Rigid body dynamics using Euler's equations, Runge-Kutta and quaternions.

Indrek Mandre <indrek@mare.ee>
http://www.mare.ee/indrek/

February 26, 2008

1 Motivation

I became interested in the angular dynamics of arbitrarily shaped rigid bodies and started to look around the Internet. I didn't find a consistent look on this problem, also there are many different ways to approach this problem.

This motivated me to do a bit of research and derive my own set of equations for numeric integration. I avoid matrices as much as possible and use quaternions[1] to represent body orientation. I base my algorithms on the Euler's equations[2] and the fourth order Runge-Kutta[3, 4] numeric integration method.

I don't address any proofs or even go very deep into different physical concepts. Rather I try to lay out the useful concepts and show how to practically use them for calculations. So this document is more like a manual for an engineer.

2 Rigid body in space

A rigid body moving freely through space with no forces applied on it has the following property: its center of mass moves in a straight line and at constant velocity. The body may also rotate around its center of mass during this straight movement. The rotation can be described through two properties: angular velocity $\boldsymbol{\omega}$ which is the rate of rotation in radians per second and angular momentum $\mathbf{L}$. In case the angular velocity and momentum vectors are not on the same axis ($|\boldsymbol{\omega} \times \mathbf{L}| \neq 0$), the angular velocity vector will move around causing the body to additionally wobble as it rotates. That is: the angular velocity vector of a body that has no forces applied on it does not have to be constant but can change as the body rotates. This wobbling is in some cases called the "torque free precession" and is caused because the angular momentum has to be conserved.

In case a force $\mathbf{F}$ is applied on the body's center of mass the angular movement of the body is unaffected. Only the path of the center of mass changes. This happens through acceleration caused by the force (the familiar $\mathbf{F} = m\mathbf{a}$).

Now lets look at a more complex case when the force is applied away from the center of mass. In that case the force can change two things at the same time: the path of the center of mass and also the angular movement.

Let the vector from the center of mass to the point where the force is applied be $\mathbf{r} = \mathbf{p_F} - \mathbf{p_C}$. We can divide the force vector $\mathbf{F}$ into two components in relation to $\mathbf{r}$: the part perpendicular to $\mathbf{r}$ (angular force $\mathbf{F}_\perp$ that affects the angular movement but not the movement of the center of mass) and the part along $\mathbf{r}$:

$$\begin{aligned} \mathbf{F} &= \mathbf{F}_\parallel + \mathbf{F}_\perp \\ \mathbf{F}_\parallel &= (\mathbf{r} \cdot \mathbf{F}) \frac{1}{\mathbf{r} \cdot \mathbf{r}} \mathbf{r} \\ \mathbf{F}_\perp &= \mathbf{F} - \mathbf{F}_\parallel \end{aligned}$$

The center of mass accelerates again based on the full force $\mathbf{F}$ ($\mathbf{F} = m\mathbf{a}$)[1]. The angular force is much more interesting. The effect a force can affect on the angular movement of a body depends on the distance from the center of mass. The principle is similar to that of a lever - the longer the lever the greater the effect. This force-distance quantity is called torque and is the principal concept used when dealing with the angular movement of a body:

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F}_\perp$$

In case of torque the cross product already takes care of removing the collinear component from the force vector, so we can simply write:

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F}$$

3 Moment of inertia tensor

When applying force on the center of mass of a body the body accelerates. This is according to the equation $\mathbf{F} = m\mathbf{a}$. The acceleration clearly depends on the mass of the body. Applying the same valued force on a larger mass will result in smaller acceleration - resulting in a smaller effect.

This same concept can be applied to angular movement. Applying torque on bodies with different masses will result in smaller or bigger effects. But this is not all - the change is not only dependent on the total mass of the body but also on its shape (that is distribution of its mass). And to make things even more complicated: the effect depends on the direction of the torque vector in relation to the body frame - meaning it depends on the shape of the body around the torque vector.

[1] Previously I had stated here it only depended on the part along $\mathbf{r}$: $\mathbf{F}_\parallel$ - this was wrong as pointed out by Harald Fischer, Dec. 2009 - the acceleration of the center of mass depends on the full force $\mathbf{F}$ no matter in what part of the body or in what direction it is applied to.

The concept that completely (for use at any axis of rotation or torque) describes this behavior is called the moment of inertia tensor. Mathematically it is a simple 3×3 matrix:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

The moment of inertia tensor depends on the orientation of the body. So in case the body rotates its moment of inertia tensor in the world frame will change as well. Let the body's rotation be represented by a 3×3 matrix $\mathbf{A}$. The moment of inertia tensor can be translated from the body frame into the world frame using this equation:

$$\mathbf{I}^{\mathbf{W}} = \mathbf{A}\mathbf{I}\mathbf{A}^{\mathbf{T}}$$

The W here marks the world frame and we also assume that the center of mass of the body is at the center of the world frame. In the body's frame of reference the moment of inertia tensor is of course a constant as the distribution of mass around the coordinate axes in the body frame is constant.

One easy way to calculate the moment of inertia tensor for a rigid body is to split the body into simple geometric shapes, find their individual moment of inertia tensors in reference to the center of mass of the body and add them up. The moment of inertia tensors for many geometric shapes can be seen from this URL:

`http://en.wikipedia.org/wiki/List_of_moment_of_inertia_tensors`

So let the body consist of multiple geometric shapes. To calculate the total moment of inertia tensor for the entire body we have to iterate through the geometric shapes, calculate their individual moment of inertia tensors, transforms them from the geometric shape frame into the body frame and add them up. Aside from the rotational transform we usually have to also do a positional transform on the tensor. So let the vector from the center of mass of the body to the center of mass of the geometric shape be $\mathbf{r}$. The positional transform looks like this:

$$\mathbf{I}^{displaced} = \mathbf{I}^{center} + m\left[(\mathbf{r} \cdot \mathbf{r})\,\mathbf{1} - \mathbf{r} \otimes \mathbf{r}\right]$$

Here m is the mass of the geometric shape, $\mathbf{1}$ is the 3×3 identity matrix and $\otimes$ marks the outer product.

So the complete moment of inertia tensor for a rigid body that consists of n geometric shapes is:

$$\mathbf{I} \quad = \quad \sum_{i=1}^{n} \left[\mathbf{A_i}\mathbf{I_i}\mathbf{A_i^T} + m_i\left((\mathbf{r_i} \cdot \mathbf{r_i})\,\mathbf{1} - \mathbf{r_i} \otimes \mathbf{r_i}\right)\right]$$

4 Properties of a rigid body in space

So lets list all the values that are needed to completely describe and calculate a rigid body's movement in space. I assume here that the center of the body frame (the coordinate $(0,0,0)$) is also the center of mass of the body.

- constant: the total mass m of the rigid body
- constant: the moment of inertia tensor $\mathbf{I}$ in the frame of the body
- the position $\mathbf{p}$ of the body's center of mass in the world frame
- the orientation of the body in reference to the world frame, this can be described either through a rotation quaternion q or a 3×3 transformation matrix $\mathbf{A}$. These together with the position $\mathbf{p}$ can be used to transform a point in the body's frame to the world frame.
- the speed vector $\mathbf{v}$ describing the movement of the body's center of mass in the world frame
- the angular velocity vector $\boldsymbol{\omega}^{\mathbf{W}}$ (or simply $\boldsymbol{\omega}$ in the body frame) describing the axis and rate of rotation the body currently has
- the force $\mathbf{F}$ affecting the path of the center of mass
- the torque $\boldsymbol{\tau}^{\mathbf{W}}$ (or simply $\boldsymbol{\tau}$ in the body frame) affecting the rotation of the body

To map a point $\mathbf{x}$ from the body frame into the world frame or reverse:

$$\begin{aligned} \mathbf{x}^{\mathbf{W}} &= \mathbf{p} + q \diamond \mathbf{x} \diamond \overline{q} \\ \mathbf{x} &= \overline{q} \diamond \left(\mathbf{x}^{\mathbf{W}} - \mathbf{p}\right) \diamond q \\ & \quad \text{or alternatively} \\ \mathbf{x}^{\mathbf{W}} &= \mathbf{p} + \mathbf{A}\mathbf{x} \\ \mathbf{x} &= \mathbf{A}^{-1}\left(\mathbf{x}^{\mathbf{W}} - \mathbf{p}\right) \end{aligned}$$

This same formula can be applied for mapping the angular velocity or torque between the reference frames.

To find the world frame speed $\mathbf{v}_{\mathbf{x}}^{\mathbf{W}}$ of a body frame point $\mathbf{x}$ we need to find the tangential velocity of the point due to the angular velocity $\boldsymbol{\omega}$ and add that to the body speed vector $\mathbf{v}$:

$$\begin{aligned} \mathbf{v}_{\mathbf{x}}^{\mathbf{W}} &= \mathbf{v} + \boldsymbol{\omega}^{\mathbf{W}} \times \left(\mathbf{x}^{\mathbf{W}} - \mathbf{p}\right) \\ \mathbf{v}_{\mathbf{x}}^{\mathbf{W}} &= \mathbf{v} + \boldsymbol{\omega}^{\mathbf{W}} \times \left(q \diamond \mathbf{x} \diamond \overline{q}\right) \\ \mathbf{v}_{\mathbf{x}}^{\mathbf{W}} &= \mathbf{v} + q \diamond \left(\boldsymbol{\omega} \times \mathbf{x}\right) \diamond \overline{q} \end{aligned}$$

Now that we have described all the properties of a rigid body we can start to simulate how it moves in space. The idea behind numeric integration is that we estimate the values of the speed, position, etc. after a fixed time-step in relation to the current speed, position, etc. Then we use those new values for a new time-step and so on. The motion of a rigid body in space can be divided into two distinct parts: linear movement - based on movement and forces on the center of mass, and angular movement - based on the angular velocity and torque causing motion of the body around its center of mass.

5 Linear movement of a rigid body

The derivatives of the position and speed of the rigid body are following:

$$\begin{aligned}\dot{\mathbf{p}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{a} \\ \mathbf{a} &= \frac{\mathbf{F}}{m}\end{aligned}$$

This forms a second-order ordinary differential equation. Lets use the Runge-Kutta-Nyström [4] method to evaluate it. This is a modification of the fourth-order Runge-Kutta method (RK4) for a second-order ordinary differential equation. Let the position and velocity at the start of the time-step be $\mathbf{p_0}$, $\mathbf{v_0}$ and at the end of the time-step $\mathbf{p_1}$, $\mathbf{v_1}$ and let the force function be defined as $\mathbf{F}(\mathbf{p},\mathbf{v},t)$. The equations for updating the position and speed come:

$$\begin{aligned}\mathbf{p_1} &= \mathbf{p_0} + h\mathbf{v_0} + \frac{h^2}{6}(\mathbf{k_1} + \mathbf{k_2} + \mathbf{k_3}) \\ \mathbf{v_1} &= \mathbf{v}_0 + \frac{h}{6}(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4})\end{aligned}$$

The coefficients come out like this:

$$\begin{aligned}\mathbf{k_1} &= \frac{1}{m}\mathbf{F}(\mathbf{p_0}, \mathbf{v_0}, t) \\ \mathbf{k_2} &= \frac{1}{m}\mathbf{F}(\mathbf{p_0} + \frac{h}{2}\mathbf{v_0} + \frac{h^2}{8}\mathbf{k_1},\ v_0 + \frac{h}{2}\mathbf{k_1}, t + \frac{h}{2}) \\ \mathbf{k_3} &= \frac{1}{m}\mathbf{F}(\mathbf{p_0} + \frac{h}{2}\mathbf{v_0} + \frac{h^2}{8}\mathbf{k_1},\ v_0 + \frac{h}{2}\mathbf{k_2}, t + \frac{h}{2}) \\ \mathbf{k_4} &= \frac{1}{m}\mathbf{F}(\mathbf{p_0} + h\mathbf{v_0} + \frac{h^2}{2}\mathbf{k_3},\ \mathbf{v_0} + h\mathbf{k_3}, t + h)\end{aligned}$$

The force function depends on the position, speed and time. For example a particle moving in an electromagnetic field might encounter a force that depends on the position (to get the field value) and also the speed (Lorentz's $\mathbf{F} = q(\mathbf{E}(\mathbf{p}) + \mathbf{v} \times \mathbf{B}(\mathbf{p}))$.

In case the force is constant during a time-step the equations reduce to the familiar direct integration:

$$\begin{aligned}\mathbf{a} &= \frac{\mathbf{F}}{m} \\ \mathbf{p_1} &= \mathbf{p_0} + \mathbf{v_0}h + \frac{\mathbf{a}h^2}{2} \\ \mathbf{v_1} &= \mathbf{v_0} + \mathbf{a}h\end{aligned}$$

6 Angular movement of a rigid body

In the frame of the rotating rigid body the Euler's equations specify the relations between the body's moment of inertia tensor $\mathbf{I}$, angular velocity $\boldsymbol{\omega}$ and torque $\boldsymbol{\tau}$ as following:

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) = \boldsymbol{\tau}$$

The moment of inertia tensor $\mathbf{I}$ is taken to be constant as the frame of reference moves with the rotating body. In case there are no external forces applying to the rotating body the torque $\boldsymbol{\tau}$ would be 0.

From here we get the angular velocity time derivative:

$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}))$$

Let the quaternion q represent the body's rotation in reference to the world frame (transforms points from the body frame into the world frame). As time passes the rotation changes due to the angular velocity $\boldsymbol{\omega}^{\mathbf{W}}$. The W marks that the vector is in the world frame of reference. The time derivative of the rotation quaternion q is:

$$\dot{q} = \frac{1}{2}\boldsymbol{\omega}^{\mathbf{W}} \diamond q$$

For clarifying the algebra: to multiply a vector $\mathbf{v}$ with a quaternion $q = (a, \mathbf{p})$ we do: $(0, \mathbf{v}) \diamond (a, \mathbf{p})$ where $\diamond$ is the Grassmann product. And when vector is again needed the vector component is extracted from the quaternion.

The angular velocity in the world frame $\boldsymbol{\omega}^{\mathbf{W}}$ can be replaced with the one in the body frame.

$$\begin{aligned} \dot{q} &= \frac{1}{2} q \diamond \boldsymbol{\omega} \diamond \overline{q} \diamond q \\ &= \frac{1}{2} q \diamond \boldsymbol{\omega} \end{aligned}$$

This allows us to define the angular velocity through the quaternion and its derivative:

$$\begin{aligned} \boldsymbol{\omega} &= 2\overline{q} \diamond \dot{q} \\ \dot{\boldsymbol{\omega}} &= \mathbf{I}^{-1}(\boldsymbol{\tau} - (2\overline{q} \diamond \dot{q}) \times (\mathbf{I}(2\overline{q} \diamond \dot{q}))) \\ &= \mathbf{I}^{-1}(\boldsymbol{\tau} - 4(\overline{q} \diamond \dot{q}) \times (\mathbf{I}(\overline{q} \diamond \dot{q}))) \end{aligned}$$

Taking a second derivative of q and replacing the angular velocities we get:

$$\begin{aligned} \ddot{q} &= \frac{1}{2}(\dot{q} \diamond \boldsymbol{\omega} + q \diamond \dot{\boldsymbol{\omega}}) \\ &= \dot{q} \diamond \overline{q} \diamond \dot{q} + \frac{1}{2} q \diamond \left[\mathbf{I}^{-1}(\boldsymbol{\tau} - 4(\overline{q} \diamond \dot{q}) \times (\mathbf{I}(\overline{q} \diamond \dot{q})))\right] \end{aligned}$$

So as the result of our labor we get a second-order ordinary differential equation:

$$\ddot{q}(q,\dot{q},t) = \dot{q} \diamond \overline{q} \diamond \dot{q} + \frac{1}{2} q \diamond \left[\mathbf{I}^{-1} \left(\boldsymbol{\tau}(t,q,\boldsymbol{\omega}) - 4 \left(\overline{q} \diamond \dot{q} \right) \times \left(\mathbf{I} \left(\overline{q} \diamond \dot{q} \right) \right) \right) \right]$$

As we can see the second derivative of the quaternion depends on time, quaternion itself, first derivative of the quaternion and torque applied on the body. Torque's arguments are time, the quaternion and angular velocity which can be calculated from the available arguments as $\boldsymbol{\omega} = 2\overline{q} \diamond \dot{q}$.

Lets use the Runge-Kutta-Nyström [4] method to numerically integrate the values. Let q_0 and $\boldsymbol{\omega}_0$ be the rotation quaternion and angular velocity at the start of a time-step and q_1 and $\boldsymbol{\omega}_1$ be the rotation quaternion and angular velocity at the end of the time-step. And let the time at the start of the time-step be t and at the end of the time-step $t+h$ where h is the length of the time-step:

$$\begin{aligned}
\dot{q}_0 &= \frac{1}{2} q_0 \diamond \boldsymbol{\omega}_0 \\
q_1 &= q_0 + h\dot{q}_0 + \frac{h^2}{6}(k_1 + k_2 + k_3) \\
\dot{q}_1 &= \dot{q}_0 + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
\boldsymbol{\omega}_1 &= 2\overline{q_1} \diamond \dot{q}_1 \\
k_1 &= \ddot{q}(q_0, \dot{q}_0, t) \\
k_2 &= \ddot{q}(q_0 + \frac{h}{2}\dot{q}_0 + \frac{h^2}{8}k_1, \dot{q}_0 + \frac{h}{2}k_1, t + \frac{h}{2}) \\
k_3 &= \ddot{q}(q_0 + \frac{h}{2}\dot{q}_0 + \frac{h^2}{8}k_1, \dot{q}_0 + \frac{h}{2}k_2, t + \frac{h}{2}) \\
k_4 &= \ddot{q}(q_0 + h\dot{q}_0 + \frac{h^2}{2}k_3, \dot{q}_0 + hk_3, t + h)
\end{aligned}$$

From time to time we should normalize the quaternion as it might drift from unit length. How often depends on our time-step size, the specific angular velocity and the accuracy we are after. Doing this every 1000 iterations might be good enough, so the frequency is quite low. The sign operator $sgn()$ is used to do this:

$$sgn(q) = \frac{q}{|q|}$$

7 Optimizations

There are various optimizations that can be applied to these algorithms to make them computationally more efficient or accurate.

7.1 Principal moments of inertia

Another thing we can do is optimize the calculations with the moment of inertia tensor **I**. Since the moment of inertia tensor is symmetric it is possible to find a coordinate system in which it is diagonal, having the form

$$\mathbf{I} = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix}$$

where I_1, I_2, I_3 are the principal moments of inertia and the coordinate axes corresponding to this form are called the principal axes.

Naturally multiplying a diagonal matrix with a vector takes less calculations. To find this form and the new set of coordinate axes to act as the body's local frame, we have to find the eigenvalues and eigenvectors of the generic moment of inertia tensor matrix. The eigenvalues are the principal moments of inertia and the eigenvectors form the new coordinate system.

Let the generic moment of inertia tensor be

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

As the tensor is symmetric we can state that $I_{xy} = I_{yz}$, $I_{xz} = I_{zx}$, $I_{yz} = I_{zy}$.

Let the principal moments of inertia (the eigenvalues) be I_1, I_2, I_3 and let the new coordinate system axes (the eigenvectors) be $\mathbf{n_1}$, $\mathbf{n_2}$, $\mathbf{n_3}$. We can draw the following relations:

$$\begin{cases} (\mathbf{I} - I_1\mathbf{1})\mathbf{n_1} &= 0 \\ (\mathbf{I} - I_2\mathbf{1})\mathbf{n_2} &= 0 \\ (\mathbf{I} - I_3\mathbf{1})\mathbf{n_3} &= 0 \\ |\mathbf{n_1}| &= 1 \\ |\mathbf{n_2}| &= 1 \\ |\mathbf{n_3}| &= 1 \\ \mathbf{n_1} \times \mathbf{n_2} &= \mathbf{n_3} \\ \mathbf{n_2} \times \mathbf{n_3} &= \mathbf{n_1} \\ \mathbf{n_3} \times \mathbf{n_1} &= \mathbf{n_2} \end{cases}$$

Here **1** is the identity matrix.

Finding a solution that matches these conditions is a nontrivial problem. One of the most common algorithms is to use the Jacobi iteration. There is also public domain code available on the Internet that does most of this for us, for example the "Java Matrix library JAMA" contains routines that do this.

But it's also possible using a direct approach. We can find the eigenvalues themselves by solving this determinant:

$$
\begin{aligned}
|\mathbf{I}-\lambda\mathbf{1}| &= 0 \\
\begin{vmatrix} I_{xx}-\lambda & I_{xy} & I_{xz} \\ I_{xy} & I_{yy}-\lambda & I_{yz} \\ I_{xz} & I_{yz} & I_{zz}-\lambda \end{vmatrix} &= 0
\end{aligned}
$$

As a result we get a cubic equation with the following coefficients:

$$
\begin{aligned}
a\lambda^3+b\lambda^2+c\lambda+d &= 0 \\
a &= -1 \\
b &= I_{xx}+I_{yy}+I_{zz} \\
c &= I_{xy}^2+I_{xz}^2+I_{yz}^2-I_{xx}I_{yy}-I_{xx}I_{zz}-I_{yy}I_{zz} \\
d &= I_{xx}I_{yy}I_{zz}+2I_{xy}I_{xz}I_{yz}-I_{xx}I_{yz}^2-I_{yy}I_{xz}^2-I_{zz}I_{xy}^2
\end{aligned}
$$

Solving this cubic equation we get three real values that are the principal moments of inertia $I_1,\ I_2,\ I_3$. Now to get the eigenvectors we have to solve three different 3-unknowns homogeneous linear systems. This can be done using iterative methods like the Gaussian elimination. For example for I_1:

$$
\begin{aligned}
(\mathbf{I}-I_1\mathbf{1})\mathbf{n_1} &= 0 \\
\begin{bmatrix} I_{xx}-I_1 & I_{xy} & I_{xz} \\ I_{xy} & I_{yy}-I_1 & I_{yz} \\ I_{xz} & I_{yz} & I_{zz}-I_1 \end{bmatrix} \mathbf{n_1} &= 0
\end{aligned}
$$

The solution vector we get should be normalized. Also in case the body is symmetric on some axis the principal moments of inertia will match. In case all three match it's a uniform distribution and any orthonormal vector system will do (no need for any optimization). In case two of them match we only need to find the eigenvector on the third value and from that can derive the other two vectors.

Once we have the eigenvectors and eigenvalues we have to make sure that the principal axes are sane. For that we need to swap the moment and axis value pairs around (or negate one of the normal vectors) to make sure the following cross products stand:

$$
\begin{aligned}
\mathbf{n_1}\times\mathbf{n_2} &= \mathbf{n_3} \\
\mathbf{n_2}\times\mathbf{n_3} &= \mathbf{n_1} \\
\mathbf{n_3}\times\mathbf{n_1} &= \mathbf{n_2}
\end{aligned}
$$

Or equivalently that this determinant equals $+1$:

$$
\begin{vmatrix} n_{1_x} & n_{2_x} & n_{3_x} \\ n_{1_y} & n_{2_y} & n_{3_y} \\ n_{1_z} & n_{2_z} & n_{3_z} \end{vmatrix} = 1
$$

We have a new coordinate system and to complete the optimization we just have to convert the body from the old frame to the new frame:

$$\begin{aligned}
\mathbf{B} &= \begin{bmatrix} n_{1_x} & n_{2_x} & n_{3_x} \\ n_{1_y} & n_{2_y} & n_{3_y} \\ n_{1_z} & n_{2_z} & n_{3_z} \end{bmatrix} \\
\mathbf{x}' &= \mathbf{B}^T\mathbf{x} \\
\boldsymbol{\omega}' &= \mathbf{B}^T\boldsymbol{\omega} \\
\boldsymbol{\tau}' &= \mathbf{B}^T\boldsymbol{\tau} \\
q' &= matrix2quatern(\mathbf{B}) \diamond q \\
\mathbf{A}' &= \mathbf{BA}
\end{aligned}$$

In case we have a body frame rotation-translation pair $q_L/\mathbf{A_L}$, $\mathbf{c}$ then they are translated to the new coordinate system:

$$\begin{aligned}
\mathbf{c}' &= \mathbf{B}^T c \\
q_L' &= matrix2quatern(\mathbf{B}) \diamond q_L \\
\mathbf{A_L}' &= \mathbf{B}^T\mathbf{A_L}
\end{aligned}$$

7.2 Angular velocity normalization

To get increased accuracy in case of torque free rotation we can do angular velocity normalization. In case there are no external forces acting on the body (torque $\boldsymbol{\tau}$ is 0) the angular momentum $\mathbf{L}$ is conserved, that means the angular momentum $\mathbf{L^W}$ is constant in the world frame. Angular momentum is defined as $\mathbf{L} = \mathbf{I}\boldsymbol{\omega}$ in the body frame. We can transform it into the world frame:

$$\mathbf{L^W} = q \diamond (\mathbf{I}\boldsymbol{\omega}) \diamond \overline{q}$$

During the torque free rotation of the body the $\mathbf{L^W}$ must be constant - so we can use it to normalize the angular velocity vector:

$$\boldsymbol{\omega} = \mathbf{I}^{-1}\left(\overline{q} \diamond \mathbf{L^W} \diamond q\right)$$

7.3 Mid-step and sub-step force and torque interpolation

At numeric integration the force and torque values are requested at the start, in the middle and at the end of a step. Also when the implementation is using sub-steps to increase accuracy many more force and torque evaluations are requested. In a simpler application those values might only be updated at main step intervals. That means the force and torque values are correct only at the start of a step. Often the force and torque values change smoothly across steps. To increase accuracy it might make sense to extrapolate the force and torque values - as we know where the values were we can predict where they are going.

The easiest way to extrapolate the values would be to use the cubic Lagrange polynomial with 3 steps of history (2 former steps and the current active).

Let the values provided from the past time-steps to the current one be $a \xrightarrow{h} b \xrightarrow{h} c$ where h is the fixed generic time-step. Let the time at the start of the current step be t_0. In this case the extrapolated value d_t at time t is:

$$\begin{aligned} x &= \frac{t_s - t}{h} \\ d_t &= \frac{1}{2}a(x+1)x - b(x+2)x + \frac{1}{2}c(x+2)(x+1) \end{aligned}$$

This equation extrapolates a scalar value. In case of a vector we can extrapolate the components individually. One needs to take care in regards to previous values at extrapolation. In case there are sudden changes (say a collision) the sharp change in values will cause the extrapolation to go haywire. In that case the past values should be reset.

7.4 Runge-Kutta sub-steps

To get precise results we need to use a smaller time-step or more sub-steps. Then again using too many sub-steps will result in unnecessary calculations. So how many sub-steps should we do? We need to do as many sub-steps as needed to keep the buildup of errors down. But how to do the error estimate? Sometimes one just has to do sub-steps to see whether the result differs significantly. At other times we might guess the need for sub-steps - for example when the angular velocity is very high but the time-steps are large the quaternion representing object rotation will build up errors much more quickly.

7.5 Combining linear and angular movement equations

The equations given for linear and angular movement before are split up. This causes the problem that at torque evaluation we don't know the current position of the body and at force evaluation we don't know the current rotation estimate. There is probably a way to merge those two to bring in estimates of current position and rotation to the other method.

7.6 Discarding conversation of angular momentum

Many 3D game physics engines discard the conversation of angular momentum component from the Euler's equations and are left with:

$$\ddot{q}(q,\dot{q},t) = \dot{q} \diamond \overline{q} \diamond \dot{q} + \frac{1}{2} q \diamond \left[\mathbf{I}^{-1} \boldsymbol{\tau}(t,q,\boldsymbol{\omega}) \right]$$

Doing this will remove the wobble from the torque-free precession. Of course games are more interested in believability than accuracy. Come to think about it - for a lay

person the way torque-induced precession (gyroscopic precession) causes objects to move would seem quite non-intuitive and odd - reducing believability of a game.

7.7 Quaternion drift

A common problem at angular movement integration is inherent quaternion drift in the algorithms. The rotation quaternion is a unit quaternion meaning its length is 1:

$$\begin{aligned} q &= (t,x,y,z) \\ |q| &= \sqrt{t^2+x^2+y^2+z^2} \\ &= 1 \end{aligned}$$

During integration steps the quaternion will drift away from this unit length.

A quaternion can be looked as a point on a 4D sphere (with radius 1). And its derivative $\dot{q}$ is basically the tangential speed of the point on the sphere. Naively integrating it, as in many algorithms ($q = q + h\dot{q}$), the point will move away from the sphere, not being constrained to its surface. And that even when using more sophisticated algorithms like the RK4. Also integrating like that will need constant and costly quaternion normalization at each step.

This drift is addressed by a patent from NASA[5]. This problem is also well described in another paper from Martin Kleppmann[6]. What NASA did was to find a way to correctly move a point across the surface of the 4D sphere. But this does not involve directly factoring in the other kinetic issues like conservation of momentum, torque, etc. Those have to be assembled, composed into a single angular velocity equation, and then based on that velocity the point on the sphere is rotated in 4D space to its next position.

But how does our algorithm compare to that? We are using a second derivative of the rotation quaternion $\ddot{q}$, and as it is a second derivative it means it is the change in the tangential speed on the surface - a change that indicates it must follow the surface of the sphere. Also it factors in the change needed to maintain angular momentum.

What this basically means is I believe what we see here is pretty much as good as what the NASA is doing, if not better[2].

7.8 Direct integration of the angular movement

In case the torque is constant it might be possible to directly integrate the second derivative of the $\ddot{q}$ resulting in an analytic equation to directly and accurately calculate the rotation and angular velocity at any time t (and also get rid of the quaternion drift). I have no idea how to do that.

[2] I believe now that this claim is slightly exaggerated and misleading. While the NASA patent offers better integration than a simple Euler method, higher order methods like the RK4 we used here offer comparable accuracy. Also the NASA method is probably more stable when calculating the position over eons of time.

8 Disclaimer

The author of this document is not a physicist nor a mathematician and so is an amateur. The information in this document is provided in the hope it will be useful, but the author takes no responsibility for any errors or problems you may encounter.

References

[1] Quaternions algebra, http://en.wikipedia.org/wiki/Quaternion

[2] Euler's equations of motion, http://en.wikipedia.org/wiki/Euler's_equations

[3] Runge-Kutta integration, http://en.wikipedia.org/wiki/Runge-Kutta

[4] "The Data Analysis BriefBook", R.K.BOCK, W.KRISCHER, http://rkb.home.cern.ch/rkb/titleA.html

[5] Closed-Form Integrator for the Quaternion (Euler Angle) Kinematics Equations (US Patent Number 6,061,611) http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080004113_2008004226.pdf

[6] Simulation of colliding constrained rigid bodies, Martin Kleppmann, University of Cambridge Computer Laboratory http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-683.pdf