Autonomous and Mobile Robotics

Prof. Giuseppe Oriolo

Humanoid Robots 4: Gait Generation

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

gait generation

generate a gait for walking while maintaining balance (also called walking pattern generation)

balance: what is it? how do we guarantee it?

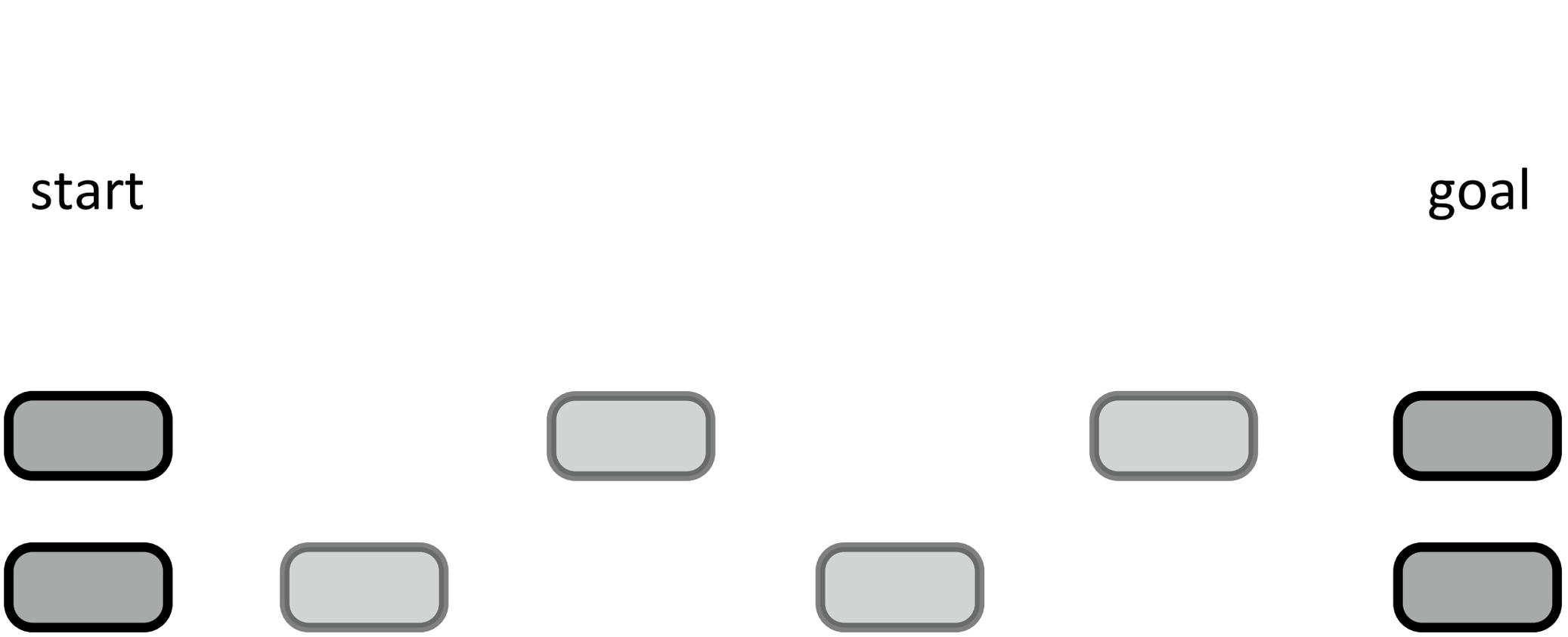
for this, different approaches available:

- nonlinear analysis
- Poincaré maps
- Viability
- Capturability
- Zero Moment Point ⟵ will use this (most popular)

ZMP-based gait generation

an algorithm based on the strategy: keep the ZMP inside the Support Polygon (SP)

1. plan the footsteps…

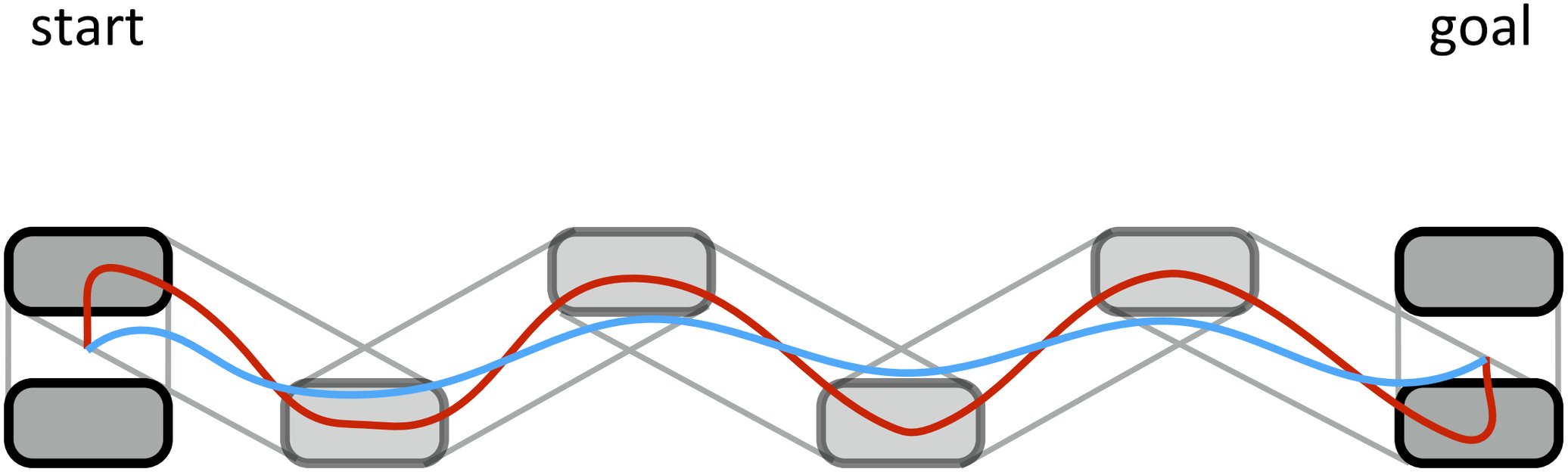


ZMP-based gait generation

…2. plan a ZMP trajectory such that is the ZMP is always inside the current SP…

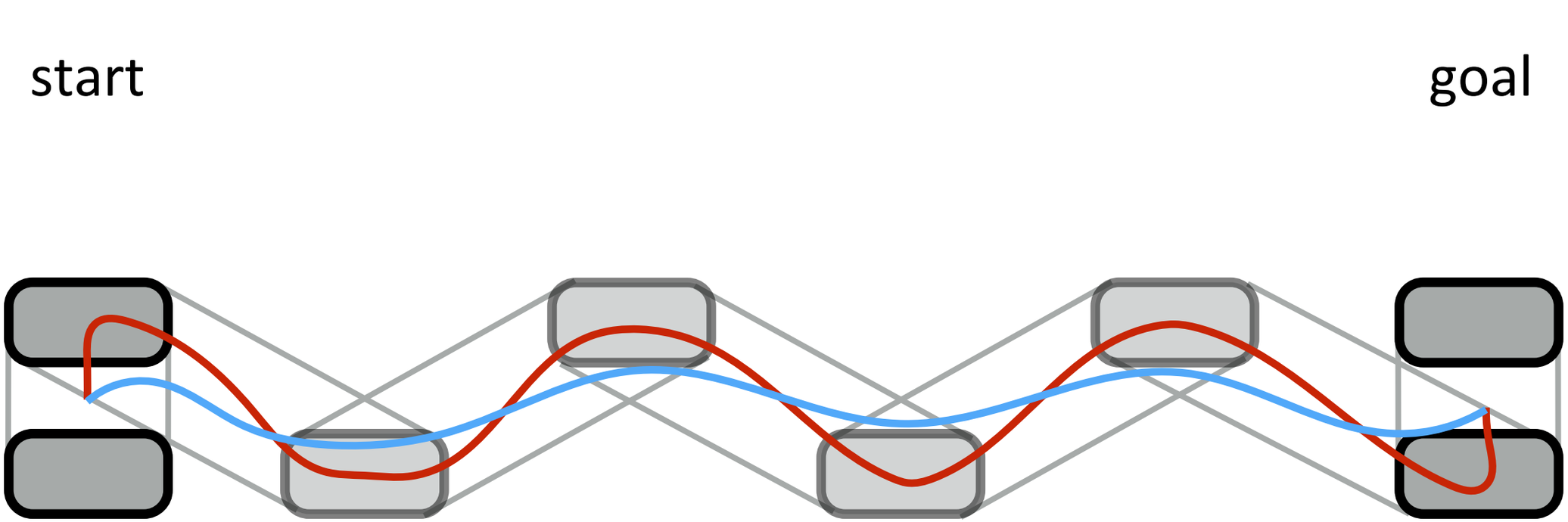
ZMP-based gait generation

…3. compute a CoM trajectory such that the ZMP moves as planned…



ZMP-based gait generation

…4. track the CoM trajectory



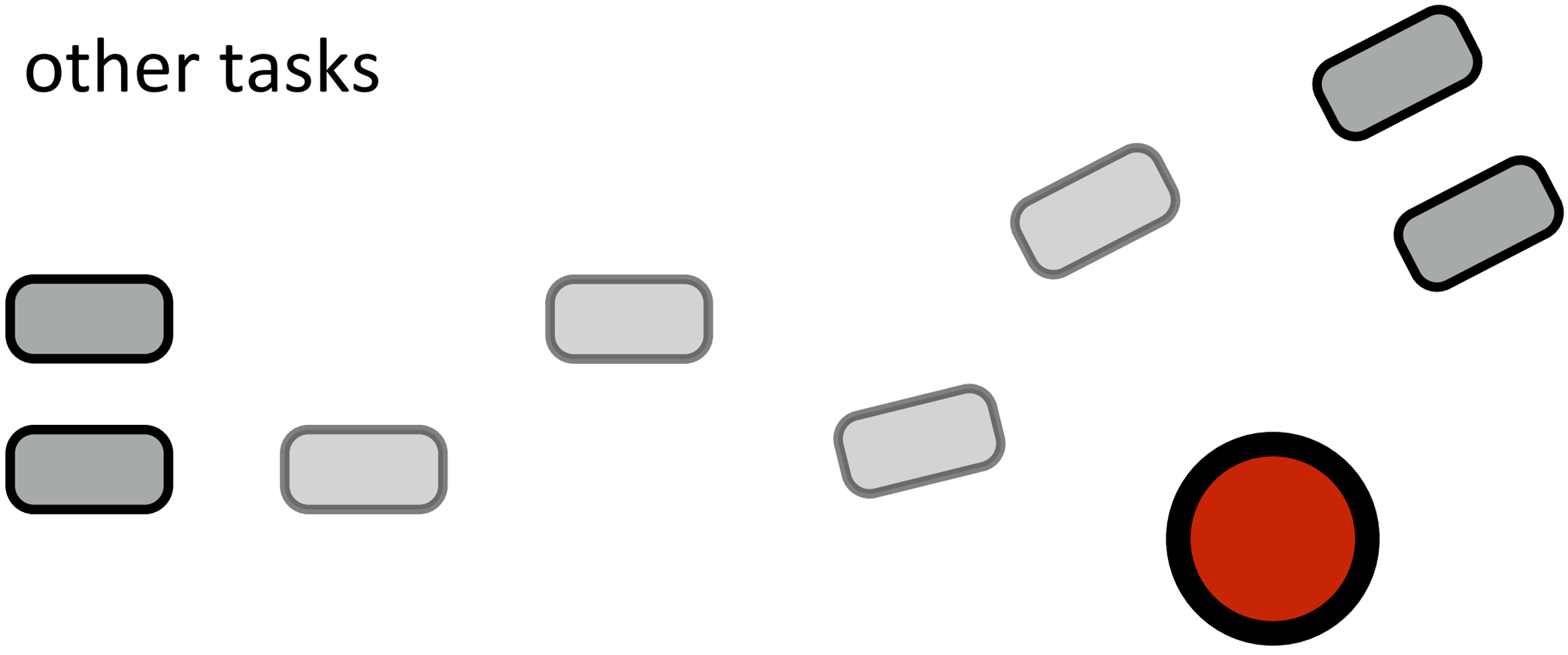
algorithm steps in detail

1 plan the footsteps (offline)

timing and lengths (desired speed)

obstacles (obstacle avoidance)

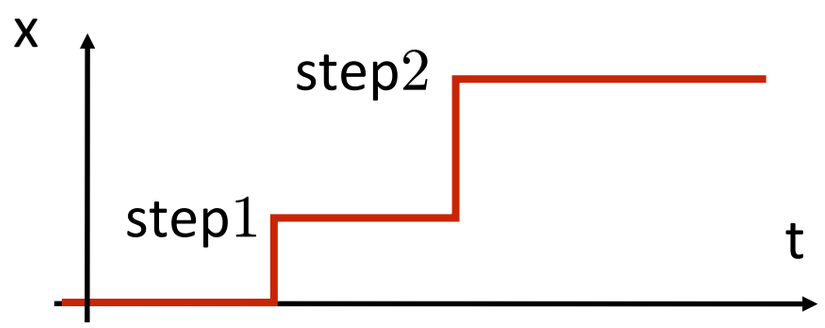
other tasks



algorithm steps in detail

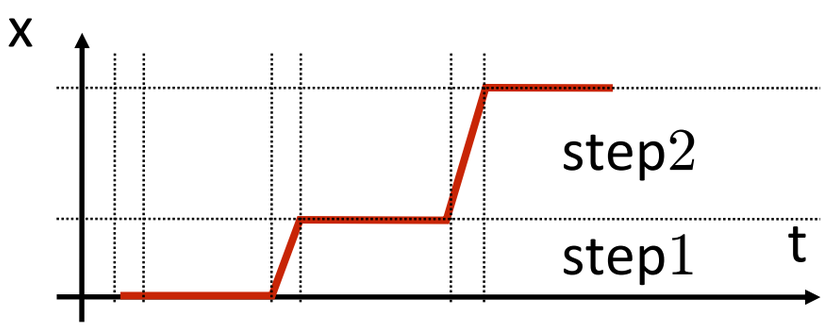
2 plan ZMP trajectory (interpolation)

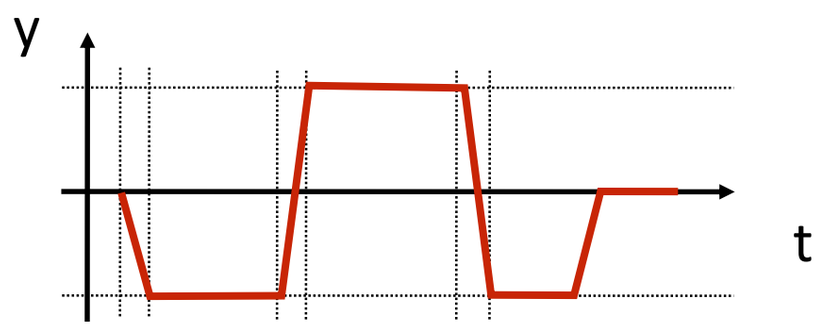
point foot (ZMP = point of contact)



finite-sized foot

cycling through
SS and DS phases





algorithm steps in detail

3 compute a (desired) CoM trajectory consistent with the planned ZMP trajectory

use the LIP model! e.g., for the sagittal direction

$$\ddot{c}^x = \frac{g^z}{c^z}\left(c^x - z^x\right)$$

planned ZMP trajectory

the CoM trajectory should be a solution of this 2nd order differential equation driven by the forcing term $z^x(t)$

potential instability problem! more on this later…

algorithm steps in detail

4 track the desired CoM trajectory

A. define a swinging foot trajectory
B. use kinematic control to obtain reference joint trajectories that realize the CoM and foot trajectories
C. send the reference joint profiles to the joint servos (for a position-controlled humanoid)

other approaches possible

instability problem: a control perspective

$$\ddot{c}^x = \frac{g^z}{c^z}\left(c^x - z^x\right) \qquad \omega^2 = \frac{g^z}{c^z}$$

pendulum frequency

ZMP → CoM

$$\frac{c^x(s)}{z^x(s)} = \frac{\omega^2}{s^2 - \omega^2} \qquad \frac{\ddot{c}^x(s)}{z^x(s)} = \frac{\omega^2 s^2}{s^2 - \omega^2}$$

inversion

$$\frac{z^x(s)}{j^x(s)} = \frac{s^2 - \omega^2}{s^3\omega^2} \qquad j^x(t) = \frac{d}{dt}\left(\ddot{c}^x(t)\right) \qquad \frac{z^x(s)}{\ddot{c}^x(s)} = \frac{s^2 - \omega^2}{s^2\omega^2}$$

jerk

CoM → ZMP

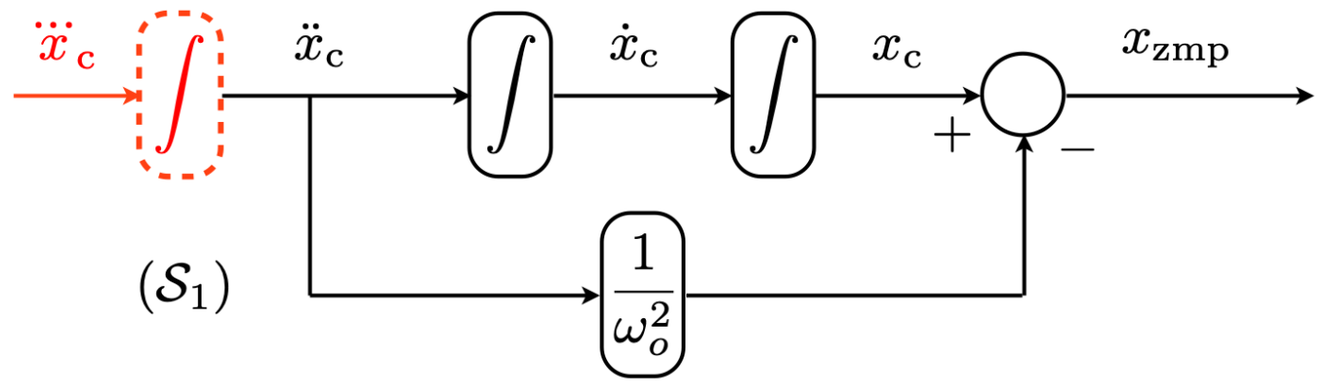
divergence problem: a control perspective

$$\ddot{c}^x = \frac{g^z}{c^z}\left(c^x - z^x\right) \qquad\qquad \omega^2 = \frac{g^z}{c^z}$$

reference ZMP

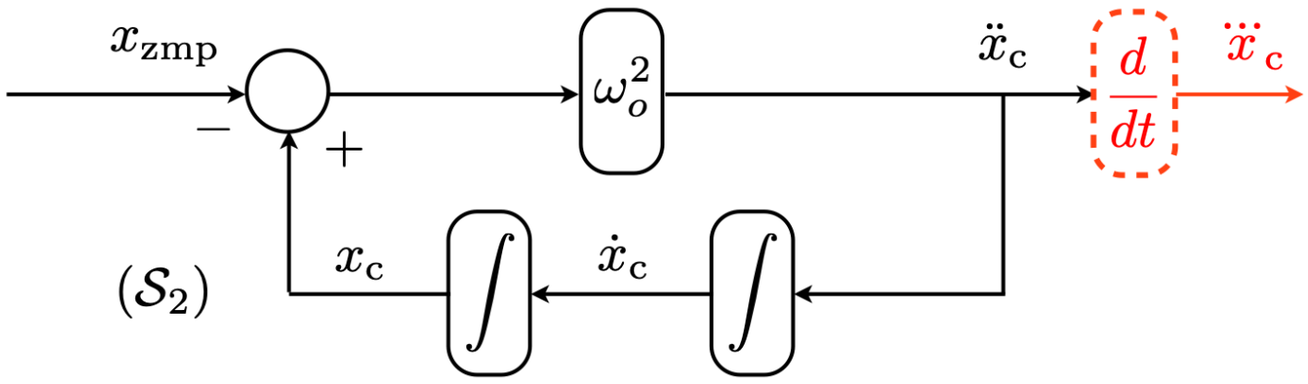
CoM ⟶ ZMP

output tracking problem



ZMP ⟶ CoM

stable inversion problem



stable inversion

- using a change of coordinates, the LIP can be decoupled in **stable** and **unstable** dynamics

$$x_s = c - \dot{c}/\eta$$
$$x_u = c + \dot{c}/\eta$$

- the decoupled dynamics are

stable $\dot{x}_s = \eta(-x_s + x_z)$

unstable $\dot{x}_u = \eta(x_u - x_z)$

- the CoM evolution is bounded if and only if

$$x_u(t_0) = \eta \int_{t_0}^{\infty} e^{-\eta(\tau - t_0)} z(\tau) d\tau$$

stability condition

linear MPC

- example: regulation using linear **Model Predictive Control**

- linear prediction model

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{cases}$$

$$x \in \mathbb{R}^n$$
$$u \in \mathbb{R}^m$$
$$y \in \mathbb{R}^p$$

- relationship between input and states

$$x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j}$$

linear MPC – cost function

- cost function

$$J(z, x_0) = x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k \qquad \begin{array}{lcl} R & = & R' \succ 0 \\ Q & = & Q' \succeq 0 \\ P & = & P' \succeq 0 \end{array} \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

- goal: find a sequence $z^\star(t)$ that minimizes $J(z,x_0)$, i.e., that steers the state x to the origin optimally

linear MPC – cost function

- cost function

$$J(z,x_0) = x_0'Qx_0 + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}' \overbrace{\begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & Q & 0 \\ 0 & 0 & \dots & 0 & P \end{bmatrix}}^{\bar{Q}} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}$$

$$+ \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}' \underbrace{\begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & R \end{bmatrix}}_{\bar{R}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \underbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}}_{\bar{S}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{z} + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\bar{T}} x_0$$

linear MPC – unconstrained case

- if the system is linear the cost function is **quadratic** in the input

$$\begin{aligned} J(z, x_0) &= (\bar{S}z + \bar{T}x_0)'\bar{Q}(\bar{S}z + \bar{T}x_0) + z'\bar{R}z + x_0'Qx_0 \\ &= \frac{1}{2}z' \underbrace{2(\bar{R} + \bar{S}'\bar{Q}\bar{S})}_{H} z + x_0' \underbrace{2\bar{T}'\bar{Q}\bar{S}}_{F'} z + \frac{1}{2}x_0' \underbrace{2(Q + \bar{T}'\bar{Q}\bar{T})}_{Y} x_0 \end{aligned} \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$J(z, x_0) = \frac{1}{2}z'Hz + x_0'F'z + \frac{1}{2}x_0'Yx_0$$

- without constraints, optimal solution by zeroing the gradient

$$\nabla_z J(z, x_0) = Hz + Fx_0 = 0 \quad \Rightarrow \quad z^* = \begin{bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{N-1}^* \end{bmatrix} = -H^{-1}Fx_0$$

linear MPC – constraints

- add output and input **constraints**

$$\begin{cases} u_{\min} \leq u(t) \leq u_{\max} \\ y_{\min} \leq y(t) \leq y_{\max} \end{cases}$$

- what results is a **constrained optimal control problem**

$$\min_{z} \quad x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k$$

$$\text{s.t.} \quad u_{\min} \leq u_k \leq u_{\max},\ k = 0, \ldots, N-1$$
$$y_{\min} \leq y_k \leq y_{\max},\ k = 1, \ldots, N$$

linear MPC – constraints

- we want to formulate this as a standard convex **Quadratic Program (QP)**

$$V(x_0) = \tfrac{1}{2}x_0' Y x_0 + \quad \min_z \quad \tfrac{1}{2}z' H z + x_0' F' z \quad \text{(quadratic objective)}$$

$$\text{s.t.} \quad Gz \leq W + Sx_0 \quad \text{(linear constraints)}$$

- we already found the cost function: how do we write the constraints?

linear MPC – constraints

- input constraints

$$\begin{cases} u_k & \leq & u_{\max} \\ -u_k & \leq & -u_{\min} \end{cases} \Rightarrow \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & 1 \\ -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & -1 \end{bmatrix} z \leq \begin{bmatrix} u_{\max} \\ u_{\max} \\ \vdots \\ u_{\max} \\ -u_{\min} \\ -u_{\min} \\ \vdots \\ -u_{\min} \end{bmatrix} \qquad z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

linear MPC – constraints

- we write the output in terms of predicted inputs using

$$y_k = CA^k x_0 + \sum_{i=0}^{k-1} CA^i B u_{k-1-i} \leq y_{\max},\ k = 1, \ldots, N$$

- the upper output constraint is

$$\begin{bmatrix} CB & 0 & \ldots & 0 \\ CAB & CB & \ldots & 0 \\ \vdots & & & \vdots \\ CA^{N-1}B & \ldots & CAB & CB \end{bmatrix} z \leq \begin{bmatrix} y_{\max} \\ y_{\max} \\ \vdots \\ y_{\max} \end{bmatrix} - \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} x_0$$

and the lower output constraint is analogous

linear MPC – algorithm

at each step:

- measure or estimate the **current state**
- compute the **prediction** (optimal control sequence) by solving the QP starting from the current state
- apply only the **first input** of the predicted sequence

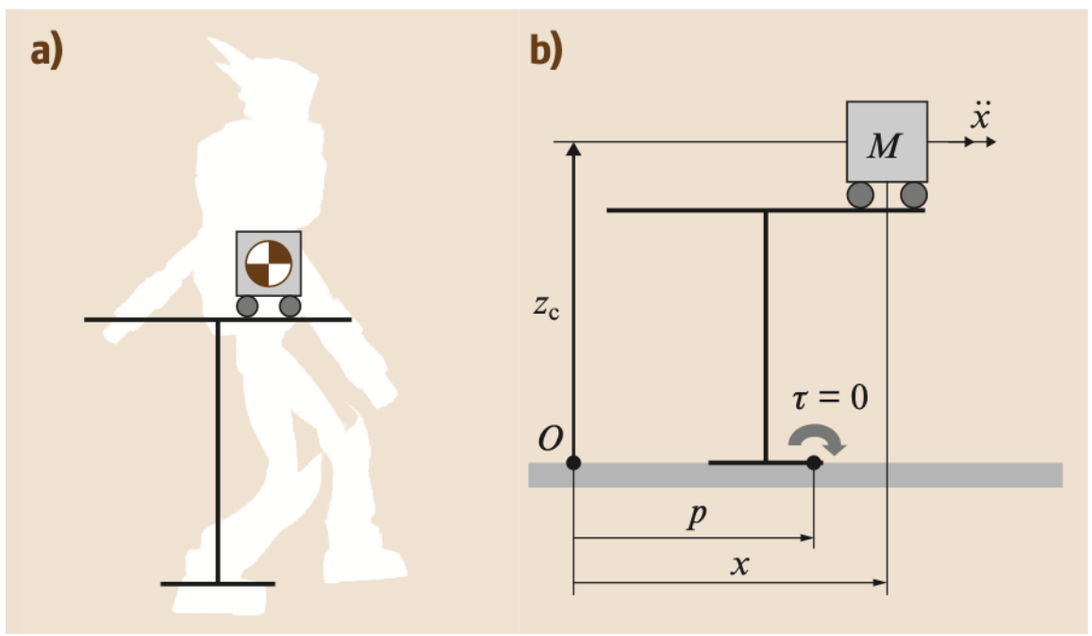
preview control

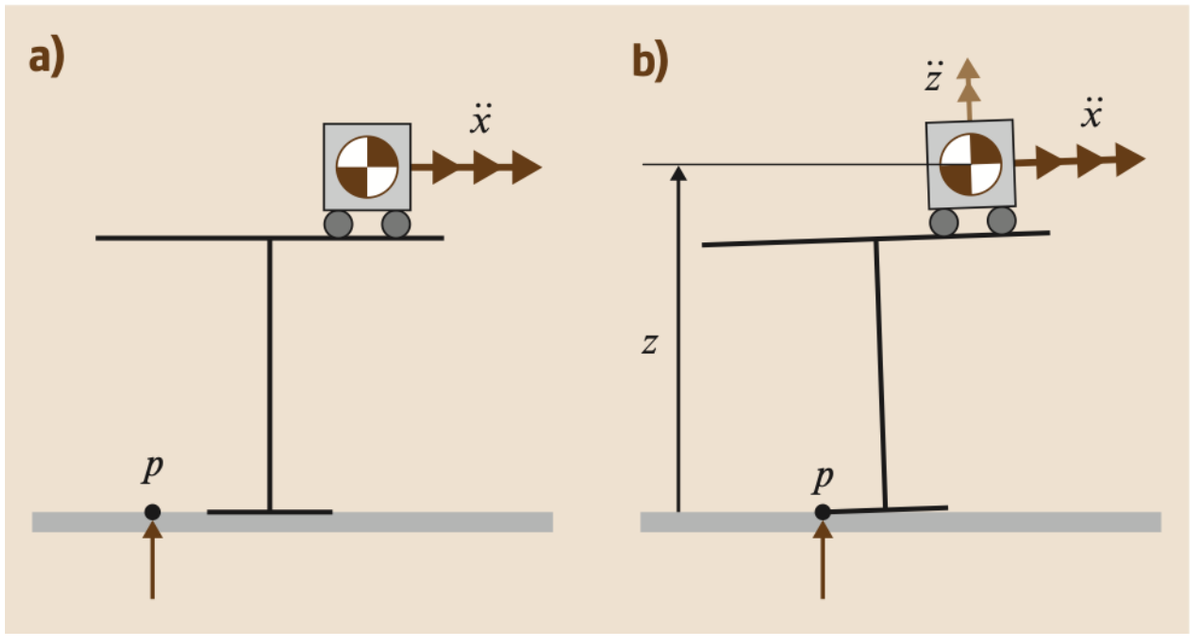
CoM trajectory generation can be seen as the

design of a ZMP tracking controller

CoM ⟶ ZMP

$$\frac{z^x(s)}{j^x(s)} = \frac{s^2 - \omega^2}{s^3\omega^2}$$





preview control

- Cart-Table (CT) model: the state of the system is defined by position, velocity and acceleration of the CoM

$$\boldsymbol{x} = [\boldsymbol{c}, \quad \dot{\boldsymbol{c}}, \quad \ddot{\boldsymbol{c}}]^T$$

- the control input is the jerk of the CoM

$$\boldsymbol{u} = \dddot{\boldsymbol{c}}$$

- the state dynamics is a triple integrator

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{pmatrix} \boldsymbol{c} \\ \dot{\boldsymbol{c}} \\ \ddot{\boldsymbol{c}} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \dot{\boldsymbol{c}} \\ \ddot{\boldsymbol{c}} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \boldsymbol{u}$$

- the output is the ZMP

$$\boldsymbol{z} = \begin{pmatrix} 1 & 0 & -c^z/g \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \dot{\boldsymbol{c}} \\ \ddot{\boldsymbol{c}} \end{pmatrix}$$

preview control

- by discretizing we obtain the dynamical system

$$\begin{cases} \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{C}\boldsymbol{u}_k \\ \boldsymbol{z}_k &= \boldsymbol{C}\boldsymbol{x}_k \end{cases}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{B} = \begin{bmatrix} T^3/6 \\ T^2/2 \\ T \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & -\frac{c^z}{g} \end{bmatrix}$$

preview control

- if the ZMP reference can be previewed for N_L future steps, the controller can be written as

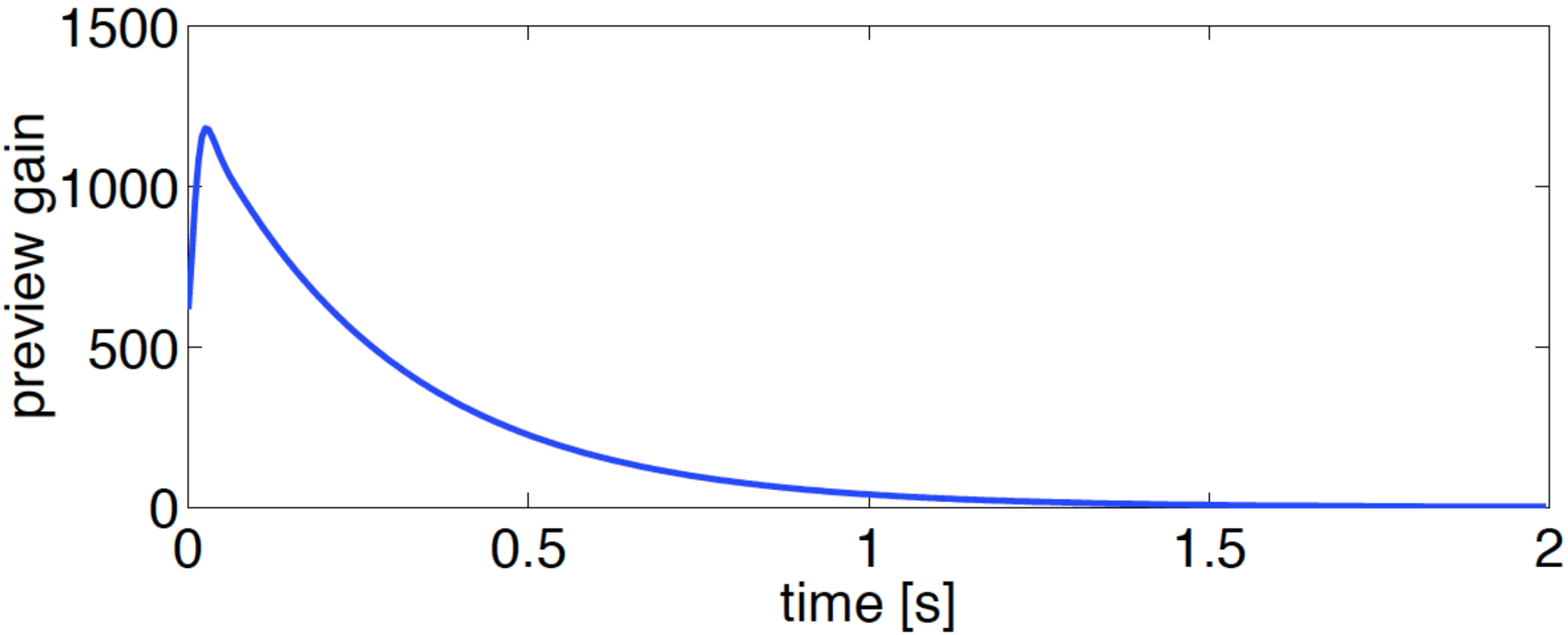
$$\boldsymbol{u}_k = \underbrace{-G_i \sum_{i=0}^{k} \boldsymbol{e}_k}_{\text{integral action on the tracking error}} - \underbrace{G_x \boldsymbol{x}_k}_{\text{state feedback}} - \underbrace{\sum_{j=1}^{N_L} G_p(j) \boldsymbol{z}^{ref}(k+j)}_{\text{preview action using future ZMP references}}$$

- this is the control law that minimizes the following cost function (unconstrained MPC)

$$J = \sum_{i=k}^{\infty} \{ Q_e \boldsymbol{e}_i^2 + \Delta \boldsymbol{x}_i^T Q_x \Delta \boldsymbol{x}_i + R \Delta \boldsymbol{u}_i^2 \}$$

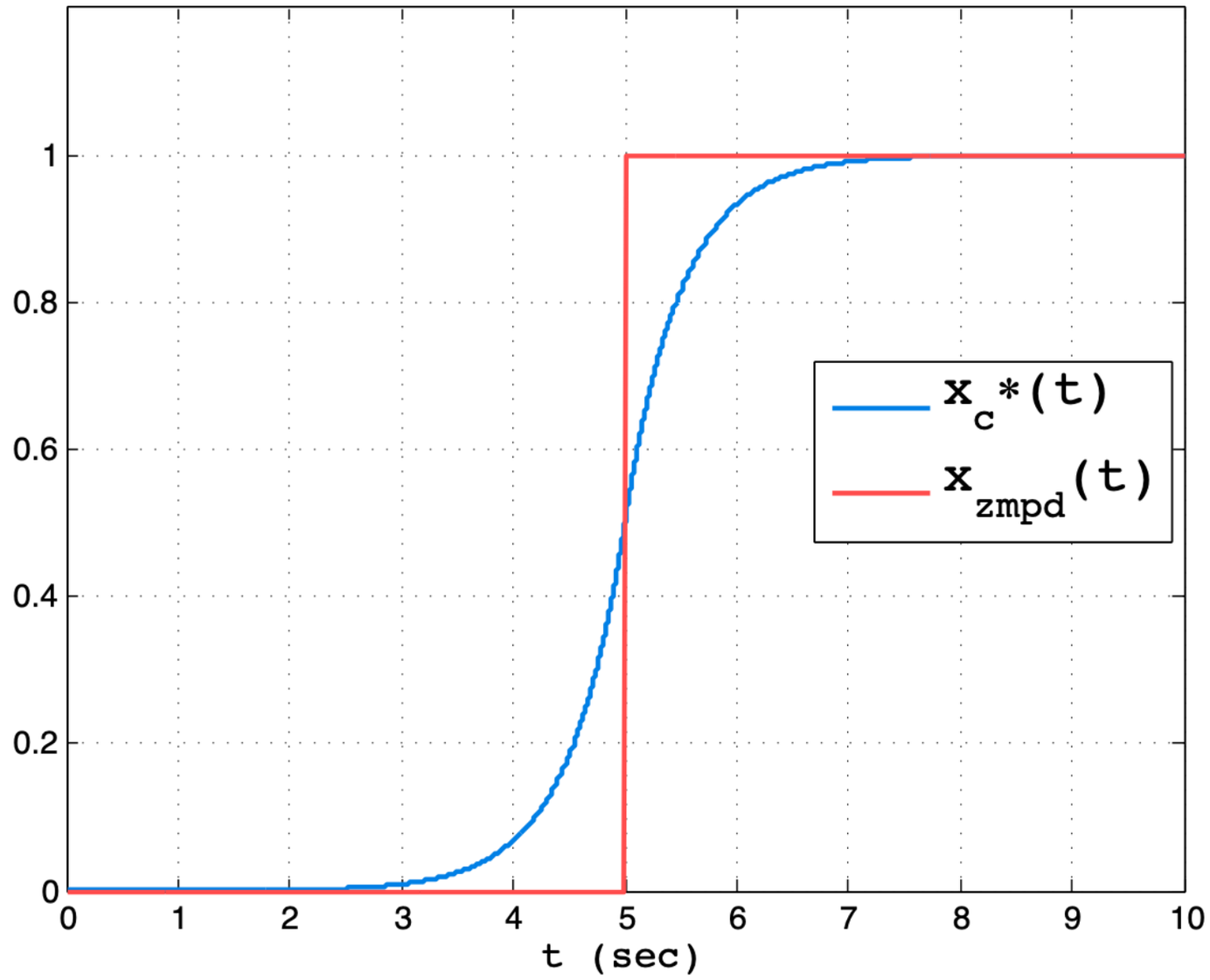
$$\boldsymbol{e}_k = \boldsymbol{z}_k - \boldsymbol{z}^{ref} \qquad \Delta \boldsymbol{x}_k = \boldsymbol{x}_k - \boldsymbol{x}_{k-1} \qquad \Delta \boldsymbol{u}_k = \boldsymbol{u}_k - \boldsymbol{u}_{k-1}$$

preview control



the magnitude of the preview gain quickly becomes very small; hence, the controller does not need information on far future

preview control



MPC gait generation

same formulation of the preview controller

- CoM jerk is the input $\boldsymbol{u} = \dddot{\boldsymbol{c}}$

- state is CoM position, velocity and acceleration

$$\boldsymbol{x} = [\boldsymbol{c}, \quad \dot{\boldsymbol{c}}, \quad \ddot{\boldsymbol{c}}]^T$$

- the state dynamics is a triple integrator

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{pmatrix} \boldsymbol{c} \\ \dot{\boldsymbol{c}} \\ \ddot{\boldsymbol{c}} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \dot{\boldsymbol{c}} \\ \ddot{\boldsymbol{c}} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \boldsymbol{u}$$

- the output is the ZMP

$$\boldsymbol{z} = \begin{pmatrix} 1 & 0 & -c^z/g \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \dot{\boldsymbol{c}} \\ \ddot{\boldsymbol{c}} \end{pmatrix}$$

MPC gait generation

- instead of tracking a ZMP reference, impose **ZMP constraints**

$$\begin{pmatrix} z_{k+1}^{x,min} \\ z_{k+2}^{x,min} \\ \vdots \\ z_{k+N}^{x,min} \\ z_{k+1}^{y,min} \\ z_{k+2}^{y,min} \\ \vdots \\ z_{k+N}^{y,min} \end{pmatrix} \leq \begin{pmatrix} z_{k+1}^{x} \\ z_{k+2}^{x} \\ \vdots \\ z_{k+N}^{x} \\ z_{k+1}^{y} \\ z_{k+2}^{y} \\ \vdots \\ z_{k+N}^{y} \end{pmatrix} \leq \begin{pmatrix} z_{k+1}^{x,max} \\ z_{k+2}^{x,max} \\ \vdots \\ z_{k+N}^{x,max} \\ z_{k+1}^{y,max} \\ z_{k+2}^{y,max} \\ \vdots \\ z_{k+N}^{y,max} \end{pmatrix}$$

(simplified case:
in general x-y
are not decoupled)

- the cost function can just minimize the square of the input over the prediction horizon

$$J = \sum_{i=k}^{k+N-1} \left((\dddot{c}_i^x)^2 + (\dddot{c}_i^y)^2 \right)$$

MPC gait generation

- the idea is to shift the the balance condition from the **cost function** (tracking a reference) to the **constraints** (ZMP in support polygon)

- more **guarantees** on the ZMP at the cost of a more **complex** controller (need to solve a constrained optimization at each step)

- good QP solvers can still guarantee **real-time** execution

MPC on the LIP model

- we can use the **Linear Inverted Pendulum (LIP)** as the prediction model of the MPC

$$\frac{d}{dt}\begin{pmatrix} c \\ \dot{c} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\eta^2 & 0 \end{pmatrix}\begin{pmatrix} c \\ \dot{c} \end{pmatrix} + \begin{pmatrix} 0 \\ \eta^2 \end{pmatrix} z$$

- the ZMP is now the **input**

- the LIP is **unstable**! how do we guarantee that we do not get a divergent CoM trajectory?

MPC on the LIP model

- decompose the LIP in **stable** and **unstable** dynamics

$$x_s = c - \dot{c}/\eta$$
$$x_u = c + \dot{c}/\eta$$

- we want to impose the condition **at every MPC iteration**

$$x_u^k = \eta \int_{t_k}^{\infty} e^{-\eta(\tau - t_k)} z(\tau) d\tau$$

stability constraint

the stability constraint

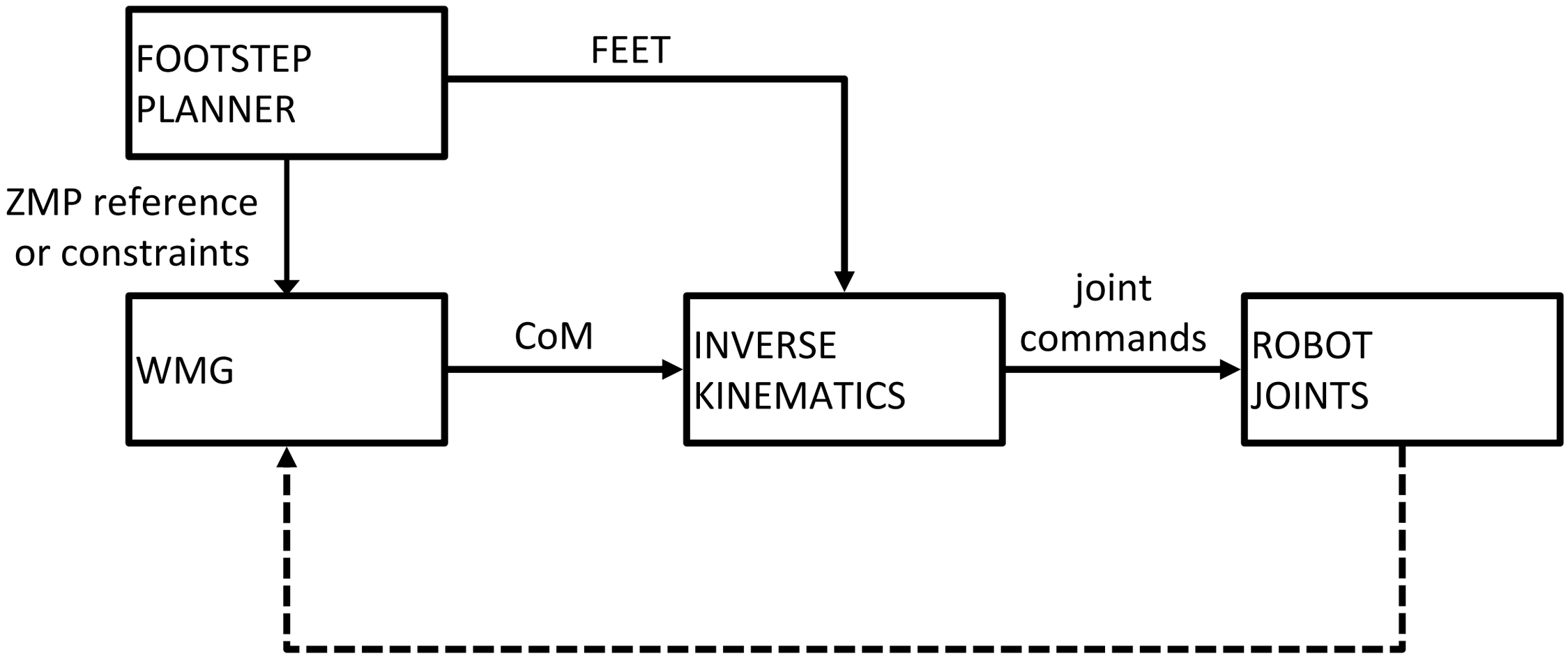
current state → $x_u^k = \eta \int_{t_k}^{\infty} e^{-\eta(\tau - t_k)} z(\tau) d\tau$ ← predicted ZMP

- the integral requires the predicted ZMP trajectory up to infinity, but MPC has a **finite** prediction horizon
- we **conjecture** the ZMP after the horizon (e.g., with the footstep plan)

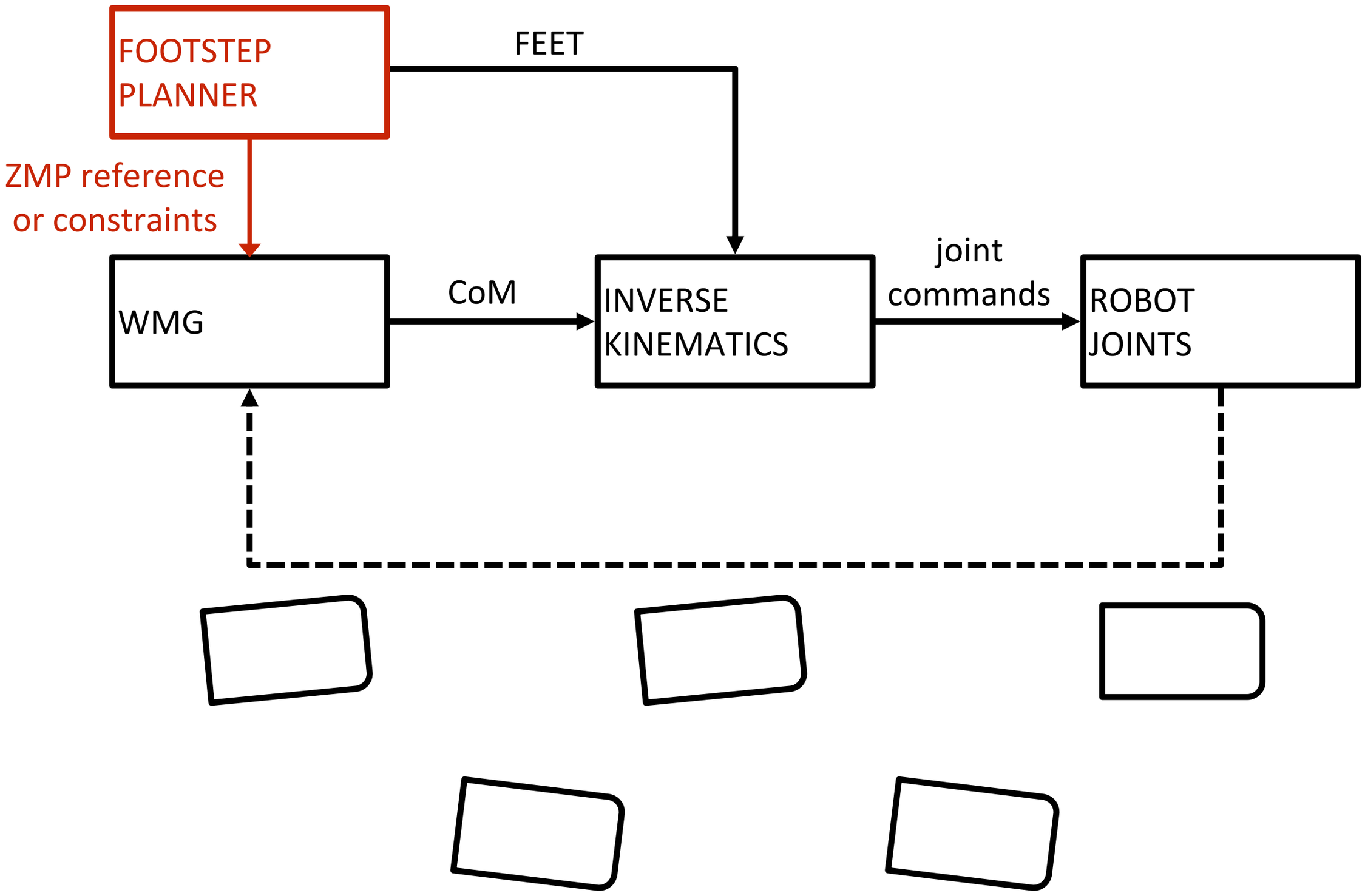
⇒ **recursive feasibility**

⇒ **stability**

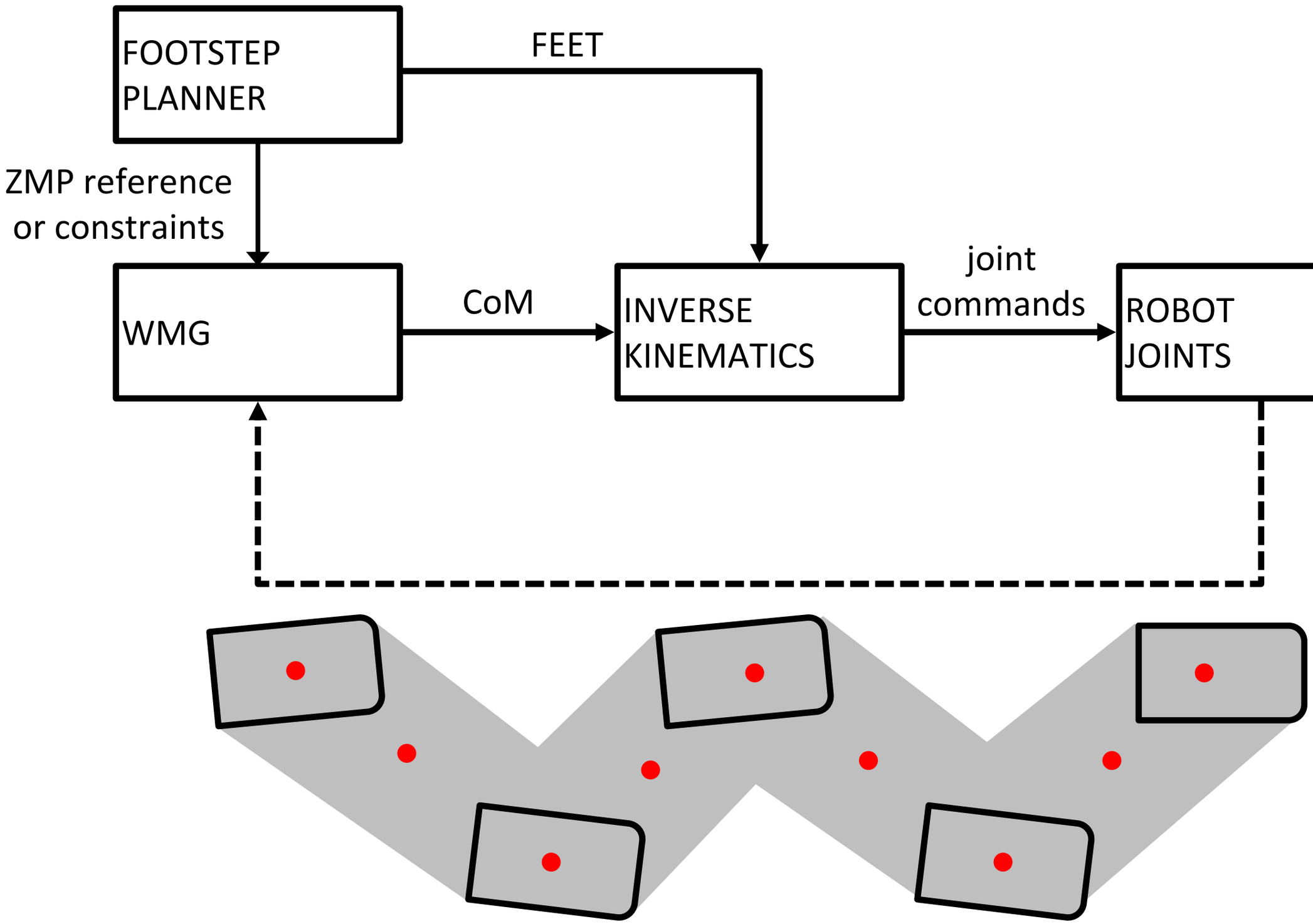
walking pattern generator (preview control)



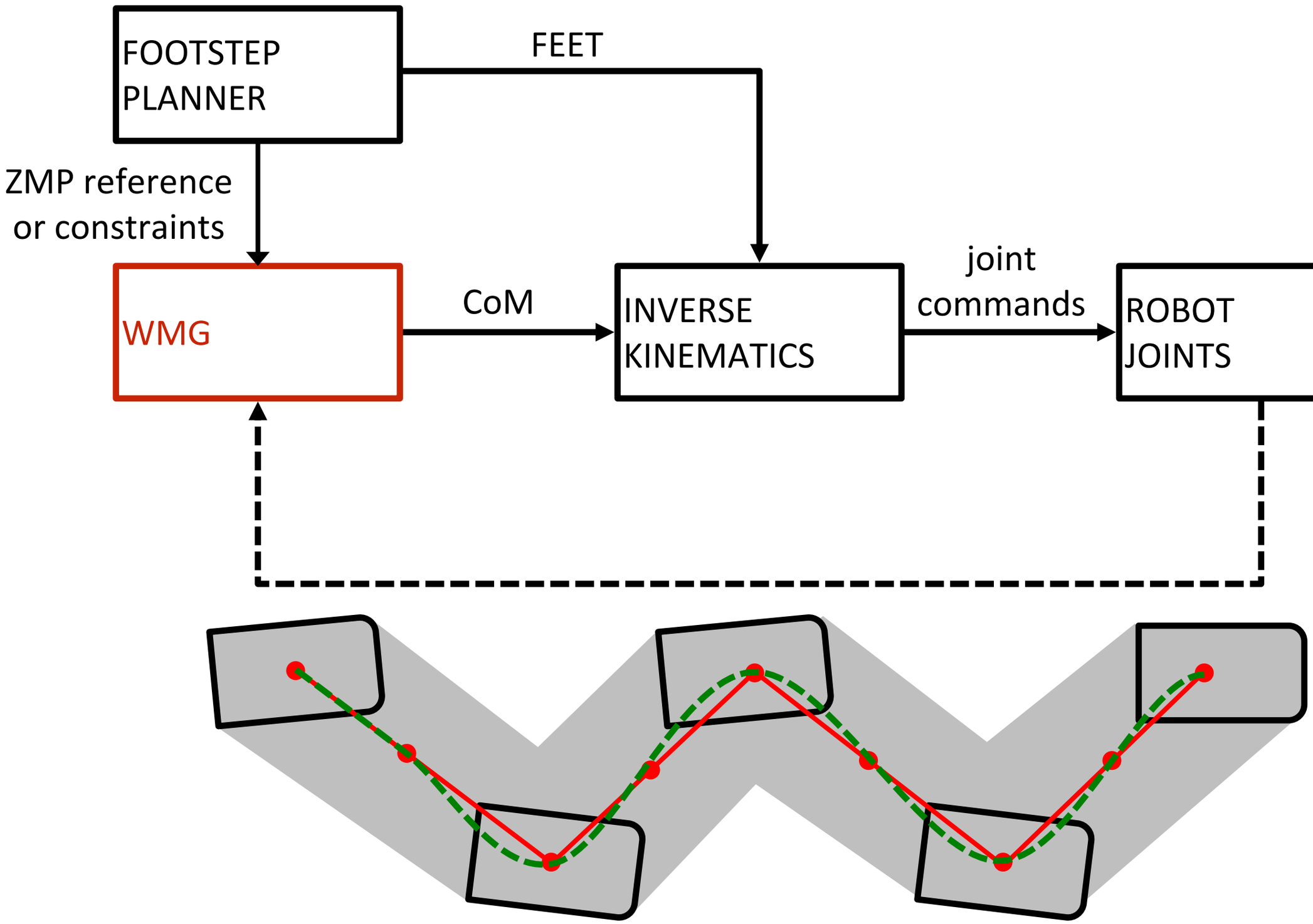
walking pattern generator



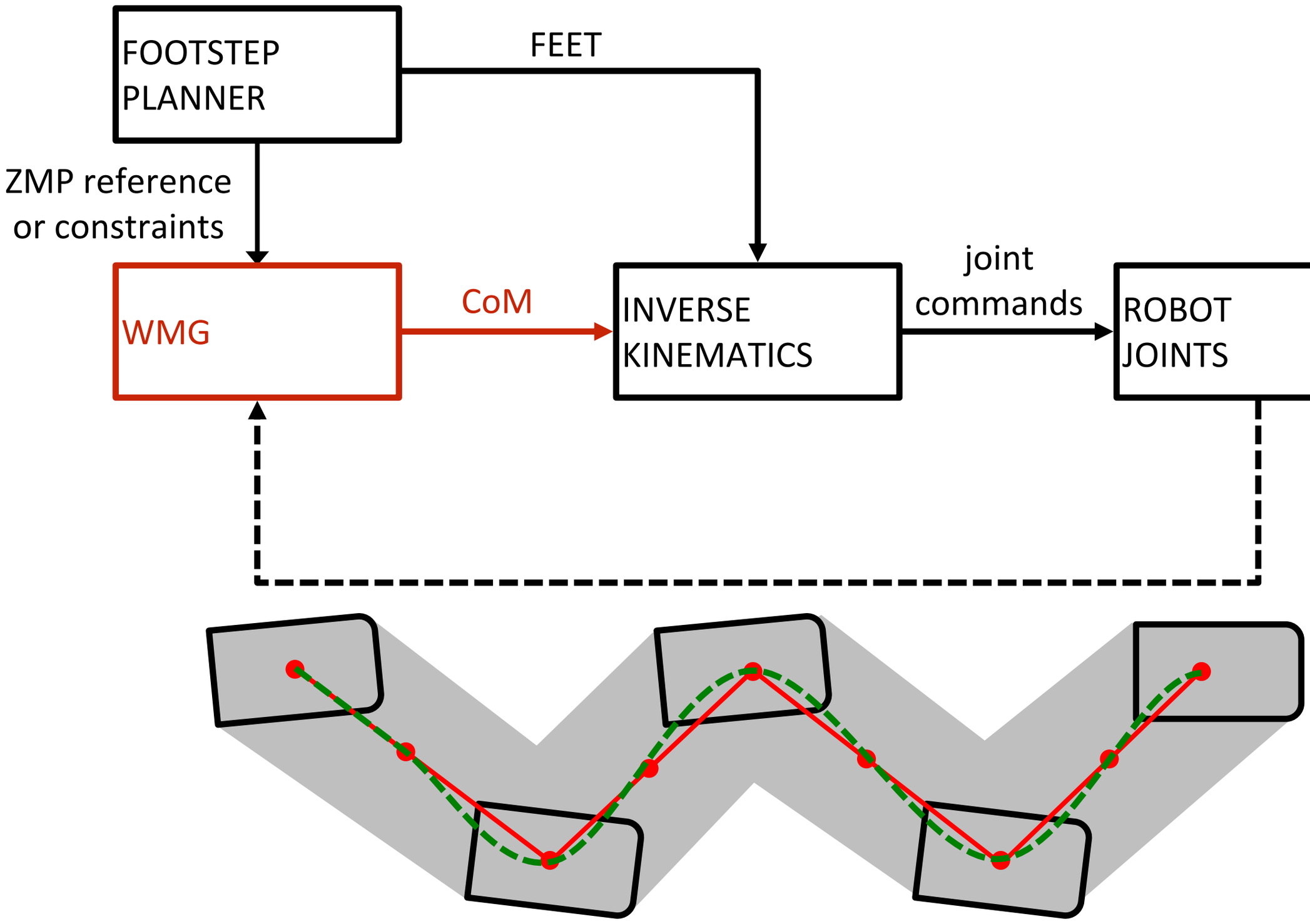
walking pattern generator



walking pattern generator



walking pattern generator



kinematic control

- how do we make the robot execute the desired CoM trajectory?
- simple solution: **kinematic tracking**

$$\boldsymbol{c} = \begin{pmatrix} c^x \\ c^y \\ c^z \end{pmatrix}, \quad \theta = \begin{pmatrix} \theta^x \\ \theta^y \\ \theta^z \end{pmatrix}$$

position of the CoM and orientation of the torso

$$\boldsymbol{f} = \begin{pmatrix} f^x \\ f^y \\ f^z \end{pmatrix}, \quad \phi = \begin{pmatrix} \phi^x \\ \phi^y \\ \phi^z \end{pmatrix}$$

position and orientation of the swing foot

everything is expressed wrt to the **current support foot**

kinematic control

- differential kinematics

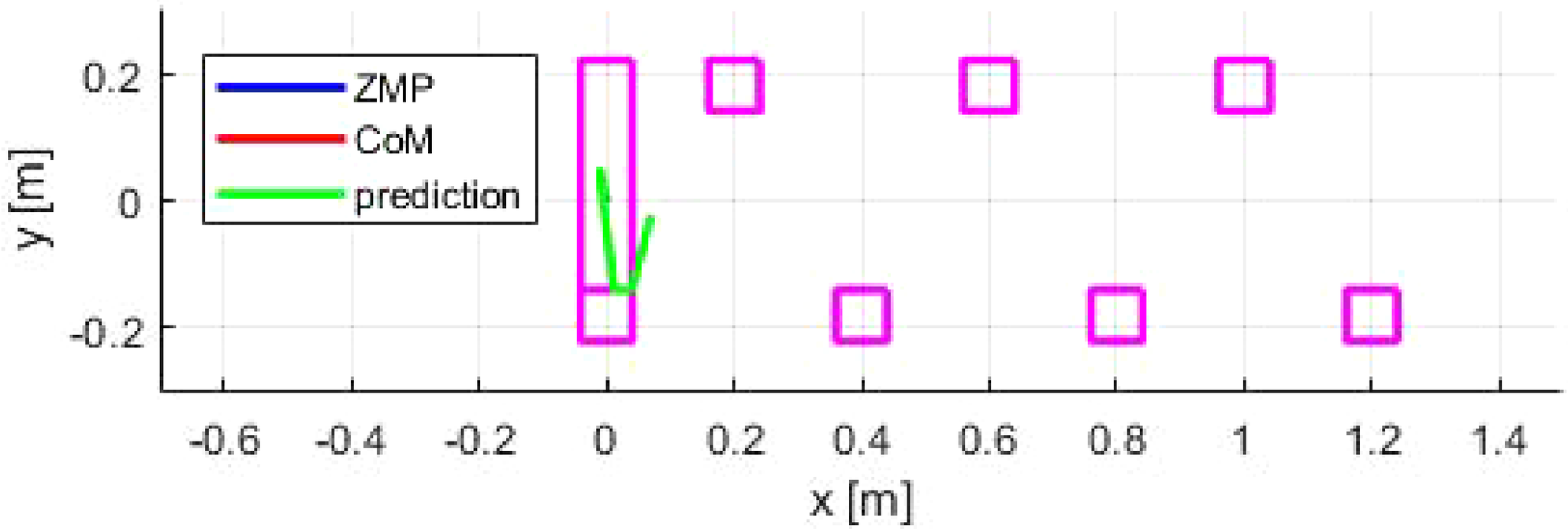
$$\dot{\boldsymbol{c}} = J_c(\boldsymbol{q})\dot{\boldsymbol{q}}$$
$$\dot{\boldsymbol{\theta}} = J_\theta(\boldsymbol{q})\dot{\boldsymbol{q}}$$
$$\dot{\boldsymbol{f}} = J_f(\boldsymbol{q})\dot{\boldsymbol{q}}$$
$$\dot{\boldsymbol{\phi}} = J_\phi(\boldsymbol{q})\dot{\boldsymbol{q}}$$

$$\begin{pmatrix} \dot{\boldsymbol{c}} \\ \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{f}} \\ \dot{\boldsymbol{\phi}} \end{pmatrix} = \begin{pmatrix} J_c(\boldsymbol{q}) \\ J_\theta(\boldsymbol{q}) \\ J_f(\boldsymbol{q}) \\ J_\phi(\boldsymbol{q}) \end{pmatrix} \dot{\boldsymbol{q}}$$

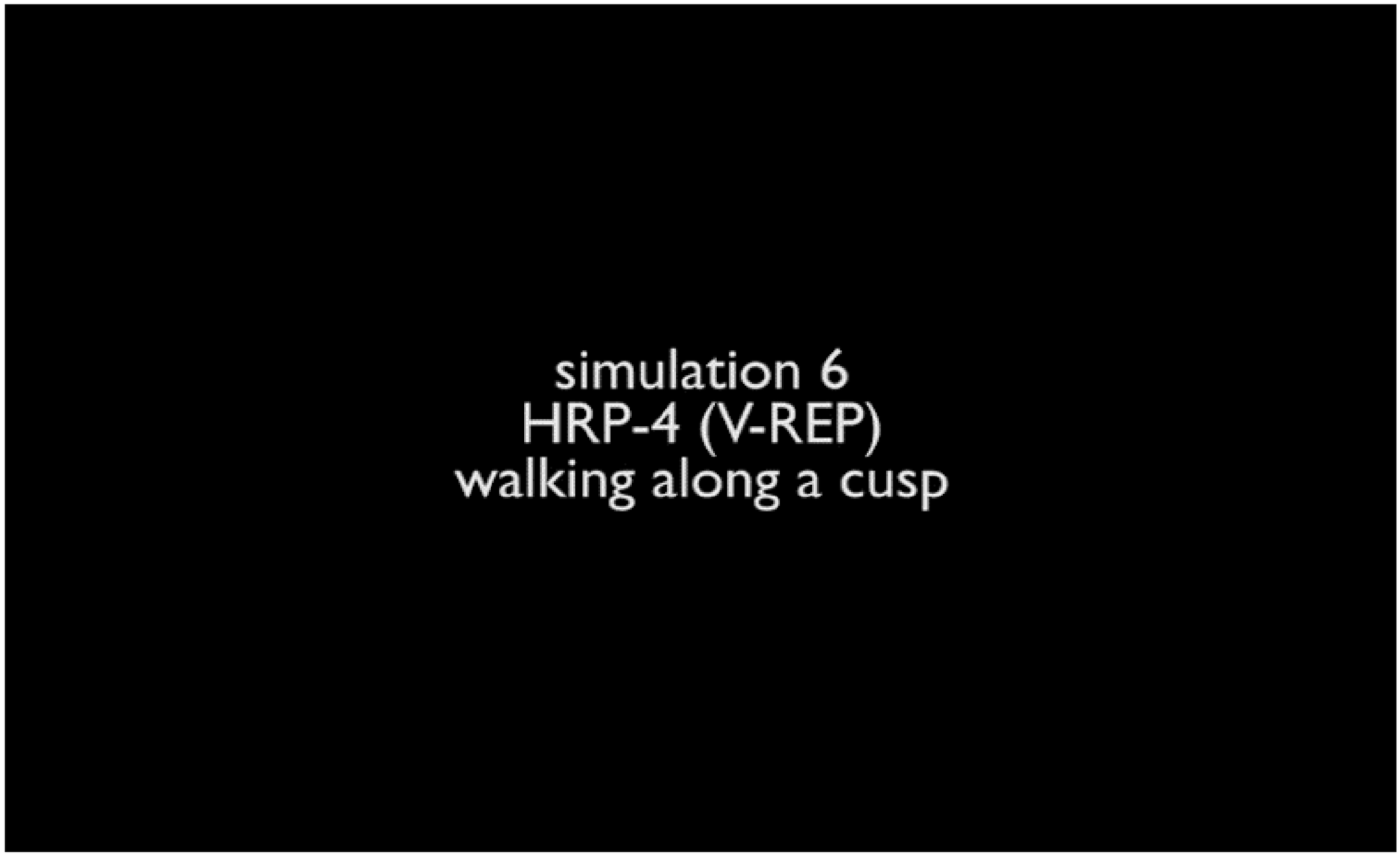
velocity task — stack of jacobians

- can be written as a **stack of tasks** $\dot{\boldsymbol{t}} = J_t(q)\dot{\boldsymbol{q}}$
- the classic solution is the **pseudoinverse** $\dot{\boldsymbol{q}} = J_t^{\#}(q)\dot{\boldsymbol{t}}_{des}$
- add a **position error** to avoid **drifting** $\dot{\boldsymbol{q}} = J_t^{\#}(q)\left(\dot{\boldsymbol{t}}_{des} + k(\boldsymbol{t}_{des} - \boldsymbol{t})\right)$

examples – MPC gait generation



examples – simulations and experiments



other relevant topics

- **robust** gait generation (for disturbances)
- vertical CoM motion (for **uneven ground**)
- more accurate models (e.g., with **angular momentum**)
- **footstep planning** in complex environments

some more examples – robust gait generation

some more examples – uneven ground