Continuous Reinforcement Learning for Delayed Dynamical Systems

A Path Signature-Based Representation Approach

Joachim Jobard

Supervisors: Lionel Mathelin, Onofrio Semeraro, Erik Fransén & Rémy Hosseinkhan-Boucher
Examiner: Arvind Kumar
Opponent: Manato Chiba

KTH Master Thesis in Computer Science

Agenda

1. Motivation
2. Theoretical Background
3. Continuous Time Reinforcement Learning Algorithms
4. Experiments
5. Limitations and Future Work

Motivation

Controlling Delayed Differential Equations (DDEs)

Ubiquitous in:

- Chemical engineering
- Population dynamics
- Network control
- ...

	ODE	DDE
Form	$\dot{x} = f(t, x(t))$	$\dot{x} = f(t, x(t), x(t - \tau_1), ..., x(t - \tau_n))$
Example	$\dot{x} = -x(t)$	$\dot{x} = -x(t) + x(t - \tau)$

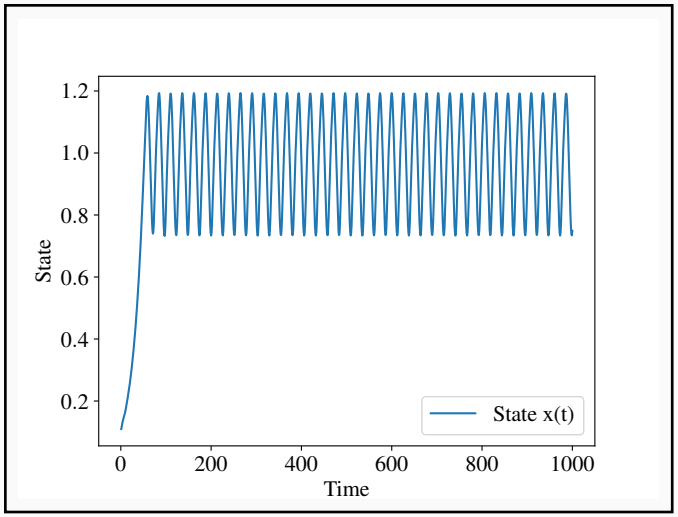
with $n \in \mathbb{N}$, $\tau_1, ..., \tau_n \in \mathbb{R}_+^*$

Problem: Usually no simple closed form solution to a DDE.

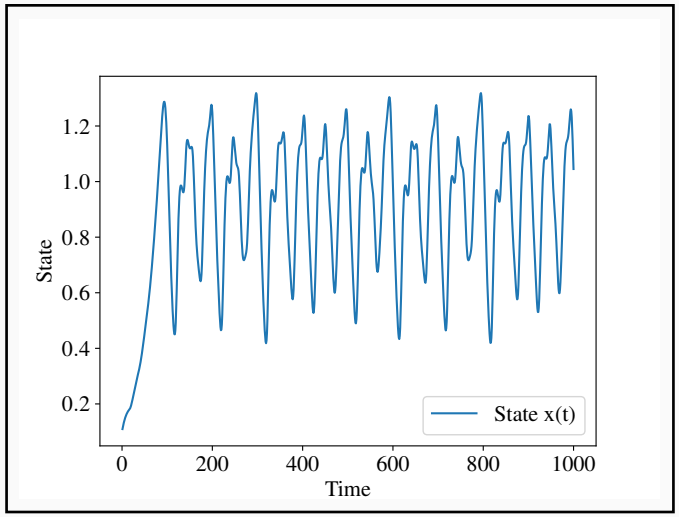
Uniqueness of solution: Up to an initial condition $x_0 : [-\tau_{\max}, 0] \to \mathbb{R}$.

Controlling Delayed Differential Equations (DDEs)

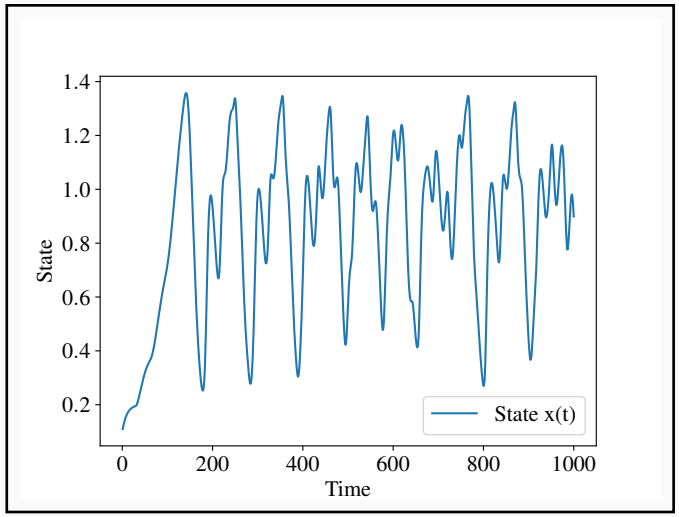
Mackey-Glass equation: $\forall t \in \mathbb{R}, \dot{x}(t) = -\mu x(t) + \frac{p x(t-\tau)}{1+x(t-\tau)^n}$ (n, μ and p fixed here). Used for modelling of healthy or pathological behaviour.



(a) $\tau = 8$



(b) $\tau = 17$



(c) $\tau = 30$

Figure 1: Mackey-Glass system for different delays (uncontrolled)

Controlling Delayed Differential Equations (DDEs)

Why are delayed systems complex to stabilize?

- Current position is not enough $\Rightarrow$ **we need knowledge of the past trajectory**
- RL framed as a Markov Decision Process ($x_{t+1} = f(x_t)$) $\Rightarrow$ **algorithms must be adapted**

Topic of this thesis: Stabilize delayed systems position to minimal energy fixed points (control)

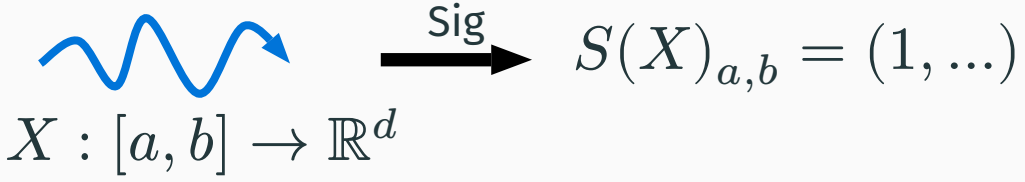
Research questions:

- Can we adapt RL algorithm to Non-Markovian dynamics?
- **How to represent the state?** (here the history function)

Theoretical Background

What is a Path Signature?

The signature of a path encodes its geometrical properties, it is a powerful feature-extractor that has been used in machine learning with promising results[1].



Input: a path $X \implies \mathcal{C}^1$ function from an interval to $\mathbb{R}^d$

Output: Infinite-size tensor with coefficients encoding the geometrical properties of the path

[1] Fermanian et al., "Framing RNN as a Kernel Method"; Kidger et al., "Deep Signature Transforms".

What is a Path Signature?

What are the coefficients?

- Take $(i_1, ..., i_N) \in \{1, ..., d\}^N$
- **Signature coefficient** associated to this tuple:

$$S_{[a,b]}^{(i_1,...,i_N)}(X) = \int_{a<u_1<...<u_n<b} ... \int \dot{X}_{i_1}(u_1)...\dot{X}_{i_N}(u_N)du_1...du_N$$

- Concatenate each coeff for $I \in \{1, ..., d\}^N$, $N \in \mathbb{N}^*$ in lexicographical order
- **Truncated Signature** (compression): stop at $N \leq \text{threshold}$

What is a Path Signature?

A more visual explanation Terms of depth (length of tuple) 1 are simply $S^{(i)}_{[a,b]}(X) = X^{(i)}(b) - X^{(i)}(a)$. Terms of depth 2 are represented in Figure 2[2]

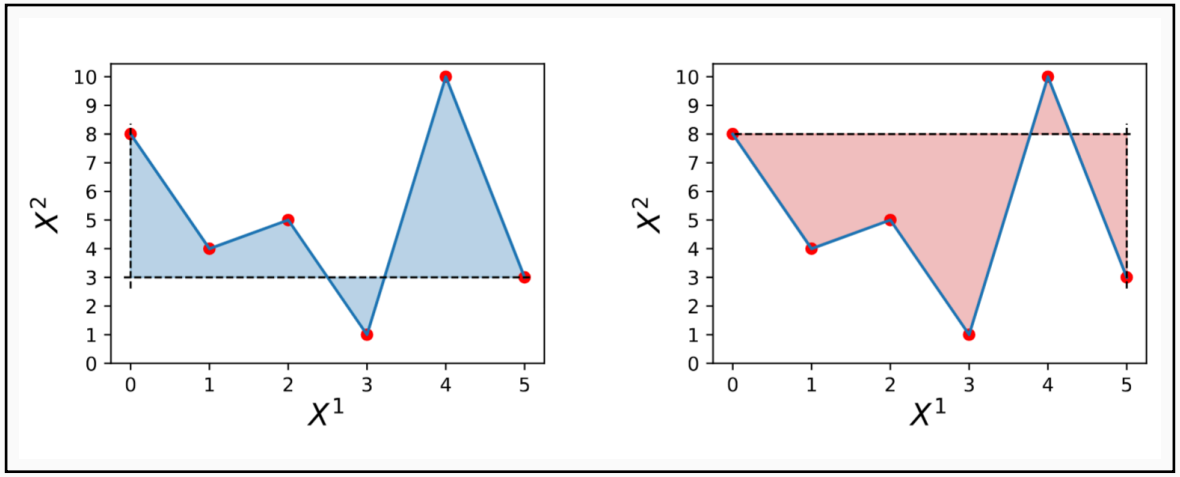


Figure 2: Signature representation for first terms $S^{(1,2)}(X)$ and $S^{(2,1)}(X)$

Deeper terms are harder to represent.

[2] Chevyrev and Kormilitzin, "A Primer on the Signature Method in Machine Learning".

What is a Path Signature?

A useful theorem[3]

For all $X : [a, b] \to \mathbb{R}^d$ $\mathcal{C}^1$ path, let $f : \mathbb{R}^d \to \mathbb{R}$ a continuous function. We define the time-augmented path $\tilde{X}(s) = (X(s), s)$ for $s \in [a, b]$.

For all $\varepsilon > 0$, there exists a linear operator $\mathcal{L}$ such that:

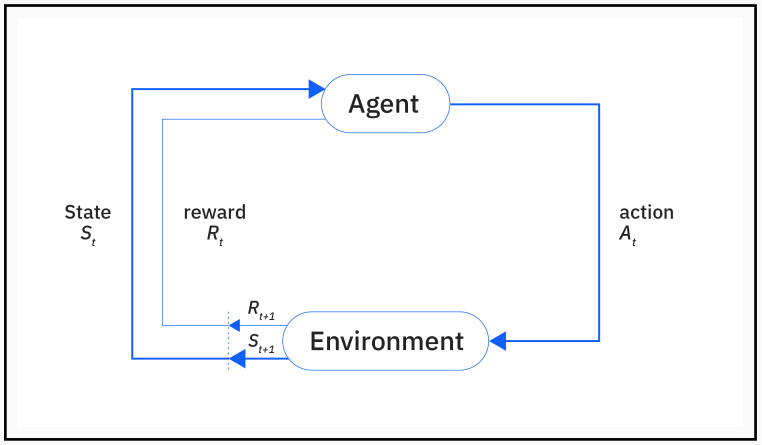
$$\left\| f(\tilde{X}) - \mathcal{L}(S(\tilde{X})) \right\| < \varepsilon$$

A similar theorem exists for truncated signatures[4].

[3] Arribas, "Derivatives Pricing Using Signature Payoffs".

[4] Chevyrev and Kormilitzin, "A Primer on the Signature Method in Machine Learning".

Discrete & Continuous Reinforcement Learning



- The Agent interacts with the environment with actions u
- u depends on x ($u(x)$ for us) $\Rightarrow$ state changes.
- Agent receives a reward $r(x,u) = \lambda_x \|x(t)\|_2^2 + \lambda_u \|u(x(t))\|_2^2$

	Value Function	Bellman equation
Discrete	$V^u(x(t)) = \sum_{i=t}^{\infty} \gamma^i r(x(i), u(x(i)))$	$V^*(x(t)) = \max_{u \in \mathcal{U}} (r(x(t),u) + \gamma V^*(x(t+1))$
Continuous	$V^u(x(t)) = \int_t^{\infty} e^{\frac{t-s}{\gamma}} r(x(s), u(x(s)))ds$	$\max_{u \in \mathcal{U}} \left(r(x(t),u) + \dot{V}^*(x(t)) \right) = -\frac{1}{\gamma} V^*(x(t))$

Continous Time RL $\Rightarrow$ well-suited for physical systems (high-frequency, different discretisation...).

Continuous Reinforcement Learning for Delayed Systems

How does the (Hamilton-Jacobi) Bellman equation translates to delayed systems?

Non-Markovian $\Rightarrow$ state of the system at t is $x_t : s \to x(t+s), s \in [-h, 0]$

HJB for DDEs/FDEs[5]:

$$\forall t, \quad \max_{u \in \mathcal{U}} (r(x_t, u) + L_u V(x_t)) = \frac{1}{\gamma} V(x_t)$$

with $L_u V(x_t) = \lim_{\Delta t \to 0} \frac{V\left(x_{t+\Delta_t}\right) - V(x_t)}{\Delta t}$

Now, how do we **learn** u and V?

[5] Kolmanovskii and Myshkis, Applied Theory of Functional Differential Equations.

Continuous Time Reinforcement Learning Algorithms

Plugging Signatures

How do we compute $V(x_t)$ and $u(x_t)$? $\Longrightarrow$ Signature approximation: $V(x_t) \approx \phi^\top S^N_{[-h,0]}(x_t)$ and $u(x_t) \approx \theta^\top S^N_{[-h,0]}(x_t)$.

Time derivative: $L_u V(x_t) = \phi^\top L_u S(x_t) = \phi^\top \lim_{\Delta t \to 0} \frac{S_{[-h,0]}(x_{t+\Delta t}) - S_{[-h,0]}(x_t)}{\Delta t}$

Can be approximated in two ways:

- $L_u S(x_t) \approx \frac{S_{[-h,0]}(x_{t+\Delta t}) - S_{[-h,0]}(x_t)}{\Delta t} \rightarrow$ compute both signature (not expensive)
- $L_u S(x_t) = S(x_t) \otimes \dot{x}(t) - \dot{x}(t-h) \otimes S(x_t)^{(*)} \rightarrow \dot{x}(t) \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$

$(*)$The proof for this equality is done using Chen's identity.[6]

[6] Chen, "Iterated Integrals and Exponential Homomorphisms".

Value-Gradient (VG)

Value-Gradient idea: learn the Value function and compute the optimal command using the learned $\hat{V}$.

- Minimize Temporal Difference $\delta(x_t) = r(x_t, u(x_t)) + L_u V(x_t) - \frac{1}{\gamma} V(x_t)$.
- Linear Quadratic Cost $\Rightarrow$ finding u is a simple optimisation problem.

With Signature-based Value function:

$$u^* = \frac{1}{2} R^{-1} g(x_t)^\top \nabla_{x(t)} V(x_t)$$

with $\nabla_{x(t)} V(x_t)$ the derivative of the Value Function at the extremity of the path $x(t)$, $g(x_t)$ the input function

Continuous Actor-Critic

- Actor $\Rightarrow$ control u
- Critic $\Rightarrow$ learn estimated Value Function $\hat{V}$.
- Critic updated with Bellman equation
- Actor updated with TD error (from Critic)

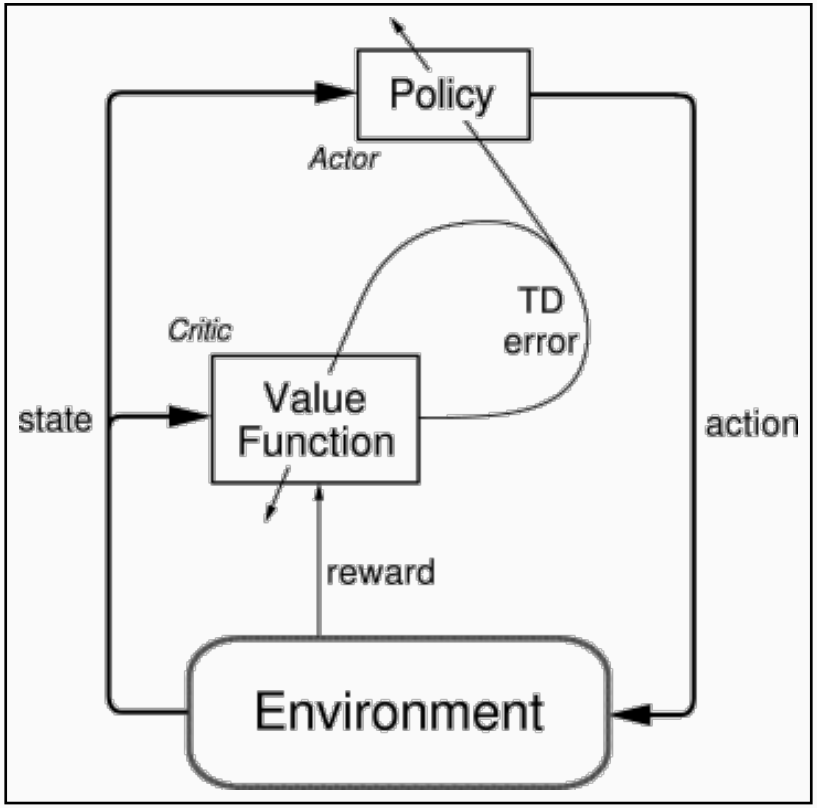


Figure 3: Actor Critic explained, from R. Sutton blog

Continuous Time Actor-Critic, delayed version

How do we adapt the Actor-Critic to Non-Markovian framework?

- Based on K. Doya's Continuous Time RL algorithms.[7]
- Instead of considering $x(t)$ as your state, take $x_t : s \mapsto x(t+s)$ for $s \in [-\tau_{\max}, 0]$ as your system state.
- Value is updated by minimizing the loss $\frac{1}{2}\delta(x_t)^2$ (like Value-Gradient).
- Actor update: $w_a \leftarrow w_a + \frac{n}{\sigma}\frac{\partial u}{\partial w_a} \Rightarrow$ **no noise implies no update**.

[7] Doya, "Reinforcement Learning in Continuous Time and Space".

Implementation

Environment:
- DDE solver compatible with RL
- RK4 scheme with linear interpolation.

Signatures:
- JAX library `signax`
- Full signature is recomputed each step (quick)

Training:
- Whole framework is coded in JAX $\Rightarrow$ compatible with the backpropagation with `signax`.

Experiments

Experimental Setup

3 different systems:

- **Simple 2D linear system:** $\dot{x} = Ax(t) + A_1 x(t-\tau) + Bu \rightarrow$ show developed methods are able to stabilize the systems
- **Mackey-Glass (chaotic 1D system):** $\dot{x} = -\mu x(t) + \frac{px(t-\tau)}{1+x(t-\tau)^n} \rightarrow$ Study behaviour on chaotic system and performance gains
- **Chemical reactor (nonlinear 4D system)**[8] $\rightarrow$ Evaluate performances on real systems & compare with literature

Each system is tested for different seeds and compared to CTAC without signatures.

[8] Dadebo and Luus, "Optimal Control of Time-Delay Systems by Dynamic Programming".

Metrics & Training

How to evaluate performances?

- **Cumulative cost**: $\int_{T_0}^{T_f} r(x,u)dt$. The lower is the better. Cost that plateaus $\Longrightarrow$ rewards going to 0 $\Longrightarrow$ system is stabilized around the origin $\Longrightarrow$ **success**.
- **State Norm** $t \mapsto \|x(t)\|_2$

Training (cluster)

- **2D system**: 1 min/run
- **Mackey-Glass**: 5 min/run to 20 min/run (dep. delay & depth)
- **Chemical Reactor**: 30 min/run

Chemical Reactor study

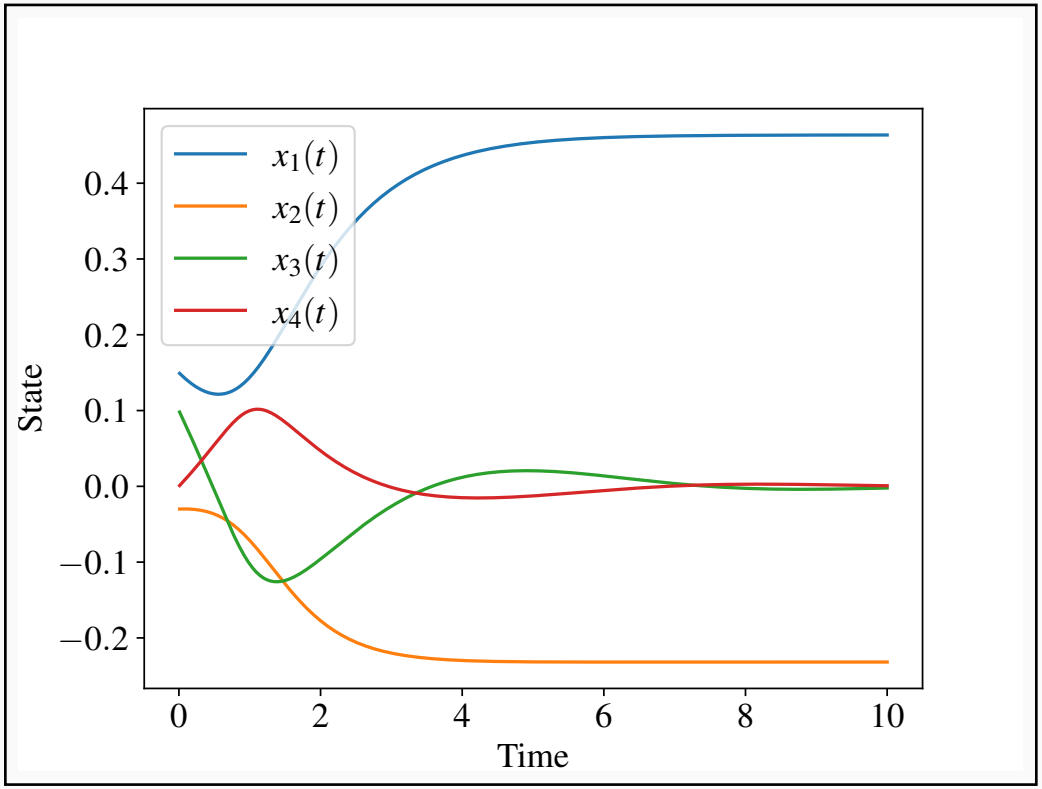


Figure 4: Chemical reactor trajectory

- $$\begin{cases} \dot{x}_1 = 0.5 - x_1(t) - R_1 \\ \dot{x}_2 = -2(x_2(t) + 0.25) - u_1(t)(x_2(t) + 0.25) + R_1 \\ \dot{x}_3 = x_1(t-\tau) - x_3(t) - R_2 + 0.25 \\ \dot{x}_4 = x_2(t-\tau) - 2x_4(t) - u_2(t)(x_4(t) + 0.25) + R_2 - 0.25 \end{cases}$$

$$\begin{cases} R_1 = (x_1(t) + 0.5)\exp\left(\frac{25x_2(t)}{x_2(t)+2}\right) \\ R_2 = (x_3(t) + 0.25)\exp\left(\frac{25x_4(t)}{x_4(t)+2}\right) \end{cases}$$

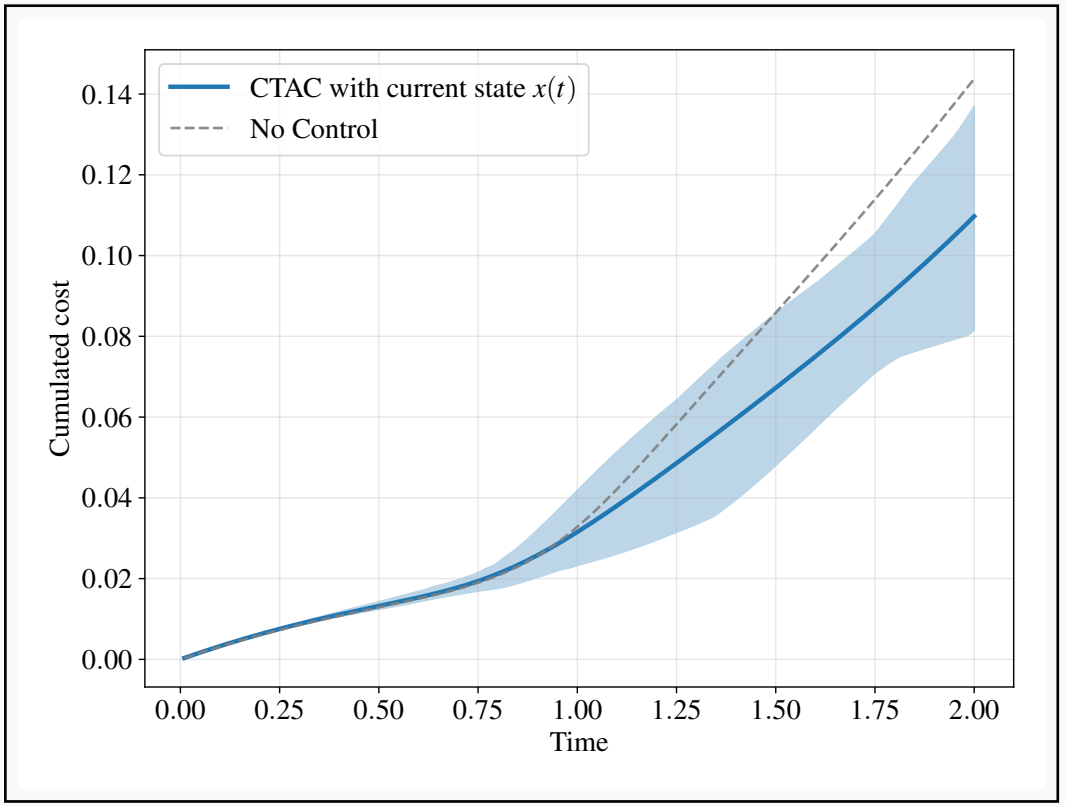
- 4D non linear system
- Optimize: $I = \int_0^2 r(x(t), u(x(t)))dt$
- **Baseline**: Dynamic Programming result $I = 0.024$

Chemical Reactor Instabilities

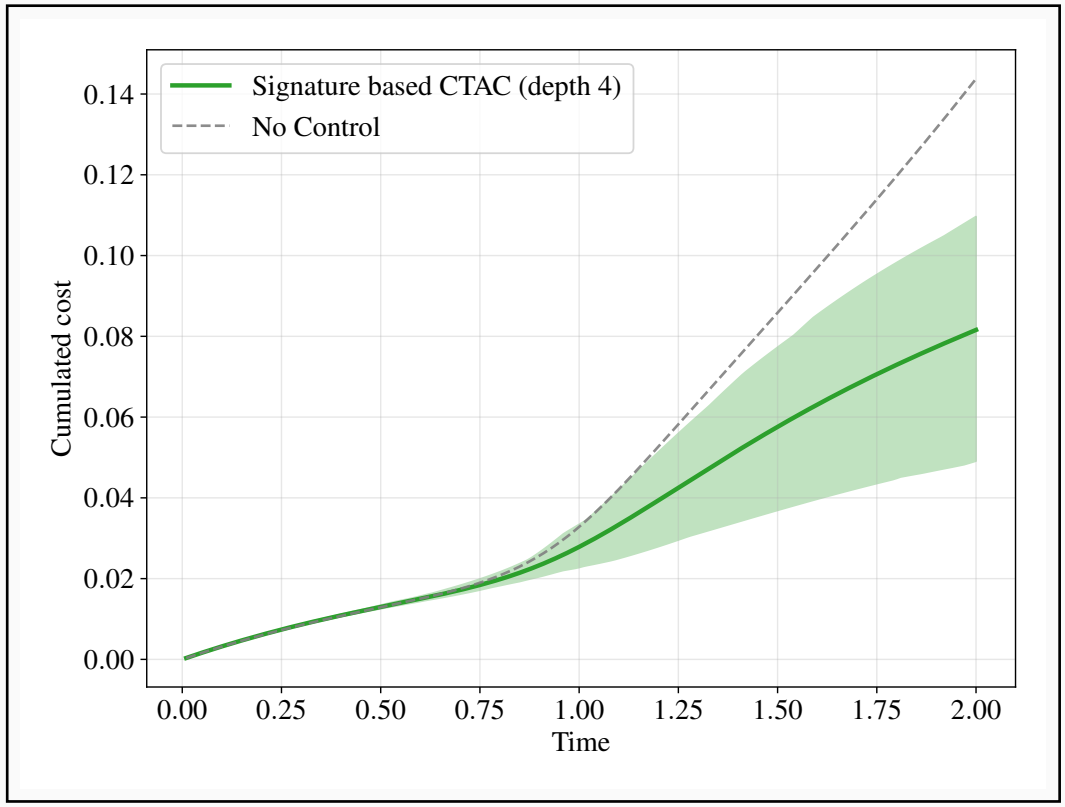
Technique	Bad perf. (%)	NaN (%)	Total rem. (%)	Best perf.	Mean perf. and std
CTAC with aug. position $(x(t), x(t-\tau))$	82.35	**0.00**	82.35	0.059	0.12 ± 0.024
CTAC with current position $x(t)$	54.90	**0.00**	54.90	0.037	0.11 ± 0.024
CTAC sig. depth 2	3.92	**0.00**	3.92	0.032	0.087 ± 0.023
CTAC sig. depth 3	**0.00**	**0.00**	**0.00**	0.036	0.085 ± 0.024
CTAC sig. depth 4	1.96	**0.00**	1.96	0.038	0.082 ± 0.024
VG sig. depth 2	47.06	**0.00**	47.06	0.036	0.11 ± 0.022
VG sig. depth 3	66.67	4.00	70.59	**0.027**	**0.073 ± 0.038**
VG sig. depth 4	52.94	**0.00**	52.94	0.032	0.076 ± 0.033
VG sig. depth 2 with more noise	16.00	**0.00**	16.00	0.030	0.11 ± 0.022
VG sig. depth 3 with more noise	34.00	4.00	38.00	0.031	**0.073 ± 0.038**
VG sig. depth 4 with more noise	40.00	**0.00**	40.00	0.037	0.076 ± 0.033

Table 1: Comparison of techniques on the chemical process: seed removal statistics and performance metrics.

Chemical Reactor results: mean



(a) CTAC with current state



(b) CTAC with signatures (depth 4)

Figure 5: Comparison of mean and quantiles (10-90 %) values for various techniques on the chemical reaction system **conditional on success**

Chemical Reactor results: best shot

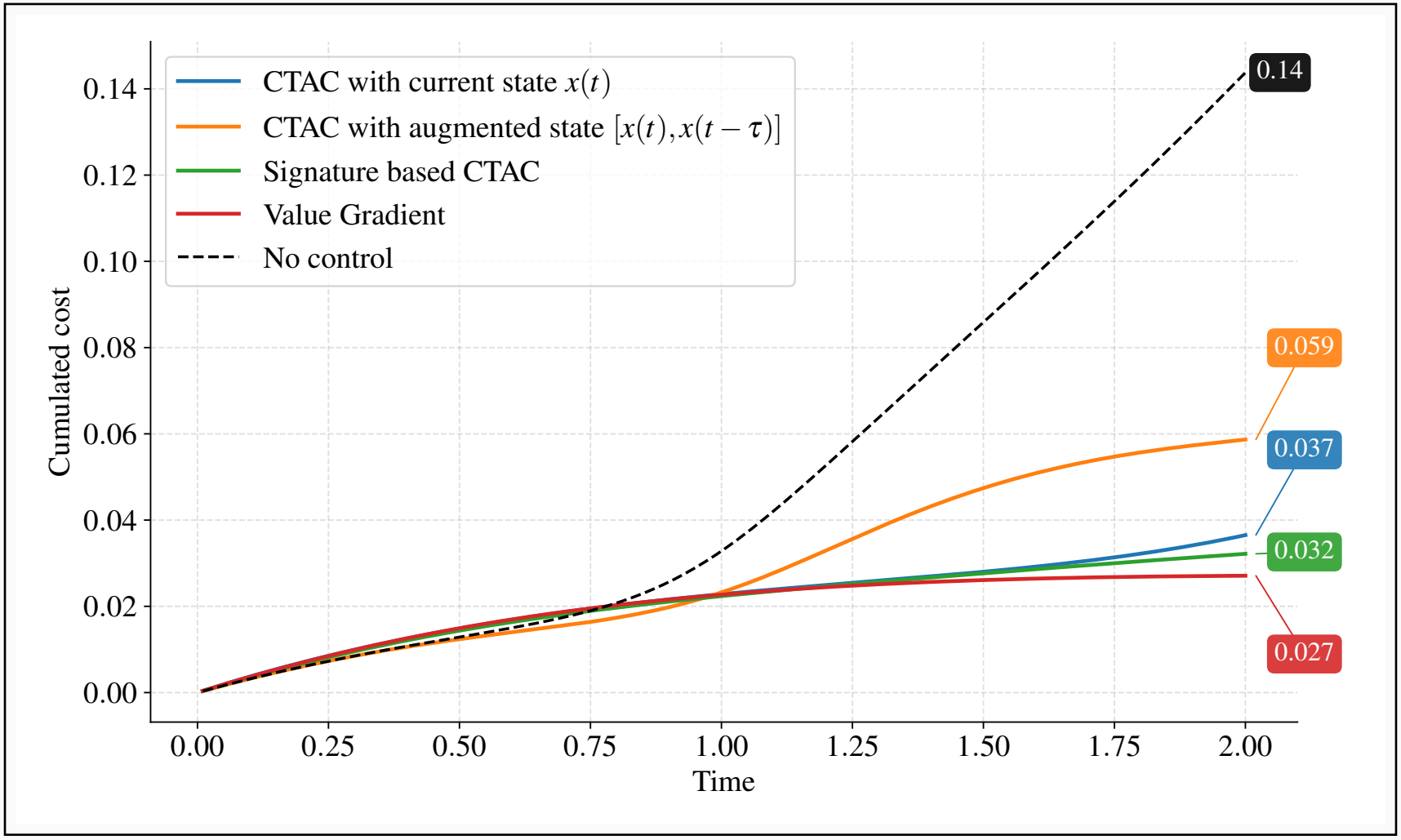


Figure 6: Best cumulative cost for various RL techniques. **Baseline: 0.024**

Limitations and Future Work

What can cause the Instabilities?

- **Curse of dimensionality**: number of signature coefficients grows as d^N, (N the depth, d system dimension). $d = 4, N = 4 \Rightarrow 4096$ coeffs (with time augmentation).
- **CTAC estimation of time derivative**: Finite difference scheme $\Rightarrow$ noisy.
- **VG convergence**: VG gets unstable as the Value Function error converges
- **Lots of Hyperparameters**:
  - lots of different parameters to play with
  - finding the sweet spot is computationally expensive.

Future Work

- **Analyse the algorithms analytically**: convergence bounds, more insights on instabilities...
- **Technical and engineering tricks**: Can we add some machine learning techniques to prevent gradient explosion?
- **Plug more complex networks?**: Current Signature & Machine Learning literature do nor use such simple approximation function: plug a Multi Layer Perceptron?
- **Partially observed systems**: Takens Theorem allows to link partially observed systems to delayed systems $\Longrightarrow$ Use the signatures on PO systems without the struggle of the delayed mechanics.

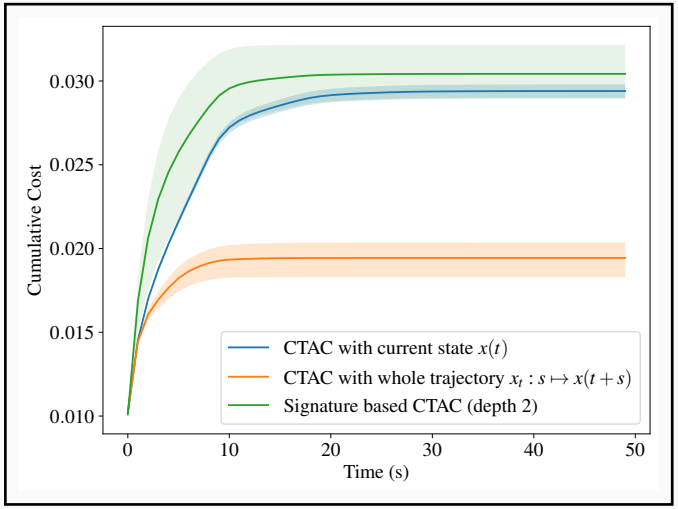
Conclusion

Conclusion

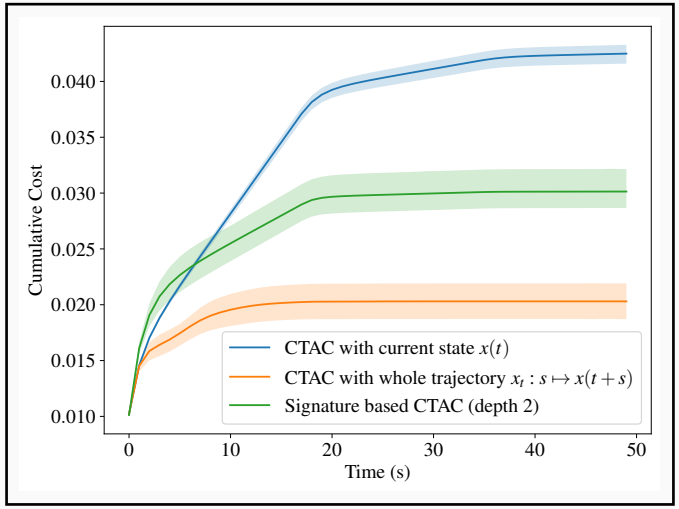
- Two algorithms for Reinforcement Learning for Delayed Systems, promising results on complex systems (chemical reactor)
- Formulation of the problem using Signatures, update rule for both algorithms
- Delayed Systems simulator and Environment to perform RL

Thank you for your attention!

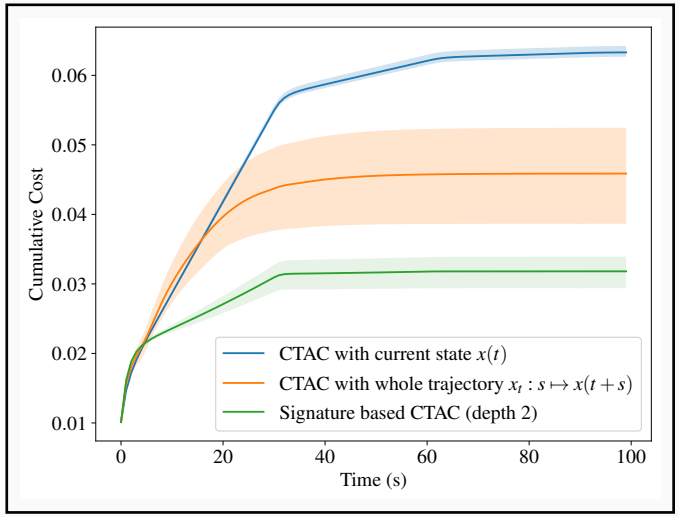
Mackey-Glass analysis



(a) $\tau = 8$



(b) $\tau = 17$



(c) $\tau = 30$

Figure 7: Comparison of cumulative cost (mean and 90% quantiles) on Mackey-Glass system for different control techniques over 10 seeds

More insights on instabilities

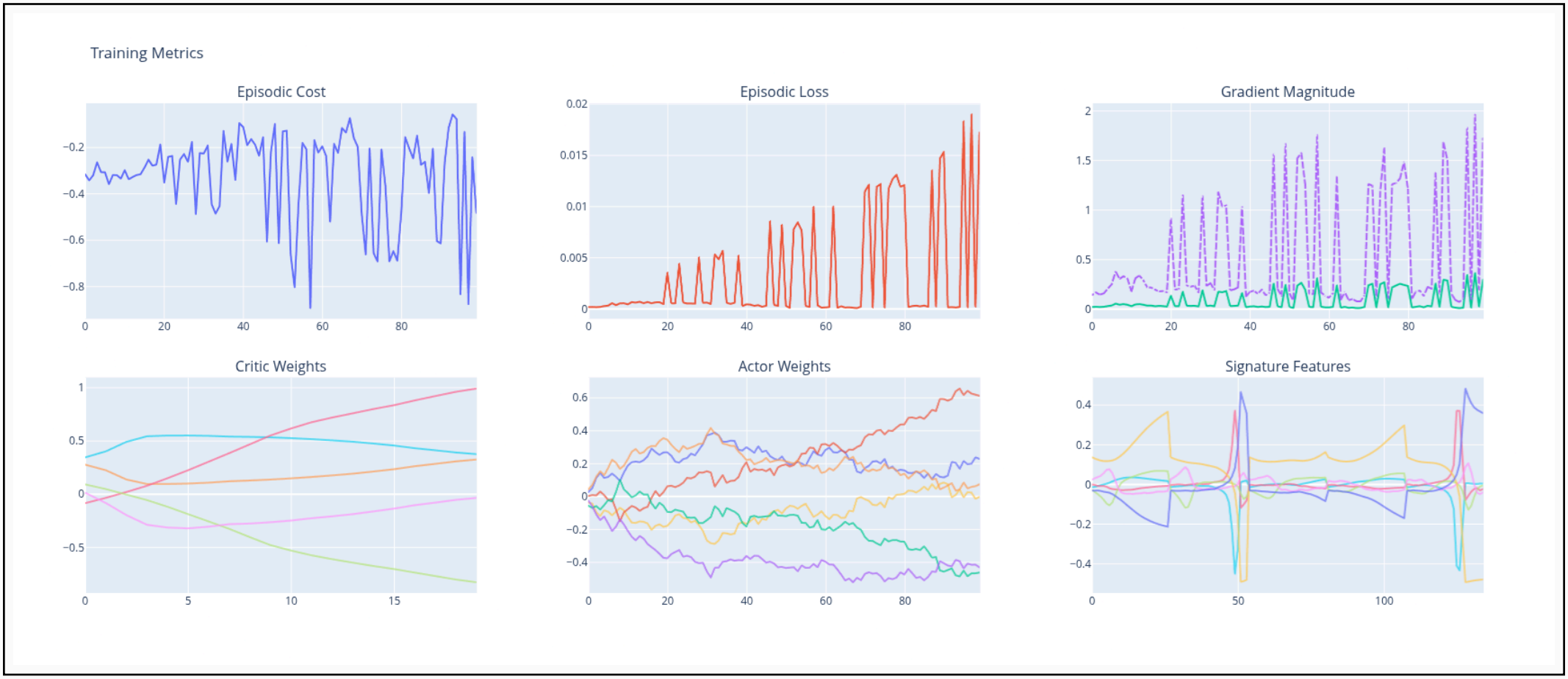


Figure 8: Training on Chemical Reactor, CTAC depth 2

More insights on instabilities

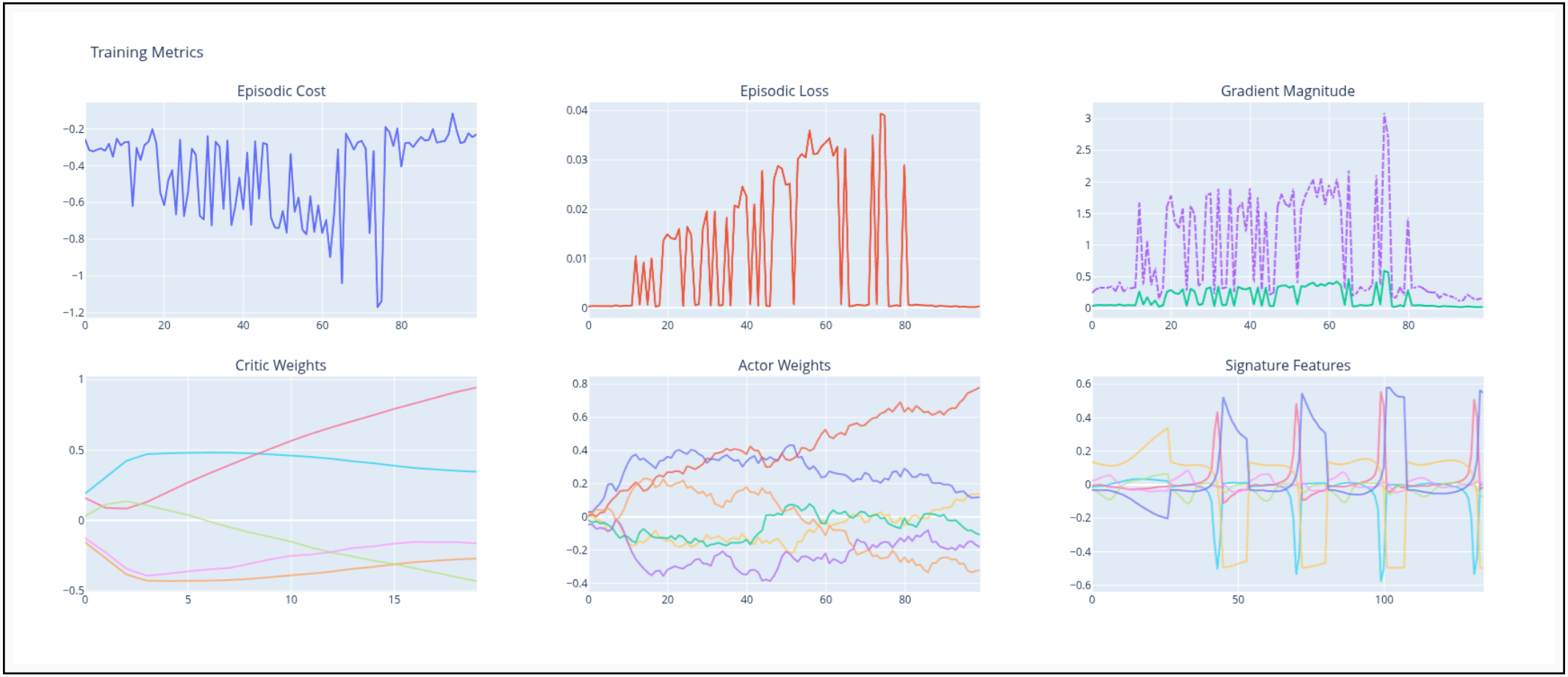


Figure 9: Training on Chemical Reactor, CTAC depth 3

More insights on instabilities

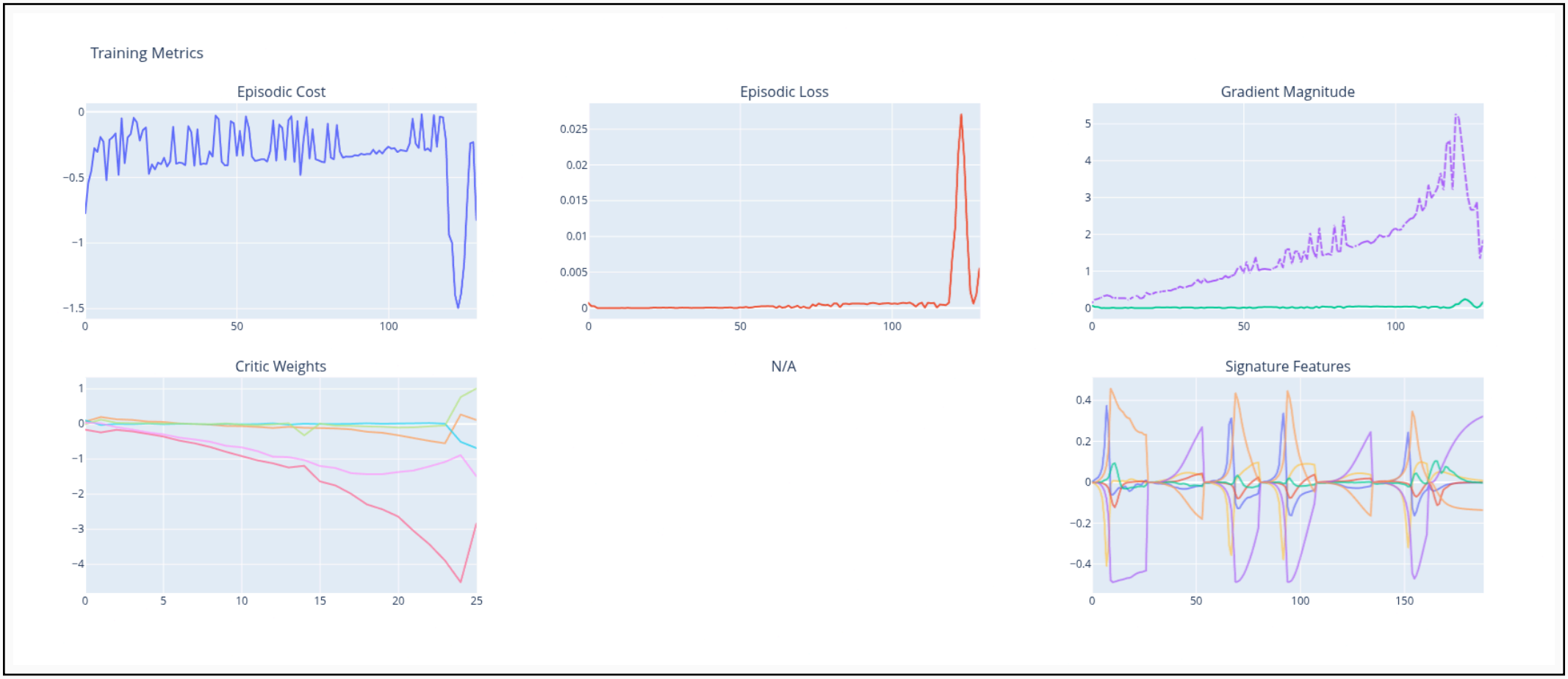


Figure 10: Training on Chemical Reactor, VG depth 2

CTAC algorithm

Algorithm 1: Continuous Time Actor Critic

Initialisation: actor function A with weights w_a and critic function V with weights
1 w_c, noise std σ, discretisation Δt, environment with step function dependent of Δt.

2 **while** not finished

3 $n \leftarrow$ random noise, std $= \sigma$

4 $x_{t+\Delta t}, r_t \leftarrow \text{step}_{\Delta t}(x_t, u + n)$

5 compute $V(x_t)$ and $V(x_{t+\Delta t})$

6 $\delta_t \leftarrow \frac{V(x_t) - V(x_{t+\Delta t})}{\Delta t} + r_t - \frac{1}{\gamma} V(x_t)$ temporal difference

7 $w_c \leftarrow w_c + \eta_c \delta_t \frac{\partial V(x_t)}{\partial w_c}$

8 $w_a \leftarrow w_a + \frac{n}{\sigma} \frac{\partial A}{\partial w_a}$

Value Gradient Algorithm

Algorithm 2: Functional Value Gradient

Initialisation: Critic function V with weights w_c, noise std σ, discretisation Δt,
environment with step function dependent of Δt, matrix $g(x_t)$, learning rate $\eta_c > 0$.

while not finished

- $\nabla_{x(t)} V(x_t) \leftarrow \frac{\partial V(x_t)}{x_t} \mid_{t=0}$
- $u \leftarrow \frac{1}{2} R^{-1} g(x_t)^\top \nabla_{x(t)} V(x_t)$
- $n \leftarrow$random noise, std $= \sigma$
- $x_{t+\Delta t}, r_t \leftarrow \text{step}_{\Delta t}(x_t, u + n)$
- compute $V(x_t)$ and $V(x_{t+\Delta t})$
- $\delta_t \leftarrow \frac{V(x_t) - V(x_{t+\Delta t})}{\Delta t} + r_t - \frac{1}{\tau} V(x_t)$ temporal difference
- $w_c \leftarrow w_c + \eta_c \delta_t \frac{\partial V(x_t)}{\partial w_c}$

Bibliography

Arribas, Imanol Perez. “Derivatives Pricing Using Signature Payoffs.” no. arXiv:1809.09466. ArXiv, September 2018. https://doi.org/10.48550/arXiv.1809.09466.

Chen, Kuo-Tsai. “Iterated Integrals and Exponential Homomorphisms.” Proceedings of the London Mathematical Society 3, no. 1 (1954): 502–12.

Chevyrev, Ilya, and Andrey Kormilitzin. “A Primer on the Signature Method in Machine Learning.” 2025. https://arxiv.org/abs/1603.03788.

Dadebo, Solomon, and Rein Luus. “Optimal Control of Time-Delay Systems by Dynamic Programming.” Optimal Control Applications and Methods 13, no. 1 (1992): 29–41. https://doi.org/10.1002/oca.4660130103.

Doya, Kenji. “Reinforcement Learning in Continuous Time and Space.” Neural Computation 12, no. 1 (2000): 219–45. https://doi.org/10.1162/089976600300015961.

Fermanian, Adeline, Pierre Marion, Jean-Philippe Vert, and Gérard Biau. “Framing RNN as a Kernel Method: A Neural ODE Approach.” no. arXiv:2106.01202. ArXiv, October 2021. https://doi.org/10.48550/arXiv.2106.01202.

Kidger, Patrick, Patric Bonnier, Imanol Perez Arribas, Cristopher Salvi, and Terry Lyons. “Deep Signature Transforms.” Advances in Neural Information Processing Systems 32 (2019).

Kolmanovskii, V., and A. Myshkis. Applied Theory of Functional Differential Equations. Springer Netherlands, 1992. https://doi.org/10.1007/978-94-015-8084-7.