MULTIBODY DYNAMICS 2005, ECCOMAS Thematic Conference
J.M. Goicolea, J. Cuadrado, J.C. García Orden (eds.)
Madrid, Spain, 21–24 June 2005

MULTIBODY FREE SOFTWARE FOR TEACHING PURPOSES

Giuseppe Quaranta[⋆], **Pierangelo Masarati**[⋆], **and Marco Morandini**[⋆]

[⋆]Dipartimento di Ingegneria Aerospaziale
Politecnico di Milano
Campus Bovisa, via La Masa 34, 20156 Milano, Italy
e-mail: `giuseppe.quaranta@polimi.it`, `pierangelo.masarati@polimi.it`,
`marco.morandini@polimi.it`,
web page: `http://www.aero.polimi.it/`

Keywords: Computational Mechanics, Higher Education, Open-Source Software, Free Software.

Abstract. *This paper presents the approach currently followed by our research team in teaching multibody dynamics to aerospace engineering students at the University "Politecnico di Milano", Italy. The high level of complexity in multibody dynamics, both in terms of the physical phenomena that are modeled and of architecture of the numerical codes needed to solve them, makes this task extremely hard, especially with current, very tight time constraints. A good trade-off between the rigor in teaching the essential theory and the complexity of the applications needs to be find to allow students to grow an adequate level of awareness in the usage of this tools. Currently, the "dilemma" is solved by giving students access to the multibody dynamics free software MBDyn for both analysis and development. This exposes students to different aspects of multibody dynamics ranging from practical programming to numerical and physical modeling of mechanical and multidisciplinary systems.*

1 INTRODUCTION

Multibody dynamics (MBD) is intrinsically an interdisciplinary topic, with foundations in classical mechanics and strong connections with system dynamics. It requires advanced skills in computational mathematics to deal with the numerical integration of differential-algebraic equations and nonlinear finite element models of deformable components. Haug [1] pointed out how the strong nonlinearities, which are often present in the kinematics and dynamics of MBD systems, leads toward problems that possess a higher degree of complexity if related to those associated with usual linear structural finite element models. As a consequence, engineers that deal with MBD models must build their skills on firm mathematical, numerical and physical foundations. For this reason lessons learned must be applied to realistic models to reach a deep understanding of these concept.

The learning of MBD principles does not require any specific programming skill, but the development of even trivial MBD applications implies non-trivial programming knowledge, which is only partially specific to MBD and may not be strictly related to the curriculum of broadly mechanical engineers. For educational purposes, it is unrealistic to expect students to write even trivial MBD software from scratch; on the contrary, they could largely benefit from the availability of a "workhorse" MBD analysis software that allows to experiment with new modules: new physical problems, physical descriptions, numerical algorithms, and more, requiring students to focus on problems with a significant scientific content without wasting resources on critical programming details. One essential feature of such tool is that, when fruitfully managed, it can largely benefit from the contributions of the students, and grow to a versatile and reliable tool. Students' work may suffer from their lack of a global vision of the problem and of the software; in short, from their lack of experience. However, a careful review by software maintainers, and the continuous peer-review operated by subsequent students and other users of the software, either to fix problems, or to expand and generalize specific capabilities, should provide an amount of feedback and cross-check that is likely unparalleled in other software. This approach can be brought to an extreme by following the free (or open source) software approach [2]. Currently, three different trends can be recognized in non-commercial MBD software, either free or for educational purposes:

1. libraries for symbolic manipulation of equations, in Maple, Mathematica, Matlab, MuPad and other proprietary/free tools; examples are EasyDyn, DynaFlex, MBSymba, SpaceLib, and ROBOTRAN.
2. libraries that serve as building blocks to develop complete models which are later analyzed in numerical form by running an interpreter/compiler for the specific language; examples are CHRONO, SpaceLib.
3. monolithic analysis software that parse models in form of input files and run the simulation using the built-in algorithms and problem libraries; in many cases the problem library can be extended by writing run-time modules or by modifying the source code, if available; an example is MBDyn.

MBDyn represents an example of multibody software whose development is driven both by research needs and by the contribution of students. It is developing a community of worldwide distributed users that participate in discussions about its usage and development on public forums, and is also being used by third-party researchers for high-end projects (e.g. by the ARL-MSCR).

2 MULTIBODY DYNAMICS TEACHING: STATUS AND REQUIREMENTS

Before going through the details of the use of MBDyn in MBD learning, we must clarify what are the main objectives, in the authors' opinion, in teaching multibody principles.

Nowadays, multibody dynamics in Italy is considered a research topic rather than a mandatory subject for the background of industrial engineers. Actually, the curriculum of industrial engineering students, a broad definition that includes mechanical, aerospace, propulsion and related branches of engineering, always included most of the topics related to multibody dynamics, starting from solid foundations in mathematics, kinematics, structural and system dynamics, vibrations, aeroelasticity and fluid-structure interactions, control theory, and many other subjects interspersed in junior and senior classes, but rarely senior classes have been tagged with the "multibody" label. However, interest is growing, as stated for example by Cavacece, Pennestrì and Sinatra [3]. For instance, starting in 2005–6, the Multibody Dynamics class currently jointly offered for the Ph.D. in Aerospace and in Mechanical Systems Engineering by the University "Politecnico di Milano" will be accessible by graduate students during the two year specialization degree (Laurea Magistralis).

The higher complexity of the underlying principles of MBD codes, when compared to, e.g., linear elastic finite element codes, requires special attention in blending the different ingredients.

Multibody software are becoming common tools for the design and analysis of industrial engineering applications. Kinematic synthesis and dynamic analysis of mechanisms constitute the basic elements of MBD modeling. To operate within this approach, the user must have basic knowledge of mechanical systems dynamics. However, this is usually not sufficient to give enough mastery in the use of MBD codes. Furthermore, the large adoption of simple and accessible graphical user interfaces, which hide the technicalities to the end user, gives the misleading impression of utmost simplicity which does not correspond to reality. This incorrect perception is potentially dangerous, because the end user may not be fully aware of the limits of the model under investigation. Being an excellent CAD practitioner does not imply being an engineer, even though good CAD capabilities definitely help in reducing the "processing time", especially when operating in an industrial design cycle.

To properly adopt MBD codes the user should understand the limitations related to: inaccuracies in input data, in physical models formulation, numerical failures or improper modeling of physical phenomena. The knowledge listed above provides the capability to discriminate between numerical or physical pathological behaviors that are root causes of numerical model failures. As a consequence, the main commitment of a University must be to provide future engineers the awareness of possible pitfalls in system modeling and analysis, rather than to provide practice in the usage of specific commercial software GUIs. With this strong background, a user will quickly become accustomed to the front-ends of the different software used by employing companies.

In addition, in the aerospace and in the mechanical engineering field in general there are specific systems where the multibody modeling paradigm represents the best approach to understand and reproduce the complex phenomena observed in the real world. Clear examples of those systems are:

a) helicopters and tiltrotors, where the correct representation of the nonlinear kinematics of the rotor and hub components is essential to reproduce the whole machine behavior [4] (Figure 1);

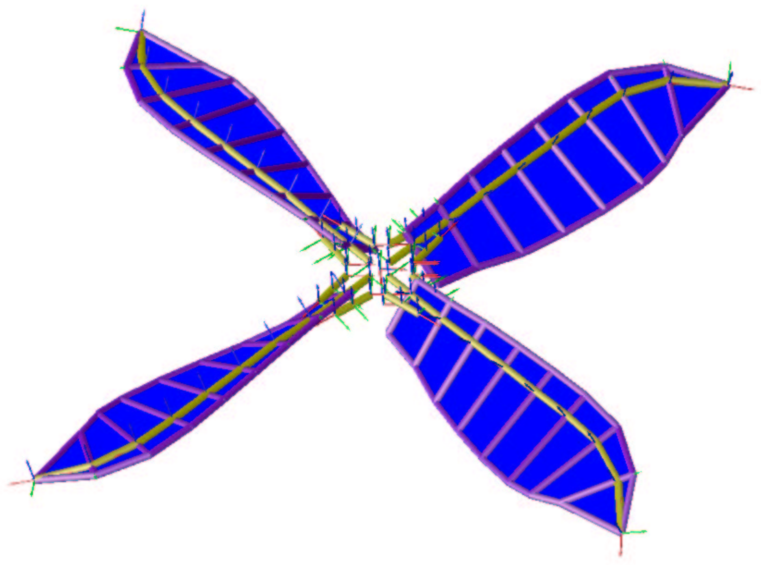

Figure 1: Multibody model of a tiltrotor

b) aircraft landing gears, with their many multidisciplinary subcomponents: the shock absorber, the tire, the brakes and the kinematic and structural properties of the overall system, including the hydraulics of the retraction mechanism [5, 6] (Figure 2).

All these systems are naturally multidisciplinary, since they require modeling of fluid dynamics forces, flexible elements, hydraulic components, and so on. As a consequence, aerospace and in general research-related applications require the capability to develop specialized modules which are usually not offered with general purpose commercial software. For this reason, in our opinion, it is extremely important for students to develop modeling skills and the numerical knowledge necessary to create interacting multidisciplinary specialized elements, useful for the specific system under investigation. The adoption of free software is essential to achieve these goals. In this case, all source component are completely known by users, which can benefit giving a look to the different strategies employed by former developers, both in terms of theoretical and programming choices. At the same time there is a complete control over the numerical sequence of operations the added component will be subjected to. It is important to note that usually commercial software offers the possibility to expand the element library, or to somehow interact with the solution process. However, first of all this capability typically represents a last resort and thus it is not easy and user-friendly at all. Furthermore, the skilled user is typically allowed to modify only selected portions of the code, and to interact only along the lines provisioned by the closed software designers. Very special needs, and in general the need to gain a deeper understanding of the internals of sophisticated simulation software, require a much deeper access to the internals of the software and, in the end, access to all of it, not only to portions.

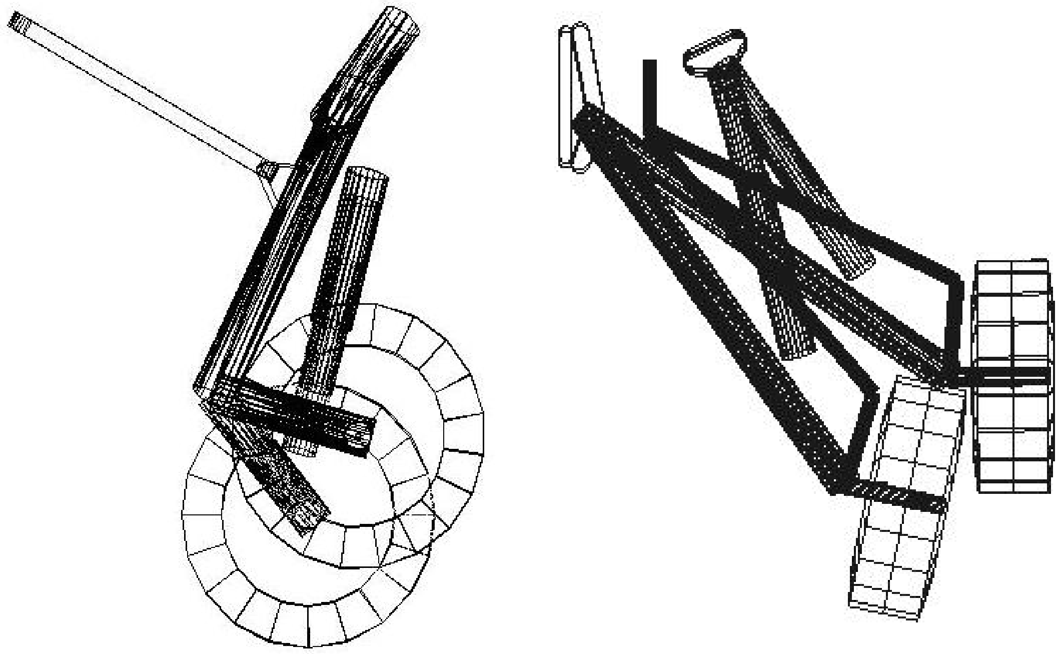

Figure 2: Multibody model of an articulated landing gear

3 MBDYN: AN OPEN SOURCE CODE

MBDyn is a Free Software[1] software for multibody/multidisciplinary analysis developed at the "Dipartimento di Ingegneria Aerospaziale" of the University "Politecnico di Milano". The code allows to model a broad number of different components, such as mechanical, electric, hydraulic, aerodynamic and so on. The user can choose between ideal, kinematically exact constraints, or deformable constraints modeling lumped or distributed structural deformations; typical examples are rods and beam elements, which are based on an intrinsic nonlinear formulation [7, 8]. Furthermore, MBDyn can exchange information with other software like: MATLAB/Simulink, or the equivalent Open Source[2] project Scilab/Scicos, for modeling control systems; fluid-dynamics codes for analysis of fluid-structure interactions. The code is written using C++ programming language to achieve an object oriented architecture. It currently lacks a friendly GUI, which is not considered right now a major drawback for the students learning phase. It essentially interacts with (free as well as commercial software) third party visualization tools.

3.1 Open source and free software

The basic concept on which Open Source software relies on is very simple: if the source code is available to highly motivated users, regardless of the level of experience they may have in a specific field, those who have the knowledge to look at it, and really need a feature, or want to get rid of a nasty problem, will eventually succeed, under the silent agreement that they will make their improvements available to the community of users and developers. Openness does not mean anarchy anyway, and to protect the open philosophy from intellectual property

[1]On the meaning of Free Software, the reader should refer to the web site of the Free Software Foundation `http://www.fsf.org/`.

[2]On the definition of Open Source Software (a somewhat limited version of Free Software), one should refer to the Open Source Initiative (OSI) `http:www.opensource.org/`.

abuses, while preserving the full availability in the broadest manner, different attempts have been made; the most remarkable resulted in the General Public License, an open license emitted by the Free Software Foundation (`http://www.fsf.org/`). The utterly simple idea of Open Source works, and its success is testified by the spread of software such as the Linux OS, Sendmail MTA, Apache web server, the Mozilla browser suite, the OpenOffice desktop environment, and thousands of applications covering most of the needs related to Information Technology (IT). It should be made clear that openness must not be viewed as in conflict with proprietary software and software companies. As a matter of fact, many firms, like Sun, IBM, HP, Novell and so on do contribute to Open Source software and use it for business. Looking at the technical and scientific field, open source software is slowly but steadily growing with very good products, such as the ASTER FEM code, Octave and Scilab mathematical environments, DAKOTA optimization environment, OpenCFD fluid dynamics analysis, to name a few.

How does learning of multibody dynamics fit into this framework? First of all, as opposed to using commercial codes, students have free access to the entire software source code, so they can take a look and learn from the analysis of the solutions adopted by other developers to a common problem. At the same time they are exposed to a quite large project, as opposed to a typical homework software, which can be used for the solution of the dynamics of realistic system and not just for simple academic test cases. The opportunity to experiment how new elements, or new numerical methodologies, impact on the solution of complex dynamical systems allow them to enjoy a great learning experience, especially when enough time is allowed to ponder on the results. The opportunity to interact with a community of users and developers focused on the same class of problems in the absence of the typical disclosure restraints of industrial processes of commercial software can be also a source of good advice for junior engineers.

On the other side the student contribution helps the growth of the code, and adds a continuous mechanism of peer-review of the different features. This continuous expansion and renewal may be a drawback for a commercial code. Instead, for the case of a research code such as MBDyn, it should be considered a positive aspect.

3.2 Multidisciplinary approach to system dynamics

Two entities constitute the foundations on which MBDyn is designed: the *nodes*, which possess and share degrees of freedom, and thus provide the fundamental bricks of connectivity equations, and the *elements* which actually write the equations based on the nodal degrees of freedom. Some elements may add “internal states” as extra degrees of freedom; for instance, algebraic kinematic constraints add Lagrange multipliers as additional unknowns. However, these degrees of freedom are internal, because they do not need to participate to model connectivity.

The numerical integration algorithms are all based on the assumption that the equations are first-order Differential-Algebraic of index 3, which allows to integrate a very broad class of physical problems. This architecture allows the integration of multidisciplinary components, where nodes may assume a very different physical significance from the classical mechanical meaning of point coordinates and reference frames. Typically, structural nodes are associated to six degrees of freedom describing the position and the orientation of a point, and to twelve states that account for momentum and momenta moment, while, for instance, hydraulic nodes, sharing pressure degrees of freedom related to flow balance equations, possess a single degree of freedom associated to a single state.

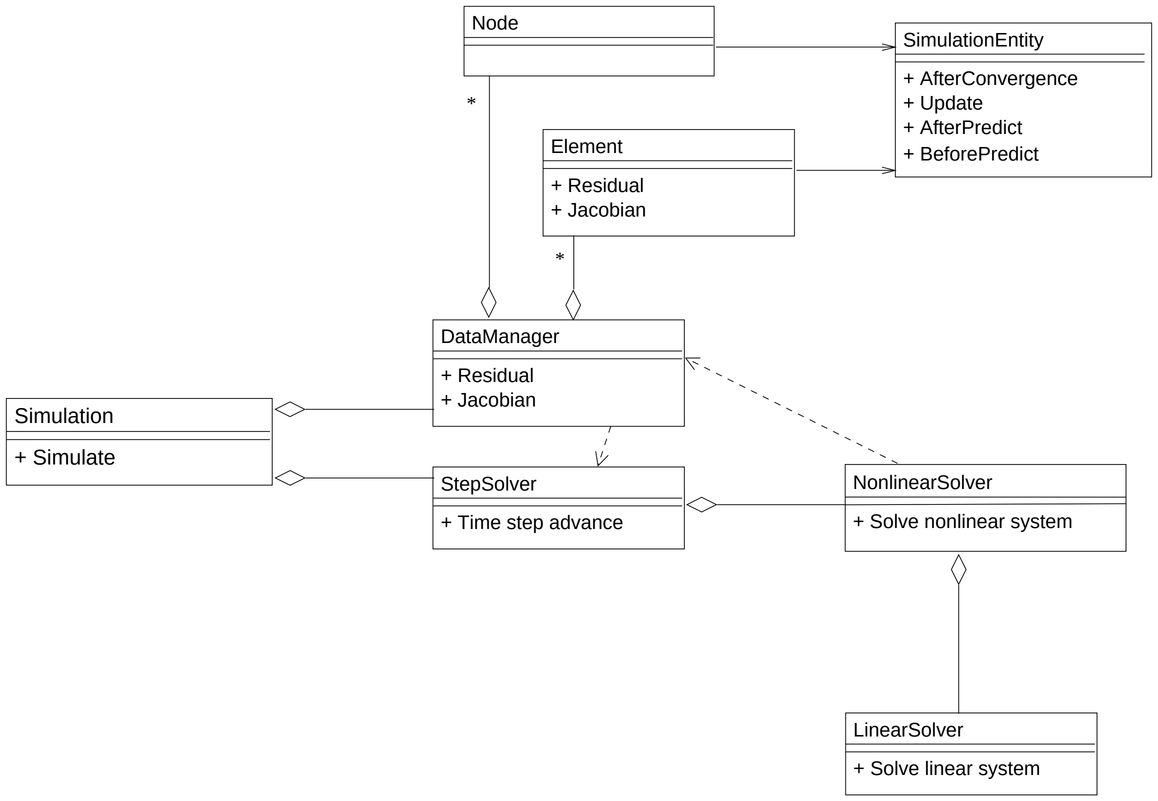

Figure 3: MBDyn main classes architecture

3.3 Object Oriented Programming

One of the main points of strength, which makes MBDyn a great candidate for teaching purposes, is its object oriented structure, based on the large adoption of the C++ programming language throughout the code. This, in common judgment, is supposed to add extra execution overhead; this may not be proven, given the high level of optimization achievable with current state of the art C++ compilers. In any case, the unparalleled programming flexibility provided by this language by means of data abstraction, class inheritance, and function and operator overloading, along with the invaluable support of strong typing and class encapsulation provided by the language itself, gives the developer a high support in the definition of clear structures and interfaces; in one word, in designing a modular software, and in improving it by refining/replacing single modules. In any case, for numerically intensive portions of the code (i.e. numerical solution of linear systems of equation), well proven libraries, written in any language of proven efficiency like C or FORTRAN can be used (and are used in MBDyn) without any design or performance penalty or limitation.

The architectural scheme (Figure 3) is based on the identification of the main operations that must be executed during simulations. This method allows to isolate the code inside them, hiding it under a standard interface for the other portions of the code. This allows to extend all the components independently without impacting the rest of the software. An external driver controls the simulation execution.

The core elements are the `DataManager` and the `StepSolver`. The first one collects all the information about the model, and provides the methods to generate the residual vector and the Jacobian matrix at any time step and with any implemented time integrator. It is also responsible for generating an initial solution compatible with the constraints.

The `StepSolver` manages the problem solution at each time step. First of all it gets access to the class of numerical time integration methods that are used when assembly the Jacobian and Residual of the problem. After the assembly of the time step problem, it calls the `NonlinearSolver` which accounts for the solution of the nonlinear problem; currently, direct Newton-Raphson or iterative GMRES or BiCGStab matrix-free methods are available. Finally, the `LinearSolutionManager` class handles the solution of the linear system of equations which may be required[3] by the `Nonlinear Solver`.

The advantages of the Object Oriented programming paradigm, in the context of student projects for multibody dynamics learning, are related mainly to these aspects:

- the possibility to define an object with a common interface that hides the details of its technical implementation. This decouples any development related to a specific area, i.e. a new element implementation, a new time integration algorithm, a new numerical scheme, and so on, from the rest of the software, e.g. from the need to recreate all data structures from scratch. As a result, students can easily experiment new ideas on the modeling of very complex numerical or physical phenomena with a limited effort.

- the operator overloading capability. It allows to define a new object with the same access interface but with a completely different technical implementation. This allows to easily test and compare different concepts for the same operation.

4 EXPERIENCE GAINED

The adoption of MBDyn for MBD learning has a long history in our department: the project started in early 1990's and went through a continuous evolution with few sharp accelerations. Two of the authors of this paper approached MBD using and developing the MBDyn code. This long history taught that the possibility to learn working with the hands directly on the core of the MBD tool is a proficient method to transfer knowledge. Among the others we must cite few very interesting features, which can be hardly found in any commercial product, and which have been developed under the pressure of the needs of student projects:

- development of elements to represent the electro-elastic behavior of piezo-electric components embedded in nonlinear beam elements;

- development of parallel algorithms specifically adapted for multibody models;

- development of a library of hydraulic elements, including pipes, vessels, valves, actuators and so on;

- development of a slider joint for flexible elements, for the analysis of deformable landing gears;

- development of a methodology for the analysis of fluid-structure interaction with multibody codes;

- methods for the extraction of modal information using Proper Orthogonal Decomposition;

[3]The direct method, based on Newton-Raphson iteration, requires a linear solver for each iteration; the matrix-free methods in principle do not require any. However, in the current implementation, a linear solver is used to precondition the vectors used for the iterative solution.

- modal elements and state space modeling of generalized unsteady forces associated with elastic movements;
- connection with a real-time Operating System (RTAI) to use the MBD code as a numeric benchmark for control system testing and as a simulator for training.

5 CONCLUSION

This paper presented the experience gained by the MBDyn developers' team in the adoption of multibody code development, programming and use as a method to spread the knowledge of MBD within engineering university students. The specific applications typically analyzed in the aerospace field often require a very high level of technical knowledge by the end users and also the capability to interact with the code to develop specific features, not always available in commercial software. The experience gained by those student is generally very positive; as a consequence, after this training, the learning time for a similar product is limited, and increases the self-confidence of new MBD code users. It is the Authors' belief that this experience can be extended, either by the adoption of MBDyn by other potential student/users for the development and testing of new ideas and techniques, or by the development of a new common Open Source product by the research community in MBD.

6 ACKNOWLEDGMENTS

The authors wish to acknowledge all the persons that contributed in various manners and to different extents to the development of MBDyn. This list includes all the students, professors, researchers and engineers around the word that supported, and are currently supporting, the growth of this project.

REFERENCES

[1] E. J. Haug, *Computer Aided Kinematics and Dynamics of Mechanical Systems. Vol. 1: Basic Methods*. Boston: Allyn and Bacon, 1989.

[2] P. Masarati, M. Morandini, G. Quaranta, and P. Mantegazza, "Open-source multibody analysis software," in *Multibody Dynamics 2003, International Conference on Advances in Computational Multibody Dynamics*, (Lisboa, Portugal), July 1–4 2003.

[3] M. Cavacece, E. Pennestrì and R. Sinatra, "Experiences in teaching multibody dynamics," in *Multibody Dynamics 2003, International Conference on Advances in Computational Multibody Dynamics*, (Lisboa, Portugal), July 1–4 2003.

[4] P. Masarati, D. J. Piatak, J. D. Singleton, and P. Mantegazza, "An investigation of soft-inplane tiltrotor aeromechanics using two multibody analyses," in *American Helicopter Society 4th Decennial Specialists' Conference on Aeromechanics*, (Fisherman's Wharf, San Francisco, CA), January 21–23 2004.

[5] S. Gualdi, P. Masarati, M. Morandini, and G. L. Ghiringhelli, "A multibody approach to the analysis of helicopter-terrain interaction," in 28th *European Rotorcraft Forum*, (Bristol, UK), pp. 72.1–12, 17–20 September 2002.

[6] S. Gualdi, M. Morandini, P. Masarati, and G. L. Ghiringhelli, “Numerical simulation of gear walk instability in an aircraft landing gear,” in *CEAS Intl. Forum on Aeroelasticity and Structural Dynamics 2005*, (Muenchen, Germany), June 28–July 1 2005.

[7] G. L. Ghiringhelli, P. Masarati, and P. Mantegazza, “A multi-body implementation of finite volume beams,” *AIAA Journal*, vol. 38, pp. 131–138, January 2000.

[8] G. Quaranta, P. Masarati, and P. Mantegazza, “Multibody analysis of controlled aeroelastic systems on parallel computers,” *Multibody System Dynamics*, vol. 8, no. 1, pp. 71–102, 2002.