Structural Dynamics Modeling of Helicopter Blades for Computational Aeroelasticity

by

Tao Cheng

B.E., Aerospace Engine, Beijing University of Aeronautics and Astronautics,
Beijing (1998)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2002
[June 2002]

Author ....
Department of Aeronautics and Astronautics
May 24, 2002

Certified by ....
Carlos E. S. Cesnik
Visiting Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by ....
Wallace E. Vander Velde
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Structural Dynamics Modeling of Helicopter Blades for Computational Aeroelasticity

by
Tao Cheng

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This thesis deals with structural dynamics modeling and simulation in time domain of helicopter blades for computational aeroelasticity. A structural model and an aeroelastic model are provided and a computer program has been developed and tested in this research. In the structural model, second-order backward Euler method is used to discretize the nonlinear intrinsic formulation for the dynamics of rotating blades in time. Newton method is used to solve the resulting nonlinear algebraic equations. The solution describes the displacement field, stress and strain field at each time step of twist composite hingeless or articulated rotor blades under the action of arbitrary external loads. Results are validated by experimental data and other numerical simulation work for various conditions. Then the aerodynamic model implemented via the GENUVP code is integrated with the structural model to form an aeroelastic simulation. The aeroelastic analysis is realized in time domain by exchanging information with two interfaces and performing consecutive aerodynamic and structural time steps. In the aeroelastic analysis, the steady state of a fixed wing at different flight speeds have been obtained and results are consistent with other methods. The time response of the active twist rotor (ATR) prototype blade in hover has also been examined. The twist response of ATR blade due to applied piezoelectric actuation is obtained and the result compared with published results. A good qualitative agreement between the present aeroelastic solution and reference results was obtained. However, quantitative discrepancies were encountered that strongly suggest that further improvements on the coupling between the two codes are needed. For all the aeroelastic test cases using the GENUVP code, no sub-iterations within a time step was used. A study considering a simple quasi-steady aerodynamics indicated that a sub-iteration in each time step may be critical to the accuracy of the final aeroelastic result. Recommendations for further work is provided at the end.

Thesis Supervisor: Carlos E. S. Cesnik
Title: Visiting Associate Professor of Aeronautics and Astronautics

Acknowledgements

I would like to thank my advisor, Professor Carlos E. S. Cesnik, for his encouragement and guidance during my graduate studies, for his patience with me during the thesis writing stage.

I would also like to thank Daniel G. Opoku and Professor Fred Nitzsche at Carleton University, Canada. Without their collaboration in this project, the aeroelastic model would not exit.

I sincerely thank Professor John Dugundji and Mark Spearing and other members of TELAC for their help over the last two years. I have learned a lot from them.

I am grateful to Dr. SangJoon Shin for his help during the difficult times. Special thanks to Eric Brown for providing the validation results. The discussion with them always turned out to be useful.

I really appreciate Professor Jaime Peraire and Klaus-Jürgen Bathe for their feedback about the time integration scheme.

Many thanks to my family in China, my parents and brother, for their long distance support and encouragement. I would like to thank my husband Yiben who gives me his deep adoration and care throughout. Without them, none of this would have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The helicopter impulsive noise can severely restrict the usage of rotorcraft in both civilian and military operations. This noise has two aerodynamic sources [1]. One is the compressible flow field due to high tip Mach number on the rotor's advancing side. It is called high-speed impulsive noise. The other is the unsteady pressure fluctuation on the rotor blades due to interactions with vortices generated by preceding blades. It is called blade-vortex interaction (BVI) noise, which is known as the most annoying noise source for residential community. A considerable amount of research over the decades on BVI noise has substantially improved the physical understanding of its generating mechanisms. And several design concepts have been presented and investigated to control the BVI noise. In recent years, many applications of active control methods are being attempted to reduce helicopter rotor BVI noise and vibrations [2]. These active control devices include higher harmonic blade pitch control (HHC), individual blade control (IBC), and active twist control. Some of these devices have been tested and significant mid-frequency noise reductions of 5-6 dB have been reported [3].

However, the accurate prediction of the unsteady airloads and resulting aeroacoustics is required in order to further increase the flexibility of input control variables. To satisfy this demand, the location of the wake and the strength of the vortical elements should be predicted accurately. And the precise and flexible control of input variables also requires accurate prediction of the dynamic response of the blade. Both of these requirements forms the basis of a numerical simulation for the analysis of the BVI noise. Therefore, the numerical simulation of the BVI noise control requires the combination of a nonlinear dynamics of moving beams and high-resolution prediction of the unsteady flow.

The motivation of this thesis is to provide an elementary numerical simulation tool for BVI noise reduction control for helicopter rotors. It is part of an ongoing collaboration between Carleton University (Prof. F. Nitzsche) and the University of Michigan (Prof. C. E. S. Cesnik).

1.2 Previous Work

Regarding the structural modeling, some research has been done in the mixed variational formulation for dynamics of moving beams. Hodges [5] has presented a nonlinear intrinsic formulation for the dynamics of initially curved and twisted beams in a moving frame. It has been integrated with the finite-state dynamic inflow theory for aeroelastic stability analysis of hingeless composite rotors in hover, Ref. [7] (a detailed history of the mixed variational formulation for dynamics of moving beams is also presented there). In Ref. [8], an asymptotical formulation is presented to analyze multi-cell composite helicopter rotor blades with integral anisotropic active plies. A linear two-dimensional analysis over the cross section and a geometrically non-linear beam analysis along the blade span were used to take into account the presence of distributed actuators. It is an extension of the exact intrinsic equations for the one-dimensional analysis of rotating beams considering small strains and finite rotations and takes into account the presence of distributed actuators. In Ref. [6], frequency response characteristics of the active twist rotor (ATR) blades and the dynamic characteristics of ATR blades are investigated analytically and experimentally.

In Ref.[9], the development of computational schemes for the dynamic analysis of non-linear elastic systems based on the displacement-based formulation is presented. The beam formulation is geometrically exact as in Ref. [5]. This scheme is based on time-discontinuous Galerkin approximations. A high frequency numerical dissipation is also obtained in this scheme [9]. The multi-body dynamics code DYMORE developed by Bauchau and co-workers is the realization of this formulation. It has been integrated with the aerodynamics of Peters and He [10] for aeroelastic simulation of helicopter rotors. The elements in the multi-body dynamics code involve rigid bodies, composite capable beams and shells, and joint models [11]. With proper definition of a multi-body model such as a rotor blade system, the static, dynamic, stability, and trim analyses can be performed on the model. In Ref. [12], the multi-body dynamics code is modified to perform the analysis of integrally twisted active rotor system during forward flight.

1.3 Present Work

The objective of this thesis is to provide a time-domain structural simulation of a rotor system to be used in a tightly-coupled computational aeroelastic solver. This will be ultimately used to study BVI noise reduction control for helicopter rotors. The mixed variational formulation based on exact intrinsic equations of motion for dynamics of moving beams and the general unsteady vortex particle aerodynamic theory [4] are integrated together to form an aeroelastic model for analysis of rotating blades in time domain.

In the structural model, an asymptotical analysis takes the electromechanical three-dimensional problem and reduces it to a set of two analyses: a linear analysis over the cross section and a nonlinear analysis of the resulting beam reference line. The nonlinear 1-D global analysis considering small strains, finite rotations, and effects of embedded piezocomposite actuators used by Shin and Cesnik (based on Ref. [5]) is solved in the time domain. After the finite element discretization in the space domain, a set of first-order ordinary differential equations is obtained. To get the time integration results, second-order backward Euler method is used to discretize in time. And Newton method is used to solve the nonlinear algebraic equations. The solution describes the displacement field, stress and strain field at each time step.

The aerodynamic model is implemented via the GENUVP code: GENeral Unsteady Vortex Particle code [4]. It is a tool for high-resolution prediction of unsteady flow for multi-component configurations such as helicopters and wind turbines. It was developed at the National Technical University of Athens (NTUA), Greece, and it has been modified at Carleton University, Canada [24]. The domain decomposition concept is used in this model. The flow field is decomposed into the near-field and the far-field. In the near-field, the regions close and around the solid boundaries are contained and the far-field contains the wakes of the different components. Appendix C presents a summary of the formulation's main features.

For the aeroelastic solution, the structural and aerodynamic modules are coupled together by two interfaces: one communicates aerodynamic loads to the structural model; the other communicates structural deformations and rates of deformation to the aerodynamic model. The aeroelastic analysis is realized in time domain by performing consecutive aerodynamic and structural time steps.

To validate the result of the present approach, results of related analyses and experiments for structural and aeroelastic modeling are used wherever possible.

Chapter 2

Theoretical Structural Modeling

In this structural model, an asymptotic analysis takes the electromechanical three-dimensional problem and reduces it to a set of two analyses: a linear analysis over the cross section and a nonlinear analysis of the resulting beam reference line.
The nonlinear 1-D global analysis considering small strains, finite rotations, and effects of embedded piezocomposite actuators is solved in the time domain.

2.1 Mixed Formulation for Dynamics of Moving Beams with Actuators

The nonlinear 1-D global analysis considering small strains, finite rotations, and effects of embedded piezocompostite actuators used by Shin and Cesnik [6] is based on the mixed variational intrinsic formulation for dynamics of moving beams originally presented by Hodges [5], and implemented by Shang and Hodges [7].

The notation used here is based on matrix notation and is consistent with the original work of Hodges [5] and Shin [12]. Some steps of the original work are repeated here to help understanding the mixed variational intrinsic formulation for dynamics of moving beams with actuators.

Three frames used by the mixed formulation for dynamics of moving beams are shown in Fig. 2-1. The global frame named a with its axes labeled a_1, a_2 and a_3 is rotating with the rotor. The undeformed blade reference frame is named b, with its axes labeled b_1, b_2 and b_3, and the deformed blade reference frame is named B with its axes labeled B_1, B_2 and B_3.

Using transformation matrices, any arbitrary vector U represented by its components in one of the frame may be converted to another frame.

$$U_B = C^{Ba} U_a \quad , U_b = C^{ba} U_a \tag{2.1}$$

where C^{Ba} is the transformation matrix from frame a to frame B, and C^{ba} is that from frame a to frame b. C^{Ba} contains the unknown rotation variables, while C^{ba} can be expressed in terms of direction cosines from the geometry of the undeformed rotor blade.

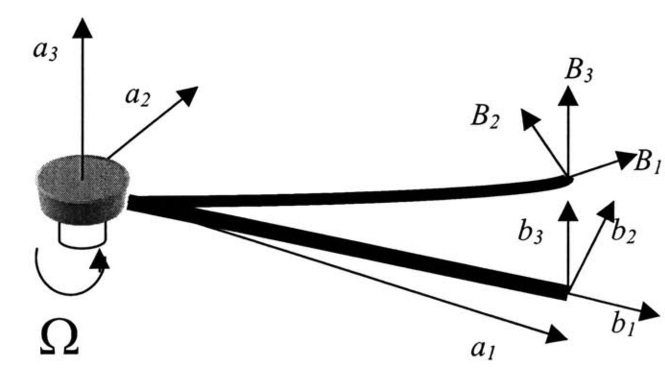


Figure 2-1: Global reference frame a, undeformed beam reference frame b and deformed beam reference frame B

The variational formulation is derived from Hamilton's principle, which can be written as

$$\int_{t_1}^{t_2}\int_0^l \left[\delta(K-U)+\overline{\delta W}\right]dx_1\,dt = \overline{\delta A} \tag{2.2}$$

where t_1 and t_2 are arbitrarily fixed times, l is the length of the beam, K and U are the kinetic and potential energy densities per unit length, respectively. $\overline{\delta A}$ is the virtual action at the ends of the beam and at the ends of the time interval, and $\overline{\delta W}$ is the virtual work of applied loads per unit length.

The variation of the kinetic energy terms is with respect to the linear velocity column vector V_B and angular velocity column vector Ω_B respectively. The velocities are all measured in the deformed blade frame B. The variation of the potential energy terms is with respect to the generalized strain column vectors γ and κ. The force and moment strain vectors γ and κ are measured in the undeformed blade frame b.

Following Hodges [5], the partial derivatives of U and K are identified section stress resultants and momenta resultants

$$F_B = \left(\frac{\partial U}{\partial \gamma}\right)^T, \quad M_B = \left(\frac{\partial U}{\partial \kappa}\right)^T$$

$$P_B = \left(\frac{\partial K}{\partial V_B}\right)^T, \quad H_B = \left(\frac{\partial K}{\partial \Omega_B}\right)^T \tag{2.3}$$

where F_B and M_B are the internal force and moment column vectors and P_B and H_B are the linear and angular momenta column vectors. The subscripts indicate the frame which the measure numbers are obtained from. The first element of F_B is the axial force and the second and third elements are shear forces in the deformed frame B. Similarly, the first element of M_B is the twisting moment and the second and third elements are bending moments.

The geometrically exact kinematical relations defined in Ref. [7] are given as:

$$\gamma = C^{Ba}(C^{ba}e_1 + u_a') - e_1$$

$$\kappa = C^{ba}\left(\frac{\Delta - \frac{\tilde{\theta}}{2}}{1+\frac{\theta^T\theta}{4}}\right)\theta'$$

$$V_B = C^{Ba}(v_a + \dot{u}_a + \tilde{\omega}_a u_a)$$

$$\Omega_B = C^{ba}\left(\frac{\Delta - \frac{\tilde{\theta}}{2}}{1+\frac{\theta^T\theta}{4}}\right)\dot{\theta} + C^{Ba}\omega_a \tag{2.4}$$

where u_a is the displacement vector measured in the a frame, θ is the rotation vector expressed in terms of Rodrigues parameters $\theta_i = 2e_i \tan(\alpha/2)$ which are defined in terms of a rotation of magnitude α about a unit vector $\boldsymbol{e}=e_i\boldsymbol{b}_i$, e_1 is the unit vector $[1, 0, 0]^T$, Δ is the 3 x 3 identity matrix, v_a and w_a are the initial velocity and initial angular velocity of a generic point on the a frame. $\dot{u}_a$ and $\dot{\theta}$ are time derivatives of displacement and rotation. u'_a and θ' are derivatives with respect to the spanwise curvilinear coordinate. The rotation matrix $C=C^{ab}C^{Ba}$ is expressed in terms of θ as following:

$$C = \frac{(1-\frac{\theta^T\theta}{4})\Delta - \tilde{\theta} + \frac{\theta\theta^T}{2}}{1+\frac{\theta^T\theta}{4}} \tag{2.5}$$

where $\tilde{\theta}$ operator converts a column vector to its dual matrix:

$$\tilde{\theta} = \begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix} \tag{2.6}$$

To form a mixed formulation, Lagrange's multipliers are used to enforce the satisfaction of the kinematical equations, Eq. (2.4). Using the rotation matrix C, some transformations can be performed so that all δ quantities, displacement and rotation quantities are measured in global frame a and the strains, velocities, forces and momenta are measured in deformed blade reference frame B. Thus, the a frame version of the variational formulation based on exact intrinsic equations for dynamics of moving beams can be obtained as

$$\int_{t_1}^{t_2} \delta\Pi_a dt = 0 \tag{2.7}$$

where

$$\begin{aligned}
\delta\Pi_a = \int_0^l \{ & \delta u_a'^T C^T C^{ab} F_B + \delta u^T{}_a \left[\left(C^T C^{ab} P_B \right)^{\bullet} + \widetilde{\omega}_a C^T C^{ab} P_B \right] \\
& + \overline{\delta\psi}'{}_a C^T C^{ab} M_B - \overline{\delta\psi}_a C^T C^{ab} (\widetilde{e}_1 + \widetilde{\gamma}) F_B \\
& + \overline{\delta\psi}_a{}^T [(C^T C^{ab} H_B)^{\bullet} + \widetilde{\omega}_a C^T C^{ab} H_B + C^T C^{ab} \widetilde{V}_B P_B] \\
& - \overline{\delta F}_a{}^T [C^T C^{ab} (e_1 + \gamma) - C^{ab} e_1] - \overline{\delta F}'{}_a{}^T u_a \\
& - \overline{\delta M}_a{}^T (\Delta + \frac{\widetilde{\theta}}{2} + \frac{\theta\theta^T}{4}) C^{ab} \kappa - \overline{\delta M}'{}_a \theta \\
& + \overline{\delta P}_a{}^T (C^T C^{ab} V_B - v_a - \widetilde{\omega}_a u(_a)) - \overline{\delta P}_a{}^T \dot{u}_a \\
& + \overline{\delta H}_a{}^T (\Delta - \frac{\widetilde{\theta}}{2} + \frac{\theta\theta^T}{4}) (C^T C^{ab} \Omega_B - \omega_a) \\
& - \overline{\delta H}_a{}^T \dot{\theta} - \delta u_a{}^T f_a - \overline{\delta\psi}_a{}^T m_a \} dx_1 \\
& - (\delta u_a{}^T \hat{F}_a + \overline{\delta\psi}_a{}^T \hat{M}_a - \overline{\delta F}_a{}^T \hat{u}_a - \overline{\delta M}_a{}^T \hat{\theta}) \Big|_0^l
\end{aligned} \tag{2.8}$$

In Eq. (2.8), f_a and m_a are the external force and moment vectors respectively, which results from aerodynamic loads. And ($\delta u_a{}^T f_a + \overline{\delta\psi}_a{}^T m_a$) is the virtual work of the applied loads per unit length, which is $\overline{\delta W}$ in Eq. (2.2).

The $\hat{F}_a, \hat{M}_a, \hat{u}_a, \hat{\theta}$ terms are boundary values of the corresponding quantities that depend on the boundary conditions. For example, in the case of a hingeless rotor blade, tip forces and moments $\hat{F}_a, \hat{M}_a$ are zeros and root displacements and rotations $\hat{u}_a$, $\hat{\theta}$ are zeros. And $(\delta u_a{}^T \hat{F}_a + \overline{\delta\psi}_a{}^T \hat{M}_a - \overline{\delta F}_a{}^T \hat{u}_a - \overline{\delta M}_a{}^T \hat{\theta})\Big|_0^l$ are the boundary terms associated with the virtual action at the ends of the time interval ($\overline{\delta A}$ in Eq. (2.2)).

The generalized strain and force measures, and velocity and momenta measures are related through the constitutive relations in the following form:

$$\begin{Bmatrix} F_B \\ M_B \end{Bmatrix} = [K]\begin{Bmatrix} \gamma \\ \kappa \end{Bmatrix} - \begin{Bmatrix} F_B{}^{(a)} \\ M_B{}^{(a)} \end{Bmatrix}$$

$$\begin{Bmatrix} P_B \\ H_B \end{Bmatrix} = [M]\begin{Bmatrix} V_B \\ \Omega_B \end{Bmatrix} \tag{2.9}$$

where $F_B{}^{(a)}$ and $M_B{}^{(a)}$ are actuation forces and moments which depend on the geometry, material distribution, and applied electric field. The stiffness $[K]$ is in general a 6 x 6 matrix, depending on material distribution and cross sectional geometry. Detailed expressions for the stiffness and mass matrices and actuation vector can be found in Ref. [8].

Eq. (2.9) are solved for γ, κ, V_B and Ω_B as functions of the other measures and constants in the following forms:

$$\begin{Bmatrix} \gamma \\ \kappa \end{Bmatrix} = [K]^{-1}\left(\begin{Bmatrix} F_B \\ M_B \end{Bmatrix} + \begin{Bmatrix} F_B{}^{(a)} \\ M_B{}^{(a)} \end{Bmatrix}\right)$$

$$\begin{Bmatrix} V_B \\ \Omega_B \end{Bmatrix} = [M]^{-1}\begin{Bmatrix} P_B \\ H_B \end{Bmatrix} \tag{2.10}$$

where F_B and M_B are internal force and moment column vectors which are unknown variables and the actuation forces and moments $F_B{}^{(a)}$ and $M_B{}^{(a)}$ are given functions of time. These equations are substituted into Eq. (2.8) with the actuation forces and moments as control inputs.

2.2 Finite Element Discretization and System Equations

Adopting the finite element method by discretizing the space domain of the blade into N elements, Eq. (2.7) is written as

$$\int_{t_1}^{t_2} \sum_{i=1}^{N} \delta\Pi_i dt = 0 \tag{2.11}$$

where index i indicates the ith element with length dl, $\delta\Pi_i$ is the corresponding spatial integration of the function in Eq. (2.8) over the ith element. Due to the formulation's weakest form, the simplest shape functions can be used. Therefore, the following transformation and interpolation are applied within each element as presented in Ref. [7]:

$$x = x_i + \xi\Delta l_i, \quad dx = \Delta l_i d\xi, \quad (\ )' = \frac{1}{\Delta l_i}\frac{d}{d\xi}(\ )$$

$$\begin{aligned}
\delta u_a &= \delta u_i(1-\xi) + \delta u_{i+1}\xi & u_a &= u_i \\
\overline{\delta\psi}_a &= \overline{\delta\psi}_i(1-\xi) + \overline{\delta\psi}_{i+1}\xi & \theta &= \theta_i \\
\overline{\delta F}_a &= \overline{\delta F}_i(1-\xi) + \overline{\delta F}_{i+1}\xi & F_B &= F_i \\
\overline{\delta M}_a &= \overline{\delta M}_i(1-\xi) + \overline{\delta M}_{i+1}\xi & M_B &= M_i \\
\overline{\delta P}_a &= \overline{\delta P}_i & P_B &= P_i \\
\overline{\delta H}_a &= \overline{\delta H}_i & H_B &= H_i
\end{aligned} \tag{2.12}$$

where u_i, θ_i, F_i, M_i, P_i and H_i are constant vectors and all δ quantities are arbitrary. ξ varies from 0 to 1.

With these shape functions, the spatial integration of Eq. (2.11) can be performed explicitly to give:

$$\sum_{i=1}^{N}\{\delta u_i^T f_{u_i} + \overline{\delta\psi}_i^T f_{\psi_i} + \overline{\delta F}_i^T f_{F_i} + \overline{\delta M}_i^T f_{M_i} + \overline{\delta P}_i^T f_{P_i} + \overline{\delta H}_i^T f_{H_i}$$
$$+ \delta u_{i+1}^T f_{u_{i+1}} + \overline{\delta\psi}_{i+1}^T f_{\psi_{i+1}} + \overline{\delta F}_{i+1}^T f_{F_{i+1}} + \overline{\delta M}_{i+1}^T f_{M_{i+1}} + \overline{\delta P}_{i+1}^T f_{P_{i+1}} + \overline{\delta H}_{i+1}^T f_{H_{i+1}}\} = \quad (2.13)$$
$$\delta u_{N+1}^T \hat{F}_{N+1} + \overline{\delta\psi}_{N+1}^T \hat{M}_{N+1} - \overline{\delta F}_{N+1}^T \hat{u}_{N+1} - \overline{\delta M}_{N+1}^T \hat{\theta}_{N+1}$$
$$- \delta u_1^T \hat{F}_1 - \overline{\delta\psi}_1^T \hat{M}_1 + \overline{\delta F}_1^T \hat{u}_1 + \overline{\delta M}_1^T \hat{\theta}_1$$

where the f_{ui} , $f_{\psi i}$,. . ., f_{Mi+1} are the element functions explicitly integrated from the formulation. There expressions are as follows:

$$f_{u_i} = -C^T C^{ab} F_i + \frac{\Delta l_i}{2}\widetilde{\omega}_a C^T C^{ab} P_i + \frac{\Delta l_i}{2}(C^T C^{ab} P_i)^{\bullet} - \overline{f_i}$$
$$f_{\psi_i} = -C^T C^{ab} M_i - \frac{\Delta l_i}{2} C^T C^{ab}(\widetilde{e}_1 + \widetilde{\gamma})F_i$$
$$+ \frac{\Delta l_i}{2}(\widetilde{\omega}_a C^T C^{ab} H_i + C^T C^{ab}\widetilde{V}_i P_i) + \frac{\Delta l_i}{2}(C^T C^{ab} H_i)^{\bullet} - \overline{m_i}$$
$$f_{F_i} = u_i - \frac{\Delta l_i}{2}[C^T C^{ab}(\widetilde{e}_1 + \widetilde{\gamma}) - C^{ab} e_1]$$
$$f_{M_i} = \theta_i - \frac{\Delta l_i}{2}(\Delta + \frac{\widetilde{\theta}}{2} + \frac{\theta\theta^T}{4})C^{ab}\kappa_i$$
$$f_{P_i} = C^T C^{ab} V_i - v_a - \widetilde{\omega}_a u_i - \dot{u}_i$$
$$f_{H_i} = \Omega_i - C^{ab} C\omega_a - C^{ab}\left(\frac{\Delta - \frac{\widetilde{\theta}_i}{2}}{1 + \frac{\theta_i^T\theta_i}{4}}\right)\dot{\theta}_i$$
$$f_{u_{i+1}} = C^T C^{ab} F_i + \frac{\Delta l_i}{2}\widetilde{\omega}_a C^T C^{ab} P_i + \frac{\Delta l_i}{2}(C^T C^{ab} P_i)^{\bullet} - \overline{f_{i+1}}$$
$$f_{\psi_{i+1}} = C^T C^{ab} M_i - \frac{\Delta l_i}{2} C^T C^{ab}(\widetilde{e}_1 + \widetilde{\gamma})F_i$$
$$+ \frac{\Delta l_i}{2}(\widetilde{\omega}_a C^T C^{ab} H_i + C^T C^{ab}\widetilde{V}_i P_i) + \frac{\Delta l_i}{2}(C^T C^{ab} H_i)^{\bullet} - \overline{m_{i+1}} \quad (2.14)$$
$$f_{F_{i+1}} = -u_i - \frac{\Delta l_i}{2}[C^T C^{ab}(\widetilde{e}_1 + \widetilde{\gamma}) - C^{ab} e_1]$$
$$f_{M_{+1i}} = -\theta_i - \frac{\Delta l_i}{2}(\Delta + \frac{\widetilde{\theta}}{2} + \frac{\theta\theta^T}{4})C^{ab}\kappa_i$$

where $\overline{f}_i$, $\overline{f}_{i+1}$, $\overline{m}_i$ and $\overline{m}_{i+1}$ are the effective nodal load vectors. For example, the distributed aerodynamic force can be expressed as the effective nodal load vectors using the relation:

$$\bar{f}_i = \int_{l_i}(1-\xi) f_a dx_1, \qquad \bar{f}_{i+1} = \int_{l_i}\xi f_a dx_1,$$

$$\overline{m}_i = \int_{l_i}(1-\xi) m_a dx_1, \qquad \overline{m}_{i+1} = \int_{l_i}\xi m_a dx_1 \tag{2.15}$$

In Eq. (2.14), the generalized strains γ, κ and velocities V_B, Ω_B are given by the constitutive relations, Eq. (2.10). The effects of the embedded anisotropic piezocomposite actuators come with the expressions of γ and κ.

Since each δ-quantity is arbitrary, Eq. (2.13) yields a set of partial differential equations that can be written in matrix notation as:

$$F_S(X,\dot{X}) - F_L = 0 \tag{2.16}$$

where F_S is the structural operator, F_L is the load operator, X is the unknown vector consisting of structural variables. In these equations, the actuation forces and moments $F_B^{(a)}$ and $M_B^{(a)}$ are time dependent input parameters associated with F_S. Explicit expressions of the structural and load operators are as follows:

$$
F_S = \begin{bmatrix} f_{u_1}^{(1)} + \hat{F}_1 \\ f_{\psi_1}^{(1)} + \hat{M}_1 \\ f_{F_1}^{(1)} - \hat{u}_1 \\ f_{M_1}^{(1)} - \hat{\theta}_1 \\ f_{P_1}^{(1)} \\ f_{H_1}^{(1)} \\ f_{u_2}^{(1)} + f_{u_2}^{(2)} \\ f_{\psi_2}^{(1)} + f_{\psi_2}^{(2)} \\ f_{F_2}^{(1)} + f_{F_2}^{(2)} \\ f_{M_2}^{(1)} + f_{M_2}^{(2)} \\ f_{P_2}^{(2)} \\ f_{P_2}^{(2)} \\ f_{u_3}^{(2)} + f_{u_3}^{(3)} \\ f_{\psi_3}^{(2)} + f_{\psi_3}^{(3)} \\ f_{F_3}^{(2)} + f_{F_3}^{(3)} \\ f_{M_3}^{(2)} + f_{M_3}^{(3)} \\ \vdots \\ f_{u_{N+1}}^{(N)} - \hat{F}_{N+1} \\ f_{\psi_{N+1}}^{(N)} - \hat{M}_{N+1} \\ f_{F_{N+1}}^{(N)} + \hat{u}_{N+1} \\ f_{M_{N+1}}^{(N)} + \hat{\theta}_{N+1} \end{bmatrix}, \qquad F_L = \begin{bmatrix} \bar{f}_1^{(1)} \\ \overline{m}_1^{(1)} \\ 0 \\ 0 \\ 0 \\ 0 \\ f_2^{(1)} + f_2^{(2)} \\ m_2^{(1)} + m_2^{(2)} \\ 0 \\ 0 \\ 0 \\ 0 \\ f_3^{(2)} + f_3^{(3)} \\ m_3^{(2)} + m_3^{(3)} \\ 0 \\ 0 \\ \vdots \\ f_{N+1}^{(N)} \\ m_{N+1}^{(N)} \\ 0 \\ 0 \end{bmatrix} \tag{2.17}
$$

In the above, superscripts indicate the element number and subscripts indicate the node number. It can be seen that the dimension of F_S and F_L is $18N+12$. The components of the unknown structural variables in X depend on the boundary condition. For a hingeless rotor blade, X is organized as

$$
\begin{aligned} X = [\hat{F}_1^T \;\; \hat{M}_1^T \;\; u_1^T \;\; \theta_1^T \;\; F_1^T \;\; M_1^T \;\; P_1^T \;\; H_1^T \cdots \\ \cdots \;\; u_N^T \;\; \theta_N^T \;\; F_N^T \;\; M_N^T \;\; P_N^T \;\; H_N^T \;\; \hat{u}_{N+1}^T \;\; \hat{\theta}_{N+1}^T]^T \end{aligned} \tag{2.18}
$$

For an articulated blade, there are some modifications of the unknown vector because of different unknown variables. Specifically, the two internal bending moments at the root of the articulated blade are zeros. However, the two bending rotation angles at the root are not zeros any more and become unknown variables. Therefore, the unknown vector $\hat{M}_1$ is modified as following:

$$\hat{M}_1 = \begin{Bmatrix} \hat{M}_{11} \\ \theta_{02} \\ \theta_{03} \end{Bmatrix} \tag{2.19}$$

where $\hat{M}_{11}$ is the twisting moment at the root of the blade and θ_{02}, θ_{03} are the lead-lag and flap rotations at the root in terms of Rodrigues parameters, respectively. Consequently, the two internal bending moments $\hat{M}_{12}$ and $\hat{M}_{13}$ are zeros, yielding free rotation at the articulation hinge.

2.3 Hinge Dampers

In what has been presented before, there is no structural damping in the mixed formulation for dynamics of moving beams. If damping is needed especially for an articulated blade, hinge dampers can be added into the formulation by modifying the two internal bending moments at the root with proper damping coefficients. For example,

$$\hat{M}_{12} = C_{02}\dot{\theta}_{02} \tag{2.20}$$

where $\hat{M}_{12}$ is the internal flap bending moment, C_{02} is the flap damping coefficient, and $\dot{\theta}_{02}$ is the flap angular velocity at the hinge. The flap angular velocity is obtained from the time derivative of the flap rotation at the hinge point. Similarly, $\hat{M}_{13}$ can be obtained using the lead-lag angular velocity at the hinge. Then these two terms are used in Eq. (2.16) as external forces.

2.4 Finite Difference Discretization and Time Integration

After the finite element discretization in the space domain, a set of first-order ordinary differential equations, Eq. (2.16) is obtained. To integrate those equations in time, second-order backward Euler method [14] is used to discretize Eq. (2.16) in time. Therefore, the following finite difference discretization is applied at each time step n.

$$\dot{P}_i^n = \frac{3P_i^n - 4P_i^{n-1} + P_i^{n-2}}{2\Delta t}$$

$$\dot{H}_i^n = \frac{3H_i^n - 4H_i^{n-1} + H_i^{n-2}}{2\Delta t}$$

$$\dot{u}_i^n = \frac{3u_i^n - 4u_i^{n-1} + u_i^{n-2}}{2\Delta t}$$

$$\dot{\theta}_i^n = \frac{3\theta_i^n - 4\theta_i^{n-1} + \theta_i^{n-2}}{2\Delta t} \tag{2.21}$$

where Δt is the time step size. Superscripts indicate the time step and subscripts indicate the node number.

Writing Eq. (2.16) at time step n and using Eq. (2.21), a set of nonlinear algebraic equations is obtained as

$$F_S(X^n) - F_L = 0 \tag{2.22}$$

where X^n is the unknown structural vector at time step n. Newton method is used to solve the nonlinear algebraic equations given by Eq. (2.22). The Jacobian matrix can be derived explicitly by differentiation

$$[J] = [\frac{\partial Fs}{\partial X}] \tag{2.23}$$

whose expressions are listed in Appendix A.

The solution of Eq. (2.22) describes the displacement field, stress and strain field at each time step.

2.5 Solution Flow

Fig. 2-2 shows the block diagram of the solution process for the structural analysis just presented.

In the first block of the program, data for the geometry of the blade, time integration parameters, rotating speed, pitch control information, finite element mesh, and material properties are input and processed. Appendix B shows a sample case of the input format. And the actuation forces or moments are input as a function of time. Then the unknown vector X is defined and the initial values are given.

The next block is the time integration block whose kernel part is the Newton nonlinear equation solver. First, the external forces or actuation forces at the present time step are read. After that, the initial velocity, pitch angle, matrix C^{ab} and actuation forces for each element are evaluated. Then Newton method is used to solve the nonlinear equations and it is shown in Fig. 2-3. In the Newton solver, the initial guess of the variables is set equal to the values of last time step. In each iteration in Newton solver, system equations and Jacobian matrix are calculated. The iteration stops when it converges and the value of the unknown vector X is obtained and saved. Then, the next time step begins.

The last block encompasses post processing and output. All the values of the unknown vector X at each time step are saved. For future aeroelastic integration, deformation information to be passed to the aerodynamic code is processed within this block.

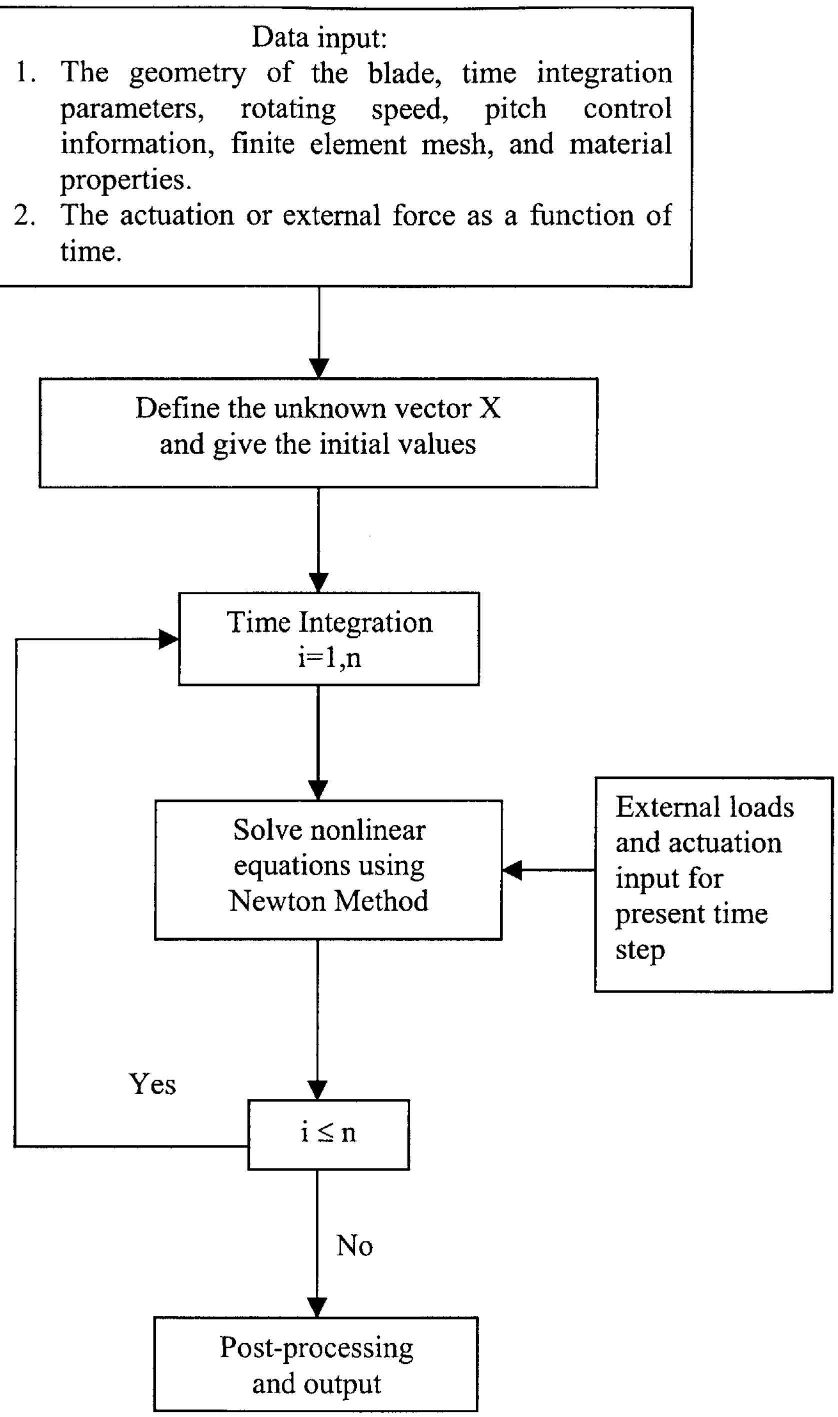


Figure 2-2: Block-Diagram of the solution process for the structural analysis

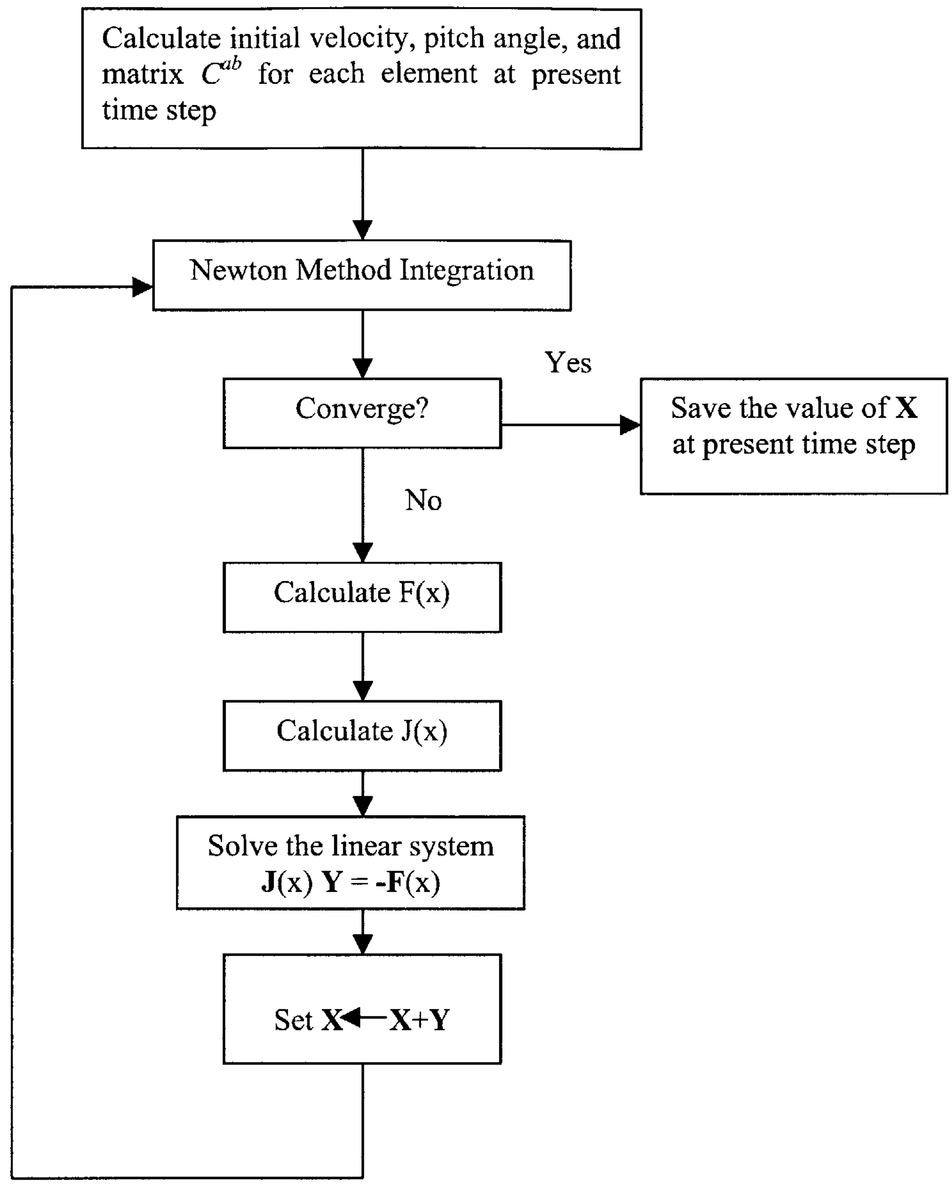


Figure 2-3: Block-Diagram of the Newton method

Chapter 3

Aeroelastic Modeling

In this chapter, structural and aerodynamic models are coupled together by two interfaces: one communicates aerodynamic loads to the structural model; the other communicates structural deformations and rates of deformation to the aerodynamic model. The aeroelastic analysis is realized in time domain by performing consecutive aerodynamic and structural time steps.

3.1 Model Overview

Fig. 3-1 gives an overview of the active aeroelastic model. The aerodynamic model is implemented via the GENUVP code: GENeral Unsteady Vortex Particle code [4]. It is a tool for high-resolution prediction of unsteady flow for multi-component configurations such as helicopters and wind turbines. It was developed at the National Technical University of Athens (NTUA), Greece, and it has been modified at Carleton University, Canada. Further information about the unsteady aerodynamic code is presented in

Appendix C. The structural model is the finite element representation described in Chapter 2. Performing consecutive aerodynamic and structural time steps, the aeroelastic analysis is realized in the time domain. There are two coupling interfaces between the aerodynamic model and the structural model. The piezocomposite actuation is an input to the structural model.

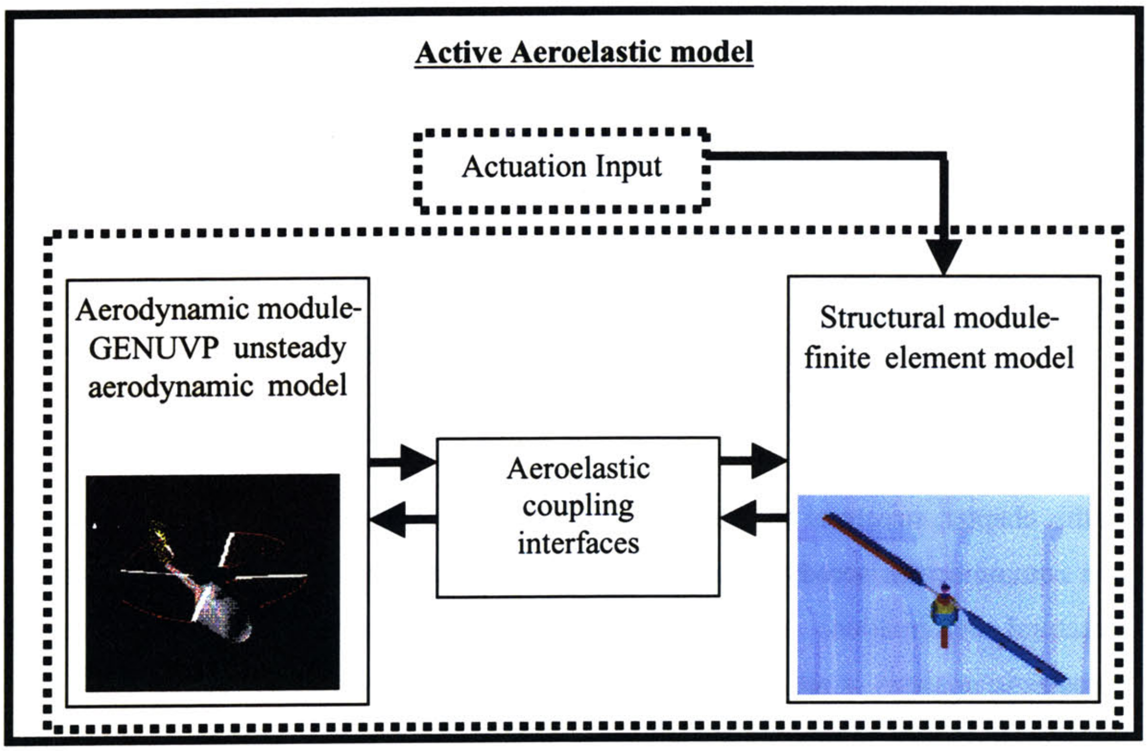


Figure 3-1: Active aeroelastic model overview

3.2 Aeroelastic Coupling Interfaces

Fig. 3-2 shows the aeroelastic coupling. Rotor geometry and structure, flight conditions, and active control are inputs to the aerodynamic and structural components. There are two coupling interfaces defined: one communicates aerodynamic loads to the structural model; the other communicates structural (blade) deformation and rates of deformation to the aerodynamic model. The outputs are rotor flow field, aerodynamic loading, structural deformation, trim conditions, acoustic field, and hub vibration.

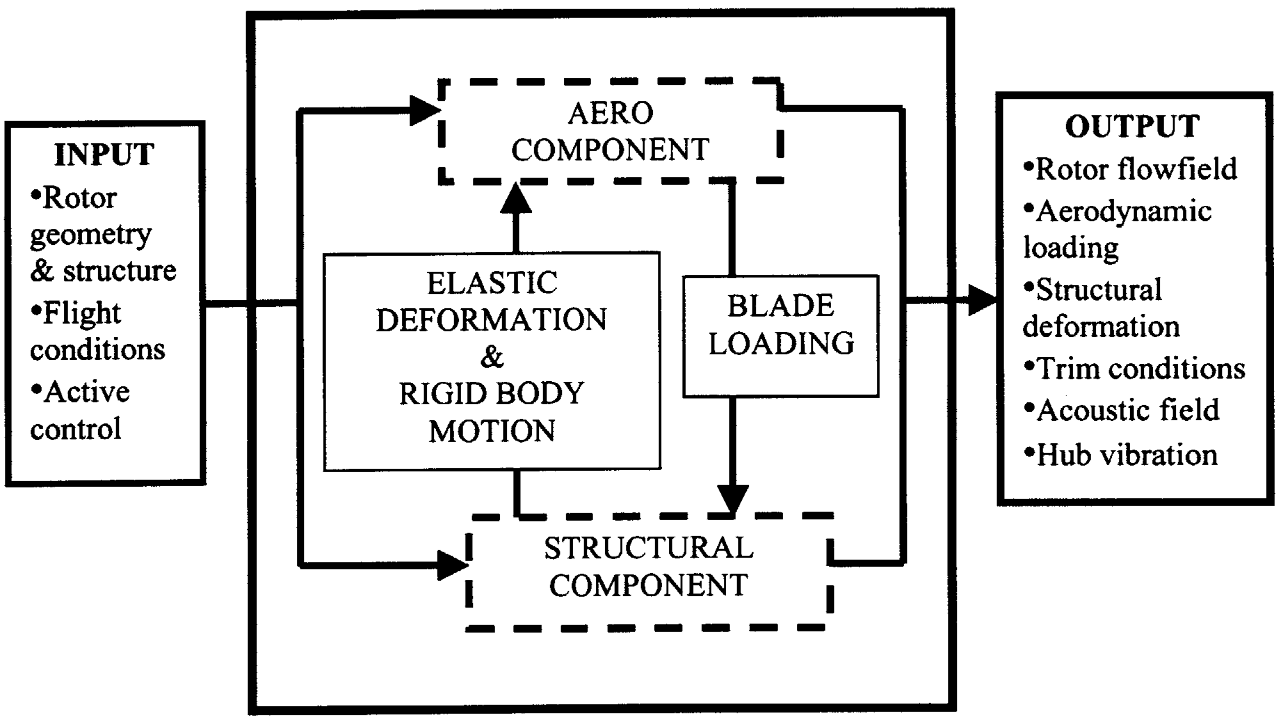


Figure 3-2: Aeroelastic solver

In order to minimize interpolation error, coincident spanwise meshing is used in the aerodynamic and structural components as shown in Fig. 3-3.

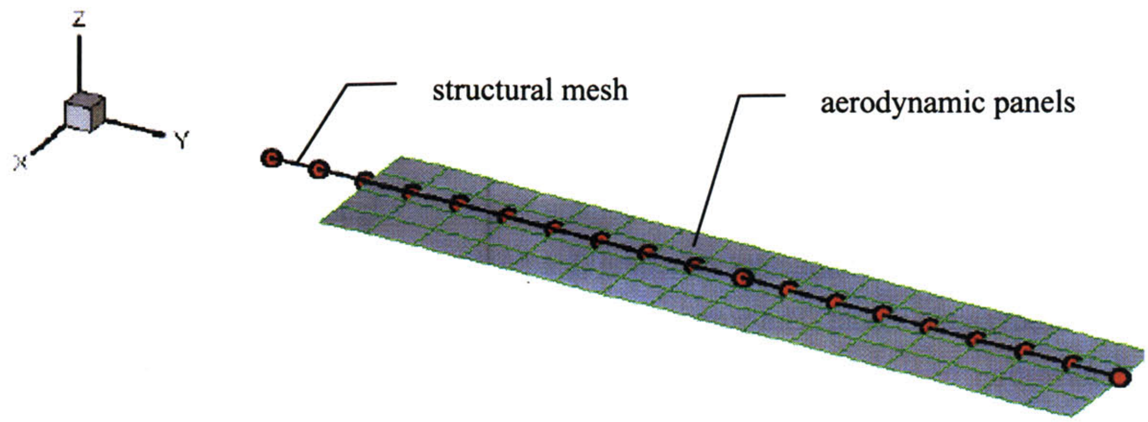


Figure 3-3: Coincident spanwise meshing

In the aerodynamic component, effective angle of attack is calculated at each spanwise station and viscous correction is applied using 2D airfoil data. Then the distributed aerodynamic loads calculated from the aerodynamic module are used to calculate concentrated loads applied at each structural node. With the aerodynamic loads applied, two kinds of deformation data are calculated in the structural module. Rigid body motion is calculated at the hinge point and elastic deformation is obtained at each blade point by subtracting the rigid body motion from the total deformations. Then for the aerodynamic module, rigid body feedback is applied as body motion, elastic deformation is used to alter aerodynamic mesh shape and the rate of elastic deformation alters aerodynamic system boundary conditions.

3.3 Solution Flow

A general active aeroelastic rotorcraft code capable of modeling various rotors is developed. FORTRAN 77 is used for programming. *Matlab* is used to generate the Jacobian matrix symbolically. The *Numerical Recipes*' LU decomposition subroutines [15] are used to construct the linear equation solver.

The block diagram of the aeroelastic solver is presented in Fig. 3-4.

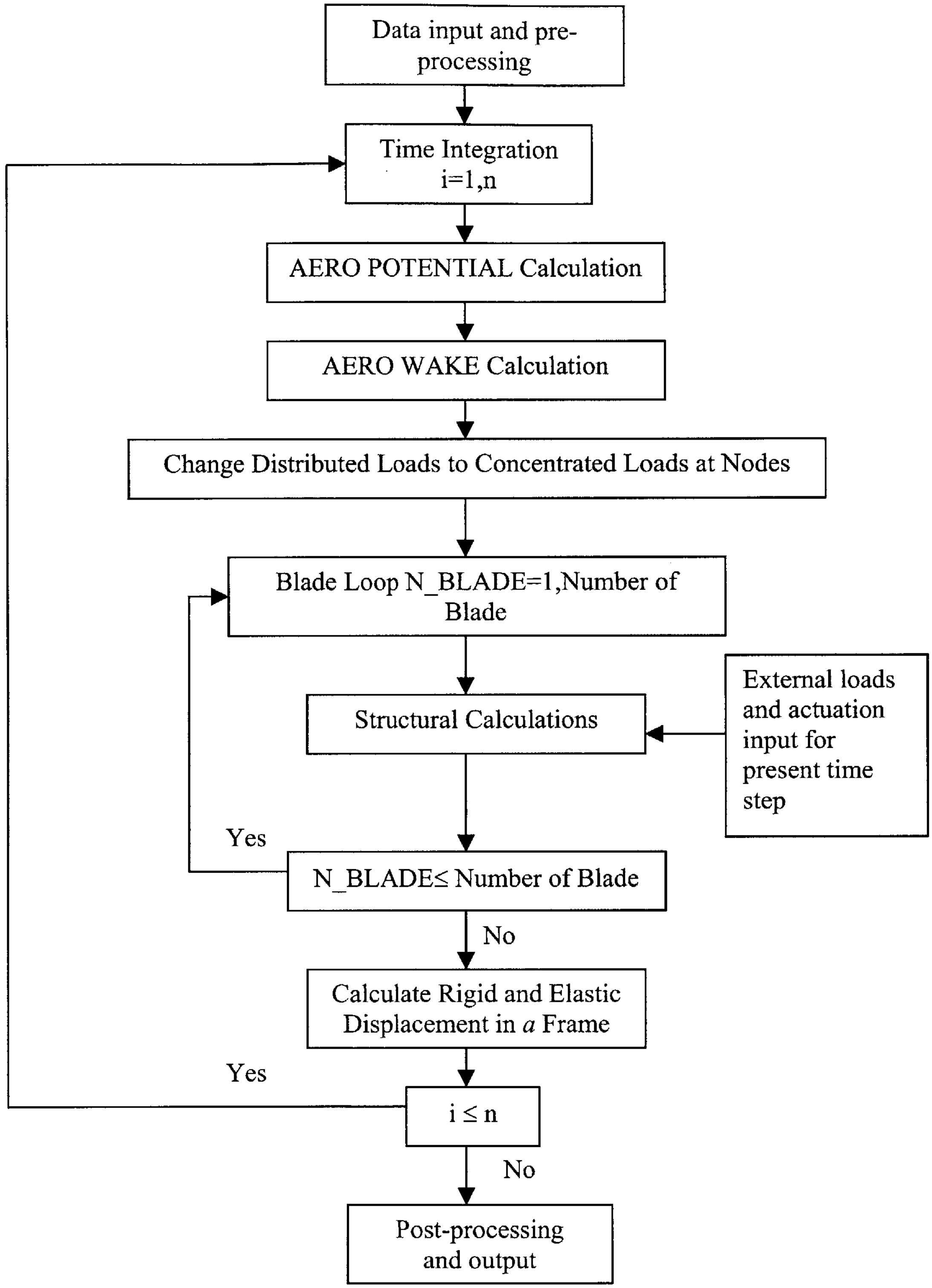


Figure 3-4: Block diagram of the aeroelastic code

In the first block of the program, data for the geometry of the blade, time integration parameters, rotating speed, pitch control information, finite element mesh, active control parameters and material properties are input and processed.

In the time integration block, the aerodynamic code is called to perform the aero potential calculation and aero wake calculation. Distributed loads calculated from the aerodynamic code are converted to concentrated loads and transferred to the structural code. In the aerodynamic and structural components, different global frames are used as shown in Fig. 3-5. Frame A is the global frame in the aerodynamic module and frame a is the global frame in the structural module. Therefore, the aerodynamic forces calculated in frame A in the aerodynamic component should be transformed to frame a when they are applied to the structural component. With the aerodynamic loads, the deformations of each blade are obtained using the structural code and are separated into rigid and elastic deformation. These data are transferred from frame a to frame A and then are used in the aerodynamic code at next time step. The time integration goes on until it reaches the last time step.

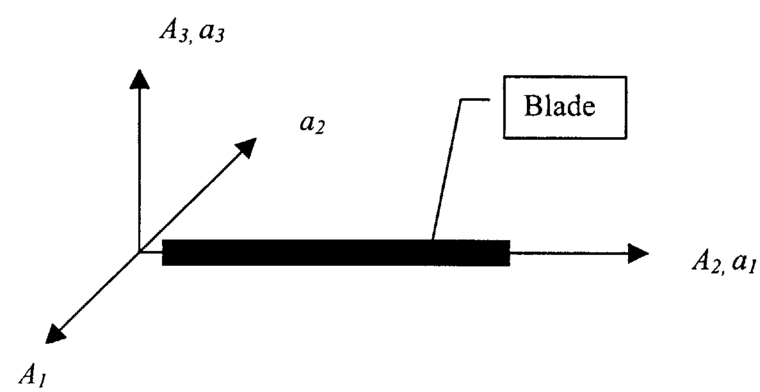


Figure 3-5: Global frames in the aerodynamic (A) and structural (a) solutions

In the post-processing and output block, the rotor flowfield, aerodynamic loading, and structural deformation are saved to different files for plotting.

3.4 Solution Process

In the aeroelastic analysis of the rotating blades, the calculation is separated into two steps: one is the steady analysis of the rotating blade in vacuum; the other is the dynamic analysis of the blade in air.

The steady analysis is derived from the dynamic analysis by eliminating the time derivative terms in all the structural equations. It can be used to calculate the steady state of blades under any kind of loading and the steady state of blades at any rotating speed. When calculating the steady state of rotating blades, the rotating speed should be given as a ramp function in order to get convergent results.

The deformations, internal forces and moments, and momenta of a rotating blade in vacuum are obtained by the steady state analysis. They are used in the dynamic analysis (with aerodynamic loads) as the initial rotating condition input.

In summary, the response of a rotating blade to aerodynamic loads is obtained using the following two-step solution.

1. Steady analysis: calculates the deformations, internal forces and moments, momenta of a rotating blade in vacuum
2. Dynamic analysis: calculates the dynamic response of rotating blades to aerodynamic loads, using the results obtained from the steady analysis as its initial condition.

In the dynamic analysis, the blades are rotating at their full speed at the first structural time step. This is due to the initial condition of the steady state of blades being that of rotating at their full speed. However, in the aerodynamic component, the rotating speed should be increased from zero to the full speed. This is to avoid suddenly applied aerodynamic forces, which may result in large numerical blade oscillations.

3.5 Time Step Size

The time step sizes in the aerodynamic and structural modules can be defined separately. If they are the same, the two components exchange data at every time step using the interfaces between them. If the time step sizes used in the two components are different, the data exchange does not occur at every time step. For example, if the time step size in the structural module is larger than that in the aerodynamic one, several time integrations are needed within the aerodynamic module so to exchange data with the structural module.

The choice of the time step size for the structural component has two requirements. First of all, the time step size should be small enough to make sure the scheme is stable. Secondly, the time step size has to be chosen to yield an accurate and effective solution. If the time step size is not small enough, the results will have period elongations and amplitude decays. In general, the numerical integrations are accurate when $\Delta t/T$ is smaller than about 0.01 [16], where T is the smallest modal period of interest. Usually, only the low frequency responses are important for the analysis. Therefore, the time step Δt should be only small enough that the responses in all modes that are significant to the total structural response are calculated accurately. The other modal response components may not be evaluated accurately. However, the errors are not important because the response measured in those components can be neglected. So the choice of the time step size depends on the highest frequency of interest and varies from case to case.

Chapter 4

Numerical Validation for Structural Modeling

In order to validate the structural modeling, a verification study is carried out for static and dynamic response in vacuum. Results are compared with experiments and other related analytical methods. Correlations demonstrated the current structural formulation is correctly implemented and ready to be integrated into the aeroelastic solver.

4.1 Reference Solution

Two related analytical methods are used as reference solutions in order to validate the structural modeling presented here: DYMORE [18] and Cesnik and Brown [19].

DYMORE is a finite element-based tool for the analysis of nonlinear flexible multibody systems [18]. It was developed by Bauchau and co-workers and it is base on the exact displacement-based formulation for dynamics of moving beams. It has the aerodynamics of Peters and He [10] built-in in the code. In DYMORE, a time-discontinuous integration scheme is used. Presenting high-frequency numerical dissipation, this scheme has energy decaying characteristics [17]. After proper definition of a mutibody model such as a rotor blade system, the static, dynamic, stability, and trim analyses can be performed on the model.

The other analysis method is presented by Cesnik and Brown [19]. A nonlinear strain-based beam model is used without considering the extensional and shear forces. The aerodynamic model is the same Peters and He [10]. The structural equations and the aerodynamic equations are integrated together and presented in a state space format. The steady state deformation and fully nonlinear time marching of the wing can be obtained by this model.

4.2 Static Test

Various tests for static response are carried out for different beams, different boundary condition and different load distribution.

4.2.1 Simple Beam Deflection Test

The length of the beam used in all this set of tests is 1.00 meter. Table 4.1 presents the material properties of this beam. In all these calculations, the effect of the beam weight has not been considered.

Table 4.1: Material Properties of the test beam

Mass per unit span (kgm^{-1})	0.2
I_{xx} (kgm)	$1.0*10^{-4}$
I_{yy} (kgm)	$1.0*10^{-6}$
I_{zz} (kgm)	$1.0*10^{-4}$
K_{11} (N)	$1.0*10^{-6}$
K_{22} (N)	$1.0*10^{20}$
K_{33} (N)	$1.0*10^{20}$
K_{44} (Nm^2)	50
K_{55} (Nm^2)	50
K_{66} (Nm^2)	$1.0*10^{3}$

4.2.1.1 Test case 1

The boundary condition of the beam for Test case 1 is one end clamped and one end free. The load is concentrated tip force along a_3 varying from 0 to 150N as shown in Fig. 4-1. The comparisons of the tip position at a_1 and tip position at a_3 obtained form the present model, Ref. [17] model and DYMORE are shown in Figs. 4-2 and 4-3 respectively. It is seen that the present results correlate very well with the other results.

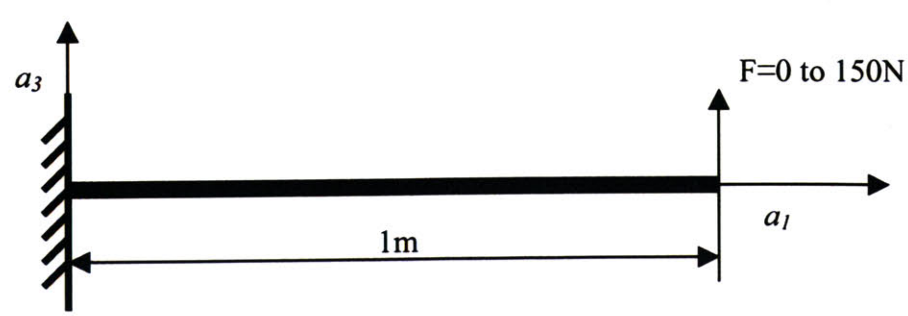


Figure 4-1: Beam model for Test case 1

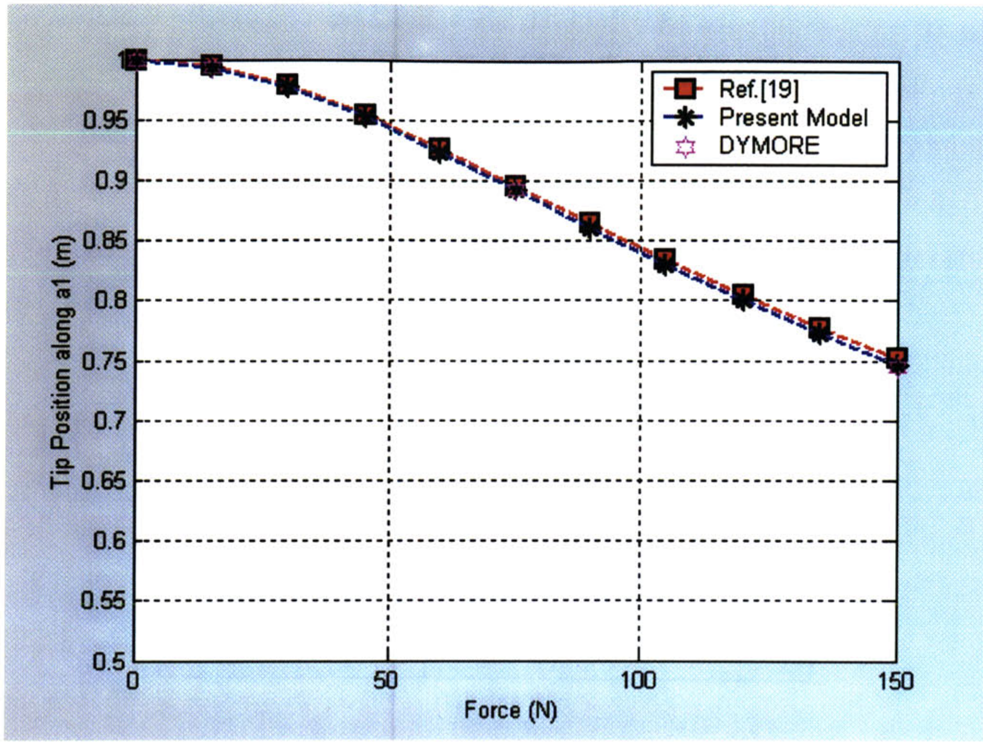


Figure 4-2: Comparison of tip position at a_1 for Test case 1

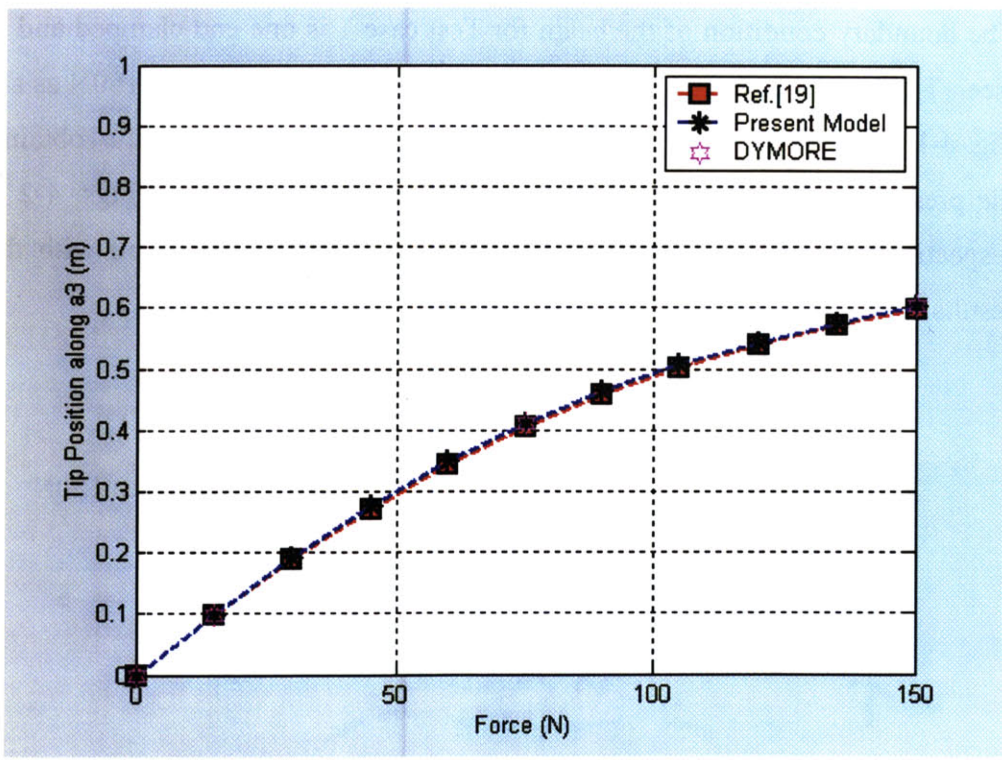


Figure 4-3: Comparison of tip position at a_3 for Test case 1

4.2.1.2 Test case 2

The boundary condition of the beam for Test case 2 is one end clamped and one end free as shown in Fig. 4-4. The load is tip moment along a_2 varying from 0 to −90Nm. The comparisons are plotted in Figs. 4-5 and 4-6. The results are consistent with each other.

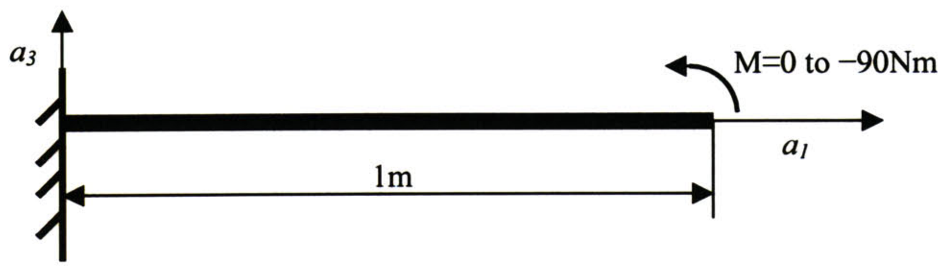


Figure 4-4: Beam model for Test case 2

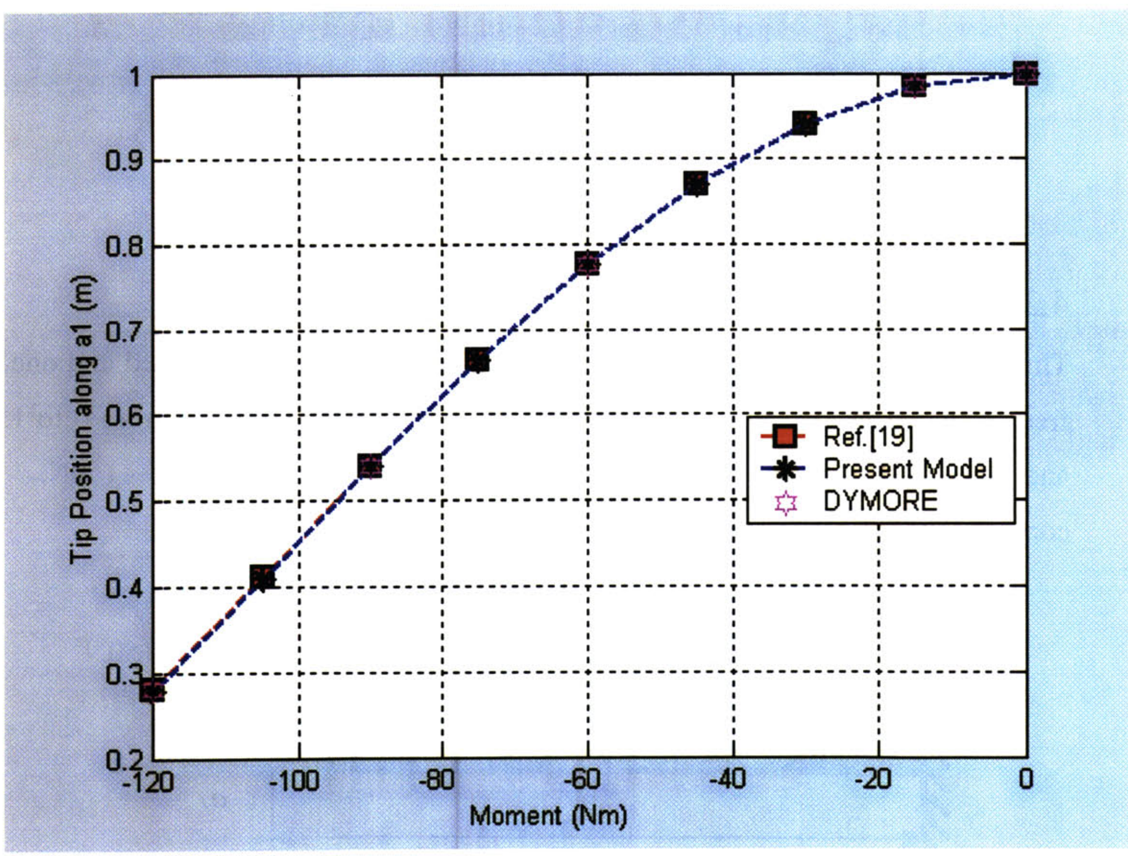


Figure 4-5: Comparison of tip position at a_1 for Test case 2

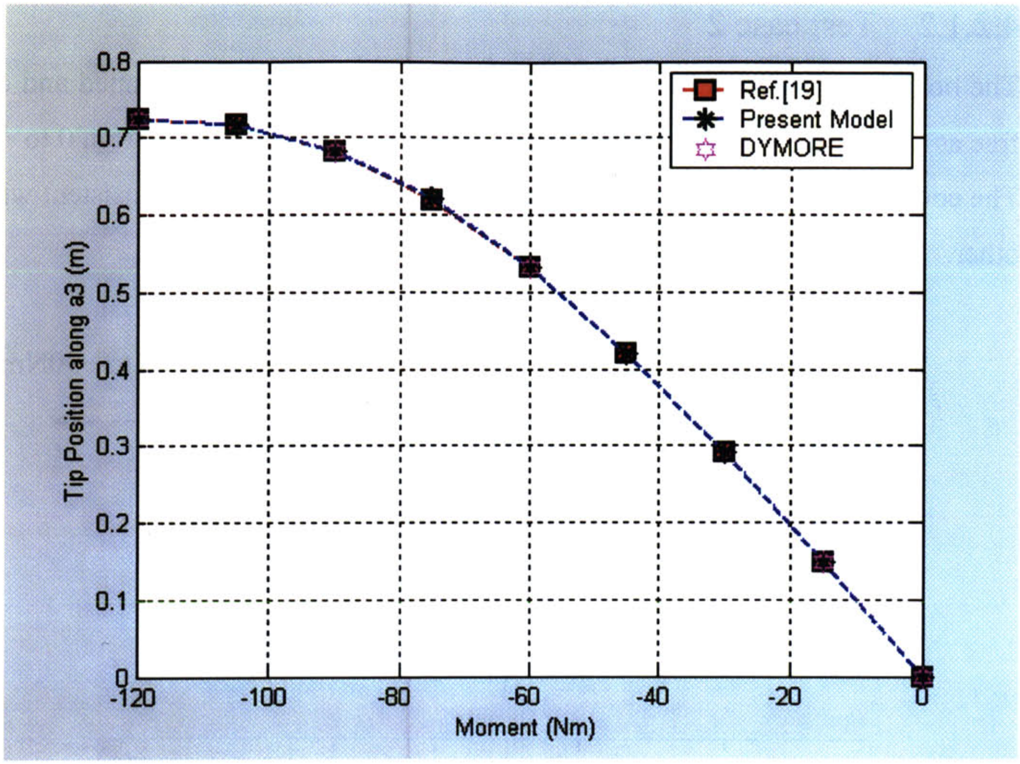


Figure 4-6: Comparison of tip position at a_3 for Test case 2

4.2.1.3 Test case 3

The boundary condition of the beam for Test case 3 is one end clamped and one end free as shown in Fig. 4-7. The loads are tip force along a_3 varying from 0 to 150N and force along a_3 in the middle of the beam with the value of –150N. The comparisons are plotted in Figs. 4-8 and 4-9. The three results overlap.

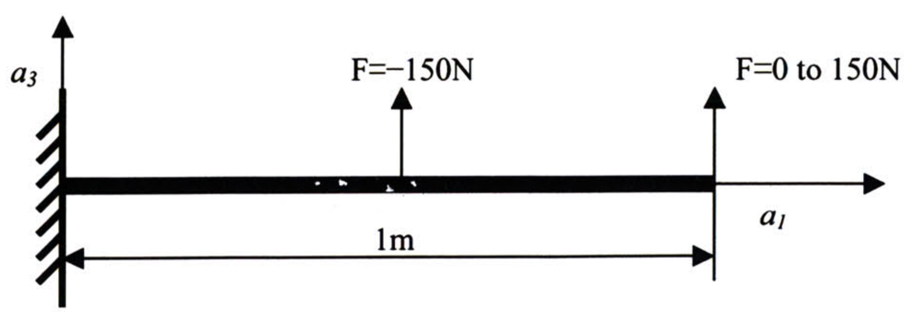


Figure 4-7: Beam model for Test case 3

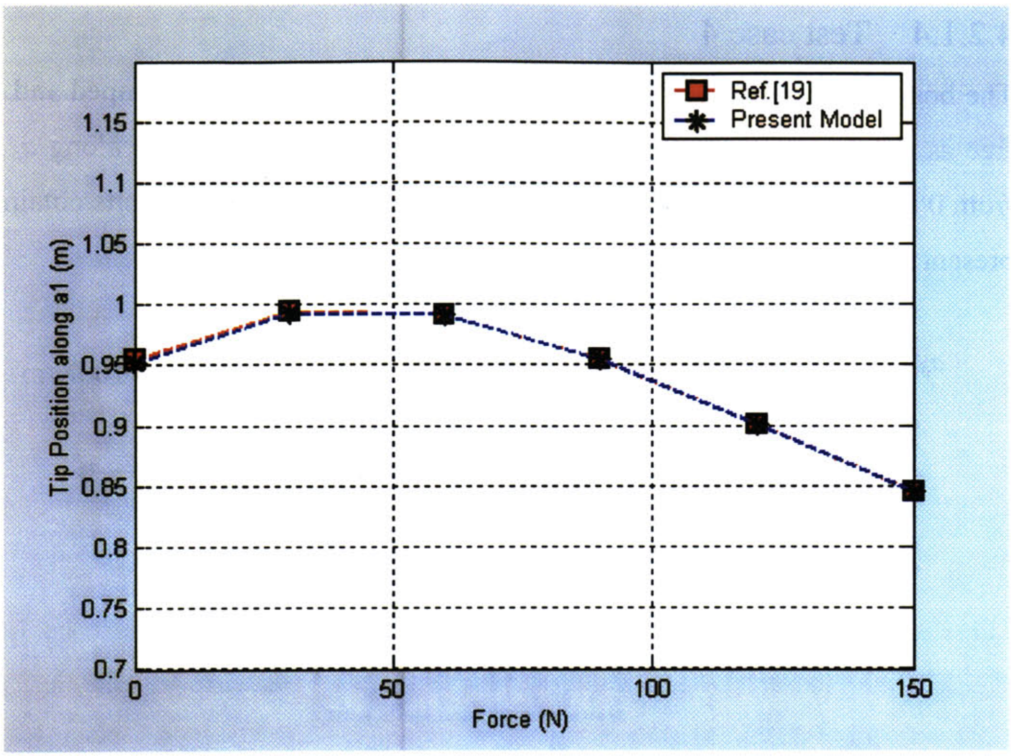


Figure 4-8: Comparison of tip position at a_1 for Test case 3

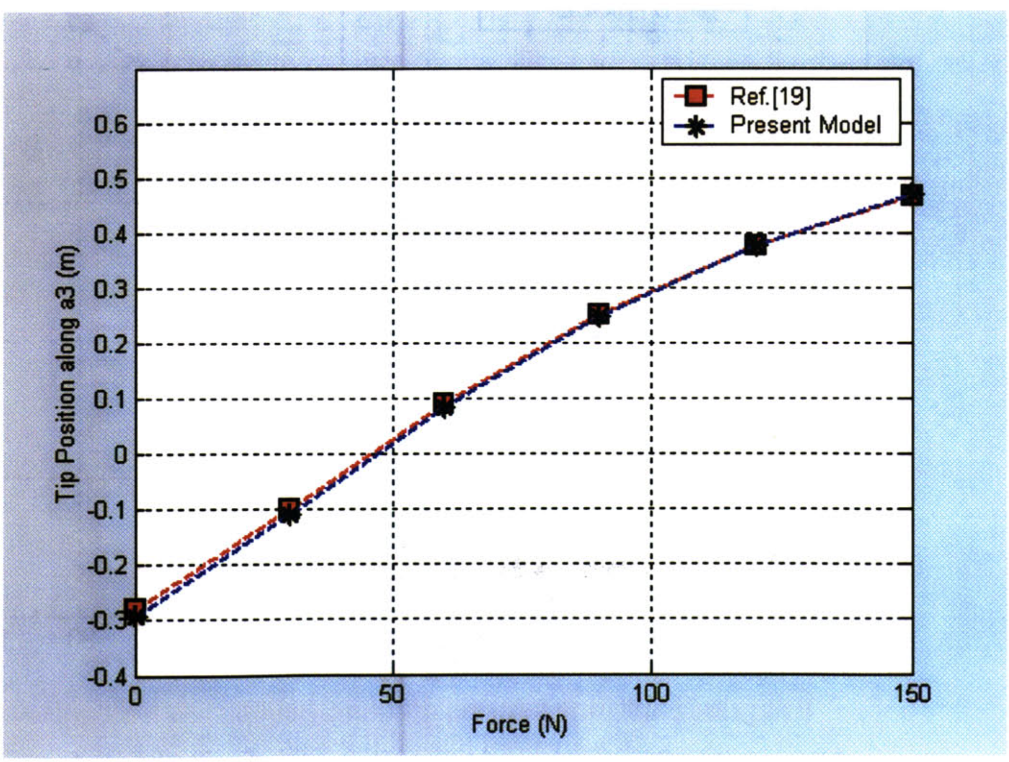


Figure 4-9: Comparison of tip position at a_3 for Test case 3

4.2.1.4 Test case 4

The boundary condition of the beam for Test case 4 is one end clamped and one end free as shown in Fig. 4-10. The loads are evenly distributed force along a_3 varying from 0 to 100N/m. The comparison is plotted in Fig. 4-11. The results obtained from present model overlap those obtained from DYMORE.

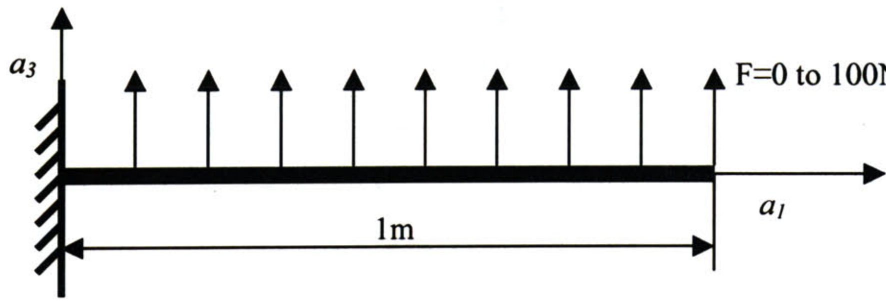


Figure 4-10: Beam model for Test case 4

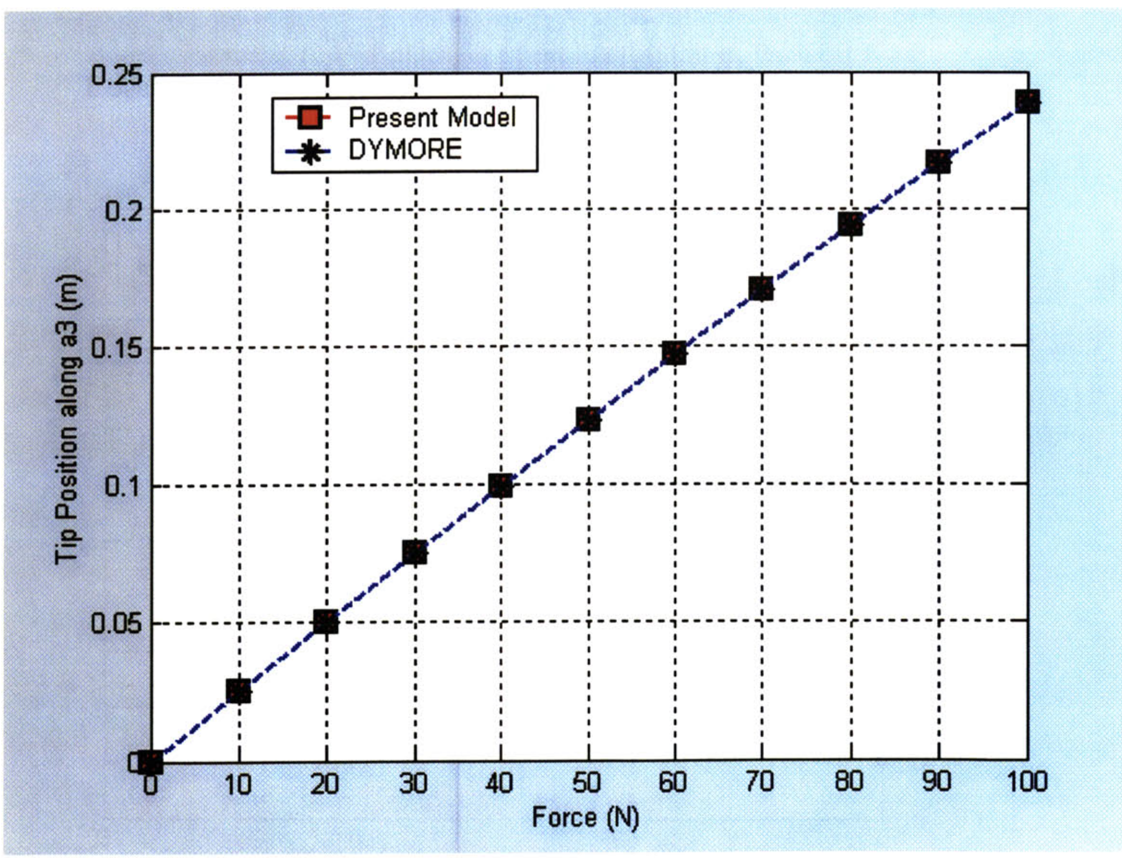


Figure 4-11: Comparison of tip position at a_3 for Test case 4

4.2.1.5 Test case 5

The boundary condition of the beam for Test case 5 is simply supported at both ends as shown in Fig. 4-12. The load is at the middle of the beam with the value from 0 to 1000N. The comparison displacement at the middle of the beam is plotted in Fig. 4-13.

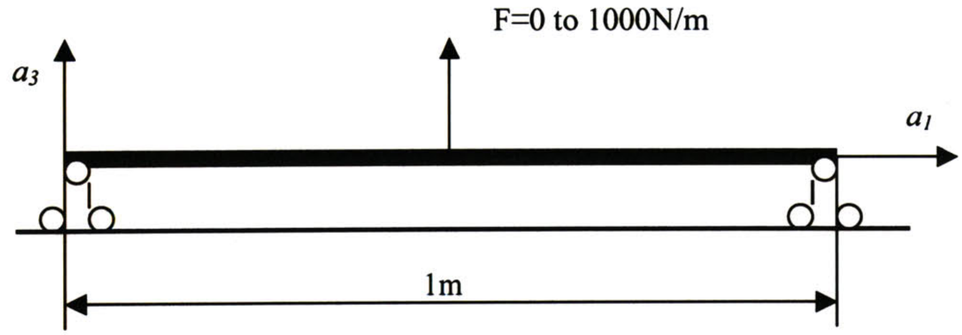


Figure 4-12: Simply supported beam model

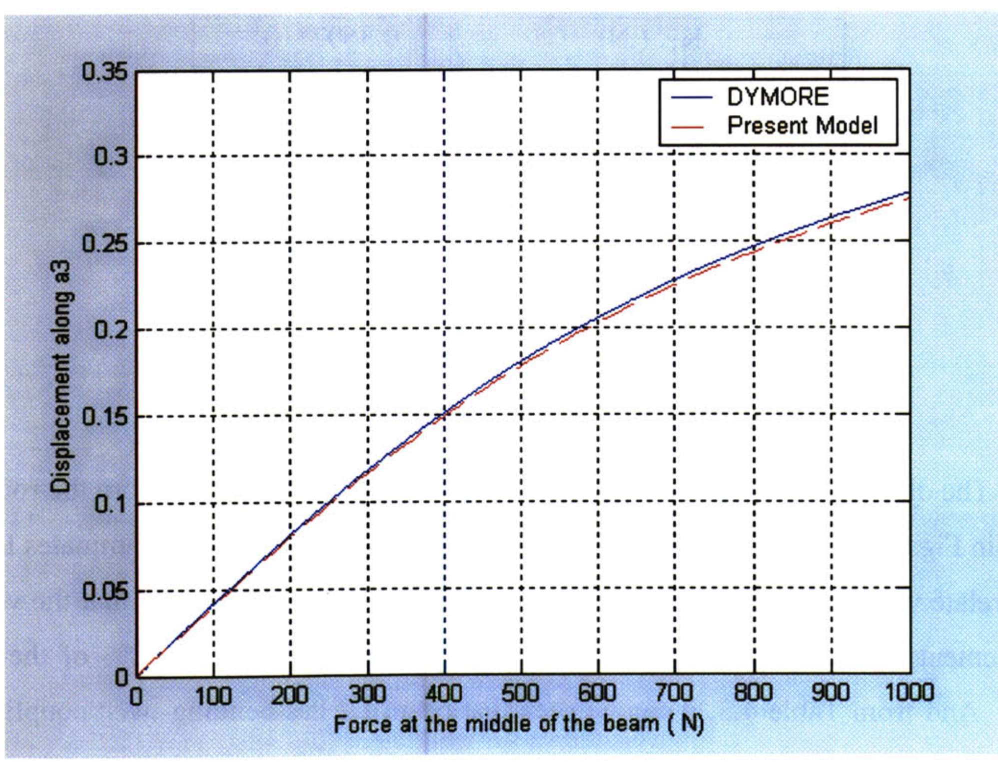


Figure 4-13: Comparison of displacement (m) at the middle of the beam for Test case 5

4.2.2 Composite Beam Deflection Test

This group of beams taken from the experiments of Ref. [20] are made of AS4/3501-6 Graphite/Epoxy. There are two different laminates: $[0^o/90^o]_{3s}$ (L1) and $[45^o/0^o]_{3s}$ (L2), both with length 0.56 m; the rectangular cross section has a horizontal dimension of 0.3m. Their stiffness constants were reported in Ref. [21] and are reproduced in Tables 4.2 and 4.3.

Table 4.2: Stiffness of L1

K_{11} (N)	$0.3412*10^7$
K_{22} (N)	$1.0*10^{20}$
K_{33} (N)	$1.0*10^{20}$
K_{44} (Nm^2)	0.1901
K_{55} (Nm^2)	0.7677
K_{66} (Nm^2)	$0.2559*10^4$

Table 4.3: Stiffness of L2

K_{11} (N)	$0.3607*10^7$
K_{22} (N)	$1.0*10^{20}$
K_{33} (N)	$1.0*10^{20}$
K_{44} (Nm^2)	0.4096
K_{45} (Nm^2)	$0.9864*10^{-1}$
K_{55} (Nm^2)	0.5297
K_{66} (Nm^2)	$0.2628*10^4$

The displacements for this beam are measured at a station 0.5m from the root. As shown in Figs.4-14, 4-15, the displacements using the present model for laminates L1 and L2 correlate very well with the experiment from Ref. [20]. It can be seen that the vertical displacement at the tip when the maximum load is applied is about 30% of the beam length. And from Table 4.3, it can be seen that beam L2 has bending-twist coupling. It shows that the formulation and numerical procedure perform very well for composite beams with large deformation.

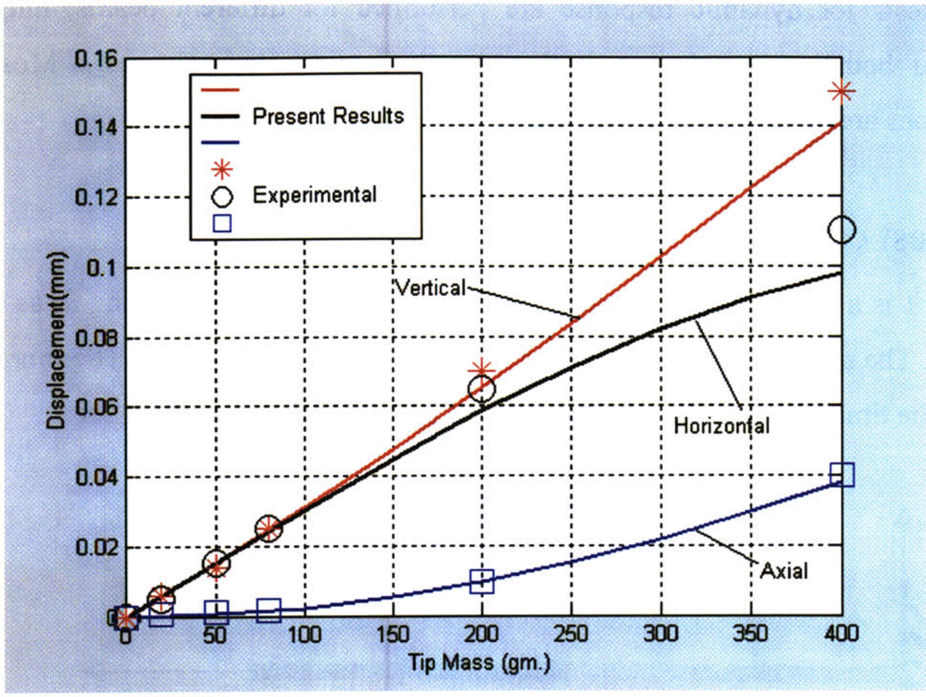


Figure 4-14:Tip Displacements for a $[0/90]_{3s}$ beam with its root at 45°

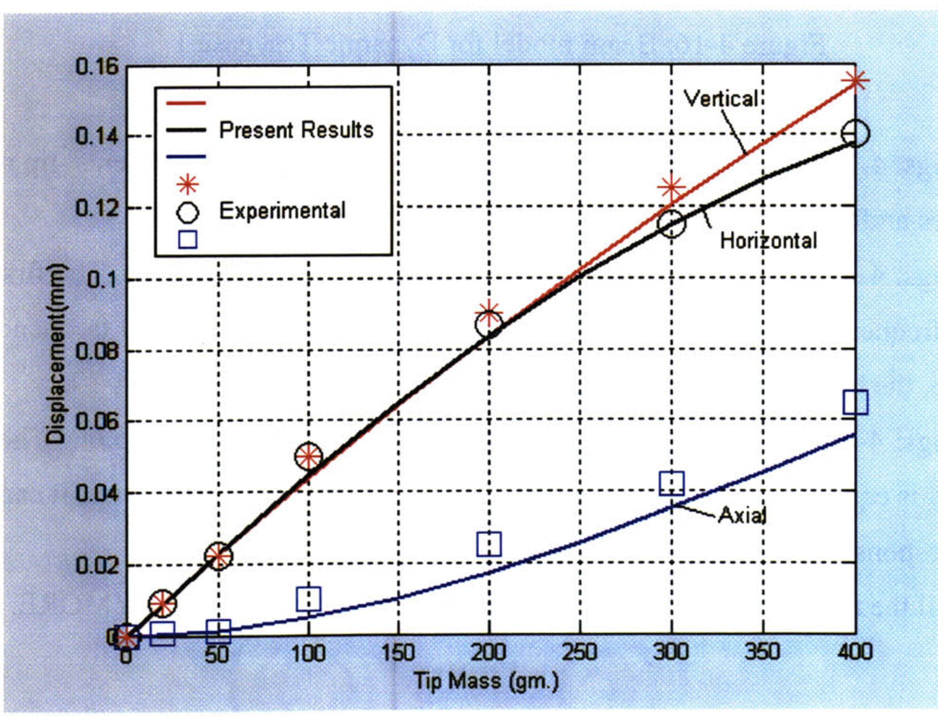


Figure 4-15:Tip Displacements for a $[45/0]_{3s}$ beam with its root at -45°

4.3 Dynamic Test

Various tests for dynamic response are performed for different beams, hingeless or articulated boundary conditions, and nonrotating and rotating cases. Most of the comparisons are conducted between the present model and DYMORE.

4.3.1 Test case 1

Test case 1 is a nonrotating beam with different tip dynamic forces along a_3 as shown in Fig. 4-16. The material properties of this beam are shown in Table 4.1. The time step size used for the time integration in DYMORE and present model is $1.0*10^{-3}$sec.

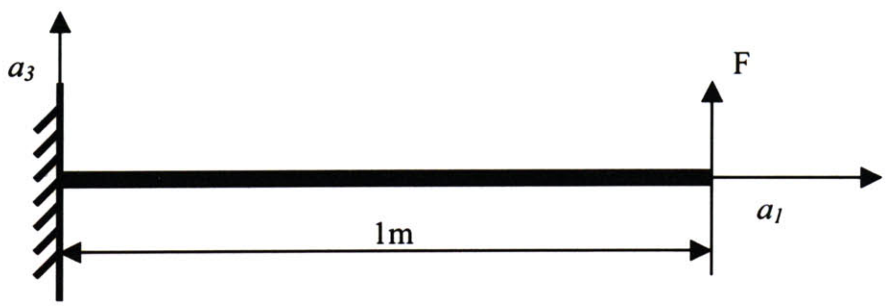


Figure 4-16: Beam model for Dynamic Test case 1

Figs. 4-17 to Fig. 4-20 present the comparisons of tip displacements, tip rotations, root forces and root moments when the applied force F is 10sin20t.

Figs. 4-21 to Fig. 4-24 are when the applied force F is 10sin50t. The first natural bending frequency of this beam is 55.6Hz which is close to the excited frequency 50Hz. Therefore, the beating phenomena can been seen from Fig. 4-16 to Fig. 4-19.

Figs. 4-25 to Fig. 4-38 are when the applied force F is 10sin55.6t. The excited frequency is equal to the natural frequency that results in resonance. Hence, it can be seen that the response of this case is unstable.

All the results are consistent with the solutions obtained from DYMORE.

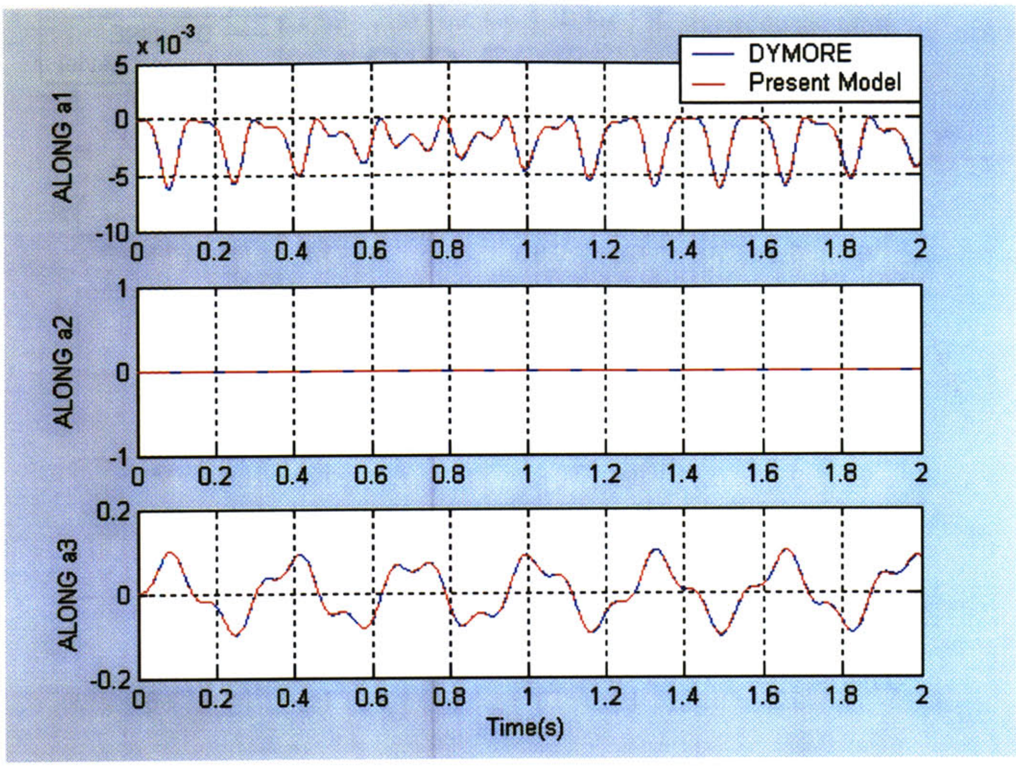


Figure 4-17: Tip displacements (m) comparison for F=10sin20t

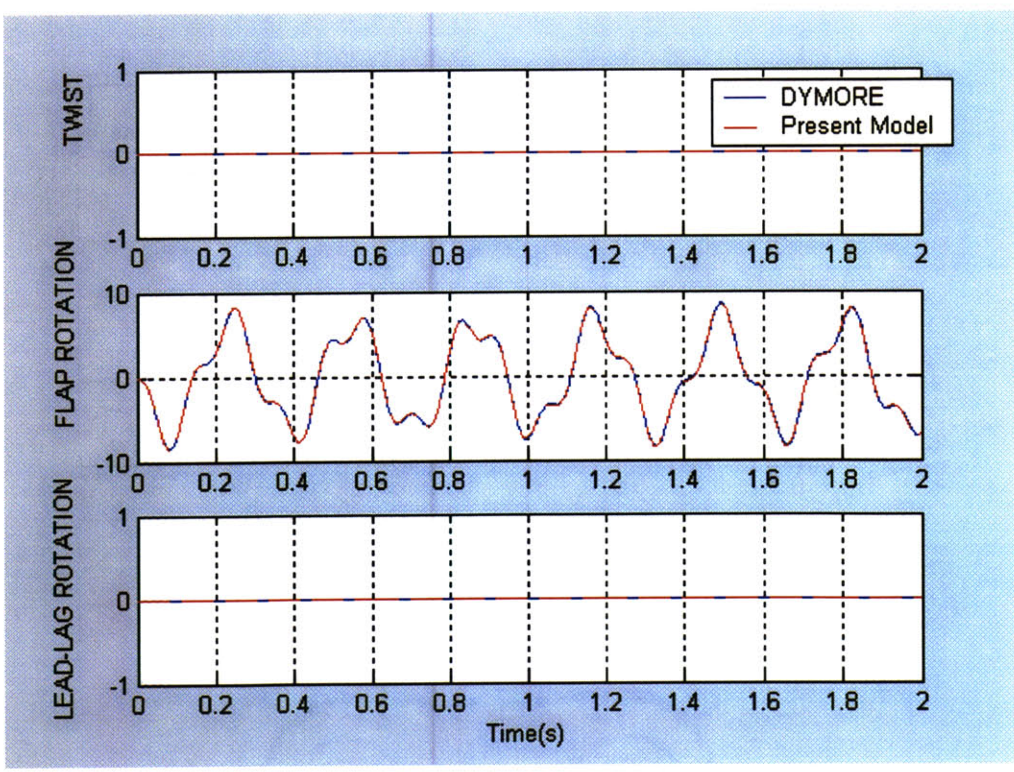


Figure 4-18: Tip rotations (degree) comparison for F=10sin20t

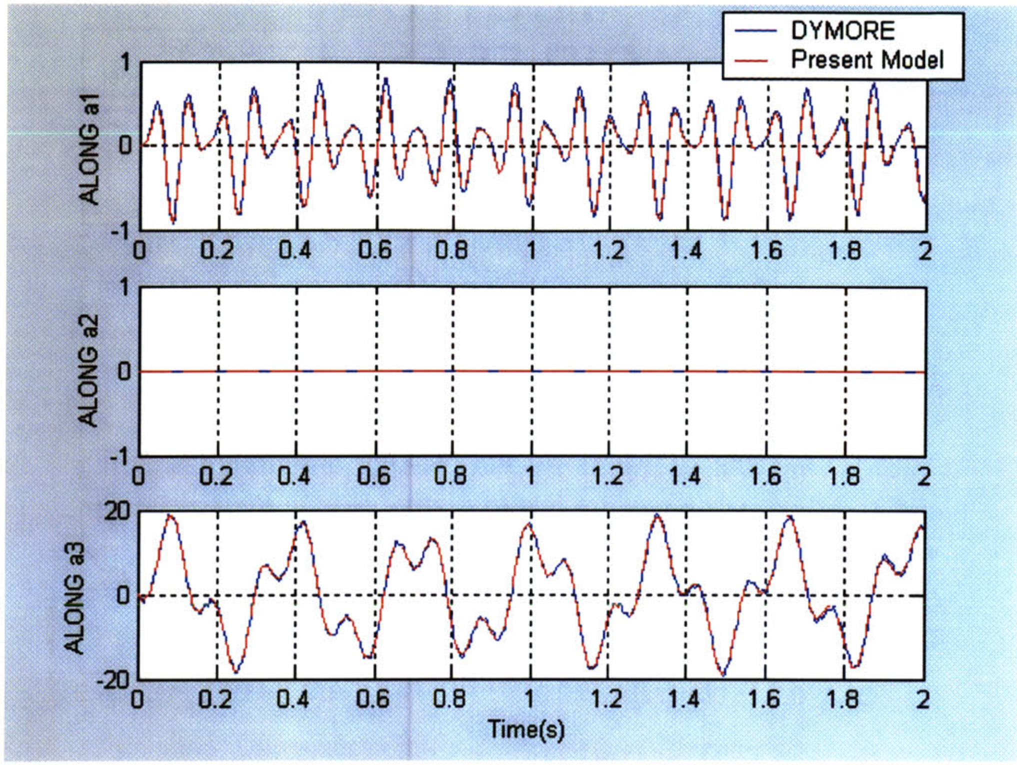


Figure 4-19: Root forces (N) comparison for F=10sin20t

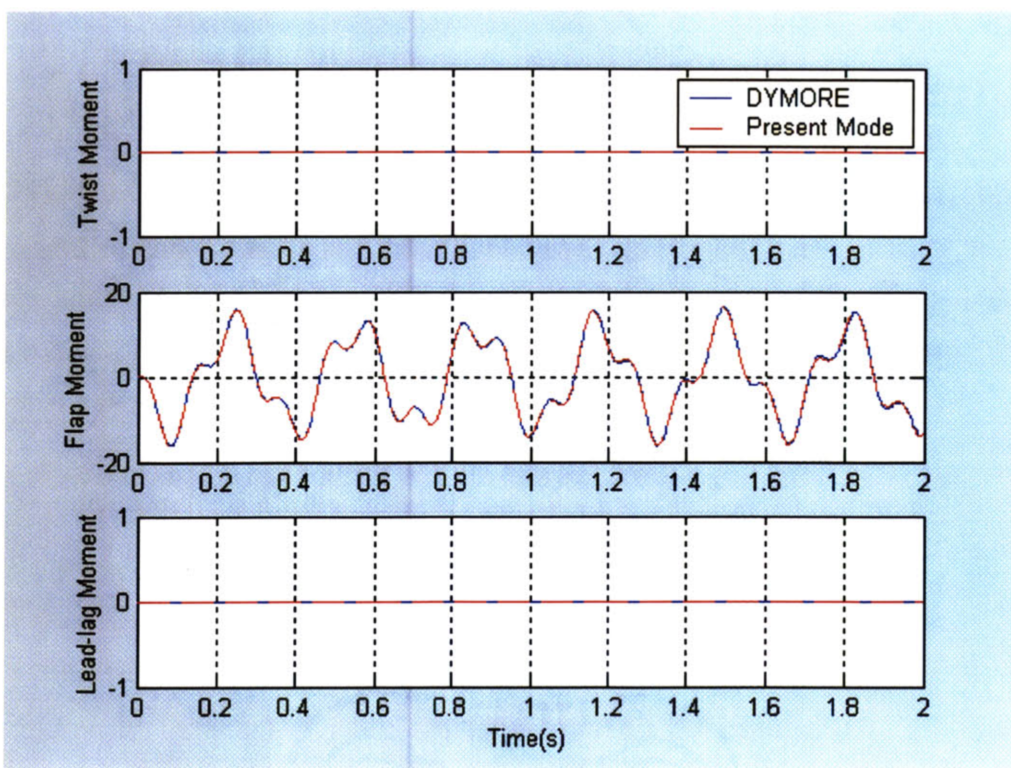


Figure 4-20: Root moments (Nm) comparison for F=10sin20t

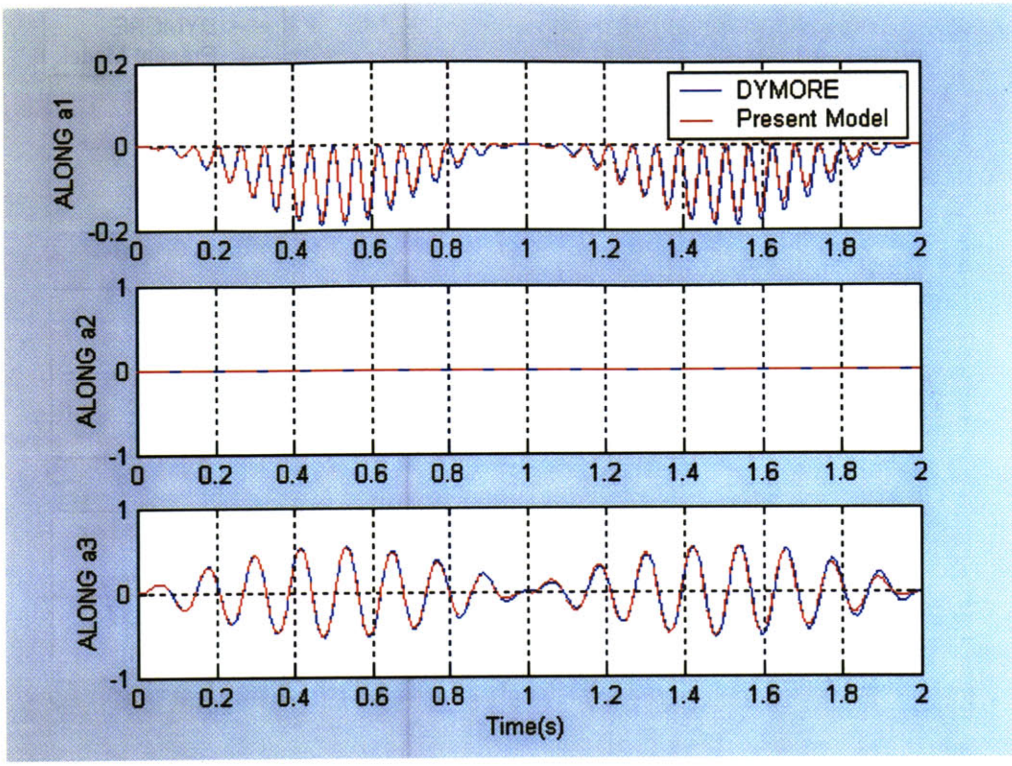


Figure 4-21: Tip displacements (m) comparison for F=10sin50t

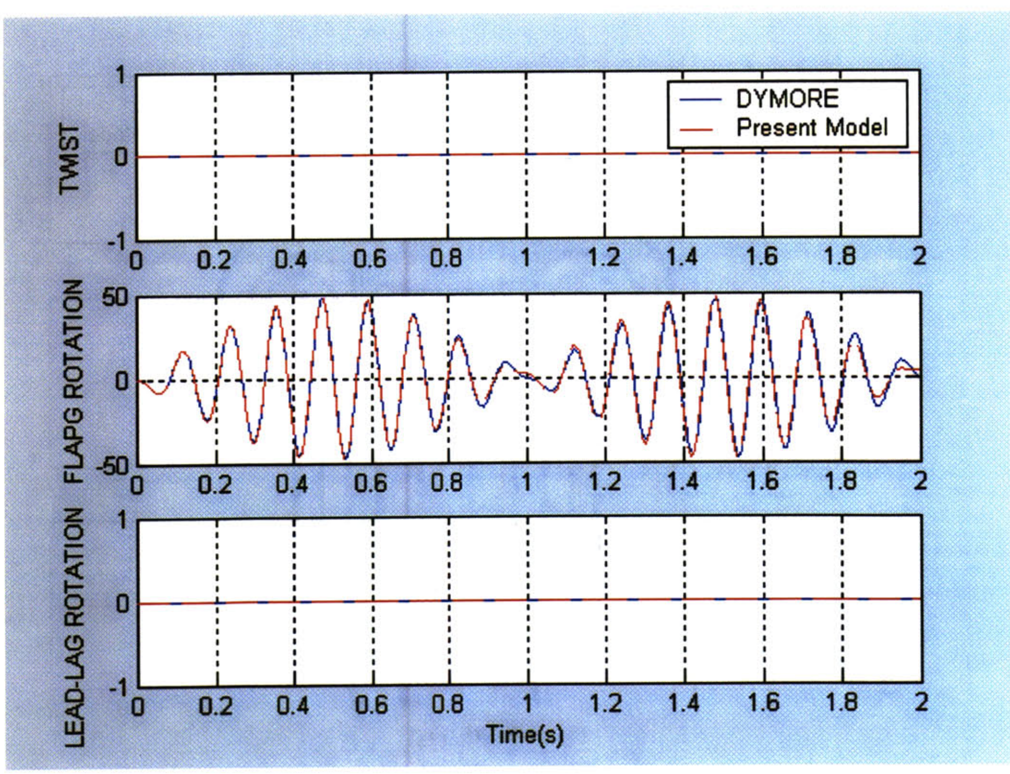


Figure 4-22: Tip rotations (degree) comparison for F=10sin50t

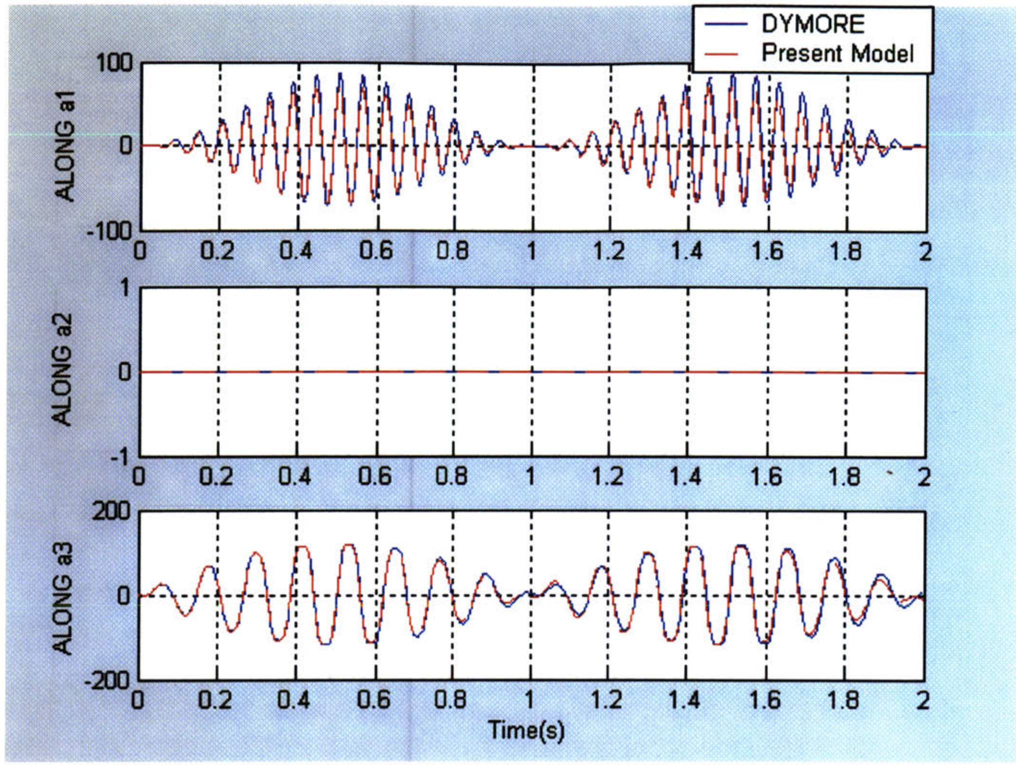


Figure 4-23: Root forces (N) comparison for F=10sin50t

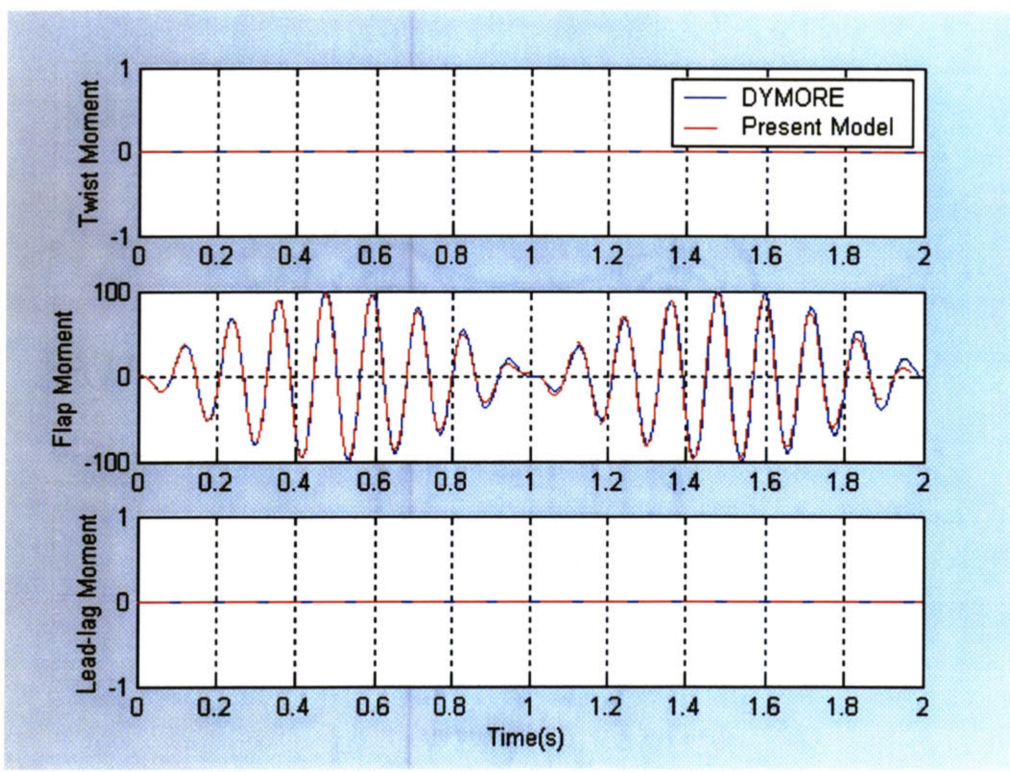


Figure 4-24: Root moments (Nm) comparison for F=10sin50t

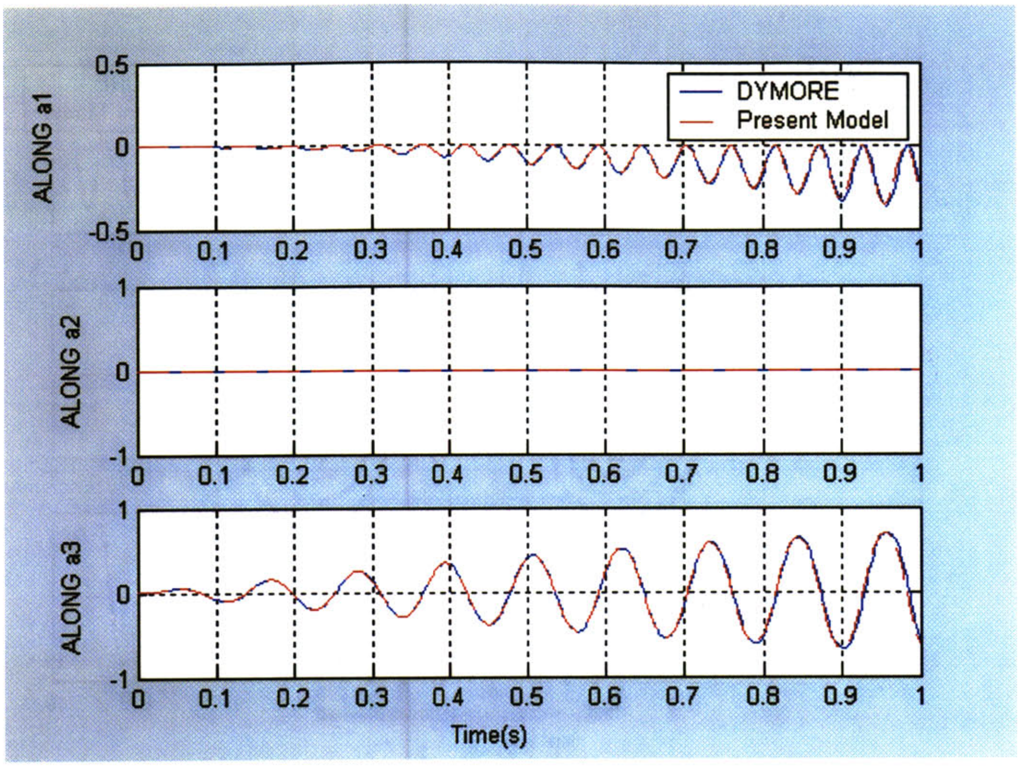


Figure 4-25: Tip displacements (m) comparison for F=10sin55.6t

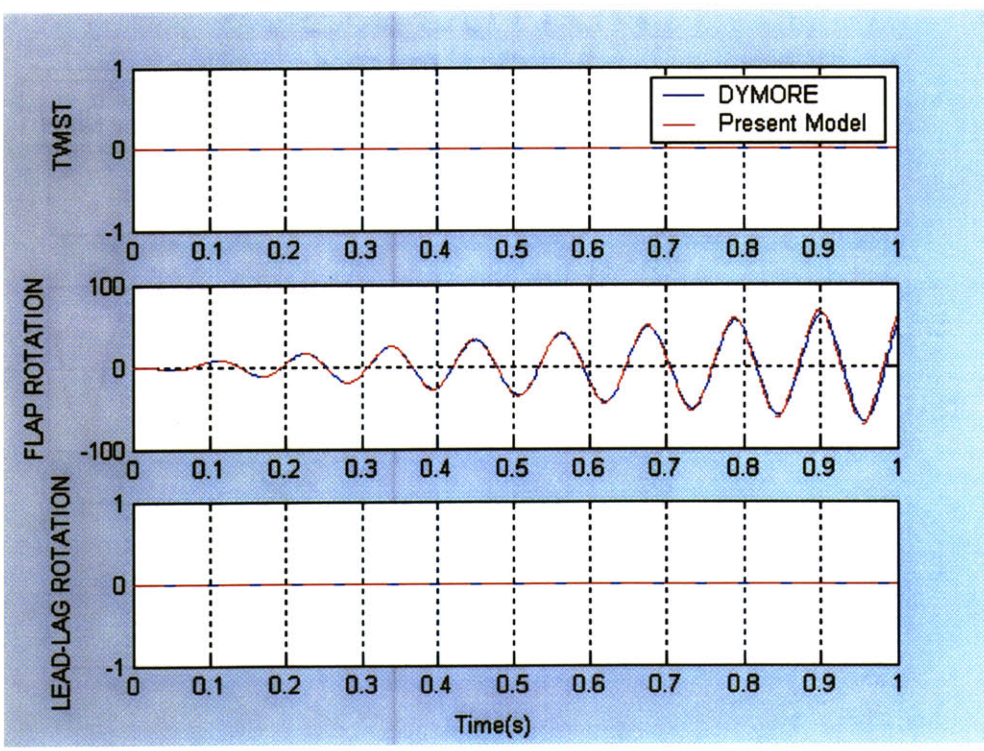


Figure 4-26: Tip rotations (degree) comparison for F=10sin55.6t

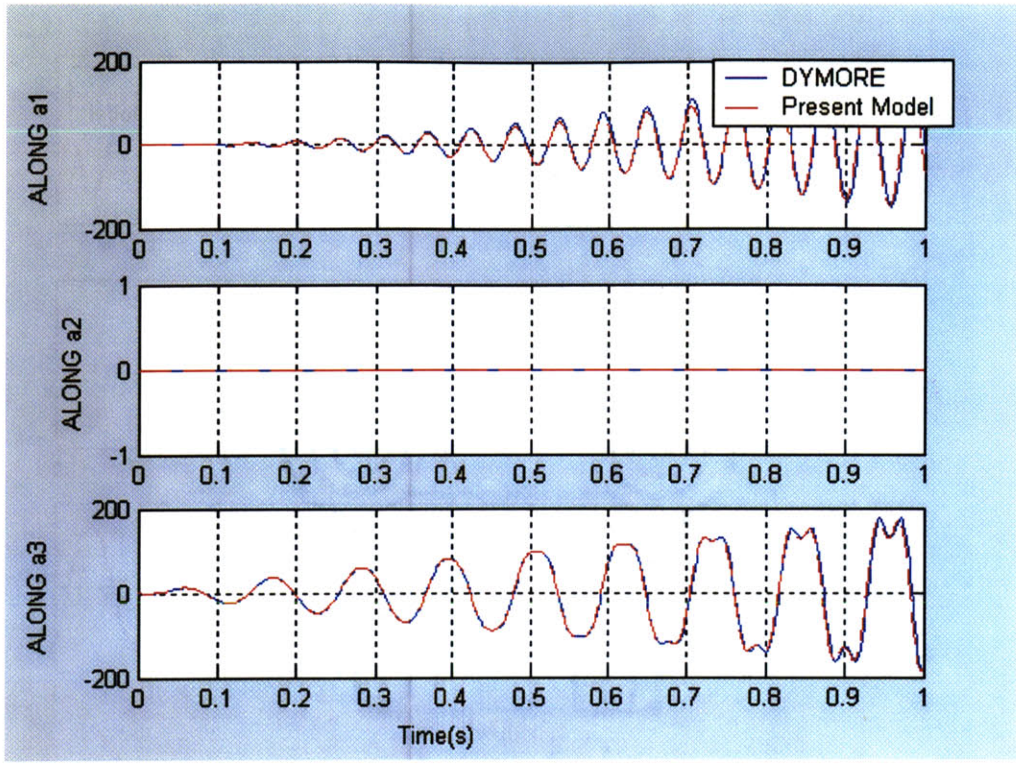


Figure 4-27: Root forces (N) comparison for F=10sin55.6t

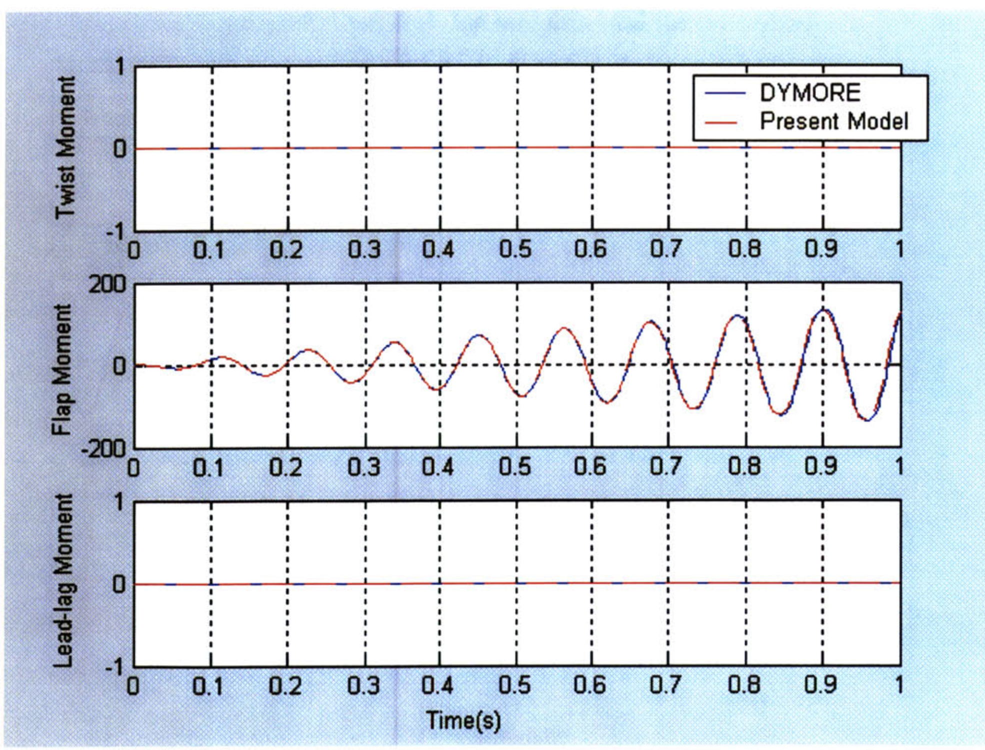


Figure 4-28: Root moments (Nm) comparison for F=10sin55.6t

4.3.2 Test case 2

Test case 2 deals with the nonrotating beam used in Ref. [17]. It is a 2.4-m long uniform straight beam articulated at the root, so as to allow rotation about the a_2 axis, and free at the tip as shown in Fig. 4-29. The material properties of this beam are shown in Table 4.4.

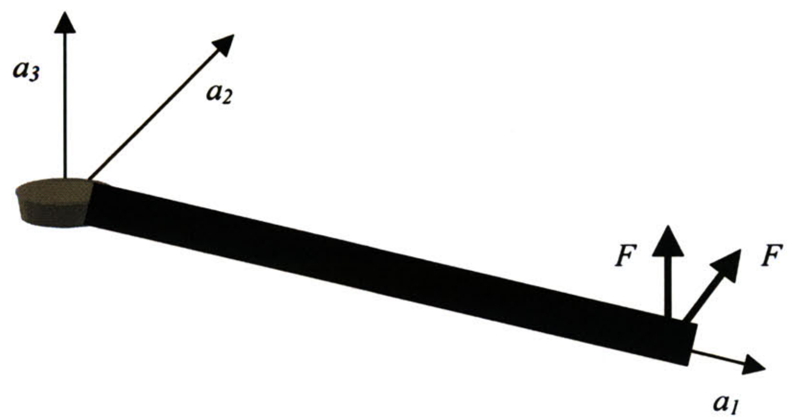


Figure 4-29: Beam model for Dynamic Test case 2

The applied load consists of a triangular pulse tip load, starting at t=0, peaking at t=0.0025 and terminating at t=0.05s, with 1000N peak components in both the a_2 and a_3 directions as shown in Fig. 4-30.

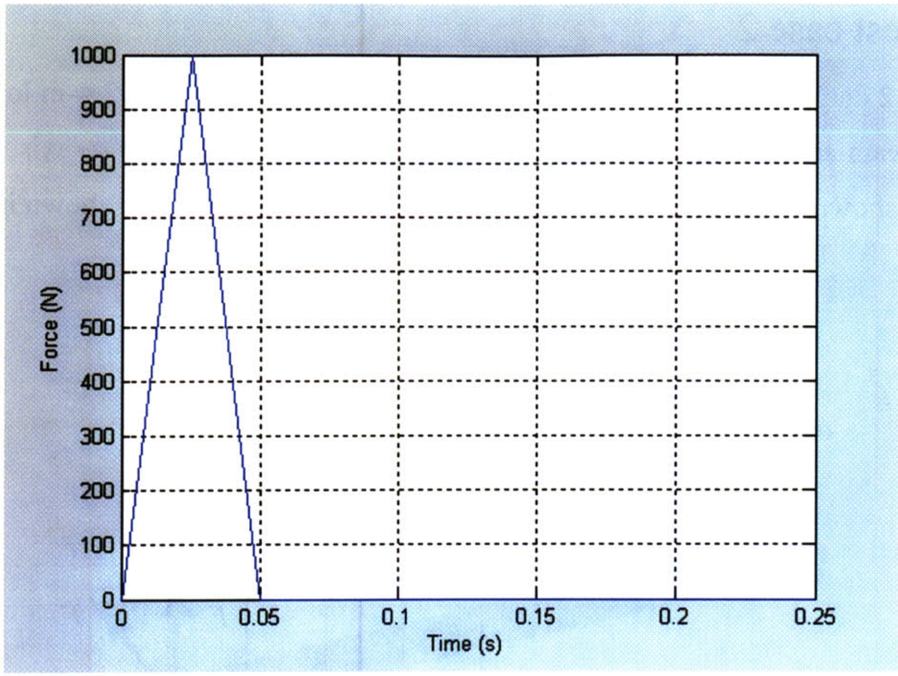


Figure 4-30: Tip force applied in both the a_2 and a_3 directions for Dynamic Test case 2

Table 4.4: Material Properties for Dynamic Test case 2

mass per length	1.60920 kg/m
Ixx	$1.19092*10^{-2}$ kg m
Iyy	$8.60200*10^{-4}$ kg m
Izz	$1.10490*10^{-2}$kg m
K_{11} (extension)	$4.35080*10^{7}$ N
K_{22}(Shear Stiffness in a_2 direction)	$1.40385*10^{7}$ N
K_{33}(Shear Stiffness in a_3 direction)	$2.80769*10^{6}$ N
K_{44} (twist)	$2.80514*10^{4}$ Nm^2
K_{55} (flat bend)	$2.32577*10^{4}$ Nm^2
K_{66} (chord bend)	$2.98731*10^{5}$ Nm^2

The results are consistent with the results by energy decaying method in Ref. [17] and the results obtained from DYMORE. The time step size used for the time integration in DYMORE and present model is $1.0*10^{-3}$sec. Figs. 4-31 and 4-32 present the comparisons of root transverse shear force and root torsional moment. These results indicates that the accuracy of the numerical procedure of the present model and its high frequency numerical dissipation characteristics are similar to the energy decaying method used in Ref. [17].

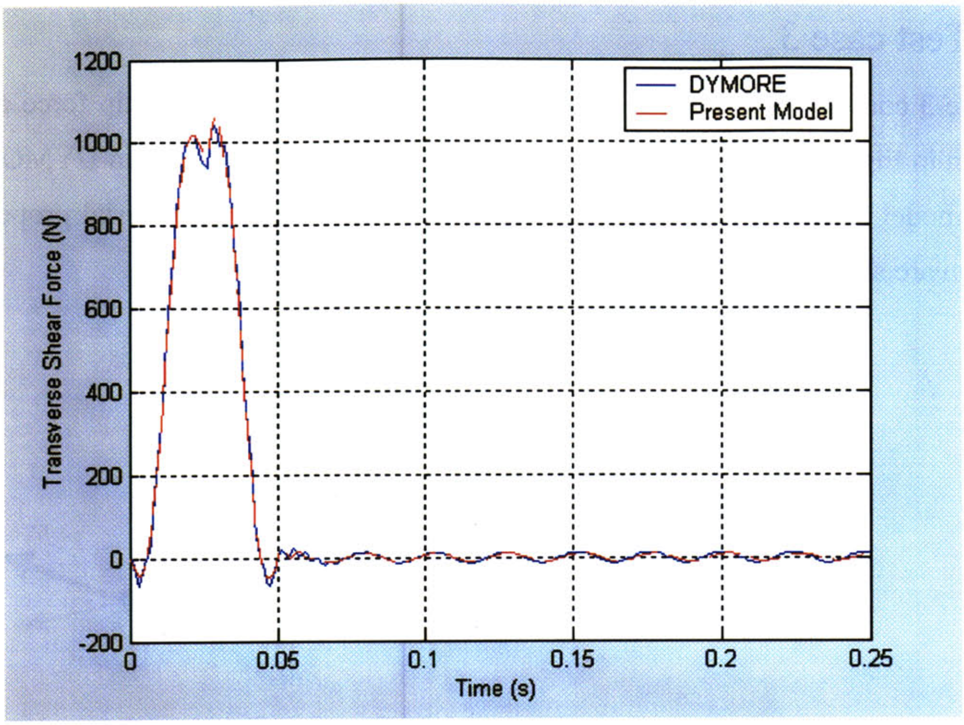


Figure 4-31: Root transverse shear forces (N) for Dynamic Test case 2

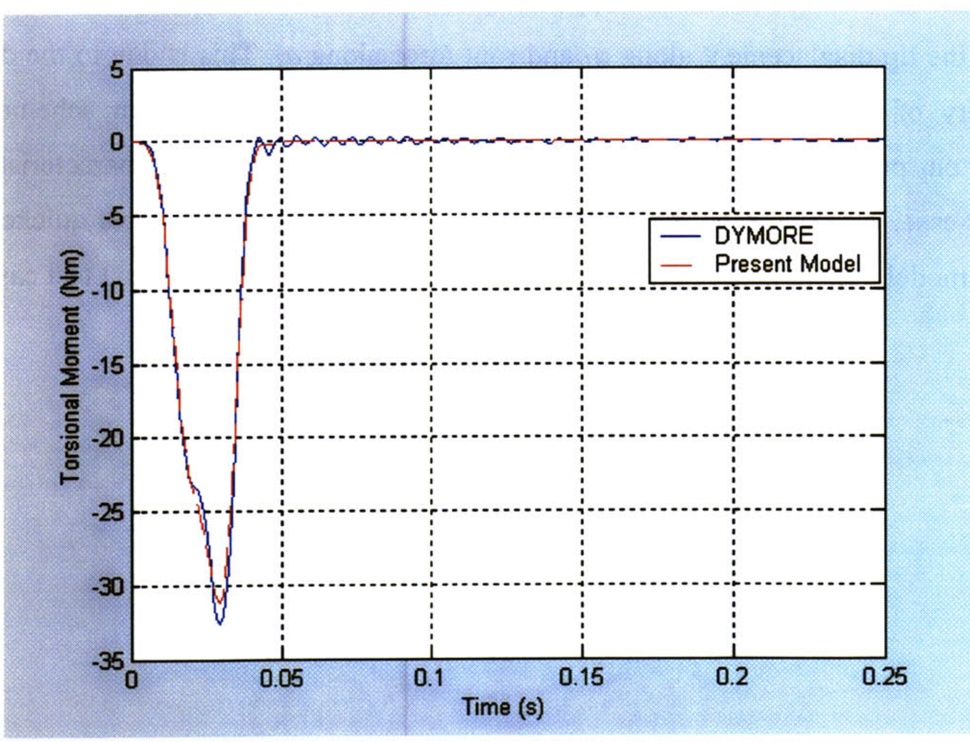


Figure 4-32: Root torsional moment (Nm) for Dynamic Test case 2

4.3.3 Test case 3

Test case 3 consists of a rotating beam clamped at the root and with a tip force along a_2 as shown in Fig. 4-33. The time step size used for the time integration in DYMORE and present model is $1.0*10^{-3}$sec. The rotating speed is 70 rad/s. The material properties of this beam are shown in Table 4.1.

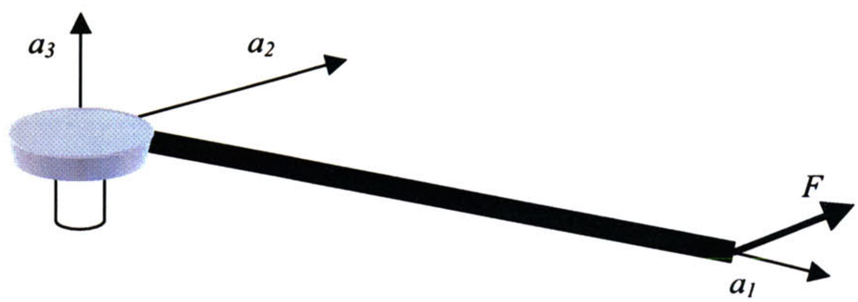


Figure 4-33: Beam model for Dynamic Test case 3

Figs. 4-34 to 4-37 present the comparisons of tip displacements, tip rotations, root forces, and root moments with the applied tip force of 10sin20t. Some differences can be seen in the tip displacement along a_1 and root force along a_1. This is due to the different capability of high frequency dissipation between the two integration schemes. As pointed out previously, both of these schemes have energy decaying characteristics. In this test case, it can be seen that the high frequency component dissipates quicker in the present model than in DYMORE. This will be further discussed in the next test case.

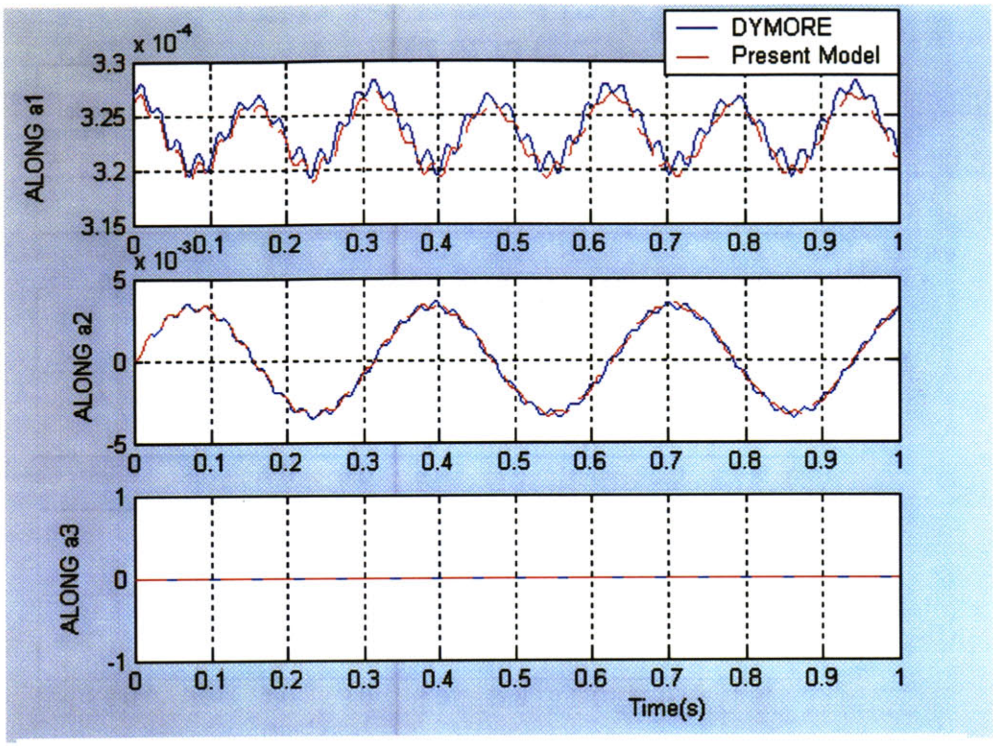


Figure 4-34: Tip displacements (m) for Dynamic Test case 3

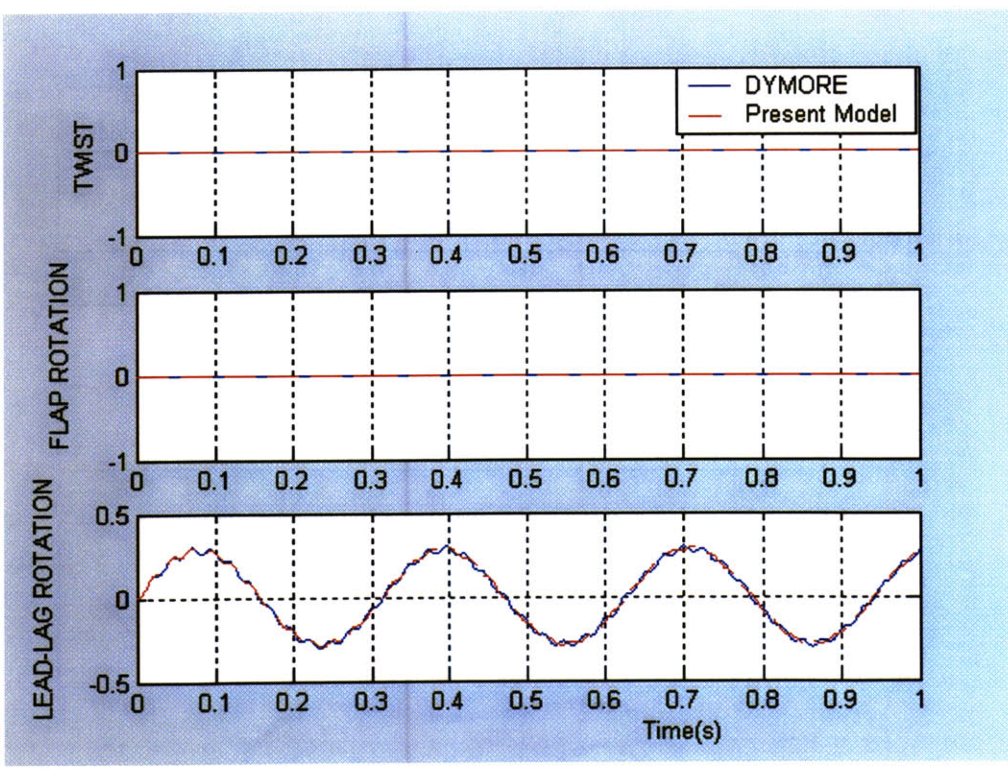


Figure 4-35: Tip rotations (degree) for Dynamic Test case 3

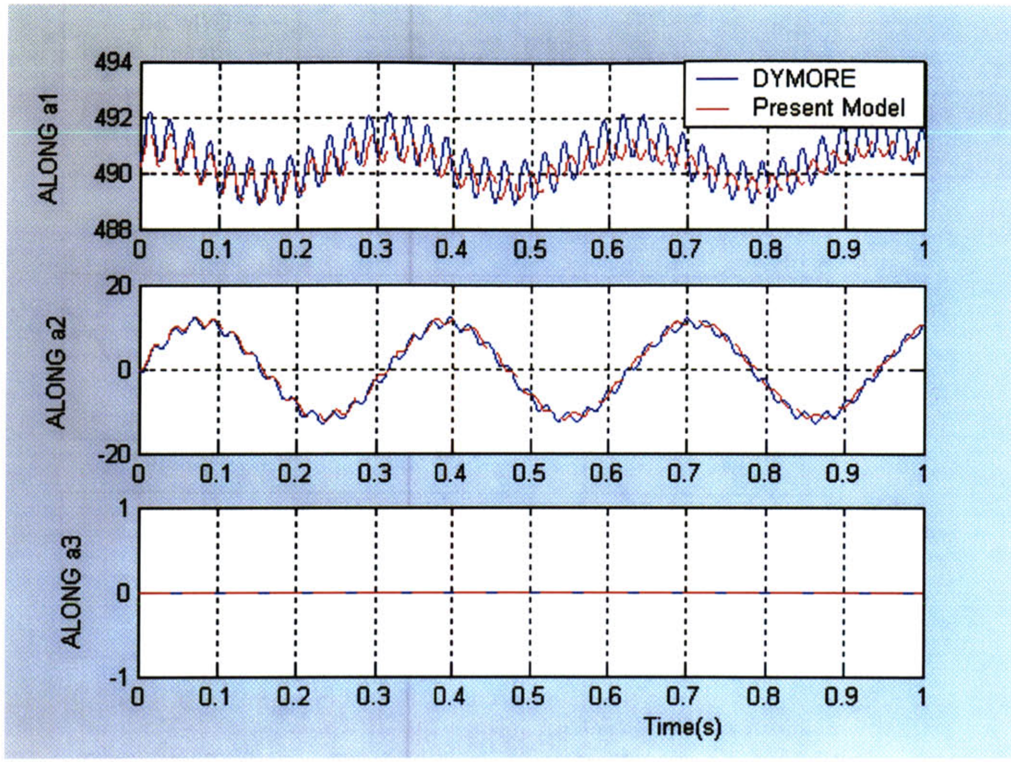


Figure 4-36: Root forces (N) for Dynamic Test case 3

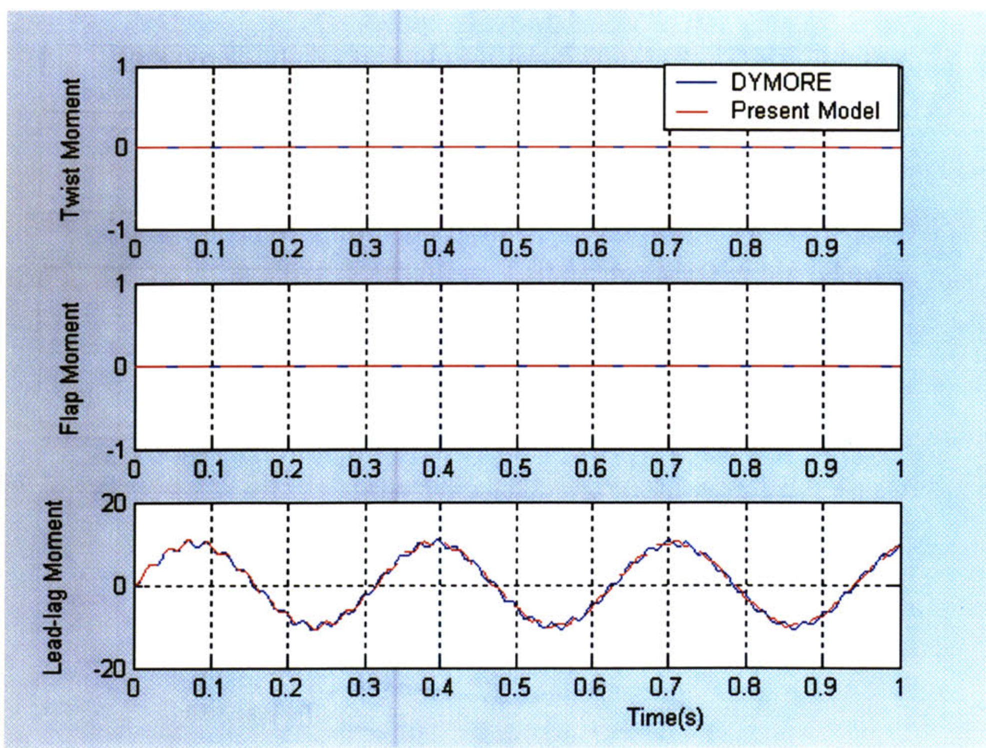


Figure 4-37: Root moments (Nm) for Dynamic Test case 3

4.3.4 Test case 4

Test case 4 is the same as Test case 3 but for longer duration. Figs. 4-38 to 4-41 present the comparisons of tip displacements, tip rotations, root forces, and root moments for 30sec duration. Figs. 4-42 and 4-43 present the zoom in plot of the root force and tip displacement between 3sec to 4sec. It can be seen that the high frequency component in present model has been dissipated away by 3sec while it still exists in DYMORE. And Figs. 4-44 and 4-45 present the zoom in plot of the root force and tip displacement between 29sec to 30sec. A phase lag can be seen from both plots. But the period elongation and amplitude decay are not obvious.

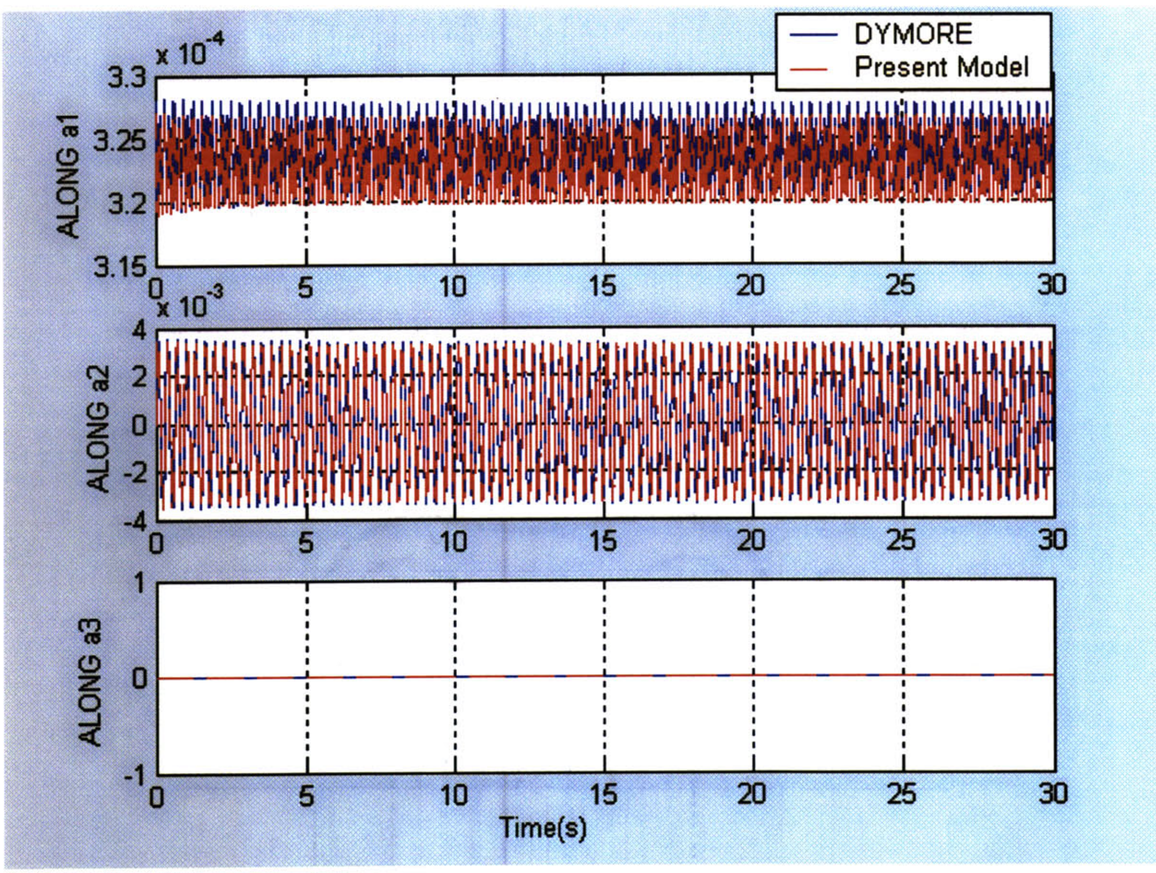


Figure 4-38: Tip displacements (m) for Dynamic Test case 4

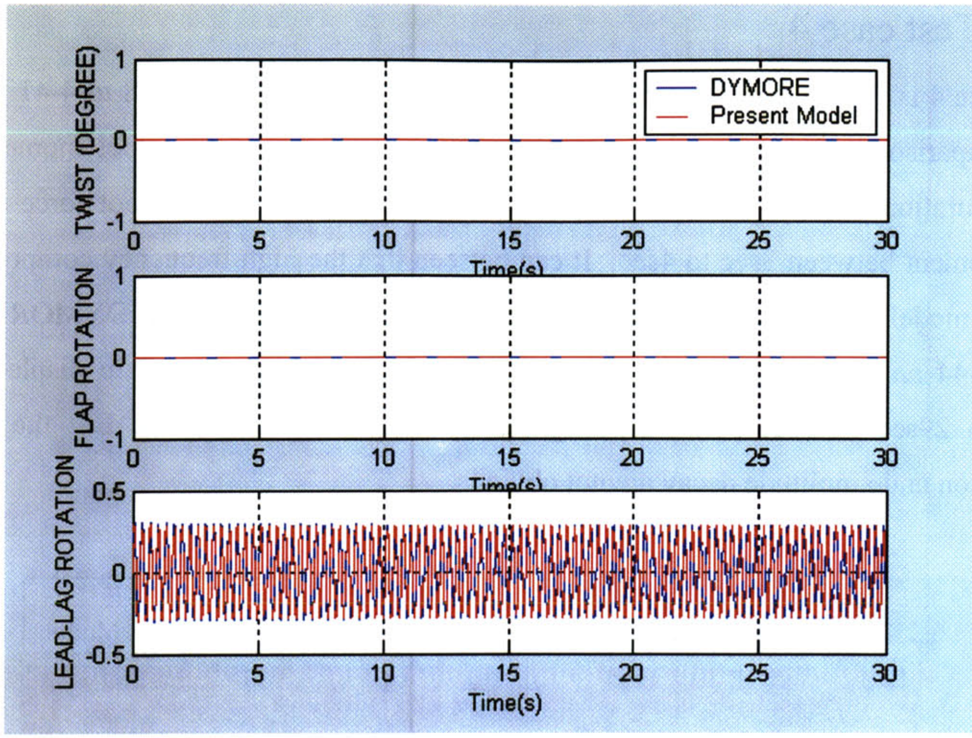


Figure 4-39: Tip rotations (degree) for Dynamic Test case 4

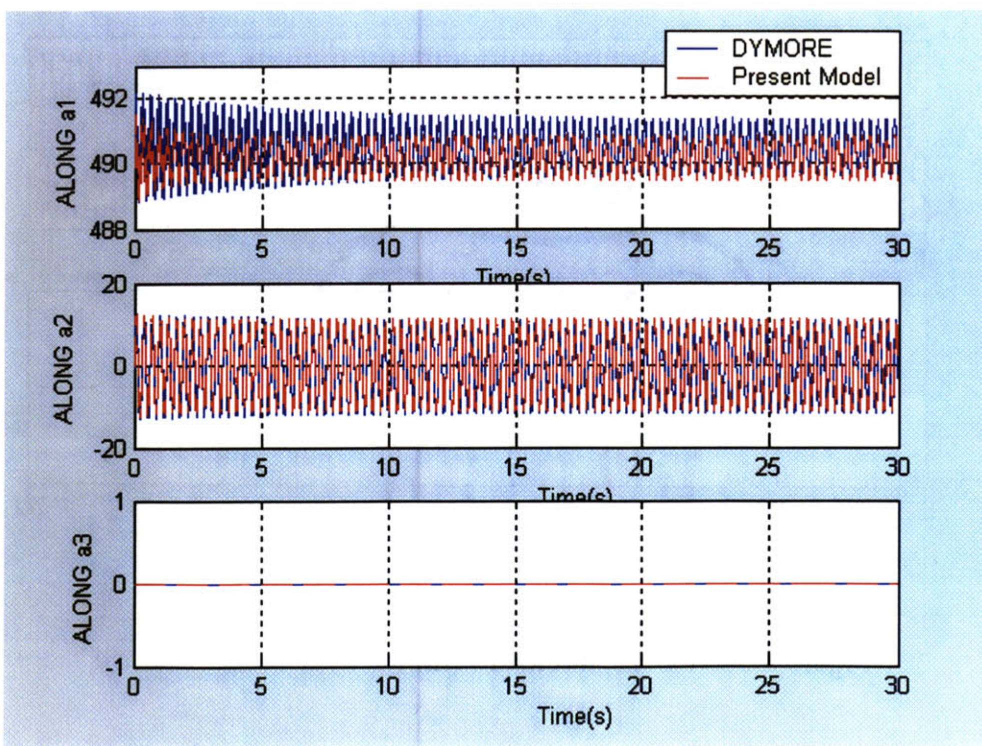


Figure 4-40: Root forces (N) for Dynamic Test case 4

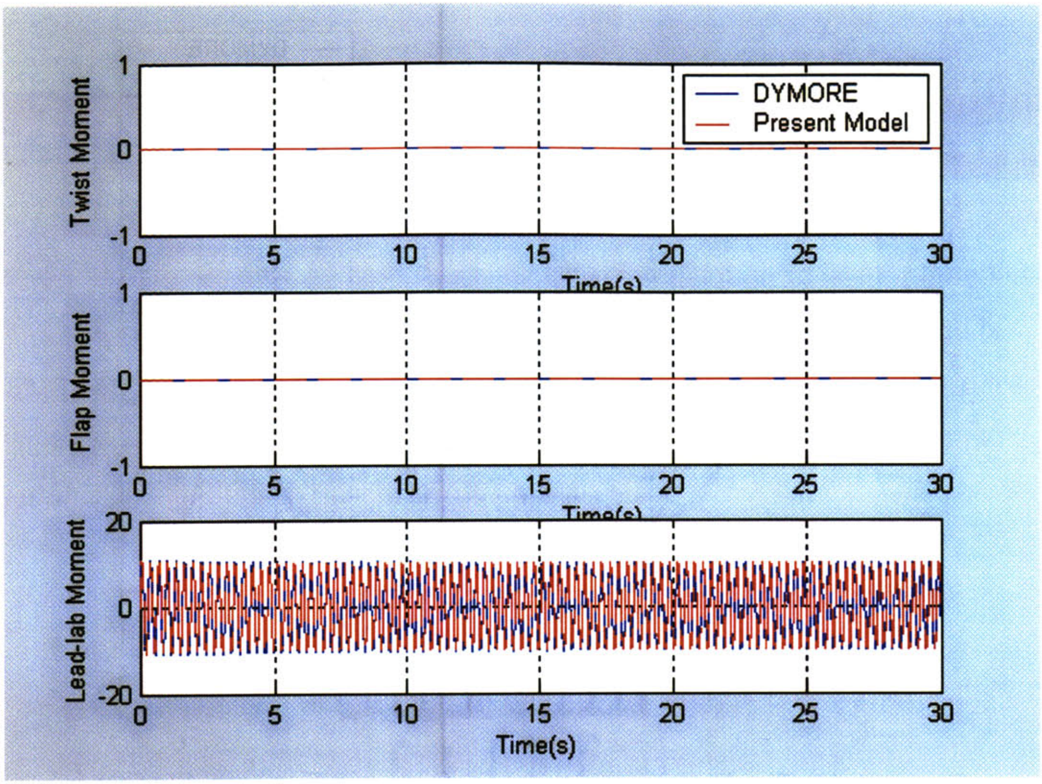


Figure 4-41: Root moments (Nm) for Dynamic Test case 4

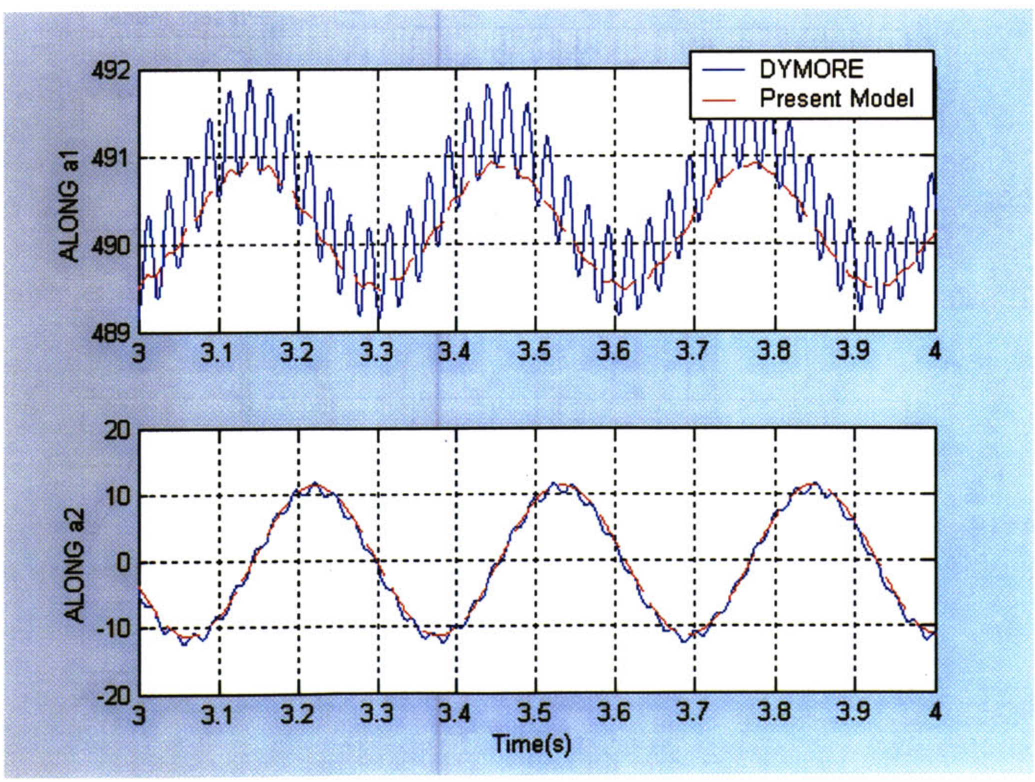


Figure 4-42: Root forces (N) for Dynamic Test case 4 (zoom in)

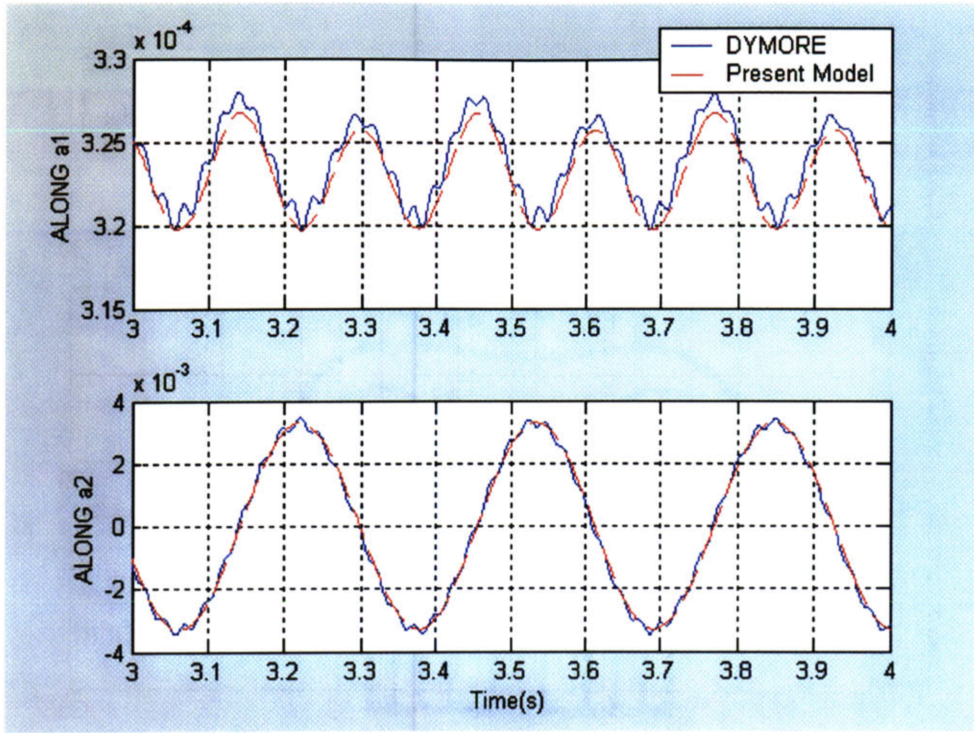


Figure 4-43: Tip displacements (m) for Dynamic Test case 4 (zoom in)

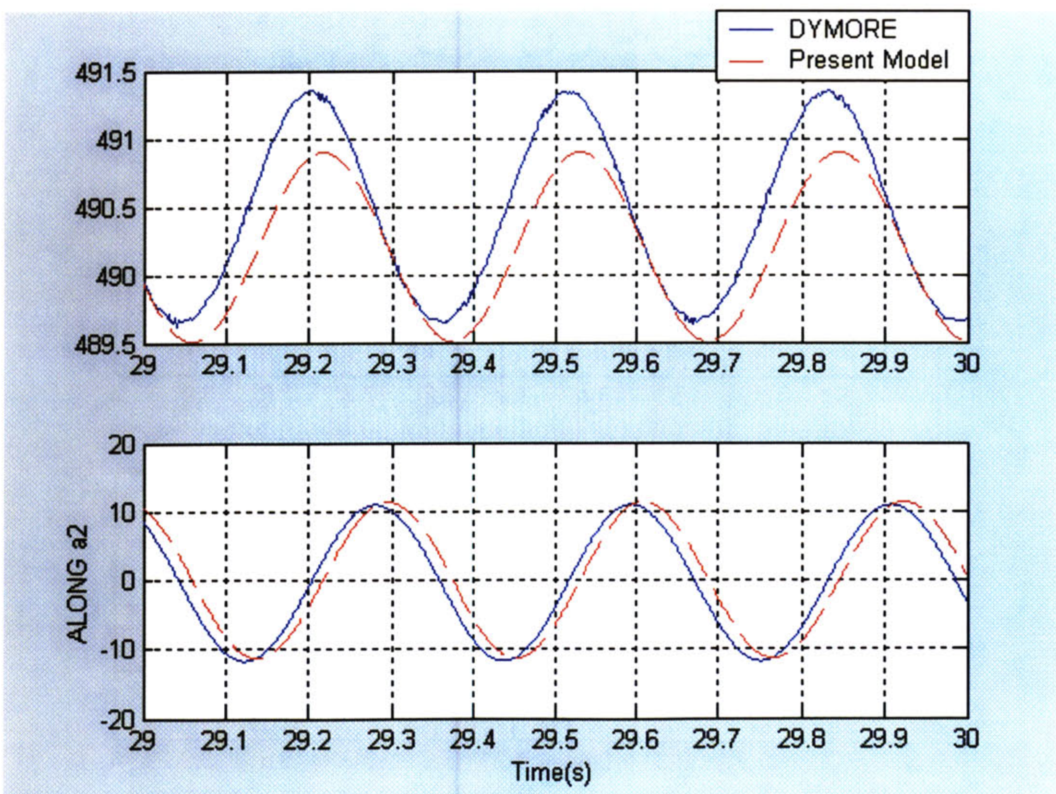


Figure 4-44: Root forces (N) for Dynamic Test case 4 (zoom in)

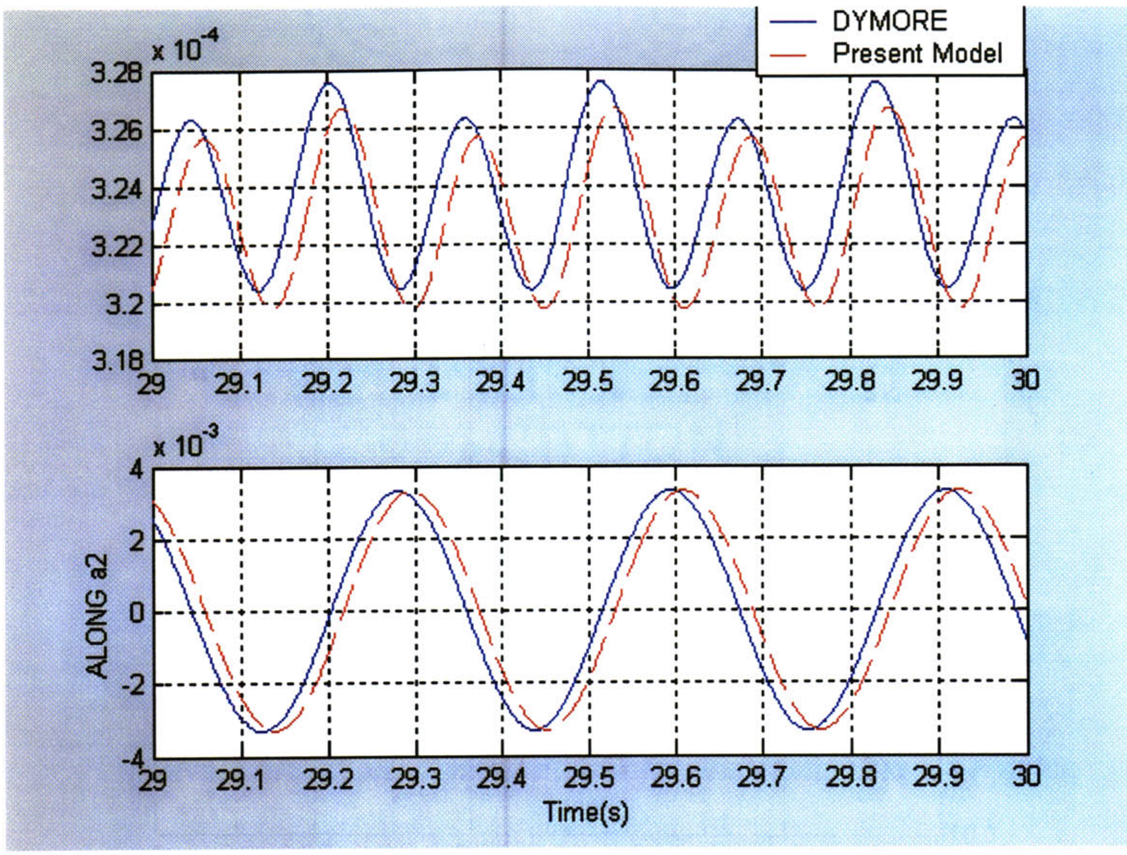


Figure 4-45: Tip displacements (m) for Dynamic Test case 4 (zoom in)

4.3.5 Test case 5

Test case 5 is a rotating beam clamped at the root and with a tip force along a_3 as shown in Fig. 4-46. The time step size used for the time integration in DYMORE and present model is $1.0*10^{-3}$sec. The rotating speed is 70 rad/s. The material properties of this beam are shown in Table 4.1. Figs. 4-47 to 4-50 present the comparisons of tip displacements, tip rotations, root forces and root moments for a tip force (*F*) of 50sin20t N. As one can see, the two results virtually overlap. The only exception is on the axial force component where there is a maximum difference of less than 2%. The source of this difference is not known at this moment.

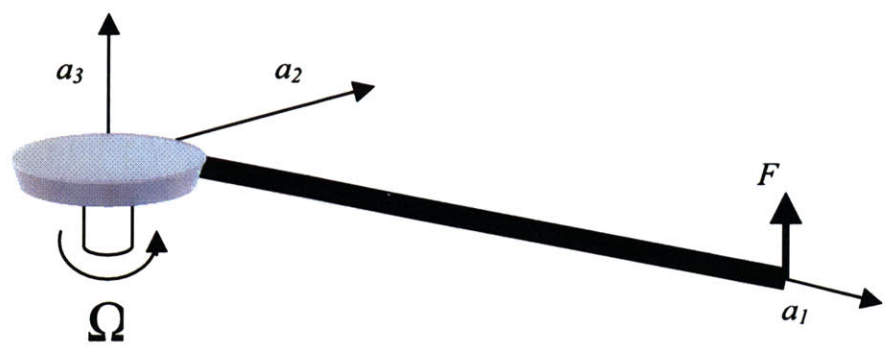


Figure 4-46: Beam model for Dynamic Test case 5

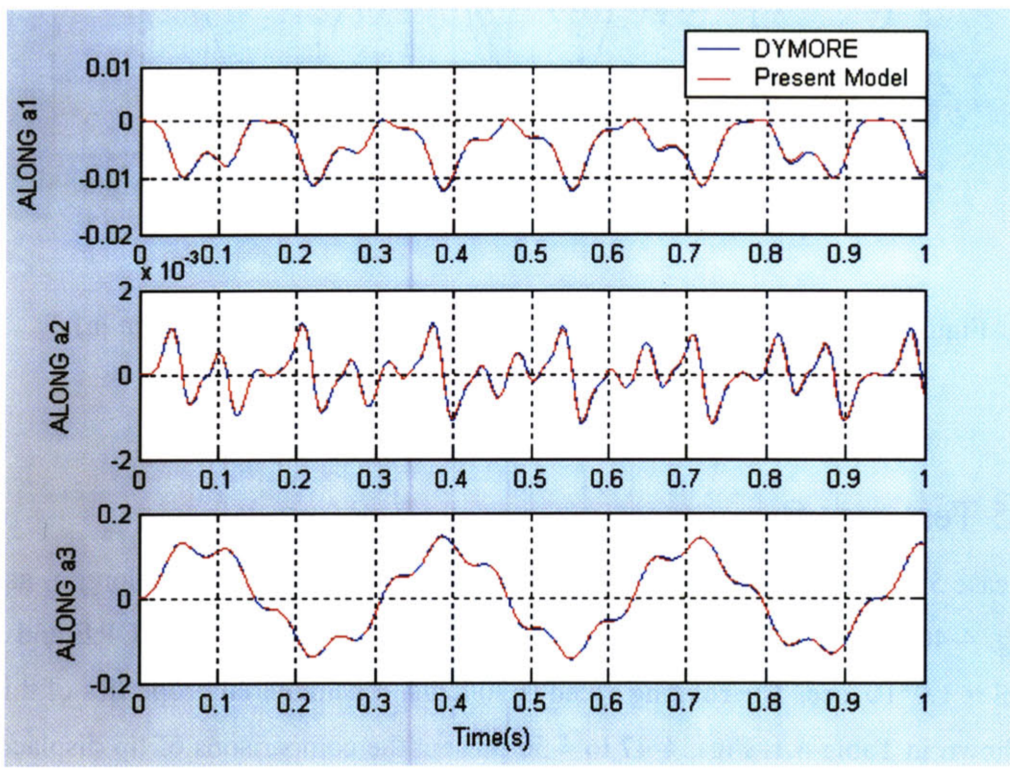


Figure 4-47: Tip displacements (m) for Dynamic Test case 5

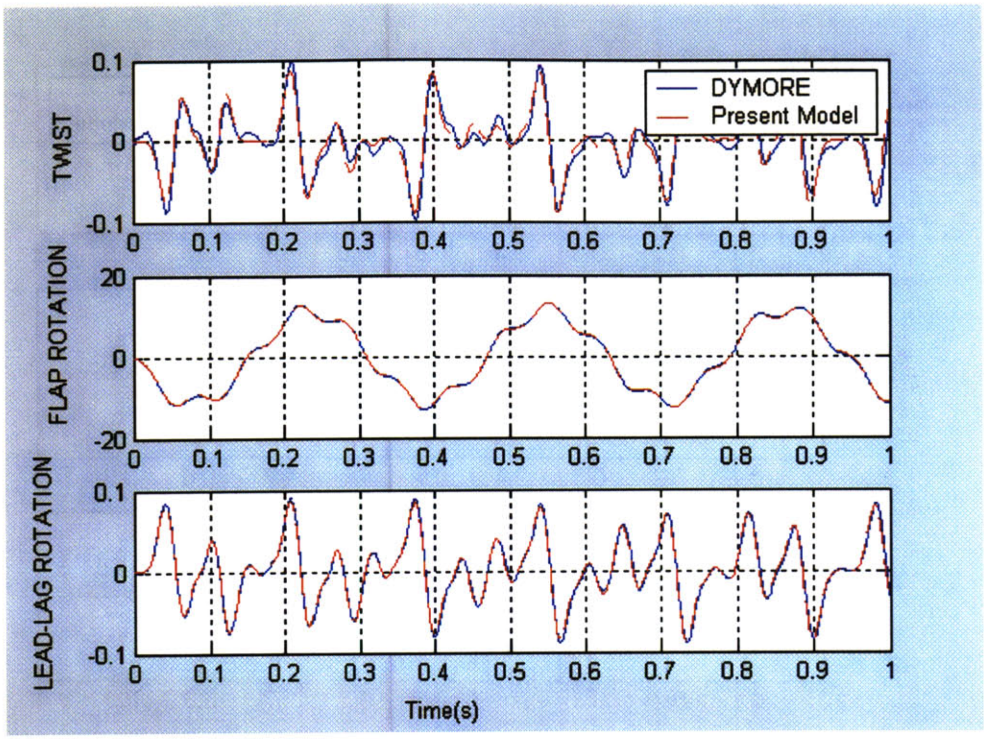


Figure 4-48: Tip rotations (degree) for Dynamic Test case 5

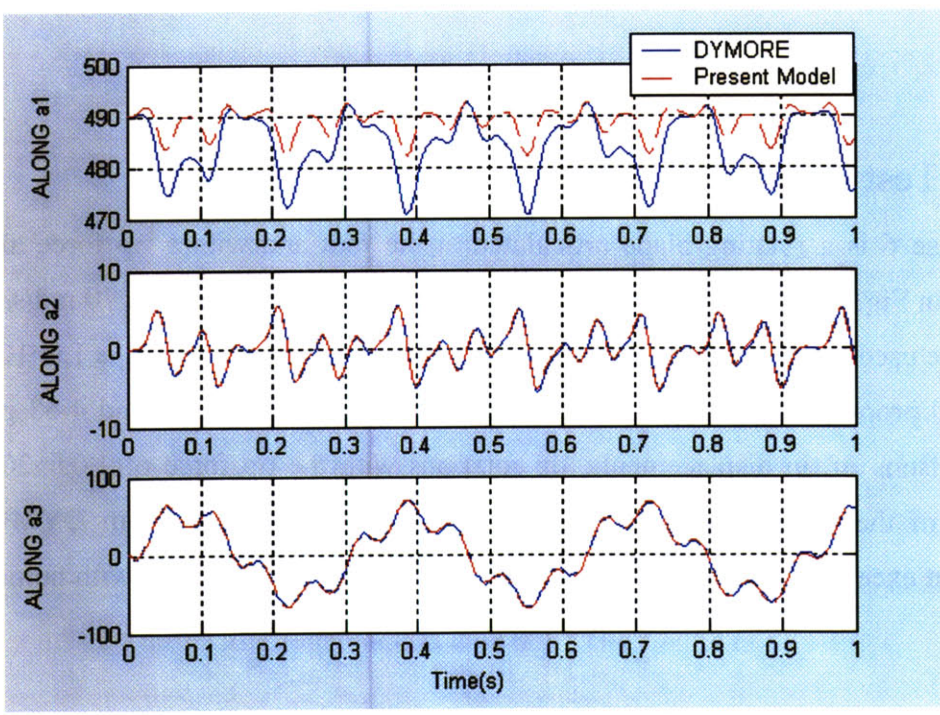


Figure 4-49: Root forces (N) for Dynamic Test case 5

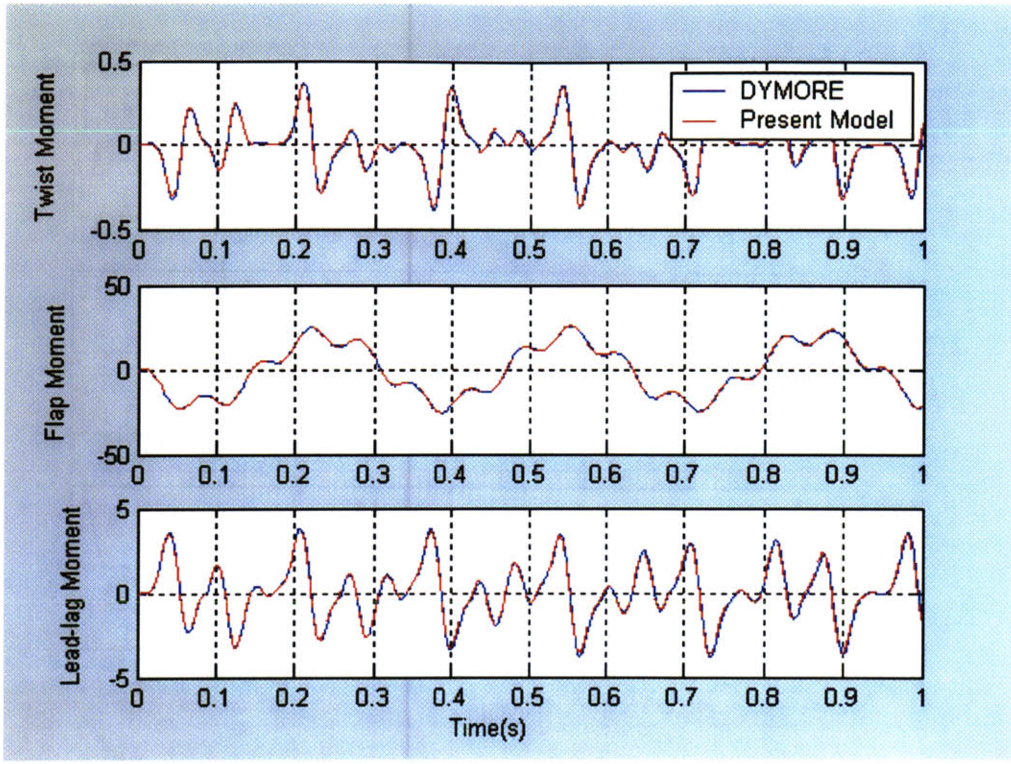


Figure 4-50: Root moments (Nm) for Dynamic Test case 5

4.3.6 Test case 6

Test case 6 is a rotating blade articulated at the root and with a tip force along a_2 as shown in Fig. 4-51. The root offset is 0.1 meter. The rotating speed is 70 rad/s. The time step size used for the time integration in DYMORE and present model is $1.0*10^{-3}$sec. The material properties of this blade are given in Table 4.1. Figs. 4-52 and 4-53 present the comparisons of tip displacements, tip rotations with the tip force of 1.0sin20t N. The results of the present model correlate very well with the ones from DYMORE. The apparent exception of noisy solutions from DYMORE are associated with numerical zero values.

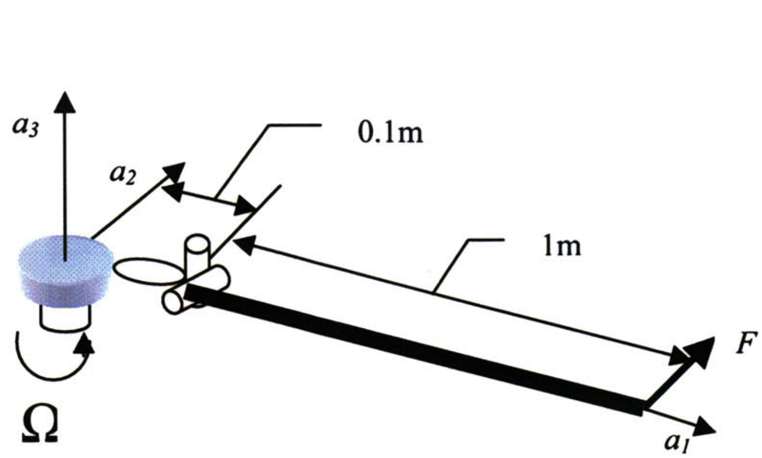


Figure 4-51: Beam model for Dynamic Test case 6

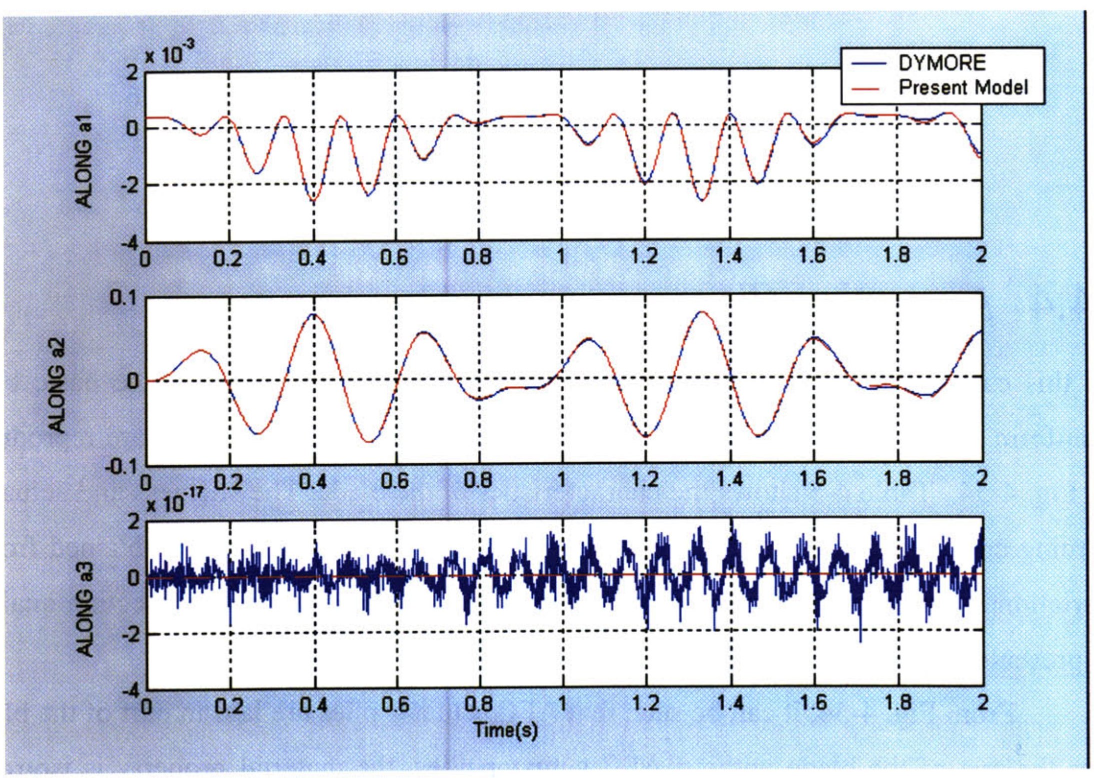


Figure 4-52: Tip displacements (m) for Dynamic Test case 6

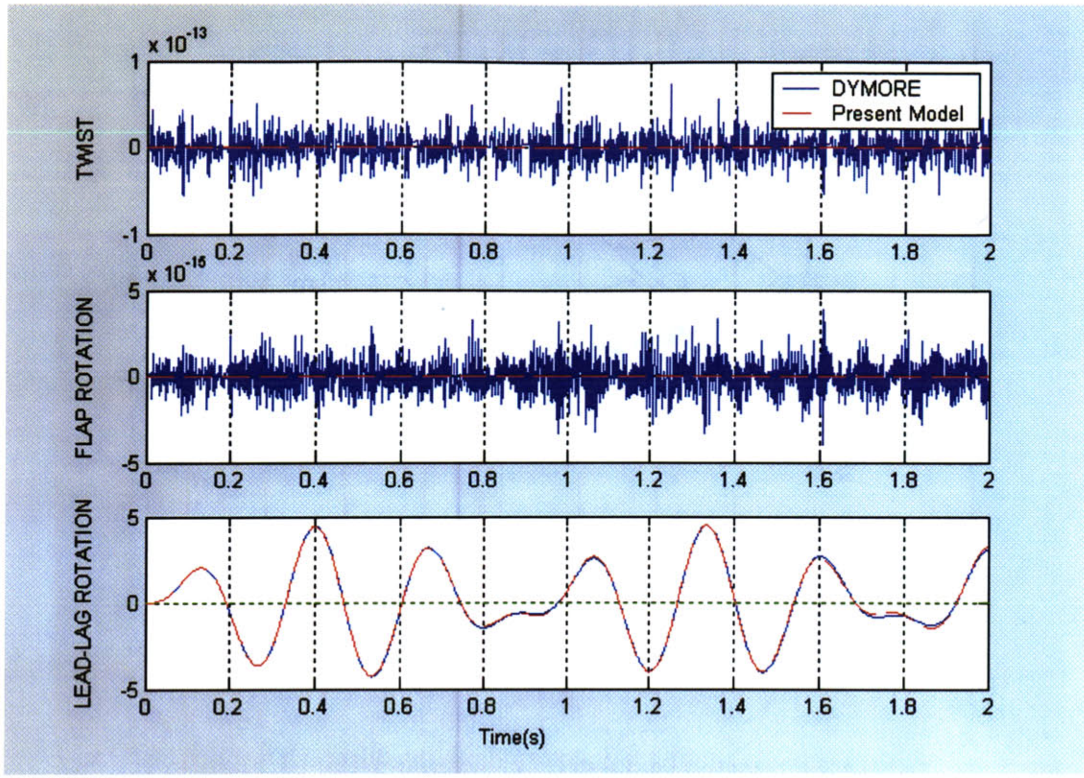
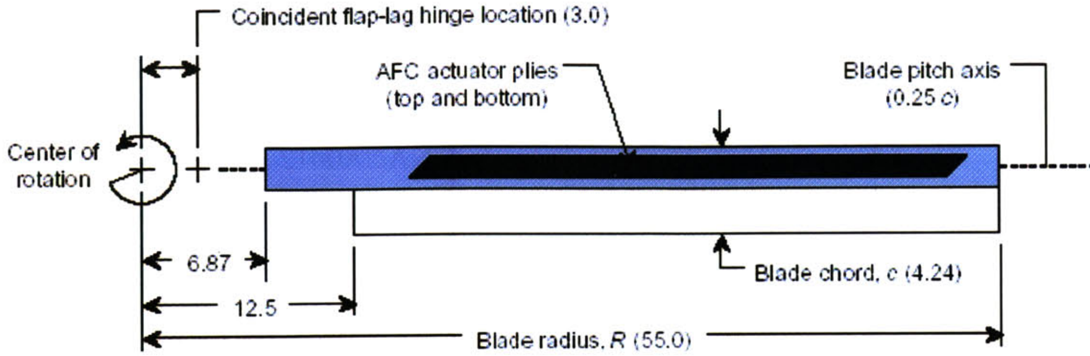


Figure 4-53: Tip rotations (degree) for Dynamic Test case 6

4.4 Actuation Test

In this case, ATR prototype blade investigated in Ref. [12] is used. The basic blade planform and cross section characteristics were reported in Ref. [12] and are reproduced in Fig. 4-54. Table 4-5 shows the characteristics of the blade. The stiffness and actuation forcing constants for an active anisotropic beam in its cross section are obtained from a variational-asymptotical formulation. The detailed information of cross-section analysis is presented in Ref. [13].

From Fig. 4-54, it can be seen that AFC actuator plies are laid in part of the blade. For the rest of the blade without AFC actuator plies, the material property is isotropic. Therefore, different mass and stiffness matrices are needed to define the active and passive regions of the blade. The corresponding mass and stiffness constants of the blade are given in Tables 4.6-4.9.



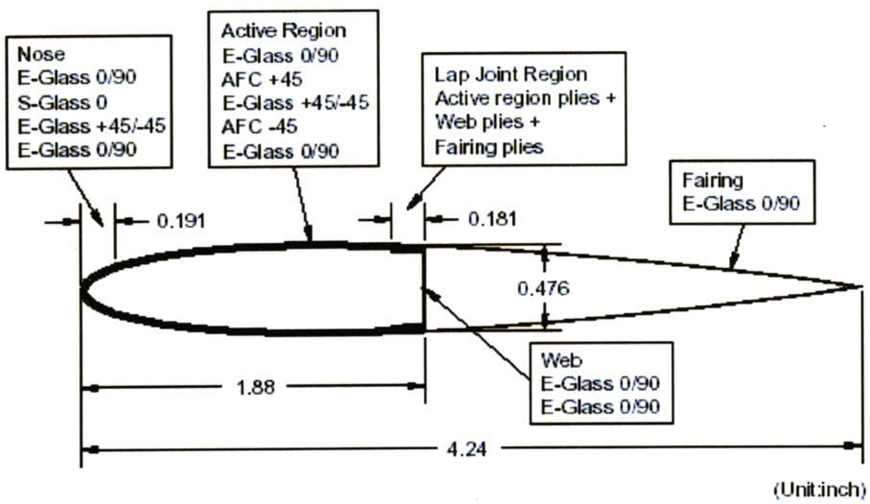


Figure 4-54: Planform and cross-section of the ATR prototype blade [12]

Table 4.5: Characteristics of the ATR prototype blade

Rotor type	Fully articulated
Blade chord (cm)	10.77
Blade radius (m)	1.397
Airfoil section	NACA0012
Hinge offset (cm)	7.62
Root cutout (cm)	31.75
Pitch axis	25% chord
Elastic axis	25% chord
Center of gravity	25% chord

Table 4.6: Non-zero inertia constants for the ATR prototype blade (active regions)

I_{11} (kgm)	0.6960
I_{22} (kgm)	0.6960
I_{33} (kgm)	0.6960
I_{44} (kgm)	$0.3307*10^{-3}$
I_{55} (kgm)	$0.6599*10^{-5}$
I_{66} (kgm)	$0.3241*10^{-3}$

Table 4.7: Non-zero inertia constants for the ATR prototype blade (passive regions)

I_{11} (kgm)	3.927
I_{22} (kgm)	3.927
I_{33} (kgm)	3.927
I_{44} (kgm)	$5.2631*10^{-4}$
I_{55} (kgm)	$2.6626*10^{-5}$
I_{66} (kgm)	$4.9969*10^{-4}$

Table 4.8: Non-zero stiffness constants for the ATR prototype blade (active regions)

K_{11} (N)	$0.1637*10^7$
K_{22} (N)	$0.1000*10^{21}$
K_{33} (N)	$0.1000*10^{21}$
K_{44} (Nm^2)	$0.3622*10^2$
K_{55} (Nm^2)	$0.4023*10^2$
K_{66} (Nm^2)	$0.1094*10^4$

Table 4.9: Non-zero stiffness constants for the ATR prototype blade (passive regions)

K_{11} (N)	$9.78476*10^6$
K_{22} (N)	$3.7634*10^6$
K_{33} (N)	$3.7634*10^6$
K_{44} (Nm^2)	$5.04357*10^2$
K_{55} (Nm^2)	$6.6339*10^1$
K_{66} (Nm^2)	$1.24499*10^3$

In order to get the frequency response of the ATR prototype blade on the bench (cantilever boundary conditions), a sine-sweep signal of actuation ranging from 0 to100 Hz is applied as shown in Fig. 4-55. The time step size used for the time integration in present model is $2.0*10^{-4}$sec. The corresponding tip twist response in time from the current model is shown in Fig. 4-56. The transfer function is estimated by dividing the FFT of the twist response by the FFT of the input actuation signal. In Fig. 4-57, the peak-to-peak tip twist response of the blade is compared with the experimental data from [12]. Overall, the correlation is very good. The resonant peak is captured around 78 Hz. It can be seen however that the response of the blade using the present model has more oscillation than that from the experiment. This may be associated with the fact that the real structure has certain level of internal damping which helps damping the motion. The current model has no structural damping.

The CPU time involved in obtaining the results presented in Figs. 4-56 is on the order of 2h in an Intel Pentium III 800MHz machine.

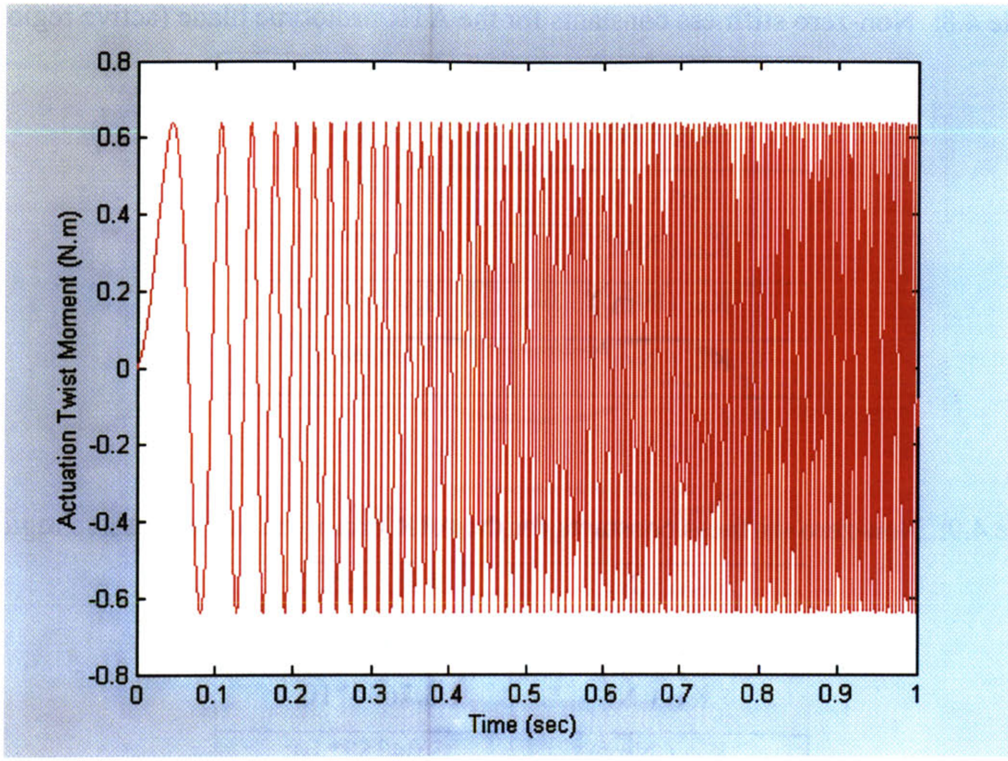


Figure 4-55: Active input of twist moment

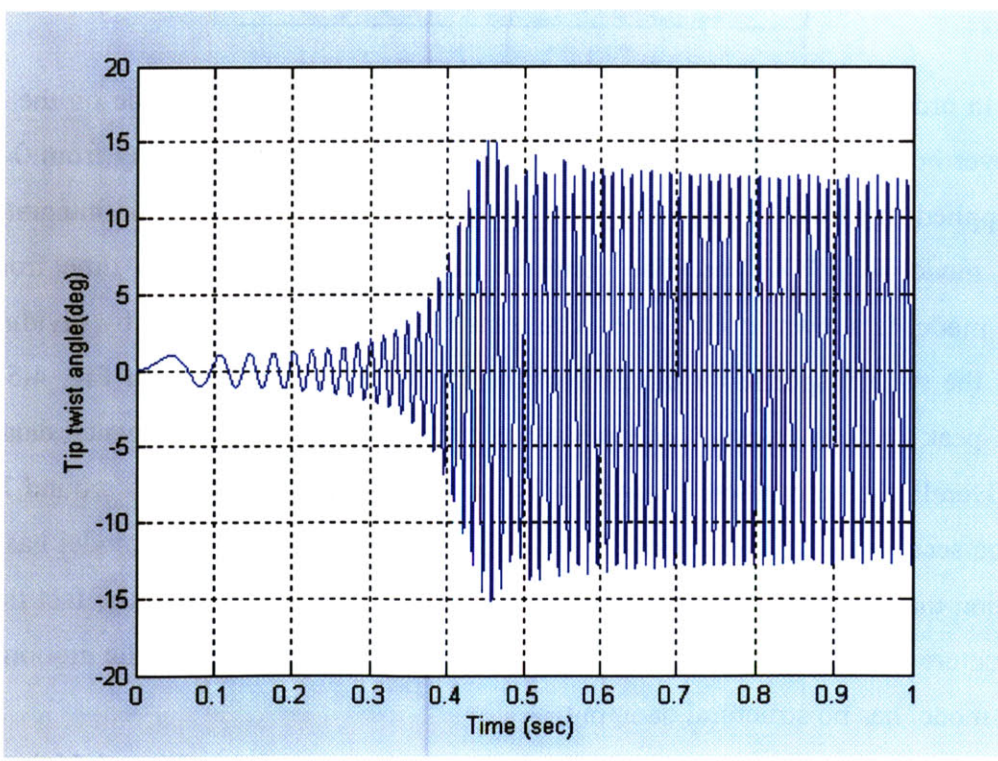


Figure 4-56: Time history of tip twist angle of the ATR prototype blade on the bench by a sine-sweep actuation signal

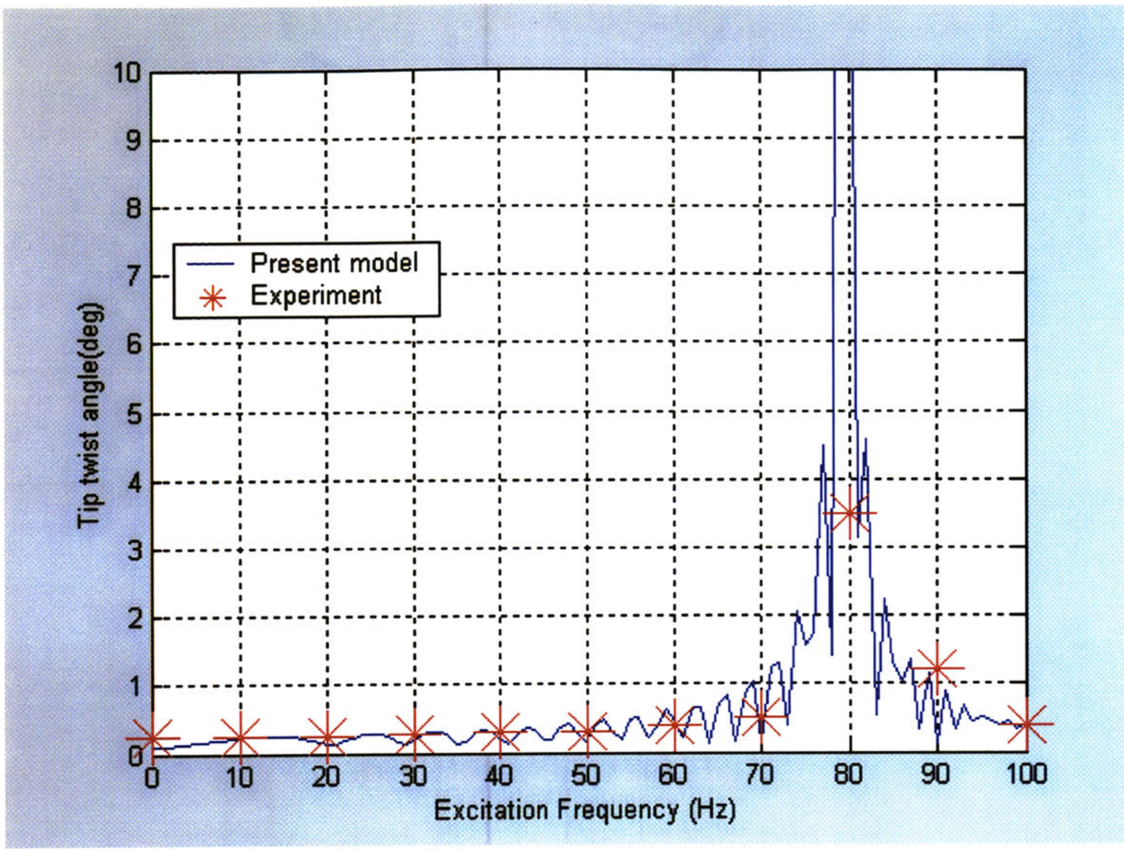


Figure 4-57: Tip twist response of the ATR prototype blade on the bench

4.5 Performance Benchmark

Table 4.10 presents a comparison of CPU time involved in obtaining the solution of some of the test cases. All times were obtained in an Intel Pentium III 800MHz machine. It can be seen that DYMORE runs five to ten times faster than the present implementation and further work should be pursuit in the future to improve the performance of the present method to comparable levels.

Table 4.10: Comparison of CPU time

	DYMORE	Present Model
Test case in Section 4.2.1.4	1.0 s	6.7 s
Test case in Section 4.3.1	96 s	1195 s
Test case in Section 4.3.3	100 s	501 s

Chapter 5

Numerical Validation of the Aeroelastic Modeling

In order to validate the aeroelastic modeling as shown in Chapter 3, a fixed wing case and the ATR prototype blade have been used to test and results are compared with other related analytical methods. The steady state of fixed wing under different flight speeds has been obtained. The time and frequency response of the ATR prototype blade on hover condition has also been tested. In these cases, the aerodynamic code GENUVP is integrated with the structural code using the interfaces. The structural-aero coupled response obtained from the coupled code is tested. The actuation test of the ATR prototype blade on hover condition is also tested. At last, a sub-iteration study is conducted.

5.1 Fixed Wing

In order to validate the aeroelastic modeling, a fixed wing case is run first. The material properties of the fixed wing used in this test is shown in Table 5.1. The fixed wing is 1 meter long, the semichord is 0.05385 meter, and the root angle of attack is 5°. The air density is 1.049kg/m^3. The flight speed tested is ranged from 0 to 40 m/s.

Table 5.1: Material properties of the test fixed wing

Mass per unit span (kgm^{-1})	0.2363
I_{xx} (kgm)	$0.1117*10^{-3}$
I_{yy} (kgm)	0
I_{zz} (kgm)	$0.1052*10^{-3}$
K_{11} (N)	$1.6284*10^{6}$
K_{22} (N)	$1.0*10^{20}$
K_{33} (N)	$1.0*10^{20}$
K_{44} (Nm^2)	37.31
K_{55} (Nm^2)	39.38
K_{66} (Nm^2)	1037.2
K_{16} (Nm)	750.53

From the aerodynamics theory [23], the lift force per length can be obtained approximately using the relation as follow:

$$F = 2\pi\rho U^2 b\alpha \tag{5.1}$$

where α is the collective pitch angle or angle of attack, b is the semichord length and ρ is the air density.

From the linear elastic theory [22], the tip displacement of the beam loaded with a distributed force as Fig. 5-1 should equal to

$$\Delta = FL^4 / 8EI \tag{5.2}$$

where Δ is the tip displacement, EI=K_{55}=39.80 N.m^2 (Table 5-1), and L is the wing span.

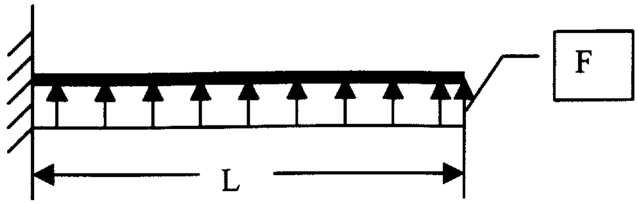


Figure 5-1: Evenly distributed load on the fixed wing

Therefore, one can use Eq. 5.2 to estimate the linear tip displacement of the fixed wing. Results from the present code are also compared with the results obtained from the nonlinear model of Ref. [19]. It can be seen from Fig. 5-2, as expected, that the linear analytical result of Eq. 5.2 presents a quadratic dependence on the uniform flow speed. For the highest uniform flow speed considered, an approximately 0.15-m wing tip deflection is obtained. This corresponds to a relative tip deflection of 15% span which starts exciting geometrically nonlinear effects. Therefore, a discrepancy between the linear analytical result and the other two results in Fig. 5-2 at high uniform flow speeds is expected. However, there is a discrepancy between the two nonlinear solutions. Fig. 5-3 presents the time responses for two of the conditions depicted in Fig. 5-2. Results are obtained by increasing the speed from rest to its maximum speed (in this example, either 30 or 40 m/s). The ramping-up ratios used between the two codes were different, and that explains the different times required for reaching steady state. As one can see, very good agreement between the two solvers on the steady state results at 30 m/s. However, error on the order of 10% is present between the two solutions for 40 m/s. These are in accordance to the results shown in Fig. 5-2. Even though the aerodynamics comes from different sources, the steady aerodynamics should be the same. Therefore, possible source of error resides in the interface between the aerodynamics and structures for the present model. Further investigation is needed to pinpoint the cause.

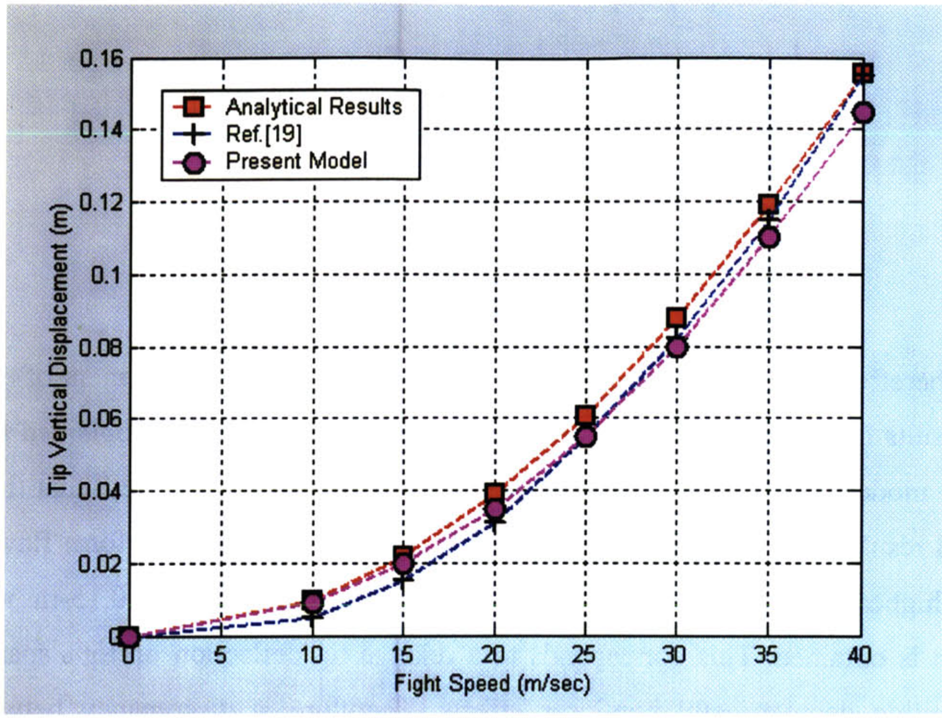


Figure 5-2: Comparison of the fixed wing tip vertical displacement under different flight speeds

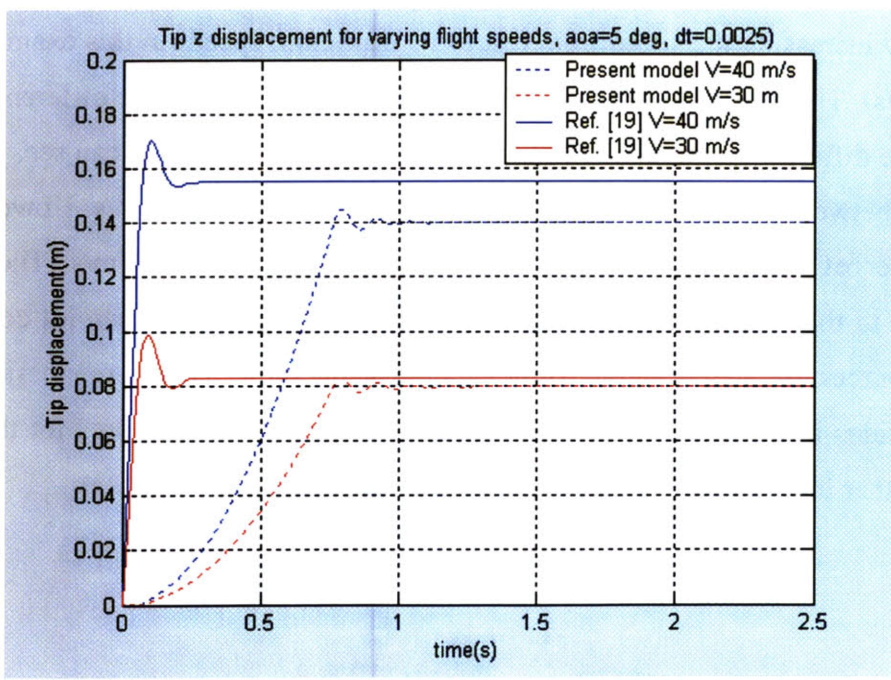


Figure 5-3: Time responses of wing tip vertical displacement under different flight speeds

5.2 ATR Prototype Blade for the Hover Condition

In this test, the tip displacement and rotation of the ATR prototype blade are obtained for the hover condition. The blade is the same as the one described in section 4.4. The mass and stiffness constants of the blade are given in Tables 4.6-4.9.

The rotating speed of this rotor is 688 rpm and the collective pitch angle is 8°. The medium density is 2.432kg/m^3. There is –10° built-in pretwist from the root to the blade tip. The root offset of the blade is 0.0762 m. Only the lead-lag damper was used and the damping coefficient is 10.0 Nm/(rad/s).

As mentioned in Chapter 3, the aeroelastic analysis of the rotating blades is separated into two steps: one is the steady analysis of the rotating blade in vacuum; the other is the dynamic analysis of the blade in air. The deformations, internal forces and moments, and momenta of a rotating blade in vacuum are obtained by the steady state analysis. They are used in the dynamic analysis (with aerodynamic loads) as the initial rotating condition input. In the dynamic analysis, the blades are rotating at their full speed at the first structural time step. However, in the aerodynamic module, the rotating speed should be increased from zero to the full speed generally in order to avoid suddenly applied aerodynamic forces, which result in large numerical blade oscillations. In this case, the time for the rotating speed to reach full speed is 1.0 second.

Figs. 5-4 and 5-5 present the tip displacements and rotations of the ATR blade in hover. They show that the blade flaps up and lag back with the aerodynamic forces and after the full rotating speed is reached, it oscillates a little especially in the twist rotation.

As a first investigation on those oscillations, the fan plot of the ATR prototype blade from Ref. [6] is used and reproduced in Fig. 5-6. It can be seen that the rigid lag frequency of the ATR prototype blade at the rotating speed of 688 rpm is approximately 4 Hz, the rigid flap frequency is 12 Hz, and first elastic flap frequency is 30 Hz. Figs. 5-7 to 5-9 present the FFT results of the three tip rotations. The two peaks in the frequency response of the tip twist as shown in Fig. 5-7 is at 4 Hz and 27 Hz, which are close to the rigid lag and 1st elastic flap frequencies, respectively. The peaks in the frequency response of the tip flap are at 4 Hz, 12 Hz and 27.5 Hz, which are around the rigid lag, rigid flap, and 1st elastic flap frequencies of the ATR blade, respectively. The peak in the frequency response of the tip lead-lag rotation is at 4 Hz, which is associated with the rigid lag frequency.

The most probable cause of these oscillations is a drift on the phase of the loads and deformations when they are passed back and forth between the structures and aerodynamics modules. As shown in the block diagram of the present aeroelastic code (Fig. 3-4), there are no sub-iterations between the aerodynamic and structural components at each time step. For a tightly-coupled aeroelastic analysis, the sub-iterations at each time step is required to allow the codes to converge. This must be addressed in the future.

The CPU time involved in obtaining the results presented in Figs. 5-4 and 5-5 is 1.8 hours in an Intel Pentium III 800MHz machine. The fraction of CPU time used by the aerodynamic code and structural code was not recorded.

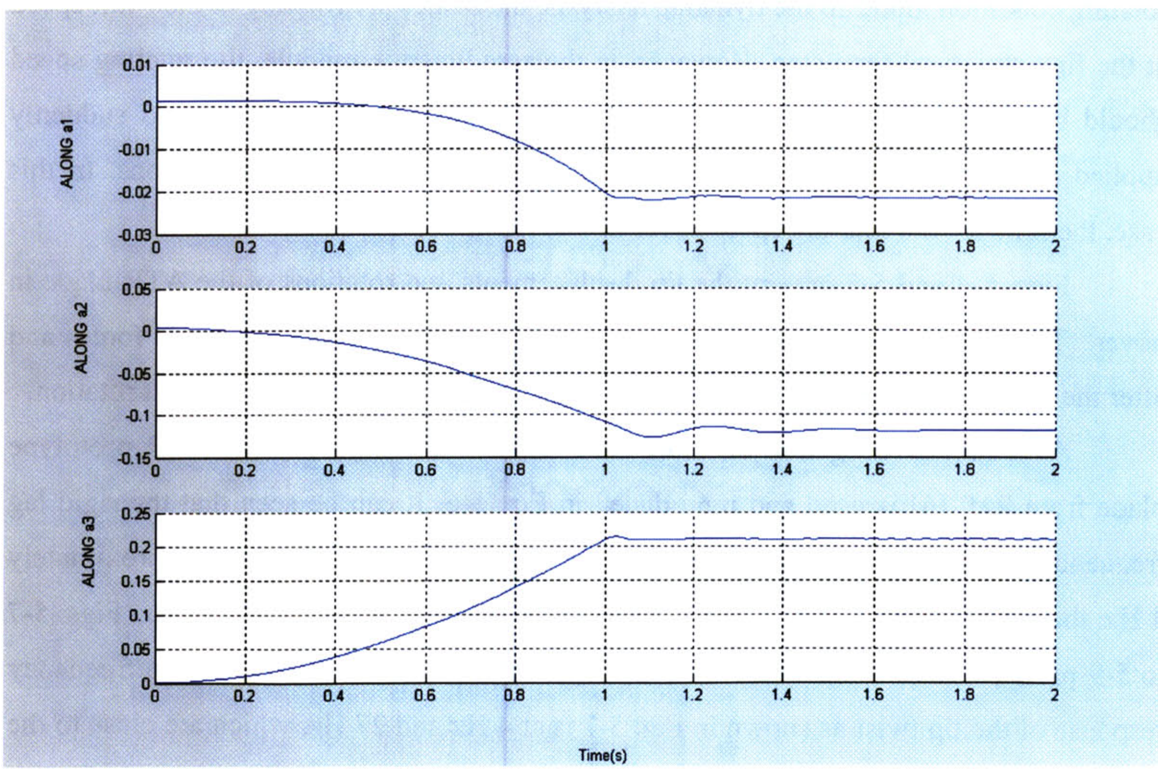


Figure 5-4: Tip displacements of the ATR prototype blade in hover (Δt=0.001s)

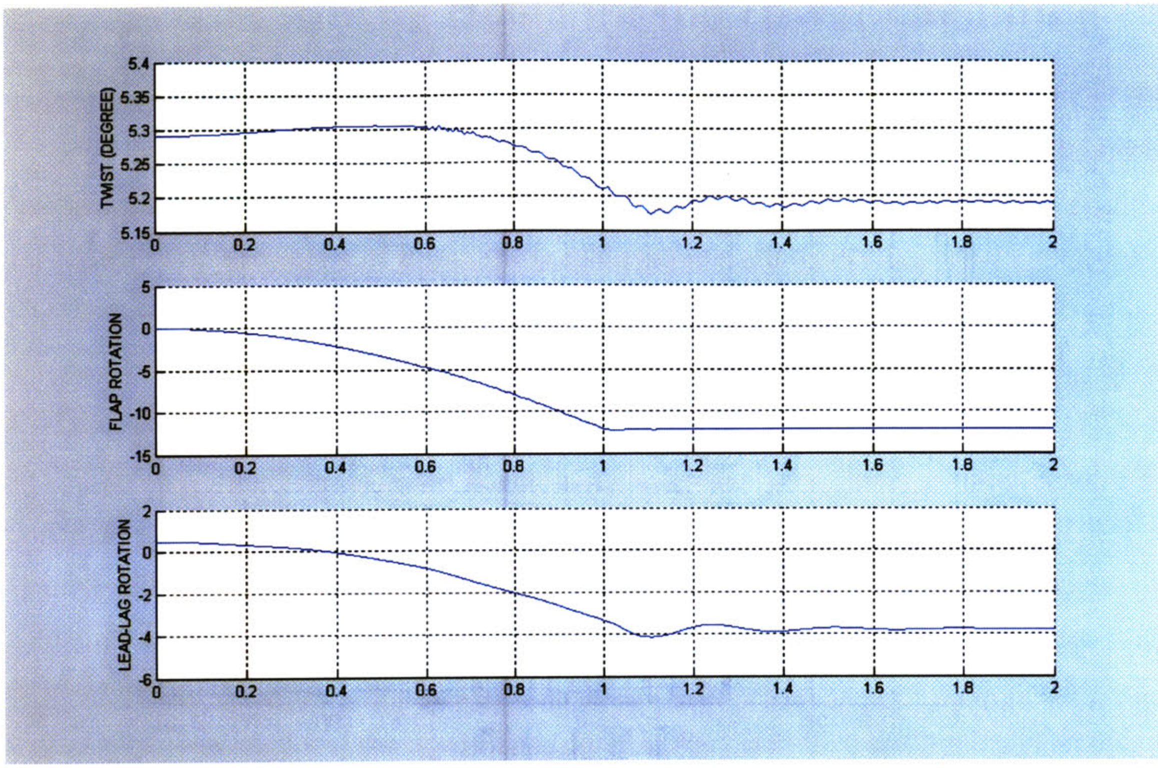


Figure 5-5: Tip rotations of the ATR prototype blade in hover (Δt=0.001s)

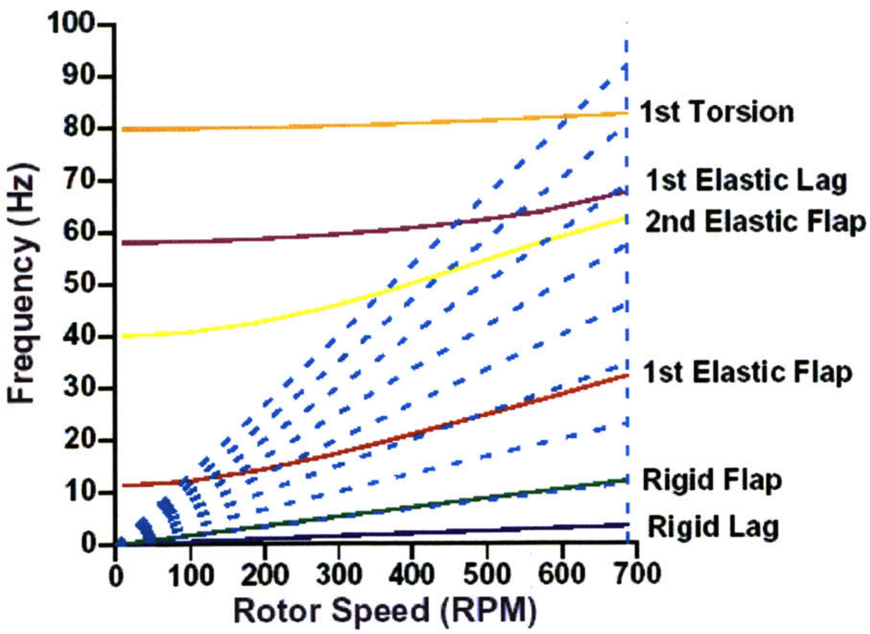


Figure 5-6: Fan plot of the ATR prototype blade [6]

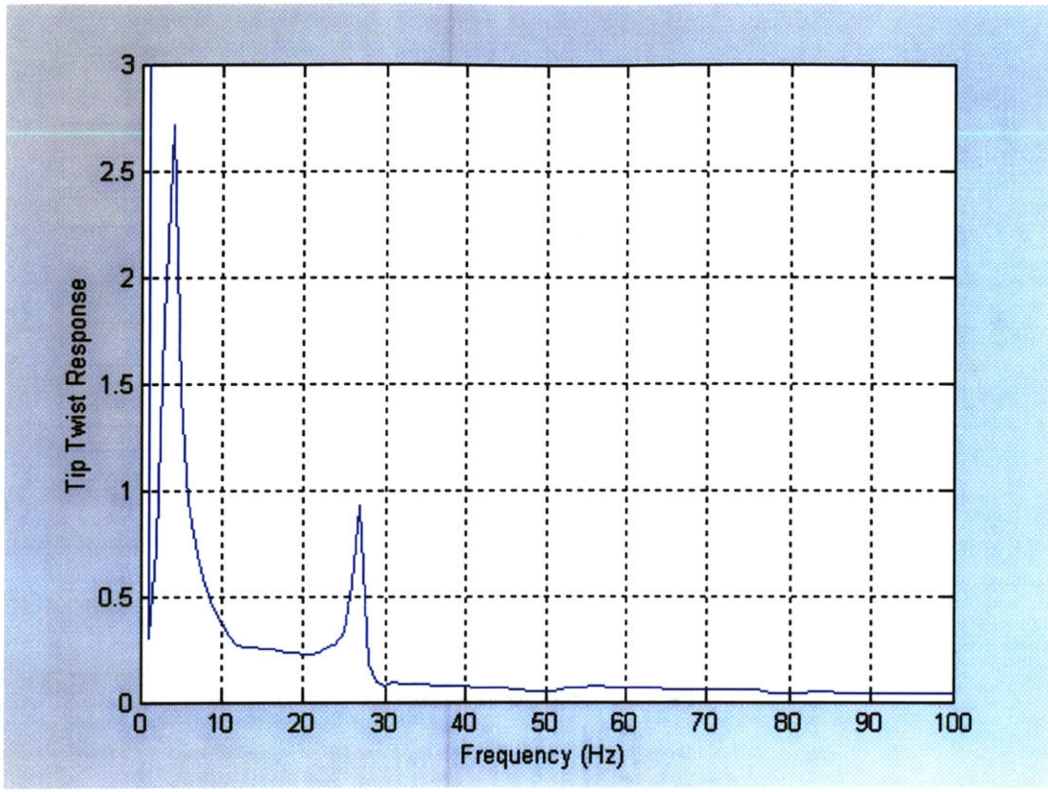


Figure 5-7: Frequency response of the tip twist rotation of the ATR prototype blade in hover

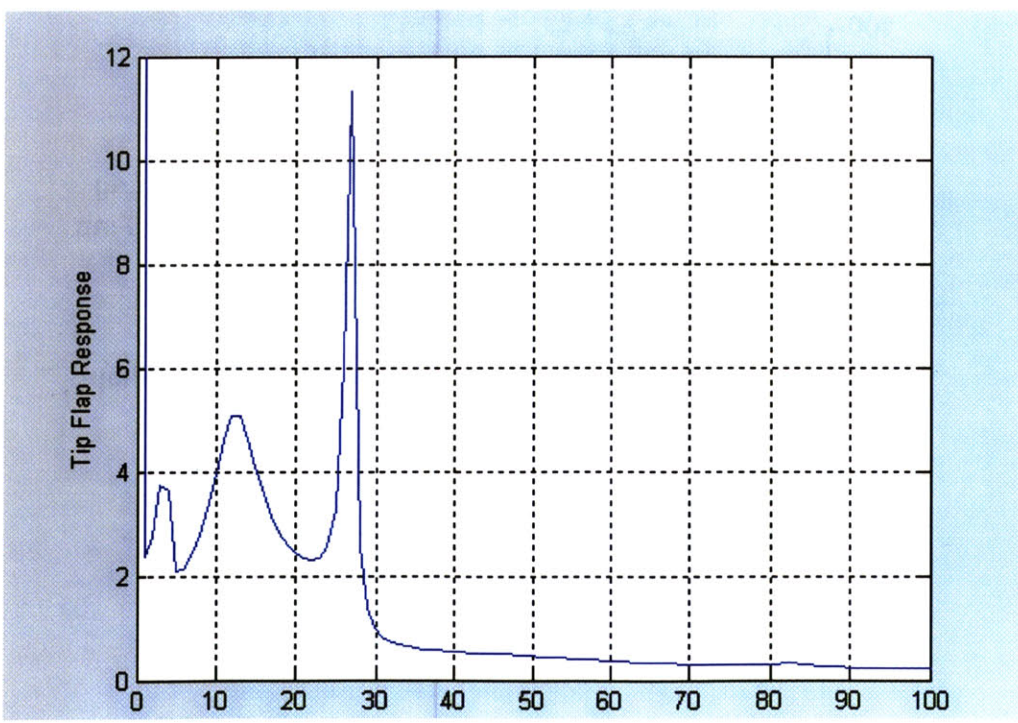


Figure 5-8: Frequency response of the tip flap rotation of the ATR prototype blade in hover

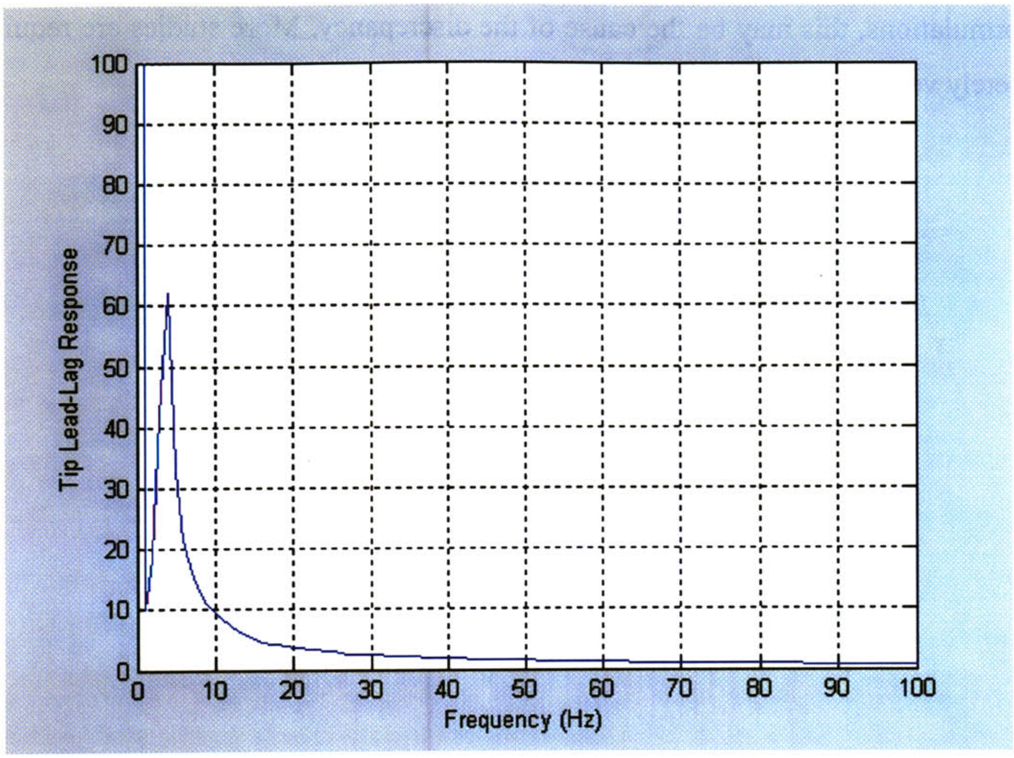


Figure 5-9: Frequency response of the tip lead-lag rotation of the ATR prototype blade in hover

5.3 Actuation Test of ATR Prototype Blade for the Hover Condition

In this case, the same ATR prototype blade was tested for active twist actuation using the present aeroelastic formulation. Other parameters are the same as the articulated case showed above. The actuation signal is a sine-sweep signal ranging from 0 to 100 Hz as shown in Fig.5-10. This actuation twist moment is applied after the full rotating speed is reached. The time step size used for the time integration in this case is $1.0*10^{-3}$sec. The corresponding tip twist response in trim is shown in Fig. 5-11. The transfer function estimatied using the same way as in Section 4.4 is shown in Fig. 5-12. The resonant peak is captured around 76 Hz. Compared with Fig. 4-10 in Ref. [12], which tested the same blade in the same condition, the resonant peak is higher and the resonant frequency is also higher than those in Ref. [12]. Since the aerodynamic different is different from the two formulations, this may be the cause of the discrepancy. More studies are required to completely verify this hypothesis.

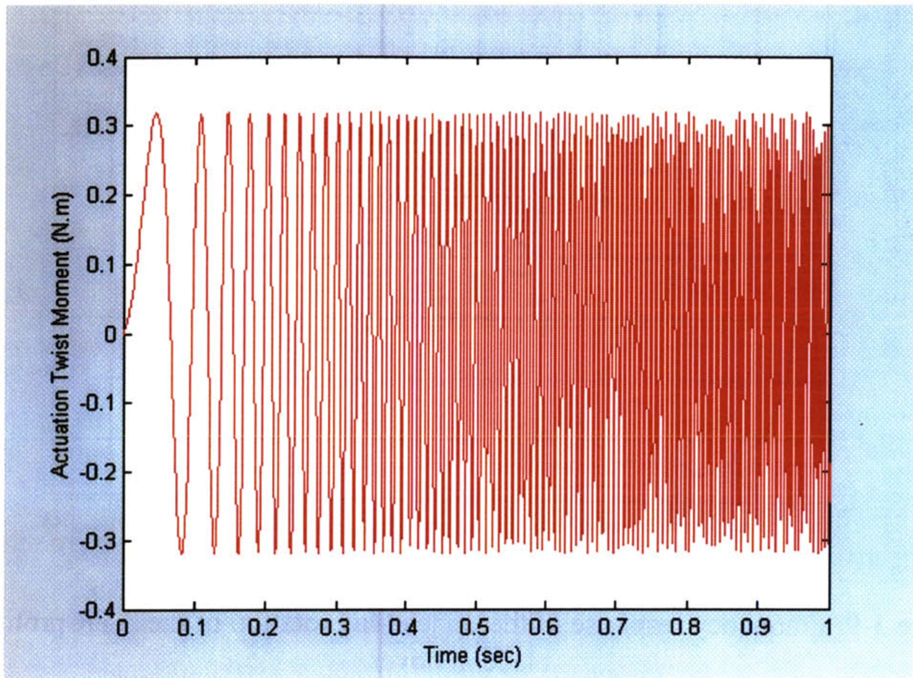


Figure 5-10: Active input of twist moment

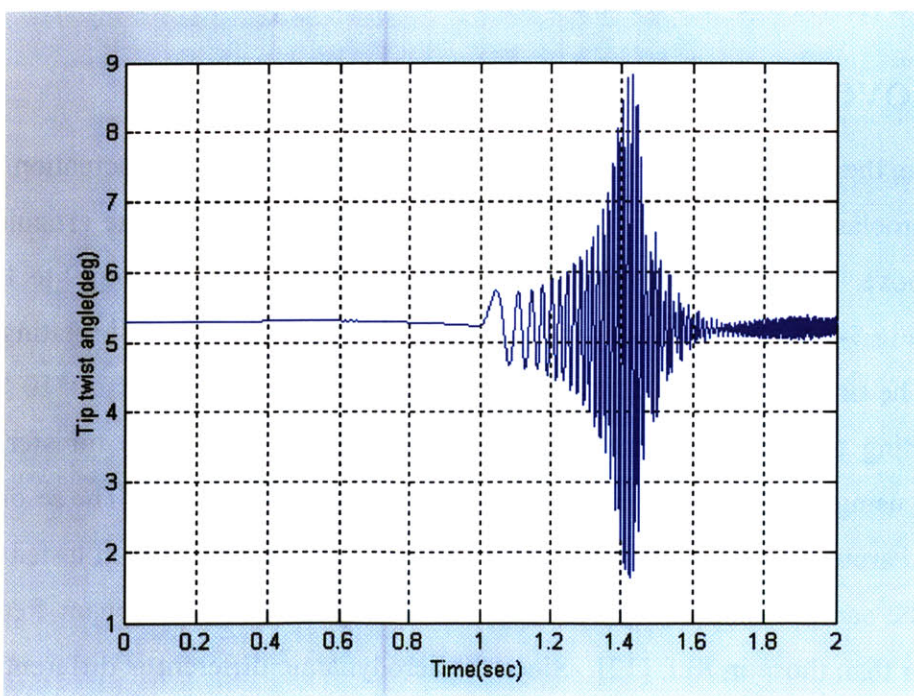


Figure 5-11: Time history of tip twist angle in hover by a sine sweep actuation after 1 sec

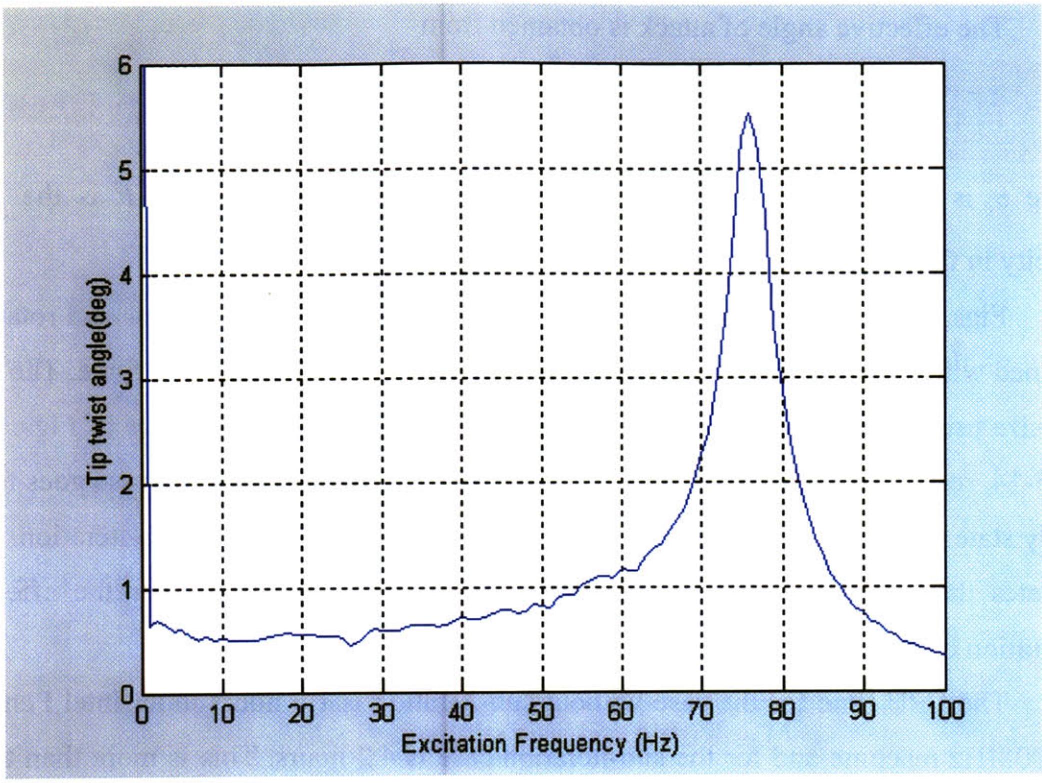


Figure 5-12: Tip twist amplitude response of the ATR prototype blade in hover

5.4 Sub-iteration Study

As mentioned before, the sub-iterations at each time step is usually required for a tightly-coupled aeroelastic analysis. In order to get a sense of how important this sub-iteration is, a simple aerodynamic model is used (instead of GENUVP) and integrated with the present structure model for a sub-iteration study.

For comparison, the same ATR prototype blade was tested. All parameters are the same as in Section 5.2. The lift force is obtained using the following relation:

$$F = \rho U^2 b 2\pi \alpha_{effective} \tag{5.3}$$

where b is the semichord length, ρ is the air density, $\alpha_{effective}$ is the effective angle of attack, U is the linear velocity defined as:

$$U = \omega * r \tag{5.4}$$

where ω is the rotating speed and r is the radius in the middle of each element.

The effective angle of attack is obtained from

$$\alpha_{effective} = \theta_0 + \theta_1 - \frac{\dot{h}}{U} \tag{5.5}$$

where θ_0 is the elastic twist, θ_1 is the pretwist of each element, and $\dot{h}$ is the local velocity in flapping.

Figs. 5-13 and 5-14 present the comparison of the tip displacements and rotations obtained with and without sub-iteration using the simply aerodynamic lift force. The time step size used in both cases is $1.0*10^{-3}$. Figs. 5-15 and 5-16 are close-ups of Figs. 5-13 and 5-14, respectively. It can be seen from the plots that the sub-iteration case goes to the steady state much faster with smaller oscillation than the case without sub-iteration. This indicates that sub-iteration should be investigated in further details for the effective integration of the present structure model and GENUVP.

The CPU time for the case without sub-iteration is 1.6 hours in an Intel Pentium III 800MHz machine and for the sub-iteration case is 4.2 hours. This is more than twice longer than the case without sub-iteration, since three to four sub-iterations were required to get convergence at each time step in the sub-iteration case. Since the number of required sub-iterations is aerodynamic formulation dependent, it is expected that the GENUVP will take even longer.

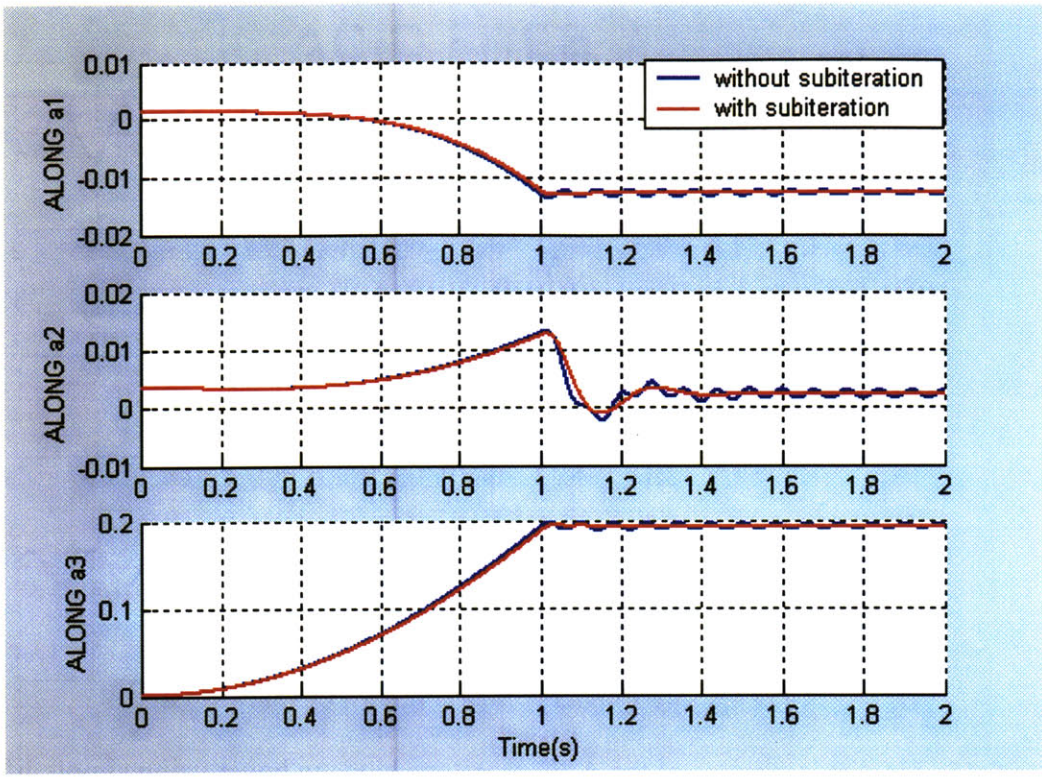


Figure 5-13: Tip displacements of the ATR prototype blade in hover using simple lift force

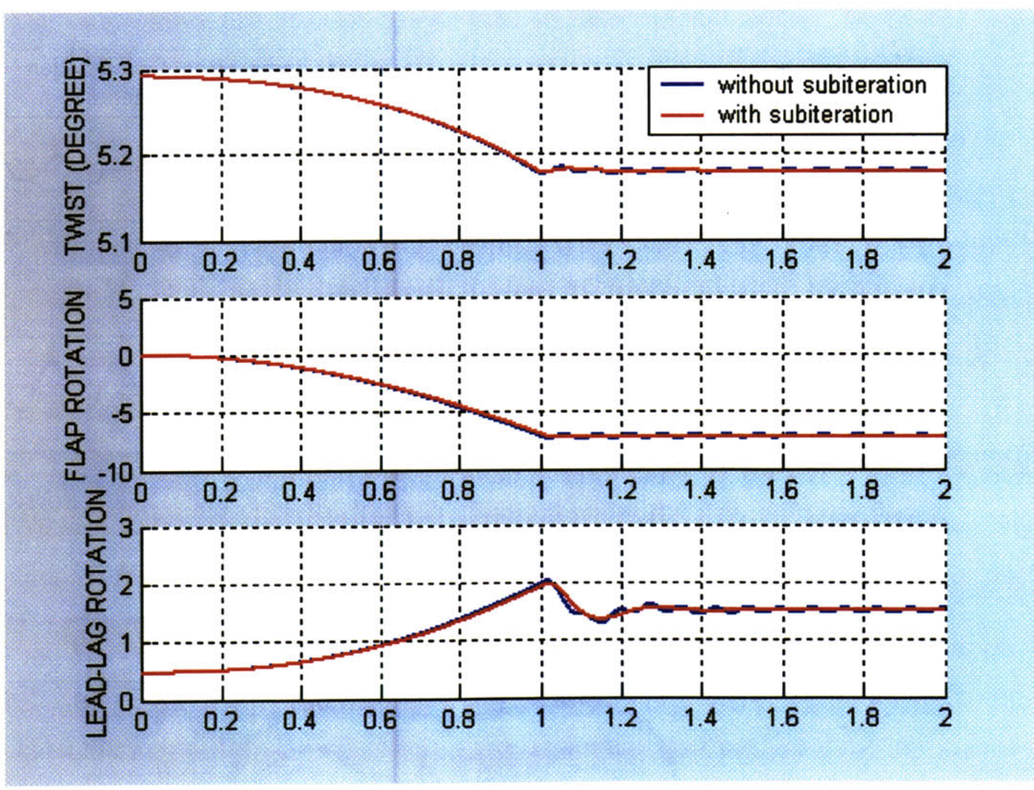


Figure 5-14: Tip rotations of the ATR prototype blade in hover using simple lift force

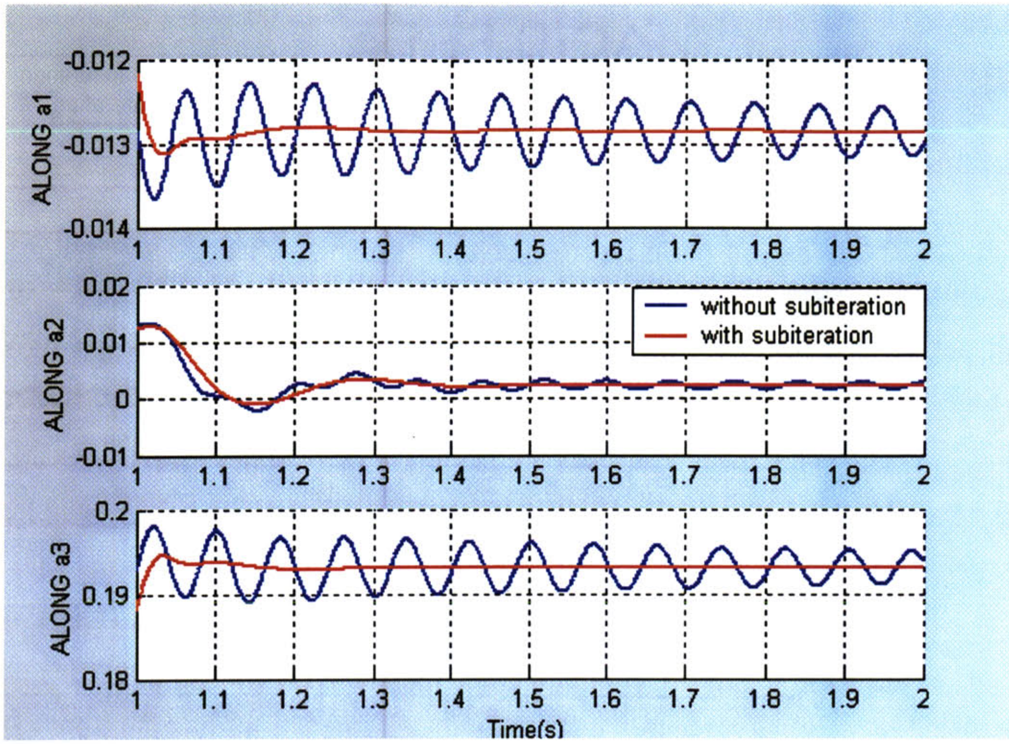


Figure 5-15: Tip displacements of the ATR prototype blade in hover using simple lift force (zoom in)

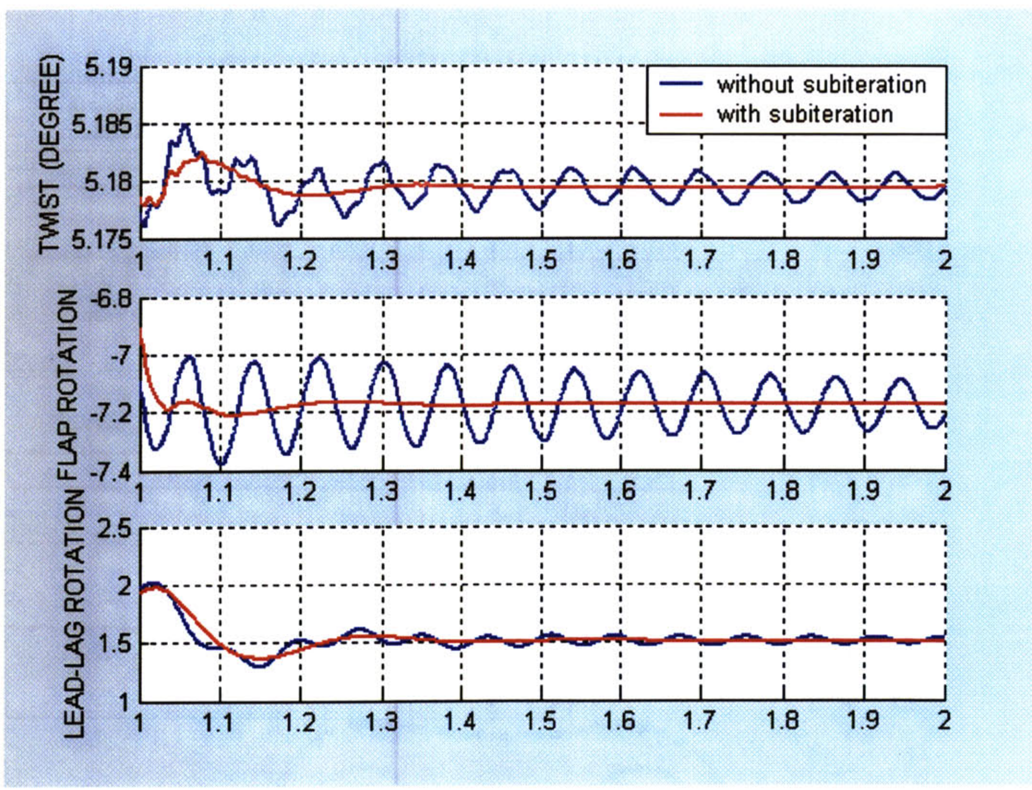


Figure 5-16: Tip rotations of the ATR prototype blade in hover using simple lift force (zoom in)

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

This thesis presented a time-domain structural simulation of a rotor system to be used in a tightly-coupled computational aeroelastic solver.

On the structural side, an asymptotical analysis takes the electromechanical three-dimensional problem and reduces it to a set of two analyses: a linear analysis over the cross section and a nonlinear analysis of the resulting beam reference line. The nonlinear 1-D global analysis considering small strains, finite rotations, and effects of embedded piezocomposite actuators used by Shin and Cesnik (based on mixed variational intrinsic formulation of Hodges, 1990) is solved in the time domain. After the finite element discretization in the space domain, a set of first-order ordinary differential equations is obtained. To get the time integration results, second-order backward Euler method is used to discretize in time. Newton method is used to solve the nonlinear algebraic equations. The solution describes the displacement field, stress and strain field at each time step.

A computer program has been developed in this research to obtain numerical solutions to the above problems. This program can be used to generate solutions of the static and dynamic responses of curved and twist composite hingeless or articulated rotor blades under the action of arbitrary external loads.

The developed structural code is integrated with an unsteady vortex particle code (GENUVP) to form an aeroelastic simulation. The structural and aerodynamic modules are coupled together by two interfaces: one communicates aerodynamic loads to the structural model; the other communicates structural deformations and rates of deformation to the aerodynamic model. The aeroelastic analysis is realized in time domain by performing consecutive aerodynamic and structural time steps. A computer program has been developed to realize this aeroelastic modeling with which aeroelastic problems of fixed and rotating wings can be studied.

In the structural analysis, solutions of the present formulation are validated by experimental data and other numerical simulation results. The static and dynamic responses were tested for various conditions as follows:

- Isotropic and anisotropic blades
- Hingeless and articulated blades
- Blades with concentrated and distributed loads
- Active twist rotor with actuation

Of particular interest, the nonlinear accuracy of the method was verified against DYMORE. However, the present implementation in mixed form requires five to ten times more CPU time than the displacement-based formulation of DYMORE. This indicated that considerable improvements to the implementation of the code are possible and should be pursuit in the future.

In the aeroelastic analysis, the steady state of a fixed wing under different flight speeds have been obtained and results are consistent with other methods. The time response of the ATR blade in hover has also been tested, and blade twist as function of the applied piezoelectric-induced actuation is also investigated. The rotary-wing results obtained with the aeroelastic code using GENUVP and the present structural code lack in accuracy when compared to published results, even though they have good qualitatively agreement. For all these results, a single iteration between the aerodynamic and structural solvers were conducted at a given time step. To study the importance of sub-iterations within a given time step, a quasi-steady aerodynamic model was used in place of GENUVP. The coupling routines were the same. The results from this test showed that sub-iterations may be needed to improve stability and accuracy of the solution.

6.2 Recommendations

Though the modeling and programs show good results for most of the tested cases, they still need improvements.

- In the aeroelastic modeling, the sub-iterations between the aerodynamic and structural components at every time step may be required to get more accurate and stable results. This will increase the computational cost of each time step in the aeroelastic solution.
- The aeroelastic program is capable of simulating the forward flight cases. To this end, some effort in the preparation of input files for the aerodynamic module is needed.
- A theoretical analysis of the scheme used for the time integration of the nonlinear structural component is desirable. The best scheme to be used on this problem should present the following characteristics: unconditional stability, second order accuracy, and high frequency numerical damping.

Appendix A

Jacobian Matrix for Newton Method

$$[J] = [\frac{\partial Fs}{\partial X}]$$

The expression for ith element is:

$$\frac{\partial Fs}{\partial X} = \begin{bmatrix} 0 & \frac{\partial f_{u_i}}{\partial \theta} & \frac{\partial f_{u_i}}{\partial F} & 0 & \frac{\partial f_{u_i}}{\partial P} & 0 \\ 0 & \frac{\partial f_{\psi_i}}{\partial \theta} & \frac{\partial f_{\psi_i}}{\partial F} & \frac{\partial f_{\psi_i}}{\partial M} & \frac{\partial f_{\psi_i}}{\partial P} & \frac{\partial f_{\psi_i}}{\partial H} \\ \frac{\partial f_{F_i}}{\partial u} & \frac{\partial f_{F_i}}{\partial \theta} & \frac{\partial f_{F_i}}{\partial F} & \frac{\partial f_{F_i}}{\partial M} & 0 & 0 \\ 0 & \frac{\partial f_{M_i}}{\partial \theta} & \frac{\partial f_{M_i}}{\partial F} & \frac{\partial f_{M_i}}{\partial M} & 0 & 0 \\ \frac{\partial f_{P_i}}{\partial u} & \frac{\partial f_{P_i}}{\partial \theta} & 0 & 0 & \frac{\partial f_{P_i}}{\partial P} & \frac{\partial f_{P_i}}{\partial H} \\ 0 & \frac{\partial f_{H_i}}{\partial \theta} & 0 & 0 & \frac{\partial f_{H_i}}{\partial P} & \frac{\partial f_{H_i}}{\partial H} \\ 0 & \frac{\partial f_{u_j}}{\partial \theta} & \frac{\partial f_{u_j}}{\partial F} & 0 & \frac{\partial f_{u_j}}{\partial P} & 0 \\ 0 & \frac{\partial f_{\psi_j}}{\partial \theta} & \frac{\partial f_{\psi_j}}{\partial F} & \frac{\partial f_{\psi_j}}{\partial M} & \frac{\partial f_{\psi_j}}{\partial P} & \frac{\partial f_{\psi_j}}{\partial H} \\ \frac{\partial f_{F_j}}{\partial u} & \frac{\partial f_{F_j}}{\partial \theta} & \frac{\partial f_{F_j}}{\partial F} & \frac{\partial f_{F_j}}{\partial M} & \frac{\partial f_{F_j}}{\partial P} & \frac{\partial f_{F_j}}{\partial H} \\ 0 & \frac{\partial f_{M_j}}{\partial \theta} & \frac{\partial f_{M_j}}{\partial F} & \frac{\partial f_{M_j}}{\partial M} & 0 & 0 \end{bmatrix}$$

with $\frac{\partial C}{\partial \theta_k}=\frac{1}{1+\frac{\theta^T\theta}{4}}\left[-\frac{e_k^T\theta}{2}(\Delta+C)-\widetilde{e}_k+\frac{1}{2}(e_k\theta^T+\theta e_k^T)\right]$ for k=1,2,3,

$$\frac{\partial \dot{P}_i}{\partial P}=\frac{\partial\left(\frac{3P_i^n-4P_i^{n-1}+P_i^{n-2}}{2\Delta t}\right)}{\partial P^n}=\frac{3}{2\Delta t}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$$

$$\frac{\partial \dot{H}_i}{\partial H}=\frac{\partial\left(\frac{3H_i^n-4H_i^{n-1}+H_i^{n-2}}{2\Delta t}\right)}{\partial H^n}=\frac{3}{2\Delta t}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$$

$$\frac{\partial \dot{u}_i}{\partial u}=\frac{\partial\left(\frac{3u_i^n-4u_i^{n-1}+u_i^{n-2}}{2\Delta t}\right)}{\partial u^n}=\frac{3}{2\Delta t}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$$

the submatrices in the above are derived as:

$$\frac{\partial f_{u_i}}{\partial \theta_k}=-\frac{\partial C^T}{\partial \theta_k}C^{ab}F_i+\frac{\Delta l_i}{2}\widetilde{\omega}_a\frac{\partial C^T}{\partial \theta_k}C^{ab}P_i+\frac{\Delta l_i}{2}(\frac{\partial \dot{C}^T}{\partial \theta_k}C^{ab}P_i+\frac{\partial C^T}{\partial \theta_k}C^{ab}\dot{P}_i)$$

$$\frac{\partial f_{u_i}}{\partial F}=-C^TC^{ab}$$

$$\frac{\partial f_{u_i}}{\partial P}=\frac{\Delta l_i}{2}\widetilde{\omega}_aC^TC^{ab}+\frac{\Delta l_i}{2}(\dot{C}^TC^{ab}+C^TC^{ab}\frac{\partial \dot{P}_i}{\partial P})$$

$$\frac{\partial f_{\psi_i}}{\partial \theta_k}=-\frac{\partial C^T}{\partial \theta_k}C^{ab}\{M_i+\frac{\Delta l_i}{2}[(\widetilde{e}_1+\widetilde{\gamma}_i)F_i-\widetilde{V}P_i]\}$$
$$+\frac{\Delta l_i}{2}\widetilde{\omega}_a\frac{\partial C^T}{\partial \theta_k}C^{ab}H_i+\frac{\Delta l_i}{2}(\frac{\partial \dot{C}^T}{\partial \theta_k}C^{ab}H_i+\frac{\partial C^T}{\partial \theta_k}C^{ab}\dot{H}_i)$$

$$\frac{\partial f_{\psi_i}}{\partial F}=-\frac{\Delta l_i}{2}C^TC^{ab}(\widetilde{e}_1+\widetilde{\gamma}_i-\widetilde{F}_i\frac{\partial \gamma}{\partial F})$$

$$\frac{\partial f_{\psi_i}}{\partial M}=-C^TC^{ab}(\Delta-\frac{\Delta l_i}{2}\widetilde{F}_i\frac{\partial \gamma}{\partial M})$$

$$\frac{\partial f_{\psi_i}}{\partial P}=\frac{\Delta l_i}{2}C^TC^{ab}(\widetilde{V}_i-\widetilde{P}_i\frac{\partial V}{\partial P})$$

$$\frac{\partial f_{\psi_i}}{\partial H} = \frac{\Delta l_i}{2}\widetilde{\omega}_a C^T C^{ab} - \frac{\Delta l_i}{2} C^T C^{ab}\widetilde{P}_i \frac{\partial V}{\partial H} + \frac{\Delta l_i}{2}(\dot{C}^T C^{ab} H_i + C^T C^{ab}\frac{\partial \dot{H}_i}{\partial H})$$

$$\frac{\partial f_{F_i}}{\partial u} = \Delta$$

$$\frac{\partial f_{F_i}}{\partial \theta} = -\frac{\Delta l_i}{2}\frac{\partial C^T}{\partial \theta_k} C^{ab}(e_1 + \gamma_i)$$

$$\frac{\partial f_{F_i}}{\partial F} = -\frac{\Delta l_i}{2} C^T C^{ab}\frac{\partial \gamma}{\partial F}$$

$$\frac{\partial f_{F_i}}{\partial M} = -\frac{\Delta l_i}{2} C^T C^{ab}\frac{\partial \gamma}{\partial M}$$

$$\frac{\partial f_{M_i}}{\partial \theta} = \Delta - \frac{\Delta l_i}{2}(\frac{\widetilde{e}_k}{2} + \frac{e_k\theta^T + \theta e_k^T}{2})C^{ab}\kappa_i$$

$$\frac{\partial f_{M_i}}{\partial F} = -\frac{\Delta l_i}{2}(\Delta + \frac{\widetilde{\theta}_i}{2} + \frac{\theta_i\theta_i^T}{2})C^{ab}\frac{\partial \kappa}{\partial F}$$

$$\frac{\partial f_{M_i}}{\partial M} = -\frac{\Delta l_i}{2}(\Delta + \frac{\widetilde{\theta}_i}{2} + \frac{\theta_i\theta_i^T}{2})C^{ab}\frac{\partial \kappa}{\partial M}$$

$$\frac{\partial f_{P_i}}{\partial u} = -\widetilde{\omega}_a - \frac{\partial \dot{u}_i}{\partial u}$$

$$\frac{\partial f_{P_i}}{\partial \theta_k} = \frac{\partial C^T}{\partial \theta_k} C^{ab} V_i$$

$$\frac{\partial f_{P_i}}{\partial P} = C^T C^{ab}\frac{\partial V_i}{\partial P}$$

$$\frac{\partial f_{P_i}}{\partial H} = C^T C^{ab}\frac{\partial V_i}{\partial H}$$

$$\frac{\partial f_{H_i}}{\partial \theta_k} = -C^{ba}\frac{\partial C}{\partial \theta_k}\omega_a - C^{ba}\frac{\partial\left(\dfrac{\Delta - \dfrac{\widetilde{\theta}_i}{2}}{1 + \dfrac{\theta_i^T\theta_i}{4}}\right)\dot{\theta}_i}{\partial \theta_k}$$

$$\frac{\partial f_{H_i}}{\partial P} = \frac{\partial \Omega}{\partial P}$$

$$\frac{\partial f_{H_i}}{\partial H} = \frac{\partial \Omega}{\partial H}$$

$$\frac{\partial f_{u_j}}{\partial \theta_k} = \frac{\partial C^T}{\partial \theta_k} C^{ab} F_i + \frac{\Delta l_i}{2} \widetilde{\omega}_a \frac{\partial C^T}{\partial \theta_k} C^{ab} P_i + \frac{\Delta l_i}{2} (\frac{\partial \dot{C}^T}{\partial \theta_k} C^{ab} P_i + \frac{\partial C^T}{\partial \theta_k} C^{ab} \dot{P}_i)$$

$$\frac{\partial f_{u_j}}{\partial F} = C^T C^{ab}$$

$$\frac{\partial f_{u_j}}{\partial P} = \frac{\Delta l_i}{2} \widetilde{\omega}_a C^T C^{ab} + \frac{\Delta l_i}{2} (\dot{C}^T C^{ab} + C^T C^{ab} \frac{\partial \dot{P}_i}{\partial P})$$

$$\frac{\partial f_{\psi_j}}{\partial \theta_k} = \frac{\partial C^T}{\partial \theta_k} C^{ab} \{M_i - \frac{\Delta l_i}{2} [(\widetilde{e}_1 + \widetilde{\gamma}_i) F_i - \widetilde{V} P_i]\}$$

$$+ \frac{\Delta l_i}{2} \widetilde{\omega}_a \frac{\partial C^T}{\partial \theta_k} C^{ab} H_i + \frac{\Delta l_i}{2} (\frac{\partial \dot{C}^T}{\partial \theta_k} C^{ab} H_i + \frac{\partial C^T}{\partial \theta_k} C^{ab} \dot{H}_i)$$

$$\frac{\partial f_{\psi_j}}{\partial F} = -\frac{\Delta l_i}{2} C^T C^{ab} (\widetilde{e}_1 + \widetilde{\gamma}_i - \widetilde{F}_i \frac{\partial \gamma}{\partial F})$$

$$\frac{\partial f_{\psi_j}}{\partial M} = C^T C^{ab} (\Delta + \frac{\Delta l_i}{2} \widetilde{F}_i \frac{\partial \gamma}{\partial M})$$

$$\frac{\partial f_{\psi_j}}{\partial P} = \frac{\Delta l_i}{2} C^T C^{ab} (\widetilde{V}_i - \widetilde{P}_i \frac{\partial V}{\partial P})$$

$$\frac{\partial f_{\psi_j}}{\partial H} = \frac{\Delta l_i}{2} \widetilde{\omega}_a C^T C^{ab} - \frac{\Delta l_i}{2} C^T C^{ab} \widetilde{P}_i \frac{\partial V}{\partial H} + \frac{\Delta l_i}{2} (\dot{C}^T C^{ab} H_i + C^T C^{ab} \frac{\partial \dot{H}_i}{\partial H})$$

$$\frac{\partial f_{F_j}}{\partial u} = -\Delta$$

$$\frac{\partial f_{F_j}}{\partial \theta} = -\frac{\Delta l_i}{2} \frac{\partial C^T}{\partial \theta_k} C^{ab} (e_1 + \gamma_i)$$

$$\frac{\partial f_{F_j}}{\partial F} = -\frac{\Delta l_i}{2} C^T C^{ab} \frac{\partial \gamma}{\partial F}$$

$$\frac{\partial f_{F_j}}{\partial M} = -\frac{\Delta l_i}{2} C^T C^{ab} \frac{\partial \gamma}{\partial M}$$

$$\frac{\partial f_{M_j}}{\partial \theta} = -\Delta - \frac{\Delta l_i}{2} (\frac{\widetilde{e}_k}{2} + \frac{e_k \theta^T + \theta e_k^T}{2}) C^{ab} \kappa_i$$

$$\frac{\partial f_{M_j}}{\partial F} = -\frac{\Delta l_i}{2}(\Delta + \frac{\widetilde{\theta}_i}{2} + \frac{\theta_i \theta_i^T}{2}) C^{ab} \frac{\partial \kappa}{\partial F}$$

$$\frac{\partial f_{M_j}}{\partial M} = -\frac{\Delta l_i}{2}(\Delta + \frac{\widetilde{\theta}_i}{2} + \frac{\theta_i \theta_i^T}{2}) C^{ab} \frac{\partial \kappa}{\partial M}$$

Appendix B

A Sample Case of the Input Format

This is a sample case of the input file used in the structural model code. This case is corresponding to the case of the ATR prototype blade for the hover condition which is described in Chapter 5. If the active control is on, another input file which has the control value at every time step is required. A MATLAB program which creates the control input file corresponding to the active input of twist moment in Fig. 4-55 is included at the end of this appendix.

Basic Input File Format

```
1     !BOUNDARY CONDITION 0-BENCH,1-HINGE
0     !ACTIVE CONTROL 0-WITHOUT ACTIVE CONTROL, 1-WITH ACTIVE CONTROL
1     !NUMBER OF BLADE
15    !NUMBER OF ELEMENT
5000      !NUMBER OF INTEGRATION TIME STEPS
1.0e-3   !TIME STEP SIZE
1.3208D0  !LENGTH OF BEAM
0.0762D0  !ROOT OFFSET
0.0983   !DL(1)
0.1430   !DL(2)
0.0597   !DL(3)
0.0850   !DL(4)
0.0850   !DL(5)
0.0850   !DL(6)
0.0850   !DL(7)
0.0850   !DL(8)
0.0850   !DL(9)
0.0850    !DL(10)
```

```
0.0850   !DL(11)
0.0850   !DL(12)
0.0850   !DL(13)
0.0850   !DL(14)
0.0850   !DL(15)
72.0d0   !ROTATING VELOCITY
1        !INCREASE SPEED
8.0D0    !PITCH CONSTANT
0.0D0    !PCOS
0.0D0    !PSIN
  -0.3511        !PRETWIST-ELEMENT1
  -1.2129        !PRETWIST-ELEMENT2
  -1.9368              !PRETWIST-ELEMENT3
  -2.4536              !PRETWIST-ELEMENT4
  -3.0607              !PRETWIST-ELEMENT5
  -3.6679              !PRETWIST-ELEMENT6
  -4.2750              !PRETWIST-ELEMENT7
  -4.8821              !PRETWIST-ELEMENT8
  -5.4893              !PRETWIST-ELEMENT9
  -6.0964              !PRETWIST-ELEMENT10
  -6.7036              !PRETWIST-ELEMENT11
  -7.3107              !PRETWIST-ELEMENT12
  -7.9179              !PRETWIST-ELEMENT13
  -8.5250              !PRETWIST-ELEMENT14
  -9.1321              !PRETWIST-ELEMENT15
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 1
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 1
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 2
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 2
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
```

0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 3
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 3
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 4
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 4
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 5
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 5
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 6
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 6
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000

```
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 7
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 7
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 8
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 8
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 9
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 9
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 10
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 10
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
```

0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 11
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 11
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 12
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 12
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 13
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 13
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 14
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 14
1.02e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.65e-007 0.00e+000 0.00e+000 0.00e+000

```
0.00e+000 0.00e+000 0.00e+000 1.98e-003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 1.50e-002 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 8.03e-004
!INVERSE OF STIFFNESS MATRIX FOR ELEMENT 15
2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 2.54e-001 0.00e+000 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 1.90e+003 0.00e+000 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 3.75e+004 0.00e+000
0.00e+000 0.00e+000 0.00e+000 0.00e+000 0.00e+000 2.00e+003
!INVERSE OF MASS MATRIX FOR ELEMENT 15
```

Program for Active Control Input File

```
% This program creates active signal file 'volt.dat'
% It is used as an input file if the active control in the structural code.
% Time range
start_time = 0.0;
end_time = 1.0;
% dt
time_int = 2.0e-04;
tim = 0.:time_int:(end_time-start_time);
% Amplitude
amplitude = 0.6372067;
phase = 0.;
% Frequancy range
start_freq = 1.;
end_freq = 100.;
%
N = length(tim);
freq_inc = (end_freq - start_freq) / N ;
freq = start_freq:freq_inc:end_freq;
freq1 = freq(1:N);
%
volt = amplitude*sin(2.*pi*(freq1.*tim+phase));
% Plot
figure(1);
plot(tim, volt,'r');
xlabel('Time (sec)');
ylabel('Actuation Twist Moment (N.m)')
% Output
fid = fopen('volt.dat','w');
for i=1:N
  fprintf(fid,'%12.8f\n',volt(i));
end
fclose(fid);
```

Appendix C

High-Resolution Aerodynamic Analysis

The aerodynamic model is implemented via the GENUVP code: GENeral Unsteady Vortex Particle code [4]. It was developed at the National Technical University of Athens (NTUA), Greece, and it has been modified at Carleton University, Canada. It is a tool for high-resolution prediction of unsteady flow for multi-component configurations such as helicopters and wind turbines.

The domain decomposition concept is used in this model. The velocity $\vec{u}$ is decomposed as follows:

$$\vec{u}(\vec{x},t) = \vec{u}_{ext}(\vec{x},t) + \vec{u}_{wake}(\vec{x},t) + \vec{u}_{solid}(\vec{x},t) \qquad \text{(C.1)}$$

where $\vec{u}_{ext}(\vec{x},t)$ and $\vec{u}_{wake}(\vec{x},t)$ denote the external flow and wake flow respectively. And $\vec{u}_{solid}$ is associated with surface singularity distributions by Green's theorem. Obviously the rotational part of velocity $\vec{u}$ is associated to the wake flow $\vec{u}_{wake}(\vec{x},t)$, while the irrotational part $\vec{u}_{solid}$ takes account for the flow induced by moving solid boundaries. For the rotational part of the flow, Helmhotz decomposition is expressed as volume convolution of the vorticity. For the potential (irrotational part) $\vec{u}_{solid}$, boundary integral methods can be used to represent it.

Boundary Integral Methods are used to approximate the potential part. Two integral equations are defined for the velocity potential and velocity:

$$\phi(\vec{x}_0) = -\int_S \sigma(\vec{x}) \frac{1}{4\pi r} dS - \int_S \mu(\vec{x}) \frac{\partial}{\partial v} \frac{1}{4\pi r} dS \tag{C.2}$$

$$\vec{u}(\vec{x}_0) = \int_S \sigma(\vec{x}) \frac{\vec{r}}{4\pi r^3} dS + \int_S \vec{\gamma}(\vec{x}) \frac{\vec{r}}{4\pi r^3} dS \tag{C.3}$$

where ϕ is the velocity potential, $\vec{u}$ is a velocity field, $\vec{v}$ is the unit normal to the surface boundary S with direction towards the flow field D, σ and μ denote surface distributions of the jumps of $\frac{\partial \phi}{\partial v}$ (sources) and $-\phi$ (dipoles) respectively, and $\vec{\gamma}$ which is the surface vorticity defined as $\vec{\gamma} = \nabla\mu \times \vec{v}$. A zero order BEM is used and sub-grid techniques are applied to reduce the cost when dense paneling is required.

Vortex blob approximations are used for the wake to reduce the computational cost. Biot-Savart law gives:

$$\vec{u}_{wake}(\vec{x}_0, t) = \int_{D_\omega(t)} \frac{\vec{\omega}(\vec{x}, t) \times (\vec{x}_0 - \vec{x})}{4\pi \mid \vec{x}_0 - \vec{x} \mid} dD \tag{C.4}$$

where $D_\omega(t)$ is the support of vorticity and is decomposed into volume elements $D_{\omega,j}(t), j \in J(t)$ to each of which a point vortex is defined. According to Biot-Savart law, the velocity and the deformation at every blob position are needed for conventional vortex methods. Convection of free vorticity is carried out in Lagrangian description. The Biot-Savart law is used in the areas of great importance to make sure the accuracy and the Particle-Mesh (PM) techniques is applied downstream to reduce the computational cost.

As for the near-to-far field coupling conditions, the separation is modeled using the double wake concept. It assumes that the principle consequence of separation is the formation of a pronounced shear layer. Separation point can be predicted internally or specified externally. In order to approximate wakes, they are introduced as vortex sheets. In GENUVP, only the strip of the wake, which was obtained during the current time step, keeps its "surface" character.

Cost effective model is achieved by using the domain decomposition concept. The flow field can be classified either as *near-field* or as *far-field*. The *near-field* is the region close to the solid boundaries. It contains weak shock waves, boundary layer regions and the part of the wakes contacting the solid boundaries. The *far-field* is the region that contains the different components of the wake. By using grid-free vortex methods, it is capable of calculating multi-component configurations with the near-field analysis, all sharing the same far-field analysis.

Above all, the Helmholtz decomposition was used to formulate cost effective numerical schemes of high resolution for unsteady flow simulation around multi-component configurations in GENUVP code.

Appendix D

Structural Model Code

What follows shows the source code of the structure code. It is written in FORTRAN 77 and it was tested in SunOS Unix FORTRAN complier and in Digital FORTRAN (former Microsoft FORTRAN) for Windows 2000.

```
C*********************************************************************
C
C     THE INPUT/OUTPUT FILES
C
C     COMMON BLOCKS FILES    'cATRc.f'
C     ATR_INITIALIZE---INITIALIZE ALL THE VARIABLES
C     STRUCTURAL_COMPONENT---MAIN PROGRAM FOR THE STRUCTURE CODE
C     equations---Caculate Fx
C     Jacobi--- Calculate Jacobi matrix
C     unknows---Calculat each variable from vector X
C     ludcmp2---LU decomposition
C     lubksb2---Back substitution
C     t---Convert a vector to its dual matrix
C     ctlyb---Calculate transpose of C matrix from theta
C     CTd---Calculate dCT/dt,dP/dt,dH/dt,du/dt,dtheta/dt using
C          finite difference method
C     dif---Calculate dCTdot/dtheta
C     mm---Multiply of two matrices
C     mv---Multiply of matrix and vector
C     vv---Multiply of two vectors
C     m_m---Plus of two matrices
C     v_v---Plus of two vectors
C     cross---Convert a column vector to its dual matrix
C     OUTPUT--- Output unknown variables to different files for plot
C
C*********************************************************************
```

```
C*********************************************************************
C   INITIALIZE ALL THE VARIABLES
C*********************************************************************

        subroutine ATR_INITIALIZE
        include 'cATRc.f'
        integer i,j,k,N_BLADE
        DOUBLE PRECISION L,anglerad
C   E IS A IDENTITY MATRIX
      do i=1,3
      do j=1,3
      e(i,j)=0.0d0
      end do
      end do
     do 2 i=1,3
2    e(i,i)=1.0d0
C       READ PARAMETERS FROM INPUT FILE
   OPEN(UNIT=4,FILE='inputc.dat')
        READ(4,*)BC
   READ(4,*)ACT
        READ(4,*)NOB
        READ(4,*)NES
        READ(4,*)INT
   READ(4,*)DTs
        READ(4,*)L
        READ(4,*)BEAMROOT
        DO I=1,NES
        READ(4,*)DL(I)
        END DO
C   ROTATE VELOCITY
   Read(4,*)w
   Read(4,*)is
C   CONSTANTS OF CONTROL PITCH
   READ(4,*)PCON
        READ(4,*)PCOS
        READ(4,*)PSIN

C   READ PRETWIST
        DO I=1,NES
        READ(4,*)anglerad
        PRETWIST(I)=anglerad*PiPi/180
        END DO

C   Read Material Properties
C   THE INVERSE OF STIFFNESS AND MASS MATRICES FOR EACH ELEMENT
     do j=1,NES
     do 204 i=1,3
    Read(4,*)dgamadF(i,1,J),dgamadF(i,2,J),
  $dgamadF(i,3,J),
  $dgamadM(i,1,J),dgamadM(i,2,J),dgamadM(i,3,J)
 204  continue
     do 205 i=1,3
   Read(4,*)dkapadF(i,1,J),dkapadF(i,2,J),
  $dkapadF(i,3,J),
  $dkapadM(i,1,J),dkapadM(i,2,J),dkapadM(i,3,J)
```

```
 205  continue
      do 206 i=1,3
     Read(4,*)dVdP(i,1,J),dVdP(i,2,J),dVdP(i,3,J),
    $  dVdH(i,1,J),dVdH(i,2,J),dVdH(i,3,J)
 206  continue
      do 207 i=1,3
      Read(4,*)dOmegadP(i,1,J),dOmegadP(i,2,J),
    $  dOmegadP(i,3,J),
    $dOmegadH(i,1,J),dOmegadH(i,2,J),dOmegadH(i,3,J)
 207  continue
     end do

C    FOR ACTUATION CASE, READ ACTUATION SIGNAL FROM INPUT FILE
         if(ACT.gt.0)then
         open(unit=50,file='volt.dat')
         do i=1,INT/2
         read(50,*)twistactive(i)
         end do
         end if

C    INITIALIZE
         do N_BLADE=1,NOB

C         INITIALIZE Cab
         DO k=1,NES
         DO i=1,3
         do j=1,3
         Cab(j,i,k,N_BLADE)=0.0D0
         end do
         end do
         end do
C     Initial Condition
     open(unit=5,file='xwholel.dat')
         do i=1,18*NES+12
         read(5,*)xwhole(i,0,N_BLADE)
         enddo
         close(5)
         do i=1,18*NES+12
      X(i,N_BLADE)=Xwhole(i,0,N_BLADE)
     end do

      do 202 i=1,3*NES
       va(i,N_BLADE)=0.0d0
       wa(i,N_BLADE)=0.0d0
 202   continue
C     Initialize Forces
      do 203 i=1,3*NES+3
       fa(i,N_BLADE)=0.0d0
       ma(i,N_BLADE)=0.0d0
 203  continue
      do i=1,3*NES
         Factive(i,N_BLADE)=0.0d0
         Mactive(i,N_BLADE)=0.0d0
          end do

          end do
```

```
    close(4)
C   CALCULATE RADIUS AT THE MIDDLE OF EACH ELEMENT
    DO I=1,NES
        RADIUS(I)=0.0D0
        DO J=1,I
        RADIUS(I)=RADIUS(I)+DL(J)
        END DO
        RADIUS(I)=RADIUS(I)-DL(I)/2.0D0+beamroot
        END DO
        RADIUS(NES+1)=L+beamroot

    END


C*******************************************************************
C   MAIN PROGRAM FOR THE STRUCTURE CODE
C*******************************************************************

        SUBROUTINE STRUCTURAL_COMPONENT(i,N_BLADE)
     INCLUDE 'cATRc.f'
        double precision Fx(18*NCWM+12),Jx(18*NCWM+12,18*NCWM+12)
        double precision ynorm
    double precision fiangle
        integer i,j,newton,iii,ii,N_BLADE
        Integer inin(18*NCWM+12)
        double precision d
        double precision Xn(18*NCWM+12),Xn1(18*NCWM+12)

C   FOR ROTATING AND ACTUATION CASE, ACTUATION SIGNAL
C   SHOULD BE INPUTED AFTER THE ROTATING SPEED IS REACHED
C   FOR BOTH AERO AND STRUCTURE CODES.
C    if(i.lt.501)then
C       act=0
C       else
C       act=1
C       endif

C   PRESENT TIME STEP IN STUCTURAL COMPONENT
        write(*,*)i
C   INITIAL VELOCITY(ANGULAR AND LINEAR VELOCITIES FOR EACH ELEMENT)
     if (i.lt.is)then
     DO j=1,NES
     WA(3*j,N_BLADE)=i*w/is
     END DO
     fiangle=0.5*WA(3*j,N_BLADE)*(I*DTS)
     else
     DO j=1,NES
     WA(3*j,N_BLADE)=w
     END DO
     fiangle=0.5*WA(3*j,N_BLADE)*(IS*DTS)
    $       +WA(3*j,N_BLADE)*(I-IS)*DTS
     end if
C   LINEAR VELOCITY FOR EACH ELEMENT FOR HOVER
     DO j=1,NES
     VA(3*j-1,N_BLADE)=RADIUS(j)*WA(3*j,N_BLADE)
```

```
      END DO

C   CACULATE PITCH ANGLE (COLLECTIVE PITCH ANGLE+SOME TIME FUNCTION)
     PITCHANGLE(I,N_BLADE)=(PCON
    $       +PCOS*COS(fiangle+N_BLADE*PiPi/2)
    $       +PSIN*SIN(fiangle+N_BLADE*PiPi/2))*PiPi/180

C   CACULATE TRANSFORMATION MATRIX Cab BETWEEN GLOBAL FRAME a AND
C   UNDEFORMED BLADE FRAME b DUE TO PRETWIST OF THE BLADE
     DO II=1,NES
          Cab(1,1,II,N_BLADE)=1.0D0
          DO J=2,3
          Cab(J,J,II,N_BLADE)=
    $     COS(PRETWIST(II))
          END DO
          Cab(2,3,II,N_BLADE)=
    $-sin(PRETWIST(II))
     Cab(3,2,II,N_BLADE)=
    $sin(PRETWIST(II))
          END DO
C   Cba IS THE TRANSPOSE OF Cab
          DO II=1,NES
          DO J=1,3
          DO JJ=1,3
          Cba(JJ,J,II,N_BLADE)=Cab(J,JJ,II,N_BLADE)
          END DO
          END DO
          END DO

C   FOR ACTUATION CASE, SET THE ACTUATION FORCES VECTOR
     if (ACT.eq.1)then
          do 300 j=3,NES-1
          Factive(3*j+1,N_BLADE)=0.0d0
          Factive(3*j+2,N_BLADE)=0.0d0
          Factive(3*j+3,N_BLADE)=0.0d0
          Mactive(3*j+1,N_BLADE)=twistactive(i-500)
          Mactive(3*j+2,N_BLADE)=0.0
          Mactive(3*j+3,N_BLADE)=0.0
300   continue
     end if

C    INITIALIZE Fx
           do 20 j=1,18*NES+12
           Fx(j)=1.0d0
20     continue

C    Second-order Euler Method
C    GIVE THE VALUE OF LAST TWO TIME STEPS
             do 100 j=1,18*NES+12
               Xn(j)=Xwhole(j,i-1,N_BLADE)
               Xn1(j)=0.0d0
               if (i.gt.1)then
                Xn1(j)=Xwhole(j,i-2,N_BLADE)
               end if
100      Continue
```

```
C    MAXIMUM ITERATION TIME
          iii=1000

C     Newton Method
     do 70 newton=1,iii

C     Check for convergence
            ynorm=0.0
              do 80 j=1,18*NES+12
                ynorm=ynorm+Fx(j)**2
80            continue
        ynorm=sqrt(ynorm)
          write(*,*)ynorm
C    GET THE VALUE OF ALL THE VARIABLES FROM THE UNKNOW VECTOR X
      call unknows(I,N_BLADE)

C     Check for convergence
            if (ynorm.gt.1e-8)then

C          Convergence not reached Prepare next iteration
C    CALCULATE EQUATION VALUE
           Call equations(i,Xn1,Xn,Fx,N_BLADE)
          do 101 j=1,18*NES+12
          Fx(j)=-1*Fx(j)
101  continue

C    CALCULATE JACOBI MATRIX
          Call Jacobi(i,Jx,N_BLADE)

C     Solve equations
          Call ludcmp2(Jx,18*NES+12,18*NCWM+12,inin,d)
          Call lubksb2(Jx,18*NES+12,18*NCWM+12,inin,Fx)

C    CACULATE NEW X FOR PRESENT TIME STEP
          do 90 j=1,18*NES+12
                X(j,N_BLADE)=X(j,N_BLADE)+Fx(j)
90     continue

            else
C    Convergence reached AT PRESENT TIME STEP
C    THEN GO TO NEXT TIME STEP
                goto 1000
            end    if

70      Continue

C    SAVE X OF PRESENT TIME STEP    TO XWHOLE
1000  do 110 j=1,18*NES+12
    Xwhole(j,i,N_BLADE)=X(j,N_BLADE)
110  continue

          END

C*****************************************************************
C    unknows-Calculat each variable from vector X
```

```
C*********************************************************************

        Subroutine unknows(IP,N_BLADE)
    INCLUDE 'cATRc.f'
        double precision theta3(3),dtheta3
    integer i,j,N_BLADE,IP

C   GIVE THE VALUE OF ALL VARIABLES FROM X
        do 10 i=1,3
    F1(i)=X(i,N_BLADE)
        uN1(i)=X(18*NES+6+i,N_BLADE)
    thetaN1(i)=X(18*NES+9+i,N_BLADE)
 10 continue

         if(BC.eq.1)then  !ARTICULATED CASE
                M1(1)=X(4,N_BLADE)
                M1(2)=0.0d0
C   ADD LEADLAG AND FLAP DAMPER FOR HINGE CASE
    if(IP.GE.2)then
    dtheta3= (XWHOLE(6,IP-1,N_BLADE)-XWHOLE(6,IP-2,N_BLADE)) / DTs
        dtheta2= (XWHOLE(5,IP-1,N_BLADE)-XWHOLE(5,IP-2,N_BLADE)) / DTs
    else
    dtheta3=0.0d0
    dtheta2=0.0d0
        end if
    M1(2)=1.0d0*dtheta2
        M1(3)=10*dtheta3

C   SET HINGE TWIST EQUAL TO COLLECTIVE PITCH ANGLE
                theta0(1)=PITCHANGLE(IP,N_BLADE)
C   FOR HINGE CASE, LEADLAG AND FLAP
C   ANGLE AT HINGE ARE UNKNOWN VARIABLES
                theta0(2)=X(5,N_BLADE)
                theta0(3)=X(6,N_BLADE)

          else              !HINGELESS CASE
        do i=1,3
        M1(i)=X(3+i,N_BLADE)
        theta0(i)=0.0d0
        end do
C   SET HINGE TWIST EQUAL TO COLLECTIVE PITCH ANGLE
        theta0(1)=PITCHANGLE(IP,N_BLADE)
    end if

    do 20 i=1,NES
        do 30 j=1,3
    u(3*(i-1)+j)=X(18*(i-1)+6+j,N_BLADE)
    theta(3*(i-1)+j)=X(18*(i-1)+9+j,N_BLADE)
    theta3(j)=theta(3*(i-1)+j)
    F(3*(i-1)+j)=X(18*(i-1)+12+j,N_BLADE)
    M(3*(i-1)+j)=X(18*(i-1)+15+j,N_BLADE)
    PM(3*(i-1)+j)=X(18*(i-1)+18+j,N_BLADE)
    H(3*(i-1)+j)=X(18*(i-1)+21+j,N_BLADE)
30  continue

C   CONSTITUTIVE RELATION AND ACTIVE MODIFICATION
```

```
      do 40 j=1,3
      gama(3*(i-1)+j)=
     $ dgamadF(j,1,I)*(F(3*(i-1)+1)+Factive(3*(i-1)+1,N_BLADE))
     $ +dgamadF(j,2,I)*(F(3*(i-1)+2)+Factive(3*(i-1)+2,N_BLADE))
     $ +dgamadF(j,3,I)*(F(3*(i-1)+3)+Factive(3*(i-1)+3,N_BLADE))
     $ +dgamadM(j,1,I)*(M(3*(i-1)+1)+Mactive(3*(i-1)+1,N_BLADE))
     $ +dgamadM(j,2,I)*(M(3*(i-1)+2)+Mactive(3*(i-1)+2,N_BLADE))
     $ +dgamadM(j,3,I)*(M(3*(i-1)+3)+Mactive(3*(i-1)+3,N_BLADE))

      kapa(3*(i-1)+j)=
     $ dkapadF(j,1,I)*(F(3*(i-1)+1)+Factive(3*(i-1)+1,N_BLADE))
     $ +dkapadF(j,2,I)*(F(3*(i-1)+2)+Factive(3*(i-1)+2,N_BLADE))
     $ +dkapadF(j,3,I)*(F(3*(i-1)+3)+Factive(3*(i-1)+3,N_BLADE))
     $ +dkapadM(j,1,I)*(M(3*(i-1)+1)+Mactive(3*(i-1)+1,N_BLADE))
     $ +dkapadM(j,2,I)*(M(3*(i-1)+2)+Mactive(3*(i-1)+2,N_BLADE))
     $ +dkapadM(j,3,I)*(M(3*(i-1)+3)+Mactive(3*(i-1)+3,N_BLADE))

      V(3*(i-1)+j)=
     $dVdP(j,1,I)*PM(3*(i-1)+1)
     $+dVdP(j,2,I)*PM(3*(i-1)+2)
     $+dVdP(j,3,I)*PM(3*(i-1)+3)
     $+dVdH(j,1,I)*H(3*(i-1)+1)
     $+dVdH(j,2,I)*H(3*(i-1)+2)
     $+dVdH(j,3,I)*H(3*(i-1)+3)

      Omega(3*(i-1)+j)=
     $dOmegadP(j,1,I)*PM(3*(i-1)+1)
     $+dOmegadP(j,2,I)*PM(3*(i-1)+2)
     $+dOmegadP(j,3,I)*PM(3*(i-1)+3)
     $+dOmegadH(j,1,I)*H(3*(i-1)+1)
     $+dOmegadH(j,2,I)*H(3*(i-1)+2)
     $+dOmegadH(j,3,I)*H(3*(i-1)+3)

 40   continue

        call ctlyb(CTR(1,1,i),theta3)

      call mm(CTCab(1,1,i),CTR(1,1,i),Cab(1,1,i,N_BLADE))

20    continue

      END


C*******************************************************************
C     equations- Caculate Fx
C*******************************************************************

        Subroutine equations(j,Xn1,Xn,Fx,N_BLADE)
      INCLUDE 'cATRc.f'
        double precision Xn(18*NCWM+12),Xn1(18*NCWM+12)
        double precision CabCT(3,3,NCWM)
        double precision Fx(18*NCWM+12)
        integer i,j,ii,jj,kk,N_BLADE
        double precision dm
        double precision CTCabF(3),CTCabM(3),twaCTCabP(3),twaCTCabH(3),
```

```
     $  CTdotCabP(3),CTdotCabH(3),CTCabPdot(3),CTCabHdot(3),
     $  CTCabte1tgamaF(3),CTCabtVP(3),eCabkapa(3),Cabe1(3),e1gama(3),
     $  CTCabV(3),twau(3),Cbathetadot(3),CabCTwa(3),CTCabe1gama(3)
      double precision twa(3,3),twaCTCab(3,3),CTdotCab(3,3),te1(3,3),
     $     tgama(3,3),te1tgama(3,3),CTCabte1tgama(3,3),tV(3,3),
     $   CTCabtV(3,3),
     $   ttheta(3,3),ettheta(3,3),thetatheta(3,3),
     $   ettheta2(3,3),eCab(3,3),e_ttheta(3,3),Cbae(3,3)

C   INITIALIZE
         do 20 i=1,18*NES+12
         Fx(i)=0.0d0
20   Continue

         do 10 i=1,NES
C   SET THE VALUE OF THE LAST TWO TIME STEPS
         do 10 ii=1,3
      un_1(3*(i-1)+ii)=Xn1(18*(i-1)+6+ii)
      thetan_1(3*(i-1)+ii)=Xn1(18*(i-1)+9+ii)
      Pn1(3*(i-1)+ii)=Xn1(18*(i-1)+18+ii)
      Hn1(3*(i-1)+ii)=Xn1(18*(i-1)+21+ii)

      un(3*(i-1)+ii)=Xn(18*(i-1)+6+ii)
      thetan(3*(i-1)+ii)=Xn(18*(i-1)+9+ii)
      Pn(3*(i-1)+ii)=Xn(18*(i-1)+18+ii)
      Hn(3*(i-1)+ii)=Xn(18*(i-1)+21+ii)
10   Continue

         do 30 i=1,NES
         call CTd(dm,j,i)
     call mv(CTCabF,CTCab(1,1,i),F(3*(i-1)+1))
     call mv(CTCabM,CTCab(1,1,i),M(3*(i-1)+1))
     call cross(twa,wa(3*(i-1)+1,N_BLADE))
     call mm(twaCTCab,twa,CTCab(1,1,i))
      call mv(twaCTCabP,twaCTCab,PM(3*(i-1)+1))
          call mm(CTdotCab,CTdot,Cab(1,1,i,N_BLADE))
          call mv(CTdotCabP,CTdotCab,PM(3*(i-1)+1))
      call mv(CTdotCabH,CTdotCab,H(3*(i-1)+1))
          call mv(CTCabPdot,CTCab(1,1,i),Pdot)
          call mv(CTCabHdot,CTCab(1,1,i),Hdot)
          call cross(te1,e(1,1))
          call cross(tgama,gama(3*(i-1)+1))
          call m_m(te1tgama,te1,tgama)
          call mm(CTCabte1tgama,CTCab(1,1,i),te1tgama)
      call mv(CTCabte1tgamaF,CTCabte1tgama,F(3*(i-1)+1))
      call mv(twaCTCabH,twaCTCab,H(3*(i-1)+1))
      call cross(tV,V(3*(i-1)+1))
          call mm(CTCabtV,CTCab(1,1,i),tV)
          call mv(CTCabtVP,CTCabtV,PM(3*(i-1)+1))
          call v_v(e1gama,e(1,1),gama(3*(i-1)+1))
          call mv(CTCabe1gama,CTCab(1,1,i),e1gama)
          call mv(Cabe1,Cab(1,1,i,N_BLADE),e(1,1))
      call cross(ttheta,theta(3*(i-1)+1))

          do 80 ii=1,3
          do 80 jj=1,3
```

```
        ttheta(ii,jj)=ttheta(ii,jj)/2.0
80    continue

        call m_m(ettheta,e,ttheta)
        call vv(thetatheta,theta(3*(i-1)+1)
   $    ,theta(3*(i-1)+1))

        do 81 ii=1,3
        do 81 jj=1,3
        thetatheta(ii,jj)=thetatheta(ii,jj)/4.0
81    continue

        call m_m(ettheta2,ettheta,thetatheta)
        call mm(eCab,ettheta2,Cab(1,1,i,N_BLADE))
        call mv(eCabkapa,eCab,kapa(3*(i-1)+1))
      call mv(CTCabV,CTCab(1,1,i),V(3*(i-1)+1))
        call mv(twau,twa,u(3*(i-1)+1))

        do 90 ii=1,3
        do 90 jj=1,3
        CabCT(ii,jj,i)=CTCab(jj,ii,i)
90    Continue

      call mv(CabCTwa,CabCT(1,1,i),wa(3*(i-1)+1,N_BLADE))

        do 100 ii=1,3
        do 100 jj=1,3
        ttheta(ii,jj)=-1*ttheta(ii,jj)
100   continue

        call m_m(e_ttheta,e,ttheta)

        do 105 ii=1,3
        do 105 jj=1,3
        e_ttheta(ii,jj)=e_ttheta(ii,jj)/dm
105    Continue

     call mm(Cbae,Cba(1,1,i,N_BLADE),e_ttheta)
        call mv(Cbathetadot,Cbae,thetadot)

            do 70 kk=1,3

        Fx(18*(i-1)+kk)=Fx(18*(i-1)+kk)-CTCabF(kk)
    $+DL(I)/2*twaCTCabP(kk)+DL(I)/2*(CTdotCabP(kk)+CTCabPdot(kk))

     Fx(18*(i-1)+18+kk)=Fx(18*(i-1)+18+kk)+CTCabF(kk)
    $+DL(I)/2*twaCTCabP(kk)+DL(I)/2*(CTdotCabP(kk)+CTCabPdot(kk))

     Fx(18*(i-1)+3+kk)=Fx(18*(i-1)+3+kk)-CTCabM(kk)
    $-DL(I)/2*CTCabte1tgamaF(kk)+DL(I)/2*(twaCTCabH(kk)+CTCabtVP(kk))
    $+DL(I)/2*(CTdotCabH(kk)+CTCabHdot(kk))

     Fx(18*(i-1)+21+kk)=Fx(18*(i-1)+21+kk)+CTCabM(kk)
    $-DL(I)/2*CTCabte1tgamaF(kk)+DL(I)/2*(twaCTCabH(kk)+CTCabtVP(kk))
    $+DL(I)/2*(CTdotCabH(kk)+CTCabHdot(kk))
```

```
      Fx(18*(i-1)+6+kk)=Fx(18*(i-1)+6+kk)
     $+u(3*(i-1)+kk)-DL(I)/2*(CTCabe1gama(kk)-Cabe1(kk))

      Fx(18*(i-1)+24+kk)=Fx(18*(i-1)+24+kk)
     $-u(3*(i-1)+kk)-DL(I)/2.0*(CTCabe1gama(kk)-Cabe1(kk))

      Fx(18*(i-1)+9+kk)=Fx(18*(i-1)+9+kk)
     $+theta(3*(i-1)+kk)-DL(I)/2.0*eCabkapa(kk)

      Fx(18*(i-1)+27+kk)=Fx(18*(i-1)+27+kk)
     $-theta(3*(i-1)+kk)-DL(I)/2.0*eCabkapa(kk)

      Fx(18*(i-1)+12+kk)=Fx(18*(i-1)+12+kk)
     $+CTCabV(kk)-va(3*(i-1)+kk,N_BLADE)-twau(kk)-udot(kk)

      Fx(18*(i-1)+15+kk)=Fx(18*(i-1)+15+kk)
     $+Omega(3*(i-1)+kk)-CabCTwa(kk)-Cbathetadot(kk)

70    continue

30    continue

C     Add ForceS and MomentS AT EACH NODES
      do ii=0,NES
          do kk=1,3
          Fx(18*ii+kk)=Fx(18*ii+kk)-Fa(3*ii+kk,N_BLADE)
          Fx(18*ii+kk+3)=Fx(18*ii+kk+3)-Ma(3*ii+kk,N_BLADE)
          end do
          end do

C     ADD ROOT BONDARY VALUE
          do ii=1,3
          Fx(ii)=Fx(ii)+F1(ii)
          Fx(ii+3)=Fx(ii+3)+M1(ii)
          Fx(ii+9)=Fx(ii+9)-theta0(ii)
          end do

          do ii=1,3
          Fx(18*NES+6+ii)=Fx(18*NES+6+ii)+uN1(ii)
          Fx(18*NES+9+ii)=Fx(18*NES+9+ii)+thetaN1(ii)
      end do

          END

C***********************************************************************
C     Jacobi- Calculate Jacobi matrix
C***********************************************************************

          Subroutine Jacobi(j,Jx,N_BLADE)
       INCLUDE 'cATRc.f'
       double precision Jx(18*NCWM+12,18*NCWM+12),
     $            Jxm(18*NCWM+12,18*NCWM)
          double precision C(3,3,NCWM)
          double precision  CTdotwhole(3,3,NCWM),Pdotwhole(3,NCWM),
     $              Hdotwhole(3,NCWM)
```

```
      double precision dPdP(3,3),dHdH(3,3),dudu(3,3),dtheta(3,3)
      double precision dCdtheta(3,3,3),dCTdtheta(3,3,3)
      double precision dm,theta_tran
      double precision theta1,theta2,theta3,theta1n,theta2n,theta3n,
   $                     theta1n1,theta2n1,theta3n1
      double precision eC(3,3),tek1(3,3),ekek(3,3),ektheta(3,3)
   $  ,thetaek(3,3)
      double precision dCTdot(3,3,3)
      double precision tek(3,3,3)
      double precision dCab(3,3),twa(3,3),twadCab(3,3),dCTCab(3,3),
   $  te1(3,3),tgama(3,3),te1tgama(3,3),tV(3,3),
   $  ethetae(3,3),eCab(3,3),CbadC(3,3),
   $  twaCTCab(3,3),CTdotCab(3,3),CTCabte1tgama(3,3),
   $  tF(3,3),tFdgamadF(3,3),etF(3,3),CTCabetF(3,3),
   $  tP(3,3),CTCabdgamadF(3,3),CTCabdgamadM(3,3),
   $  thth(3,3),ethth(3,3),eththCab(3,3),ttheta(3,3),
   $  eththCabdkapadF(3,3),eththCabdkapadM(3,3),etheta(3,3),
   $  thetae(3,3),CTCabdVdH(3,3),dfhdtheta(3,3),CTCabPdot(3,3),
   $ CTCabHdot(3,3),CTCabtFdgamadF(3,3),tFdgamadM(3,3),tVtP(3,3),
   $  CTCabtVtP(3,3),dVdPtP(3,3),CTCabdVdP(3,3),
   $ CTCabtP(3,3),CTCabtPdVdH(3,3)
      double precision dCabF(3),twadCabP(3),dCTCabP(3),dCabPdot(3),
   $  te1tgamaF(3),tVP(3),MFP(3),M_FP(3),dCabM_FP(3),
   $  dCabMFP(3),dCabH(3),twadCabH(3),dCTCabH(3),dCabHdot(3),
   $  e1gama(3),dCabe1gama(3),
   $  eCabkapa(3),dCabV(3),CbadCwa(3),Cbadfhdtheta(3)
   integer i,j,ii,jj,kk,k,N_BLADE

   do 300 ii=1,3
   do 300 jj=1,3
   dPdP(ii,jj)=0.0d0
   dHdH(ii,jj)=0.0d0
   dudu(ii,jj)=0.0d0
300   dtheta(ii,jj)=0.0d0
C   INITIALIZE Jx
      do 200 ii=1,18*NCWM+12
      do 200 jj=1,18*NCWM+12
   Jx(ii,jj)=0.0
200  continue

      do 201 ii=1,18*NCWM+12
      do 201 jj=1,18*NES
   Jxm(ii,jj)=0.0
201  continue

      if(j.eq.1)then
   do 1 i=1,NES
      do 1 ii=1,3
   Pdotwhole(ii,i)=Pdotwhole1(ii,i)
   Hdotwhole(ii,i)=Hdotwhole1(ii,i)
      do 1 jj=1,3
   CTdotwhole(ii,jj,i)=CTdotwhole1(ii,jj,i)
1  continue

   do 3 ii=1,3
   dPdP(ii,ii)=1.0/dts
```

```
      dHdH(ii,ii)=1.0/dts
      dudu(ii,ii)=1.0/dts
      dtheta(ii,ii)=1.0/dts
3     continue

      else

      do 2 i=1,NES
            do 2 ii=1,3
      Pdotwhole(ii,i)=Pdotwhole2(ii,i)
      Hdotwhole(ii,i)=Hdotwhole2(ii,i)
            do 2 jj=1,3
      CTdotwhole(ii,jj,i)=CTdotwhole2(ii,jj,i)
2     continue

      do 4 ii=1,3
      dPdP(ii,ii)=3.0/2.0/dts
      dHdH(ii,ii)=3.0/2.0/dts
      dudu(ii,ii)=3.0/2.0/dts
      dtheta(ii,ii)=3.0/2.0/dts
4     continue

            end if

      do 10 i=1,NES

            theta_tran=theta(3*(i-1)+1)**2+theta(3*(i-1)+2)**2+
     $      theta(3*(i-1)+3)**2
            dm=1+theta_tran/4.0

      do 20 ii=1,3

            do 30 jj=1,3
            do 30 kk=1,3
30          C(jj,kk,i)=CTR(kk,jj,i)

            call m_m(eC,e,C(1,1,i))
      call cross(tek1,e(1,ii))
            call vv(ektheta,e(1,ii),theta(3*(i-1)+1))
            call vv(thetaek,theta(3*(i-1)+1),e(1,ii))
            call m_m(ekek,ektheta,thetaek)
       do 60 jj=1,3
            do 60 kk=1,3
            dCdtheta(jj,kk,ii)=1.0/dm*(-1*theta(3*(i-1)+ii)/2*eC(jj,kk)
     $       -tek1(jj,kk)+0.5*ekek(jj,kk))
             dCTdtheta(kk,jj,ii)=dCdtheta(jj,kk,ii)
60    continue
20    continue

      theta1=theta(3*(i-1)+1)
      theta2=theta(3*(i-1)+2)
      theta3=theta(3*(i-1)+3)
      theta1n=thetan(3*(i-1)+1)
      theta2n=thetan(3*(i-1)+2)
      theta3n=thetan(3*(i-1)+3)
      theta1n1=thetan_1(3*(i-1)+1)
```

```
      theta2n1=thetan_1(3*(i-1)+2)
      theta3n1=thetan_1(3*(i-1)+3)

          call dif(j,dfhdtheta,dCTdot,theta1,theta2,theta3,
     $   theta1n,theta2n,theta3n,theta1n1,
     $   theta2n1,theta3n1)
                 call cross(tek(1,1,1),e(1,1))
                 call cross(tek(1,1,2),e(1,2))
                 call cross(tek(1,1,3),e(1,3))

           do 100  k=1,3
       call mm(dCab,dCTdtheta(1,1,k),Cab(1,1,i,N_BLADE))
          call mv(dCabF,dCab,F(3*(i-1)+1))
          call cross(twa,wa(3*(i-1)+1,N_BLADE))
          call mm(twadCab,twa,dCab)
          call mv(twadCabP,twadCab,PM(3*(i-1)+1))
          call mm(dCTCab,dCTdot(1,1,k),Cab(1,1,i,N_BLADE))
          call mv(dCTCabP,dCTCab,PM(3*(i-1)+1))
          call mv(dCabPdot,dCab,Pdotwhole(1,i))
          call cross(te1,e(1,1))
          call cross(tgama,gama(3*(i-1)+1))
          call m_m(te1tgama,te1,tgama)
          call mv(te1tgamaF,te1tgama,F(3*(i-1)+1))
          call cross(tV,V(3*(i-1)+1))
          call mv(tVP,tV,PM(3*(i-1)+1))

          do 70 ii=1,3
           MFP(ii)=M(3*(i-1)+ii)+DL(I)/2*(te1tgamaF(ii)-tVP(ii))
           M_FP(ii)=M(3*(i-1)+ii)-DL(I)/2*(te1tgamaF(ii)-tVP(ii))
70     Continue
      call mv(dCabM_FP,dCab,M_FP)
      call mv(dCabMFP,dCab,MFP)
          call mv(dCabH,dCab,H(3*(i-1)+1))
      call mv(twadCabH,twa,dCabH)
          call mv(dCTCabH,dCTCab,H(3*(i-1)+1))
          call mv(dCabHdot,dCab,Hdotwhole(1,i))
          call v_v(e1gama,e(1,1),gama(3*(i-1)+1))
          call mv(dCabe1gama,dCab,e1gama)
          call vv(etheta,e(1,k),theta(3*(i-1)+1))
          call vv(thetae,theta(3*(i-1)+1),e(1,k))

          do 80 ii=1,3
          do 80 jj=1,3
80     ethetae(ii,jj)=tek(ii,jj,k)/2+(etheta(ii,jj)+thetae(ii,jj))/4

      call mm(eCab,ethetae,Cab(1,1,i,N_BLADE))
          call mv(eCabkapa,eCab,kapa(3*(i-1)+1))
          call mv(dCabV,dCab,V(3*(i-1)+1))
          call mm(CbadC,Cba(1,1,i,N_BLADE),dCdtheta(1,1,k))
          call mv(CbadCwa,CbadC,wa(3*(i-1)+1,N_BLADE))
      call mv(Cbadfhdtheta,Cba(1,1,i,N_BLADE),dfhdtheta(1,k))

          do 101 ii=1,3
          Jxm(18*(i-1)+ii,18*(i-1)+k+3)=-1*dCabF(ii)+DL(I)/2*twadCabP(ii)
     $   +DL(I)/2*(dCTCabP(ii)+dCabPdot(ii))
```

```
 Jxm(18*(i-1)+18+ii,18*(i-1)+k+3)=dCabF(ii)+DL(I)/2*twadCabP(ii)
$   +DL(I)/2*(dCTCabP(ii)+dCabPdot(ii))

 Jxm(18*(i-1)+3+ii,18*(i-1)+k+3)=-1*dCabMFP(ii)+
$   DL(I)/2*twadCabH(ii)
$      +DL(I)/2*(dCTCabH(ii)+dCabHdot(ii))

 Jxm(18*(i-1)+21+ii,18*(i-1)+k+3)=dCabM_FP(ii)
$+DL(I)/2*twadCabH(ii)
$      +DL(I)/2*(dCTCabH(ii)+dCabHdot(ii))

      Jxm(18*(i-1)+6+ii,18*(i-1)+k+3)=-DL(I)/2*dCabe1gama(ii)

      Jxm(18*(i-1)+24+ii,18*(i-1)+k+3)=-DL(I)/2*dCabe1gama(ii)

      Jxm(18*(i-1)+9+ii,18*(i-1)+k+3)=e(ii,k)-DL(I)/2*eCabkapa(ii)

 Jxm(18*(i-1)+27+ii,18*(i-1)+k+3)=-1*e(ii,k)-DL(I)/2*eCabkapa(ii)

      Jxm(18*(i-1)+12+ii,18*(i-1)+k+3)=dCabV(ii)

 Jxm(18*(i-1)+15+ii,18*(i-1)+k+3)=-1*CbadCwa(ii)-Cbadfhdtheta(ii)

101  continue

100  continue

      call mm(twaCTCab,twa,CTCab(1,1,i))
 call cross(tP,PM(3*(i-1)+1))
      call mm(CTCabtP,CTCab(1,1,i),tP)
      call mm(CTCabtPdVdH,CTCabtP,dVdH(1,1,I))
      call mm(CTdotCab,CTdotwhole(1,1,i),Cab(1,1,i,N_BLADE))
      call mm(CTCabPdot,CTCab(1,1,i),dPdP(1,1))
      call mm(CTCabHdot,CTCab(1,1,i),dHdH(1,1))
      call mm(CTCabte1tgama,CTCab(1,1,i),te1tgama)
      call cross(tF,F(3*(i-1)+1))
      call mm(tFdgamadF,tF,dgamadF(1,1,I))
      call mm(CTCabtFdgamadF,CTCab(1,1,i),tFdgamadF)
      call mm(tFdgamadM,tF,dgamadM(1,1,I))

      do 110 ii=1,3
      do 110 jj=1,3
110   etF(ii,jj)=e(ii,jj)-DL(I)/2.0*tFdgamadM(ii,jj)

      call mm(CTCabetF,CTCab(1,1,i),etF)
 call mm(dVdPtP,dVdP(1,1,I),tP)

      do 111 ii=1,3
      do 111 jj=1,3
111   tVtP(ii,jj)=tV(ii,jj)-dVdPtP(ii,jj)

 call mm(CTCabtVtP,CTCab(1,1,i),tVtP)
      call mm(CTCabdgamadF,CTCab(1,1,i),dgamadF(1,1,I))
      call mm(CTCabdgamadM,CTCab(1,1,i),dgamadM(1,1,I))
      call vv(thth,theta(3*(i-1)+1),
$     theta(3*(i-1)+1))
```

```
      call cross(ttheta,theta(3*(i-1)+1))

            do 120 ii=1,3
            do 120 jj=1,3
120         ethth(ii,jj)=e(ii,jj)+ttheta(ii,jj)/2+thth(ii,jj)/4

      call mm(eththCab,ethth,Cab(1,1,i,N_BLADE))
            call mm(eththCabdkapadF,eththCab,dkapadF(1,1,I))
            call mm(eththCabdkapadM,eththCab,dkapadM(1,1,I))
            call mm(CTCabdVdH,CTCab(1,1,i),dVdH(1,1,I))
      call mm(CTCabdVdP,CTCab(1,1,i),dVdP(1,1,I))
C     CALCULATE ELEMENT JACOBI MATRIX
      do 102 ii=1,3
            do 102 jj=1,3
            Jxm(18*(i-1)+ii,18*(i-1)+6+jj)=-1*CTCab(ii,jj,i)

            Jxm(18*(i-1)+18+ii,18*(i-1)+6+jj)=CTCab(ii,jj,i)

      Jxm(18*(i-1)+ii,18*(i-1)+12+jj)=DL(I)/2*twaCTCab(ii,jj)
     $  +DL(I)/2*(CTdotCab(ii,jj)+CTCabPdot(ii,jj))

      Jxm(18*(i-1)+18+ii,18*(i-1)+12+jj)=DL(I)/2*twaCTCab(ii,jj)
     $        +DL(I)/2*(CTdotCab(ii,jj)+CTCabPdot(ii,jj))

      Jxm(18*(i-1)+3+ii,18*(i-1)+6+jj)=-DL(I)/2*(CTCabte1tgama(ii,jj)

     $      -CTCabtFdgamadF(ii,jj))

      Jxm(18*(i-1)+21+ii,18*(i-1)+6+jj)=-DL(I)/2*(CTCabte1tgama(ii,jj)

     $      -CTCabtFdgamadF(ii,jj))

            Jxm(18*(i-1)+3+ii,18*(i-1)+9+jj)=-1*CTCabetF(ii,jj)

      Jxm(18*(i-1)+21+ii,18*(i-1)+9+jj)=CTCabetF(ii,jj)

      Jxm(18*(i-1)+3+ii,18*(i-1)+12+jj)=DL(I)/2*CTCabtVtP(ii,jj)

            Jxm(18*(i-1)+21+ii,18*(i-1)+12+jj)=DL(I)/2*CTCabtVtP(ii,jj)

      Jxm(18*(i-1)+3+ii,18*(i-1)+15+jj)= DL(I)/2*twaCTCab(ii,jj)
     $  -dl(I)/2*CTCabtPdVdH(ii,jj)
     $        +DL(I)/2*(CTdotCab(ii,jj)+CTCabHdot(ii,jj))

            Jxm(18*(i-1)+21+ii,18*(i-1)+15+jj)=DL(I)/2*twaCTCab(ii,jj)
     $  -dl(I)/2*CTCabtPdVdH(ii,jj)
     $        +DL(I)/2*(CTdotCab(ii,jj)+CTCabHdot(ii,jj))

            Jxm(18*(i-1)+6+ii,18*(i-1)+jj)=e(ii,jj)

            Jxm(18*(i-1)+24+ii,18*(i-1)+jj)=-1*e(ii,jj)

            Jxm(18*(i-1)+6+ii,18*(i-1)+6+jj)=-DL(I)/2.0*CTCabdgamadF(ii,jj)

      Jxm(18*(i-1)+24+ii,18*(i-1)+6+jj)=-DL(I)/2.0*CTCabdgamadF(ii,jj)
```

```
      Jxm(18*(i-1)+6+ii,18*(i-1)+9+jj)=-DL(I)/2.0*CTCabdgamadM(ii,jj)

      Jxm(18*(i-1)+24+ii,18*(i-1)+9+jj)=-DL(I)/2.0*CTCabdgamadM(ii,jj)

      Jxm(18*(i-1)+9+ii,18*(i-1)+6+jj)=-DL(I)/2.0
     $   *eththCabdkapadF(ii,jj)

      Jxm(18*(i-1)+27+ii,18*(i-1)+6+jj)=
     $     -DL(I)/2.0*eththCabdkapadF(ii,jj)

      Jxm(18*(i-1)+9+ii,18*(i-1)+9+jj)=-DL(I)/2.0
     $     *eththCabdkapadM(ii,jj)

      Jxm(18*(i-1)+27+ii,18*(i-1)+9+jj)=
     $       -DL(I)/2.0*eththCabdkapadM(ii,jj)

      Jxm(18*(i-1)+12+ii,18*(i-1)+jj)=-1*twa(ii,jj)-dudu(ii,jj)

            Jxm(18*(i-1)+12+ii,18*(i-1)+12+jj)=CTCabdVdP(ii,jj)

      Jxm(18*(i-1)+12+ii,18*(i-1)+15+jj)=CTCabdVdH(ii,jj)

            Jxm(18*(i-1)+15+ii,18*(i-1)+12+jj)=dOmegadP(ii,jj,I)

            Jxm(18*(i-1)+15+ii,18*(i-1)+15+jj)=dOmegadH(ii,jj,I)

102   Continue

10    continue

C     ASSEMBLE ELEMENT JACOBI MATRICES TO THE WHOLE JACOBI MATRIX
            do ii=1,6
       Jx(ii+18*NES+6,ii+18*NES+6)=1.0d0
            end do

      do ii=1,3
            Jx(ii,ii)=1.0d0
      end do

            if (BC.eq.1)then   !HINGELESS
                  Jx(4,4)=1.0d0
                  Jx(11,5)=-1.0d0
                  Jx(12,6)=-1.0d0
            else                          !ARTICULATED
                  do ii=1,3
                  Jx(ii+3,ii+3)=1.0d0
                  end do
            end if


      do 104 ii=1,18*NES+12
            do 104 jj=7,18*NES+6
104         Jx(ii,jj)=Jxm(ii,jj-6)
      end
```

```
C*********************************************************************
C     lubksb2-Back substitution FROM FORTRAN RECIPE
C*********************************************************************

      SUBROUTINE lubksb2(a,NSOLVE,NWHOLE,inin,b)
      implicit none
      INTEGER NSOLVE,inin(NSOLVE),NWHOLE
      double precision a(NWHOLE,NWHOLE),b(NSOLVE)
      INTEGER i,ii,j,ll
      double precision sum

      ii=0
      do i=1,NSOLVE
      ll=inin(i)
      sum=b(ll)
      b(ll)=b(i)

      if (ii.ne.0)then
      do j=ii,i-1
      sum=sum-a(i,j)*b(j)
      enddo
      else if (sum.ne.0.) then
      ii=i
      endif
      b(i)=sum
      enddo
      do i=NSOLVE,1,-1
      sum=b(i)
      do j=i+1,NSOLVE
      sum=sum-a(i,j)*b(j)
      enddo
            b(i)=sum/a(i,i)
      enddo
           return
      END

C*********************************************************************
C     ludcmp2-LU decomposition
C*********************************************************************

            SUBROUTINE ludcmp2(a,NSOLVE,NWHOLE,inin,d)
      implicit none
      INTEGER NSOLVE,inin(NSOLVE),NMAX,NWHOLE
      double precision d,a(NWHOLE,NWHOLE),TINY
      PARAMETER (NMAX=500,TINY=1.0e-20)
      INTEGER i,imax,j,k
      double precision aamax,dum,sum,vv(NMAX)

      do  i=1,NSOLVE
      aamax=0.
      do j=1,NSOLVE

      if (abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
      enddo
      if (aamax.eq.0.) pause 'singular matrix in ludcmp'
      vv(i)=1./aamax
```

```
      enddo
      do  j=1,NSOLVE
      do  i=1,j-1
      sum=a(i,j)
      do  k=1,i-1
      sum=sum-a(i,k)*a(k,j)
      enddo
      a(i,j)=sum
      enddo

      aamax=0.

      do  i=j,NSOLVE
      sum=a(i,j)
      do  k=1,j-1
      sum=sum-a(i,k)*a(k,j)
      enddo
      a(i,j)=sum
      dum=vv(i)*abs(sum)
      if (dum.ge.aamax) then
      imax=i
      aamax=dum
      endif
      enddo

      if (j.ne.imax)then
      do k=1,NSOLVE
      dum=a(imax,k)
      a(imax,k)=a(j,k)
      a(j,k)=dum
      enddo
      d=-d
      vv(imax)=vv(j)
      endif
      inin(j)=imax
      if(a(j,j).eq.0.)a(j,j)=TINY
      if(j.ne.NSOLVE)then
      dum=1./a(j,j)
      do  i=j+1,NSOLVE
      a(i,j)=a(i,j)*dum
      enddo
      endif
      enddo
      return
      END

C**********************************************************************
C     m_m---Plus of two matrices
C**********************************************************************

            Subroutine m_m(A,B,C)
      implicit none
      integer i,j
      double precision A(3,3),B(3,3),C(3,3)
```

```
        do 10 i=1,3
        do 10 j=1,3
        A(i,j)=B(i,j)+C(i,j)
10   continue
     end

C*******************************************************************
C     v_v---Plus of two vectors
C*******************************************************************

        Subroutine v_v(A,B,C)
     implicit none
     integer i
     double precision A(3),B(3),C(3)

        do 10 i=1,3
        A(i)=B(i)+C(i)
10   continue
     end


C*******************************************************************
C     mm---Multiply of two matrices
C*******************************************************************

        Subroutine mm(A,B,C)
     implicit none
     integer i,j,k
     double precision A(3,3),B(3,3),C(3,3)

        do 20 i=1,3
        do 20 j=1,3
              A(i,j)=0.0
        do 10 k=1,3
        A(i,j)=A(i,j)+B(i,k)*C(k,j)
10   continue
20   continue
     end


C*******************************************************************
C     mv---Multiply of matrix and vector
C*******************************************************************

        Subroutine mv(A,B,C)
     implicit none
     integer i,j
     double precision A(3),B(3,3),C(3)
        do 20 i=1,3
     A(i)=0.0
        do 20 j=1,3
         A(i)=A(i)+B(i,j)*C(j)
20       continue
     end

C*******************************************************************
```

```
C     vv---Multiply of two vectors
C*******************************************************************

          Subroutine vv(A,B,C)
      implicit none
      integer i,j
      double precision A(3,3),B(3),C(3)

          do 10 i=1,3
          do 10 j=1,3
10        A(i,j)=B(i)*C(j)
      end


C*******************************************************************
C    cross---Convert a column vector to its dual matrix
C*******************************************************************

          subroutine cross(theta3t,theta3)
      implicit none
      integer i,j
          double precision theta3t(3,3),theta3(3)

          do 10 i=1,3
          do 10 j=1,3
10     theta3t(i,j)=0.0d0
       theta3t(1,2)=-theta3(3)
       theta3t(1,3)=theta3(2)
       theta3t(2,1)=theta3(3)
       theta3t(2,3)=-theta3(1)
       theta3t(3,1)=-theta3(2)
       theta3t(3,2)=theta3(1)
          end


C*******************************************************************
C     CTd---Calculate dCT/dt,dP/dt,dH/dt,du/dt,dtheta/dt using
C           finite difference method
C*******************************************************************

          Subroutine CTd(dm,j,i)
      INCLUDE 'cATRc.f'
      double precision theta_tran,dm,ddm
          integer i,j,ii,jj,kk

          do 10 ii=1,3
          do 10 jj=1,3
 10     CTdot(ii,jj)=0.0d0
          theta_tran=theta(3*(i-1)+1)**2+theta(3*(i-1)+2)**2+
     $    theta(3*(i-1)+3)**2
          dm=1.0d0+theta_tran/4.0d0
          if (j.eq.1)then
          ddm=0.5/dts*(theta(3*(i-1)+1)*(theta(3*(i-1)+1)
     $-thetan(3*(i-1)+1))
     $       +theta(3*(i-1)+2)*(theta(3*(i-1)+2)-thetan(3*(i-1)+2))
     $   +theta(3*(i-1)+3)*(theta(3*(i-1)+3)-thetan(3*(i-1)+3)))
```

```
 CTdot(1,1)=(-ddm+theta(3*(i-1)+1)*(theta(3*(i-1)+1)
$        -thetan(3*(i-1)+1))/dts)*dm-CTR(1,1,i)*ddm

 CTdot(2,2)=(-ddm+theta(3*(i-1)+2)*(theta(3*(i-1)+2)
$        -thetan(3*(i-1)+2))/dts)*dm-CTR(2,2,i)*ddm

 CTdot(3,3)=(-ddm+theta(3*(i-1)+3)*(theta(3*(i-1)+3)
$        -thetan(3*(i-1)+3))/dts)*dm-CTR(3,3,i)*ddm

 CTdot(1,2)=(-(theta(3*(i-1)+3)-thetan(3*(i-1)+3))
$      +0.5d0*theta(3*(i-1)+2)*(theta(3*(i-1)+1)-thetan(3*(i-1)+1))
$ +0.5*theta(3*(i-1)+1)*(theta(3*(i-1)+2)
$ -thetan(3*(i-1)+2)))/dts*dm
$ -CTR(1,2,i)*ddm

 CTdot(1,3)=((theta(3*(i-1)+2)-thetan(3*(i-1)+2))
$     +0.5*theta(3*(i-1)+3)*(theta(3*(i-1)+1)-thetan(3*(i-1)+1))
$ +0.5*theta(3*(i-1)+1)*(theta(3*(i-1)+3)
$ -thetan(3*(i-1)+3)))/dts*dm
$ -CTR(1,3,i)*ddm

 CTdot(2,1)=((theta(3*(i-1)+3)-thetan(3*(i-1)+3))
$  +0.5*theta(3*(i-1)+2)*(theta(3*(i-1)+1)-thetan(3*(i-1)+1))
$ +0.5*theta(3*(i-1)+1)*(theta(3*(i-1)+2)
$-thetan(3*(i-1)+2)))/dts*dm
$ -CTR(2,1,i)*ddm

 CTdot(2,3)=(-(theta(3*(i-1)+1)-thetan(3*(i-1)+1))
$  +0.5*theta(3*(i-1)+3)*(theta(3*(i-1)+2)-thetan(3*(i-1)+2))
$ +0.5*theta(3*(i-1)+2)*(theta(3*(i-1)+3)
$-thetan(3*(i-1)+3)))/dts*dm
$ -CTR(2,3,i)*ddm

 CTdot(3,1)=(-(theta(3*(i-1)+2)-thetan(3*(i-1)+2))
$ +0.5*theta(3*(i-1)+3)*(theta(3*(i-1)+1)-thetan(3*(i-1)+1))
$ +0.5*theta(3*(i-1)+1)*(theta(3*(i-1)+3)
$-thetan(3*(i-1)+3)))/dts*dm
$ -CTR(3,1,i)*ddm

 CTdot(3,2)=((theta(3*(i-1)+1)-thetan(3*(i-1)+1))
$ +0.5*theta(3*(i-1)+3)*(theta(3*(i-1)+2)-thetan(3*(i-1)+2))
$ +0.5*theta(3*(i-1)+2)*(theta(3*(i-1)+3)
$ -thetan(3*(i-1)+3)))/dts*dm
$ -CTR(3,2,i)*ddm

 do 31 ii=1,3
       do 31 jj=1,3
       CTdot(ii,jj)=CTdot(ii,jj)/dm**2
       CTdotwhole1(ii,jj,i)=CTdot(ii,jj)
31   continue

       else

       ddm=0.5/2/dts*(theta(3*(i-1)+1)*(3*theta(3*(i-1)+1)
$   -4*thetan(3*(i-1)+1)+thetan_1(3*(i-1)+1))+theta(3*(i-1)+2)
```

```
$    *(3*theta(3*(i-1)+2)-4*thetan(3*(i-1)+2)+thetan_1(3*(i-1)+2))
$    +theta(3*(i-1)+3)*(3*theta(3*(i-1)+3)-4*thetan(3*(i-1)+3)
$    +thetan_1(3*(i-1)+3)))

 CTdot(1,1)=(-ddm+theta(3*(i-1)+1)*(3*theta(3*(i-1)+1)

$  -4*thetan(3*(i-1)+1)+thetan_1(3*(i-1)+1))/2/dts)
$ *dm-CTR(1,1,i)*ddm

 CTdot(2,2)=(-ddm+theta(3*(i-1)+2)*(3*theta(3*(i-1)+2)

$  -4*thetan(3*(i-1)+2)+thetan_1(3*(i-1)+2))/2/dts)*dm
$-CTR(2,2,i)*ddm

 CTdot(3,3)=(-ddm+theta(3*(i-1)+3)*(3*theta(3*(i-1)+3)

$  -4*thetan(3*(i-1)+3)+thetan_1(3*(i-1)+3))/2/dts)
$  *dm-CTR(3,3,i)*ddm

 CTdot(1,2)=(-(3*theta(3*(i-1)+3)-4*thetan(3*(i-1)+3)

$+thetan_1(3*(i-1)+3))+0.5*theta(3*(i-1)+2)*(3*theta(3*(i-1)+1)

$-4*thetan(3*(i-1)+1)+thetan_1(3*(i-1)+1))+0.5*theta(3*(i-1)+1)

$*(3*theta(3*(i-1)+2)-4*thetan(3*(i-1)+2)+thetan_1(3*(i-1)+2)))

$       /2/dts*dm-CTR(1,2,i)*ddm

 CTdot(1,3)=((3*theta(3*(i-1)+2)-4*thetan(3*(i-1)+2)

$+thetan_1(3*(i-1)+2))+0.5*theta(3*(i-1)+3)*(3*theta(3*(i-1)+1)

$-4*thetan(3*(i-1)+1)+thetan_1(3*(i-1)+1))+0.5*theta(3*(i-1)+1)

$*(3*theta(3*(i-1)+3)-4*thetan(3*(i-1)+3)+thetan_1(3*(i-1)+3)))

$       /2/dts*dm-CTR(1,3,i)*ddm

 CTdot(2,1)=((3*theta(3*(i-1)+3)-4*thetan(3*(i-1)+3)

$+thetan_1(3*(i-1)+3))+0.5*theta(3*(i-1)+2)*(3*theta(3*(i-1)+1)

$-4*thetan(3*(i-1)+1)+thetan_1(3*(i-1)+1))+0.5*theta(3*(i-1)+1)

$*(3*theta(3*(i-1)+2)-4*thetan(3*(i-1)+2)+thetan_1(3*(i-1)+2)))

$       /2/dts*dm-CTR(2,1,i)*ddm

 CTdot(2,3)=(-(3*theta(3*(i-1)+1)-4*thetan(3*(i-1)+1)

$+thetan_1(3*(i-1)+1))+0.5*theta(3*(i-1)+3)*(3*theta(3*(i-1)+2)

$  -4*thetan(3*(i-1)+2)+thetan_1(3*(i-1)+2))+0.5*theta(3*(i-1)+2)

$  *(3*theta(3*(i-1)+3)-4*thetan(3*(i-1)+3)+thetan_1(3*(i-1)+3)))
```

```
$  /2/dts*dm-CTR(2,3,i)*ddm

 CTdot(3,1)=(-(3*theta(3*(i-1)+2)-4*thetan(3*(i-1)+2)
$+thetan_1(3*(i-1)+2))+0.5*theta(3*(i-1)+3)*(3*theta(3*(i-1)+1)
$-4*thetan(3*(i-1)+1)+thetan_1(3*(i-1)+1))+0.5*theta(3*(i-1)+1)
$*(3*theta(3*(i-1)+3)-4*thetan(3*(i-1)+3)+thetan_1(3*(i-1)+3)))
$       /2/dts*dm-CTR(3,1,i)*ddm

 CTdot(3,2)=((3*theta(3*(i-1)+1)-4*thetan(3*(i-1)+1)
$+thetan_1(3*(i-1)+1))+0.5*theta(3*(i-1)+3)*(3*theta(3*(i-1)+2)
$-4*thetan(3*(i-1)+2)+thetan_1(3*(i-1)+2))+0.5*theta(3*(i-1)+2)
$*(3*theta(3*(i-1)+3)-4*thetan(3*(i-1)+3)+thetan_1(3*(i-1)+3)))
$       /2/dts*dm-CTR(3,2,i)*ddm

        do 32 ii=1,3
        do 32 jj=1,3
        CTdot(ii,jj)=CTdot(ii,jj)/dm**2
        CTdotwhole2(ii,jj,i)=CTdot(ii,jj)
32   continue

        end if

        do 50 kk=1,3

        if (j.eq.1)then
        Pdot(kk)=(PM(3*(i-1)+kk)-Pn(3*(i-1)+kk))/dts
 Hdot(kk)=(H(3*(i-1)+kk)-Hn(3*(i-1)+kk))/dts
 udot(kk)=(u(3*(i-1)+kk)-un(3*(i-1)+kk))/dts
 thetadot(kk)=(theta(3*(i-1)+kk)-thetan(3*(i-1)+kk))/dts
        Pdotwhole1(kk,i)=Pdot(kk)
        Hdotwhole1(kk,i)=Hdot(kk)
        else
 Pdot(kk)=(3*PM(3*(i-1)+kk)-4*Pn(3*(i-1)+kk)+Pn1(3*(i-1)+kk))/2/dts
 Hdot(kk)=(3*H(3*(i-1)+kk)-4*Hn(3*(i-1)+kk)+Hn1(3*(i-1)+kk))/2/dts
 udot(kk)=(3*u(3*(i-1)+kk)-4*un(3*(i-1)+kk)
$  +un_1(3*(i-1)+kk))/2/dts
 thetadot(kk)=(3*theta(3*(i-1)+kk)-4*thetan(3*(i-1)+kk)
$            +thetan_1(3*(i-1)+kk))/2/dts
 Pdotwhole2(kk,i)=Pdot(kk)
 Hdotwhole2(kk,i)=Hdot(kk)
        end if

50   continue

        end

C*******************************************************************
```

```
C     ctlyb---Calculate transpose of C matrix from theta
C*******************************************************************

        Subroutine ctlyb(CT1,theta3)
        implicit none
        double precision CT1(3,3),theta3(3),dm
    integer i,j
        do 20 i=1,3
        do 20 j=1,3
20   CT1(i,j)=0.0d0
    dm=1+(theta3(1)**2+theta3(2)**2+theta3(3)**2)*0.25d0
    CT1(1,1)=(2-dm)+0.5*theta3(1)**2
    CT1(2,2)=(2-dm)+0.5*theta3(2)**2
    CT1(3,3)=(2-dm)+0.5*theta3(3)**2
    CT1(1,2)=-theta3(3)+0.5*theta3(2)*theta3(1)
    CT1(1,3)=theta3(2)+0.5*theta3(3)*theta3(1)
    CT1(2,1)=theta3(3)+0.5*theta3(2)*theta3(1)
    CT1(2,3)=-theta3(1)+0.5*theta3(2)*theta3(3)
    CT1(3,1)=-theta3(2)+0.5*theta3(3)*theta3(1)
    CT1(3,2)=theta3(1)+0.5*theta3(2)*theta3(3)
    do 10 i=1,3
          do 10 j=1,3
           CT1(i,j)=CT1(i,j)/dm
10   continue
    end

C*******************************************************************
C     dif---Calculate dCTdot/dtheta
C*******************************************************************

        Subroutine dif(j,dfhdtheta,dCTdot,theta1,theta2,theta3,
   $   theta1n,theta2n,theta3n,theta1n1,
   $   theta2n1,theta3n1)
        INCLUDE 'cATRc.f'

        double precision theta1,theta2,theta3,
   $   theta1n,theta2n,theta3n,theta1n1,
   $   theta2n1,theta3n1
        double precision dCTdot(3,3,3),dfhdtheta(3,3)
        integer j
        if (j.eq.1)then
         dfhdtheta(1,1)=1.0/2.0*theta1*(theta1-theta1n)/dts
   $       +(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
   $+1.0/4.0*theta3**2)/dts+1.0/4.0
   $       *theta1*theta3*(theta2-theta2n)/dts-1.0/4.0*theta1*theta2
   $       *(theta3-theta3n)/dts
    dfhdtheta(1,2)=1.0/2.0*theta2*(theta1-theta1n)/dts
   $       +1.0/4.0*theta2*theta3*(theta2-theta2n)/dts+1.0/2.0*(1
   $       +1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0
   $*theta3**2)*theta3/dts
   $       -1.0/4.0*theta2**2*(theta3-theta3n)/dts-1.0/2.0*(1+1.0/4.0
   $       *theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
   $*(theta3-theta3n)/dts
         dfhdtheta(1,3)=1.0/2.0*theta3*(theta1-theta1n)/dts
   $       +1.0/4.0*theta3**2*(theta2-theta2n)/dts+1.0/2.0
   $*(1+1.0/4.0*theta1**2
```

```
$      +1.0/4.0*theta2**2+1.0/4.0*theta3**2)*(theta2-theta2n)/dts
$-1.0/4.0
$      *theta3*theta2*(theta3-theta3n)/dts-1.0/2.0*(1
$+1.0/4.0*theta1**2
$      +1.0/4.0*theta2**2+1.0/4.0*theta3**2)*theta2/dts
 dfhdtheta(2,1)=-1.0/4.0*theta1*theta3*(theta1-theta1n)/dts
$      -1.0/2.0*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0
$*theta3**2)
$      *theta3/dts+1.0/2.0*theta1*(theta2-theta2n)/dts+1.0/4.0
$      *theta1**2*(theta3-theta3n)/dts+1.0/2.0*(1+1.0/4.0*theta1**2
$      +1.0/4.0*theta2**2+1.0/4.0*theta3**2)*(theta3-theta3n)/dts
 dfhdtheta(2,2)=-1.0/4.0*theta2*theta3*(theta1-theta1n)
$      /dts+1.0/2.0*theta2*(theta2-theta2n)/dts+(1+1.0/4.0*theta1**2
$      +1.0/4.0*theta2**2+1.0/4.0*theta3**2)/dts
$+1.0/4.0*theta1*theta2*(theta3-theta3n)/dts
 dfhdtheta(2,3)=-1.0/4.0*theta3**2*(theta1-theta1n)
$      /dts-1.0/2.0*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$      *(theta1-theta1n)/dts+1.0/2.0*theta3*(theta2-theta2n)/dts
$ +1.0/4.0*theta3*theta1*(theta3-theta3n)/dts+1.0/2.0*(1+1.0/4.0
$      *theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)*theta1/dts
 dfhdtheta(3,1)=1.0/4.0*theta1*theta2*(theta1-theta1n)
$      /dts+1.0/2.0*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$      *theta2/dts-1.0/4.0*theta1**2*(theta2-theta2n)/dts-1.0/2.0
$      *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$      *(theta2-theta2n)/dts+1.0/2.0*theta1*(theta3-theta3n)/dts
 dfhdtheta(3,2)=1.0/4.0*theta2**2*(theta1-theta1n)/dts
$      +1.0/2.0*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$      *(theta1-theta1n)/dts-1.0/4.0*theta2*theta1*(theta2
$      -theta2n)/dts-1.0/2.0*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$      +1.0/4.0*theta3**2)*theta1/dts+1.0/2.0*theta2
$*(theta3-theta3n)/dts
 dfhdtheta(3,3)=1.0/4.0*theta2*theta3*(theta1-theta1n)
$      /dts-1.0/4.0*theta1*theta3*(theta2-theta2n)/dts+1.0/2.0
$      *theta3*(theta3-theta3n)/dts+(1+1.0/4.0*theta1**2+1.0/4.0
$      *theta2**2+1.0/4.0*theta3**2)/dts

 dCTdot(1,1,1)=((-(theta1-1.0/2.0*theta1n)/dts
$ +(theta1-theta1n)/dts
$      +theta1/dts)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$ +1.0/2.0*(-(1.0/2.0*theta1*(theta1-theta1n)
$+1.0/2.0*theta2*(theta2-theta2n)
$      +1.0/2.0*theta3*(theta3-theta3n))/dts+theta1
$*(theta1-theta1n)/dts)
$ *theta1-1.0/2.0*theta1*(1.0/2.0*theta1*(theta1-theta1n)
$      +1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0
$*theta3*(theta3-theta3n))/dts
$  -(1+1.0/4.0*theta1**2-1.0/4.0*theta2**2
$-1.0/4.0*theta3**2)*(theta1
$-1.0/2.0*theta1n)/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
$      -((-(1.0/2.0*theta1*(theta1-theta1n)
$+1.0/2.0*theta2*(theta2-theta2n)
```

```
$       +1.0/2.0*theta3*(theta3-theta3n))/dts
$+theta1*(theta1-theta1n)/dts)
$  *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(1+1.0/4.0*theta1**2
$       -1.0/4.0*theta2**2-1.0/4.0*theta3**2)
$*(1.0/2.0*theta1*(theta1-theta1n)
$  +1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3*(theta3-theta3n))/dts)
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta1


 dCTdot(1,1,2)=(-(theta2-1.0/2.0*theta2n)/dts*(1+1.0/4.0*theta1**2
$ +1.0/4.0*theta2**2+1.0/4.0*theta3**2)+1.0/2.0*(
$-(1.0/2.0*theta1*(theta1-theta1n)
$       +1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0
$*theta3*(theta3-theta3n))/dts
$       +theta1*(theta1-theta1n)/dts)*theta2+1.0/2.0*theta2*
$       (1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0*theta2
$*(theta2-theta2n)
$+1.0/2.0*theta3*(theta3-theta3n))/dts-(1
$+1.0/4.0*theta1**2-1.0/4.0*theta2**2
$       -1.0/4.0*theta3**2)*(theta2
$-1.0/2.0*theta2n)/dts)/(1+1.0/4.0*theta1**2
$ +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
$-((-(1.0/2.0*theta1*(theta1-theta1n)
$       +1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3*(theta3-theta3n))/dts
$       +theta1*(theta1-theta1n)/dts)
$*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$ +1.0/4.0*theta3**2)-(1+1.0/4.0*theta1**2
$-1.0/4.0*theta2**2-1.0/4.0*theta3**2)
$       *(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0
$*theta2*(theta2-theta2n)
$ +1.0/2.0*theta3*(theta3-theta3n))/dts)/(1+1.0/4.0
$*theta1**2+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)**3*theta2


 dCTdot(1,1,3)=(-(theta3-1.0/2.0*theta3n)/dts*(1+1.0/4.0*theta1**2
$  +1.0/4.0*theta2**2+1.0/4.0*theta3**2)+1.0/2.0
$*(-(1.0/2.0*theta1*(theta1-theta1n)
$       +1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0
$*theta3*(theta3-theta3n))/dts
$       +theta1*(theta1-theta1n)/dts)*theta3+1.0/2.0
$*theta3*(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3*
$(theta3-theta3n))/dts-(1+1.0/4.0*theta1**2-1.0/4.0*theta2**2
$-1.0/4.0*theta3**2)
$       *(theta3-1.0/2.0*theta3n)/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$  +1.0/4.0*theta3**2)**2-((-(1.0/2.0*theta1
$*(theta1-theta1n)+1.0/2.0*theta2
$*(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts+theta1
$*(theta1-theta1n)/dts)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
```

```
$   +1.0/4.0*theta3**2)
$       -(1+1.0/4.0*theta1**2-1.0/4.0*theta2**2
$-1.0/4.0*theta3**2)*(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3
$       *(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)**3*theta3

  dCTdot(2,1,1)=((1.0/2.0/dts*theta2+(1.0/2.0*theta2
$-1.0/2.0*theta2n)/dts)
$       *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$       +1.0/2.0*((theta3-theta3n)/dts+(1.0/2.0*theta1-1.0/2.0
$*theta1n)/dts
$       *theta2+(1.0/2.0*theta2-1.0/2.0*theta2n)/dts*theta1)*theta1
$       -1.0/2.0*theta2*(1.0/2.0*theta1*(theta1-theta1n)
$+1.0/2.0*theta2
$       *(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts
$ -(theta3+1.0/2.0*theta2*theta1)*(theta1-1.0/2.0*theta1n)/dts)
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2-(((theta3
$       -theta3n)/dts+(1.0/2.0*theta1-1.0/2.0*theta1n)/dts
$*theta2+(1.0/2.0*theta2
$       -1.0/2.0*theta2n)/dts*theta1)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)-(theta3+1.0/2.0*theta2*theta1)
$*(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3
$       *(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)**3*theta1


  dCTdot(2,1,2)=(((1.0/2.0*theta1-1.0/2.0*theta1n)/dts
$+1.0/2.0/dts*theta1)
$       *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$       +1.0/2.0*((theta3-theta3n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)/dts
$       *theta2+(1.0/2.0*theta2-1.0/2.0*theta2n)/dts*theta1)*theta2
$       -1.0/2.0*theta1*(1.0/2.0*theta1*(theta1-theta1n)
$+1.0/2.0*theta2*(theta2
$       -theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts-(theta3
$ +1.0/2.0*theta2*theta1)*(theta2-1.0/2.0*theta2n)/dts)
$/(1+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
$-(((theta3-theta3n)/dts
$+(1.0/2.0*theta1-1.0/2.0*theta1n)/dts*theta2+(1.0/2.0*theta2
$       -1.0/2.0*theta2n)/dts*theta1)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)-(theta3+1.0/2.0*theta2*theta1)
$*(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3
$       *(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
```

```
$       +1.0/4.0*theta3**2)**3*theta2


  dCTdot(2,1,3)=(1/dts*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$       +1.0/2.0*((theta3-theta3n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)/dts*theta2
$ +(1.0/2.0*theta2-1.0/2.0*theta2n)/dts*theta1)
$*theta3-(1.0/2.0*theta1*(theta1
$       -theta1n)+1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0*theta3
$       *(theta3-theta3n))/dts-(theta3+1.0/2.0*theta2*theta1)
$       *(theta3-1.0/2.0*theta3n)/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2-(((theta3-theta3n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)/dts*theta2+(1.0/2.0*theta2-1.0/2.0*theta2n)
$       /dts*theta1)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$       -(theta3+1.0/2.0*theta2*theta1)*(1.0/2.0*theta1
$*(theta1-theta1n)
$ +1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0*theta3
$*(theta3-theta3n))/dts)
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta3


   dCTdot(1,3,1)=((1.0/2.0*theta3/dts+(1.0/2.0*theta3
$-1.0/2.0*theta3n)/dts)
$       *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$       +1.0/2.0*((theta2-theta2n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)/dts
$       *theta3+(1.0/2.0*theta3-1.0/2.0*theta3n)/dts*theta1)*theta1
$       -1.0/2.0*theta3*(1.0/2.0*theta1*(theta1-theta1n)
$+1.0/2.0*theta2
$       *(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts
$-(theta2+1.0/2.0*theta3*theta1)*(theta1-1.0/2.0*theta1n)/dts)
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0
$*theta3**2)**2-(((theta2
$       -theta2n)/dts+(1.0/2.0*theta1-1.0/2.0*theta1n)/dts*theta3
$ +(1.0/2.0*theta3-1.0/2.0*theta3n)/dts*theta1)*(1
$+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)-(theta2
$+1.0/2.0*theta3*theta1)
$       *(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0
$*theta2*(theta2-theta2n)
$       +1.0/2.0*theta3*(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3*theta1

   dCTdot(1,3,2)=(1/dts*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)+1.0/2.0*((theta2-theta2n)
$/dts+(1.0/2.0*theta1
$       -1.0/2.0*theta1n)/dts*theta3+(1.0/2.0*theta3
$-1.0/2.0*theta3n)/dts*theta1)
$       *theta2-(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0
$*theta2*(theta2
$       -theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts-(theta2
$       +1.0/2.0*theta3*theta1)*(theta2-1.0/2.0*theta2n)/dts)/(1
```

```
$ +1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
$-(((theta2-theta2n)
$       /dts+(1.0/2.0*theta1-1.0/2.0*theta1n)/dts*theta3
$+(1.0/2.0*theta3
$       -1.0/2.0*theta3n)/dts*theta1)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)-(theta2+1.0/2.0*theta3*theta1)
$*(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$       +1.0/2.0*theta3*(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3*theta2

  dCTdot(1,3,3)=(((1.0/2.0*theta1-1.0/2.0*theta1n)/dts
$+1.0/2.0/dts*theta1)
$       *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$       +1.0/2.0*((theta2-theta2n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)
$       /dts*theta3+(1.0/2.0*theta3-1.0/2.0*theta3n)
$/dts*theta1)*theta3
$       -1.0/2.0*theta1*(1.0/2.0*theta1*(theta1-theta1n)
$+1.0/2.0*theta2
$       *(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts
$-(theta2+1.0/2.0*theta3*theta1)*(theta3-1.0/2.0*theta3n)/dts)
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2
$       -(((theta2-theta2n)/dts+(1.0/2.0*theta1-1.0/2.0*theta1n)/dts
$       *theta3+(1.0/2.0*theta3-1.0/2.0*theta3n)/dts
$*theta1)*(1+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$-(theta2+1.0/2.0*theta3*theta1)
$       *(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0
$*theta2*(theta2-theta2n)
$       +1.0/2.0*theta3*(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3*theta3

  dCTdot(3,1,1)=((1.0/2.0/dts*theta3+(1.0/2.0*theta3
$-1.0/2.0*theta3n)
$       /dts)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$       +1.0/2.0*((-theta2+theta2n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)/dts
$       *theta3+(1.0/2.0*theta3-1.0/2.0*theta3n)/dts*theta1)*theta1
$       -1.0/2.0*theta3*(1.0/2.0*theta1*(theta1
$-theta1n)+1.0/2.0*theta2
$       *(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts
$       -(-theta2+1.0/2.0*theta3*theta1)*
$(theta1-1.0/2.0*theta1n)/dts)
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2
$-(((-theta2+theta2n)/dts+(1.0/2.0*theta1-1.0/2.0*theta1n)/dts
$       *theta3+(1.0/2.0*theta3-1.0/2.0*theta3n)/dts*theta1)
$       *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$       -(-theta2+1.0/2.0*theta3*theta1)*
$(1.0/2.0*theta1*(theta1-theta1n)
$       +1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3*(theta3-theta3n))
```

```
$       /dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta1


 dCTdot(3,1,2)=(-1/dts*(1+1.0/4.0*theta1**2+1.0/4.0
$*theta2**2+1.0/4.0*theta3**2)
$ +1.0/2.0*((-theta2+theta2n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)/dts*theta3
$       +(1.0/2.0*theta3-1.0/2.0*theta3n)/dts*theta1)
$*theta2+(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2
$-theta2n)+1.0/2.0*theta3
$       *(theta3-theta3n))/dts-(-theta2+1.0/2.0*theta3*theta1)
$       *(theta2-1.0/2.0*theta2n)/dts)/(1+1.0/4.0
$*theta1**2+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)**2-(((-theta2+theta2n)/dts
$+(1.0/2.0*theta1
$       -1.0/2.0*theta1n)/dts*theta3+(1.0/2.0*theta3
$-1.0/2.0*theta3n)/dts*theta1)
$       *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)-
$       (-theta2+1.0/2.0*theta3*theta1)*(1.0/2.0
$*theta1*(theta1-theta1n)
$       +1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3*(theta3-theta3n))
$       /dts)/(1+1.0/4.0*theta1**2+1.0/4.0
$*theta2**2+1.0/4.0*theta3**2)**3*theta2


  dCTdot(3,1,3)=(((1.0/2.0*theta1-1.0/2.0*theta1n)/dts
$+1.0/2.0/dts*theta1)
$       *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$       +1.0/2.0*((-theta2+theta2n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)/dts
$       *theta3+(1.0/2.0*theta3-1.0/2.0*theta3n)/dts*theta1)*theta3
$       -1.0/2.0*theta1*(1.0/2.0*theta1*(theta1
$-theta1n)+1.0/2.0*theta2
$       *(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts-
$(-theta2+1.0/2.0*theta3*theta1)*(theta3-1.0/2.0*theta3n)/dts)
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2
$-(((-theta2+theta2n)/dts+(1.0/2.0*theta1-1.0/2.0*theta1n)/dts
$       *theta3+(1.0/2.0*theta3-1.0/2.0*theta3n)
$/dts*theta1)*(1+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$-(-theta2+1.0/2.0*theta3*theta1)
$       *(1.0/2.0*theta1*(theta1-theta1n)
$+1.0/2.0*theta2*(theta2-theta2n)
$       +1.0/2.0*theta3*(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3*theta3


  dCTdot(2,3,1)=(-1/dts*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)+1.0/2.0*((-theta1+theta1n)
$/dts+(1.0/2.0*theta2
$       -1.0/2.0*theta2n)/dts*theta3+(1.0/2.0*theta3
$-1.0/2.0*theta3n)/dts*theta2)
```

```
$       *theta1+(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0*theta2
$       *(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts
$       -(-theta1+1.0/2.0*theta3*theta2)*(theta1
$-1.0/2.0*theta1n)/dts)
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2-
$       (((-theta1+theta1n)/dts+(1.0/2.0*theta2-1.0/2.0*theta2n)/dts
$       *theta3+(1.0/2.0*theta3-1.0/2.0*theta3n)/dts*theta2)*(1
$       +1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(-theta1
$       +1.0/2.0*theta3*theta2)*(1.0/2.0*theta1*(theta1-theta1n)
$       +1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0*theta3*(theta3
$  -theta3n))/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3
$       *theta1
  dCTdot(2,3,2)=((1.0/2.0*theta3/dts
$+(1.0/2.0*theta3-1.0/2.0*theta3n)
$       /dts)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0*((
$       -theta1+theta1n)/dts+(1.0/2.0*theta2-1.0/2.0
$*theta2n)/dts*theta3
$       +(1.0/2.0*theta3-1.0/2.0*theta3n)/dts*theta2)
$*theta2-1.0/2.0*theta3
$       *(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0
$*theta2*(theta2-theta2n)
$       +1.0/2.0*theta3*(theta3-theta3n))/dts-
$(-theta1+1.0/2.0*theta3*theta2)
$       *(theta2-1.0/2.0*theta2n)/dts)
$/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)**2-(((-theta1+theta1n)
$/dts+(1.0/2.0*theta2
$       -1.0/2.0*theta2n)/dts*theta3+(1.0/2.0*theta3
$-1.0/2.0*theta3n)/dts
$       *theta2)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$       -(-theta1+1.0/2.0*theta3*theta2)*(1.0/2.0*theta1*(theta1
$       -theta1n)+1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0
$*theta3*(theta3
$       -theta3n))/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0
$       *theta3**2)**3*theta2

  dCTdot(2,3,3)=((1.0/2.0/dts*theta2+(1.0/2.0*theta2
$-1.0/2.0*theta2n)
$       /dts)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$       +1.0/2.0*((-theta1+theta1n)/dts+(1.0/2.0*theta2
$-1.0/2.0*theta2n)
$       /dts*theta3+(1.0/2.0*theta3-1.0/2.0*theta3n)/dts
$*theta2)*theta3
$       -1.0/2.0*theta2*(1.0/2.0*theta1*(theta1-theta1n)
$+1.0/2.0*theta2
$       *(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts
$       -(-theta1+1.0/2.0*theta3*theta2)*(theta3
$-1.0/2.0*theta3n)/dts)
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
```

```
$+1.0/4.0*theta3**2)**2-(((-theta1
$       +theta1n)/dts+(1.0/2.0*theta2-1.0/2.0*theta2n)
$/dts*theta3+(1.0/2.0*theta3
$       -1.0/2.0*theta3n)/dts*theta2)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)-(-theta1+1.0/2.0*theta3*theta2)
$*(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3
$       *(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2+1.0/4.0
$*theta2**2+1.0/4.0
$       *theta3**2)**3*theta3

  dCTdot(3,2,1)=(1/dts*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$       +1.0/2.0*((theta1-theta1n)/dts+(1.0/2.0*theta2
$-1.0/2.0*theta2n)/dts*theta3
$       +(1.0/2.0*theta3-1.0/2.0*theta3n)/dts*theta2)
$*theta1-(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3
$*(theta3-theta3n))/dts-(theta1+1.0/2.0*theta3*theta2)*(theta1
$ -1.0/2.0*theta1n)/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$       **2-(((theta1-theta1n)/dts+(1.0/2.0*theta2
$-1.0/2.0*theta2n)/dts*theta3
$       +(1.0/2.0*theta3-1.0/2.0*theta3n)/dts*theta2)
$*(1+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$-(theta1+1.0/2.0*theta3*theta2)
$       *(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0
$*theta2*(theta2-theta2n)
$       +1.0/2.0*theta3*(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2
$           +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3*theta1


   dCTdot(3,2,2)=((1.0/2.0/dts*theta3+(1.0/2.0*theta3
$-1.0/2.0*theta3n)/dts)
$       *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0
$*theta3**2)+1.0/2.0*((theta1
$       -theta1n)/dts+(1.0/2.0*theta2-1.0/2.0*theta2n)
$/dts*theta3+(1.0/2.0*theta3
$       -1.0/2.0*theta3n)/dts*theta2)*theta2-1.0/2.0
$*theta3*(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3
$       *(theta3-theta3n))/dts-(theta1+1.0/2.0*theta3*theta2)
$       *(theta2-1.0/2.0*theta2n)/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)**2-(((theta1-theta1n)/dts
$+(1.0/2.0*theta2
$       -1.0/2.0*theta2n)/dts*theta3+(1.0/2.0*theta3
$-1.0/2.0*theta3n)/dts
$       *theta2)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$       -(theta1+1.0/2.0*theta3*theta2)*(1.0/2.0*theta1*(theta1
```

```
$       -theta1n)+1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0*theta3
$       *(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)**3*theta2


  dCTdot(3,2,3)=((1.0/2.0/dts*theta2+(1.0/2.0*theta2
$-1.0/2.0*theta2n)/dts)
$       *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0*((theta1
$       -theta1n)/dts+(1.0/2.0*theta2-1.0/2.0*theta2n)
$/dts*theta3+(1.0/2.0*theta3
$       -1.0/2.0*theta3n)/dts*theta2)*theta3
$-1.0/2.0*theta2*(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3
$       *(theta3-theta3n))/dts-(theta1+1.0/2.0*theta3*theta2)
$       *(theta3-1.0/2.0*theta3n)/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2-(((theta1-theta1n)/dts+(1.0/2.0*theta2
$       -1.0/2.0*theta2n)/dts*theta3+(1.0/2.0*theta3
$-1.0/2.0*theta3n)/dts
$       *theta2)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$       -(theta1+1.0/2.0*theta3*theta2)*(1.0/2.0*theta1
$*(theta1-theta1n)
$       +1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0*theta3
$*(theta3-theta3n))
$       /dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta3


  dCTdot(2,2,1)=(-(theta1-1.0/2.0*theta1n)
$/dts*(1+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$+1.0/2.0*(-(1.0/2.0*theta1*(theta1
$       -theta1n)+1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0*theta3
$       *(theta3-theta3n))/dts+theta2*(theta2-theta2n)/dts)*theta1
$       +1.0/2.0*theta1*(1.0/2.0*theta1
$*(theta1-theta1n)+1.0/2.0*theta2*(theta2
$       -theta2n)+1.0/2.0*theta3*(theta3-theta3n))
$/dts-(1-1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2-1.0/4.0*theta3**2)
$*(theta1-1.0/2.0*theta1n)/dts)/(1
$  +1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0
$*theta3**2)**2-((-(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2*(theta2
$-theta2n)+1.0/2.0*theta3
$       *(theta3-theta3n))/dts+theta2
$*(theta2-theta2n)/dts)*(1+1.0/4.0
$  *theta1**2+1.0/4.0*theta2**2+1.0/4.0
$*theta3**2)-(1-1.0/4.0*theta1**2+1.0/4.0
$       *theta2**2-1.0/4.0*theta3**2)*(1.0/2.0
$*theta1*(theta1-theta1n)+1.0/2.0
$       *theta2*(theta2-theta2n)
$+1.0/2.0*theta3*(theta3-theta3n))/dts)
```

```
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta1

   dCTdot(2,2,2)=((-(theta2-1.0/2.0*theta2n)/dts+(theta2-theta2n)
$  /dts+1/dts*theta2)*(1+1.0/4.0*theta1**2+1.0/4.0
$*theta2**2+1.0/4.0*theta3**2)
$       +1.0/2.0*(-(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0*theta2
$       *(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts
$+theta2*(theta2-theta2n)/dts)*theta2-1.0/2.0*theta2*(1.0/2.0
$       *theta1*(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$       +1.0/2.0*theta3*(theta3-theta3n))/dts
$-(1-1.0/4.0*theta1**2+1.0/4.0
$       *theta2**2-1.0/4.0*theta3**2)*(theta2

$-1.0/2.0*theta2n)/dts)/(1
$       +1.0/4.0*theta1**2+1.0/4.0*theta2**2

$+1.0/4.0*theta3**2)**2-((-(1.0/2.0

$       *theta1*(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)

$  +1.0/2.0*theta3*(theta3-theta3n))
$/dts+theta2*(theta2-theta2n)/dts)

$  *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(1-1.0/4.0*theta1**2

$       +1.0/4.0*theta2**2-1.0/4.0*theta3**2)

$*(1.0/2.0*theta1*(theta1-theta1n)

$       +1.0/2.0*theta2*(theta2-theta2n)

$+1.0/2.0*theta3*(theta3-theta3n))
$  /dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta2

   dCTdot(2,2,3)=(-(theta3-1.0/2.0*theta3n)/dts*(1
$+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$+1.0/2.0*(-(1.0/2.0*theta1*(theta1
$       -theta1n)+1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3*(theta3
$       -theta3n))/dts+theta2*(theta2-theta2n)/dts)*theta3+1.0/2.0
$       *theta3*(1.0/2.0*theta1*(theta1-theta1n)
$+1.0/2.0*theta2*(theta2
$       -theta2n)+1.0/2.0*theta3*(theta3-theta3n))
$/dts-(1-1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2-1.0/4.0*theta3**2)
$*(theta3-1.0/2.0*theta3n)/dts)/(1
$  +1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2-((-(1.0/2.0*theta1
$       *(theta1-theta1n)+1.0/2.0*theta2
$*(theta2-theta2n)+1.0/2.0*theta3
$       *(theta3-theta3n))/dts+theta2
$*(theta2-theta2n)/dts)*(1+1.0/4.0
$  *theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
```

```
     $-(1-1.0/4.0*theta1**2+1.0/4.0
     $     *theta2**2-1.0/4.0*theta3**2)
     $*(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0
     $     *theta2*(theta2-theta2n)
     $+1.0/2.0*theta3*(theta3-theta3n))/dts)
     $     /(1+1.0/4.0*theta1**2+1.0/4.0
     $*theta2**2+1.0/4.0*theta3**2)**3*theta3

      dCTdot(3,3,1)=(-(theta1-1.0/2.0*theta1n)
     $/dts*(1+1.0/4.0*theta1**2+1.0/4.0
     $ *theta2**2+1.0/4.0*theta3**2)+1.0/2.0
     $*(-(1.0/2.0*theta1*(theta1-theta1n)
     $     +1.0/2.0*theta2*(theta2-theta2n)
     $+1.0/2.0*theta3*(theta3-theta3n))
     $     /dts+theta3*(theta3-theta3n)/dts)*theta1
     $+1.0/2.0*theta1*(1.0/2.0
     $     *theta1*(theta1-theta1n)+1.0/2.0
     $*theta2*(theta2-theta2n)
     $     +1.0/2.0*theta3*(theta3-theta3n))
     $/dts-(1-1.0/4.0*theta1**2-1.0/4.0
     $     *theta2**2+1.0/4.0*theta3**2)
     $*(theta1-1.0/2.0*theta1n)/dts)/(1
     $     +1.0/4.0*theta1**2+1.0/4.0*theta2**2
     $+1.0/4.0*theta3**2)**2-((-(1.0/2.0
     $     *theta1*(theta1-theta1n)+1.0/2.0*theta2
     $*(theta2-theta2n)
     $     +1.0/2.0*theta3*(theta3-theta3n))/dts
     $+theta3*(theta3-theta3n)
     $     /dts)*(1+1.0/4.0*theta1**2+1.0/4.0
     $*theta2**2+1.0/4.0*theta3**2)-(1
     $     -1.0/4.0*theta1**2-1.0/4.0*theta2**2
     $+1.0/4.0*theta3**2)*(1.0/2.0*theta1
     $     *(theta1-theta1n)+1.0/2.0*theta2
     $*(theta2-theta2n)+1.0/2.0
     $     *theta3*(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2+1.0/4.0
     $     *theta2**2+1.0/4.0*theta3**2)**3*theta1

      dCTdot(3,3,2)=(-(theta2-1.0/2.0*theta2n)/dts
     $*(1+1.0/4.0*theta1**2
     $     +1.0/4.0*theta2**2+1.0/4.0*theta3**2)
     $+1.0/2.0*(-(1.0/2.0*theta1*(theta1
     $     -theta1n)+1.0/2.0*theta2*(theta2-theta2n)
     $+1.0/2.0*theta3*(theta3
     $     -theta3n))/dts+theta3*(theta3-theta3n)
     $/dts)*theta2+1.0/2.0
     $     *theta2*(1.0/2.0*theta1*(theta1-theta1n)
     $+1.0/2.0*theta2*(theta2
     $     -theta2n)+1.0/2.0*theta3*(theta3-theta3n))
     $/dts-(1-1.0/4.0*theta1**2
     $     -1.0/4.0*theta2**2+1.0/4.0*theta3**2)
     $*(theta2-1.0/2.0*theta2n)/dts)/(1
     $ +1.0/4.0*theta1**2+1.0/4.0*theta2**2
     $+1.0/4.0*theta3**2)**2-((-(1.0/2.0*theta1
     $     *(theta1-theta1n)+1.0/2.0*theta2
     $*(theta2-theta2n)+1.0/2.0*theta3
     $     *(theta3-theta3n))/dts+theta3*(theta3-theta3n)/dts)*(1
```

```
$  +1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(1-1.0/4.0*theta1**2
$        -1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$*(1.0/2.0*theta1*(theta1-theta1n)
$        +1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3*(theta3-theta3n))
$  /dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta2

   dCTdot(3,3,3)=((-(theta3-1.0/2.0*theta3n)/dts+(theta3-theta3n)
$  /dts+theta3/dts)*(1+1.0/4.0*theta1**2+1.0/4.0
$*theta2**2+1.0/4.0*theta3**2)
$        +1.0/2.0*(-(1.0/2.0*theta1*(theta1
$-theta1n)+1.0/2.0*theta2*(theta2
$        -theta2n)+1.0/2.0*theta3*(theta3
$-theta3n))/dts+theta3*(theta3
$  -theta3n)/dts)*theta3-1.0/2.0*theta3
$*(1.0/2.0*theta1*(theta1-theta1n)
$  +1.0/2.0*theta2*(theta2-theta2n)
$+1.0/2.0*theta3*(theta3-theta3n))/dts
$        -(1-1.0/4.0*theta1**2-1.0/4.0
$*theta2**2+1.0/4.0*theta3**2)*(theta3-1.0/2.0
$  *theta3n)/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2
$  -((-(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0*theta2
$*(theta2-theta2n)
$        +1.0/2.0*theta3*(theta3-theta3n))/dts
$+theta3*(theta3-theta3n)
$        /dts)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(1
$        -1.0/4.0*theta1**2-1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)*(1.0/2.0*theta1
$        *(theta1-theta1n)+1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0
$        *theta3*(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2+1.0/4.0
$        *theta2**2+1.0/4.0*theta3**2)**3*theta3
   dCTdot(1,2,1)=((1.0/2.0/dts*theta2+(1.0/2.0*theta2
$-1.0/2.0*theta2n)
$              /dts)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$              +1.0/2.0*((-theta3+theta3n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)
$              /dts*theta2+(1.0/2.0*theta2-1.0/2.0*theta2n)/dts
$*theta1)*theta1
$              -1.0/2.0*theta2*(1.0/2.0*theta1*(theta1-theta1n)
$+1.0/2.0*theta2
$              *(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts
$     -(-theta3+1.0/2.0*theta2*theta1)*(theta1-1.0/2.0*theta1n)
$              /dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2
$              -(((-theta3+theta3n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)/dts
$              *theta2+(1.0/2.0*theta2-1.0/2.0*theta2n)/dts*theta1)*(1
$              +1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(-theta3+1.0/2.0
$     *theta2*theta1)*(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0
$     *theta2*(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))
```

```
$     /dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta1

    dCTdot(1,2,2)=(((1.0/2.0*theta1-1.0/2.0*theta1n)/dts
$     +1.0/2.0/dts*theta1)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$              +1.0/2.0*((-theta3+theta3n)/dts+(1.0/2.0*theta1
$-1.0/2.0*theta1n)
$ /dts*theta2+(1.0/2.0*theta2-1.0/2.0*theta2n)/dts*theta1)
$              *theta2-1.0/2.0*theta1*(1.0/2.0*theta1*(theta1-theta1n)
$              +1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0*theta3*(theta3
$              -theta3n))/dts-(-theta3+1.0/2.0*theta2*theta1)*(theta2
$              -1.0/2.0*theta2n)/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$              +1.0/4.0*theta3**2)**2-(((-theta3+theta3n)/dts
$+(1.0/2.0*theta1
$              -1.0/2.0*theta1n)/dts*theta2+(1.0/2.0*theta2
$-1.0/2.0*theta2n)
$              /dts*theta1)*(1+1.0/4.0*theta1**2+1.0/4.0
$*theta2**2+1.0/4.0*theta3**2)
$     -(-theta3+1.0/2.0*theta2*theta1)*(1.0/2.0*theta1*(theta1
$     -theta1n)+1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0*theta3
$              *(theta3-theta3n))/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$              +1.0/4.0*theta3**2)**3*theta2

   dCTdot(1,2,3)=(-1/dts*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)+1.0/2.0*((-theta3+theta3n)/dts
$+(1.0/2.0*theta1
$-1.0/2.0*theta1n)/dts*theta2+(1.0/2.0*theta2-1.0/2.0*theta2n)
$       /dts*theta1)*theta3+(1.0/2.0*theta1*(theta1-theta1n)
$       +1.0/2.0*theta2*(theta2-theta2n)+1.0/2.0*theta3*(theta3
$       -theta3n))/dts-(-theta3+1.0/2.0*theta2*theta1)*
$       (theta3-1.0/2.0*theta3n)/dts)/(1+1.0/4.0*theta1**2+1.0/4.0
$       *theta2**2+1.0/4.0*theta3**2)**2-(((-theta3+theta3n)
$       /dts+(1.0/2.0*theta1-1.0/2.0*theta1n)/dts*theta2+(1.0/2.0
$       *theta2-1.0/2.0*theta2n)/dts*theta1)*(1+1.0/4.0*theta1**2
$       +1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$-(-theta3+1.0/2.0*theta2
$       *theta1)*(1.0/2.0*theta1*(theta1-theta1n)+1.0/2.0*theta2
$       *(theta2-theta2n)+1.0/2.0*theta3*(theta3-theta3n))/dts)
$       /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta3
        else
 dfhdtheta(1,1)=1.0/2.0*theta1*(3.0/2.0*theta1-2*theta1n+
$     1.0/2.0*theta1n1)/dts+3.0/2.0*(1.0+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)/dts+1.0/4.0*theta3*theta1*(3.0/2.0*theta2-
$2*theta2n+1.0/2.0*theta2n1)/dts-1.0/4.0*theta1*theta2*(3.0/2.0*
$theta3-2*theta3n+1.0/2.0*theta3n1)/dts


  dfhdtheta(1,2)=1.0/2.0*theta2*(3.0/2.0*theta1-2*theta1n
$+1.0/2.0*theta1n1)/dts+1.0/4.0*theta2*theta3*(3.0/2.0*theta2-2
$ *theta2n+1.0/2.0*theta2n1)/dts+3.0d0/4.0d0*(1
$+1.0/4.0*theta1**2+1.0/4.0
```

```
$       *theta2**2+1.0/4.0*theta3**2)*theta3/dts-1.0/4.0*theta2**2
$       *(3.0/2.0*theta3-2*theta3n+1.0/2.0*theta3n1)/dts-1.0/2.0*(1
$+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)*(3.0/2.0
$       *theta3-2*theta3n+1.0/2.0*theta3n1)/dts
 dfhdtheta(1,3)=1.0/2.0*theta3*(3.0/2.0*theta1-2*theta1n
$       +1.0/2.0*theta1n1)/dts+1.0/4.0*theta3**2*(3.0/2.0*theta2-2
$       *theta2n+1.0/2.0*theta2n1)/dts+1.0/2.0*(1+1.0/4.0*theta1**2
$ +1.0/4.0*theta2**2+1.0/4.0*theta3**2)*(3.0/2.0*theta2-2*theta2n
$+1.0/2.0*theta2n1)/dts-1.0/4.0*theta3*theta2*(3.0/2.0*theta3-2
$ *theta3n+1.0/2.0*theta3n1)/dts-3.0d0/4.0d0*(1
$+1.0/4.0*theta1**2+1.0/4.0
$       *theta2**2+1.0/4.0*theta3**2)*theta2/dts
 dfhdtheta(2,1)=-1.0/4.0*theta1*theta3*(3.0/2.0*theta1-2
$ *theta1n+1.0/2.0*theta1n1)/dts-3.0d0/4.0d0
$*(1+1.0/4.0*theta1**2+1.0/4.0
$ *theta2**2+1.0/4.0*theta3**2)*theta3/dts+1.0/2.0*theta1*(3.0/2.0
$ *theta2-2*theta2n+1.0/2.0*theta2n1)/dts+1.0/4.0*theta1**2
$       *(3.0/2.0*theta3-2*theta3n+1.0/2.0*theta3n1)/dts+1.0/2.0*(1
$ +1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)*(3.0/2.0
$       *theta3-2*theta3n+1.0/2.0*theta3n1)/dts
 dfhdtheta(2,2)=-1.0/4.0*theta2*theta3*(3.0/2.0*theta1-2
$       *theta1n+1.0/2.0*theta1n1)/dts+1.0/2.0*theta2*(3.0/2.0*theta2
$ -2*theta2n+1.0/2.0*theta2n1)/dts+3.0/2.0*(1
$+1.0/4.0*theta1**2+1.0/4.0
$ *theta2**2+1.0/4.0*theta3**2)/dts+1.0/4.0*theta1*theta2*(3.0/2.0
$       *theta3-2*theta3n+1.0/2.0*theta3n1)/dts
 dfhdtheta(2,3)=-1.0/4.0*theta3**2*(3.0/2.0*theta1-2*theta1n
$ +1.0/2.0*theta1n1)/dts-1.0/2.0*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)*(3.0/2.0*theta1-2*theta1n
$+1.0/2.0*theta1n1)/dts
$ +1.0/2.0*theta3*(3.0/2.0*theta2-2*theta2n+1.0/2.0*theta2n1)/dts
$       +1.0/4.0*theta3*theta1*(3.0/2.0*theta3-2*theta3n+1.0/2.0
$*theta3n1)/dts+3.0d0/4.0d0*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)*theta1/dts
 dfhdtheta(3,1)=1.0/4.0*theta1*theta2*(3.0/2.0*theta1
$-2*theta1n+1.0/2.0*theta1n1)/dts+3.0d0/4.0d0*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)*theta2/dts
$ -1.0/4.0*theta1**2
$*(3.0/2.0*theta2-2*theta2n+1.0/2.0*theta2n1)/dts
$-1.0/2.0*(1+1.0/4.0
$ *theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)*(3.0/2.0*theta2-2
$ *theta2n+1.0/2.0*theta2n1)/dts+1.0/2.0*theta1*(3.0/2.0*theta3-2
$       *theta3n+1.0/2.0*theta3n1)/dts
 dfhdtheta(3,2)=1.0/4.0*theta2**2*(3.0/2.0*theta1-2*theta1n
$+1.0/2.0*theta1n1)/dts+1.0/2.0*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$ +1.0/4.0*theta3**2)*(3.0/2.0*theta1-2*theta1n+1.0/2.0*theta1n1)
$       /dts-1.0/4.0*theta2*theta1*(3.0/2.0*theta2-2*theta2n+1.0/2.0
$*theta2n1)/dts-3.0d0/4.0d0*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$       +1.0/4.0*theta3**2)*theta1/dts+1.0/2.0*theta2*(3.0/2.0*theta3
$       -2*theta3n+1.0/2.0*theta3n1)/dts
 dfhdtheta(3,3)=1.0/4.0*theta2*theta3*(3.0/2.0*theta1
$       -2*theta1n+1.0/2.0*theta1n1)/dts-1.0/4.0*theta3*theta1
$       *(3.0/2.0*theta2-2*theta2n+1.0/2.0*theta2n1)/dts+1.0/2.0
```

```
$     *theta3*(3.0/2.0*theta3-2*theta3n+1.0/2.0*theta3n1)/dts
$ +3.0/2.0*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)/dts

 dCTdot(1,1,1)=((-(3.0/2.0*theta1-theta1n+1.0/4.0*theta1n1)/dts
$     +1.0/2.0*(3*theta1-4*theta1n+theta1n1)/dts+3.0/2.0*theta1/dts)
$      *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0*(
$   -(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$   *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$   *(3*theta3-4*theta3n+theta3n1))/dts+1.0/2.0*theta1*(3
$   *theta1-4*theta1n+theta1n1)/dts)*theta1-1.0/2.0*theta1
$   *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$   *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$   *(3*theta3-4*theta3n+theta3n1))/dts-(1+1.0/4.0*theta1**2
$ -1.0/4.0*theta2**2-1.0/4.0*theta3**2)*(3.0/2.0*theta1-theta1n
$ +1.0/4.0*theta1n1)/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$  +1.0/4.0*theta3**2)**2-((-(1.0/4.0*theta1*(3*theta1-4*theta1n
$   +theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts+1.0/2.0*theta1
$   *(3*theta1-4*theta1n+theta1n1)/dts)*(1+1.0/4.0*theta1**2
$   +1.0/4.0*theta2**2+1.0/4.0*theta3**2)-(1
$+1.0/4.0*theta1**2-1.0/4.0
$   *theta2**2-1.0/4.0*theta3**2)*(1.0/4.0*theta1*(3*theta1
$   -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4
$   *theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4
$*theta3n+theta3n1))/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$   +1.0/4.0*theta3**2)**3*theta1
 dCTdot(1,1,2)=(-(3.0/2.0*theta2-theta2n+1.0/4.0*theta2n1)/dts
$*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0
$   *(-(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$   *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$   *(3*theta3-4*theta3n+theta3n1))/dts+1.0/2.0*theta1*(3
$   *theta1-4*theta1n+theta1n1)/dts)*theta2+1.0/2.0*theta2
$   *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$   *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$   *(3*theta3-4*theta3n+theta3n1))/dts-(1+1.0/4.0*theta1**2
$  -1.0/4.0*theta2**2-1.0/4.0*theta3**2)*(3.0/2.0*theta2-theta2n
$+1.0/4.0*theta2n1)/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$  +1.0/4.0*theta3**2)**2-((-(1.0/4.0*theta1*(3*theta1-4*theta1n
$   +theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$   +1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts+1.0/2.0
$   *theta1*(3*theta1-4*theta1n+theta1n1)/dts)*(1+1.0/4.0
$  *theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(1+1.0/4.0*theta1**2
$ -1.0/4.0*theta2**2-1.0/4.0*theta3**2)*(1.0/4.0*theta1*(3*theta1
$   -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$   +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$ /dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta2

 dCTdot(1,1,3)=(-(3.0/2.0*theta3-theta3n+1.0/4.0*theta3n1)
$     /dts*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0
$   *(-(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
```

```
$    *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$    *(3*theta3-4*theta3n+theta3n1))/dts+1.0/2.0*theta1*(3
$    *theta1-4*theta1n+theta1n1)/dts)*theta3+1.0/2.0*theta3
$   *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$    *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4
$*theta3n+theta3n1))/dts-(1+1.0/4.0*theta1**2-1.0/4.0*theta2**2
$-1.0/4.0*theta3**2)*(3.0/2.0*theta3-theta3n+1.0/4.0
$ *theta3n1)/dts)
$ /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2-((
$   -(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$    *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$   -4*theta3n+theta3n1))/dts+1.0/2.0*theta1*(3*theta1-4*theta1n
$  +theta1n1)/dts)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$-(1+1.0/4.0*theta1**2-1.0/4.0*theta2**2
$-1.0/4.0*theta3**2)*(1.0/4.0*theta1
$    *(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2
$   -4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n
$+theta3n1))/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0
$    *theta3**2)**3*theta3


 dCTdot(2,1,1)=((3.0/4.0/dts*theta2+(3.0/4.0*theta2
$      -theta2n+1.0/4.0*theta2n1)/dts)*(1+1.0/4.0*theta1**2+1.0/4.0
$ *theta2**2+1.0/4.0*theta3**2)+1.0/2.0*((3.0/2.0*theta3-2*theta3n
$ +1.0/2.0*theta3n1)/dts+(3.0/4.0*theta1-theta1n+1.0/4.0*theta1n1)
$/dts*theta2+(3.0/4.0*theta2-theta2n+1.0/4.0*theta2n1)/dts*theta1)
$    *theta1-1.0/2.0*theta2*(1.0/4.0*theta1*(3*theta1-4*theta1n
$    +theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$    +1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts
$    -(theta3+1.0/2.0*theta2*theta1)*(3.0/2.0*theta1-theta1n
$ +1.0/4.0*theta1n1)/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$    +1.0/4.0*theta3**2)**2-(((3.0/2.0*theta3-2*theta3n+1.0/2.0
$    *theta3n1)/dts+(3.0/4.0*theta1-theta1n+1.0/4.0*theta1n1
$    )/dts*theta2+(3.0/4.0*theta2-theta2n+1.0/4.0*theta2n1)
$    /dts*theta1)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$    -(theta3+1.0/2.0*theta2*theta1)*(1.0/4.0*theta1*(3*theta1
$    -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$    +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$ +1.0/4.0*theta3**2)**3
$    *theta1

 dCTdot(2,1,2)=(((3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts
$      +3.0d0/4.0d0/dts*theta1)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2
$**2+1.0/4.0
$ *theta3**2)+1.0/2.0*((3.0/2.0*theta3-2*theta3n+1.0/2.0*theta3n1)
$/dts+(3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts*theta2
$+(3.0d0/4.0d0
$    *theta2-theta2n+1.0/4.0*theta2n1)/dts*theta1)*theta2-1.0/2.0
$   *theta1*(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$    *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3
$*theta3-4*theta3n+theta3n1))/dts-(theta3+1.0/2.0*theta2*theta1)
$    *(3.0/2.0*theta2-theta2n+1.0/4.0*theta2n1)/dts)
$/(1+1.0/4.0*theta1**2
```

```
$    +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
$-(((3.0/2.0*theta3-2*theta3n
$+1.0/2.0*theta3n1)/dts+(3.0d0/4.0d0*theta1-theta1n
$+1.0/4.0*theta1n1)
$/dts*theta2+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0*theta2n1)
$/dts*theta1)
$    *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$    -(theta3+1.0/2.0*theta2*theta1)*(1.0/4.0*theta1*(3*theta1
$    -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$    +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta2

 dCTdot(2,1,3)=(3.0/2.0/dts*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0
$*theta3**2)+1.0/2.0*((3.0/2.0*theta3
$-2*theta3n+1.0/2.0*theta3n1)/dts
$+(3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts*theta2
$+(3.0d0/4.0d0*theta2
$-theta2n+1.0/4.0*theta2n1)/dts*theta1)*theta3-(1.0/4.0*theta1
$    *(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4
$    *theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$-4*theta3n+theta3n1))
$    /dts-(theta3+1.0/2.0*theta2*theta1)*(3.0/2.0*theta3-theta3n
$    +1.0/4.0*theta3n1)/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0
$    *theta3**2)**2-(((3.0/2.0*theta3-2*theta3n
$+1.0/2.0*theta3n1)/dts
$    +(3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts*theta2
$+(3.0d0/4.0d0*theta2
$    -theta2n+1.0/4.0*theta2n1)/dts*theta1)*(1+1.0/4.0*theta1**2
$    +1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$-(theta3+1.0/2.0*theta2*theta1)
$  *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$    *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4
$    *theta3n+theta3n1))/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0
$    *theta3**2)**3*theta3

 dCTdot(1,3,1)=((3.0d0/4.0d0*theta3/dts+(3.0d0/4.0d0*theta3
$-theta3n+1.0/4.0*theta3n1)/dts)*(1
$+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$ +1.0/4.0*theta3**2)+1.0/2.0*((3.0/2.0*theta2
$-2*theta2n+1.0/2.0*theta2n1)/dts
$ +(3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts*theta3
$+(3.0d0/4.0d0*theta3-theta3n+1.0/4.0*theta3n1)/dts*theta1)*theta1
$ -1.0/2.0*theta3*(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)
$ +1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$+1.0/4.0*theta3*(3*theta3
$ -4*theta3n+theta3n1))/dts-(theta2+1.0/2.0*theta3*theta1)
$*(3.0/2.0*theta1
$ -theta1n+1.0/4.0*theta1n1)/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$ +1.0/4.0*theta3**2)**2-(((3.0/2.0*theta2
$-2*theta2n+1.0/2.0*theta2n1)/dts
$ +(3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts
```

```
$*theta3+(3.0d0/4.0d0
$*theta3-theta3n+1.0/4.0*theta3n1)/dts*theta1)
$ *(1+1.0/4.0*theta1**2
$ +1.0/4.0*theta2**2+1.0/4.0*theta3**2)-(theta2
$+1.0/2.0*theta3*theta1)
$ *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)
$+1.0/4.0*theta2*(3*theta2
$ -4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$-4*theta3n+theta3n1))/dts)
$ /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta1

 dCTdot(1,3,2)=(3.0/2.0/dts*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$ +1.0/4.0*theta3**2)+1.0/2.0*((3.0/2.0*theta2
$-2*theta2n+1.0/2.0*theta2n1)/dts
$ +(3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts*theta3
$+(3.0d0/4.0d0
$ *theta3-theta3n+1.0/4.0*theta3n1)/dts
$*theta1)*theta2-(1.0/4.0*theta1
$*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$ +theta2n1)+1.0/4.0*theta3*(3*theta3
$-4*theta3n+theta3n1))/dts-(theta2
$+1.0/2.0*theta3*theta1)*(3.0/2.0*theta2-theta2n
$+1.0/4.0*theta2n1)/dts)
$ /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
$-(((3.0/2.0*theta2-2*theta2n+1.0/2.0*theta2n1)/dts
$+(3.0d0/4.0d0*theta1
$-theta1n+1.0/4.0*theta1n1)/dts*theta3+(3.0d0/4.0d0*theta3-theta3n
$ +1.0/4.0*theta3n1)/dts*theta1)*(1
$+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$ +1.0/4.0*theta3**2)-(theta2+1.0/2.0*theta3*theta1)
$*(1.0/4.0*theta1
$*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$ +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts)
$ /(1+1.0/4.0*theta1**2+1.0/4.0
$*theta2**2+1.0/4.0*theta3**2)**3*theta2

 dCTdot(1,3,3)=(((3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts
$      +3.0d0/4.0d0/dts*theta1)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$    +1.0/2.0*((3.0/2.0*theta2-2*theta2n
$+1.0/2.0*theta2n1)/dts+(3.0d0/4.0d0*theta1
$    -theta1n+1.0/4.0*theta1n1)/dts*theta3+(3.0d0/4.0d0*theta3
$-theta3n+1.0/4.0
$*theta3n1)/dts*theta1)*theta3-1.0/2.0*theta1*(1.0/4.0*theta1*(3
$    *theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4
$    *theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n
$  +theta3n1))/dts-(theta2+1.0/2.0*theta3*theta1)*(3.0/2.0*theta3
$  -theta3n+1.0/4.0*theta3n1)/dts)
$/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$    +1.0/4.0*theta3**2)**2-(((3.0/2.0*theta2
$-2*theta2n+1.0/2.0*theta2n1)
$/dts+(3.0d0/4.0d0*theta1-theta1n
$+1.0/4.0*theta1n1)/dts*theta3+(3.0d0/4.0d0
$    *theta3-theta3n+1.0/4.0*theta3n1)/dts*theta1)
$*(1+1.0/4.0*theta1**2
```

```
$    +1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$-(theta2+1.0/2.0*theta3*theta1)
$ *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$    *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$    -4*theta3n+theta3n1))/dts)/(1+1.0/4.0*theta1**2+1.0/4.0
$    *theta2**2+1.0/4.0*theta3**2)**3*theta3



 dCTdot(3,1,1)=(((-3.0/2.0*theta2+2*theta2n-1.0/2.0*theta2n1)
$     /dts+(3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts*theta3

$+(3.0d0/4.0d0*theta3-theta3n+1.0/4.0*theta3n1)/dts*theta1)
$*(1+1.0/4.0*theta1**2

$    +1.0/4.0*theta2**2+1.0/4.0*theta3**2)

$-(-theta2+1.0/2.0*theta3*theta1)
$ *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$   *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4

$*theta3n+theta3n1))/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2

$    +1.0/4.0*theta3**2)**2


 dCTdot(3,1,2)=((3.0d0/4.0d0*theta3/dts
$+(3.0d0/4.0d0*theta3-theta3n
$+1.0/4.0*theta3n1)/dts)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0
$*theta3**2)+1.0/2.0*((-3.0/2.0*theta2+2*theta2n-1.0/2.0*theta2n1)
$/dts+(3.0d0/4.0d0*theta1-theta1n
$+1.0/4.0*theta1n1)/dts*theta3+(3.0d0/4.0d0
$ *theta3-theta3n+1.0/4.0*theta3n1)/dts*theta1)*theta1-1.0/2.0
$ *theta3*(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$    *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$    *(3*theta3-4*theta3n+theta3n1))/dts-(-theta2+1.0/2.0
$ *theta3*theta1)*(3.0/2.0*theta1-theta1n+1.0/4.0*theta1n1)/dts)
$  /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
$-(((-3.0/2.0*theta2+2*theta2n-1.0/2.0*theta2n1)/dts+(3.0d0/4.0d0
$ *theta1-theta1n+1.0/4.0*theta1n1)/dts*theta3+(3.0d0/4.0d0*theta3
$    -theta3n+1.0/4.0*theta3n1)/dts*theta1)*(1+1.0/4.0*theta1**2
$    +1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$-(-theta2+1.0/2.0*theta3*theta1)
$ *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$    *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4
$  *theta3n+theta3n1))/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$    +1.0/4.0*theta3**2)**3*theta1

 dCTdot(3,1,3)=(((3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts
$      +3.0d0/4.0d0/dts*theta1)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$    +1.0/2.0*((-3.0/2.0*theta2+2*theta2n
$-1.0/2.0*theta2n1)/dts+(3.0d0/4.0d0*theta1
$-theta1n+1.0/4.0*theta1n1)/dts*theta3+(3.0d0/4.0d0*theta3-theta3n
$    +1.0/4.0*theta3n1)/dts*theta1)*theta3
```

```
$-1.0/2.0*theta1*(1.0/4.0*theta1
$    *(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4
$ *theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$    /dts-(-theta2+1.0/2.0*theta3*theta1)*(3.0/2.0*theta3-theta3n
$    +1.0/4.0*theta3n1)/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0
$ *theta3**2)**2-(((-3.0/2.0*theta2+2*theta2n-1.0/2.0*theta2n1)
$/dts+(3.0d0/4.0d0*theta1-theta1n
$+1.0/4.0*theta1n1)/dts*theta3+(3.0d0/4.0d0
$    *theta3-theta3n+1.0/4.0*theta3n1)/dts*theta1)
$*(1+1.0/4.0*theta1**2
$    +1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$-(-theta2+1.0/2.0*theta3*theta1)
$ *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$    *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4
$    *theta3n+theta3n1))/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$    +1.0/4.0*theta3**2)**3*theta3

 dCTdot(2,3,1)=(-3.0/2.0/dts*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2
$      +1.0/4.0*theta3**2)+1.0/2.0*((-3.0/2.0
$*theta1+2*theta1n-1.0/2.0*theta1n1)
$ /dts+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0*theta2n1)/dts*theta3
$+(3.0d0/4.0d0*theta3-theta3n+1.0/4.0*theta3n1)
$/dts*theta2)*theta1+(1.0/4.0*theta1*(3*theta1
$    -4*theta1n+theta1n1)+1.0/4.0*theta2*(3
$*theta2-4*theta2n+theta2n1)
$    +1.0/4.0*theta3*(3*theta3-4*theta3n
$+theta3n1))/dts-(-theta1+1.0/2.0
$*theta3*theta2)*(3.0/2.0*theta1-theta1n+1.0/4.0*theta1n1)/dts)
$ /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2-(((-3.0/2.0*theta1
$    +2*theta1n-1.0/2.0*theta1n1)/dts
$+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0
$    *theta2n1)/dts*theta3+(3.0d0/4.0d0*theta3
$-theta3n+1.0/4.0*theta3n1)
$    /dts*theta2)*(1+1.0/4.0*theta1**2+1.0/4.0
$*theta2**2+1.0/4.0*theta3**2)
$    -(-theta1+1.0/2.0*theta3*theta2)*(1.0/4.0*theta1*(3*theta1-4
$    *theta1n+theta1n1)+1.0/4.0*theta2
$*(3*theta2-4*theta2n+theta2n1)
$    +1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts)/(1
$    +1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta1

 dCTdot(2,3,2)=((3.0d0/4.0d0*theta3/dts
$+(3.0d0/4.0d0*theta3-theta3n
$      +1.0/4.0*theta3n1)/dts)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0
$    *theta3**2)+1.0/2.0*((-3.0/2.0*theta1+2*theta1n
$-1.0/2.0*theta1n1)
$/dts+(3.0d0/4.0d0*theta2-theta2n
$+1.0/4.0*theta2n1)/dts*theta3+(3.0d0/4.0d0
$    *theta3-theta3n+1.0/4.0*theta3n1)/dts*theta2)*theta2-1.0/2.0
$   *theta3*(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
```

```
$     *theta2*(3*theta2-4*theta2n+theta2n1)
$+1.0/4.0*theta3*(3*theta3
$ -4*theta3n+theta3n1))/dts
$-(-theta1+1.0/2.0*theta3*theta2)*(3.0/2.0
$*theta2-theta2n+1.0/4.0*theta2n1)/dts)/(1
$+1.0/4.0*theta1**2+1.0/4.0
$ *theta2**2+1.0/4.0*theta3**2)**2-(((-3.0/2.0*theta1+2*theta1n
$-1.0/2.0*theta1n1)/dts+(3.0d0/4.0d0
$*theta2-theta2n+1.0/4.0*theta2n1)
$     /dts*theta3+(3.0d0/4.0d0*theta3
$-theta3n+1.0/4.0*theta3n1)/dts*theta2)
$     *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$-(-theta1
$     +1.0/2.0*theta3*theta2)*(1.0/4.0*theta1*(3*theta1-4*theta1n
$     +theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$     +1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts)/(1
$     +1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta2

 dCTdot(2,3,3)=((3.0d0/4.0d0/dts*theta2
$+(3.0d0/4.0d0*theta2-theta2n
$      +1.0/4.0*theta2n1)/dts)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0
$     *theta3**2)+1.0/2.0*((-3.0/2.0*theta1+2*theta1n
$-1.0/2.0*theta1n1)
$/dts+(3.0d0/4.0d0*theta2-theta2n
$+1.0/4.0*theta2n1)/dts*theta3+(3.0d0/4.0d0
$     *theta3-theta3n+1.0/4.0*theta3n1)/dts*theta2)*theta3-1.0/2.0
$ *theta2*(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$     *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3
$     *theta3-4*theta3n+theta3n1))/dts-(-theta1+1.0/2.0*theta3
$     *theta2)*(3.0/2.0*theta3-theta3n
$+1.0/4.0*theta3n1)/dts)/(1+1.0/4.0
$     *theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$**2-(((-3.0/2.0*theta1+2
$     *theta1n-1.0/2.0*theta1n1)/dts+(3.0d0/4.0d0*theta2-theta2n
$+1.0/4.0*theta2n1)
$/dts*theta3+(3.0d0/4.0d0*theta3-theta3n+1.0/4.0*theta3n1)/dts
$     *theta2)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)
$     -(-theta1+1.0/2.0*theta3*theta2)*(1.0/4.0*theta1*(3*theta1
$     -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$     +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$     /dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta3

 dCTdot(3,2,1)=(3.0/2.0/dts*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$      +1.0/4.0*theta3**2)+1.0/2.0*((3.0/2.0*theta1-2*theta1n
$+1.0/2.0*theta1n1)
$/dts+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0*theta2n1)/dts*theta3
$+(3.0d0/4.0d0*theta3-theta3n+1.0/4.0*theta3n1)/dts*theta2)*theta1
$-(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$     *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4
$     *theta3n+theta3n1))/dts-(theta1
$+1.0/2.0*theta3*theta2)*(3.0/2.0
$     *theta1-theta1n+1.0/4.0*theta1n1)/dts)/(1
```

```
$+1.0/4.0*theta1**2+1.0/4.0
$   *theta2**2+1.0/4.0*theta3**2)**2-(((3.0/2.0*theta1
$-2*theta1n+1.0/2.0
$ *theta1n1)/dts+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0*theta2n1)/dts
$*theta3+(3.0d0/4.0d0*theta3-theta3n+1.0/4.0*theta3n1)/dts*theta2)
$   *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$   +1.0/4.0*theta3**2)-(theta1
$   +1.0/2.0*theta3*theta2)*(1.0/4.0*theta1*(3*theta1-4*theta1n
$   +theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$   +1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts)/(1
$+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3*theta1

 dCTdot(3,2,2)=((3.0d0/4.0d0*theta3/dts+(3.0d0/4.0d0*theta3
$-theta3n+1.0/4.0
$*theta3n1)/dts)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$   +1.0/2.0*((3.0/2.0*theta1-2*theta1n
$+1.0/2.0*theta1n1)/dts+(3.0d0/4.0d0*theta2
$   -theta2n+1.0/4.0*theta2n1)/dts*theta3
$+(3.0d0/4.0d0*theta3-theta3n
$+1.0/4.0*theta3n1)/dts*theta2)*theta2-1.0/2.0*theta3*(1.0/4.0
$   *theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$   *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$   -4*theta3n+theta3n1))/dts-(theta1+1.0/2.0*theta3*theta2)
$   *(3.0/2.0*theta2-theta2n+1.0/4.0*theta2n1)/dts)
$/(1+1.0/4.0*theta1**2
$   +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
$-(((3.0/2.0*theta1-2*theta1n
$ +1.0/2.0*theta1n1)/dts+(3.0d0/4.0d0*theta2-theta2n
$+1.0/4.0*theta2n1)/dts
$   *theta3+(3.0d0/4.0d0*theta3-theta3n
$+1.0/4.0*theta3n1)/dts*theta2)
$   *(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(theta1
$   +1.0/2.0*theta3*theta2)*(1.0/4.0*theta1*(3*theta1-4*theta1n
$   +theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$   +1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts)/(1
$   +1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta2

 dCTdot(3,2,3)=((3.0d0/4.0d0/dts*theta2
$+(3.0d0/4.0d0*theta2-theta2n
$     +1.0/4.0*theta2n1)/dts)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0
$   *theta3**2)+1.0/2.0*((3.0/2.0*theta1-2*theta1n
$+1.0/2.0*theta1n1)
$/dts+(3.0d0/4.0d0*theta2-theta2n
$+1.0/4.0*theta2n1)/dts*theta3+(3.0d0/4.0d0
$   *theta3-theta3n+1.0/4.0*theta3n1)/dts*theta2)*theta3-1.0/2.0
$   *theta2*(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)
$   +1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$   *(3*theta3-4*theta3n+theta3n1))/dts-(theta1+1.0/2.0*theta3
$   *theta2)*(3.0/2.0*theta3-theta3n
$+1.0/4.0*theta3n1)/dts)/(1+1.0/4.0
$   *theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2-(((3.0/2.0*theta1
```

```
$    -2*theta1n+1.0/2.0*theta1n1)/dts
$+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0
$    *theta2n1)/dts*theta3+(3.0d0/4.0d0
$*theta3-theta3n+1.0/4.0*theta3n1)
$    /dts*theta2)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$    -(theta1+1.0/2.0*theta3*theta2)*(1.0/4.0*theta1*(3*theta1-4
$    *theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$    +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$    /dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3*theta3

 dCTdot(2,2,1)=(-(3.0/2.0*theta1-theta1n+1.0/4.0*theta1n1)
$      /dts*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0*(
$-(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$    *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4*
$    theta3n+theta3n1))/dts+1.0/2.0*theta2*(3*theta2-4*theta2n
$ +theta2n1)/dts)*theta1+1.0/2.0*theta1*(1.0/4.0*theta1*(3*theta1
$    -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$    +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$/dts-(1-1.0/4.0*theta1**2+1.0/4.0*theta2**2
$-1.0/4.0*theta3**2)*(3.0/2.0
$    *theta1-theta1n+1.0/4.0*theta1n1)/dts)/(1+1.0/4.0*theta1**2
$    +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
$-((-(1.0/4.0*theta1*(3*theta1
$    -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$    +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$    /dts+1.0/2.0*theta2*(3*theta2-4*theta2n+theta2n1)/dts)*
$    (1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(1-1.0/4.0
$ *theta1**2+1.0/4.0*theta2**2-1.0/4.0*theta3**2)*(1.0/4.0*theta1
$    *(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2
$    -4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n
$ +theta3n1))/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0
$    *theta3**2)**3*theta1

 dCTdot(2,2,2)=((-(3.0/2.0*theta2-theta2n+1.0/4.0*theta2n1)
$      /dts+1.0/2.0*(3*theta2-4*theta2n+theta2n1)/dts+3.0/2.0*theta2
$ /dts)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0
$    *(-(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$    *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$    *(3*theta3-4*theta3n+theta3n1))/dts+1.0/2.0*theta2*(3
$    *theta2-4*theta2n+theta2n1)/dts)*theta2-1.0/2.0*theta2
$ *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$    *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$ -4*theta3n+theta3n1))/dts-(1-1.0/4.0*theta1**2+1.0/4.0*theta2**2
$-1.0/4.0*theta3**2)*(3.0/2.0*theta2
$-theta2n+1.0/4.0*theta2n1)/dts)
$ /(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2-((
$-(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$    *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$    -4*theta3n+theta3n1))/dts+1.0/2.0*theta2*(3*theta2-4
$ *theta2n+theta2n1)/dts)*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(1-1.0/4.0
```

```
$*theta1**2+1.0/4.0*theta2**2-1.0/4.0
$    *theta3**2)*(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)
$    +1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$    *(3*theta3-4*theta3n+theta3n1))/dts)/(1+1.0/4.0*theta1**2
$    +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3*theta2

 dCTdot(2,2,3)=(-(3.0/2.0*theta3-theta3n+1.0/4.0*theta3n1)
$/dts*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0
$    *(-(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$    *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$    *(3*theta3-4*theta3n+theta3n1))/dts+1.0/2.0*theta2*(3
$    *theta2-4*theta2n+theta2n1)/dts)*theta3+1.0/2.0*theta3
$    *(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$    *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$    *(3*theta3-4*theta3n+theta3n1))/dts-(1-1.0/4.0*theta1**2
$ +1.0/4.0*theta2**2-1.0/4.0*theta3**2)*(3.0/2.0*theta3-theta3n
$ +1.0/4.0*theta3n1)/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$ +1.0/4.0*theta3**2)**2-((-(1.0/4.0*theta1*(3*theta1-4*theta1n
$    +theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$    +1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts+1.0/2.0
$    *theta2*(3*theta2-4*theta2n+theta2n1)/dts)*(1+1.0/4.0
$    *theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(1-1.0/4.0*theta1**2
$ +1.0/4.0*theta2**2-1.0/4.0*theta3**2)*(1.0/4.0*theta1*(3*theta1
$    -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$    +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$    /dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3*theta3


 dCTdot(3,3,1)=(-(3.0/2.0*theta1-theta1n+1.0/4.0*theta1n1)/dts*(1
$      +1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0*(-(1.0/4.0
$    *theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3
$    *theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4
$    *theta3n+theta3n1))/dts+1.0/2.0*theta3*(3*theta3-4*theta3n
$ +theta3n1)/dts)*theta1+1.0/2.0*theta1*(1.0/4.0*theta1*(3*theta1
$    -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$    +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$ /dts-(1-1.0/4.0*theta1**2-1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)*(3.0/2.0
$*theta1-theta1n+1.0/4.0*theta1n1)/dts)
$/(1+1.0/4.0*theta1**2+1.0/4.0
$ *theta2**2+1.0/4.0*theta3**2)**2-((-(1.0/4.0*theta1*(3*theta1-4
$    *theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$    +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$    /dts+1.0/2.0*theta3*(3*theta3-4*theta3n+theta3n1)/dts)*(1
$+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)-(1-1.0/4.0*theta1**2
$    -1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$*(1.0/4.0*theta1*(3*theta1
$    -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$    +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$    /dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**3*theta1
```

```
 dCTdot(3,3,2)=(-(3.0/2.0*theta2-theta2n+1.0/4.0*theta2n1)/dts
$*(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0*(-(1.0/4.0
$    *theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3
$     *theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4
$     *theta3n+theta3n1))/dts+1.0/2.0*theta3*(3*theta3-4*theta3n
$  +theta3n1)/dts)*theta2+1.0/2.0*theta2*(1.0/4.0*theta1*(3*theta1
$     -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$     +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))
$ /dts-(1-1.0/4.0*theta1**2-1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)*(3.0/2.0
$     *theta2-theta2n+1.0/4.0*theta2n1)/dts)/(1
$+1.0/4.0*theta1**2+1.0/4.0
$     *theta2**2+1.0/4.0*theta3**2)**2-((
$-(1.0/4.0*theta1*(3*theta1-4
$ *theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts+1.0/2.0*theta3
$*(3*theta3-4*theta3n+theta3n1)/dts)*(1+1.0/4.0*theta1**2+1.0/4.0
$     *theta2**2+1.0/4.0*theta3**2)
$-(1-1.0/4.0*theta1**2-1.0/4.0*theta2**2+1.0/4.0
$     *theta3**2)*(1.0/4.0*theta1*(3*theta1
$-4*theta1n+theta1n1)+1.0/4.0
$  *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$    -4*theta3n+theta3n1))/dts)/(1+1.0/4.0
$*theta1**2+1.0/4.0*theta2**2+1.0/4.0
$     *theta3**2)**3*theta2

 dCTdot(3,3,3)=((-(3.0/2.0*theta3-theta3n+1.0/4.0*theta3n1)/dts
$+1.0/2.0*(3*theta3-4*theta3n+theta3n1)/dts+3.0/2.0*theta3/dts)*(1
$     +1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)+1.0/2.0*(-(1.0/4.0
$     *theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3
$     *theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4
$     *theta3n+theta3n1))/dts+1.0/2.0*theta3*(3*theta3-4*theta3n
$ +theta3n1)/dts)*theta3-1.0/2.0*theta3*(1.0/4.0*theta1*(3*theta1
$     -4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n
$     +theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts
$-(1-1.0/4.0*theta1**2-1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)*(3.0/2.0*theta3
$     -theta3n+1.0/4.0*theta3n1)/dts)/(1+1.0/4.0
$*theta1**2+1.0/4.0*theta2**2
$ +1.0/4.0*theta3**2)**2-((-(1.0/4.0*theta1*(3*theta1-4*theta1n
$     +theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$+1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts+1.0/2.0*theta3
$ *(3*theta3-4*theta3n+theta3n1)/dts)*(1+1.0/4.0*theta1**2+1.0/4.0
$*theta2**2+1.0/4.0*theta3**2)-(1
$-1.0/4.0*theta1**2-1.0/4.0*theta2**2+1.0/4.0
$*theta3**2)*(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0
$ *theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$-4*theta3n+theta3n1))/dts)/(1+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$     +1.0/4.0*theta3**2)**3*theta3

 dCTdot(1,2,1)=((3.0d0/4.0d0/dts*theta2
$+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0
```

```
$*theta2n1)/dts)*(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)
$   +1.0/2.0*((-3.0/2.0*theta3+2*theta3n
$-1.0/2.0*theta3n1)/dts+(3.0d0/4.0d0*theta1
$-theta1n+1.0/4.0*theta1n1)/dts*theta2+(3.0d0/4.0d0*theta2-theta2n
$+1.0/4.0*theta2n1)/dts*theta1)*theta1-1.0/2.0*theta2*(1.0/4.0
$   *theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$   *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$   -4*theta3n+theta3n1))/dts-(-theta3+1.0/2.0*theta2*theta1)
$   *(3.0/2.0*theta1-theta1n+1.0/4.0*theta1n1)/dts)
$/(1+1.0/4.0*theta1**2
$   +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
$-(((-3.0/2.0*theta3+2*theta3n
$ -1.0/2.0*theta3n1)/dts+(3.0d0/4.0d0*theta1
$-theta1n+1.0/4.0*theta1n1)/dts
$*theta2+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0*theta2n1)/dts*theta1)
$   *(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)-(-theta3
$   +1.0/2.0*theta2*theta1)*(1.0/4.0*theta1*(3*theta1-4*theta1n
$   +theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
$   +1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts)/(1
$+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3*theta1

 dCTdot(1,2,2)=(((3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts
$     +3.0d0/4.0d0/dts*theta1)*(1
$+1.0/4.0*theta1**2+1.0/4.0*theta2**2+1.0/4.0
$*theta3**2)+1.0/2.0*((-3.0/2.0*theta3+2*theta3n-1.0/2.0*theta3n1)
$/dts+(3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts*theta2
$+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0*theta2n1)/dts*theta1)*theta2
$ -1.0/2.0*theta1*(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)
$   +1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3
$   *(3*theta3-4*theta3n+theta3n1))/dts-(-theta3+1.0/2.0*theta2
$   *theta1)*(3.0/2.0*theta2-theta2n+1.0/4.0*theta2n1)/dts)/(1
$   +1.0/4.0*theta1**2+1.0/4.0*theta2**2
$+1.0/4.0*theta3**2)**2-(((-3.0/2.0
$   *theta3+2*theta3n-1.0/2.0*theta3n1)/dts+(3.0d0/4.0d0*theta1
$-theta1n+1.0/4.0*theta1n1)/dts*theta2+(3.0d0/4.0d0*theta2-theta2n
$   +1.0/4.0*theta2n1)/dts*theta1)*(1
$+1.0/4.0*theta1**2+1.0/4.0*theta2**2
$   +1.0/4.0*theta3**2)-(-theta3+1.0/2.0*theta2
$*theta1)*(1.0/4.0*theta1
$   *(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2*(3*theta2
$   -4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3-4*theta3n
$ +theta3n1))/dts)/(1+1.0/4.0*theta1**2
$+1.0/4.0*theta2**2+1.0/4.0*theta3**2)**3
$   *theta2

 dCTdot(1,2,3)=(-3.0/2.0/dts*(1+1.0/4.0
$*theta1**2+1.0/4.0*theta2**2
$     +1.0/4.0*theta3**2)+1.0/2.0*((
$-3.0/2.0*theta3+2*theta3n-1.0/2.0*theta3n1)
$/dts+(3.0d0/4.0d0*theta1-theta1n+1.0/4.0*theta1n1)/dts*theta2
$+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0*theta2n1)/dts*theta1)*theta3
$ +(1.0/4.0*theta1*(3*theta1-4*theta1n+theta1n1)+1.0/4.0*theta2
$   *(3*theta2-4*theta2n+theta2n1)+1.0/4.0*theta3*(3*theta3
$   -4*theta3n+theta3n1))/dts-(-theta3+1.0/2.0*theta2*theta1)
```

```
     $    *(3.0/2.0*theta3-theta3n+1.0/4.0
     $*theta3n1)/dts)/(1+1.0/4.0*theta1**2
     $    +1.0/4.0*theta2**2+1.0/4.0*theta3**2)**2
     $-(((-3.0/2.0*theta3+2*theta3n
     $ -1.0/2.0*theta3n1)/dts+(3.0d0/4.0d0*theta1-theta1n
     $+1.0/4.0*theta1n1)/dts
     $*theta2+(3.0d0/4.0d0*theta2-theta2n+1.0/4.0*theta2n1)/dts*theta1)
     $    *(1+1.0/4.0*theta1**2+1.0/4.0
     $*theta2**2+1.0/4.0*theta3**2)-(-theta3
     $    +1.0/2.0*theta2*theta1)*(1.0/4.0*theta1*(3*theta1-4*theta1n
     $    +theta1n1)+1.0/4.0*theta2*(3*theta2-4*theta2n+theta2n1)
     $    +1.0/4.0*theta3*(3*theta3-4*theta3n+theta3n1))/dts)/(1
     $    +1.0/4.0*theta1**2+1.0/4.0*theta2**2
     $+1.0/4.0*theta3**2)**3*theta3
        end if
        end


C*********************************************************************
C    OUTPUT--- Output unknown variables to different files for plot
C*********************************************************************

      Subroutine OUTPUT
      INCLUDE 'cATRc.f'
        double precision Pwhole(3*NCWM,0:NTIMERM,NBODTM),
     $Hwhole(3*NCWM,0:NTIMERM,NBODTM),
     $ Fwhole(3*NCWM,0:NTIMERM,NBODTM),Mwhole(3*NCWM,0:NTIMERM,NBODTM),
     $uwhole(3*NCWM,0:NTIMERM,NBODTM),
     $ thetawhole(3*NCWM,0:NTIMERM,NBODTM),F1whole(3,0:NTIMERM,NBODTM),
     $M1whole(3,0:NTIMERM,NBODTM),theta0whole(3,0:NTIMERM,NBODTM),
     $ uN1whole(3,0:NTIMERM,NBODTM),thetaN1whole(3,0:NTIMERM,NBODTM),
     $gamawhole(3*NCWM,0:NTIMERM,NBODTM),
     $ kapawhole(3*NCWM,0:NTIMERM,NBODTM),
     $Vwhole(3*NCWM,0:NTIMERM,NBODTM),
     $Omegawhole(3*NCWM,0:NTIMERM,NBODTM)
        common /whole/ Pwhole,Hwhole,Fwhole,Mwhole,uwhole,thetawhole,
     $F1whole,M1whole,uN1whole,thetaN1whole,gamawhole,kapawhole,
     $Vwhole,Omegawhole,theta0whole
      INTEGER i,j,ii,jj,N_BLADE
        do N_BLADE=1,NOB
       do j=0,Int
         do i=1,3
       F1whole(i,j,N_BLADE)=Xwhole(i,j,N_BLADE)
       uN1whole(i,j,N_BLADE)=Xwhole(18*NES+6+i,j,N_BLADE)
      thetaN1whole(i,j,N_BLADE)=Xwhole(18*NES+9+i,j,N_BLADE)*180/PiPi
         end do
        end do
       IF (BC.EQ.1)THEN
        do j=0,Int
        M1WHOLE(1,J,N_BLADE)=XWHOLE(4,J,N_BLADE)
        M1WHOLE(2,J,N_BLADE)=0.0D0
        M1WHOLE(3,J,N_BLADE)=0.0D0
        IF (J.EQ.0)THEN
       THETA0WHOLE(1,J,N_BLADE)=0.0D0
        ELSE
        THETA0WHOLE(1,J,N_BLADE)=PITCHANGLE(j,N_BLADE)*180/PiPi
```

```
      END IF
      THETA0WHOLE(2,J,N_BLADE)=XWHOLE(5,J,N_BLADE)*180/PiPi
      THETA0WHOLE(3,J,N_BLADE)=XWHOLE(6,J,N_BLADE)*180/PiPi
      end do
      ELSE
      do j=0,Int
      DO I=1,3
      M1WHOLE(I,J,N_BLADE)=XWHOLE(I+3,J,N_BLADE)
      THETA0WHOLE(I,J,N_BLADE)=0.0D0*180/PiPi
      END DO
      end do
      END IF
      do 10 j=0,Int
  do 30 ii=1,NES
      do 30 jj=1,3
  uwhole(3*(ii-1)+jj,j,N_BLADE)=Xwhole(18*(ii-1)+6+jj,j,N_BLADE)
  thetawhole(3*(ii-1)+jj,j,N_BLADE)=
 $ Xwhole(18*(ii-1)+9+jj,j,N_BLADE)*180/PiPi
  Fwhole(3*(ii-1)+jj,j,N_BLADE)=Xwhole(18*(ii-1)+12+jj,j,N_BLADE)
  Mwhole(3*(ii-1)+jj,j,N_BLADE)=Xwhole(18*(ii-1)+15+jj,j,N_BLADE)
  Pwhole(3*(ii-1)+jj,j,N_BLADE)=Xwhole(18*(ii-1)+18+jj,j,N_BLADE)
  Hwhole(3*(ii-1)+jj,j,N_BLADE)=Xwhole(18*(ii-1)+21+jj,j,N_BLADE)
30  continue
10  continue

  open(unit=5,file='tipdis.dat')
  do 40 i=0,Int
  write(5,*) uN1whole(1,i,N_BLADE),
 $uN1whole(2,i,N_BLADE),uN1whole(3,i,N_BLADE)
40   continue

  open(unit=6,file='tiprot.dat')
  do 50 i=0,Int
  write(6,*)thetaN1whole(1,i,N_BLADE),thetaN1whole(2,i,N_BLADE),
 $ thetaN1whole(3,i,N_BLADE)
50   continue

  open(unit=7,file='rootforce.dat')
  do 60 i=0,Int
  write(7,*)F1whole(1,i,N_BLADE),F1whole(2,i,N_BLADE)
 $,F1whole(3,i,N_BLADE)
60   continue

  open(unit=8,file='rootmom.dat')
  do 70 i=0,Int
  write(8,*)M1whole(1,i,N_BLADE),M1whole(2,i,N_BLADE),
 $ M1whole(3,i,N_BLADE)
70   continue

  open(unit=9,file='theta0.dat')
  do i=1,Int
      write(9,*)theta0whole(1,i,N_BLADE),theta0whole(2,i,N_BLADE),
 $theta0whole(3,i,N_BLADE)
  end do

  open(unit=10,file='elasticT.dat')
```

```
  do i=1,NTIME
  write(10,*)TWIST_1D(i,N_BLADE,NES),TWIST_2D(i,N_BLADE,NES)
$  ,TWIST_3D(i,N_BLADE,NES)
  end do

  open(unit=11,file='elasticD.dat')
  do i=1,NTIME
  write(11,*)DISP_1D(i,N_BLADE,NES),DISP_2D(i,N_BLADE,NES),
$DISP_3D(i,N_BLADE,NES)
  end do

  open(unit=15,file='deform.dat')
  do i=1,NES
  write(15,*)uwhole(3*(i-1)+3,INT,N_BLADE)
  end do

 open(unit=17,file='xwhole.dat')
      do j=1,int
      do i=1,18*NES+12
      write(17,*)xwhole(i,j,n_blade)
      enddo
      enddo

      end do
      END
```

```
C*********************************************************************
C   DEFORMATION OBTAINED FROM STRUCTRUE CODE IS THE TOTAL DEFORMATION
C   INCLUDING RIGID BODY MOTION AND ELASTIC DEFORMATION
C   THIS SUBROUTINE SEPERATES THE RIGID DEFORMATION AND
C   ELASTIC DEFORMATION FOR THE USE OF AERODYNAMIC CODE.
C*********************************************************************

   SUBROUTINE TRAN(IASTEP,N_BLADE)
C       IMPLICIT NONE
        INCLUDE 'cATRc.f'
        INTEGER IASTEP,N_BLADE,NELE,I,J,K

C   RIGID FEEDBACK DATA ( ROTATION AT THE HINGE )
   A_PITCHD(IASTEP,N_BLADE)=THETA0(1)
        A_FLAPD(IASTEP,N_BLADE)=-THETA0(2)
        A_LLAGD(IASTEP,N_BLADE)=THETA0(3)

C   ELASTIC FEEDBACK DATA AT THE MIDDLE OF EACH ELEMENT
C   ( = TOTAL DEFORMATION - RIGID DEFORMATION )
        DO NELE=1,NES
        TWIST_1D(IASTEP,N_BLADE,NELE)=THETA(3*(NELE-1)+1)-THETA0(1)
        TWIST_2D(IASTEP,N_BLADE,NELE)=THETA(3*(NELE-1)+2)-THETA0(2)
        TWIST_3D(IASTEP,N_BLADE,NELE)=THETA(3*(NELE-1)+3)-THETA0(3)
        DISP_1D(IASTEP,N_BLADE,NELE)=U(3*(NELE-1)+1)
  $-(RADIUS(NELE)-beamroot)*
  $   (2-COS(THETA0(2))-COS(THETA0(3)))
        DISP_2D(IASTEP,N_BLADE,NELE)=U(3*(NELE-1)+2)
  $-(RADIUS(NELE)-beamroot)*
  $   SIN(THETA0(3))
        DISP_3D(IASTEP,N_BLADE,NELE)=U(3*(NELE-1)+3)
  $+(RADIUS(NELE)-beamroot)*
  $   SIN(THETA0(2))
        END DO

C   DEFORMATION AT THE BLADE TIP POINT
   DISP_1DTIP(IASTEP,N_BLADE)=uN1(1)
  $-(RADIUS(NES+1)-beamroot)*
  $   (2-COS(THETA0(2))-COS(THETA0(3)))
   DISP_2DTIP(IASTEP,N_BLADE)=uN1(2)
  $-(RADIUS(NES+1)-beamroot)*
  $   SIN(THETA0(3))
   DISP_3DTIP(IASTEP,N_BLADE)=uN1(3)
  $+(RADIUS(NES+1)-beamroot)*
  $   SIN(THETA0(2))
        TWIST_1DTIP(IASTEP,N_BLADE)=THETAN1(1)-THETA0(1)
        TWIST_2DTIP(IASTEP,N_BLADE)=THETAN1(2)-THETA0(2)
        TWIST_3DTIP(IASTEP,N_BLADE)=THETAN1(3)-THETA0(3)
        END
```

```
C*********************************************************************
C    CALCULATE AERODYNAMIC FORCES AND MOMENT AT NODES FROM
C    THE DISTRIBUTED FORCES AND MOMENTS OBTAINED FROM AERO COMPONENT
C*********************************************************************

      SUBROUTINE FORCE_NODE
C       implicit none
      INCLUDE 'cATRc.f'
      DOUBLE PRECISION FSTRG_AD(3,NCWM,NBODTM),
     $     PMOMSTRG_AD(3,NCWM,NBODTM)
      INTEGER I,J,N_BLADE

C       Save AERODYNAMIC FORCE
       If(MESH_OFFSET.EQ.0)THEN
       DO I=1,3
       DO J=1,NES
       DO N_BLADE=1,NOB
       FSTRG_AD(I,J,N_BLADE)=FSTRG_A(I,J,N_BLADE)
       PMOMSTRG_AD(I,J,N_BLADE)=PMOMSTRG_A(I,J,N_BLADE)
       END DO
       END DO
       END DO
       ELSE

       DO I=1,3
       DO N_BLADE=1,NOB
       DO J=1,MESH_OFFSET
       FSTRG_AD(I,J,N_BLADE)=0.0D0
       PMOMSTRG_AD(I,J,N_BLADE)=0.0D0
      END DO
       DO J=1+MESH_OFFSET,NES
       FSTRG_AD(I,J,N_BLADE)=FSTRG_A(I,J-MESH_OFFSET,N_BLADE)
       PMOMSTRG_AD(I,J,N_BLADE)=PMOMSTRG_A(I,J-MESH_OFFSET,N_BLADE)
       END DO

       END DO
       END DO

       END IF

C     INTEGRATE DISTRIBUTED FORCE TO CONCENTRATED NODE FORCE
      do N_BLADE=1,NOB
      do  I=2,NES
      FA(3*(I-1)+1,N_BLADE)=FSTRG_AD(1,I-1,N_BLADE)*DL(I-1)/2.0D0
     $  + FSTRG_AD(1,I,N_BLADE)*DL(I)/2.0D0
      FA(3*(I-1)+2,N_BLADE)=FSTRG_AD(2,I-1,N_BLADE)*DL(I-1)/2.0D0
     $  + FSTRG_AD(2,I,N_BLADE)*DL(I)/2.0D0
      FA(3*(I-1)+3,N_BLADE)=FSTRG_AD(3,I-1,N_BLADE)*DL(I-1)/2.0D0
     $  + FSTRG_AD(3,I,N_BLADE)*DL(I)/2.0D0
      MA(3*(I-1)+1,N_BLADE)=PMOMSTRG_AD(1,I-1,N_BLADE)*DL(I-1)/2.0D0
     $  + PMOMSTRG_AD (1,I,N_BLADE)*DL(I)/2.0D0
      MA(3*(I-1)+2,N_BLADE)= PMOMSTRG_AD (2,I-1,N_BLADE)*DL(I-1)/2.0D0
     $  + PMOMSTRG_AD (2,I,N_BLADE)*DL(I)/2.0D0
      MA(3*(I-1)+3,N_BLADE)= PMOMSTRG_AD (3,I-1,N_BLADE)*DL(I-1)/2.0D0
     $  + PMOMSTRG_AD (3,I,N_BLADE)*DL(I)/2.0D0
      END DO
```

```
FA(1,N_BLADE)=FSTRG_AD(1, 1,N_BLADE)*DL(1)/2.0D0
FA(2,N_BLADE)=FSTRG_AD(2, 1,N_BLADE)*DL(1)/2.0D0
FA(3,N_BLADE)=FSTRG_AD(3, 1,N_BLADE)*DL(1)/2.0D0
MA(1,N_BLADE)=PMOMSTRG_AD(1, 1,N_BLADE)*DL(1)/2.0D0
MA(2,N_BLADE)= PMOMSTRG_AD (2, 1,N_BLADE)*DL(1)/2.0D0
MA(3,N_BLADE)= PMOMSTRG_AD (3, 1,N_BLADE)*DL(1)/2.0D0
FA(3*NES+1,N_BLADE)=FSTRG_AD(1, NES,N_BLADE)*DL(NES)/2.0D0
FA(3*NES+2,N_BLADE)=FSTRG_AD(2, NES,N_BLADE)*DL(NES)/2.0D0
FA(3*NES+3,N_BLADE)=FSTRG_AD(3, NES,N_BLADE)*DL(NES)/2.0D0
MA(3*NES+1,N_BLADE)=PMOMSTRG_AD(1, NES,N_BLADE)*DL(NES)/2.0D0
MA(3*NES+2,N_BLADE)= PMOMSTRG_AD(2, NES,N_BLADE)*DL(NES)/2.0D0
MA(3*NES+3,N_BLADE)= PMOMSTRG_AD (3, NES,N_BLADE)*DL(NES)/2.0D0
END DO
END
```

```
C*********************************************************************
C
C     BC-=1,Hinge boundary condition; =0,Bench condition
C     act-=1,active control
C     rotate-=1,rotating;=1 none rotation
C     NCWM--Number of elements
C     NTIMERM--Number of time step
C     L--Length of the blade
C     dt--Time step
C     dl--Length of each element
C     S---Stiffness Matrix
C     mass--Mass matrix
C     P--Linear Momenta
C     H--Angular Momenta
C     F--Internal force
C     M--Internal moment
C     u--Displacement vector
C     theta--rotation vector
C     F1--INTERNAL FORCE AT ROOT
C     M1--INTERNAL MOMENT AT ROOT
C     uN1--DISPLACEMENT AT TIP
C     tN1--ROTATION AT TIP
C     gama--strain
C     kapa--strain
C     V--Linear velocity
C     Omega--angular velocity
C     va--Initial velocity
C     wa--Initial angular velocity
C     Cab--Transformation matix from b to a
C     Cba--Transformation matix from a to b
C     C--Cab*CBa
C     CTR--Transpose of C
C     fa--AERODYNAMIC FORCE
C     ma--AERODYNAMIC MOMENT
C
C*********************************************************************
C     implicit none
    include 'cinterface.f'
    double precision PiPi
         Parameter(PiPi=3.141592654e0)
         double precision dl(NCWM)
         common /dll/ dl
         DOUBLE PRECISION RADIUS(NCWM+1)
         COMMON /R/ RADIUS
         double precision dts
         common /timestep/ dts
         double precision PM(3*NCWM),H(3*NCWM),F(3*NCWM),M(3*NCWM),
     $          u(3*NCWM),theta(3*NCWM),
     $          F1(3),M1(3),theta0(3),uN1(3),thetaN1(3)
         common /variables/ PM,H,F,M,u,theta,F1,M1,theta0,uN1,thetaN1
         double precision  gama(3*NCWM),kapa(3*NCWM)
         common /strain/ gama,kapa
         double precision  V(3*NCWM),Omega(3*NCWM)
         common /speed/ V,Omega
         double precision  va(3*NCWM,NBODTM),wa(3*NCWM,NBODTM)
         common /initial_speed/ va,wa
```

```
      double precision  Cab(3,3,NCWM,NBODTM),Cba(3,3,NCWM,NBODTM)
      common /ab/       Cab,Cba
      double precision  CTR(3,3,NCWM)
      common /CTtheta/          CTR
      double precision  CTdotwhole1(3,3,NCWM),Pdotwhole1(3,NCWM),
$              Hdotwhole1(3,NCWM)
      common /dotwhole1/ CTdotwhole1,Pdotwhole1,Hdotwhole1
      double precision  CTdotwhole2(3,3,NCWM),Pdotwhole2(3,NCWM),
$              Hdotwhole2(3,NCWM)
      common /dotwhole2/ CTdotwhole2,Pdotwhole2,Hdotwhole2
      double precision  fa(3*NCWM+3,NBODTM),ma(3*NCWM+3,NBODTM)
      common /force/ fa,ma
      double precision e(3,3)
      common /ee/       e
      double precision Xwhole(18*NCWM+12,0:NTIMERM,NBODTM)
      common /X_whole/ Xwhole
      double precision X(18*NCWM+12,NBODTM)
      common /X_X/ X
      double precision dgamadF(3,3,NCWM),dgamadM(3,3,NCWM),
$   dkapadF(3,3,NCWM),
$      dkapadM(3,3,NCWM),dVdP(3,3,NCWM), dVdH(3,3,NCWM),
$dOmegadP(3,3,NCWM),dOmegadH(3,3,NCWM)
 common /dstraindF/  dgamadF,dgamadM,dkapadF,
$      dkapadM,dVdP, dVdH,dOmegadP,dOmegadH
      double precision Factive(3*NCWM,NBODTM),Mactive(3*NCWM,NBODTM)
 common /Active/ Factive,Mactive
 double precision CTCab(3,3,NCWM)
      common /CT_Cab/CTCab
      double precision CTdot(3,3),Pdot(3),Hdot(3),udot(3),thetadot(3)
      common /dot/ CTdot,Pdot,Hdot,udot,thetadot
 double precision un_1(3*NCWM),Pn1(3*NCWM),Hn1(3*NCWM),
$      un(3*NCWM),Pn(3*NCWM),Hn(3*NCWM)
 common /old/ un_1,Pn1,Hn1,un,Pn,Hn
      double precision  thetan_1(3*NCWM),thetan(3*NCWM)
      common /n_1theta/ thetan_1,thetan
 double precision w
 common /rotatespeed/ w
 integer is
 common /ispeed/ is
 INTEGER BC,ACT
      COMMON /BOUNDARYCONDITION/ BC,ACT
      INTEGER NES,INT,NOB
      COMMON /CONSTANTS/ NES,INT,NOB
      DOUBLE PRECISION PRETWIST(NCWM)
 COMMON /TWIST/ PRETWIST
      DOUBLE PRECISION PITCHANGLE(NTIMERM,NBODTM),PCON,PCOS,PSIN,POMEGA
      COMMON  /CONTROLPITCH/ PITCHANGLE,PCON,PCOS,PSIN,POMEGA
      double precision beamroot
      common /lengthroot/beamroot
 double precision twistactive(NTIMERM)
 common /activevector/ twistactive
```

Bibliography

[1]Y. H. Yu, B. Gmelin, W. Splettstoesser, J. J. Philippe, J. Prieur and T. F. Brooks. Reduction of helicopter blade-vortex interaction noise by active rotor control technology. *Progress of Aerospace Science*, 33:647-687, 1997.

[2] P. C Chen, J. D. Baeder, R. A. D. Evans and J. Niemczuk. Blade-vortex interaction noise reduction with active twist smart rotor technology. *Smart Materials and Structures*, 10:77-85, 2001.

[3] Y. H. Yu. Miss-distance for rotor blade-vortex interaction noise reduction. AIAA paper No. 96-1738.

[4] S. G. Voutsinas and D. G. Triantos. High-resolution aerodynamic analysis of full helicopter configurations. Proceedings of the *25th European Rotorcraft Forum*, paper C11, September 14-16, 1999.

[5] D. H. Hodges. Mixed variational formulation based on exact intrinsic equations for dynamics of moving beams. *International Journal of Solids and Structures*, 26-11, pp. 1253-1273, 1990.

[6] C. E. S. Cesnik, S. -J. Shin, and M. L. Wilbur. Dynamic response of active twist rotor blades. *Smart Materials and Structures*, 10:62-76, 2001.

[7] X. Shang. *Aeroelastic stability of composite hingeless rotors with finite-state unsteady aerodynamics*. PhD dissertation, Georgia Institute of Technology, 1995.

[8] C. E. S. Cesnik and S. -J. Shin. On the modeling of active helicopter blades. *International Journal of Solids and Structures*, 38(10-13):1765-1789, 2001.

[9] O. A. Bauchau and T. Joo. Computational schemes for non-linear elasto-dynamics. *International Journal for Numerical Methods in Engineering*, 45: 693-719, 1999.

[10] D. A. Peters and C. J. He. Finite state induced flow models part II: three dimensional rotor disk, *Journal of Aircraft*, 32(2): 323-333, 1995.

[11] O. A. Bauchau, C. L. Bottasso, and Y. G. Nikishkow. Modeling rotorcraft dynamics with finite element multibody procedures. *Mathematical and Computer Modelling*. 33: 1113-1137, 2001.

[12] S. -J. Shin. *Integral twist actuation of helicopter rotor blades for vibration reduction*. PhD dissertation, Massachusetts Institute of Technology, 2001.

[13] S. -J. Shin and C. E. S. Cesnik. Design, manufacturing and testing of an active twist rotor. AMSL #99-3, Massachusetts Institute of Technology, June 1999.

[14] K. A. Hoffmann and S. T. Chiang. *Computational fluid dynamics for engineering, Volume I.* A Publication of Engineering Education System, Wichita, Kansas, 1993.

[15] W. H. Press. *Numerical recipes in FORTRAN 77 and FORTRAN 90 [computer file]: the art of scientific and parallel computing*. Cambridge University Press, 1996.

[16] K. J. Bathe. *Finite element procedures*. Prentice Hall, Upper Saddle River, New Jersey 07458, 1996.

[17] O. A. Bauchau and N. J. Theron. energy decaying scheme for non-linear beam models. *Computer Methods in Applied Mechanics and Engineering*. 134:37-56, 1996.

[18] O. A. Bauchau, G. L. Bottasso, and Y. G. Nikishkov. Modeling rotorcraft dynamics with finite element multibody procedures. *Mathematical and Computer Modelling*. 33: 1113-1137, 2001.

[19] C. E. S. Cesnik and E. Brown. Modeling of high aspect ratio active flexible wings for roll control. *Proceedings of the 43rd Structures, Structural Dynamics, and Material Conference*, Denver, Colorado, 2002, AIAA Paper No. 2002-1719.

[20] P. Minguet. *Static and dynamic behavior of composite helicopter rotor blades under large deflection.* PhD dissertation, Massachusetts Institute of Technology, 1989.

[21] A. R. Atilgan, D. H. Hodges, M. V. Fulton, and C. E. S. Cesnik. Application of the variational-asymptotical method to static and dynamic behavior of elastic beams. AIAA Paper No. 91-1026-CP.

[22] S. P. Timoshenko and J. M. Gere. *Theory of elasticity*. McGraw-Hill, 1961.

[23] J. D. Anderson. *Fundamentals of aerodynamics*, Third Edition, McGraw Hill, 2001.

[24] D. Opoku and F. Nitzsche. Personal communications, 2001.