Coefficient of restitution

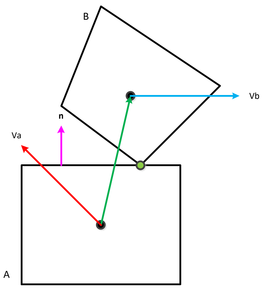
From Wikipedia, the free encyclopedia

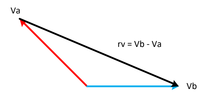
The **coefficient of restitution** (**COR**), also denoted by (**e**), is the ratio of the final to initial relative velocity between two objects after they collide. It normally ranges from 0 to 1 where 1 would be a perfectly elastic collision. A perfectly inelastic collision has a coefficient of 0, but a 0 value does not have to be perfectly inelastic. It is measured in the Leeb rebound hardness test, expressed as 1000 times the COR, but it is only a valid COR for the test, not as a universal COR for the material being tested.

The value is almost always less than one due to initial translational kinetic energy being lost to rotational kinetic energy, plastic deformation, and heat. It can be more than 1 if there is an energy gain during the collision from a chemical reaction, a reduction in rotational energy, or another internal energy decrease that contributes to the post-collision velocity.

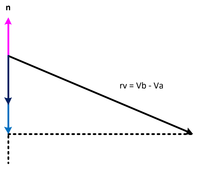
$$\text{Coefficient of restitution } (e) = \frac{|\text{Relative velocity after collision}|}{|\text{Relative velocity before collision}|}$$

A bouncing ball captured with a stroboscopic flash at 25 images per second: Ignoring air resistance, the square root of the ratio of the height of one bounce to that of the preceding bounce gives the coefficient of restitution for the ball/surface impact.





$$V'=\epsilon\,\vec{V}$$
$$(V_b'-V_a')\cdot\vec{n}=-\epsilon(V_b-V_a)\cdot n$$



Impulse (physics)

From Wikipedia, the free encyclopedia

In classical mechanics, **impulse** (symbolized by *J* or **Imp**) is the integral of a force, *F*, over the time interval, *t*, for which it acts. Since force is a vector quantity, impulse is also a vector quantity. Impulse applied to an object produces an equivalent vector change in its linear momentum, also in the resultant direction. The SI unit of impulse is the newton second (N·s), and the dimensionally equivalent unit of momentum is the kilogram meter per second (kg·m/s). The corresponding English engineering unit is the pound-second (lbf·s), and in the British Gravitational System, the unit is the slug-foot per second (slug·ft/s).

Mathematical derivation in the case of an object of constant mass [edit]

Impulse **J** produced from time t_1 to t_2 is defined to be[2]

$$\mathbf{J} = \int_{t_1}^{t_2} \mathbf{F}\,\mathrm{d}t$$

where **F** is the resultant force applied from t_1 to t_2.

From Newton's second law, force is related to momentum **p** by

$$\mathbf{F} = \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t}$$

Therefore,

$$\begin{aligned}\mathbf{J} &= \int_{t_1}^{t_2} \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t}\,\mathrm{d}t \\ &= \int_{\mathbf{p}_1}^{\mathbf{p}_2} \mathrm{d}\mathbf{p} \\ &= \mathbf{p}_2 - \mathbf{p}_1 = \Delta\mathbf{p}\end{aligned}$$

where **Δp** is the change in linear momentum from time t_1 to t_2. This is often called the impulse-momentu theorem).

As a result, an impulse may also be regarded as the change in momentum of an object to which a resulta in a simpler form when the mass is constant:

$$\mathbf{J} = \int_{t_1}^{t_2} \mathbf{F}\,\mathrm{d}t = \Delta\mathbf{p} = m\mathbf{v_2} - m\mathbf{v_1}$$

$$V'=\epsilon\,\vec{V}$$
$$(V_b'-V_a')\cdot\vec{n}=-\epsilon(V_b-V_a)\cdot n$$

$$\frac{J}{m}=\Delta V$$
$$J=j\,\vec{n}$$

$$V_a'=V_a-\frac{j\,\vec{n}}{mass_a}$$
$$V_b'=V_b+\frac{j\,\vec{n}}{mass_b}$$

$$(V_b'-V_a')\cdot\vec{n}=-\epsilon(V_b-V_a)\cdot n$$

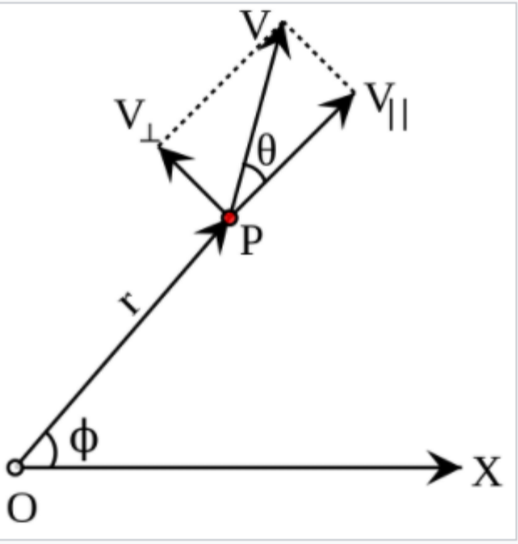
$$(V_b+\frac{j\,\vec{n}}{mass_b}-(V_a-\frac{j\,\vec{n}}{mass_a}))\cdot\vec{n}=-\epsilon(V_b-V_a)\cdot\vec{n}$$
$$(V_b+\frac{j\,\vec{n}}{mass_b}-V_a+\frac{j\,\vec{n}}{mass_a})\cdot\vec{n}+\epsilon(V_b-V_a)\cdot\vec{n}=0$$
$$(V_b-V_a+\frac{j\,\vec{n}}{mass_b}+\frac{j\,\vec{n}}{mass_a})\cdot\vec{n}+\epsilon(V_b-V_a)\cdot\vec{n}=0$$
$$(V_b-V_a)\cdot\vec{n}+(\frac{j\,\vec{n}}{mass_b}+\frac{j\,\vec{n}}{mass_a})\cdot\vec{n}+\epsilon(V_b-V_a)\cdot\vec{n}=0$$
$$\vec{n}\cdot\vec{n}=1$$
$$(V_b-V_a)\cdot\vec{n}+j(\frac{1}{mass_b}+\frac{1}{mass_a})+\epsilon(V_b-V_a)\cdot\vec{n}=0$$
$$(1+\epsilon)(V_b-V_a)\cdot\vec{n}+j(\frac{1}{mass_b}+\frac{1}{mass_a})=0$$
$$j(\frac{1}{mass_b}+\frac{1}{mass_a})=-(1+\epsilon)(V_b-V_a)\cdot\vec{n}$$

$$j=\frac{-(1+\epsilon)(V_b-V_a)\cdot\vec{n}}{(\frac{1}{mass_b}+\frac{1}{mass_a})}$$

Angular velocity

From Wikipedia, the free encyclopedia

In physics, **angular velocity** (**ω** or **Ω**), also known as **angular frequency vector**,[1] is a vector measure of rotation rate, that refers to how fast an object rotates or revolves relative to another point, i.e. how fast the angular position or orientation of an object changes with time.



The angular velocity of the particle at *P* with respect to the origin *O* is determined by the perpendicular component of the velocity vector **v**.

Particle in three dimensions [edit]

In three-dimensional space, we again have the position vector **r** of a moving particle. Here, orbital angular velocity is a pseudovector whose magnitude is the rate at which **r** sweeps out angle, and whose direction is perpendicular to the instantaneous plane in which **r** sweeps out angle (i.e. the plane spanned by **r** and **v**). However, as there are *two* directions perpendicular to any plane, an additional condition is necessary to uniquely specify the direction of the angular velocity; conventionally, the right-hand rule is used.

Let the pseudovector $\mathbf{u}$ be the unit vector perpendicular to the plane spanned by **r** and **v**, so that the right-hand rule is satisfied (i.e. the instantaneous direction of angular displacement is counter-clockwise looking from the top of $\mathbf{u}$). Taking polar coordinates (r, ϕ) in this plane, as in the two-dimensional case above, one may define the orbital angular velocity vector as:

$$\boldsymbol{\omega} = \omega\mathbf{u} = \frac{d\phi}{dt}\mathbf{u} = \frac{v\sin(\theta)}{r}\mathbf{u},$$

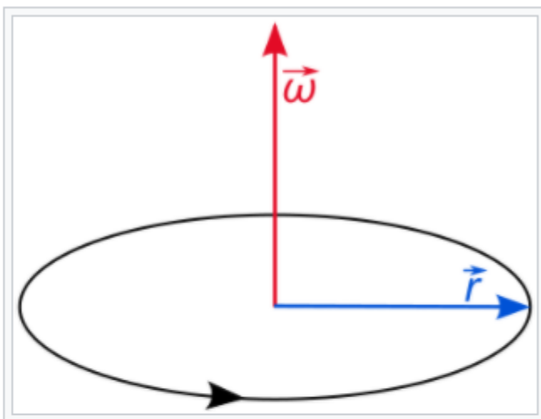
where θ is the angle between **r** and **v**. In terms of the cross product, this is:

$$\boldsymbol{\omega} = \frac{\mathbf{r} \times \mathbf{v}}{r^2}.$$

From the above equation, one can recover the tangential velocity as:

$$\mathbf{v}_\perp = \boldsymbol{\omega} \times \mathbf{r}$$

Note that the above expression for $\boldsymbol{\omega}$ is only valid if $\boldsymbol{r}$ is in the same plane as the motion.



The orbital angular velocity vector encodes the time rate of change of angular position, as well as the instantaneous plane of angular displacement. In this case (counter-clockwise circular motion) the vector points up.

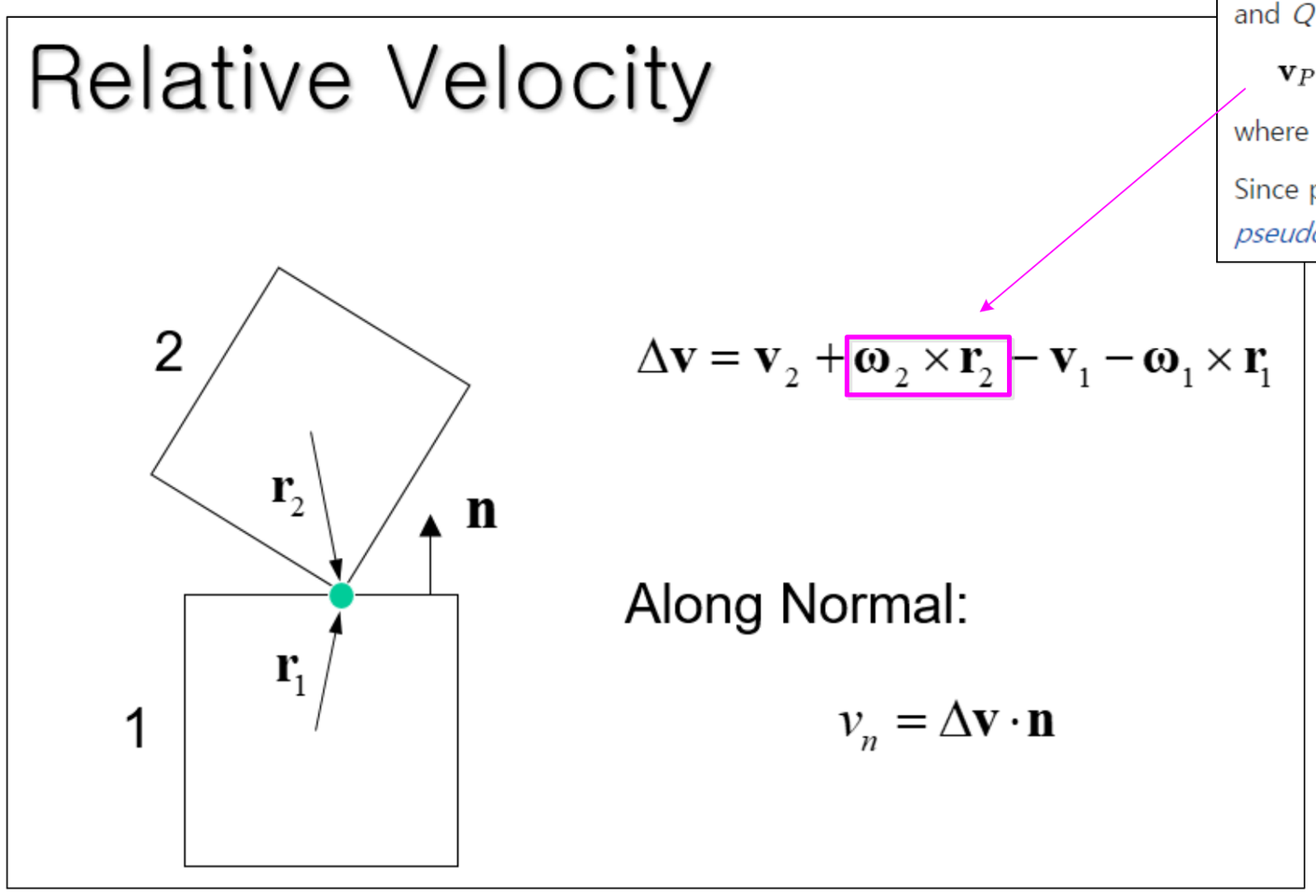
Rigid body [edit]

The cross product frequently appears in the description of rigid motions. Two points *P* and *Q* on a rigid body can be related by:

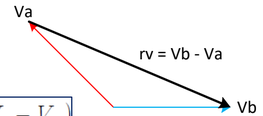
$$\mathbf{v}_P - \mathbf{v}_Q = \boldsymbol{\omega} \times (\mathbf{r}_P - \mathbf{r}_Q)$$

where **r** is the point's position, **v** is its velocity and $\boldsymbol{\omega}$ is the body's angular velocity.

Since position **r** and velocity **v** are *true* vectors, the angular velocity $\boldsymbol{\omega}$ is a *pseudovector* or *axial vector*.



```
for (uint32 i = 0; i < contact_count; ++i)
{
    // Calculate radii from COM to contact
    KVector2 ra = contacts[i] - rigidbodyA->position;
    KVector2 rb = contacts[i] - rigidbodyB->position;

    // Relative velocity
    KVector2 rv = rigidbodyB->velocity + KVector2::Cross(rigidbodyB->angularVelocity, rb) -
        rigidbodyA->velocity - KVector2::Cross(rigidbodyA->angularVelocity, ra);

    // Relative velocity along the normal
    float contactVel = KVector2::Dot(rv, normal);

    // Do not resolve if velocities are separating
    if (contactVel > 0)
        return;

    float raCrossN = KVector2::Cross(ra, normal);
    float rbCrossN = KVector2::Cross(rb, normal);
    float invMassSum = rigidbodyA->m_invMass + rigidbodyB->m_invMass
        + Square(raCrossN) * rigidbodyA->m_invI + Square(rbCrossN) * rigidbodyB->m_invI;

    // Calculate impulse scalar
    float j = -(1.0f + restitution) * contactVel;
    j /= invMassSum;
    j /= (float)contact_count;

    // Apply impulse
    KVector2 impulse = normal * j;
    rigidbodyA->ApplyImpulse(-impulse, ra);
    rigidbodyB->ApplyImpulse(impulse, rb);
```

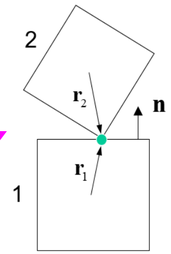


$(V_b - V_a)$

$(V_b - V_a)\cdot n$

$$j = \frac{-(1+\epsilon)(V_b - V_a)\cdot \vec{n}}{\left(\frac{1}{mass_b} + \frac{1}{mass_a}\right)}$$

Relative Velocity



$$\Delta\mathbf{v} = \mathbf{v}_2 + \boldsymbol{\omega}_2 \times \mathbf{r}_2 - \mathbf{v}_1 - \boldsymbol{\omega}_1 \times \mathbf{r}_1$$

Along Normal:

$$v_n = \Delta\mathbf{v} \cdot \mathbf{n}$$

$$\tau = I\alpha.$$

For a simple pendulum, this definition yields a formula for the moment of inertia I in terms of the mass m of the pendulum and its distance r from the pivot point as,

$$I = mr^2.$$

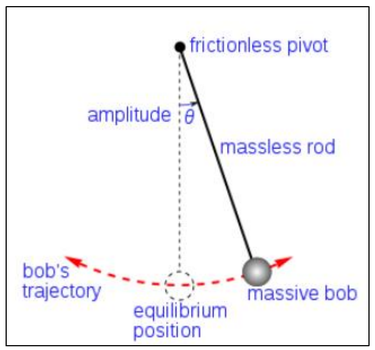
Thus, moment of inertia of the pendulum depends on both the mass m of a body and its geometry, or shape, as defined by the distance r to the axis of rotation.

This simple formula generalizes to define moment of inertia for an arbitrarily shaped body as the sum of all the elemental point masses dm each multiplied by the square of its perpendicular distance r to an axis k. An arbitrary object's moment of inertia thus depends on the spatial distribution of its mass.

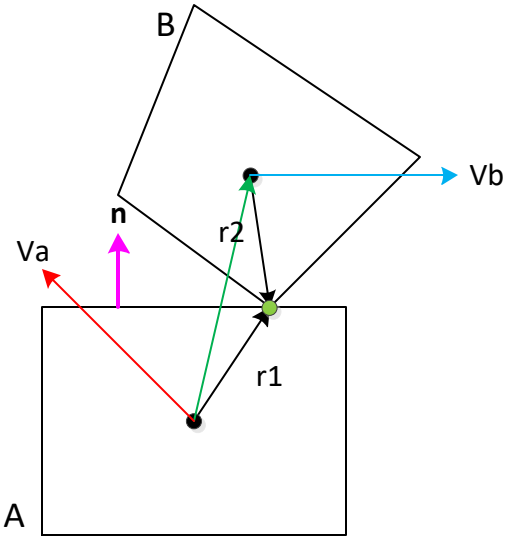
In general, given an object of mass m, an effective radius k can be defined, dependent on a particular axis of rotation, with such a value that its moment of inertia around the axis is

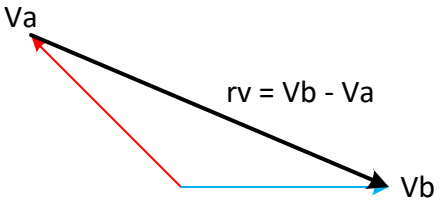
$$I = mk^2,$$

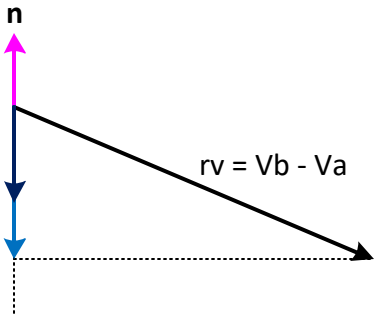
where k is known as the radius of gyration around the axis.




```
/// Friction mixing law. The idea is to allow either fixture to drive the restitution to zero.
/// For example, anything slides on ice.
inline float32 b2MixFriction(float32 friction1, float32 friction2)
{
    return b2Sqrt(friction1 * friction2);
}

/// Restitution mixing law. The idea is allow for anything to bounce off an inelastic surface.
/// For example, a superball bounces on anything.
inline float32 b2MixRestitution(float32 restitution1, float32 restitution2)
{
    return restitution1 > restitution2 ? restitution1 : restitution2;
}
```


B
Vb
n
r2
Va
r1
A


Va
rv = Vb - Va
Vb


n
rv = Vb - Va