RL-augmented MPC for Humanoid Robot Locomotion

Coletta Emanuele
Sartori Giulio
Sperduto Gianluca

Department of Computer, Control, and Management Engineering Antonio Ruberti

Sapienza
Università di Roma

Autonomous and Mobile Robotics
A.A. 2025/2026
Proff: Giuseppe Oriolo
Nicola Scianca

Table of contents

INTRODUCTION

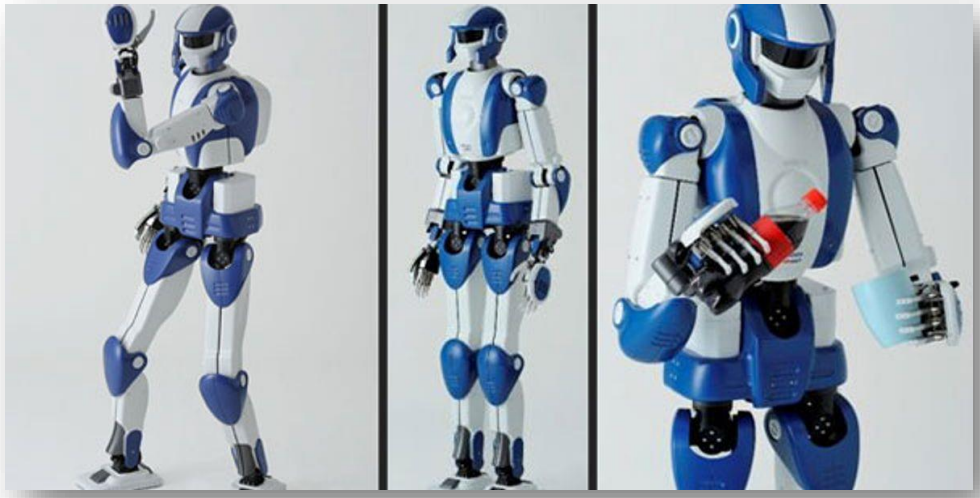
Introduction

Humanoid robots

- More and more ubiquitous
- Many DOFs, underactuated
- Models are complex, need simplifying assumptions

In this project:

- Walking is based on the LIP dynamics: control CoM-ZMP system, footsteps as constraints
- We add a RL-trained agent to modify footsteps to react to external disturbance forces and terrain inclination



Introduction

LIP Dynamics

- Linear model obtained assuming that
  - Center of mass is at constant height: $z_c = h$
  - Internal angular momentum is constant: $\dot{L}_c = 0$

- Uses ZMP to control CoM + dynamic extension

with $\eta = \sqrt{\frac{g}{h}}$

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

Introduction

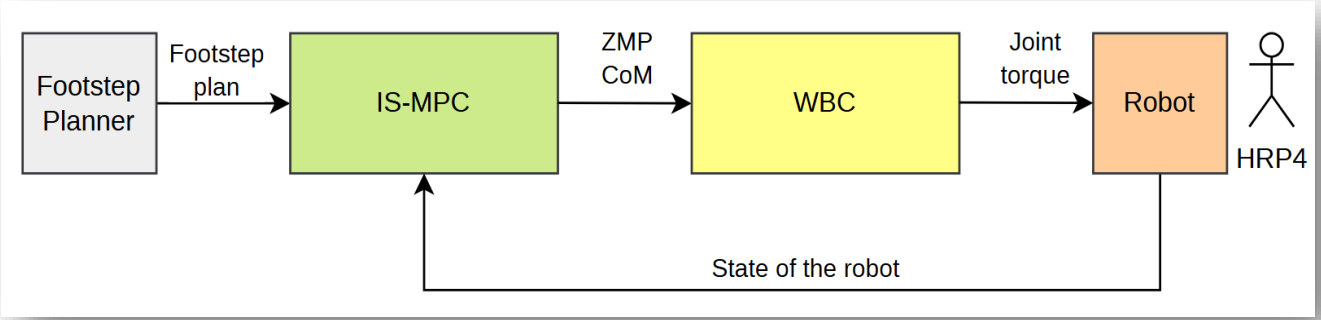
IS-MPC

- Extension to classical MPC that includes **stability contraints**
- Uses LIP model for prediction
  - Uses ZMP to control CoM
  - CoM position and velocity and ZMP position as state
  - ZMP velocity as control input (dynamic extension)

- Footsteps are used to generate constraints for support polygon
- Formulation includes constraints on **boundedness of CoM trajectory** w.r.t ZMP trajectory
  - If these constraints are satisfied , the dynamics is stable

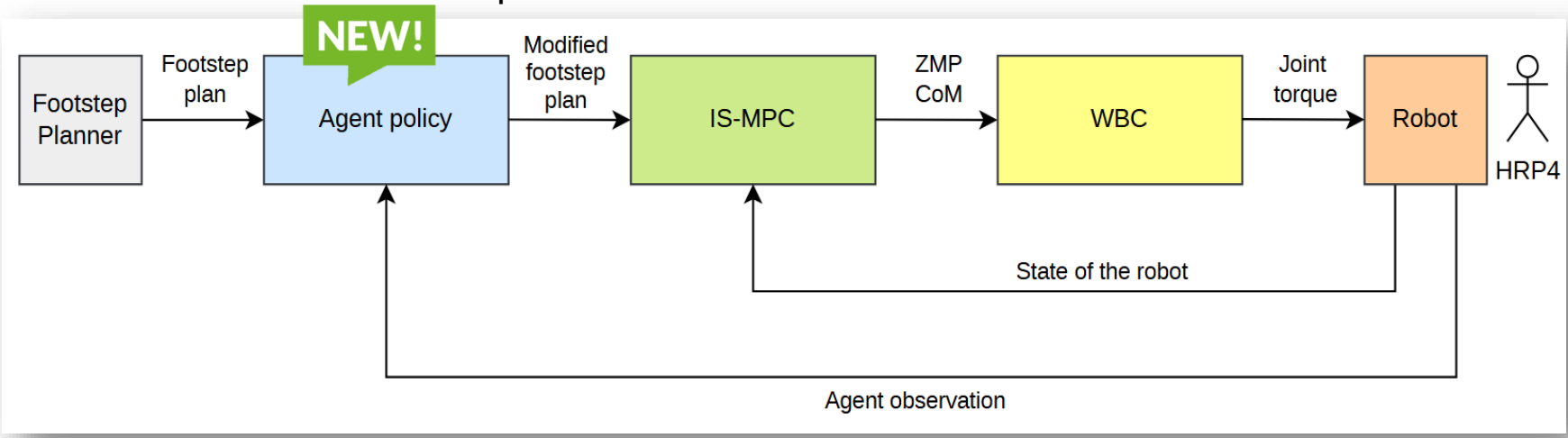
- Fixed timings for single and double support phases
  - Reference paper considers them variable

Introduction



Reinforcement Learning approach

- IS-MPC prediction model is not accurate in the presence of high disturbance forces and terrain inclination
- Augment the scheme with a RL-trained agent which displaces the footsteps to make the robot keep balance



RL approach

PPO – On Policy Algorithms

1) **Act** according to the current policy while collecting data (Roll-out)
2) **Process** data to compute the **Advantage** (used in loss function)
3) **Update** the policy network w.r.t. PPO **Loss** to maximize the **Expected Return**
4) Repeat

- ❑ When enough data is collected the simulations are paused and network is updated
- ❑ **Advantage** is used to compute part of the PPO Surrogate Loss
- ❑ Then Roll-outs are restarted and simulations are resumed
- ❑ This process is repeated for a large number of episodes
- ❑ Our implementation runs **16 parallelized environments** each Roll-out

Actor-Critic algorithm

The **advantage** function express how good it is to take a certain action *a* in state *s* w.r.t. the other actions

It is defined as:

Estimated by a NN

$$A(s,a) = Q(s,a) - V(s)$$

Expected return from taking action *a* in state *s*

Mean expected return of the policy in state *s*

- It is used to compute an **estimator of the policy gradient**
- **Gradient ascent algorithm** to update the policy network (**actor** component)
- **V(s)** estimated by the **critic** **component** of the neural policy
- **V(s)** estimation updated according to a squared-error loss w.r.t. a **V(s)** target

RL Curriculum Learning

Curriculum Learning

- Start by training the agent on simple environments
- **Gradually increase the difficulty** of the environment in **levels (**not included in reward function)
- Difficulty corresponds to range of possible terrain inclinations
  - Range [3.4°, 6.8°] * level / 100

- An agent starts from 0 points.
  - ✓ It is awarded 2 points for successfully completing a foostep plan with no failure.
  - ✓ 1 point subtracted on any failure
- An agent **levels up** when it accumulates a total of 6 points

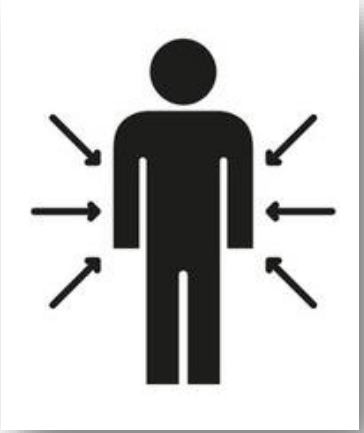
Disturbances

External forces

- Random external forces between 30 and 87N are applied to random bodies of the robot (torso, body, feet)

Ground Slope

- The ground has a slope (both in x and y direction) that is not modelled in MPC



GENERAL FRAMEWORK

Observations

Observations

$$\mathcal{S} = \begin{bmatrix} \sigma_{\mathbf{k}} & \Delta t_{r,k} & \mathbf{p_{c,k}} & \dot{\mathbf{p}}_{\mathbf{c,k}} & \mathbf{p_{z,k}} & \dot{\mathbf{p}}^{\mathbf{r}}_{\mathbf{c,k}} & \mathbf{p}^{\mathbf{r}}_{\mathbf{z,k}} & \mathbf{L} & \boldsymbol{\Delta}\sigma_{\mathbf{k+1}} & \boldsymbol{\Delta}\sigma^{\mathbf{r}}_{\mathbf{k+1}} \end{bmatrix}^T \in \mathbb{R}^{25} \times \mathbb{SO}^2(2)$$

- $\sigma_{\mathbf{k}}$ Support foot encoding
- $\Delta t_{r,k}$ Remaining time in swing
- $\mathbf{p_{c,k}}$ $\dot{\mathbf{p}}_{\mathbf{c,k}}$ CoM position and velocity
- $\mathbf{p_{z,k}}$ ZMP position
- $\dot{\mathbf{p}}^{\mathbf{r}}_{\mathbf{c,k}}$ CoM reference velocity
- $\mathbf{p}^{\mathbf{r}}_{\mathbf{z,k}}$ ZMP reference position
- $\mathbf{L}$ Angular momentum
- $\boldsymbol{\Delta}\sigma_{\mathbf{k+1}}$ Next footstep position
- $\boldsymbol{\Delta}\sigma^{\mathbf{r}}_{\mathbf{k+1}}$ Next footstep reference position

All components of the state are relative to the current support foot

Displacements given in this form

$$\Delta\sigma_{k+1} = \begin{bmatrix} \Delta x & \Delta y & \Delta\varphi \end{bmatrix}^T \in \mathbb{R}^2 \times \mathbb{SO}(2)$$

Rewards

Reward function given to the Agent

$$R(s_k, a_k, s_{k+1}) = r_{\text{state}} + r_{\text{phase}} + r_{\text{cp}} + r_{\text{CoM}} + r_{\text{fs}}$$

$$r_{\text{state}} = \begin{cases} \text{alive bonus} & \text{if robot alive} \\ \text{penalty} & \text{if IS-MPC solver solution innacurate} \\ \text{penalty} \cdot 0.5 & \text{if IS-MPC solver max iters.} \\ \text{penalty} \cdot 3 & \text{if Inverse Dynamics not feasible} \\ \text{penalty} \cdot 5 & \text{if feet collide} \end{cases}$$

$$r_{\text{phase}} = \begin{cases} r_{\text{swing}} & \text{if is in swing phase} \\ r_{\text{ds}} & \text{if is in double support} \end{cases}$$

$$r_{\text{swing}} = -w_h(z_c - h)^2 + w_a \frac{a^T a}{\Delta t_r + 10^{-3}}$$

$$r_{ds} = -w_a a^T a$$

$$r_{\text{cp}} = \begin{cases} \text{checkpoint bonus} & \text{if robot reaches multiple of 3 footsteps} \\ \text{checkpoint + end bonus} & \text{if robot reaches the end of the plan} \\ 0 & \text{otherwise} \end{cases}$$

$$r_{fs} = r_{fs_1} + r_{fs_2}$$
$$r_{fs_1} = -\text{Ker}_f(\Delta f^2)$$
$$r_{fs_2} = \text{Ker}_f(|\Delta f^r|^2)$$

$$r_{CoM} = \text{Ker}_{\dot{x}_c}(e_{\dot{x}_c, \dot{x}_c^r}) + \text{Ker}_{\dot{y}_c}(e_{\dot{y}_c, \dot{y}_c^r})$$

$$\text{Ker}_v(e) = w_v \exp(-(e/\sigma_v)^2)$$

Different Approaches

Different approaches to footstep displacement

- Approach 1: displace only the next footstep.
- Approach 2: displace the whole plan.
- Approach 3: fully displace the next footstep, then geometrically scale displacement of successive footsteps.

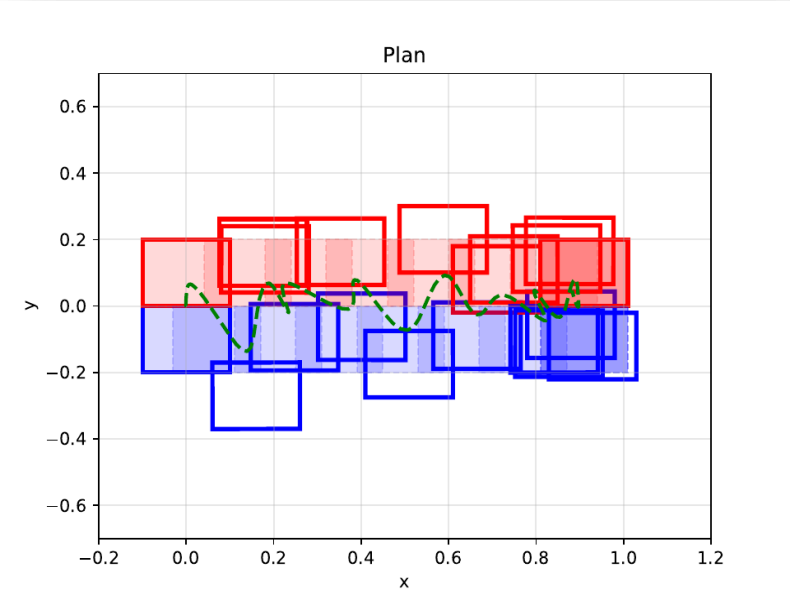
- Saturations are considered on the maximum displacement of the next footstep to ensure it stays feasible after displacement (for all agents).

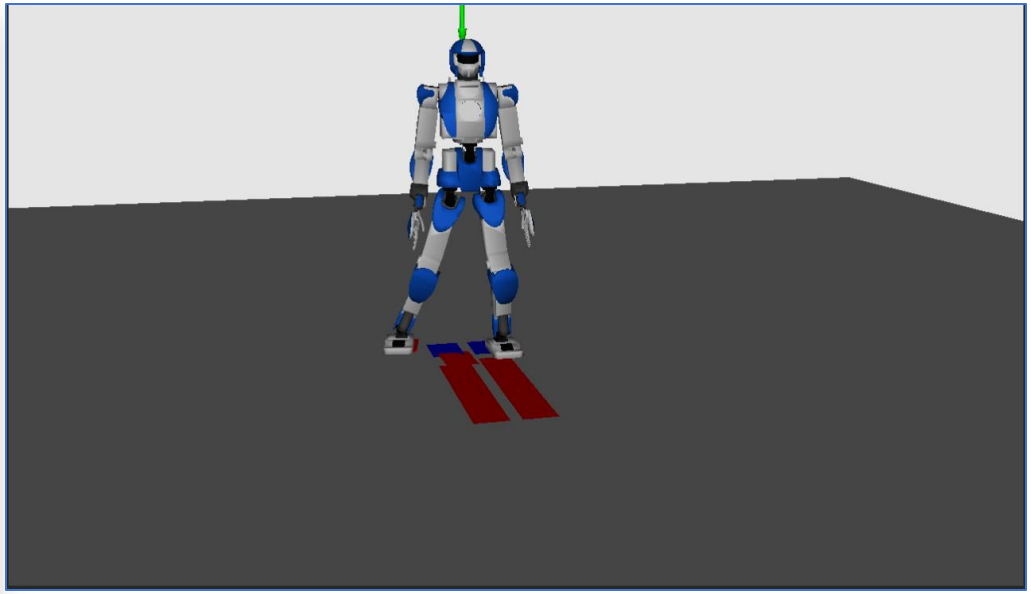
- Since in PPO actions are sampled from a Gaussian policy, in general we expect to see some displacement even when no disturbance is present, even on a very well trained agent.

Footstep displacement

Approach 1

- Only the next foostep is displaced
- Little displacement, easily track the nominal footstep plan
- Displacement might make the plan unfeasible, especially when caused by external forces or terrain inclination
  - Feasibility of current displacement depends on future (unknown) actions!

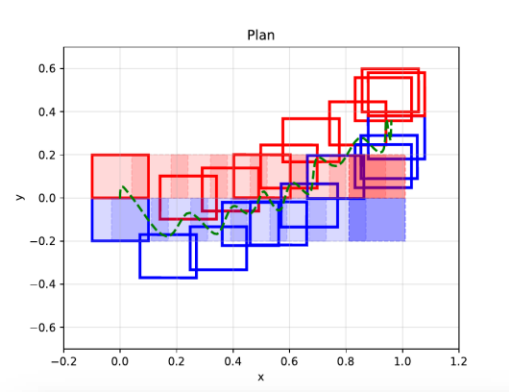


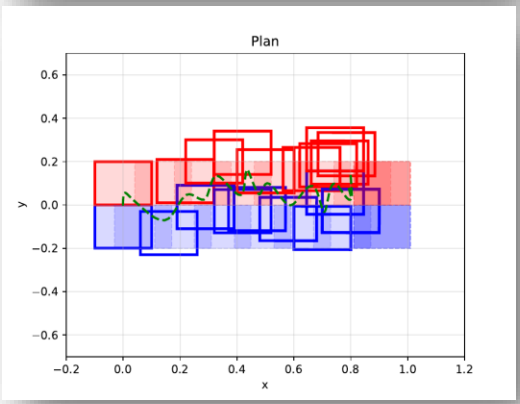


Footstep displacement

Approach 2

- The whole plan is displaced
- Displaced plan is feasible if both the original plan and the current displacement are
- Displacements accumulate, the robot easily strays from the nominal plan
  - Can add appropriate reward to encourage better tracking, but this doesn't always work



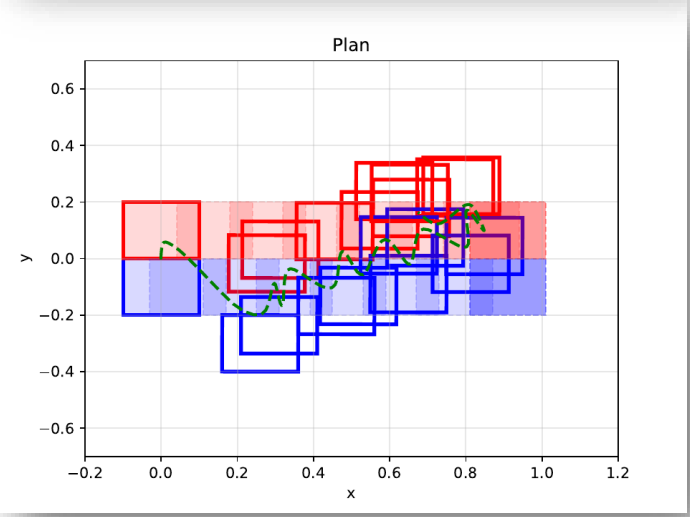


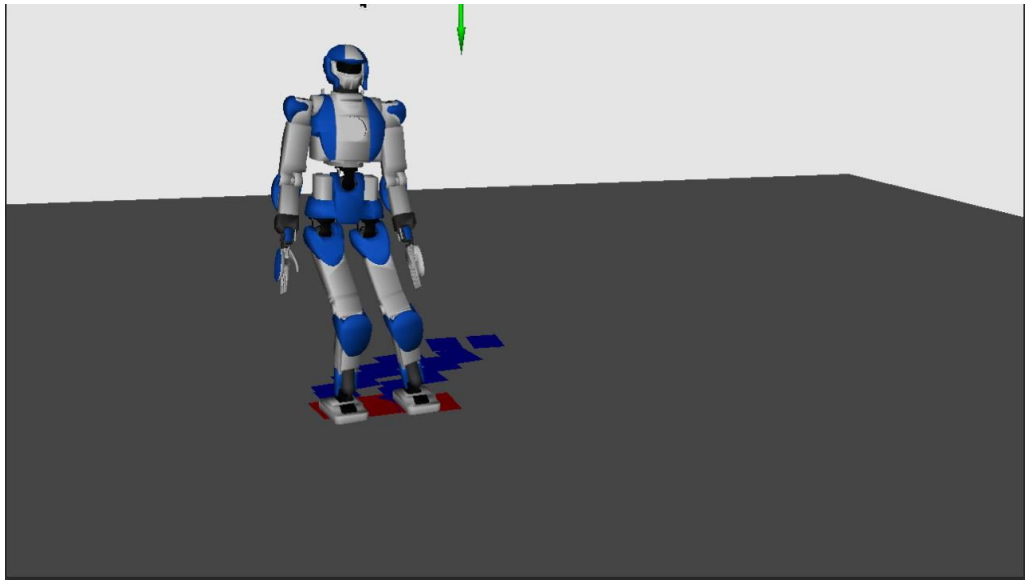
Footstep displacement

Approach 3

- Tradeoff between approaches 1 and 2
- The first footstep is completely displaced, for the successive ones the displacement is geometrically scaled by a fixed factor (hyperparameter)
- Better tracking that approach 2, displacements are feasible with higher probability than approach 1.

$$FS'_{t+i} = FS_{t+i} + \xi^{i-1}\Delta_t, \; i = 1, 2, ..., \Sigma$$





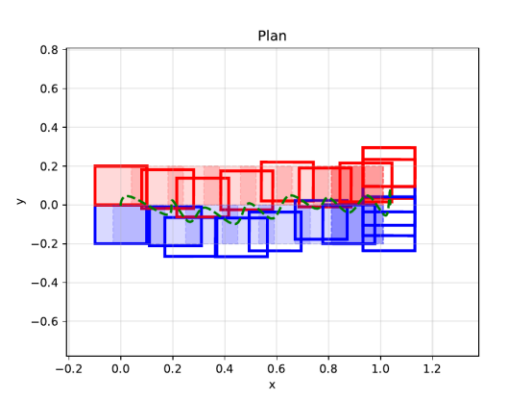
Footstep displacement

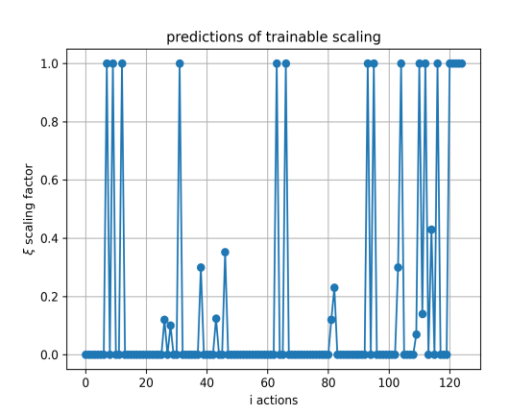
Approach 3- Extension

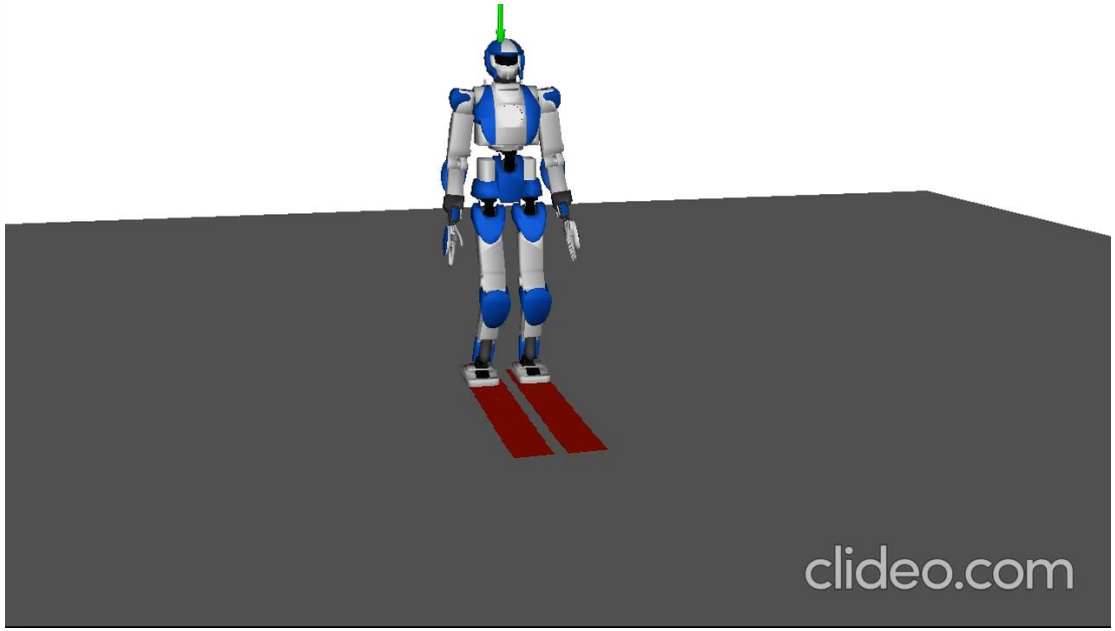
Approach 3 with the scaling parameter that is **learned.**

$$A_{int} = \{\Delta_x, \Delta_y, \Delta_\theta\}$$

$$A_p = \{\xi\}$$

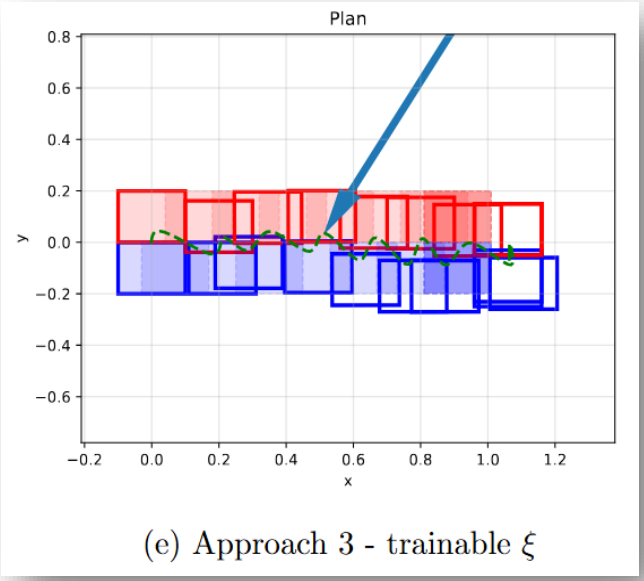






Nominal Conditions: displace only 1st footstep

Foostep displacement

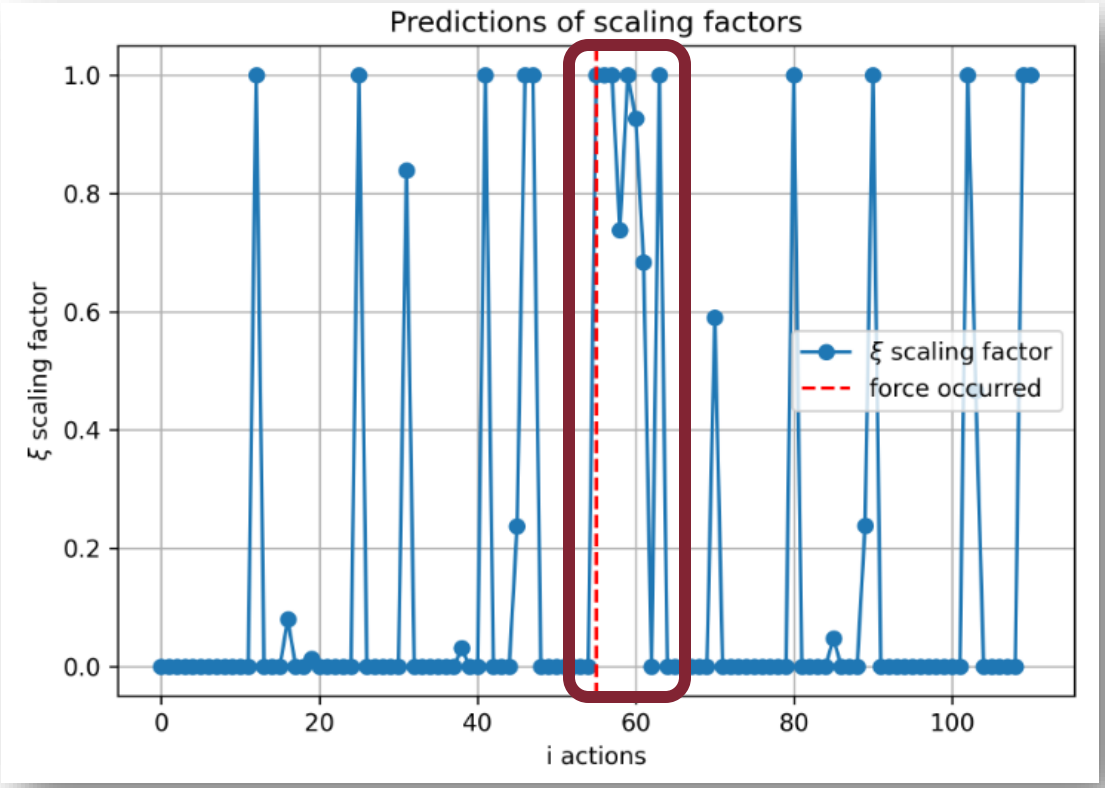


(e) Approach 3 - trainable ξ

More uncertain behavior if **ground slope** is introduced

Only external forces

Learned to modify all the plan if it is needed to recover **dynamical balance** after **force disturbances**



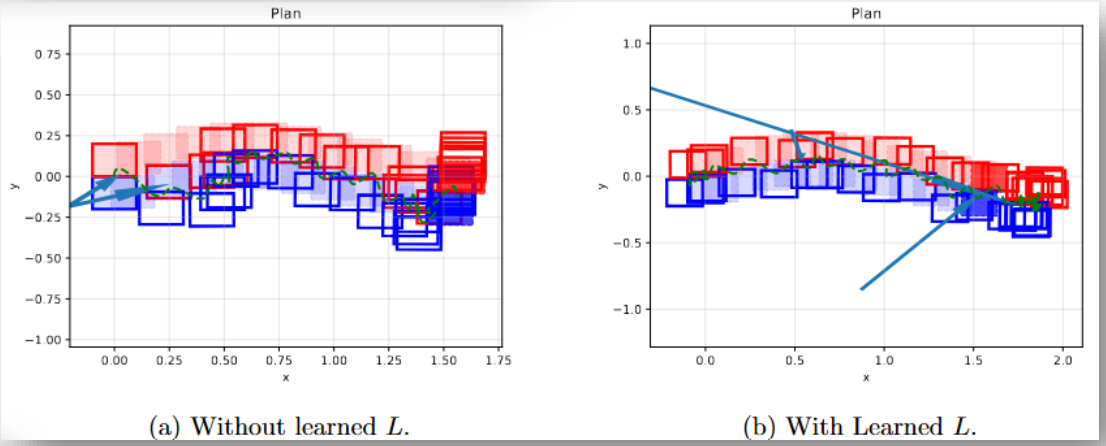
Learning Angular Momentum

Idea: Learning to reproduce a Desired Angular Momentum about the contact point

We get an Angular Momentum reference performing a simulation in **nominal conditions** with **only IS-MPC**

$$L = [L_x, L_y] \qquad \mathbf{L_{des}} = [L_{xd}, L_{yd}] \qquad \mathbf{e_L} = \mathbf{L_{des}} - \mathbf{L}$$

augmented agent observation space



(a) Without learned L. (b) With Learned L.

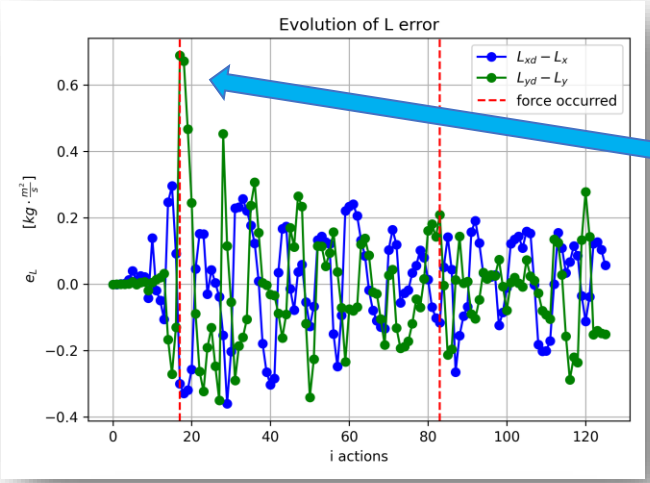
Promoting tracking with the **reward term:**

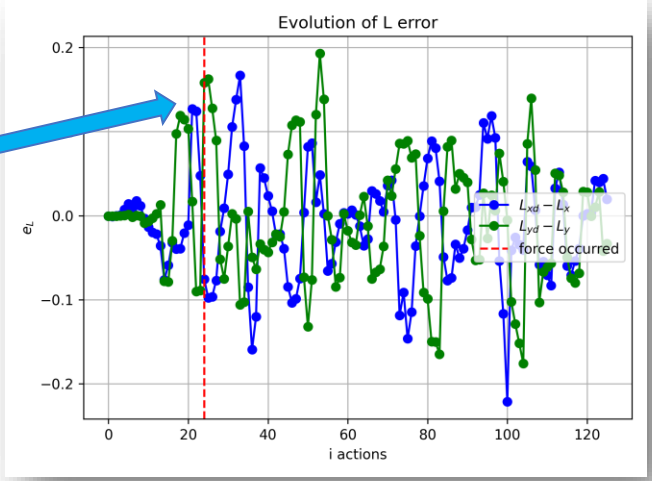
$$r_{L_{des}} = Ker_L(|\mathbf{e_L}|)$$

better **reference reproduction** capabilites

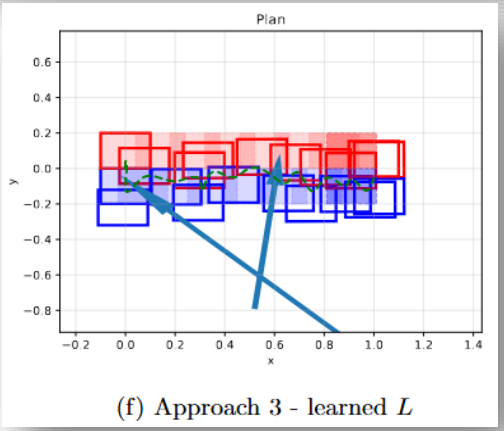
Learning Angular Momentum

Idea: Learning to reproduce a Desired Angular Momentum about the contact point



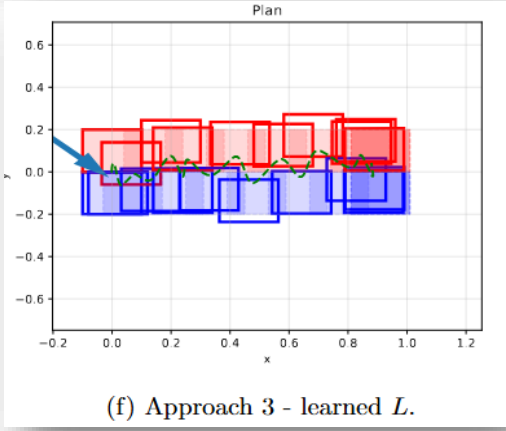


Spikes in the error as an **external force** occurs



(f) Approach 3 - learned L

not so effective in presence of **ground slope**



(f) Approach 3 - learned L.

SIMULATIONS

Simulations

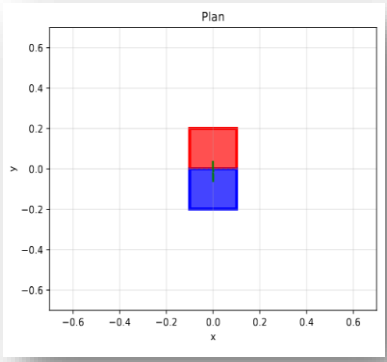
Setup

- **Sampling time** 0.01s
- IS-MPC **prediction horizon** to 150 samples (1.5s)
- Policy is a **MLP**, two hidden layers of 64 nodes each and *tanh* activations
- Saturation on total displacement of the next foostep is set to 0.2m
- Only approach 3 is shown for brevity, scaling factor set to 0.9
  - The report includes each simulation with all three approaches
- Random **forces between [30, 87] [N]** in magnitude, applied with 3% chance each step to a random part of the robot between torso, body and feet
- **Feet collision** is considered a failure, same as crashing MPC or Whole Body Controller solvers

Simulations

Sim 2 – **Walking on the spot**, with forces, no terrain inclination

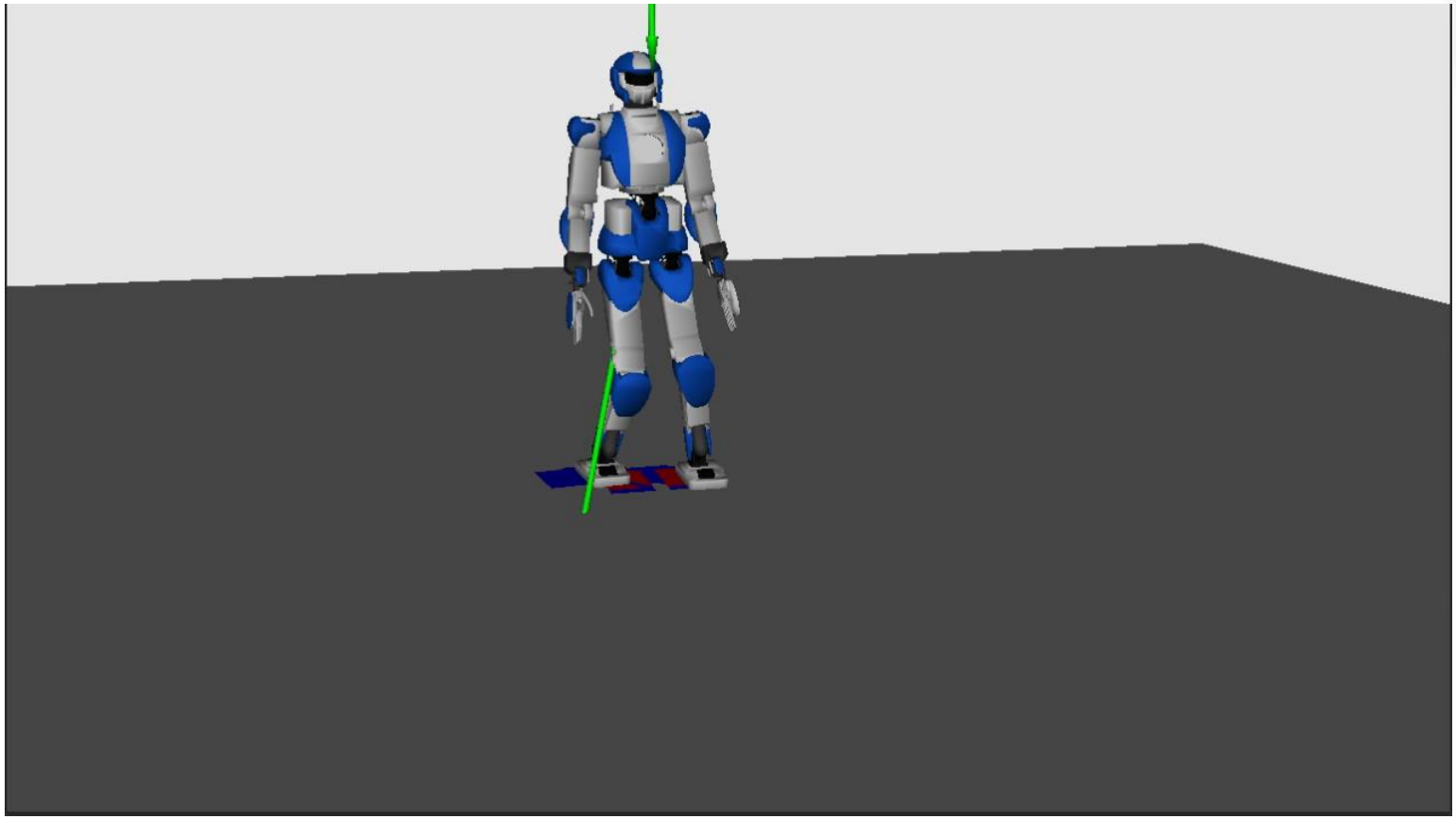
- Nominal case (IS-MPC only)

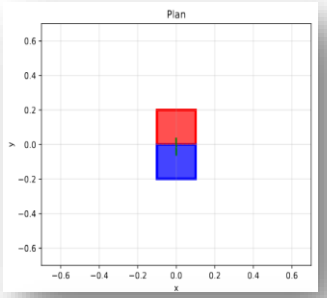


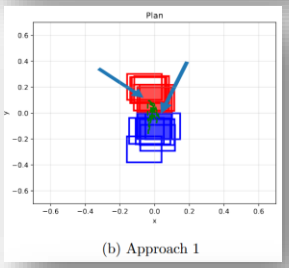
Simulations

Sim 2 – **Walking on the spot**, with forces, no terrain inclination

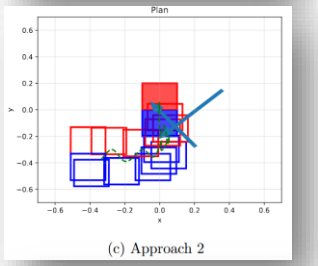
- Approach 3



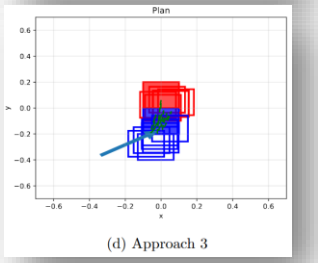




(b) Approach 1



(c) Approach 2

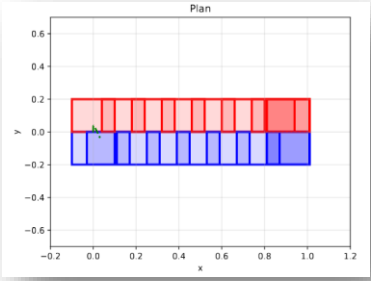


(d) Approach 3

Simulations

Sim 3 – **Straight line trajectory**, with forces and inclination (x: 3° , y: 1°)

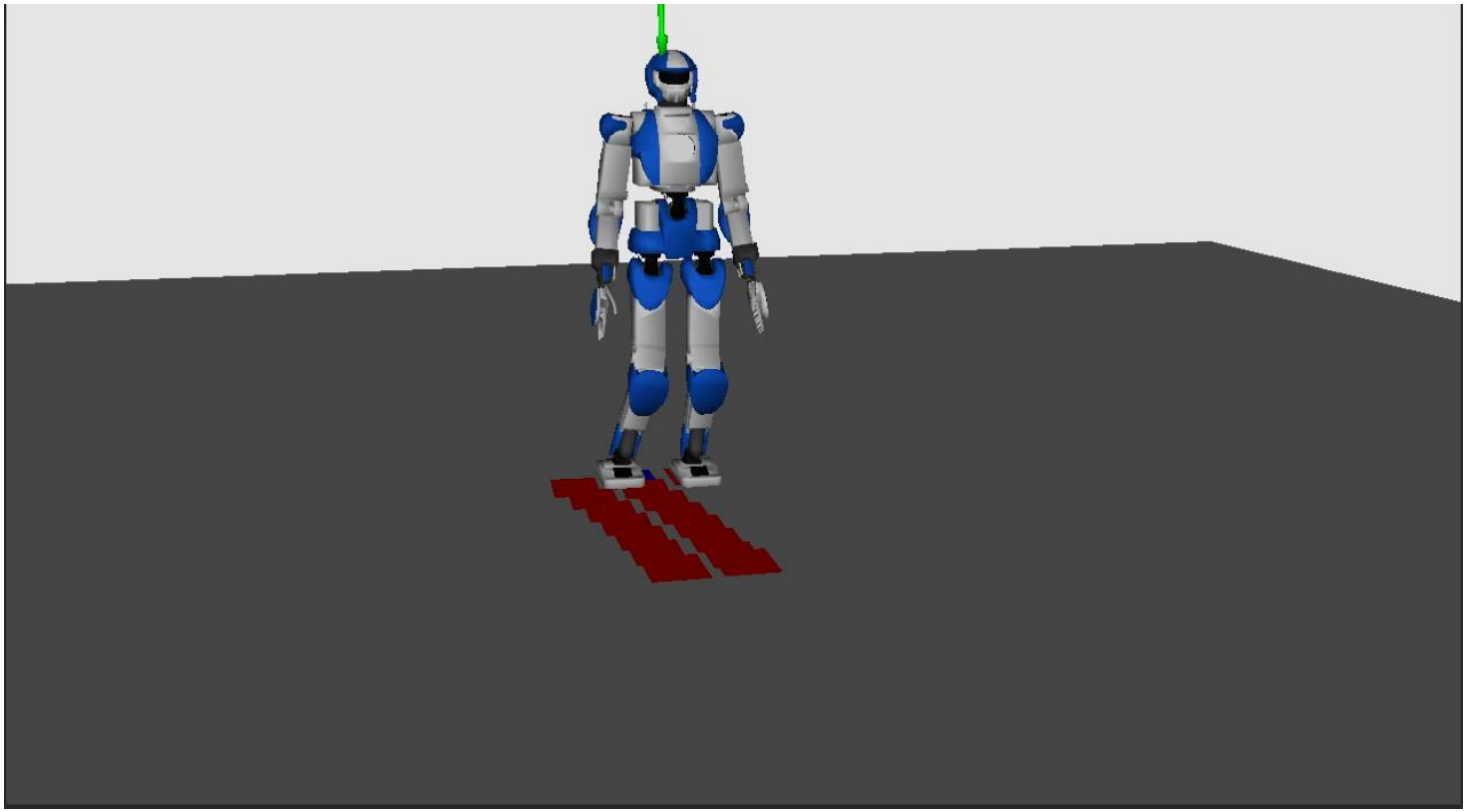
- Nominal (IS-MPC only)

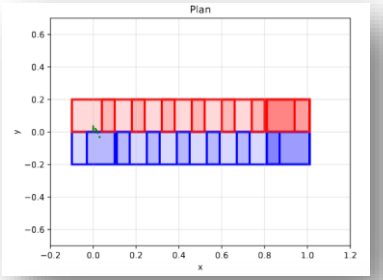


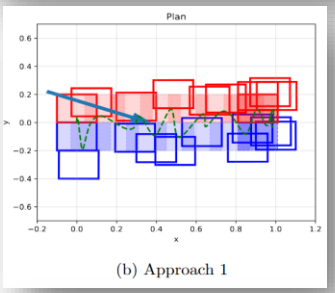
Simulations

Sim 3 – **Straight line trajectory**, with forces and inclination (x: 3° , y: 1°

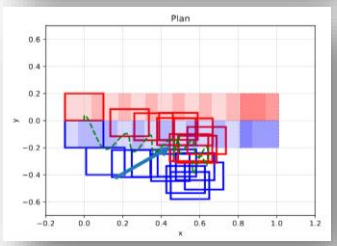
- Approach 3

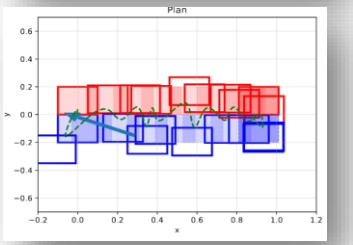






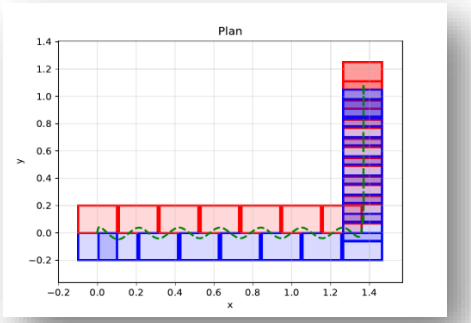
(b) Approach 1



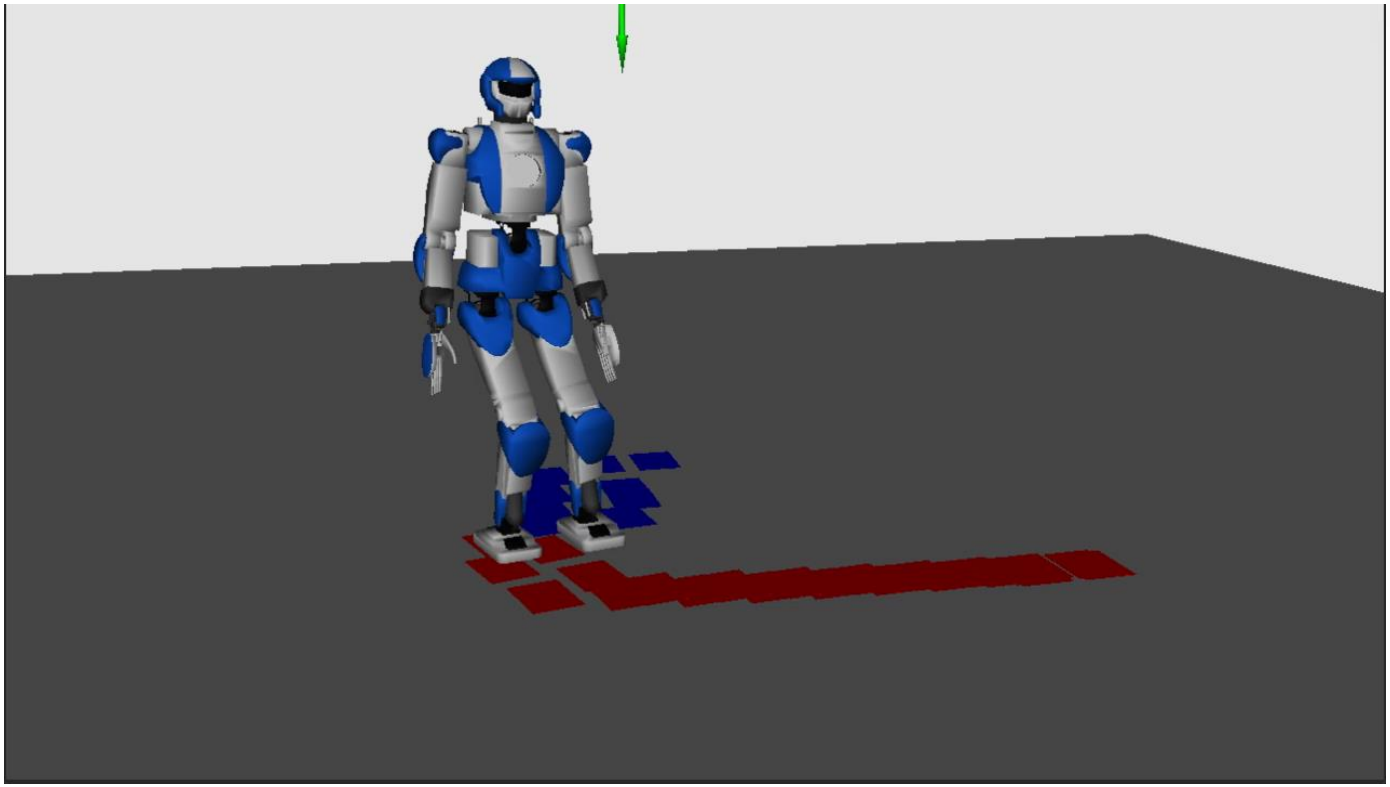


Simulations

Sim 4 – **Validation trajectory**, with forces, no inclination



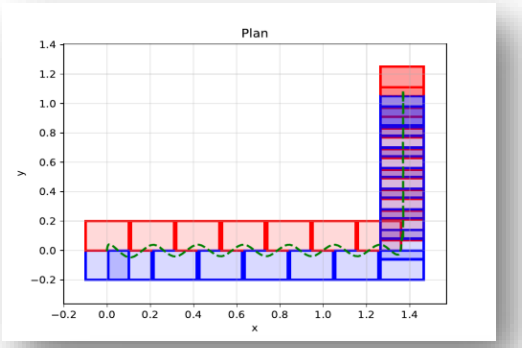
- Approach 3
- Most common modes of failure are feet collision or solver failure when changing direction: the agent is **not generalizing** well

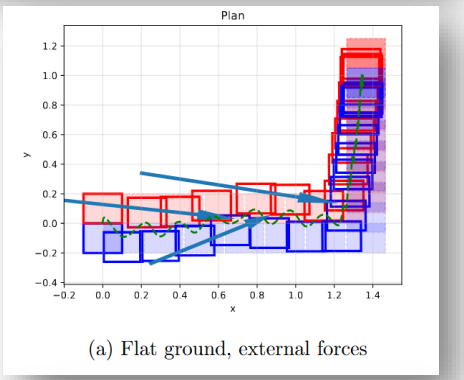


Simulations

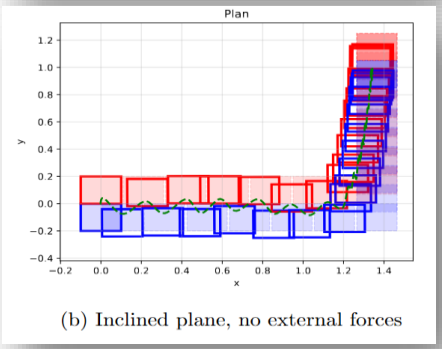
Sim 5 – **Validation trajectory**, with forces, inclination (x: 3° , y: 1°) and saturated actions

- Approach 3 – Trainable scaling factor and learned L
- Actions are saturated to prevent feet collision and for max. displacement, model trained from scratch





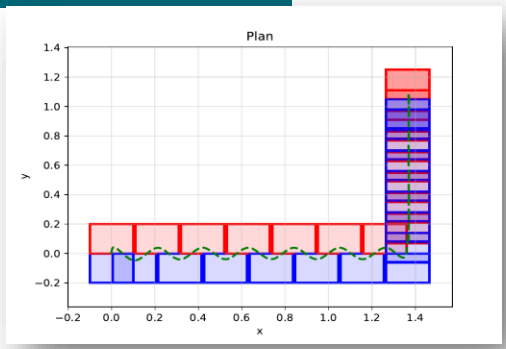
(a) Flat ground, external forces



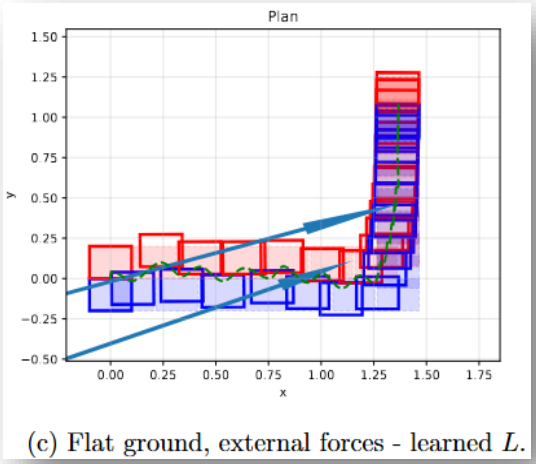
(b) Inclined plane, no external forces

Simulations

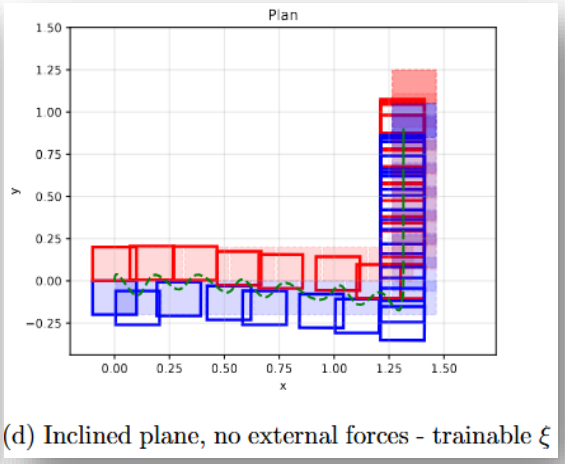
Sim 5 – **Validation trajectory**, with forces, inclination (x: 4° , y: 1°) and saturated actions



- Approach 3, trainable scaling factor and learned L

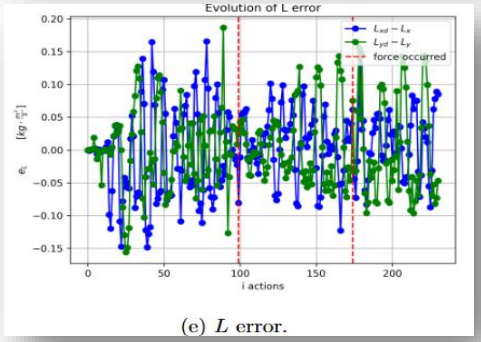


(c) Flat ground, external forces - learned L.

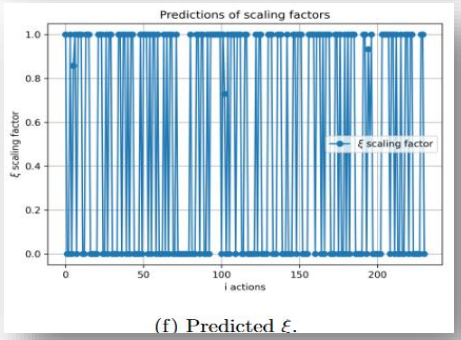


(d) Inclined plane, no external forces - trainable ξ

actions saturation allows to walk sideways



(e) L error.



(f) Predicted ξ.

more involved scaling factor predictions if **ground** is **tilted**

When To Use The Agent?

Use the agent **only** when is needed

- The agent will always modify the plan
  - But if well trained, the displacements will be close to zero when in near-nominal conditions
- It is interesting to find a way to "turn-off" and "turn-on" the agent only when a force is applied or the terrain is inclined.
- How to detect if there are perturbations in the environment?

Energy Observer

Hybrid Gaussian-Bernoullian decision policy

Energy Observer Derivation

Energy observer for **perturbation detection**

- The energy of the LIP dynamics can be monitored with a **residual**
- If the residual is above a threshold, something is perturbing the model and the Agent must take actions

$$E(t) = K(t) = \frac{1}{2\eta^2}\dot{p}^T\dot{p}$$

$$\dot{E}(t) = \frac{1}{\eta^2}\dot{p}_c^T\ddot{p}_c = \dot{p}_c^T(p_c - p_z) - \frac{g}{\eta^2}y_c$$

$$\ddot{p}_c = \begin{bmatrix} \ddot{x}_c \\ \ddot{y}_x \\ \ddot{z}_c \end{bmatrix} = \begin{bmatrix} \eta^2(x_c - x_z) \\ \eta^2(y_c - y_z) \\ \eta^2(z_c - z_z) - g \end{bmatrix}$$

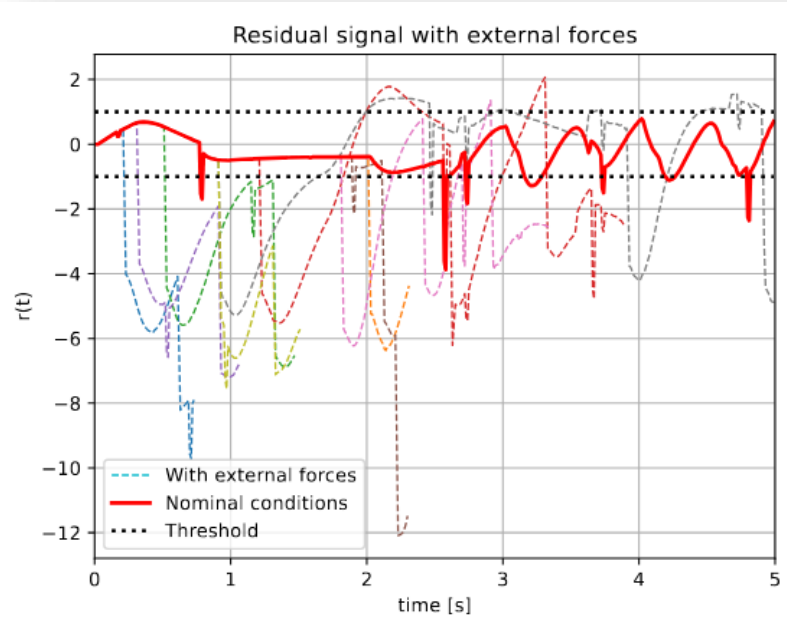
$$\dot{r}(t) = G\left(\dot{E}(t) - \dot{p}_c^T(p_c - p_z) + \frac{g}{\eta^2}y_c - r(t)\right) = G\Big(n(t) - r(t)\Big)$$

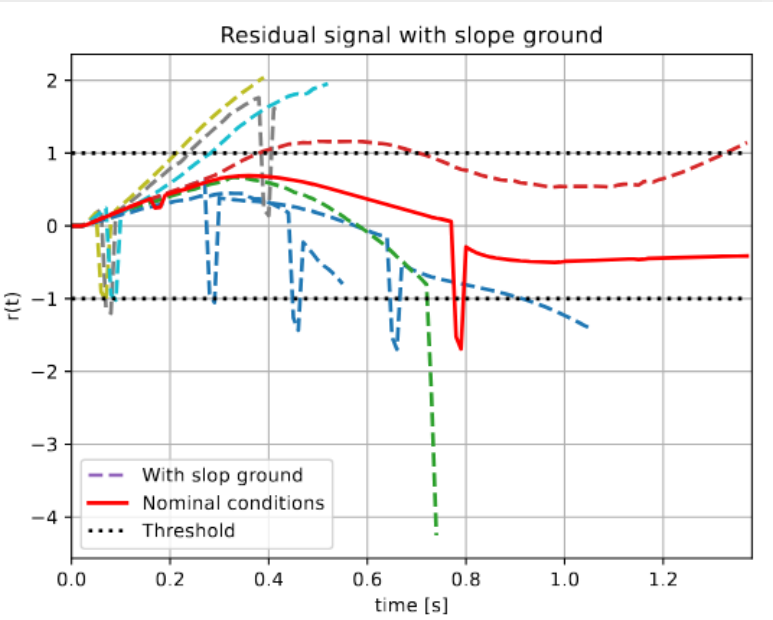
$$\mathcal{L}\{\dot{r}(t)\} = sr(s) = G(n(s) - r(s)) \Rightarrow r(s) = \frac{G}{G+s}n(s)$$

Energy Observer

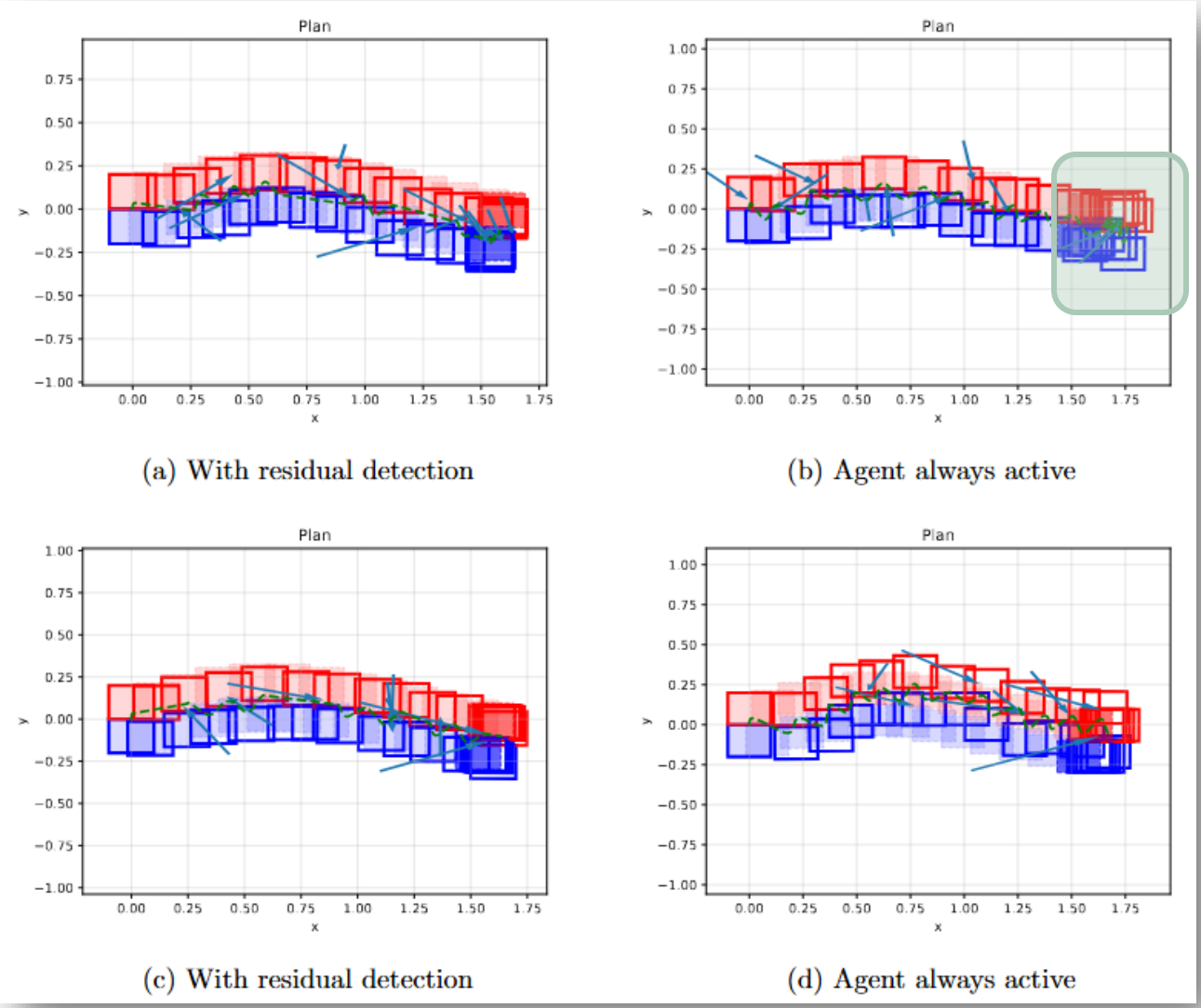
Energy observer for **perturbation detection**

$$r(t) = G\left(E(t) - \int_0^t \left(\dot{p}_c^T (p_c - p_z) - \frac{g}{\eta^2} y_c + r \right) d\tau - E(0) \right)$$





Energy Observer Behaviour



Final displacement

General displacement from original plan

Hybrid Policy

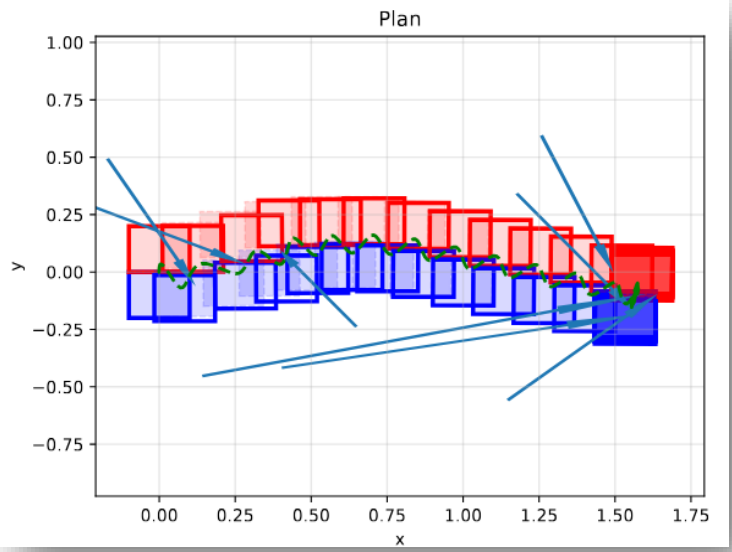
Hybrid Gaussian-Bernoullian policy to decide if acting or not

- Trained to provide predictions both on displacement and on a **binary decision**
- Hybrid action space: continous and discrete

$$p_\theta(s) \in [0, 1]$$

$$b \sim \text{Bernoulli}(p_\theta(s))$$

$$\Delta \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$$



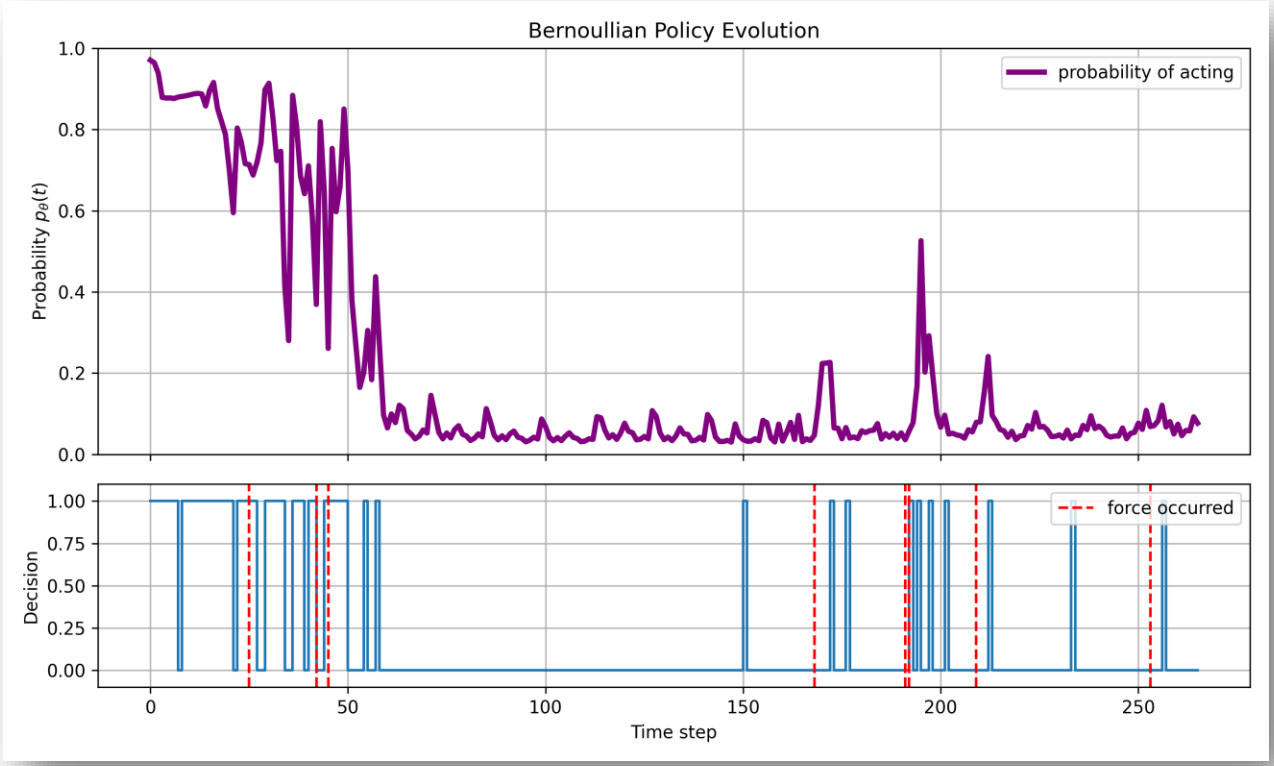
The agent learns to disable corrections to better reproduce the reference

Interventing just in the presence of **external forces** to recover balance

no **ground slope** here

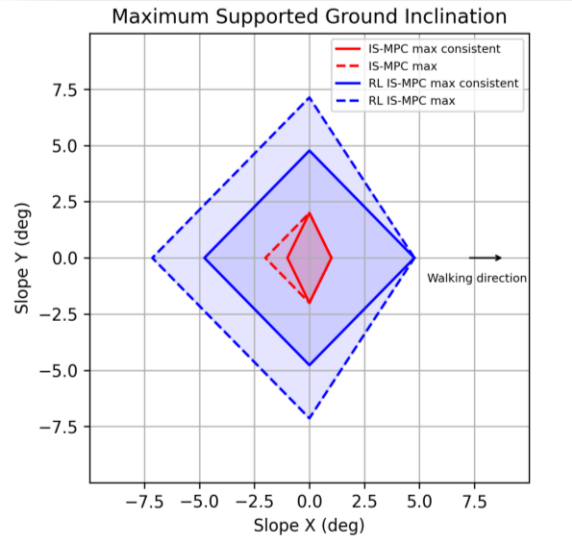
Hybrid Policy

Hybrid Gaussian-Bernoullian policy to decide if acting or not

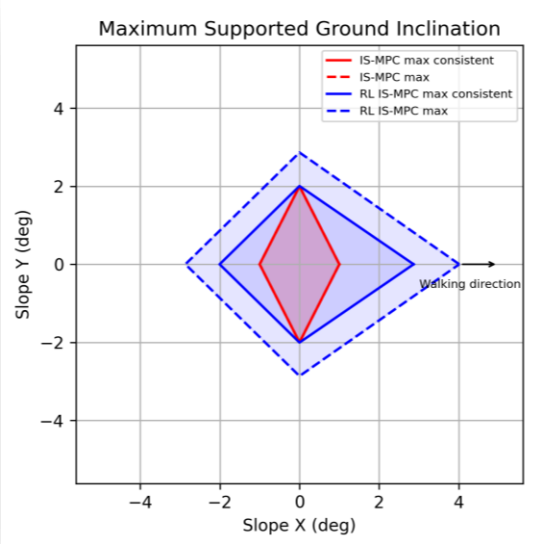


probability of acting increases as a **force** occurs behaves as a **Neural Residual**

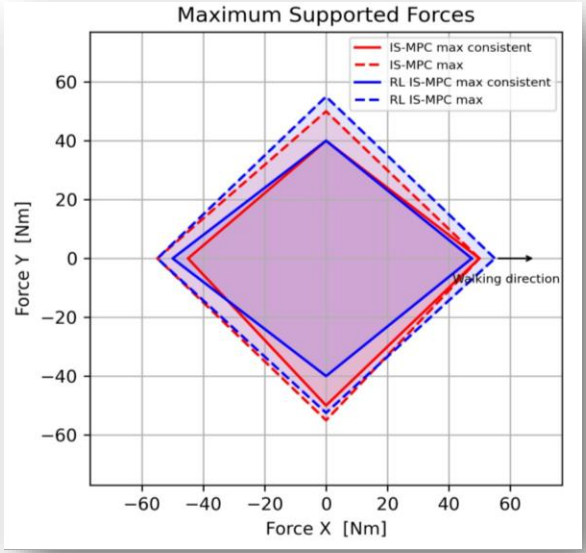
Final comparisons



Slope comparison:
Hybrid Gaussian-Bernoullian policy
VS
pure IS-MPC



Slope comparison:
Approach 3 scaled 0.9
VS
pure IS-MPC



Forces comparison:
Approach 3 scaled 0.9
VS
pure IS-MPC

CONCLUSIONS

Conclusions

Recap of our work

- We augmented a MPC-based control scheme for humanoid locomotion with a Reinforcement-Learning based agent.
- The agent is trained with the **PPO** algorithm, and displaces the footsteps of the plan to **reject disturbances** and keep balance on inclined terrain.
- We explored **three different approaches** to displacement:
  - Only the next footstep: displacement may invalidate the plan. Very good tracking, but poor response to disturbance forces.
  - Whole plan: Saturated displacements are always feasible, but poor tracking.
  - Scaled: best of both worlds.
    - ✓ Better performance when scaling parameter is part of the action space

Conclusions

On generalization

- The agents do not generalize well to new footstep plans:
  - ✓ It's hard to modify the **ZMP/CoM relationship** by just moving footsteps (thus only changing MPC constraints)
  - ✓ IS-MPC assumes **fixed timings** for single and double support phases, in the reference paper they are computed by MPC

- **Saturations** to prevent feet collision lead to better generalization and behaviour in general, by effectively reducing the action space
  - ✓ Less failures -> more regular learning
  - ✓ The agent doesn't know about saturations, but better learns that displacement should be near zero in general, greater only with disturbances

- We also looked to methods for deciding ***when*** to apply the action, to displace footsteps only when strictly necessary, more precise than just saturating actions