# Numerical control of the heat equation with reinforcement learning


S. Kadri Harouna

Laboratoire de Mathématiques, Image et Applications (MIA)
Avenue Michel Crépeau 17042 La Rochelle


**Joint work** with K. Ammari (University of Monastir, Tunisia) & Ghazi Bel Mufti (ESSAIT, University of Carthage, Tunisia).

# Outline

1 Introduction

# Outline

1. Introduction

2. Wavelet-based Galerkin method

# Outline


1. Introduction
2. Wavelet-based Galerkin method
3. Reinforcement learning to control first order system

# Outline


1. Introduction
2. Wavelet-based Galerkin method
3. Reinforcement learning to control first order system
4. Numerical results

# Introduction

The initial-boundary value problem for the one-dimensional heat conduction that we considered is:

$$\begin{cases} \partial_t u(t,x) = \nu \partial_x^2 u(t,x) + f(t,x), \; x \in [0,1] \text{ and } t \in ]0,T], \\ \\ u(0,x) = u_0(x), \end{cases} \tag{1}$$

where $\nu > 0$ is the diffusion coefficient, $f$ is the source term and $T > 0$.
Homogeneous Dirichlet boundary conditions are assumed: $u(t,0) = u(t,1) = 0$.

## Objective

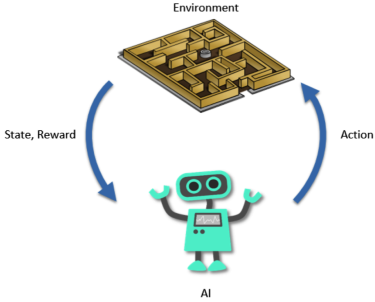
- Given a target $u_T \in L^2(0,1)$, find a source term $f(t,.) \in L^2(0,1)$, such that:

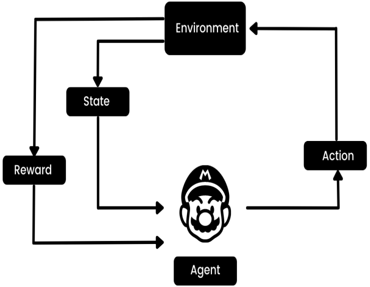
$$\|u(T,.) - u_T\|_{L^2(0,1)} \leq \epsilon \quad \text{for} \quad \epsilon > 0.$$

$\longrightarrow$ Numerical exact control remains elusive.

# Introduction

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns the optimal action for a given task through its repeated interaction with a dynamic environment that either rewards or punishes the agent action.

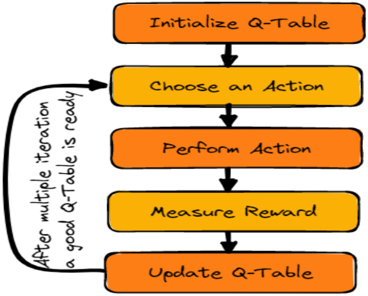


# Introduction

Q-learning is a model-free, value-based, off-policy algorithm that will find the best series of actions based on the agent's current state. The Q stands for quality. Quality represents how valuable the action is in maximizing future rewards.

Q-Table: the agent maintains the Q-table of sets of states and actions.




⟶ **Objective:** to learn a Q-table of state and action.

# Introduction

- States: $s_t$, the current position of the agent in the environment.

$$s_t = u(t, .)$$

- Action: $a_t$, a step taken by the agent in a particular state.

$$a_t = f(t, .)$$

- Rewards: $R_t$, for every action, the agent receives a reward and penalty.

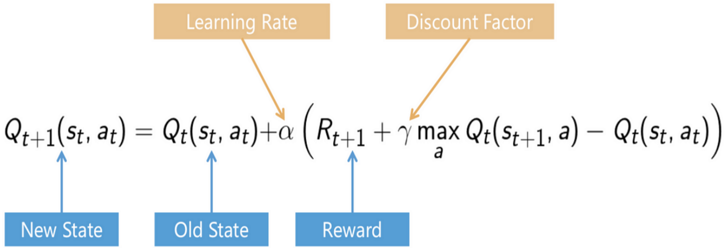
$$R_t = ?$$

- Episodes: the end of the stage, where agents can take new action. It happens when the agent has achieved the goal or failed.

- $Q_t(s_{t+1}, a)$: expected optimal Q-value of doing the action in a particular state.

# Introduction

Q-function uses the Bellman equation as a simple value iteration update, using the weighted average of the current value and the new information:




with $0 < \alpha \leq 1$ and $0 \leq \gamma \leq 1$.

# Introduction

$\longrightarrow$ Is it possible to use this approach to solve the previous control problem?

$\longrightarrow$ How accurate is the method that results from this?

$\longrightarrow$ What kind of improvements can be made?

# Introduction

[E. Hernandez, D.Kalise, E. Otárola, 09]: Numerical approximation of the LQR problem in a strongly damped wave equation.

[M.A. Bucci, et al, 19]: Control of chaotic systems by deep reinforcement learning.

[K. Ammari, G. Bel Mufti, 23]: Controlling a dynamic system through reinforcement learning

[G. Novati, L. Mahadevan, P. Koumoutsakos, 19]: Controlled gliding and perching through deep-reinforcement-learning.

$\longrightarrow$ Wavelet approach satisfying physical boundary condition.

# Biorthogonal wavelet basis

Multi-scale projection of $f \in L^2(0,1)$:

$$\mathcal{P}_j(f) = \sum_{k\in\mathbb{Z}} \langle f, \tilde{\varphi}_{j,k}\rangle \varphi_{j,k} \quad \text{and} \quad \mathcal{Q}_j(f) = \sum_{k\in\mathbb{Z}} \langle f, \tilde{\psi}_{j,k}\rangle \psi_{j,k} \tag{2}$$

with:

$$V_j = span\{\varphi_{j,k}\} \text{ and } W_j = span\{\psi_{j,k}\} = V_{j+1} \cap \tilde{V}_j^{\perp}.$$

Multi-scale decomposition of $f \in L^2(0,1)$ :

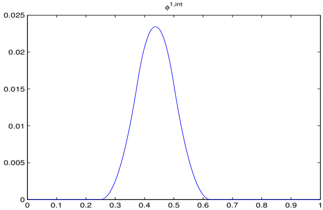
$$f = \mathcal{P}_j(f) + \sum_{\ell \geq j} \mathcal{Q}_\ell(f) \text{ with } \mathcal{Q}_j(f) = \mathcal{P}_{j+1}(f) - \mathcal{P}_j(f).$$

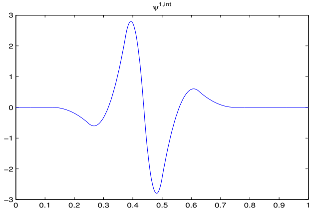
Given $f \in H^s(0,1)$, we have the following Jackson and Bernstein inequalities:

$\|\mathcal{P}_j(f) - f\|_{L^2(0,1)} \leq C2^{-js}\|f\|_{H^s(0,1)}$ and $\|\mathcal{P}_j(f)\|_{H^s(0,1)} \leq C2^{js}\|\mathcal{P}_j(f)\|_{L^2(0,1)}$, $s > 0$.

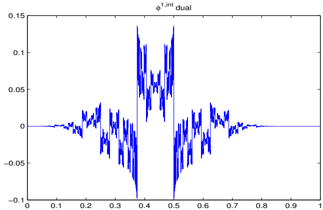
# Biorthogonal B-Spline wavelets (3 vanishing moments)

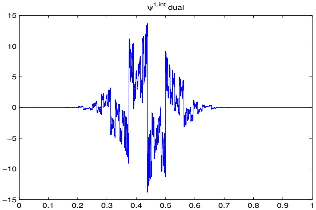
**Primal scaling function (left) and associated wavelet (right):**



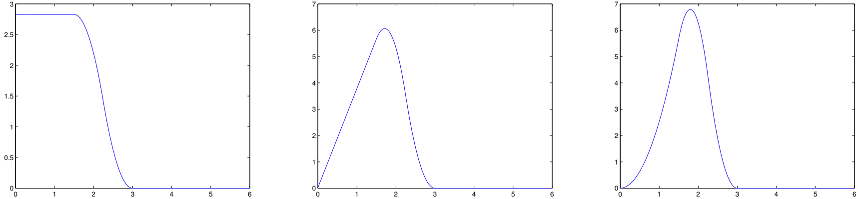

**Dual scaling function (left) and associated wavelet (right):**



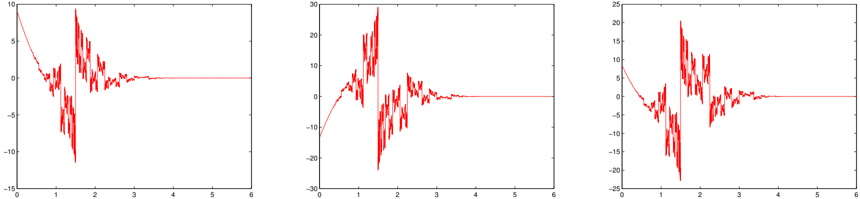



# Wavelet basis satisfying boundary conditions

**Edge** 0 **scaling function of** $V_j^1$**: B-Spline** 3.3

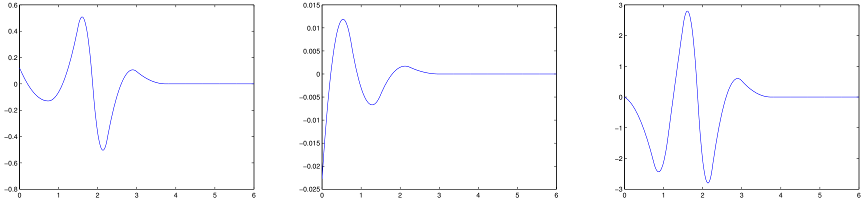



**Edge** 0 **scaling function of** $\tilde{V}_j^1$**: B-Spline** 3.3

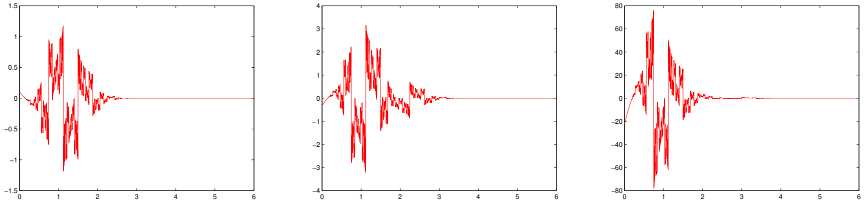


# Wavelet basis satisfying boundary conditions

**Edge 0 wavelets of $W_j^1$: B-Spline 3.3**



**Edge 0 wavelets of $\tilde{W}_j^1$: B-Spline 3.3**



# Wavelet-based Galerkin method for the heat equation

The solution $u_j \in V_j$ of (1) is searched in the following discrete form:

$$u_j(t,x) = \sum_{k=1}^{N_j} \langle u, \tilde{\psi}_{j,k} \rangle \psi_{j,k}(x) = \sum_{k=1}^{N_j} d_{j,k}(t)\psi_{j,k}(x). \tag{3}$$

For $m = 1, \ldots, N_j$, integration by part and the boundary conditions lead to:

$$\sum_{k=1}^{N_j} \left[ d'_{j,k}(t)\langle \psi_{j,k}, \psi_{j,m} \rangle + \nu d_{j,k}(t)\langle \psi'_{j,k}, \psi'_{j,m} \rangle \right] = \langle f(t,.), \psi_{j,m} \rangle. \tag{4}$$

Thus, the coefficients $(d_{j,k})$ are solution of a differential system:

$$\mathcal{A}_j \left[ d'_{j,k}(t) \right] + \mathcal{R}_j \left[ d_{j,k}(t) \right] = \mathcal{A}_j \left[ f_{j,k}(t) \right], \tag{5}$$

with

$$[\mathcal{A}_j]_{k,m} = \int_0^1 \psi_{j,k}(x)\psi_{j,m}(x)dx \quad \text{and} \quad [\mathcal{R}_j]_{k,m} = \nu \int_0^1 \psi'_{j,k}(x)\psi'_{j,m}(x)dx. \tag{6}$$

$\longrightarrow$ Symmetric and positive definite matrices with diagonal preconditioners.

# Wavelet-based Galerkin method for the heat equation

**Posteriori error estimate**

## Proposition

Let $u$ and $u_j$ be solutions of (1) and (4), respectively. If the initial conditions $u_0(x)$ and the wavelet basis are *regular enough*, then we have:

$$\|u_j - u\|_{L^2(0,1)} \leq C2^{-js}, \tag{7}$$

for all $j \geq j_{min}$ and $s > 0$.

Then, we have:

$$\begin{aligned} \|u(T) - u_T\|_{L^2(0,1)} &\leq \|u(T) - \mathcal{P}_j(u(T))\|_{L^2(0,1)} + \|u_T - \mathcal{P}_j(u_T)\|_{L^2(0,1)} \\ &+ \|u_j - \mathcal{P}_j(u_T)\|_{L^2(0,1)} \leq C2^{-js} + \epsilon. \end{aligned}$$

$\longrightarrow$ $j_{min}$ the smallest resolution to avoid boundary functions support overlapping

# Wavelet coefficients control

Given $d_{j,k}^T \sim \mathcal{P}_j(u^T)$, we aim to find $[f_{j,k}(t)] = \mathcal{B}_j\,[v_{j,k}(t)]\,,$ such that:

$$\|d_{j,k}(T) - d_{j,k}^T(t)\|_{\ell^2} \leq \epsilon,$$

with $v_j = \sum_{k=1}^{N_j} v_{j,k}(t)\psi_{j,k}(x)$ and $\mathcal{B}_j$ a suitable real matrix of rank less than $N_j$.

System (5) rewrites:

$$\left[d'_{j,k}(t)\right] + \mathcal{M}_j\,[d_{j,k}(t)] = \mathcal{B}_j\,[v_{j,k}(t)] \quad \text{with} \quad \mathcal{M}_j = \mathcal{A}_j^{-1}\mathcal{R}_j. \tag{8}$$

$\longrightarrow$ ODE system control: Kalman rank criterion for $\mathcal{M}_j$ and $\mathcal{B}_j$.

# Time discretization

For a time step $\delta t > 0$ and integer $n \geq 0$, we search:

$$x_{t_n} \approx d_{j,k}(n\delta t) \text{ and } v_{t_n} \approx v_{j,k}(n\delta t).$$

An explicit Euler scheme leads to:

$$x_{t_{n+1}} = f(x_{t_n}, v_{t_n}) = A_{\delta t} x_{t_n} + B_{\delta t} v_{t_n}, \quad (9)$$

where

$$A_{\delta t} = I + \delta t \mathcal{M}_j \text{ and } B_{\delta t} = \delta t \mathcal{B}_j.$$

$\longrightarrow$ Implicite numerical schemes can be used.

# ODE system control by reinforcement learning

Usually, to obtain control for (9), a linear feedback controller is designed

$$v_{t_n} = P_{t_n} x_{t_n}.$$

The matrix $P_{t_n}$ is obtained from the solution of the algebraic Riccati equation, when minimizing the following quadratic cost function

$$J_N = \frac{\delta t}{2} \sum_{n=0}^{N} [\langle E_{\delta t} x_{t_n}, x_{t_n} \rangle + \langle R_{\delta t} v_{t_n}, v_{t_n} \rangle] + \frac{1}{2} \langle E_N x_{t_N}, x_{t_N} \rangle, \quad T_N = N\delta t = T,$$

under constraints defined by (9).

$\longrightarrow$ LQR regularization.

# ODE system control by reinforcement learning

Linear feedback can also be used in improved policy Q-learning approach:

$$r_{t_n} = r(x_{t_n}, v_{t_n}) =< x_{t_n}, E_{\delta t} x_{t_n} > + < v_{t_n}, R_{\delta t} v_{t_n} > . \quad (10)$$

The value of the total cost obtained for $x_{t_n}$ under policy $P_{t_n}$ is:

$$V_{P_{t_n}}(x_{t_n}) = \sum_{i=0}^{N-1} \gamma^i r_{t_n+i} =< x_{t_n}, K_{t_n} x_{t_n} >, \quad 0 < \gamma < 1,$$

where $K_{t_n}$ denotes the cost matrix related to the policy defined by $P_{t_n}$.

The $Q$-function:

$$Q_{t_n}(x, v) = r(x, v) + \gamma V_{P_{t_n}}(f(x, v)).$$

# ODE system control by reinforcement learning

The Q-function's value at the next time step is:

$$Q_{t_{n+1}}(x_{t_n}, v_{t_n}) = (1-\alpha)Q_{t_n}(x_{t_n}, v_{t_n}) + \alpha\left[r(x_{t_n}, v_{t_n}) + \gamma Q_{t_n}(x_{t_{n+1}}, v_{t_{n+1}})\right],$$

where

$$v_{t_{n+1}} = P_{t_{n+1}} x_{t_{n+1}}.$$

The matrix $P_{t_{n+1}}$ is the improved policy matrix computed from $P_{t_n}$ such that:

$$P_{t_{n+1}} x = \arg\min_v [r(x, v) + \gamma V_{P_{t_n}}(f(x, v))]. \tag{11}$$

Using forward calculations, we see that:

$$P_{t_{n+1}} = -\gamma(R_{\delta t} + \gamma B^*_{\delta t} K_{t_n} B_{\delta t})^{-1} B^*_{\delta t} K_{t_n} A_{\delta t}$$

$\longrightarrow$ $P_{t_n}$ and $K_{t_n}$ are obtained by means of a dynamic programming procedure.

# ODE system control by reinforcement learning

**Classical Q-learning algorithm**

**Input:** $\mathcal{S}$, $\mathcal{A}$, $\alpha$, $\gamma$
**Output:** $Q-$table
**for** *each episode* **do**
  Initialize the first state
  **for** *each step* **do**
    Given current state $s$, select action $a$ with an $\epsilon$-greedy policy
    Observe $r$ and $s'$ from the environment
    Update the $Q$-table:
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    Update $s$
    **until** end of the episode
  **end**
**end**

**Special case:**

$\longrightarrow Q_{t_{n+1}}(x_{t_n}, v_{t_n}) = Q_{t_n}(x_{t_n}, v_{t_n}) + \alpha\left[r(x_{t_n}, v_{t_n}) + \gamma Q_{t_n}(x_{t_{n+1}}, v_{t_{n+1}}) - Q_{t_n}(x_{t_n}, v_{t_n})\right]$

# Numerical results

To evaluate our method, we compared it to the HUM approach [Lions 88, Glowinski-Lions 90]. As analytical solution, we used:

$$u(t,x) = \exp(1-t)\sin^3(2\pi x) + 8x(1-x)^2, \quad x \in [0,1], \tag{12}$$

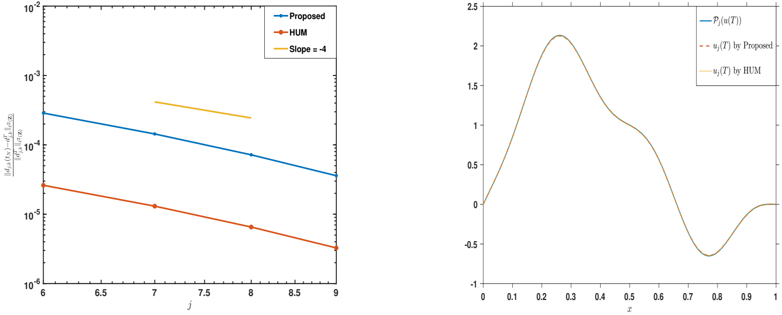
with $\delta t = 1/100$ and diffusion coefficient $\nu = 1/4\pi^2$.

First we study the discretization error:

$$\mathbf{e}_j = \frac{\|\mathcal{P}_j[u(.,t)] - u_j(.,t)\|}{\|\mathcal{P}_j u(.,t)\|}.$$

# Numerical results

## Galerkin discretization error




(a) (b)

Figure: Error $\|u_j(T) - \mathcal{P}_j(u_T)\|_{\ell^2}$ according to the resolution $j$ in loglog scale (left) and plot of the obtained end states (right) for the spatial resolution $j = 7$.

# Numerical results

The performance indicators considered are $\ell^2$ error:

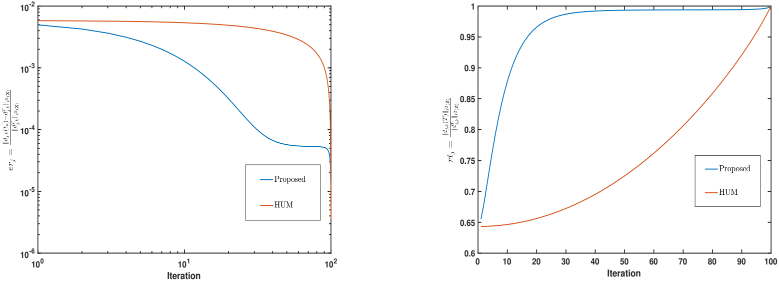
$$er_j = \frac{\|d_{j,k}(T) - d_{j,k}^T\|_{\ell^2(\mathbb{Z})}}{\|d_{j,k}^T\|_{\ell^2(\mathbb{Z})}}.$$

and the convergence ratio with respect to the change of the policy:

$$rt_j(n) = \frac{\|d_{j,k}(t_n)\|_{\ell^2(\mathbb{Z})}}{\|d_{j,k}^T\|_{\ell^2(\mathbb{Z})}}, \quad 0 \leq n \leq N.$$

# Numerical results

## Error at grid points

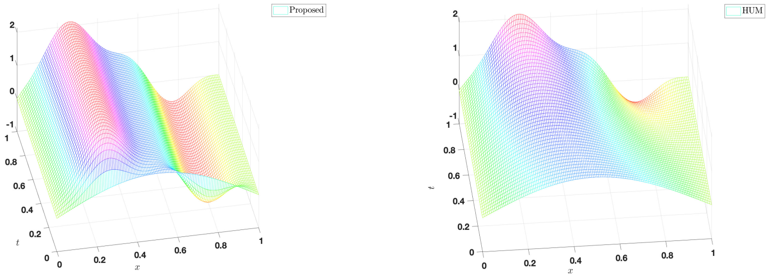



(a) (b)

Figure: Comparison of the $\ell_2$-error between the HUM method and the proposed one. Relative error $er_j$ (left ) and the convergence ratio $rt_j$ (right), according to the number of iterations.

# Numerical results

**Evolution of the error on the target state and the convergence ratio**




(a) (b)

Figure: Plot of the time evolution of the solution $u_j(t_n)$ at grid points: $0 \leq t_n \leq 1$. Proposed method (left) and the HUM method (right).

# Numerical results

| Two-dimensional space | | | | |
|---|---|---|---|---|
| $j$ | 6 | 7 | 8 | Order |
| $er_j$ | $9.5593 \times 10^{-4}$ | $5.4462 \times 10^{-4}$ | $3.1054 \times 10^{-4}$ | 3.9825 |
| $rt_j$ | 0.9916161 | 0.9916162 | 0.9916028 | |
| CPU(s) | 0.0700 | 0.1900 | 0.4800 | |

| Three-dimensional space | | | | |
|---|---|---|---|---|
| $j$ | 6 | 7 | 8 | Order |
| $er_j$ | $8.1188 \times 10^{-4}$ | $4.5918 \times 10^{-4}$ | $2.5971 \times 10^{-4}$ | 3.9552 |
| $rt_j$ | 0.99157743 | 0.9915775 | 0.99157754 | |
| CPU(s) | 1.8300 | 22.0200 | 243.3400 | |

Table: Heat equation results obtained with the proposed method in higher dimension.

Thank you for your attention

- K. Ammari, G. Bel Mufti, S. Kadri Harouna, *Reinforcement learning for the control of parabolic and hyperbolic differential equations*, in the pipeline.