Physics Simulation Tutorial

From Custom Integrators to MuJoCo

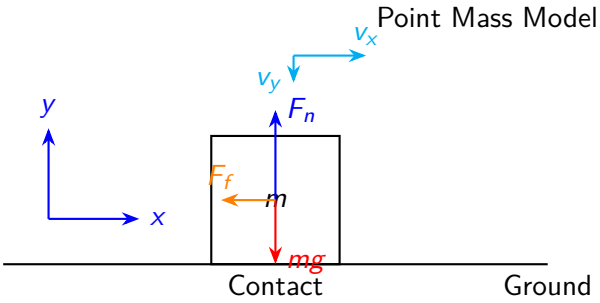
Chunwei Xing

October 16, 2025

Table of Contents

Table of Contents

Point Mass Physics Model



- Point-mass state $\mathbf{x} = [x, y, v_x, v_y]$ with gravity mg.
- Contact height $d = \max(0, h_{\text{box}} - y)$, normal force $F_n = kd + c(-v_y)$.
- Friction force $F_f = -\mu F_n \tanh(\alpha v_x)$ provides horizontal damping.
- Semi-implicit Euler: $v_{t+1} = v_t + \frac{\mathbf{F}}{m}\Delta t$, $x_{t+1} = x_t + v_{t+1}\Delta t$.

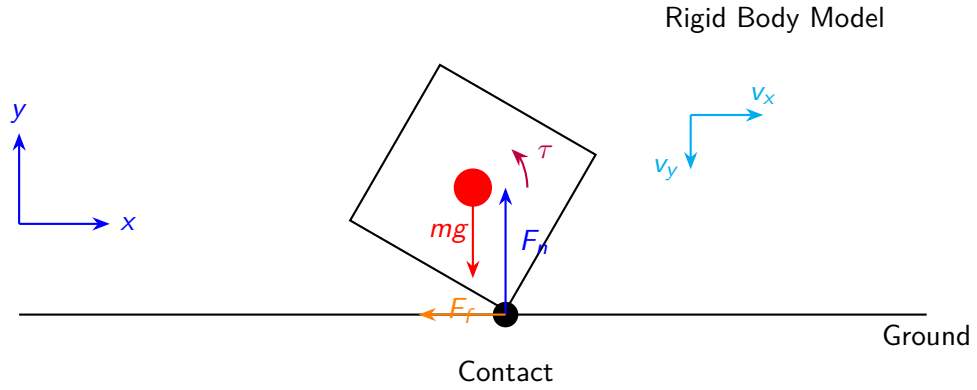
Point Mass Simulation Algorithm

Algorithm 1 Falling Box Simulation

$\mathbf{x} \leftarrow [x_0, y_0, v_{x0}, v_{y0}]$ ▷ initial state
for $t = 0$ to $N - 1$ **do**
    $d \leftarrow \max(0, h - y)$ ▷ check contact
    $F_n \leftarrow kd + c\max(0, -v_y)$ ▷ compute normal force
    $F_f \leftarrow -\mu F_n \tanh(\alpha v_x)$ ▷ compute friction force
    $\mathbf{F} \leftarrow [F_f, -mg + F_n]$ ▷ compute total force
    $\mathbf{v} \leftarrow \mathbf{v} + (\mathbf{F}/m)\Delta t$ ▷ update velocity
    $\mathbf{p} \leftarrow \mathbf{p} + \mathbf{v}\Delta t$ ▷ update position
end for

Table of Contents

Rigid Body Dynamics Model



- State $\mathbf{x} = [\mathbf{p}, \mathbf{v}, \theta, \omega]$ with inertia $I = \frac{1}{12} m(w^2 + h^2)$.
- Corners: $\mathbf{c}_i = \mathbf{p} + R(\theta)\mathbf{c}_i^{\text{local}}$.
- Contact per corner: if $c_{i,y} < 0$, compute (F_t, F_n); torque $\tau += \mathbf{r}_i \times \mathbf{F}_i$.

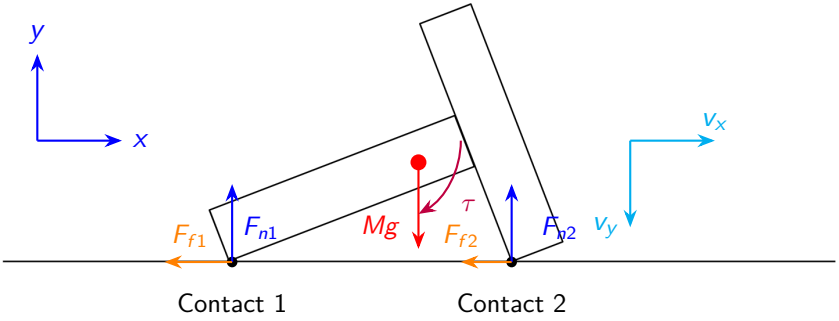
Rigid Body Simulation Algorithm

Algorithm 2 Rigid Box Simulation

precompute local corner coordinates $\{\mathbf{c}_i^{\text{local}}\}$ ▷ contact points
for $t = 0$ to $N - 1$ **do**
    $\mathbf{F} \leftarrow [0, -mg]$, $\tau \leftarrow 0$ ▷ reset force and torque
    for each corner i **do**
        $\mathbf{r}_i \leftarrow R(\theta)\mathbf{c}_i^{\text{local}}$ ▷ contact point
        $\mathbf{c}_i \leftarrow \mathbf{p} + \mathbf{r}_i$ ▷ contact point in world frame
        if $c_{i,y} < 0$ **then**
            $\mathbf{v}_i \leftarrow \mathbf{v} + \omega[-r_{i,y}, r_{i,x}]$ ▷ contact point velocity
            $\mathbf{F}_i \leftarrow \textsc{ContactForce}(\mathbf{c}_i, \mathbf{v}_i)$
            $\mathbf{F} \leftarrow \mathbf{F} + \mathbf{F}_i$ ▷ update force
            $\tau \leftarrow \tau + \mathbf{r}_i \times \mathbf{F}_i$ ▷ update torque
        end if
    end for
    update $(\mathbf{v}, \omega)$ and $(\mathbf{p}, \theta)$ ▷ update velocity and angular velocity
end for

Table of Contents

Composite Body Assembly



- Multiple contact points: each corner can generate forces and torques
- Complex geometry: T-shape creates asymmetric contact patterns
- Total torque: $\tau = \sum_i \mathbf{r}_i \times \mathbf{F}_i$ from all contacts

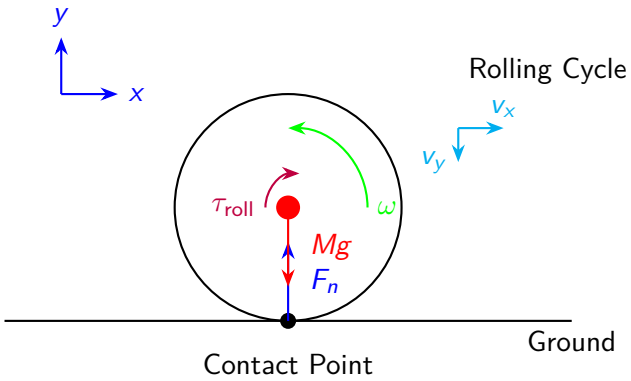
Composite Body Simulation Algorithm

Algorithm 3 Compound Body Simulation

precompute geometry list $\{\text{geom}_j\}$ with local vertices and masses
compute total mass, COM offset, inertia about COM
for $t = 0$ to $N - 1$ **do**
    $\mathbf{F} \leftarrow [0, -Mg]$, $\tau \leftarrow 0$ ▷ reset force and torque
    for each geometry j **do**
        for vertex $\mathbf{c}^{\text{local}}$ in geom_j **do**
            $\mathbf{r} \leftarrow R(\theta)\mathbf{c}^{\text{local}}$ ▷ contact point
            $\mathbf{c} \leftarrow \mathbf{p} + \mathbf{r}$ ▷ contact point in world frame
            if $c_y < 0$ **then**
                compute contact force and add to $\mathbf{F}$, τ
            end if
        end for
    end for
    integrate $(\mathbf{v}, \omega)$ and $(\mathbf{p}, \theta)$
end for

Table of Contents

Rolling Circle Dynamics



- Consider a single contact point beneath centre $\mathbf{r} = [0, -R]$. (multiple points?)
- Consider only the rolling friction, not the sliding friction. (both frictions?)

Rolling Circle Simulation Algorithm

Algorithm 4 Rolling Circle Simulation

for $t = 0$ to $N - 1$ **do**
  $\mathbf{F} \leftarrow [0, -Mg]$, $\tau \leftarrow 0$ ▷ reset force and torque
  $\mathbf{r} \leftarrow [0, -R]$ ▷ contact point
  $\mathbf{c} \leftarrow \mathbf{p} + \mathbf{r}$ ▷ contact point in world frame
  if $c_y < 0$ **then**
    $\mathbf{v}_{\text{contact}} \leftarrow \mathbf{v} + \omega[-r_y, r_x]$ ▷ contact point velocity
    $(\mathbf{F}_c, F_n) \leftarrow \textsc{ContactForce}(\mathbf{c}, \mathbf{v}_{\text{contact}})$ ▷ compute contact forces
    $\mathbf{F} \leftarrow \mathbf{F} + \mathbf{F}_c$ ▷ update force
    $\tau \leftarrow \tau + \mathbf{r} \times \mathbf{F}_c - \mu_{\text{roll}} F_n \operatorname{sign}(\omega)$ ▷ update torque
  end if
  update velocity and angular velocity
end for

Table of Contents

MuJoCo Configuration

```
<!-- Compiler settings -->
<compiler angle="radian" coordinate="local"/>

<!-- Simulation options -->
<option timestep="0.001" gravity="0 0 -9.81" integrator="Euler">
    <flag warmstart="disable"/>
</option>

<!-- Contact parameters matching custom implementation -->
<default>
    <geom friction="0.0 0.0 0.01" condim="6" solimp="0.99 0.995 0.001" solref="0.01 1.0"/>
```

```
<!-- World body (fixed objects) -->
<worldbody>
    <!-- Ground plane -->
    <geom name="ground" type="plane" size="5 5 0.1" rgba="0.5 0.5 0.5 1" material="grid"/>

    <!-- Light source -->
    <light directional="true" diffuse="0.8 0.8 0.8" specular="0.2 0.2 0.2" pos="0 0 5" dir="0 0 -1"/>

    <!-- Thin disk (wheel) -->
    <body name="disk" pos="0 0 1.0">
        <inertial pos="0 0 0" mass="1.0" diaginertia="0.010033 0.020000 0.010033"/>
        <!-- Disk geometry: cylinder with axle along Y-axis -->
        <geom name="disk_geom" type="cylinder" size="0.2 0.01" euler="1.57079632679 0 0" rgba="0.2 0.4 0.8 1"/>
        <!-- Visual indicator to show disk rotation -->
        <geom name="disk_indicator" type="capsule" fromto="0 0 0  0.2 0 0" size="0.01" density="0" contype="0" conaffinity="0" rgba="1 0.2 0.2 1"/>

        <!-- Initial velocity (set via qvel in simulation) -->
        <freejoint/>
    </body>
</worldbody>
```

- `condim`=6 enables sliding/torsional/rolling friction; tuple $(\mu_{\text{slide}}, \mu_{\text{torsion}}, \mu_{\text{roll}})$.
- `solref` from k, c: timeconst $= \sqrt{m/k}$, dampratio $= c/(2\sqrt{mk})$.
- Inertial properties: $I_y = 0.5mR^2$, $I_x = I_z = 0.25mR^2 + \frac{1}{12}m(2R)^2$.

Numerical Integration for Rigid Bodies

Explicit Euler

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{a}_t \, \Delta t$$
$$\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{v}_t \, \Delta t$$

- Less stable
- More energy drift
- Simple to implement

Semi-implicit Euler

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{a}_t \, \Delta t$$
$$\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{v}_{t+1} \, \Delta t$$

- Most common
- Good balance of stability and simplicity

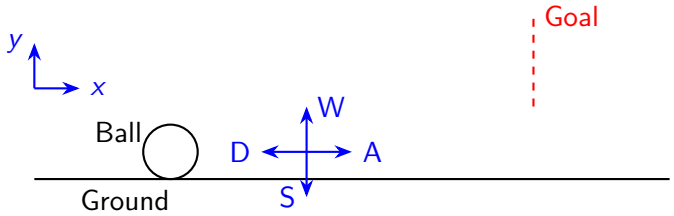
Implicit Euler

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{a}_{t+1} \, \Delta t$$
$$\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{v}_{t+1} \, \Delta t$$

- Most stable
- Requires solving for $\mathbf{a}_{t+1}$
- Good for stiff contacts

Other integrators: RK4 (more accurate, more stable, more expensive).

Assignment - Interactive Simulator



Task: Design an interactive simulator, and you can **apply forces** to the circle using your keyboard.

Rules: Assume you get a passing ball from your teammate, and you need to kick at least $1m$ away from the goal.

Bonus Points: For people who submit the code and a video of themselves (only your hand and your laptop's screen) playing with it to score a goal.