Simulation III

Agimus Winter School 2023

Quentin Le Lidec and Justin Carpentier

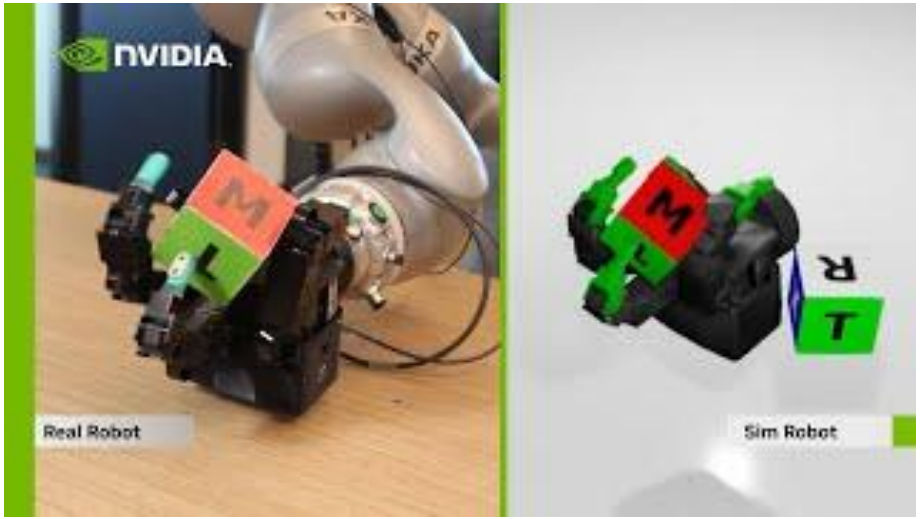
Outline

Contact simulation in robotics

1. Models in robotics
2. Modeling the physics of contacts
3. Algorithmic variations
4. Experiments

Models in robotics

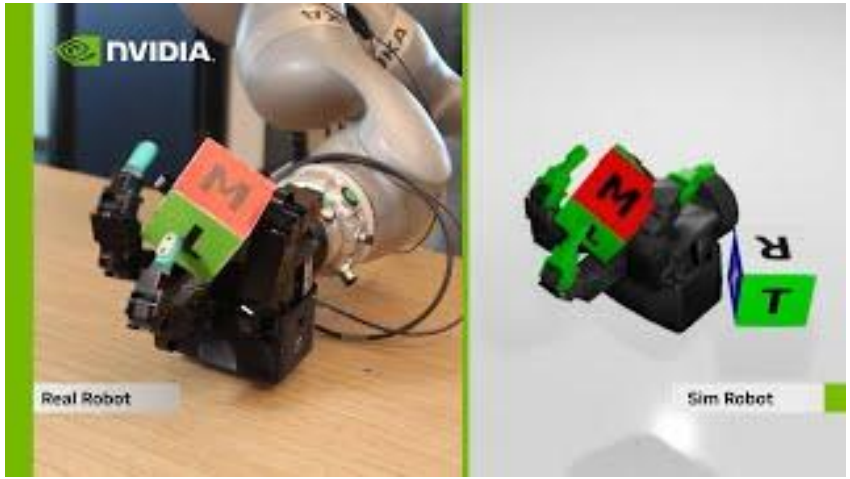
Model-based vs. model-free



Models in the era of “model-free” RL

Parameter	Type	Distribution	Initial Range	ADR-Discovered Range
Hand				
Mass	Scaling	uniform	[0.4, 1.5]	[0.4, 1.5]
Scale	Scaling	uniform	[0.95, 1.05]	[0.95, 1.05]
Friction	Scaling	uniform	[0.8, 1.2]	[0.54, 1.58]
Armature	Scaling	uniform	[0.8, 1.02]	[0.31, 1.24]
Effort	Scaling	uniform	[0.9, 1.1]	[0.9, 2.49]
Joint Stiffness	Scaling	loguniform	[0.3, 3.0]	[0.3, 3.52]
Joint Damping	Scaling	loguniform	[0.75, 1.5]	[0.43, 1.6]
Restitution	Additive	uniform	[0.0, 0.4]	[0.0, 0.4]
Object				
Mass	Scaling	uniform	[0.4, 1.6]	[0.4, 1.6]
Friction	Scaling	uniform	[0.3, 0.9]	[0.01, 1.60]
Scale	Scaling	uniform	[0.95, 1.05]	[0.95, 1.05]
External Forces	Additive	Refer to [1]	–	–
Restitution	Additive	uniform	[0.0, 0.4]	[0.0, 0.4]
Observation				
Obj. Pose Delay Prob.	Set Value	uniform	[0.0, 0.05]	[0.0, 0.47]
Obj. Pose Freq.	Set Value	uniform	[1.0, 1.0]	[1.0, 6.0]
Obs Corr. Noise	Additive	gaussian	[0.0, 0.04]	[0.0, 0.12]
Obs Uncorr. Noise	Additive	gaussian	[0.0, 0.04]	[0.0, 0.14]
Random Pose Injection	Set Value	uniform	[0.3, 0.3]	[0.3, 0.3]
Action				
Action Delay Prob.	Set Value	uniform	[0.0, 0.05]	[0.0, 0.31]
Action Latency	Set Value	uniform	[0.0, 0.0]	[0.0, 1.5]
Action Corr. Noise	Additive	gaussian	[0.0, 0.04]	[0.0, 0.32]
Action Uncorr. Noise	Additive	gaussian	[0.0, 0.04]	[0.0, 0.48]
RNA α	Set Value	uniform	[0.0, 0.0]	[0.0, 0.16]
Environment				
Gravity (each coord.)	Additive	normal	[0, 0.5]	[0, 0.5]

Table 3: Domain randomisation parameter ranges for policy learning

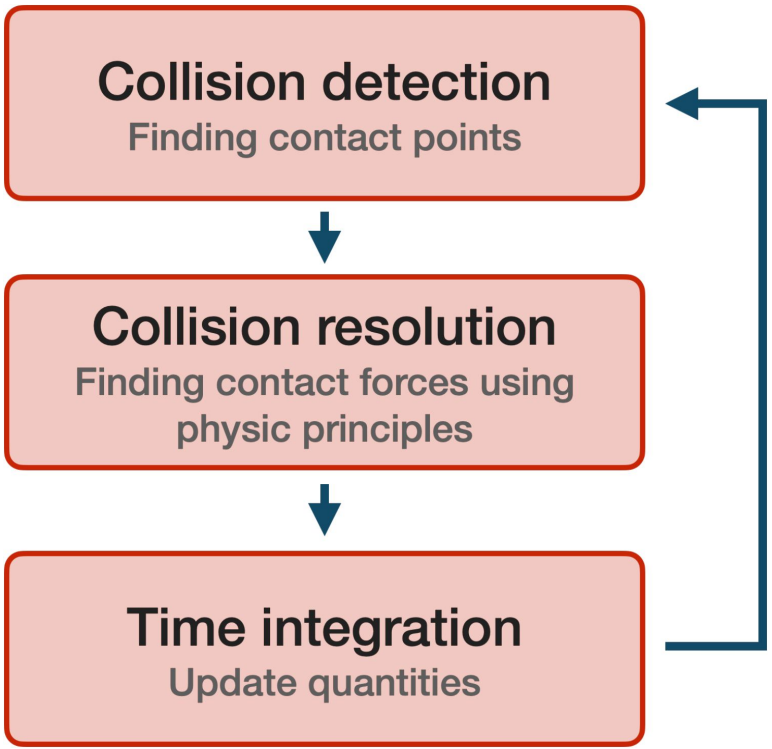


1 training: 60 hours with 16384 environments

Approx. 100e9 calls to the simulator ~ 55 years of simulation

1000$, 55 kg CO2e ~ 300km by car

What is a simulator ?



Main simulators in robotics

- Bullet (Google): computer graphics
- MuJoCo (Deepmind): invertible, RL
- Drake (TRI): invertible, robotics manipulation
- RaiSim (ETHZ): quadrupedal locomotion
- Isaac Gym (NVIDIA): GPU acceleration

These simulators have been designed for different purposes: they are not interchangeable in general !

Difference may come from both physical modeling and numerical implementations

Contact models in robotics: a comparative analysis

Contact Models in Robotics: a Comparative Analysis

Quentin Le Lidec[1,†], Wilson Jallet[1,2], Louis Montaut[1,3], Ivan Laptev[1], Cordelia Schmid[1], and Justin Carpentier[1]

***Abstract*—Physics simulation is ubiquitous in robotics. Whether in model-based approaches (*e.g.*, trajectory optimization), or model-free algorithms (*e.g.*, reinforcement learning), physics simulators are a central component of modern control pipelines in robotics. Over the past decades, several robotic simulators have been developed, each with dedicated contact modeling assumptions and algorithmic solutions. In this article, we survey the main contact models and the associated numerical methods commonly used in robotics for simulating advanced robot motions involving contact interactions. In particular, we recall the physical laws underlying contacts and friction (*i.e.*, Signorini condition, Coulomb's law, and the maximum dissipation principle), and how they are transcribed in current simulators. For each physics engine, we expose their inherent physical relaxations along with their limitations due to the numerical techniques employed. Based on our study, we propose theoretically grounded quantitative criteria on which we build benchmarks assessing both the physical and computational aspects of simulation. We support our work with an open-source and efficient C++ implementation of the existing algorithmic variations. Our results demonstrate that some approximations or algorithms commonly used in robotics can severely widen the reality gap and impact target applications. We hope this work will help motivate the development of new contact models, contact solvers, and robotic simulators in general, at the root of recent progress in motion generation in robotics.**

***Index Terms*—Physical simulation, Numerical optimization.**

I. Introduction

SIMULATION is a fundamental tool in robotics. Control algorithms, like trajectory optimization (TO) or model predictive control (MPC) algorithms, rely on physics simulators to evaluate the dynamics of the controlled system. Reinforcement Learning (RL) algorithms operate by trial and error and require a simulator to avoid time-consuming and costly failures on real hardware. Robot co-design aims at finding optimal hardware design and morphology and thus extensively rely on simulation to prevent tedious physical validation. In practice, roboticists also usually perform simulated safety checks before running a new controller on their robots. These applications are evidence for a wide range of research areas in robotics where simulation is critical.

To be effective and valuable in practice, robot simulators must meet some fidelity or efficiency levels, depending on the use case. For instance, trajectory optimization algorithms, *e.g.* iLQR[1] or DDP [2], [3], use physics simulation to evaluate the system dynamics and leverage finite differences or the recent advent of differentiable simulators [4], [5], [6], [7], [8] to compute derivatives. If the solution lacks precision, the real and planned trajectories may quickly diverge, impacting *de facto* the capacity of such control solutions to be deployed on real hardware. To absorb such errors, the Model Predictive Control (MPC) [9], [10] control paradigm exploits state feedback by repeatedly running Optimal Control (OC) algorithms at high-frequency rates (e.g., 1kHz) [11], [12]. The frequency rate is one factor determining the robustness of this closed-loop algorithm to modeling errors and perturbations; thus, the efficiency of the simulation becomes critical. Although RL [13] is considered as a model-free approach, physical models are still at work to generate the samples that are indispensable for learning control policies. In fact, the vast number of required samples is the main bottleneck during training, as days or years of simulation, which corresponds to billions of calls to a simulator, are necessary [14], [15], [16]. Therefore, the efficiency of the simulator directly determines

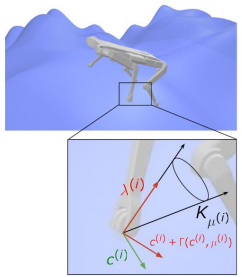


Fig. 1. **Illustration of the dynamics of frictional contacts** between rigid bodies which are governed by the Signorini condition, the Coulomb's law, and the maximum dissipation principle. The combination of these three principles leads to the Non-linear Complementarity Problem (14).

[1]Inria - Département d'Informatique de l'École normale supérieure, PSL Research University. Email: `first.name.lastname@inria.fr`
[2]LAAS-CNRS, 7 av. du Colonel Roche, 31400 Toulouse
[3]Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University, Prague, Czech Republic
[†]Corresponding author

	Signorini	Coulomb	MDP	No shift	No internal forces	Robust	Convergence guarantees
LCP							
PGS [30], [29], [60], [31]	✓			✓			✓
Staggered projections [34]	✓			✓	✓	✓	
CCP							
PGS [61]		✓	✓	✓			✓
MuJoCo [32]		✓	✓		✓	✓	✓
ADMM (Alg. 3)		✓	✓	✓	✓	✓	✓
RaiSim [33]	✓	✓		✓			
NCP							
PGS	✓	✓	✓	✓			✓
Staggered projections [6]	✓	✓	✓	✓	✓	✓	

Modeling the physics of contacts

Free motion with generalized coordinates

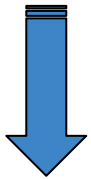
Lagrangian equations of motion:

$$M(q)\dot{v} + b(q, v) = \tau$$

Free motion with generalized coordinates

Lagrangian equations of motion:

$$M(q)\dot{v} + b(q, v) = \tau$$



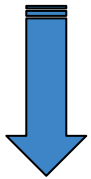
Discrete version (time-stepping methods):

$$Mv^{t+1} = Mv^{t} + (\tau - b)\Delta t$$

Free motion with generalized coordinates

Lagrangian equations of motion:

$$M(q)\dot{v} + b(q, v) = \tau$$



Discrete version (time-stepping methods):

terms are evaluated explicitly at q^t, v^t

$$Mv^{t+1} = Mv^t + (\tau - b)\Delta t$$

Contact modelling hypothesis

Contacts in modern simulators are modelled via 4 elementary principles:

- Rigid bodies
- Unilateral contacts
- Coulomb’s law of friction
- Maximum Dissipation Principle

Contact modelling hypothesis

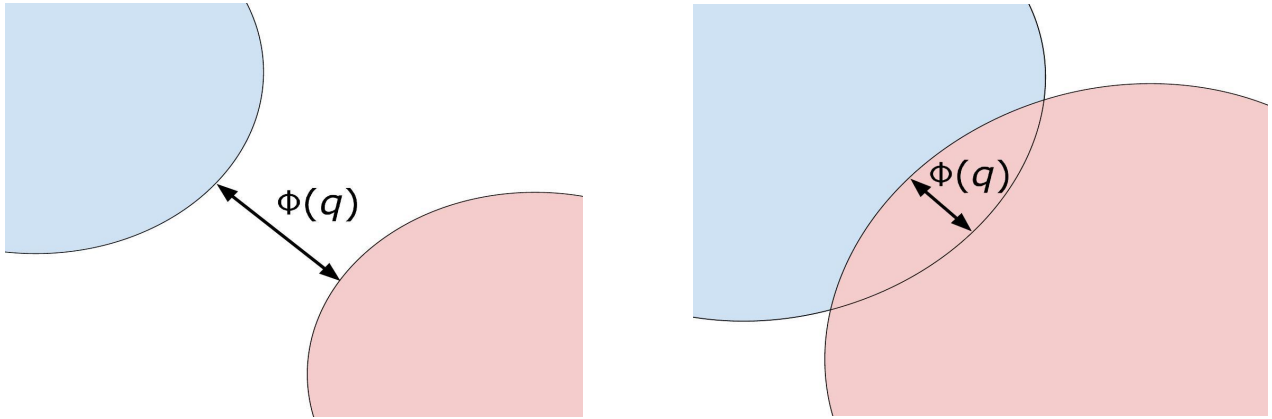
Contacts in modern simulators are modelled via 4 elementary principles:

- Rigid bodies
- Unilateral contacts

} Signorini condition

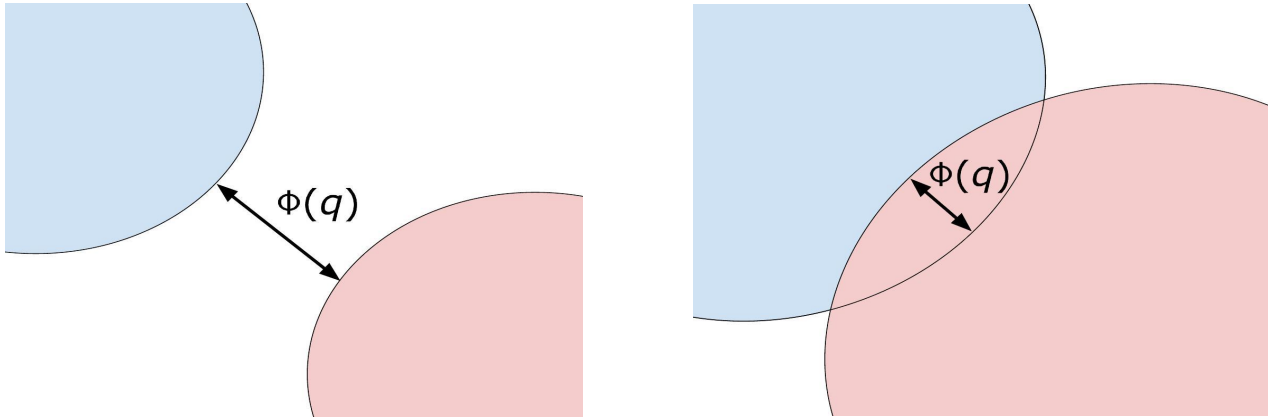
- Coulomb’s law of friction
- Maximum Dissipation Principle

Modelling unilateral contacts



No interpenetration: $\Phi(q)_N \geq 0$

Modelling unilateral contacts

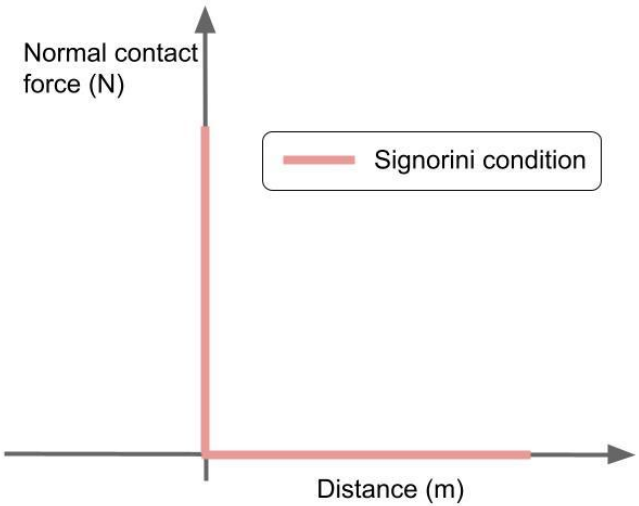


No interpenetration: $\Phi(q)_N \geq 0 \Rightarrow c_N - c_N^* \geq 0$

where: $c = Jv^{t+1}$ and $J = \partial\Phi/\partial q$

Modelling unilateral contacts

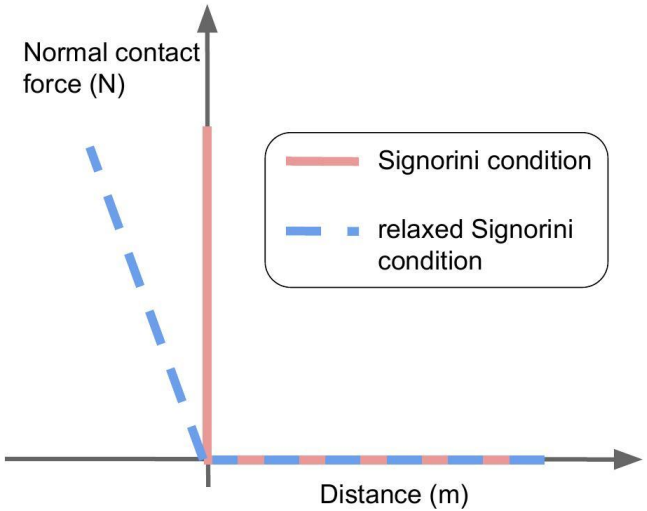
Signorini condition: $0 \leq \lambda_N \perp c_N - c_N^* \geq 0$



Modelling compliant unilateral contacts

Relaxed Signorini condition:

$$0 \leq \lambda_N \perp c_N - c_N^* + R_N \lambda_N \geq 0$$

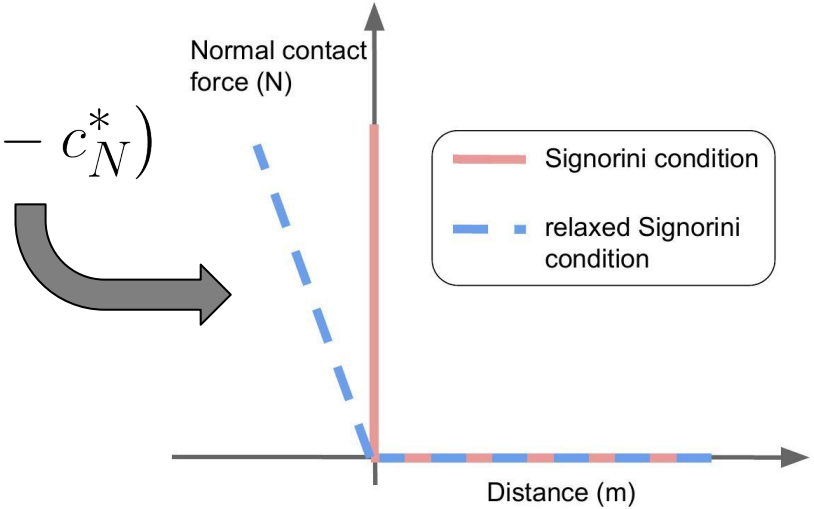


Modelling compliant unilateral contacts

Relaxed Signorini condition:

$$0 \leq \lambda_N \perp c_N - c_N^* + R_N \lambda_N \geq 0$$

$$\lambda_N = -R_N^{-1} \left(c_N - c_N^* \right)$$

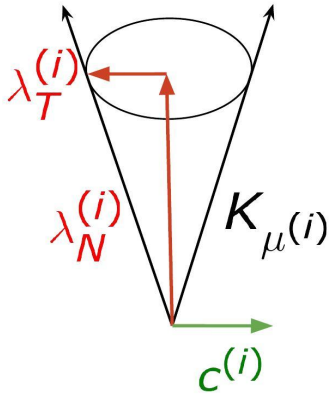


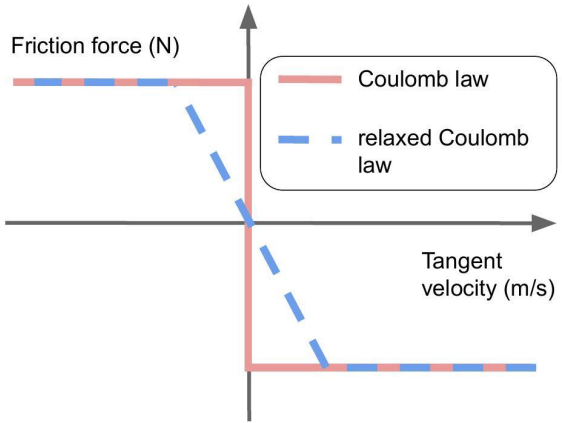
Modelling frictions

Coulomb's law of friction: $\lambda \in K_\mu = \prod_{i=1}^{n_c} K_{\mu^{(i)}}$

$$K_{\mu^{(i)}} = \left\{ \lambda | \lambda_N \geq 0, \|\lambda_T\|_2 \leq \mu^{(i)} \lambda_N \right\}$$

Maximum dissipation principle: $\forall i,\ \lambda_T^{(i)} = -\mu^{(i)}\, \lambda_N^{(i)} \frac{c_T^{(i)}}{\|c_T\|}, \text{if } \|c_T^{(i)}\| > 0$





The frictional contact problem

Eventually, simulating contacts and frictions requires to solve:

$$\begin{cases} \lambda^{(i)} \in K_{\mu^{(i)}}, \text{if } c^{(i)} = 0 \\ \lambda^{(i)} = 0, \text{if } c_N^{(i)} > 0 \\ \lambda^{(i)} \in \partial K_{\mu^{(i)}}, \exists \alpha > 0, \lambda_T^{(i)} = -\alpha c_T^{(i)} \text{ otherwise.} \end{cases}$$

where: $c = G\lambda + g$, $G = JM^{-1}J^\top$ and $g = Jv^f$

The frictional contact problem

Eventually, simulating contacts and frictions requires to solve:

$$\begin{cases} \lambda^{(i)} \in K_{\mu^{(i)}}, \text{if } c^{(i)} = 0 \\ \lambda^{(i)} = 0, \text{if } c_N^{(i)} > 0 \\ \lambda^{(i)} \in \partial K_{\mu^{(i)}}, \exists \alpha > 0, \lambda_T^{(i)} = -\alpha c_T^{(i)} \text{ otherwise.} \end{cases}$$

➡ Sticking

where: $c = G\lambda + g$, $G = JM^{-1}J^\top$ and $g = Jv^f$

The frictional contact problem

Eventually, simulating contacts and frictions requires to solve:

$$\begin{cases} \lambda^{(i)} \in K_{\mu^{(i)}}, \text{if } c^{(i)} = 0 \Rightarrow \text{Sticking} \\ \lambda^{(i)} = 0, \text{if } c_N^{(i)} > 0 \Rightarrow \text{Breaking} \\ \lambda^{(i)} \in \partial K_{\mu^{(i)}}, \exists \alpha > 0, \lambda_T^{(i)} = -\alpha c_T^{(i)} \text{ otherwise.} \end{cases}$$

where: $c = G\lambda + g$, $G = JM^{-1}J^\top$ and $g = Jv^f$

The frictional contact problem

Eventually, simulating contacts and frictions requires to solve:

$$\begin{cases} \lambda^{(i)} \in K_{\mu^{(i)}}, \text{if } c^{(i)} = 0 & \Rightarrow \text{Sticking} \\ \lambda^{(i)} = 0, \text{if } c_N^{(i)} > 0 & \Rightarrow \text{Breaking} \\ \lambda^{(i)} \in \partial K_{\mu^{(i)}}, \exists \alpha > 0, \lambda_T^{(i)} = -\alpha c_T^{(i)} \text{ otherwise.} & \Rightarrow \text{Slipping} \end{cases}$$

where: $c = G\lambda + g$, $G = JM^{-1}J^\top$ and $g = Jv^f$

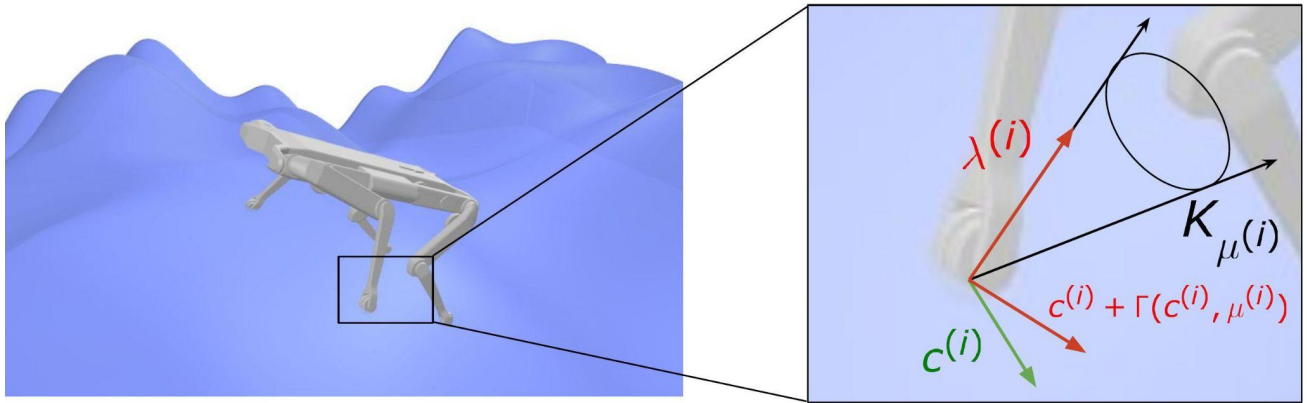
The frictional contact problem

Simulating contacts and frictions requires to solve:

$$\forall i, K_{\mu^{(i)}} \ni \lambda^{(i)} \perp c^{(i)} + \Gamma\left(c^{(i)}, \mu^{(i)}\right) \in K^*_{\mu^{(i)}}$$

$$c = G\lambda + g$$

where the de Saxcé correction is: $\Gamma : (c, \mu) \in \mathbb{R}^3 \times \mathbb{R} \mapsto [0, 0, \mu \|c_T\|_2]$



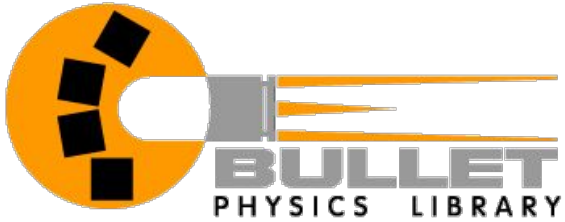
Algorithmic variations of the contact problem: the simulators

How simulators differ

Two main sources of difference:

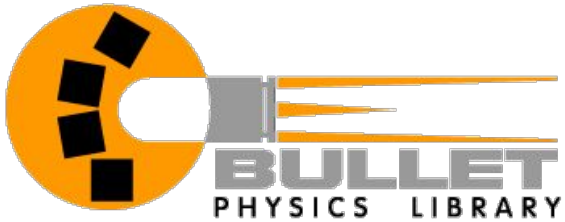
- Physical assumptions: NCP can be relaxed to make it more tractable
- Choice of numerical algorithm: PGS, Newton etc.

How does it impact robotics simulation ?



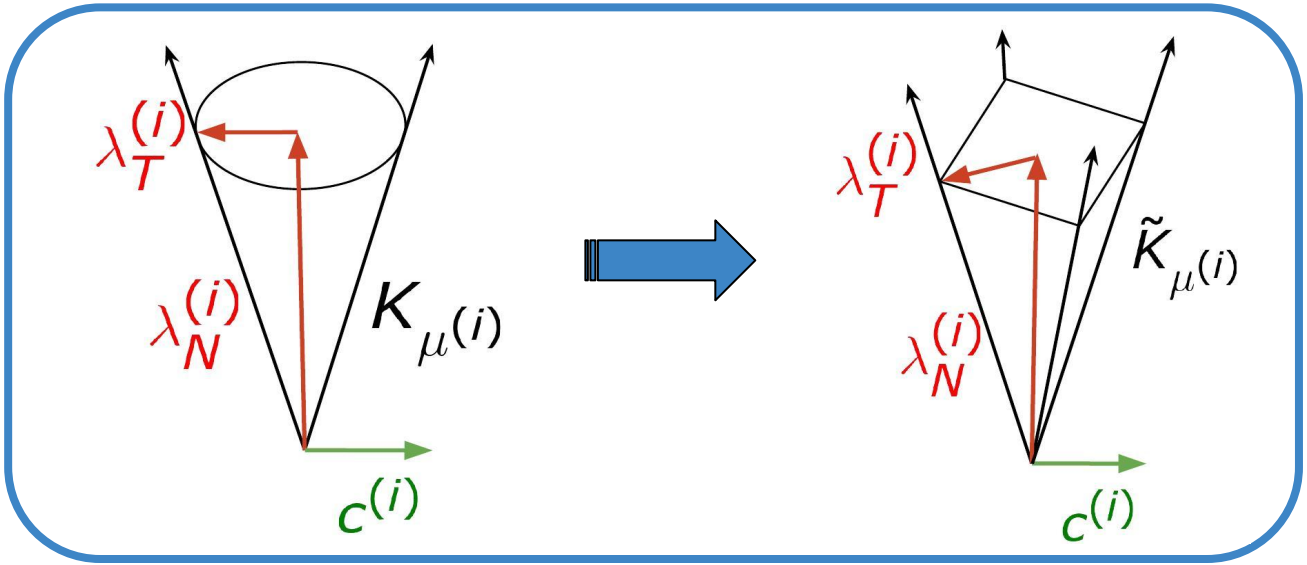
Developed by Erwin Coumans (previously Google, now NVIDIA) it was primarily designed for graphics (video games) but still popular in robotics

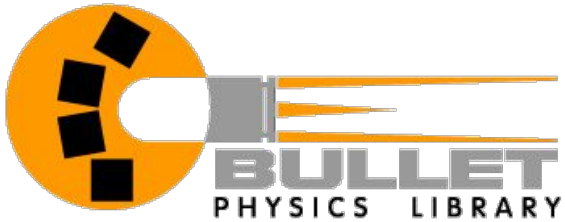
Motivations: Fast and open source physics engine



The second order cone is approximated by its linearization:

$$\tilde{K}_{\mu^{(i)}} = \left\{ \lambda | \lambda_N \geq 0, \|\lambda_T\|_\infty \leq \mu^{(i)} \lambda_N \right\}$$





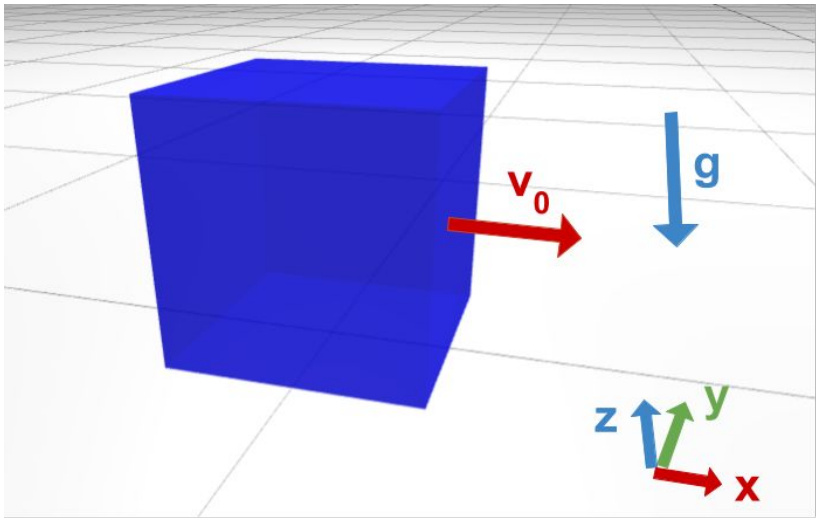
The problem can be solved numerically by a Projected Gauss Seidel algorithm:

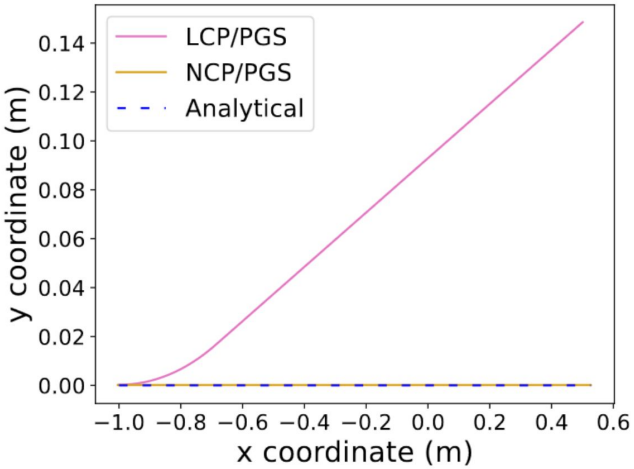
Algorithm 1: Pseudo-code of the projected Gauss-Seidel (PGS) algorithm for solving LCPs.

Input: Delassus matrix: G, free velocity: g, friction cones: K_μ

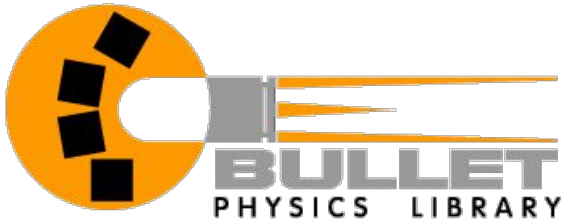
Output: Contact forces: λ

for $k = 1$ **to** n_{iter} **do**
  for $i = 1$ **to** n_c **do**
    $\lambda_N^{(i)} \leftarrow \lambda_N^{(i)} - \frac{1}{G_{NN}^{(ii)}} (G\lambda + g)_N^{(i)}$;
    $\lambda_N^{(i)} \leftarrow \max(0, \lambda_N^{(i)})$;
    $\lambda_T^{(i)} \leftarrow \lambda_T^{(i)} - \frac{1}{\min(G_{T_x T_x}^{(ii)}, G_{T_y T_y}^{(ii)})} (G\lambda + g)_T^{(i)}$;
    $\lambda_T^{(i)} \leftarrow \text{clamp}(\lambda_T^{(i)}, \mu_i \lambda_N)$;
  end
end



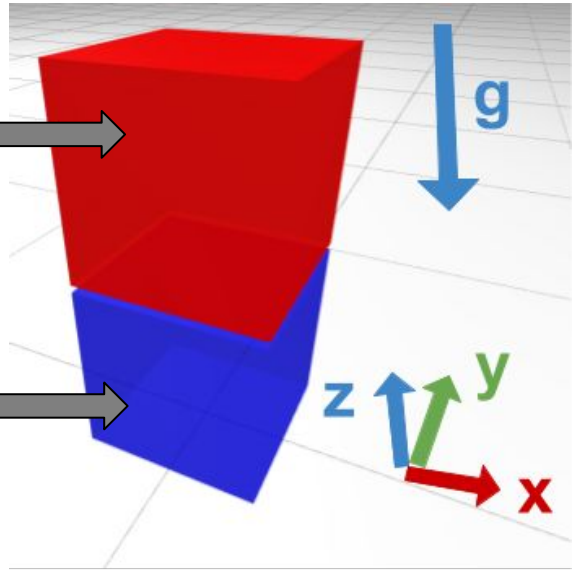


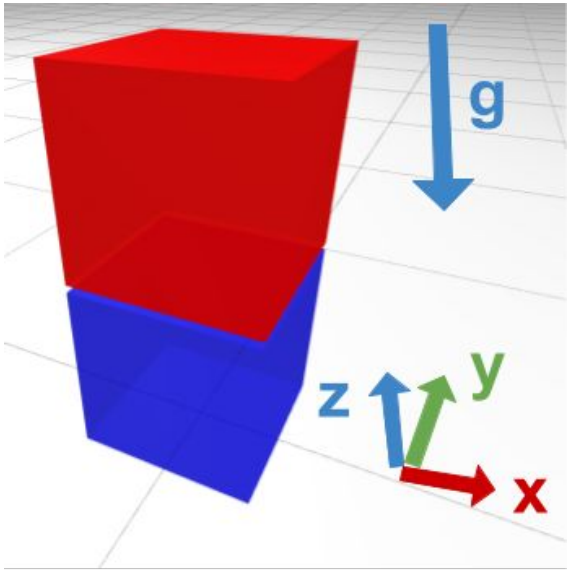
Pyramidal approximation leads to non-physical behaviour

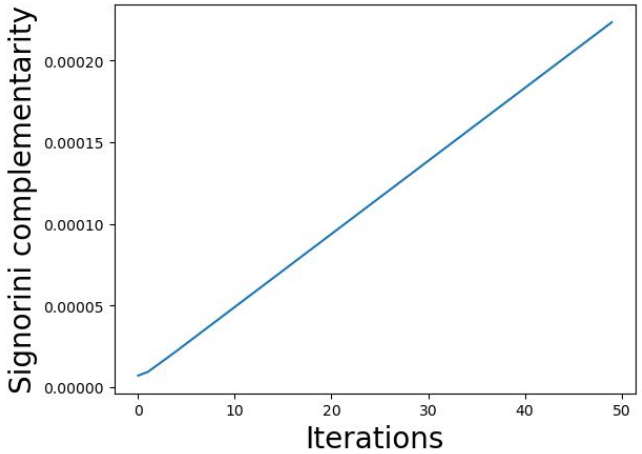


heavy cube (1e3 kg)

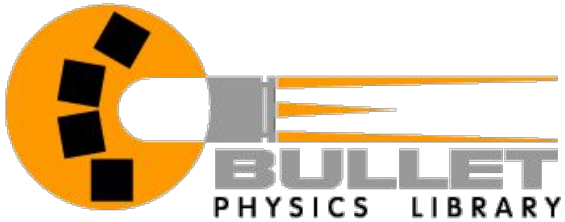
light cube (1e-3 kg)







PGS is not robust to numerical ill-conditioning

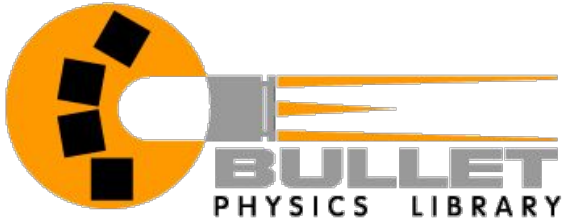


(+) Advantages:

- For coarse simulation, PGS is fast

(-) Drawbacks:

- Linearization of the cone tends to bias forces towards the corners which induces violation of the “original” MDP
- PGS is a first-order per-contact approach making it fail on ill-conditioned cases



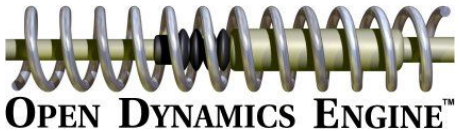
(+) Advantages:

- For coarse simulation, PGS is fast

(-) Drawbacks:

- Linearization of the cone tends to bias forces towards the corners which induces violation of the “original” MDP
- PGS is a first-order per-contact approach making it fail on ill-conditioned cases

This approach is used by many simulators:



MuJoCo

Originally developed by Emo Todorov and now by Deepmind, general purpose physics simulator. It became a standard benchmark in RL community.

Motivations: strictly convex problem for stable numerics and invertible dynamics

MuJoCo

Reminder: Non-linear complementarity problem

$$\forall i, K_{\mu^{(i)}} \ni \lambda^{(i)} \perp c^{(i)} + \Gamma\left(c^{(i)}, \mu^{(i)}\right) \in K^*_{\mu^{(i)}}$$

$$c = G\lambda + g$$

MuJoCo

Reminder: Non-linear complementarity problem

$$\forall i, K_{\mu^{(i)}} \ni \lambda^{(i)} \perp c^{(i)} + \Gamma\left(c^{(i)}, \mu^{(i)}\right) \in K^*_{\mu^{(i)}}$$

$$c = G\lambda + g$$

MuJoCo

The complementarity is relaxed:

$$K_\mu \ni \lambda \perp c \in K_\mu^*$$

$$c = G\lambda + g$$

Conic complementarity writes:

$$\lambda^\top c = 0$$

MuJoCo

The complementarity is relaxed and we get a Conic Complementarity Problem (CCP):

$$K_\mu \ni \lambda \perp c \in K_\mu^*$$
$$c = G\lambda + g$$

Conic complementarity writes:

$$\lambda_N^\top c_N = -\lambda_T^\top c_T \neq 0$$

MuJoCo

The complementarity is relaxed and we get a Conic Complementarity Problem (CCP):

$$K_\mu \ni \lambda \perp c \in K_\mu^*$$

$$c = G\lambda + g$$

Conic complementarity writes:

Signorini condition is violated for sliding contacts !

$$\lambda_N^\top c_N = -\lambda_T^\top c_T \neq 0$$

MuJoCo

CCP is equivalent to a QCQP (it exactly corresponds to KKT optimality conditions):

$$\min_{\lambda \in K_\mu} \frac{1}{2}\lambda^\top G\lambda + g^\top \lambda$$

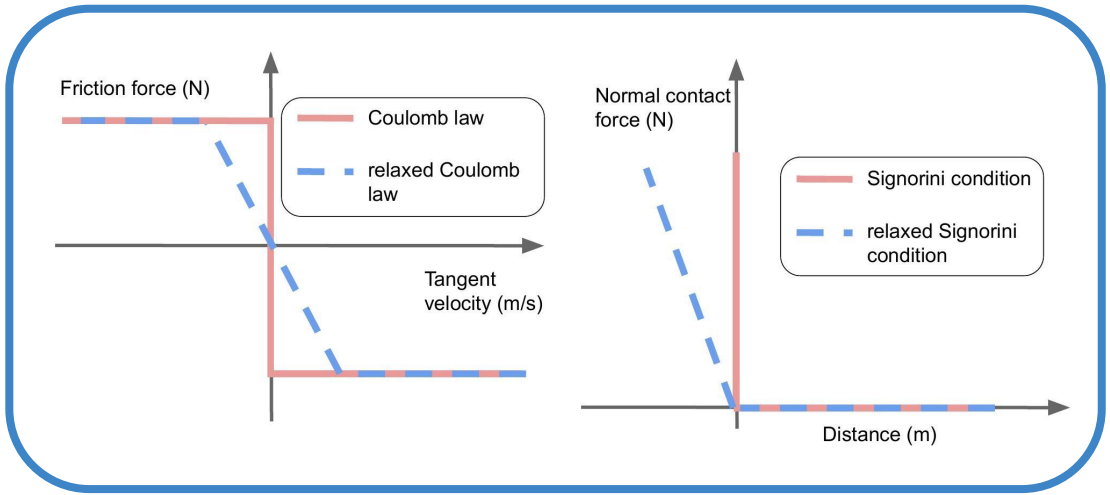
Any optimization algorithm can be used !

In particular, Newton-like algorithms leads to improved convergence rates

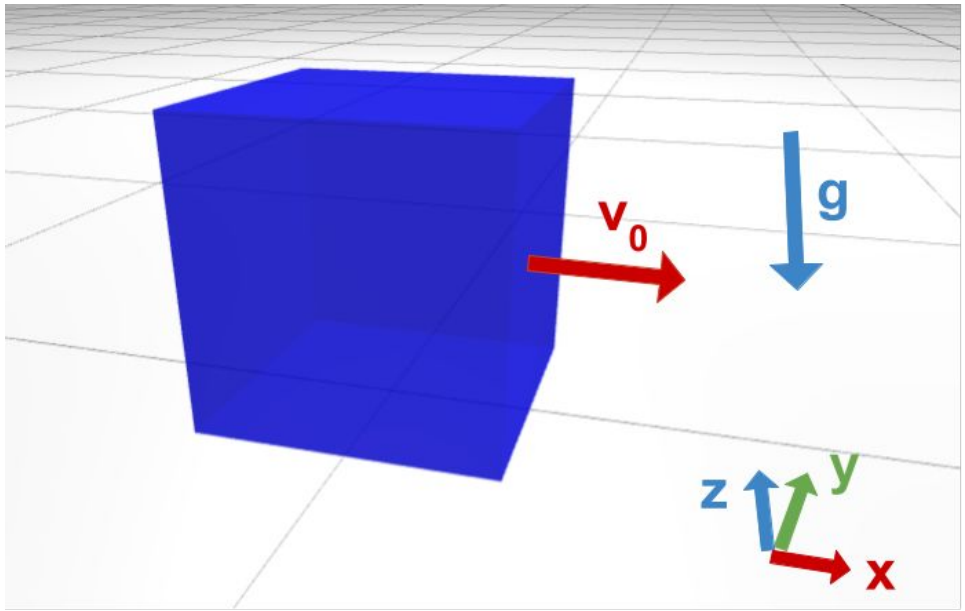
MuJoCo

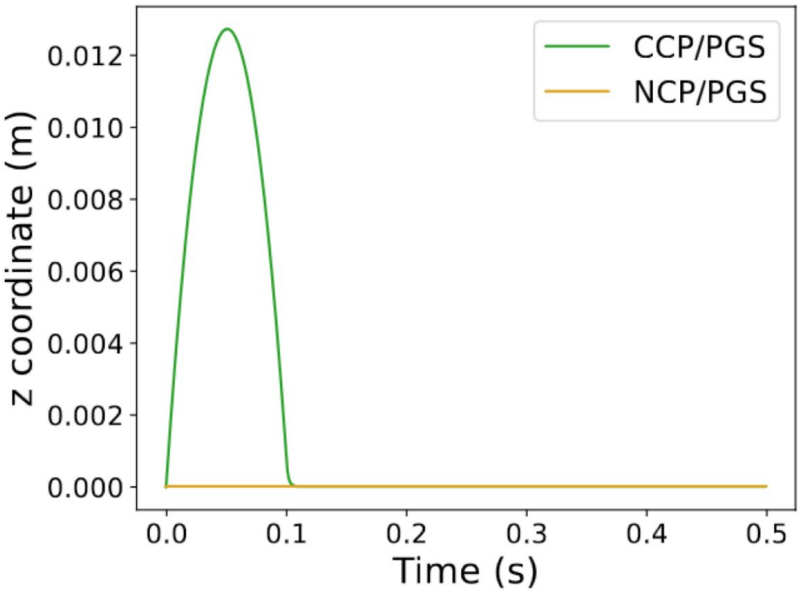
MuJoCo additionally introduces an artificial compliance

$$G \Rightarrow \tilde{G} = G + R$$



MuJoCo


v0
g
z
y
x


CCP/PGS
NCP/PGS
z coordinate (m)
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0.0
0.1
0.2
0.3
0.4
0.5
Time (s)

MuJoCo

(+) Advantages:

- Solving a convex optimization problem makes it robust

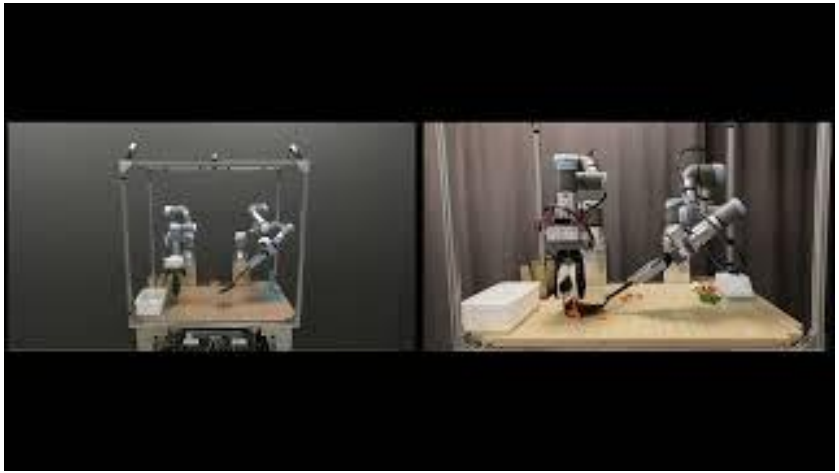
(-) Drawbacks:

- Signorini condition is relaxed which leads to “magic” forces
- For numerical reasons, materials are made artificially compliant

DRAKE

Developed by Toyota Research Institute (TRI), it is a complete model-based toolbox for robots control (simulation + control)

Motivations: more realistic contacts for robotics



DRAKE

(+) Advantages:

- Newton algorithm can deal with ill-conditioning
- Sparse algebra backend improves efficiency
- Compliance is set to get more realistic behaviours than MuJoCo

(-) Drawbacks:

- Strictly rigid contacts cannot be handled
- Signorini condition is relaxed which leads to “magic” forces for sliding contacts

RaiSim

Developed by Jemin Hwangbo (previously at ETHZ and now at KAIST)

Motivations: simulator dedicated to robotics and, in particular, to quadrupedal locomotion

Reminder: CCP is equivalent to a QCQP:

$$\min_{\lambda \in K_\mu} \frac{1}{2}\lambda^\top G\lambda + g^\top \lambda$$

Its solution violates the Signorini condition for sliding contacts:

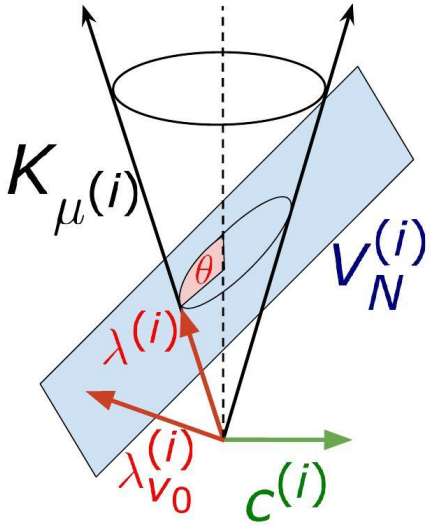
$$\lambda_N^\top c_N = -\lambda_T^\top c_T \neq 0$$

RaiSim

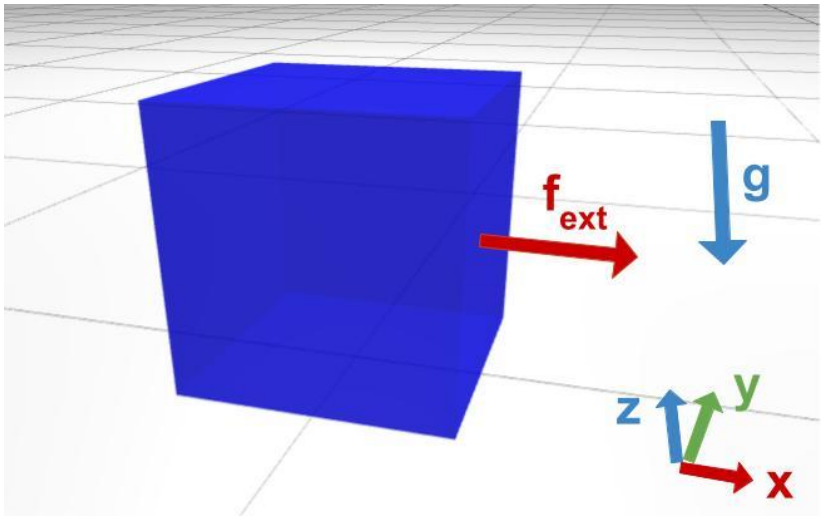
RaiSim is a Gauss-Seidel-like approach which aims at correcting drawbacks from the CCP of MuJoCo by enforcing the Signorini condition :

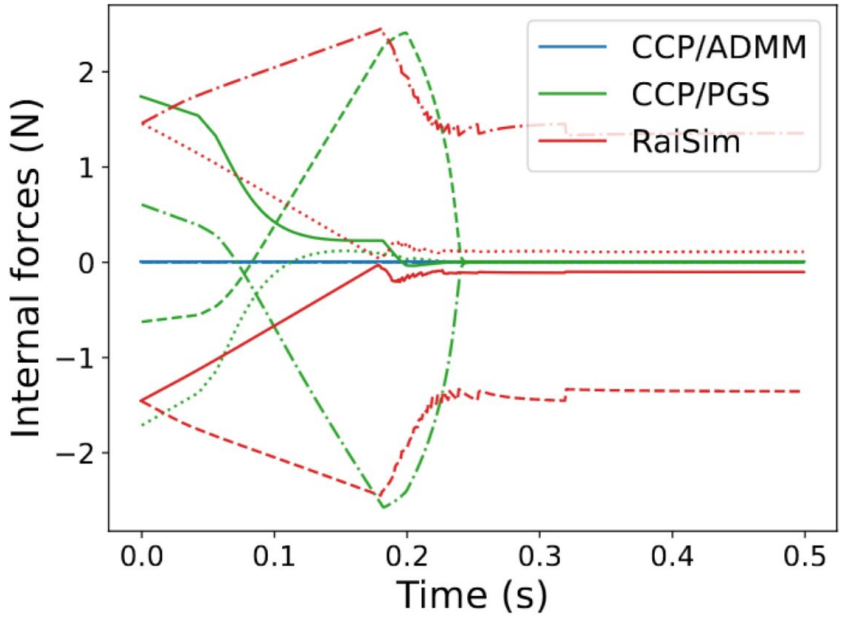
$$\min_{\lambda \in K_{\mu(i)} \cap V_N^{(i)}} \frac{1}{2}\lambda^\top G^{(i)}\lambda + \tilde{g}^{(i)\top}\lambda$$

where: $V_N^{(i)} = \{\lambda | \underbrace{G_N^{(i)}\lambda + g_N^{(i)}}_{c_N^{(i)}} = 0\}$



RaiSim


f ext
g
z
y
x


CCP/ADMM
CCP/PGS
RaiSim
Internal forces (N)
Time (s)
2
1
0
−1
−2
0.0
0.1
0.2
0.3
0.4
0.5

(+) Advantages:

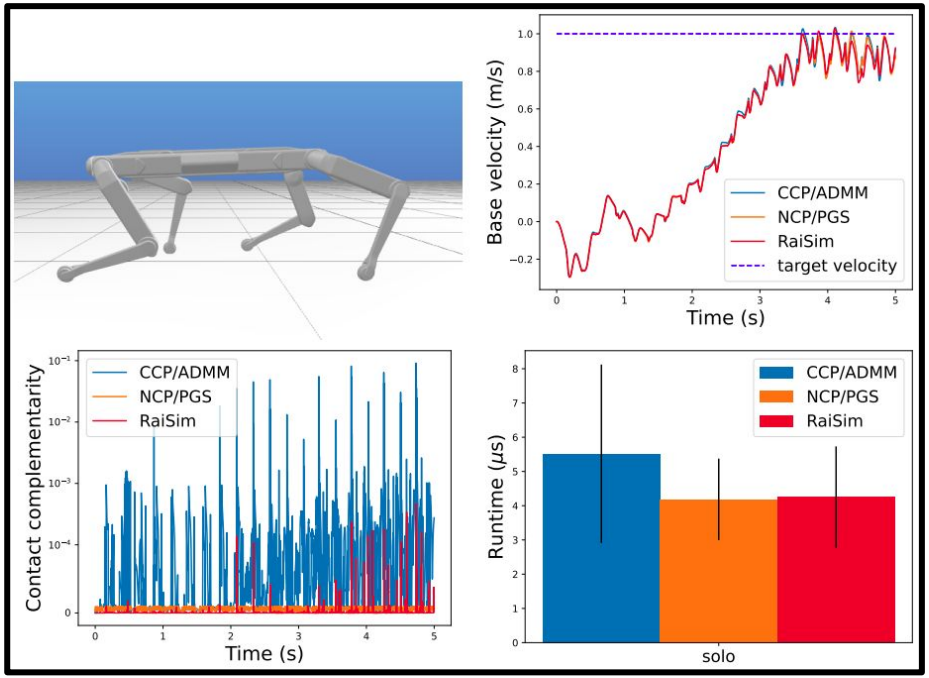
- efficient for low accuracy requirements

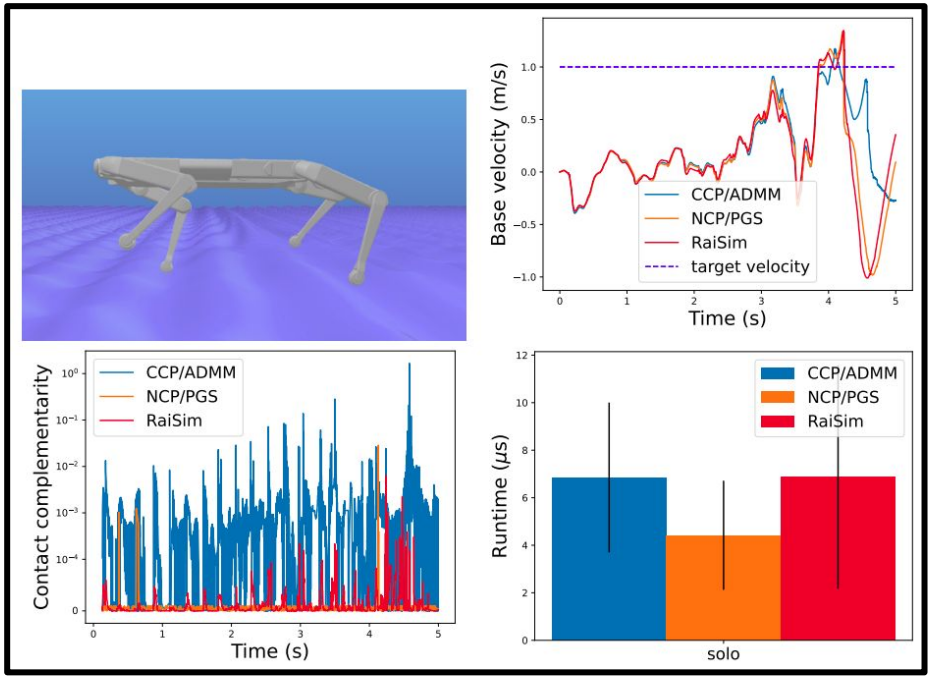
(-) Drawbacks:

- the per-contact approach induces issues for ill-conditioned problems
- the proposed correction retrieves the Signorini condition but loses the MDP

How simulation impacts robotics applications

Impact on locomotion with MPC





Conclusion

- Simulating rigid bodies with contacts and frictions requires to solve a NCP
- The NCP is relaxed to make it easier to solve, thus inducing physical artifacts
- Numerical implementations may also induce additional artifacts
- These artifacts can affect downstream applications (e.g. reality gap issues)