# Underactuated Robots
# Architectures, Whole-Body Control, MPC

Nicola Scianca

December 2025

# outline

- **General architecture**: layers, interfaces, time scales
- **Whole-Body Control (WBC)**: tasks + constraints $\rightarrow$ whole-body QP
- **Model Predictive Control (MPC)**: TO vs MPC (conceptually), receding horizon, example

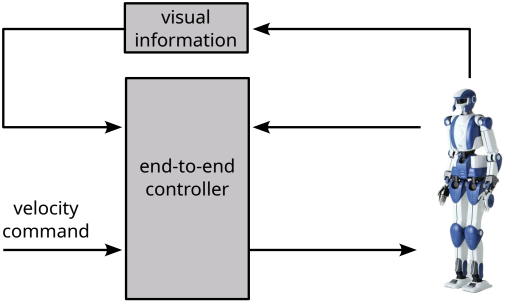
# architectures

- to generate the commands necessary to achieve these motions we can use different architectures

- let us take a look at some examples, before exploring in depth a specific architecture
  - **end-to-end** architectures
  - architectures that use **whole-body predictive controllers**
  - architectures that perform predictive control with a **simplified model**

# end-to-end architectures

- end-to-end architectures are typically **data-driven** (e.g., based on **reinforcement learning** or **imitation learning**)

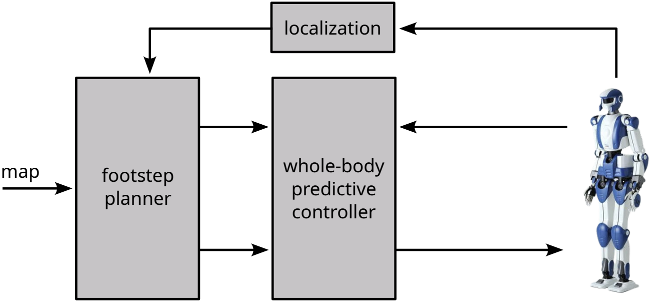


# end-to-end architectures

- the strength of these techniques lies in the fact that they can directly use **visual information**, e.g., coming from an RGB camera, to perceive the environment
- challenges include the **sim-to-real gap** usually found in learning-based controllers, and the fact that performing a different task usually requires **retraining**
- as of now, these approaches work well on **quadrupeds** but are more challenging on humanoids

# whole-body predictive controllers

- some architectures are based on perform predictive control on the **whole-body model** of the robot

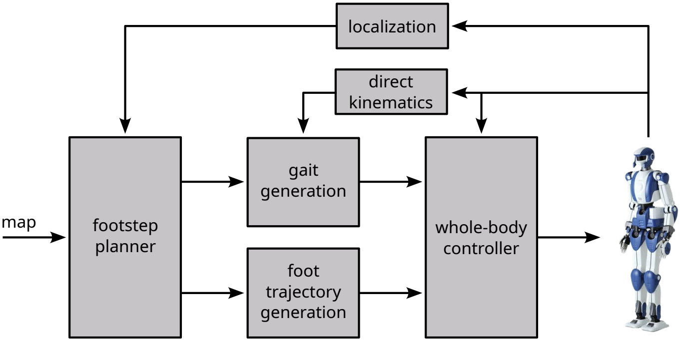


# whole-body predictive controllers

- these techniques are potentially capable of performing very **dynamic** motions, such as running and jumping
- they require **heavy computations**, and often a good **initial guess**, to avoid the optimization getting stuck in a bad **local minimum**
- long-term goals must be planned using a separate technique (e.g., a footstep planner to reach a goal) and some form of localization

# gait generation with a simplified model

- a common approach is to have a separate controller optimize the trajectory using a **simplified model**, that can then be tracked by a whole-body controller



# gait generation with a simplified model

- the range of applicability of these architectures is limited by the range of validity of the assumptions used to derive the **simplified model**
- for achieving locomotion they can be quite effective, and with proper choice of the simplified model they even allow jumping and running
- let us now analyze more in detail a typical architecture of this kind, starting from the whole-body controller

# humanoid robot control architectures

- in this lecture we will focus on **hierarchical** architectures based on **model predictive control** (MPC)
- MPC is used to optimize a trajectory for a **simplified model** of the dynamics
- a whole-body controller is used to generate commands for the robot

# general architecture

- **state estimation** → robot state, contacts
- **planner** → discrete decisions (contacts/steps/goals)
- **predictive layer** (often MPC) → short-horizon **references**
- **whole-body controller (WBC)** → feasible **torques/forces**
- **low-level actuation** → motor control

# interfaces between layers

- planner provides: contact schedule, targets, timings, bounds
- predictive layer provides: reference trajectories (CoM/feet/torso/momentum,. . . )
- WBC accepts references and outputs joint torques

# time scales (typical)

- planner: slow (Hz or less)
- predictive layer: medium (10–200 Hz)
- WBC: fast (500–1000 Hz)
- motor control: fastest (kHz)

# models at different layers

- planning/prediction: **simplified** models to keep optimization fast
- WBC: **full rigid-body dynamics** + contacts to guarantee feasibility
- key trade-off: computation vs physical accuracy

# Whole-Body Control (WBC)

- WBC coordinates all joints to realize multiple **tasks** while satisfying **physical constraints**
- goal: compute feasible commands (typically **joint torques** $\tau$) that
  - track desired motions (CoM, feet, hands, torso, ...)
  - respect contact consistency and contact feasibility

# floating-base dynamics

$$M(q)\dot{\nu} + n(q,\nu) = S^\top \tau + \sum_{i=1}^{n_c} J_{c,i}^\top(q)\, w_i$$

- $q = (q_{\mathrm{fb}}, q_{\mathrm{act}})$: floating base + actuated joints
- $S$: selection matrix (picks actuated DoFs), $\tau$: joint torques
- $J_{c,i}$: Jacobian of contact $i$, $w_i$: contact wrench (force+torque)

# kinematic task

- a **kinematic task** specifies a desired behavior for a task variable

$y_j = \phi_j(q)$ (e.g., CoM position, foot pose, hand position, torso o

- its velocity and acceleration satisfy

$$\dot{y}_j = J_j(q)\dot{q}, \qquad \ddot{y}_j = J_j(q)\ddot{q} + \dot{J}_j(q,\dot{q})\dot{q}$$

where $J_j(q) = \frac{\partial \phi_j}{\partial q}$ is the task Jacobian

# kinematic task tracking

- choose a desired task acceleration (feedforward + PD)

$$\ddot{y}_j^d = \ddot{y}_j^{\mathrm{ff}} + k_p(y_j^d - y_j) + k_d(\dot{y}_j^d - \dot{y}_j)$$

- tracking can be written as a **linear expression in $\ddot{q}$**:

$$J_j(q)\ddot{q} + \dot{J}_j(q, \dot{q})\dot{q} \approx \ddot{y}_j^d$$

- in a QP we typically minimize the squared tracking error of this relation

# examples of kinematic tasks

- CoM tracking: $y = p_G(q) \in \mathbb{R}^3$
- swing foot pose: $y = {}^W T_{\text{foot}}(q)$ (position + orientation)
- torso orientation: $y = R_{\text{torso}}(q)$
- hand position/orientation: $y = {}^W T_{\text{hand}}(q)$
- posture regularization: $y = q_{\text{act}}$

all become constraints/objectives through $\ddot{y} = J\ddot{q} + \dot{J}\dot{q}$.

# Contact constraints: kinematic

- if contact $i$ is rigid (stance), the contact point/pose must not move

$$v_{c,i} = J_{c,i}(q)\dot{q} = 0$$

- enforcing it at the acceleration level gives a **linear constraint** in $\ddot{q}$:

$$a_{c,i} = J_{c,i}(q)\ddot{q} + \dot{J}_{c,i}(q,\dot{q})\dot{q} = 0$$

- we impose this for all active contacts $i = 1, \ldots, n_c$

# contact constraints: force feasibility

- contact wrench $w_i = (f_i, \tau_i)$ must be physically realizable
- typical conditions (polyhedral approximation):
  - **unilaterality**: $f_{z,i} \geq 0$
  - **friction cone**: $\sqrt{f_{x,i}^2 + f_{y,i}^2} \leq \mu f_{z,i}$ (linearized)
  - **no tilting / CoP bounds**: center of pressure inside support polygon
- compactly, we write them as linear inequalities:

$$A_i \, w_i \leq b_i$$

# linearized friction cone constraints

- replace the cone with a pyramid using $\bar{\mu} < \mu$:

$$-\bar{\mu} f_z \leq f_x \leq \bar{\mu} f_z, \qquad -\bar{\mu} f_z \leq f_y \leq \bar{\mu} f_z$$

- together with unilaterality $f_z \geq 0$, these become part of $A_i w_i \leq b_i$

# decision variables in WBC

- typical QP variables:

$$\ddot{q},\ \tau,\ w_1, \ldots, w_{n_c}$$

- sometimes we also add **slack variables** to soften low-priority tasks
- the QP must satisfy:
  - floating-base dynamics
  - contact kinematics
  - contact wrench feasibility
  - (optional) actuator bounds / rates

# whole-body QP

$$\min_{\ddot{q},\tau,w_i} \;\; \sum_{j=1}^{n_t} \alpha_j \left\| J_j(q)\ddot{q} + \dot{J}_j(q,\dot{q})\dot{q} - \ddot{y}_j^d \right\|^2 + \lambda_\tau \|\tau\|^2 + \lambda_w \sum_{i=1}^{n_c} \|w_i\|^2$$

$$\begin{aligned}
\text{s.t.} \;\; & M(q)\ddot{q} + n(q,\dot{q}) = S^\top \tau + \sum_{i=1}^{n_c} J_{c,i}^\top(q)\, w_i \\
& J_{c,i}(q)\ddot{q} + \dot{J}_{c,i}(q,\dot{q})\dot{q} = 0 \qquad i = 1,\ldots,n_c \\
& A_i w_i \le b_i \qquad i = 1,\ldots,n_c
\end{aligned}$$

# hard constraints vs soft objectives

- **hard constraints**: must always hold
  - dynamics, contact kinematics, wrench feasibility, actuator bounds
- **soft tasks**: tracked "as well as possible"

$$\text{minimize } \|J\ddot{q} + \dot{J}\dot{q} - \ddot{y}^d\|^2$$

- priorities can be encoded via weights $\alpha_j$ or strict hierarchical QPs

# interpretation of the solution

- the QP returns:
  - $\tau$: feasible torques to send to the robot
  - $w_i$: predicted contact wrenches (useful for monitoring feasibility margins)
  - $\ddot{q}$: internally consistent accelerations (optional for internal integration)
- if tasks conflict or are unachievable, the optimizer trades tracking errors using weights

# typical tasks for humanoid locomotion

- CoM tracking (balance and progression)
- stance foot: contact kinematics constraints
- swing foot tracking
- torso orientation
- angular momentum / centroidal regulation
- posture regularization (resolve redundancy)

# example: whole-body QP for walking

# TO vs control

- trajectory optimization (TO) is great for finding **reference trajectories**
- usually done **offline**, then executed via **trajectory tracking**
- issue: if the robot deviates significantly, the planned trajectory may be suboptimal or unsafe

# TO vs control

- MPC "looks like TO" but it runs **online**
- at every control cycle:
  - re-plan from the **measured current state**
  - apply only the **first action**
  - repeat at the next cycle (receding horizon)
- result: a controller that continuously adapts to disturbances and modeling errors

# TO vs control

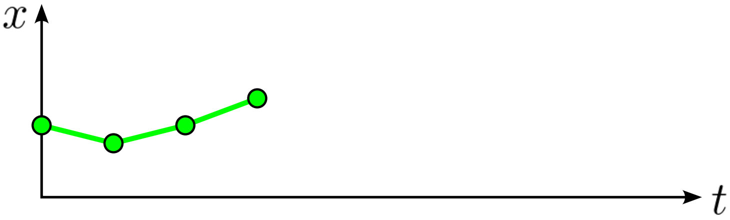
- the optimization is always initialized at the **current measured state**
- so any mismatch between prediction and reality gets corrected at the next solve
- MPC trades “solve once offline” for “solve small problems repeatedly online”

# MPC practical challenges

- MPC must run fast (often around 100 Hz or more)
- therefore we usually:
  - shorten the horizon (optimize only the immediate future)
  - simplify the model (capture what matters most)
  - warm-start from the previous solution (avoid big jumps)

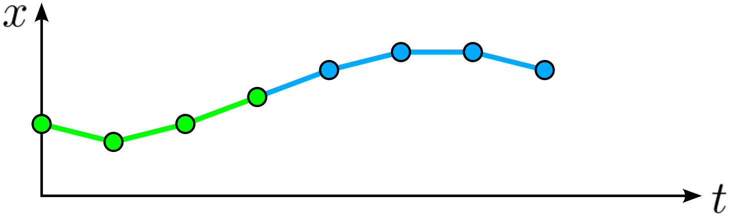
# MPC example (1/4)

- in green: the **realized trajectory** up to now



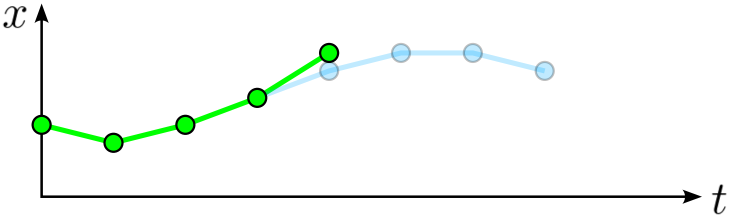
# MPC example (2/4)

- we **predict** an optimal trajectory from the current state
- we apply the **first** input of the predicted sequence



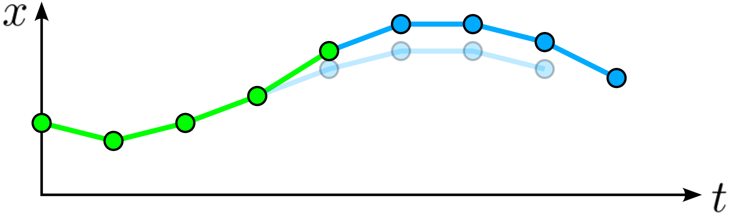
# MPC example (3/4)

- the new state differs from prediction due to:
  - model mismatch
  - disturbances



# MPC example (4/4)

- re-solve from the **new measured state**
- this repeated correction is what makes MPC robust in practice



# MPC in an architecture (high-level)

- MPC provides **short-horizon references** (e.g., CoM/foot targets, velocities)
- WBC tracks those references while enforcing full-body feasibility
- planner handles longer-horizon decisions (contacts/steps/goals)