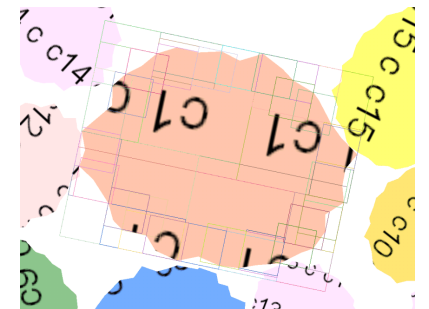
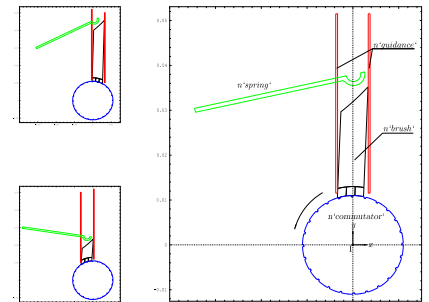
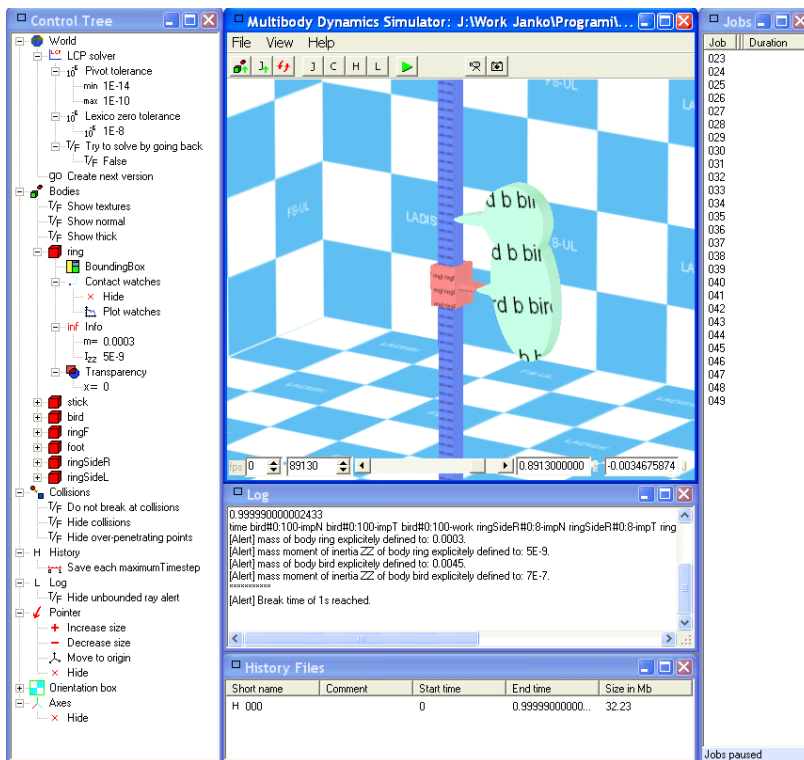




Multibody Dynamics Simulator

dr. Janko SLAVIČ



<http://www.fs.uni-lj.si/ladisk/~slavic/mbd/>

August 28, 2009

Contents

1	Introduction	4
2	Running the program	5
2.1	Front panel	5
2.1.1	Control window	5
2.1.2	Jobs window	6
2.1.3	History window	6
2.1.4	Plot window	6
2.2	Defining a Model - 2 Mass Example	9
2.2.1	The environment	9
2.2.2	Bodies	9
2.2.3	Generalized coordinates	10
2.2.4	Dependent coordinates	11
2.2.5	Mass matrix	11
2.2.6	H Vector	11
2.2.7	Constants	11
2.2.8	Mechanical energy	11
2.2.9	Contact properties	12
2.2.10	Watches	12
2.2.11	Running the model	12
2.2.12	Results	13
2.3	Defining a model - Woodpecker 4 DoF Example	13
2.3.1	Defining the model with Mathematica packages	13
2.3.2	The environment	16
2.3.3	Defining body shape, contact properties, etc	17
2.4	Defining a model - WearBalls	19

3 Specification	22
3.1 Units	22
3.2 Mathematical expression	22
3.3 Shortcut keys	24
3.4 File specifications	25
3.4.1 World	25
3.4.2 Job	27
3.4.3 Body	28
3.4.4 ExtrusionShape	30
3.4.5 GeneralizedCoordinates	30
3.4.6 DependentCoordinates	30
3.4.7 Constants	31
3.4.8 HVector	31
3.4.9 MechanicalEnergy	32
3.4.10 MassMatrix	33
3.4.11 ContactProperties	33
3.4.12 Watches and Reshaping	34
References	37

Chapter 1

Introduction

Multibody Dynamics Simulator resulted from the PhD work [1] on planar dynamics of rigid bodies with unilateral contacts (as Linear Complementarity Problems-LCP). Most of the theory is covered in the book by Pfeiffer and Glocker [2]. There are many more references, but if you need one book to start, this is it!

Some extensions to the Pfeiffer and Glocker formulations built into the program are:

- arbitrary shape of planar discrete bodies that can have unilateral contacts [3],
- tracking mechanical energy lost at contact surfaces [4],
- geometrical roughness phenomena and run-in wear [4, 5].

The program is written in *Delphi 7* and uses *GLScene* Open GL library for graphics and a modified *ParseExpr* for handling symbolic mathematical expressions at run-time.

To run the program you need *MultiBodyDynamicsSimulator.exe* and the *qtintf70.dll*; it is recommended that both files are in path.

Chapter 2

Running the program

In this section a quick overview on running the program will be given. First a the program in the section 2.1 the Front panel is presented and than in the Section 2.2 the program in shown by a simple example model.

You can skip the Front panel section if you are eager to start using the program ASAP.

2.1 Front panel

Figure 2.1 shows the *Multibody Dynamics Simulator* windows: *Main window*, *Control window*, *Log window*, *History window*, *Jobs window* and the *Plot window*.

2.1.1 Control window

Expanded *Control window* is shown in Figure 2.2 and has this sections:

- *World*
 - *LCP Solver*. Used to change parameters of the Linear Complementarity Problem Solver (section 3.4.1)
 - *Create next version*. Creates next version of the current model. If the current model is in directory 000, then the new directory will be 001. Usually, this option is used to create the next version of the model if the model is changing due to wear (section 3.4.2).
- *Bodies*. Used to change the way the bodies are displayed
 - *Body*. Used to show each body' Bounding Box, Contact Watches (section 3.4.12), get body' mass and mass moment of inertia and also to set transparency (section 3.4.3).
- *Collisions*. Used to control what happens at collision (section 3.4.1).
- *History*. Used to control frequency of saved data (section 3.4.1).
- *Log*. Used to control what goes int the log file/window (section 3.4.1).

- *Pointer*. Used to set the focus.
- *Orientation box*. Used to control the orientation box (the room).
- *Axes*. Used to show the inertial coordinate system.

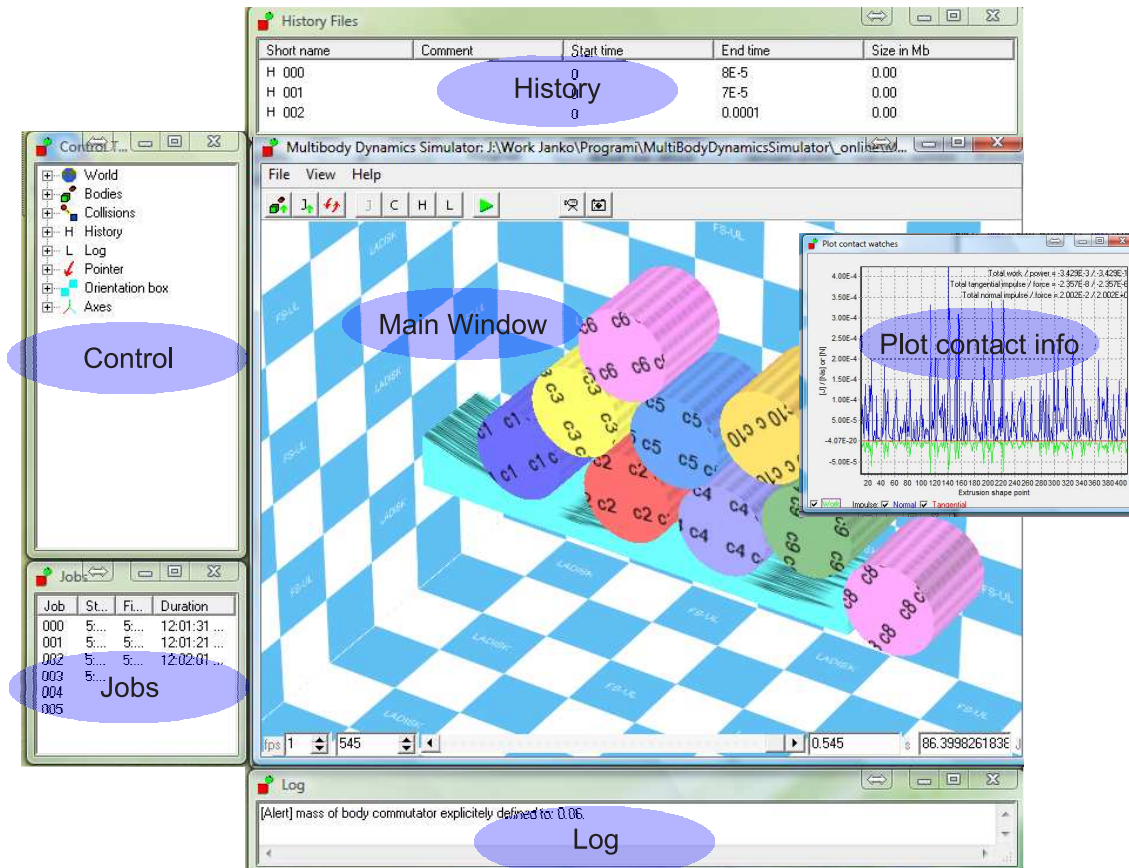


Figure 2.1: MultiBody Dynamics Simulator.

2.1.2 Jobs window

Jobs window is active only if a *Job* is running (section 3.4.2). Via this window the execution of jobs can be controlled.

2.1.3 History window

Each model has its own folder and each simulation run has a 3digit numerated subfolder (e.g. 000, 001, ...). In the history window this subfolders are listed for loading, deleting, resampling. The option *Start with this* is used to start a new simulation where the selected simulation in the history window ended.

2.1.4 Plot window

The *Plot window* is used to show *watches* at selected contact points (section 3.4.12).

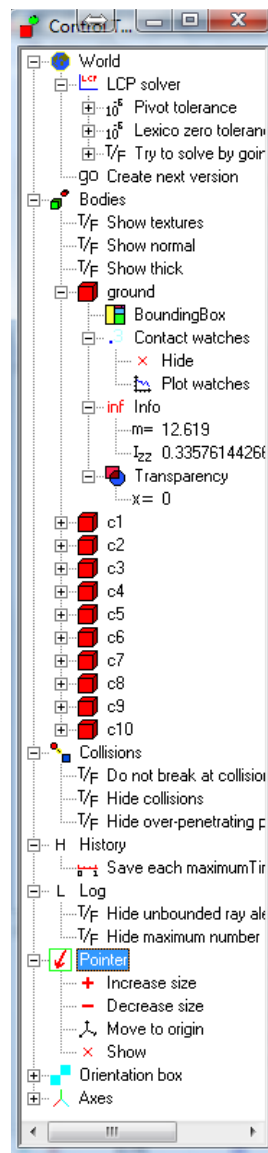


Figure 2.2: Control window.

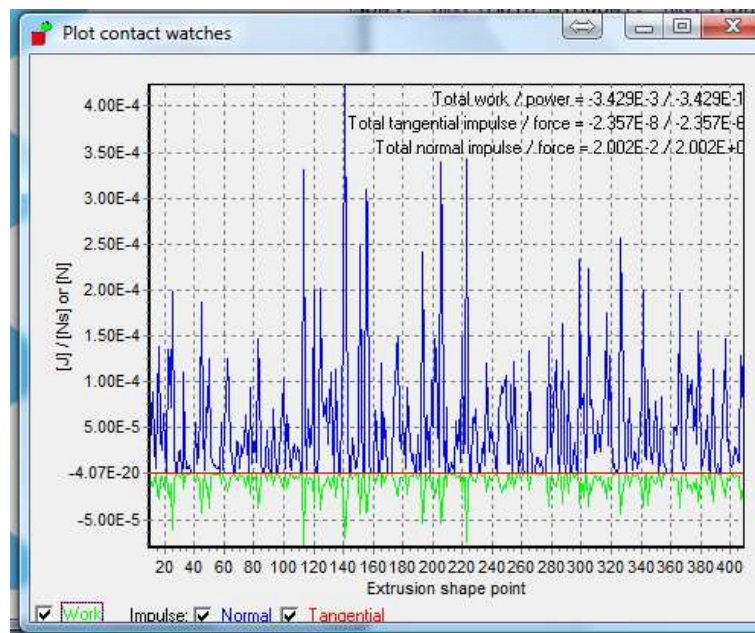


Figure 2.3: Plot results window.

2.2 Defining a Model - 2 Mass Example

The dynamical model is defined by several files (mass matrix, shapes, contact parameters,...). It is advised to create a separate folder for each model.

Each run of a model will create a three digital sub-folder starting with 000 (following with 001,002,...). In this way each simulation can have different parameters or shapes; this is especially useful in wear-in simulations where the body-shapes change with each simulation.

The creation of a simple *2Mass model* will shown here.

2.2.1 The environment

The main file of a model is the *World file*.

A simple world file is:

```
[main]
numberOfGeneralizedCoordinates = 3
minimumTimestep = 1e-10
maximumTimestep = 5e-4
maximumPenetration = 1e-5
breakTime = 0.5

[bodies]
body1 = mass1.bdy
body2 = mass2.bdy
```

Detailed specifications for the *World file* are given in section 3.4.1.

2.2.2 Bodies

Each body defined in the *World file* needs to have a *bdy* file. A typical body file named *mass1.bdy* is (section 3.4.3):

```

[main]
name = mass1
RGBcolor = 0 1 0
shapeType = extrusionSolid
shapeFile = mass1.exs
density = 2500
thickness = 0.02
correctOrigin = 0

[startPosition]
x = -0.005
y = 0.15
z = 0
Rx = 0
Ry = 0
Rz = 3.1459

[startVelocity]
x = 0
y = -1
z = 0
Rx = 0
Ry = 0
Rz = 0

```

As defined in the body file (*shapeFile = mass1.exs*), the shape is defined in *mass1.exs shapeFile* (section 3.4.4). A simple shape is shown below with listing the x and y points that define the edge of the body.

```

points = 6
%%data
0.07 0.00 -0.07 -0.07 0.07 0.07
0.05 0.06 0.05 -0.05 -0.05 0.05
%%enddata

```

2.2.3 Generalized coordinates

Generalized coordinates are defined in the *gc* file (section 3.4.5).

In this simple example we would like the body named *mass1* to have three degrees of freedom (x, y and Rz)¹.

The *2Mass.gc* file is:

¹The program uses *contexts* to organize names of symbols. All bodies are in the context *b* and each body has its own context that is the same as the name defined in the body file (*bdy*).

```
b'mass1'x[t]
b'mass1'y[t]
b'mass1'Rz[t]
```

2.2.4 Dependent coordinates

Generalized coordinates are defined in the *dc* file (section 3.4.6).

For this simple example the *2Mass.dc* file is empty as there are no dependent coordinates. For a non-empty *dc* file, see section 2.3.

2.2.5 Mass matrix

The mass matrix is defined in the *mmx* file (section 3.4.10).

The *2Mass.mmx* file is:

```
b'mass1'm 0 0
0 b'mass1'm 0
0 0 b'mass1'Jzz
```

2.2.6 H Vector

The equations of motion are written in the form $\mathbf{M}\ddot{\mathbf{q}} = \mathbf{H}$ and the \mathbf{H} vector is used to describe the dynamics that is not covered by the mass matrix \mathbf{M} multiplied by the generalized acceleration vector $\ddot{\mathbf{q}}$. The \mathbf{H} is defined in the *hv* file (section 3.4.8).

The *2Mass.hv* file is:

```
0
-(c'gravity*b'mass1'm)
0
```

2.2.7 Constants

A dynamic model can have several constants that are covered in the *cns* file (section 3.4.7).

The *2Mass.cns* file is:

```
c'gravity = 9.81
```

2.2.8 Mechanical energy

Definition of the mechanical energy is not necessary for running a simulation. However, mechanical energy defined in the file with the *me* extension (section 3.4.9) is useful for keeping track of the lost energy in the system.

The *2Mass.me* file is:

```
c'gravity*b'mass1'm*b'mass1'y[t] + (b'mass1'Jzz*Derivative[1][b'mass1'Rz][t]^2)/2 +
(b'mass1'm*(Derivative[1][b'mass1'x][t]^2 + Derivative[1][b'mass1'y][t]^2))/2
```

2.2.9 Contact properties

The contact properties are defined in the *cop* file. To define contact properties between the bodies the names defined in the body file are used and for each body pair (section 3.4.11)

The *2Mass.cop* file is:

```
[mass1-mass2]
friction=0.1
restitutionNormal=0.7
restitutionTangent=0.2
parameterNi=0
initialTemperature=0
```

2.2.10 Watches

What contact properties should the program keep track off is defined in the Watches file, extension *wtc* (section 3.4.12).

To keep track of points 0,1 and 2 of the body mass1 in the file looks like this (*2Mass.wtc*):

```
[mass1]
contactWatch = 0-2
```

The watches are written to the file *watches.txt*, where the header of the watches file is written in the *log.txt* (section 2.2.12)

2.2.11 Running the model

Once the model is defined run the *Multibody Dynamics Simulator*, go to *File/Open* and find the folder with the model.wld file. The simulations starts with a click on the green play button. Reduce the Frames Per Second (FPS) in the lower left edge of the *Main window*, if you want more computer power for calculation and less for visualization. You can rotate the model with the mouse and use *short-keys* (section 3.3) to zoom in or out. Use the *Pointer* in the *Control window* to set focus to an arbitrary point (by default focus is set to the first body of the system).

2.2.12 Results

Log file For this example the header looks like this:

```
time mechanicalEnergy b'mass1'x b'mass1'x' b'mass1'x" b'mass1'y b'mass1'y'...
-
0
0.49995
time mass1#0-impN mass1#0-impT mass1#0-work mass1#1-impN mass1#1-impT...
*****
[Alert] Break time of 0.50005s reached.
```

The first line is the header of the `data.txt` file.

The 3rd and 4th lines are start and end time, respectively.

The 5th line is the header of the `watches.txt` file. Lines from 7th on are notes about the simulation.

Data file `data.txt` stores the time series of the simulation.

Watches file `watches.txt` stores the watches data of the simulation. For each body the accumulated watches data can be shown in the *Plot window*, Figure 2.3.

2.3 Defining a model - Woodpecker 4 DoF Example

The Woodpecker 3DoF and 4DoF models were presented and analyzed in detail in the research [3], see Figure 2.4.

2.3.1 Defining the model with Mathematica packages

For the *Multibody Dynamics Simulator* a set of *Mathematica* packages was developed to help with defining the model. The *Mathematica* notebook for the *2 Mass* model is shown in Figure 2.5. And for the *Woodpecker 4 DoF* model in Figure 2.6. While the Woodpecker model is more complicated than the *2 Mass* model, the *Mathematica* notebook is still simple.

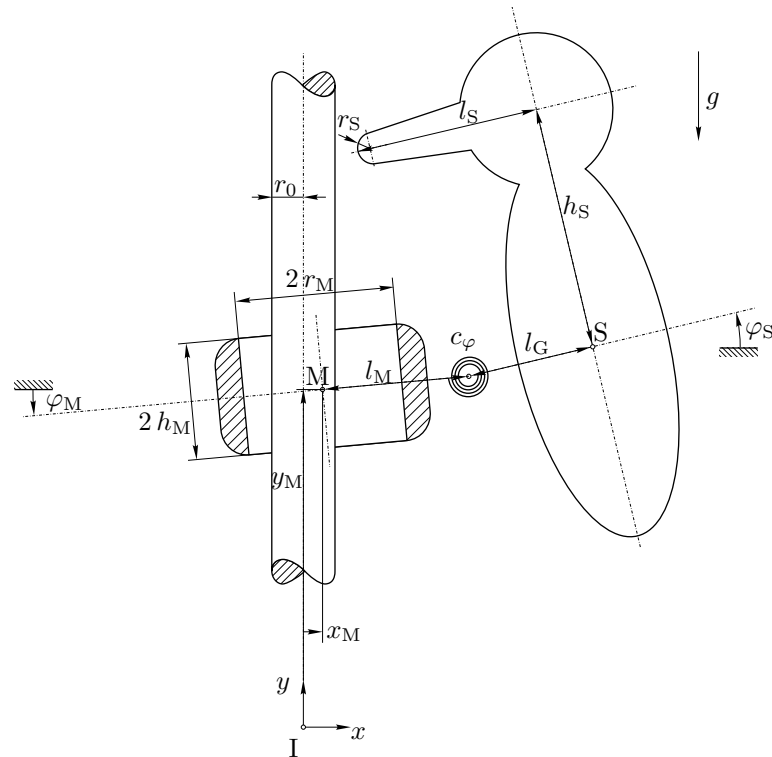


Figure 2.4: Woodpecker 4 DoF model.

Libraries

Path to custom libraries

```
AppendTo[$Path, "j:/Work Janko/CommonFiles/Math"];
Needs["MultiBodyDynamics`MultiBodyDynamics`"];
```

2 Mass System

dr. Janko Stavic, 2009
www.fs.uni-lj.si/ladisk/-stavic/

Coordinates

Generalized coordinates

```
GC := {
  b`mass1`x[t], b`mass1`y[t], b`mass1`Rz[t];
```

Dependent coordinates

```
DC := {};
```

Kinetic, potential energy and virtual work

Kinetic energy

```
Ek[t_] :=
(*z1*)
1/2 b`mass1`m + vr1[t]^2 + 1/2 b`mass1`Jzz + b`mass1`Rz'[t]^2
```

```
vr1[t_] := sqrt(b`mass1`x'[t]^2 + b`mass1`y'[t]^2)
```

Potential energy

```
Ep[t_] := b`mass1`m c`gravity b`mass1`y[t]
```

Virtual work

```
dW[t_] := 0
```

Equations of motion

```
EM = EquationsOfMotion[Ek[t], Ep[t], GeneralizedForces[dW[t], GC], GC
```

```
{m, h} = MassMatrixAndHVector[Ek[t], Ep[t], GeneralizedForces[dW[t],
```

MatrixForm[m]

```
{
  b`mass1`m    0    0
  0    b`mass1`m    0
  0    0    b`mass1`Jzz
```

MatrixForm[h]

```
{
  0
  -c`gravity b`mass1`m
  0
```

Name of model, folder name

```
filename = "2Mass";
folder = "j:/";
```

Write mass matrix, h vector and mechanical energy

```
MXMWrite[folder <> filename <> ".mxm", m]
HVWrite[folder <> filename <> ".hv", h]
HWWrite[folder <> filename <> ".me", Ek[t] + Ep[t]]
```

Write generalized and dependent coordinates

```
GCWrite[folder <> filename <> ".gc", GC]
DCWrite[folder <> filename <> ".dc", PrepareDependentCoordinates[Defi
```

Define and write constants

```
c`gravity = 9.81;
```

```
CNSWrite[folder <> filename <> ".cns", Names["c`*"], ToExpression[Name
```

Figure 2.5: Mathematica notebook for the 2 Mass system.

Libraries

Path to custom libraries

```
AppendTo[$Path, "j:/Work Janko/CommonFiles/Math"];
Needs["MultiBodyDynamics`MultiBodyDynamics`*"];
```

WoodpeckerToy - 4 DoF

dr. Janko Slavic, 2009
www.fs.uni-lj.si/ladisk/-slavic/

Coordinates

Generalized

```
GC := {
  b`ring`x[t], b`ring`y[t], b`ring`Rz[t], b`bird`Rz[t];
```

Dependent

```
DC := {b`bird`x[t], b`bird`y[t]};
```

```
b`bird`x[t] := c`lM * Cos[b`ring`Rz[t]] + c`lG * Cos[b`bird`Rz[t]] + b`
b`bird`y[t] := b`ring`y[t] + c`lM * Sin[b`ring`Rz[t]] + c`lG * Sin[b`bird`Rz[t]]
```

Kinetic, potential energy and virtual work

Kinetic energy

```
Ek[t_] :=
(*ring*)
1/2 b`ring`m * vRing[t]^2 + 1/2 b`ring`Jzz * b`ring`Rz'[t]^2 +
(*bird*)
1/2 b`bird`m * vBird[t]^2 + 1/2 b`bird`Jzz * b`bird`Rz'[t]^2
```

```
vRing[t_] := Sqrt[b`ring`x'[t]^2 + b`ring`y'[t]^2]
vBird[t_] := Sqrt[b`bird`x'[t]^2 + b`bird`y'[t]^2]
```

Potential energy

WoodpeckerToy 4DoF.nb

```
Ep[t_] :=
c`gravity * b`ring`m * b`ring`y[t] +
c`gravity * b`bird`m * b`bird`y[t] +
1/2 c`spring * (b`bird`Rz[t] - b`ring`Rz[t])^2
```

Virtual work

```
deltaW[t_] := 0
```

Equations of motion

```
EM = EquationsOfMotion[Ek[t], Ep[t], GeneralizedForces[deltaW[t], GC], GC]
```

Mass matrix and h vector

```
{m, h} = MassMatrixAndHVector[EM[t], Ep[t], GeneralizedForces[deltaW[t], GC], GC]
```

Save model for MultiBody Dynamics Simulator

Name of model

```
filename = "WoodpeckerToy";
folder = "j:/";
```

Write mass matrix, h vector and mechanical energy

```
MXXWrite[folder <> filename <> ".mxx", m]
HVWrite[folder <> filename <> ".hv", h]
HVWrite[folder <> filename <> ".me", EK[t] + Ep[t]]
```

Write generalized and dependent coordinates

```
GCWrite[folder <> filename <> ".gc", GC]
GCWrite[folder <> filename <> ".dc", PrepareDependentCoordinates[Defi
```

Define and write constants

```
c`gravity = 9.81;
c`lM = 0.01;
c`lG = 0.015;
c`spring = 0.0056;
CNSWrite[folder <> filename <> ".cns", Names["c`*"], ToExpression[Name
```

Figure 2.6: Mathematica notebook for the Woodpecker 4 DoF system.

With help of the *Mathematica* packages the Jacobian vector and the Jacobian scalar of the dependent coordinates are automatically generated (section 3.4.6) [2, 3].

The *WoodpeckerToy.dc* file for the Woodpecker 4 DoF model is shown here:

```

%%new%%
b'bird'x[t]
%%definition c'IG*Cos[b'bird'Rz[t]] + c'IM*Cos[b'ring'Rz[t]] + ...
%%J
1
0
-(c'IM*Sin[b'ring'Rz[t]])
-(c'IG*Sin[b'bird'Rz[t]])
%%j
-(c'IG*Cos[b'bird'Rz[t]]*Derivative[1][b'bird'Rz[t]^2])...

%%new%%
b'bird'y[t]
%%definition
c'IG*Sin[b'bird'Rz[t]] + c'IM*Sin[b'ring'Rz[t]]...
...

```

2.3.2 The environment

The *WoodpeckerToy.wld* file looks like this:

```

[main]
decimalSeparator = .
numberOfGeneralizedCoordinates = 4
minimumTimestep = 1e-12
maximumTimestep = 1e-5
maximumPenetration = 1e-6
writeToHistoryMultipleOfMaximumTimestep = 1
breakTime = 1.0

[bodies]
body1 = ring.bdy
body2 = stick.bdy
body3 = bird.bdy
body4 = ringF.bdy
body5 = foot.bdy
body6 = ringSideR.bdy
body7 = ringSideL.bdy

```

From the World file we can see that there are seven bodies in the system (Figure 2.7).

There is a problem with the ring to stick interaction. The ring is embracing the stick and by using only extrusion bodies this needs to be done by combining four bodies: *ring*, *ringF*, *ringSideR* and *ringSideL*.

However, to speed-up the computation all the mass properties can be added to one body (e.g. *ring*) and other bodies can be defined its virtual bodies (see option *virtualToBody* in section 3.4.3). In this way the *ringSideR* and *ringSideL* can have unilateral contact with the stick, while the contact dynamics is actually acting on the body *ring*.

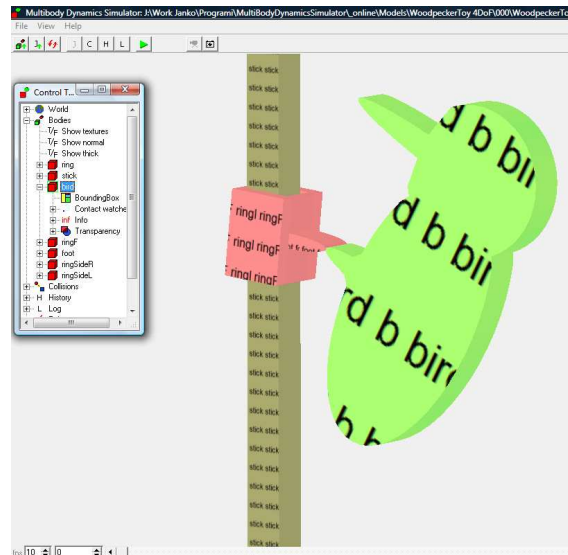


Figure 2.7: Woodpecker in the Multibody Dynamics Simulator.

2.3.3 Defining body shape, contact properties, etc

The *Mathematica* packages can also help you with creating complex body shapes. Figure 2.8 shows *Mathematica* notebook for creating the *bird* shape (*exs* file).

The contact properties for the system are:

2
Bird.nb

Libraries

Path to custom libraries

```
AppendTo[$Path, "j:/Work Janko/CommonFiles/Math*"];
Needs["MultiBodyDynamics`MultiBodyDynamics`*"];
```

WoodpeckerToy - Bird

dr. Janko Slavic, 2009
[www.fs.uni-lj.si/ladisk-slavic/](http://www.fs.uni-lj.si/ladisk/slavic/)

This file is used to create the exact shape of the bird

```
Circle[x_, n_] := Module[{s, i},
  Table[{x Cos[2 π  $\frac{i}{n}$ ], x Sin[2 π  $\frac{i}{n}$ ]}, {i, 0, n}]
]

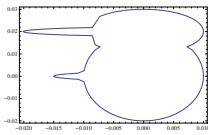
Ellipse[x1_, x2_, n_] := Module[{s, i},
  Table[{x1 Cos[2 π  $\frac{i}{n}$ ], x2 Sin[2 π  $\frac{i}{n}$ ]}, {i, 0, n}]
]

body = Ellipse[0.01, 0.02, 50];
beak = Ellipse[0.0201, 0.002, 50];
head = Ellipse[0.01, 0.01, 50];
feet = Ellipse[0.015, 0.002, 50];

TranslateData[data_, x_, y_] := Module[{}],
  Transpose[{x, y} + Transpose[data]]]
```

```
data = Join[
  Take[body, {1, 6}],
  TranslateData[Take[head, {45, 50}], 0, 0.02],
  TranslateData[Take[head, {1, 20}], 0, 0.02],
  TranslateData[Take[beak, {17, 35}], 0, 0.02],
  TranslateData[Take[head, {31, 32}], 0, 0.02],
  TranslateData[Take[body, {21, 25}], 0, 0],
  TranslateData[Take[feet, {20, 32}], 0, 0],
  TranslateData[Take[body, {27, 51}], 0, 0]
];

ListPlot[data, Frame -> True, Axes -> False, PlotJoined -> True]
```



```
a = Min[Transpose[data][[1]]]
-0.0201

Position[data, a]
{{42, 1}}

data[[42, 2]]
0.02
```

Write/ReadFile

```
filename = "j:/bird.exs";
EXSWrite[filename, data]

dateread = EXSRead[filename];
```

Figure 2.8: Mathematica notebook for creation of the Bird shape.

```
[stick-ringSideR]
friction=0.3
restitutionNormal=0
restitutionTangent=0
parameterNi=0
initialTemperature=0
```

```
[stick-ringSideL]
friction=0.3
restitutionNormal=0
restitutionTangent=0
parameterNi=0
initialTemperature=0
```

```
[stick-bird]
friction=0.3
restitutionNormal=0.5
restitutionTangent=0
parameterNi=0
initialTemperature=0
```

2.4 Defining a model - WearBalls

WearBalls is a simple model of five balls bouncing and colliding with side-wall and sinusoidally moving ground, Figure 2.9. Here we will discuss how to deal with wear which is defined in the *watches* file.

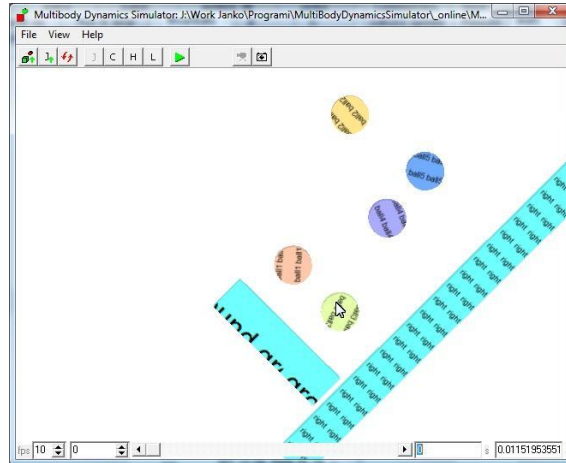


Figure 2.9: WearBalls model.

The *watches* file looks like this:

```
[ball1]
contactWatch = all
directionInside=1
maxChange=0.002

[ball2]
contactWatch = all
directionInside=1
maxChange=0.002

[ball3]
contactWatch = all
directionInside=1
maxChange=0.002

[ball4]
contactWatch = all
directionInside=1
maxChange=0.002

[ball5]
contactWatch = all
directionInside=1
maxChange=0.002
```

The *watches* file will cause the shape of each ball to change (wear) regarding to the loss of mechanical energy at the surface of the body; where the point with maximum loss of mechanical energy will reshape to the inside for 0.002 m. Running the first version for a few moments create the 000 folder that can be used as a starting model version for job run.

The *job* file:

```
[main]
jobs=000-013
maxMinutesForJob=10
maxMinutesForJobs=300
createNextVersionAtEnd=1
```

will start running the (already created) version 000 and at the end it will create the version 001 and start running it. This will continue until the version 013. The shapes of the version 013 are shown in the Figure 2.10.

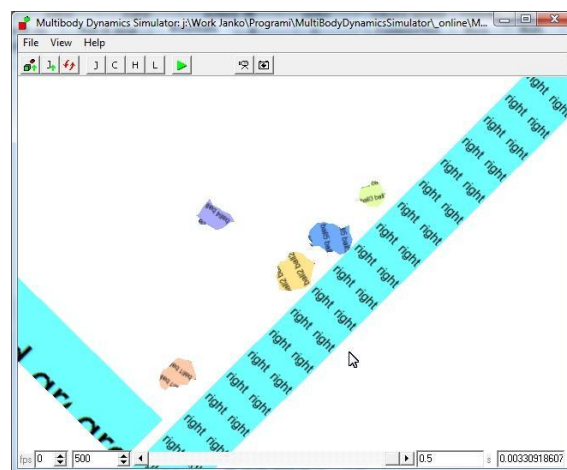


Figure 2.10: WearBalls model after 13 wear cycles.

If the *watches* file defines the *bodyBalancedUpdate* wear:

```
[bodyBalancedUpdate]
body = ball1,ball2,ball3,ball4,ball5
maxChange=0.002
```

then the wear all bodies is balanced: with maximum wear at the point of maximum loss of mechanical energy of all bodies. The result of version 013 with *bodyBalancedUpdate* is shown in Figure 2.11.

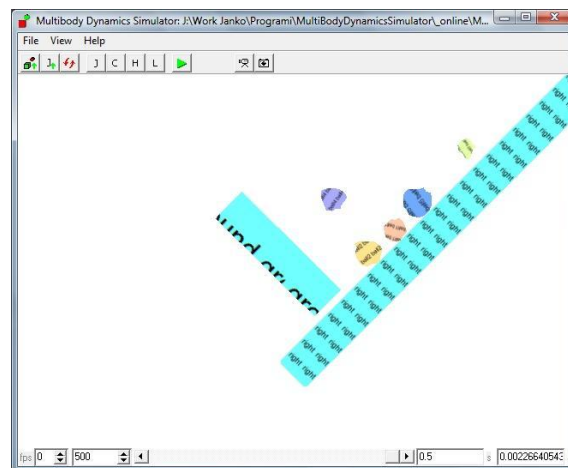


Figure 2.11: WearBalls model after 13 body-balanced wear cycles.

Chapter 3

Specification

3.1 Units

All units are kg, m, s, rad, °C.

3.2 Mathematical expression

The program uses *contexts* to organize names of symbols.

For example all bodies are in the context b and each body has its own context that is the same as the name defined in the body file. To access to the coordinate x of the body named *mass1* the mathematical expression is: $b'mass1'x[t]$.

List of contexts:

b bodies (use `bodyName`, see also section 3.4.3)

c constants

coordinate possible values: x, y, z, Rx, Ry, Rz

g denotes generalized coordinate which is not directly bounded to a body

© whenever mathematical expression can be used this sign will denote this.

Mathematical expressions are

Mathematical expression can contain:

- *Generalized coordinates* (see section 3.4.5) and its derivatives (e.g. $b'rect'x[t]$).
- *Mass properties* (e.g. $b'rect'm$, $b'rect'Jzz, \dots$).
- *Constants* (see section 3.4.7, e.g. $c'gravity$).

- *Mathematical functions* like $\sin(x)$, $\cos(x)$, $\operatorname{arcsinh}(x)$, $\log_{10}(x)$, $\ln(x)$, $\log_N(\text{base}, x)$, $\operatorname{sqr}(x)$,...
- *Operands*: $x!$, x^y , $*$, $+$, $-$,...

Note: First derivative (velocity) is denoted by `Derivative[1][b'rod'x][t]` or by `b'rod'x'[t]`.

Note: `[t]` is removed, i.e.: `b'rect'x[t]` is replaced by `b'rect'x`. You **cannot** use functions `Sin[t]`, but you can `Sin[1*t]`!

3.3 Shortcut keys

A zoom in.

shift+A zoom-in with adjusting the focus.

Y zoom out.

shift+Y zoom out with adjusting the focus.

S change font scale (used for contact watches).

B changes background color to white and back.

3.4 File specifications

3.4.1 World

File extension: wld

Example of a file with default values (used if key is not given):

```
[main]
decimalSeparator = .
workingDirectory = current
numberOfGeneralizedCoordinates = 1
LCPPivotToleranceMin = 1e-14
LCPPivotToleranceMax = 1e-10
LCPLexicoZeroTolerance = 1e-8
logLCPUnboundedRay = 0
logLCPMaxNumberOfIterations = 0
ZeroVelocityTolerance = 1e-18
TryToSolveByGoingBack = 0    ← 1=True, 0=False
minimumTimestep = 1e-12
maximumTimestep = 1e-5
maximumPenetration = 1e-6
maximumDisplacementFactor = 0.5
writeToHistoryMultipleOfMaximumTimestep = 1
writeToHistoryAtCollision = 0    ← 1=True, 0=False
writeToHistoryEachTimeStep = 0    ← 1=True, 0=False
constantListFile = current.cns
generalizedCoordinatesFile = current.gc
dependentCoordinatesFile = current.gc
massMatrixFile = current.mmx
hVectorFile = current.hv
mechanicalEnergyFile = current.me
contactPropertiesFile = current.cop
watchesFile = current.wtc
breakTime = 1    ← Ⓣ

[bodies]
body1 = rect.bdy
body2 = circle.bdy
```

LCPPivotToleranceMin, *LCPPivotToleranceMax*, *LCPLexicoZeroTolerance* define the parameters for the LCP Solver. In case of problems with obtaining the solution try to change this settings. For more accurate results and for geometrically smaller bodies smaller tolerances should be used.

minimumTimestep and *maximumTimestep* are the minimum and the maximum allowed time step of the numerical integration, respectively. The maximum time step is in general defined by the numerical integration of the differential equations, while the minimum time step is defined by collisions. The

program automatically adjusts the time step between min and max to achieve the penetration limit *maximumPenetration*.

TryToSolveByGoingBack if true at unsolvable time-step goes two time-steps back to the history and tries again.

maximumDisplacementFactor maximal displacement is estimated according to the geometry and current velocity and then multiplied by the *maximumDisplacementFactor*. *maximumDisplacementFactor* should be between 0.1 (quick) and 10 (slow); smaller values can lead to over-penetrations. The body-defined *maximumDisplacementFactor* overrides the world defined (see 3.4.3).

breakTime can be a mathematical expression; however, it is calculated at start only.

logLCPUnboundedRay if True, LCP unbounded ray alert is written to log.

logLCPMaxNumberOfIterations if True, LCP maximum number of iterations alert is written to log.

ZeroVelocityTolerance Is used to create the set of non-impacting contacts.

Note: *current* is a special token that can be used to replace for example working directory by the directory of world file.

Token can also be used in the filenames, i.e.: “current.cns” is replaced with “worldFileName.cns”.

3.4.2 Job

File extension: job

Example of a job file with default values (used if key is not given):

```
[main]
worldFile=default.wld
jobs=      ← example: 001,005,009
lexicoRandomizationRange=0.0,0.0
createNextVersionAtEnd=0    ← 1=True, 0=False
maxLinesInLog=0
maxMinutesForJob=MaxInt
maxMinutesForJobs=MaxInt
framesPerSecond=0
delayStartMinutes=0
exitAtFinish=0    ← 1=True, 0=False
```

Note: the model versions (saved in folders) that are about to be ran need to exist. If there are iterative models that create a new version at the end at least the first version needs to exist.

worldfile if not present then the name of the *job* file is used.

jobs lists (000,001,005) and ranges can be used (005-015).

lexicoRandomizationRange if set then the *LCPLexicoZeroTolerance* (see .wld file, section 3.4.1) is randomized at each time-step.

createNextVersionAtEnd if enabled then at the end of job next version is created with blank history files.

command-prompt jobs can be started as parameters in command-prompt.

Example: `start \low \hold MultiBodyDynamicsSimulator subdir\BrushSystem.job`

maxLinesInLog can be used to break simulation if the maximum number of allowed log lines is exceeded. If `maxLinesInLog=0` the log checking is off.

3.4.3 Body

File extension: **bdy**

Example of a file:

```
[main]
name = rect
virtualToBody =
RGBColor = 0 1 0
shapeType = extrusionSolid
shapeFile = rect.exs
density = 7900
thickness = 0.1
maximumDisplacementFactor = 0.1
skinThickness = 0.001    ← not needed by default
correctOrigin = 1      ← 1=True, 0=False

[Mass properties]
mass = 1
InertiaXX = 0.004
InertiaXY = 0.004
InertiaXZ = 0.004
InertiaYY = 0.004
InertiaYZ = 0.004
InertiaZZ = 0.004

[startPosition]
x = 0
y = 1.1
z = 0
Rx = 0
Ry = 0
Rz = 0

[startVelocity]
x = 0
y = 1.1
z = 0
Rx = 0
Ry = 0
Rz = 0
```

Note: *skinThickness* defines the maximum depth of body which is checked for penetration (if not set then the default value is: $100 \times \text{maximumPenetration}$, see 3.4.1).

Note: section *Mass properties* is optional. If mass and mass moment of inertia *ZZ* are not defined, then they are calculated by using shape data and density.

Note: if body is virtual to another body use the key *virtualToBody* to define to which body is virtual to.

Virtual body means that the referenced body uses only the shape of the virtual body to detect collisions.

Note: If the extrusion shape file (see 3.4.4) origin is not at the center of mass and the *correctOrigin* is set to True, then the origin is corrected to start at the center of mass.

maximumDisplacementFactor: the body-defined *maximumDisplacementFactor* overrides the world defined (see 3.4.1).

3.4.4 ExtrusionShape

File extension: **exs**

Example of a file:

```
points = 4
%%data
1.0  0.125  -0.52  1.0  %← x
1.0  0.50   -1.31  1.0  %← y
%%enddata
```

Note: extrusion shape needs to be closed (first and last points are the same). It is also important that the inside of the body is on the left hand side in one moves in the direction from first to the last point.

3.4.5 GeneralizedCoordinates

File extension: **gc**

Example of a file:

```
b'rect'x[t]
b'rect'y[t]
b'rect'Rz[t]
g'relative1[t]
```

See also section 3.2.

3.4.6 DependentCoordinates

File extension: **dc**

Example of a file:

```

%%new%%
b'bird'x[t] ← coordinate name, see 3.4.5.
%%definition
c'lG + c'lM ← position ©
0 ← velocity ©
0 ← acceleration ©
%%J ← Jacobian vector
0 ← according to 1st generalized coordinate ©
0 ← according to 2nd generalized coordinate ©
0 ← according to 3rd generalized coordinate ©
%%j ← Jacobian scalar
c'lG + c'lM ← ©

%%new%%
b'bird'y[t]
%%definition
:

```

3.4.7 Constants

File extension: **cns**

Example of a file:

```

c'forcex = 1
c'forcey = 2
c'gravityz = 9.81

```

Notation: c'constantName

c denotes constant object
constantName can be whatever

3.4.8 HVector

File extension: **hv**

Example of a file:

```

c'forcex ← ©
c'forcey + c'gravityz b'rect'm ← ©
c'forcex (c'jly Cos[b'rect'Rz[t]] - c'jlx Sin[Derivative[1][b'rod'x][t]]) ← ©
+...

```

Note: vector length equals the number of generalized coordinates. Mathematical expressions are used ©

3.4.9 MechanicalEnergy

File extension: **me**

Example of a file:

```
c'gravity*b'rod'm*b'rod'y[t] + (b'rod'Jzz*Derivative[1][b'rod'Rz][t]^2)/2 + ... ← ©
```

Note: mechanical energy should be written in first line.

3.4.10 MassMatrix

File extension: **mmx**

Example of a file:

```
b'rect'm 0 0 ← ⊕
0 b'rect'm 0 ← ⊕
0 0 b'rect'Jzz ← ⊕
```

See section 3.2.

3.4.11 ContactProperties

File extension: **cop**

Example of a file:

```
[rect1-rect2]      % there should be names of bodies, see also section 3.4.3
  resistingPenetration=T:T
  friction=0.5-1/vRel
  restitutionNormal=0.5-1/temp
  restitutionTangent=0.5
  parameterNi=0
  initialTemperature=0

[rect1:all]
  resistingPenetration=T:T
  friction=0.5
  restitutionNormal=0.5
  restitutionTangent=0.5
  parameterNi=0
  initialTemperature=0
```

Note: contact can be defined between two bodies (e.g. [rect-ball]) or between a body and all other (e.g. [rect:all]). First body:all is read and then overwritten with body-body properties.

Note: contact between two bodies is defined by:

<i>resistingPenetration</i>	defines if bodies resist penetration. Possible cases: T:T, T:F, F:T.
<i>friction</i>	coefficient of friction function
<i>restitutionNormal</i>	coefficient of restitution function in normal direction
<i>restitutionTangent</i>	coefficient of restitution function in tangential direction
<i>parameterNi</i>	parameter function of tangential restitution function
<i>initialTemperature</i>	the initial temperature of a contact

Functions of contact can contain any constant values and variables “temp” and “vRel”.

<i>temp</i>	denotes the contact temperature
<i>vRel</i>	denotes the absolute value of the relative tangential velocity
<i>initialTemperature</i>	the initial temperature of a contact

3.4.12 Watches and Reshaping

File extension: **wtc**

Example of a file:

```
[common]
timeConstant = 0
reshapingCycles = 1
smoothWorkLeftRight = 0
convertImpulseToForce = 0    ← 1=True, 0=False
cummulativeWork = 0    ← 1=True, 0=False
writeAveragedTDivN = 0    ← 1=True, 0=False
absoluteAngleInsteadOfWork = 0    ← 1=True, 0=False

[constantsUpdate]    % this is body balanced updating (according to the loss of ME)
body = brushpin1,brushpin2,brushpin3
constant = c'brushpin1y,c'brushpin2y,c'brushpin3y
maxChange = 0.2e-6

[changeConstants]
constant = c'springA,c'damping
change = -1,-0.03

[reshapeWindow]
cutHiLowValues = 0.0
standardDeviationRange = -100,+100

[bodyBalancedUpdate]
body = brushpin1,brushpin2,brushpin3
maxChange=0.1e-6

[rect1]    % there should be name of the body, see also section 3.4.3
contactWatch=1,2,5-10
[rect2]
contactWatch=5
[rect3]
contactWatch=all
directionX = 0
directionY = 0
directionInside = 0
maxChange =0
```

```

[rect3:0]      % in case of reshapingCycles = 2 this would be used for even history num-
bers
contactWatch=1-5
directionInside = 1
maxChange =1.e-6
[rect3:1]      % in case of reshapingCycles = 2 this would be used for odd history num-
bers
contactWatch=6-15
directionInside = 1
maxChange =1.e-6,-1.e-6      % wear, grow combination
[rect4]
contactWatch=all:
[rect5]
contactWatch=1:3,[4-5]

```

Watches. Here are some notes on how to define *Watches*: the values that will be watched.

contactWatch: define which points of a body should be watched.

contactWatch: use single point (e.g. 1), ranges (e.g. 1-3), combination (e.g. 1,2,3,7-10,15) or all.

contactWatch: use a colon to combine contact points to a single watch (e.g. 1:3 or all:).

contactWatch: if definition inside square brackets (e.g. []) then the impulses are converted from relative normal-tangential coordinate system to the absolute x-y coordinate system (e.g. [1:3] or [all:]).

timeConstant: if >0 then the output impulse is decayed. This is a simulation of measuring the forces with piezo-electric sensor.

cummulativeWork: if is true then the locally lost work is cummulative (summed).

writeAveragedTDivN: if is true then the watches.txt file includes the running average of the tangential/normal impulse (force).

absoluteAngleInsteadOfWork: instead of work the relative angle of contact point is watched. The output is angle between the normal vector and the absolute x axis (in rad). For consistent output: use only with the option *writeToHistoryAtCollision*, see section 3.4.1 and use only with not combined point ranges (e.g. 30-98 is ok, 30:98 is not).

smoothWorkLeftRight if not zero then, the moving average smoothing is applied to the mechanical energy lost at discrete points that define the body.

bodyBalancedUpdate if set then then the statistical properties (mean, standard deviation,...) are calculated according to all listed bodies (by default the properties are calculated for each body individually). *maxChange* in the section *bodyBalancedUpdate* is used instead of the *maxChange* defined for each particular body. Calculation is effected by the *reshapeWindow* section.

Reshaping. The selected points that are being watched can also be reshaped, see example in Section 2.4. This is always done in iteratively: e.g. if the current run is numbered 000 and the option *createNextVersionAtEnd* is enabled (section 3.4.2) then the next version's (001) shape will be reshaped according to the options given below.

Theoretically the approach was presented in [4, 5]: The basic idea is that the amount of wear at selected contact points is proportional to the rate of lost mechanical energy. However, at each iteration the maximum change is defined by *maxChange* occurring at contact point with highest loss of mechanical energy. The direction of reshaping of the contact point is either defined by the body-relative-coordinate: (*directionX*, *directionY*) or by the option *directionInside*. If *directionInside* is 1 or -1 (1=inside, -1=outside) then the *directionX* and *directionY* are calculated for each shape point according to the two neighboring points.

maxChange: defines the maximum reshaping size at the contact point with the highest loss of mechanical energy. *maxChange* can be as single value or as a range (e.g. *maxChange=1e-6, -1e-6*). In case of a range the first value is used as maximum change for contact points with mechanical energy below average (left side of Gaussian) and the second value is used as maximum change for contact points on right side of Gaussian. For example: *directionInside=1* and *maxChange=1e-6,-1e-6* would cause the points with high loss of mechanical energy (left side of Gaussian) to wear and the points with low or no loss of mechanical energy to grow.

reshapeWindow. Sometimes the loss of mechanical energy is irregular with few points taking over all the mechanical load; consequentially, proportional wear would not be appropriate. The *reshapeWindow* options set a filter window for the loss of mechanical energy where *cutHiLowValues* defines the percentage of data cut at low and at high end (to exclude the irregularities). The option *standardDeviationRange* defines the range around the mean loss of mechanical energy that is used to calculate the proportional wear.

The *reshapeWindow* effects: *bodyBalancedUpdate*, *constantsUpdate* and all bodies.

Updating constants. There are two ways for changing constants when a next version of a model is created. The first one is called **constantsUpdate** where the constants are updated according to accumulated mechanical energy of selected bodies. The second one is called **changeConstants** where the constants are changed for a constant value in the next version. If both options are enabled, the *constantsUpdate* is applied first.

In the example above the constants *c'brushpin1y*, *c'brushpin2y*, *c'brushpin3y* will be change proportionally to the total work of bodies *brushpin1*, *brushpin2*, *brushpin3* for a maximum change +0.2e-6. Calculation of the total work is effected by the *reshapeWindow* section (*cutHiLowValues*).

changeConstants: a list of constants that are being iteratively updated when creating a new job. the change is constant according to the value set by *change*. Several constants can be separated by a comma. *changeConstants* is applied after *constantsUpdate*.

reshapingCycles In case different reshaping parameters should be used for different versions of the model (each run/version has its own folder, i.e. 001) **reshapingCycles** can be used. If for example *reshapingCycles=2* is used then the section [bodyName:0] is used for even history names (000, 002, 004, ...) and [bodyName:1] for the odd history names (001, 003, 005, ...). Similarly, *reshapingCycles=3* would designate versions 000, 003, 006, ...

The program first searches for [bodyName:?] sections and if not successful then for [bodyName] section.

Bibliography

- [1] J. Slavič. *Nonlinear and nonsmooth dynamics of discretely defined system of rigid bodies with unilateral contacts*. PhD thesis, Faculty of mechanical engineering, University of Ljubljana, 2005. In Slovene.
- [2] F Pfeiffer and Ch Glocker. *Multibody Dynamics with Unilateral Contacts*. John Wiley & Sons, Inc, New York, 1996.
- [3] J Slavič and M Boltežar. Nonlinearity and non-smoothness in multi body dynamics: application to woodpecker toy. *Journal of Mechanical Engineering Science*, 220(3):285–296, 2006.
- [4] J Slavič and M Boltežar. Simulating multibody dynamics with rough contact surfaces and run-in wear. *Nonlinear Dynamics*, 45(3-4):353–365, August 2006.
- [5] J Slavič, M D. Bryant, and M Boltežar. A new approach to roughness-induced vibrations on a slider. *Journal of Sound and Vibration*, 306(3-5):732–750, Oct 2007.

Index

- ©, 22
- 2 Mass, 13
- 2Mass model, 9
- 2Mass.cns, 11
- 2Mass.cop, 12
- 2Mass.dc, 11
- 2Mass.gc, 10
- 2Mass.hv, 11
- 2Mass.me, 12
- 2Mass.mmx, 11
- 2Mass.wtc, 12

- A, 24
- absoluteAngleInsteadOfWork, 35
- Axes, 6

- B, 24
- b, 10, 22
- bdy, 9, 10, 28
- Bodies, 5
- Body, 5
- bodyBalancedUpdate, 20, 35, 36
- breakTime, 26
- brushpin1,brushpin2,brushpin3, 36

- c, 31
- c'brushpin1y,c'brushpin2y,c'brushpin3y, 36
- change, 36
- changeConstants, 36
- cns, 11, 31
- Collisions, 5
- command-prompt, 27
- constantName, 31
- Constants, 22
- constantsUpdate, 36
- contact watch, 6
- contactWatch, 35
- contexts, 10, 22
- Control window, 5, 12

- cop, 12, 33
- correctOrigin, 29
- Create next version, 5
- createNextVersionAtEnd, 27, 35
- cummulativeWork, 35
- current, 26
- cutHiLowValues, 36

- data.txt, 13
- dc, 11, 30
- default, 25, 27
- Delphi 7, 4
- directionInside, 36
- directionX, 36
- directionX, directionY, 36
- directionY, 36

- exs, 17, 30

- File/Open, 12
- friction, 33

- gc, 10, 30
- Generalized coordinates, 22
- GLScene, 4

- History, 5
- History window, 5
- hv, 11, 31

- initialTemperature, 33

- Job, 6
- job, 20, 27
- jobs, 27
- Jobs window, 5

- LCP Solver, 5
- LCPLexicoZeroTolerance, 25, 27
- LCPPivotToleranceMax, 25
- LCPPivotToleranceMin, 25

- lexicoRandomizationRange, 27
- Log, 5
- Log window, 5
- log.txt, 12
- logLCPMaxNumberOfIterations, 26
- logLCPUnboundedRay, 26

- Main window, 5, 12
- Mass properties, 22, 28
- mass1, 22
- mass1.bdy, 9
- mass1.exs, 10
- Mathematica, 13, 16, 17
- Mathematical functions, 23
- maxChange, 35, 36
- maximumDisplacementFactor, 26, 29
- maximumDisplacementFactor:, 29
- maximumPenetration, 26
- maximumTimestep, 25
- maxLinesInLog, 27
- me, 11, 32
- minimumTimestep, 25
- mmx, 11, 33
- Multibody Dynamics Simulator, 4, 5, 12, 13
- MultiBodyDynamicsSimulator.exe, 4

- Operands:, 23
- Orientation box, 6

- parameterNi, 33
- ParseExpr, 4
- Plot window, 5, 6, 13
- Pointer, 6, 12

- qtintf70.dll, 4

- reshapeWindow, 35, 36
- reshapingCycles, 36
- resistingPenetration, 33
- restitutionNormal, 33
- restitutionTangent, 33
- ring, 16, 17
- ringF, 16
- ringSideL, 17
- ringSideR, 16, 17

- S, 24
- shapeFile, 10
- shapeFile = mass1.exs, 10
- shift+A, 24
- shift+Y, 24
- short-keys, 12
- skinThickness, 28
- smoothWorkLeftRight, 35
- standardDeviationRange, 36
- Start with this, 6

- temp, 33
- timeConstant, 35
- TryToSolveByGoingBack, 26

- virtualToBody, 17, 28
- vRel, 33

- Watches, 35
- watches, 19, 20
- watches.txt, 12, 13
- wld, 25
- Woodpecker 4 DoF, 13
- WoodpeckerToy.dc, 16
- WoodpeckerToy.wld, 16
- World, 5
- World file, 9
- worldfile, 27
- writeAveragedTDivN, 35
- wtc, 12, 34

- Y, 24

- ZeroVelocityTolerance, 26