Advanced Control Strategies Based on Reinforcement Learning for Linear Actuators

Stat.AI

Damian Tamburi, MSc. (dtamburi@stataisolutions.com)
Cristian Napole, PhD. (cnapole@stataisolutions.com)

Scope

The objective is to develop and evaluate a controller based on Reinforcement Learning for a second-order dynamic model with application in linear actuators and compare it with a classical PID control method.

Supported by

BIC ARABA UP! EUSKADI

GRUPO spri TALDEA

EUSKO JAURLARITZA
GOBIERNO VASCO
INDUSTRIA, TRANTSIZIO ENERGETIKO ETA JASANGARRITASUN SAILA
DEPARTAMENTO DE INDUSTRIA, TRANSICIÓN ENERGÉTICA Y SOSTENIBILIDAD

Index

Application Example

Application Example

Stewart Platform

Reinforcement Learning Algorithms

Reinforcement Learning Algorithms

Uses Cases

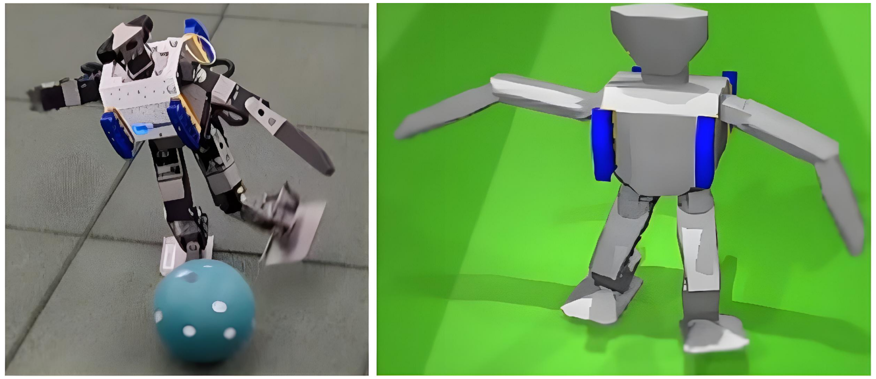
off policy

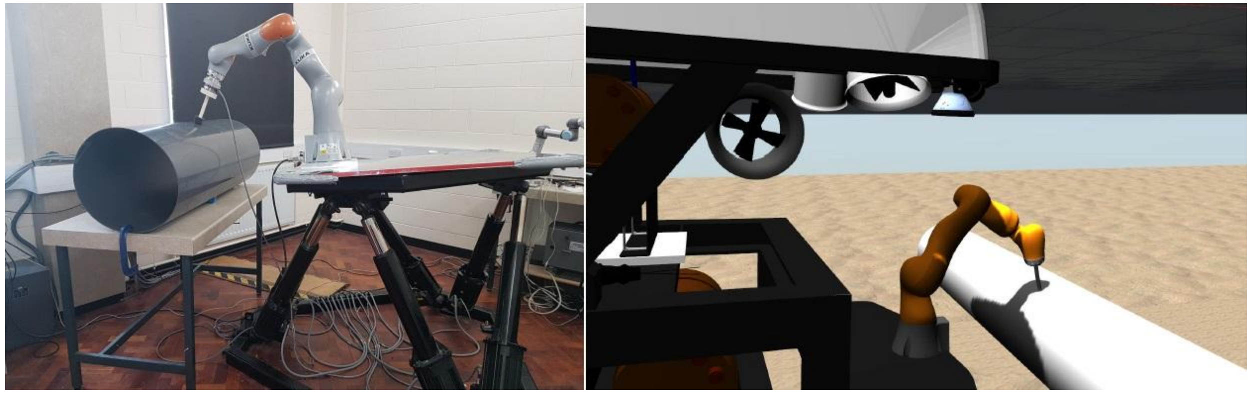
Deep Q Learning:
Discrete Actions.

On policy

Hybrid

PPO/ DDPG:
Continuous Actions.





Reinforcement Learning Algorithms

Basic Reinforcement Learning

$$\text{Experience} = \{(s_t, a_t, r_t, s_{t+1} + \text{extra}_t)\}_{t=0}^{T}$$

- Off Policy
- On Policy

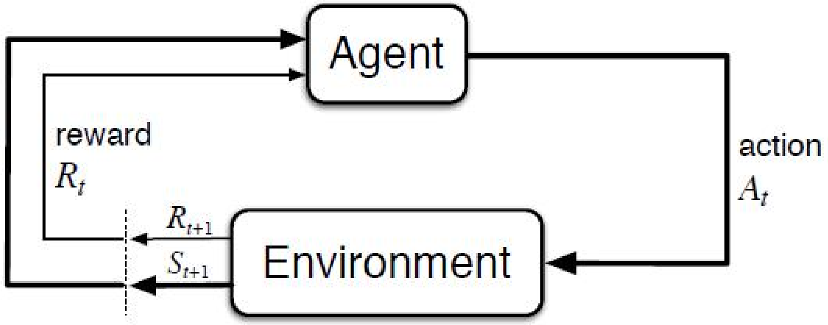
Actor

$$T = [S_t, A_t, R_t, S'_t]$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Critic

$$v_\pi(s) = E_\pi[G_t | s]$$

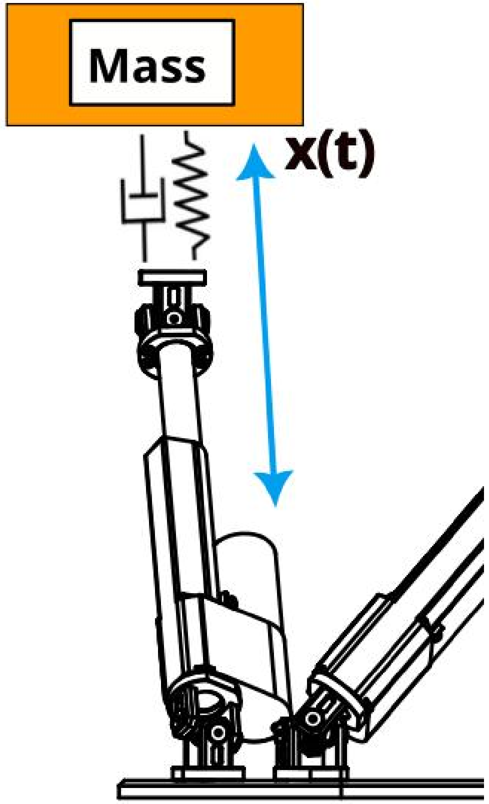


Markov Process

Controller Implementation

Controller Implementation

Reinforcement Learning environment.



Gymnasium

$$m\ddot{x} + b\dot{x} + kx = V$$
$$V = K.F$$

State Space

- Position
- Velocity
- Error = Position - goal
- Cumulative error

Action Space:

- V [-1 , 1]

[-1 m, 1 m] randomly each episode

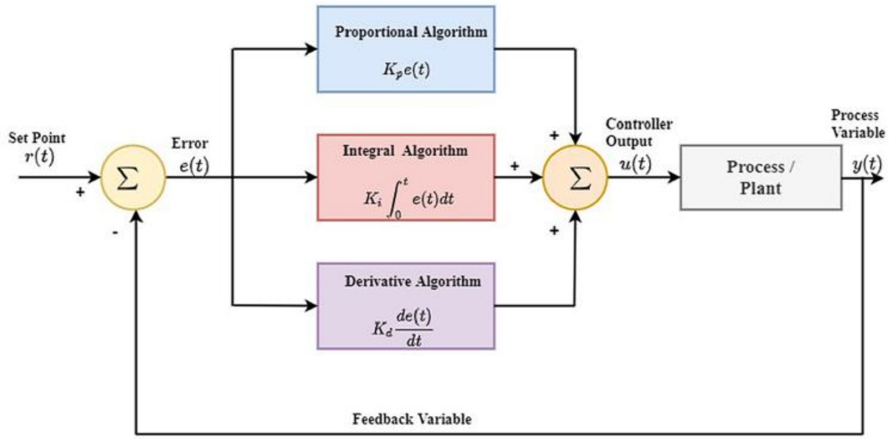
Done: if $(|x - \text{goal}| < 0.001$ and $|\text{Velocity}| < 0.001)$ OR current step $\geq$ max steps

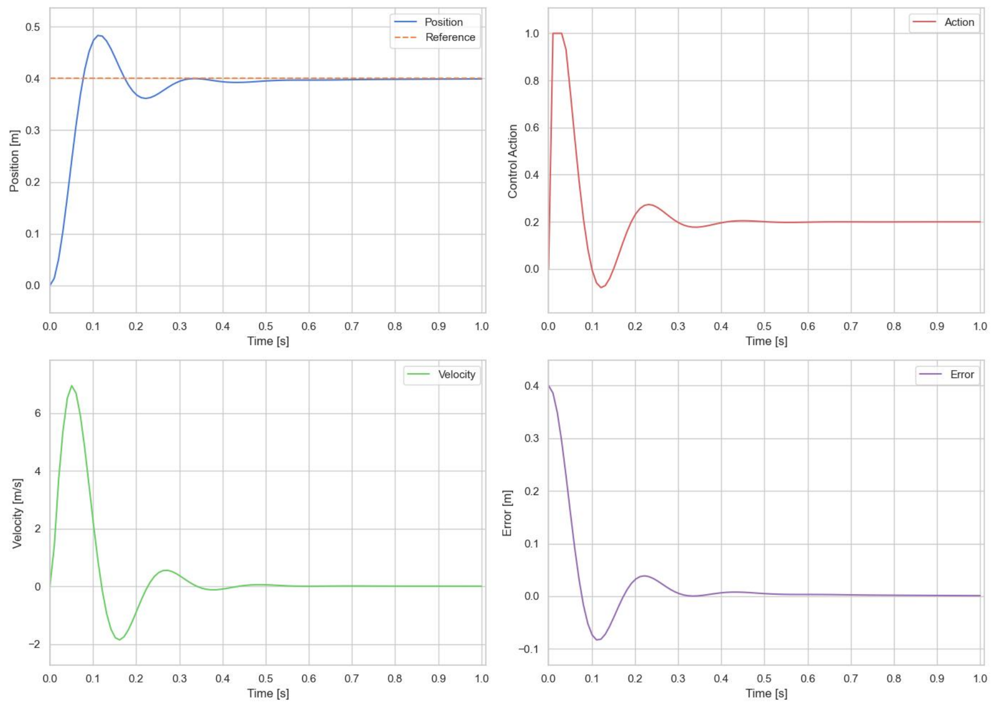
Parameter	Value	Description
K	300 N	Constant K
ζ	1.1	Damping Factor
k	100 N/m	Spring Constant
m	1 kg	Mass

$$\zeta = \frac{b}{2\sqrt{km}}$$

Controller Implementation

PID Tuning





Parameter	Value
Setpoint Step	0.4 m
Initial Distance (x_0)	0 m
Initial Speed (v_0)	0 m/s
Step Time	0.01 s
Number of Steps	100
Simulation Time	1 s
Proportional Gain (K_p)	2.9
Integral Gain (K_i)	10
Derivative Gain (K_d)	0.0125

Controller Implementation

Reinforcement Learning Training configuration.

Parameter	Parameters Phase 1	Parameters Phase 2
rank	4	4
net_arch	[32, 32, 16, 8]	[32, 32, 16, 8]
Optimizer	Adam	Adam
activation_fn	th.nn.Tanh	th.nn.Tanh
dropout_p	0	0
use_batch_norm	False	False
norm_obs	True	True
norm_reward	True	True
gamma	0.99	0.99
n_steps	**256**	**512**
ent_coef	0.1	0.1
learning_rate	0.000025	0.000025
vf_coef	0.5	0.5
max_grad_norm	0.5	0.5
gae_lambda	0.95	0.95
n_epochs	4	4
batch_size	**64**	**128**
clip_range	0.2	0.2

A2C

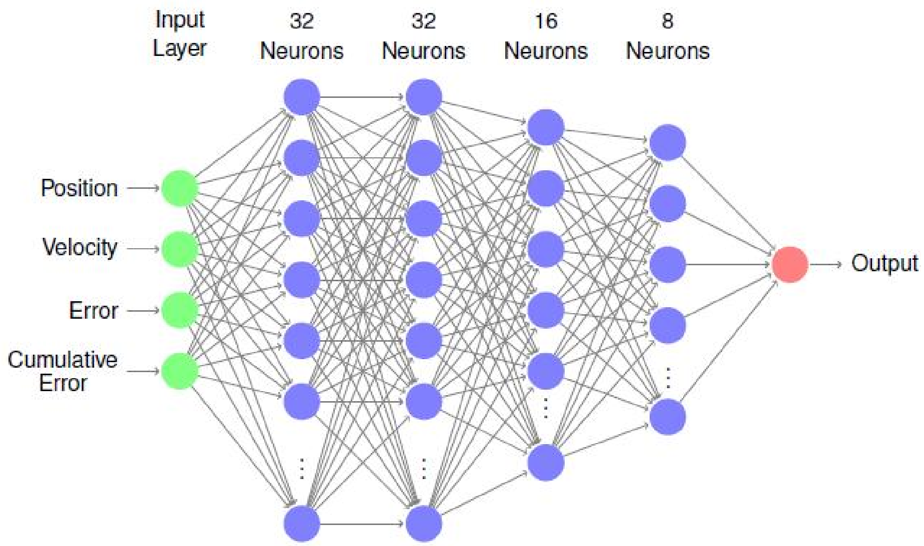
T (steps)

Learning Rate

Batch Size

Stable Baseline 3

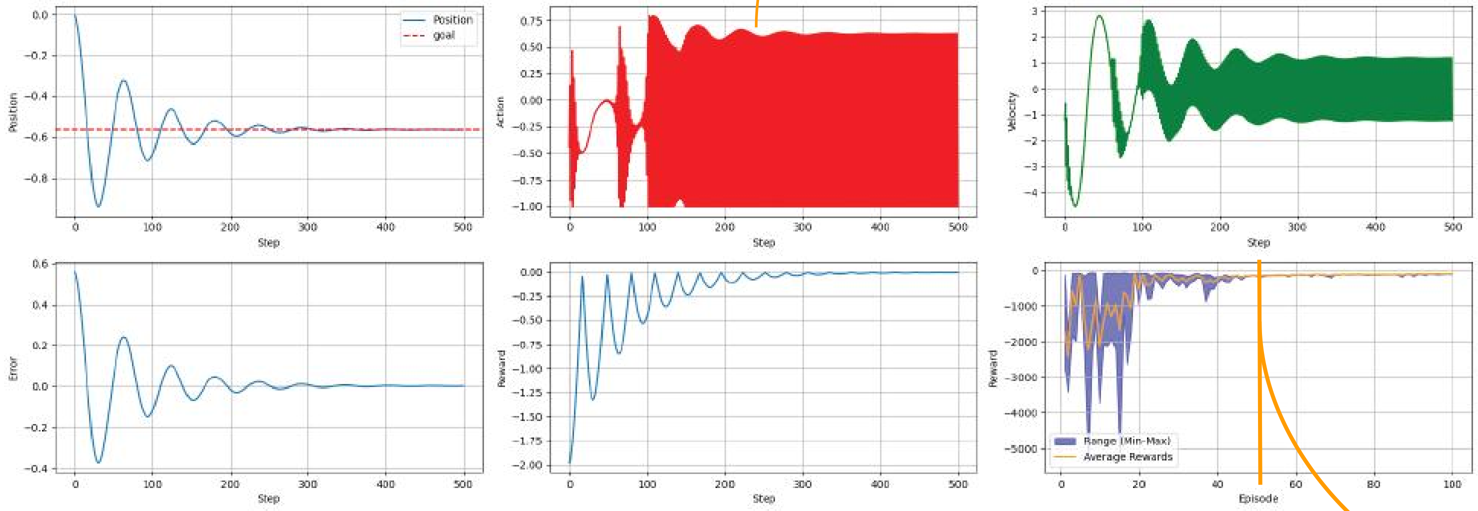
PyTorch



Controller Implementation

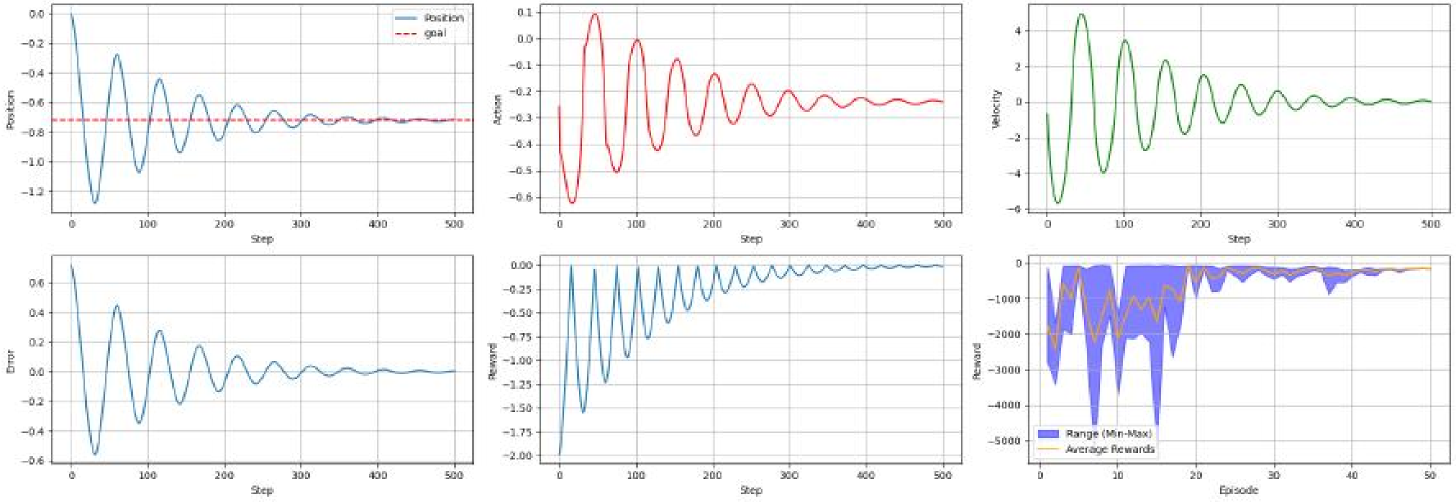
Reinforcement Learning Phase 1.

Control Action Oscillation.



$$-c_1 \times \frac{|x - \text{goal}|}{\max(|\text{goal}|, 0.01)}$$

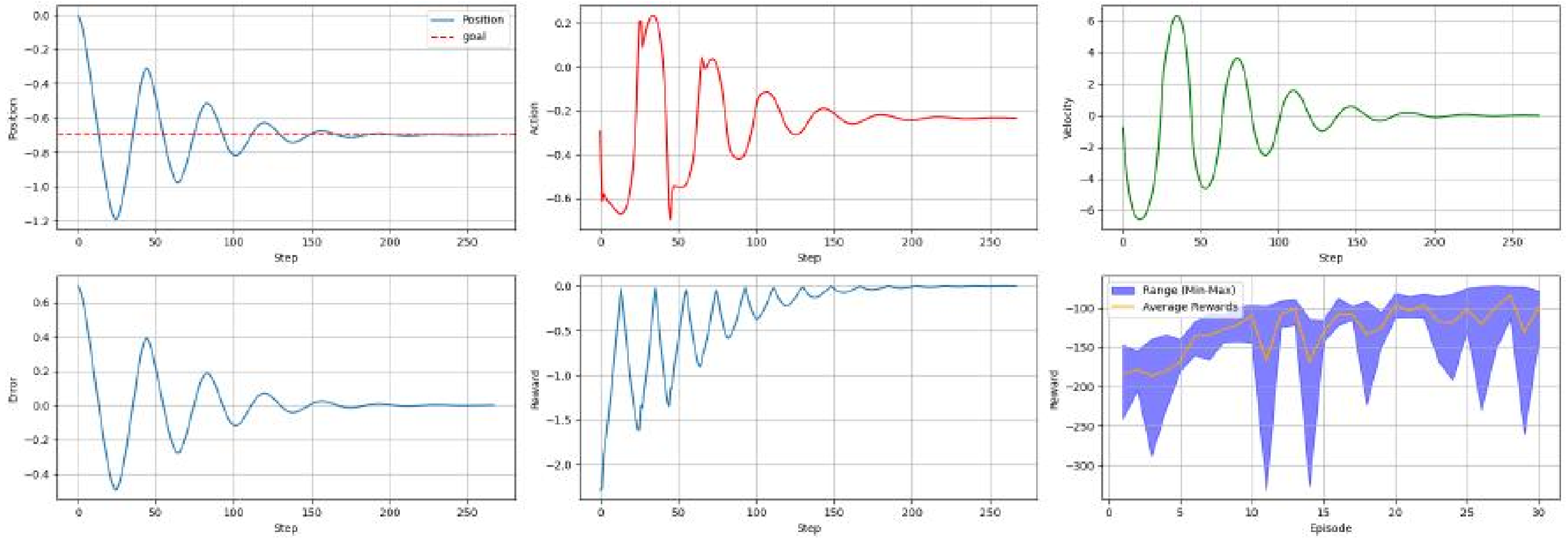
Reward Function.



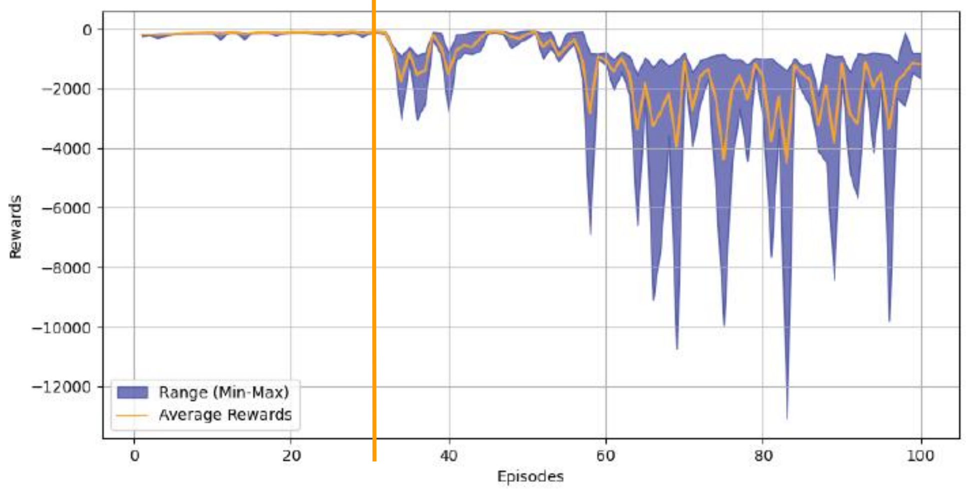
End of Phase 1

Controller Implementation

Reinforcement Learning Phase 2.



End of Phase 2.



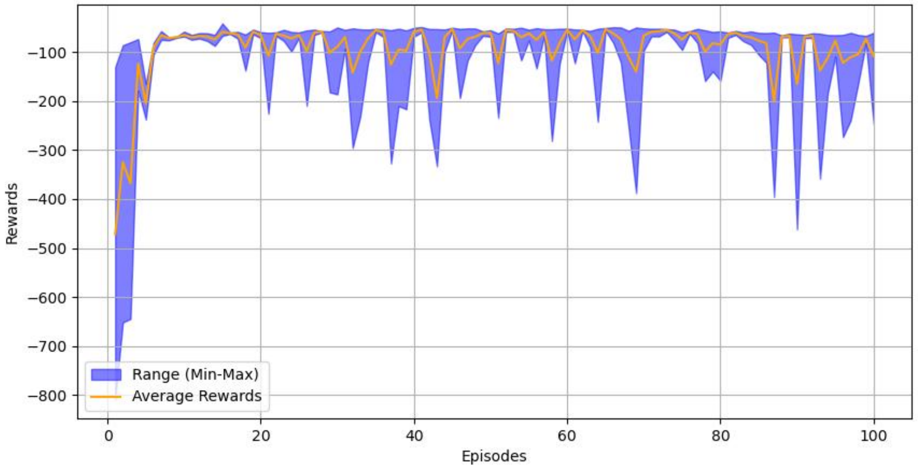
Reward Function.

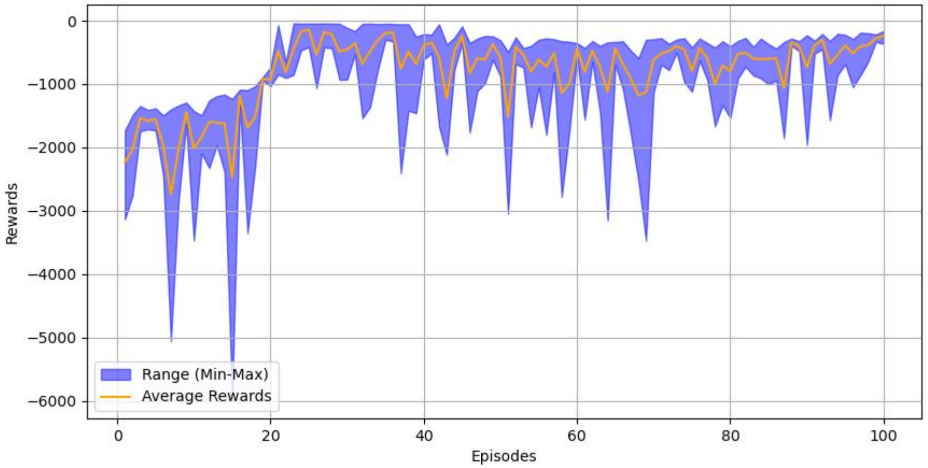
$$-c_1 \times \frac{|\mathrm{x} - \mathrm{goal}|}{\max(|\mathrm{goal}|, 0.01)} - c_2 \times |\mathrm{action} - \mathrm{previous\ action}|$$

Controller Implementation

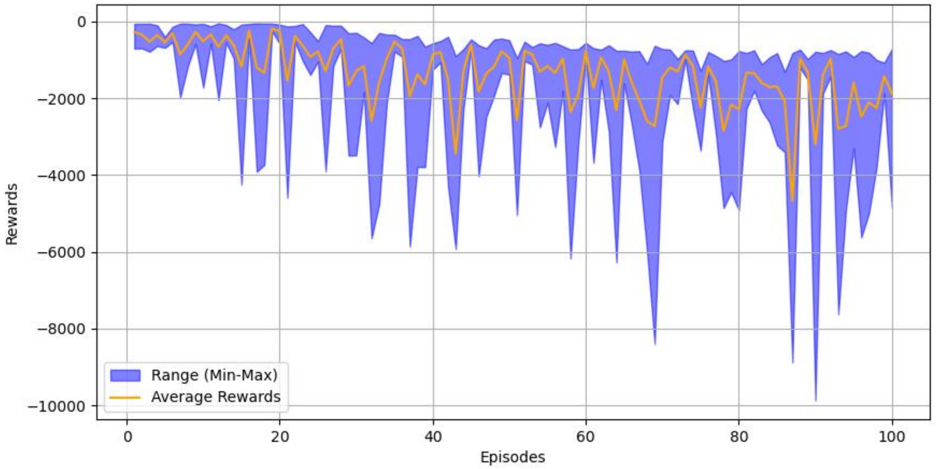
Other Hyperparameters

Simple Neural Network 32_16_8





Learning Rate increasing to 0.0025

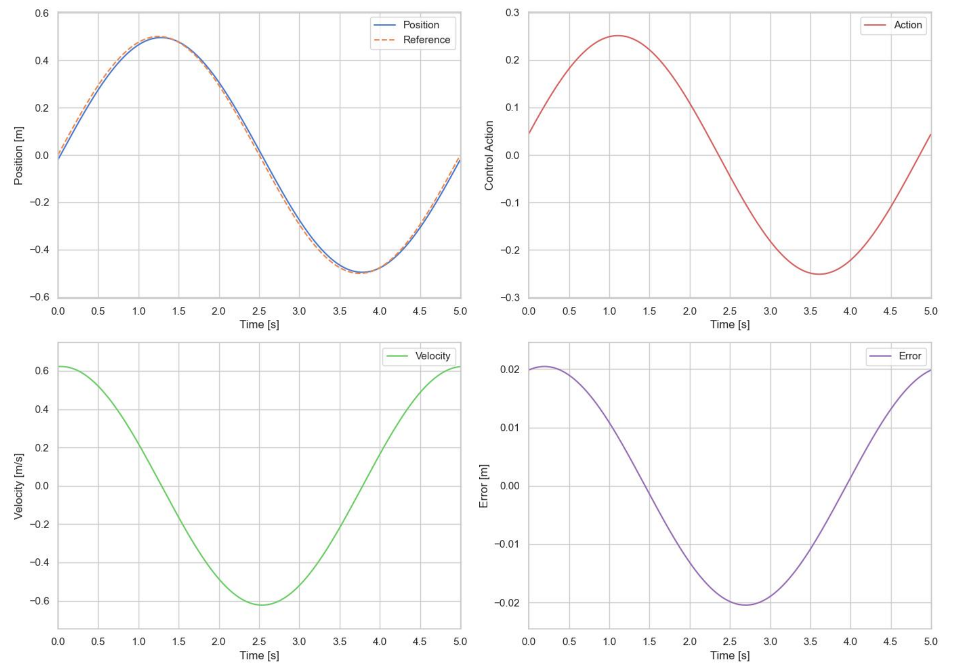


Simple Neural Network 8_16_32

Controller Implementation

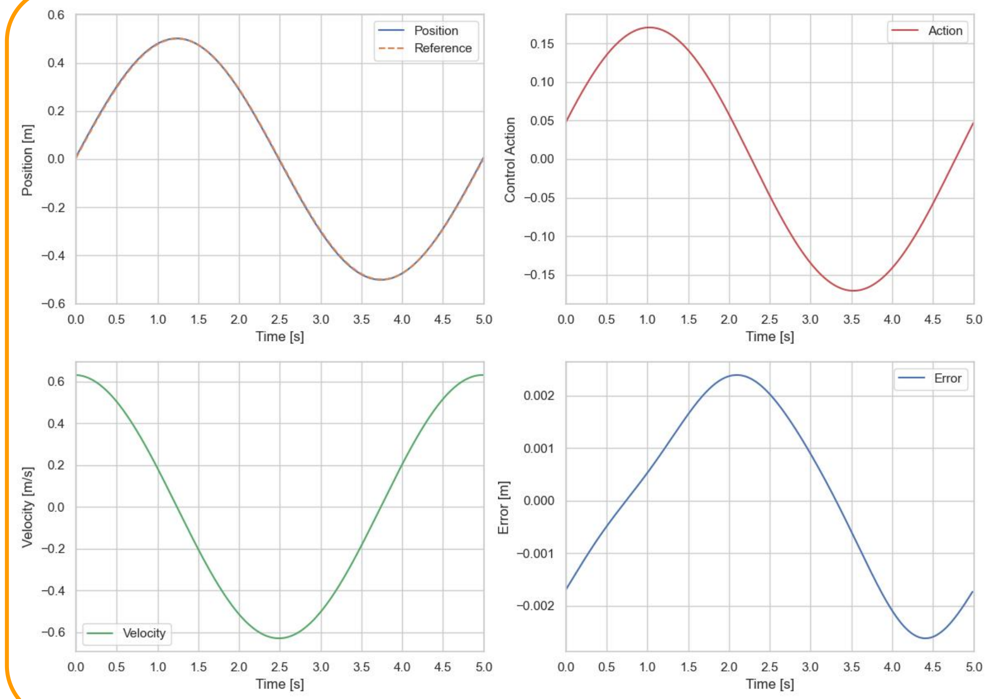
Result Comparison

PID



Absolute Error ≈ 0.02m

RL

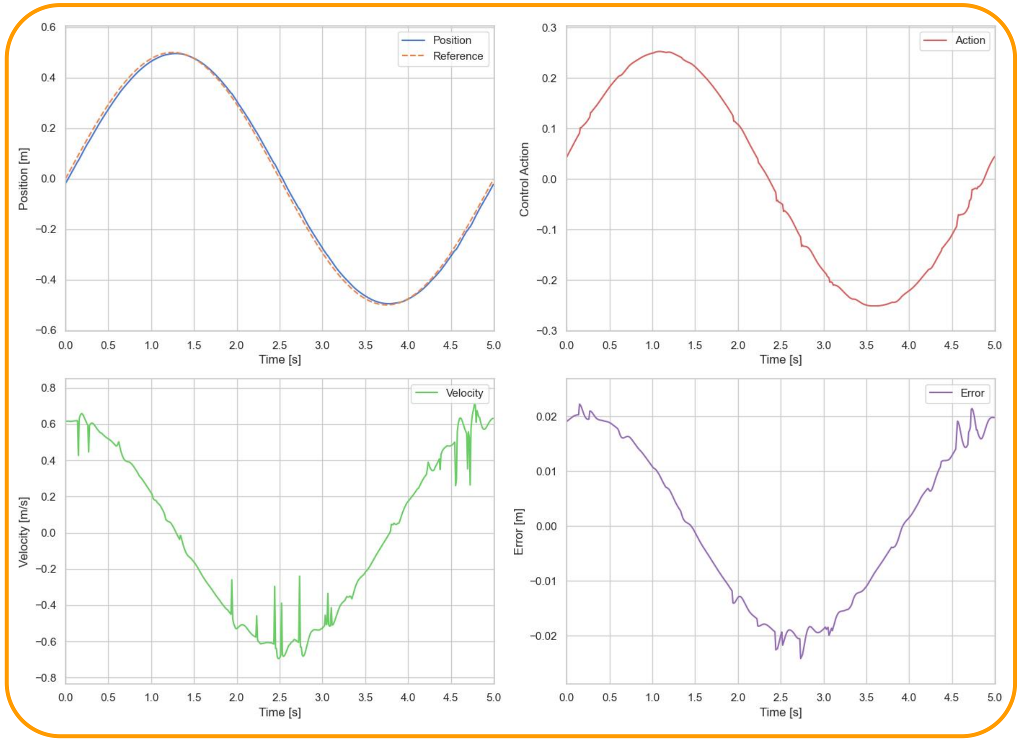


Absolute Error ≈ 0.0025m

Controller Implementation

Dynamic Noise Analysis.

PID



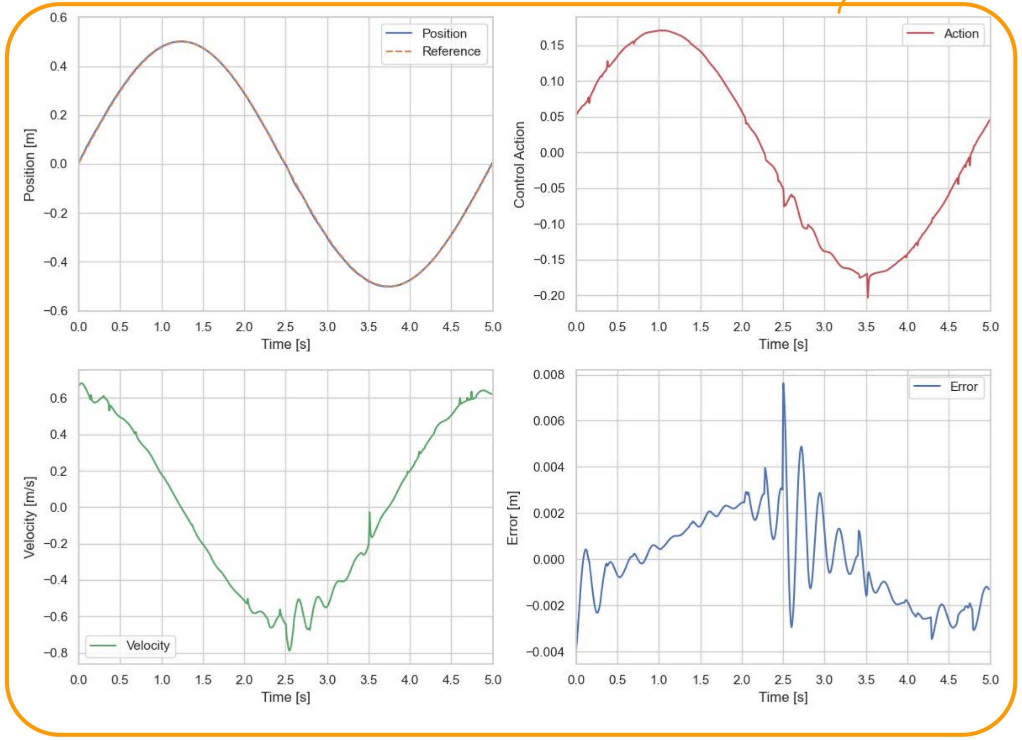
Absolute Error ≈ 0.025m

[-0.1K, 0.1K]
with a 20% of probability
each time step.

$$\frac{dv}{dt} = \frac{KF - bv - kx + \text{External Force}}{m}$$

RL

Action Peaks



Absolute Error ≈ 0.008m

Conclusions and Future Activities

Conclusions and Future Activities

Conclusions

- Reinforcement Learning algorithms are constantly evolving, improving control in continuous robotic environments.
- A reinforcement learning environment was designed for a second-order dynamic model using the Gymnasium library.
- A PID controller was implemented and tuned using the Ziegler-Nichols method, then manually readjusted.
- A PPO model was trained in 2 phases , with adaptive rewards..
- Sinusoidal references were applied to compare both controllers showing how PPO outperformed PID in terms of absolute errors and amplitude action reduction .
- Although the reinforcement learning controller was more effective, its training is complex and computationally expensive compared to PID.
- Despite its complexity, these technologies have great potential to complement traditional methods in engineering. The reinforcement learning controller can adapt to nonlinear systems and changing conditions.

Conclusions and Future Activities

Future Activities

- Improve the environment by changing parameters such as mass, spring, and damper coefficients, emulating classic active impedance controls, and adding disturbances to simulate failures and changing conditions.
- To validate the behavior of the trained Neural Network, it is necessary to conduct experimental tests in a physical environment and evaluate its real-time viability.
- Automating hyperparameter tuning is necessary to improve results and simplify the process.
- Fine-tune controllers on more complex signals, such as triangular ones.

Thanks!

https://stataisolutions.com/

Contact us:
dtamburi@stataisolutions.com
cnapole@stataisolutions.com