

A Collaborative Benchmarking Framework for Multibody System Dynamics

Manuel González, Francisco González, Alberto Luaces, Javier Cuadrado

This is a post-peer-review, pre-copyedit version of an article published in Engineering with Computers. This version of the article has been accepted for publication, after peer review and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: <https://doi.org/10.1007/s00366-009-0139-0>.

A collaborative benchmarking framework for multibody system dynamics

Manuel González, Francisco González, Alberto Luaces, Javier Cuadrado
Escuela Politécnica Superior, Universidad de A Coruña.
Mendizábal s/n, 15403 Ferrol, Spain. Tel.: (+34) 981.33.74.00 - ext 3870.
E-mail: lolo@cdf.udc.es, fgonzalez@udc.es, aluaces@udc.es, javicuad@cdf.udc.es

Abstract

Despite the importance given to the computational efficiency of multibody system (MBS) simulation tools, there is a lack of standard benchmarks to measure the performance of these kinds of numerical simulations. This work proposes a collaborative benchmarking framework to measure and compare the performance of different MBS simulation methods. The framework is made up of two main components: (a) an on-line repository of test problems with reference solutions and standardized procedures to measure computational efficiency and (b) a prototype implementation of a collaborative web-based application to collect, organize and share information about performance results in an intuitive and graphical form. The proposed benchmarking framework has been tested to evaluate the performance of a commercial MBS simulation software, and it proved to be an effective tool to collect and analyze information about the numerous factors which affect the computational efficiency of dynamic simulations of multibody systems.

Keywords: Multibody dynamics, simulation, performance, efficiency, benchmark, web-based system

1. Introduction

Dynamic simulation of multibody systems (MBS) is of great interest for dynamics of machinery, road and rail vehicle design, robotics, and biomechanics. Numerical simulations performed by MBS dynamics simulation tools lead to more reliable, optimized designs, and

significant reductions in cost and time of the product development cycle. Computational efficiency of these tools is very important for two reasons. First, there are some applications, like hardware-in-the-loop settings or human-in-the-loop simulations, which cannot be developed unless MBS simulation is performed in real-time. And second, when MBS simulation is used in product virtual prototyping, faster simulations allow the design engineer to perform what-if-analyses and optimizations in shorter times, increasing productivity and model interaction. Therefore, computational efficiency is an active area of research in MBS, and a great variety of methods to improve simulation speed have been proposed during the last years [1–5]. Despite the existing interest in fast MBS dynamic simulation tools, there is a lack of standard benchmarks to measure performance of these numerical simulations. Benchmarking is done on an individual basis: different authors use different sets of problems to evaluate the performance of the new proposed methods; the procedures and conditions considered to measure computational efficiency are also different. When results are published, complex test problems are usually described briefly and in a qualitative way due to space limitations: detailed model data are not always available and therefore, other authors cannot replicate the problems in order to use them in future comparisons. In addition, results are scattered across different sources (textbooks, proceedings, journal papers and reports) and difficult to collect. In this scenario, it becomes almost impossible to compare the performance of the different available simulation methods in an objective and quantitative way.

In a previous contribution [6], the authors presented a detailed review of the state of the art in this field, corroborating the situation described in the previous paragraph; in addition, a set of problems involving rigid bodies was proposed as a standard benchmark. However, standard test problems are insufficient to make up a benchmarking system for multibody dynamics: a collaborative, centralized platform to collect, organize, and share information about the

performance of different MBS simulation methods is required. Such a centralized repository would help the research community to easily detect emerging better-than-average performing methods in order to concentrate resources in their development and improvement. Vendors of commercial simulation tools and industrial users could also use this benchmarking platform to monitor progress achieved by the research community, in order to select and incorporate state-of-the-art methods into their products. In summary, a collaborative benchmarking infrastructure for MBS dynamic simulations would transform the scattered, non-comparable performance data available now into valuable knowledge for both the research and user community of this field.

The rest of the article is organized as follows. Section 2 identifies and describes different factors influencing simulation performance in MBS dynamics. Section 3 describes the proposed benchmarking system. Section 4 describes a prototype implementation of a Internet based management system for the proposed benchmark infrastructure. Finally, Sect. 5 provides conclusions and areas of future research.

2. Factors influencing simulation performance

In order to develop a benchmarking system for MBS simulation codes, factors which affect simulation performance must be identified. Information about them shall be collected and taken into account at the time of measuring the computational efficiency of different simulation methods. These factors can be grouped into four main components shown in Fig. 1: (a) the model to simulate, (b) the formalism chosen to perform the simulation, (c) the implementation of the formalism into a computer program, and (d) the computer used to run the program. These four components are highly coupled, and the right choice of each of them is crucial to get the best possible performance.

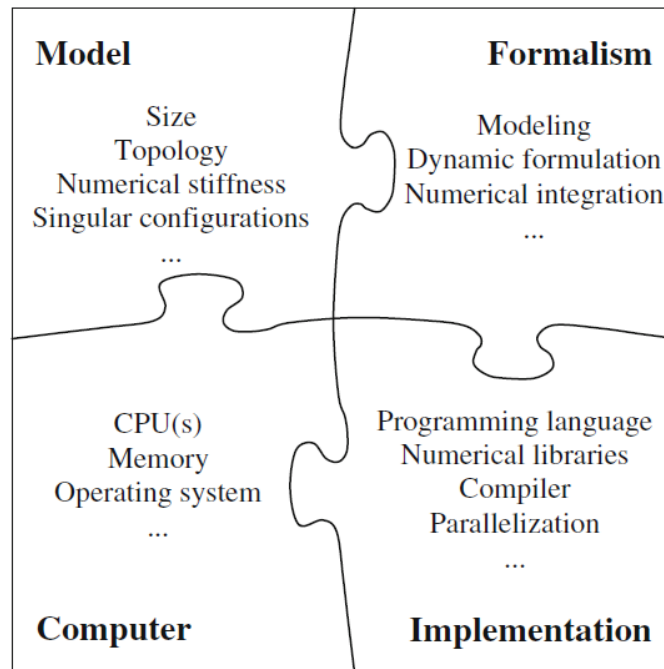


Figure 1: The multibody dynamic simulation puzzle

2.1. Problem

The problem to solve is a given component in the simulation process. The problem is characterized by many properties: size (number of bodies, number of constraints, degrees of freedom), topology (open loop vs. closed loop; fixed vs. changing configuration), type of constraints (scleronomous vs. rheonomous; holonomic vs. non-holonomic), presence of redundant constraints or singular configurations, flexibility of the bodies (rigid vs. flexible), presence of stiffness, presence of discontinuous effects (impacts, clearances, friction, hysteresis), etc. Each of these properties impose different requirements to the formalism used to solve the problem, since no formalism can handle all of them in a proper way.

2.2. Formalism

The formalism used to solve the problem involves three main components: modeling, formulation, and numerical integration.

The modeling of the system involves the selection of a set of parameters or coordinates in order to unequivocally define at all times the position, velocity, and acceleration of the multibody system. The most important types of coordinates currently used are relative coordinates, reference point coordinates, and natural coordinates. Their features and influence on the solution process have been studied in [1]. The second component is the dynamic formulation, obtained from the application of the principles in dynamics, that will lead to the final form of the equations of motion. Some formulations lead to a representation in descriptor form, constituting a set of index-3 differential algebraic equations (DAE). The addition of stabilization techniques reduces the index and makes the solution tractable by means of standard ODE solvers. Other formulations transform the equations of motion to a minimum set of coordinates or state-space form, which is directly solvable by ODE methods. State-space representations may also be obtained by means of velocity transformations, typically used in recursive methods. Descriptions and references are provided in [7, 8]. These sets of DAE or ODE must be solved using a numerical integration scheme, which is the third component of the formalism. Again, a broad family of integrators is available [9, 10].

It is important to state that there is not an optimal formalism for all kind of problems: the performance heavily depends on the size of the multibody system and its properties: changing topologies, singular configurations, stiffness, redundant constraints, etc. [7]. A particular combination of modeling, formulation and numerical integration scheme may give the best performance for a particular problem, and, however, provide poor performance or even fail for other problems.

2.3. Implementation

The implementation refers to the translation of the selected formalism into an executable computer program. This is a key factor for computational efficiency, since a naive implementation can spoil the potential of a good formalism.

Hardware aspects of modern computer architectures influence the design of software for numerical simulations: each architecture has its own features and strengths, which must be exploited to get the best computational efficiency of a given formalism. Careful selection of optimal data structures and lineal algebra implementations can significantly increase the performance of MBS dynamic simulations, as demonstrated in [5]. Sometimes, optimization at the implementation stage is not enough, and simulation algorithms must be completely redesigned to exploit modern hardware designs [11].

Parallelization is another approach to speed up MBS dynamic simulations. Its benefits have been already shown [12, 13], and it does not require expensive hardware any more, since nowadays commodity multi-core/multi-CPU workstations and computing cluster solutions are quite affordable. Some MBS dynamic formulations have been designed to exploit parallel computing environments [14], but they are competitive only when applied to big-sized models.

2.4. Computer

Measuring the computational cost of a dynamic simulation method by counting the number of floating-point arithmetic operations (FLOPs) per function evaluation was the standard method to

compare the performance of different formalisms. This technique does not make sense any longer, since in modern desktop hardware the cost of FLOPs is of the same order as integer operations. In these new machines with a hierarchical memory model, the logic of the algorithm and an efficient use of the cache memory can be more important than the number of FLOPs [1]. As explained in the previous section, hardware characteristics (mainly processor type and number, and memory size) have a significant effect on the actual performance of a given multibody formalism and its implementation. This is specially true in hardware-in-the-loop settings where the dynamic simulation runs in a embedded microprocessor, a common setting in the automotive industry [15] (e.g., design of advanced Electronic Stability Control systems): the limited memory resources often discards implementations with are optimized for speed at the expense of a higher memory footprint, and benefits state-space-form formulations since they require smaller data sets than descriptor form formulations. Hard real-time requirements may also discard formalisms which use iterative procedures without a deterministic execution time, or numerical integration schemes which require inputs at points in time prior to their occurrence [16]. As a consequence, top-performing MBS simulation codes (formalism and implementation) designed for offline simulations in desktop computers may be unusable in hardware-in-the-loop computing environments.

3. Benchmarking system overview

The complexity of achieving an optimal combination of formalism and implementation for a given problem and computing environment, described in the previous Section, reinforces the need of a collaborative benchmarking system for MBS dynamic simulations. This system shall achieve

two goals: (a) standardize the method used to measure performance; and (b) provide a collaborative, centralized platform to collect, organize and share information about performance of different simulation strategies. The first goal is addressed in this Section, and the second goal will be addressed in Sect. 4.

In order to standardize the method to benchmark MBS dynamic simulation codes, three components are required: a standard problem set, reference solutions for these problems, and a standard procedure to measure performance.

3.1. Standard problem set

As explained in Sect. 2, the properties of a multibody model often determine which formalisms can simulate it in an accurate and efficient way. Hence, it is very convenient to develop a set of small and simple test problems which isolate a specific characteristic of multibody systems (stiffness, presence of singular positions, redundant constraints, etc.), designed to evaluate the response of a MBS simulation code to that particular characteristic. Simple test problems have another important advantage: users need to invest little time to solve them, which increases the number of potential users of the benchmark. These test problems shall be classified into groups, according to the kind of multibody system: systems with rigid bodies, with flexible bodies, undergoing contact-impact, etc.

Despite the advantages of small test problems, industry usually demands demonstrations with complex, real-life problems. A standard benchmark should satisfy both demands, including two additional categories orthogonal to the previous ones: “Basic problems”, designed for the above-mentioned purposes, and “Industrial applications”, designed to fulfill industry requirements. This

last class may also contain test problems for specific sectors like the automotive or railway industry. The structure of the proposed benchmark problem set is shown in Fig. 2.

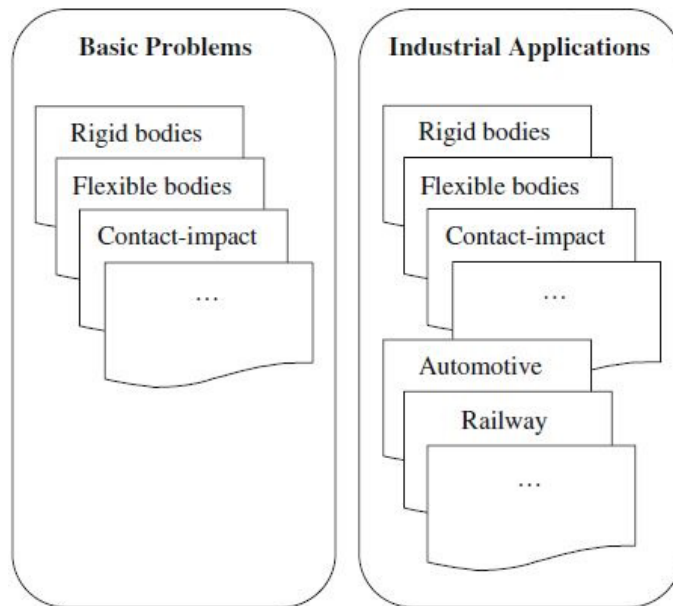


Figure 2: Structure of the proposed benchmark problem set

Two groups of problem in the category of rigid multibody systems have been defined: group A in the class of “Basic Problems” (Table 1), and Group B in the class of “Industrial Applications” (Table 2). Problems in Group A have been described in detail in a previous contribution [6], providing model data and simulation conditions; all of them proved to be good quality benchmark problems since they can reach the limits of simulation methods available in commercial MBS simulation codes. Problems in Group B proposed in this contribution are undergoing a similar validation process.

Table 1: Group A – Basic problems for rigid MBS

Code	Name	Characteristic
A01	Simple pendulum	Example problem (2D)
A02	N-four bar mechanism	Singular positions (2D)
A03	Andrew's mechanism	Very small time scale (2D)
A04	Bricard's mechanism	Redundant constraints
A05	Flyball governor	Stiff system

Table 2: Group B – Industrial applications for rigid MBS

Code	Name	Characteristic
B01	Iltis Vehicle	Automotive
B02	Dornier's antenna	Aerospace
B03	Human body	Biomechanics
B04	PUMA robot	Robotics (serial)
B05	Stewart platform	Robotics (parallel)

3.2. Reference solutions

Computational efficiency of a particular MBS dynamic simulation code depends on several factors, as explained in Sect. 2: problem, modeling, implementation, and computer. Once these factors are fixed, the computational efficiency of the simulation code is still a function of the desired accuracy, since users can tune the method parameters (integration step, tolerances, etc.) to decrease CPU-times at the expense of decreasing the precision of the solution. Therefore, when comparing the performance (i.e., speed) of two different methods for a certain test problem, the

same precision level must be required. The only way to ensure that the solutions generated by both solutions have the same precision is by comparing them with a reference solution.

In [6], reference solutions for problems in group A were obtained using commercial and in-house developed MBS codes. For each test problem, the corresponding model was build and solved using different formalisms using decreasing integrator tolerances and time steps, until high convergence was achieved in all of the obtained solutions. For problems in Group B, reference solutions are more difficult to obtain: due to the complexity of these problems the solutions obtained with different simulation methods and codes do not converge to the same values, and therefore the reference solution shall be computed as an average value of the obtained solutions.

3.3. Performance measurement procedure

The proposed measurement procedure for a test problem consists on solving the problem with a given required accuracy and measure the elapsed CPU-time. The required accuracy is defined for each problem as a maximum error between the reference solution and the obtained solution. In general, solutions are time-histories of several components (positions, velocities, forces, etc.), and each test problem specifies an expression to evaluate the error in the solution. The user must tune the simulation method (adjusting integration step, tolerances, etc.) to achieve the required precision with a minimum CPU-time.

However, comparing CPU-times has two problems. First, CPU-time not only measures the performance of the formalism and the implementation, but also the performance of the computer running the simulation, and this is not desirable in many situations. And second, since CPU-time

is proportional to the final time of a dynamics simulation (given in the problem description), CPU-times for similar test problems with different final times cannot be compared.

To solve the first problem, a custom computer benchmarking utility is used. The utility is quite simple: a dynamic simulation of a road vehicle (the reference problem) is performed by an in-house developed code (the *reference solver*). The simulation last 20 s, and the averaged CPU-time of 3 simulation runs is measured. The utility calculates a *Hardware Performance Ratio (H.P.R.)* using Eq. 1. This ratio measures the computer performance when used to run MBS dynamic simulations:

$$H.P.R. = \frac{\text{Simulation time}_{reference\ problem}}{CPU - time_{reference\ solver}^{reference\ problem}} \quad (1)$$

As explained in Sects. 2.3 and 2.4, the performance of a given computer depends on the particular characteristics of the running application. For that reason, the *H.P.R.* is not completely independent on the reference problem and reference solver used to measure it. However, the proposed custom computer benchmarking utility uses computer resources in a similar way to most multibody simulations (medium memory footprint and high number of floating-point operations), and therefore that dependency is small and acceptable.

Then, the performance of a given combination of formalism and implementation can be calculated for a test problem with the *Software Performance Ratio (S.P.R.)* defined in Eq. 2:

$$S.P.R._{test\ problem\ i} = \left(\frac{1}{H.P.R.} \right) \frac{\text{Simulation time}_{test\ problem\ i}}{CPU - time_{test\ problem\ i}} \quad (2)$$

This ratio tries to remove the dependency of the CPU-time for *test problem i* on:

- a) The computer performance. This is achieved by introducing the *H.P.R.* in the definition of *S.P.R.*, which normalizes CPU-times as if they were measured in computers with $H.P.R. = 1$. This allows to compare *S.P.R.* values measured in different computers.
- b) The final simulation time of the test problem. This allows to get *S.P.R.* values of the same order of magnitude even for test problems with very different final simulation times, a property which is very convenient to make graphical comparisons of *S.P.R.* values for different problems.

As a result, the *S.P.R.* for a given *test problem i* mainly depends on the simulation code used to solve the problem (formalism and implementation). Numerical experiments performed by the authors with different computers and simulation codes confirm this fact, and therefore the *S.P.R.* can be considered an adequate measure of the simulation code performance for a particular type of problem.

4. Implementation of a collaborative benchmarking platform

As stated in the Sect. 1, standard test problems and performance measurement procedures shall be enclosed in a collaborative, centralized platform which collects, organizes, and shares information about performance of different MBS dynamic simulation methods in a homogeneous format, in order to make this information helpful to the multibody systems dynamics community.

A prototype of a web-based collaborative management platform for the proposed MBS benchmark has been developed. The platform is made up of three main components: (a) a repository of benchmark problems with detailed documentation, (b) a database of benchmark

results, and (c) services to submit, search and compare performance results stored in the database from a web-browser. The following subsection describe each components in more detail.

Benchmark Specifications

Here you will find specifications and models for the benchmark problem collection. Specifications are published in XML format. Models are available in MbsML and ADAMS format.

Group A: Rigid bodies - Basic Problems

ID	Problem Name	Specifications	Models
A01	Simple pendulum	XML	MbsML ADAMS
A02	N-four-bar mechanism	XML	MbsML ADAMS
A03	Andrew's mechanism	XML	MbsML ADAMS
A04	Bricard's mechanism	XML	MbsML ADAMS
A05	Flyball governor	XML	MbsML ADAMS

Group B: Rigid bodies - Industrial Applications

ID	Problem Name	Specifications	Models
B01	Iltis Vehicle	XML	MbsML ADAMS
B02	Dornier antenna	XML	MbsML ADAMS
B03	Human body	XML	MbsML ADAMS
B04	PUMA robot	XML	MbsML ADAMS
B05	Stewart platform	XML	MbsML ADAMS

Figure 3: Repository of benchmark problems

4.1. Problem repository

Currently, the problem repository contains documentation for the problems in the aforementioned Group A and Group B, as shown in Fig. 3. Documentation for each problem (Fig. 4) includes the problem specification (a brief description of the multibody system, the analysis to be performed and instructions concerning precision), detailed multibody models encoded in different formats, and reference solutions consisting on time-histories of selected variables, in tabular and graphical form, and animations of the resulting motion. Since no standard data format

exist in the field of multibody dynamics, multibody models are provided in a neutral XML-based format which can be easily parsed and translated into other formats. Models for problems in Group A are also provided in ADAMS/Solver format [17], a commercial MBS simulation tool.

Multi-Body Systems Benchmark - Problem Specification

A04 : Bricard Mechanism

[2005-08-30] http://lim.i.i.udc.es/mbsbenchmark/dist/A04/A04_specification.xml

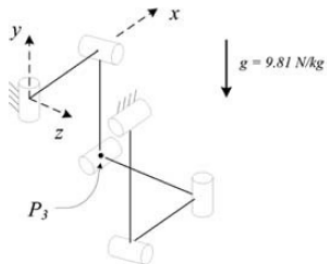
Dynamic simulation of the Bricard mechanism

Author: Manuel González Castro lojo@cdf.udc.es
 Universidad de la Coruña - Laboratorio de Ingeniería Mecánica
 Mendizabal s/n, Ferrol, 15403, España.
<http://lim.i.i.udc.es>

Multibody System Description:

The Bricard mechanism is a classic example of overconstrained system. It is composed by 5 rods of 1 m length with a uniformly distributed mass of 1 kg, and 6 revolute joints. The system is under gravity effects (9.81 N/kg acting in the negative y direction).

Grübler formula gives 0 degrees-of-freedom for this mechanism, but the particular orientation of the revolute pairs yields a system with 1 degree-of-freedom.



Analysis Description:

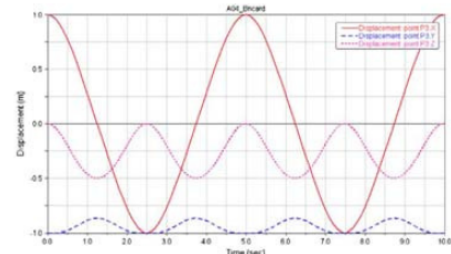
The analysis to be performed is a dynamic simulation with a duration of 600 s. Initially, the system is at rest in the position shown in the figure.

Reference solution:

The reference solution for this problem is the time-history of the position of point P3 (x, y and z coordinates). The maximum allowed errors are 1.0E-1 (low precision) and 1.0E-3 (high precision), measured with 3000 output steps and a threshold value of 1.0E-3.

The following information is provided:

1. Time-history of the position of point P3, as text data file: [A04_solution_data.txt](#)
2. Animation of the system, as AVI movie (first 10 s): [A04_solution_movie.avi](#)
3. Time-history of the position of point P3, as plot (first 10 s):



Multi-Body System Model:

A full quantitative description of the problem is available in [MbsML](#) format: [A04_model_mbsml.xml](#)

Measures:

Name	Description
Time	CPU-time spent by the software to solve the problem (in s).

Figure 4: Specifications for a benchmark problem

4.2. Results database

Section 2 illustrate the various factors with affect the performance of a MBS dynamic simulation; information about them shall be collected in a database of performance results. This prototype implementation uses a relational database with the structure shown in Table 3: data are

classified into eight tables; the first field in each table acts as unique identifier (primary key), and fields in italics are linked to other tables.

Table 3: Structure of the results database

Table	Fields
Problems	ProblemID, Name, URL
Organizations	OrganizationID, Name, Address, City, Country, URL
Users	UserID, <i>OrganizationID</i> , FirstName, LastName, Email, Password
Computers	ComputerID, <i>OrganizationID</i> , Nickname, Brand, Model, Motherboard, CPUModel, CPUnumber, Memory, OSname, PerformanceRatio
Softwares	SoftwareID, Name, Author, URL
Builds	BuildID, <i>SoftwareID</i> , Version, BuildSystem, BuildOptions, Libraries
Methods	MethodID, <i>SoftwareID</i> , Name, Coordinates, Formulation, Integrator
Results	ResultID, <i>ProblemID</i> , <i>UserID</i> , <i>ComputerID</i> , <i>SoftwareID</i> , <i>BuildID</i> , <i>MethodID</i> , Tags, IntegrationStep, CPUtime, relativeError, Comments

The first table, *Problems*, holds information about each problem (problem name and web-address of its documentation). Tables *Organizations* and *Users* hold contact information about the organizations and persons that submitted results to the system, since result submission is not anonymous. Table *Computers* holds information about the computing environment used to solve the test problems. In order to compare performance results produced in different computers without taking into account the computer power, this table also stores the Hardware Performance Ratio of the computer. Information about the simulation tool is split into three tables. Table *Softwares* holds general information about the software implementation (name, author and website). Table *Builds* holds technical information about the software implementation; for

commercial codes this information is the application version; for non-commercial codes, information about source code version, programming language, compiler, optimization flags, and numerical libraries can be entered. Table *Methods* holds information about the formalism (modeling technique, formulation and integrator) used to solve the problem. This information is not integrated with the software technical description, because the same software can provide several methods. Finally, table *Results* holds the performance results for a particular problem using a given combination of computer, implementation (software and build) and method.

The level of categorization of the results database, with 54 fields grouped in 8 tables, tries to fit the requirements of expert developers of MBS simulation software, while keeping a structure simple enough for the average user of commercial MBS simulation packages. Some of the fields in tables *Builds* and *Methods* could be divided in a more detailed taxonomy, but the resulting database would probably confuse user of commercial software with no control on many details about the implementation and the simulation method.

4.3. Management of benchmark results

Registered users can upload results to the database. The registration process only requires minimal contact details, and provides login and password. Results submission has three steps (Fig. 5). In the first step, the user chooses the test problem for which results are to be submitted, and the computer and software used to solve it. Information about new computers or software systems can be entered at this stage. When the user enters information about the computer, its Hardware Performance Ratio must be provided. In the second step, the user chooses the build environment for the software (i.e., details about the implementation) and the method used to perform the

dynamic simulation (details about the formalism). Again, information about new build environments or methods can be introduced. Finally, the user enters the measured CPU-time. The user can also enter several optional fields: the number of integration steps, the error in the obtained solution, annotations giving details about how the results were obtained, and a tag to allow an easy retrieval of this result in the future (results can be filtered by a particular tag value).

Anonymous users can retrieve performance results stored in the database. Several types of queries can be run to filter and compare performance results:

- *Basic query* shows all the results submitted for a selected test problem. Information is presented in tabular form and in graphical form using a bar graph (Fig. 6). The bar length represents the *Software Performance Ratio* of each simulator.
- *Aggregated performance* shows the aggregated performance of different simulators for a selected set of problems. As in the previous query, information is presented in tabular form and in graphical form using a bar graph. The bar length represents the average *Software Performance Ratio* of each simulator over the selected problems, and the bar color represents the percentage of problems that can be solved by that simulator.
- *Compare two simulators* shows the average performance of two selected simulators over a given range of problems.

In all types of queries, filters can be applied to on any of the database fields in order to restrict the results range. In addition, detailed information about a particular result can be examined, including author contact information.

The screenshot shows a web browser window titled "MBS Benchmark - Microsoft Internet Explorer" with the address bar displaying "http://fm.il.udc.es/mbsbenchm". The page content includes navigation links: "Submit results", "Delete Information", "Change my details", "Hello, lolol!", and "[log off]".

Submit Results

Step 1 of 3

Problem:

Computer: [Add new](#)

Software: [Add new](#)

Step 2 of 3

Software Information

Build: [Add new](#)

Method: [Add new](#)

Step 3 of 3

Result Information

CPU time

Integration Steps

Error (%)

Tag

Annotations

Figure 5: Steps to submit a performance result

In order to test the prototype benchmarking platform, problems in Group A have been solved using the well-known commercial MBS simulation tool ADAMS/Solver [17]. Three numerical integrators have been tested (named GSTIFF, WSTIFF and CONSTANT_BDF) combined with two formulations (the index-3 formulation I3 and the stabilized-index two SI2). The ABAM integrator (Adams Bashforth-Adams Moulton) combined with a coordinate partitioning formulation has also been tested. Detailed information about these dynamic simulation methods can be found in the ADAMS/Solver User's Guide. All performance results generated from the benchmark tests have been up-loaded in the web-based application, and can be used as a baseline for future comparisons of other MBS analysis codes. The described prototype implementation proved to be an effective tool to analyze and compare different MBS simulation methods in a collaborative way.

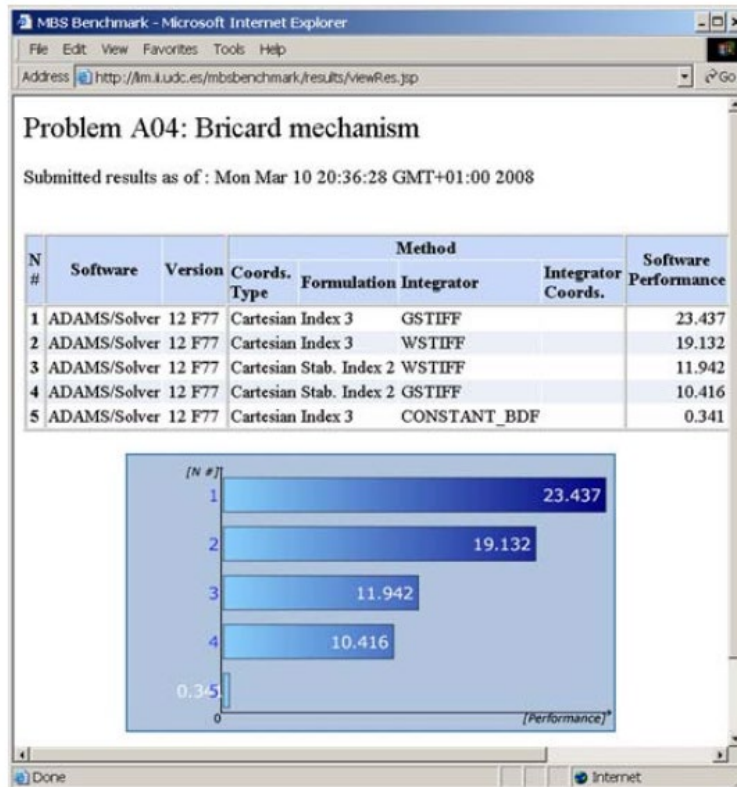


Figure 6: Result of a database query in tabular and graphical form

5. Conclusions and future work

The proposed benchmarking systems is the first attempt to develop a collaborative benchmarking system for multi-body system dynamics. The system architecture is made up of two main components: (a) a standard problem set, with detailed models, reference solutions and procedures to measure computational efficiency performance, and (b) a web-based collaborative platform to store and manage benchmark results, which allows users to search compare the performance of different simulation methods in a intuitive, graphical form. The system could be used locally by a single research team, to monitor progress of a given simulation tool during its development cycle, or globally by all the MBS community, to compare commercial and academic simulators developed by geographically distant teams. Some guidelines for future work are:

- a) To define new groups of problems to benchmark simulator performance when dealing with other phenomena like flexibility, contact, impacts, etc. Some interesting test problems of these types have been proposed during the last years: reference solutions must be validated and standardized problem documentation must be generated. However, the lack of a standard neutral data format to describe multibody systems is an important obstacle to develop a repository of standard test problems.
- b) To generate validated reference solutions for problems in group “Industrial applications”. The authors are carrying out this task for some of the problems proposed in Group B, and it proves to be quite involved due to the complexity of these models. It will require the use of several simulation tools and, probably, the participation of several research teams or industrial partners.

- c) To develop a detailed classification of the various components of a multibody dynamics formalism (modeling, formulation and method). This would help to replace the content of most fields in the performance results database from the current free-text user description to a standard list of choices, in order to make possible to filter results in a more controlled, standard form and allow data mining techniques to be applied if large data sets are stored in the database.
- d) To develop software tools to automate the benchmarking procedure. Some tasks that could be automated are the evaluation of the error in a given solution compared with the reference solution (this goal requires a standard data format for simulation results), and the submission of results to the central database, avoiding the manual work of filling HTML forms. In this way, developers of research MBS simulation software could use the collaborative benchmarking system to control the quality and monitor the improvements in every new release of their code: without human intervention, all problems in the benchmark would be solved and performance results would be automatically computed and submitted to the central database. In few minutes, developers would get an overall view of the performance of their new simulation methods and how they compare with alternative existing methods.

Finally, support from international organizations is key to achieve a true standard benchmarking system for multibody system dynamics.

Acknowledgments

The authors gratefully acknowledge the support of the Spanish MEC under Grant No. DPI2003-05547-C02-01 and the Galician DGID under Grant No. PGIDT04PXIC16601PN.

References

1. García de Jalón J, Bayo E (1994) Kinematic and dynamic simulation of multibody systems - the real-time challenge. Springer-Verlag, New York
2. Bae DS, Lee JK, Cho HJ, Yae H (2000) An explicit integration method for realtime simulation of multibody vehicle models. *Comput Methods Appl Mech Eng* 187:337–350
3. Anderson KS, Critchley JH (2003) Improved ‘Order-N’ performance algorithm for the simulation of constrained multi-rigid-body dynamic systems. *Multibody Syst Dyn* 9:185–212
4. Anderson K, Mukherjee R, Critchley J, Ziegler J, Lipton S (2007) POEMS: parallelizable open-source efficient multibody software. *Eng Comput* 23:11–23
5. Gonzalez M, Gonzalez F, Dopico D, Luaces A (2008) On the effect of linear algebra implementations in real-time multibody system dynamics. *Comput Mech* 41:607–615
6. Gonzalez M, Dopico D, Lugrís U, Cuadrado J (2006) A benchmarking system for MBS simulation software: problem standardization and performance measurement. *Multibody Syst Dyn* 16:179–190
7. Cuadrado J, Cardenal J, Morer P (1997) Modeling and solution methods for efficient real-time simulation of multibody dynamics. *Multibody Syst Dyn* 1:259–280
8. Cuadrado J, Cardenal J, Morer P, Bayo E (2000) Intelligent simulation of multibody dynamics: space-state and descriptor methods in sequential and parallel computing environments. *Multibody Syst Dyn* 4:55–73

9. Hairer E, Nørsett SP, Wanner G (1987) Solving ordinary differential equations I. Nonstiff problems. Springer-Verlag, New York
10. Hairer E, Wanner G (1991) Solving ordinary differential equations II. Stiff and differential-algebraic problems. Springer-Verlag, New York
11. Becker C, Kilian S, Turek S (1999) Consequences of modern hardware design for numerical simulations and their realization in FEAST, Euro-Par'99, Parallel Processing. In: Proceedings of the 5th international Euro-Par conference. Lecture notes in computer science, vol 1685. pp 643–650
12. Eichberger A, Fuhrer C, Schwertassek R (1993) The benefits of parallel multibody simulation and its application to vehicle dynamics. In: Advanced multibody system dynamics: simulation and software tools. Kluwer Academic Publishers, Dordrecht, pp 107–126
13. Quaranta G, Masarati P, Mantegazza P (2002) Multibody analysis of controlled aeroelastic systems on parallel computers. *Multibody Syst Dyn* 8:71–102
14. Duan S, Anderson KS (2000) Parallel implementation of a low order algorithm for dynamics of multibody systems on a distributed memory computing system. *Eng Comput* 16:96–108
15. Glesner M, Kirschbaum A, Renner FM, Voss B (2002) State-of-the-art in rapid prototyping for mechatronic systems. *Mechatronics* 12:987–998
16. Arnold M, Burgermeister B, Eichberger A (2007) Linearly implicit time integration methods in real-time applications: DAEs and stiff ODEs. *Multibody Syst Dyn* 17:99–117
17. MSC.Software Corporation (2004) ADAMS. <http://www.mscsoftware.com/>