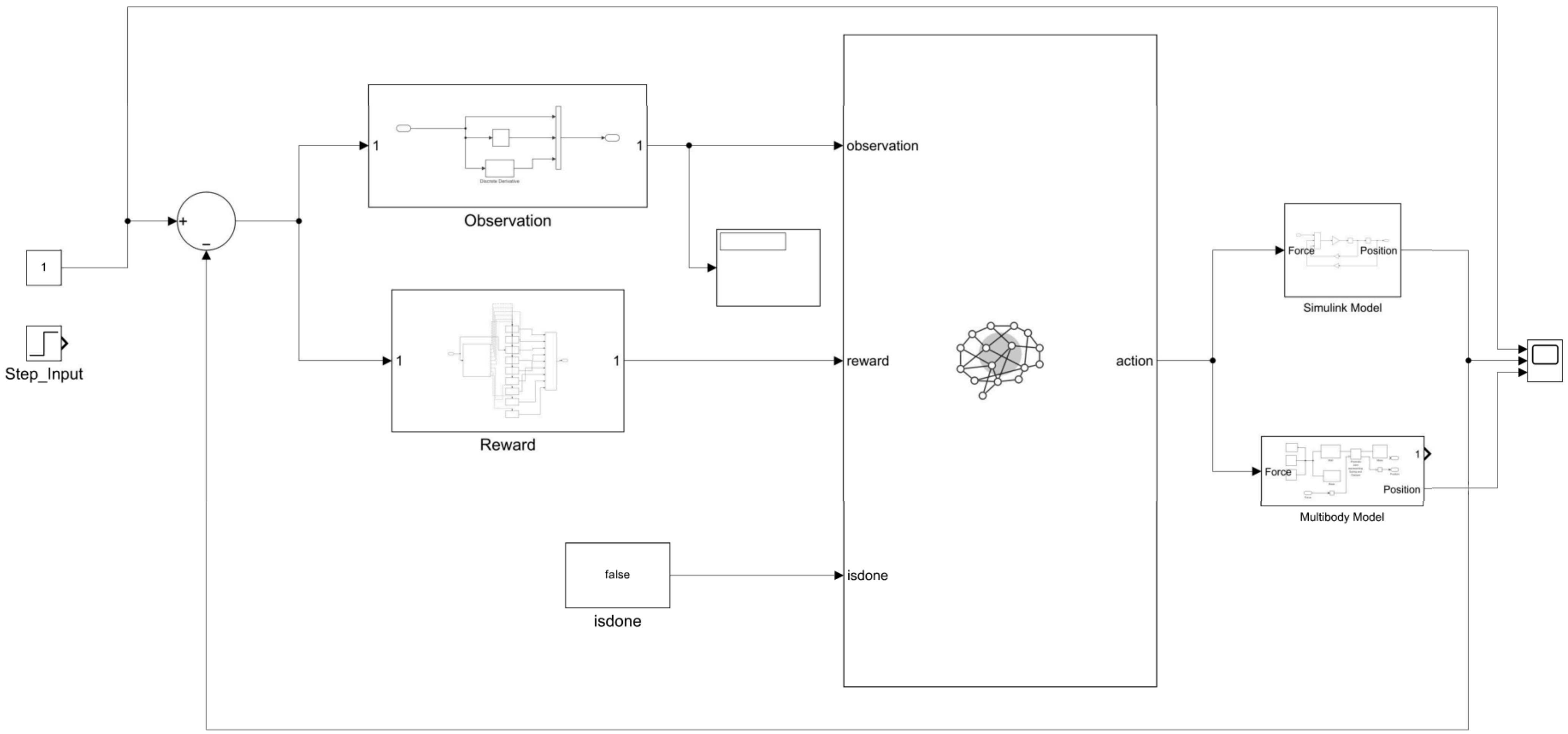
RL Agent (As PID) For Mass-spring-damper-system



This document explains how a Reinforcement Learning (RL) agent can be used as a PID controller to control a mass-spring-damper system. Traditional PID controllers often require manual tuning, especially for nonlinear systems. In contrast, RL-based methods can automatically adapt and optimize control actions for such systems.

The project aims to:

1. Show how RL can control a physical system.
2. Highlight the advantages of RL, like adaptability and efficiency, over traditional methods.

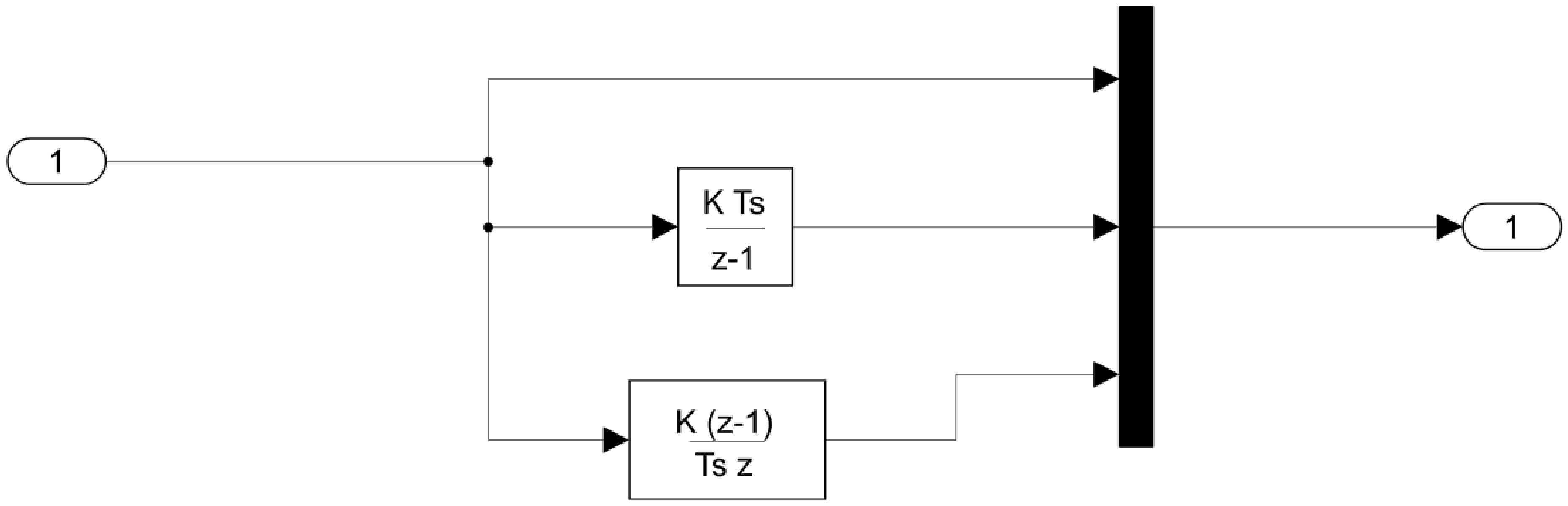
The key parts of the project are:

- **Observation Block:** Processes the system's state for the RL agent.
- **Reward Block:** Defines the rules for evaluating the agent's performance.
- **Simulink Model:** Simulates the system's dynamics.
- **Simscape Model:** Provides a physical model of the system with mass, spring, and damper components.

This project connects theoretical concepts to practical implementation, demonstrating how RL can enhance control strategies for dynamic systems.

The RL agent acts as the controller by analyzing system inputs and generating control actions. Unlike traditional controllers, it doesn’t rely on manual tuning but learns and adapts automatically.

Observation Block



The **Observation Block** collects and processes key information about the system, such as:

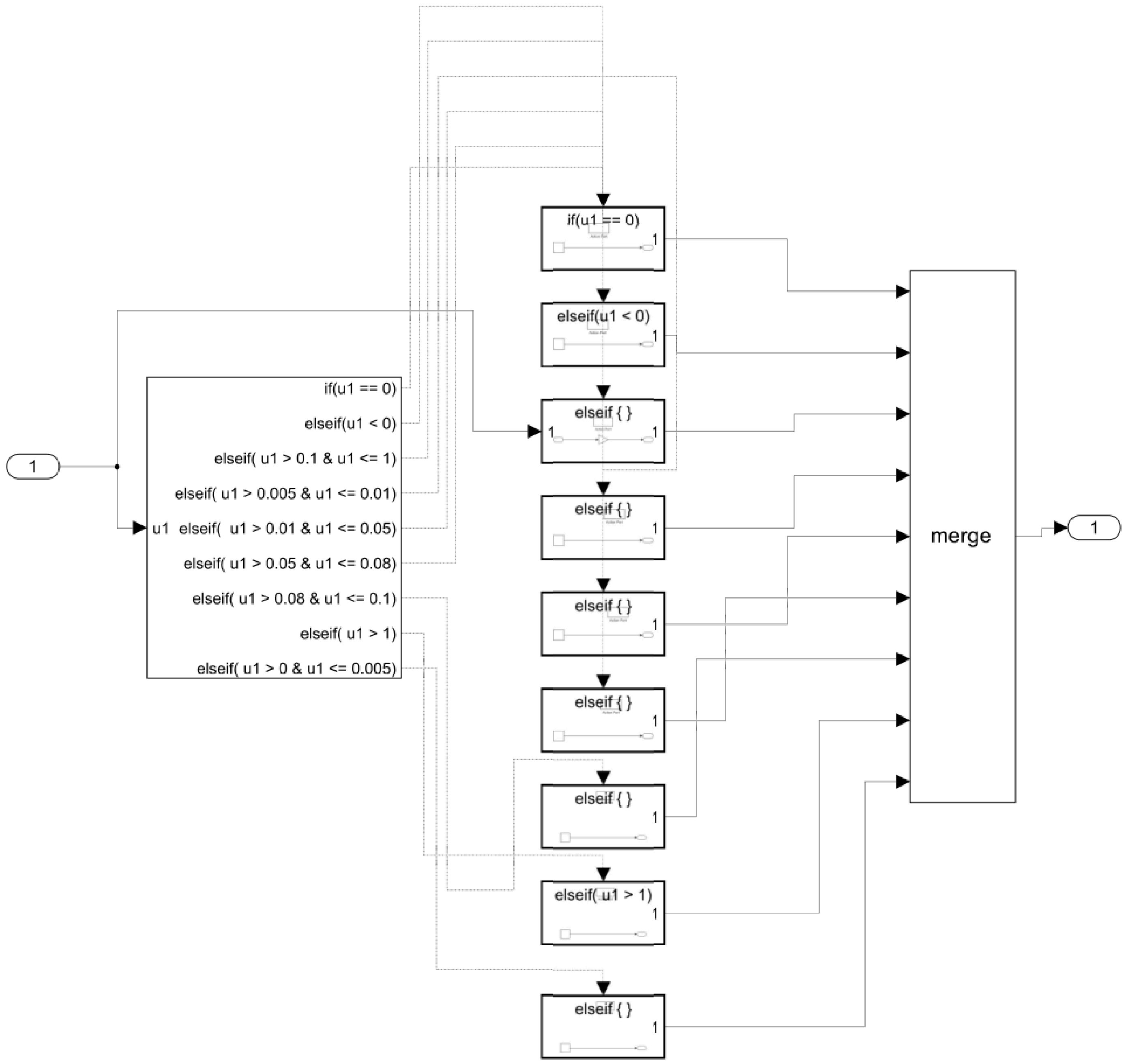
- The **position** and **velocity** of the mass.

This information is sent to the RL agent as an observation vector. The observation block:

- Filters and standardizes data to ensure accuracy.
- Provides real-time feedback for decision-making.

The RL agent uses this processed information to calculate actions that reduce errors and stabilize the system.

Reward Block


1
u1
if(u1 == 0)
elseif(u1 < 0)
elseif(u1 > 0.1 & u1 <= 1)
elseif(u1 > 0.005 & u1 <= 0.01)
elseif(u1 > 0.01 & u1 <= 0.05)
elseif(u1 > 0.05 & u1 <= 0.08)
elseif(u1 > 0.08 & u1 <= 0.1)
elseif(u1 > 1)
elseif(u1 > 0 & u1 <= 0.005)
if(u1 == 0)
elseif(u1 < 0)
elseif { }
elseif { }
elseif { }
elseif { }
elseif { }
elseif(u1 > 1)
elseif { }
merge
1

Reward Function Description for TD3 RL Agent Training

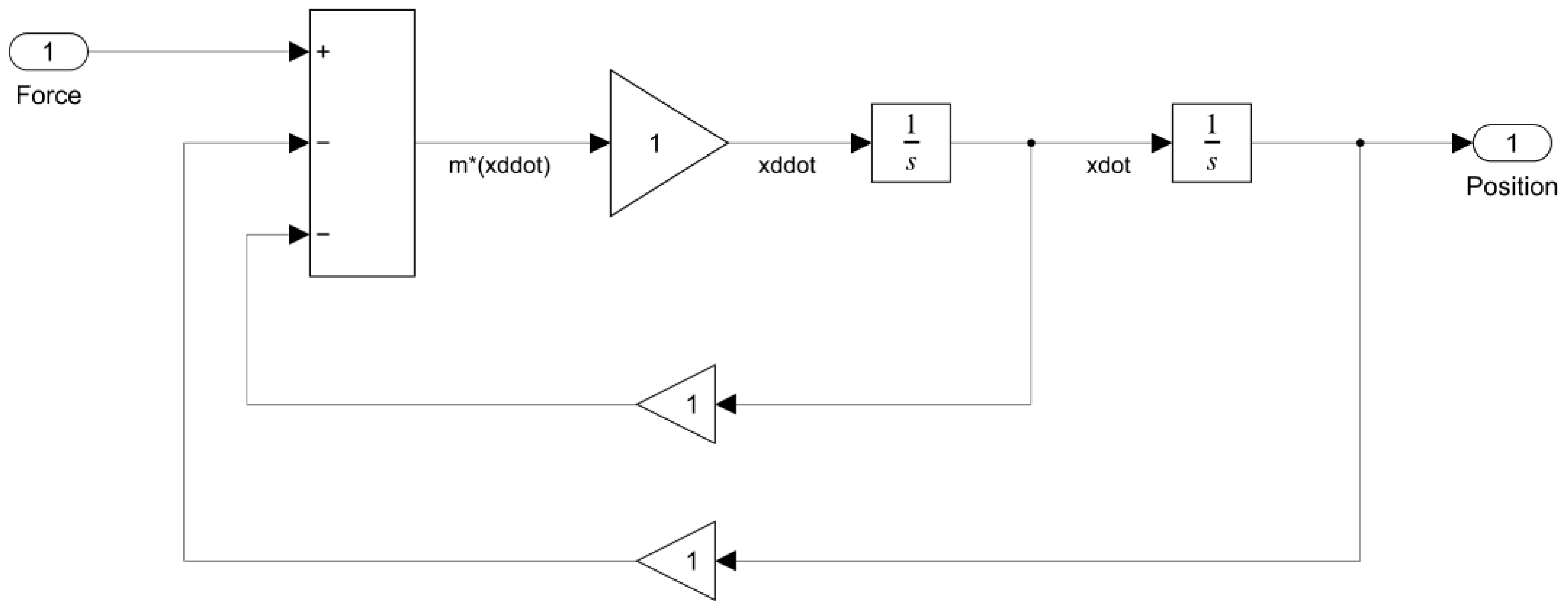
In the TD3 reinforcement learning (RL) agent, the reward function is designed to guide the agent towards effective PID tuning while penalizing undesired behaviors. Below is a detailed description of the components of the reward block:

1. **Penalty for Overshooting**: A large negative penalty of `-2000` is applied if the agent's output overshoots from the given input, discouraging large deviations from the target.
2. **Reward for Staying Within Target**: A small positive reward of `+3` is awarded if the agent maintains its output near the given input for a certain period of time, encouraging stability in the system.
3. **Penalty for Staying in the 0.1 to 1 Region**: A penalty is applied if the agent's output remains in the region between `0.1` and `1` for too long, as this range is considered suboptimal for control performance. The exact penalty increases based on how far the agent's output is from the desired region.
4. **Reward for Specific Output Ranges**:
   - A reward of **+2** is given when the agent's output is between `0` and `0.005`, promoting fine-tuned control in this range.
   - A reward of **+1** is awarded when the agent's output falls between `0.005` and `0.01`, encouraging the agent to focus on this narrow range.
   - A reward of **+0.4** is given when the agent's output is between `0.01` and `0.05`, incentivizing moderate stability.
   - A reward of **+0.2** is applied for outputs between `0.05` and `0.08`, encouraging fine adjustments within this range.
   - A small reward of **+0.1** is awarded when the agent's output is between `0.08` and `1`, promoting overall system performance.
5. **Penalty for Deviation Beyond Target**: A severe penalty of `-100` is given if the agent's output moves far from the input target, i.e., when the value exceeds `1` between target and system output. This discourages the agent from moving too far from the target and promotes efficient control within the acceptable range.

Overall Reward Strategy

The reward function is structured to balance stability and precision in the agent's actions. The agent receives higher rewards for staying close to the target input range and adjusting with minimal overshoot. Meanwhile, large deviations or extended stays outside the target range result in penalties, guiding the agent towards optimal performance over time.

Simulink Model



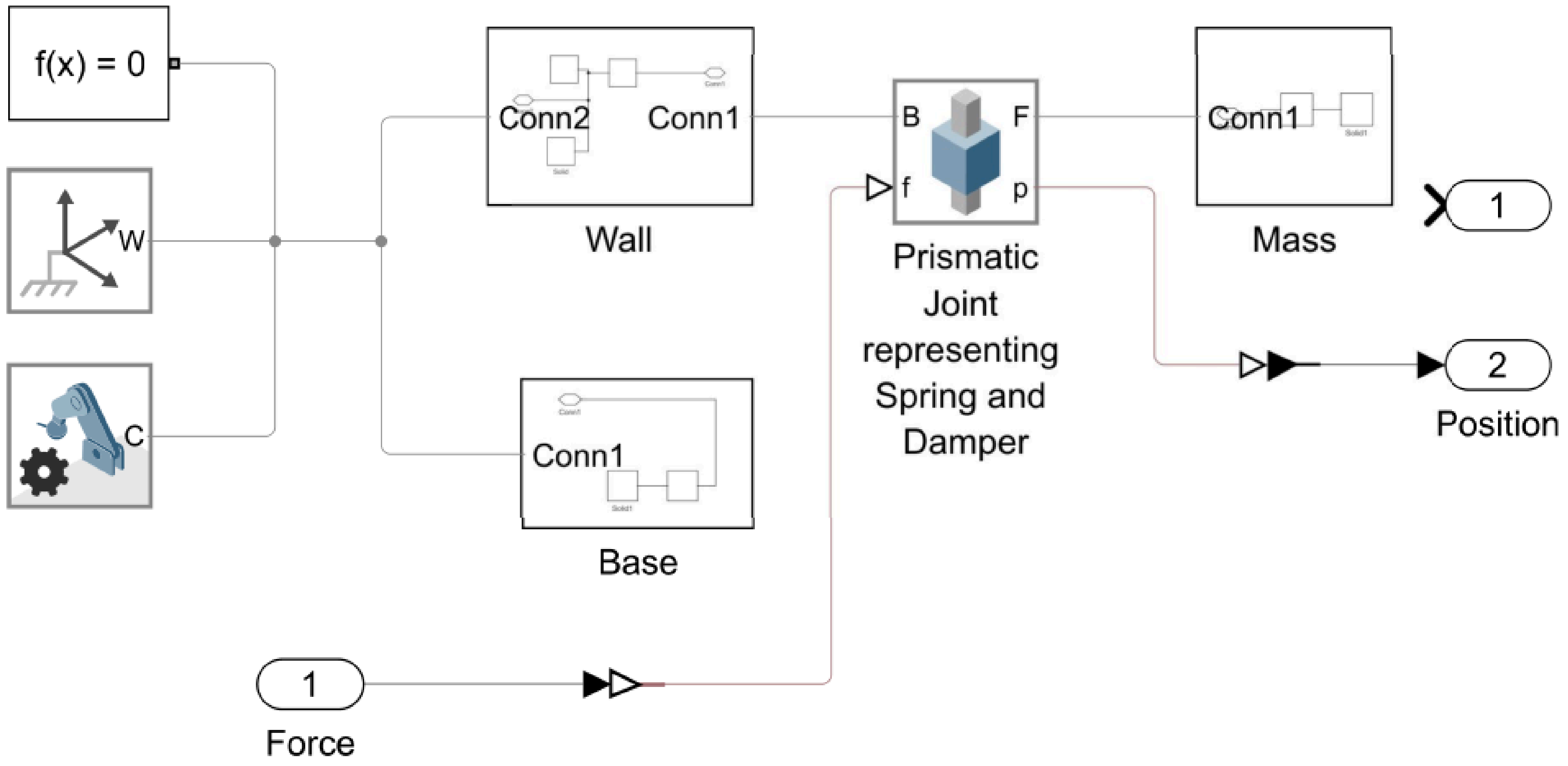
The **Simulink Model** provides a virtual environment to simulate the behavior of the mass-spring-damper system. It integrates the RL agent with the system's dynamics, enabling the testing of control strategies.

Features of the Simulink model:

- **System Simulation:** Simulates how the mass-spring-damper system behaves.
- **RL Integration:** Allows the RL agent to interact with the system using observation and reward blocks.
- **Parameter Adjustment:** Enables tuning of system properties like spring stiffness or damping.

The Simulink model acts as a testing ground to ensure that the RL agent can control the system effectively before applying it in real-world scenarios.

Simscape Model



The **Simscape Model** provides a physical representation of the mass-spring-damper system. This model includes:

1. **Wall and Base:** Represent fixed boundaries.
2. **Prismatic Joint:** Simulates linear motion with spring and damper properties.
3. **Mass:** Represents the movable body.
4. **Position Sensor:** Measures the mass's displacement.

The position sensor connects to the simulation through a **PS-Simulink Converter**, allowing data to flow between the Simscape and Simulink models. This connection enables the RL agent to control the physical model effectively.

The Simscape model helps visualize the system in a more realistic way, showing how the control strategies work in a physical setup.