

DCAP (Dynamics and Control Analysis Package) an Effective Tool for Modelling and Simulating of Coupled Controlled Rigid Flexible Structure in Space Environment

Stefano Portigliotti^a, Massimo Dumontela^a,
Gianluigi Baldesi^b, Donato Sciacovelli^b

^aGuidance Navigation and Control Section,
System Avionics and Operations Division, Alenia Spazio SpA, Torino, Italy

^bStructures Section, Thermal and Structures Division,
ESA/ESTEC, Noordwijk, The Netherlands

Abstract

The paper presents the software DCAP (Dynamics and Control Analysis Package): a suite of fast, effective computer programs that provides the user with capability to model, simulate and analyze the dynamics and control performances of coupled rigid and flexible (NASTRAN interfaced) structural systems subjected to possibly time varying structural characteristics and space environment loads.

It uses the formulation for the dynamics of multi-rigid/flexible-body systems based on Order(n). This avoids the explicit computation of a system mass matrix and its inversion, and it results in a minimum-dimension formulation exhibiting close to Order(n) behavior, n being the number of system degrees of freedom. A dedicated symbolic manipulation pre-processor is further used in the coding optimization.

The modeling capability is completed with the possibility of user-defined software, allowing for modeling of specific control feature not directly included in the dynamic package's library. For the latter control modeling, a new interface is being developed allowing to describe the user control directly by Matlab/Simulink blocks.

An application example of 3D trajectory flight of a first stage launch vehicle with flexible time varying structural properties is presented.

Brief overview of Dynamics and Control Analysis Package

DCAP is a suite of fast, effective computer programs that provides the user with a powerful tool for designing and verifying the dynamics and control performance of coupled rigid and flexible structural systems. The software modules that it contains can be grouped into four general categories: Pre-Processing, Processing, Post-Processing and Utility including the MATDCAP I/F to Matlab/Simulink, [1]. Communication between the modules is achieved via the dedicated file structure. The package provides the user with an outstanding capability to model, simulate and analyze a complex multi-body system. The latter can be connected in open- and closed-loop topologies, where relative motion is defined through 'hinges'. Each

hinge allows from zero to six relative degrees of freedom, as it can be free, locked or constrained to pre-defined motion.

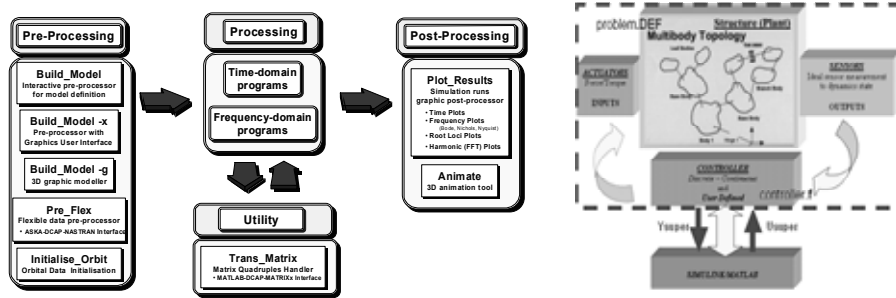


Figure 1: DCAP software modules

Transition modes between hinge states can be implemented as well, allowing locking and releasing commanded through monitoring of suitable dynamic states during system dynamic evolution.

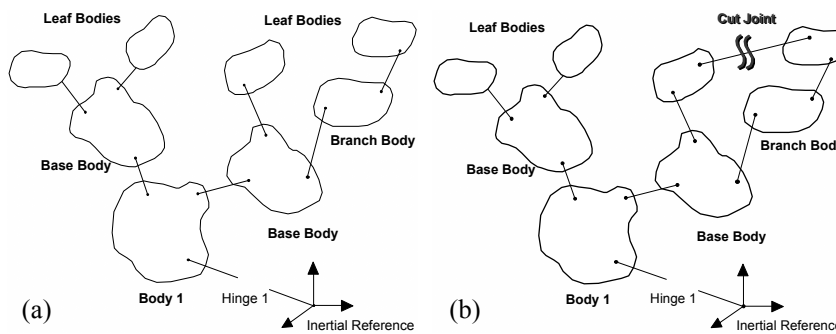


Figure 2: Open-loop (a) and closed-loop (b) tree topologies

From a general viewpoint, the DCAP model is the combination of four components:

- **'structure'** or **'plant'**: the multi- body topology itself
- **'sensors'**: the feature that allows the extraction of system motion information for performance monitoring
- **'actuators'**: the feature through which forces and torques are applied to the structure
- **'controller'**: the part of the system driving the multi-body structure in order to achieve the allocated objective.

DCAP capabilities can be summarized to include: Rigid and Flexible Body Chains, Open- and Closed- Loop Topology, Six Degree-of-Freedom Hinges, Large-Angle Rotations, Linear and Non-linear Time Domain Simulations, Order (N) Algorithm, Symbolic Code Generation, Orbital Environment, Built-in Sensors and Actuators,

Non-linear Devices (including Coulomb Dampers), Numerical Linearization, Continuous/Discrete/User Controllers (Transfer Function, State Space, Block Diagram), File Interface to MATLAB, File Interface to NASTRAN, Frequency Response, Interactive Pre- and Post-processing and MATDCAP code I/F to Matlab/Simulink.

The core of the package consists of the computational modules, particularly oriented towards the simulation of the non-linear dynamics of multi-body systems. In addition, programs for numerical linearization, matrix representation manipulation (state space) and system combination allow linear problems in both the time and frequency domains to be handled.

Interfaces to popular finite-element software packages such as NASTRAN (MSC and COSMIC) and ASKA, are available and give a direct capability to define complex flexible structures. Similar interfaces are provided to control design packages like MATLAB/Simulink. The modeling capability is completed with the possibility of user-defined software integration, allowing for the use of specific features (sensor and actuator dynamics, user controller, etc.) not directly included in the package's library.

The user friendliness of DCAP is significantly increased using Matlab. In order to achieve this challenging target a Matlab S-function has been created. DCAP allows for code generation both in FORTRAN and in C. These two options are both compatible with coding the S-functions in MATLAB Simulink. Consistently with availability in time of the DCAP upgrades, the FORTRAN I/F option is currently implemented in first instance for MATDCAP. Similar I/F in using C code is planned to be in a subsequent phase where relative merits could be envisaged.

This s-function can help the user to design the system, in fact, used like a Simulink block, it can be linked directly to any other blocks. This makes the modeling of the control part much easier and efficient.

To let the user the possibility to decide which integrators of DCAP or of MATLAB Simulink would have to take control of the simulation, there have been developed two different interfaces:

- I. The MATLAB Simulink integrator drives the simulation. In this case the S-function is limited to the calculation of the time derivatives, it interfaces sensors and actuators in the continuous and possibly in the sampled domain. However under the constraints of being driven by the timing imposed by the MATLAB Simulink scheme. Consequently special features of the DCAP requiring adaptation of the integration step, such as locking, stiction, could not be run in this scheme. The scheme is run with constant integration step.
- II. The latest one gives the user the possibility to have independent integrators for the MATLAB Simulink model and for the DCAP model This case allows for the two models to exchange information at discrete time instants only. Special features of the DCAP can be used without problem.

The analysis of results is supported by a specific plotting package that can handle time-domain plots, harmonic analysis (direct and inverse FFT), and Bode, Nichols,

Nyquist and Root locus charts. Performance can also be inspected via a powerful 3D animation tool, which helps the user to visualize the simulation results.

The DCAP development has been done on VAX/VMS computers, with later extension to UNIX (Sun, Silicon Graphics) workstations, recent adaptation to Linux machines and on Windows (MATDCAP) machines. The software package is complemented by a full set of manuals (User, Theory, Demonstration and Installation).

Dynamics formulation

Introduction

Early approaches to the dynamics formulation for multibody systems led to the equations of motion, for open-loop tree topologies, of the form

$$M \ddot{\underline{q}} = \underline{F} \quad (1)$$

where, M is an $n \times n$ mass matrix, $\underline{q} = [q_1 \ q_2 \ \dots \ q_n]^T$ is an $n \times 1$ column matrix representing the generalized coordinates and F is the column matrix containing the contributions from centrifugal, Coriolis, gravitational and external forces. For a numerical simulation of such a system, the mass matrix must be inverted. Since the inversion of an $n \times n$ matrix involves operations of the Order(n^3), this is called an Order(n^3) approach. As the number of degrees of freedom increases, this matrix inversion, for every integration step, becomes computationally expensive. Thus, researchers have sought methodologies to circumvent the mass matrix inversion, to improve computational efficiency. The need for computational efficiency first arose in the area of robotics, where efficient controller designs incorporating the system dynamics were sought. The research into improvements in formulations that increase computational speed resulted in - what are today called - Order(n) algorithms. The reason for this nomenclature is that the computational burden in these schemes increases only linearly with n .

The equations of motion are derived using Kane's method of generalized speeds [8,9] for the definition of dynamics with respect to active degrees of freedom (dofs). The method consists essentially of two parts: a frontal part, in which the inertia and active forces associated with each body are shifted to its inboard body and this process repeated until the core body is reached, and a back substitution part in which all the accelerations are obtained in terms of the core body accelerations and all the external forces acting on the system.

A multibody system in a topological tree is shown in Figure 2. Body 1 is an arbitrarily selected reference body assumed to be connected to an imaginary inertially fixed body, numbered 0. An accounting system is developed that supplies a unique procedure for writing the kinematics of any body in the system. At the heart of this system is the definition of body pairs j and $L(j)$. For Body j , $L(j)$ is the adjacent body leading inward to Body 0 (or to the core body, Body 1). Body $L(j)$ is then defined as the body directly inboard of Body j . Note that, since tree topologies are considered, a body can have more than one outboard body.

With this definition of j and $L(j)$, the remaining system topology variables used for the accounting procedure are defined as follows. Leaf bodies are bodies with no bodies directly outboard. Branches are defined as a set of bodies making up a chain topology. Branch bodies are bodies in a branch, and thus contain outboard as well as inboard bodies. The system configuration is then completed by defining the system degrees of freedom (DOF).

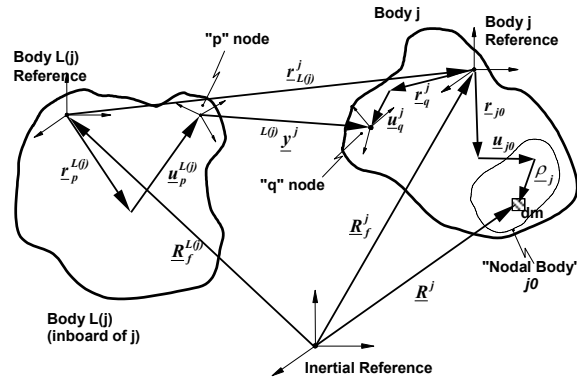


Figure 3: Body j - $L(j)$ pair

These are categorized into rigid body and elastic degrees of freedom. The former are defined through the definition of joints that permit relative motion, and the latter are defined using the assumed modes method. A joint, shown in Figure 3, is defined between a pair of material points, one on each of the adjoining bodies j and $L(j)$. The rigid-body DOF are then characterized by the relative translation and rotation of two sets of reference axes located at the two nodes, q_j on body j and p_j on the body inboard of body j - i.e., on $L(j)$, making up the j th joint. These joints can have up to six degrees of freedom.

The following symbols are used to describe the system configuration:

NB = Number of bodies

NT_j = Number of translational DOF at j^{th} hinge

NR_j = Number of rotational DOF at j^{th} hinge

NM_j = Number of deformational DOF of Body j

NS = Total number of degrees of freedom of the tree

Formulation of equations of motion

The equations of motion are derived via Kane's method. The formulation and the corresponding solution algorithm are based on the kinematic relationships between body pairs j and $L(j)$. Referring to Figure 3, we proceed as follows.

The vector locating an elemental mass dm of Body j , by a recursive form in the inertial frame, is given by

$$\underline{R}_j = \underline{R}_f^{L(j)} + \underline{r}_p^{L(j)} + \underline{u}_p^{L(j)} + \sum_{l=1}^{L(j)} \underline{y}^j - (\underline{r}_q^j + \underline{u}_q^j) + \underline{r}^j + \underline{u}^j \quad (2)$$

where \underline{r}^j is a vector that defines the undeformed configuration of the elemental mass dm in \mathbf{b}^j reference frame, and $\underline{u}^j = \sum_{l=1}^{NM_j} \phi_l^j(\underline{r}^j) \eta_l^j(t)$ represents the elastic deformation (vector) experienced by dm as the sum of the product of a set of assumed mode shape vectors $\phi_l^j(\underline{r}^j)$ and their time-varying amplitudes $\eta_l^j(t)$.

Using a set of generalized speeds W_1, \dots, W_{NS} (time derivatives of coordinates characterizing the configuration, [9] which represent the DOF of the system, the $\dot{\underline{R}}_j$ can always be written as:

$$\dot{\underline{R}}_j = \sum_{k=1}^{NS} \underline{V}_k^j W_k + \underline{V}_t^j \quad (3)$$

Where, \underline{V}_k^j are the coefficients of the generalized speeds, and \underline{V}_k^j and \underline{V}_t^j are functions of the NS generalized coordinates and time
Newton's law for the tree structure can be written as

$$\sum_{k=1}^{NB} \int_{B_j} \underline{V}_k^j \cdot (d\underline{f} - \ddot{\underline{R}}_j dm) + 0, \quad k = 1, 2, \dots, NS \quad (4)$$

where, $d\underline{f}$ is the force on the differential element and $\ddot{\underline{R}}^j$ is the inertial acceleration of dm .

The system equations of motion (Equation (4)) results in an NS×NS mass matrix. Thus, during numerical simulation the above algorithm requires the inversion of an NS×NS mass matrix, implying O(n³) operations over a reduced set of active dofs.

The steps towards achievement of an Order(n) behavior are described in [4,5,6,7], with DCAP implementation details in [10] and summarized in the following section.

Order(n) solution algorithm

The solution algorithm, presented in details in [10], can be thought of as consisting of two basic parts - a "frontal" part and a "back substitution" part, which are based on recursive formulation of the body j - $L(j)$ kinematics.

The displacement of body- j reference can be expressed as

$$\underline{\mathbf{R}}_r^j = \underline{\mathbf{R}}_r^{L(j)} + \left(\underline{\mathbf{r}}_p^{L(j)} + \underline{\mathbf{u}}_p^{L(j)} \right) + {}^{L(j)}\underline{\mathbf{y}}^j - \left(\underline{\mathbf{r}}_q^j + \underline{\mathbf{u}}_q^j \right) \quad (5)$$

and differentiated to

$$\dot{\underline{\mathbf{R}}}_j = \dot{\underline{\mathbf{R}}}_r^j + \underline{\omega}_j \times \left(\underline{\mathbf{r}}_{j0} + \underline{\mathbf{u}}_{j0} \right) + \dot{\underline{\mathbf{u}}}_{j0} + \underline{\omega}_0^j \times \underline{\rho}_j \quad (6)$$

where the open dot denotes differentiation with respect to time in Body-j frame and

$$\underline{\omega}_j = \underline{\omega}_{L(j)} + \underline{\mathbf{u}}_p^{\prime L(j)} + {}^{L(j)}\underline{\omega}^j + \underline{\mathbf{u}}_q^{\prime j} \quad (7)$$

$$\underline{\omega}_0^j = \underline{\omega}_j + \underline{\mathbf{u}}_{j0}^{\prime}$$

where the prime identifies the rotation due to flexibility, as detailed in later sections.

With the generalized velocities for Body-j defined as

$$\{\dot{\mathbf{q}}_j\}^T = \left\{ \left[\dot{y}_1^j \dots \dot{y}_{NT_j}^j \right] \left[\dot{\theta}_1^j \dots \dot{\theta}_{NR_j}^j \right] \left[\dot{\eta}_1^j \dots \dot{\eta}_{NM_j}^j \right] \right\} \quad (8)$$

for the j^{th} hinge NT_j translational and NR_j rotational (Euler angle) dofs and j^{th} body modal coordinates η^j , the expression of (3) becomes

$$\frac{\partial \dot{\underline{\mathbf{R}}}_j}{\partial \dot{\mathbf{q}}_k} = \frac{\partial \dot{\underline{\mathbf{R}}}_r^j}{\partial \dot{\mathbf{q}}_k} + \frac{\partial \underline{\omega}^j}{\partial \dot{\mathbf{q}}_k} \times \left(\underline{\mathbf{r}}_{j0} + \underline{\mathbf{u}}_{j0} + \underline{\rho}_j \right) + \frac{\partial \dot{\underline{\mathbf{u}}}_{j0}}{\partial \dot{\mathbf{q}}_k} + \frac{\partial \underline{\omega}_0^j}{\partial \dot{\mathbf{q}}_k} \times \underline{\rho}_j \quad (9)$$

or

$$\frac{\partial \dot{\underline{\mathbf{R}}}_j}{\partial \dot{\mathbf{q}}_k} = \underline{\mathbf{V}}_k^{f_j} + \underline{\omega}_k^j \times \left(\underline{\mathbf{r}}_{j0} + \underline{\mathbf{u}}_{j0} + \underline{\rho}_{j0} \right) + \underline{\mathbf{V}}_k^{\dot{\eta}_j} \quad (10)$$

With the coefficients of generalized speeds defined as

$$\underline{\mathbf{V}}_k^{f_j} = \frac{\partial \dot{\underline{\mathbf{R}}}_r^j}{\partial \dot{y}_k^j} \quad \underline{\omega}_k^j = \frac{\partial \underline{\omega}^j}{\partial \dot{\theta}_k^j} \quad \underline{\mathbf{V}}_k^{\dot{\eta}_j} = \frac{\partial \dot{\underline{\mathbf{u}}}_{j0}}{\partial \dot{\eta}_k^j} + \frac{\partial \underline{\omega}_0^j}{\partial \dot{\eta}_k^j} \times \underline{\rho}_j \quad (11)$$

The generalized inertial forces can then be expressed as

$$\underline{f}_{-T_j}^* = \int_{b_j} \underline{\ddot{\mathbf{R}}}_j dm \quad \underline{f}_{-R_j}^* = \int_{b_j} (\underline{\mathbf{r}}_j + \underline{\mathbf{u}}_j) \times \underline{\ddot{\mathbf{R}}}_j dm \quad \left\{ \underline{f}_{\eta_j}^* \right\}_k = \int_{b_j} \underline{V}_{-k}^{\dot{\eta}_j} \cdot \underline{\ddot{\mathbf{R}}}_j dm \quad (12)$$

And equivalently the generalized active forces determined by dotting the active forces $d\mathbf{f}$ and torques $d\mathbf{\tau}$ to coefficients of generalized speeds

$$\underline{f}_{-T_j} = \int_{b_j} \underline{V}_{-j}^{f_j} \cdot d\mathbf{f}_j \quad \underline{f}_{-R_j} = \int_{b_j} \underline{\omega}_k^j \cdot (d\mathbf{\tau}_j + (\underline{\mathbf{r}}_j + \underline{\mathbf{u}}_j) \times d\mathbf{f}_j) \quad \left\{ \underline{f}_{\eta_j} \right\}_k = \int_{b_j} \underline{V}_{-k}^{\dot{\eta}_j} \cdot d\mathbf{f}_j \quad (13)$$

The above expressions for the inertial forces will be function of $\{ \underline{\ddot{\mathbf{R}}}_{L(j)}^j \}, \{ \dot{\underline{\omega}}_{L(j)}^j \}, \{ \ddot{y}^j \}, \{ \ddot{\theta}^j \}, \{ \ddot{\eta}^j \}$.

The frontal part consists of the steps needed to perform the elimination of a body's modal $\{\ddot{\eta}^j\}$ and relative joint degrees of freedom $\{\ddot{y}^j\}, \{\ddot{\theta}^j\}$ in terms of its inboard body motions $\{\ddot{\underline{R}}_{L(j)}^j\}, \{\dot{\underline{\omega}}_{L(j)}^j\}$ and known external forces. Once performed the step for a leaf body- j , the obtained equations are “shifted” to inner body- $L(j)$ reference obtaining equations “augmented” due to the contribution of outboard body- j . The process is repeated until the core Body-1 is reached, for all branches, and augmented dynamic equations are then solved with respect to $\{\ddot{y}^l\}, \{\ddot{\theta}^l\}$ dofs. Note that, for a core body, $\ddot{\underline{R}}_{L(l)}^l$ and $\dot{\underline{\omega}}_{L(l)}^l$ are null vectors.

The back substitution part consists of indeed “back-substituting” along the tree topology and determining the body/joint- j DOF accelerations in terms of the motion variables of its inboard body- $L(j)$.

The solution algorithm, which is demonstrated to behave as Order(n), can then be summarized as follows:

Step 1: Define the system topology and initialize the joint variables and their rates.

Step 2: Compute all kinematic relationship and generalized inertial and active force quantities.

Step 3: Frontal Part:

- a) For all leaf bodies in the branch, perform the modal and joint DOF elimination's. Setup the inertia forces of body j and the active forces.
- b) For a generic branch body, first obtain the inertia forces of all of its outboard bodies in terms of the current branch body and its inboard body accelerations. Follow steps 2 and 3a, except that the contribution from its outboard bodies must be considered for the inertia and active forces, respectively. A generic branch body may have more than one outboard body and hence the order in which these elimination's are carried out is important.

Complete steps 2-3 for all the bodies in the system until the core body is reached.

Step 4: Solve for the accelerations associated with the augmented core body DOF in terms of all the active forces on the system.

Step 5: Assemble the system state vector derivatives for numerical integration, including additional states coming from user code

Step 6: Back Substitution: Working from Body 1 outward, solve for all the body.

Note that, in the present algorithm the system mass matrix is never explicitly inverted, thus avoiding the need for $O(n^3)$ operations. The computational burden increases only linearly as the number of bodies n increases, because the present algorithm requires computations of Order(n).

The Order(n) dynamics formulation is extended to closed-loop topologies (Figure 2.b). The method consists of rendering the system open-loop by introducing a cut-joint in the system. In addition to the active forces present in the system, the (unknown) constraint forces and torques introduced by the cut-joint must be accounted for. These unknowns will be eliminated by using the compatibility conditions at the cut-joint. Thus, this procedure will involve two frontal and two back substitution steps for every closed-loop in the system. Details on DCAP Order(n) formulation for closed-loop topologies is given in [10].

Symbolic processing

Symbolic processing of the equations of motion can result in a substantially more efficient simulation. This increase in efficiency is achieved through simplifications that are possible because of special configuration characteristics, as well as arithmetic and algebraic simplifications.

A schematic of the symbolic processing and simulation modules, along with their relationship with the DCAP package, is shown in Figure 4. They receive their input from three sources:

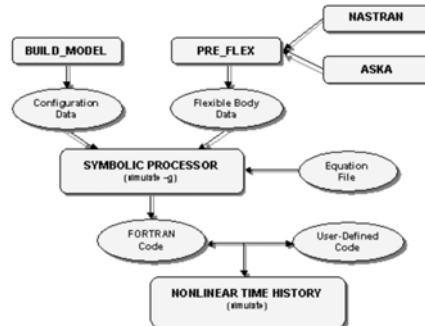


Figure 4: DCAP Simulation Context diagram

- a configuration data file which describes the multi-body system being simulated, its topology and properties, generated through DCAP command line or Graphic User Interface modules
- a flexible-body data set that contains data relating to the flexibility properties of each flexible body in the system, obtained from dedicated interface to general finite element codes or generated internally for simplified flexible body types (see later section)
- an equation file containing the templates of the equations of motion of a generic multi-body system, implementing the formal coding of all the kinematic, frontal and back substitution equations consistent with the formalism outlined in previous section

The symbolic processor, coded in C-language, produces in output a set of FORTRAN source files containing the implementation of the specific set of equations of motion that are applicable to the multi-body configuration defined. This source code is then compiled and linked with the simulation library to generate the executable module. An additional set of user-defined routines is conceived in DCAP in terms of user-continuous-controller, user discrete-controller or user-function generator. The user-controllers implementation is revisited latter in this paper hen describing the Matlab interface in MATDCAP section.

Flexible bodies

The modal formulation used in DCAP for flexible bodies depends on the definition of assumed mode shape vectors $\underline{\phi}_i^j(\underline{r}^j)$ being multiplied by time-varying amplitudes $\eta_i^j(t)$ for i^{th} mode. The local deformation is then expressed, conforming the “nodal-body2” notation shown in Figure 3, with flexible displacement and rotation

$$\underline{u}_{j0} = \sum_{i=1}^{NM_j} \underline{\Phi}_i^j(\underline{r}_{j0}) \eta_i^j \quad \underline{u}'_{j0} = \sum_{i=1}^{NM_j} \underline{\Phi}'_i^j(\underline{r}_{j0}) \eta_i^j \quad (14)$$

The modal shapes $\underline{\phi}_i^j$ and modal slopes $\underline{\phi}'_i^j$ can be either obtained from popular finite element codes, or expressed analytically for simple flexible body configuration such as beams or cables. Modal representation allows the definition of a mass matrix for body-j that is configurable up to 2nd order nonlinear dependency on modal coordinates.

$$\begin{aligned} [M^j] = & \begin{bmatrix} [M_R^j] & [L^j] \\ [L^j]^T & [m^j] \end{bmatrix} + \sum_{i=1}^{NM} \begin{bmatrix} [M_i^j + N_i^j] & [Y_i^j] \\ [Y_i^j]^T & [0] \end{bmatrix} \eta_i^j + \\ & + \sum_{i=1}^{NM_j} \sum_{k=1}^{NM_j} \begin{bmatrix} [P_{ik}^j] & [0] \\ [0]^T & [0] \end{bmatrix} \eta_i^j \eta_k^j \end{aligned} \quad (15)$$

With the sub-matrix partitions applying for the 6×6 rigid body components identified by the rigid body mass matrix $[M_R^j]$, the $NM_j \times NM_j$ modal mass matrix $[m^j]$. At zero order, the coupling with flexible *dofs* is defined through the modal participations $[L^j]$ sized $6 \times NM_j$, defined with respect to body reference as

$$[L^j] = \begin{bmatrix} \int_{b_j} [\underline{r}^j \times \underline{\phi}_1^j \dots \underline{r}^j \times \underline{\phi}_{NM_j}^j] dm \\ \int_{b_j} [\underline{\phi}_1^j \dots \underline{\phi}_{NM_j}^j] dm \end{bmatrix} \quad (16)$$

with the integration defined at finite element code level. It should be noticed that in DCAP formulation, the rotational terms are located in row/column 1÷3, the translational in 4÷6.

At first order, the $[M_i^j + N_i^j]$ 6×6 matrices allow the implementation of first order dependency of the center of mass and inertia tensor on deformation η_i^j , while the $[Y_i^j]$ 6× NM_j matrices gives the first order change in modal participations. With the 3×3 partitioning

$$[\mathbf{M}_i^j + \mathbf{N}_i^j] = \begin{bmatrix} -\int_{b_j} (\tilde{\phi}_i^j \tilde{r}^j + \tilde{r}^j \tilde{\phi}_i^j) dm & -\int_{b_j} \tilde{\phi}_i^j dm \\ \int_{b_j} \tilde{\phi}_i^j dm & 0 \end{bmatrix} \quad (17)$$

and

$$[\mathbf{Y}_i^j] = \begin{bmatrix} \int_{b_j} [\underline{\phi}_k^j \times \underline{\phi}_i^j \dots \underline{\phi}_k^j \times \underline{\phi}_{NM_j}^j] dm \\ 0 \end{bmatrix} \quad (18)$$

At second order, the $[\mathbf{P}_{ik}^j]$ matrices allow the implementation of second order dependency of the inertia tensor on deformations $\underline{\eta}_i^j \cdot \underline{\eta}_k^j$.

$$[\mathbf{P}_{ik}^j] = \begin{bmatrix} -\int_{b_j} (\tilde{\phi}_i^j \tilde{\phi}_k^j) dm & 0 \\ 0 & 0 \end{bmatrix} \quad (19)$$

It is worth pointing out that selection of modal basis is left to the user definition of the constraints in the finite element model. In the general approach, selection of normal modes of vibration allows implementation of diagonal modal mass $[m^j]$ and modal stiffness $[k^j]$ matrices, with the former selectable to identity. The general purpose "Alter" routines provided with DCAP package allow direct computation within NASTRAN of the required modal integrals to be handled by *pre_flex* flexible data pre-processor.

As a very useful extension, Craig-Bampton approaches are also implemented in the interface routines. The fixed-interface normal modes basis is augmented with "static modes" (unit displacement at interface *dofs*), which allow usage of reduced modal bases for improved convergence when redundant constraints are identified for the flexible body in the tree topology. Clearly, for Craig-Bampton approaches the modal mass matrix will be no longer diagonal.

Properties of the eigenfunctions of a fixed-free beam

The assembly of modal integrals starting from finite element models allow the integration in DCAP models of complex flexible bodies. Nevertheless, for simple shapes such as beams and cables, the modal integrals can be obtained internally with implementation of analytical definition of mode shapes.

The boundary value problem associated with a uniform beam with fixed-free end conditions is defined by

$$EIu''''(x) - \lambda\mu u(x) = 0 \quad (20)$$

with the associated boundary conditions, $u'(0)=0$ and $Elu''''(L)=0$ where, EI and μ are the stiffness and mass per unit length, respectively, and $u(x)$ is the elastic displacement of a point along the beam with spatial coordinate x . Primes denote differentiation with respect to x .

By solving the eigenvalue problem, the characteristic equation is obtained as:

$$\cos\beta L \cdot \cosh\beta L = -1 \quad (21)$$

and, the eigenfunctions associated with each of the solutions of the characteristic equation are given by

$$\phi_r(x) = \frac{1}{\sqrt{\mu L}} \left[(\cos\beta_r x - \cosh\beta_r x) + A_r (\sin\beta_r x - \sinh\beta_r x) \right] \quad (22)$$

Where the r^{th} eigenvalue is defined as

$$\beta^4 = \frac{\lambda \mu}{EI} \quad (23)$$

and

$$A_r = \frac{\sin\beta_r L - \sinh\beta_r L}{\cos\beta_r L + \cosh\beta_r L} \quad (24)$$

and the eigenfunctions are unit modal mass normalized such that

$$(\phi_r(x), \mu \phi_r(x)) \equiv \int_0^L \mu \phi_r^2(x) dx = 1 \quad (25)$$

The vibration frequencies are obtained from the relation, $\omega^2 = \lambda$. An important observation is in order with regards to obtaining the frequencies of fixed-free beams with different lengths, which is the case when a beam with a prismatic end condition is travelling while the joint itself remains stationary. The characteristic equation, Eq.(21), is a function of the product βL , and not a function of β or L alone. It implies that the characteristic equation can be solved once and for all, the solutions of the characteristic equation, namely βL 's can be tabulated, and the frequencies of a uniform fixed-free beam of any length can be obtained from such a table and Eq.(23). Mathematically,

$$\frac{d}{dL}(\beta L) = 0 \quad (26)$$

Expanding Eq.(26), the variation of β with respect to L can be obtained as

$$\frac{d\beta}{dL} = -\frac{\beta}{L} \quad (27)$$

Or, in terms of the frequencies,

$$\frac{d\omega}{dL} = -\frac{2\omega}{L} \quad (28)$$

A closed-form solution for ω is given by

$$\omega(L) = D/L^2 \quad (29)$$

where the length dependency of the frequencies is stated explicitly and $D = \omega(L_0)L_0^2$ is a constant. Since the frequencies and the associated eigenfunctions for any length of the beam can be obtained, a question is raised as to

how the eigenfunctions can be related for two beams of different lengths. To this end, it can be shown that

$$\frac{d\phi_r(x,L)}{dL} = -\frac{1}{2L}\phi_r(x,L) - \frac{x}{L}\frac{d\phi_r(x,L)}{dx} \quad (30)$$

where, the length L on the right side of Eq.(22) is also treated as a variable in obtaining the above result.

The above analysis is valid even if the boundary conditions are different but still represent the “classical” end conditions such as pinned or guided; then, Eqs.(20) and (22) will be appropriately modified. Although the normalization constant will not retain its simple form as in Eq.(23), it will contain the product βL and hence $\frac{d(\beta L)}{dL} = 0$.

Dynamics of flexible structures during deployment

An extension allowable in DCAP with respect to the analytical modeling of beams and cables is represented by the implementation of models for deploying structures. The complex formulation of a flexible beam or cable in deployment is internally handled by DCAP through the symbolic processor, allowing the automatic implementation of the time-varying flexibility of the body being deployed.

The equations of motion for a structure during deployment from and retraction into a base that is part of an open-loop multi-body chain. The continuously changing elastic behavior of such a structure is modeled through instantaneous frequencies and mode shapes, using a continuum approach. The properties of the eigenfunctions of a fixed-free beam are exploited to express various domain integral terms as explicit functions of the instantaneous deployed length. The equations reflect the effects of rigid-body and elastic motions on each other. Thus, this complete model can be used in the controller design, if desired. The deployment rate and the base/hinge rotational motion can also be prescribed, if one is only interested in studying the effect of rigid-body motion on the elastic behavior of the structure.

A major obstacle to the modeling of structures during deployment is that the elastic behavior of the structure undergoes continuous change. Consider the deployment of a space truss from a canister. As the deployed length increases, the flexibility of the structure increases. Since this is a continuous process, the frequencies of the structure also undergo continuous change.

Consider the elastic behavior of a beam under deployment from a fixed base. It is usually assumed that translating flexible links can be modeled as beams (Euler-Bernoulli model) in flexure with fixed-free end conditions [11,12]. Tabarrok *et al.* [11] derived certain properties of the mode shapes of fixed-free beams in flexure. The results of [11] have been used in [12] to study the effect of the coupling between the translational (rigid-body) and flexible motions.

The canister is assumed to be attached to the inboard body through a hinge that permits only relative rotational motion. Thus $\sum^{L(j)} y^j = 0$. Let the deployed portion of the space truss be modeled as a beam having uniform stiffness and mass properties. Assuming that the stacked portion of the structure behaves like a rigid-body, the deployed portion of the structure can be modeled as a flexible beam that is instantaneously fixed to the stacked part. Let the translational (deployment) velocity of the beam be denoted by \dot{L} . As the beam is extended, the length of the vibrating section of the beam increases.

We denote the length of the beam outside the support by L and assign the spatial variable x measured from the fixed end to denote the material point on the deployed portion of the beam. Let $x=0$ and $x=L$ correspond to the fixed- and free-ends, respectively. Now, the part of the beam outside the support can be modeled as a fixed-free beam at any time instant.

Variable mass and variable stiffness bodies

In later implementation, the capability for having bodies changing their mass and stiffness properties during the dynamic simulation has been extended to both rigid bodies (for the mass, center of mass location and inertia tensor variation) and generic flexible bodies obtained from finite element models (for both mass properties and stiffness properties).

The implementation has been allowing the configuration of specific “rigid-variable” and “flexible-variable” body types. For these body types, mass, center of mass, inertia tensor, modal mass and stiffness matrices are made time-dependent through the definition of a specific number of user-defined configuration data. For flexible bodies, this means availability of a set of finite element code runs for a fixed structural topology at some reference “key configuration”. For rigid bodies, a dedicated file hosting several configuration data points is implemented.

The variability can be selected to be either function of time or function of mass. It is relevant to notice that no internal consistency check is made for the forces arising from mass variation. In other words, when defining the dynamics symbolically as

$$\underline{F}(t) = \frac{d}{dt}(M\dot{\underline{R}}) = M(t)\ddot{\underline{R}}(t) + \dot{M}(t)\dot{\underline{R}}(t) \quad (31)$$

The right side term $\dot{M}(t)\dot{\underline{R}}(t)$ should be implemented by the user as external force consistently with the specified $M(t)$. A typical case is represented by a launch vehicle. In the user-defined FORTRAN routines, the engine can be modeled by the user through the specific impulse and the mass flow rate rates. The DCAP user continuous controller allows internal coding of states to be integrated in line with system dynamics, and therefore the modeled mass flow rate can be integrated to obtain the instantaneous mass of fuel burnt and the instantaneous mass of the vehicle. The latter can be made available as output of the user-controller and linked as variable driving the tabular mass/stiffness interpolation for time-varying data.

The number of allowable configuration can be selected by the user, and the run-time update performed. This implementation is clearly optimal in terms of run time for the Order(n) formulation, where no mass matrix inversion is directly required. In classical Order(n^3) formulation the approach is instead rather time consuming, requiring besides mass matrices assembly their explicit inversion at each integration step.

An application: LV GNC simulator

The LV guidance, navigation and control function is performed by an On-Board program flight Computer (OBC), which executes the navigation calculations and implements the guidance law (Figure 5). This function is performed with the aid of an Inertial Measurement Unit (IMU), which delivers navigation and attitude data to the computer. Attitude control is performed:

- During first three stages flight by vectoring of the motor thrust achieved by deflection of the motor nozzle. This function is implemented by the Thrust Vector Control system
- During exo-atmospheric flight by attitude control thrusters actuation

Motor nozzle deflection commands are transmitted to the TVC system electronic control unit IPDU (Integrated Power Distribution Unit), which implements a local feed-back control system to deflect the nozzle and keep it in the desired position working against loads applied to the nozzle.

Computational tools are sought necessary to perform a) non-linear trajectory dynamics time simulation under variable mass, b) control design evaluation at discrete trajectory points in linear domain, c) stage separation dynamics. The DCAP software has been selected for the tasks involving non-linear dynamics simulation, including modeling capabilities of contact dynamics, and stiction-friction. For the control design analysis, as for the case of TVC related activities, advantages exist also here in using the MATLAB environment. Particularly with respect to control tools, the post processing analysis and use of the numerous computational tools (tool boxes) of MATLAB.

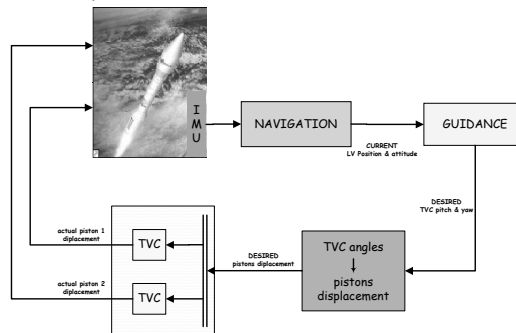


Figure 5: LV GNC description

Results

Linear analysis

This analysis option is implemented in DCAP by means of numerical linearisation of the non-linear dynamic equations about an assumed equilibrium state. The model is implemented for a 6DOF configuration with two pistons now actuating on the nozzle. Figure 6 shows examples of frequency response from demanded piston 1 displacement to actual piston 1 displacement, while Figure 7 shows the corresponding cross-coupling from demanded piston 1 displacement to piston 2 displacement. These results are relevant to a case with LV flexibility beginning of life, big loop open, and no force feedback in the piston control.

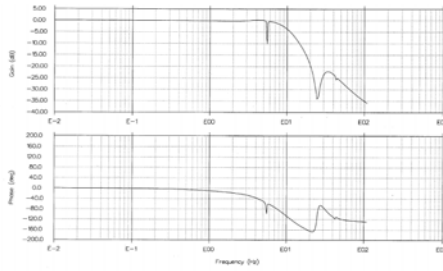


Figure 6: Frequency response from demanded piston 1 displacement to actual piston 1 displacement

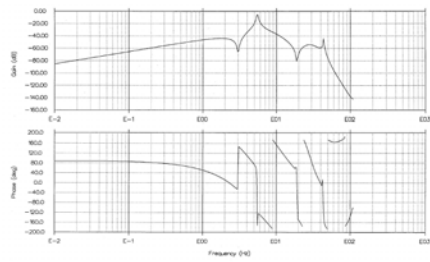


Figure 7: Cross-coupling from demanded piston 1 displacement to piston 2 displacement

Non-linear analysis

DCAP also allows the non-linear time simulation of the LV trajectory. Options are available for including rigid and flexible time varying dynamics. Currently the software runs on DCAP Linux Platform and on MATDCAP in Simulink/Matlab Windows environment. The following figures show the main results of a 1st stage 3D non-linear trajectory simulation.

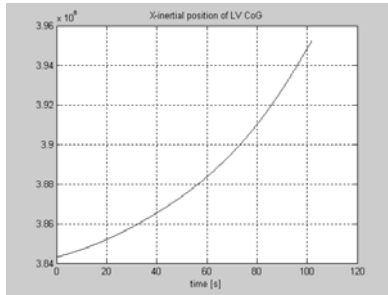


Figure 8: X-inertial position of LV CoG

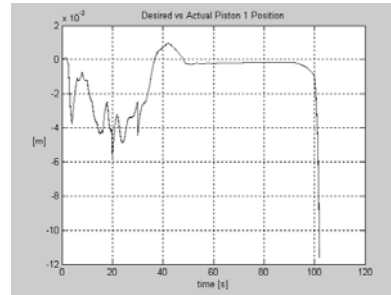


Figure 9: Desired vs Actual Piston 1 Position

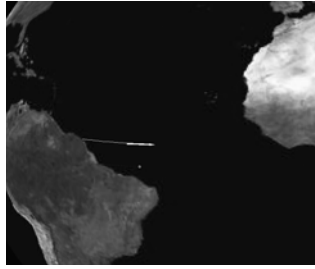


Figure 10: Fly Alongside view

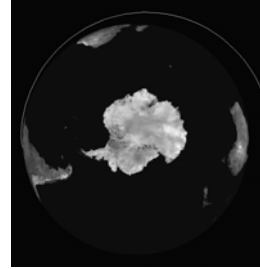


Figure 11: Fixed Position view

Conclusion

Generic modeling capabilities and computational speed are strong assets of the DCAP. Recent advances address sensitivity modules towards MonteCarlo type of analysis and MATLAB-Simulink I/F. The latter makes the control design task much simpler thus when implemented as a user dedicated function within DCAP. This way of operation is expected to enhance very much the user friendliness and thus the number of users of the package. Current developments are expected to focus on extending further functionalities of the modeling, GUI interface, and 3D modeling-animation.

References

- [1] DCAP Release 7, ESA Contract 7971/88/NL/JG, Alenia, 1994.
- [2] R. Franco, M. L. Dumontel, S. Portigliotti & R. Venugopal, The Dynamics and Control Analysis Package (DCAP) - A versatile tool for satellite control, *ESA Bulletin*, Nr. 87, August, 1996.
(<http://esapub.esrin.esa.it/bulletin/bullet87/franco87.htm>)

- [3] R. Franco et al, Enhanced Dynamics and Control Analysis Package (DCAP), *Proceedings of ESA WPP-041, ESA Workshop on Spacecraft Guidance Navigation and Control*, September 1992.
- [4] Singh, R.P., VanderVoort, R.J., and Likins, P.W., Dynamics of Flexible Bodies in Tree Topology - A Computer Oriented Approach, *Journal of Guidance, Control and Dynamics*, Vol. 10, No. 5, September-October 1985.
- [5] Singh, R.P. and Schubele, B., Computationally Efficient Algorithm for Dynamics of Multi-link Mechanism, *AIAA GN&C Conference*, Boston, MA, August 1989.
- [6] Irons, B.M., A Frontal Solution Program, *International Journal of Numerical Methods in Engineering*, N. 2, pp. 5-32, 1970.
- [7] Nielan, P.E., Efficient Computer Simulation of Motion of Multi-body Systems, PhD Dissertation, Stanford University, September 1986.
- [8] Kane, T.R., and Levinson, D.A., *Dynamics, Theory and Applications*, McGraw-Hill, New York, 1985.
- [9] Kane, T.R., Likins, P.W., and Levinson, D.A., *Spacecraft Dynamics*, McGraw-Hill, New York, 1983.
- [10] M. L. Dumontel, S. Portigliotti, R. Venugopal, DCAP: A Tool for Analysis and Simulation of Multi-Body Systems, *45th IAF Conference*, Oslo, October 1995.
- [11] B. Tabarrok, Leech C.M. and Y.I. Kim, On the dynamics of an axially moving beam, *Journal of the Franklin Institute*, 297(3):201-220, March 1974.
- [12] S.K. Tadikonda and H. Baruh, Dynamics and Control of a Translating Flexible Beam, *Journal of Dynamic Systems, Measurement and Control*, to appear.
- [13] Hippmann, G., An Algorithm for compliant contact between complexly shaped surfaces in multi-body dynamics, *Proceedings of the International Conference on Advances in Computational Multibody Dynamics*, Lisbon, Portugal, July 1-4, 2003.
- [14] D. Sciacovelli, S. Kiryenko, G. Baldesi, A. Thirkettle, R. Redondo & P. D. Resta, Vega Prototype 3D Simulation Software with Time Varying structural characteristics, *5th Inter. Conference on Space Launchers: Missions, Control and Avionics*, Madrid, Spain, November 25-27, 2003.
- [15] G. Baldesi, Modelling Launch Vehicle Nozzle and TVC loop with DCAP & Importing the Non-Linear Dynamics in Simulink/Matlab Environment, report in Fulfillment of Master in Satellites and Orbiting Platforms, University "La Sapienza", Rome, Italy 19-11-2003.