Multibody Dynamics Teaching Experience at Politecnico di Milano

FRANCESCO BRAGHIN[*], FEDERICO CHELI[*],
PAOLO MANTEGAZZA[†], PIERANGELO MASARATI[†], GIUSEPPE QUARANTA[†]

Dipartimento di Ingegneria Meccanica[*]/Aerospaziale[†], Politecnico di Milano

Abstract: - This paper illustrates the experience gained at the Faculty of Industrial Engineering of the University "Politecnico di Milano" in teaching multibody dynamics to Ph.D. students. 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 demanding, especially under the very tight time constraints that currently characterize higher education classes. A good trade-off between the rigor in teaching the essential theory and in the illustration of the complexity of real-life applications needs to be found to allow students to grow an adequate level of awareness in the usage of these tools. It is believed that an essential phase of learning how to manage the complexity of the problem goes through the practical solution of problems by developing dedicated software; however, there are limitations on the amount of time a student can dedicate to software development, while software development often requires a considerable effort to address details that are not essential or specific to multibody dynamics. Currently, the "dilemma" is solved by giving students access to existing software, either commercial or internally developed, in order to require only that portion of software development that is essential to the solution of their specific problems. The possibility to access all portions of a reasonably complex and complete simulation software is important since it exposes students to different aspects of multibody dynamics ranging from practical programming to numerical and physical modeling of mechanical and multidisciplinary systems. This aspect can be magnified by the use of free software, so that the community of potential users can grow outside of a single group of work, department or university.

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

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, lead 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 concepts.

The learning of MBD principles does not require any specific programming skill, but the develop-

ment 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, to some extent, 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).

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 mostly 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 interaction, 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);

b) railway and tramcar vehicles having very different design solutions and requiring the modeling of multidisciplinary subsystems such as pneumatic suspensions, hydraulic tilting systems, complex interaction between the vehicle and the track/substructure to correctly predict structure-borne vibrations and noise, etc. [10, 11, 12, 13] (Figure 2);

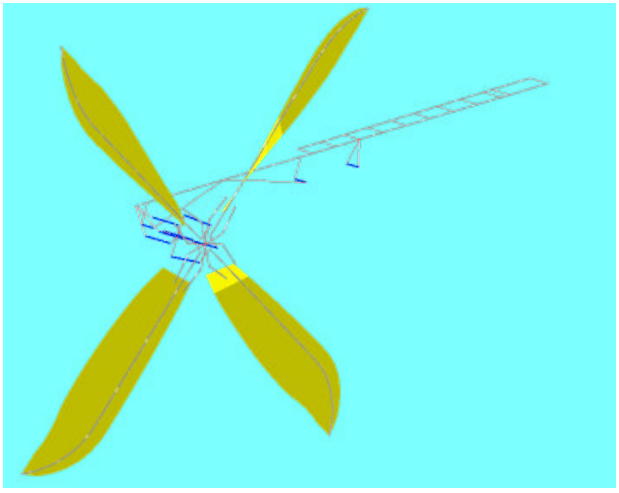

Figure 1: Multibody model of a tiltrotor

c) 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 3);

d) cars where the nonlinear kinematics of the suspension system as well as the deformability of several components, such as the driveline and the bushings, and the interaction between the vehicle's dynamics and active control systems, such as ABS, ESP, active suspensions and 4WD, have to be correctly modelled [7, 8, 9] (Figure 4).

All these systems are naturally multidisciplinary, since they require modeling of fluid dynamics forces, flexible elements, hydraulic components, and so on. As a consequence, 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

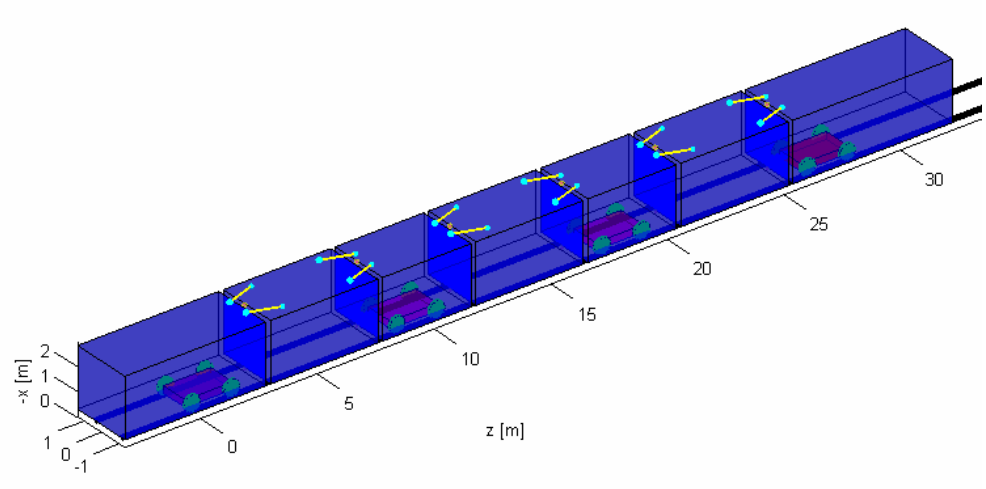

Figure 2: Multibody model of a tramcar vehicle

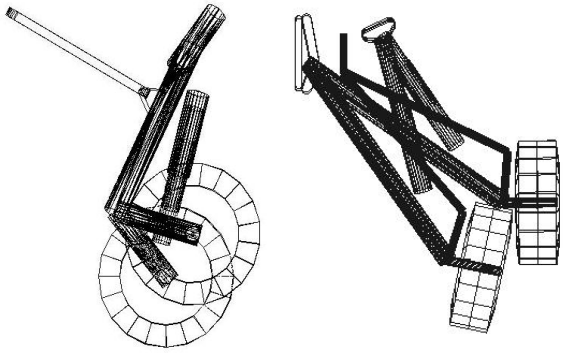

Figure 3: Multibody model of an articulated landing gear

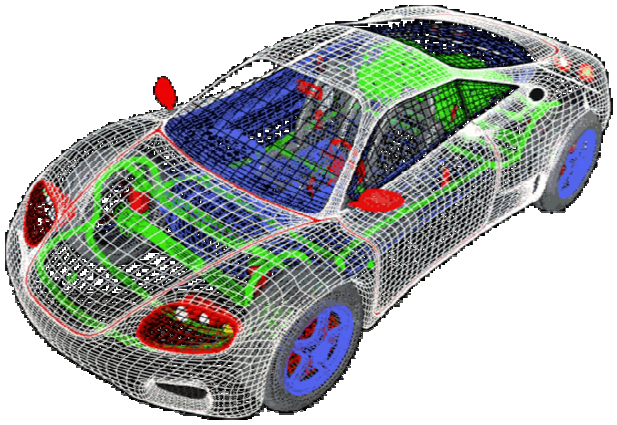

Figure 4: Multibody model of a sports car

this case, all source component are completely accessible by the users, so they can benefit from looking at the different strategies used 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, this capability typically represents a last resort, and thus it may not be easy to access and user-friendly. 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.

3 MBDyn: an Example of Free Software

MBDyn (`http://www.mbdyn.org/`) is a Free Software[1] for general-purpose multi-

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

body/multidisciplinary analysis developed at the "Dipartimento di Ingegneria Aerospaziale" of the University "Politecnico di Milano". The code allows to model a significant variety of components, such as mechanical, electric, hydraulic, aerodynamic and so on. The user can choose between ideal, kinematically exact constraints, deformable constraints modeling lumped or distributed structural components; typical examples are rods and beam elements, which are based on an intrinsic nonlinear formulation [14, 15], and component mode synthesis elements. Furthermore, MBDyn can exchange information with other software like: MATLAB/Simulink, or the equivalent Open Source[2] project Scilab/Scicos, for control systems modeling, and with fluid-dynamics codes for the analysis of fluid-structure interaction. The code is written using the C++ programming language to exploit the Object-Oriented (OO) programming paradigm. 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 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 source code of the software, so they can look at, 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 allows 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.

Furthermore, the contribution of each student helps the growth of the code, and builds up into a continuous peer-review of the different features. This continuing expansion and renewal may be a drawback for a commercial code; however, for a research code like MBDyn, it may be considered a positive aspect.

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

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 need, and add, "internal states" as extra degrees of freedom; for example, algebraic kinematic constraints add Lagrange multipliers as additional unknowns. 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, written in implicit form; which allows to integrate a very broad class of physical problems. This 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.

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 extensive adoption of the C++ programming language throughout the code. In common judgment, higher-level programming langeages are supposed to add extra execution overhead; given the high level of optimization achievable with current state of the art C++ compilers, this is not true. 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 is based on the identification of the main operations that must be executed during simulations. This approach 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, while exploiting extensions that comply with the common programming interface.

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.

3.4 Experience Gained

The adoption of MBDyn for MBD learning has a long history in the Department of Aerospace Engineering of Politecnico di Milano: 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 MBDyn in partial fulfilment of their graduation and Ph.D. thesis. This experience taught that the possibility to learn working with the hands directly on the core of the MBD tool is a proficient method to gain and 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;
- investigation and development of methods for the extraction of modal information using Proper Orthogonal Decomposition;
- development of component mode synthesis elements and state space modeling of generalized unsteady forces associated with elastic movements;
- connection with a free software real-time Operating System (RTAI) to use the MBD code as a simulated experiment for control system testing and as a simulator for training.

4 Conclusions

This paper discussed how teaching Multibody Dynamics to etherogeneous classes of studies in different, yet related, Ph.D. courses, confirmed the importance of software development or, at least, software complexity awareness for the comprehension of computational mechanics. The possibility to access the source code of nontrivial software allows students to focus on their task by building their own contribution on top of existing, well-defined and tested foundations, without "reinventing the wheel" all times. This process is magnified when accessing code that is shared among a large set of users and developers, that goes beyond a single research group, department or university. As an example, the paper illustrates the experience gained by the developers of MBDyn 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 and mechanical engineering fields 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, which may not be 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 or similar software by other potential students/users for the development and testing of new ideas and techniques, or by the development of a new common Free Software product by the research community in MBD.

5 Acknowledges

The authors acknowledge all the persons that contribute to teaching the Ph.D. Multibody Dynamics class for the Ph.D. schools of Aerospace Engineering and Mechanical Systems Engineering at Politecnico di Milano. The authors also 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 world 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 4$^{\text{th}}$ 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 28$^{\text{th}}$ *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] F. Cheli, E. Leo, F. Mancosu, and S. Melzi, "A 14 dof model for the evaluation of vehicle's dynamics, numerical-experimental comparison," *Meccanica*, vol. 41, no. 1, pp. 35–43, 2006.

[8] F. Cheli, M. Pedrinelli, and A. Zorzutti, "Integrated modeling of vehicle and powertrain dynamics," in *Proceedings of ESDA2006, 8$^{\text{th}}$ Biennial ASME Conference on Engineering Systems Design and Analysis*, (Torino, Italy), July 4–7 2006.

[9] F. Braghin and E. Sabbioni, "A 4WS control strategy for improving race car performances," in *Proceedings of ESDA2006, 8$^{\text{th}}$ Biennial ASME Conference on Engineering Systems Design and Analysis*, (Torino, Italy), July 4–7 2006.

[10] G. Diana, G. Traini, S. Bruni, A. Collina, R. D. Bianco, and R. Viganò, "A.D.Tre.S.: a software for railway runnability analysis," in *Proceedings of WCRR '97*, vol. B, (Florence, Italy), pp. 441–447, 16–19 November 1997.

[11] S. Bruni, F. Cheli, G. Diana, and F. Resta, "Active control of the running behaviour of a railway vehicle: Stability and curving performances," *Vehicle System Dynamics (Supplement)*, vol. 478–489, pp. 157–170, 2002.

[12] P. Belforte, F. Cheli, R. Corradi, and A. Facchinetti, "Software for the numerical simulation of tramcar vehicle dynamics," *Heavy Vehicle Systems - International Journal of Vehicle Design*, vol. 10, no. 1/2, pp. 48–69, 2003.

[13] F. Braghin, S. Bruni, and G. Diana, "Experimental and numerical investigation on the derailment of a railway wheelset with solid axle," *Vehicle System Dynamics*, vol. 44, pp. 305–325, 2006.

[14] 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.

[15] 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.