Robotic Systems II

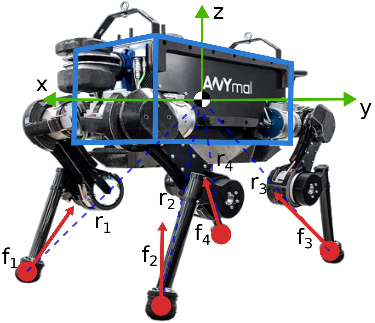
Lecture 10: Optimal Control for Complex Robots

Konstantinos Chatzilygeroudis - costashatz@upatras.gr

Department of Electrical and Computer Engineering
University of Patras

Template made by Panagiotis Papagiannopoulos

Single Rigid Body Dynamics (SRBD) Model



Key Ideas of the SRBD model:

- We assume that the feet are massless, but can push on the ground!
- We only care about the movement of the root body!

SRBD Details

In the SRBD model:

- The rigid body has a mass $m \in \mathbb{R}^+$ and moment of inertia $\boldsymbol{I} \in \mathbb{R}^{3\times 3}$

$$\boldsymbol{x}_{\text{body}} = \begin{bmatrix} \boldsymbol{p}_w \in \mathbb{R}^3 \\ \dot{\boldsymbol{p}}_w \in \mathbb{R}^3 \\ \boldsymbol{R}_w \in \mathcal{SO}(3) \\ \boldsymbol{\omega}_b \in \mathbb{R}^3 \end{bmatrix}$$

- Each leg i has a position $\boldsymbol{p}_i \in \mathbb{R}^3$, it can generate contact forces $\boldsymbol{f}_i \in \mathbb{R}^3$

$$\ddot{\boldsymbol{p}}_w = \frac{\sum_i \boldsymbol{f}_i + m\,\boldsymbol{g}}{m}$$
$$\dot{\boldsymbol{\omega}}_b = \boldsymbol{I}^{-1}(\boldsymbol{R}_w^T(\sum_i(\boldsymbol{p}_i - \boldsymbol{p}_w) \times \boldsymbol{f}_i) - \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega})$$

SRBD Details (2)

Let's write it cleaner:

$$\ddot{\boldsymbol{p}}_w = \frac{\boldsymbol{f}_{\text{total}}}{m}$$
$$\dot{\boldsymbol{\omega}}_b = \boldsymbol{I}^{-1}(\boldsymbol{R}_w^T \boldsymbol{\tau}_{\text{total}} - \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega})$$

where

$$\boldsymbol{f}_{\text{total}} = \sum_i \boldsymbol{f}_i + m\boldsymbol{g}$$
$$\boldsymbol{\tau}_{\text{total}} = \sum_i \boldsymbol{r}_i \times \boldsymbol{f}_i$$
$$\boldsymbol{r}_i = \boldsymbol{p}_i - \boldsymbol{p}_w$$

Now we can integrate via RK4, Semi-Implicit Euler or any integrator that we like!

SRBD Details (3)

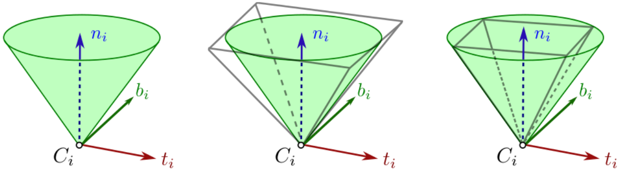
Remember: We need to have constraints for the forces!

$$(\boldsymbol{f} \cdot \boldsymbol{n}) > 0, \text{ the force cannot pull/grasp!}$$

$$\|\boldsymbol{f} \cdot \boldsymbol{t}\|_2 \leqslant \mu(\boldsymbol{f} \cdot \boldsymbol{n})$$

$$\|\boldsymbol{f} \cdot \boldsymbol{b}\|_2 \leqslant \mu(\boldsymbol{f} \cdot \boldsymbol{n})$$

These are "Conic" constraints! We usually linearize them to help the solvers:



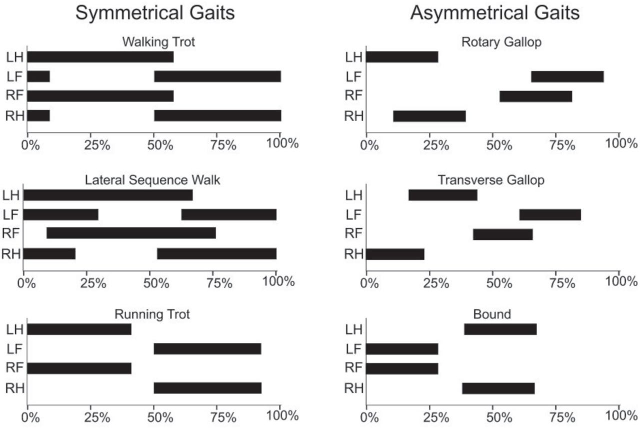
From `https://scaron.info/`

$$(\boldsymbol{f} \cdot \boldsymbol{n}) > 0, \text{ the force cannot pull/grasp!}$$

$$|\boldsymbol{f} \cdot \boldsymbol{t}| \leqslant \mu(\boldsymbol{f} \cdot \boldsymbol{n}) \text{ or } |\boldsymbol{f} \cdot \boldsymbol{t}| \leqslant \frac{\mu}{\sqrt{2}}(\boldsymbol{f} \cdot \boldsymbol{n})$$

$$|\boldsymbol{f} \cdot \boldsymbol{b}| \leqslant \mu(\boldsymbol{f} \cdot \boldsymbol{n}) \text{ or } |\boldsymbol{f} \cdot \boldsymbol{b}| \leqslant \frac{\mu}{\sqrt{2}}(\boldsymbol{f} \cdot \boldsymbol{n})$$

But the feet are not moving!



Trajectory Optimization with the SRBD model

- We have K knot points. For each knot point k and for each foot i, we **select beforehand** if the foot is in contact or not!
- We have the following optimization variables:

$$\boldsymbol{\theta}_k = \begin{bmatrix} \boldsymbol{p}_k & \dot{\boldsymbol{p}}_k & \boldsymbol{R}_k & \boldsymbol{\omega}_k & \boldsymbol{p}_{ik} & \boldsymbol{f}_{ik} \end{bmatrix}^T$$

- We interpolate each foot position $\boldsymbol{p}_i$ when in air! Or we can constrain their movement with respect to the body!
- How can we optimize/represent for each $\boldsymbol{R}_k$?

Trajectory Optimization with the SRBD model

- We have K knot points. For each knot point k and for each foot i, we **select beforehand** if the foot is in contact or not!
- We have the following optimization variables:

$$\boldsymbol{\theta}_k = \begin{bmatrix} \boldsymbol{p}_k & \dot{\boldsymbol{p}}_k & \boldsymbol{R}_k & \boldsymbol{\omega}_k & \boldsymbol{p}_{ik} & \boldsymbol{f}_{ik} \end{bmatrix}^T$$

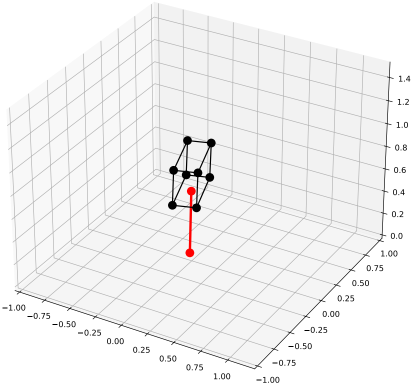
- We interpolate each foot position $\boldsymbol{p}_i$ when in air! Or we can constrain their movement with respect to the body!
- How can we optimize/represent for each $\boldsymbol{R}_k$?
- We now just have different $\oplus$, $\ominus$ operators! We also know their gradients!
- We can also use an axis-angle representation to reduce the optimization variables: $\boldsymbol{\theta}_k = \begin{bmatrix} \boldsymbol{p}_k & \dot{\boldsymbol{p}}_k & \phi_k & \boldsymbol{\omega}_k & \boldsymbol{p}_{ik} & \boldsymbol{f}_{ik} \end{bmatrix}^T$

Full Monopod Example

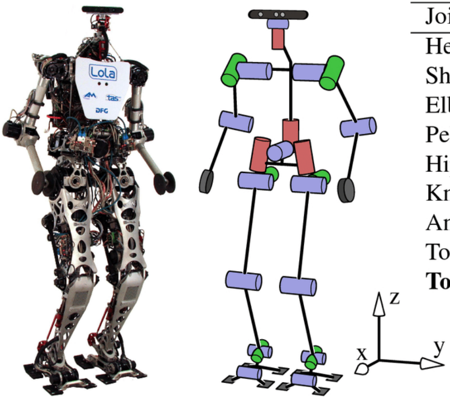
Let's make the problem a bit more complicated:

- One foot (Monopod!)
- We use orientation this time!
- Euler Integration (implicit)
- Total time: $2.5\,s$
- Gait Sequence:
  `Contact`$(0.5\,s)$, `Air`$(0.3\,s)$, `Contact`$(0.5\,s)$, `Air`$(0.2\,s)$, `Contact`$(1\,s)$
- We can choose K as we like!
- We could use splines instead of samples!

Full Monopod - Code Example



But our robots have many degrees of freedom!



Joint	DoF
Head	2
Shoulder	2
Elbow	1
Pelvis	2
Hip	3
Knee	1
Ankle	2
Toe	1
Total	**24**

From Hildebrandt, Arne-Christoph, et al. "Kinematic optimization for bipedal robots: a framework for real-time collision avoidance." Autonomous Robots 43 (2019): 1187-1205.

Manipulator Equation Reminder:

$$\underbrace{\boldsymbol{M}(\boldsymbol{q})}_{\text{“Mass Matrix”}} \dot{\boldsymbol{v}} + \underbrace{\boldsymbol{C}(\boldsymbol{q}, \boldsymbol{v})}_{\text{“Coriolis/Gravity Forces”}} = \boldsymbol{B}(\boldsymbol{q}) \underbrace{\boldsymbol{u}}_{\text{“Usually } \boldsymbol{\tau}\text{”}} + \underbrace{\boldsymbol{F}_{\text{ext}}}_{\text{“External forces”}}$$

Velocity Kinematics:

$$\dot{\boldsymbol{q}} = \boldsymbol{G}(\boldsymbol{q})\boldsymbol{v}$$

and thus,

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{G}(\boldsymbol{q})\boldsymbol{v} \\ \boldsymbol{M}^{-1}(\boldsymbol{q})\Big(\boldsymbol{B}(\boldsymbol{q})\boldsymbol{u} + \boldsymbol{F}_{\text{ext}} - \boldsymbol{C}(\boldsymbol{q}, \boldsymbol{v})\Big) \end{bmatrix}$$

We usually refer to $\boldsymbol{q}$ as the “Generalized Reduced Coordinates” of the system.

- Easy: we set $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{v} \end{bmatrix}$ and proceed as normal

- Easy: we set $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{v} \end{bmatrix}$ and proceed as normal

- Not so fast! The acceleration constraints are of the form:

$$\boldsymbol{g} = \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{q}, \boldsymbol{v}) - \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\tau} = 0$$

- We need to take the derivatives of this! How?

- Easy: we set $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{v} \end{bmatrix}$ and proceed as normal

- Not so fast! The acceleration constraints are of the form:

$$\boldsymbol{g} = \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{q}, \boldsymbol{v}) - \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\tau} = 0$$

- We need to take the derivatives of this! How?

- **Luckily,** there are tools that we can use and get the derivatives of the whole expression (e.g. Pinocchio)! This is based on the Featherstone's algorithm.

- So we have access to terms:

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}}, \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{v}}, \frac{\partial \boldsymbol{g}}{\partial \dot{\boldsymbol{v}}}, \dots$$

Generalized Reduced Coordinates and Contacts?

- Easy: we set $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{v} \end{bmatrix}$ and proceed as normal
- What happens if we have contacts?

Generalized Reduced Coordinates and Contacts?

- Easy: we set $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{v} \end{bmatrix}$ and proceed as normal
- What happens if we have contacts?

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{q}, \boldsymbol{v}) = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\tau} + \boldsymbol{J}^T \boldsymbol{F}_{\text{contacts}}$$

- But the **jump map** and **end effector constraints** operate in 3D points/space!
- How can we enforce the jump to the generalized coordinates?

Jump Map in Generalized Reduced Coordinates

- The jump map is usually a **position constraint**, i.e. $j(\boldsymbol{q}) = 0$
- Let's take the time derivatives of this:

$$\dot{j} = \boldsymbol{H}\boldsymbol{v}$$
$$\ddot{j} = \dot{\boldsymbol{H}}\boldsymbol{v} + \boldsymbol{H}\dot{\boldsymbol{v}}$$

  where $\boldsymbol{H} = \frac{\partial j}{\partial \boldsymbol{q}}$.

- And what can we do with this?

Jump Map in Generalized Reduced Coordinates

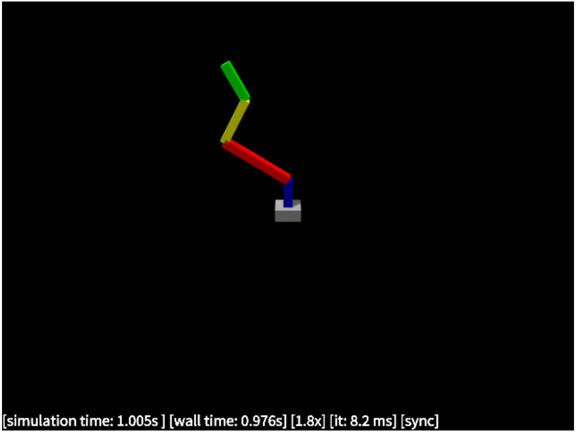
- The jump map is usually a **position constraint**, i.e. $j(\boldsymbol{q}) = 0$
- Let's take the time derivatives of this:

$$\dot{j} = \boldsymbol{H}\boldsymbol{v}$$
$$\ddot{j} = \dot{\boldsymbol{H}}\boldsymbol{v} + \boldsymbol{H}\dot{\boldsymbol{v}}$$

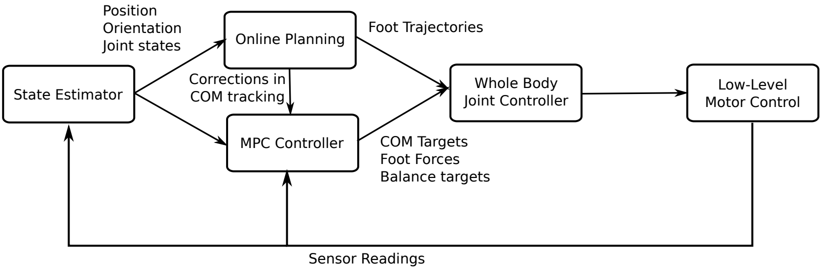
  where $\boldsymbol{H} = \frac{\partial j}{\partial \boldsymbol{q}}$.

- And what can we do with this?
- For the jump maps that we care about: $\boldsymbol{H} \equiv \boldsymbol{J}$ (Jacobian at the contact point)!!
- So now we can add the constraints: $\dot{\boldsymbol{J}}\boldsymbol{v} + \boldsymbol{J}\dot{\boldsymbol{v}} = \boldsymbol{0}$ at the "jump" points!
- **Baumgarte's stabilization technique**: $\dot{\boldsymbol{J}}\boldsymbol{v} + \boldsymbol{J}\dot{\boldsymbol{v}} = -2\alpha\dot{j} + \alpha^2 j$

4DoF Arm - Code Example



So how can we control a complex real robot?



Final Verdict

- Optimal Control is a strong framework for generating and controlling robots
- Now you can control your own robots
- But, it is not enough! We need to combine it with robust state estimation and low-level controllers
- When you can, linearize!
- Implementing MPC on real hardware is a challenge on its own!
- Hot research topics: combine optimal control with learning and Reinforcement Learning!

Thank you

- **Any Questions?**
- **Office Hours**:
  - **Tue & Thu** *(09:00-11:00)*
  - 24/7 by email (`costashatz@upatras.gr`, subject: *ECE_RSII_AM*)
- **Material and Announcements**

Laboratory of Automation & Robotics