CONTROL OF AN INVERTED PENDULUM ON A CART

Name	Matric No.
Tashfin Muqtasid	2119609
Ahmad Faiz Najmi Bin Ahmad Fairus	2116267
Muhammad Irfan Bin Mohd Helmy	2112727

System Introduction

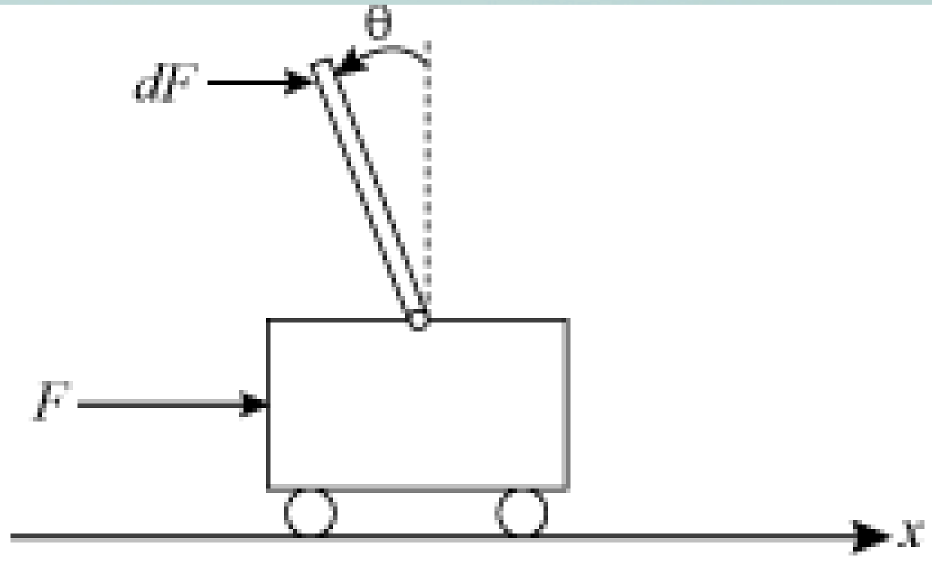
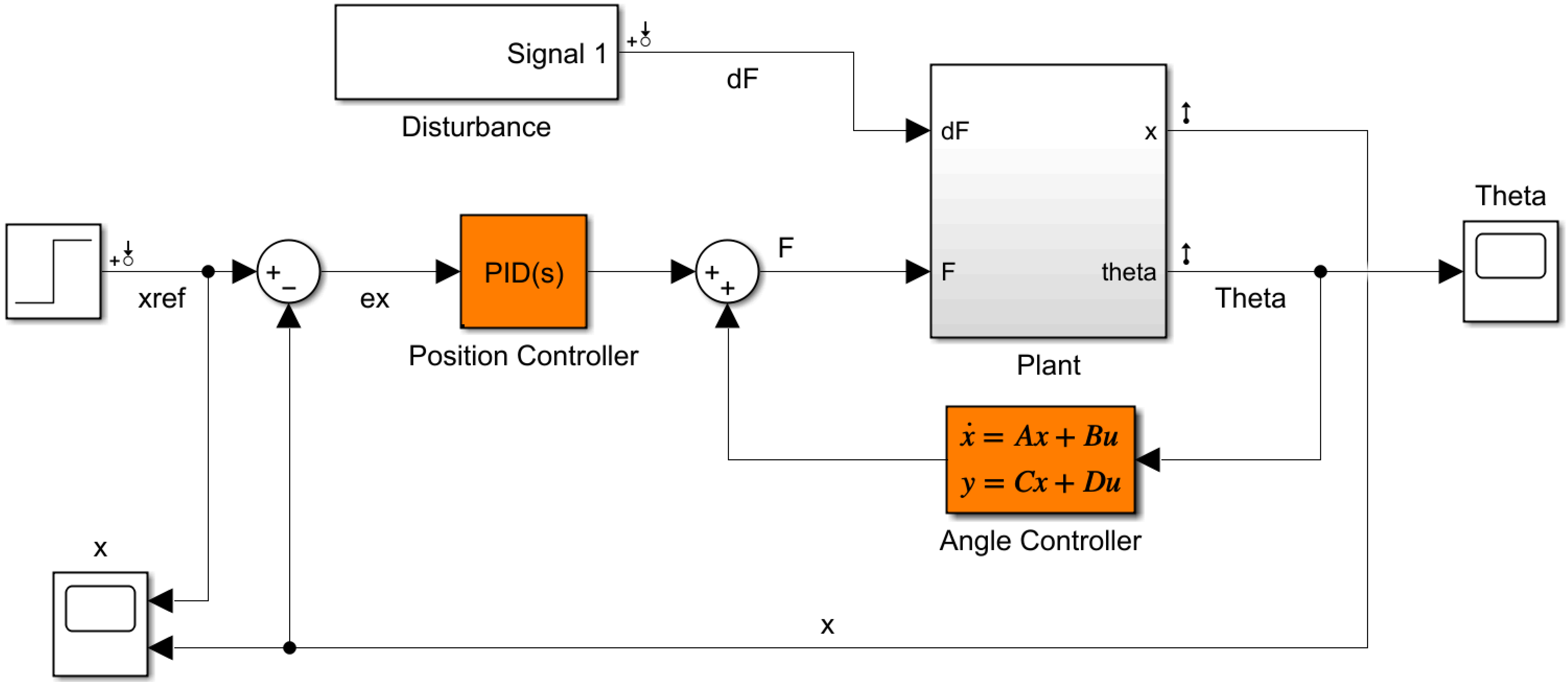


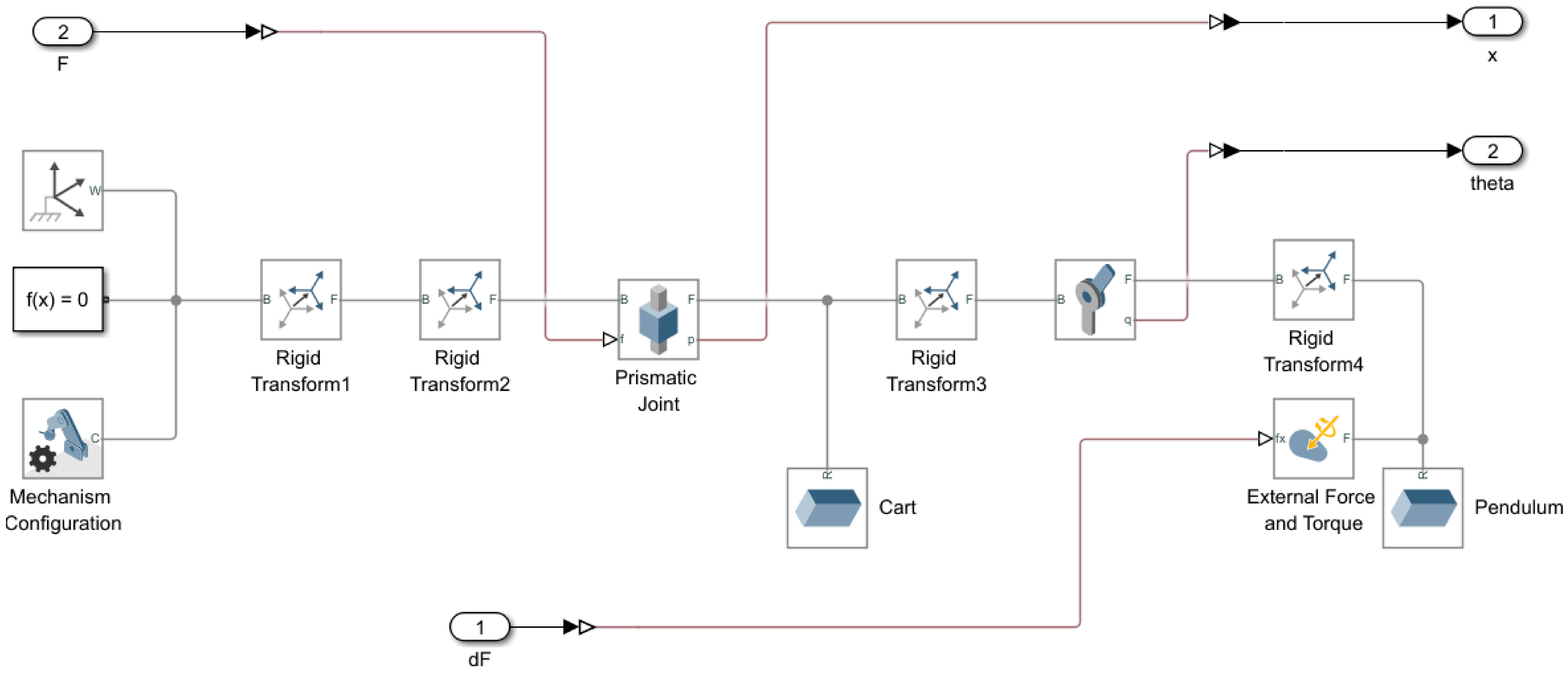
Figure 1: Inverted pendulum on a cart

- Small cart with a pole (pendulum) attached to it. The natural state of the pendulum is to fall down (theta = 180)
- **Objective:** Keep the pendulum upright. (theta = 0)
- **Control:** Move the cart left or right with a horizontal force *F*, which is the input to the system. By moving the cart, we can adjust the position of the pendulum.

Simulink Model



Plant


2
F
1
x
2
theta
W
f(x) = 0
C
Mechanism Configuration
B
F
Rigid Transform1
B
F
Rigid Transform2
B
F
f
p
Prismatic Joint
R
Cart
B
F
Rigid Transform3
B
F
q
B
F
Rigid Transform4
fx
F
External Force and Torque
R
Pendulum
1
dF

Controller Design

1. Angle Controller

- State Space Controller (2nd Order) to determine force required.
- **Input** : Pendulum angle
- **Output:** Force needed to move pendulum to desired angle.

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

2. Position Controller

- PD Controller
- **Input** : Xref - X
- **Output:** Force needed to reach target position of the cart.

$$C(s) = K_p + K_d \frac{s}{T_f s + 1}$$

Controller Tuning

- Set objectives of the system.
- Use MATLAB's systune function to automatically determine parameters

Objectives

- Position Tracking: Settling within 3 seconds.
- Disturbance Rejection: After a nudge, the controller must stabilize the pendulum, prioritizing a small angle deviation and minimal force.
- Robust Stability: Error Buffer (6dB) to avoid aggressive motion
- Smooth Response: By using damping (min 0.5)

```
% Tracking of x command
req1 = TuningGoal.Tracking('xref','x',3);
% Rejection of impulse disturbance dF
Qxu = diag([16 1 0.01]);
req2 = TuningGoal.LQG('dF',{'Theta','x','F'},1,Qxu);
% Stability margins
req3 = TuningGoal.Margins('F',6,40);
% Pole locations
MinDamping = 0.5;
MaxFrequency = 45;
req4 = TuningGoal.Poles(0,MinDamping,MaxFrequency);
% Connecting to Simulink
ST0 = slTuner('rct_pendulum',{'Position Controller','Angle Controller'})
addPoint(ST0,'F');
% Tuning
rng(0)
Options = systuneOptions('RandomStart',5);
[ST, fSoft] = systune(ST0,[req1,req2],[req3,req4],Options);
```

Output:

```
C1 =

                s
  Kp + Kd * --------
             Tf*s+1

  with Kp = 5.84, Kd = 1.89, Tf = 0.0498
```

Position Controller

```
C2 =

  -1579.2 (s+12.96) (s+4.239)
  ---------------------------
     (s+133.5) (s-14.07)
```

Angle Controller

RL Control with DDPG

Action: Force (between -15N to 15N)

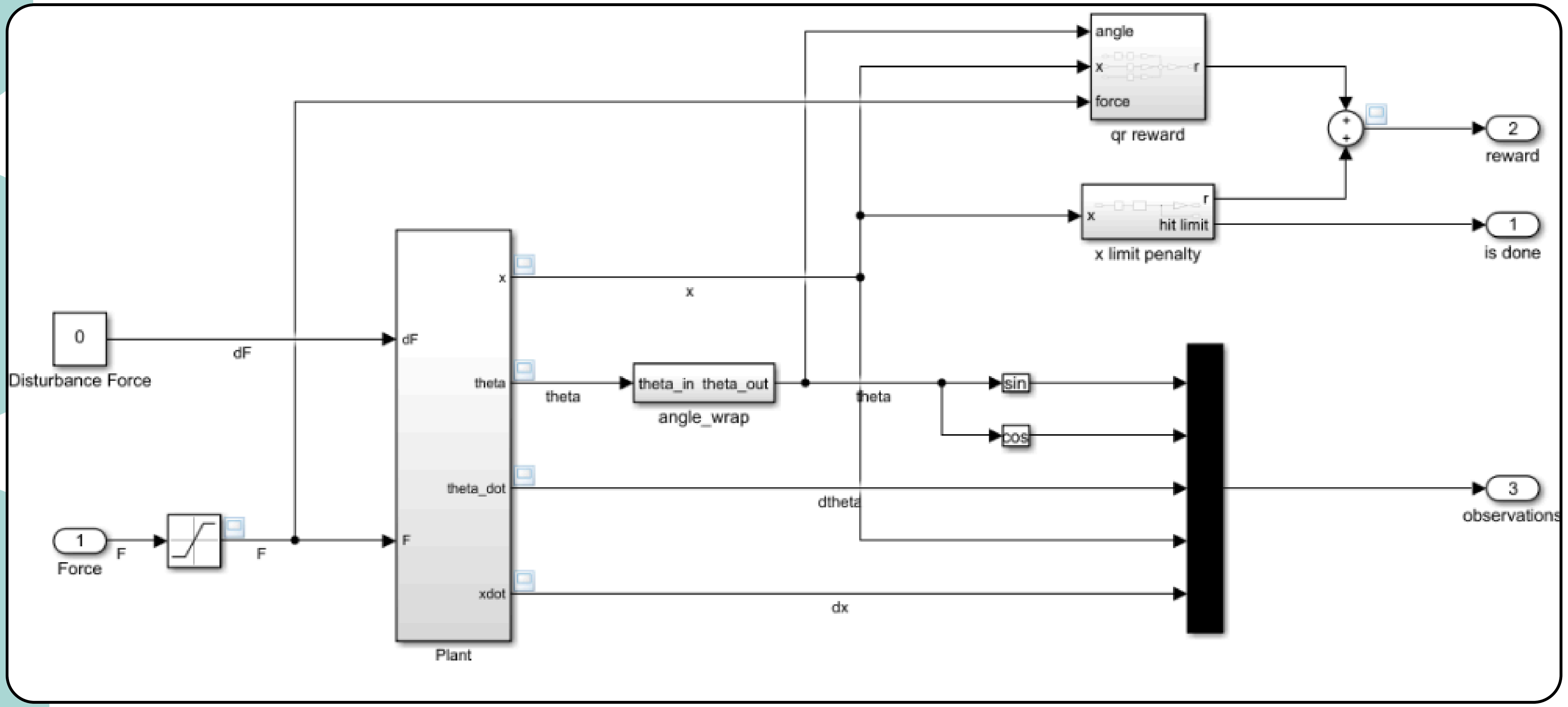
State (Observations): Position, Velocity, Angle

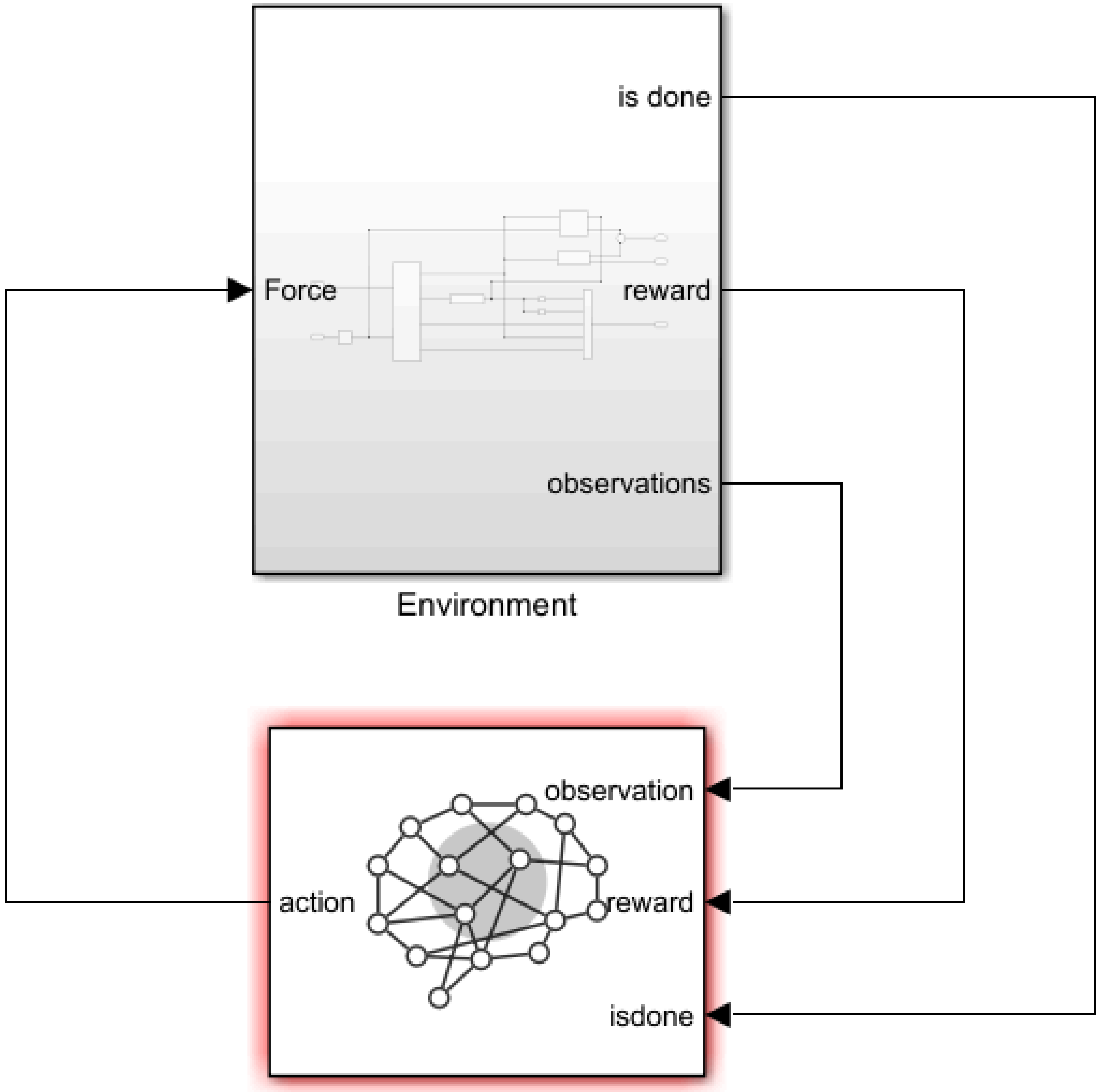
Reward Function:

$$r_t = -0.1\left(5\theta_t^2 + x_t^2 + 0.05u_{t-1}^2\right) - 100B$$

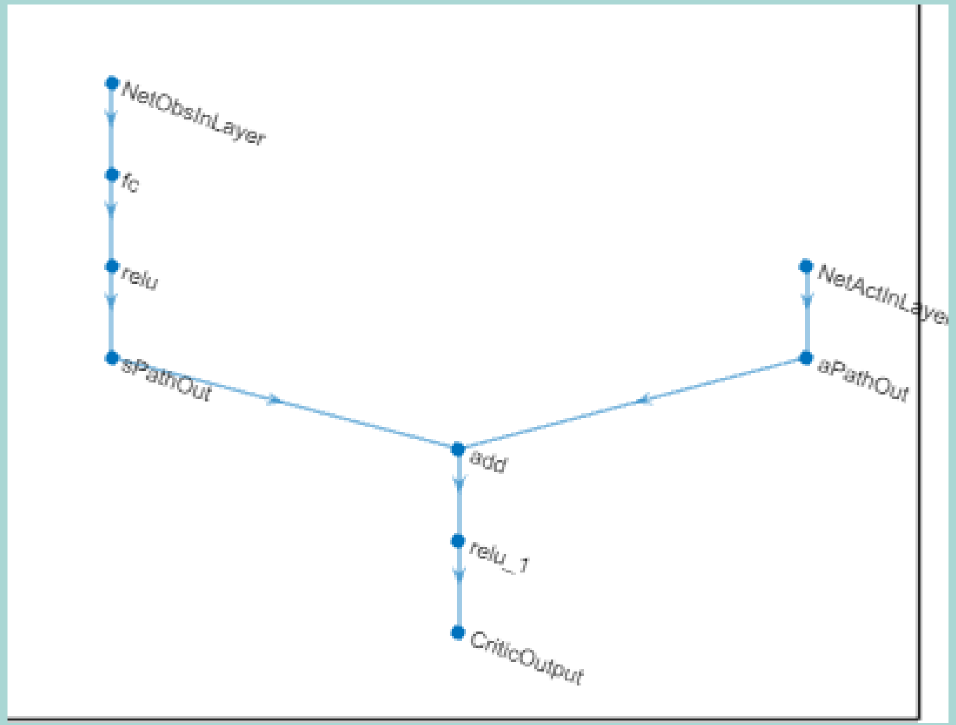
Here:

- θ_t is the angle of displacement from the upright position of the pole.
- x_t is the position displacement from the center position of the cart.
- u_{t-1} is the control effort from the previous time step.
- B is a flag (1 or 0) that indicates whether the cart is out of bounds.

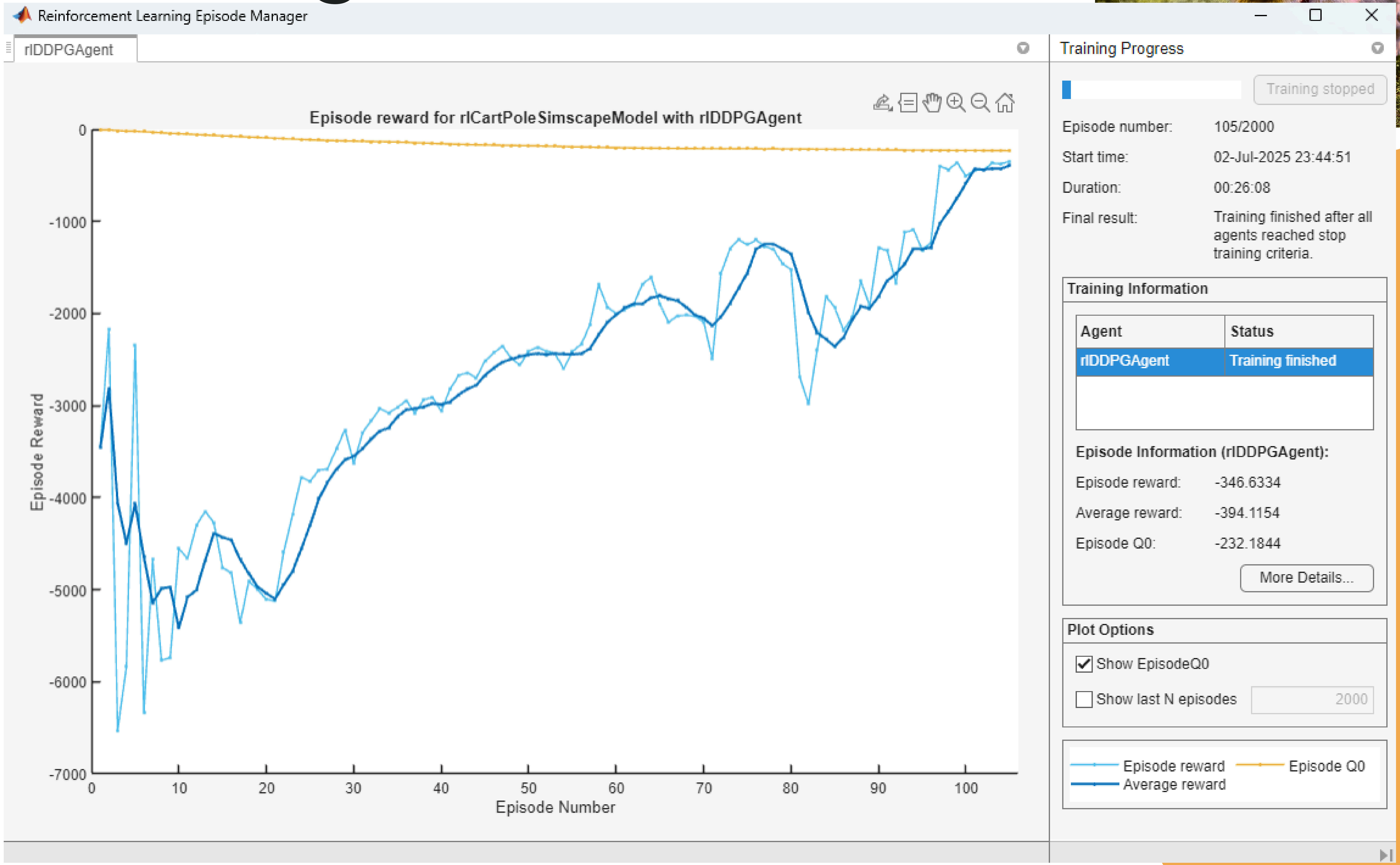


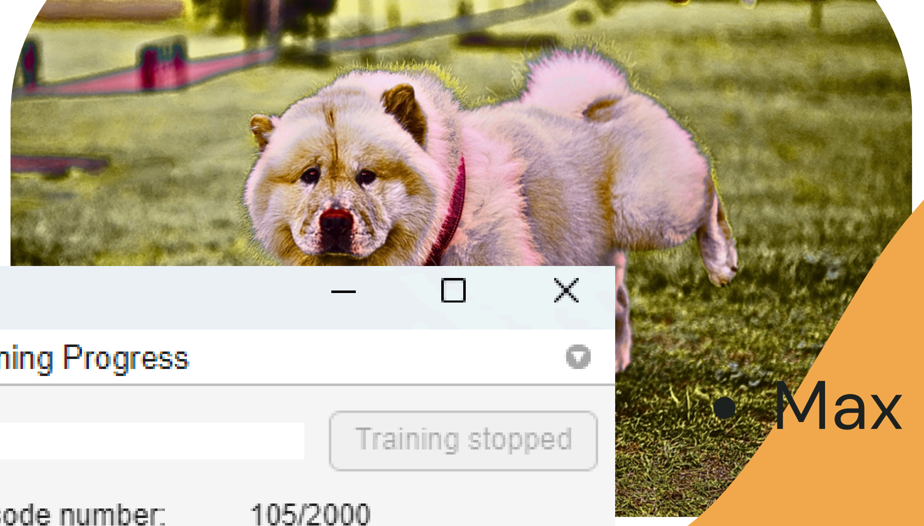


- **Actor Network:**
  - 5-128-200-2 neurons
  - LearnRate=1e-03
- **Critic Network:**



Training





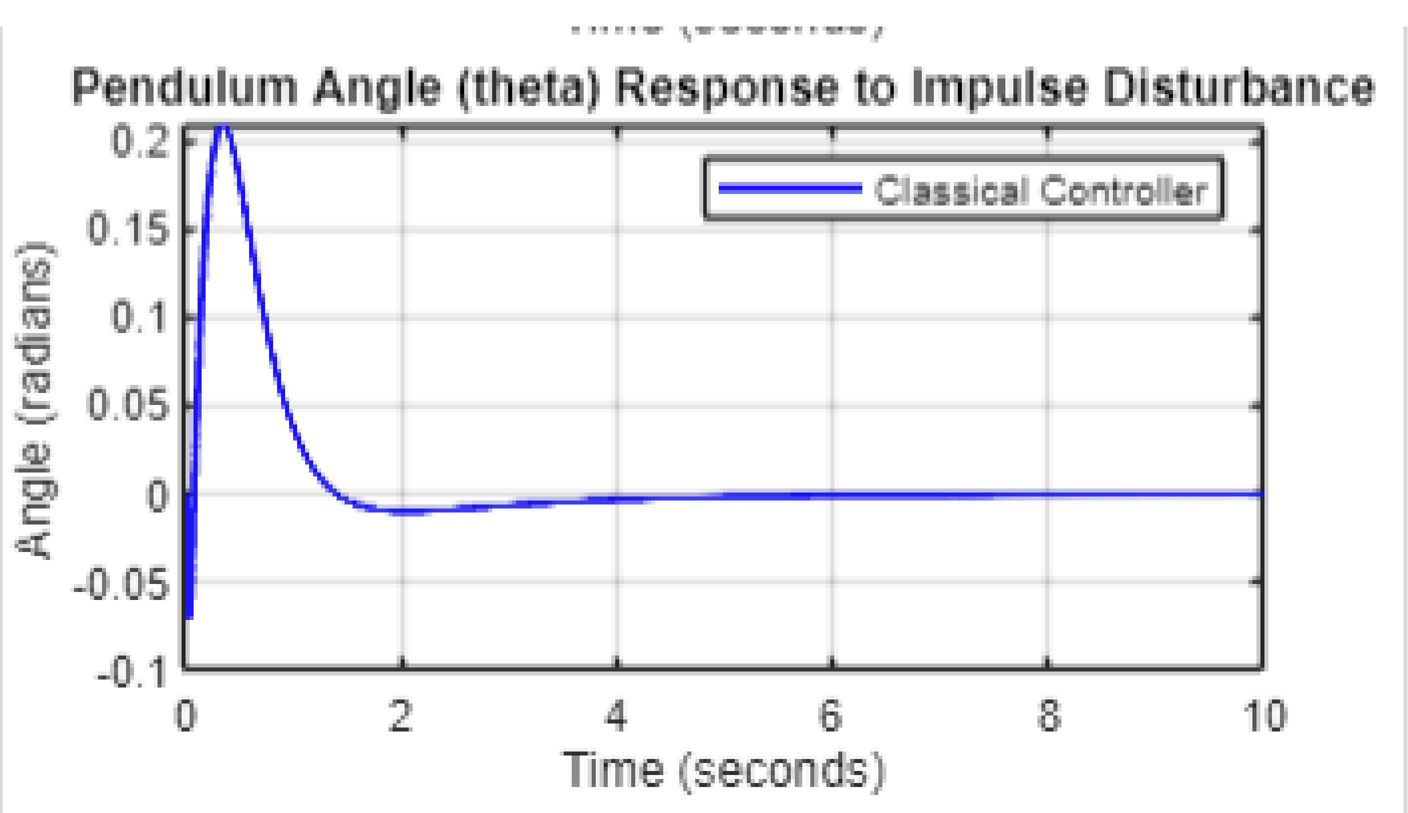
- Max Ep. : 2000
- Steps per Ep. : 1250
- Terminate if:
  - >3.5m deviation
- End training:
  - Avg. Reward: -400 over 5 epochs
- Training ended at epoch: 105

Evaluation

Inverted Pendulum

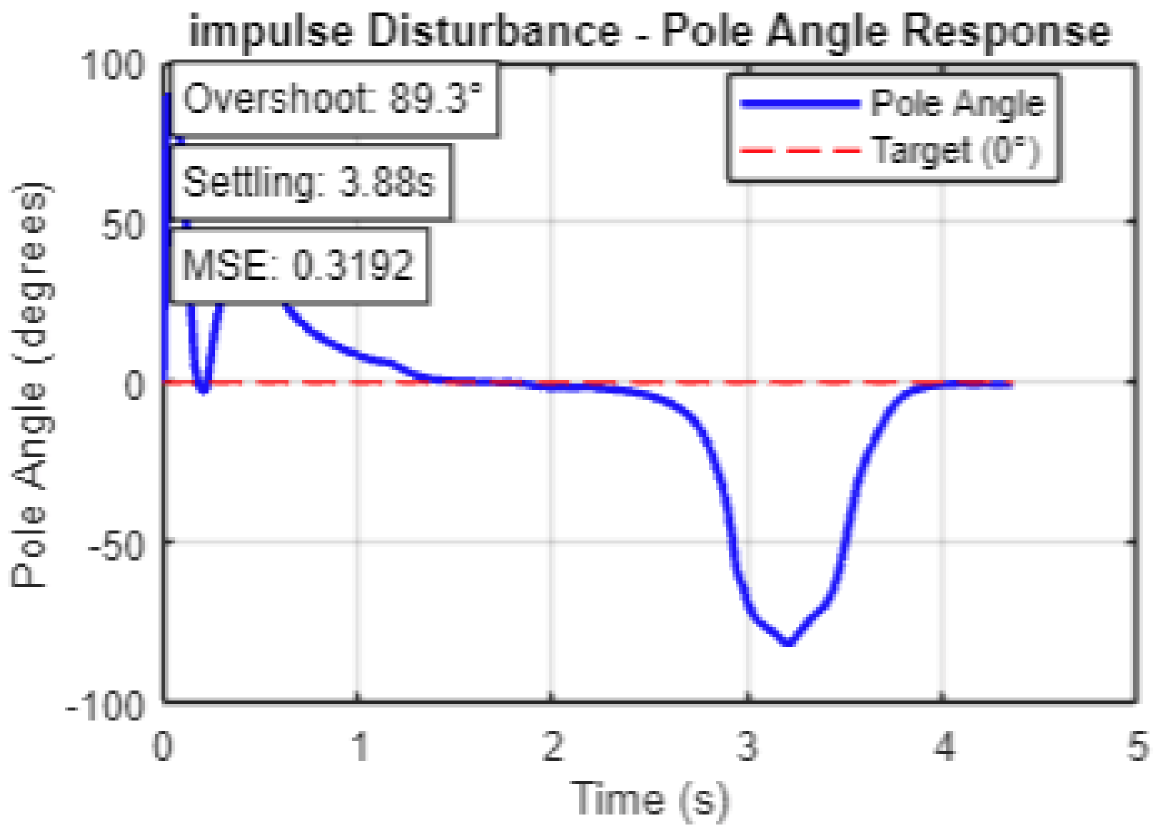
Classical PD

```
Overshoot (Peak Deviation)   | 0.2087
Rise Time (s)                | 0.2396
Settling Time (s)            | 3.5265
Mean Squared Error (MSE)     | 0.001925
```



Reinforcement Learning

```
Overshoot: 89.31%
Settling Time: 3.880 s
Rise Time: 0.000 s
MSE: 0.319172
```

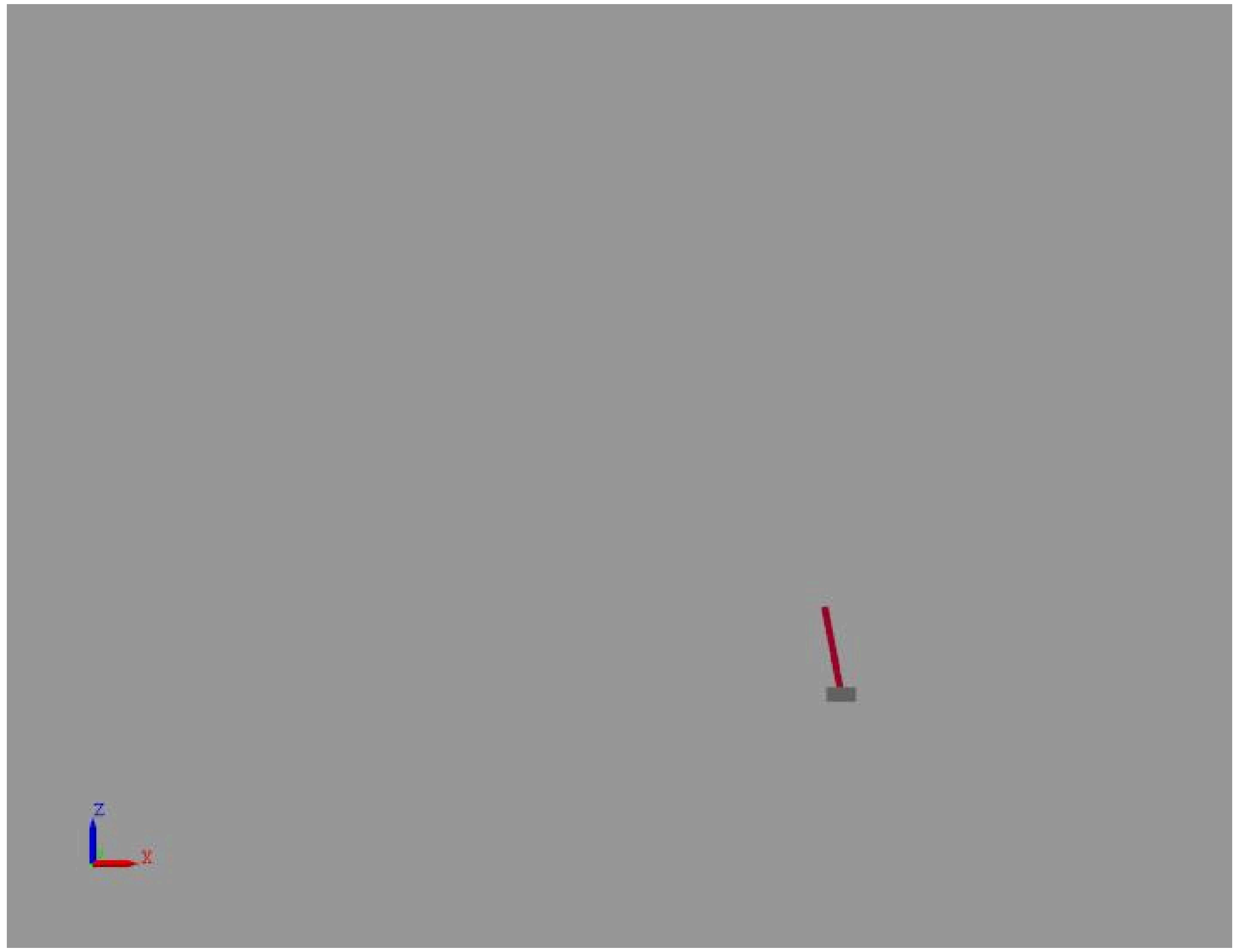


Demonstration

Inverted Pendulum

Classical PD

Reinforcement Learning



Thank you very much!

PRESENTED BY THIS GROUP