

Bridging Model-based Safety and Model-free Reinforcement Learning through System Identification of Low Dimensional Linear Models

*Zhongyu Li,
Jun Zeng,
Akshay Thirugnanam,
Koushil Sreenath (UC Berkeley)*

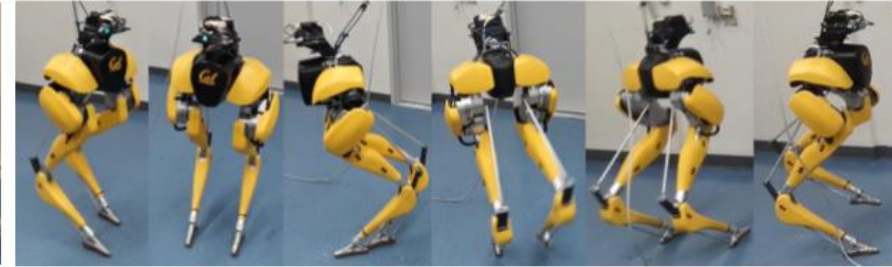
presented by Yancheng Liang
07/05/2022



(a) Fast Walking Outdoor



(b) Side Walking



(c) Turning



(d) Recover from Foot Sliding



(e) Unknown Load



(f) Anti-Slip

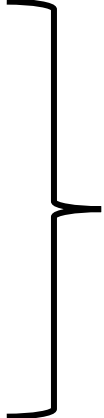


(g) Slippery

[1]

- RL controllers are flexible and perturbation-robust
- Questions: Where is the robustness from? How to formally show the stability?

Overall Framework

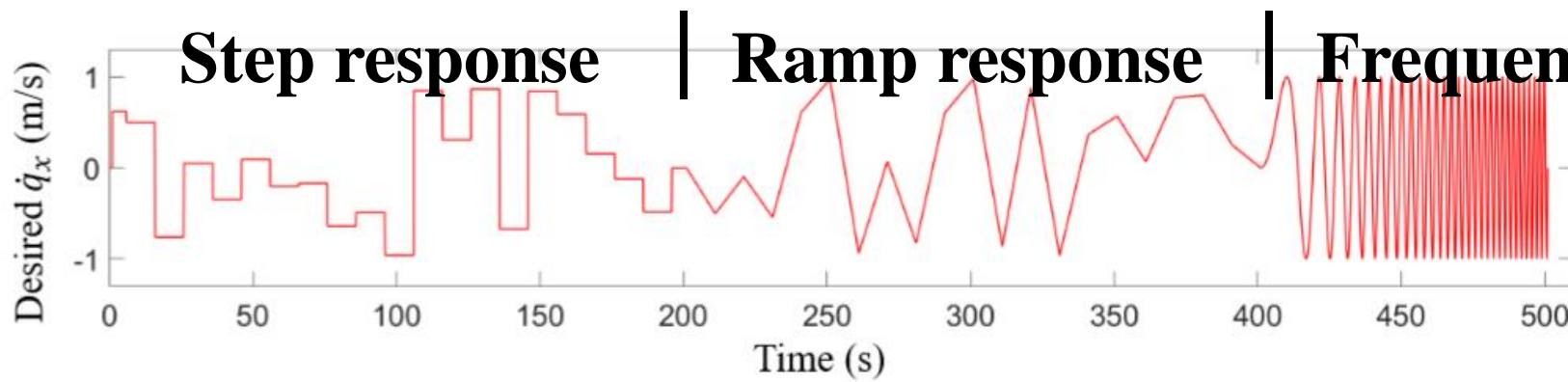
- Step 1: train a RL-based agent
 - Step 2: linearize the closed-loop system compromising the RL-controlled agent
 - Step 3: use the linear model to do safety-critical planning
- 
- There is a pipeline to get the linear model

Methodology Pipeline I: Train RL Policy

- First learn a RL policy $s_t \rightarrow a_t$, which can be arbitrary complex
 - The input contains reference motion, current and 4 timesteps past states
 - The output is the desired motor position which is then deployed by a PD controller
 - Two versions: a MLP-based policy and a CNN-based policy (convolute temporal information)
- In this study, bipedal robot Cassie is used
- The policy is trained to track a four-dimensional reference control
 - Sagittal speed \dot{q}_x^d , lateral speed \dot{q}_y^d , height q_z^d and turning yaw rate \dot{q}_ϕ^d

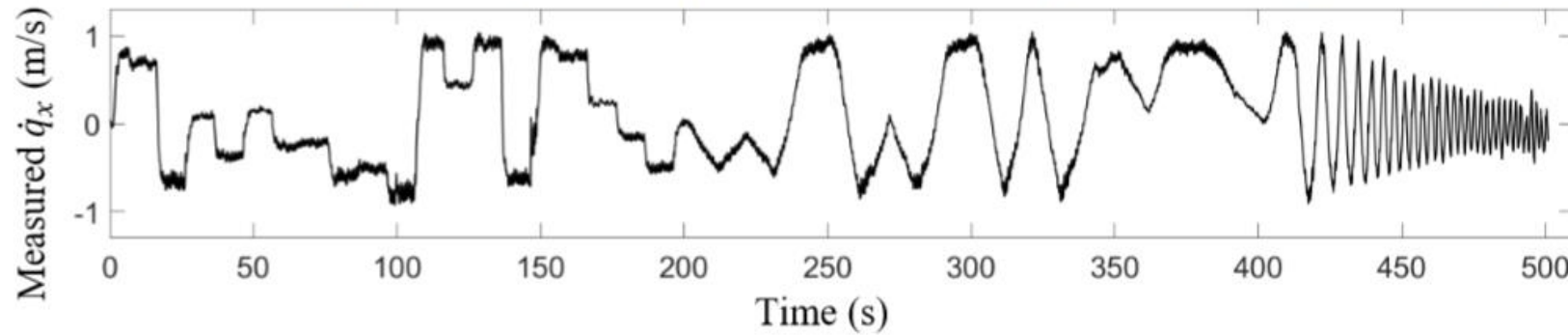
Methodology Pipeline II: Closed-Loop System

- Then identify a low-dimensional
 - The RL-controlled agent and the external environment compose a closed-loop system
 - Identify the closed-loop system with a low-dimensional model
- The input of the system is the control reference $u = [\dot{q}_x^d, \dot{q}_y^d, q_z^d, \dot{q}_\phi^d]$
- The output of the system is the observation $y = [\widehat{\dot{q}}_x^d, \widehat{\dot{q}}_y^d, \widehat{q}_z^d, \widehat{\dot{q}}_\phi^d]$ (after the agent takes actions according to RL policy)



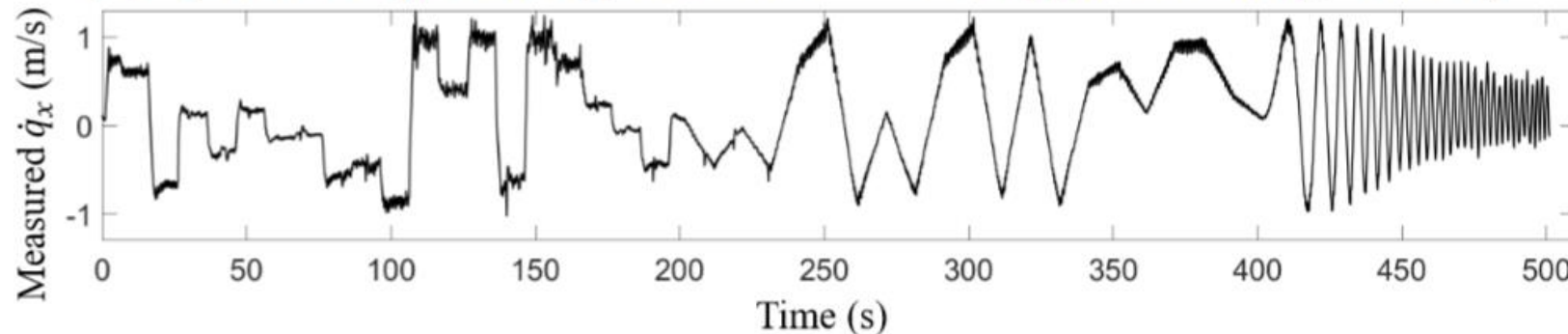
Control reference (input) u

(a) An Example of System Inputs: Desired Sagittal Walking Velocity



Observation (output) y with
MLP RL policy

(b) System Outputs using MLP: Measured Sagittal Walking Velocity



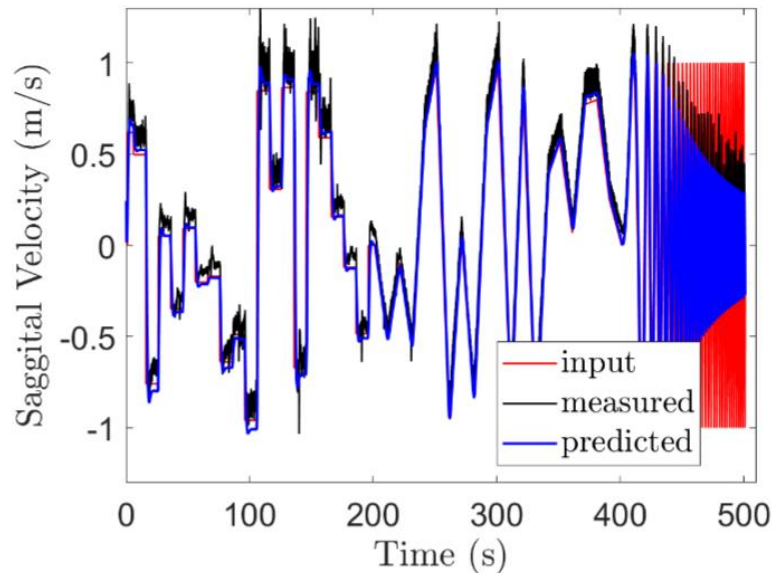
Observation (output) y with
CNN RL policy

(c) System Outputs using CNN: Measured Sagittal Walking Velocity

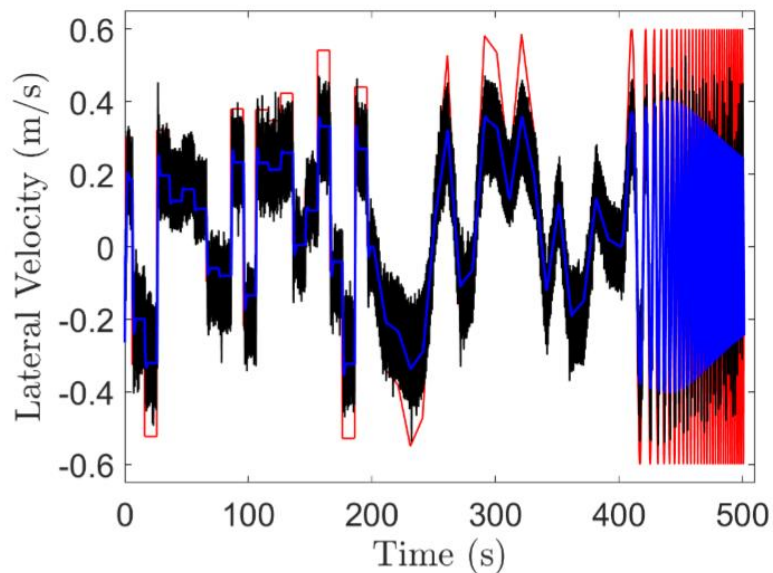
Methodology Pipeline III: Linearization

Input: $u = [\dot{q}_x^d, \dot{q}_y^d, q_z^d, \dot{q}_\phi^d]$ output: $y = [\widehat{\dot{q}_x^d}, \widehat{\dot{q}_y^d}, \widehat{q_z^d}, \widehat{\dot{q}_\phi^d}]$

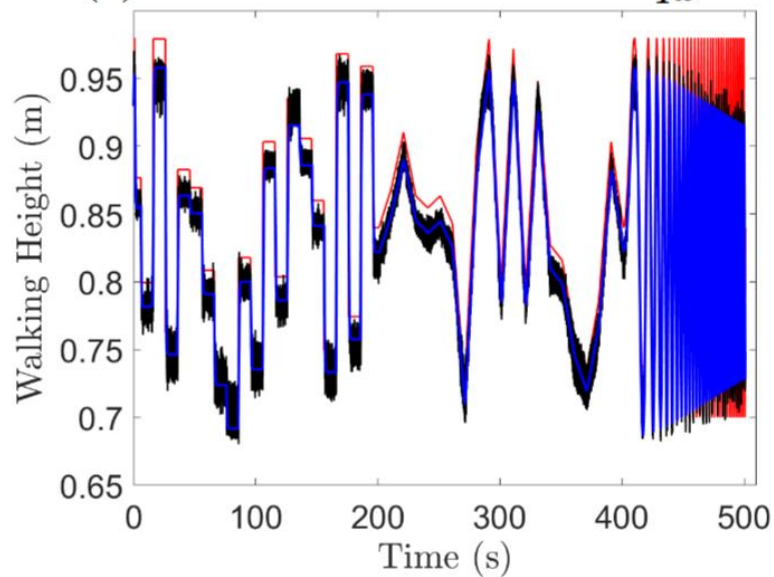
- Collect input-output pairs (u, y) of this system in a simulator
- Use linear model $\dot{x} = Ax + Bu, y = Cx$ to fit the collected data
 - [Fitting curves](#)
 - [Numerical results](#)
 - The linearity preserves when input frequency is under some threshold (0.6Hz)
- Finally we have a MIMO system with input $u \in R^4$ and output $y \in R^4$
 - Some [further analysis](#)



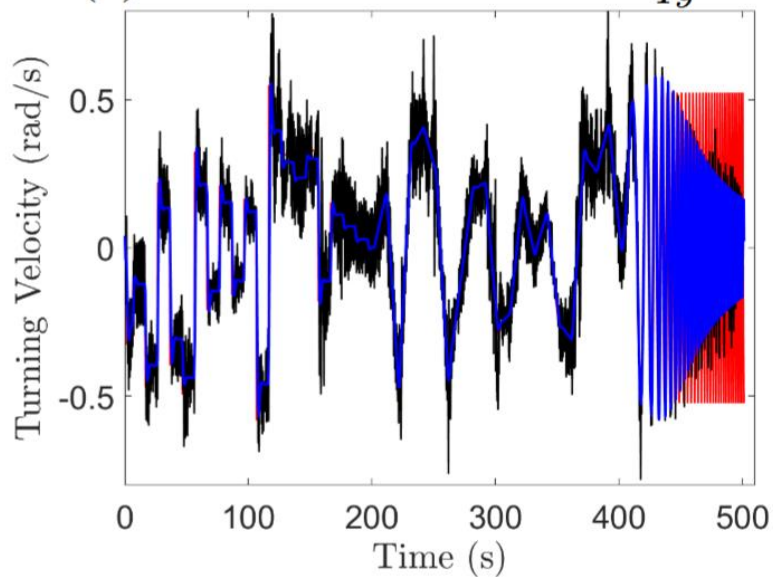
(a) Fitted Linear Model of \dot{q}_x



(b) Fitted Linear Model of \dot{q}_y



(c) Fitted Linear Model of q_z



(d) Fitted Linear Model of \dot{q}_ϕ

Red: system input u

Gray: system output
measured (ground truth)
 y

Blue: system output
predicted by the linear
model y_{pred}

[Back](#)

Transfer functions (CNN-based)

$$Y_{\dot{x}}(s) = \frac{0.4694s^2 + 6.089s + 8.697}{s^3 + 6.432s^2 + 11.03s + 8.274} U_{\dot{x}}(s) \quad (1)$$

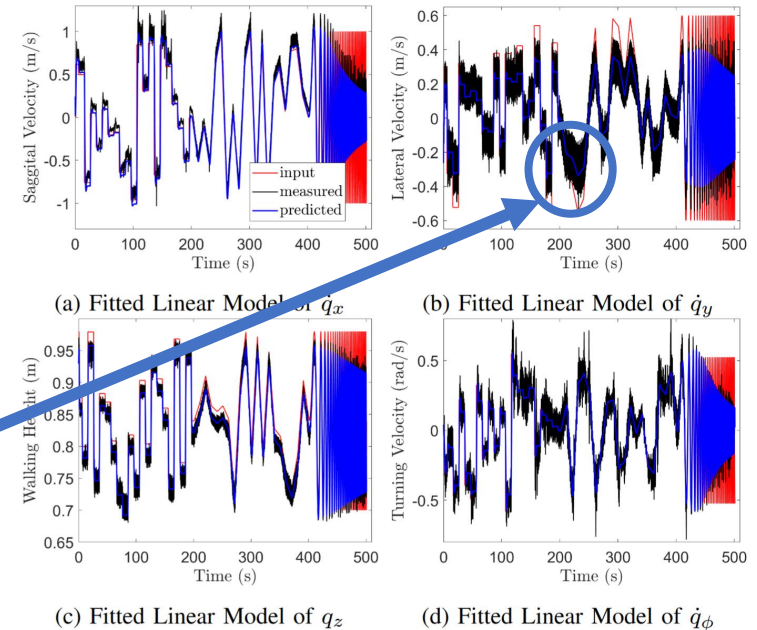
$$Y_{\dot{y}}(s) = \frac{13.59s + 24.56}{s^3 + 11.48s^2 + 32.5s + 40.13} U_{\dot{y}}(s) \quad (2)$$

$$Y_z(s) = \frac{145.9s + 37.55}{s^3 + 46.43s^2 + 161s + 38.38} U_z(s) \quad (3)$$

$$Y_{\dot{\phi}}(s) = \frac{0.3078s^2 + 5.267s + 5.553}{s^3 + 4.595s^2 + 7.528s + 6.045} U_{\dot{\phi}}(s) \quad (4)$$

Input: $u = [\dot{q}_x^d, \dot{q}_y^d, q_z^d, \dot{q}_\phi^d]$ output: $y = [\hat{q}_x^d, \hat{q}_y^d, \hat{q}_z^d, \hat{q}_\phi^d]$

- The fitting accuracy is around 80% (relative error) after filtering the **oscillation** with lowpass filter



Poles and Zeros

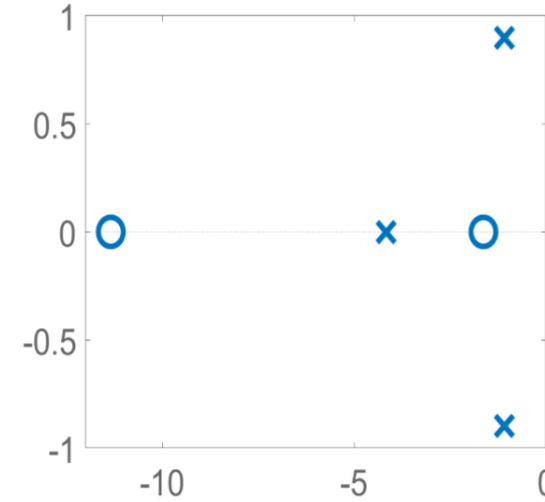
$$Y_{\dot{x}}(s) = \frac{0.4694s^2 + 6.089s + 8.697}{s^3 + 6.432s^2 + 11.03s + 8.274}U_{\dot{x}}(s) \quad (1)$$

$$Y_{\dot{y}}(s) = \frac{13.59s + 24.56}{s^3 + 11.48s^2 + 32.5s + 40.13}U_{\dot{y}}(s) \quad (2)$$

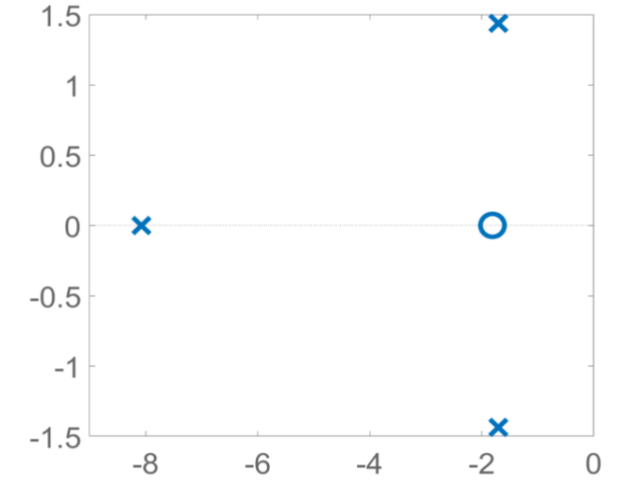
$$Y_z(s) = \frac{145.9s + 37.55}{s^3 + 46.43s^2 + 161s + 38.38}U_z(s) \quad (3)$$

$$Y_{\dot{\phi}}(s) = \frac{0.3078s^2 + 5.267s + 5.553}{s^3 + 4.595s^2 + 7.528s + 6.045}U_{\dot{\phi}}(s) \quad (4)$$

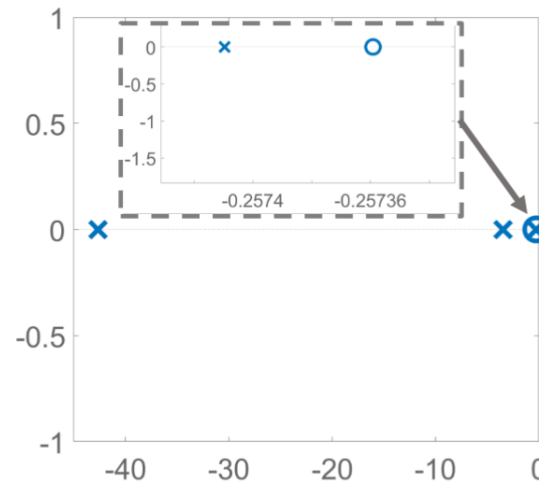
For CNN-based RL policy, all poles and zeros are in left hand plate (LHP);
For MLP-based RL policy, some zeros are not in LHP



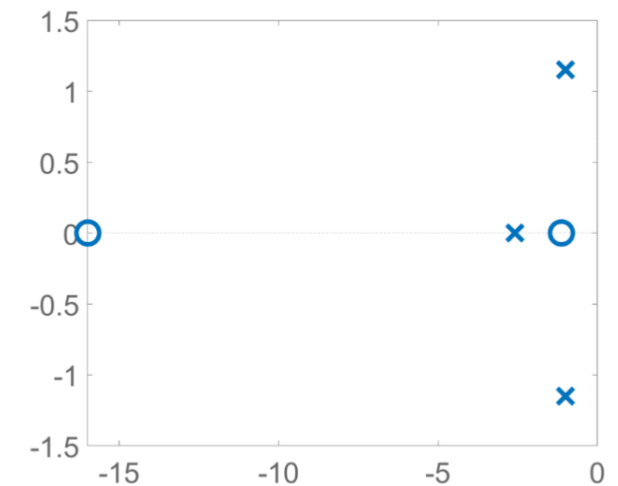
(a) Pole-Zero Plot of \dot{q}_x



(b) Pole-Zero Plot of \dot{q}_y



(c) Pole-Zero Plot of q_z

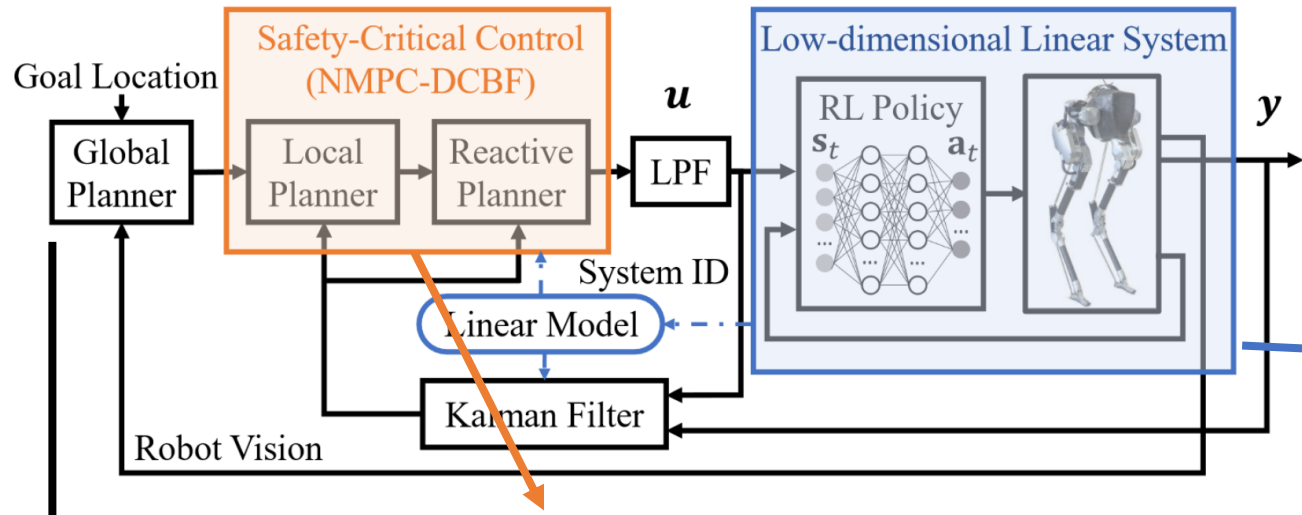


(d) Pole-Zero Plot of \dot{q}_ϕ

Analysis

- They also test the dependences among the four inputs and find
 - With chirp signal **frequency less than the threshold** (0.6Hz), **the system is considerably decoupled**
 - Higher frequency may excite the coupled dynamics (like between \dot{q}_x and q_z)
- They do some ablation study by substitute the policy with other policies with different training process
 - Linearity is not affected by rewards/NN architectures or other hyperparameters
 - If the RL policy is not converged, there is poor linearity (**convergence is important for linearity**)

Case Study: Safe Navigation



$$\begin{bmatrix} \dot{\mathbf{x}}_{\dot{x},\dot{y},z,\dot{\phi}} \\ \mathbf{y}_{\dot{x},\dot{y},z,\dot{\phi}} \end{bmatrix} = \begin{bmatrix} A_{\dot{x},\dot{y},z,\dot{\phi}} & B_{\dot{x},\dot{y},z,\dot{\phi}} \\ C_{\dot{x},\dot{y},z,\dot{\phi}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\dot{x},\dot{y},z,\dot{\phi}} \\ \mathbf{u}_{\dot{x},\dot{y},z,\dot{\phi}} \end{bmatrix}$$

NMPC-DCBF Framework [2]

$$\min_{\mathbf{x}, \mathbf{u}, \delta} J(\mathbf{x}, \mathbf{u}, \delta, \omega),$$

$$\text{s.t. } \mathbf{x}_{k+1} = A^d \mathbf{x}_k + B^d \mathbf{u}_k$$

$$\mathbf{x}_0 = \mathbf{x}_{init}$$

$$\mathbf{x}_k \in \mathcal{X}_{adm}, \mathbf{u}_k \in \mathcal{U}_{adm},$$

$$x_{k+1}^g = x_k^g + C_{\dot{x}}^d \mathbf{x}_{\dot{x},k} \cos(C_{\dot{\phi}}^d \mathbf{x}_{\phi,k}) dt,$$

$$y_{k+1}^g = y_k^g + C_{\dot{y}}^d \mathbf{x}_{\dot{y},k} \sin(C_{\dot{\phi}}^d \mathbf{x}_{\phi,k}) dt,$$

$$d_{k+1}^{o_i} \geq \omega_k \alpha_{DCBF} d_k^{o_i}, \quad \omega_k \geq 0,$$

$$[x_N^g, y_N^g, \mathbf{x}_{\phi,N}]^T = [x_f^g, y_f^g, \mathbf{x}_{\phi,f}]^T + \delta$$

Cost function

$$J(\mathbf{x}, \mathbf{u}, \delta, \omega) = \sum_{i=1}^{N-1} (\|\mathbf{x}_i\|_Q^2 + \|\mathbf{u}_i\|_R^2 + \|\mathbf{x}_{i+1} - \mathbf{x}_i\|_{d_Q}^2 + \rho(1 - \omega_k)^2) + \|\delta\|_K^2.$$

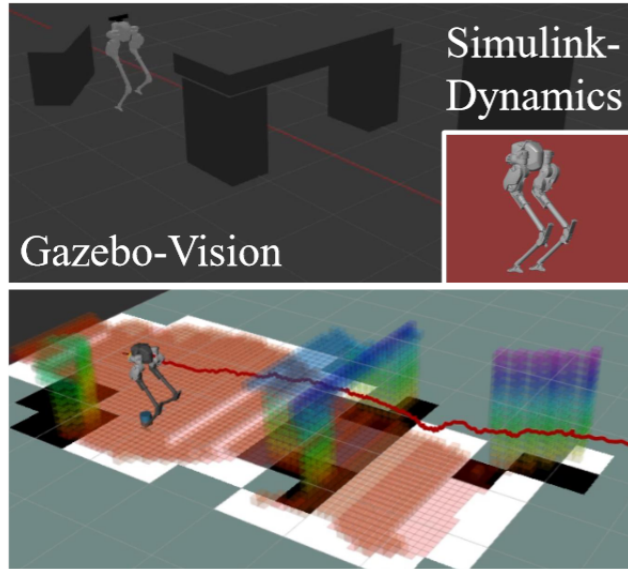
Linear model

Discrete-time CBF constraints
with distance-based CBF

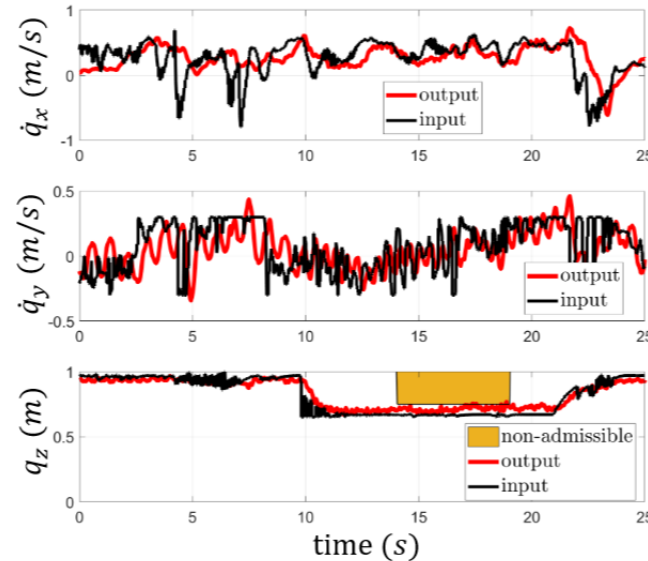
$$d_k^{o_i} = (x_k^g - x^{o_i})^2 + (y_k^g - y^{o_i})^2 - (R^{robot} + R^{o_i})^2,$$

Generate a
collision-free
trajectory

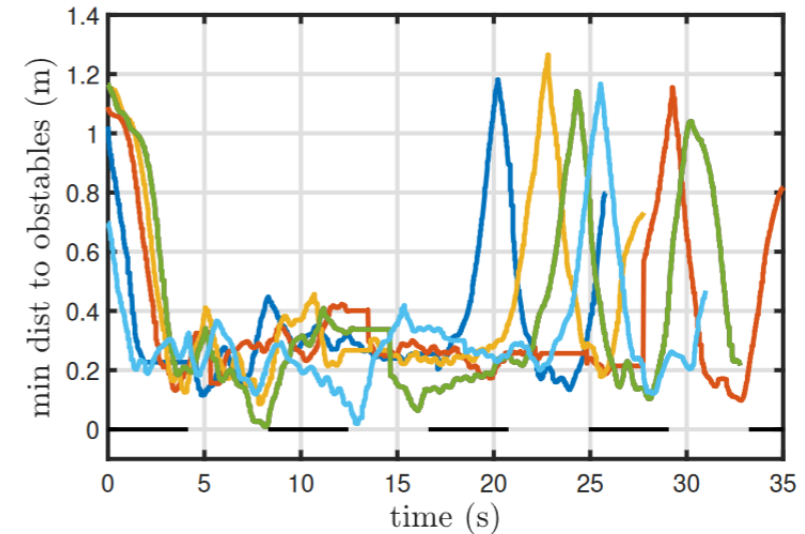
Case Study: Safe Navigation. Results



(a) Joint simulation of vision and dynamics using the safe navigation autonomy in a congested space



(b) The input (desired commands) and output (robot measured states) during navigation in Fig. 9a

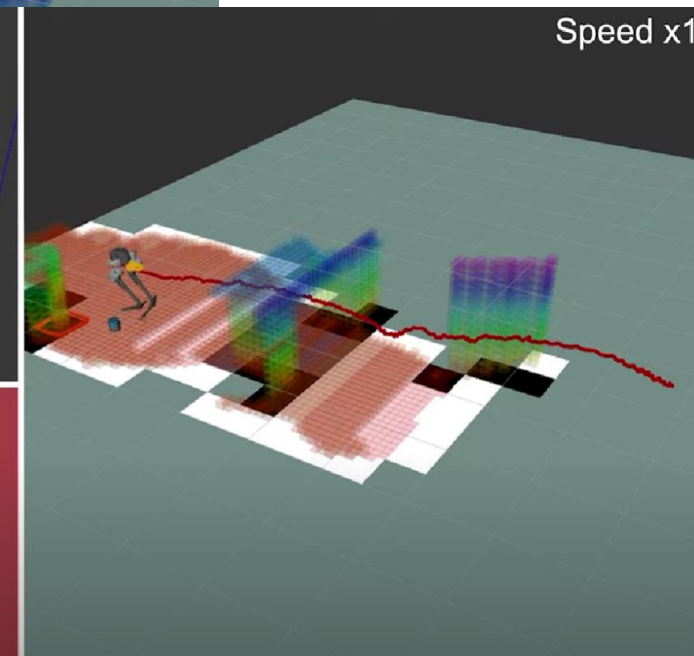
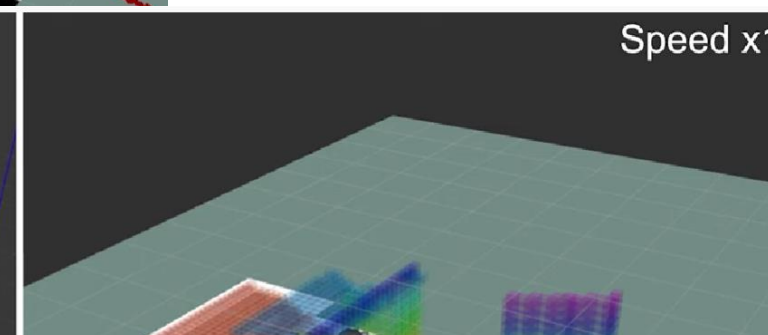
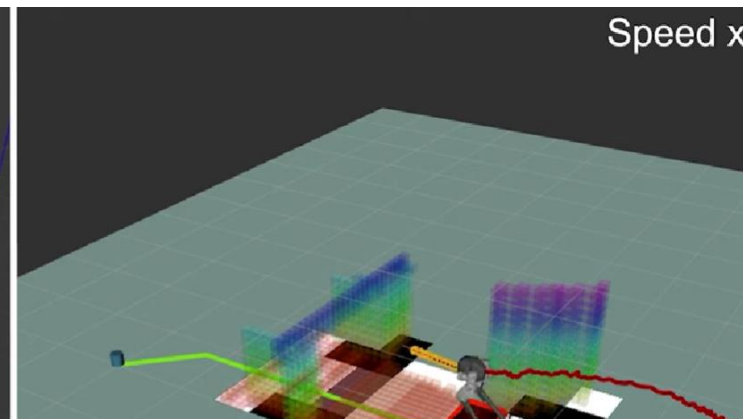


(c) The recorded distance between the robot and its closet obstacles for 5 repeated tests in the same scenario but with randomized initial conditions

Track the reference
control well

Avoid collision

- Compared with baseline (HZD-based controller, which does not have the linear model as in this study), 0.1s optimization (5x faster). And the navigation task can be finished in 25s (2x faster)



<https://sites.google.com/berkeley.edu/rl-sysid-rss2022>

Recap

- Suppose we have a RL locomotion policy, when
 - The RL policy is trained to converge
 - The input does not contain high-frequency signals
- We can build a closed-loop system which is linear and low-dimensional
- Then we can use this linear system to do safety-critical planning to produce a control reference for the RL-based policy to obtain the final policy

Ref.

- [1] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 2811–2817, 2021.
- [2] Jun Zeng, Zhongyu Li, and Koushil Sreenath. Enhancing feasibility and safety of nonlinear model predictive control with discrete-time control barrier functions. In 2021 60th IEEE Conference on Decision and Control (CDC), pages 6137–6144, 2021.
- [3] Zhongyu Li, Jun Zeng, Shuxiao Chen, and Koushil Sreenath. Vision-aided autonomous navigation of underactuated bipedal robots in height-constrained environments. arXiv preprint arXiv:2109.05714, 2021.