# How to make new joints in ODE

February 24, 2002

## Contents

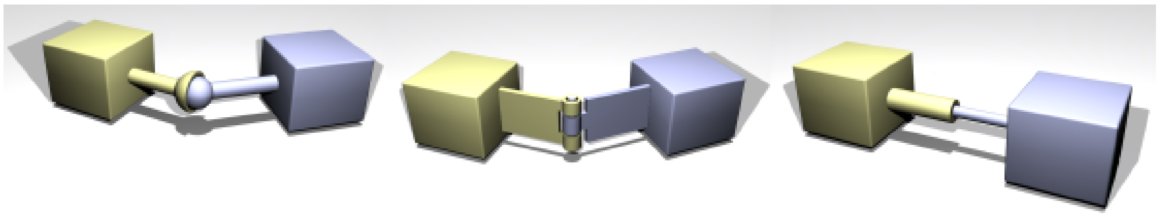

Figure 1: Three different constraint types: A ball joint, a hinge and a slider.

## 1 Introduction

The Open Dynamics Engine (ODE) is a free, industrial quality library for simulating articulated rigid body dynamics. For example, it is good for simulating ground vehicles, legged creatures, and moving objects in VR environments. It is fast, flexible and robust, and it has built-in collision detection. ODE is being developed by Russell Smith, and the web site is `http://www.q12.org/ode/`.

This is a short implementors guide for creating new joint constraints in ODE.

### 1.1 What is a joint?

In real life a joint is something like a hinge, that is used to connect two objects. In ODE a joint is very similar: It is a relationship that is enforced between two bodies so that they can only have certain positions and orientations relative to each other. This relationship is called a constraint — the words joint and constraint are often used interchangeably. Figure 1 shows three different constraint types.

Each time the integrator takes a step all the joints apply constraint forces to the bodies they affect. These forces are automatically calculated so that the body motions will preserve all the joint relationships.

Each joint has a number of parameters controlling its geometry. An example is the position of the ball in a ball-and-socket joint.

## 2 Mathematical preliminaries

The following notation will be used when deriving the example joint. It is assumed that the reader has at least a basic understanding of matrix algebra.

### 2.1 Notation

- A $3 \times 1$ vector $a'$ is deemed to be relative to the frame of reference of some body. The corresponding vector $a$ is in the global (or absolute) frame of reference. $a$ is a rotated version of $a'$, i.e.

$$a \quad = \quad R\,a' \tag{1}$$

  where $R$ is a $3 \times 3$ orthonormal (“rotation”) matrix.

- The $n \times n$ identity matrix is called $1_n$.
- The $n \times 1$ vector with each element set to infinity is $\infty_n$.

### 2.2 Cross product algebra

If $a$ and $b$ are $3 \times 1$ vectors then we define

$$a \times b \;\triangleq\; \hat{a}\, b \tag{2}$$

where

$$\hat{a} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{3}$$

$(a_1, a_2, a_3$ are the elements of $a)$. Some identities are:

$$\begin{aligned}
\hat{a} &= -\hat{a}^T && (4)\\
\hat{a}a &= 0 && (5)\\
\hat{a}b &= -\hat{b}a && (6)\\
\hat{a}\hat{b} &= (\hat{b}\hat{a})^T && (7)\\
\hat{a}\hat{b}\hat{b}a &= -\hat{b}\hat{a}\hat{a}b && (8)\\
\widehat{Ab} &= A\hat{b}A^T \qquad \text{if } A^TA = 1_3 && (9)\\
\text{thus} \qquad Ab \times Ac &= A(b \times c) \qquad \text{if } A^TA = 1_3 && (10)
\end{aligned}$$

## 3 The body state variables

Joints in ODE connect rigid bodies together. Each body has a number of state variables: position, orientation, linear and angular velocity. Body parameters that usually remain fixed during the simulation, such as mass properties and geometrical shape, are not regarded as state variables.

### 3.1 Position

A body's $3 \times 1$ position vector contains the position of the body's center of mass:

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{11}$$

### 3.2 Orientation

A body's orientation is represented in ODE as either a $3 \times 3$ rotation matrix $R$, or as a $4 \times 1$ quaternion vector $q$. $R$ has the property that for a point $a'$ relative to the body (i.e. in the body frame) the corresponding global position $a$ is:

$$a = R\,a' + p \tag{12}$$

Note that $R$ is an orthonormal matrix, so that

$$R^{-1} = R^T \tag{13}$$

Let us look at how $R$ transforms the global x axis vector (called $u_x$):

$$R\,u_x = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \end{bmatrix} \tag{14}$$

We can see that the first column of $R$ is the transformed x axis (the second and third columns are the transformed y and z axes). These transformed axes are called $u_1$, $u_2$, and $u_3$, so that

$$R = \begin{bmatrix} | & | & | \\ u_1 & u_2 & u_3 \\ | & | & | \end{bmatrix} \tag{15}$$

The $u_i$ vectors can be considered to be the body's own local x,y and z axes, that are fixed to the body and move relative to it (but that are expressed in global coordinates).

## 3.3 Linear velocity

A body's $3 \times 1$ linear velocity vector is

$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \mathrm{d}p_x/\mathrm{d}t \\ \mathrm{d}p_y/\mathrm{d}t \\ \mathrm{d}p_z/\mathrm{d}t \end{bmatrix} \tag{16}$$

This is the time derivative of the position vector.

## 3.4 Angular velocity

A body's $3 \times 1$ angular velocity vector is

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{17}$$

If $a$ is a global-frame vector giving the offset of a body point relative to the center of mass, then the velocity of that point (in the global frame, relative to the center of mass) is:

$$\mathrm{d}a/\mathrm{d}t = \dot{a} = \omega \times a \tag{18}$$

If $a'$ is a body-relative vector giving the offset of a point in the body, then the global velocity of that point is

$$\mathrm{d}a/\mathrm{d}t = \dot{a} = \omega \times (R\,a') + v \tag{19}$$

The derivative of $R$ is often useful in deriving constraint equations:

$$\begin{aligned}
\dot{R} &= \begin{bmatrix} \dot{u}_1 & \dot{u}_2 & \dot{u}_3 \end{bmatrix} && (20)\\
&= \begin{bmatrix} \omega \times u_1 & \omega \times u_2 & \omega \times u_3 \end{bmatrix} && (21)\\
&= \begin{bmatrix} \widehat{\omega} u_1 & \widehat{\omega} u_2 & \widehat{\omega} u_3 \end{bmatrix} && (22)\\
&= \widehat{\omega} R && (23)
\end{aligned}$$

### 3.5 Note

State vectors in ODE are expressed in global (not body-local) coordinates.

## 4 The constraint equation

Every joint has three associated equations that looks like this:

$$\begin{aligned}
J_1 v_1 + \Omega_1 \omega_1 + J_2 v_2 + \Omega_2 \omega_2 &= c + C\lambda && (24)\\
\lambda &\geq \ell && (25)\\
\lambda &\leq h && (26)
\end{aligned}$$

The quantities on the left and right hand sides are vectors of size $m \times 1$ (we say this constraint has $m$ rows). $J$ and $\Omega$ are $m \times 3$ Jacobian matrices. The linear and angular velocity vectors for the first body are $v_1$ and $\omega_1$. Similarly $v_2$ and $\omega_2$ correspond to the second body. $c$ is an $m \times 1$ "right hand side vector".

$\lambda$ is an $m \times 1$ constraint force that is applied to the bodies to ensure that the constraint equation (equation 24) is satisfied. $\lambda$ is automatically calculated by ODE. Equation 25 and equation 26 indicate that the elements of $\lambda$ can be restricted to be within a lower ($\ell$) and upper ($h$) bound.

$C$ is a diagonal $m \times m$ matrix called the "constraint force mixing" (CFM) matrix. It allows the constraint force $\lambda$ to be part of the constraint equation. $C$ can be manipulated to get certain interesting effects, described below. For many "normal" constraints, $C$ is set to 0.

## 5 Example 1: A ball joint

Let us derive the constraint equation for the ball joint shown in Figure 2.

### 5.1 Step 1: Position invariant

First, a kinematic constraint equation is written in terms of the position and orientation of the two bodies. This is the equation that must always be true while the joint is connecting the two bodies.

For the ball joint, if we define $a'_1$ (and $a'_2$) to be the "attachment" vectors from $p_1$ (and $p_2$) to the center of the ball, relative to bodies 1 (and 2), then the constraint is:

$$p_1 + R_1 a'_1 = p_2 + R_2 a'_2 \qquad (27)$$

because the ball of body 2 must remain in the socket of body 1.

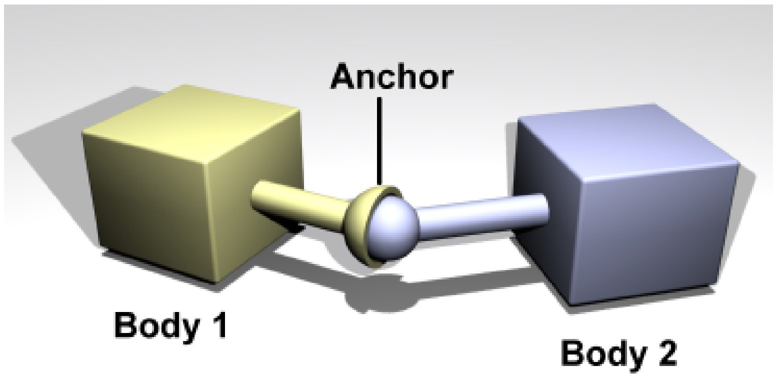

Figure 2: A ball joint.

## 5.2 Step 2: Velocity invariant

Take the time derivative of equation 27, so that the constraint is expressed in terms of the body velocities:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\Big[p_1 + R_1 a_1'\Big] &= \frac{\mathrm{d}}{\mathrm{d}t}\Big[p_2 + R_2 a_2'\Big] && (28)\\
v_1 + \dot{R}_1 a_1' &= v_2 + \dot{R}_2 a_2' && (29)\\
v_1 + \widehat{\omega_1} R_1 a_1' &= v_2 + \widehat{\omega_2} R_2 a_2' && (30)\\
v_1 + \widehat{\omega_1} a_1 &= v_2 + \widehat{\omega_2} a_2 && (31)\\
v_1 - \widehat{a_1}\omega_1 - v_2 + \widehat{a_2}\omega_2 &= 0 && (32)
\end{aligned}$$

Thus

$$\begin{aligned}
J_1 &= 1_3 && (33)\\
\Omega_1 &= -\widehat{a_1} && (34)\\
J_2 &= -1_3 && (35)\\
\Omega_2 &= \widehat{a_2} && (36)\\
c &= 0 && (37)
\end{aligned}$$

## 5.3 Step 3: Choose $\ell$, $h$ and CFM

There will be no limits on $\lambda$, and no CFM so we also have

$$\begin{aligned}
\ell &= -\infty_3 && (38)\\
h &= \infty_3 && (39)\\
C &= 0 && (40)
\end{aligned}$$

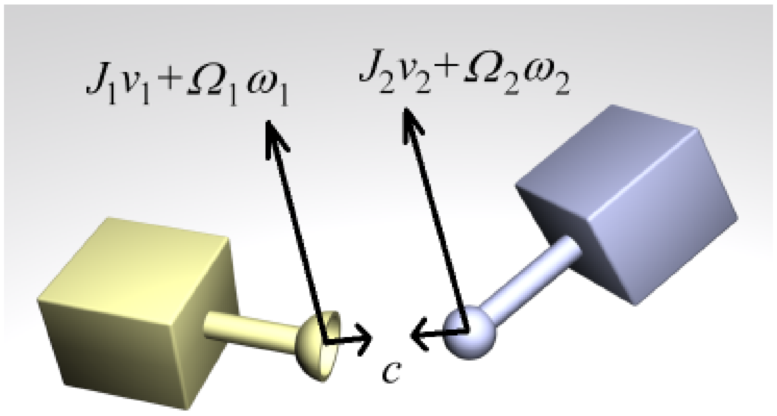

Figure 3: The meaning of the constraint equation for a ball joint.

## 5.4 Step 4: Error reduction

It seems that we have found all the values required for this constraint. However, if you actually implement the ball joint this way you will discover a problem. As the simulation proceeds the joint will gradually "come apart", i.e. the position of the ball and socket with respect to the first and second bodies will gradually diverge. This is referred to as joint error.

But shouldn't equation 27 explicitly prevent this from happening? No: the enforced constraint is expressed in velocities, not positions. This would not matter if we were able to simulate with perfect accuracy, but in fact all simulators have various numerical errors. The result is that the body positions and orientations do not evolve over time exactly as they should.

Joint error can also occur if the user sets the position/orientation of one body without correctly setting the position/orientation of the other body.

The solution to this problem is to use a nonzero $c$. The meanings of $c$ and the other terms in the constraint equation 24 are illustrated in figure 3. $J_1 v_1 + \Omega_1 \omega_1$ is the velocity of the socket as measured from body 1. Similarly, $J_2 v_2 + \Omega_2 \omega_2$ is the velocity of the ball as measured from body 2. Both of these velocities are in global coordinates.

If $c$ is zero then both velocities must be equal. If $c$ nonzero then they are not equal. If we choose $c$ to be proportional to the positional error vector between the ball and the socket:

$$c \quad = \quad \epsilon \Big[ (p_1 + a_1) - (p_2 + a_2) \Big] \tag{41}$$

then the ball and socket velocities will be constrained so that they move closer to each other (if they are already together then $c = 0$). This means that any positional error introduced by the simulation will be reduced over time.

The factor $\epsilon$ controls how quickly this error reduction is performed. $\epsilon$ is computed from the global "error reduction parameter" (ERP). The meaning of ERP is:

- If ERP $= 0$ then no error reduction is performed, and $c = 0$.
- If ERP $= 1$ then $c$ should be set so that the position error is completely eliminated in one time step.

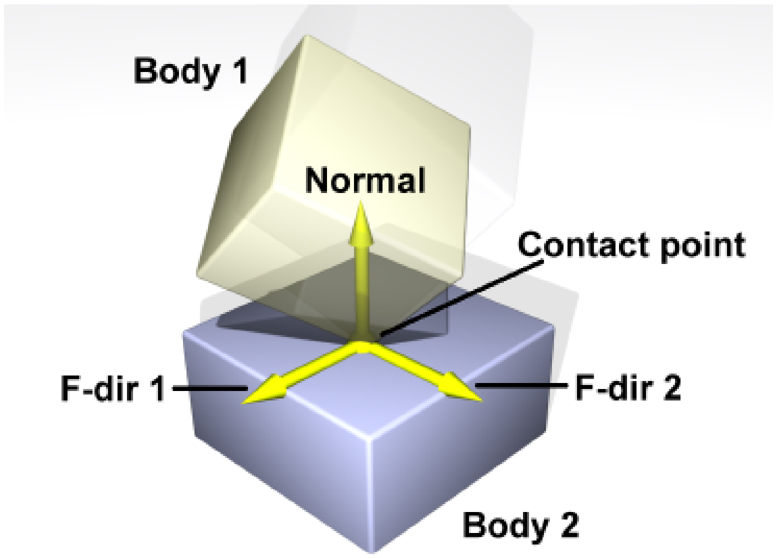

Figure 4: A contact joint.

- For $0 < \text{ERP} < 1$, the position error should be reduced by a fraction ERP in one time step.

As $c$ has the units of velocity, for a time step $s$:

$$\begin{aligned} c &= \frac{\text{distance}}{\text{time}} && (42) \\ &= \frac{\text{ERP}\left[(p_1 + R_1 a_1') - (p_2 + R_2 a_2')\right]}{s} && (43) \end{aligned}$$

so we can see that

$$\epsilon = \frac{\text{ERP}}{s} \tag{44}$$

It would seem that setting $\text{ERP} = 1$ is the best approach, since this eliminates all error in one time step. However, setting $\text{ERP} = 1$ is not recommended as it will not have the intended effect due to various internal approximations. A value of $\text{ERP} = 0.1$ to 0.8 is recommended in practice.

## 6 Example 2: A contact joint

Let us derive the constraint equation for the frictionless contact joint shown in Figure 4.

### 6.1 Step 1: Position invariant

We will skip this step and write the constraint directly in terms of velocity. This is sometimes the easiest approach, as a constraint may have a complex position form but a simple velocity form.

## 6.2 Step 2: Velocity invariant

Define $a_1$ (and $a_2$) to be the offset vectors (in global coordinates) from $p_1$ (and $p_2$) to the contact point. The contact normal vector is $n$. The constraint must state that the velocity of the contact point with respect to the two bodies must be the same along the direction of the normal:

$$\begin{aligned} n^T v_1 + n^T \widehat{\omega_1} a_1 &= n^T v_2 + n^T \widehat{\omega_2} a_2 & (45) \\ n^T v_1 - n^T \widehat{a_1} \omega_1 - n^T v_2 + n^T \widehat{a_2} \omega_2 &= 0 & (46) \end{aligned}$$

Thus $m = 1$ and

$$\begin{aligned} J_1 &= n^T & (47) \\ \Omega_1 &= -n^T \widehat{a_1} & (48) \\ J_2 &= -n^T & (49) \\ \Omega_2 &= n^T \widehat{a_2} & (50) \end{aligned}$$

## 6.3 Step 3: Choose $\ell$, $h$ and CFM

The constraint force $\lambda$ will be oriented along the normal $n$. We want this force to push the bodies apart, but not to suck them together. Without limits on $\lambda$ this contact joint would be "sticky", i.e. it would not allow the contact to be broken. Thus we set

$$\begin{aligned} \ell &= 0 & (51) \\ h &= \infty & (52) \end{aligned}$$

so that

$$0 \leq \lambda \leq \infty \qquad (53)$$

The CFM matrix $C$ is actually a $1 \times 1$ scalar in this case. Let us examine the effect of making $C$ nonzero. First note that the quantities in equation 46 are velocities along the contact normal vector. By examining equation 24 we see that if the term $C\lambda$ is nonzero it effectively adds to $c$. The result is that the contact point velocity with respect to the two bodies will not be equal—in other words, we will get some relative velocity at the contact point in the normal direction (the bodies will interpenetrate).

If the bodies are being pressed together lightly, the constraint force $\lambda$ will be small. The product $C\lambda$ will be small, and there will be a small interpenetration velocity.

If the bodies are being pressed together with great force, a large constraint force $\lambda$ will be necessary to keep then apart. The product $C\lambda$ will be large, and there will be a large interpenetration velocity.

Thus we will be able to push the bodies in to each other at the contact, with a velocity that depends on how hard we push (i.e. what force is used). This is somewhat similar to if the bodies were made of clay or some other yielding substance. The magnitude of this effect is controlled by the scale factor $C$. $C$ can be chosen by the user to get various material effects.

## 6.4 Step 4: Error reduction

As noted above, the quantities in equation 46 are velocities along the contact normal vector. We can choose $c$ based on the position error along the normal vector, in a manner similar to the ball-and-socket joint:

$$c = \frac{\text{ERP}}{s} \, n^T \Big[ (p_1 + a_1) - (p_2 + a_2) \Big] \tag{54}$$

Setting ERP $> 0$ will tend to counter the effects of setting $C$ is nonzero, as the error reduction process is essentially a restoring velocity to counter the $C\lambda$ interpenetration velocity.

By allowing the user to choose ERP and CFM independently for the contact constraint, various interesting material effects can be produced—for example “softness” or “springiness”.

## 7 How to use ERP and CFM

The following is taken from the ODE manual:

ERP and CFM can be independently set in many ODE joints. They can be set in contact joints, in joint limits and various other places, to control the sponginess and springiness of the joint (or joint limit).

If CFM is set to zero, the constraint will be hard. If CFM is set to a positive value, it will be possible to violate the constraint by “pushing on it” (for example, for contact constraints by forcing the two contacting objects together). In other words the constraint will be soft, and the softness will increase as CFM increases. What is actually happening here is that the constraint is allowed to be violated by an amount proportional to CFM times the restoring force that is needed to enforce the constraint. Note that setting CFM to a negative value can have undesirable bad effects, such as instability. Don’t do it.

By adjusting the values of ERP and CFM, you can achieve various effects. For example you can simulate springy constraints, where the two bodies oscillate as though connected by springs. Or you can simulate more spongy constraints, without the oscillation. In fact, ERP and CFM can be selected to have the same effect as any desired spring and damper constants. If you have a spring constant $k_p$ and damping constant $k_d$, then the corresponding ODE constants are:

$$\text{ERP} = \frac{s k_p}{s k_p + k_d} \tag{55}$$

$$\text{CFM} = \frac{1}{s k_p + k_d} \tag{56}$$

where $s$ is the step size. These values will give the same effect as a spring-and-damper system simulated with implicit first order integration.

Increasing CFM, especially the global CFM, can reduce the numerical errors in the simulation. If the system is near-singular, then this can markedly increase stability. In fact, if the system is misbehaving, one of the first things to try is to increase the global CFM.