This presentation in its animated version is available here :
https://lucas-perrin.github.io/canum22/

The code is in the following Github repository :
https://github.com/lucas-perrin/canum22

Time parallelisation for data assimilation

Paraexp and Luenberger observer

Lucas Perrin

Julien Salomon

INRIA Paris, ANGE Team / Laboratoire Jacques-Louis Lions

30 Mai 2022

Time parallelisation for data assimilation

→ Present a sequential data assimilation : the Luenberger observer

→ Explain the parallel in time scheme used : Paraexp

→ Expose our PinT method for sequential data assimilation

Data assimilation : Luenberger observer & dynamical systems

state dynamical system :

$$\begin{cases} \dot{x} = Ax(t) + Bu(t) \\ y(t) = Cx(t) \\ x(0) = x_0, \quad t \geq 0 \end{cases}$$

- $\rightarrow x(t)$: *state* vector
- $\rightarrow y(t)$: *output* vector
- $\rightarrow A \in \mathcal{M}_{m\times m}(\mathbb{R})$,
  $B \in \mathcal{M}_{m\times p}(\mathbb{R})$,
  $C \in \mathcal{M}_{q\times m}(\mathbb{R})$
- $\rightarrow x_0$ is unknown

observer system :

$$\begin{cases} \dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L[y(t) - \hat{y}(t)] \\ \hat{y}(t) = C\hat{x}(t) \\ \hat{x}(0) = \hat{x}_0, \quad t \geq 0 \end{cases}$$

- $\rightarrow \hat{x}(t)$: *observer* vector
- $\rightarrow L \in \mathcal{M}_{m\times q}(\mathbb{R})$
- $\rightarrow \hat{x}_0$ chosen as we want

$$\dot{x}(t) - \dot{\hat{x}}(t) = (A - LC)(x(t) - \hat{x}(t))$$

Data assimilation : Luenberger observer & dynamical systems

state dynamical system :

$$\begin{cases} \dot{x} = Ax(t) + Bu(t) \\ y(t) = Cx(t) \\ x(0) = x_0, \quad t \geq 0 \end{cases}$$

→ $x(t)$: *state* vector

→ $y(t)$: *output* vector

→ $A \in \mathcal{M}_{m\times m}(\mathbb{R})$,
$B \in \mathcal{M}_{m\times p}(\mathbb{R})$,
$C \in \mathcal{M}_{q\times m}(\mathbb{R})$

→ x_0 is unknown

observer system :

$$\begin{cases} \dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L[y(t) - \hat{y}(t)] \\ \hat{y}(t) = C\hat{x}(t) \\ \hat{x}(0) = \hat{x}_0, \quad t \geq 0 \end{cases}$$

→ $\hat{x}(t)$: *observer* vector

→ $L \in \mathcal{M}_{m\times q}(\mathbb{R})$

→ $\hat{x}_0$ chosen as we want

$$\epsilon(t) = \mathrm{e}^{(A-LC)t}(x(0) - \hat{x}(0))$$

Data assimilation : Luenberger observer & dynamical systems

state dynamical system :

$$\begin{cases} \dot{x} = Ax(t) + Bu(t) \\ y(t) = Cx(t) \\ x(0) = x_0, \quad t \geq 0 \end{cases}$$

observer system :

$$\begin{cases} \dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L[y(t) - \hat{y}(t)] \\ \hat{y}(t) = C\hat{x}(t) \\ \hat{x}(0) = \hat{x}_0, \quad t \geq 0 \end{cases}$$

How do we choose L ?

so that : $\hat{x}(t) \longrightarrow x(t)$

$\|\epsilon(t)\| \longrightarrow 0$

$\mathfrak{Re}(\Lambda = \sigma(A - LC)) < 0$

→ ODE : $L =$ `place`(Λ, A, C)

'pole placement procedure'

→ dsicretized PDE : $L = \gamma C^T \ (\gamma \in \mathbb{R})$

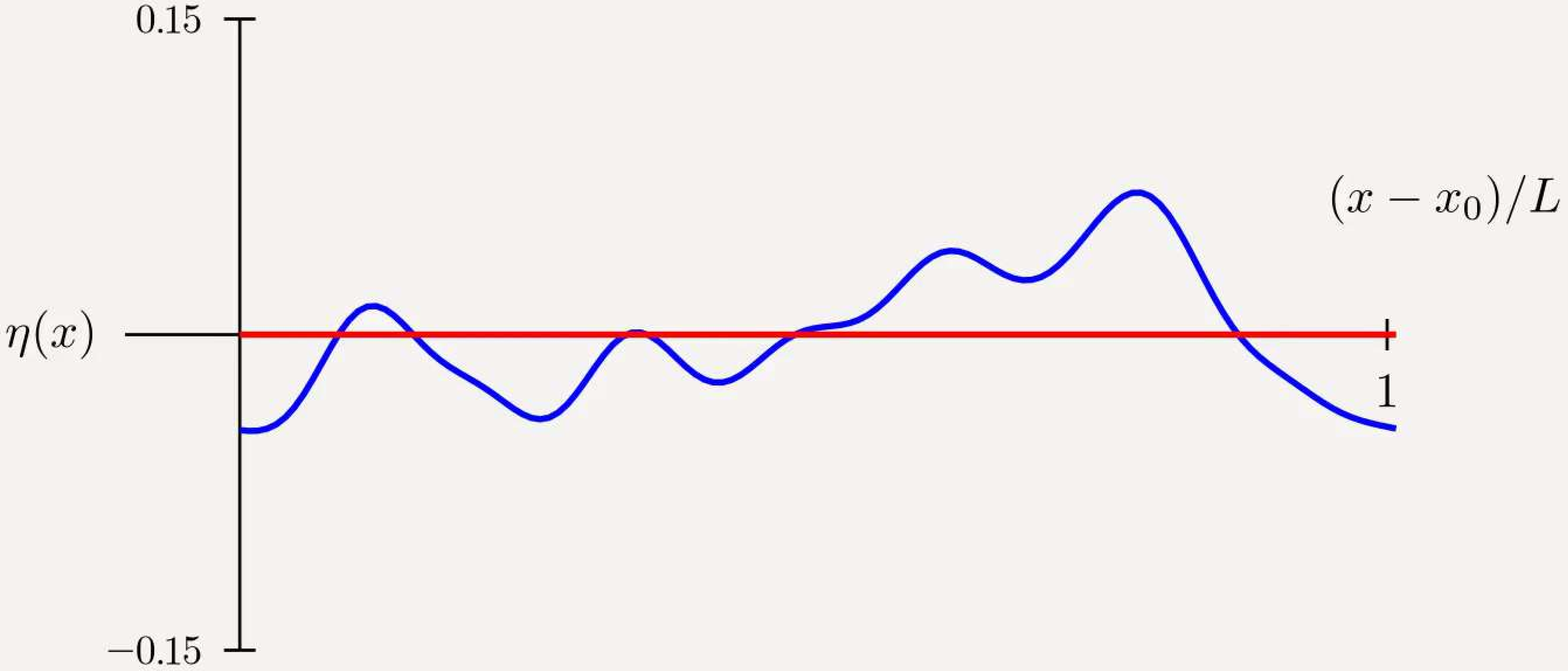
$$\|\epsilon(t)\| = \|e^{(A-LC)t}\epsilon(0)\| \leq \|x(0) - \hat{x}(0)\| \cdot \kappa(X) \cdot e^{-\mu t} \quad \rightarrow \mu = \min\{|\Lambda|\}$$

$$\rightarrow X = \text{e.v. of } A - LC$$

[1] Kautsky, Nichols, Van Dooren.'Robust pole assignment in linear state feedback.' (1985)

[2] Liu. 'Locally distributed control and damping for the conservative systems.' (1987)

Data assimilation : Luenberger observer & dynamical systems

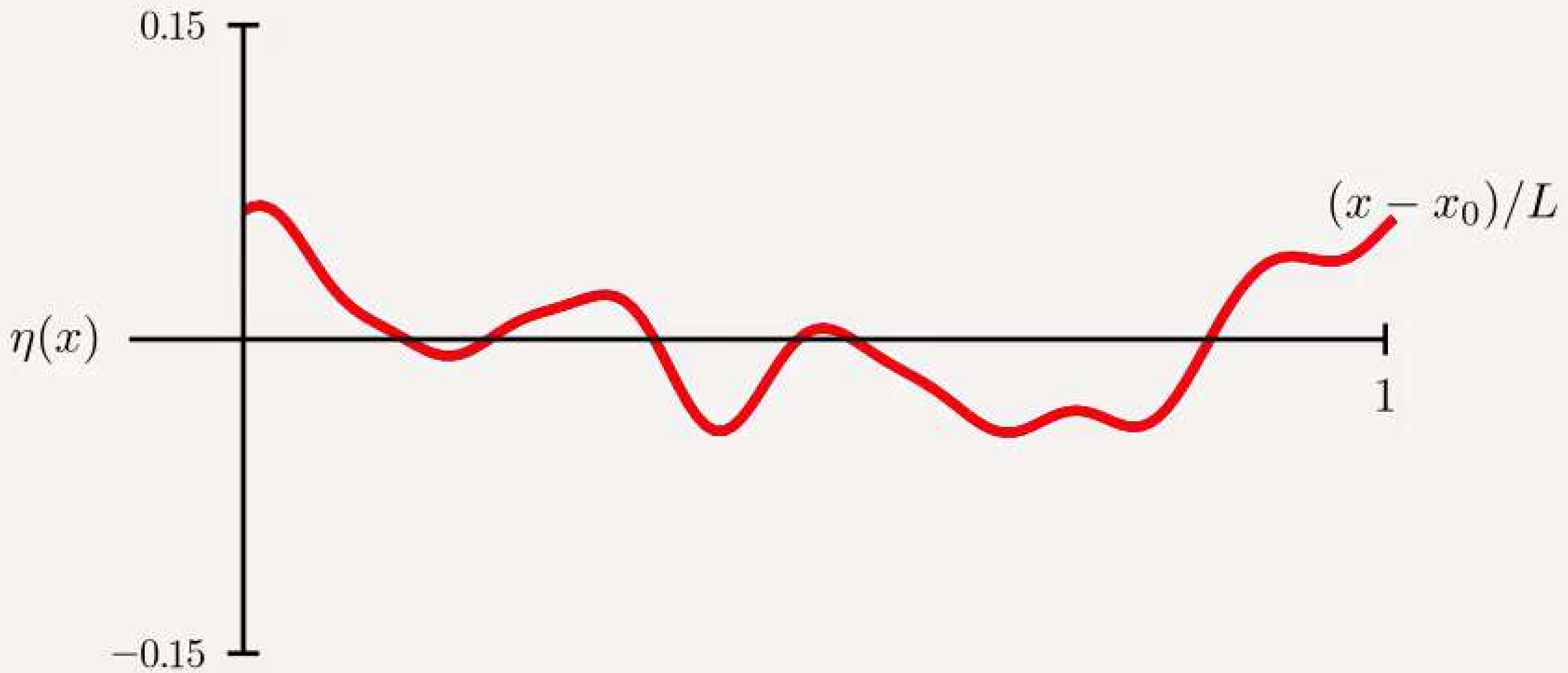


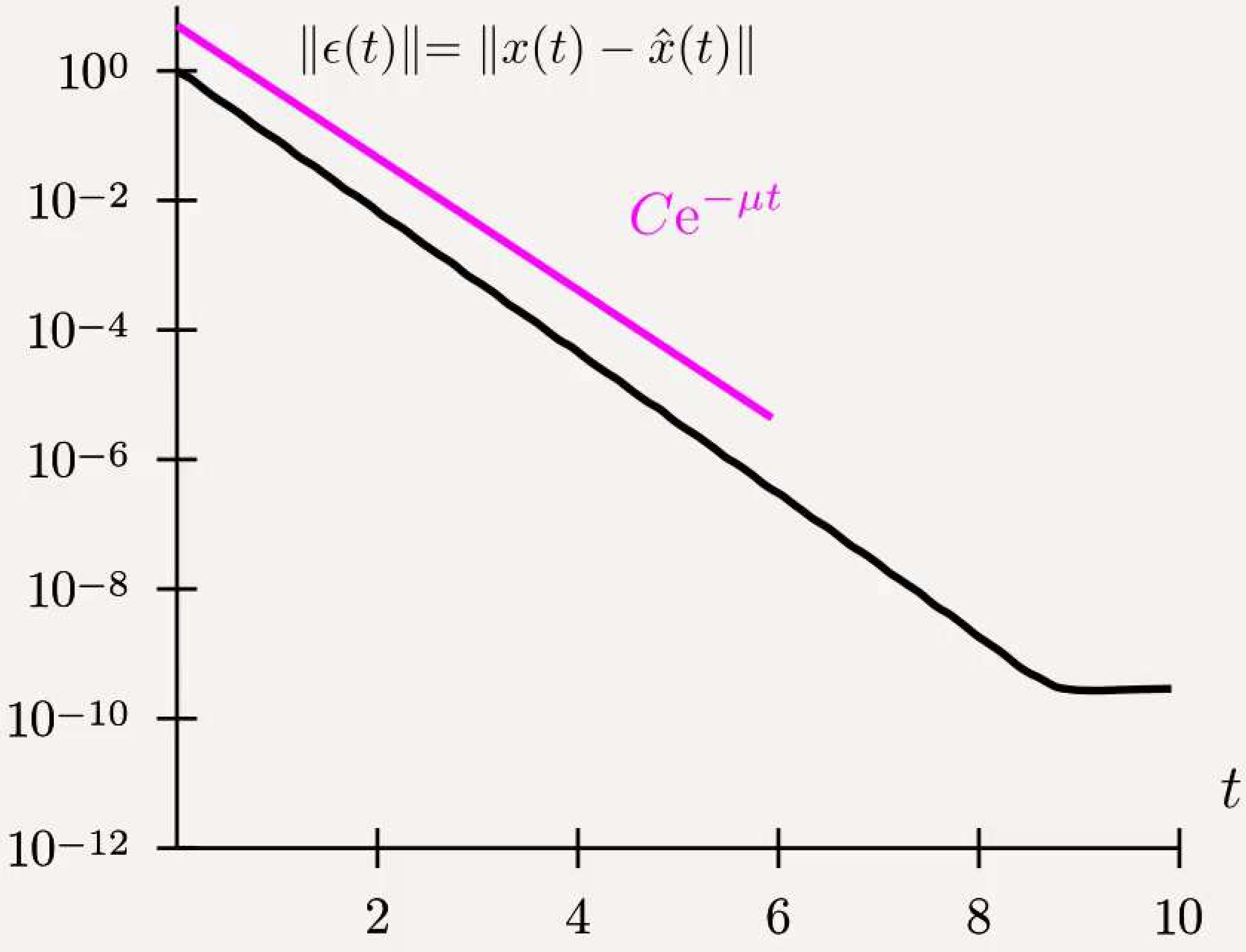
$$\begin{cases} \dot{x} = Ax(t) + Bu(t) \\ y(t) = Cx(t) \\ x(0) = x_0, \quad t \geq 0 \end{cases}$$

$$\begin{cases} \dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L[y(t) - \hat{y}(t)] \\ \hat{y}(t) = C\hat{x}(t) \\ \hat{x}(0) = \hat{x}_0, \quad t \geq 0 \end{cases}$$

[3] Yu, Pei, Xu. 'Estimation of velocity potential of water waves using a Luenberger-like observer.' (2020)

Data assimilation : Luenberger observer & dynamical systems





$$\|\epsilon(t)\| = \|e^{(A-LC)t}\epsilon(0)\| \leq \|x(0) - \hat{x}(0)\| \cdot \kappa(X) \cdot e^{-\mu t}$$

[3] Yu, Pei, Xu. 'Estimation of velocity potential of water waves using a Luenberger-like observer.' (2020)

Time parallelization : the Paraexp algorithm

$$\begin{cases} \dot{x}(t) = Mx(t) + g(t), & t \in [0,T] \\ x(0) = x_0 \end{cases}$$

$\rightarrow M \in \mathcal{M}_{m\times m}(\mathbb{C})$

$\rightarrow x(t), g(t) \in \mathbb{C}^m$

$$x(t) = \sum_{j=1}^{p} v_j(t) + \sum_{j=1}^{p} w_j(t)$$

Euler

Runge-Kutta $\leftarrow$ $\begin{cases} \dot{v}_j(t) = Mv_j(t) + g(t) \\ v_j(t_{j-1}) = 0 \end{cases}$

$\begin{cases} \dot{w}_j(t) = Mw_j(t) \\ w_j(t_{j-1}) = v_j(t_j) \end{cases} \Rightarrow w(t) = \mathrm{e}^{(tM)} w(0)$

'Type 1' on $[t_{j-1}, t_j]$

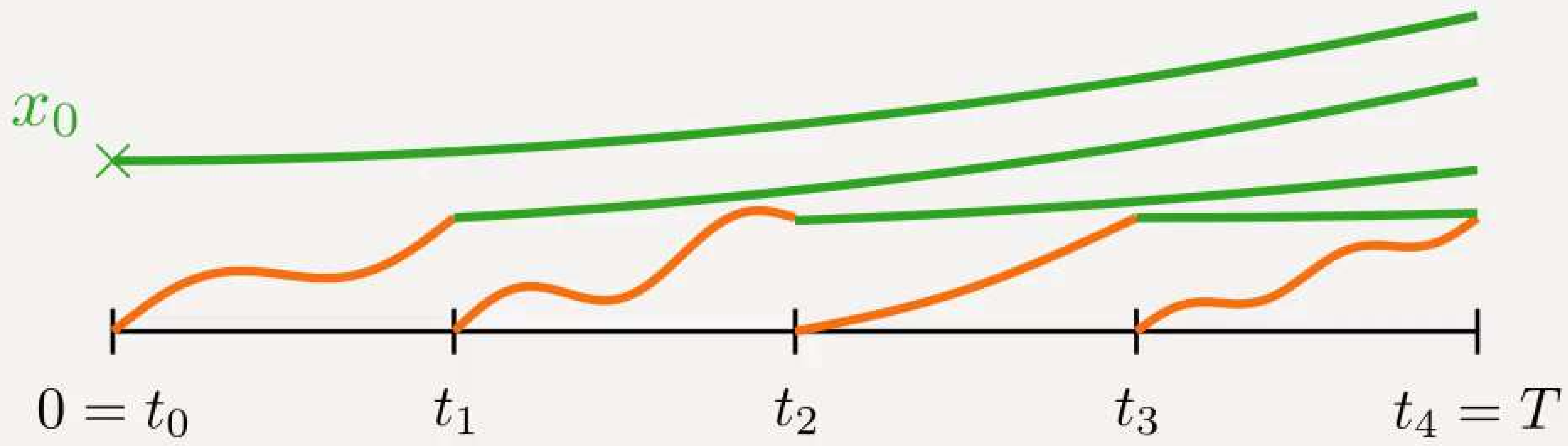
'Type 2' on $[t_{j-1}, T]$

$\downarrow$

Rational Krylov

Chebyshev polynomials

$p = 4$ computers :



[4] Gander, Güttel. 'Paraexp : a parallel integrator for linear initial-value problems.' (2013)

Coupling PinT & Data assimilation

Objective : PinT(data assimilation)

→ PinT algorithms are on a bounded time interval, data assimilation is on an unbounded time interval

→ To optimize PinT, we want to start with a coarse approximation and refine it over time

→ We want to preserve the property of the data assimilation scheme : in our case the convergence rate μ

Coupling PinT & Data assimilation

$\rightarrow$ PinT algorithms are on a bounded time interval, data assimilation is on an unbounded time interval

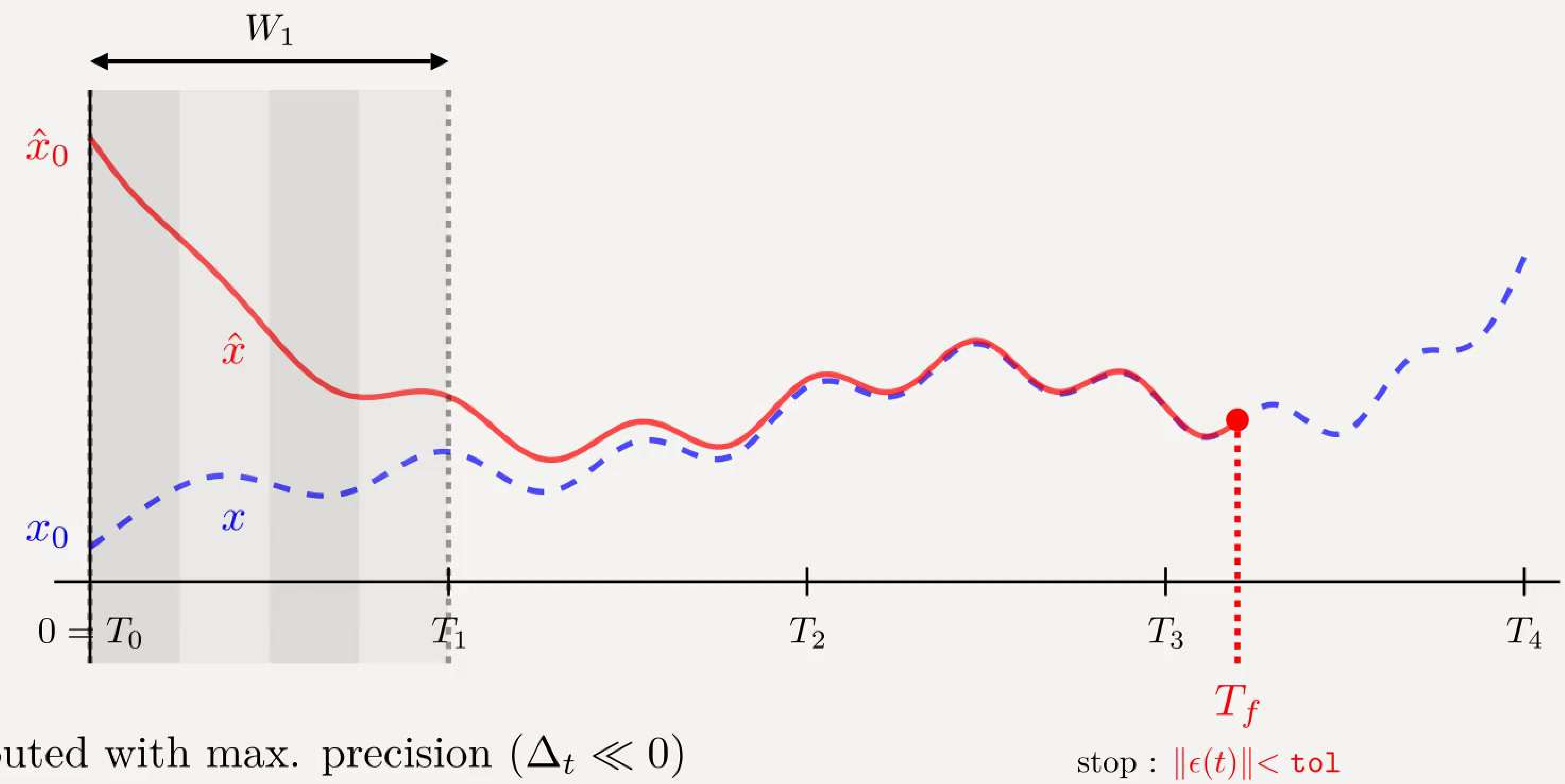
1) Divide the unbonded interval into 'windows' of size T :
$W_\ell = (T_{\ell-1}, T_\ell), \ell \leq 0$

2) Apply time parallelization scheme on each 'window'

3) Estimate the error at the end of each 'window' to go (or not) onto the next one

Coupling PinT & Data assimilation

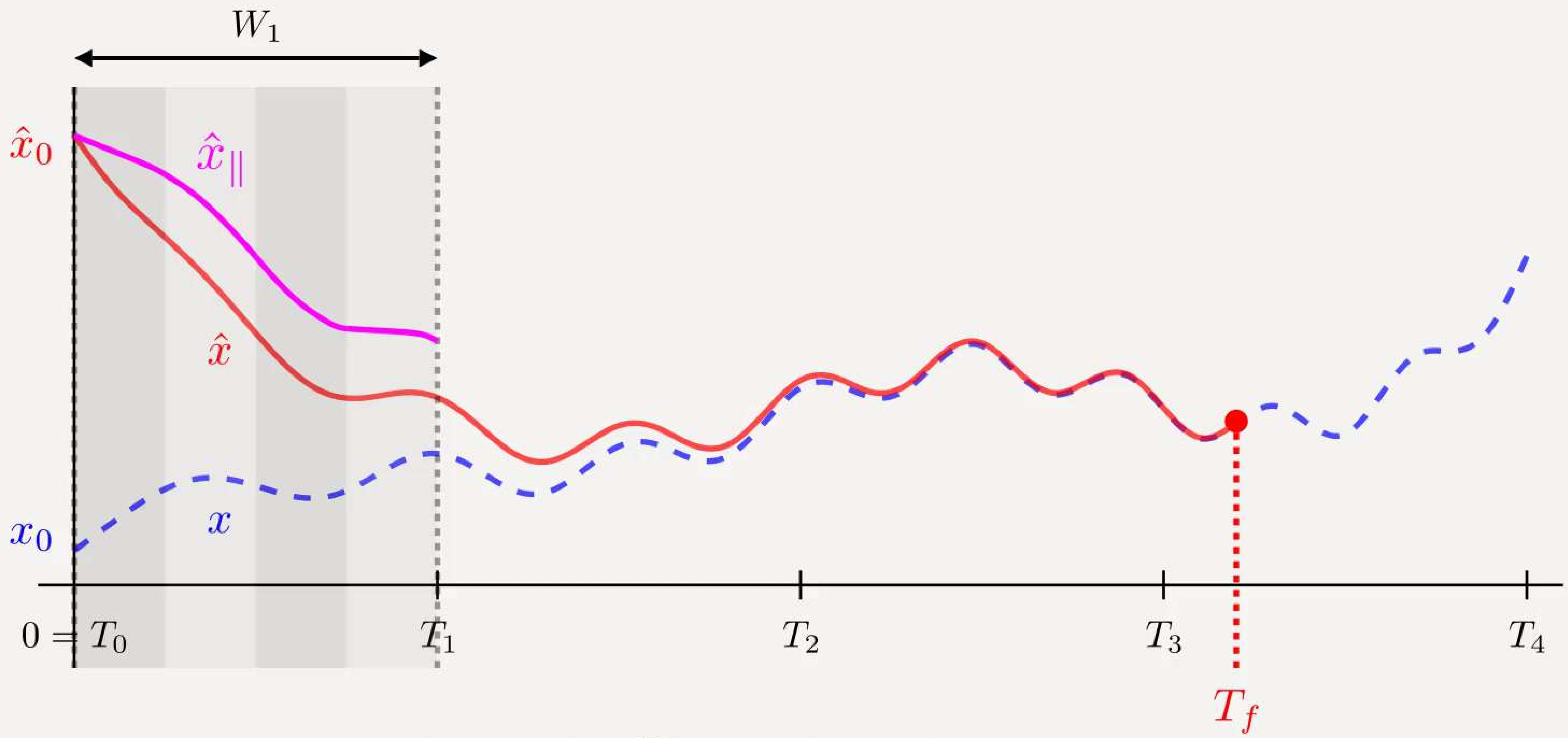
→ PinT algorithms are on a bounded time interval, data assimilation is on an unbounded time interval



$\hat{x}$ computed with max. precision ($\Delta_t \ll 0$)

Coupling PinT & Data assimilation

→ PinT algorithms are on a bounded time interval, data assimilation is on an unbounded time interval

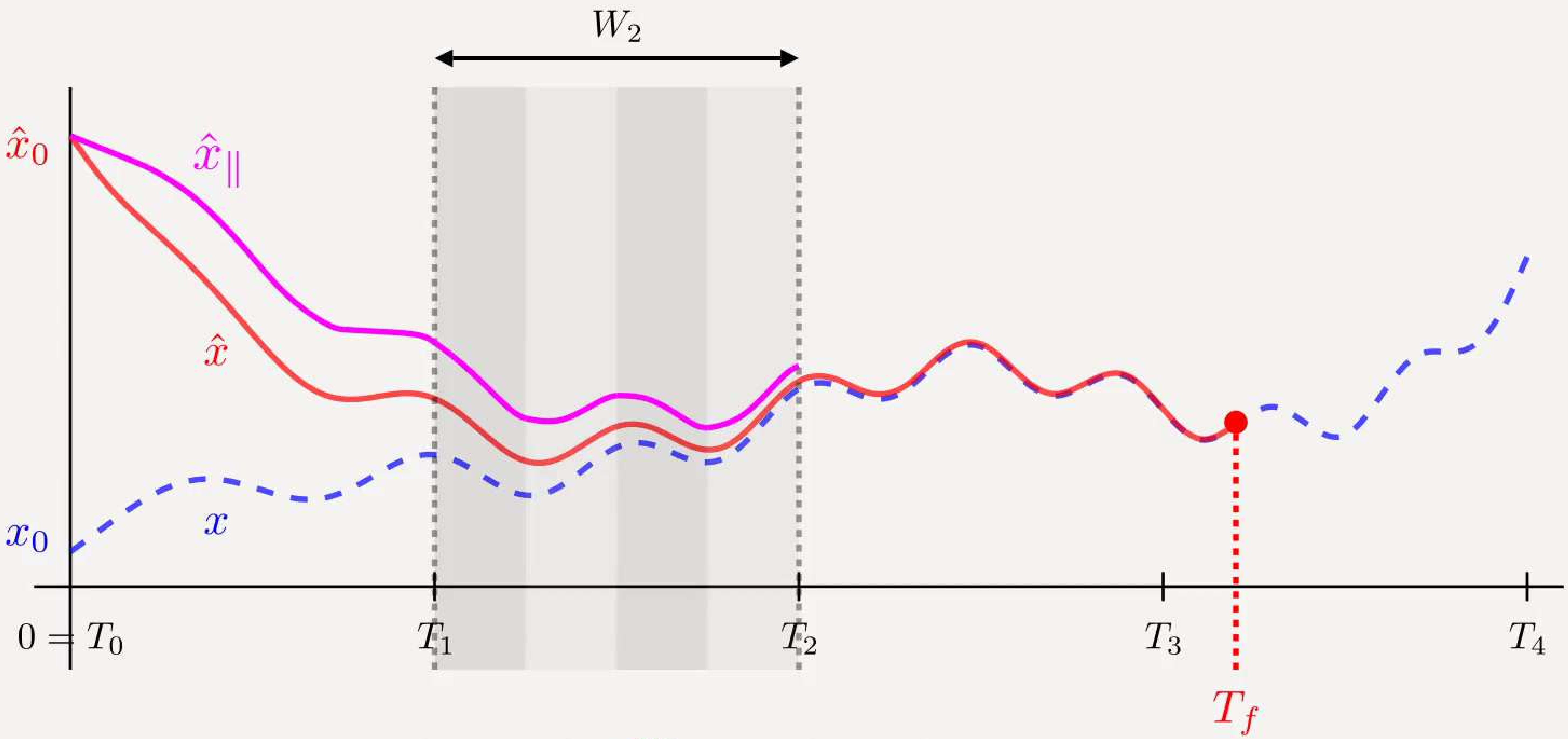


$\hat{x}_\parallel$: $\hat{x}_\parallel^{T1}$: with *some* precision ($\Delta_t \ll 0$), $\hat{x}_\parallel^{T2}$ exact (expm)

stop : $\|\epsilon(t)\| <$ `tol`

Coupling PinT & Data assimilation

→ PinT algorithms are on a bounded time interval, data assimilation is on an unbounded time interval

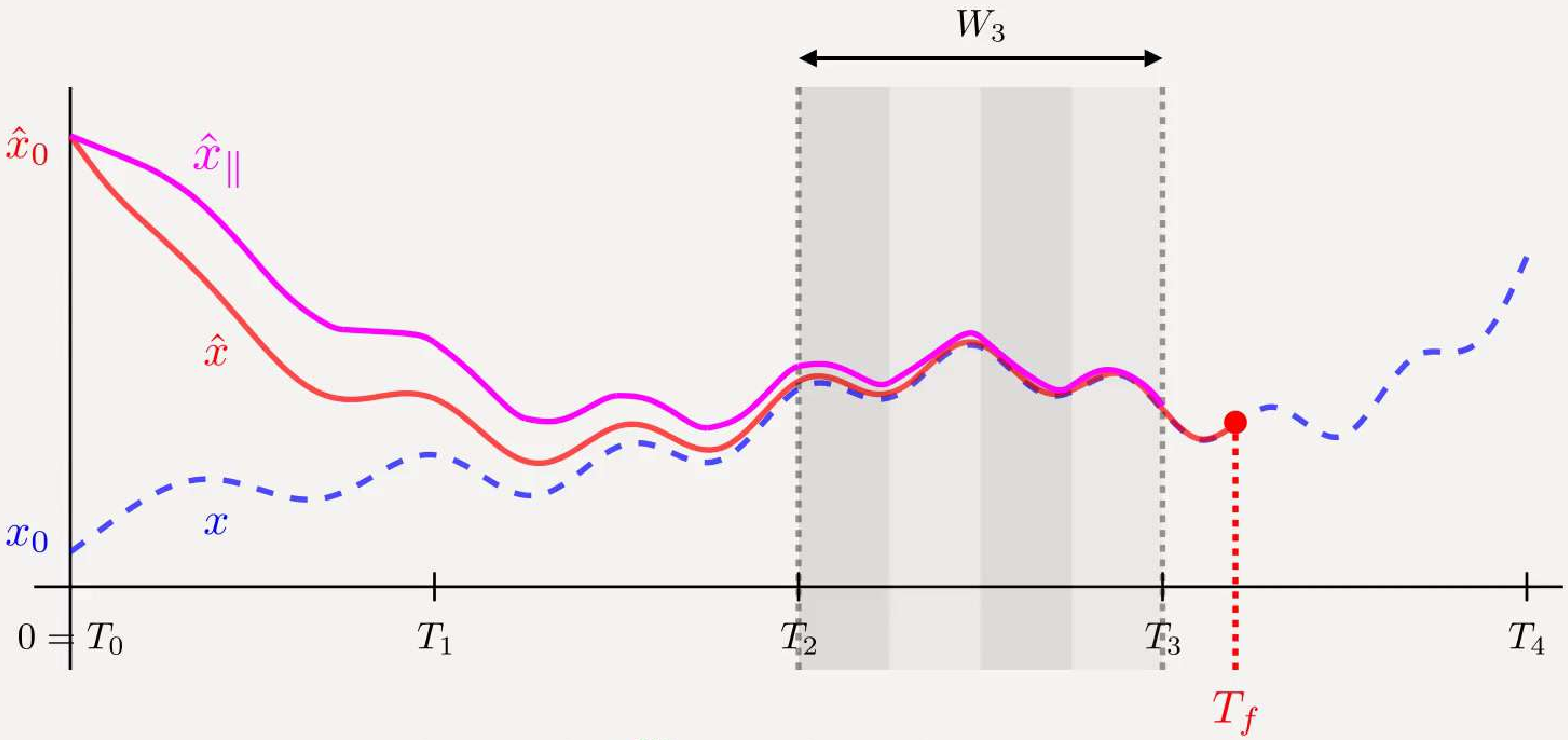


$\hat{x}_\parallel$: $\hat{x}_\parallel^{T1}$: with *some* precision ($\Delta_t \ll 0$), $\hat{x}_\parallel^{T2}$ exact (expm)

stop : $\|\epsilon(t)\| <$ `tol`

Coupling PinT & Data assimilation

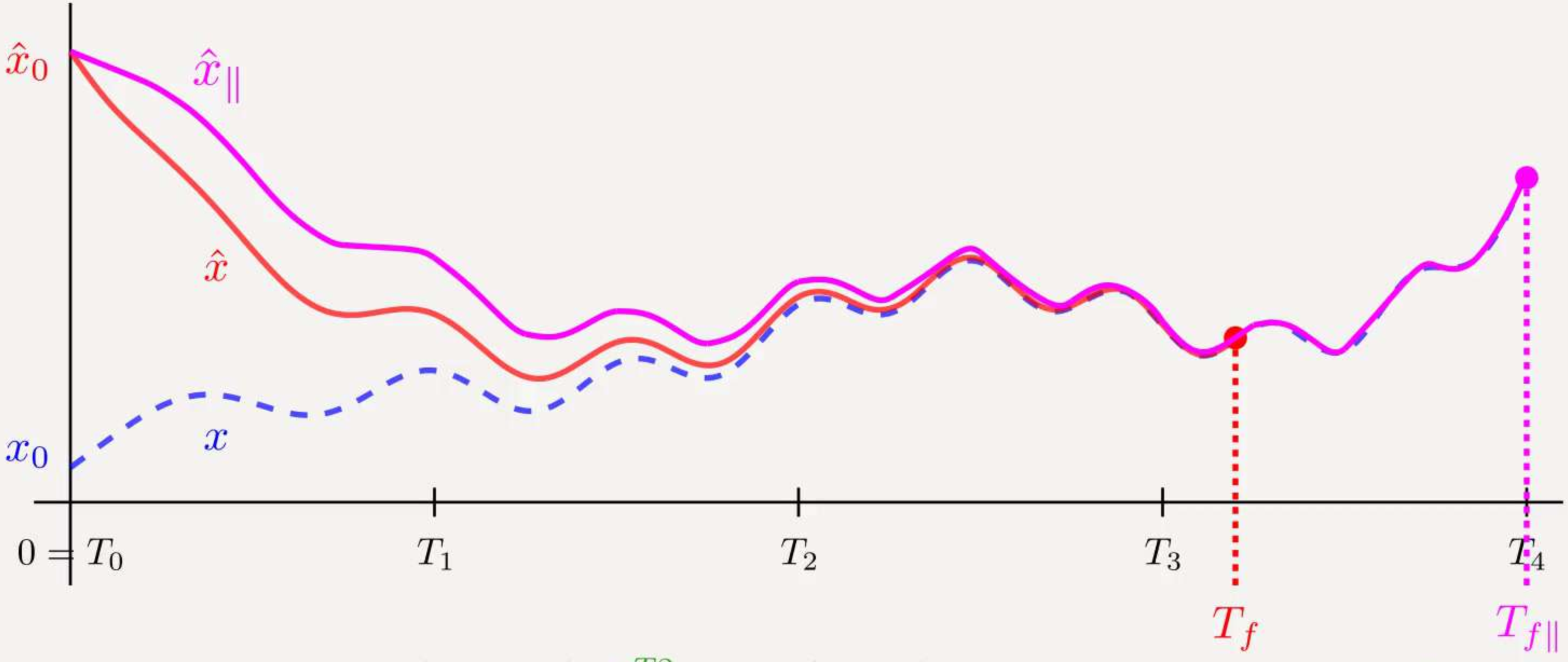
→ PinT algorithms are on a bounded time interval, data assimilation is on an unbounded time interval



$\hat{x}_\parallel$: $\hat{x}_\parallel^{T1}$: with *some* precision ($\Delta_t \ll 0$), $\hat{x}_\parallel^{T2}$ exact (expm) stop : $\|\epsilon(t)\| <$ `tol`

Coupling PinT & Data assimilation

→ PinT algorithms are on a bounded time interval, data assimilation is on an unbounded time interval



$\hat{x}_{\|}$: $\hat{x}_{\|}^{T1}$: with *some* precision ($\Delta_t \ll 0$), $\hat{x}_{\|}^{T2}$ exact (expm) stop : $\|\epsilon(t)\| <$ `tol` $\|\epsilon_{\|}(T_\ell)\| <$ `tol`

Coupling PinT & Data assimilation

→ To optimize PinT, we want to start with a coarse approximation and refine it over time, while conserving the convergence rate μ

$$\boxed{\|\epsilon_{\|}(T_\ell)\| = \|\hat{x}_{\|}(T_\ell) - x(T_\ell)\| \leq \|\epsilon(T_\ell)\| + \|\hat{x}(T_\ell) - \hat{x}_{\|}(T_\ell)\|}$$

$$\begin{aligned}
\epsilon_{\|} &= \hat{x}_{\|} - x \\
&= \hat{x}_{\|}^{T1} + \hat{x}_{\|}^{T2} - x \\
&= \hat{x}_{\|}^{T1} + \hat{x}_{\|}^{T2} - \hat{x}^{T1} + \hat{x}^{T1} - x \\
&= \hat{x}_{\|}^{T1} - \hat{x}^{T1} + \hat{x}_{\|}^{T2} + \hat{x}^{T1} - x \\
&= \hat{x}_{\|}^{T1} - \hat{x}^{T1} + \hat{x} - x
\end{aligned}$$

$$\boxed{\|\hat{x}(T_\ell) - \hat{x}_{\|}(T_\ell)\| = \|\hat{x}_{\|}^{T1}(T_\ell) - \hat{x}^{T1}(T_\ell)\|}$$

$$\boxed{\|\epsilon(T_\ell)\| \approx C\mathrm{e}^{-\mu T_\ell}}$$

$$\left\{\begin{aligned}
\hat{x}(T_\ell) &= \hat{x}^{T1}(T_\ell) + \hat{x}^{T2}(T_\ell) \\
\hat{x}_{\|}(T_\ell) &= \hat{x}_{\|}^{T1}(T_\ell) + \hat{x}_{\|}^{T2}(T_\ell)
\end{aligned}\right.$$

Coupling PinT & Data assimilation

→ To optimize PinT, we want to start with a coarse approximation and refine it over time, while conserving the convergence rate μ

$$\|\epsilon(T_\ell)\| \approx C\mathrm{e}^{-\mu T_\ell}$$

$$\|\epsilon_{\|}(T_\ell)\| = \|\hat{x}_{\|}(T_\ell) - x(T_\ell)\| \leq \|\epsilon(T_\ell)\| + \|\hat{x}(T_\ell) - \hat{x}_{\|}(T_\ell)\|$$

$$\|\hat{x}(T_\ell) - \hat{x}_{\|}(T_\ell)\| = \|\hat{x}_{\|}^{T1}(T_\ell) - \hat{x}^{T1}(T_\ell)\|$$

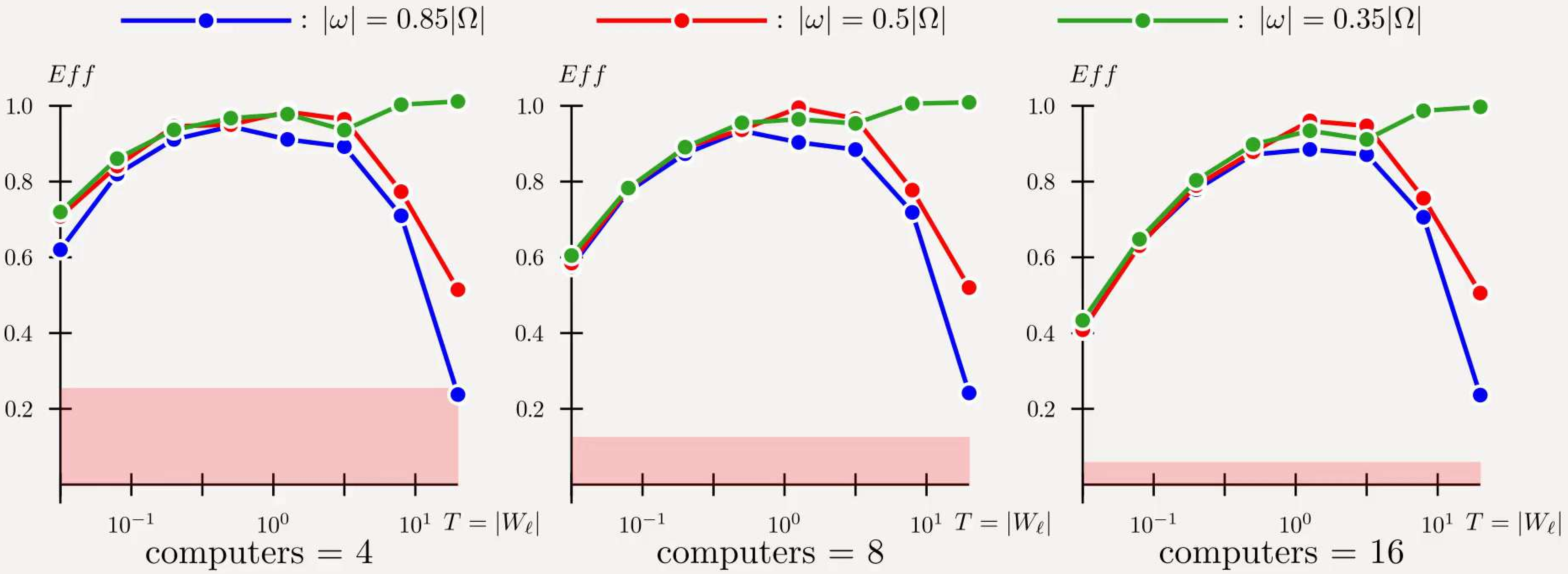
We must have $\|\hat{x}_{\|}^{T1}(T_\ell) - \hat{x}^{T1}(T_\ell)\| \approx C_{\|}\mathrm{e}^{-\mu T_\ell}$

If RK4 : $(\Delta_t)_{\ell+1} \leq \left(((\Delta_t)_\ell)^4 \mathrm{e}^{-\mu T}\right)^{1/4}, \quad \forall \ell \leq 1$

Results : a wave equation

2D Wave eq., on $\Omega = [0, 2\pi]^2$, $N_x = 9$, obs. space : ω

$$\text{Efficiency} = \frac{cputime(non\text{-}parallel)}{\#\ computers \times cputime(parallel)}$$

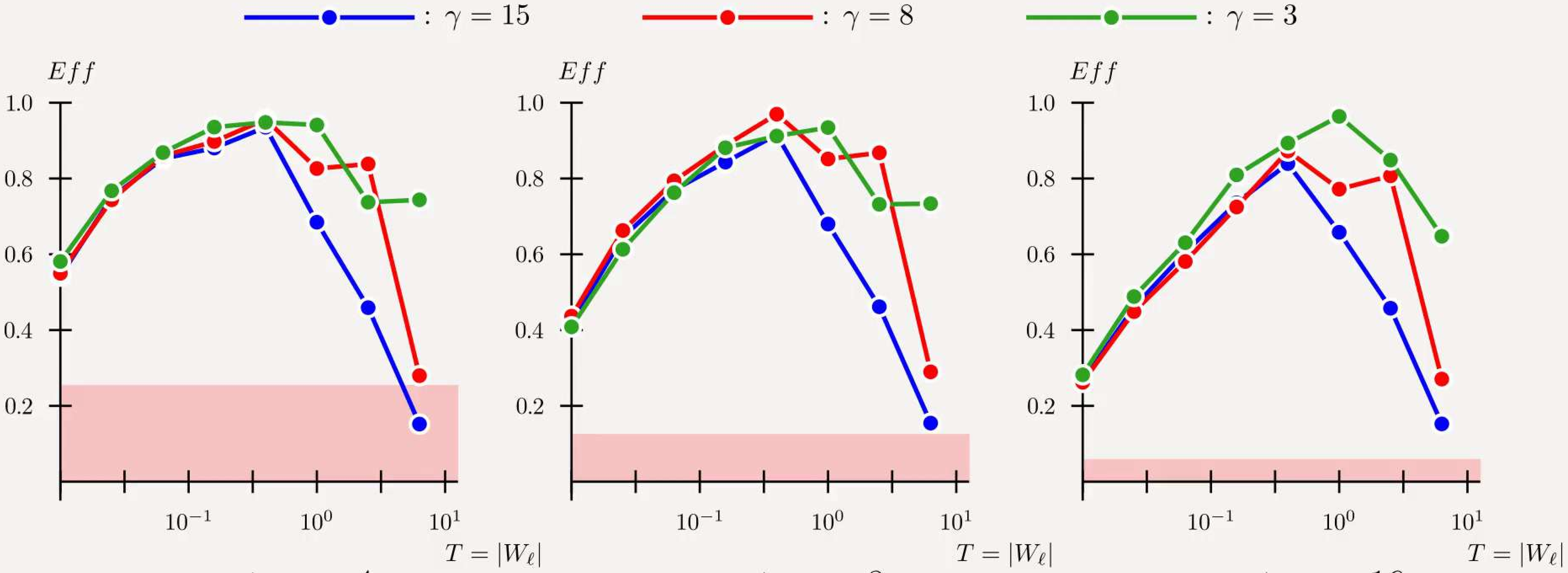


[5] Bardos, Lebau, Rauch. 'Sharp sufficient conditions for the observation, control, and stabilization of waves from the boundary.' (1992)

Results : linear water waves equations

LWWE, $N_x = 128$, $L = 1$, $y(x,t) = \eta(x,t)$

$$\text{Efficiency} = \frac{cputime(non\text{-}parallel)}{\#\ computers \times cputime(parallel)}$$



computers = 4 computers = 8 computers = 16

Results : following & leads

$\rightarrow$ Works similarly for heat equation (1D & 2D)

$\rightarrow$ Application to linear water wave equations : (in progress with N. Desmars)

- Convergence of the observer only when surface fully observed
  $(y(x,t) = \eta(x,t))$
- Naive description of the data assimilation setting
- Go to a probabilistic setting ? $(y(x,t) = C\eta(x,t) + \varepsilon(x,t))$

Thanks for your attention !

[1] Kautsky, Nichols, Van Dooren. 'Robust pole assignment in linear state feedback.' (1985).

[2] Haine, Ramdani. 'Observateurs itératifs, en horizon fini. Application à la reconstruction de données initiales pour des EDP d'évolution.' (2011).

[3] Yu, Pei, Xu. 'Estimation of velocity potential of water waves using a Luenberger-like observer.' (2020).

[4] Gander, Güttel. 'Paraexp : a parallel integrator for linear initial-value problems.' (2013).

[5] Bardos, Lebau, Rauch. 'Sharp sufficient conditions for the observation, control, and stabilization of waves from the boundary.' (1992).

[6] The Manim Community Developers. Manim – Mathematical Animation Framework (Version v0.15.2). https://www.manim.community/. (2022).

Annex : linearised water wave equations

$$\begin{cases} \Delta\Phi = 0 & \text{in the fluid domain} \\ \partial_t\Phi = -g\eta - \frac{1}{2}\left|\nabla\Phi\right|^2 & \text{on } z = \eta(x,t) \\ \partial_t\eta = \partial_z\Phi - \partial_x\Phi\partial_x\eta & \text{on } z = \eta(x,t) \\ \partial_z\Phi = 0 & \text{on } z = -h \end{cases}$$

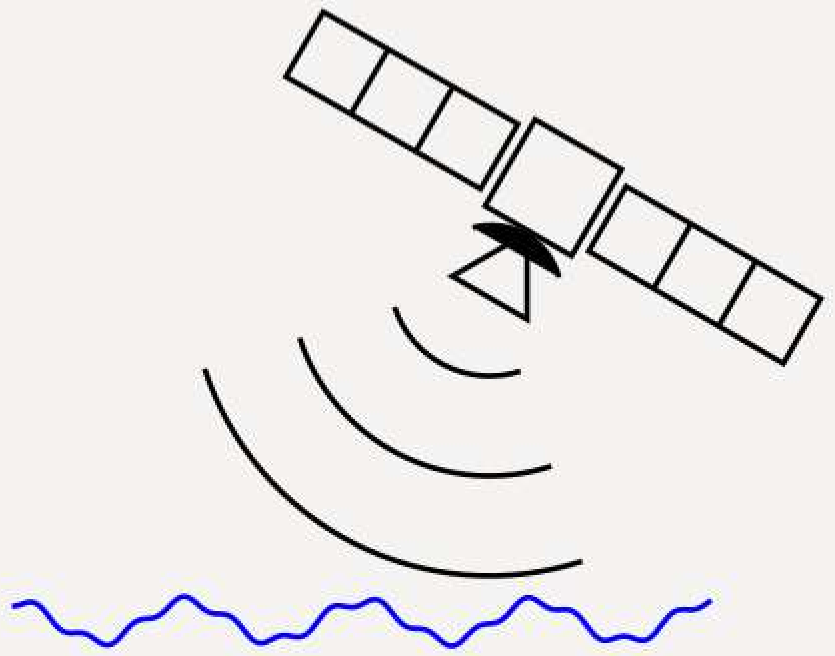
$$U(x,t) = [\Phi(x,0,t), \eta(x,t)]^T$$

$$\hat{U}(x,t) = [\hat{\Phi}(x,0,t), \hat{\eta}(x,t)]^T$$

$$A = \begin{pmatrix} 0 & -gI_d \\ \partial_z(\cdot) & 0 \end{pmatrix}$$

$$\begin{cases} \partial_t\Phi(x,z,t) + g\eta(x,t) = 0 & \text{on } z = 0 \\ \partial_t\eta(x,t) - \partial_z\Phi(x,z,t) = 0 & \text{on } z = 0 \end{cases}$$

$$\begin{cases} \partial_t U(x,t) = AU(x,t) \\ y(x,t) = \eta(x,t) \\ U_0(x) = U(x,0) \end{cases}$$



$$\begin{cases} \partial_t \hat{U}(x,t) = A\hat{U}(x,t) + L[y(x,t) - \hat{y}(x,t)] \\ \hat{y}(x,t) = \hat{\eta}(x,t) \\ U_0(x) = U(x,0) \end{cases}$$