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

- we have developed models that describe humanoid robot dynamics with varying levels of complexity:
how do we use these models to **make a robot walk**?

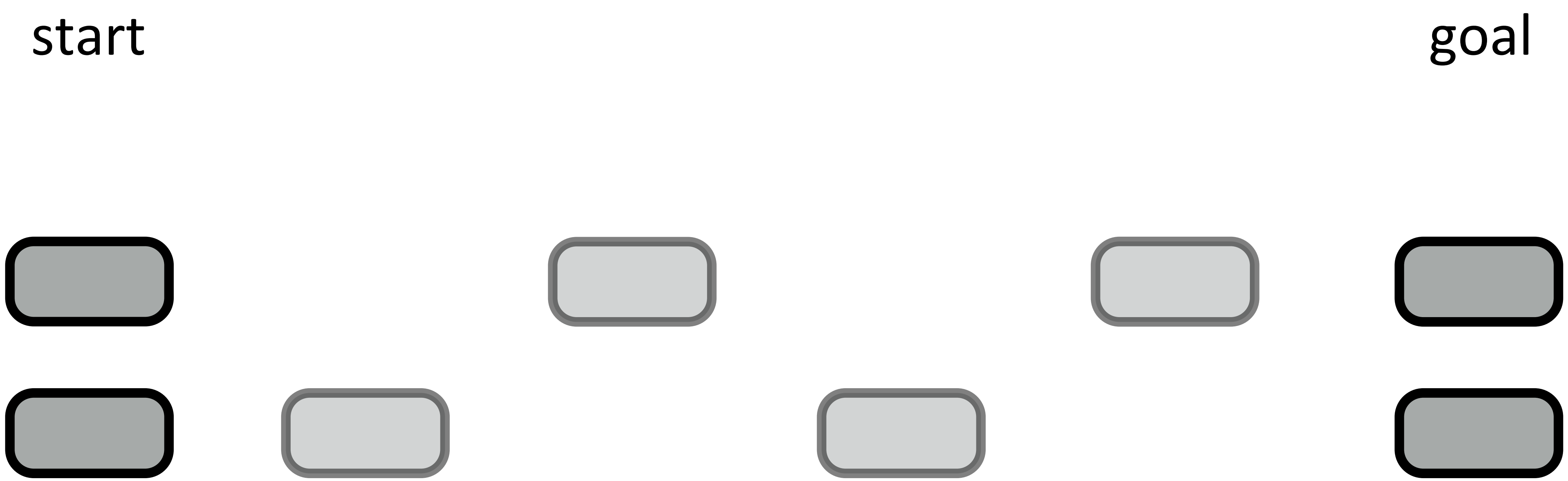
- in these slides we will take a look at a few techniques that can be used to **generate a walking gait** using the LIP model

- main topics will be:
  - stable inversion
  - model predictive control
  - control scheme architecture and kinematic tracking
  - examples

ZMP-based gait generation

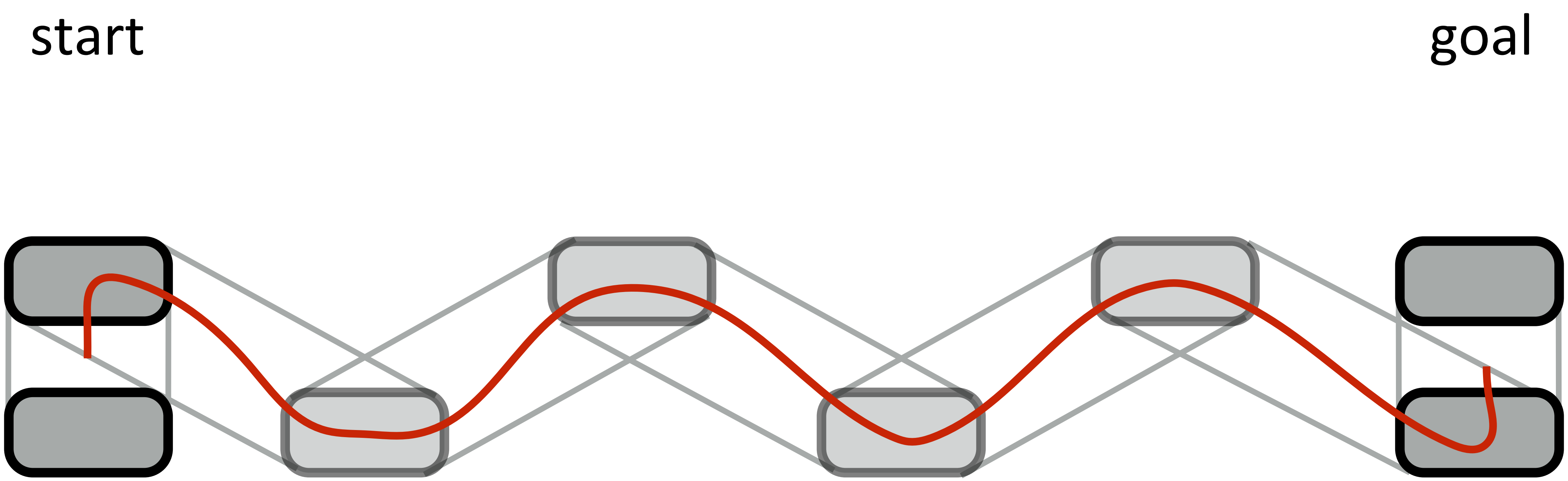
an algorithm based on the strategy: keep the ZMP inside the Support Polygon (SP)

1. plan the footsteps…



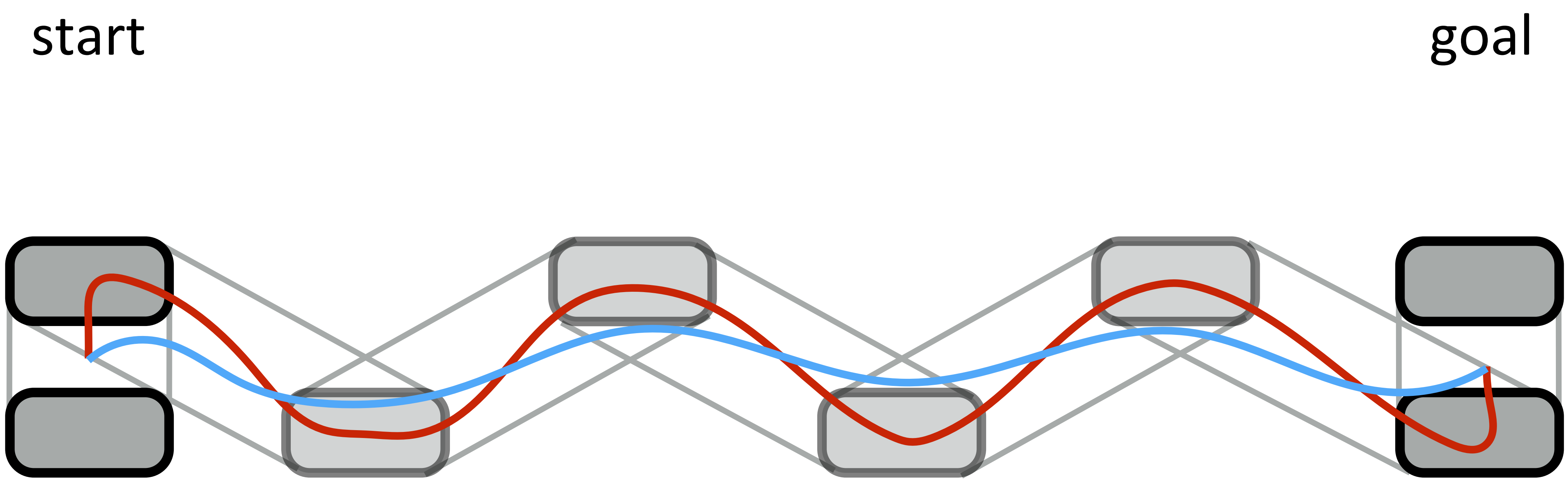
ZMP-based gait generation

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



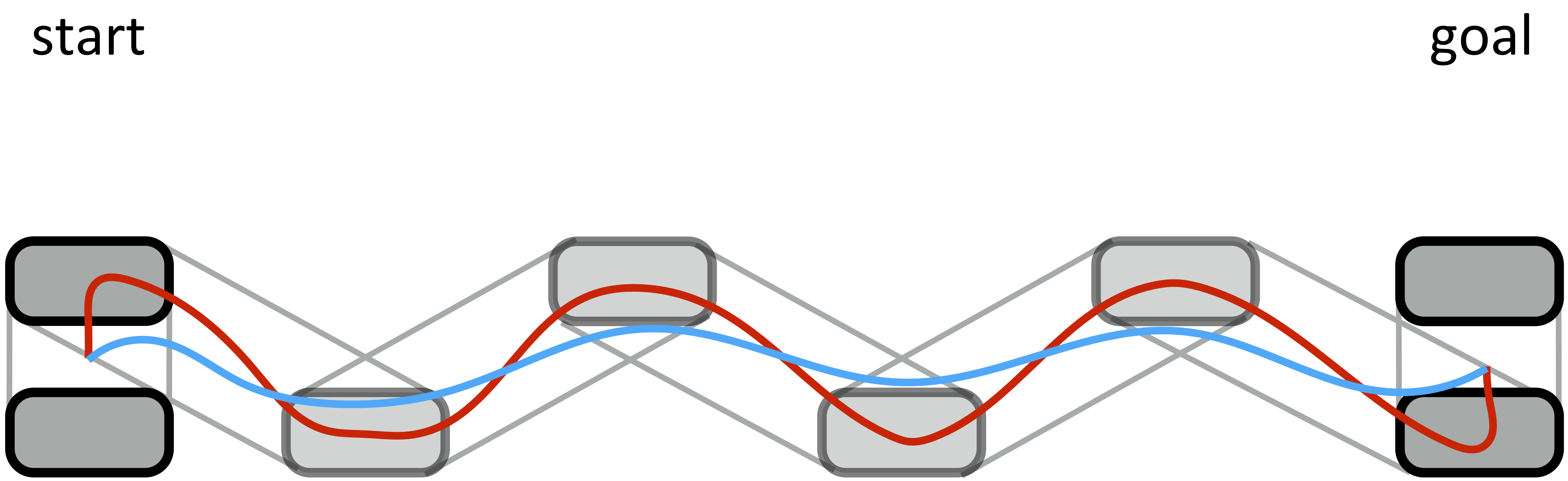
ZMP-based gait generation

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



ZMP-based gait generation

…4. track the CoM trajectory



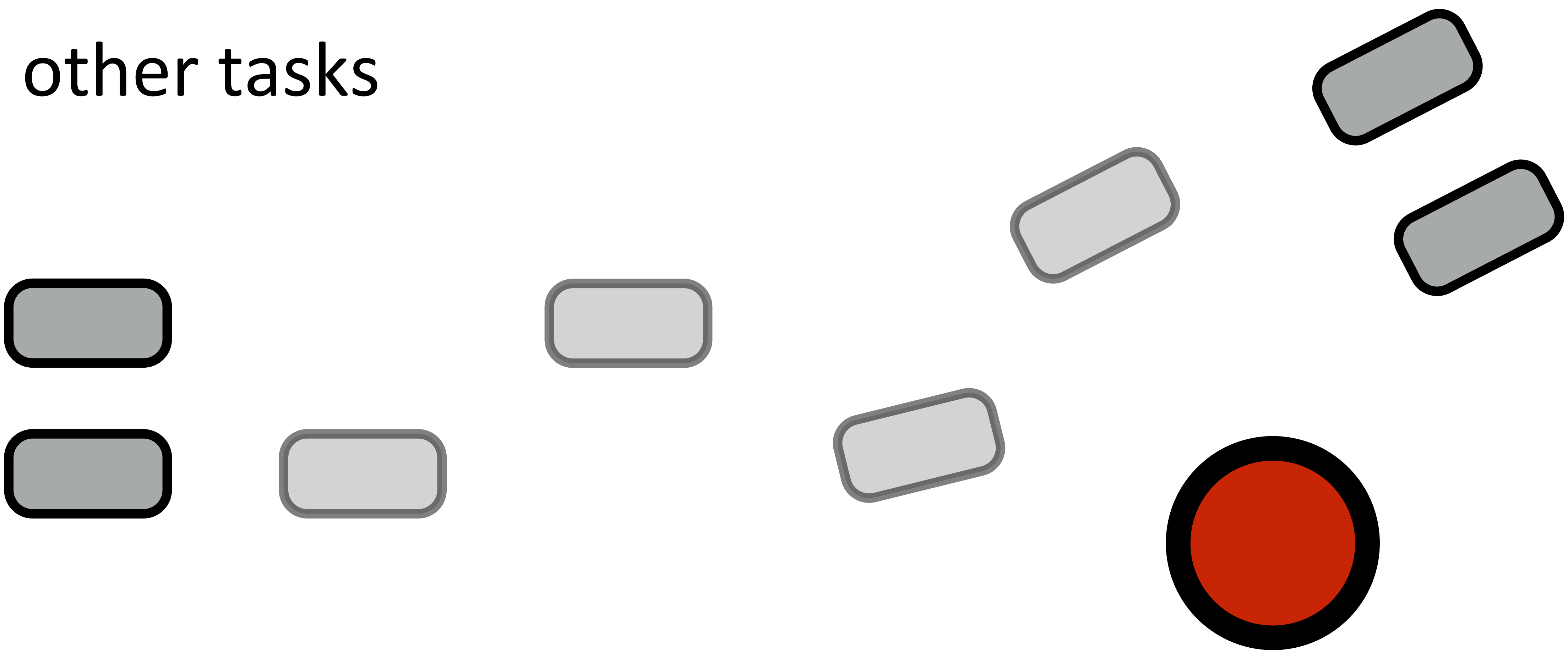
algorithm steps in detail

1 plan the footsteps (offline)

timing and lengths (desired speed)

obstacles (obstacle avoidance)

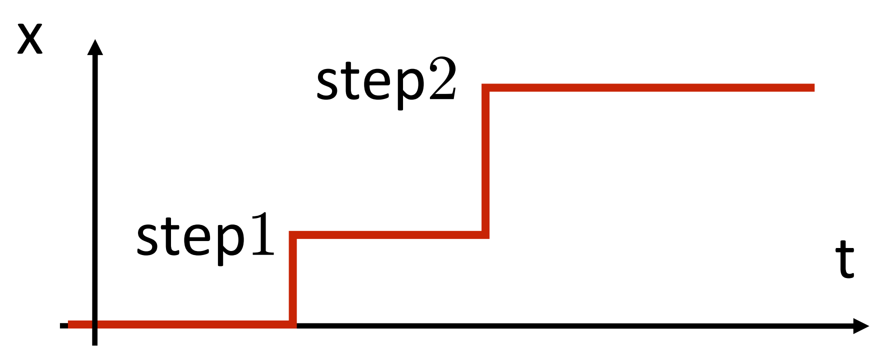
other tasks



algorithm steps in detail

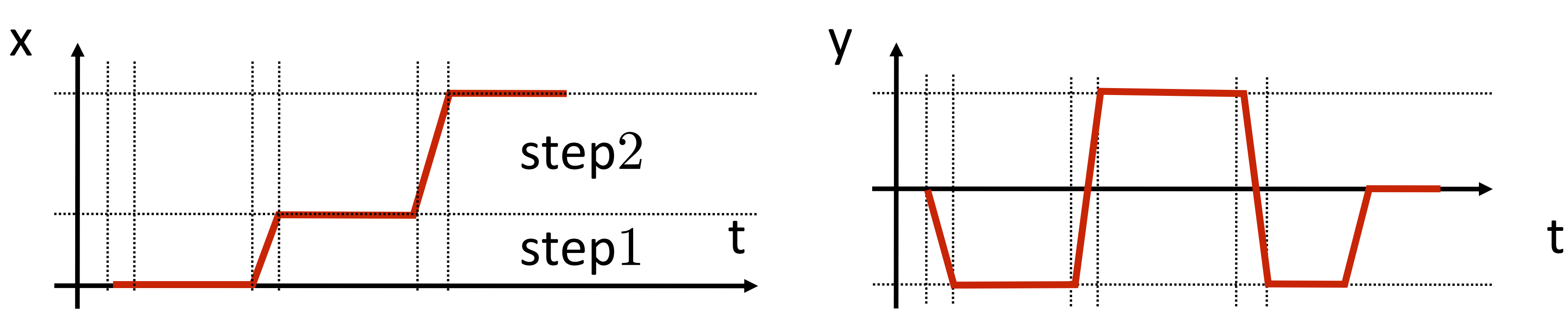
2 plan ZMP trajectory

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

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

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 $\longrightarrow$ 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 $\longrightarrow$ ZMP

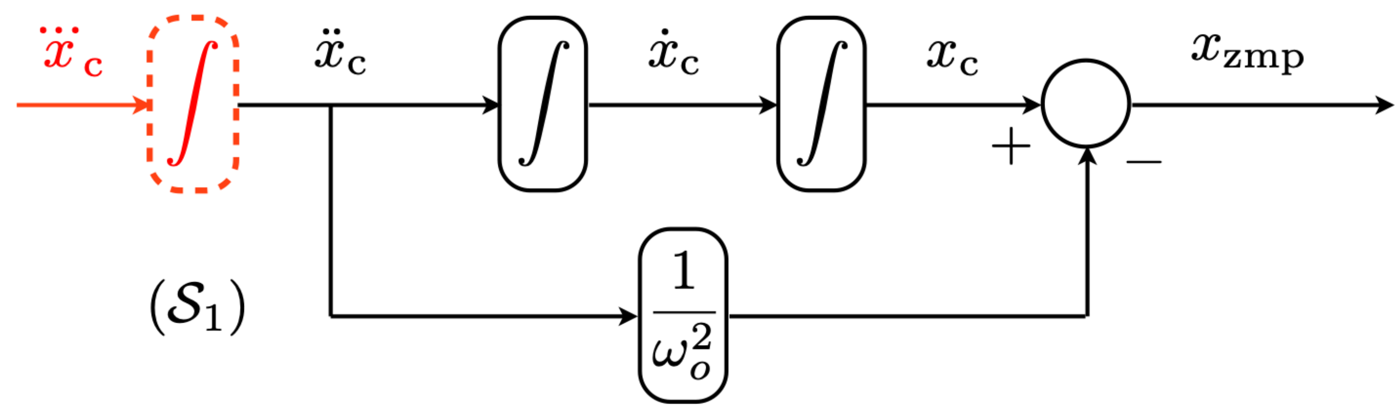
instability problem: a control perspective

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

reference ZMP

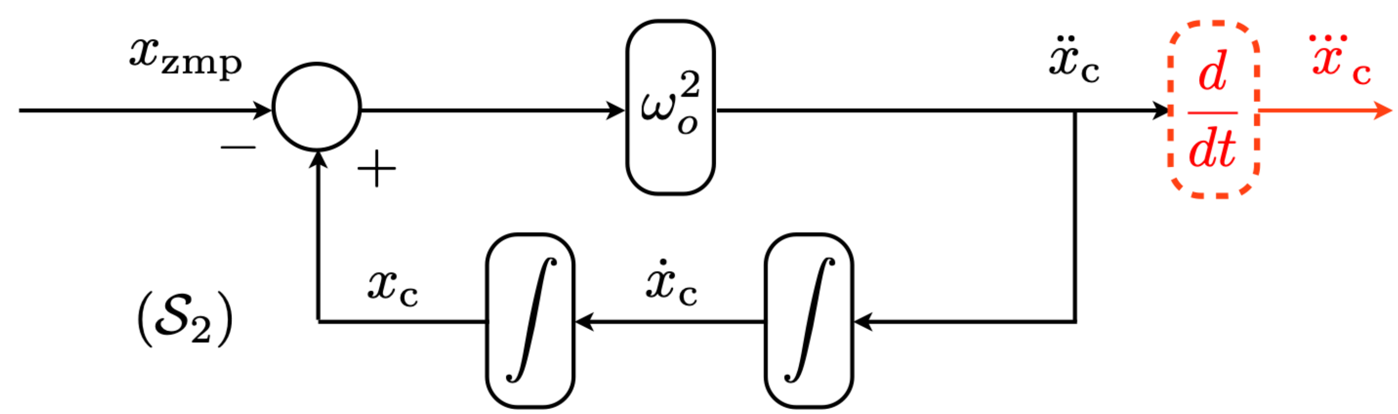
CoM → ZMP

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

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

also known as:
Divergent Component of Motion (DCM)
Capture Point (CP)
Extrapolated Center of Mass (xCoM)

- 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

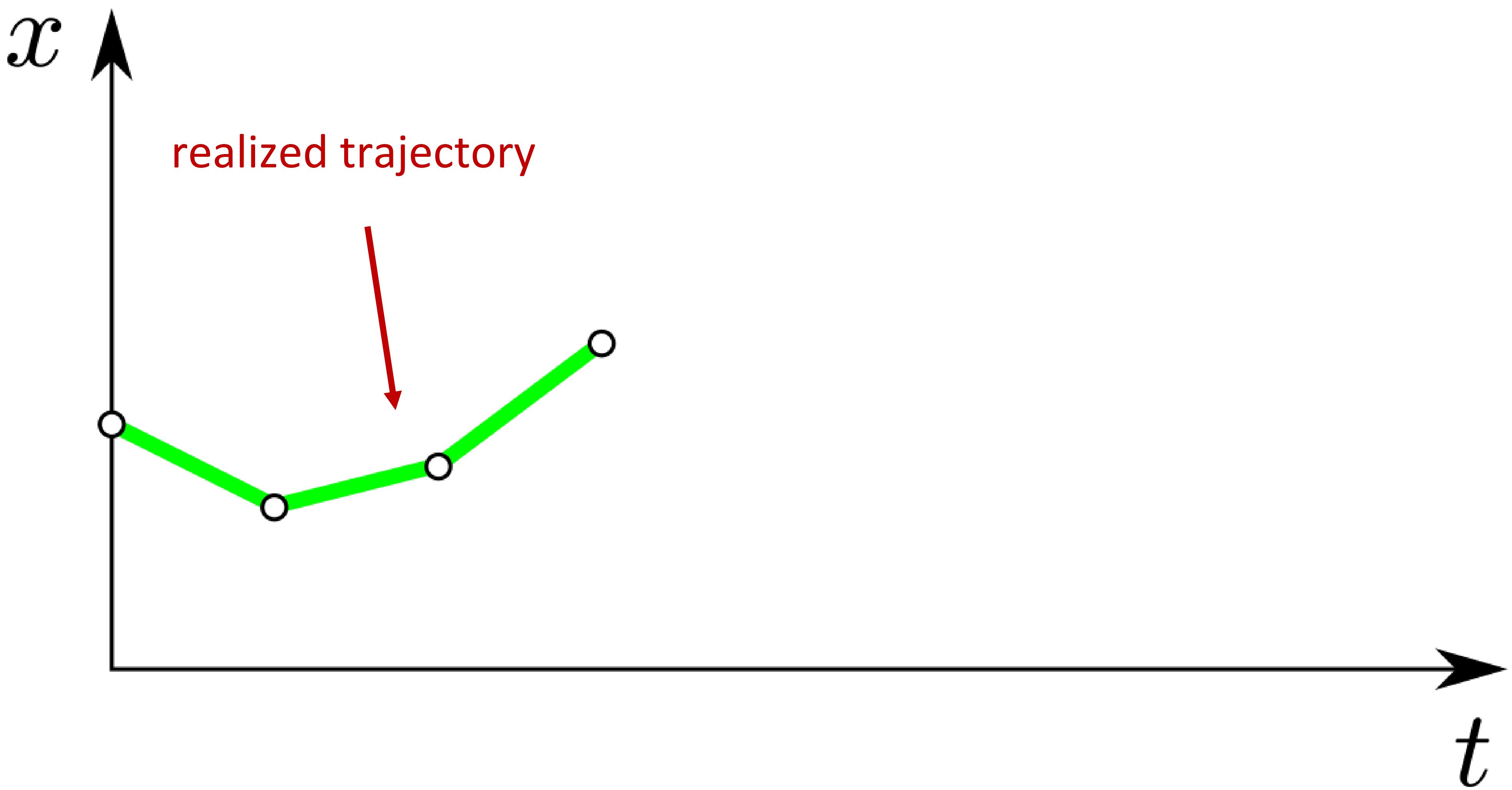
linear MPC

- **Model Predictive Control** is a general control technique, especially useful on underactuated systems with constraints

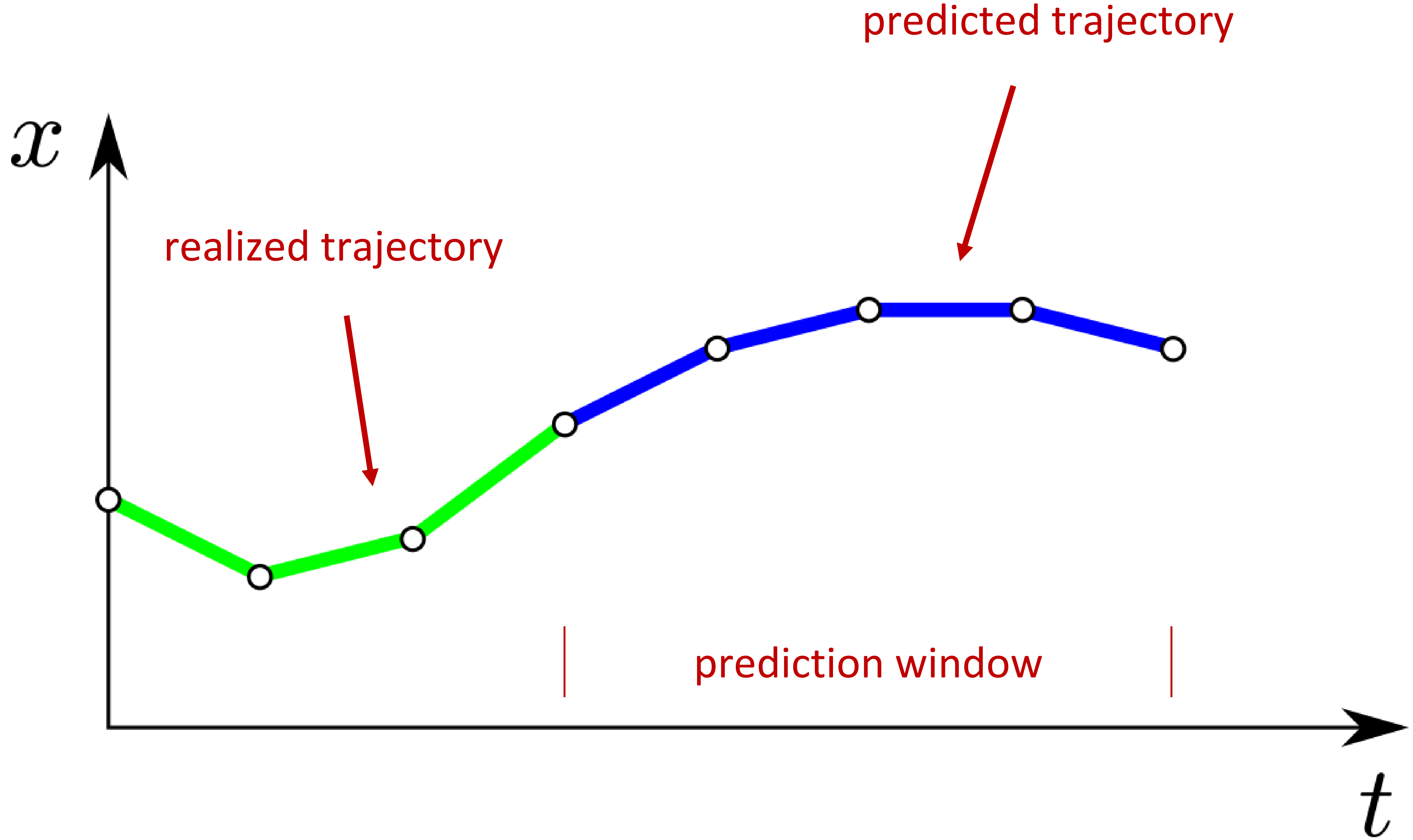
- it uses a model to forecast the evolution of the system over a short **prediction window**

- the window shifts forward at each control time-step: **receding horizon control**

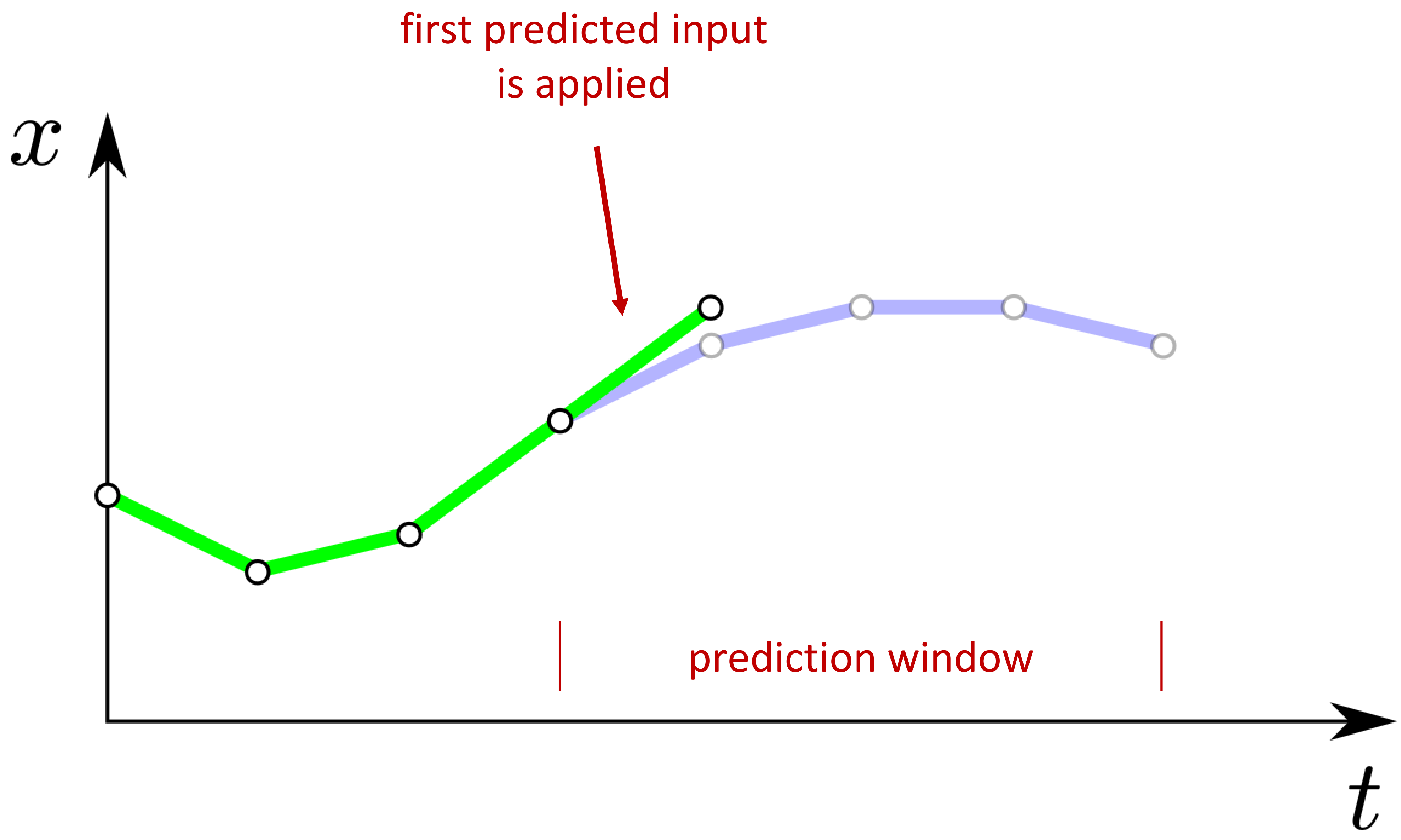
linear MPC – example



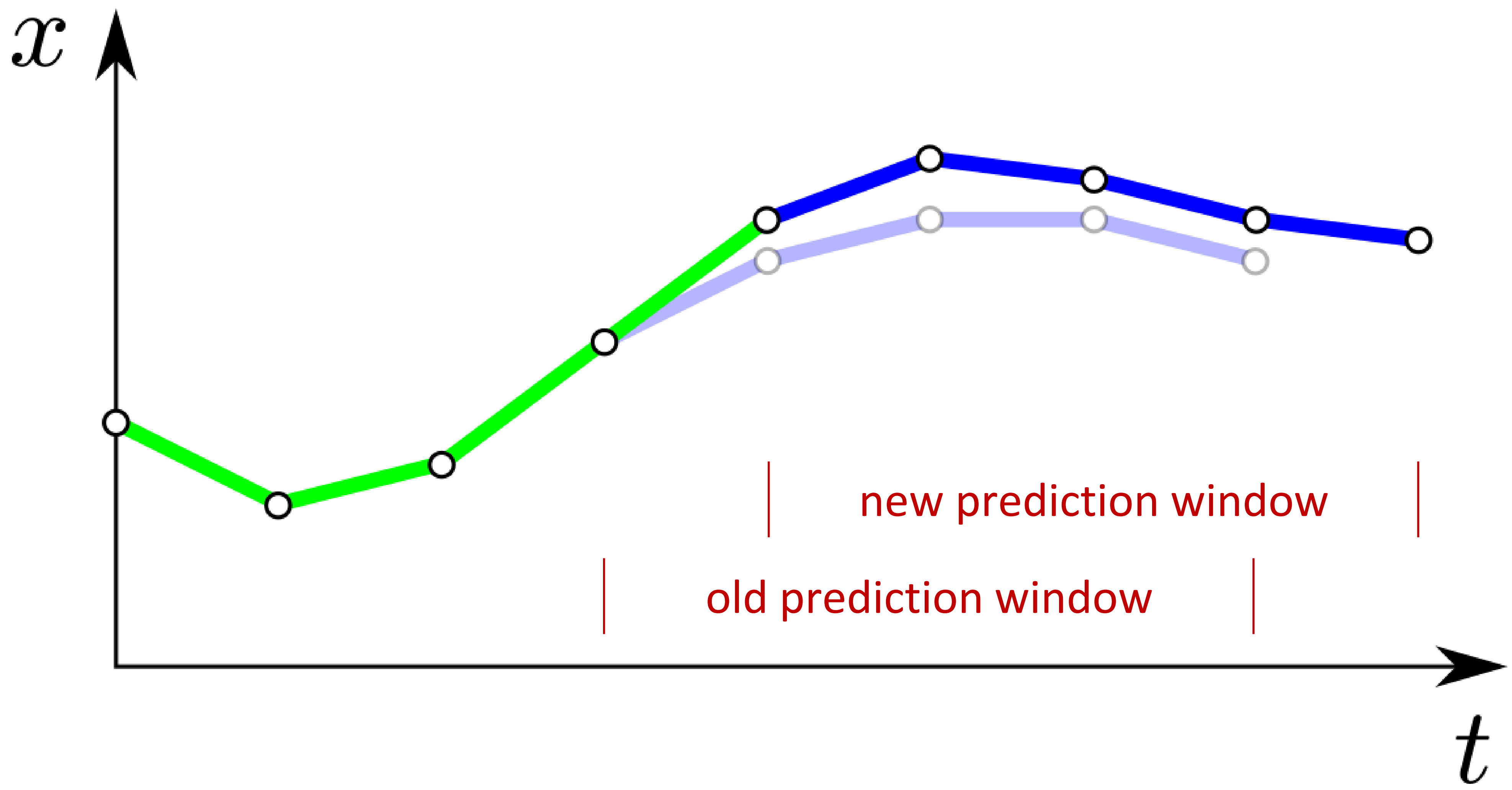
linear MPC – example



linear MPC – example



linear MPC – example



linear MPC

- example: regulation using linear MPC

- the **prediction model** is given by the linear discrete-time system

$$x_{k+1} = Ax_k + Bu_k \qquad \begin{array}{l} x \in \mathbb{R}^n \\ u \in \mathbb{R}^m \end{array}$$

linear MPC

- we want to use this model to forecast the evolution of the system over a **prediction horizon** of N discrete time-steps

- starting at k, let's move ahead two time-steps, then perform some substitutions

$$x_{k+1} = Ax_k + Bu_k$$
$$x_{k+2} = Ax_{k+1} + Bu_{k+1}$$

$$\Downarrow$$

$$x_{k+1} = Ax_k + Bu_k$$
$$x_{k+2} = A^2x_k + ABu_k + Bu_{k+1}$$

linear MPC

- if we keep going foward in time, we arrive at the following

$$x_{k+1} = Ax_k + Bu_k$$
$$x_{k+2} = A^2x_k + ABu_k + Bu_{k+1}$$
$$x_{k+3} = A^3x_k + A^2Bu_k + ABu_{k+1} + Bu_{k+2}$$
$$\vdots$$

in which the i-th term looks like

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

which is the discrete equivalent of a **convolution**

linear MPC

- in matrix form

$$\underbrace{\begin{pmatrix} x_{k+1} \\ x_{k+2} \\ x_{k+3} \\ \vdots \\ x_{k+N} \end{pmatrix}}_{X_{k+1}} = \underbrace{\begin{pmatrix} B & 0 & 0 & & 0 \\ AB & B & 0 & \dots & 0 \\ A^2B & AB & B & & 0 \\ & \vdots & & \ddots & \\ A^{N-1}B & A^{N-2}B & A^{N-3}B & \dots & B \end{pmatrix}}_{\bar{S}} \underbrace{\begin{pmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ \vdots \\ u_{k+N-1} \end{pmatrix}}_{U_k} + \underbrace{\begin{pmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^N \end{pmatrix}}_{\bar{T}} x_k$$

$$X_{k+1} = \bar{S}U_k + \bar{T}x_k$$

- this expresses the vector of **predicted states** X_{k+1} in terms of the **current state** x_k and the vector of **predicted inputs** U_k

linear MPC – cost function

- cost function

$$J = \sum_{j=0}^{N-1} \left[x_{k+i+1}^T Q x_{k+i+1} + u_{k+i}^T R u_{k+i} \right]$$

- goal: find a sequence U_k that minimizes J, i.e., that steers the state x to the origin "optimally"
- similar to LQR, but the horizon is **finite**!

linear MPC – cost function

- cost function

$$J = \begin{pmatrix} x_{k+1} \\ \vdots \\ x_{k+N} \end{pmatrix}^T \underbrace{\begin{pmatrix} Q & & 0 \\ & \ddots & \\ 0 & & Q \end{pmatrix}}_{\bar{Q}} \begin{pmatrix} x_{k+1} \\ \vdots \\ x_{k+N} \end{pmatrix} + \begin{pmatrix} u_k \\ \vdots \\ u_{k+N-1} \end{pmatrix}^T \underbrace{\begin{pmatrix} R & & 0 \\ & \ddots & \\ 0 & & R \end{pmatrix}}_{\bar{R}} \begin{pmatrix} u_k \\ \vdots \\ u_{k+N-1} \end{pmatrix}$$

$$J = X_{k+1}^T \bar{Q} X_{k+1} + U_k^T \bar{R} U_k$$

- we can also write the cost function in terms of vectors U_k and X_{k+1} using block-diagonal weight matrices

linear MPC – cost function

- substitute the state prediction term

$$J = \underbrace{\left(\bar{S}U_k + \bar{T}x_k\right)^T}_{X_{k+1}^T} \bar{Q} \underbrace{\left(\bar{S}U_k + \bar{T}x_k\right)}_{X_{k+1}} + U_k^T \bar{R} U_k$$

$$J = \underbrace{U_k^T \left(\bar{S}^T \bar{Q} \bar{S} + \bar{R}\right) U_k}_{\text{quadratic term}} + \underbrace{2U_k^T \bar{S}^T \bar{T} x_k}_{\text{linear term}} + \underbrace{\cancel{x_k^T \bar{T}^T \bar{Q} \bar{T} x_k}}_{\substack{\text{constant term} \\ \text{(does not change the min)}}}$$

- we can omit the constant term (independent of U_k) as it only changes the min value, not the min location

linear MPC – cost function

- minimizing the cost function

$$\min \frac{1}{2} U_k^T H U_k + U_k^T F x_k$$

$$H = 2\bar{S}^T \bar{Q} \bar{S} + 2\bar{R}$$

$$F = 2\bar{S}^T \bar{T}$$

- with no constraints, this can be solved by zeroing the gradient

$$\nabla J = H U_k^* + F x_k = 0 \implies U_k^* = -H^{-1} F x_k$$

linear MPC – unconstrained case

$$u_k^* = -I_{\text{sel}} H^{-1} F x_k$$

- we isolate the first input of the optimal sequence using a selection matrix I_{sel}
- this input is **applied** to the system, then the whole process is repeated starting from the new state x_{k+1}
- when no constraint are enforced, MPC is simply **linear state feedback**

linear MPC – constraints

- add linear **constraints** on output and input

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

- it would be nice if we could turn the problem in the form

$$\begin{aligned} &\min \frac{1}{2} U_k^T H U_k + U_k^T F x_k \\ &\text{s.t.} \quad \bar{A} U_k \le \bar{b} \end{aligned}$$

because this is a standard **Quadratic Programming** problem and can be solved using off-the-shelf software

linear MPC – constraints

- input constraints $$u \le u_{\max}$$ $$-u \le -u_{\min}$$

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & -1 \end{pmatrix} U_k \le \begin{pmatrix} u_{\max} \\ u_{\max} \\ \vdots \\ u_{\max} \\ -u_{\min} \\ -u_{\min} \\ \vdots \\ -u_{\min} \end{pmatrix}$$

linear MPC – constraints

- we write the output in terms of predicted inputs using

$$y_{k+i} = CA^i x_k + \sum_{j=0}^{i-1} CA^{i-j-1} B u_{k+j}$$

- the upper output constraint is

$$\begin{pmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \dots & CB \end{pmatrix} U_k \leq \begin{pmatrix} y_{\max} \\ y_{\max} \\ \vdots \\ y_{\max} \end{pmatrix} - \begin{pmatrix} CA \\ CA^2 \\ \vdots \\ CA^N \end{pmatrix} x^k$$

and the lower output constraint can be written similarly using the minus sign trick (see input constraints)

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

this is similar to planning with optimal control, but the continuous replanning (trajectory is recomputed at each iteration) introduces a form of "robustness"

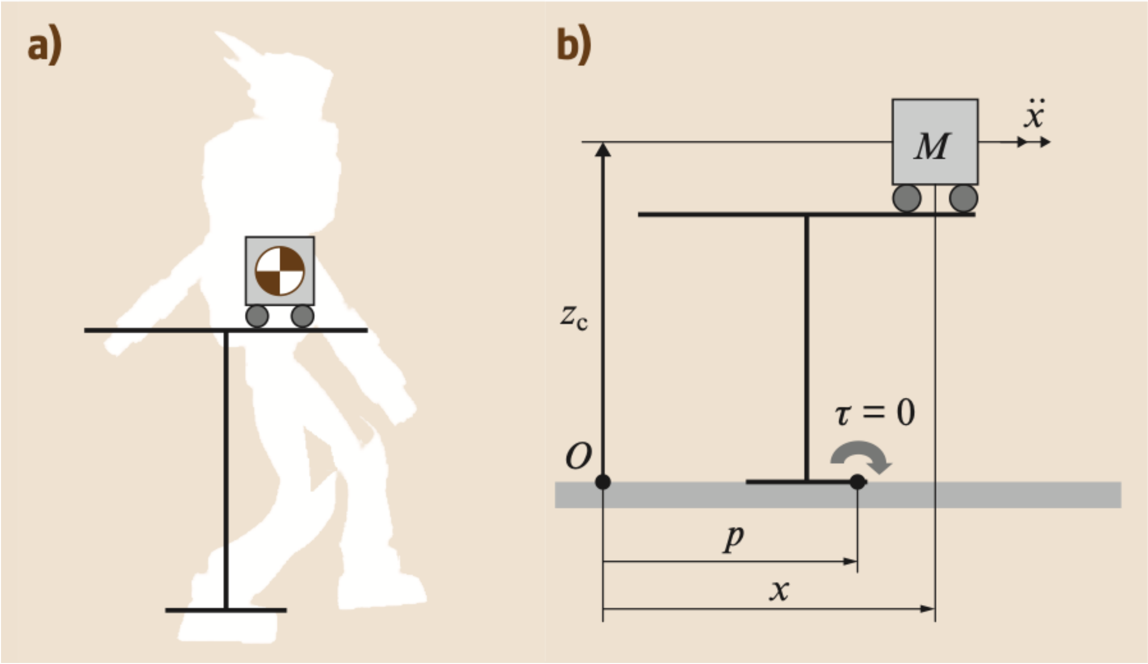
preview control

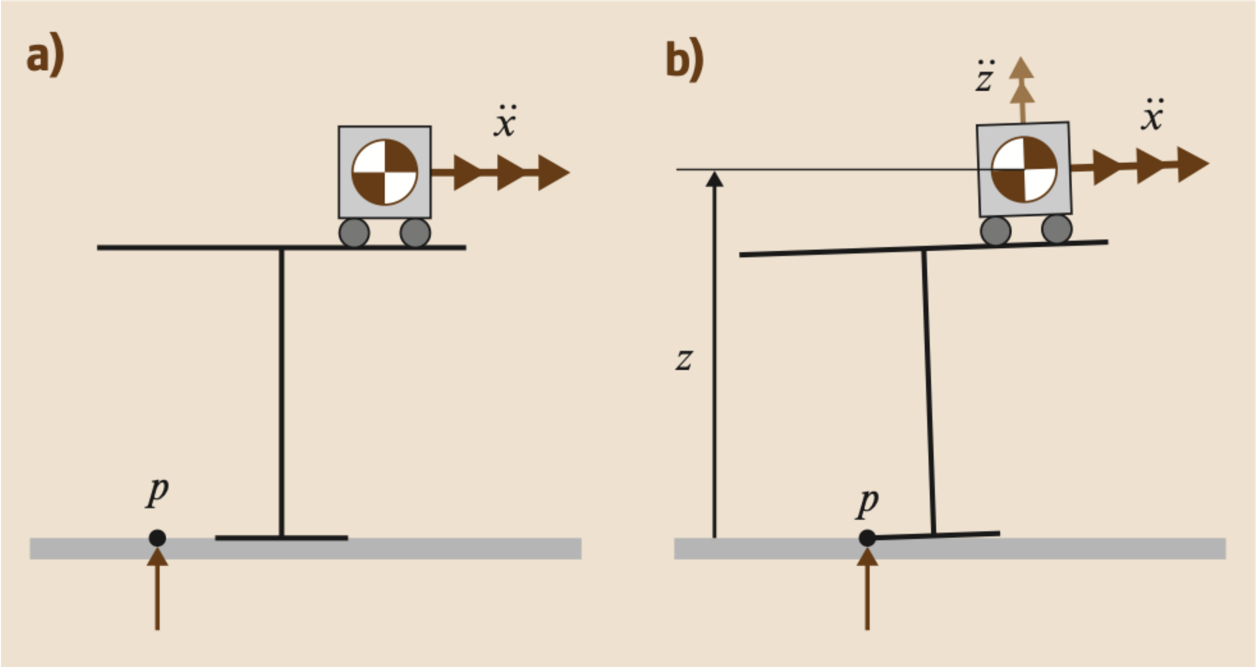
CoM trajectory generation can be seen as the

design of a ZMP tracking controller

CoM $\longrightarrow$ 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

- define a **cost function** that achieves tracking of the desired ZMP, while penalizing divergence of the CoM trajectory

- the simplest one: a weighted sum of two terms

$$J = \sum_{i=k}^{k+N-1} \underbrace{(\boldsymbol{z}_{i+1} - \boldsymbol{z}^{\text{ref}}_{i+1})^T \boldsymbol{Q} (\boldsymbol{z}_{i+1} - \boldsymbol{z}^{\text{ref}}_{i+1})}_{\text{tracking error minimization}} + \underbrace{\dddot{\boldsymbol{x}}_i^T \boldsymbol{R} \dddot{\boldsymbol{x}}_i}_{\text{input minimization}}$$

- this is unconstrained MPC, so the control law can be expressed as a **state feedback** linear control law in **closed form**

preview control

- requires **planning ahead**, but the reference trajectory far in the far future is weighted less (exponetially)

- it is computationally **very fast** (closed form) because of the absence of constraints

- drawback: the balance condition on the ZMP is **not guaranteed** (ZMP might exit the support polygon)

- a valid solution is obtained by proper **tuning** of the cost function weights, and by designing a **good ZMP reference**

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 time 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

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

- 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**

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

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

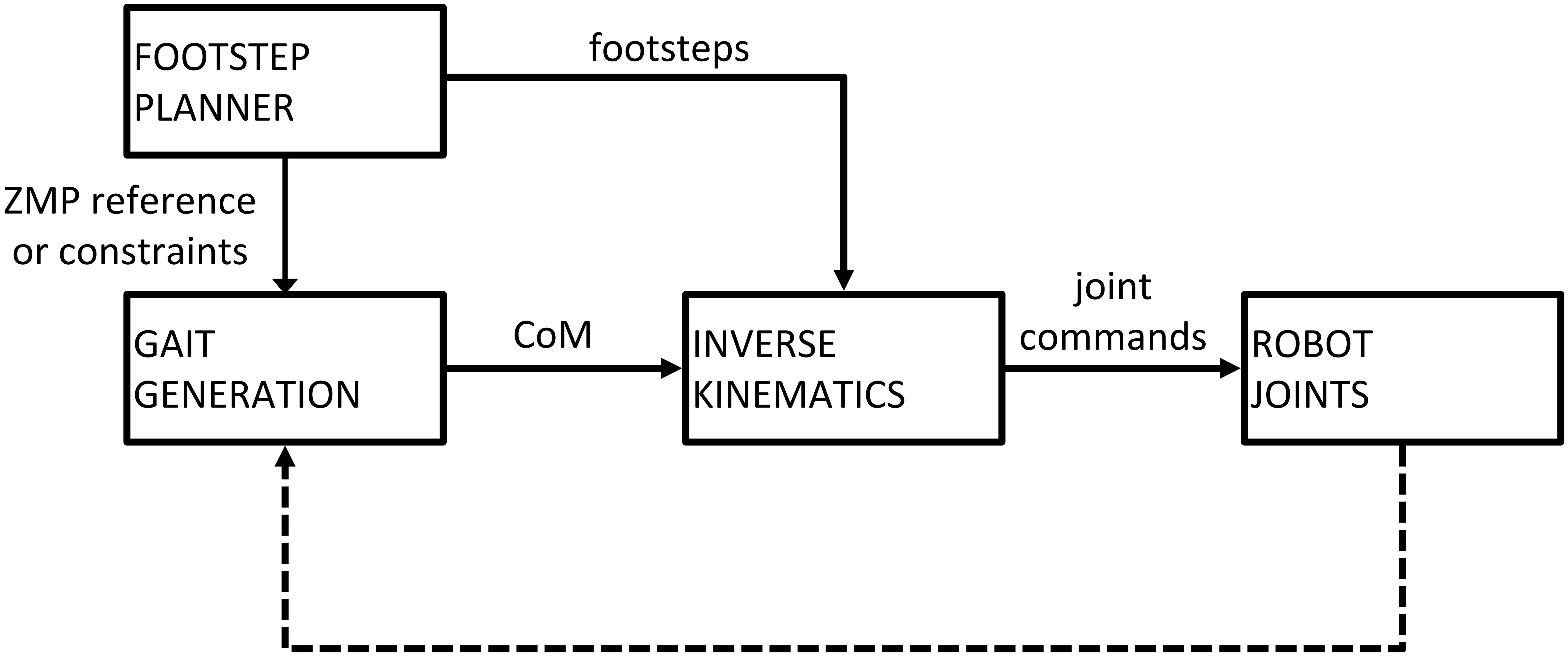
- the integral requires the predicted ZMP trajectory up to infinity, but MPC has a **finite** prediction horizon

- we **conjecture** the ZMP after the horizon (e.g., with the footstep plan)

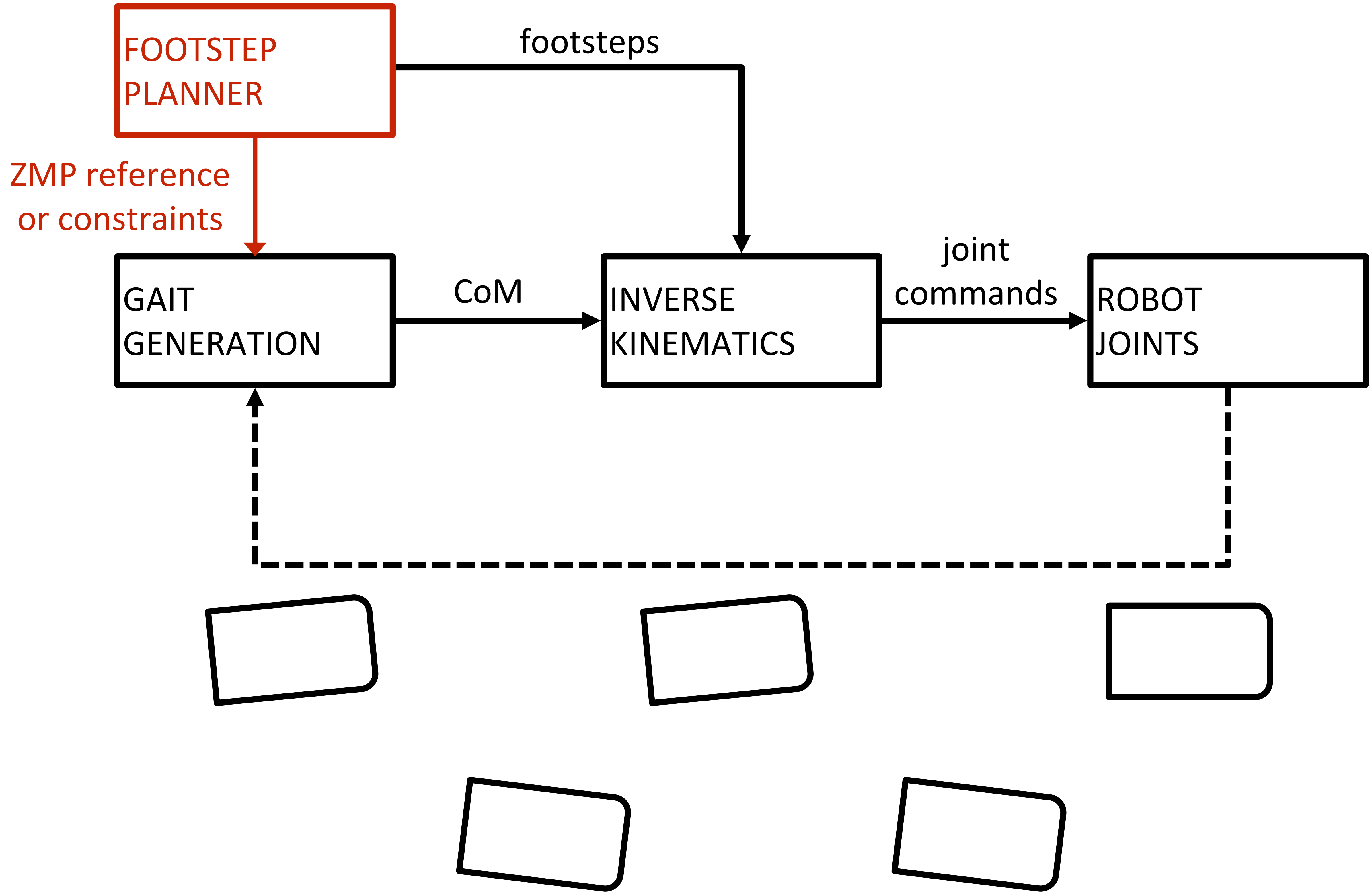
⇒ **recursive feasibility**

⇒ **stability**

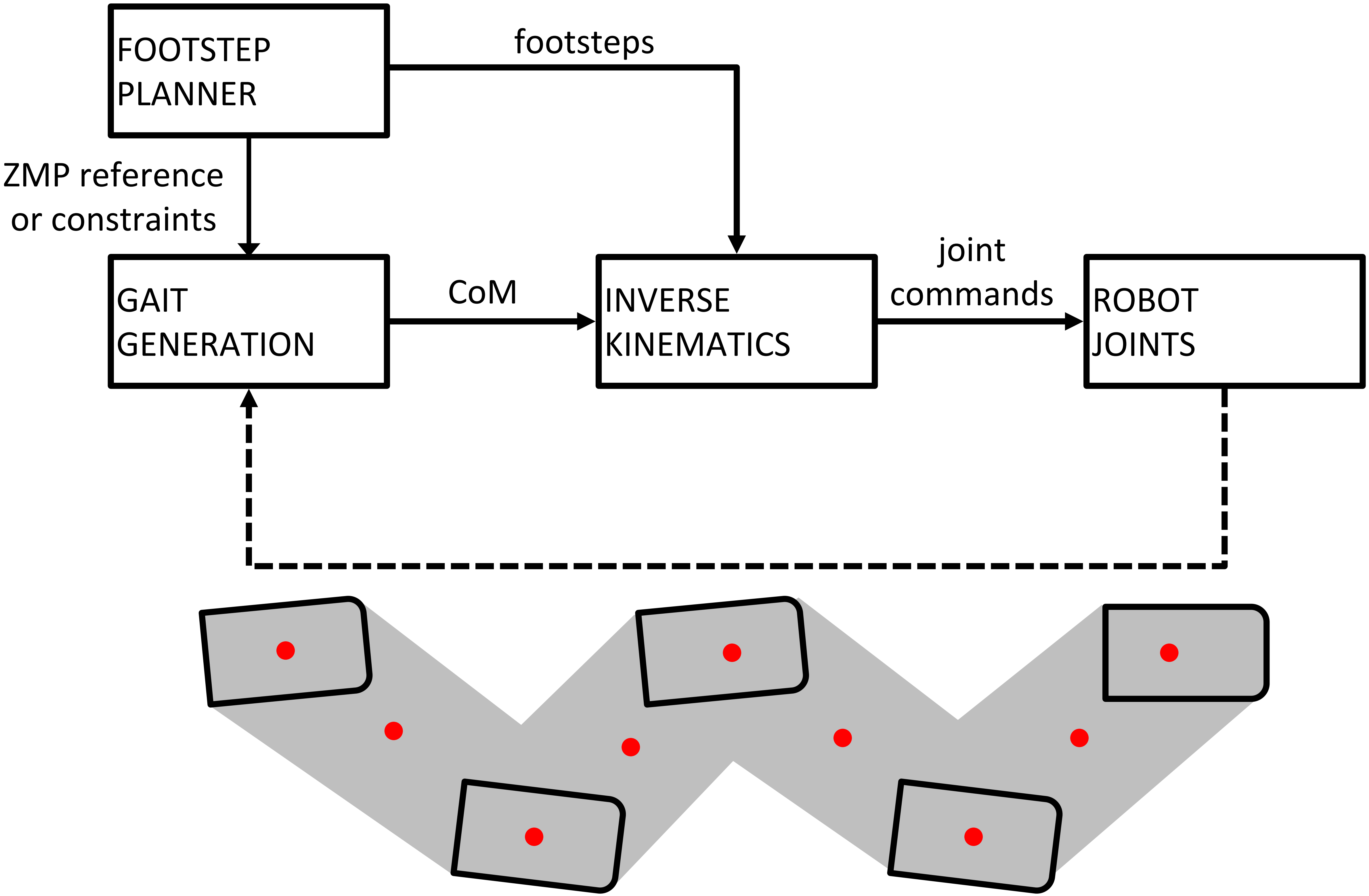
general controller architecture



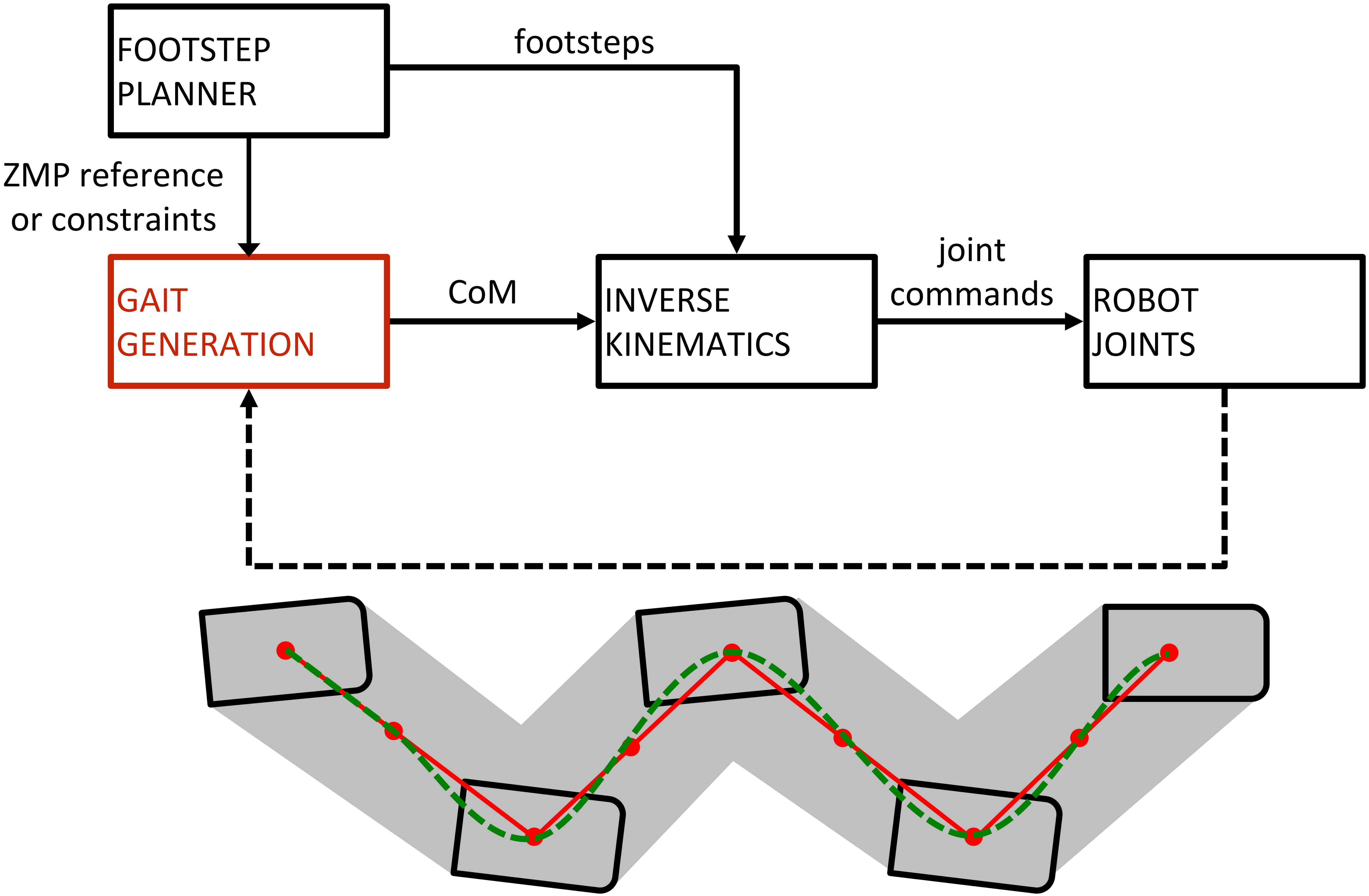
general controller architecture



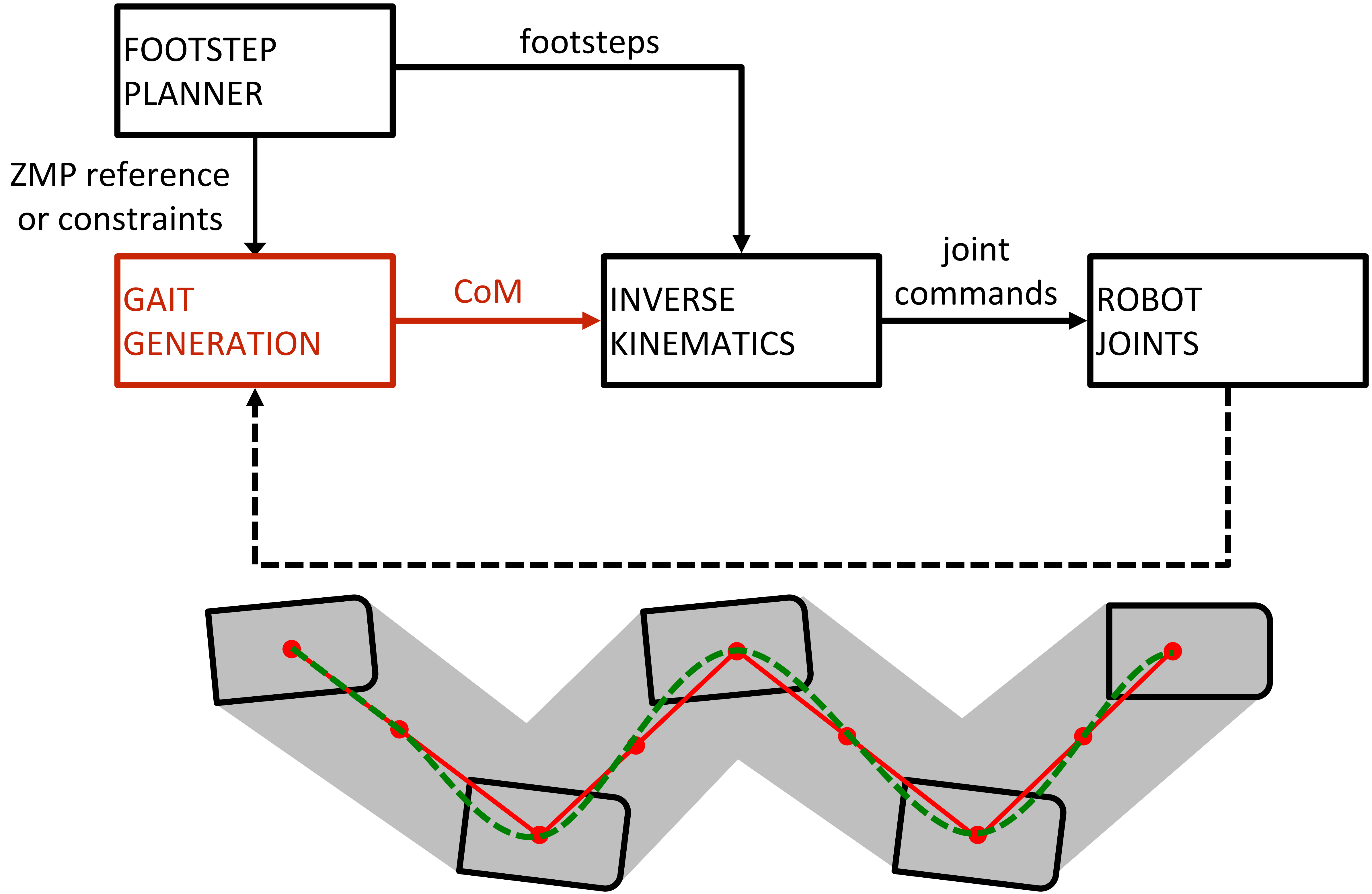
general controller architecture



general controller architecture



general controller architecture



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

$$\dot{\boldsymbol{c}} = J_c(\boldsymbol{q})\dot{\boldsymbol{q}}$$

$$\dot{\boldsymbol{\theta}} = J_\theta(\boldsymbol{q})\dot{\boldsymbol{q}}$$

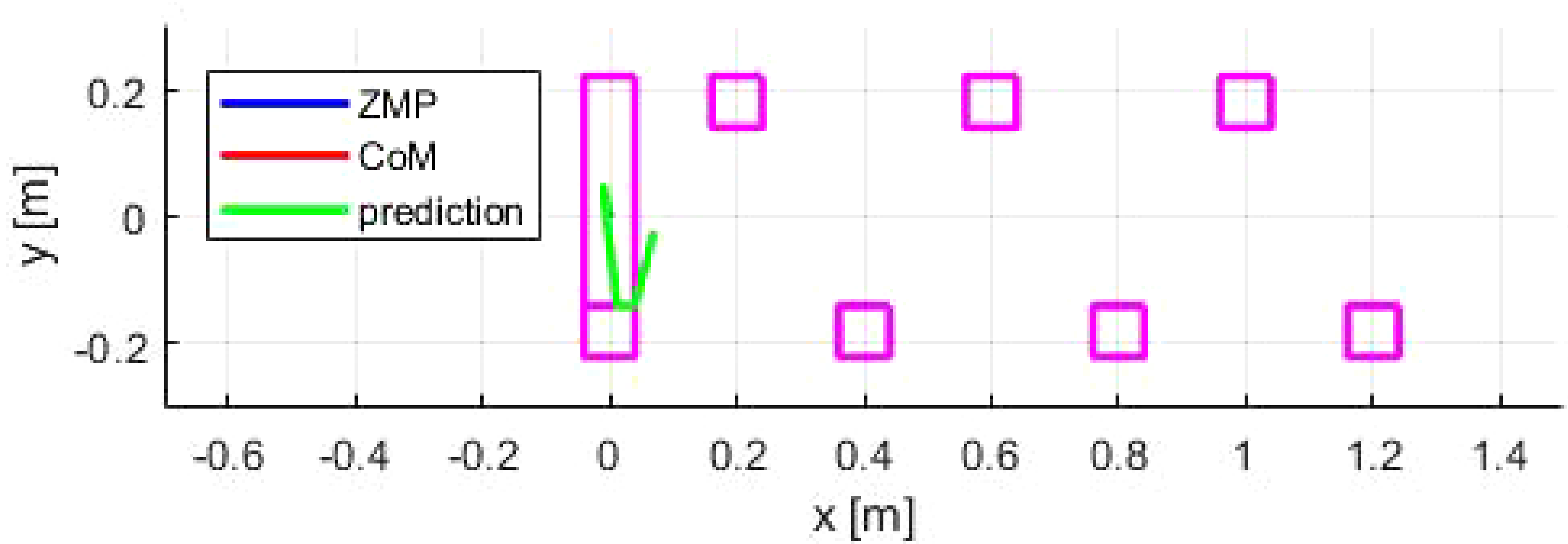
$$\dot{\boldsymbol{f}} = J_f(\boldsymbol{q})\dot{\boldsymbol{q}}$$

$$\dot{\boldsymbol{\phi}} = J_\phi(\boldsymbol{q})\dot{\boldsymbol{q}}$$

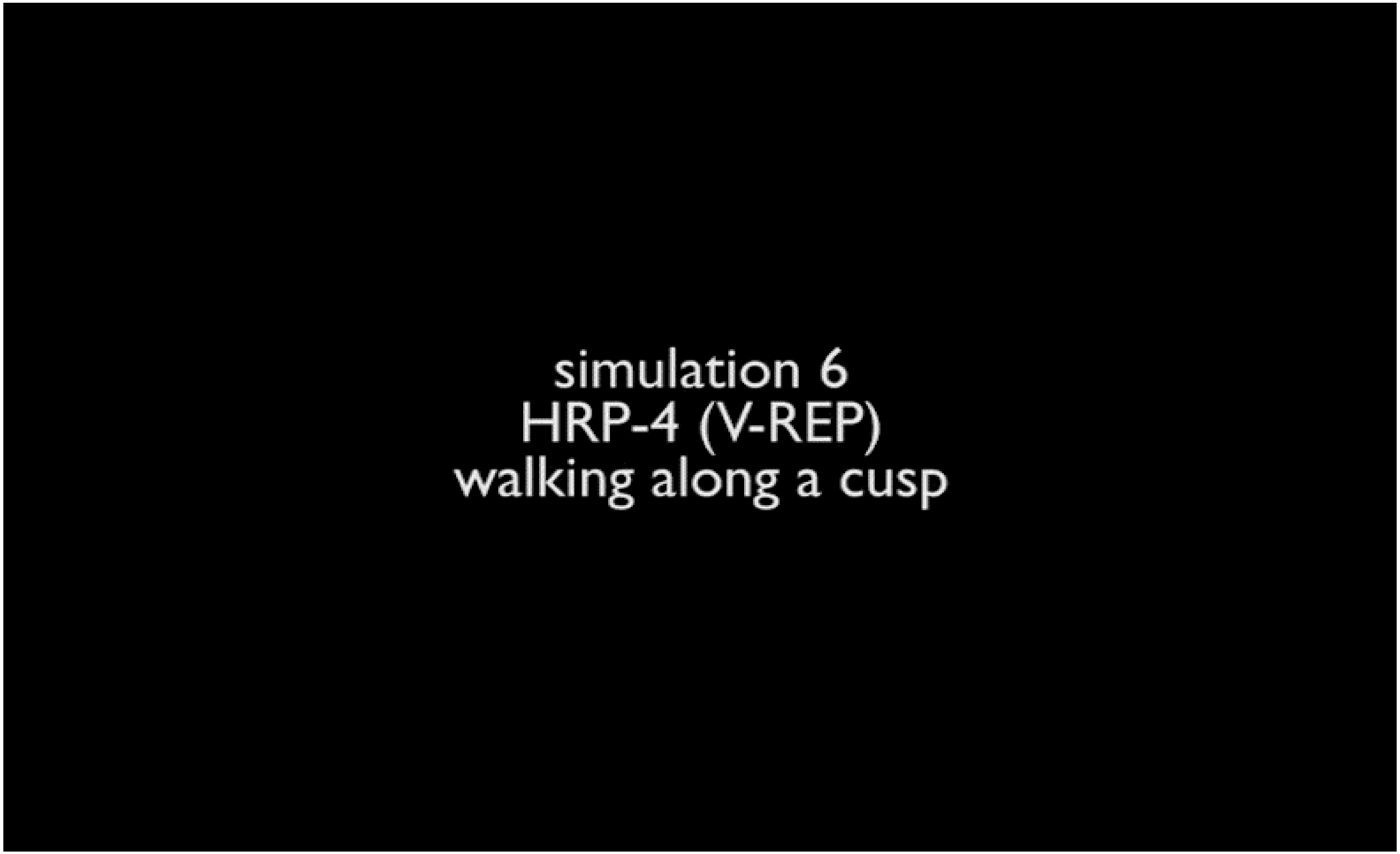
$$\Longrightarrow \quad \underbrace{\begin{pmatrix} \dot{\boldsymbol{c}} \\ \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{f}} \\ \dot{\boldsymbol{\phi}} \end{pmatrix}}_{\text{velocity task}} = \underbrace{\begin{pmatrix} J_c(\boldsymbol{q}) \\ J_\theta(\boldsymbol{q}) \\ J_f(\boldsymbol{q}) \\ J_\phi(\boldsymbol{q}) \end{pmatrix}}_{\text{stack of jacobians}} \dot{\boldsymbol{q}}$$

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

examples – MPC gait generation



examples – simulations and experiments



other relevant topics

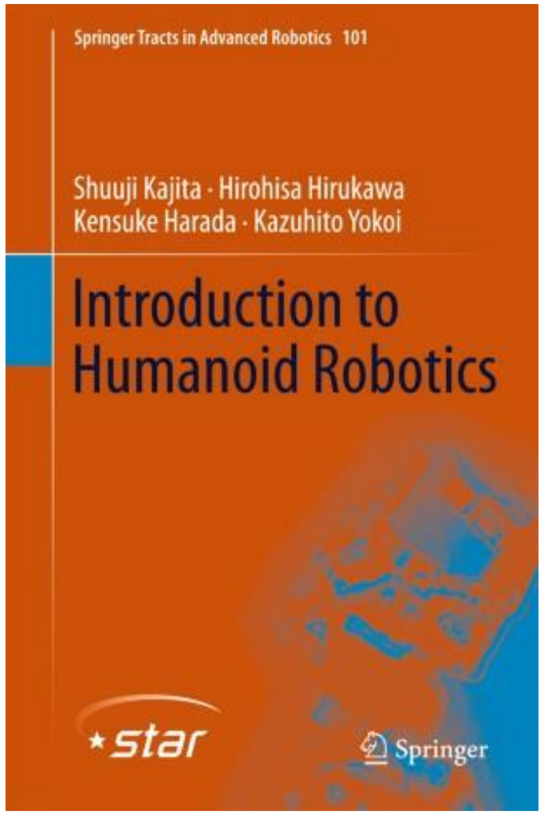
- **robust** gait generation (for disturbances)
- vertical CoM motion (for **uneven ground**)
- more accurate models (e.g., with **angular momentum**)
- **footstep planning** in complex environments

more examples – robust gait generation

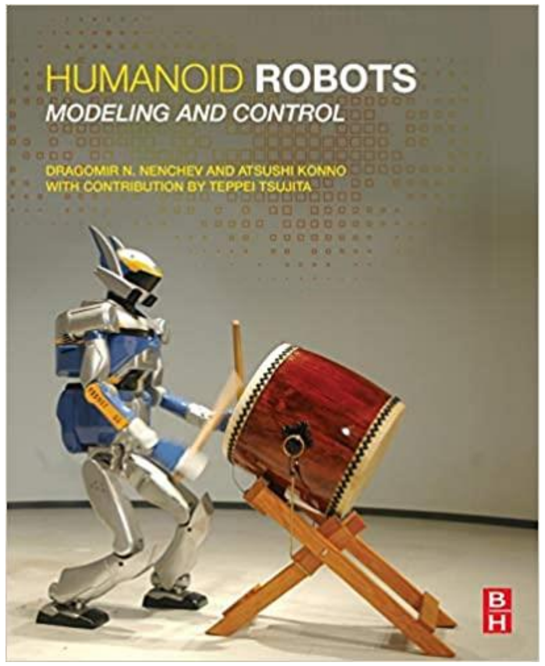
more examples – uneven ground

material – books

Kajita, Hirukawa, Harada, Yokoi
"Introduction to Humanoid Robots"
Springer



Nenchev, Konno, Tsujita
"Humanoid Robots: Modeling and Control"
Butterworth-Heinemann



material – books

material on model predictive control:
prof. Alberto Bemporad slides, from his course on MPC
http://cse.lab.imtlucca.it/~bemporad/mpc_course.html

Kajita, Kanehiro, Kaneko, Fujiwara, Harada, Yokoi, Hirukawa
“Biped walking pattern generation by using preview control of zero-moment point”
Int. Conf. on Robotics and Automation, 2003

Wieber
“Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations”
Int. Conf. on Humanoid Robots, 2006