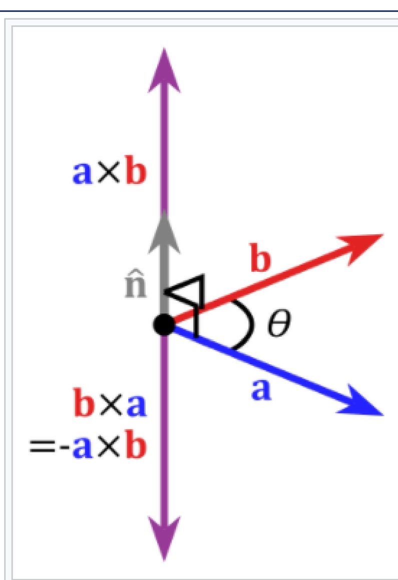
Cross product

From Wikipedia, the free encyclopedia



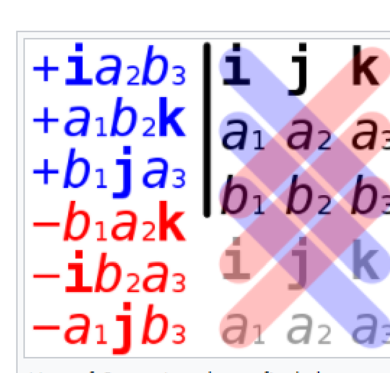
The cross product with respect to a right-handed coordinate system

Matrix notation [edit]

The cross product can also be expressed as the formal determinant:[note 1][2]

$$\mathbf{a}\times\mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

This determinant can be computed using Sarrus's rule or cofactor expansion. Using Sarrus's rule, it expands to



Use of Sarrus's rule to find the cross product of **a** and **b**

$$\begin{aligned}\mathbf{a}\times\mathbf{b} &= (a_2b_3\mathbf{i} + a_3b_1\mathbf{j} + a_1b_2\mathbf{k}) - (a_3b_2\mathbf{i} + a_1b_3\mathbf{j} + a_2b_1\mathbf{k}) \\ &= (a_2b_3 - a_3b_2)\mathbf{i} + (a_3b_1 - a_1b_3)\mathbf{j} + (a_1b_2 - a_2b_1)\mathbf{k}.\end{aligned}$$

Geometric meaning [edit]

See also: Triple product

The magnitude of the cross product can be interpreted as the positive area of the parallelogram having **a** and **b** as sides (see Figure 1):[2]

$$\|\mathbf{a}\times\mathbf{b}\| = \|\mathbf{a}\|\,\|\mathbf{b}\|\,|\sin\theta|.$$

Indeed, one can also compute the volume *V* of a parallelepiped having **a**, **b** and **c** as edges by using a combination of a cross product and a dot product, called scalar triple product (see Figure 2):

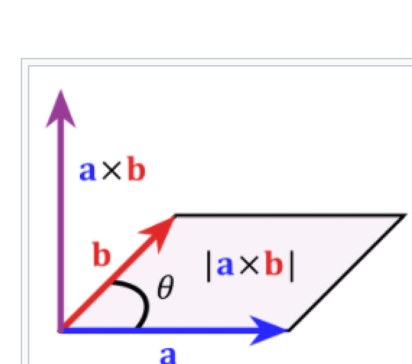
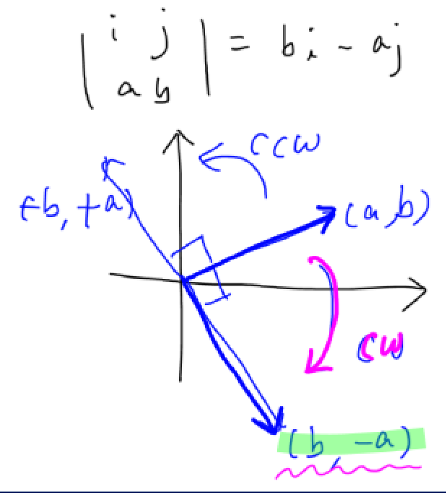


Figure 1. The area of a parallelogram as the magnitude of a cross product

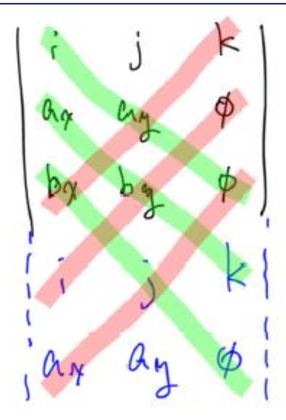
*** 2D Cross Product.**

$$\begin{vmatrix} i & j \\ a & b \end{vmatrix} = bi - aj$$

(b, −a)
clockwise 90°



$$\begin{vmatrix} i & j & k \\ a_x & a_y & \phi \\ b_x & b_y & \phi \end{vmatrix}$$



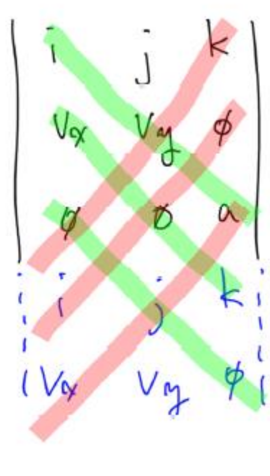
$(0,\ 0,\ a_x b_y k - a_y b_x k)$

⇓

Scalar : $a_x b_y - a_y b_x$

```
float KVector2::Cross(const KVector2& a, const KVector2& b)
{
    return a.x * b.y - a.y * b.x;
}
```

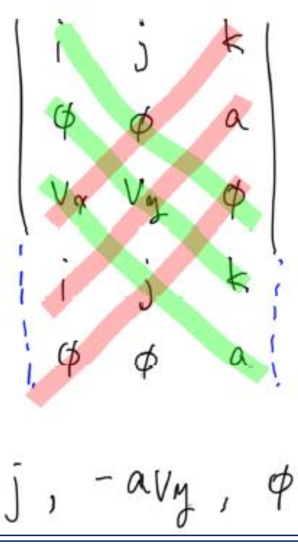
$$\begin{vmatrix} i & j & k \\ V_x & V_y & \phi \\ \phi & \phi & a \end{vmatrix}$$



$(aV_y i,\ -aV_x j,\ \phi)$

```
KVector2 KVector2::Cross(const KVector2& v, float a)
{
    // v is rotated 90-degree CW
    return KVector2(a * v.y, -a * v.x);
}
```

$$\begin{vmatrix} i & j & k \\ \phi & \phi & a \\ V_x & V_y & \phi \end{vmatrix}$$



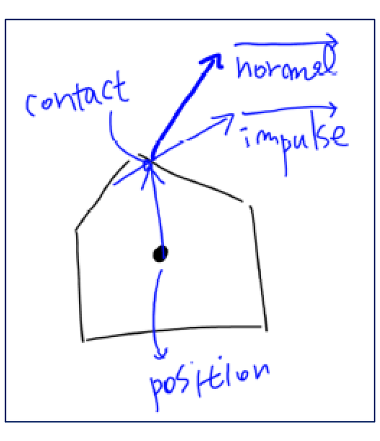
$(aV_x j,\ -aV_y,\ \phi)$

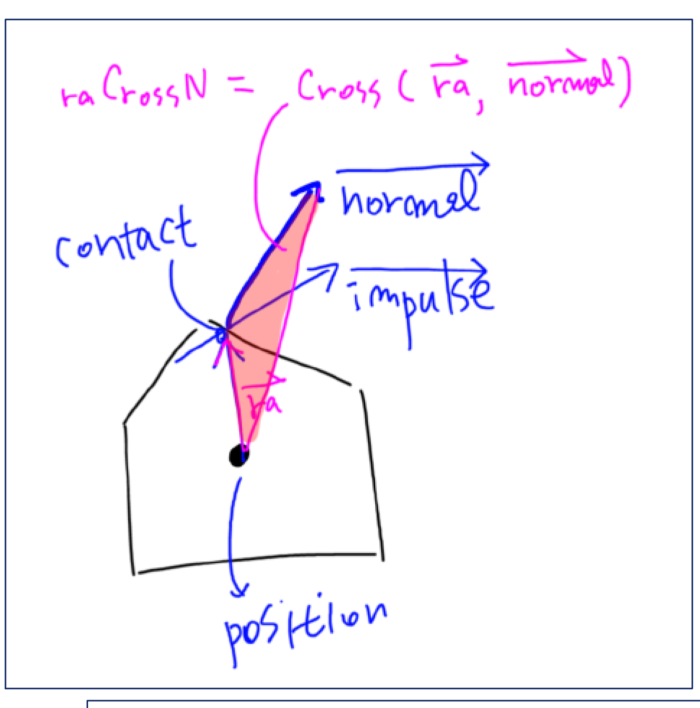
```
KVector2 KVector2::Cross(float a, const KVector2& v)
{
    // v is rotated 90-degree CCW
    return KVector2(-a * v.y, a * v.x);
}
```

angularVelocity and torque is 3rd component(z-component) of 3D-Vector!

```
struct KRigidbody
{
    KRigidbody(KShape *shape_, int32 x, int32 y);
    void ApplyForce(const KVector2& f);
    void ApplyImpulse(const KVector2& impulse, const KVector2& contactVector);
    void SetStatic();
    void SetRotation(float radians);
    KVector2 position;
    KVector2 lastPosition;
    KVector2 velocity;

    float angularVelocity;  // 3d vector (0, 0, angularVelocity)
    float torque;           // 3d vector (0, 0, torque)

    float rotation; // radians

    KVector2 force;

    // Set by shape
    float m_I;  // moment of inertia
    float m_invI; // inverse inertia
    float m_mass;  // mass
    float m_invMass; // inverse mass
```

2D Cross Product as Area






```
void KRigidbody::ApplyImpulse(const KVector2& impulse, const KVector2& contactVector)
{
    velocity += m_invMass * impulse;
    angularVelocity += m_invI * KVector2::Cross(contactVector, impulse);
}
```

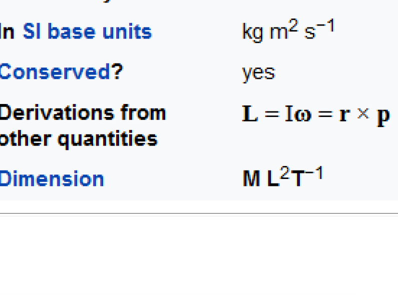
Angular velocity	
Common symbols	ω
In SI base units	s^{-1}
Extensive?	yes
Intensive?	yes (for rigid body only)
Conserved?	no
Behaviour under coord transformation	pseudovector
Derivations from other quantities	$\omega = d\theta / dt$
Dimension	T^{-1}

Impulse (physics)

$$\mathbf{J} = \int_{t_1}^{t_2} \frac{d\mathbf{p}}{dt}\,dt = \int_{\mathbf{p}_1}^{\mathbf{p}_2} d\mathbf{p} = \mathbf{p}_2 - \mathbf{p}_1 = \Delta\mathbf{p}$$

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

Angular momentum

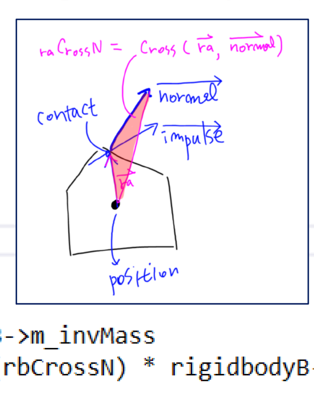


Relationship between force (**F**), torque (**τ**), momentum (**p**), and angular momentum (**L**) vectors in a rotating system. **r** is the position vector.

$$\begin{aligned}\mathbf{L} &= (r^2 m)\left(\frac{\mathbf{r}\times\mathbf{v}}{r^2}\right) \\ &= m\,(\mathbf{r}\times\mathbf{v}) \\ &= \mathbf{r}\times m\mathbf{v} \\ &= \mathbf{r}\times\mathbf{p},\end{aligned}$$

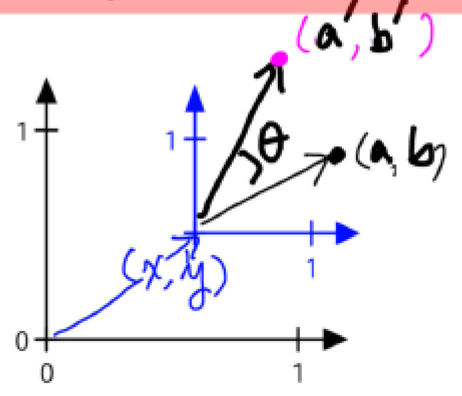
Common symbols	L
In SI base units	$kg\ m^2\ s^{-1}$
Conserved?	yes
Derivations from other quantities	$\mathbf{L} = I\boldsymbol{\omega} = \mathbf{r}\times\mathbf{p}$
Dimension	$M L^2 T^{-1}$

```
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;
```



***2D Rotation with 3D Vector**

quaternion idea



$\swarrow$

(x, y, θ)

$(a', b') = (x, y, \theta) \otimes (a, b, \phi)$

$\Rightarrow (a - x,\ b - y,\ -\theta)$

$\Downarrow$

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} a - x \\ b - y \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix}$$

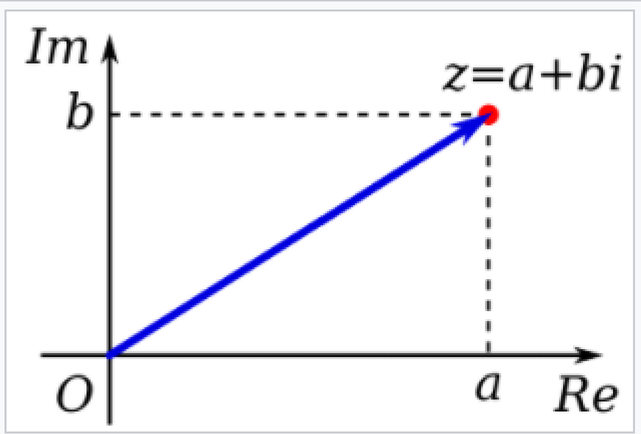
$$= \left((a-x)\cos\theta - (b-y)\sin\theta,\ (a-x)\sin\theta + (b-y)\cos\theta,\ 0 \right)$$

$$= \begin{pmatrix} (a-x)\cos\theta - (b-y)\sin\theta + x, \\ (b-y)\cos\theta + (a-x)\sin\theta + y, \\ 0 \end{pmatrix}$$

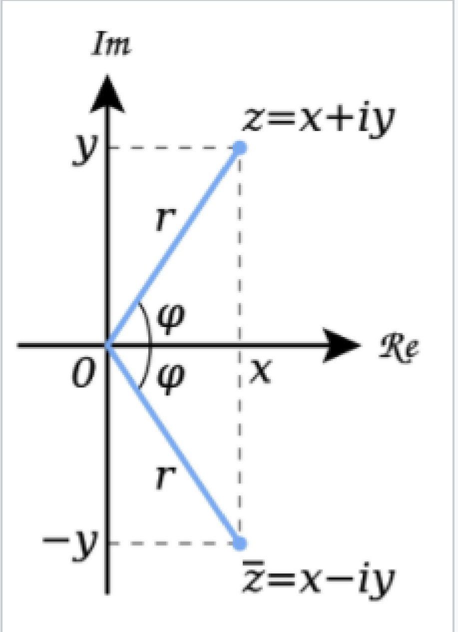
$\cos\theta - \sin\theta\, i$; when $x = \phi,\ y = \phi$

$\cos\theta + \sin\theta\, i$

Complex number



A complex number can be visually represented as a pair of numbers (a, b) forming a vector on a diagram called an Argand diagram, representing the complex plane. $\mathcal{Re}$ is the real axis, $\mathcal{Im}$ is the imaginary axis, and i is the "*imaginary unit*" that satisfies $i^2 = -1$.



Geometric representation of z and its conjugate $\bar{z}$ in the complex plane

Euler's formula [edit]

Euler's formula states that, for any real number y,

$$e^{iy} = \cos y + i \sin y.$$

The functional equation implies thus that, if x and y are real, one has

$$e^{x+iy} = e^x(\cos y + i \sin y) = e^x \cos y + ie^x \sin y,$$

which is the decomposition of the exponential function into its real and imaginary parts.

Quaternion

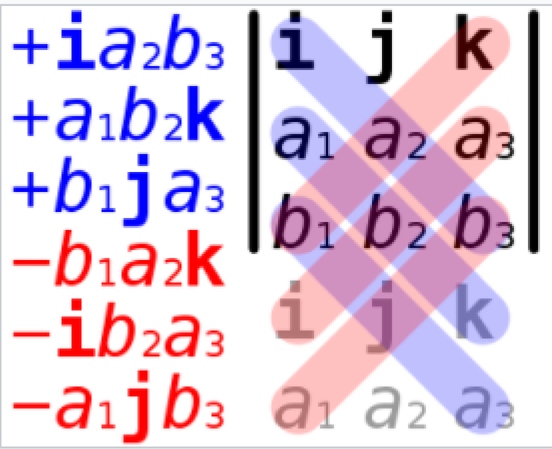
Quaternions are generally represented in the form

$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$$

Quaternion multiplication table

	1	**i**	**j**	**k**
1	1	**i**	**j**	**k**
i	**i**	−1	**k**	−**j**
j	**j**	−**k**	−1	**i**
k	**k**	**j**	−**i**	−1

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$



Use of Sarrus's rule to find the cross product of **a** and **b**

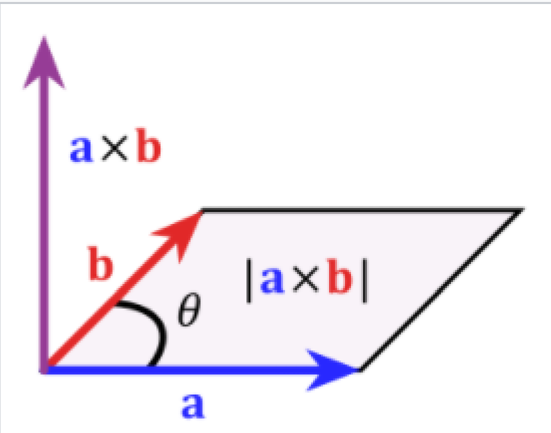


Figure 1. The area of a parallelogram as the magnitude of a cross product