# Simulation

**Nicolas Mansard**

Gepetto
LAAS-CNRS & ANITI

# Overview

NVIDIA ISAAC

MuJoCo

DART
Dynamic Animation and Robotics Toolkit

RaiSim

DRAKE

Simple
The Simple simulator BY Inria

BULLET PHYSICS LIBRARY

**Collision detection**
contact points and normals

**Problem formulation**
intertia, biais forces, jacobians

**Problem resolution**
search forces and accelerations

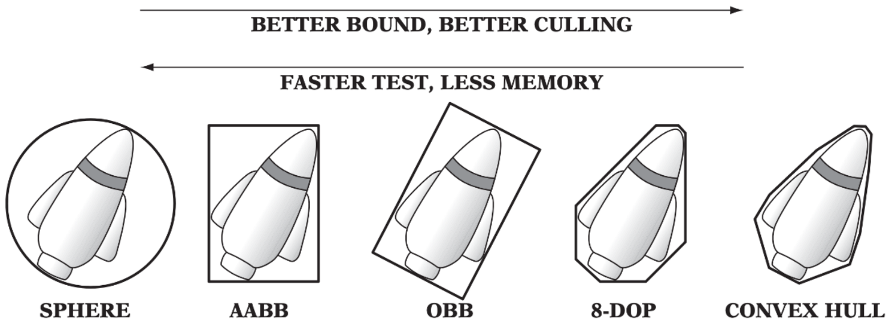
**Temporal integration**
numerical integration (event loop)

# Collision model

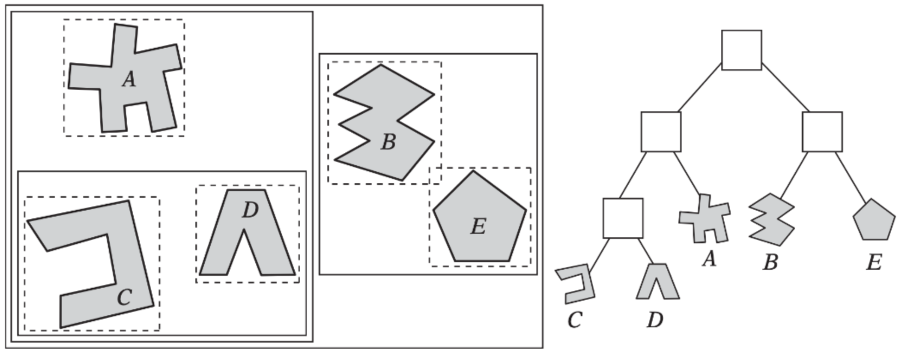
- Documentation of the kinematic tree
  - Placement wrt joint
  - Simple shape
    - Sphere, capsules, boxes …

  OR
  - Polygon soup
  - Collision pairs

- Ad-hoc structure for broad phase

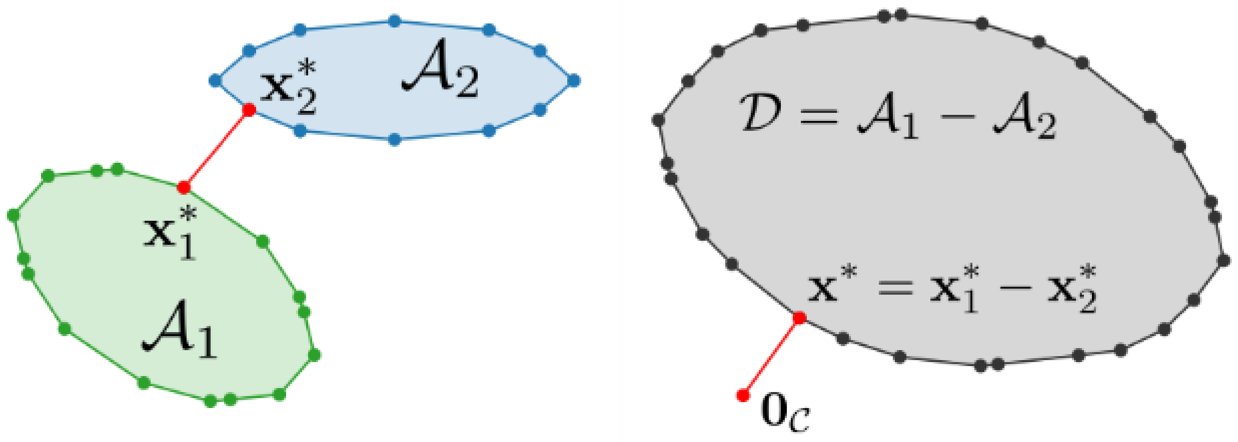



Credit: *Real-time Collision Detection*, Ericson




Credit: *Real-time Collision Detection*, Ericson

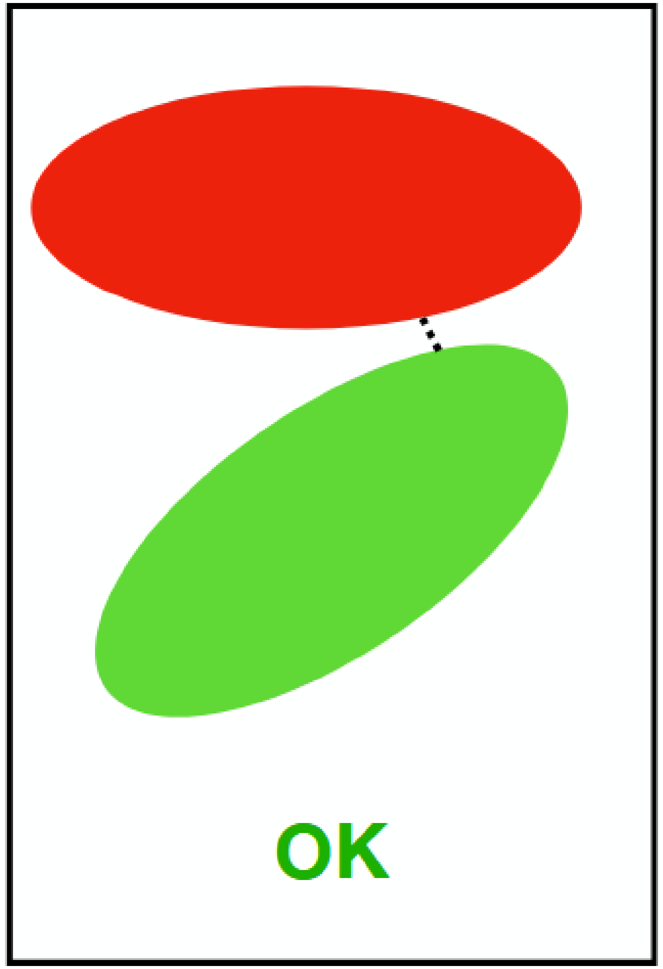
# Collision detection and distances



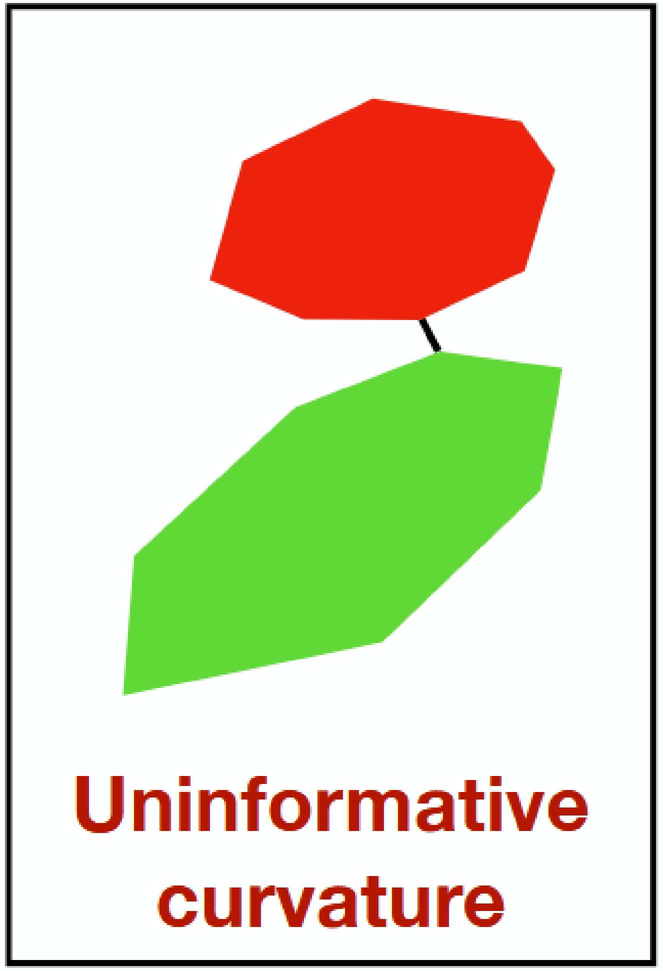

$$\min_{x_1 \in \mathcal{A}_1, x_2 \in \mathcal{A}_2} \frac{1}{2} ||x_1 - x_2||^2 \longrightarrow \boxed{\min_{x \in \mathcal{D}} \frac{1}{2} ||x||^2}$$

From Louis Montaut thesis, *must read*

# Differentiability








From Louis Montaut thesis, *must read*

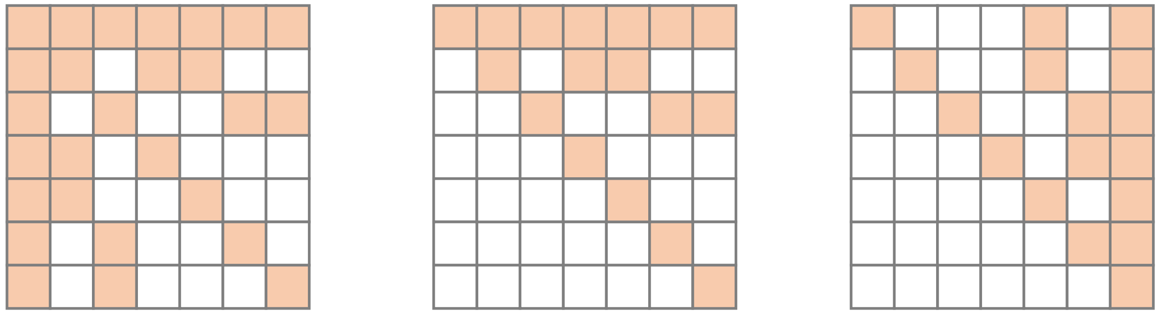
# Dynamics with (bilateral) constraints

- With no constraint:

$$Ma + b = \tau$$

- Solve with triangular decomposition

$$M \quad = \quad U \quad * \quad U^T$$



Featherstone textbook, 2008

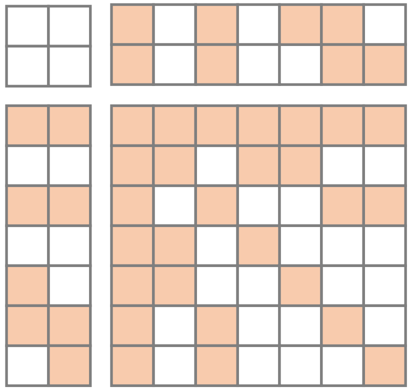
# Dynamics with (bilateral) constraints

- With constraints:

$$Ma + b = \tau + J^T f$$
$$Ja + a_0 = 0$$

- Corresponding KKT matrix

$$\begin{pmatrix} 0 & J \\ J^T & M \end{pmatrix} = \begin{pmatrix} U_D & X \\ & U \end{pmatrix} \begin{pmatrix} -1 & \\ & 1 \end{pmatrix} \begin{pmatrix} U_D^T & 0 \\ X^T & U^T \end{pmatrix}$$



with $\mathrm{U} = \sqrt{M}, \mathrm{X} = \mathrm{J}\sqrt{M^{-T}}, U_D = \sqrt{JM^{-1}J^T}$

Carpentier, Budhiraja, Mansard, 2021

# Friction-less contacts

- Complementarity problem

$$\ddot{q} = M^{-1}(\tau - b + J^T f)$$

$$J\ddot{q} + \dot{J}\dot{q} \geq 0 \quad \perp \quad f \geq 0$$

*no penetration* *no pulling*

*one or the other*

- Equivalent to a principled QP

$$\min_{\ddot{q}} \left\| \ddot{q} - \ddot{q}_{free} \right\|_M$$

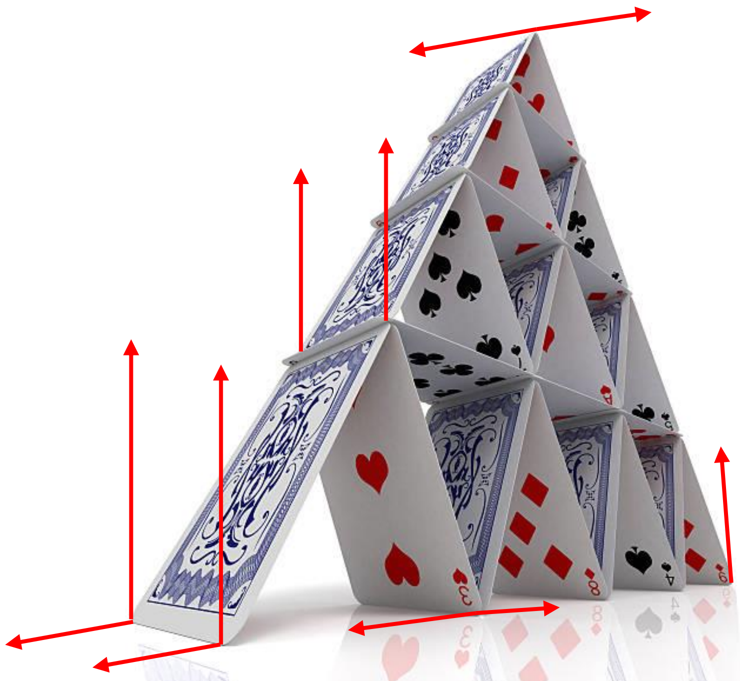
*Subject to* $J\ddot{q} + \dot{J}\dot{q} \geq 0$

# Maximum dissipation

$f \in K$ with $J v_q = 0$ ... sticking contact

$f = 0$ with $J_{\perp} v_q > 0$ ... breaking contact

$f \in \partial K$ with $J_{//} v_q = \alpha f_{//}, \alpha > 0$ ... sliding contact

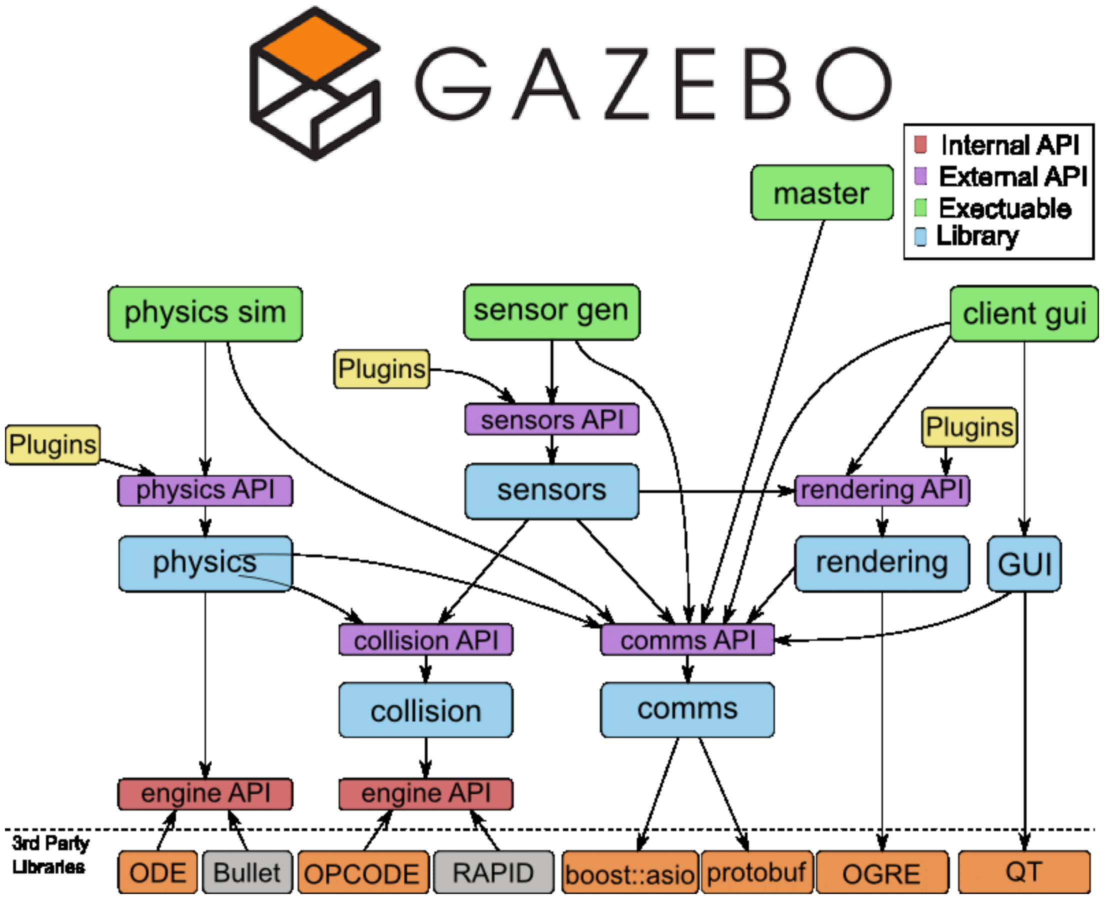


# Overview of the main simulators

| Simulator | Collision | Contact/Friction | Constraint Solving | Speed | Accuracy | GPU Support | RL Support | Open Source |
|---|---|---|---|---|---|---|---|---|
| **MuJoCo** | Convex + Mesh | Regularized contact model | Impl. Newton + constraints | Very fast | High | ⚠ through MJX | ☑ | ☑ (Apache, DeepMind lead) |
| **Isaac Gym/Sim** | Mesh + SDF | Penalty + Coulomb | GPU-based PGS | Extremely fast | Good | ☑ | ☑ | ⚠ (recently open, TBC) |
| **PyBullet** | Convex + Mesh (GImpact) | LCP + Coulomb | PGS + constraints | Fast | Medium | ⚠ (Partial) | ☑ | ☑ |
| **DART** | Mesh + primitives | LCP + Penalty | DAE-based LCP | Medium | High | ✕ | ⚠ (Limited) | ☑ |
| **Drake** | Convex + Analytic | Hydroelastic + Penalty | Implicit SCS / LCP | Medium | Very high | ✕ | ☑ | ☑ |
| **RaiSim** | Convex + Mesh | Rigid contact + Coulomb | NCP via PGS | Very fast | High | ✕ (CPU only) | ☑ | ✕ (Free academic) |
| **ODE** | Primitives | Penalty-based | PGS | Medium | Low-Med | ✕ | ⚠ (Limited) | ☑ |
| **Simple** | Convex + Mesh + Primitives | Rigid with exact regularization | ADMM with prox | Fast | Excellent | ✕ (CPU only) | ✕ | ☑ BSDv2 |

# System simulator

- A **contact** simulator is only a part of a **system** simulator
  - Sensors
  - Software interface
  - Rendering
  - …



# SimScape

- Simulation par pénalité
  - Difficile à régler
  - Intégrateur fin (couteux en temps de calcul)

- Peu de documentation sur le modèle de collision
  - Ad-hoc? Défini par l’utilisateur?

- Intégration impossible
  - Pas de support RL ou GPU
  - Lien avec ROS ?