APPLICATION OF A COMBINED RL AND PID ALGORITHM IN A MASK CONTROLLER FOR APNEA

PIME-TD3 APPLYED TO CPAP

Victor Gabriel Tenório Oliveira
Departamento da Computação, UFRPE, Brasil
victor6g0@gmail.com

Pablo Sampaio
Departamento da Computação, UFRPE, Brasil
pablo.sampaio@ufrpe.br

Hatus Vianna Wanderley (Coautor)
Massachusetts General Hospital, Harvard Medical School, USA
hwanderley@mgh.harvard.edu

28 de março de 2025

ABSTRACT

Este estudo investiga a integração do algoritmo *Twin Delayed Deep Deterministic Policy Gradient* (TD3) com o controle Proporcional-Integral-Derivativo (PID) para gerenciar sistemas não lineares, com foco no controle de Pressão Positiva Contínua nas Vias Aéreas (CPAP), buscando mostrar a viabilidade dessa integração. Um modelo matemático foi desenvolvido para representar a dinâmica do CPAP, e o algoritmo PIME-TD3 foi implementado e testado. Experimentos foram conduzidos nos ambientes *Cascade Water Tank* e CPAP, utilizando *Optuna* para otimizar hiperparâmetros. No ambiente *Cascade Water Tank*, o PIME-TD3 apresentou dificuldades em replicar os resultados da literatura, exigindo um ajuste exaustivo de hiperparâmetros e estrutura da rede. Já no ambiente CPAP, os resultados indicaram que as limitações da simulação impactaram o aprendizado, impedindo um controle eficaz da pressão. O estudo destaca a importância de modelos matemáticos realistas para treinar agentes de aprendizado por reforço e a necessidade de validação em hardware real para aplicação clínica. Além disso, sugere direções para pesquisas futuras, como a melhoria da modelagem do CPAP, experimentação com diferentes arquiteturas de rede e validação do controlador em dispositivos reais.

Keywords Aprendizado por Reforço Profundo · Equipamento Pressão Positiva Contínua nas Vias Aéreas · Controlador PID · Teoria do Controle · Apneia

1 Introduction

A engenharia de controle define controladores que automatizam tarefas encontradas em biorreatores [1], sistemas da indústria do petróleo [2], máscaras CPAP e etc. Em particular, o equipamento CPAP, utilizado no tratamento da apneia do sono, é o foco deste trabalho, onde aplicamos uma combinação de aprendizado por reforço com um controlador PID para solucionar os desafios desse ambiente dinâmico e não linear.

Nesse contexto, os ambientes de aprendizado encapsulam modelos matemáticos baseados em sistemas reais não lineares e os controladores que atuam nesse ambiente buscam manter alguma variável numérica o mais próxima possível de um valor objetivo que varia durante o tempo. No CPAP, essa variável corresponde à pressão do ar fornecida ao paciente; nos sistemas da indústria do petróleo, está associada à taxa de extração; e, em biorreatores, refere-se à densidade populacional de microrganismos.

Síndrome de apneia obstrutiva do sono (SAOS) é caracterizada por episódios repetidos de apneia e hipopneia durante o sono. Ambos os episódios se referem à interrupção do fluxo de ar por 10 ou mais segundos. Enquanto a apneia é caracterizada pela interrupção total ou parcial, a hipopneia é apenas caracterizada pela interrupção parcial [3].

A SAOS pode trazer vários efeitos negativos na saúde, como doenças cardiovasculares, incluindo hipertensão, enfarte do miocárdio e acidente vascular cerebral [3]. Além dessas doenças, pode ocorrer hipersonolência diurna [3] e atraso no desenvolvimento de crianças [4]. Considerando as alterações cardiovasculares e a necessidade de dormir bem para o bom funcionamento do corpo, tratar esse problema é de grande relevância para a sociedade.

O tratamento dessa síndrome pode se dar de diversas formas, sendo as modalidades de tratamento não cirúrgicas amplamente exploradas na literatura. As abordagens não cirúrgicas incluem modificação de comportamento, terapia medicamentosa e utilização de um dispositivo CPAP (Pressão Positiva Contínua nas Vias Aéreas do inglês *Continuous Positive Airway Pressure*) [3]. Também há a possibilidade de tratamento cirúrgico [3]. Dentre essas abordagens, o dispositivo mecânico nasal CPAP é considerado a principal e mais importante abordagem de tratamento [3, 5]. Esse dispositivo é projetado para manter as vias aéreas abertas durante o sono, proporcionando um fluxo constante de ar.

Para operar eficientemente, os equipamentos CPAP dependem de um ventilador, que atua como controlador [5] e pode ser treinado em uma simulação do ambiente ou no mundo real. A simulação é preferível, pois evita riscos reais, consome menos tempo, provê dados em abundância para algoritmos de aprendizado por reforço e permite o uso de técnicas existentes e já estudadas que superam as discrepâncias entre o mundo real e a simulação. O controlador é um sistema que opera o equipamento de CPAP com o objetivo de atingir uma pressão desejada, chamada de *setpoint*, que muda durante o tempo. Para operar corretamente, é necessário otimizar o desempenho do controlador com técnicas da engenharia.

A simulação do sistema real do CPAP é um modelo matemático não linear e difícil de modelar perfeitamente. Além do ventilador, esse modelo pode considerar as características do paciente, como resistência da passagem de ar e complacência do sistema respiratório, que determinam como a pressão gerada se comporta internamente. Esse comportamento não linear impõe desafios de modelagem e controle adicionais para o engenheiro, tornando o desafio mais complexo.

Uma abordagem adequada para esse tipo de problema é o aprendizado por reforço, no qual o sistema aprende por tentativa e erro, recebendo sinais de reforço na forma de recompensas. Quando combinado com redes neurais, o aprendizado por reforço é capaz de lidar com problemas de forma independente do modelo matemático (*model-free*), generalizando para diferentes condições e se adaptando a sistemas não lineares com maior eficiência. O fato de não depender de um modelo exato permite um treinamento mais rápido e menos custoso. Além disso, redes neurais são flexíveis o suficiente para lidar naturalmente com sistemas altamente não lineares [6]. Com base nessas características, o algoritmo PIME-PPO foi escolhido para ser aplicado ao controle do CPAP, visando obter bons resultados. Dessa forma, a principal contribuição deste trabalho é a implementação, adaptação e avaliação do PIME-PPO em um ambiente que encapsula a simulação matemática do equipamento CPAP.

2 Fundamentação Teórica

Esse trabalho é construído em cima de conceitos advindos da inteligência artificial, teoria do controle e medicina. Portanto, essa seção busca fundamentar os conceitos matemáticos importantes para este trabalho.

2.1 Controlador PID

Teoria dos controladores é um campo da matemática que estuda como escolher um *input* em um modelo de equações com o objetivo de chegar em um estado desejado [7]. Os sistemas podem ser de *loop* aberto (*feedfoward*) ou *loop* fechado (*feedback*). No sistema *feedfoward*, a saída não influencia o próximo *input*, enquanto no sistema *feedback*, a saída é usada para ajustar a entrada. Na perspectiva do aprendizado por reforço, o termo 'sistema' pode se referir tanto ao ambiente quanto ao conjunto formado pelo ambiente e o controlador, enquanto o agente inteligente assume o papel de controlador.

No campo do controle, os sistemas dinâmicos são modelados matematicamente com o objetivo de descrever seu comportamento ao longo do tempo. A partir desses modelos, é possível programar controladores capazes de garantir propriedades desejáveis, como estabilidade e resposta rápida a perturbações. Descrever um modelo exato de um sistema real é extremamente difícil, pois envolve múltiplas variáveis, incertezas e dinâmicas complexas que muitas vezes não podem ser totalmente capturadas matematicamente. Além disso, equações mais detalhadas aumentam significativamente o custo computacional, tornando inviável seu uso em tempo real. Por essa razão, modelos aproximados são frequentemente utilizados, equilibrando precisão e viabilidade computacional.

O Proporcional-Integral-Derivativo (PID) é um exemplo de controlador que calcula uma saída para manter um processo em algum estado desejado. O processo refere-se ao sistema físico ou lógico a ser controlado, e o valor de referência desse processo, denominado *setpoint*, pode variar ao longo do tempo. Utilizando a diferença entre o *setpoint* atual $y^{current}$ medido por algum sensor e o *setpoint* objetivo y^{ref} como erro, o controlador PID faz a soma de três fórmulas para calcular a saída, buscando manter o *setpoint* desejado.

$$\text{Error} = \varepsilon_t = y_t^{ref} - y_t^{current}$$

$$\text{Proportional} = P = K_p \cdot \varepsilon_t$$

$$\text{Integral} = I = K_i \cdot \int_0^t \varepsilon_t dt$$

$$\text{Derivative} = D = K_d \cdot \frac{d\varepsilon_t}{dt}$$

$$\text{saída} = U = P + I + D$$

Para o PID fazer o ambiente manter o *setpoint*, é necessário escolher os valores constantes K_p, K_i e K_d, um processo conhecido como *ajuste de parâmetros*. Esses valores dependem da aplicação específica e, mesmo com algoritmos para automatizar essa escolha, ajustes manuais de um profissional ainda são frequentemente necessários [8].

Duas propriedades específicas do controlador precisam ser mencionadas. Ao utilizar $K_p \neq 0$, $K_i = 0$ e $K_d = 0$, o y obtido como resultado se aproxima rapidamente do y^{ref} ao longo do tempo, mas o erro permanece constante ao chegar perto demais. Diminuir P desacelera a aproximação e diminui o erro constante, mas o erro ainda existe e, para a maioria das aplicações, ainda é significativamente grande. A outra propriedade está no ajuste de parâmetros que é feito para uma pequena faixa de *setpoints*. A performance do PID degrada quando os objetivos estão muito distantes dos *setpoints* previstos no ajuste.

2.2 Aprendizado por reforço

Um agente inteligente é um sistema que percebe seu ambiente através de sensores, realiza ações apropriadas para atingir seus objetivos através de atuadores, aprende com a experiência e se adapta às mudanças em seu ambiente [9].

Aprendizado por reforço (RL do inglês *reinforcement learning*) é um campo de aprendizado de máquina (ML do inglês *machine learning*) que estuda problemas e suas soluções em que os agentes inteligentes, por meio da interação com seu ambiente, aprendem a maximizar a média dos retornos. Um retorno é o somatório de recompensas descontadas calculadas a cada passo [10]. Um episódio é uma sequência de passos e cada episódio tem um retorno associado.

O estado do ambiente é a representação das informações relevantes que descrevem sua condição em um determinado instante, servindo como base para que o agente tome decisões e ajuste seu comportamento de acordo com as mudanças observadas. Um passo equivale a um ciclo de iteração do agente com o ambiente e uma observação é um subconjunto do estado s, sendo este subconjunto o que, de fato, chega nas percepções do agente.

Todo algoritmo RL tem uma política que é aprendida para maximizar a esperança (média) dos retornos. Se a ação for contínua, como é o caso no CPAP, a política representada por μ retorna a ação $a \in A$ ou uma distribuição estatística que representa a probabilidade de escolher a ação a dado o estado s. π é usada para representar a política de ação discreta.

Algoritmos de RL podem adotar várias abordagens, a exemplo de, *on-policy*, *off-policy*, *model-free*, *model-based*, *policy-gradient*, *actor-critic*, *Monte Carlo*, *Deep RL*, etc. É necessário detalhar que a abordagem *on-policy* atualiza a política com base nos dados gerados pela decisão dessa mesma política. Em algoritmos *policy-gradient*, a política é um modelo derivável, o agente aprende políticas de ação diretamente (em vez de estimar valores de estado ou ação) e o algoritmo busca encontrar a política de ação que maximiza a recompensa esperada em um ambiente. No caso do aprendizado por reforço profundo (*Deep RL*), redes neurais são utilizadas para representar funções de valor $V(s)$, a política $\mu(s)$ ou ambas, permitindo que o agente aprenda representações mais complexas do ambiente. Um exemplo notável de algoritmo RL com as abordagens *on-policy*, *model-free*, e *policy-gradient* é o PPO (*Proximal Policy Optimization*). Esse algoritmo foi usado em parte do treinamento de algumas versões do ChatGPT e adaptado neste trabalho para controlar o equipamento CPAP. Para utilizar a teoria dos controladores junto com o aprendizado por reforço é preciso observar relações entre os conceitos dessas duas áreas. Essas relações estão na tabela 1.

Tabela 1: Tabela de relações.

Relação	**Descrição**
Observação e Estado	Observação $\tilde{x}_t$ é equivalente ao estado do sistema de controle unido com outras variáveis úteis para o treinamento.
Exploração e Excitação do Controlador	Explorar o espaço de estados é o mesmo que excitar o controlador [6].
Recompensa, Erro e objetivo	A recompensa é maximizada enquanto o erro é minimizado, ou seja, $Reward \propto -Error$ [11]. O objetivo y_t^{ref} usado para calcular a recompensa varia ao longo do tempo, razão pela qual é indexada por t.
Termo Integral e Acumulador de Erro	O termo integral em um controlador PID pode ser visto como o acumulador de erro em RL, incorporado ao estado como uma memória que influencia as ações futuras, ajudando a corrigir erros estáticos [6, 11].
Sinônimos de Ação	*Input* u_t, Sinal de controle, saída de controle, ação de controle e ação de um agente inteligente são sinônimos.
Saturação do Controlador	A saturação de um controlador significa atingir o limite máximo e mínimo do espaço de ações, impedindo que a saída não possa mais ser aumentada ou diminuída. Isso pode limitar a exploração de novos estados importantes [6].
Sistema Controlado e Agente Inteligente	Na perspectiva de aprendizado por reforço, o processo controlado é o ambiente e o agente inteligente é o controlador.
Notação matemática do estado	O estado é o vetor $x_t = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^{n_x}$, tal que n_x é a quantidade de elementos. O *output* x_{t+1} do modelo matemático f_p é o próximo estado do ambiente. $y_t = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^{n_y}$ é formado por valores dentro de x_t, ou seja, os valores dentro de x_t tentam se igualar ao vetor y_t^{ref}. Por fim, y^{ref} e y_t são escalares nos ambientes deste trabalho.

3 Trabalhos relacionados

A literatura recente aplicou algoritmos RL combinados com algum controlador da engenharia em sistemas de *loop* fechado não-lineares com objetivo variável. Nesses trabalhos é observado que RL sozinho enfrenta vários problemas que não só inviabilizam a eficácia do treinamento, como também podem ser resolvidos através da combinação com o PID, um controlador simples e robusto. Em especial, os artigos [11] e [12] propõem formas inovadoras e importantes de serem analisadas para esse trabalho.

Em [12], vários algoritmos RL (PPO, DDPG e A2C) foram combinados com PID, de tal forma que o agente inteligente tem o papel de ajustar dinamicamente os ganhos K_p, K_i, K_d. Esse artigo mostra que algoritmos RL não substituem o PID, mas servem como um supervisor. Para além dessa combinação, modificações nos algoritmos RL foram necessárias para melhorar a robustez, estabilidade e capacidade de adaptação a sistemas não lineares.

A outra abordagem descrita em [11], utiliza a soma da ação de um algoritmo RL *policy gradient* com a saída do PID para gerar a ação final de controle. A combinação foi implementada com o PPO, gerando o PIME-PPO, junto com três mudanças importantes para melhorar o desempenho do algoritmo PIME-PPO. Essas mudanças são, em resumo:

1. Utilizar *model ensemble* (figura 1) em modelo de simulações para aprimorar a robustez às incertezas nos parâmetros desse modelo de simulação; O modelo de simulação é uma equação que descreve um sistema da realidade; os parâmetros são variáveis dessa equação; Nos experimentos de [11], cada parâmetro p é aleatoriamente amostrado de uma distribuição P escolhida pelo autor.

2. Modificar a representação do vetor de estado x_t para $\tilde{x}_t$, tal que

$$
\begin{aligned}
\tilde{x}_t &= [x_t^T \quad z_t]^T \\
z_{t+1} &= z_t + \varepsilon_t \\
z_0 &= 0 \\
\varepsilon_t &= y^{ref} - y^{current} \\
y^{current} &= x_t^{tracked} \in x_t \\
x_0 &= \text{vetor inicializado aleatoriamente}
\end{aligned}
$$

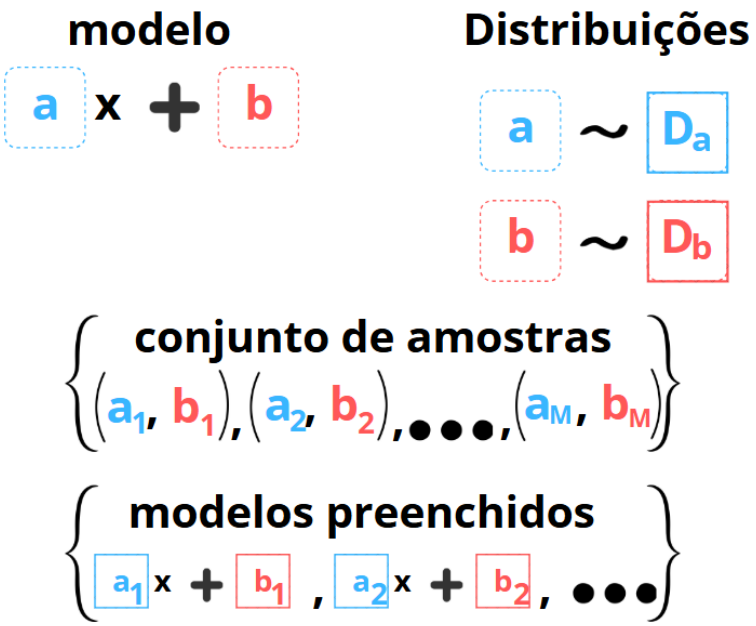

Figura 1: Nesse exemplo ilustrativo de *model ensemble*, o modelo $ax + b$ possui parâmetros a e b amostrados de distribuições independentes D_a e D_b. Cada amostra gera um modelo distinto, e o conjunto dos modelos preenchidos forma o *ensemble* de tamanho M.

y_{ref} é o *setpoint*, $y^{current}$ é o valor que se quer igualar ao *setpoint*, z_t é o acumulador de erro contendo a soma de todos os ε_t anteriores e ε_t é o erro. Isso ajuda a treinar uma política que pode lidar com alterações nos parâmetros do sistema e distúrbios externos não modelados.

3. Começar com um controlador anterior PI (PID sem D) sem ajuste de parâmetros para orientar a política π_θ a se aproximar rapidamente do *setpoint* y^{ref}, ajudar a exploração do espaço de estados, superar dinâmicas desconhecidas do sistema e evitar ações inseguras que possam levar o sistema a estados inviáveis ou instáveis.

$$\text{ação\_PID} = u_t^{PID} = k(\tilde{x}_t, y_t^{ref}) \tag{1}$$

Um desafio observado em [11] e [6] é a possibilidade de ações inseguras que ocorrem quando o agente escolhe ações que podem levar o sistema a estados perigosos ou inviáveis, como ultrapassar limites físicos, causar instabilidade ou danificar o sistema controlado. Além disso, a exploração ineficiente do espaço de estados refere-se à dificuldade do agente em testar uma gama ampla e representativa de ações e estados relevantes, o que pode resultar em aprendizado limitado e subótimo. [11] explica que o uso de simulação permite ao agente explorar o ambiente em um modelo matemático do sistema real, eliminando riscos no mundo físico na fase de treinamento e fornecendo dados abundantes, ou seja, melhora a eficiência da exploração.

Aleatorização de Domínio (AD)[13, 14] é uma técnica utilizada em aprendizado por reforço para facilitar a transferência de políticas aprendidas em simulações para o mundo real e garantir que a política aprendida seja robusta a variações que ela possa encontrar no mundo real. Essa técnica consiste em escolher valores aleatoriamente para parâmetros do domínio durante o treinamento do modelo. No treinamento descrito na seção de metodologia deste trabalho, o *Model Ensemble* é uma forma de utilizar a técnica AD para gerar uma política robusta.

Os algoritmos RL utilizados seguem a abordagem *policy gradient* e isso significa que possuem uma rede neural. Redes neurais são flexíveis e, portanto, capazes de atuar em um espaço de estados muito vasto. A estrutura da rede influencia a facilidade de aprendizado, eficiência computacional e capacidade de lidar com distúrbios e variabilidade nos dados observados durante e depois do treinamento.

4 Formulação Matemática do Problema

Considerando um modelo de espaço de estados não linear em tempo discreto

$$x_{t+1} = f_p(x_t, u_t, w_t) \tag{2}$$
$$y_t = h_p(x_t) \tag{3}$$

onde f_p e h_p são funções desconhecidas e parametrizadas por p, $x_t \in \mathbb{R}^{n_x}$ que é o estado atual do sistema, $u_t \in \mathbb{R}^{n_u}$ que é a ação gerada pelo controlador (ou seja, o *input*), $y_t \in \mathbb{R}^{n_y}$ que é o *output*, e $w_t \in \mathbb{R}^{n_w}$ que é o ruído do sistema. A tarefa é combinar um algoritmo RL com PID para treinar um agente inteligente que faz o papel de um controlador de estados por *feedback* de *setpoint* com o objetivo de rastrear um *setpoint* y^{ref} variável no tempo. Para lidar com vários *setpoints* em RL, um *framework multi-goal* RL [15] é considerado. o objetivo é, então, encontrar um controlador

$$u_t = g_\theta(x_t, y_t^{ref}) \tag{4}$$

tal que θ é um vetor de parâmetros que maximiza o retorno descontado

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{5}$$

tal que $\gamma \in [0,1]$ é o fator de desconto e $r_t = R(x_t, u_t, y_t^{ref})$ é a função de recompensa no passo de tempo t.

5 Metodologia

Aplicando ao controlador da equação (4) as modificações propostas em [11], o controlador final é descrito como a combinação de 2 controladores

$$u_t = \pi_\theta(PID(\varepsilon_t, z_t) + g_\theta(\tilde{x}_t, y_t^{ref})) \tag{6}$$

A abordagem PIME-TD3 foi selecionada para ser treinada e avaliada nos ambientes Cascade Water Tank e CPAP. Para a implementação, utilizamos o algoritmo TD3, enquanto os ambientes foram desenvolvidos com a biblioteca Gymnasium [16], amplamente usada para desenvolver ambientes para algoritmos de RL.

Antes do treinamento final, realizamos uma busca de hiperparâmetros combinando ajustes manuais e otimização automatizada com a biblioteca *Optuna* [17]. Todo o código produzido para este trabalho está disponível no GitHub [1].

5.1 Desenvolvimento do PIME-TD3

É importante destacar que a implementação do PIME-TD3 precisou ser um pouco diferente de [11] devido a lacunas em detalhes cruciais de implementação deixados de lado pelo artigo.

Durante a inicialização do PIME-TD3, o PID precisa ser inicializado com ganhos de alguma forma e isso pode impactar significativamente os resultados do treinamento. Existem métodos de inicialização do PID que dependem de modelo ou iteração humana, mas é preferível não depender do modelo e ser programático, uma vez que isso é aplicável em qualquer ambiente sem intervenção humana. Como [11] não especifica esse detalhe, foi testada a inicialização com valores manuais, com método Ziegler-Nichols (ZN) e com *Optuna* executando um treinamento do PI sozinho (sem ajuda do TD3) em cada ambiente. O método ZN gerou valores muito altos de K_p e K_i, gerando instabilidade no ambiente *Cascade Water Tank* e não foi possível aplicar esse método nas equações do CPAP. Por essas razões, nenhum experimento com ZN no CPAP foi realizado. Enquanto os testes manuais do PIME-TD3 apontaram valores de K_p bons na faixa de 3 até 7 e K_i por volta de 0.015 no ambiente *Cascade Water Tank*, os testes no *CPAP* apontaram valores de $K_p = 1.5$ e K_i por volta de 0.005 no CPAP. Experimento com *Optuna* resultou nos valores $K_p = 9.1$ e $K_i = 4.31$ para *Cascade Water Tank* e $K_p = 10.0$ e $K_i = 9.2$ para *CPAP*. Apesar dos valores encontrados manualmente e programaticamente com *Optuna* parecerem distantes, a pontuação (somatório das recompensas descontadas do episódio) e os gráficos de y_t sobre o tempo são semelhantes e aceitáveis para ambos os ambientes.

A ação final escolhida pelo PIME-TD3 é descrita como

[1] *link do código fonte*

$$\eta \sim \mathcal{N}(0, \sigma^2) \tag{7}$$

$$z_t = clip(z_t, -25, +25) \tag{8}$$

$$g_\theta(\tilde{x}_t, y_t^{ref}) = clip\Big(u_\theta^{TD3} + \eta,\ a_{min},\ a_{max}\Big) \tag{9}$$

$$u_t = \pi_\theta\Big(PID(\varepsilon_t, z_t) + g_\theta(\tilde{x}_t, y_t^{ref})\Big) \tag{10}$$

tal que η é o ruído para aumentar a exploração e z_t é limitado entre -25 e +25, conforme [11] sugere.

A estrutura da rede neural utilizada está ilustrada nas figuras 2 e 3. A inicialização da rede recebe parâmetros que definem qual a função de ativação usada, se a ativação será aplicada no final, qual a quantidade de neurônios por camada e se deve existir uma ramificação na rede neural. A rede é composta por três camadas lineares e uma ramificação para se assemelhar à estrutura da referência [11].

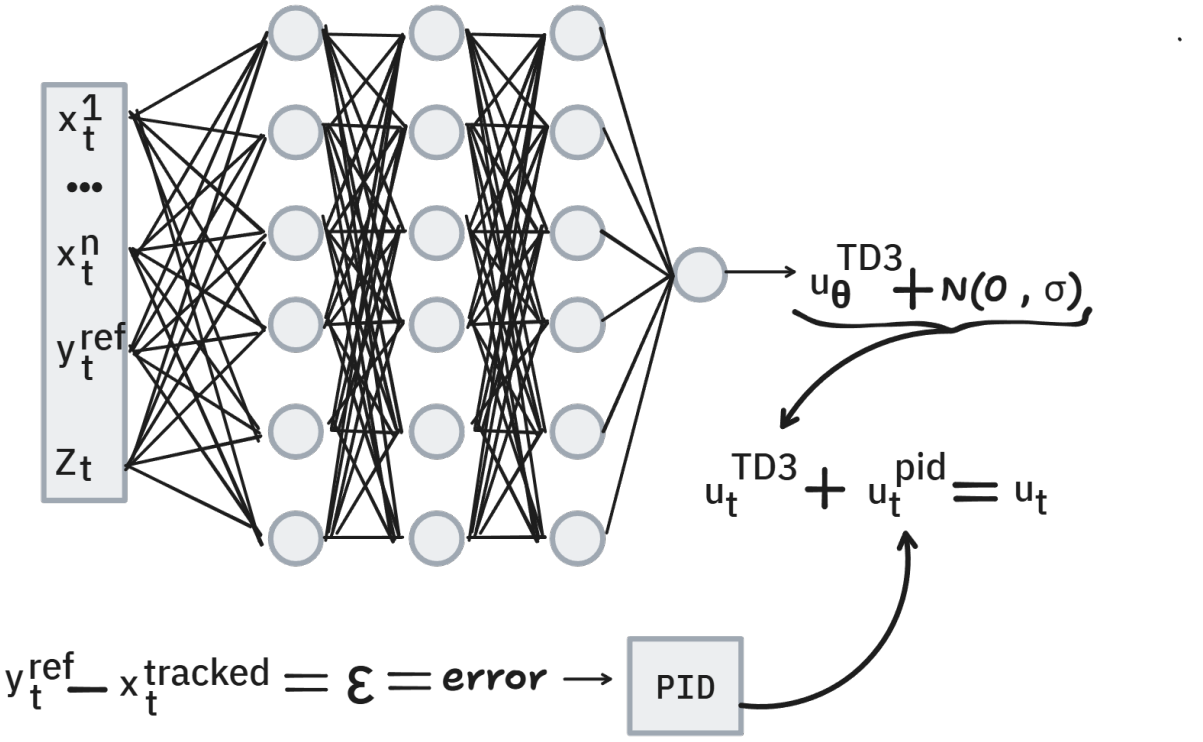

Figura 2: Estrutura da rede neural do PIME-PPO adaptado sem separação.

Com essas modificações, e considerando que o PIME-TD3 trata os valores K_p e K_i como hiperparâmetros otimizáveis por métodos que não dependem do conhecimento explícito do modelo nem de ajustes manuais, essa versão aprimorada será denominada PIME-TD3* e está descrita no algoritmo 1.

5.2 Ambientes

Os ambientes utilizam

- Função de recompensa $R(x_t, y_t^{ref}) = -(x_t - y_t^{ref})^2$, que representa o erro quadrático entre a saída do sistema x_t e o valor desejado y^{ref} em cada instante. Ou seja, essa função penaliza desvios entre a saída real e o *setpoint*, incentivando o agente a minimizar a diferença ao longo do tempo. Como x_t é um vetor atualizado pela simulação, ele contém o valor específico $x_t^{tracked}$, que é a variável de interesse sendo controlada e comparada com o *setpoint*.

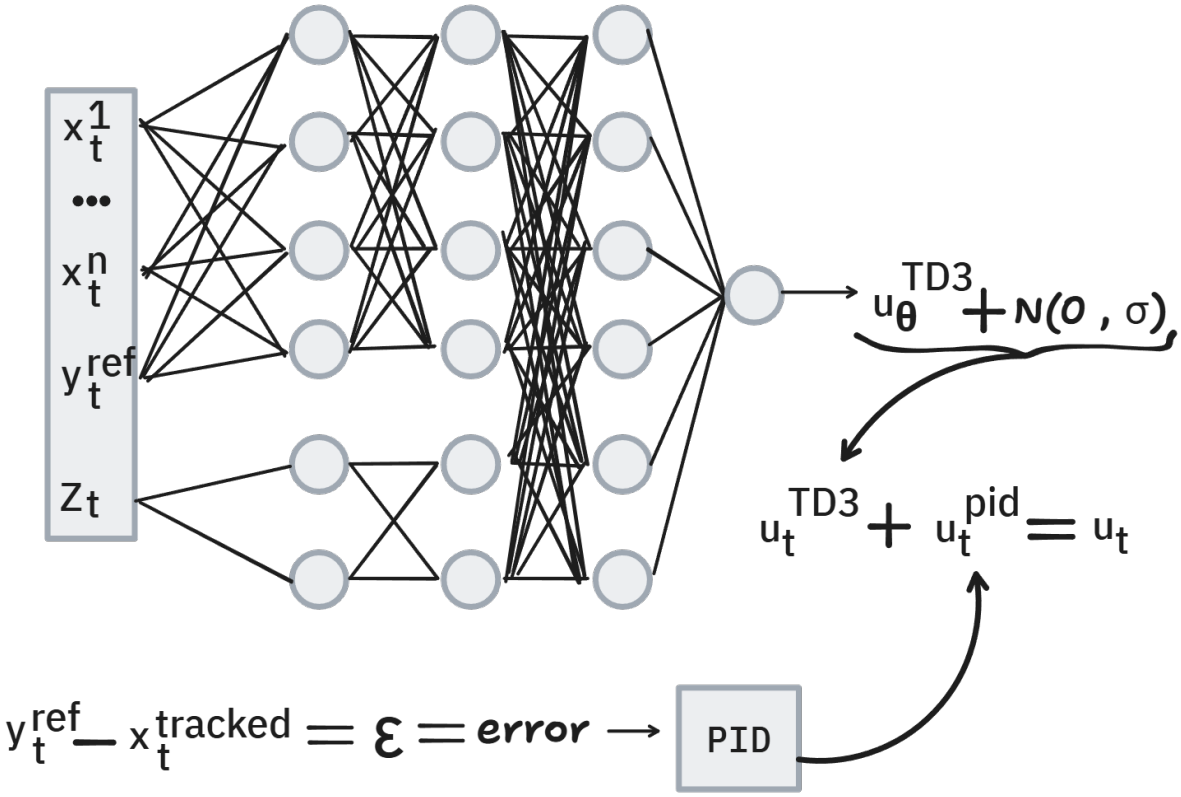

Figura 3: Estrutura da rede neural do PIME-PPO adaptado com separação. As duas camadas lineares na parte superior tem 66% dos neurônios, enquanto a inferior tem 33%.

- Fila de objetivos com intervalos de duração para cada objetivo.

```
## Scheduller
"set_points": [3, 6, 9, 4, 2],
"intervals": [400, 400, 400, 400, 400],
```

- Método *reset* que escolhe valores de estados $(x_1, \ldots x_n)$ aleatoriamente.
- Método *simulation_model* que recebe a ação u_t, o vetor de estados estendido $\tilde{x}$, e uma amostra de parâmetros provenientes de um *Model Ensemble* e retorna o novo vetor de estado x.
- Cada parâmetro do *Model Ensemble* é gerado aleatoriamente (AD) a partir de uma distribuição.

```
## EnsembleGenerator
"distributions": {
    "a": ("constant",
         {"constant": 981}),
    "b": ("uniform",
         {"low": 0.0, "high": 10.0}),
    "c": ("normal",
         {"mean": 1, "sigma": 2})
}
```

- Regra de finalização que indica quando o episódio será finalizado.
  - parada por intervalos: finaliza após cada objetivo ter sido utilizado por toda a sua duração uma vez.
  - limite de passos: finaliza ao atingir um número máximo de passos.

Algorithm 1 Pseudocódigo do PIME-TD3*

Entrada: Ambiente *multi-goal env* e hiperparâmetros da tabela 2.

$steps\_per\_iteraction \leftarrow steps\_per\_episode \cdot size(ensemble) \cdot episodes\_per\_sample$
$iterations \leftarrow \lceil steps\_to\_run / steps\_per\_iteration \rceil$
Para $iteration = 1, 2, \ldots, iteractions$ **faça**
    Para $m = 1, 2, \ldots, size(ensemble)$ **faça**
        Para $j = 1, \ldots, episodes\_per\_sample$ **faça**
            Carrega os parâmetros de um modelo do *model samples* no ambiente *env*.
            Reseta o ambiente para o estado inicial:
                $\tilde{x_t} \leftarrow reset(env)$.
            Enquanto not done **faça**
                $pi\_action \leftarrow PID(error)$.
                $td3\_action \leftarrow g_\theta(\tilde{x_t})$.
                $u_t \leftarrow pi\_action + td3\_action$.
                Aplica ação no ambiente e atualiza variáveis:
                    $r_t, \tilde{x}_{t+1}, done \leftarrow step(env, u_t)$.
                Aarmazena no *buffer*: $(\tilde{\tilde{x}_t}, action, r_t, done, \tilde{x_{t+1}})$.
                Atualiza a observação:
                    $\tilde{x_t} \leftarrow \tilde{x}_{t+1}$.
            Fim Enquanto
            Treina a rede e esvazia o *buffer*.
        Fim Para
    Fim Para
Fim Para

Tabela 2: Hiperparâmetros utilizados no PIME-TD3 para os experimentos Cascade Water Tank e CPAP. Os hiperparâmetros marcadas com * foram otimizados com *Optuna*.

Hiperparâmetros	**Cascade Water Tank**	**CPAP**
Objetivos	[5, 2, 6, 8, 3]	[3, 12, 5, 7, 5]
Duração de cada objetivo	[400, 400, 400, 400, 400]	[2000, 2000, 2000, 2000, 2000]
Decaimento de alvo	0.97	0.97
Limites do integrador do PID e z_t	[-25, 25]	[-25, 25]
Desconto γ	0.995	0.94
Épocas de treinamento*	16	4
Frequência de atualização da política*	32	200
Coeficiente de aprendizado α*	2.3305045634697055e-05,	3.357890528561397e-05
Tamanho do *batch**	256	64
Neurônios por camada*	42	48
Camadas divididas*	True	True
Função de ativação*	tanh	tanh
Ruído de exploração	0.01	0.2
Limite de rauído	0.05	0.5
Episódios por amostra de ensemble	5	5
Tamanho do ensemble M	100	100
x_i rastreado	"x_2"	"x_3"
Passos	1,000,000	1,000,000
Semente usada no ensemble	42	42
Tipo de PID	PI	PI
Kp, Ki, Kd	9.31, 0.015, 0	1.0, 1.2, 0
Limites da ação do TD3	(-30, 30)	(-30, 30)
Limites máximos de cada x_i	(10, 10)	(∞, ∞, ∞)
Limites mínimos de cada x_i	(0, 0)	$(-\infty, -\infty, -\infty)$

5.2.1 Cascade Water Tank

O ambiente não linear discretizado com o método de Euler está descrito nas fórmulas abaixo:

$$\Delta l1_t = (-p_1 \cdot \sqrt{2gl1_t} + p_3 \cdot u_t) \cdot dt, \tag{11}$$

$$\Delta l2_t = (p_1 \cdot \sqrt{2gl1_t} - p_2 \cdot \sqrt{2gl2_t}) \cdot dt \tag{12}$$

Nesse ambiente, o objetivo rastreado é a altura em cm do segundo tanque de água ($l2_t$) e os valores dos tanques de água têm uma capacidade máxima de 10 cm. A Ação u_t é medida em volts e valores negativos são mapeados para 0, uma vez que não é possível remover água.

As distribuições, unidades de medida e definições dos parâmetros do ensemble estão listadas abaixo:

- $p1$, distribuição uniforme $U_1(0.0015, 0.0024)$, é a razão entre a área da abertura do tanque superior e a área da seção transversal do tanque superior.
- $p2$, distribuição uniforme $U_2(0.0015, 0.0024)$, é a razão entre a área da abertura do tanque inferior e a área da seção transversal do tanque inferior.
- $p3$, distribuição uniforme $U_3(0.07, 0.17)$, é a razão entre a constante da bomba e a área do tanque superior.
- g, constante igual a $981 cm/s^2$, gravidade.
- dt, constante igual a $2s$, intervalo de tempo que implementa discretização de Euler.

O vetor x_t é composto por $(l1_t, l2_t, u_{t-1})$.

5.2.2 CPAP

O ambiente foi descrito em duas partes, *ventilator* e *lung* (Figura 4), que foram modeladas com a ajuda de um especialista engenheiro de equipamentos biomédicos. A fórmula do *ventilator* 14 recebe uma voltagem e transforma-a em pressão. As fórmulas de *lung* atualizam o fluxo, volume e pressão do ar conforme a fase da respiração (expirando, inspirando, pausa).

$$u_t\_filtered = u_{t-1}\_filtered + \frac{dt}{tau_f} * (u_t - u_{t-1}\_filtered) \tag{13}$$

$$\text{PEEP} = k_v \cdot u_t\_filtered \cdot \exp(\frac{-dt}{t_v}) \tag{14}$$

Nas equações 13 e 14, k_v é o ganho do ventilador, t_v e tau_f são constantes de tempo de atuação e dt é a quantidade discreta de tempo passado em cada iteração. Devido às limitações do PID isolado (ou mesmo combinado com TD3) na

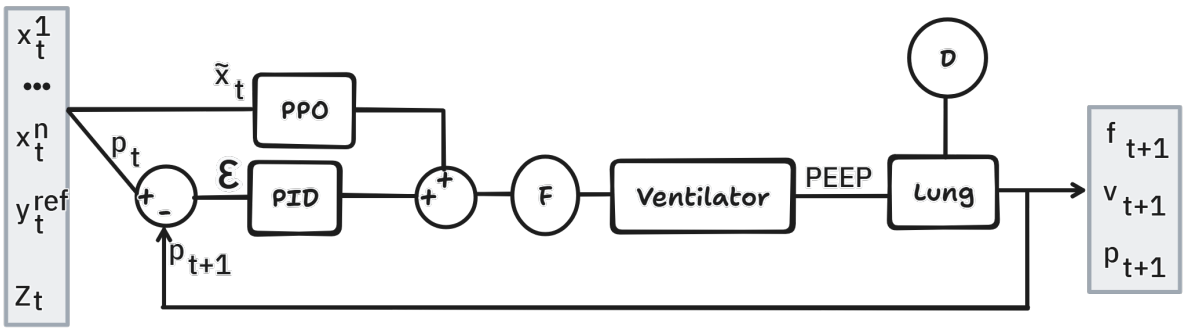

Figura 4: Diagrama do sistema com *feedback* do ambiente CPAP. O *ventilator* recebe uma voltagem filtrada e transforma em pressão. A PEEP é aplicada no *lung* para atualizar o fluxo, volume e pressão do estado x_t. D representa os distúrbios aleatórios utilizados no *lung*.

estabilização da pressão do CPAP [5], um filtro (equação 13) de passa-baixa foi adicionado ao sistema de controle. Esse filtro suaviza o sinal de tensão u_t do PID, ou seja, ele atenua variações abruptas e ruídos de alta frequência, permitindo que apenas mudanças mais lentas e significativas passem adiante. Dessa maneira, o filtro torna o sistema mais estável, evitando que oscilações bruscas na saída do controlador instabilizem a pressão observada no ambiente.

O *lung* é uma função de transferência não linear do CPAP modelado com três equações que mudam de acordo com a fase da respiração (expiração, inspiração, pausa). Essas fórmulas utilizam um ruído aleatório $noise_t$ gerado com a distribuição normal $\mathcal{N}(0.0, 10.0)$ a cada iteração. Enquanto está expirando, as equações usadas são:

$$elapsed\_phase\_time = t - t_0^{phase} \tag{15}$$

$$lpp = \begin{cases} peep, \text{primeira iteração} \\ p_{t-1}, \text{primeira iteração ao trocar de fase} \\ lpp, \text{não modifica caso contrário} \end{cases} \tag{16}$$

$$\tag{17}$$

$$f_{t+1} = \frac{(peep - lpp)}{\hat{r}_{aw}} \cdot \exp(\frac{-(ept)}{(\hat{r}_{aw} \cdot c_{rs})}) + noise_t \tag{18}$$

$$v_{t+1} = v_t + f_{t+1} \cdot dt \tag{19}$$

$$p_{t+1} = f_{t+1} \cdot \hat{r}_{aw} + \frac{v_{t+1}}{c_{rs}} + peep \tag{20}$$

Enquanto está inspirando, as equações usadas são:

$$f_{t+1} = f_i + noise_t \tag{21}$$

$$v_{t+1} = v_t + (f_{t+1} \cdot dt) \tag{22}$$

$$p_{t+1} = f_{t+1} \cdot \hat{r}_{aw} + \frac{v_{t+1}}{c_{rs}} + peep \tag{23}$$

Enquanto está pausado entre a fase de inspiração e expiração, as equações são:

$$f_{t+1} = noise_t \tag{24}$$

$$v_{t+1} = v_t + (f_{t+1} \cdot dt) \tag{25}$$

$$p_{t+1} = f_{t+1} \cdot \hat{r}_{aw} + \frac{v_{t+1}}{c_{rs}} + peep \tag{26}$$

A descrição e unidade de medida das variáveis estão na tabela 3. Nesse ambiente, o objetivo rastreado y_t^{ref} é a pressão p_t medida em cmH2O pelo equipamento. As distribuições dos parâmetros do ensemble estão listadas abaixo:

- r_{aw}, distribuição uniforme $U_4(2,5)$, resistência das vias aéreas.
- c_{rs}, distribuição uniforme $U_5(35,60)$, complacência do paciente.
- v_t, distribuição uniforme $U_6(300,400)$, volume de ar entregue pelo sistema de ventilação.
- rr, distribuição uniforme $U_7(10,20)$, taxa de respiração do paciente.
- t_i, distribuição uniforme $U_8(0.8,1.2)$, tempo respirando.
- t_ip, distribuição uniforme $U_9(0.1,0.3)$, tempo na fase de pausa.
- k_v, constante igual a $0.99cm^3/s/V$, ganho do *ventilator* (equação 14).
- t_v, constante igual a $0.01s$, constante de tempo do *ventilator* (equação 14).
- tau_f, constante igual a $0.1s$, constante de tempo do filtro passa-baixo (equação 13).
- f_s, constante igual a $1000Hz$, frequência de amostragem.
- dt, constante igual a $0.001s$, intervalo de tempo que implementa discretização de Euler.

O vetor x_t é composto por
$(f_t, v_t, p_t, p_{t-1}, one\_hot\_phase, u_{t-1})$, tal que a codificação *one-hot* da fase respiratória é definida como:

Tabela 3: Descrição das variáveis usadas no ambiente CPAP

Símbolo	Descrição	Unidade de medida
u_t	Voltagem aplicada no *ventilator*	V
k_v	Ganho do *ventilator*	ml/s/V
t_v	Constante de tempo	s
r_{aw}	Resistência das vias aéreas	cmH2O/L/s
$\hat{r}_{aw}$	Resistência das vias aéreas convertida	cmH2O/ml/s
c_{rs}	Complacência do sistema respiratório	ml/cmH2O
v_t	Volume corrente	ml
PEEP	Pressão gerada pelo ventilator (equação 14)	cmH2O
rr	Frequência respiratória	min^{-1}
$\hat{rr}$	Frequência respiratória convertida	Hz
t_i	Tempo no passo i	s
t_{ip}	Tempo de pausa inspiratória	s
t_c	Tempo de ciclo	s
t_e	Tempo expiratório	s
f_i	Fluxo inspiratório	ml/s
dt	Paso de tempo discreto	s
t_0^{phase}	Tempo inicial da fase atual	s
lpp	*last pause pressure*	cmH2O
ept	*elapsed phase time*	s
f_{t+1}	Fluxo de ar	ml/s
v_{t+1}	Volume de ar	ml
p_{t+1}	Pressão de ar	cmH2O

Expiração: $(\ldots, \textbf{True}, False, False, \ldots)$
Inspiração: $(\ldots, False, \textbf{True}, False, \ldots)$
Pausa: $(\ldots, False, False, \textbf{True}, \ldots)$

5.3 Pipeline de treinamento

A preparação dos ambientes e do algoritmo está detalhada no algoritmo 2.

Algorithm 2 O pseudo código que explica a preparação do ambiente e treinamento está descrito abaixo.

1: Inicializar hiperparâmetros do TD3, PID e ambiente, incluindo as distribuições do *model ensemble* e durações de cada *setpoint*.
2: Usar hiperparâmetros, *Scheduller* e *EnsembleGenerator* para inicializar ambiente gymnasium.
3: Instanciar PIME-TD3* com o ambiente, hiperparâmetros, *Scheduller* e *EnsembleGenerator*.
4: Invocar método *train* do PIME-TD3* com uma quantidade de passos desejada. Essa quantidade pode ser automaticamente modificada pelo método para se adequar à quantidade de passos guardados dentro do *buffer* de treinamento.
5: Executa o pseudocódigo do PIME-TD3* (algoritmo 1) e salva os resultados em arquivos csv.

Para replicar os resultados de [11], é preciso escolher valores para hiperparâmetros descritos na tabela 2. Primeiro, experimentos com valores manuais foram realizados. Em seguida, uma busca de hiperparâmetros foi realizada em ambos os ambientes. Os espaços de busca de cada ambiente estão definidos na tabela 4 abaixo.

Utilizando os melhores valores de ganhos do PID, foram realizados 170 ensaios no ambiente *Cascade Water Tank* e 170 ensaios no ambiente CPAP. Ambos os estudos foram conduzidos em 17 threads, com cada ensaio executando 400.000 passos. Os hiperparâmetros usados estão na tabela 2 e as definições do *model ensemble* estão detalhadas nas subseções de cada ambiente.

Tabela 4: Espaços de busca usados no *Optuna* em cada ambiente

Hiperparâmetros	**Cascade Water Tank**	**CPAP**
Épocas de treinamento	2 a 20 (step 2)	0.01 a 10.0 (log=True)
Frequência de atualização da política	2 a 32 (step=2)	2 a 32 (step=2)
batch_size	[32, 64, 128, 256]	[32, 64, 128, 256]
Coeficiente de aprendizado α	1e-5 a 3e-4 (log=True)	1e-5 a 3e-4 (log=True)
Neurônios por camada	6 a 60 (step=6)	6 a 60 (step=6)
Utiliza rede neural ramificada	[True, False]	[True, False]
Função de ativação	["no activation", "relu", "tanh"]	["no activation", "relu", "tanh"]
Tamanho do *ensemble*	1	1

6 Experimentos

Ao testar os ganhos do PID gerados pelo *Optuna* no ambiente *Cascade Water Tank*, o nível de água oscilou muito longe do *setpoint* devido ao *Ki* alto. Isso foi corrigido utilizando um $ki = 0.015$ escolhido manualmente.

Conforme ilustrado na figura 5, surgiram dificuldades em reproduzir o mesmo desempenho obtido em [11], mesmo utilizando os hiperparâmetros fornecidos no estudo original. Essas discrepâncias são atribuídas a incertezas nos valores das constantes K_p e K_i, variações na implementação da rede neural e hiperparâmetros não especificados, como o número de épocas de treinamento.

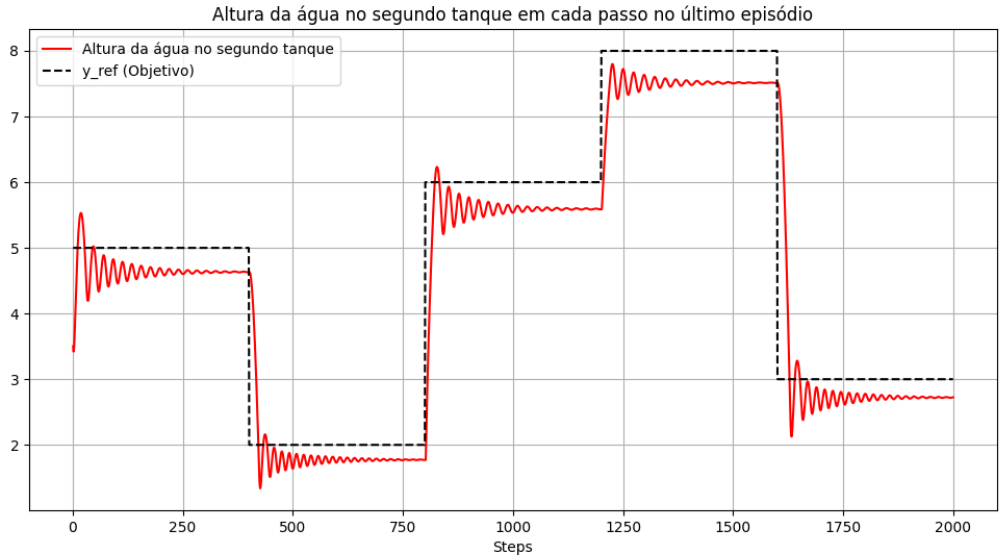

Figura 5: Gráfico da altura da água em cm no segundo tanque em cada passo. $K_p = 9.31$, $K_i = 0.015$, 6 neurônios por camada, nenhuma ativação, divisão dos inputs nas duas primeiras camadas, ensemble com tamanho igual a 100.

Ao testar modificações, como funções de ativação diferentes, aumento no número de neurônios, estruturas de redes diferentes, foram obtidos resultados mais promissores. O resultado promissor da figura 6 utiliza os hiperparâmetros otimizados por *Optuna*.

No ambiente CPAP, o TD3 demonstrou um desempenho inferior ao esperado, o que pode ser atribuído à simplificação da simulação matemática. A modelagem utilizada não captura todas as nuances da dinâmica real do CPAP, o que limita a capacidade do agente de aprender uma política eficaz de controle. A figura 7 mostra que, mesmo com os hiperparâmetros otimizados, o controle da pressão não foi satisfatório.

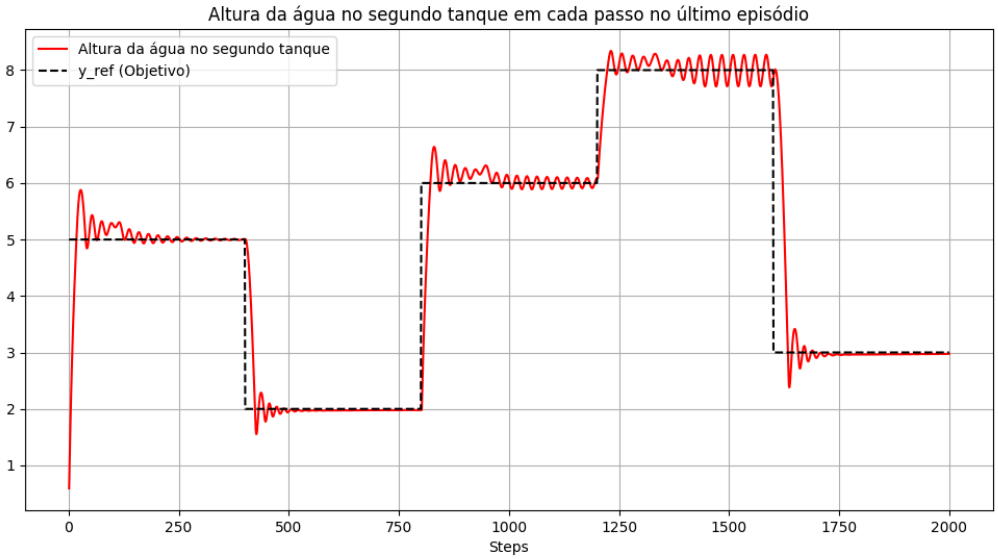

Figura 6: Gráfico da altura da água em cm no segundo tanque em cada passo. $K_p = 9.31$, $K_i = 0.015$, 42 neurônios por camada, ativação tanh, divisão dos inputs nas duas primeiras camadas. O resto dos parâmetros podem ser consultados na tabela 2.

7 Discussão dos resultados

O ambiente *Cascade Water Tank* foi utilizado para replicar os resultados apresentados em [11]. No entanto, a combinação do PIME com TD3 não apresentou a mesma estabilidade e robustez descritas no estudo original.

Dificuldades em replicar experimentos são comuns na literatura científica, especialmente quando os detalhes da implementação não são completamente descritos [18]. Neste caso, a ausência do código-fonte de [11] representou um desafio significativo, tornando o processo de reprodução dos resultados mais trabalhoso. A disponibilização do código-fonte original seria uma boa prática, pois facilitaria a replicação e possibilitaria comparações mais precisas entre abordagens.

Para obter um desempenho adequado, foi necessário um grande esforço de ajuste manual, testando diferentes combinações de hiperparâmetros e estruturas de rede. Foram exploradas diversas variações, incluindo:

- Ajuste do número de neurônios por camada, variando de 6 a 60;
- Testes com diferentes funções de ativação, como ReLU, tanh e ausência de ativação na última camada;
- Modificação da estrutura da rede, incluindo normalização em lote (*batch normalization*) e rede sem ramificação;
- Atualizar ou não os *biases* da rede durante o treinamento;
- Modificação dos limites da ação do agente, variando entre [-1, 1] e [-30, 30];
- Testes com a soma do PID antes e depois da amostragem da ação pela rede neural.
- Diferentes combinações de entrada do vetor x da rede neural.

Apesar desse esforço, o TD3 não foi capaz de refinar significativamente os valores do PID, gerando instabilidade no ambiente *Cascade Water Tank*. Além disso, observou-se que redes com mais de 60 neurônios por camada aumentavam drasticamente o tempo de treinamento, tornando inviável uma busca exaustiva por hiperparâmetros com uma grande quantidade de neurônios.

Mudando o foco para o ambiente CPAP, os desafios são ainda mais evidentes devido à complexidade inerente ao controle de sistemas não lineares. O filtro passa-baixa colocado tem o papel de evitar a instabilidade nos primeiros momentos da simulação da fase de expiração. Sem o filtro, o PID desestabiliza o ambiente, não guiando o agente RL e colocando a pressão em valores cada vez mais distantes do *setpoint*. Diferentemente do *Cascade Water Tank*, que é um sistema relativamente bem compreendido e com equações dinâmicas conhecidas, o CPAP envolve interações

Figura 7: Gráfico da pressão em cmH2O. $K_p = 10.0$, $K_i = 1.2$, 48 neurônios por camada, ativação tanh, divisão dos inputs nas duas primeiras camadas. O resto dos parâmetros podem ser consultados na tabela 2.

complexas entre o ventilador e o sistema respiratório do paciente. Pequenos erros na modelagem podem levar a grandes discrepâncias na resposta do controlador, exigindo um modelo matemático mais realista para que o agente RL possa ser treinado adequadamente.

Outro desafio desse estudo é a ausência de um laboratório para validar os resultados em um equipamento CPAP real. Embora a simulação seja uma etapa fundamental para evitar riscos ao paciente e reduzir custos experimentais, a validação em hardware real é a próxima etapa para comprovar a eficácia do controlador em um ambiente clínico.

Essas descobertas indicam possíveis direções para análises futuras. Investigações adicionais poderiam explorar:

1. Modelar de forma mais precisa e realista o *ventilator* e *lung* de forma que os sinais de pressão, fluxo e volume da simulação correspondessem com precisão aos dados clínicos reais (ver figura 8).
2. Conduzir experimentos em ambientes reais para validar a metodologia e transformar o controlador em um modelo confiável.
3. Melhores formas de modelar o impacto de distúrbios respiratórios. Talvez usando um *dataset* contendo dados reais de medições de pressão e fluxo de ar possa ser usado para calcular o distúrbio e modelar uma função que replica esses distúrbios.
4. A eficácia de outras formas de combinar o PID com algoritmos RL, a exemplo de [12].
5. Investigar modificações na arquitetura da rede neural do PIME-PPO e PIME-TD3, bem como na estrutura do vetor de *inputs*, visando aprimorar a capacidade de aprendizado e desempenho do modelo.

8 Conclusão

O principal objetivo deste estudo foi verificar se o PIME-TD3 é um algoritmo adequado para o controle do sistema CPAP, que possui características não lineares. Para isso, modelos matemáticos foram desenvolvidos para representar a dinâmica do CPAP, e o algoritmo PIME-TD3* foi implementado e testado. Todo o código-fonte utilizado neste trabalho foi disponibilizado publicamente no GitHub(*link do código fonte*).

Os experimentos mostraram que, embora o TD3 tenha apresentado um desempenho razoável no ambiente *Cascade Water Tank*, seu uso no CPAP foi limitado pela simplificação da modelagem matemática. Isso ressalta a necessidade de um modelo mais realista para que o aprendizado por reforço seja eficaz nesse tipo de aplicação.

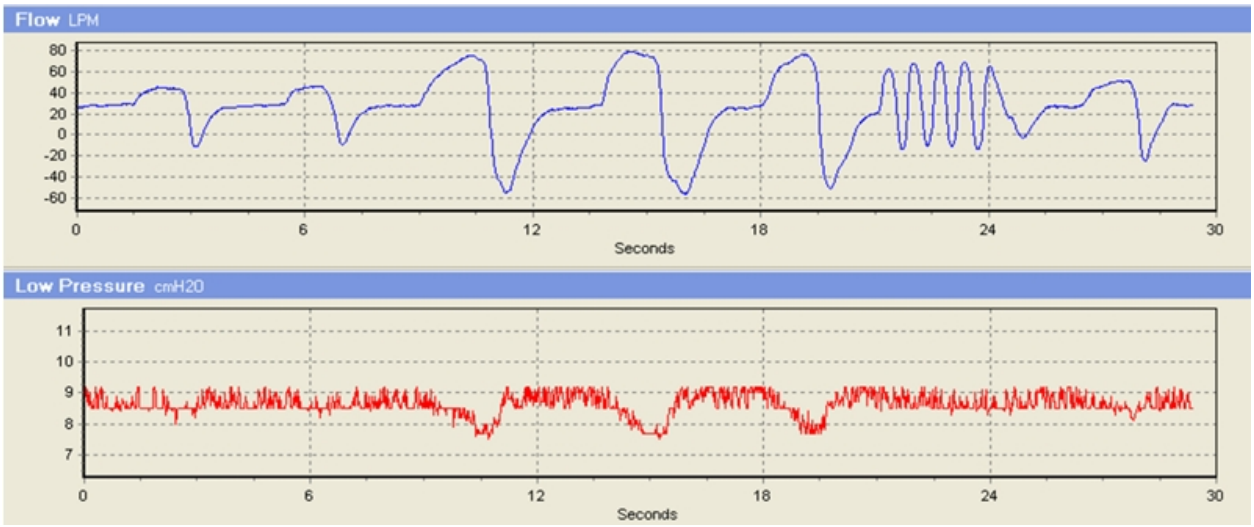

Figura 8: Exemplo de gráficos de fluxo e pressão gerados por pacientes e equipamentos reais em teste de laboratório [5]. A pressão está variando entorno do *setpoint* 8.5 cmH2O devido ao controlador *feedforward* e PID utilizados.

Além disso, este estudo reforça a importância da replicabilidade em pesquisas envolvendo aprendizado por reforço, evidenciando as dificuldades encontradas na tentativa de reproduzir resultados anteriores sem acesso ao código-fonte original.

Trabalhos futuros podem explorar modelos mais realistas do CPAP, realizar experimentos em hardware real e testar diferentes abordagens para integrar aprendizado por reforço com técnicas clássicas de controle.

9 Author contributions

Victor G. T. Oliveira and Pablo S. conducted the literature review and wrote the manuscript. Victor G. T. O. developed the code implementation under the supervision of Pablo S. Pablo S. designed the methodology, and Hatus V. W. developed the mathematical model for the CPAP system.

References

[1] Sara Maria Brancato, Davide Salzano, Francesco De Lellis, Davide Fiore, Giovanni Russo, and Mario di Bernardo. In vivo learning-based control of microbial populations density in bioreactors. In *6th Annual Learning for Dynamics & Control Conference*, pages 941–953. PMLR, 2024.

[2] Ruan de Rezende Faria, Bruno Didier Olivier Capron, Argimiro Resende Secchi, and Maurício Bezerra de Souza Jr. Gas-lift optimization using physics-informed deep reinforcement learning. *Industrial & Engineering Chemistry Research*, 63(32):14199–14210, 2024.

[3] Aaron E. Sher, Kenneth B. Schechtman, and Jay F. Piccirillo. The Efficacy of Surgical Modifications of the Upper Airway in Adults With Obstructive Sleep Apnea Syndrome. *Sleep*, 19(2):156–177, 03 1996.

[4] Renata C Di Francesco, Paula Andreya Junqueira, Ronaldo Frizzarini, and Fabio Elias Zerati. Crescimento pôndero-estatural de crianças após adenoamigdalectomia. *Revista Brasileira de Otorrinolaringologia*, 69:193–196, 2003.

[5] Zheng-Long Chen, Zhao-Yan Hu, and Hou-De Dai. Control system design for a continuous positive airway pressure ventilator. *Biomedical engineering online*, 11(1):1–13, 2012.

[6] Torbjörn Wigren. Integration of feedback and feed-forward techniques in reinforcement learning. *Technical Reports from the Department of Information Technology, Uppsala University*, 2023.

[7] Brian Douglas. Control systems lectures - closed loop control. YouTube video, 2012. Accessed on 27 November 2023.

[8] Shweta Goyal, Amalraj Shankar, Kailash Chandra Das, Akhilesh Singh, Sanjay Oli, and Madan Mohan Sati. Manual and adaptive tuned pid controllers for industrial application. In *2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pages 1953–1958, 2024.

[9] Peter Norvig Stuart Russell. Inteligencia artificial tradução da 3ª edição, 2013.

[10] M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

[11] Ruoqi Zhang, Per Mattsson, and Torbjörn Wigren. Robust nonlinear set-point control with reinforcement learning. In *2023 American Control Conference (ACC)*, pages 84–91. IEEE, 2023.

[12] T Shuprajhaa, Shiva Kanth Sujit, and K Srinivasan. Reinforcement learning based adaptive pid controller design for control of linear/nonlinear unstable processes. *Applied Soft Computing*, 128:109450, 2022.

[13] Yecheng Jason Ma, William Liang, Hung-Ju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. Dreureka: Language model guided sim-to-real transfer, 2024.

[14] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019.

[15] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Farama gymnasium. *arXiv preprint arXiv:1606.01540*, 2016.

[17] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902, 2019.

[18] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice, 2022.