

MBDyn Installation Manual

Version 1.2.1

Pierangelo Masarati

DIPARTIMENTO DI INGEGNERIA AEROSPAZIALE
POLITECNICO DI MILANO

Automatically generated August 16, 2004

Contents

1	Introduction	3
2	Getting the package	4
2.1	MBDyn	4
2.2	Mathematical Utilities	4
2.2.1	ATLAS	4
2.2.2	BLAS	5
2.2.3	Metis	5
2.3	Linear Solvers	5
2.3.1	Naive	5
2.3.2	Y12	5
2.3.3	Umfpack	5
2.3.4	Lapack	6
2.3.5	SuperLU (experimental)	6
2.3.6	TAUCS (experimental)	6
2.3.7	Harwell (historical)	6
2.3.8	Meschach (historical)	6
2.4	Utilities	6
2.5	Communication	6
2.5.1	MPI	6
2.5.2	Metis	6
2.6	Real-Time	6
2.6.1	RTAI	6
3	Building	7
3.1	Configuring	7
3.1.1	Option List	7
3.1.2	Multithread Assembly/Solution	8
3.1.3	Schur Parallel Solver	8
3.1.4	Real-Time Simulator	9
4	Installing	10

5	Executing	11
5.1	Regular Execution	11
5.2	Parallel Execution	11
5.3	Real-Time Execution	11
5.4	External Execution	11
6	Troubleshooting	12
7	HOWTOs	13
7.1	Schur Solver	13
7.2	Real-Time Simulator	15
7.3	Simulink Interface	16
8	Developers	18

Chapter 1

Introduction

This document describes how to download, build, install and execute MBDyn — MultiBody Analysis program, a suite of tools for multibody/multidisciplinary analysis of complex systems.

For any question or problem, to fix typos, bugs, for comments and suggestions, please contact the Development Team without hesitation:

Pierangelo Masarati,
MBDyn Development Team
Dipartimento di Ingegneria Aerospaziale
Politecnico di Milano
via La Masa 34, 20156 Milano, Italy
Fax: +39 02 2399 8334
E-mail: mbdyn@aero.polimi.it
Web: <http://www.aero.polimi.it/~mbdyn/>

This document is also available online at
<http://mbdyn.aero.polimi.it/~masarati/MBDyn-input/install/>

Chapter 2

Getting the package

2.1 MBDyn

The package can be downloaded in source form from

<http://mbdyn.aero.polimi.it/~mbdyn/>

Binary releases and snapshots are also available for Windows 2000/XP at <http://mbdyn.aero.polimi.it/~masarati/Download/mbdyn/>, compiled with Cygwin (see <http://www.cygwin.com/>) and thus require `cyglttdl-3.dll`; they are provided to Windows users to save them the burden of installing Cygwin (very easy and straightforward, though) and to compile a package under an unfamiliar environment. However, no support is provided for those builds, unless problems are easily identifiable as related to the sources and not to the OS, and they impact the UN*X version as well.

MBDyn may use a wide range of packages, if available on the host system and correctly detected by `configure`.

2.2 Mathematical Utilities

MBDyn may exploit the availability of some mathematical utilities; neither of them is required for a basic compilation, but they may be useful or required for specific features.

2.2.1 ATLAS

The Automatically Tunable Linear Algebra Subroutines are a replacement of the standard BLAS. They are exploited by the linear solver Umfpack (see Section 2.3.3) and can be used by Lapack (see Section ??) and other packages that require BLAS (see Section 2.2.2).

2.2.2 BLAS

The Basic Linear Algebra Subroutines can be used by Umfpack (see Section 2.3.3) and other packages. Specially tuned binaries for each architecture (processor type, cache size and otehr special hardware features) are advisable; otherwise, instead of compiling them of your own, ATLAS (see Section 2.2.1) are considered a better replacement.

2.2.3 Metis

2.3 Linear Solvers

MBDyn may use a variety of linear solvers

2.3.1 Naive

The Naive solver is also native. It is especially meant for medium size problems (between 100 and 1000 unknowns) and is essentially a sparse solver optimized for speed rather than memory usage.

2.3.2 Y12

The Y12 solver is also builtin. the MBDyn Project distributes the Y12 library **AS IS** and **WITHOUT ANY WARRANTY** as part of the source code, with no copyright statement and no license. Of course credits go to the original Authors: Zahari Zlatev, Jerzy Wasniewski, and Kjeld Schaumburg, Comp. Sci., Math. Inst., University of Aarhus, Ny Munkegade, DK 8000 Aarhus. This solver is recommended for moderate to large problems, if Umfpack (see Section 2.3.3) is not available.

2.3.3 Umfpack

The Umfpack linear sparse solver library must be downloaded separately, from <http://www.cise.ufl.edu/research/sparse/umfpack/>. Credit goes to Timothy A. Davis, University of Florida. Umfpack is used by permission; please read its Copyright, License and Availability note. It is used as the standard sparse solver by MATLAB (see <http://www.mathworks.com/>). This solver is recommended for very large problems, and as a general purpose solver.

- 2.3.4 Lapack**
- 2.3.5 SuperLU (experimental)**
- 2.3.6 TAUCS (experimental)**
- 2.3.7 Harwell (historical)**
- 2.3.8 Meschach (historical)**
- 2.4 Utilities**
- 2.5 Communication**
 - 2.5.1 MPI**
 - 2.5.2 Metis**
- 2.6 Real-Time**
 - 2.6.1 RTAI**

Chapter 3

Building

The configuration of the MBDyn package is based on GNU's autotools (see <http://www.gnu.org/> for details).

3.1 Configuring

3.1.1 Option List

Specific options (from `configure --help`):

```
--enable-debug          enable debugging (no)
--with-debug-mode[={none|mem}]  with debug mode {none|mem} (none)
--enable-socket-drives  enable socket drives (auto)
--enable-runtime-loading  enable runtime loading (auto)
--with-static-modules    build (known) modules as static (auto)
--enable-crypt          enable crypt (deprecated) (no)
--enable-schur          enable Schur parallel solver
                        (needs MPI and either Metis or Chaco) (auto)
--enable-multithread    enable multithread solution (no)
--enable-adams          enable MSC.ADAMS output (no)
--enable-motionview    enable Altair's Motion View output (no)
--with-tcl              with tcl interpreters (auto)
--with-libf2c[={f2c|g2c}]  with f2c library (auto)
--with-fs[={unix|dos}]  filesystem type (unix)
--with-mpi              with MPI support (=pmpi for profiling) (auto)
--enable-debug-mpi      enable MPI debugging (no)
--with-metis            with Metis model partitioning support (auto)
--with-threads          with threads (auto)
--with-rtai             with RTAI support (no)

math libraries:
--with-blas              with (C)BLAS math library (auto)
--with-goto=lib(s)      with Goto BLAS implementation
--with-ginac            with GiNaC support
```

(ginac-config must be in \$PATH) (auto)

linear algebra solvers (naive is enabled by default):

--with-y12 with Y12 sparse math library (yes)
--with-umfpack with Umfpack math library (auto)
--with-lapack with LAPACK math library (auto)
--with-hsl with HSL (Harwell) sparse math library - historical (auto)
--with-meschach with Meschach math library - historical (auto)
--with-superlu with SuperLU math library - eXperimental (auto)

misc security libraries:

--with-pam with PAM support (auto)
--with-sasl2 with Cyrus SASL2 support (auto)

supported features:

--with-struct with structural elements (yes)
--with-elec with electric stuff (yes)
--with-aero with aerodynamic stuff (yes)
--with-aero-output={std,gauss,node} aerodynamic output mode (auto)
--with-hydr with hydraulic stuff (yes)

--with-module=<list> build listed modules (see modules/)

3.1.2 Multithread Assembly/Solution

The switch `--enable-multithread` enables multithreaded assembly; it defaults to `auto`, i.e. if the system meets the following requirements, it is automatically enabled. Essentially, a working `-lpthread` library and a POSIX compliant `pthread.h` header must be available. Currently, only the SuperLU sparse threaded solver is available; it is experimental. A threaded implementation of the builtin `naive` sparse solver is under development.

3.1.3 Schur Parallel Solver

The Schur parallel solver is enabled by using the switch `--enable-schur`. It requires a working MPI library with the `ch` driver and the C++ interface, and a partitioning library. The MPI library is selected by the switch `--with-mpi`, which allows to specify the value `pmpi` if the profiling version of the library is to be used. Note that recent MPI releases by default only build the profiling version of the C++ library, so the `pmpi` value is mandatory. Currently, the only partitioning library that is supported by MBDyn is METIS; a patch to support Chaco is being incorporated, and may be available in future releases. The Schur solver is not compatible with the multithreaded assembly/solution, so if no switch is specified and the system meets the requirements for both, the multithreaded assembly wins over the Schur solver. As a consequence, the `schur` solver should be explicitly enabled.

3.1.4 Real-Time Simulator

The real-time simulator is enabled by using the switch `--with-rtai`, which essentially detects the availability of the mandatory headers of the Linux Real-Time Application Interface. These headers changed between 2.4.13 and 3.X; both versions are automatically detected.

Chapter 4

Installing

Chapter 5

Executing

5.1 Regular Execution

5.2 Parallel Execution

5.3 Real-Time Execution

5.4 External Execution

Chapter 6

Troubleshooting

Chapter 7

HOWTOs

This chapter contains mini-howtos about typical significant configurations of MBDyn as performed by the developers. Feel free to contribute your own if you had to do any unusual configuring to meet special needs.

7.1 Schur Solver

This section describes how the Schur parallel solver available within the MBDyn package has been successfully compiled and executed on a dual Athlon SMP machine. By no means it is intended to suggest how the related packages should be built for other purposes, nor it may represent a replacement for the original build and install procedures. Please refer to the documentation available with each package for more details or for troubleshooting.

Software prerequisites:

- gcc/g++/g77 \geq 3.0 (tested with gcc/g++/g77 3.2.1, 3.3.1 and 3.4.0)

Software requirements:

- MBDyn 1.2.1
- mpich 1.2.5.2
- Metis 4.0

MPI

- download `mpich.tar.gz` from <http://www-unix.mcs.anl.gov/mpi/>
- untar the package in a temporary directory
- configure the package with

```
$ ./configure -rsh=ssh --disable-f77 --prefix=/usr/local/mpich
```

- `make` the package
- `make install`
- edit `/usr/local/mpich/share/machines.<arch>` to enumerate the max number instances of the MBDyn process you want to allow on each machine; in my case `<arch>` is LINUX (more details in `/usr/local/mpich/doc/mpichman-chp4.pdf`).

Metis

- download XXX from <http://www-users.cs.umn.edu/~karypis/metis/metis/>
- untar it in a temporary directory
- apply the patch `metis-namespace-cleanup.patch` from <http://mbdyn.aero.polimi.it/~masarati/Download/mbdyn/>
- `make` the package
- ...
- copy the library `libmetis.a` in `/usr/local/lib` and the header files `defs.h`, `macros.h`, `metis.h`, `proto.h`, `rename.h` and `struct.h` in `/usr/local/include/metis`.

MBDyn

- download `mbdyn-1.2.1.tar.gz` from <http://mbdyn.aero.polimi.it/~masarati/Download/mbdyn/>
- untar it in a temporary directory
- make sure the compiler can find headers from MPI and Metis by defining

```
CPPFLAGS="-I/usr/local/mpich/include \
-I/usr/local/mpich/include/mpi2c++ \
-I/usr/local/include/metis"
```

- make sure the linker can find the libraries from MPI and Metis by defining

```
LDFLAGS="-L/usr/local/mpich/lib \
-L/usr/local/lib"
```

- configure MBDyn by running

```
$ ./configure --with-mpi=mpich --with-metis --enable-schur \
--prefix=/usr/local
```

- `make` the package
- `make install` the package

Run It!

To test the system, one needs a test input file; the example `cantilever2` from <http://www.aero.polimi.it/~mbdyn/documentation/examples/> should do the trick. The input file does not need any specific change, unless special features are required. To run it on an SMP machine, simply execute

```
$ /usr/local/mpich/bin/mpirun -np 2 mbdyn -f cantilever2 -o /tmp/ -ss
```

This generates a set of files `/tmp/cantilever2.0.*` and `/tmp/cantilever2.1.*` with the outputs from processes 0 and 1.

7.2 Real-Time Simulator

This section describes how the Real-Time simulator available within the MBDyn package has been successfully compiled and executed. By no means it is intended to suggest how the related packages should be built for other purposes, nor it may represent a replacement for the original build and install procedures. Please refer to the documentation available with each package for more details or for troubleshooting.

Software prerequisites:

- `gcc/g++/g77 >= 3.0` (tested with `gcc 3.3.1` and `3.4.0`)

Software requirements:

- MBDyn 1.2.1
- RTAI 3.0

RTAI

- download `rtai-3.0.tar.gz` from <http://www.rtai.org/>
- untar the package in a temporary directory
- configure the package with

```
$ ./configure
```

- make the package
- make install

MBDyn

- download `mbdyn-1.2.1.tar.gz` from <http://mbdyn.aero.polimi.it/~masarati/Download/mbdyn/>
- untar it in a temporary directory

- make sure the compiler can find headers from RTAI by defining

```
CPPFLAGS="-I/home/realtime"
```

- configure MBDyn by running

```
$ ./configure --with-rtai
```

- make the package
- make install the package

Run It!

To test the system, one needs a test input file; the example RT-MBDyn/pendulum from <http://www.aero.polimi.it/~mbdyn/documentation/examples/> should do the trick. Simply execute

```
$ mbdyn -f pendulum -ss
```

...

7.3 Simulink Interface

This section describes how the Simulink Interface available within the MBDyn package has been successfully compiled and executed. By no means it is intended to suggest how the related packages should be built for other purposes, nor it may represent a replacement for the original build and install procedures. Please refer to the documentation available with each package for more details or for troubleshooting.

Software prerequisites:

- gcc/g++/g77 \geq 3.0 (tested with gcc 3.3.1 and 3.4.0)
- Matlab/Simulink (tested with XXX)

Software requirements:

- MBDyn 1.2.1

MBDyn

- download `mbdyn-1.2.1.tar.gz` from <http://mbdyn.aero.polimi.it/~masarati/Download/mbdyn/>
- untar it in a temporary directory
- configure MBDyn by running

```
$ ./configure
```

- `make` the package
- `make install` the package
- in directory `contrib/SimulinkInterface`, edit the file `Makefile` such that the appropriate `mex` compiler is used; in my system it is `/opt/matlab/bin/mex`
- `make` the package

Run It!

To test the system, one needs a test input file; the example `pendulum` in the subdirectory `examples` should do the trick.

- start Matlab
- add the subdirectory `contrib/SimulinkInterface` to Matlab's path, by executing

```
> path('<path/to>/contrib/SimulinkInterface', path);
```

- execute simulink by clicking on the related icon, or by running

```
> simulink
```

- open the model by clicking `File->Open` on the menubar
- run the model by clicking `Simulation->Start` on the menubar. some times, the first run fails; we're still trying to track that down. In case, just try again...
- for more details on creating your own model, or on editing the interface parameters, refer to the `README` in `contrib/SimulinkInterface`
- an analogous interface with Scicos is under development.

Chapter 8

Developers

Bibliography