Sintonia de Controladores PI baseada em *Augmented Random Search*: Estudo de Caso do Processo CSTR

Santino M. Bitarães [*] **Moises T. Silva** [**] **Thiago A.M. Euzébio** [**]

[*] *Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração, Universidade Federal de Ouro Preto e Instituto Tecnológico Vale, Ouro Preto, MG, (e-mail: santino.bitaraes@aluno.itv.org).*
[**] *Instituto Tecnológico Vale, Ouro Preto, MG, (e-mail: moises.silva@pq.itv.org, thiago.euzebio@itv.org)*

Abstract: The ARS algorithm is a reinforcement learning method that seeks to map the best actions to the process operating conditions. Initially, the algorithm has no instructions and no knowledge of the process dynamics. Thus, this technique seeks to learn while interacting with the process and the search for the best actions is guided only by a numerical reward signal. The continuous stirring tank (CSTR) process, simulated in Python, is used as an interaction environment for the ARS algorithm. The main objective of this work is to apply the ARS algorithm to tune a PI controller of the CSTR process. The states are the setpoint values applied to the process before and after its variation. Actions are the PI controller parameters for each reference set (states). The reward was defined as the inverse of the sum of the error module. The tunings proposed by the ARS are 8.3% (same operating point) better than the tuning benchmark chosen for comparison.

Resumo: O algoritmo ARS é uma método de aprendizagem por reforço que busca mapear as melhores ações para as condições de operação do processo. Inicialmente o algoritmo não possui instruções e nem algum conhecimento da dinâmica do processo. Assim, essa técnica busca o aprendizado enquanto interage com o processo e a busca pelas melhores ações é norteada somente por um sinal numérico de recompensa. O processo do tanque de agitação contínua (CSTR), simulado em *Python*, é utilizado como ambiente de interação do algoritmo ARS. O principal objetivo desse trabalho é aplicar o algoritmo ARS para sintonizar um controlador PI do processo CSTR. Os estados são os valores dos *setpoint* aplicado ao processo antes e após sua variação. As ações são os parâmetros do controlador PI para cada conjunto de referência (estados). A recompensa foi definida como o inverso do somatório do módulo do erro. As sintonias propostas pelo ARS são 8,3% (mesmo ponto de operação) melhores que o *benchmark* de sintonia escolhido para comparação.

Keywords: Artificial Intelligence, Reinforcement Learning, PI control, CSTR.
Palavras-chaves: Inteligência Artificial, Aprendizagem por reforço, Controlador PI, CSTR.

1. INTRODUÇÃO

A aprendizagem por reforço é uma área que vem apresentando grande relevância no meio científico. Esta técnica é uma área de pesquisa ativa dentro da inteligência artificial. A sua origem é dada na pesquisa operacional e na ciência da computação para resolver problemas de tomada de decisão sequencial. Trata-se de uma técnica que mapeia ações para situações de modo a obter uma maior recompensa do ambiente. Inicialmente o algoritmo não possui instruções de quais ações tomar dadas as situações, dessa forma, a técnica de aprendizagem por reforço deverá descobrir quais são as melhores ações que geram maior recompensa, após experimentá-las (Spielberg et al., 2019) (Sutton and Barto, 2018).

De acordo com Wiering and van Otterlo (2012) a aprendizagem por reforço é uma classe geral de algoritmos de aprendizado de máquina que visa fazer com que o algoritmo tome ações em um ambiente onde o único *feedback* consiste em um sinal de recompensa escalar. O objetivo do agente é realizar ações que maximizem o sinal de recompensa no longo prazo.

Diversos trabalhos na literatura aplicaram algoritmos de aprendizagem por reforço para sintonia de controladores Proporcional-Integral-Derivativo (PID). Em Howell and Best (2000) é automatizado o ajuste de um motor Zetec da Ford Motors. O algoritmo, denominado *Continuous Action Reinforcement Learning Automata* (CARLA), foi usado para ajustar controladores PIDs após os parâmetros iniciais serem definidos usando métodos como Ziegler-Nichols. Os resultados mostraram uma redução de 60% na função custo após o ajuste da aprendizagem por reforço. Por outro lado, um algoritmo de aprendizagem por reforço chamado de Ator-Critico foi usado por song WANG et al. (2007) para ajustar os parâmetros PID de maneira adaptativa em um sistema complexo e não linear. Quando comparado com um controlador PID convencional, os resultados da simulação mostram que o controlador proposto é eficiente para sistemas não lineares complexos, adaptável e robusto.

O trabalho de Koszaka et al. (2006) utiliza uma abordagem de aprendizado por reforço baseado em *Q-learning* para ajuste adaptativo de um controlador PID. Os processos controlados foram simulados como funções de transferência no Matlab. Foi demonstrado que, para uma função de recompensa simples, o algoritmo proposto forneceu soluções satisfatórias e manteve o sobressinal e erro de estado estacionário sob restrições estabelecidas.

Em Brujeni et al. (2010) foi usado o algoritmo SARSA (*State-Action-Reward-State-Action*) para ajustar dinamicamente um controlador PI usado para controlar o processo CSTR. O agente foi primeiro pré-treinado em um modelo estimado para o processo e, em seguida, foi implementado para ajustar continuamente o aquecedor do tanque on-line. O agente teve como objetivo rejeitar distúrbios e rastrear o ponto de operação. Ao final, os autores compararam o desempenho do controlador sintonizado pelo algoritmo SARSA com os métodos de ajuste de controle do modelo interno. Verificou-se que algoritmo SARSA foi o método de ajuste superior devido à sua natureza adaptativa contínua. Em contraste, o artigo de Reddy et al. (2020) propõe um estimador de modelo polinomial recursivo do controlador PI. A proposta foi projetar o controlador para o reator químico simulado com objetivo de manter o pH do produto e nível de reagente. Os resultados da simulação mostram um bom desempenho do controlador PI adaptativo.

Huang et al. (1982) propõe um sistema de controle adaptativo para o controle do processo CSTR. O sistema é projetado para ser adaptado às mudanças de parâmetros devido à variação das condições de operação e também às mudanças de concentração não mensuráveis na entrada. A adaptação dos parâmetros foi feita por sensibilidades no estado estacionário. Através dos experimentos realizados em um reator tanque agitado parcialmente simulado, o autor apresentou resultados do controle adaptativo proposto para o processo CSTR bastante satisfatórios. Para o controle de temperatura de um reator, Kumar et al. (2020) apresenta dois controladores: controle PID e controle preditivo baseado em modelo (MPC). O controle efetivo da temperatura foi obtido pelo uso do modelo de controle preditivo em termos de estabilidade, eliminando o efeito de perturbação e rastreamento de referência.

Neste artigo, o algoritmo de aprendizagem por reforço ARS proposto por Mania et al. (2018) é aplicado para sintonizar um controlador PI. Este algoritmo é usado em uma camada de controle avançado para definição dos parâmetros do controlador PI de um processo CSTR em diferentes pontos de operação. A principal motivação para uso do algoritmo ARS deve-se a sua simples implementação, possibilitando seu uso em CLPs e SDCDs. De acordo com a revisão da literatura, não há estudos sobre o uso de ARS para sintonia de controladores em processos contínuos.

Esse trabalho está organizado da seguinte forma. Na Seção 2 são apresentados os fundamentos da aprendizagem por reforço, enquanto na Seção 3 é feito um descritivo do algoritmo ARS, já na Seção 4 é apresentado detalhes do processo CSTR. Na Seção 5 é descrito a metodologia utilizada para sintonia do controlador PI usando ARS. Os resultados e conclusões são apresentadas nas Seções 6 e 7, respectivamente.

2. FUNDAMENTOS DE APRENDIZAGEM POR REFORÇO

A estrutura básica da técnica de aprendizagem por reforço é apresentada na Figura 1. Esta técnica é baseada no aprendizado adquirido ao decorrer da iteração entre um agente e um ambiente em uma sequência de passos de tempo discretos, $(t = 1,2,3,\dots)$. A cada passo de tempo t, o agente recebe do ambiente um estado $s_t \in S$, onde S é um conjunto de possíveis estados, com base nesse estado seleciona uma ação $a_t \in A$, onde A é um conjunto de ações possíveis e aplica no ambiente. O ambiente retorna uma recompensa numérica r_t, que indica o quão bom foi a ação tomada, e um novo estado s_{t+1}. A cada passo de tempo o agente faz um mapeamento dos estados de acordo com as probabilidades de selecionar cada possível ação a_t. O agente aprende como melhor interagir com o ambiente pela maximização das recompensas que recebe.

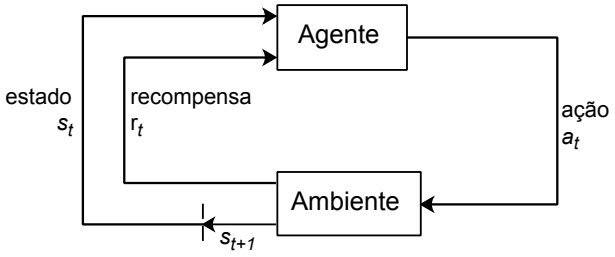

Figura 1. Estrutura básica da técnica de aprendizagem por reforço.

De acordo com Sutton and Barto (2018) os principais elementos da aprendizagem por reforço são: o agente, a ação, o estado, a recompensa e o ambiente. Estes elementos são descritos a seguir.

- **Agente**: O agente consiste no algoritmo que toma ações e aprende. O agente escolhe uma ação a_t em cada estado s_t, e as percepções que o agente obtém do ambiente são o estado do ambiente após cada ação e o sinal de recompensa escalar r_t em cada passo.
- **Ambiente**: O ambiente é tudo o que o agente não pode controlar. Em outras palavras, o ambiente é o processo em que o agente interage.
- **Ações**: As ações são o conjunto de formas que o agente pode interagir com ambiente. Esse conjunto pode ser composto de valores numéricos ou comandos direcionais, por exemplo.
- **Estados**: Os estados são as possíveis percepções que o agente pode ter do ambiente. Por exemplo, quando o ambiente for um tabuleiro de xadrez, os estados serão as possíveis posições que as peças podem ocupar.
- **Recompensa**: A recompensa é um valor escalar que indica numericamente o quão bom foi para o agente tomar uma ação em cada passo de tempo. Analogamente, em um sistema biológico podemos pensar nas recompensas como experiência de prazer e dor. Em geral, os sinais de recompensa podem estar em função do estado, do ambiente e das ações realizadas.
- **Política**: Uma política, denotada por π, é uma regra pela qual o agente seleciona ações em função dos estados, assim definindo o comportamento do agente. Sendo o núcleo de um agente de aprendizagem por reforço no sentido de que por si só é suficiente para determinar

o comportamento. A política pode ser determinística ou estocástica, na determinística é feito um mapeamento de cada estado para uma ação enquanto na estocástica é feito um mapeamento de cada estado para uma probabilidade de selecionar cada ação possível. A política pode ser uma tabela de pesquisa, uma função simples, um método de busca e até mesmo uma rede neural.

A aprendizagem por reforço possui dois conjuntos de métodos de solução: O conjunto dos métodos de solução tabular e o conjunto dos métodos de solução aproximada. Os métodos de solução tabular consideram um espaço finito de estados e ações na construção da aprendizagem por reforço e todas as combinações de ações, estados e a política precisa ser armazenada, na forma de tabela, para estar sempre disponível para consulta durante as iterações. Enquanto os métodos de solução aproximada são capazes de considerar um espaço infinito de estados e ações na maioria dos casos. O conhecimento (política) é armazenado na forma parametrizada, seja por uma, tabela parametrizada, função parametrizada ou uma rede neural profunda.

3. O ALGORITMO *AUGMENTED RANDOM SEARCH* (ARS)

O algoritmo *Augmented Random Search (ARS)* propõe um método de aprendizagem por reforço livre de modelo mais simples de implementação. O método é proposto por Mania et al. (2018) e aborda um novo método de aprendizagem por reforço baseado em busca aleatória simples com melhorias no processamento das recompensas e dos estados, baseadas em heurísticas aplicadas com sucesso na aprendizagem por reforço profunda. A proposta desse método surgiu devido aos outros métodos necessitarem de muitos dados para atingir um bom desempenho e apresentarem implementação complexa.

Para obter o método livre de modelo, duas abordagens foram combinadas para se criar o método ARS, sendo elas: o método proposto por Salimans et al. (2017), na qual a política é otimizada sem uso de derivadas e o método proposto por Rajeswaran et al. (2018), o qual mostra que as políticas lineares podem ser treinadas por meio de gradiente de política simples e obter desempenho competitivo com políticas de rede neural complexas. O pseudocódigo do algoritmo ARS é apresentado no Algoritmo 1.

Algoritmo 1 : ARS

Hiperparâmetros:

$n \leftarrow 2$ *número de entradas*
$p \leftarrow 2$ *número de saídas*
$\alpha \leftarrow 0,3$ *taxa de aprendizagem*
$\upsilon \leftarrow 0,2$ *ruído de exploração*
passos $\leftarrow 3000$ *número de iterações*
$N \leftarrow 16$ *número de direções por direção de ajuste*
$b \leftarrow 8$ *número de direções com os melhores desempenhos*

Inicializa:

$M_0 \leftarrow 0 \in R^{p \times n}$ *matriz de pesos da perceptron*
$\mu_0 \leftarrow 0 \in R^n$ *média das entradas*
$j \leftarrow 0$
Para $j \leftarrow 0$ **até** *steps* **Faça**
    Crie as matrizes $\delta_1, \delta_2, \ldots, \delta_N$ em $R^{p \times n}$ preenchidas com valores uniformes entre 0 e 1.
    Crie dois vetores $r(\pi_{j,k,-})$ e $r(\pi_{j,k,+})$ em R^N vazios.
    Para $k \leftarrow 0$ **até** N **Faça**
        $r(\pi_{j,k,-}) \leftarrow$ recompensa de $\pi_{j,k,-}(x) = (M_j - \upsilon\delta_k)$ diag $(\sum_j)^{-\frac{1}{2}}(x - \mu_j)$.
        $r(\pi_{j,k,+}) \leftarrow$ recompensa de $\pi_{j,k,+}(x) = (M_j + \upsilon\delta_k)$ diag $(\sum_j)^{-\frac{1}{2}}(x - \mu_j)$.
    Fim Para
    Ordena as direções δ_k pela máxima recompensa em $\{r(\pi_{j,k,+}), r(\pi_{j,k,-})\}$, sendo então $\pi_{j,(k),+}$ e $\pi_{j,(k),-}$ as políticas respectivas.
    Atualiza
    $M_{j+1} = M_j + \frac{\alpha}{b\sigma^R}\sum_{k=1}^{b}[r(\pi_{j,k,+}) - r(\pi_{j,k,-})]\delta_k$, onde σ^R é o desvio padrão das recompensas usadas na etapa de atualização.
    Configura $\mu_{j+1}, \sum_{j+1}$ para ser a média e covariância dos estados $2NH(j+1)$ encontrados desde o início do treinamento.
    $j \leftarrow j+1$
Fim Para

4. DESCRIÇÃO DO PROCESSO CSTR

O processo sob estudo consiste em um reator de agitação contínua (CSTR). O trabalho de Antonelli and Astolfi (2003) descreve o processo CSTR em detalhes. Além disso, afirma que esse tipo de reator é amplamente utilizado na indústria.

Um reator é usado para converter um produto químico **A** em um produto químico útil **B** em um fluxo de resíduos antes de entrar em processos subsequentes. A Figura 2 apresenta um esquema do reator CSTR.

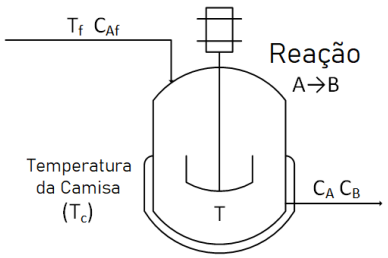

Figura 2. Representação simplificada do reator CSTR.

O processo considerado é modelado dinamicamente como um Reator de Agitação Contínua com um mecanismo cinético simplificado que descreve a conversão do reagente **A** em produto **B** com uma reação irreversível e exotérmica. Como o analisador do produto **B** não é rápido o suficiente para controle em tempo real, é desejável manter a temperatura em um ponto de ajuste constante que maximize o consumo de **A** (temperatura mais alta possível). Assim, faze-se necessário ajustar a temperatura da camisa (T_c) para manter a temperatura desejada do reator e minimizar a concentração do reagente **A**. A temperatura do reator nunca deve exceder 400 K. A temperatura

da camisa de resfriamento pode ser ajustada entre 250 K e 350 K. As equações do modelo desse processo são apresentados a seguir e seus parâmetros são exibidos na Tabela 1.

$$\dot{C_A} = \frac{q}{V}(C_{AF} - C_A) - k_0 C_A e^{-(E/RT)}$$
$$\dot{T} = \frac{q}{V}(T_f - T) - \frac{\Delta H K_0}{\rho C_p} C_A e^{-(E/RT)} + \frac{\rho_c C_{pc}}{\rho C_p V} q_c (1 - e^{-(\frac{h_A}{\rho_c C_{pc} q_c})})(T_{cf-T}).$$

Tabela 1. Parâmetros do processo CSTR Hedengren (2021).

Parâmetro	Notação	Valor
Vazão processo	q	100 m^3/s
Concentração alimentação	C_{Af}	0,877 mol/m^3
Temperatura alimentação	T_f	350 K
Temperatura entrada refrigerante	T_{cf}	350 K
Volume do reator	V	100 m^3
Coef. transferência térmica	h_A	7×10^5 cal/ min/ K
Taxa de reação constante	k_0	$7,2 \times 10^{10}$ min^{-1}
Energia de ativação	E/R	8750 K
Reação de calor	ΔH	-5×10^4 J/ mol
Densidade líquidos	ρ , ρ_c	1×10^3 kg/m^3
Calor específico	C_p,C_{pc}	0,239 J/kg/K
Limite inferior da Temp. da Camisa	$T_{c_{min}}$	250 K
Limite superior da Temp. da Camisa	$T_{c_{max}}$	350 K

5. ALGORITMO ARS APLICADO AO PROCESSO CSTR

O processo CSTR é um sistema não linear e multivariável. Neste estudo, considera-se um controlador PI que apenas atua na temperatura da camisa (T_c) para ajustar a temperatura do reator (T). As interações das outras variáveis não são consideradas. O controle do processo é realizado através de um controlador PI, dado por:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau$$

Os parâmetros do controlador PI são definidos pelo algoritmo ARS para cada referência. O diagrama de blocos do sistema de controle é apresentado na Figura 3.

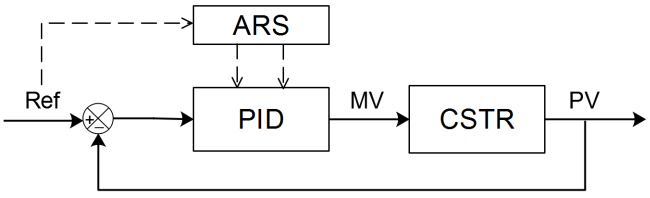

Figura 3. Diagrama de blocos do sistema de controle.

Um episódio de interação do algoritmo ARS com o processo foi definido como um conjunto de referências aplicadas ao processo. No início de cada episódio o processo é redefinido para as suas condições iniciais. A cada passo do episódio o algoritmo observa a referência atual e a próxima referência como o estado do ambiente e propõe para esse estado uma ação (parâmetros do controlador PI).

Cada ação é um conjunto de parâmetros de sintonia do controlador PI contendo os parâmetros K_p e K_i , $\in R_+$. O ambiente então executa a mudança de referência com os parâmetros por um tempo t e então retorna ao algoritmo uma recompensa e os próximos estados. Uma vez que o objetivo de controle é melhorar o rastreamento da referência define-se a recompensa como o inverso do somatório do módulo do erro. Logo, a recompensa é dada por:

$$Reward = \frac{1}{\sum |erro_t|}$$

A escolha da função de recompensa infere nas especificações da resposta do sistema de controle, pois esta indica o quão bom foi a ação de controle obtida a partir do algoritmo. Um resumo com os hiper-parâmetros considerados são apresentados na Tabela 2.

Tabela 2. Resumo dos hiper-parâmetros do algoritmo ARS.

Hiper-parâmetros	Valor
Estados	$[SP_1, SP_2;\ SP_2, SP_3;\ SP_3, SP_4;\ SP_4, SP_{n+1}]$
Ações	$[K_{p1}, K_{i1};\ K_{p2}, K_{i2};\ K_{p3}, K_{i3};\ K_{pn}, K_{in}] \in R_+$
Recompensa	$\frac{1}{\sum \lvert erro_t \rvert}$
Taxa de aprendizagem	0,3
Ruído de exploração	0,2
N Direções	16
b Melhores Direções	8
Iterações	3000
Passos por episódio	4

O algoritmo em *Python*, desenvolvido por Hedengren (2021) para o controle do processo CSTR, foi adaptado para permitir a implementação da estratégia de controle baseado no algoritmo ARS. A cada iteração do algoritmo ARS, o ambiente executa por 80 passos com a novo referência e a respectiva sintonia proposta. O algoritmo completo foi executado na plataforma online *Google Colaboratory*.

6. RESULTADOS E DISCUSSÕES

Nesta seção são apresentados os resultados da sintonia dos controladores PI para o processo CSTR usando o algoritmo ARS. O algoritmo completo foi executado por 3 mil iterações e em cada iteração o episódio foi executado por 32 (2×16) direções de ajuste, totalizando 96 mil execuções completas do processo. O gráfico com o histórico de aprendizagem é apresentado na Figura 4.

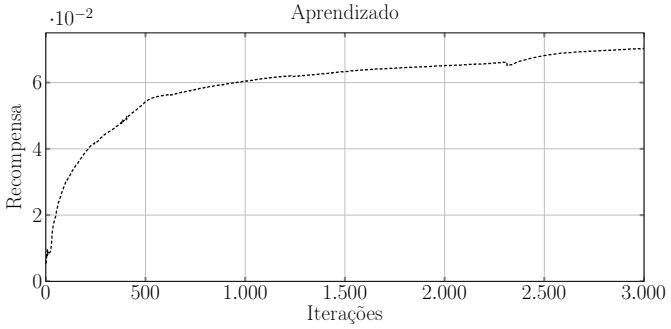

Figura 4. Aprendizado do ARS.

Conforme exibido na Figura 4 o algoritmo convergiu após 2500 iterações e teve o tempo de execução de 63 minutos. Observe que os parâmetros do controlador PI são calculados offline. Então, a sintonia que fornece a melhor recompensa a partir do algoritmo ARS é usada para avaliar o controlador PI projetado.

As métricas utilizadas para avaliação dos resultados são o IAE (integral do valor absoluto do erro) e o SII (Soma dos incrementos absolutos da entrada) são apresentadas nas equações abaixo.

$$IAE = \sum_{t=1}^{T} |e(t) - e(t-1)|$$

$$SII = \sum_{t=1}^{T} |u(t) - u(t-1)|$$

6.1. *Comparativo entre pontos de operações iguais*

A sintonia do controlador PI usada como *benchmark* foi proposta por por Hedengren (2021) para o processo CSTR no ponto de operação entre 300 e 320 K. O autor utilizou o método de sintonia de Controle por Modelo Interno (IMC) para definir os parâmetros do controlador PI. Os parâmetros de sintonia proposto por Hedengren (2021) e os parâmetros definidos pelo algoritmo ARS para o mesmo ponto de operação são apresentados na Tabela 3.

Tabela 3. Parâmetros de sintonia do controlador.

	Referência	K_p	K_i
Hedengren	300 - 320	4,62	40,44
ARS	300 - 320	50,39	7,37

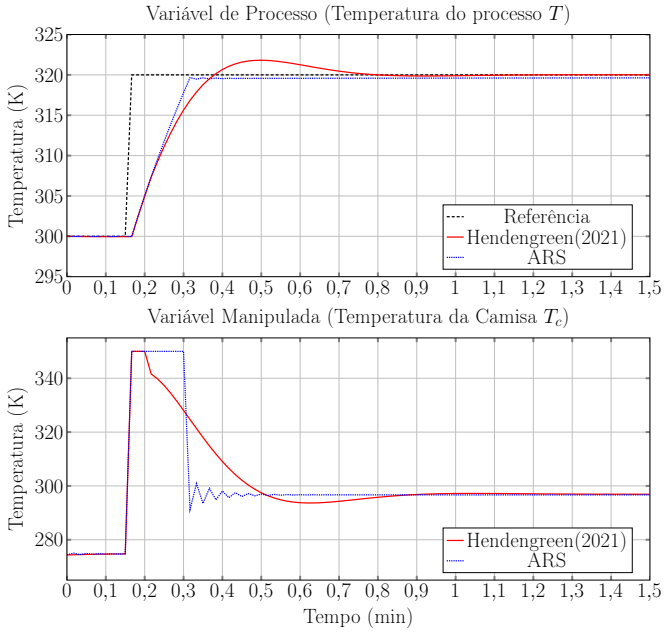

Figura 5. Variáveis do Processo CSTR.

A Figura 5 apresenta a saída do processo (T) e ação de controle (T_c) obtidas com as sintonias listadas na Tabela 3. O processo CSTR quando controlado por um PI sintonizado por Hedengren (2021) apresentou um sobressinal de 9% ($t = 0,5min$), tempo de subida (t_s) de $0,17min$, tempo de acomodação ($t_{ac(5\%)}$) de $0,47min$ e não apresentou erro em regime permanente (e_∞), enquanto a sintonia do controlador PI obtida a partir do algoritmo ARS não apresentou sobressinal, tempo de subida de $0,12min$, tempo de acomodação de $0,15min$ e erro em regime permanente de $0,4$. A variável manipulada T_c apresentou saturação em ambas as sintonias. Porém, a sintonia de Hedengren (2021) apresentou 2,4 segundos de saturação enquanto a sintonia proposta pelo algoritmo ARS apresentou 8,4 segundos de saturação. Um tempo maior de saturação da sintonia proposta pela ARS está relacionada com a escolha da função de recompensa.Uma vez que o objetivo de controle escolhido foi rastrear a variação de *setpoint* o mais rápido. Um comparativo entre os índices são apresentados na Tabela 4.

Tabela 4. Índices de Desempenho.

Método	IAE	SII	%OS	$t_s(min)$	$t_{ac}(min)$	e_∞
Hedengren	136,50	631,9	9	0,17	0,47	0
ARS	125,11	570,3	-	0,12	0,15	0,4

Os índices de desempenho obtidos com o algoritmo ARS foram 8,3% (IAE) e 9,75% (SII) melhores que os obtidos pela sintonia de Hedengren (2021). O algoritmo ARS apresentou melhores tempos de acomodação e subida.

6.2. *Avaliação do ARS em diferentes pontos de operação*

Agora, é avaliado o desempenho da sintonia do controlador PI obtida a partir do algoritmo ARS em diferentes pontos de operação. O seguinte conjunto de referências foi usada:

$$Ref = [300,\ 320,\ 330,\ 340,\ 350].$$

Tabela 5. Sintonia proposta pelo ARS.

Referência	K_p	K_i
300 - 320	50,39	7,37
320 - 330	52,40	30,21
330 - 340	21,47	27,06
340 - 350	36,79	56,52

O conjunto de ações, ou seja, as sintonias do controlador PI para cada referência são apresentadas na Tabela 5. Os valores das variáveis de processo e manipulada obtidos com as sintonias são apresentados na Figura 6.

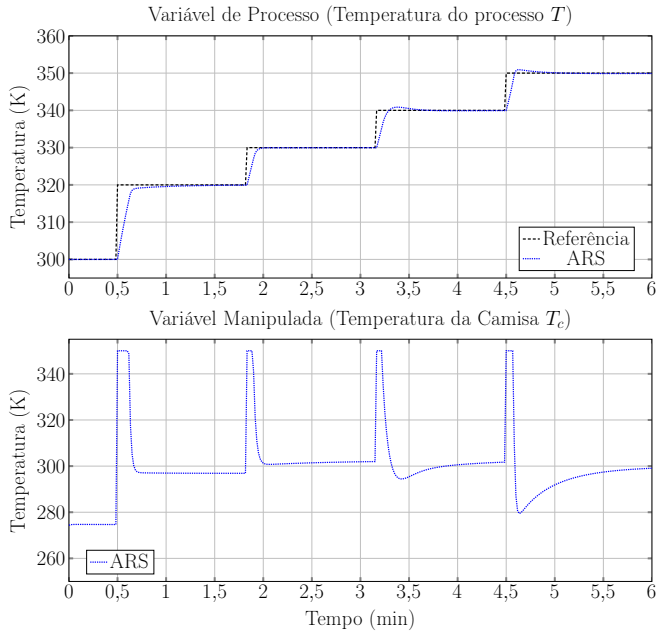

Figura 6. Variáveis do Processo CSTR em vários pontos de operação.

As sintonias propostas pelo algoritmo ARS apresentaram baixos sobressinais, apenas no terceiro e quarto degrau. A variável manipulada (T_c) apresentou saturação em todos pontos de operação, porém apresentou uma dinâmica suave em todos pontos de operação, principalmente na ultima referência aplicada. Neste caso, o índice IAE é igual 264,92.

7. CONCLUSÕES

Neste artigo foi utilizado o algoritmo ARS para sintonizar um controlador PI do processo CSTR. A partir dos resultados de simulação observa-se que o algoritmo ARS apresentou melhor desempenho para o rastreamento da referência. Além disso, a MV apresentou menor variabilidade que a sintonia usada como *benchmark*. O uso do algoritmo ARS apresenta como principal vantagem a capacidade de aprender de forma independente do usuário a tarefa de sintonizar um controlador. A sua desvantagem são as inúmeras interações, o que implica na necessidade de alto poder de processamento e um tempo considerável para o pleno aprendizado. Em trabalhos futuros deseja-se aplicar perturbações ao processo e também analisar outras formulações da recompensa, bem como adicionar ruídos de medição na saída do processo.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001; do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ), números 402759/2018-4 e 444425/2018-7; do Instituto Tecnológico Vale (ITV) e da Universidade Federal de Ouro Preto (UFOP).

Referências

Antonelli, R. and Astolfi, A. (2003). Continuous stirred tank reactors: easy to stabilise? *Automatica*, 39(10), 1817–1827.

Brujeni, L.A., Lee, J.M., and Shah, S.L. (2010). *Dynamic tuning of PI-controllers based on model-free reinforcement learning methods*. IEEE.

Hedengren, J.D. (2021). Temperature control of a stirred reactor. Disponível em: `https://apmonitor.com/pdc/index.php/Main/StirredReactor`. Acesso em: 19 de abril 2022.

Howell, M. and Best, M. (2000). On-line pid tuning for engine idle-speed control using continuous action reinforcement learning automata. *Control Engineering Practice*, 8(2), 147–154.

Huang, H.P., Chao, Y.C., and Cheng, C.L. (1982). Identification and adaptive control for a cstr process. In *1982 American Control Conference*, 551–556. doi:10.23919/ACC.1982.4787911.

Koszaka, L., Rudek, R., and Pozniak-Koszalka, I. (2006). An idea of using reinforcement learning in adaptive control systems. In *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*, 190–190. IEEE.

Kumar, U., Sharma, V., Rahi, O.P., and Kumar, V. (2020). Mpc-based temperature control of cstr process and its comparison with pid. In T. Sengodan, M. Murugappan, and S. Misra (eds.), *Advances in Electrical and Computer Technologies*, 1109–1115. Springer Singapore, Singapore.

Mania, H., Guy, A., and Recht, B. (2018). Simple random search provides a competitive approach to reinforcement learning.

Rajeswaran, A., Lowrey, K., Todorov, E., and Kakade, S. (2018). Towards generalization and simplicity in continuous control.

Reddy, B.R., Ram, K.T., and Pranavanand, S. (2020). Adaptive pi controller design and deployment for chemical reactor. In *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, 1210–1215. doi:10.1109/ICOSEC49089.2020.9215456.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning.

song WANG, X., hu CHENG, Y., and SUN, W. (2007). A proposal of adaptive pid controller based on reinforcement learning. *Journal of China University of Mining and Technology*, 17(1), 40–44.

Spielberg, S., Tulsyan, A., Lawrence, N.P., Loewen, P.D., and Bhushan Gopaluni, R. (2019). Toward self-driving processes: A deep reinforcement learning approach to control. *AIChE Journal*, 65(10).

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement learning: An introduction*. MIT press, London, England.

Wiering, M. and van Otterlo, M. (2012). *Reinforcement Learning: State-of-the-Art*. Springer, Berlin, Heidelberg.