Rigid Body Project Help

Preface

'Rigid body' section of a course of mechanics is considered as required in most technical colleges and Universities.

Sometimes it's difficult to imagine a rotation of a rigid body in the three-dimensional space. Especially, when an asymmetric top is in question.

Rigid Body project may help students to see the rotations clearly as though they were in the real world.

Introduction

The Rigid Body project demonstrates a rotation of a rigid body about a fixed point without regard for friction forces.

The project is made by using GUIDE of Matlab 7.5.0 (R2007b) and user-defined classes.

The project may be used on the seminars on mechanics in the technical educational establishments.

The terms 'rigid body' and 'top' are convertible In the Rigid Body project.

How to use Rigid Body project

The Rigid Body project consists of 3 GUI files, several m-files for user-defined classes and get_examples.m file with input data of 10 examples. All the files are in RigidBody folder.

To run Rigid Body project

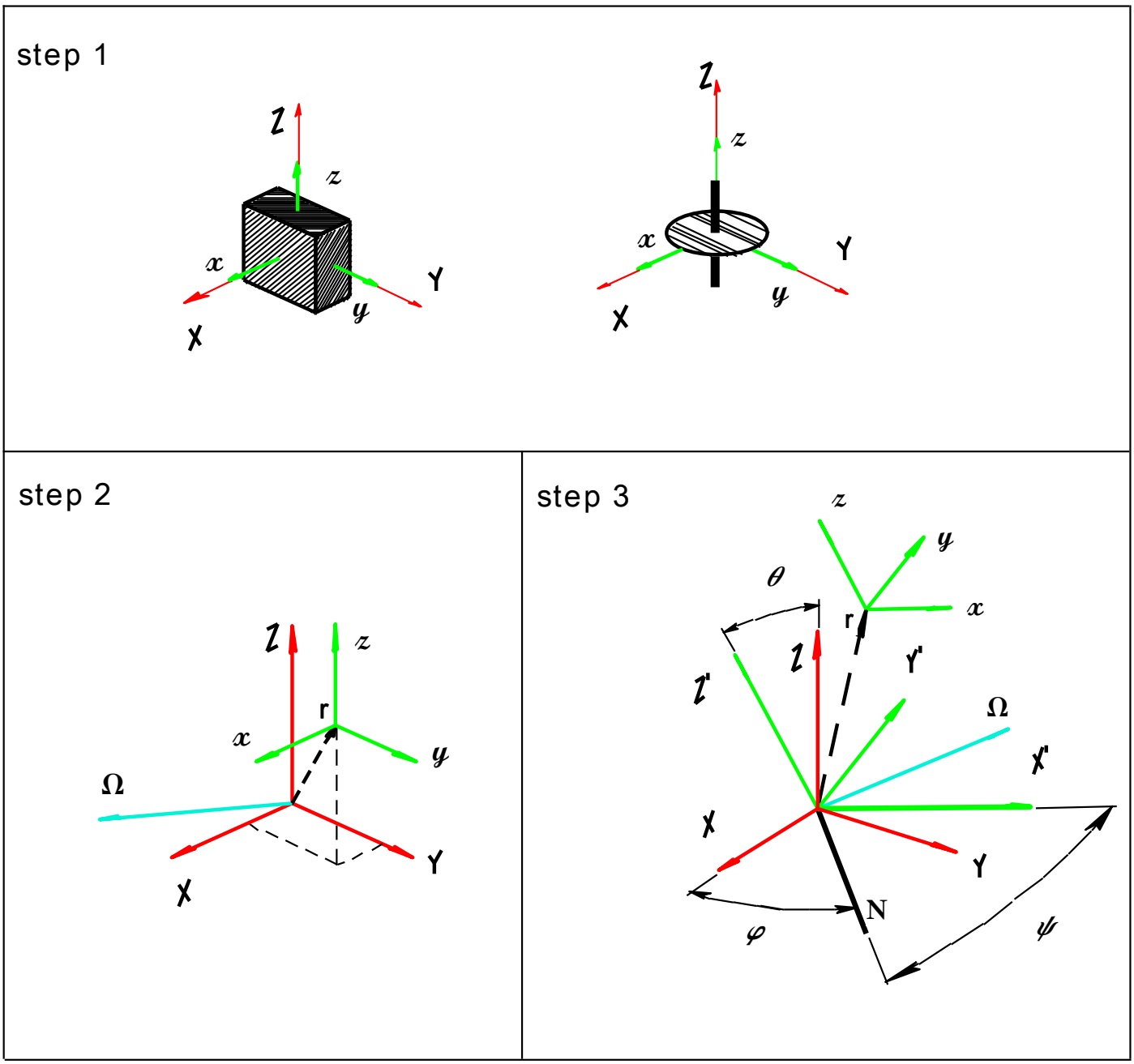
1) Add the folder path to Matlab by using 'addpath' function (For example: >>addpath C:\RigidBody).
2) Enter 'RBInterface' in the Command Window (i.e. >>RBInterface). Press Enter key.
3) Click on 'Input Data' button of 'RBInterface' figure.
4) Enter data on 'RBInputData' figure. Press 'Apply' button.
5) Click on 'Calculate' button of 'RBInterface' figure. Wait a little.
6) Click on 'Animate' button of 'RBInterface' figure.
7) Click on 'START' button of 'RBAnimation' figure.

Note, if data once calculated you need not calculate the next data again if the data does not influence on the calculation algorithm. This occurs when you change only 'Trajectory' object's properties on the blue colored panel (i.e. 'Trajectory' panel).

Input data algorithm

The algorithm works in the following way.

1) Constructs a rigid body so that the principal axes of inertia of the rigid body are oriented as the axes of the laboratory system of coordinates XYZ. The fixed point of the rigid body coincides with the origin of the laboratory system XYZ.
2) Translates the rigid body along the vector **r** that coordinates are in the 'Center of mass' panel. So the fixed point may not belong to the rigid body.
3) Rotates the rigid body about the origin of XYZ using Euler angles (theta, phi, and psi) that are in 'Initial conditions' panel. The initial angular velocity **Ω** is being rotated together with the rigid body. So the angular velocity's coordinates are being calculated in the body-fixed system of coordinates $\boldsymbol{x}\ \boldsymbol{y}\ \boldsymbol{z}$.



N – line of nodes.

Note, the angular velocity vector **Ω** is not shifted together with the rigid body in order to emphasize the fact of the angular velocity independence of this shifting. The advantage is that we can draw the line in the direction of the initial angular velocity from the fixed point (0,0,0) and consider this line as the initial instantaneous axis of a rotation. This is more natural in contract to that if we shifted the initial angular velocity together with the rigid body. The line represented the initial instantaneous axis is marked with Aqua color.

In 'RBInterface' and 'RBAnimation' figures #-symbol is used to show the fixed point (0,0,0).

Formulas

Here the primary equations are provided. They were used to make Rigid Body project.

The correspondence between the variables entered into the equations and the variables used in Rigid Body program interface and source code is given in [the list of symbols](#).

t $-time$

$$A(t)=\begin{pmatrix} a_{11}(t) & a_{12}(t) & a_{13}(t) \\ a_{21}(t) & a_{22}(t) & a_{23}(t) \\ a_{31}(t) & a_{32}(t) & a_{33}(t) \end{pmatrix} \quad -rotation\ matrix$$

$r=(r_x,r_y,r_z)$ $-center\ of\ mass$

$\Omega(t)=(\omega_x(t),\omega_y(t),\omega_z(t))$ $-angular\ velocity$

$V(t)=\omega(t)\times r$ $-velocity\ of\ the\ center\ of\ mass$

m $-mass\ of\ the\ rigid\ body$

$P(t)=m\cdot V(t)$ $-momentum$

I_x,I_y,I_z $-principal\ moments\ of\ inertia$

$M(t)=(I_x\omega_x,I_y\omega_y,I_z\omega_z)$ $-angular\ momentum$

$a=(0,0,-g)'$ $-gravitational\ acceleration\ vector$

$R(t)=(R_x(t),R_y(t),R_z(t))$ $-reaction\ force$

$F(t)=m\,A'(t)\,a+R(t),$ $A'(t)-transposed\ rotation\ matrix$

$K(t)=-r\times R(t)$ $-moment\ of\ R(t)$

$$\frac{dP(t)}{dt}+\Omega(t)\times P(t)=F(t) \quad -forces\ balance$$

$$\frac{dM(t)}{dt}+\Omega(t)\times M(t)=K(t) \quad -moments\ balance$$

$$\frac{dA(t)}{dt}=A(t)\cdot\begin{pmatrix} 0 & -\omega_z(t) & \omega_y(t) \\ \omega_z(t) & 0 & -\omega_x(t) \\ -\omega_y(t) & \omega_x(t) & 0 \end{pmatrix}$$

$$\omega_x(0)=\dot{\varphi}(0)\sin(\theta(0))\sin(\psi(0))+\dot{\theta}(0)\cos(\psi(0))$$
$$\omega_y(0)=\dot{\varphi}(0)\sin(\theta(0))\cos(\psi(0))-\dot{\theta}(0)\sin(\psi(0))$$
$$\omega_z(0)=\dot{\varphi}(0)\cos(\theta(0))+\dot{\psi}(0)$$

$$A_3(\varphi(0))=\begin{pmatrix} \cos(\varphi(0)) & -\sin(\varphi(0)) & 0 \\ \sin(\varphi(0)) & \cos(\varphi(0)) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$A_1(\theta(0))=\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta(0)) & -\sin(\theta(0)) \\ 0 & \sin(\theta(0)) & \cos(\theta(0)) \end{pmatrix}$$

$$A_3(\psi(0))=\begin{pmatrix} \cos(\psi(0)) & -\sin(\psi(0)) & 0 \\ \sin(\psi(0)) & \cos(\psi(0)) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$A(0)=A_3(\varphi(0))\cdot A_1(\theta(0))\cdot A_3(\psi(0))$$

$A, \varphi, \theta, \psi, r$ and a are given in the laboratory system of coordinates.
All other variables are given in the fixed-body system of coordinates.

The final 12 differential equations (to calculate the rotation matrixes A(t)) are embedded in ODE_rb.p file. If you wish to get them please [contact me (the author of Rigid Body project).](#) Of course there is no discovery or secret in my code. I just would like to know a person who has shown an interest in my project and his (her) opinion about it.

Examples

You may choose from 10 examples by checking one from 'Examples' list box of 'RBInputData' figure.

1) Euler case
2) Lagrange case #1
3) Lagrange case #2
4) Lagrange case #3
5) Kovalevskaja case
6) Most general case
7) Spherical Top
8) Pendulum
9) Time control
10) Light effect

When 7, 8 or 10 case is chosen the additional information is being output to the command window of Matlab after 'Apply' button is pressed.

That is the period of the pendulum in the 8 case, the check of the stability of motion in 7 case and a formula to get the light effect on your computer in 10 case.

What is the light effect?

You may match the proper rotation of the symmetric top (i.e. dpsi variable on condition that dtheta=0 and dphi =0) with the fpsRate variable which is shown on the 'RBAnimation' figure when the animation is running. The following formula gives this matching.

dpsi = 2 * pi * fpsRate / 3 ;

When dpsi ≈ 2 * pi * fpsRate / 3 , the red, green and blue colored sectors of the top's disk are being alternated with such a speed that the dirty white color appears through three given colors. In the real world you can see the almost white color. But the computer screen is a quite another story as it turned out. Note, fpsRate(It's short for frames per second rate) has no concern with frequency parameters of the computer monitor. fpsRate is being calculated inside the while loop in the pushbutton2_Callback event handler.

Classes

These are 'RigidBody', 'SymmetricTop', 'AsymmetricTop' and 'Trajectory'.

'RigidBody' is a parent class for 'SymmetricTop' and 'AsymmetricTop'.

Symmetric Top consists of the thin circular disk and the thin rod.

md – mass of the disk; mL – mass of the rod;

d – diameter of the disk; L – length of the rod;

Asymmetric Top is represented as a homogenous, solid brick.

m – mass of the brick;

a, b, c – lengths of the sides of the brick;

These parameters can be entered from 'Top' panel.

Notes:

1) If the fixed point (0, 0, 0) does not belong to the rigid body you can see the dashed segment that connects the fixed point with the center of mass of the rigid body. This segment can be considered as the rod with zero mass and which is rigidly connected with the rigid body.

2) It's clear that a symmetric top can be constructed by using 'AsymmetricTop' class too.

Trajectory is started from the point which coordinates x, y and z can be entered from 'Trajectory' panel. The point is regarded as rigidly connected with the rigid body even if it does not belong to the rigid body.

Trajectory has 2 parameters. These are tr_L and tr_S. They must be integer numbers. The length of the trajectory and its density depend on these parameters. The more tr_L the more length of the trajectory will be. The more tr_S the less the density of the trajectory will be.

x, y, z, tr_L, tr_S can be entered from 'Trajectory' blue panel. The panel is blue colored because it is rather different from other panels. The point is that tr.point, tr_L, tr_S parameters do not influent on the calculation algorithm that calculates rotation matrixes. So you need not to calculate the rotation matrixes again if the parameters from only 'Trajectory' panel have been changed.

Functions

1) Control the step of the time variable.
   It is available by the slider control of 'RBAnimation' figure.
2) Real-time rendering.
   It is on after pressing 'Real Time ON' button of 'RBAnimation' figure.
   Press 'Real Time OFF' button to make real-time rendering off.
3) Selecting the trajectory object to be rendered. Use 'Trajectory' check-box to show/hide the trajectory of the point.

Note, at this point the real-time does not mean a real-time interactivity but a (Time – t) value constancy. Where Time is a computer real time obtained with 'cputime' function and t is a variable that represents the time. So when (Time – t) ≈ Const, you can see the animation as though it were in reality.

The 8^{th} and 9^{th} examples illustrate the real-time effect very clearly.

List of symbols

In the formulas	In the program interface	In the source code	Description
t	t	i * dt	time variable
A(t)	–	rb.rm(:,i)	rotation matrix
$r = (r_x, r_y, r_z)$	x, y, z	rb.rc	center of mass
$\Omega = (\omega_x, \omega_y, \omega_z)$	–	–	angular velocity
V	–	–	velocity of the center of mass
m	m	rb.m	mass of top
P	–	–	momentum
I_x, I_y, I_z	–	rb.pmi	principal moments of inertia
M	–	–	angular momentum
g	g	rb.g	gravitational acceleration
a = (0, 0, – g)	–	–	gravitational acceleration vector
$R = (R_x, R_y, R_z)$	–	–	reaction force
K	–	–	moment of R
$\theta(0), \varphi(0), \psi(0)$	theta, phi, psi	rb.theta, rb.phi, rb.psi	initial Euler angles
$\dot{\theta}(0), \dot{\varphi}(0), \dot{\psi}(0)$	dtheta, dphi, dpsi	rb.dtheta, rb.dphi, rb.dpsi	initial speeds for Euler angles
$(\omega_x(0), \omega_y(0), \omega_z(0))$	–	rb.q0	initial angular velocity
A(0)	–	rb.rm0	initial rotation matrix
–	md	st.md	mass of disk
–	mL	st.mL	mass of rod
–	d	st.d	diameter of disk
–	L	st.L	length of rod
–	a	at.a	x-size of brick
–	b	at.b	y-size of brick
–	c	at.c	z-size of brick
–	dt	rb.dt	increment of t
–	tr_L, tr_S	tr.tr_L, tr.tr_S	parameters of trajectory
–	x, y, z	tr.point	initial point for trajectory
–	Time	cputime - newC	computer time
–	t step	step * dt	increment of t
–	fpsRate	fpsRate	frames per second rate
–	T	rb.T	time interval
–	–	rb.n	floor(rb.T/rb.dt)

Bibliography

1) L.D. Landau, E.M. Lifshitz "Mechanics", Volume 1 (Course of Theoretical Physics), 1976

2) Korn,G.A.and Korn,T.M "Mathematical Handbook for Scientists and Engineers" ,1968

Author

Author: Ferat Talat oglu

E-mail: ferattalatoglu@ymail.com

Internet: http://arithmetic.eu.pn