



# MBDyn

a time-domain tool for multibody simulations

V. Boschi

# MBDyn

## Introduction

**MultiBody Dynamics** analysis software is an *open-source* tool, widely used in aerospace industry, that features the integrated multidisciplinary analysis of multibody systems, including nonlinear mechanics of rigid and flexible constrained bodies, using numerical integration.

Features:

- It allows to simulate the behavior of heterogeneous mechanical systems based on first principles equations.
- Command-line, direct initial-value problem solver with a large, flexible multifield element library
- Motion visualization through EasyAnim 3D graphical interface
- Simulink Integration

# MBDyn

## Introduction

MBDyn is essentially a programming language with its own compiler and parser.

### Integrator properties

```

1  begin: data;
2    integrator: multistep; # the default
3  end: data;
4
5  begin: multistep;
6    initial time: 0.;
7    final time: 500.;
8    strategy: no change;
9    time step: 1.e-2;
10
11   max iterations: 100;
12   tolerance: 1.e-6;
13 end: multistep;

```

```

14 begin: control data;
15   structural nodes:
16     +1 # node in the ground
17     +1 # node in the base
18     +1 # node in the load;
19   rigid bodies:
20     +1 # node in the base
21     +1 # node in the load;
22   joints:
23     +1 # clamp
24     +1 # prismatic
25     +1 # base driver
26     +1 # spring;
27   gravity;
28 end: control data;

```

### Element Declarations

### Variable Declarations

```

29 set: integer GroundNode = 1000;
30 set: integer GroundJoint = 4000;
31 set: integer LoadFrame = 1001;
32 set: integer LoadNode = 2001;
33 set: integer LoadBody = 3001;
34 set: integer Spring = 5001;
35 set: integer BaseNode = 2004;
36 set: integer BaseBody = 3004;
37 set: integer BaseJoint = 4004;
38 set: integer BaseDrive = 5004;
39 set: real f0 = .1;
40 set: real M = 1000;
41 set: real m = 906/4;
42 set: real J_xx= 100;
43   set: real J_yy= 100;
44 set: real J_zz= 100;
45 set: real J_M= 1000;
46 set: real Lz = 1;
47 set: real kz = (2*pi*f0)^2*m;
48 set: real kx = 1e10;

```

```

49 reference: BaseFrame,
50 reference, global, null,
51 reference, global, eye,
52 reference, global, null,
53 reference, global, null;

54 reference: LoadFrame,
55 reference, BaseFrame, 0, 0, Lz,
56 reference, BaseFrame, eye,
57 reference, BaseFrame, null,
58 reference, BaseFrame, null;

```

### Reference Frames

### Nodes

```

begin: nodes;
  structural: GroundNode, static,
    reference, global, null,
    reference, global, eye,
    reference, global, null,
    reference, global, null;
  structural: BaseNode, dynamic,
    reference, BaseFrame, null,
    reference, BaseFrame, eye,
    reference, BaseFrame, null,
    reference, BaseFrame, null;
  structural: LoadNode, dynamic,
    reference, LoadFrame, null,
    reference, LoadFrame, eye,
    reference, LoadFrame, null;
end: nodes;

```

```

begin: elements;
  body: BaseBody, BaseNode,
    M,
    reference, BaseFrame, null,
    diag, J_M, J_M, J_M;

  body: LoadBody, LoadNode,
    m,
    reference, LoadFrame, null,
    diag, J_xx, J_yy, J_zz;

  joint: GroundJoint, clamp, GroundNode, node, node;

  joint: BaseJoint, prismatic,
    GroundNode,
    BaseNode;

```

### Elements

```

joint: BaseDrive, drive displacement,
  BaseNode,
  reference, BaseFrame, null,
  GroundNode,
  reference, BaseFrame, null,
  0,0,1, random,
    0.01, # amplitude
    0, # mean value
    1e-2, # initial time
    forever, # final time
    steps, 1, # steps
    seed, 0; # seed

joint: Spring, deformable displacement joint,
  BaseNode,
    reference, BaseFrame, null,
  hinge, reference, BaseFrame, eye,
  LoadNode,
    reference, LoadFrame, null,
  hinge, reference, LoadFrame, eye,
  linear elastic generic, diag , kx,kx,kz,
  prestress, 0,0,-kz*Lz-m*9.81;

```

```

gravity: 0., 0., -1., const, 9.81;
end: elements;

```

# MBDyn

## Matlab visualization tools

The compiler produce a .mov file that contains in ASCII format  $N_{\text{node}} \times N_{\text{timesteps}}$  lines formatted as:

- the node label
- the three coordinates of the position of the node
- the three Euler-like angles that define the orientation of the node (following the 1, 2, 3 convention)
- the three components of the velocity of the node
- the three components of the angular velocity of the node

For this reason an handy script that load MBDyn simulation data into the e2e global structure, used by Virginio's e2etools, has been developed.

Here are the most useful scripts:

- ***MBDynLoad***(*FileName, Nodes, SelectedNode, TimeStep[, Points]*)

Load MBDyn simulation data into the e2e global structure.

- ***MBDynTF***(*FileName, Nodes, InputNode, InputVariable, OutputNode, OutputVariable, TimeStep*)

Compute Transfer Functions from MBDyn simulation data

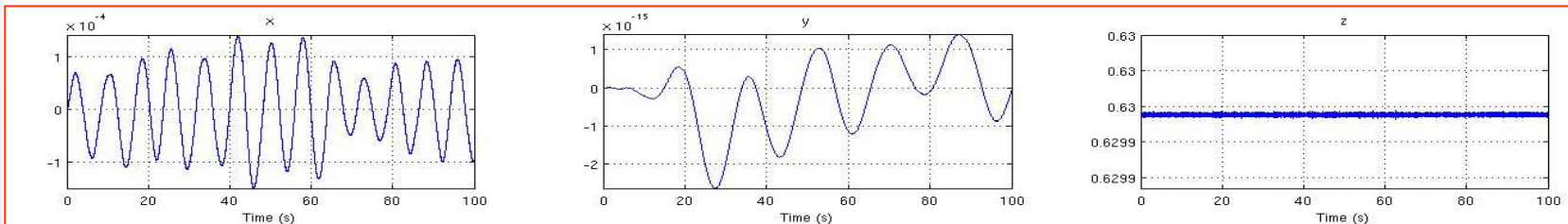


# MBDyn

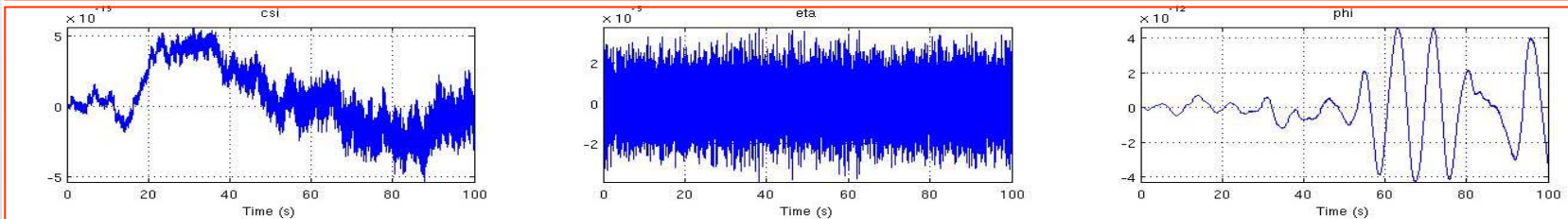
## Matlab visualization tools

In this way the data MBDyn data can be viewed and processed using Virginio's e2e Matlab tools

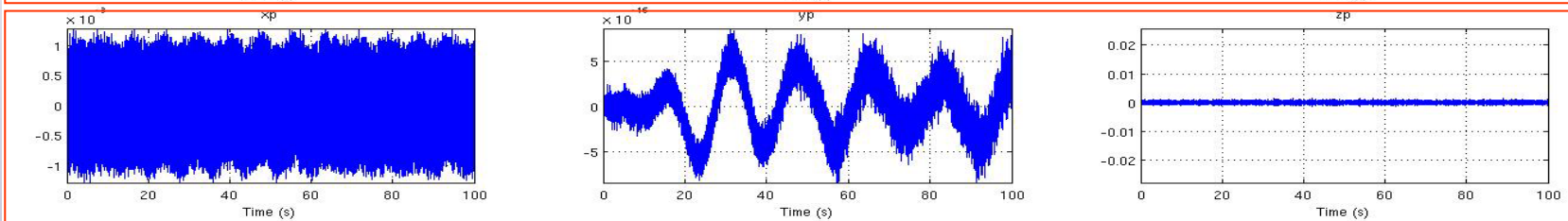
Position



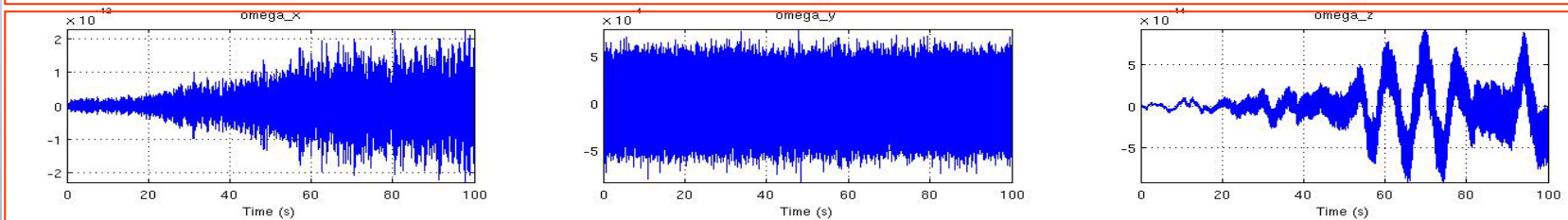
Angles



Velocities

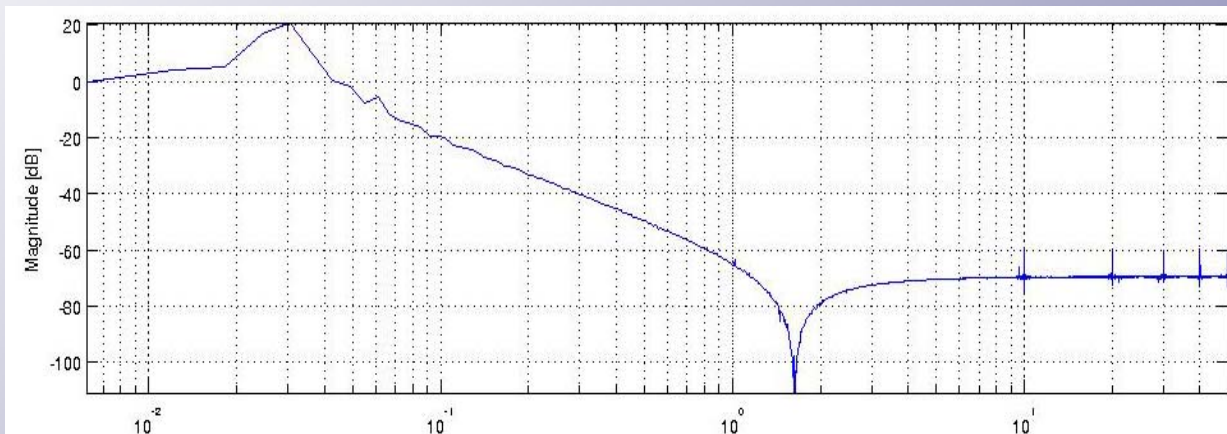
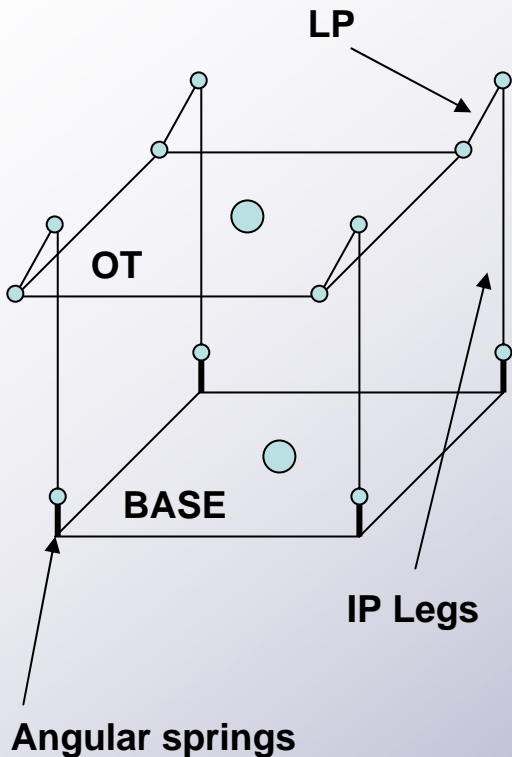


Angular Velocities

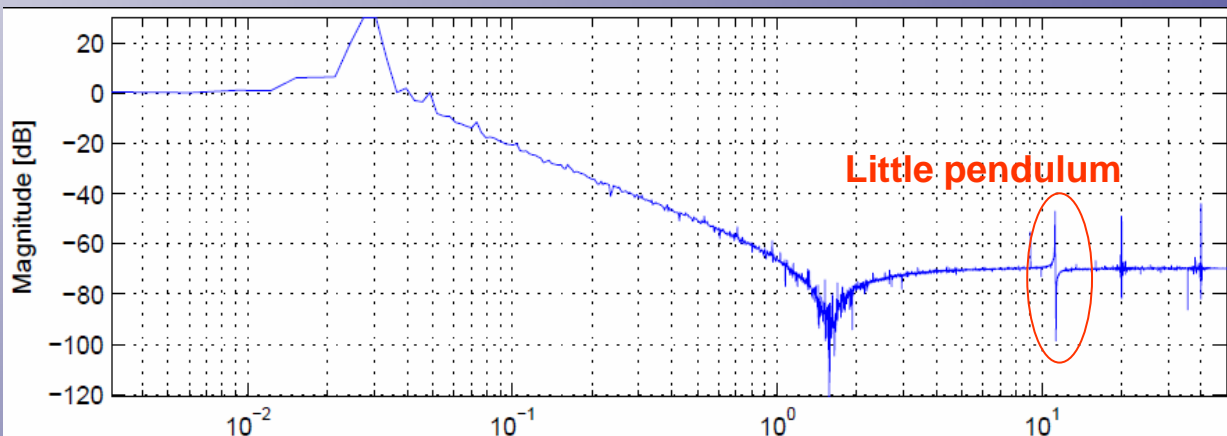


# MBDyn

## Results: IPTable



Horizontal Transmissibility without LP

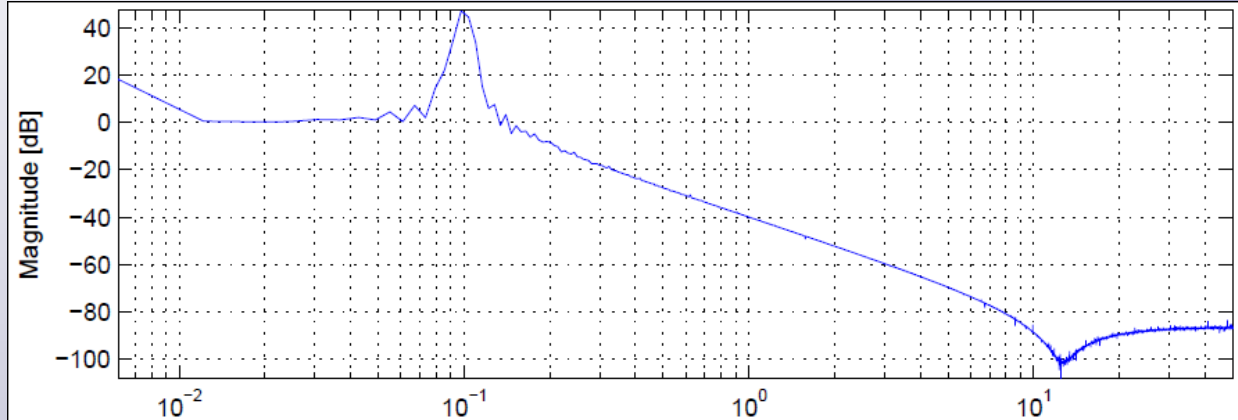
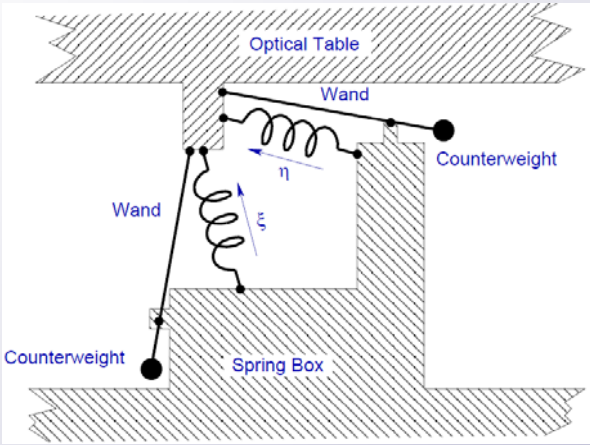


Horizontal Transmissibility with LP

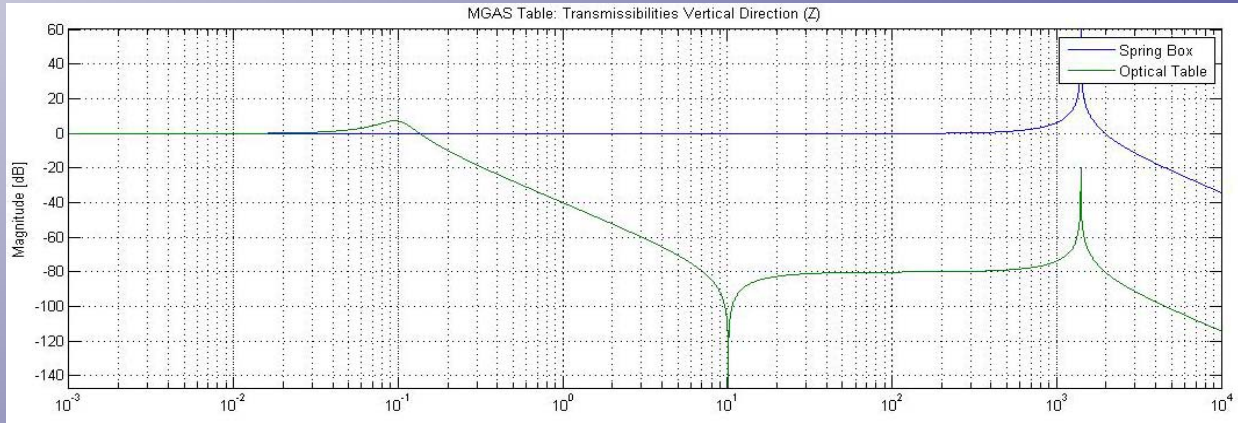
- Completely non-linear and 3D model
- Tuning matches perfectly with theory
- Very good agreement with Maple simulations

# MBDyn

## Results: MGAS Table



Vertical Transmissibility (MBDyn)



Vertical Transmissibility (Maple)

- Completely non-linear and 3D model
- Very good agreement with Maple simulations

# MBDyn

## Linearization

MBDyn works in time-domain and is intrinsically non-linear.

⇒ Extracting a state-space representation is non trivial task:

Three possibilities:

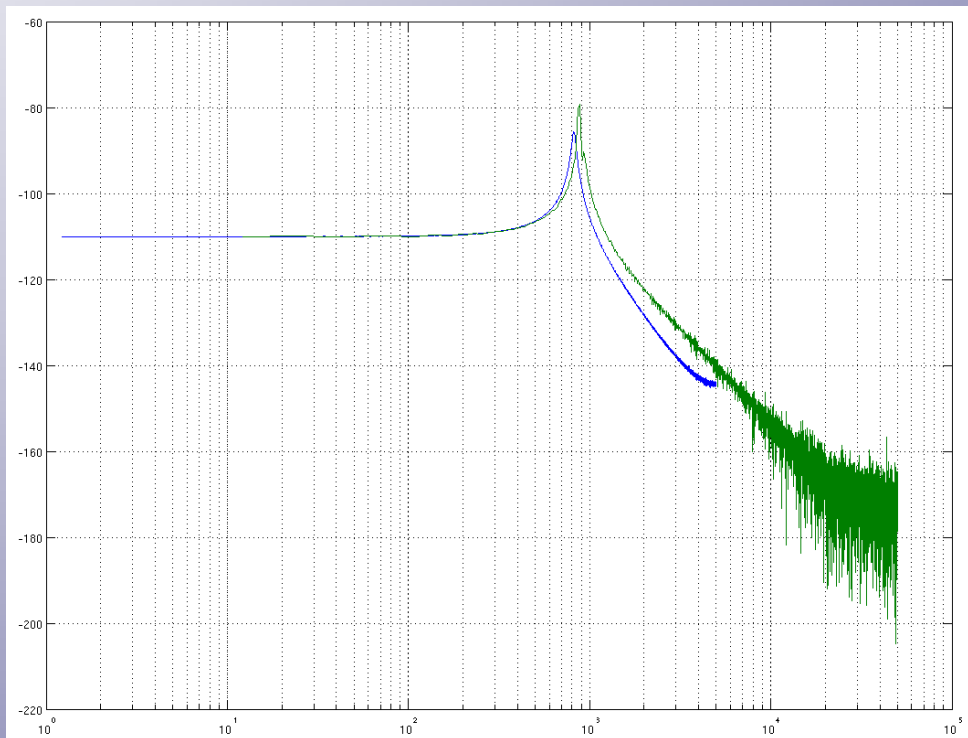
- Study the system using Simulink and Control Design Toolbox integrated linearization tools
- Use Matlab System Identification toolbox set of commands (n4sid, pem, ...) treating the simulation output as experimental data.
- Use an analytical method that has been developed by one of the authors but has not been included yet in the distribution and that we are going to test soon.



# MBDyn

## Problems

- Text only. No GUI yet.
- Time domain simulation only
- Solver parameters must be chosen appropriately
- Frequency domain behavior of some elements (e.g. Beams) is dependent from solver parameters.



# MBDyn

## Future developments

- IPTable Modeling using 2 or 3 node beams
- Linearization study of the system in collaboration with P.Masarati.  
Work for Multibody Dynamics 2007 conference.  
Abstract already submitted : *Seismic Attenuation System Synthesis by Reduced Order Models from Multibody Analysis*
- BSC-SAS modeling