Motion Laws for Javascript Animation

Stefan Gössner

26.10.2010

Coulomb Friction (Dry Friction)

The Coulomb friction basically depends on the normal force *N* between two bodies currently in contact and the coefficient of friction *μ* between them, which is solely depending on the materials used.

The coefficient of friction *μ* is dimensionless and has always a value between 0.0 and 1.0.

The equations of motion are

$$s(t) = v_0 t - \frac{1}{2} a t^2$$
$$v(t) = v_0 - a t$$

s(t)	=	distance travelled over time (pixels)
v(t)	=	velocity over time (pixels per second)
v_0	=	start velocity (pixels per second)
μ	=	coefficient of friction [0 .. 1]
a	=	deceleration – depending on normal force (pixels per second^2)

The velocity over time decreases linearly.
Using this for motion on a screen a Javascript function may look like

```
function s(v0,mu,a,t) {
   return v0*t - 0.5*mu*a*t*t;
}
```

Damped Oscillation

The damped oscillation is a harmonic oscillation, that comes to halt after some time.

The general equations of motion are

$$\omega_0 = \frac{2\pi}{T}$$
$$s(t) = A e^{-\lambda \omega_0 t} \sin(\omega_0 \sqrt{1-\lambda^2}\, t + \varphi_0)$$
$$v(t) = -A \omega_0^2 \lambda \sqrt{1-\lambda^2}\, e^{-\lambda \omega_0 t} \cos(\omega_0 \sqrt{1-\lambda^2}\, t + \varphi_0)$$

s(t)	=	distance travelled over time (pixels)
v(t)	=	velocity over time (pixels per second)
A	=	amplitute (pixels)
λ	=	damping ratio [0 .. 1] dimensionless
ω_0	=	natural (undamped) frequency (radians per second)
φ_0	=	phase (radians
T	=	time period

Two different special cases have to be discussed now with these general equations.

Case I: Static displacement from equilibrium

We assume an initial displace of s_0. Static displacement means here, no initial velocity. So the initial conditions are

$$s(t=0)=s_0 \quad \rightarrow \quad A=s_0$$
$$v(t=0)=0 \quad \rightarrow \quad \varphi_0=\frac{\pi}{2}$$

This leads to the equation of motion

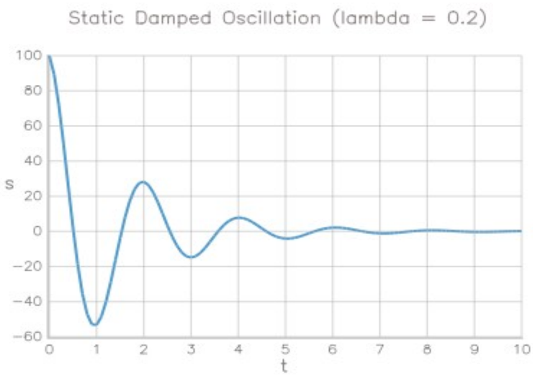
$$s(t)=s_0 e^{-\lambda \omega_0 t} \cos(\omega_0 \sqrt{a-\lambda^2} t)$$

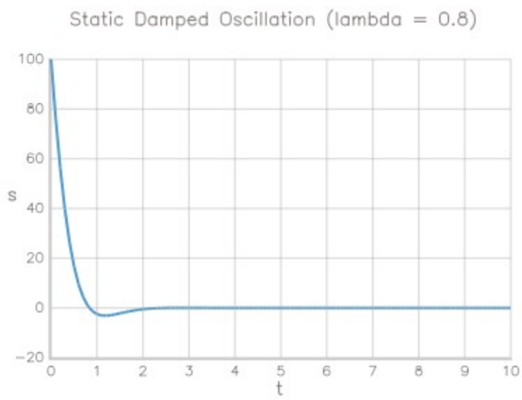
which can be easily put to a Javascript function

```
function staticDampedOscillation(s0,T,lambda,t) {
   var omega0 = 2*Math.PI/T;
   return s0*Math.exp(-lambda*omega0*t)*
             Math.cos(omega0*Math.sqrt(1-lambda*lambda)*t);
}
```

With a damping ratio of $\lambda=0$, we yield an undamped motion oscillating forever. A damping ratio of $\lambda=1$ results in *critical damping*, that goes back to equilibrium position as fast as possible without oscillation.

A line chart of this function looks like this:





Case I: Dynamic oscillation

We assume an initial velocity of v_0 without an initial displacement. So the initial conditions are now

$$s(t=0)=0 \quad \rightarrow \quad \varphi_0=0$$

$$v(t=0)=v_0 \quad \rightarrow \quad A=\frac{v_o}{\lambda\omega^2\sqrt{1-\lambda^2}}$$

This leads to the equation of motion

$$s(t)=\frac{v_o}{\lambda\omega_0^2\sqrt{1-\lambda^2}}e^{-\lambda\omega_0 t}\sin(\omega_0\sqrt{a-\lambda^2}t)$$

which can also be easily written as a Javascript function

```
function dynamicDampedOscillation(v0,T,lambda,t) {
   var omega0  = 2*Math.PI/T,
       omega   = omega0*Math.sqrt(1-lambda*lambda);
   return v0/(lambda*omega0*omega)*
          Math.exp(-lambda*omega0*t)*Math.sin(omega*t);
}
```

The damping ratio has the same behavior as with static oscillation.

A line chart of this function looks like this: