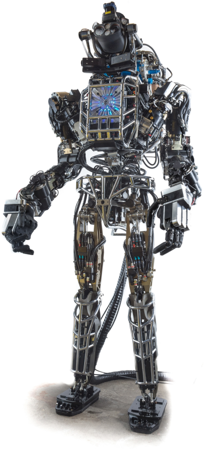
Autonomous and Mobile Robots: Humanoid Gait Generation

Nicola Scianca

December 2024

outline

- gait generation
- model predictive control
- preview control
- MPC for gait generation
- MPC on the LIP model
- examples

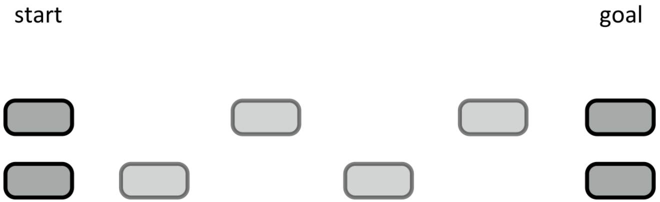


what is a gait?

- a **gait** is the way a person **walks**
- in humanoid robotics, **gait generation** the process by which we generate a trajectory for the **Center of Mass** (CoM) of the robot
- assume we want to go from a start position to a goal position: let us now look at the conceptual steps we have to go through in order to achieve this

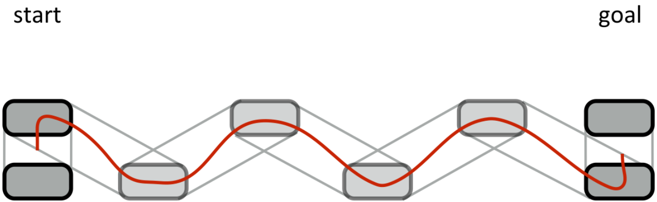
gait generation

- the first thing to do is **plan** a sequence of **footsteps** (e.g., using RRT); this defines the shape and position of the **support polygon** at each time during the walk



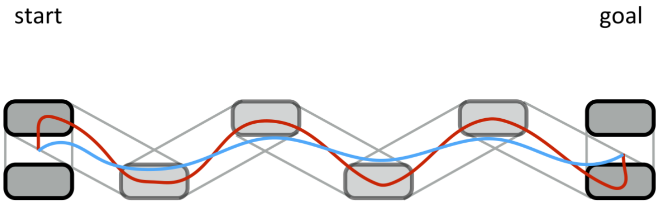
gait generation

- now we can design a **ZMP trajectory** that is always inside the **support polygon**: it uses double support phases to transition between consecutive footsteps



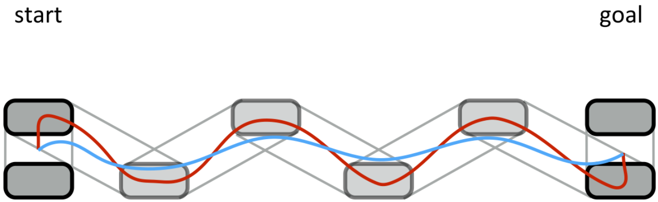
gait generation

- the next step is to generate a **CoM trajectory** that satisfies the dynamics of the **simplified model** we chose to use (e.g., the LIP model)



gait generation

- now we generate the other trajectories that we need (e.g., feet, torso orientation, ...) and track them using a **whole-body controller**



gait generation

- since the ZMP is the input of the LIP model, we could in principle design an arbitrary ZMP trajectory $p_z(t)$, plug it inside the model and **integrate** forward to obtain the CoM trajectory $p_c(t)$

$$\textbf{ZMP} \Longrightarrow \boxed{\ddot{\boldsymbol{p}}_c = \eta^2(\boldsymbol{p}_c - \boldsymbol{p}_z)} \Longrightarrow \textbf{CoM}$$

- however, the LIP is **unstable**! if we do not keep this into account we would get a **diverging** CoM trajectory

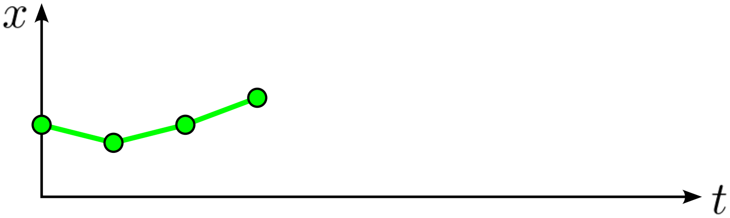
- a solution is to have an **optimization-based controller** design both the **ZMP** and the **CoM** trajectories

model predictive control

- **Model Predictive Control** (MPC) is a control technique that uses a model of the system to optimize the **trajectory** of the system over a short **prediction horizon**
- the optimization process involves minimizing a **cost function** subject to some **constraints**
- at each time-step we obtain a sequence of optimal inputs: we apply the **first** and discard the rest, then we **repeat** the optimization at the next time-step

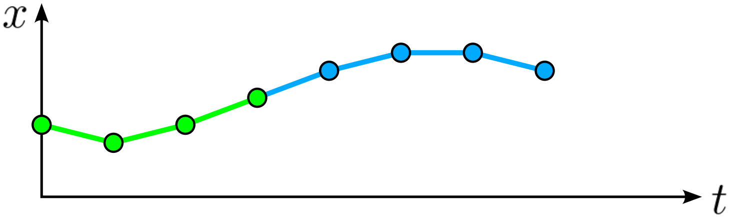
model predictive control

- let's see an example of MPC in action: in green is our **realized trajectory** up to the present moment



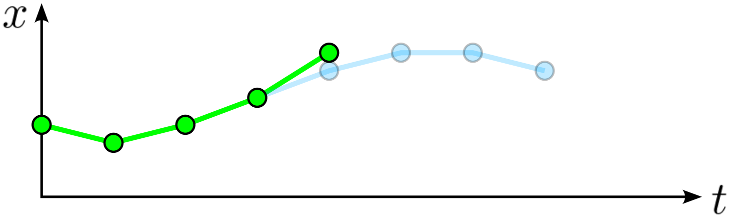
model predictive control

- we **predict** an optimal trajectory starting from the current state and apply the **first predicted input**



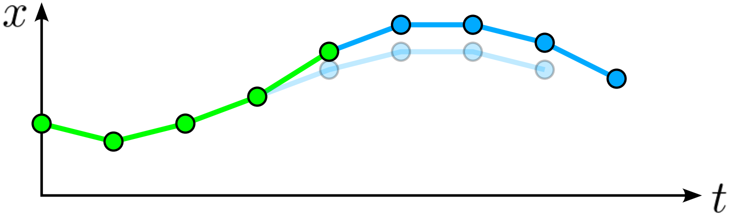
model predictive control

- the new state will be slightly different than we predicted due to **model inaccuracy** and **disturbances**



model predictive control

- now we find a new prediction starting from the current **measured state**



model predictive control

- starting from the current time-step k, we want to predict a **discrete** sequence of input and states over N future time-steps

$$\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_{k+N-1}, \boldsymbol{x}_{k+N}$$
$$\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_{k+N-1}$$

- these states and inputs are connected via the systems **dynamics**, which we write in discrete form

$$\boldsymbol{x}_{i+1} = f(\boldsymbol{x}_i, \boldsymbol{u}_i)$$

- for physical systems, which have continuous dynamics, we assume to have discretized them in some way (e.g., Euler, Runge-Kutta, ...)

model predictive control

- at time-step k, an **MPC** computes an optimal control sequence by solving an optimization problem of the form

$$\min \sum_{i=k}^{k+N-1} \underbrace{l(\boldsymbol{x}_i, \boldsymbol{u}_i)}_{\text{running cost}} + \underbrace{l_T(\boldsymbol{x}_{k+N})}_{\text{terminal cost}}$$

$$\begin{aligned} \text{s.t.} \quad \boldsymbol{x}_k &= \bar{\boldsymbol{x}}_k \\ \boldsymbol{x}_{i+1} &= f(\boldsymbol{x}_i, \boldsymbol{u}_i) \quad \text{for } i = 0, \ldots, N-1 \\ h(\boldsymbol{x}_i, \boldsymbol{u}_i) &= 0 \quad \text{for } i = 0, \ldots, N \\ g(\boldsymbol{x}_i, \boldsymbol{u}_i) &\leq 0 \quad \text{for } i = 0, \ldots, N \end{aligned}$$

- the initial state $\boldsymbol{x}_k$ of the predicted sequence is set equal to the **measured state** $\bar{\boldsymbol{x}}_k$ (**feedback!**)
- other constraints ensure that the **dynamics** are satisfied, and enforce other requirements on states and inputs

model predictive control

- generic optimization problems can be **computationally heavy**, but an MPC controller must give us solutions at a rate of about 100 Hz
- to keep computation time manageable, we typically formulate this as a **Quadratic Program** (QP), i.e., an optimization problem with **quadratic cost** and **linear constraints**
- these problems can be solved using off-the-shelf libraries in **milliseconds**

preview control

- MPC gait generation using the LIP model was introduced with the **preview controller** [Kajita et al.]

- this technique used a cost function to track a **reference ZMP** trajectory, designed in such a way to always be inside the support polygon

- it did not make use of **constraints**, therefore the solution was given in **closed form** (without the need to use a QP solver)

Kajita et al., "Biped walking pattern generation by using preview control of zero-moment point", International Conference on Robotics and Automation, 2003

preview control

- the dynamics is represented having the **CoM jerk** $\dddot{p}_c$ (third time derivative) as the input, and the **ZMP** as the output

$$\underbrace{\frac{d}{dt}\begin{pmatrix} p_c \\ \dot{p}_c \\ \ddot{p}_c \end{pmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}}_{\boldsymbol{A}} \underbrace{\begin{pmatrix} p_c \\ \dot{p}_c \\ \ddot{p}_c \end{pmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}}_{\boldsymbol{B}} \underbrace{\dddot{p}_c}_{\boldsymbol{u}}$$

$$\underbrace{p_z}_{\boldsymbol{y}} = \underbrace{\begin{pmatrix} 1 & 1 & -1/\eta^2 \end{pmatrix}}_{\boldsymbol{C}} \underbrace{\begin{pmatrix} p_c \\ \dot{p}_c \\ \ddot{p}_c \end{pmatrix}}_{\boldsymbol{x}}$$

- the jerk $\dddot{p}_c$ is used instead of the acceleration $\ddot{p}_c$ to avoid the output directly depending on the input, which might cause discontinuous ZMP trajectories

preview control

- the optimization problem can now be written as

$$\min_{\boldsymbol{x}_i,\boldsymbol{u}_i} \sum_{i=k}^{k+N-1} \left(\left(\dddot{p}_c^{\,i} \right)^2 + \beta \left(p_z^i - p_z^{i,\text{ref}} \right)^2 \right) + \beta \left(p_z^{k+N} - p_z^{k+N,\text{ref}} \right)^2$$

$$\text{s.t.} \quad \boldsymbol{x}_k = \bar{\boldsymbol{x}}_k$$

$$\boldsymbol{x}_{i+1} = \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{B}\boldsymbol{u}_i \quad \text{for } i = 0, \dots, N-1$$

- there are constraints to enforce the consistency of the predicted sequence with the **dynamics** of the model
- β is a weight that modulates how strongly we want to **track** the reference

preview control

- the predicted states can be expressed in terms of the measured current state $\bar{\boldsymbol{x}}_k$ and the predicted input sequence $\boldsymbol{u}_k, \ldots, \boldsymbol{u}_{k+N-1}$ by performing **algebraic substitutions**

$$
\begin{aligned}
\boldsymbol{x}_{k+1} &= \boldsymbol{A}\bar{\boldsymbol{x}}_k + \boldsymbol{B}\boldsymbol{u}_k \\
\boldsymbol{x}_{k+2} &= \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{u}_{k+1} = \boldsymbol{A}(\boldsymbol{A}\bar{\boldsymbol{x}}_k + \boldsymbol{B}\boldsymbol{u}_k) + \boldsymbol{B}\boldsymbol{u}_{k+1} \\
&= \boldsymbol{A}^2\bar{\boldsymbol{x}}_k + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{B}\boldsymbol{u}_{k+1} \\
\boldsymbol{x}_{k+3} &= \boldsymbol{A}\boldsymbol{x}_{k+2} + \boldsymbol{B}\boldsymbol{u}_{k+2} = \boldsymbol{A}(\boldsymbol{A}^2\bar{\boldsymbol{x}}_k + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{B}\boldsymbol{u}_{k+1}) + \boldsymbol{B}\boldsymbol{u}_{k+2} \\
&= \boldsymbol{A}^3\bar{\boldsymbol{x}}_k + \boldsymbol{A}^2\boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_{k+1} + \boldsymbol{B}\boldsymbol{u}_{k+2} \\
&\vdots \\
\boldsymbol{x}_{k+N} &= \boldsymbol{A}^N\bar{\boldsymbol{x}}_k + \boldsymbol{A}^{N-1}\boldsymbol{B}\boldsymbol{u}_k + \cdots + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_{k+N-2} + \boldsymbol{B}\boldsymbol{u}_{k+N-1}
\end{aligned}
$$

preview control

- preview control is essentially using **unconstrained MPC** to realize **tracking** of a predefined ZMP trajectory
- the absence of constraints implies that this method is quite fast, because the solution can be recovered in closed form (we have to invert a large matrix but only once)
- however, without constraints we cannot guarantee that the **ZMP** will always be inside the **support polygon**

constrained MPC

- since we know the footstep plan, it is possible to enforce **constraints** that will ensure that the ZMP is always contained in the support polygon

$$p_z^{k+i,\min} \leq \boldsymbol{C}\boldsymbol{x}_{k+i} \leq p_z^{k+i,\max}$$

- if the robot is in **single support** $p_z^{k+i,\min}$ and $p_z^{k+i,\max}$ represent the lower and upper edge of the foot

- if the robot is in **double support** the expression is a bit more involved, but still linear

Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations", International Conference on Humanoid Robots, 2006

constrained MPC

- having constrained the ZMP, we do not need to explicitly track the reference in the cost function (but we still minimize the square of the input to make the problem convex)

$$\min_{\boldsymbol{x}_i, \boldsymbol{u}_i} \sum_{i=k}^{k+N-1} \left(\dddot{p}_c^{\,i} \right)^2$$

$$\begin{aligned} \text{s.t.} \quad \boldsymbol{x}_k &= \bar{\boldsymbol{x}}_k \\ \boldsymbol{x}_{i+1} &= \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{B}\boldsymbol{u}_i \quad \text{for } i = 0, \dots, N-1 \\ p_z^{k+i,\min} \leq \boldsymbol{C}&\boldsymbol{x}_{k+i} \leq p_z^{k+i,\max} \quad \text{for } i = 1, \dots, N \end{aligned}$$

- this in principle means that the MPC is free to choose the "best" trajectory, although in practice we might still want to include a ZMP tracking term if it gives better results

MPC on the LIP

- when we derived the LIP model, we saw that it can be written in state-space form by choosing the **ZMP** as the **input**

- to avoid discontinuous ZMP trajectories we can **dynamically extend** the system with the ZMP derivative $\dot{p}_z$ as new input

$$\underbrace{\frac{d}{dt}\begin{pmatrix} p_c \\ \dot{p}_c \\ p_z \end{pmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{pmatrix} 0 & 1 & 0 \\ \eta^2 & 0 & -\eta^2 \\ 0 & 0 & 0 \end{pmatrix}}_{\boldsymbol{A}} \underbrace{\begin{pmatrix} p_c \\ \dot{p}_c \\ \ddot{p}_c \end{pmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}}_{\boldsymbol{B}} \underbrace{\dot{p}_z}_{\boldsymbol{u}}$$

MPC on the LIP

- since the LIP has a positive eigenvalue, we will get a diverging trajectory if we just set up an MPC that minimizes the square of the input
- to avoid the instability, we can enforce a **constraint** derived from the following **stability condition** on the unstable part of the dynamics [Scianca et al.]

$$\underbrace{p_x^k + \frac{\dot{p}_x^k}{\eta}}_{\text{current state}} = \eta \int_{t_k}^{\infty} e^{-\eta(\tau - t_k)} \underbrace{p_z(\tau)}_{\text{future ZMP}} \, d\tau$$

- the **future ZMP** trajectory is partly **predicted** by the MPC, and partly **conjectured** based on the footstep plan

Scianca et al.,"MPC for Humanoid Gait Generation: Stability and Feasibility", Transactions on Robotics, 2020

constrained MPC

- this condition must be enforced inside the optimization problem as an additiona **stability constraint**

$$\min_{\boldsymbol{x}_i, \boldsymbol{u}_i} \sum_{i=k}^{k+N-1} \left(\dddot{p}_z^{\,i} \right)^2$$

$$\text{s.t.} \quad \boldsymbol{x}_k = \bar{\boldsymbol{x}}_k$$

$$\boldsymbol{x}_{i+1} = \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{B}\boldsymbol{u}_i \quad \text{for } i = 0, \ldots, N-1$$

$$\text{ZMP constraints}$$

$$\text{stability constraint}$$

- however, we can find guarantees of **feasibility** and **stability** thanks to this additional constraint

constrained MPC

- this condition must be enforced inside the optimization problem as an additiona **stability constraint**

$$\begin{aligned}\min_{\boldsymbol{x}_i, \boldsymbol{u}_i} \quad & \sum_{i=k}^{k+N-1} \left(\dot{p}_z^i\right)^2 \\ \text{s.t.} \quad & \boldsymbol{x}_k = \bar{\boldsymbol{x}}_k \\ & \boldsymbol{x}_{i+1} = \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{B}\boldsymbol{u}_i \quad \text{for } i = 0, \ldots, N-1 \\ & \text{ZMP constraints} \\ & \text{stability constraint}\end{aligned}$$

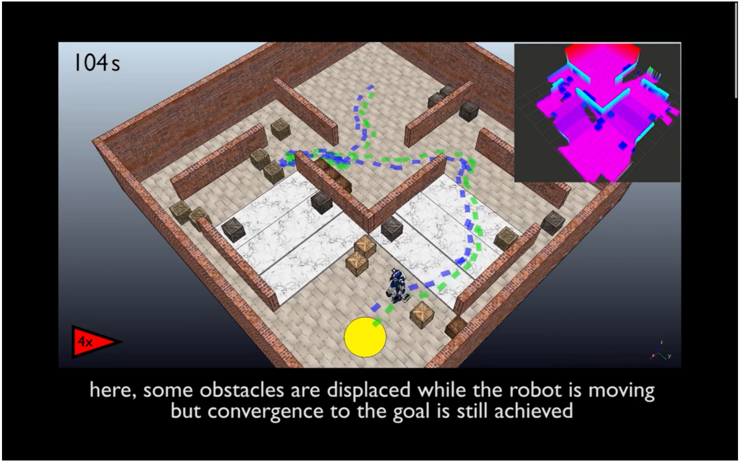
- however, we can find guarantees of **feasibility** and **stability** thanks to this additional constraint

MPC locomotion example

- a working implementation of this last MPC scheme is available at https://github.com/DIAG-Robotics-Lab/ismpc
- it is written in Python using the DART robotics toolkit
- the optimization problems are solved with OSQP, interfaced using CasADi

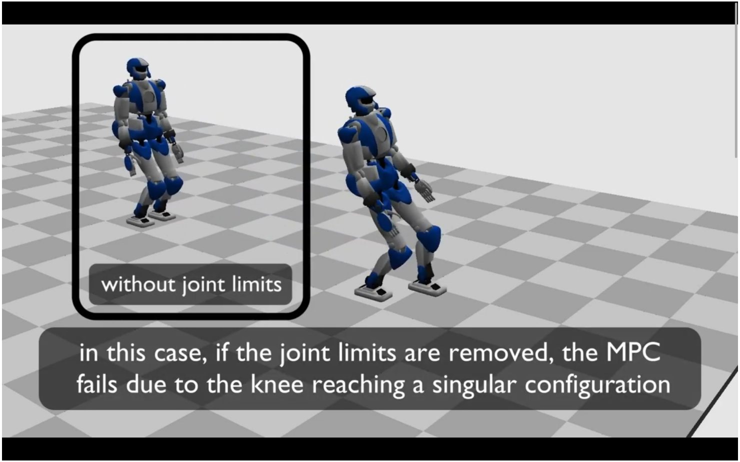
MPC locomotion example

locomotion in complex environments



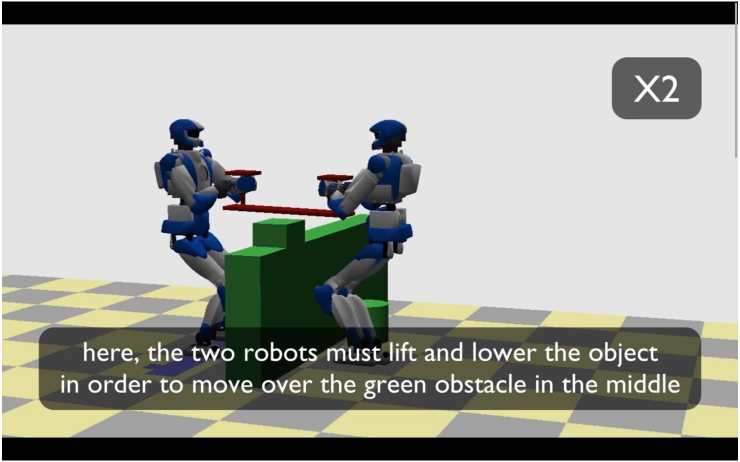
https://www.youtube.com/embed/BF43qUcx4gY

MPC using a whole-body model



https://www.youtube.com/embed/Fa6iy3mUcBY

not only locomotion



https://www.youtube.com/embed/ChfhpSJFqg0