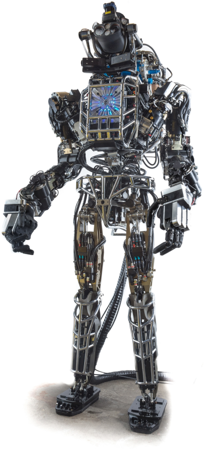
Autonomous and Mobile Robots: Architectures and Whole-Body Control

Nicola Scianca

December 2024

outline

- linear inverted pendulum model
- divergent component of motion
- legged locomotion
- architectures
- contact wrenches
- whole-body control



linear inverted pendulum

- the **linear inverted pendulum** (LIP) dynamics is

$$\boxed{\ddot{\boldsymbol{p}}_c^{x/y} = \eta^2(\boldsymbol{p}_c^{x/y} - \boldsymbol{p}_z^{x/y})} \qquad \eta = \sqrt{\frac{g}{h}}$$

- let us now study the main features of this model

linear inverted pendulum

- the LIP model is decoupled along x and y, so we can refer to a generic component p to lighten the notation

$$\ddot{p}_c = \eta^2 (p_c - p_z)$$

- we must identify state $\boldsymbol{x}$, input $\boldsymbol{u}$ and output $\boldsymbol{y}$ for this system; one possible choice is

$$\boldsymbol{x} = \begin{pmatrix} p_c \\ \dot{p}_c \end{pmatrix}, \quad \boldsymbol{u} = p_z, \quad \boldsymbol{y} = p_c$$

- now we can write the system in **state-space** form

$$\frac{d}{dt}\begin{pmatrix} p_c \\ \dot{p}_c \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \eta^2 & 0 \end{pmatrix}\begin{pmatrix} p_c \\ \dot{p}_c \end{pmatrix} + \begin{pmatrix} 0 \\ -\eta^2 \end{pmatrix} p_z$$

divergent component of motion

- the LIP has two eigenvalues: $\pm\eta$
- we can perform a **change of coordinates** to decouple the dynamics associated with these eigenvalues

$$\begin{pmatrix} p_s \\ p_u \end{pmatrix} = \begin{pmatrix} 1 & -1/\eta \\ 1 & 1/\eta \end{pmatrix} \begin{pmatrix} p_c \\ \dot{p}_c \end{pmatrix}$$

- we call p_s the one that will be associated with $-\eta$ (stable) and p_u the one that will be associated the $+\eta$ (unstable)

divergent component of motion

- to perform this change of coordinates, we first have to identify the inverse transformation

$$\begin{pmatrix} p_c \\ \dot{p}_c \end{pmatrix} = \begin{pmatrix} 1 & -1/\eta \\ 1 & 1/\eta \end{pmatrix}^{-1} \begin{pmatrix} p_s \\ p_u \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1/\eta & 1/\eta \\ -1 & 1 \end{pmatrix} \begin{pmatrix} p_s \\ p_u \end{pmatrix}$$

- we can substitute it inside the state-space dynamics

$$\frac{1}{2} \begin{pmatrix} 1/\eta & 1/\eta \\ -1 & 1 \end{pmatrix} \frac{d}{dt} \begin{pmatrix} p_s \\ p_u \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \eta^2 & 0 \end{pmatrix} \frac{1}{2} \begin{pmatrix} 1/\eta & 1/\eta \\ -1 & 1 \end{pmatrix} \begin{pmatrix} p_s \\ p_u \end{pmatrix} + \begin{pmatrix} 0 \\ -\eta^2 \end{pmatrix} p_z$$

$$\frac{d}{dt} \begin{pmatrix} p_s \\ p_u \end{pmatrix} = \begin{pmatrix} -\eta & 0 \\ 0 & \eta \end{pmatrix} \begin{pmatrix} p_s \\ p_u \end{pmatrix} + \begin{pmatrix} \eta \\ -\eta \end{pmatrix} p_z$$

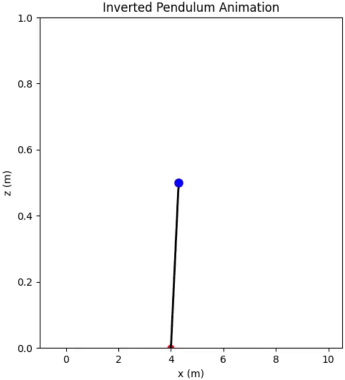
divergent component of motion

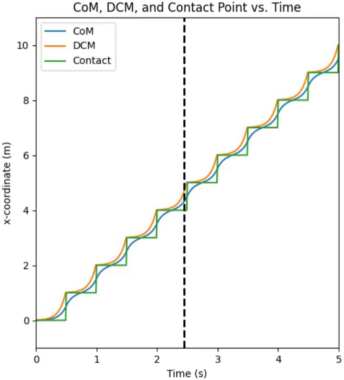
- the decoupled dynamics are

$$\dot{p}_s = \eta(p_z - p_s)$$
$$\dot{p}_u = \eta(p_u - p_z)$$

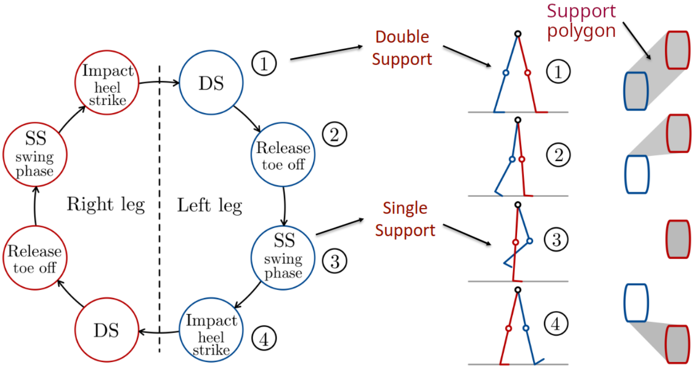
- the stable dynamics p_s is **attracted** to the ZMP
- the unstable dynamics p_u is **pushed away** from the ZMP
- in the literature, the latter is often called the **Divergent Component of Motion** (DCM): many methods focus on making sure the DCM doesn't diverge

example trajectories generated with a LIP





gait phases



- human walking typically consists of the cyclic alternation of **four phases**
- robots typically have flat non-flexible feet, and are usually limited to **single support** and **double support** phases

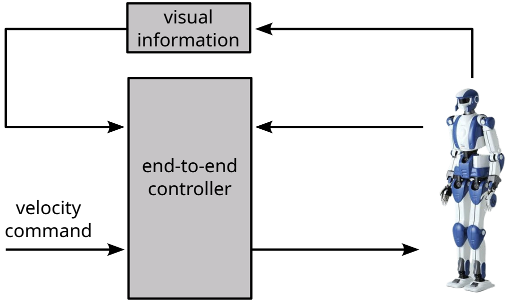
architectures

- 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**)

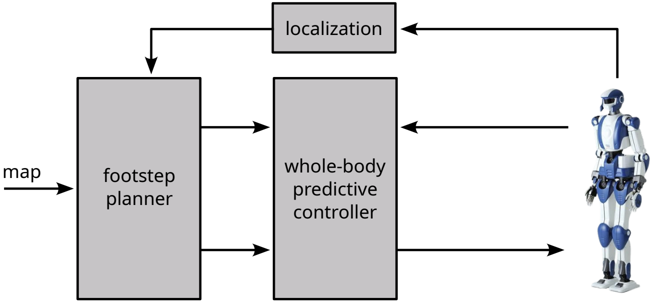


end-to-end architectures

- 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

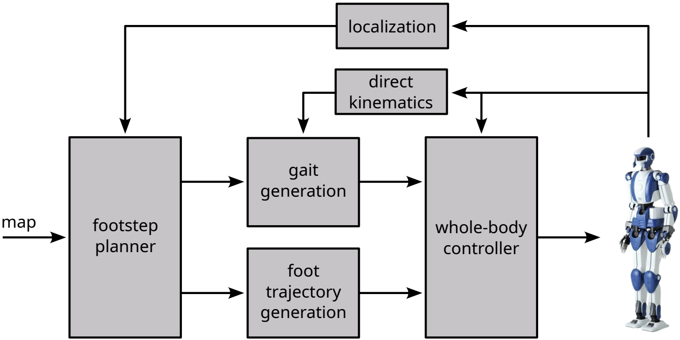
- these techniques are potentially capable of performing very **dynamic** motions, such as running and jumping

- 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

whole-body control

- **whole-body control** aims to coordinate all joints of the robot to achieve desired motions while satisfying physical constraints
- the desired motions are specified as a set of **tasks**, e.g., trajectory of the Center of Mass (CoM), the feet, the hands, etc..., that may be provided by other modules
- our goal is to find **joint torques** that realize these tasks, while also keeping the robot balanced

whole-body control

- the control torques $\boldsymbol{\tau}$ will be determined by a **constrained optimization** problem, formulated as a **Quadratic Program** (QP), i.e., a quadratic function subject to linear constraints

- a **cost function** will encode the **tasks** that we want to achieve

- we will enforce **constraints** to ensure that the contact forces are **feasible**

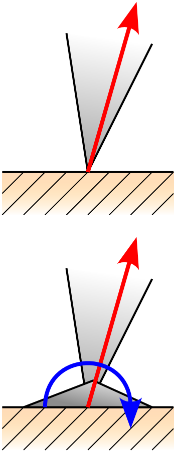
- QPs can be solved very efficiently using **off-the-shelf libraries**

contact forces

- a point contact can only apply a **force** $\boldsymbol{f}$, because it only constraint the contact position

- a surface contact also applies a **torque** $\boldsymbol{\tau}$, because it also constrains the orientation of the contact surface

- the stack of the corresponding force and torque $\boldsymbol{w} = (\boldsymbol{f}, \boldsymbol{\tau})$ is called the **contact wrench**



contact wrench constraints

- the requirements [Caron et al., 2015] that a contact wrench must satisfy are
  - **unilaterality**: the force must push **away** from the contact surface
  - **no slipping**: if we want to avoid relative motion of the contact surfaces, we must ensure sufficient **friction**
  - **no tilting**: if we want to maintain a surface contact, we must impose constraints on the **ZMP**

Caron et al.,"Stability of surface contacts for humanoid robots: Closed-form formulae of the Contact Wrench Cone for rectangular support areas", ICRA, 2015

unilaterality constraint

- the **unilaterality** constraint is relatively straightforward: we must enforce that
$$\boldsymbol{f}_z \geq 0$$

- we can write this as a constraint on the contact wrench as
$$\boldsymbol{A}\boldsymbol{w} \leq \boldsymbol{b}$$
with
$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 0 \end{pmatrix}$$

static friction

- to avoid slipping we must ensure that the contact has **sufficient friction**
- Coulomb's **static friction** model tells us that no slipping occurs as long as the vertical contact force f_z and the tangential contact force $f_t = \sqrt{f_x^2 + f_y^2}$ satisfy

$$|f_t| \leq \mu f_z \quad \Longrightarrow \quad \sqrt{f_x^2 + f_y^2} \leq \mu f_z$$

where μ is the friction coefficient, dependent on the materials

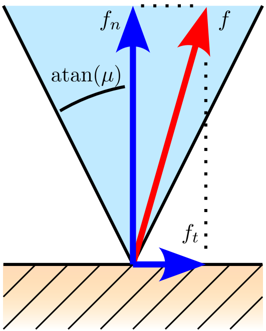
static friction

- this condition can be rewritten as

$$\left|\frac{f_t}{f_z}\right| \leq \mu$$

- there is a geometric interpretation: the **angle** between the vertical and tangential component of the force must be less than $\text{atan}(\mu)$

- the range of allowed forces defines a **friction cone**



static friction constraint

- to make it linear, we approximate it using a smaller $\bar{\mu} < \mu$

$$
\begin{aligned}
-\bar{\mu} f_z \leq f_x \leq \bar{\mu} f_z \\
-\bar{\mu} f_z \leq f_y \leq \bar{\mu} f_z
\end{aligned}
$$

- on the contact wrench, we can write it as $\boldsymbol{A}\boldsymbol{w} \leq \boldsymbol{b}$, with

$$
\boldsymbol{A} = \begin{pmatrix} 1 & 0 & -\bar{\mu} & 0 & 0 & 0 \\ -1 & 0 & -\bar{\mu} & 0 & 0 & 0 \\ 0 & 1 & -\bar{\mu} & 0 & 0 & 0 \\ 0 & -1 & -\bar{\mu} & 0 & 0 & 0 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
$$

contact wrench constraints

- there is a final condition that requires that $\tau_{\min} \leq \tau_z \leq \tau_{\max}$, with

$$\tau_{\min} = -\bar{\mu}(d_x + d_y) f_z + |d_y f_x - \bar{\mu}\tau_x| + |d_x f_y - \bar{\mu}\tau_y|$$
$$\tau_{\max} = +\bar{\mu}(d_x + d_y) f_z - |d_y f_x + \bar{\mu}\tau_x| - |d_x f_y + \bar{\mu}\tau_y|$$

- this condition looks more complex, but it can be written as a set of linear constraints on the contact wrench (see Caron et al.)

ZMP constraint

- now we must ensure that the foot does not tilt: this can be done by imposing a constraint on the **ZMP**
- we saw that a **ZMP outside** the contact surface is equivalent to **infeasible** contact forces
- therefore, we must keep the ZMP of each contact within the contact surface

ZMP constraints

- compute the horizontal moments with respect to a generic point on the ground p_o

$$f_z(x_f - x_o) - \tau_y = M_o$$
$$f_z(y_f - y_o) + \tau_x = M_o$$

- if p_o is a ZMP, by definition $M_o = 0$

$$f_z(x_f - x_z) - \tau_y = 0$$
$$f_z(y_f - y_z) + \tau_x = 0$$

- therefore, the position of the ZMP is

$$x_z - x_f = -\frac{\tau_y}{f_z}$$
$$y_z - y_f = \frac{\tau_x}{f_z}$$

ZMP constraints

- if the foot has a rectangular shape, with size $(2d_x, 2d_y)$, to keep the ZMP within the foot surface, we impose that

$$|x_z - x_f| \leq d_x$$
$$|y_z - y_f| \leq d_y$$

- this is equivalent to

$$-d_x \leq \frac{\tau_y}{f_z} \leq d_x$$
$$-d_y \leq \frac{\tau_x}{f_z} \leq d_y$$

- on the contact wrench $\boldsymbol{w}$, we can write as $\boldsymbol{A}\boldsymbol{w} \leq \boldsymbol{b}$, with

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & -d_x & 0 & 1 & 0 \\ 0 & 0 & -d_x & 0 & -1 & 0 \\ 0 & 0 & -d_y & 1 & 0 & 0 \\ 0 & 0 & -d_y & -1 & 0 & 0 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Cartesian tasks

- we define the j-th Cartesian task in terms of position $\boldsymbol{p}_j(t)$, velocity $\boldsymbol{v}_j(t)$ and acceleration $\boldsymbol{a}_j(t)$

- for each task we write a **PD + feedforward** law

$$\boldsymbol{a}_j = \boldsymbol{J}_j\dot{\boldsymbol{\nu}} + \dot{\boldsymbol{J}}_j\boldsymbol{\nu} = \underbrace{\boldsymbol{a}_j^d}_{\text{fedforward}} + \underbrace{k_p(\boldsymbol{p}_j^d - \boldsymbol{p}_j)}_{\text{proportional term}} + \underbrace{k_d(\boldsymbol{v}_j^d - \boldsymbol{v}_j)}_{\text{derivative term}}$$

- the tasks that we typically include are
  - tracking a center of mass trajectory
  - tracking trajectories of the feet for stepping
  - tracking an angular momentum reference
  - tracking a torso reference orientation
  - tracking a reference joint configuration, to resolve redundancy
  - tracking hand trajectories, for manipulation tasks

Cartesian tasks

- a particularly important kinematic task is the one relative to the foot in contact

- since this foot is in contact with the ground, both its velocity $\boldsymbol{v}_{c,i}$ and acceleration $\boldsymbol{a}_{c,i}$ must be zero

$$\boldsymbol{v}_{c,i} = 0 \quad \Longrightarrow \quad \boldsymbol{a}_{c,i} = \boldsymbol{J}_{c,i}\dot{\boldsymbol{\nu}} + \dot{\boldsymbol{J}}_{c,i}\nu = -k_v \boldsymbol{v}_{c,i}$$

- to effectively control the floating base, this task must be executed with precision, which is why it is often imposed as a **constraint**

whole-body QP

- the QP can be formulated as

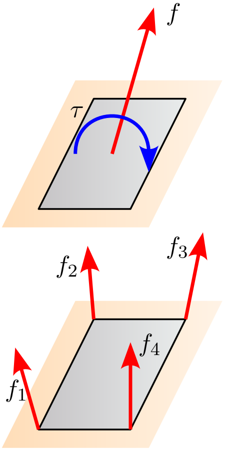
$$\min_{\dot{\boldsymbol{\nu}},\boldsymbol{\tau},\boldsymbol{w}_i} \sum_j \alpha_j \left(\boldsymbol{J}_j \dot{\boldsymbol{\nu}} + \dot{\boldsymbol{J}} \boldsymbol{\nu} - \boldsymbol{a}_j^d - k_p(\boldsymbol{p}_j^d - \boldsymbol{p}_j) - k_v(\boldsymbol{v}_j^d - \boldsymbol{v}_j) \right)^2$$

$$\begin{aligned} \text{s.t.} \quad & M\dot{\boldsymbol{\nu}} + \boldsymbol{n} = \boldsymbol{S}\boldsymbol{\tau} + \sum_i \boldsymbol{J}_i^T \boldsymbol{w}_i \\ & J_{c,i}\dot{\boldsymbol{\nu}} + \dot{\boldsymbol{J}}_{c,i}\boldsymbol{\nu} = -k_v \boldsymbol{v}_{c,i} \\ & \boldsymbol{A}\boldsymbol{w}_j \leq \boldsymbol{b} \end{aligned}$$

- $\sum_j$ sums over all the task that we want our robot to achieve, with weights α_j
- constraints include **dynamics**, **contact constraints**, and **contact wrench constraints**

four-forces contact model

- the conditions we need to impose on the contact wrenches (in particular that on τ_z) are rather complex
- a different approach is to use a **contact model** in which we specify that each contact is mediated by a predefined number of forces, and no torque (usually 4 forces)



four-forces contact model

- on each of these four forces we need to impose the usual force constraints: **unilaterality** and **friction cone**

- no need to worry about the ZMP: imposing unilaterality of the four forces is **equivalent** to imposing that the ZMP is inside the convex hull of their points of application

- some people prefer this approach to using contact wrenches, but they are equivalent