Imitation Learning of Neural Network Control Systems

Internship defense

Ahmad FARES

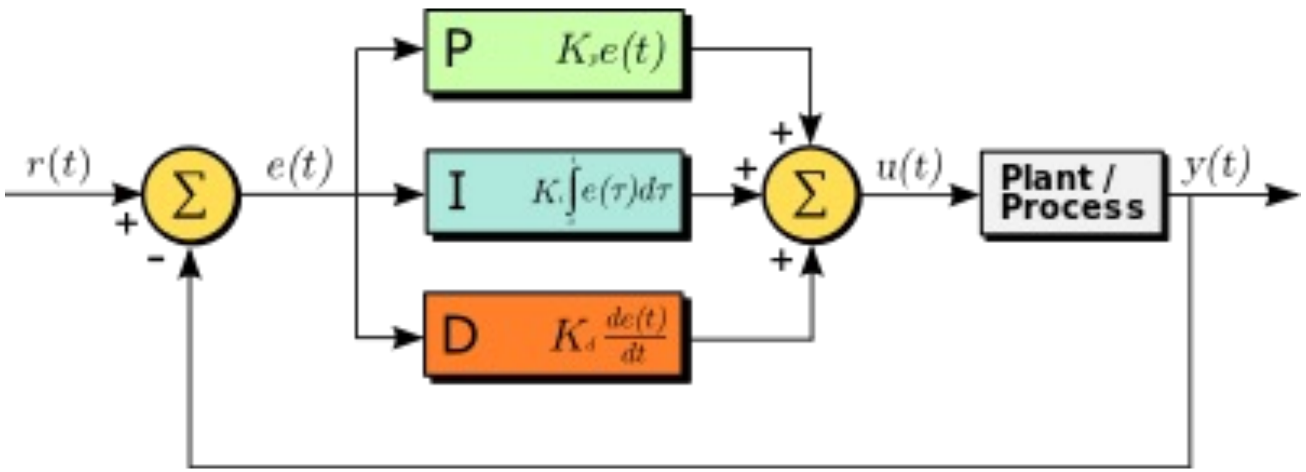
Supervisor :

- Professor Thao DANG

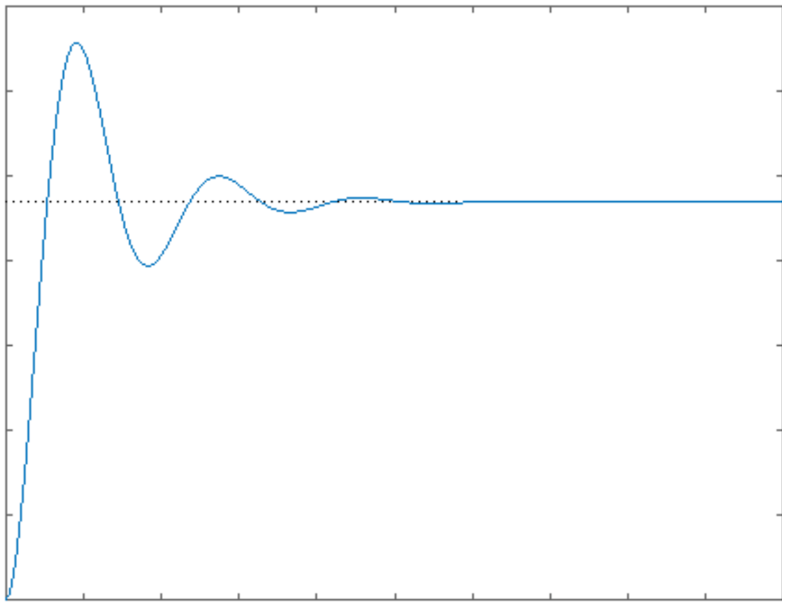
June 11, 2024

PID Controller

- Proportional Integral Derivative Controller
- Fundamental tools in industrial control systems
- provide solutions for maintaining desired levels of performance

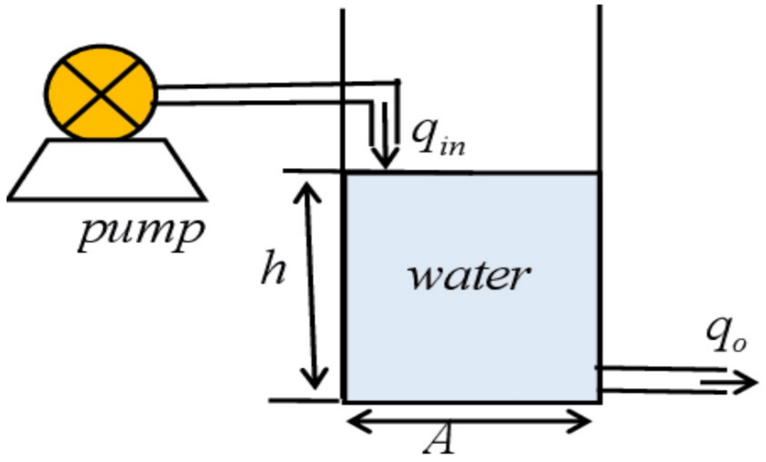


PID Loop Diagram

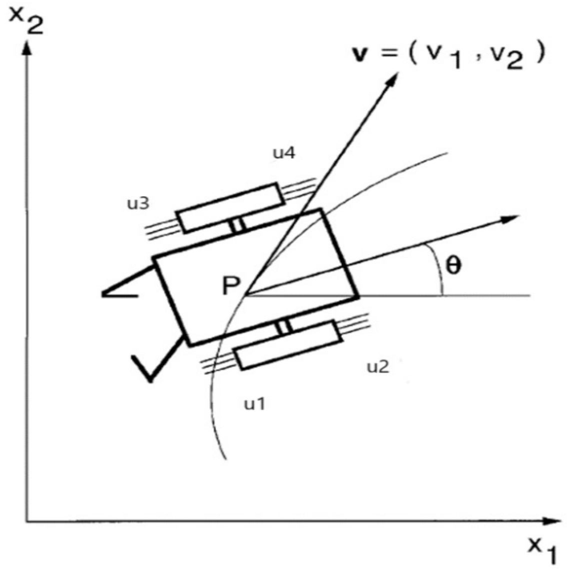


Stabilizing a System

PID Applications



Water-tank Model

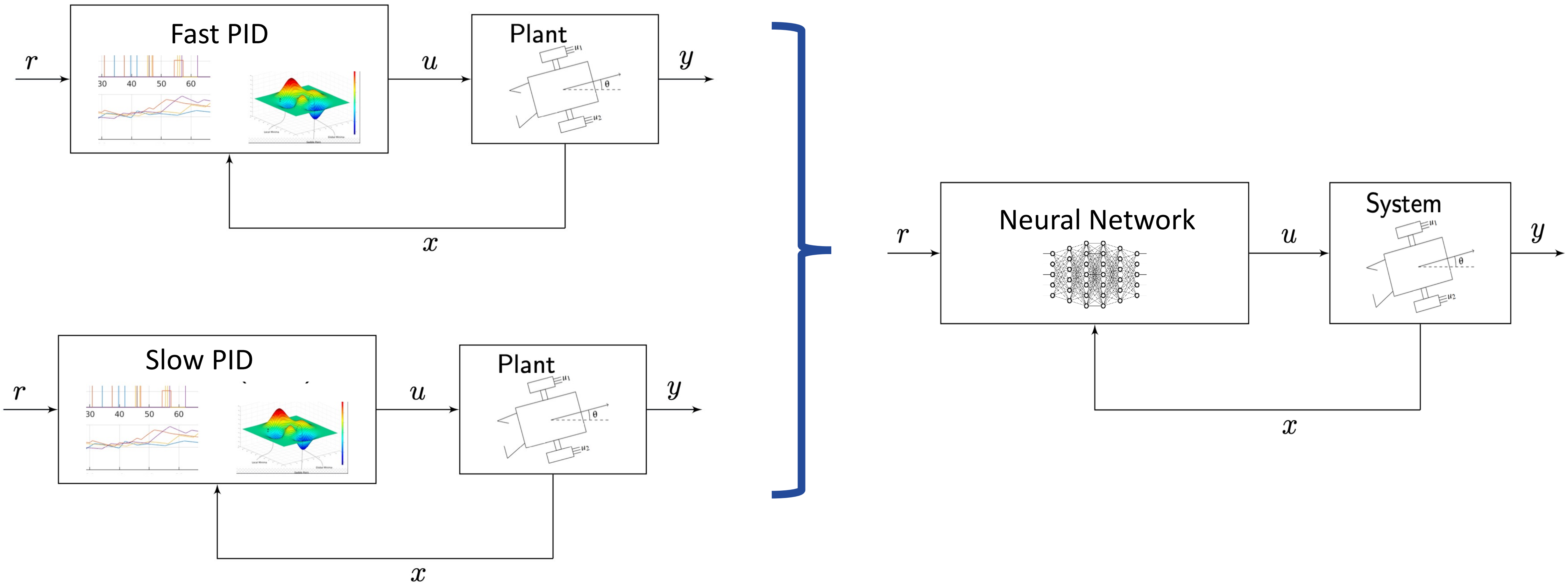


Flying Robot Model

PID Limitations

- Tied to their fixed operation domains and configurations → provide only limited performance.
- Expensive to deploy

Design a neural network controller capable of emulating the functionality of two distinct PID



Why Neural Networks?

- Exceptional function approximation capabilities
- Possess the ability to accurately approximate complex functions → well-suited for emulating the behavior of complex systems
- Deliver superior control performance and can be implemented on cost-effective, energy-efficient embedded platforms [Varshney et al., 2019]

→ NN Present an alternative to traditional PID Controllers and similar systems.

DC Motor Speed Control with Deep Learning [Cheon et al. ,2015]

- Utilized Deep Belief Network (DBN) algorithms to learn from PID controller data.
- Achieved superior motor speed control, demonstrating significant improvements over traditional PID methods.

Enhanced Balance Control for Segways [Ahmed and Saleh Alshandoli, 2020]

- Applied neural networks for managing the position and balance of Segways, traditionally controlled by PID systems.
- Neural networks provided more precise control of motion and balance, enhancing response accuracy.

Imitation Learning with Neural Networks for Flying Robots: [Dang et al., 2023]

- Developed a framework using dataset aggregation and imitation learning for training neural network-based controllers.
- Utilized Signal Temporal Logic(STL) for strategic data collection and falsification tests to validate neural network behavior.
- Tested on a model predictive controller (MPC) for a flying robot using 2-D navigation.
- Enhanced overshoot control within two iterations, though stability near target points remained challenging.

Initial Concepts:

Signal Temporal Logic (STL):

- Extends Linear Temporal Logic for real-time and real-valued systems.
- Used to specify and verify complex temporal properties of signals in continuous and hybrid systems.

Parametric Signal Temporal Logic (PSTL):

- Evolution: Incorporates parameters within STL formulas to allow for dynamic condition adjustments based on system performance.

Breach Tool:

- A MATLAB/C++ toolbox crucial for simulation-based analysis of dynamical and hybrid systems.
- Supports the evaluation of STL formulas, performs sensitivity analysis, and assists in parameter synthesis.

STL-Imitation-Algorithm:

Algorithm 1 Dataset aggregation-based training algorithm

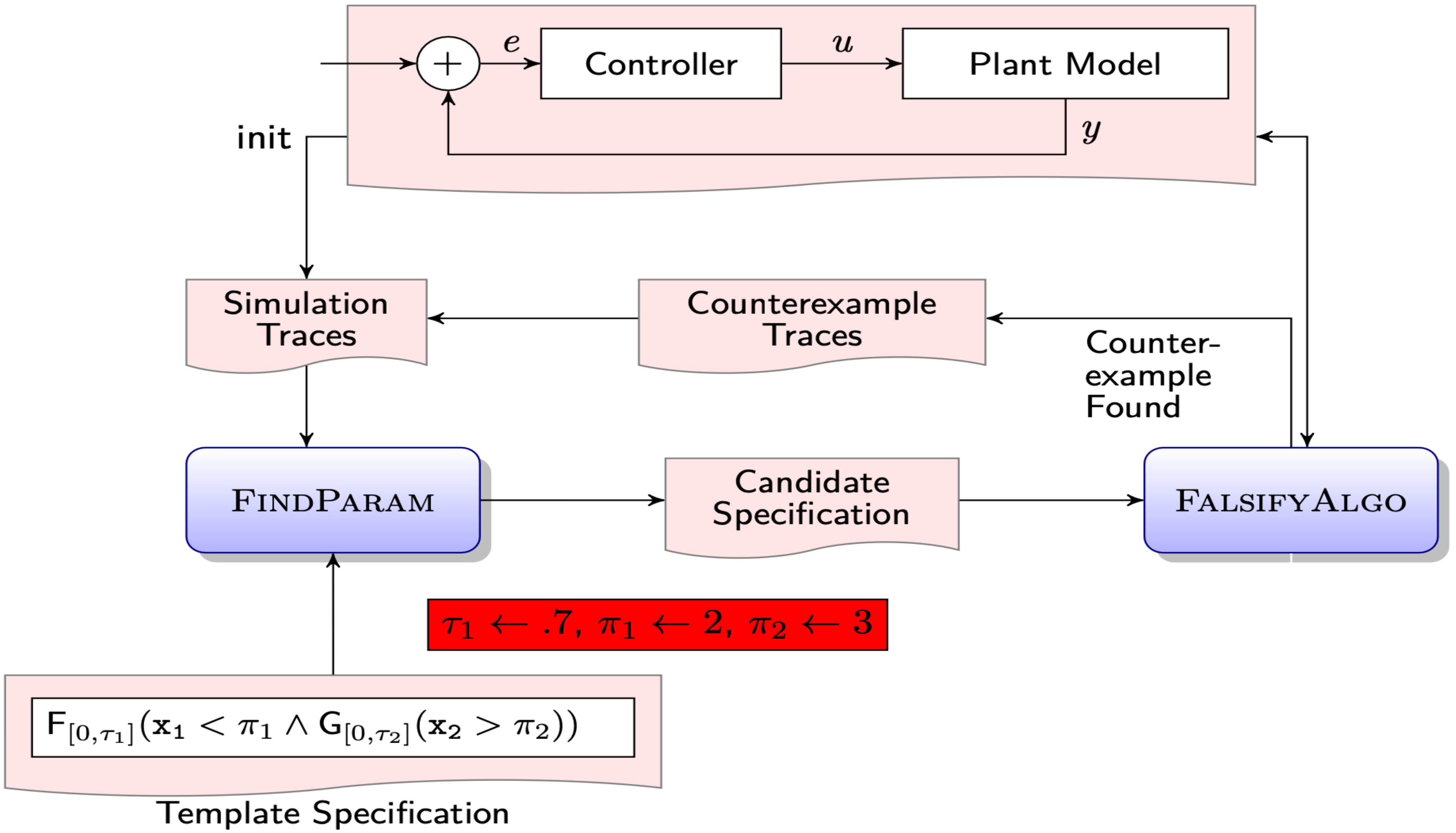
$\mathcal{N}_0 \leftarrow \emptyset, \mathcal{D}_0 \leftarrow \emptyset, k \leftarrow 1$
repeat
  $(\mathcal{D}_k, \text{Status}) \leftarrow \text{getNewData}(\mathcal{N}_{k-1}, \mathcal{D}_{k-1})$
  if $\mathcal{D}_k \neq \mathcal{D}_{k-1}$ **then**
    $\mathcal{N}_k \leftarrow \text{Train}(\mathcal{D}_k)$
    $k \leftarrow k + 1$
  end if
until $\mathcal{D}_k = \mathcal{D}_{k-1}$ or $k > k_{\text{max}}$
return $\mathcal{N}_k$, Status

Algorithm 3 Augmenting the training data from bad traces

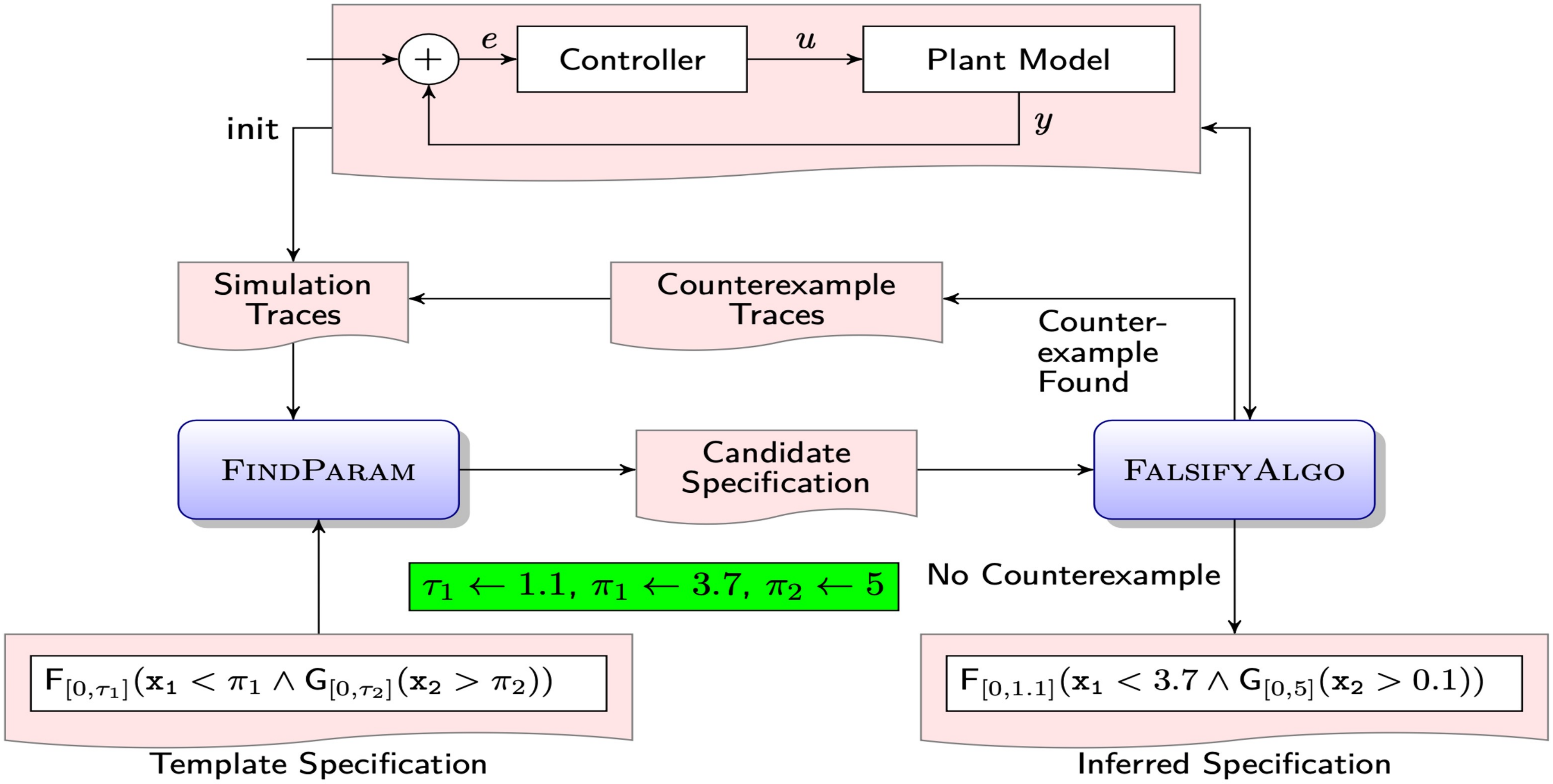
procedure FIXANDMERGE($\mathcal{D}$, CexTraces)
  NeighSamples $\leftarrow$ gridFilter(CexTraces) Cover($\mathcal{D}$)
  if NeighSamples $= \emptyset$ **then**
    status $\leftarrow$ "Counter-examples do not add new data."
  else
    FixedTraces $\leftarrow$ simNominal(NeighSamples)
    $\mathcal{D}_{\text{new}} \leftarrow \text{gridFilter}(\mathcal{D} \cup \text{FixedTraces})$
    status $\leftarrow$ "New data available for training."
  end if
  return $\mathcal{D}_{\text{new}}$, status
end procedure

Algorithm 2 New data acquisition procedure

procedure GETNEWDATA(D, N)
  if $D = \emptyset$ **then**
    InitSamples $\leftarrow$ GETINITSAMPLES
    InitTraces $\leftarrow$ SIMNOMINAL(InitSamples)
    $D_{\text{new}} \leftarrow$ GRIDFILTER(InitTraces)
    status $\leftarrow$ "New data available for training."
  else
    CexTraces $\leftarrow$ FALSIFY(N)
    if CexTraces $\neq \emptyset$ **then**
      (D_{new}, **status**) $\leftarrow$ FIXANDMERGE(D, CexTraces)
    else
      $D_{\text{new}} \leftarrow D$
      status $\leftarrow$ "No counter-example found."
    end if
  end if
  return (D_{new}, **status**)
end procedure



e
Controller
u
Plant Model
y
init
Simulation Traces
Counterexample Traces
Counter-example Found
FINDPARAM
Candidate Specification
FALSIFYALGO
$\tau_1 \leftarrow .7,\ \pi_1 \leftarrow 2,\ \pi_2 \leftarrow 3$
$\mathsf{F}_{[0,\tau_1]}(\mathbf{x}_1 < \pi_1 \wedge \mathsf{G}_{[0,\tau_2]}(\mathbf{x}_2 > \pi_2))$
Template Specification


e
Controller
u
Plant Model
y
init
Simulation Traces
Counterexample Traces
Counter-example Found
FINDPARAM
Candidate Specification
FALSIFYALGO
$\tau_1 \leftarrow 1.1,\ \pi_1 \leftarrow 3.7,\ \pi_2 \leftarrow 5$
No Counterexample
$F_{[0,\tau_1]}(x_1 < \pi_1 \wedge G_{[0,\tau_2]}(x_2 > \pi_2))$
Template Specification
$F_{[0,1.1]}(x_1 < 3.7 \wedge G_{[0,5]}(x_2 > 0.1))$
Inferred Specification

Computation of Control Signal u:

The state vector y is defined as:

$$y = [H, H_p, H_{pp}, \text{ref}, \text{ref}_p, \text{ref}_{pp}, u_p]$$

Where

- $y(1) = H$: Current height of the water tank
- $y(2) = H_p$: First derivative of the height (velocity)
- $y(3) = H_{pp}$: Second derivative of the height (acceleration)
- $y(4) = \text{ref}$: Reference height
- $y(5) = \text{ref}_p$: First derivative of the reference height
- $y(6) = \text{ref}_{pp}$: Second derivative of the reference height
- $y(7) = u_p$: Previous control input

The errors are defined as:

$$e = y(1) - y(4)$$

$$ep = y(2) - y(5)$$

$$epp = y(3) - y(6)$$

The control signal u is then computed as:

$$u = y(7) + a \cdot e + b \cdot ep + c \cdot epp$$

where the coefficients a, b, and c are given by:

$$a = Kp + \frac{Ki \cdot Ts}{2} + \frac{Kd}{Ts}$$

$$b = -Kp + \frac{Ki \cdot Ts}{2} - \frac{2 \cdot Kd}{Ts}$$

$$c = \frac{Kd}{Ts}$$

→ Kp, Ki, Kd are adjusted to change the behavior of the controller

Water-tank Model: Imitating behavior of 1 controller

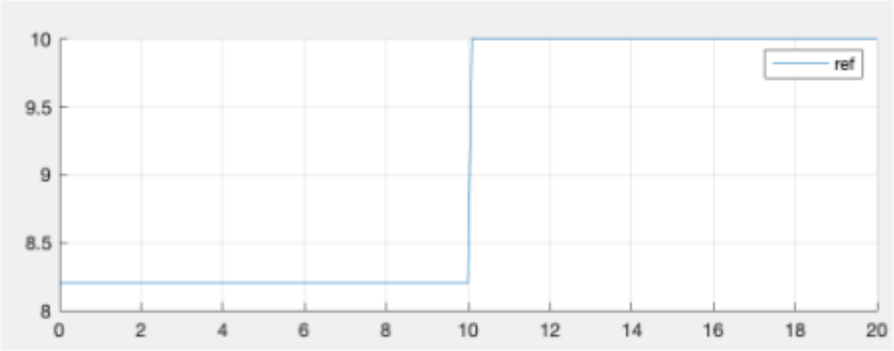
Algorithm 4 Simulation of Breach Water Tank

```
procedure SIM_BREACH_WATERTANK(control_fn, t, p)
    Initialize parameters and initial state
    for each time step k do
        Get current state
        Compute control input using the provided function
        Update plant state using ODE solver
        Store new state values
    end for
    Finalize the last control input
end procedure
```

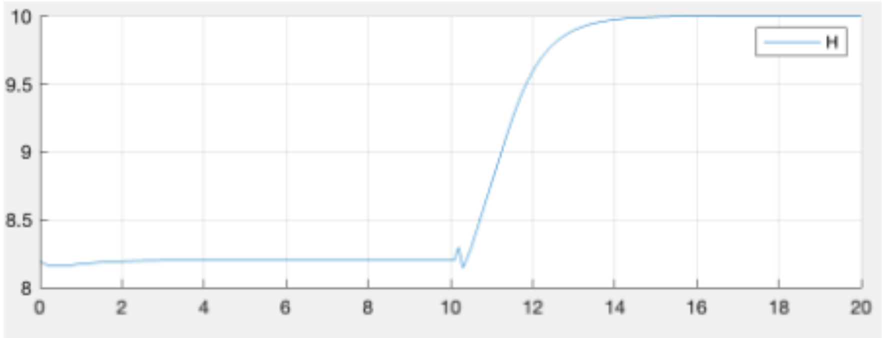
Evaluation Criteria:

- Monitoring the water level "H" in relation to a predefined step function as a reference signal.
- Controller is deemed effective if "H" aligns closely with the reference signal.

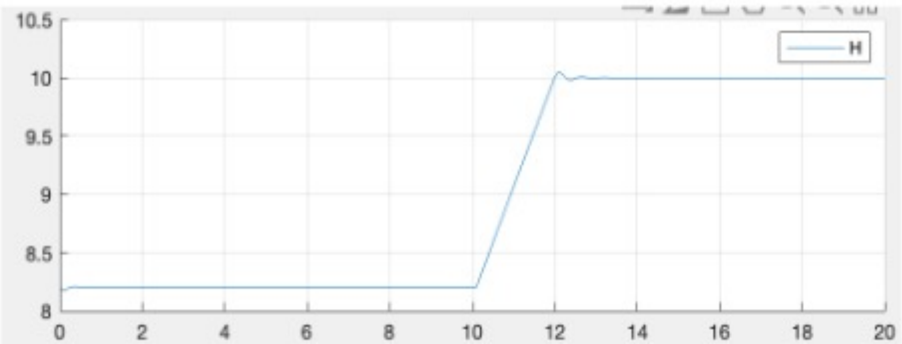
Results of simulation:



Reference Step Function



Slow Controller



Fast Controller

Proposed Approach:

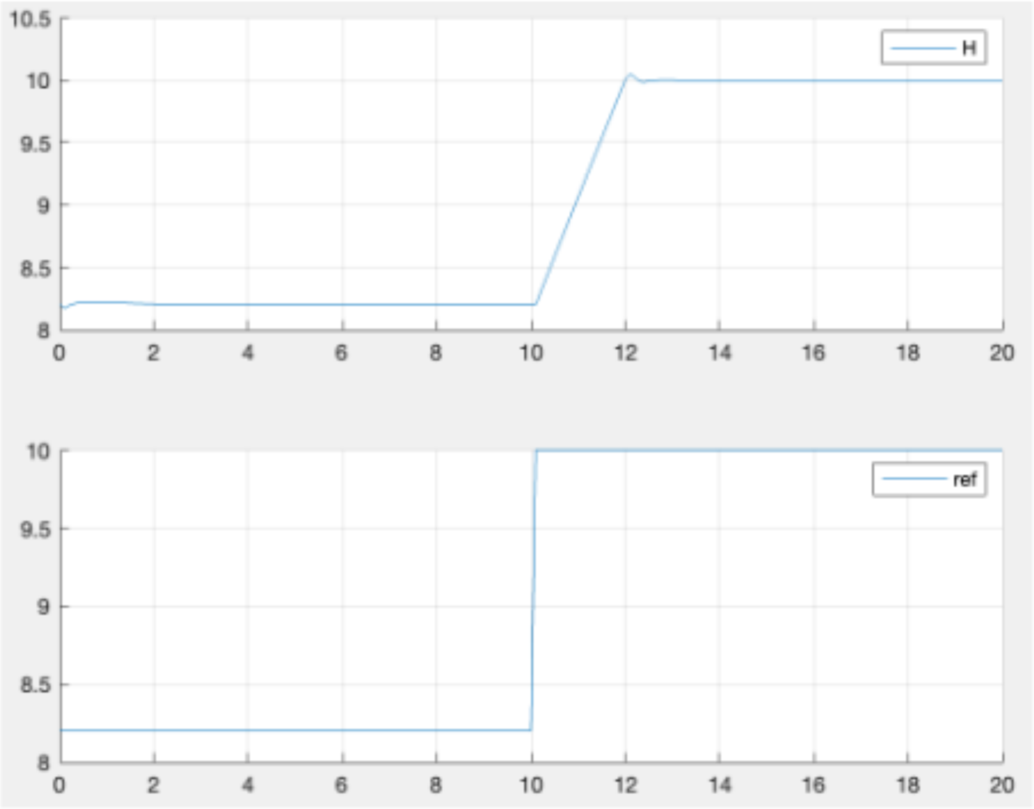
Algorithm 4 Simulation of Breach Water Tank

```
procedure SIM_BREACH_WATERTANK(control_fn, t, p)
    Initialize parameters and initial state
    for each time step k do
        Get current state
        Compute control input using the provided function
        Update plant state using ODE solver
        Store new state values
    end for
    Finalize the last control input
end procedure
```

Algorithm 5 Simulation of Breach Water Tank with Dual Controllers

```
procedure SIM_BREACH_WATERTANK(control_fn1, control_fn2, t, p)
    Initialize parameters and initial state
    for each time step k do
        Get current state
        Compute control inputs using both controllers
        for each controller 1 and 2 do
            Apply control input
            Update plant state using ODE solver
            Store new state values temporarily
            Evaluate performance
            if new state is better then
                Update best distance
                Update best state
            end if
        end for
        Update the state with the best result from this step
    end for
    The corresponding control input is stored in the data to be used for training.
end procedure
```

Initial Experimentation:



Analysis:

- The combined controller exhibited behavior similar to the fast controller, effectively tracking the reference height with high precision.
- Combined controller operates correctly and meets expected performance criteria.

→ **Further experimentation is in progress**

Achievements:

- Successfully integrated two PID controllers into a single neural network-based system.

Implications:

- The study illustrates the potential of superior neural networks that enhance traditional control systems.

Future Work:

- Further experimentation to refine the neural network training process.
- Extending into more than two controllers.