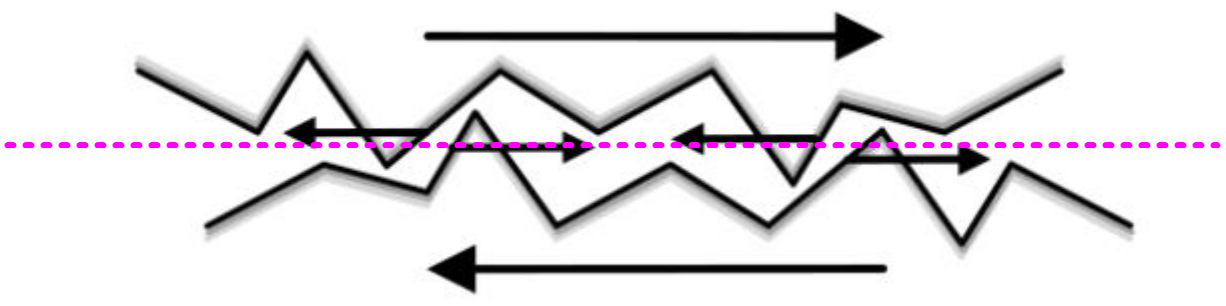
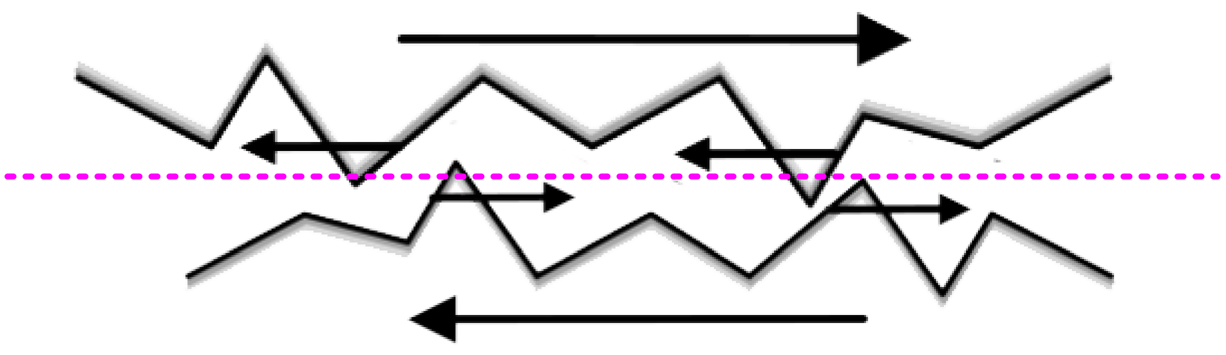
Coulomb Friction Model

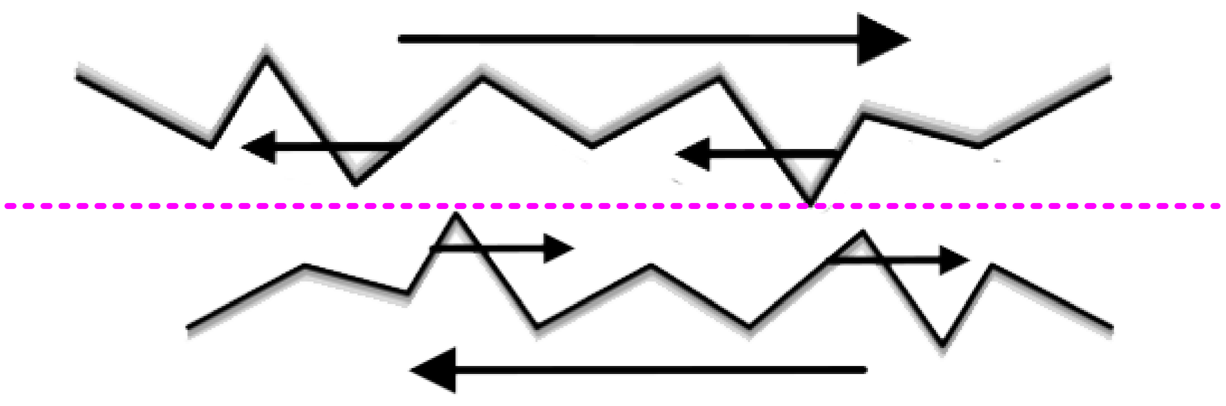
Static Friction



Dynamic Friction



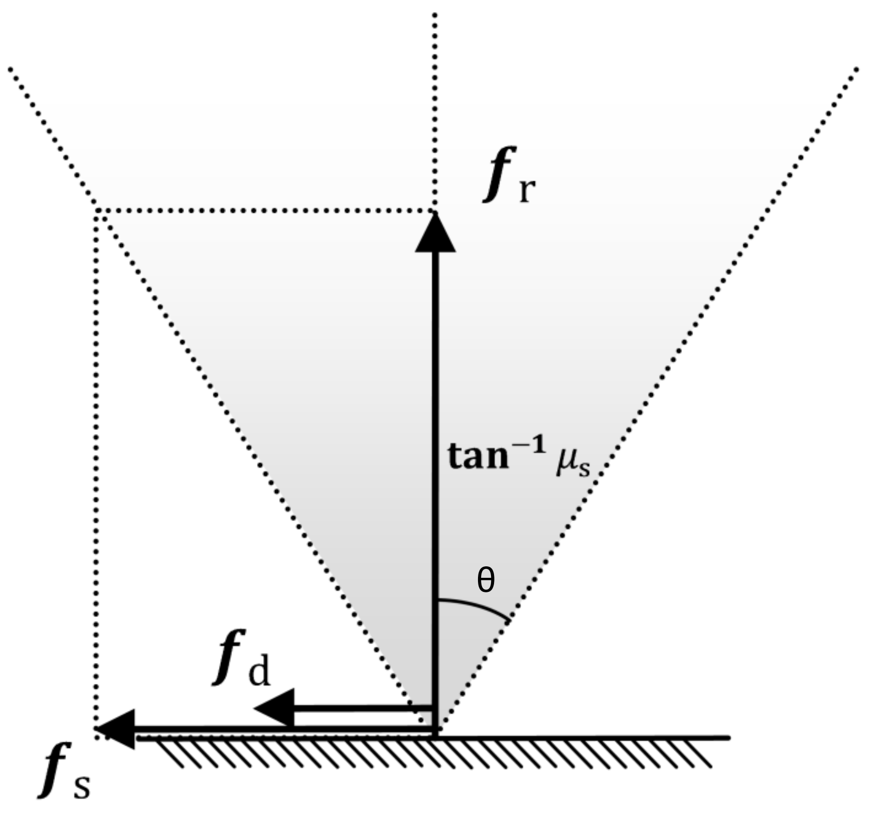
No Friction: No Contact



Impulse-based friction model [edit]

One of the most popular models for describing friction is the Coulomb friction model. This model defines coefficients of static friction $\mu_s \in \mathbb{R}$ and dynamic friction $\mu_d \in \mathbb{R}$ such that $\mu_s > \mu_d$. These coefficients describe the two types of friction forces in terms of the reaction forces acting on the bodies. More specifically, the static and dynamic friction force magnitudes $f_s, f_d \in \mathbb{R}$ are computed in terms of the reaction force magnitude $f_r = |\mathbf{f}_r|$ as follows

$$f_s = \mu_s f_r$$
$$f_d = \mu_d f_r$$

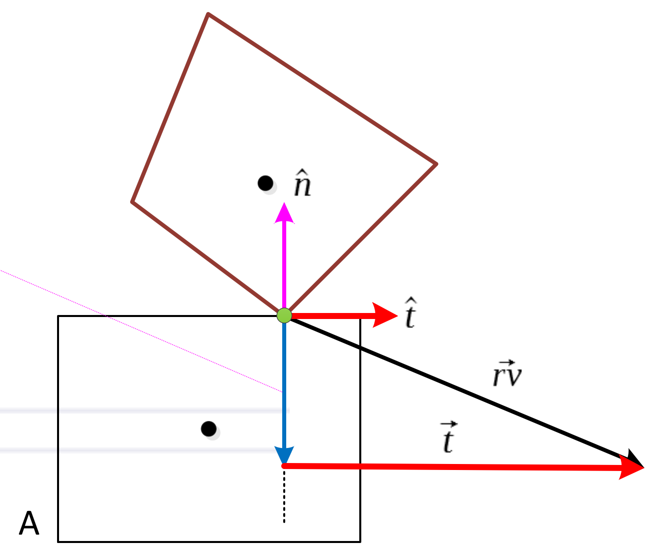


$$f_s = \mu_s f_r$$
$$f_d = \mu_d f_r$$
$$\frac{f_s}{f_r} = \mu_s$$
$$\frac{f_d}{f_r} = \mu_d$$
$$\tan(\theta) = \frac{f_s}{f_r} = \mu_s$$
$$\tan^{-1}\mu_s = \theta$$

$$j_s = \mu_s j_r$$
$$j_d = \mu_d j_r$$

$$\mathbf{j}_f = \begin{cases} -(m\mathbf{v}_r \cdot \hat{\mathbf{t}})\hat{\mathbf{t}} & \mathbf{v}_r \cdot \hat{\mathbf{t}} = 0 \quad m\mathbf{v}_r \cdot \hat{\mathbf{t}} \leq j_s \\ -j_d \hat{\mathbf{t}} & (\text{otherwise}) \end{cases}$$

```
// Friction impulse
if( g_enableFriction == true )
{
    rv = rigidbodyB->velocity + KVector2::Cross(rigidbodyB->angularVelocity, rb)
       - rigidbodyA->velocity - KVector2::Cross(rigidbodyA->angularVelocity, ra);

    KVector2 t = rv - (normal * KVector2::Dot(rv, normal));
    t.Normalize();

    // j tangent magnitude
    float jt = KVector2::Dot(rv, t);
    jt /= invMassSum;
    jt /= (float)contact_count;

    // Don't apply tiny friction impulses
    if (IsEqual(jt, 0.0f))
        return;

    // Couloumb's law
    KVector2 tangentImpulse;
    if (std::abs(jt) < j * sf)
        tangentImpulse = -(t * jt);
    else
        tangentImpulse = -(t * j * df);

    // Apply friction impulse
    rigidbodyA->ApplyImpulse(-tangentImpulse, ra);
    rigidbodyB->ApplyImpulse(tangentImpulse, rb);
}/**/
```



$$j_s = \mu_s j_r$$
$$j_d = \mu_d j_r$$

$$\mathbf{j}_f = \begin{cases} -(m\mathbf{v}_r \cdot \hat{\mathbf{t}})\hat{\mathbf{t}} & \mathbf{v}_r \cdot \hat{\mathbf{t}} = 0 \quad m\mathbf{v}_r \cdot \hat{\mathbf{t}} \leq j_s \\ -j_d \hat{\mathbf{t}} & (\text{otherwise}) \end{cases}$$

```
// Friction impulse
if( g_enableFriction == true )
{
    rv = rigidbodyB->velocity + KVector2::Cross(rigidbodyB->angularVelocity, rb)
        - rigidbodyA->velocity - KVector2::Cross(rigidbodyA->angularVelocity, ra);

    KVector2 t = rv - (normal * KVector2::Dot(rv, normal));
    t.Normalize();

    // j tangent magnitude
    float jt = KVector2::Dot(rv, t);
    jt /= invMassSum;
    jt /= (float)contact_count;

    // Don't apply tiny friction impulses
    if (IsEqual(jt, 0.0f))
        return;

    // Couloumb's law
    KVector2 tangentImpulse;
    if (std::abs(jt) < j * sf)
        tangentImpulse = -(t * jt);
    else
        tangentImpulse = -(t * j * df);

    // Apply friction impulse
    rigidbodyA->ApplyImpulse(-tangentImpulse, ra);
    rigidbodyB->ApplyImpulse(tangentImpulse, rb);
}/**/
```

$$j_s = \mu_s j_r$$

$$j_d = \mu_d j_r$$

$$\mathbf{j}_f = \begin{cases} -(m\mathbf{v}_r \cdot \hat{\mathbf{t}})\hat{\mathbf{t}} & \mathbf{v}_r \cdot \hat{\mathbf{t}} = 0 \quad m\mathbf{v}_r \cdot \hat{\mathbf{t}} \leq j_s \\ -j_d \hat{\mathbf{t}} & \text{(otherwise)} \end{cases}$$