Proceedings of IDETC/CIE 2005
ASME 2005 International Design Engineering Technical Conferences
& Computers and Information in Engineering Conference
September 24-28, 2005, Long Beach, California USA

DETC2005-85035

BENCHMARKING OF MBS SIMULATION SOFTWARE

Manuel González
University of A Coruña
Escola Politécnica Superior
Mendizábal s/n, 15403 Ferrol, Spain
lolo@cdf.udc.es

Urbano Lugrís
University of A Coruña
Escola Politécnica Superior
Mendizábal s/n, 15403 Ferrol, Spain
ulugris@cdf.udc.es

Ruth Gutiérrez
University of A Coruña
Escola Politécnica Superior
Mendizábal s/n, 15403 Ferrol, Spain
rutgut@udc.es

Javier Cuadrado
University of A Coruña
Escola Politécnica Superior
Mendizábal s/n, 15403 Ferrol, Spain
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 software applications. Benchmarking is done on an individual basis: different sets of problems are used, and the procedures and conditions considered to measure computational efficiency are also different. In this scenario, it becomes almost impossible to compare the performance of the different available simulation methods in an objective and quantitative way.

This works proposes a benchmarking system for MBS simulation tools. The system is made up by two components: (a) a collection of test problems with reference solutions and standardized procedures to measure computational efficiency; and (b) a web-based application to collect, organize and share information about performance of existing simulation methods.

The benchmarking system has been applied to evaluate the performance of ADAMS/Solver, a popular MBS simulation tool.

1. INTRODUCTION

Dynamic simulation of multibody systems (MBS) is of great interest for dynamics of machinery, road and rail vehicle design, robotics, biomechanics and spacecraft control. Computer simulations performed by MBS 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 devices, which cannot be developed unless MBS simulation is performed in real-time. And second, when MBS simulation is used in virtual prototyping, faster simulations allow the design engineer to perform what-if-analyses and optimizations in shorter times, increasing productivity and the interaction with the model. 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 [1-6].

Despite the existing interest in fast MBS dynamic simulation tools, there is a lack of standard benchmarks to measure performance of these kinds of software applications. 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 (dimensions, inertias, forces, …) 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 methods in an objective and quantitative way.

A benchmarking system for MBS dynamics is required. This system shall achieve two goals: (a) standardize the problem set and procedures used to measure computational efficiency; and (b) provide a collaborative, centralized platform to collect, organize and share information about performance of different existing methods.

Such a benchmarking system for MBS dynamics simulation would speed up research progress in this field and benefit industrial users of this technology. With a centralized repository of comparable performance metrics for simulation methods, the research community could easily detect which methods have low performance, so giving up its development, and which ones offer better-than-average performance, in order to concentrate efforts in their improvement. Resources would be optimized to work on the most promising research lines. Vendors of commercial simulation tools could use the information provided by the benchmarking system to monitor progress achieved by the research community, and incorporate the best and state-of-the-art methods into their tools. Industrial users of these tools could use the benchmark to evaluate them, and get unbiased information about their real performance. In summary, this benchmarking system would transform the scattered, non-comparable performance data available now into valuable knowledge for all the users of MBS dynamics.

This paper is organized as follows: Section 2 presents a review of previous work in the area. Section 3 identifies and describes factors influencing simulation performance. Section 4 proposes a problem set and measurement procedures for a standard benchmarking system. Section 5 describes a prototype of an Internet based management system for the proposed benchmark. Section 6 shows results from applying the benchmark to a commercial MBS simulation tool. Finally, Section 7 provides conclusions and areas of future research.

2. STATE OF THE ART

The first comprehensive comparison of available MBS simulation codes was published in the handbook by Schiehlen [7]. It offered a qualitative comparison on modeling capabilities, formalism and implementation, but it did not compare the computational efficiency or accuracy of the codes. Another effort was carried out by the International Association for Vehicle System Dynamics (IAVSD): benchmarks for road and railway vehicles were proposed [8]; the results, reported in [9], included comparisons of the solutions generated by several codes, but no information about computational efficiency was provided. The IAVSD road vehicle benchmark was made up of two benchmarks, the 4x4 Bombardier Iltis vehicle [10] and a five-link wheel suspension [11]. Subsequent works have used the Iltis vehicle [3,12-17] or the five-link suspension [18-21] evaluate the performance of MBS formulations. The original IAVSD railway benchmark proved to be inconclusive and a new set of benchmarks were agreed [22]. Results were published in [23], and full problem statements and further updated results are publicly available in a web site [24]. This benchmark offers only qualitative comparisons to assess the effect of the various techniques and approximations made, and many of the participant codes are specific for railway simulation.

The previous benchmarks were developed and proposed by international organizations, but many more problems have been used by other authors to test the performance of new simulation methods (see Table 1). Bayo and Avello [25] used a planar double four-bar mechanism as example of system undergoing singular positions, reused by Cuadrado et al [3]. Anderson and Critchley [6] generalized this example to build heavily constrained ladder systems, used to test recursive formulations. García de Jalón and Bayo [2] compared the computational efficiency of different formulations using a human body model, a heavy truck suspension, and the Bricard's mechanism. All the examples were reused by Rodríguez [20]. Von Schwerin [18] employed a long insulator chain to compare the performance of different sparse solvers used in dynamic formulations. In the field of flexible multibody systems, Jahnke et al [26] used a flexible slider crack mechanism as test problem. Variants of the same model were used by Simeon [27], Cuadrado et al [28] and Schaub and Simeon [29]. Bauchau and Theron [30] employed a hinged beam, and Bauchau and Bottasso [31] proposed a four bar crooked mechanism. These two examples were also reused by Cuadrado et al [28] and Bottasso et al [32], among others.

Table 1: Some benchmarks proposed for MBS dynamic simulation.

Author(s)	Test Problem	References
Schiehlen (1990)	Planar seven body mech.	[3,7]
Schiehlen (1990)	Serial robot	[7]
Tregenza & Anderson (1990)	Bombardier Iltis vehicle	[3,10,12-17,33]
Hiller & Frik (1993)	Five-link suspension	[11,18-21]
Bayo & Avello (1994)	Double four-bar mech.	[3,25]
García de Jalón & Bayo (1994)	Bricard mechanism	[2,20]
García de Jalón & Bayo (1994)	Human body	[2,20]
von Schwerin (1999)	Dielectric chain insulator	[18]
Jahnke et al (1993)	Flex. slider crack mech.	[26-29]
Bauchau and Theron (1996)	Flex. hinged beam	[28,30,32]
Bauchau and Bottasso (1999)	Flex. four-bar crooked mech.	[28,31,32]

Although some of the proposed problems have been reused by several authors, it is very difficult to compare their results. Some authors only use the test problem to show that a certain method can solve it, but they do not measure computational efficiency. Other authors introduce minor modifications in the original problem statement that make results not comparable. Even if the same model is used, problems arise because an authoritative reference solution is missing: since the CPU-time employed to get the solution is proportional to the required accuracy, if different authors solve the same problem with different accuracies, the CPU-times are not comparable.

3. FACTORS INFLUENCING SIMULATION PERFORMANCE

Dynamic simulation of MBS is a complex process involving several factors. 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 (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, (d) and 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.

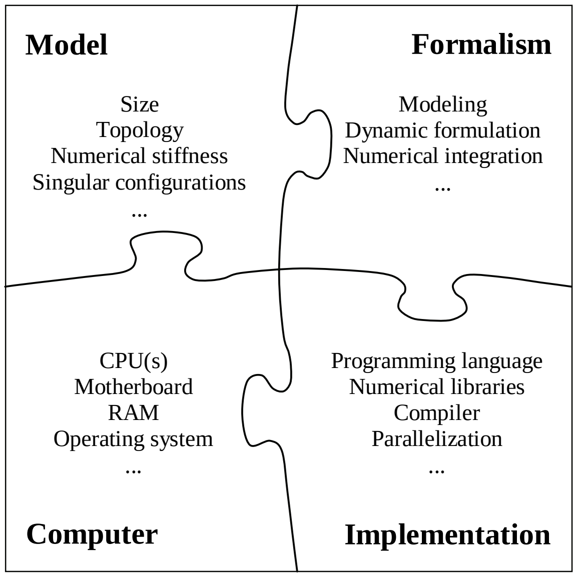


Figure 1: The simulator puzzle.

3.1. Problem

The problem to solve is a given component of the simulation process. The problem is characterized by many properties: the size (number of bodies, number of constraitns, degrees of freedom) and topology (open loop vs. closed loop; fixed vs. changing configuration), the type of constraints (scleronomous vs. rheonomous; holonomic vs. non-holonomic), the presence of redundant constraints or singular configurations, the flexibility of the bodies (rigid vs. flexible), the presence of stiffness, the presence of discontinuous effects (impacts, clearances, friction, hysteresis), etc.

3.2. Formalism

The formalism used to solve the problem involves three important 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 [2]. Another 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 state-space form, which is directly solvable by ODE methods. Descriptions and references are provided in [3,15]. These sets of DAE or ODE must be solved using a numerical integration scheme, which is the third component of the formalism.

It is important to say 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 or redundant constraints [3]. 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.

3.3. Implementation

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

The first implementation decision is to select the programming language. Fortran has been the common choice in numerical computing due to its excellent performance, but nowadays, C can compete with Fortran 77 in terms of efficiency [34], and C++ has become a rival of Fortran 90 due to the benefits of its object-oriented features and the availability of high-performance C++ compilers and libraries [35].

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. A portable way to accomplish this is using optimized compilers and numerical libraries. Modern compilers offer architectural and interprocedural optimizations that can deliver 50% speed-up for numerical simulations compared with poor compilers [36], and tuned libraries like ATLAS [37] can also help to increase performance. Sometimes, good compilers and numerical libraries are not enough, and simulation algorithms must be completely redesigned to exploit modern hardware designs [38].

Parallelization is another way to speed up MBS dynamic simulation. Its benefits have been already shown [39-41], and it does not require expensive hardware any more, since nowadays commodity, dual-CPU desktop workstations and computing cluster solutions are becoming quite affordable.

3.4. Computer

The computer is the last piece of the simulator puzzle. As explained in the previous section, its hardware characteristics (processor type and number, memory size ...) have a strong influence in the implementation of formalisms.

Similar to what happens with dynamic formulations, there is not an optimal hardware configuration for all numerical simulations: the performance of a computer depends on the characteristics of the running application. One of the causes is the hierarchical memory model used in modern commodity processors, with a growing gap between local-memory and CPU clock speeds: some CPU models perform best for applications with small data sizes, while others perform best for applications with medium or big data sizes. In MBS dynamics, the data size of the simulation code depends on the model size and the formulation used (global formulations usually need more storage). Therefore, when performance is critical, the computer must be selected taking into account the characteristics of the model and the formulation.

Measuring the performance of a dynamic simulation method by counting the number of floating-point arithmetic operations (FLOPs) per function evaluation does not make sense any longer, since in modern CPUs, the cost of FLOPs is of the same order as integer operations. In these new machines, the logic of the algorithm and an efficient use of the cache memory can be more important than the number of FLOPs [2].

4. PROPOSED BENCHMARK

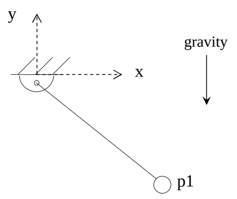
As stated in the introductory section, a benchmarking system for MBS dynamics is required. This section proposes a prototype for such a benchmark. Several test problems have been selected from examples used by authors during the last years, while others are new proposals. Reference solutions have been calculated. For problems taken from other authors, the obtained reference solution has been contrasted with the published results, provided they exist. In addition, a normalized procedure to measure performance is defined. In the next section, a web application to manage the benchmark results is presented.

4.1. Problem set

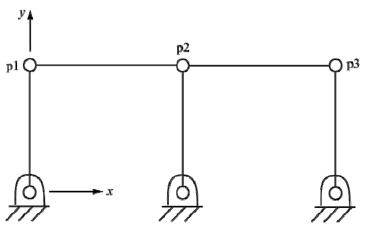
Industry demands demonstrations with complex, real-life problems, but on the other hand, small and simple problems offer several advantages: users need to invest little time to solve them, and they can isolate a specific characteristic of multibody systems (stiffness, presence of singular positions, etc.), and therefore evaluate the response of the simulation software to that particular characteristic.

It seems reasonable to split the problem collection into two main categories. The first category, named "Basic Problems", would include small, specialized problems of MBS simulation. Each of these problems should be designed to measure the ability of different simulation methods to deal with a particular characteristic of multibody systems (stiffness, singularities, etc.). They can be classified into several groups, according to the kind of multibody system: systems with rigid bodies, with flexible bodies, undergoing contact-impact, etc. This first category would serve to test new proposed methods and compare their performance with existing methods. The second category, named "Complex Problems", would include complex, real-life problems, probably involving several phenomena in a single system. This category could also be divided into several groups. It would serve to test MBS simulation tools and show results to the industrial users of them.

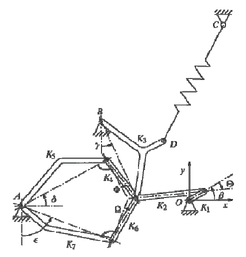
Two groups of rigid MBS are proposed in this work: group A, in the category "Basic Problems", is composed by five simple, academic problems using rigid multibody systems (Table 2). Group B, in the category "Industrial Applications", is composed by five real-life problems with rigid multibody systems (Table 3).

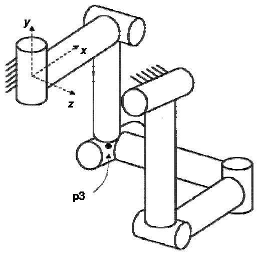


(a) Problem A01

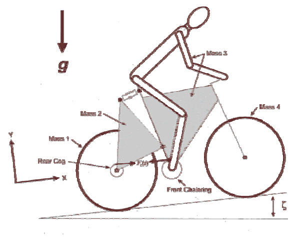


(b) Problem A02


(c) Problem A03



(d) Problem A04



(e) Problem A05

Figure 2: Basic Problems – Group A.

Table 2: Group A – Basic Problems for Rigid MBS.

Code	Name	Characteristic
A01	Pendulum	Example problem
A02	Double four bar mechanism	Singular positions
A03	Andrew's mechanism	Very small time scale
A04	Bricard's mechanism	Redundant equations
A05	Bicycle with rear suspension	Stiff system

Table 3: Group B – Complex Problems for Rigid MBS.

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

Models and test cases have been fully defined for problems in group A (see Fig. 2). Problem A01, a planar double pendulum under gravity effects, is proposed as demonstration example. Problem A02, a one degree-of-freedom assembly of two four-bar linkages, has been proposed by Bayo and Avello [25] as an example to test the robustness of a given method in cases where the mechanism undergoes singular configurations: when the mechanism reaches a horizontal position, the number of degrees of freedom instantaneously increases from 1 to 3. Many formulations have problems to overcome this situation. Problem A03, a planar mechanism composed by seven bodies and driven by a motor located at point O, has been proposed by Schiehlen as a benchmark problem [7]: it has a very small time scale, thus making it difficult to simulate for solvers that can't reach small time steps. Problem A04, known as Bricard's mechanism, is a classic example of overconstrained system: Grübler's formula gives 0 degrees-of-freedom for it, but the particular orientation of the revolute pairs yields a system with 1 degree-of-freedom. Finnaly, problem A05, a bicycle with rear suspension, is a example of stiff system proposed and studied by Good and McPhee [42].

For problems in group B, detailed models and test case definitions are available only for two of them: problems B01 and B02. Work is being conducted to finish the specifications for the remaining problems.

4.2. Reference solutions

Computational efficiency of a particular simulation method is a function of the desired accuracy, and therefore, when comparing the performance 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.

Reference solutions for problems in group A were obtained using the commercial MBS software ADAMS/Solver [43]. For each test problem, the corresponding model was build and solved using different formalisms using decreasing integrator tolerances and time steps, until converged was achieved. In addition, problems were solved using custom dynamic simulation codes programmed in Matlab, to verify the correctness of the ADAMS/Solver solution.

For problems in group B, reference solutions are much more difficult to obtain. Currently, solutions have been generated with ADAMS/Solver. Due to the complexity of these problems, other simulation tools must be used too, in order to ensure that the obtained solution can be considered as a reference solution.

4.3. Performance measurement

Since computational efficiency is a function of precision, the measurement procedure for the benchmark is as follows: each problem must be solved with *a given required accuracy*, and the CPU-time employed by the computer in solving the simulation must be measured. 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. Each problem specifies which components made up the solution:

$$\begin{array}{ll} y_j\left(t_i\right) & \textit{obtained solution for component } j \\ y_j^{ref}\left(t_i\right) & \textit{reference solution for component } j \end{array} \tag{1}$$

The error in a component at a given point is given in Eq. (2), where the threshold value is selected properly for each problem to overcome situations where the reference solution is very close to zero. The aggregated error for all components during the whole simulation is evaluated with Eq. (3).

$$e_j\left(t_i\right) = \frac{\left|y_j\left(t_i\right) - y_j^{ref}\left(t_i\right)\right|}{\max\left\{\left|y_j^{ref}\left(t_i\right)\right|, y_j^{threshold}\right\}} \tag{2}$$

$$e_{2,2} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\frac{1}{n}\sum_{j=1}^{n}\left(e_j\left(t_i\right)\right)^2} \tag{3}$$

It was decided that two precision levels could be established for all problems: a low precision level with an allowed maximum error of 10^{-1} (10% aggregated error) and a high precision level with an allowed maximum error of $5\cdot10^{-3}$ (0.5% aggregated error).

With the required precision levels defined, the benchmark allows to get comparable metrics to measure computational efficiency of a given combination of problem, method, implementation and computer. To measure the performance of a particular simulator at solving a test problem, the researcher must tune the simulator (adjusting integration step, tolerances, etc.) to achieve the required precision with a minimum CPU-time. The obtained CPU-time is directly comparable to CPU-times for the same test problem obtained with other simulators.

However, comparing CPU-times has two problems. First, in many situations it is desirable to compare performance without taking into account the power of the computer. And second, since CPU-times are proportional to simulation times, they cannot be compared between problems with different durations.

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 a custom solver implemented in Fortran 77 (the *reference solver*). The simulation last 20 seconds, and the averaged CPU-time of 3 simulation runs is measured. The utility calculates a *Hardware Performance Ratio (H.P.R.)* using Eq. (4). This ratio measures the power of the computer when used to run MBS dynamic simulations.

$$H.P.R. = \frac{simulation\ time_{reference\ problem}}{CPU\text{-}time^{reference\ solver}_{reference\ problem}} \tag{4}$$

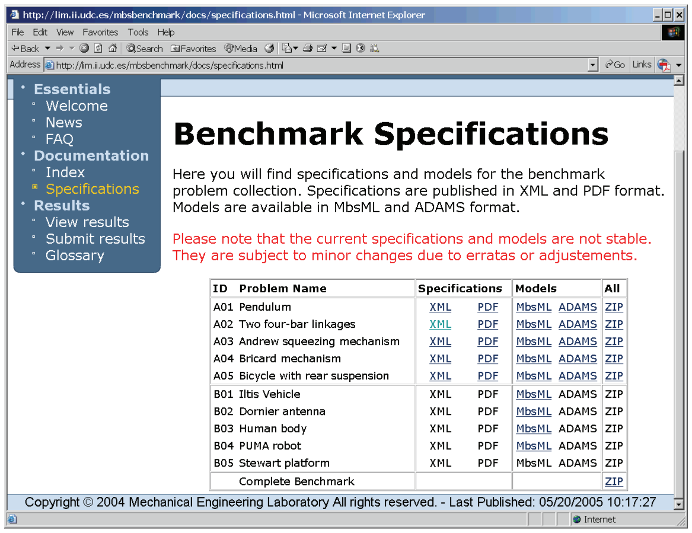
Then, the performance of a given combination of simulation method and implementation can be calculated for a given test problem with the *Software Performance Ratio (S.P.R.)* defined in Eq. (5):

$$S.P.R._{test\ problem\ i} = \left(\frac{1}{H.P.R.}\right)\cdot\frac{simulation\ time_{test\ problem\ i}}{CPU\text{-}time_{test\ problem\ i}} \tag{5}$$

This ratio is fully independent from the problem simulation time, and almost independent from the computer. The partial dependency from the computer comes from the H.P.R., which depends on the problem solved in the simulation: as explained in Section 3.3, the performance of a given computer depends on the particular characteristics of the running application.

4.4. Documentation

Finally, documentation for test problems in group A and measurement procedures has been published in a dedicated website (Fig. 3). Documentation for each problem includes: (a) problem specification (a brief description of the multibody system and the analysis to be performed, and links to other pieces of documentation); (b) detailed MBS models encoded in MbsML (an XML-based format defined by the authors [44]) and in ADAMS/Solver format (more formats will be available in the future); and (c) reference solutions (time-histories of selected variables, in tabular and graphical form) and animations of the resulting motion).


http://lim.ii.udc.es/mbsbenchmark/docs/specifications.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://lim.ii.udc.es/mbsbenchmark/docs/specifications.html
Go Links
Essentials
Welcome
News
FAQ
Documentation
Index
Specifications
Results
View results
Submit results
Glossary
Internet

Benchmark Specifications

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

Please note that the current specifications and models are not stable. They are subject to minor changes due to erratas or adjustements.

ID	Problem Name	Specifications		Models		All
A01	Pendulum	XML	PDF	MbsML	ADAMS	ZIP
A02	Two four-bar linkages	XML	PDF	MbsML	ADAMS	ZIP
A03	Andrew squeezing mechanism	XML	PDF	MbsML	ADAMS	ZIP
A04	Bricard mechanism	XML	PDF	MbsML	ADAMS	ZIP
A05	Bicycle with rear suspension	XML	PDF	MbsML	ADAMS	ZIP
B01	Iltis Vehicle	XML	PDF	MbsML	ADAMS	ZIP
B02	Dornier antenna	XML	PDF	MbsML	ADAMS	ZIP
B03	Human body	XML	PDF	MbsML	ADAMS	ZIP
B04	PUMA robot	XML	PDF	MbsML	ADAMS	ZIP
B05	Stewart platform	XML	PDF	MbsML	ADAMS	ZIP
	Complete Benchmark					ZIP

Copyright © 2004 Mechanical Engineering Laboratory All rights reserved. - Last Published: 05/20/2005 10:17:27

Figure 3: Website for benchmark documentation.

5. MANAGEMENT OF BENCHMARK RESULTS

A problem collection is not enough to build a benchmarking system for MBS dynamic simulation. Such a system must also provide a collaborative, centralized platform to collect, organize and share information about performance of different existing methods. This kind of platform is essential to publish results in a homogeneous format and to make them available to the MBS community. Today the Internet provides a ubiquitous platform for connectivity and collaboration, and it should be exploited to achieve these goals.

A prototype of a web-based management system for the proposed MBS benchmark has been developed. With this management system, users can submit performance results of a given simulator; these results are stored in a centralized database, with detailed information about the simulator used to solve the problems. The performance results stored in the database can be retrieved: users can run several types of queries to get and compare the performance results that have been submitted to the database. Users perform these tasks using a conventional web browser, so they do not need to install any special software in their computers, and the system is accessible from all over the world during all the time. The interface is quite simple and intuitive. All these features contribute to the adoption of the benchmarking system by new users.

The system is made up of three software components: (a) a relational database manager, which holds and manages the database where information about simulators and performance results is stored; (b) a web-based application, which is the link between the users and the database: it receives, interprets and executes the commands given by users through a web browser, and presents results as HTML web pages; and (c) a web server, which serves the web site to the Internet and provides an execution environment for the web application. This solution can be fully deployed at cost zero (all selected technologies and software components are freely available), and it is robust and scalable enough to allow a future implementation of the proposed prototype as an industrial-strength system suitable to be used for production.

5.1. Results database

When a user submits performance results measured for a given simulator, all the information is stored in the results database. Data are classified into eight tables. Table 4 shows the database structure and the main fields in each table. The first field in each table acts as primary key (or unique identifier), and fields in italics are links to other tables.

Table 4: Structure of the results database.

Table	Fields
Problems	ProblemID, Name, URL
Organizations	OrganizationID, Name, Adress, City, Country, URL
Users	UserID, *OrganizationID*, FirstName, LastName, Email, Password
Computers	ComputerID, *OrganizationID*, Nickname, Brand, Model, Motherboard, CPUmodel, CPUnumber, Memory, OSname, PerformanceRatio
Softwares	SoftwareID, Name, Author, URL
Builds	BuildID, *SoftwareID*, Version, BuildSystem, BuildOptions, Libraries
Methods	MethodID, *SoftwareID*, Name, Coordinates, Formulation, Integrator
Results	ResultID, *ProblemID, UserID, ComputerID, SoftwareID, BuildID, MethodID*, 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 (result submission is not anonymous).

Table *Computers* holds information about the hardware and operational system of the computer 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.

5.2. Results submission

Information is submitted to the central database by filling HTML forms from a web browser. Results submission is allowed only for registered users. In the registration process, users must provide contact information for them and their organization (university or company). Before submitting results, users must login providing name and password.

Results submission has three steps. 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. At the present moment, only problems in group A are available. Information about new computers or software systems can be entered at this stage. When the user enters information about the computer, the Hardware Performance Ratio of that computer must be provided. In the second step, the user chooses the build environment for the software and the method used to perform the dynamic simulation. Again, information about new build environments or methods can be entered. Finally, the user enters the measured CPU-time and the reached error (high precision or low precision, as specified in the problem documentation). 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).

In addition, a user can delete results that were submitted previously. This is useful if errors in the procedure to measure performance are detected after the results have been submitted, or to update results due to performance gains in the simulator.

5.3. Results retrieval and comparison

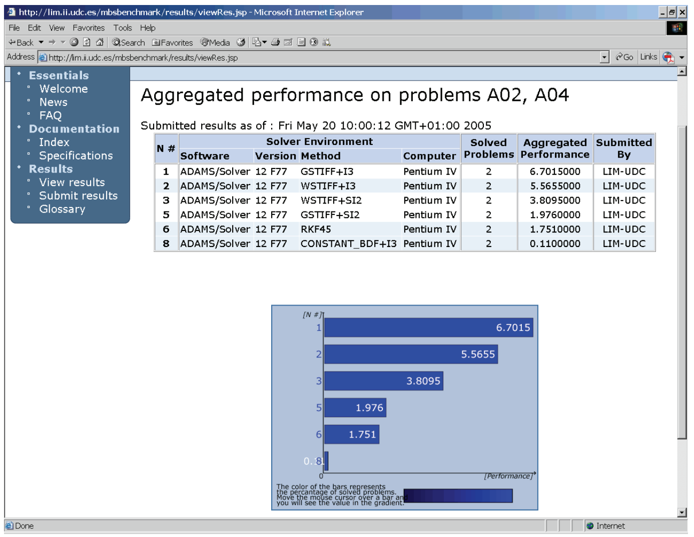
Users can retrieve performance results stored in the database. Results retrieval is anonymous. Users can run several types of queries to get and compare the performance results that have been submitted to the database:

Basic query shows all the results submitted for a test problem. Information is presented in tabular form and in graphical form using a bar graph. 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 (Fig. 4). 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 restrict the result range: the filters can be applied on any of the database fields showed in Table 4. In addition, detailed information about a particular result can be examined, including author contact information.



http://lim.ii.udc.es/mbsbenchmark/results/viewRes.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://lim.ii.udc.es/mbsbenchmark/results/viewRes.jsp
Go Links
Essentials
Welcome
News
FAQ
Documentation
Index
Specifications
Results
View results
Submit results
Glossary
Done
Internet

Aggregated performance on problems A02, A04

Submitted results as of : Fri May 20 10:00:12 GMT+01:00 2005

N #	Solver Environment				Solved Problems	Aggregated Performance	Submitted By
	Software	Version	Method	Computer			
1	ADAMS/Solver	12 F77	GSTIFF+I3	Pentium IV	2	6.7015000	LIM-UDC
2	ADAMS/Solver	12 F77	WSTIFF+I3	Pentium IV	2	5.5655000	LIM-UDC
3	ADAMS/Solver	12 F77	WSTIFF+SI2	Pentium IV	2	3.8095000	LIM-UDC
5	ADAMS/Solver	12 F77	GSTIFF+SI2	Pentium IV	2	1.9760000	LIM-UDC
6	ADAMS/Solver	12 F77	RKF45	Pentium IV	2	1.7510000	LIM-UDC
8	ADAMS/Solver	12 F77	CONSTANT_BDF+I3	Pentium IV	2	0.1100000	LIM-UDC

The color of the bars represents the percantage of solved problems. Move the mouse cursor over a bar and you will see the value in the gradient.

Figure 4: Query of performance results: aggregated performance.

6. APPLICATION TO ADAMS

In order to test the validity of the proposed benchmark, it was applied to measure the performance of the commercial MBS simulation tool ADAMS/Solver [43]. This software provides a wide range of dynamic simulation methods and it is the most popular MBS simulation tool. The performance results generated from these tests can be used as a baseline for future comparisons of other MBS analysis codes.

ADAMS/Solver provides several formulations and integrators. In this numerical experiment, four integrators have been tested. Three of them (named GSTIFF, WSTIFF and CONSTANT_BDF) are stiff, variable-order, variable-step, multi-step integrators based on Backward-Difference Formulae (BDF) that use a predictor-corrector scheme. These integrators have been combined with two formulations: I3 (index-3 formulation) and SI2 (stabilized-index two). The other integrator, ABAM (Adams Bashforth-Adams Moulton), is a non-stiff, variable-order, variable-step integrator that is used with a coordinate partitioning formulation (CP) to reduce the system of differential algebraic equations (DAE) to a condensed set of ordinary differential equations (ODE). Detailed information about the integrators and formulations can be found in the ADAMS/Solver User's Guide.

All test problems in group A were solved to measure CPU-times for low and high precision levels, following the benchmark specifications. Simulations were performed using different combinations of integrators and formulations. All performance results generated from the numerical experiments with ADAMS/Solver simulation methods have been loaded in the web-based application. Results for low and high precision levels are shown in Table 5 and Table 6. The lowest CPU-time in each problem is highlighted with bold font. If the lower precision obtained with a method is higher than the required precision, its CPU-time is between parentheses.

Table 5: CPU-time (s) for different simulation methods (low precision, error < 10^{-1}).

Problem	A01	A02	A03	A04	A05
GSTIFF + I3	33.8	**2.5**	-	**8.0**	(3.35)
GSTIFF + SI2	30.9	-	64.5	18.0	(5.16)
WSTIFF + I3	**23.8**	3.0	-	9.8	(3.38)
WSTIFF + SI2	26.7	4.5	64.1	15.7	(5.45)
CONSTANT_BDF + I3	818.2	179.0	-	548.9	**1.68**
CONSTANT_BDF + SI2	617.8	-	95.7	-	2.72
ABAM	-	-	**47.5**	-	(6.83)

Table 6: CPU-time (s) for different simulation methods (high precision, error < $5 \cdot 10^{-3}$).

Problem	A01	A02	A03	A04	A05
GSTIFF + I3	53.8	**2.7**	-	**8.1**	**3.35**
GSTIFF + SI2	47.1	-	96.3	37.8	5.16
WSTIFF + I3	40.0	3.3	-	9.8	(3.38)
WSTIFF + SI2	**35.2**	4.9	87.7	16.0	(5.45)
CONSTANT_BDF + I3	-	-	-	-	3.72
CONSTANT_BDF + SI2	-	-	-	-	6.19
ABAM	-	-	**59.6**	-	(6.83)

For low precision, two combinations of integrator and formulation (GSTIFF+I3, the default method, and WSTIFF+I3) are the fastest methods. For high precision, the ABAM integrator wins in difficult cases, like problem A03. Results show that problem A05 can be solved easily with all tested methods, and some methods reach accuracies even higher than the high precision level. This suggests that A05 is not a good candidate for a test problem, and should be retired from the benchmark collection.

Results also show that for problem A01 and A05, the winner methods for low and high precision levels are different. This fact shows the importance of specifying the required accuracy when comparing the computational efficiency of different simulation methods. In some cases, the CPU-time has a weak dependency from the precision, but in other cases the CPU-time increases by a factor greater than 2 when moving from low precision to high precision.

7. CONCLUSIONS AND FUTURE WORK

A prototype of a benchmarking system for MBS simulation tools have been proposed. The system is made up by two components:

(a) Benchmark for MBS dynamics. The structure of the benchmark problem collection was defined, and several test problems were proposed. A procedure to measure the computational efficiency of a simulator in a comparable way was described. For problems included in the group “Basic Problems”, documentation and reference solutions were generated.

(b) Web-based management system for benchmark results. This prototype demonstrator allows to store benchmark results in a relational database and manage them using a friendly web interface. From a web browser, users can submit results to the database and retrieve submitted results to compare the performance of different simulators. Information is presented in an 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.

The benchmarking system has been used to evaluate the performance of the commercial simulation tool ADAMS/Solver. Since this tool is the market leader, results can be used as a baseline for future comparisons. All problems except one (it will be replaced with a different problem) were able to reach the limits of some simulation methods, and therefore they can be considered good benchmark problems. Experiments also demonstrated that computational efficiencies measured under different accuracy requirements are not comparable at all.

Some guidelines for future work are:

(a) To generate documentation and validated reference solutions for problems in group “Complex Problems”. Due to the complexity of these problems, this task requires the use of several simulation tools and, probably, the participation of several research teams.

(b) To define new groups of problems to benchmark simulator performance when dealing with other phenomena like flexibility, contact, impacts, etc. Some good test problems of these types have been proposed during the last years: reference solutions must be obtained and standardized problem documentation must be generated.

(c) To develop software tools to automate the benchmarking procedure. Some tasks that should 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 MBS simulation software could use the benchmarking system to control the quality and monitor the improvements in every new release: without human intervention, all problems in the benchmark would be solved and performance results would be computed and submitted to the central database automatically. In few minutes, developers would get an overall view of the performance of the new software release.

(d) To conduct evaluations with commercial MBS simulation tools other than ADAMS/Solver, especially for problems in group "Complex Problems". Existing academic simulation tools could also be benchmarked. Results would be of great value for industrial users of these tools.

REFERENCES

[1] Haug, E., 1993, "High Speed Multibody Dynamic Simulation and Its Impact on Man-Machine Systems," in *Advanced multibody system dynamics: simulation and software tools*, W.Schiehlen (ed.), pp. 1-18.

[2] García de Jalón, J., and Bayo, E., 1994, *Kinematic and Dynamic Simulation of Multibody Systems - The Real-Time Challenge*, Springer-Verlag, New York.

[3] Cuadrado, J., Cardenal, J., and Morer, P., 1997, "Modeling and Solution Methods for Efficient Real-Time Simulation of Multibody Dynamics," Multibody System Dynamics, **1**(3), pp. 259-280.

[4] Bae, D. S., Lee, J. K., Cho, H. J., and Yae, H., 2000, "An Explicit Integration Method for Realtime Simulation of Multibody Vehicle Models," Computer Methods in Applied Mechanics and Engineering, **187**(1-2), pp. 337-350.

[5] García Orden, J. C., and Goicolea, J. M., 2000, "Conserving Properties in Constrained Dynamics of Flexible Multibody Systems," Multibody System Dynamics, **4**(3), pp. 225-244.

[6] Anderson, K. S., and Critchley, J. H., 2003, "Improved 'Order-N' Performance Algorithm for the Simulation of Constrained Multi-Rigid-Body Dynamic Systems," Multibody System Dynamics, **9**(2), pp. 185-212.

[7] W.Schiehlen (ed.), 1990, *Multibody Systems Handbook*, Springer-Verlag, Dordrecht.

[8] Kortum, W., and Sharp, R. S., 1991, "A Report on the State-Of-Affairs on Application of Multibody Computer Codes to Vehicle System Dynamics," Vehicle System Dynamics, **20**(3-4), pp. 177-184.

[9] Kortum, W., and Sharp, R. S., 1993, *Multibody Computer Codes in Vehicle System Dynamics*, Swets and Zeitlinger Publishers.

[10] Anderson, R. J., 1990, "Iltis Benchmark Proposal," Dep. of Mechanical Engineering, Queen's University, Kingston, Ontario, Canada.

[11] Hiller, M., and Frik, S., 1993, "Road Vehicle Benchmark 2 - Five-Link Suspension," Vehicle System Dynamics, **22**(suppl.issue), pp. 254-262.

[12] Quirt, R. C., and Anderson, R. J., 1992, "Comparisons of A Linear With A Nonlinear Multibody Simulation of An Off-Road Vehicle," Vehicle System Dynamics, **20**, pp. 490-503.

[13] Langlois, R. G., Hanna, D. M., and Anderson, R. J., 1992, "Implementing Preview Control on An Off-Road Vehicle With Active Suspension," Vehicle System Dynamics, **20**, pp. 340-353.

[14] Schiehlen, W., 1992, "Prospects of the German Multibody System Research-Project on Vehicle Dynamics Simulation," Vehicle System Dynamics, **20**, pp. 537-550.

[15] Cuadrado, J., Cardenal, J., Morer, P., and Bayo, E., 2000, "Intelligent Simulation of Multibody Dynamics: Space-State and Descriptor Methods in Sequential and Parallel Computing Environments," Multibody System Dynamics, **4**(1), pp. 55-73.

[16] Schwab, A. L., and Meijaard, J. P., 1999, "Dynamics of Flexible Multibody Systems Having Rolling Contact: Application of the Wheel Element to the Dynamics of Road Vehicles," Vehicle System Dynamics, **33**, pp. 338-349.

[17] Schumann, A. R., and Anderson, R. J., 2002, "Optimal Control of an Active Anti Roll Suspension for an Off-Road Utility Vehicle Using Interconnected Hydragas Suspension Units," Vehicle System Dynamics, **37**, pp. 145-156.

[18] von Schwerin, R., 1999, *Multibody System Simulation: Numerical Methods, Algorithms and Software*, Springer-Verlag.

[19] Minaker, B., and Anderson, R. J., 1999, "Modelling the Dynamics of a Vehicle With Active Geometry Suspension," Vehicle System Dynamics, **33**, pp. 716-727.

[20] Rodríguez, I., 2000, "Análisis Eficiente De Mecanismos 3D Con Métodos Topológicos y Tecnología De Componentes En Internet," Ph.D. Dissertation, Universidad de Navarra, San Sebastián, Spain.

[21] Simeon, B., 1996, "On the Numerical Solution of a Wheel Suspension Benchmark Problem," Journal of Computational and Applied Mathematics, **66**(1-2), pp. 443-456.

[22] Iwnicki, S., 1998, "Manchester Benchmarks for Rail Vehicle Simulation," Vehicle System Dynamics, **30**(3-4), pp. 295-313.

[23] Iwnicki, S., 1999, "The Manchester Benchmarks for Rail Vehicle Simulation," Vehicle System Dynamics, **31**, p. 1.

[24] Rail Technology Unit, Manchester Metropolitan University, 1998, "The Manchester Benchmarks for Rail Vehicle Simulation," http://www.sci-eng.mmu.ac.uk/rtu/.

[25] Bayo, E., and Avello, A., 1994, "Singularity-Free Augmented Lagrangian Algorithms for Constrained Multibody Dynamics," Nonlinear Dynamics, **5**(2), pp. 209-231.

[26] Jahnke, M., Popp, K., and Dirr, B., 1993, "Approximate Analysis of Flexible Parts in Multibody Systems Using the Finite Element Method," W.Schiehlen (ed.), Kluwer Academic Publishers, Dordrecht, Netherlands, pp. 237-256.

[27] Simeon, B., 2001, "Numerical Analysis of Flexible Multibody Systems," Multibody System Dynamics, **6**(4), pp. 305-325.

[28] Cuadrado, J., Gutierrez, R., Naya, M. A., and Morer, P., 2001, "A Comparison in Terms of Accuracy and Efficiency Between a MBS Dynamic Formulation With Stress Analysis and a Non-Linear FEA Code," International Journal for Numerical Methods in Engineering, **51**(9), pp. 1033-1052.

[29] Schaub, M., and Simeon, B., 2002, "Automatic H-Scaling for the Efficient Time Integration of Stiff Mechanical Systems," Multibody System Dynamics, **8**(3), pp. 329-345.

[30] Bauchau, O. A., and Theron, N. J., 1996, "Energy Decaying Scheme for Non-Linear Beam Models," Computer Methods in Applied Mechanics and Engineering, **134**(1-2), pp. 37-56.

[31] Bauchau, O. A., and Bottasso, C. L., 1999, "On the Design of Energy Preserving and Decaying Schemes for Flexible, Nonlinear Multi-Body Systems," Computer Methods in Applied Mechanics and Engineering, **169**(1-2), pp. 61-79.

[32] Bottasso, C. L., Borri, M., and Trainelli, L., 2001, "Integration of Elastic Multibody Systems by Invariant Conserving/Dissipating Algorithms. II. Numerical Schemes and Applications," Computer Methods in Applied Mechanics and Engineering, **190**(29-30), pp. 3701-3733.

[33] Tregenza, J. E., and Anderson, R. J., 1990, "Iltis Data Package," DL/SPEC/90/1, Dep. of Mechanical Engineering, Queen's University, Kingston, Ontario, Canada.

[34] Theurich, G., Anson, B., Hill, N. A., and Hill, A., 2001, "Making the Fortran-to-C Transition: How Painful Is It Really?," Computing in Science & Engineering, **3**(1), pp. 21-27.

[35] Cary, J. R., Shasharina, S. G., Cummings, J. C., Reynders, J. V. W., and Hinker, P. J., 1997, "Comparison of C++ and Fortran 90 for Object-Oriented Scientific Programming," Computer Physics Communications, **105**(1), pp. 20-36.

[36] Yu, J. S. K., and Yu, C. H., 2002, "Recent Advances in PC-Linux Systems for Electronic Structure Computations by Optimized Compilers and Numerical Libraries," Journal of

Chemical Information and Computer Sciences, **42**(3), pp. 673-681.

[37] Whaley, R. C., Petitet, A., and Dongarra, J. J., 2001, "Automated Empirical Optimizations of Software and the ATLAS Project," Parallel Computing, **27**(1-2), pp. 3-35.

[38] Becker, C., Kilian, S., and Turek, S., 1999, "Consequences of Modern Hardware Design for Numerical Simulations and Their Realization in FEAST," Euro-Par'99, Parallel Processing, 5th International Euro-Par Conference, Proceedings (Lecture Notes in Computer Science Vol.1685), pp. 643-650.

[39] Eichberger, A., Fuhrer, C., and Schwertassek, R., 1993, "The Benefits of Parallel Multibody Simulation and Its Application to Vehicle Dynamics," Advanced multibody system dynamics: simulation and software tools, pp. 107-126.

[40] Duan, S., and Anderson, K. S., 2000, "Parallel Implementation of a Low Order Algorithm for Dynamics of Multibody Systems on a Distributed Memory Computing System," Engineering with Computers, **16**(2), pp. 96-108.

[41] Quaranta, G., Masarati, P., and Mantegazza, P., 2002, "Multibody Analysis of Controlled Aeroelastic Systems on Parallel Computers," Multibody System Dynamics, **8**(1), pp. 71-102.

[42] Good, C., and McPhee, J., 1999, "Dynamics of Mountain Bicycles With Rear Suspensions: Modeling and Simulation.," Sports Engineering, **2**, pp. 129-143.

[43] MSC.Software Corporation, 2004, "ADAMS," http://www.mscsoftware.com/.

[44] Gonzalez, M., Dopico, D., and Cuadrado, J., 2004, "A New Software Environment for MBS Simulation Based on XML and Integrated With CAD/CAE Packages," Eleventh World Congress in Mechanism and Machine Science, Vols 1-5, Proceedings, pp. 642-646.