tops:
Classes for computing rigid body rotation user documentation

Ramses van Zon[*]

Chemical Physics Theory Group, Department of Chemistry, University of Toronto,

80 St. George Street, Toronto, Ontario M5S 3H6, Canada

May 9, 2007

Abstract

This document describes how to use the c++ Top classes defined in the header file tops.h and implemented in the file tops.cc. These classes can compute the exact rotation of an arbitrary rigid body, and have been written to be easy to use while being efficient.

Note: there is also an implementation in C, which is described in docctops.pdf.

Contents

[*]rzon@chem.utoronto.ca

1 Introduction

Rigid body dynamics is encountered in many physical models, such as for molecules, polymers, robotics and even the computer game industry. The classes defined in *tops.h* give an easy-to-use implementation of the exact rotation for arbitrary rigid bodies.

Before explaining how to use the classes in tops.h, it is necessary to explain a bit of the notation concerning rigid bodies.

A rigid body is a body in which the relative position of all its parts is fixed. This still allows for the body to translate and to rotate. For a free rigid body, translation is governed by the linear velocity $\mathbf{V}$, such that the position $\mathbf{R}_t$ of the mass center of the body at time t is related to that at time 0 by $\mathbf{R}_t = \mathbf{R}_0 + \mathbf{V}t$. This is so simple that there is no need to write a specialized c++ class for translation. Therefore, *tops* only deals with the rotational body of the motion.

The rotation, then, takes place relative to the center of mass. The body can be oriented in an arbitrary way. Suppose that a special reference orientation has been chosen. Then any other orientation can be obtained by rotating the body. A rotation can be represented by a 3x3 orthogonal matrix A, so this also represents the orientation. In this context, A is called an attitude matrix. If the position of a point on the body in the reference orientation is $\tilde{\mathbf{r}}$, then in the rotation orientation, its position is $\mathsf{A}^T \cdot \tilde{\mathbf{r}}$, where A^T is the transpose of A.

In time, the attitude matrix A can change, but it will remain orthogonal. The change is governed by the angular velocity vector ω. For a body with angular velocity vector ω, the velocity of a point $\mathbf{r}$ is $\omega \times \mathbf{r}$. Different from the velocities for translational motion, the angular velocity is not necessarily constant in time for a free body. The way it changes depends on the body inertial moment matrix. This matrix is symmetric and can be diagonalized. The orientation of the body in which the inertial moment matrix is diagonal will be taken as the reference orientation, and the resulting diagonal elements I_x, I_y and I_z are called the principal moments of inertia. The angular velocity vector is constant only if the principal moments of inertia are all the same. We then say that the body is “a spherical top” (as far as the rotation is concerned). Another kind of body is the symmetric top, for which two moments of inertia are the same. If the equal moments of inertia are smaller than the unequal one, the body is called “Oblate”, while if the equal moments of inertia are larger than the unequal one, the body is called “Prolate”. Finally, if all moments of inertia are different the body is said to be “Asymmetric”. This nomenclature is used for the names of the Top classes in tops.h.

In all cases, the rotation of the body can be solved exactly, although the mathematics gets somewhat involved for an asymmetric body[1,2,3]. This, does not mean, however, that this solution cannot be implemented efficiently, and indeed, this is what tops.h is does.

2 Header file and linkage

To use *tops*, the following general procedure should be followed:

- The Top classes for asymmetric bodies need to evaluate elliptic functions and elliptic integrals. In the current implementation, these are computed using the GNU Scientific Library (gsl), which is freely available at `http://www.gnu.org/software/gsl/`.

$\boxed{\textbf{The Top classes cannot be used if gsl is not installed!}}$

- The Top classes use vectors and matrices defined in the header file vecmat3.h. While tops.h includes this file automatically, it needs to be in the same directory as the tops.h file.[1] *Vecmat3* is an efficient implementation of three dimensional vectors and matrices which is strongly recommended.[2]

$\boxed{\textbf{The Top classes cannot be used if vecmat3.h cannot be found!}}$

- Include the header file tops.h:

  ```
  #include "tops.h"
  ```

- The classes Top, TopSpherical, TopProlate, TopOblate, TopAsymmetric and TopRecur and are now defined and can be used as explained in the next section.
- The implementation of these classes can be found in the source file “tops.cc”, which needs to be compiled and linked with any program using tops.
- When compiling or linking your program, the gsl library will need to be linked in as well, using “-lgsl -lgslcblas”. E.g., to compile testtops.cc, use

  ```
  c++ testtops.cc tops.cc -lm -lgsl -lgslcblas -o testtops
  ```

  with the files tops.h and vecmat3.h in the current directory.

3 Initialization

The Top classes can be initialized as follows:

```
TopSpherical  sphere (Ix);          // Ix = Iy = Iz
TopProlate    prolate(Ix, Iz);      // Ix < Iy = Iz
TopOblate     oblate (Ix, Iz);      // Ix = Iy < Iz
TopAsymmetric asymtop(Ix, Iy, Iz); // Ix < Iy < Iz
TopRecur      recurse(Ix, Iy, Iz);
```

Notes:

[1]or in a directory that is searched for header files by the compiler.
[2]See docvecmat3.pdf.

1. These classes require arguments, i.e., one cannot define a Top without specifying its moments of inertia.
2. For spherical top, as single moment of inertia is enough, for the two types of symmetric tops, one needs two, while in general three moments of inertia are required.
3. The ordering of the moments of inertia indicated above is required, i.e., the specified moments of inertia need to be given in ascending order.
4. As the names suggest, TopSpherical computes the rotation of a spherical top, TopProlate, that of a prolate symmetric top, TopOblate that of a oblate symmetric top, and TopAsymmetric that of an asymmetric Top. All of these classes are derived from a parent class Top, in which the member functions discussed below are virtually overloaded.
5. The class TopRecur is a variant of TopAsymmetric which works only for small enough times, as explained below, but which is considerably faster than TopAsymmetric.

4 Initial conditions

Given an object `top` of any of the above mentioned Top classes, one can set the initial angular velocity ω_0 (denoted by `omega0` in the code) and the initial attitude matrix A_0 (simply `A0`), using

```
Vector omega0 (1,2,3);
Matrix A0 (0, 1, 0,
           1, 0, 0,
           0, 0,-1);
top.Initialization( omega0, A0 );
```

5 Computing future rotations

Once the moments of inertia have been specified and the initial conditions are set, the angular velocity ω and attitude matrix A at time t can be computed as follows:

```
Vector omega;
Matrix A;
double t = 1.5; // arbitrary time
top.Evolution(t, omega, A);
cout << omega << A; // write out the result
```

Note that one can request the the angular velocity ω and attitude matrix A at several times without having to specify the moments of inertia or the initial conditions again.

6 Propagation

In some applications, one only needs to update the old ω and A to new ones a time interval *dt* later. For this purpose, the Top classes contain a function Propagate, to be used as follows:

```
Vector omega (1,2,3);
Matrix A ( 0, 1, 0,
           1, 0, 0,
           0, 0,-1);
double t = 1.5;
top.Propagate(t, omega, A);
cout << omega << A; // write out the result
```

Note that after calling Propagate, the original value of `omega` and `A` are lost.

7 Example

```
#include <fstream>
#include "tops.h"
using namespace std;
void testTop( Top & rotor, const char* filename ) {
  Vector omega0 ( 1.0, -1.0, 0.5 );
  Matrix A0 ( 1,0,0,
              0,1,0,
              0,0,1 );
  rotor.Initialization( omega0, A0 );
  ofstream f( filename );
  for ( double t = 0.0; t < 12.0; t += 0.1 ) {
    Vector omega;
    Matrix A;
    rotor.Evolution( t, omega, A );
    f << fixed << t
      << ' ' << A.row(0)
      << ' ' << A.row(1)
```

```
          << ' ' << A.row(2)
          << ' ' << omega
          << endl;
    }
  }
  int main() {
    double Ix = 1.0, Iy = 1.5, Iz = 2.0;
    TopSpherical  sphere (Ix);          // Ix = Iy = Iz
    TopProlate    prolate(Ix, Iz);      // Ix < Iy = Iz
    TopOblate     oblate (Ix, Iz);      // Ix = Iy < Iz
    TopAsymmetric asymtop(Ix, Iy, Iz); // Ix < Iy < Iz
    TopRecur      recurse(Ix, Iy, Iz);

    testTop( sphere, "sphere.dat" );
    testTop( prolate, "prolate.dat" );
    testTop( oblate, "oblate.dat" );
    testTop( asymtop, "asymtop.dat" );
    testTop( recurse, "recurse.dat" );
  }
```

Background references

[1] Lisandro Hernandez de la Pena, Ramses van Zon, Jeremy Schofield and Sheldon B. Opps, *Discontinuous molecular dynamics for semi-flexible and rigid bodies*, Journal of Chemical Physics **126**, 074105 (2007).

[2] Ramses van Zon and Jeremy Schofield, *Numerical implementation of the exact dynamics of free rigid bodies*, Journal of Computational Physics, doi:10.1016/j.jcp.2006.11.019 (2007).

[3] Ramses van Zon and Jeremy Schofield, *Symplectic algorithms for simulations of rigid-body systems using the exact solution of free motion*, Physical Review E **75**, 056701 (2007).

[4] The comments in `tops.cc`