Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

computer graphics laboratory

PHYSICALLY-BASED SIMULATION COURSE 2018

EXERCISE 2 - RIGID BODY ROTATION

Handout date: 03.10.2018

General Rules

Setup. Please go to our gitlab repository (https://gitlab.vis.ethz.ch/cglphysics/PBS18-Exercises) and carefully follow the instructions to update and run your forked project.

Goal of this exercise

In this exercise, you will rotate a rigid body object with the following ways:

- Matrix-based Rotation
- Quaternion-based Rotation
- Gyroscopic Torque with Semi-Implicit Euler
- Gyroscopic Torque with Implicit Euler

Problem 1: Matrix-based Rotation

Statement. We will rotate a rigid body object cube b of which status $\mathbf{S}_b$ and its derivative $\mathbf{S'}_b$ are defined as follows:

$$\mathbf{S}_b = \begin{pmatrix} \mathbf{b} \\ \mathbf{R} \\ \mathbf{p} \\ \mathbf{l} \end{pmatrix}, \mathbf{S'}_b = \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\Omega} \\ \mathbf{f} \\ \boldsymbol{\tau} \end{pmatrix}, \tag{1}$$

where $\mathbf{b}$ is the position of center of mass m, $\mathbf{R}$ is the rotation matrix, $\mathbf{p}$ is the linear momentum, $\mathbf{l}$ is the angular momentum, $\mathbf{v}$ is the linear velocity, $\boldsymbol{\Omega}$ is the skew matrix of the angular velocity $\boldsymbol{\omega}$,

and $\boldsymbol{\tau}$ is the torque. In this configuration, the force is applied once at the first step. Thus, a cube never stops to rotate with constant linear and angular momentum.

In this problem, you are required to implement a semi-implicit euler integrator for each state of a cube. Given pre-computed inverse of inertia tensor $\mathbf{I}_b^{-1}$ in *local coordinate*, you can (1) compute the inverse of inertia tensor $\mathbf{I}_t^{-1} = \mathbf{R}_t\mathbf{I}_b^{-1}\mathbf{R}_t^T$ in *world coordinate* at time t. Then, (2) update the angular velocity $\boldsymbol{\omega_{t+1}} = \mathbf{I}_t^{-1}\mathbf{l}_{t+1}$ to get the skew matrix of it $\boldsymbol{\Omega}_{t+1}$ which is defined as:

$$\boldsymbol{\Omega} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \tag{2}$$

Finally, we can (3) integrate the orientation with a time step Δt such as $\mathbf{R}_{t+1} = \mathbf{R}_t + \Delta t\boldsymbol{\Omega}_{t+1}\mathbf{R}_t$

Please fill in `advance()` method by implementing procedures (1)-(3).

Relevant functions. Take a look at the region *GettersAndSetters* of `RigidObject.h`

Goal. When you run simulation, you should see a spinning cube of which volume expands as seen in Fig. 1.

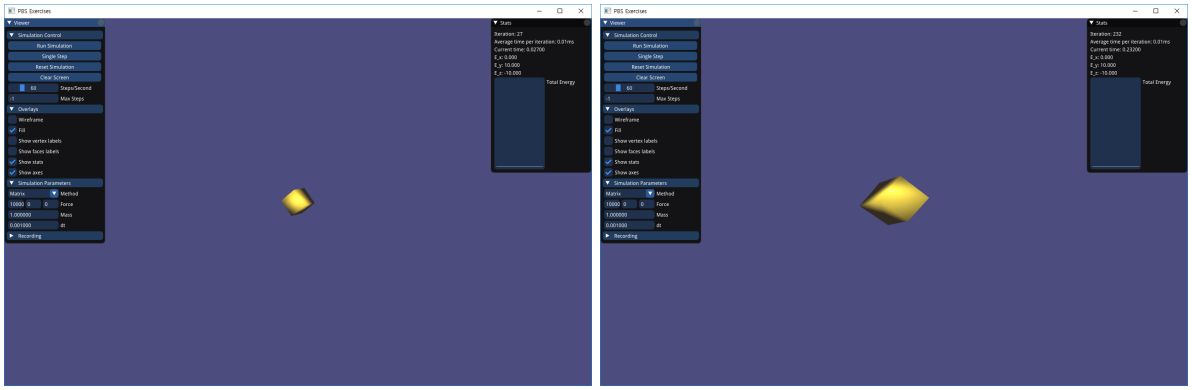

FIGURE 1. Screenshot of an example result of problem 1. The volume of a cube expands while spinning.

PROBLEM 2: QUATERNION-BASED ROTATION

Statement. In this problem, we will fix the instability problem of matrix-based rotation scheme. The integration of rotation $\mathbf{R}$ with the matrix $\boldsymbol{\Omega}$ has no constraint for the orthogonality of the group $SO(3)$. One of the solutions is to use normalized quaternions.

Replacing $\mathbf{R}$ by quaternions yields:

$$\begin{pmatrix} \mathbf{R} \rightarrow \mathbf{q} \\ \boldsymbol{\omega'} = [0, \boldsymbol{\omega}] \\ \mathbf{q^*_{t+1}} = \mathbf{q}_t + \frac{\Delta t}{2} \boldsymbol{\omega'}_{t+1} \mathbf{q}_t \\ \mathbf{q_{t+1}} = \frac{\mathbf{q^*_{t+1}}}{\|\mathbf{q^*_{t+1}}\|} \end{pmatrix} \tag{3}$$

Please fill in `advance()` method by implementing quaternion-based scheme.

Goal. You will see a spinning cube without volume expansion. What is the advantage of using quaternion instead of matrix?

Problem 3: Gyroscpoic Torque with Semi-Implicit Euler

Statement. Now it's time to incorporate gyroscopic torque $\boldsymbol{\tau}_g$ to handle the case where an angular velocity and momentum may point in different directions. Gyroscopic torque is defined as

$$\boldsymbol{\tau}_g = \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}, \tag{4}$$

and the discretized form in semi-implicit euler is

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \Delta t \mathbf{I}_t^{-1}(\boldsymbol{\tau}_t - \boldsymbol{\omega}_t \times \mathbf{I}_t \boldsymbol{\omega}_t), \tag{5}$$

where $\mathbf{I}_t^{-1} = \mathbf{q}_t \mathbf{I}_b^{-1} \mathbf{q}_t^T$. In this problem, you are required to implement the equation 5 in `advance()` method. Then, integrate orientation in a quaternion-based way just same as solved in problem 2.

Goal. You will see a spinning t-shape rigid body object showing *Dzhanibekov effect* that rotation of an object around its first and third principal axes is stable, while rotation around its second principal axis (or intermediate axis) is not. However, semi-implicit euler blows up because it extrapolates velocity. Since the gyroscopic torque is quadratic in angular velocity, the extrapolation diverges. Check if it is getting faster and the energy increases as shown in Fig. 2.

Problem 4: Gyroscpoic Torque with Implicit Euler

Statement. According to the superposition principle, the effect of multiple separate torques can be additive. Thus, it allows to use implicit Euler on the gyroscopic torque while using explicit Euler on all the torques as follows:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \Delta\boldsymbol{\omega}_t + \Delta\boldsymbol{\omega}_{t+1}, \tag{6}$$

where $\Delta\boldsymbol{\omega}_t = \Delta t \mathbf{I}_t^{-1} \boldsymbol{\tau}_t$, $\Delta\boldsymbol{\omega}_{t+1} = -\Delta t \mathbf{I}_{t+1}^{-1} \boldsymbol{\omega}_{t+1} \times \mathbf{I}_{t+1} \boldsymbol{\omega}_{t+1}$. We need to get the implicit solution of $\Delta\boldsymbol{\omega}_{t+1}$, which is $\mathbf{I}_{t+1}(\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t) + \Delta t \boldsymbol{\omega}_{t+1} \times \mathbf{I}_{t+1}\boldsymbol{\omega}_{t+1} = \mathbf{f}(\boldsymbol{\omega}_{t+1}) = 0$. This is quadratic in $\boldsymbol{\omega}_{t+1}$, so we will use Newton's method only for a *single* step.

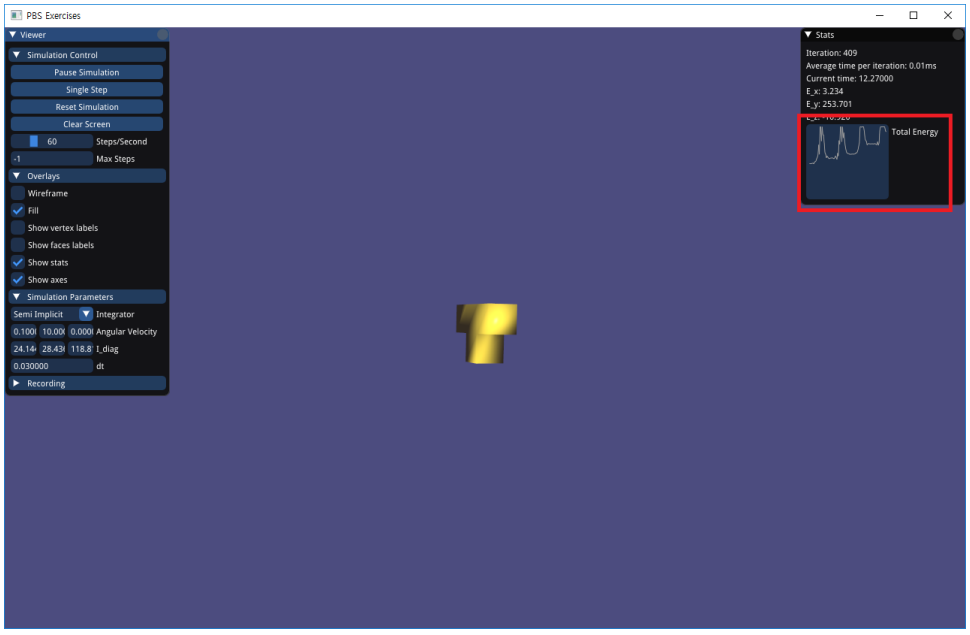

FIGURE 2. Screenshot of an example result of problem 3. The red box highlights the plot of increasing energy.

However, we cannot solve for $\mathbf{I}_{t+1}$ as the inertia tensor in world coordinates depends on the orientation of the rigid body. Instead, we will transform $\mathbf{f}(\boldsymbol{\omega}_{t+1})$ to local (body) coordinates as follows:

$$\boldsymbol{\omega}_b = \mathbf{q}^{-1}\boldsymbol{\omega}_{t+1}, \quad \mathbf{f}(\boldsymbol{\omega}_b) = \Delta t \boldsymbol{\omega}_b \times \mathbf{I}_b \boldsymbol{\omega}_b. \tag{7}$$

Transforming a single non-linear update to local coordinates yields

$$\mathbf{f}'(\boldsymbol{\omega}_b^1)\Delta\boldsymbol{\omega}_b = -\mathbf{f}(\boldsymbol{\omega}_b^0), \quad \boldsymbol{\omega}_b^0 = \mathbf{q}^{-1}\boldsymbol{\omega}_t. \tag{8}$$

Lastly, we compute the Jacobian of $\mathbf{f}'(\boldsymbol{\omega}_b^1)$ as:

$$\mathbf{J} = \mathbf{I}_b + \Delta t(\boldsymbol{\Omega}_{\boldsymbol{\omega}_b}\mathbf{I}_b - \boldsymbol{\Omega}_{\mathbf{I}_b\boldsymbol{\omega}_b}) \tag{9}$$

where $\boldsymbol{\Omega}$ is the skew-matrix of lower case.

Once you solve the equation 8, it should be transformed back to the world coordinated as follows:

$$\boldsymbol{\omega}_b = \boldsymbol{\omega}_b^0 + \Delta\boldsymbol{\omega}_b, \quad \boldsymbol{\omega}_{t+1} = \mathbf{q}\boldsymbol{\omega}_b \tag{10}$$

Relevant functions. For Newton step, try this line with $\mathbf{J}$ and $\mathbf{f}$: `J.colPivHouseholderQr().solve(f)`

Goal. You will see a spinning t-shape rigid body object showing weakly decreasing energy level as seen in Fig. 3.

FIGURE 3. Screenshot of an example result of problem 4. The red box highlights the plot of decreasing energy.