Time parallelisation for optimal control and data assimilation

Julien Salomon

Joint work with M. Gander, F. Kwok, S. Reyes-Riffo

JLL & INRIA, ANGE project-team (Inria, Cerema, UPMC, CNRS)

PinT 2018, Roscoff

Problem 1 : control on a fixed, bounded interval $[0, T]$
Given $T > 0$, consider the optimal control problem associated with the cost functional

$$J(c) = \frac{1}{2}\|x(T) - x_{target}\|^2 + \frac{\alpha}{2}\int_0^T c^2(t)dt,$$

where the state function x evolution is described by an equation :

$$\dot{x}(t) = f(x(t), c(t)),$$

with initial condition $x(0) = x_{init}$.

Objective : Given an optimal control solver, combine it with a time-parallelization.

Problem 2 : assimilation on an unbounded interval $[t_0, +\infty)$

Given a (linear) dynamic

$$\dot{x}(t) = Ax(t) + Bu(t)$$

whose initial condition **is NOT known**, and an output

$$y(t) = Cx(t),$$

which **is known**.

Objective :	Combine observer approaches with a time-parallelization.

Previous works :

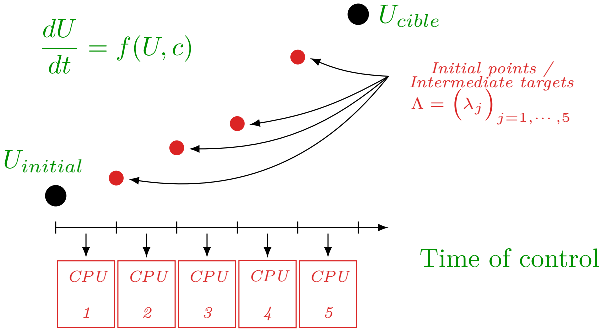
- Hackbusch, 1984 : Multgrid approach
- Borzì , 2003 : Multigrid for parabolic distributed
- Heinkenschloss, 2005 : Block symmetric Gauss-Seidel preconditioning
- Maday, Turinici, J.S. 2007 : intermediate states approach
- Mathew, Sarkis, 2010 : combination of a shooting method and parareal preconditionning

"Non-linear control" or "Bilinear control"

	Linear eq.	Non-linear eq.
"Linear" control	$\dot{y} = Ay + Bc$	$\dot{y} = f(y) + Bc$
Non-linear control	$\dot{y} = A(c)y$	$\dot{y} = f(y, c)$

- $y = y(t, x)$ state
- $c = c(t)$ or $c(t, x)$ control

Schematic description



Disclaimer : not a parareal algorithm.

Y. Maday, J. Salomon, G. Turinici, *SIAM J. Num. Anal., 45 (6), 2007.*

K. M. Riahi, J. Salomon, S. J. Glaser, D. Sugny, *Phys. Rev. A, 93 (4), 2016.*

Outline

"Linear control"

	Linear eq.	Non-linear eq.
"Linear" control	$\dot{y} = Ay + Bc$	$\dot{y} = f(y) + Bc$
Non-linear control	$\dot{y} = A(c)y$	$\dot{y} = f(y, c)$

- $y = y(t, x)$ state
- $c = c(t)$ or $c(t, x)$ control

The optimality condition then reads

$$\left\{\begin{array}{rcl} \dot{y}(t) & = & f(y(t)) + c(t), \\ \dot{\lambda}(t) & = & -(f(y(t))')^T \lambda(t), \\ \alpha c(t) & = & -\lambda(t). \end{array}\right.$$

→ **Elimination of c :**

$$\left\{\begin{array}{l} \dot{y} = f(y) - \dfrac{\lambda}{\alpha}, \\ \\ \dot{\lambda} = -(f(y)')^T \lambda, \end{array}\right.$$

and final condition $\lambda(T) = y(T) - y_{target}$.

$$\boxed{\text{Time discretization} \Rightarrow M_{\delta t} \begin{pmatrix} Y \\ \Lambda \end{pmatrix} = b}$$

Our approach is based on **two ideas** :

1. Partition the time interval $[0, T]$:
   $T_0 = 0 < T_1 < \ldots < T_L = T$.
2. Coarse approximation of the inverse : $M_{\delta t} \to M_{\Delta t}$.

Outline

1. Non-linear Control
2. Linear Control
   - Time sub-intervals decomposition
   - Use of a coarse solver
   - Numerical examples
3. Unbounded time domains and assimilation
   - Algorithm
   - Analysis
   - Numerical example

Boundary value problems notations : on the subinterval $[T_l, T_{l+1}]$ with initial condition $y(T_l) = y_l$ and final condition $\lambda(T_{l+1}) = \lambda_{l+1}$, we denote

$$\begin{pmatrix} y(T_{l+1}) \\ \lambda(T_l) \end{pmatrix} = \begin{pmatrix} P(y_l, \lambda_{l+1}) \\ Q(y_l, \lambda_{l+1}) \end{pmatrix}.$$

The optimality system is enriched :

$$\begin{array}{rclrcl} y_0 - y_{init} & = & 0 & & & \\ y_1 - P(y_0,\lambda_1) & = & 0 & \lambda_1 - Q(y_1,\lambda_2) & = & 0 \\ y_2 - P(y_1,\lambda_2) & = & 0 & \lambda_2 - Q(y_2,\lambda_3) & = & 0 \\ & \vdots & & & \vdots & \\ y_L - P(y_{L-1},\lambda_L) & = & 0 & \lambda_L - y_L + y_{target} & = & 0 \end{array} \quad (1)$$

That is : **a system of boundary value subproblems, satisfying matching conditions**.

Collecting the unknowns in the vector

$$(Y^T, \Lambda^T) := (y_0, y_1, y_2, \ldots, y_L, \lambda_1, \lambda_2, \ldots, \lambda_L),$$

we obtain the nonlinear system

$$\mathcal{F}(Y^T, \Lambda^T) := \begin{pmatrix} y_0 - y_{init} \\ y_1 - P(y_0, \lambda_1) \\ y_2 - P(y_1, \lambda_2) \\ \vdots \\ y_L - P(y_{L-1}, \lambda_L) \\ \lambda_1 - Q(y_1, \lambda_2) \\ \lambda_2 - Q(y_2, \lambda_3) \\ \vdots \\ \lambda_L - y_L + y_{target} \end{pmatrix} = 0.$$

Newton's method :

$$\mathcal{F}'\begin{pmatrix} Y^n \\ \Lambda^n \end{pmatrix}\begin{pmatrix} Y^{n+1}-Y^n \\ \Lambda^{n+1}-\Lambda^n \end{pmatrix} = -\mathcal{F}\begin{pmatrix} Y^n \\ \Lambda^n \end{pmatrix},$$

where the Jacobian matrix of $\mathcal{F}$ is given by

$$\mathcal{F}'\begin{pmatrix} Y \\ \Lambda \end{pmatrix} =$$

$$\left(\begin{array}{cccc|cccc}
1 & & & & & & & \\
-P_Y(Y_0,\Lambda_1) & 1 & & & -P_\Lambda(Y_0,\Lambda_1) & & & \\
 & \ddots & \ddots & & & \ddots & & \\
 & & -P_Y(Y_{N-1},\Lambda_N) & 1 & & & & -P_\Lambda(Y_{N-1},\Lambda_N) \\
\hline
 & -Q_Y(Y_1,\Lambda_2) & & & 1 & -Q_\Lambda(Y_1,\Lambda_2) & & \\
 & \ddots & & & & \ddots & \ddots & \\
 & & -Q_Y(Y_{N-1},\Lambda_N) & & & & 1 & -Q_\Lambda(Y_{N-1},\Lambda_N) \\
 & & & -1 & & & & 1
\end{array}\right)$$

Outline

Third idea : coarse approximation of the Jacobian

$$\mathcal{F}' \approx \text{finite difference}$$

Which concretely corresponds to :

$$\begin{aligned}
P_y(y_{\ell-1}^n, \lambda_\ell^n)(y_{\ell-1}^{n+1} - y_{\ell-1}^n) &\approx P^G(y_{\ell-1}^{n+1}, \lambda_\ell^n) - P^G(y_{\ell-1}^n, \lambda_\ell^n), \\
P_\lambda(y_{\ell-1}^n, \lambda_\ell^n)(\lambda_\ell^{n+1} - \lambda_\ell^n) &\approx P^G(y_{\ell-1}^n, \lambda_\ell^{n+1}) - P^G(y_{\ell-1}^n, \lambda_\ell^n), \\
Q_\lambda(y_{\ell-1}^n, \lambda_\ell^n)(\lambda_\ell^{n+1} - \lambda_\ell^n) &\approx Q^G(y_{\ell-1}^n, \lambda_\ell^{n+1}) - Q^G(y_{\ell-1}^n, \lambda_\ell^n), \\
Q_y(y_{\ell-1}^n, \lambda_\ell^n)(y_{\ell-1}^{n+1} - y_{\ell-1}^n) &\approx Q^G(y_{\ell-1}^{n+1}, \lambda_\ell^n) - Q^G(y_{\ell-1}^n, \lambda_\ell^n).
\end{aligned}$$

$\rightarrow$ *Inspiration from the Parareal algorithm :*

J.-L. Lions, Y. Maday, and G. Turinici. A "parareal" in time disretization of pde's. Comptes Rendus de l'Acad. des Sciences, 2001.

$\rightarrow$ *and its interpretation :*

M. Gander, S. Vandewalle, SISC 2003.

Partial summary :

- In parallel : all fine propagations on sub-intervals.
- Sequential part : only coarse solving.

Example : linear dynamics

$$\dot{y}(t) = \sigma y(t) + c(t).$$

Discretizing and setting :

$$X = \begin{pmatrix} Y \\ \Lambda \end{pmatrix},$$

we get :

$$X^{k+1} = \left(Id - M_{\Delta t}^{-1} M_{\delta t}\right) X^k + M_{\Delta t}^{-1} b.$$

Example : linear dynamics

$$\dot{y}(t) = \sigma y(t) + c(t).$$

Discretizing and setting :

$$X = \begin{pmatrix} Y \\ \Lambda \end{pmatrix},$$

we get :

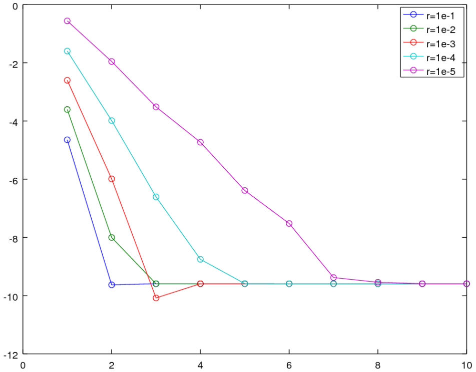
$$X^{k+1} = \left(Id - M_{\Delta t}^{-1} M_{\delta t}\right) X^k + M_{\Delta t}^{-1} b.$$

Analyze the eigenvalues of $Id - M_{\Delta t}^{-1} M_{\delta t}$!

Results for implicit Euler :

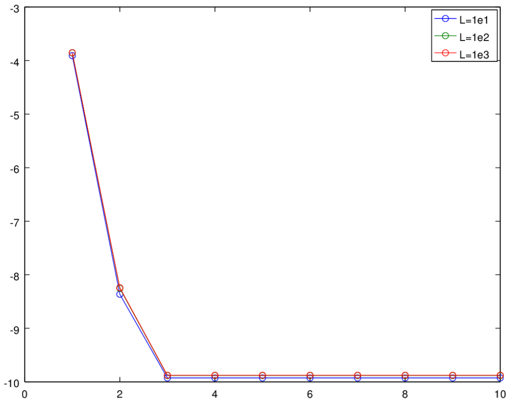
- Contraction factor : $\rho \leq C(\Delta t - \delta t)$
- For $\sigma < 0$, C can be chosen independent of σ
- For very large α, C can grow like $\log(\alpha)$ when the number of subdomains becomes large

F. Kwok, M. Gander, J. Salomon, to appear ...

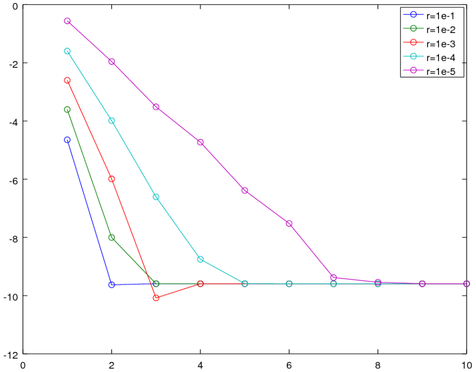


$$\dot{y}(t) = \sigma y(t) + c(t).$$

Convergence for various values of $r = \delta t/\Delta t$ for fixed $\delta t = \delta t_0$.

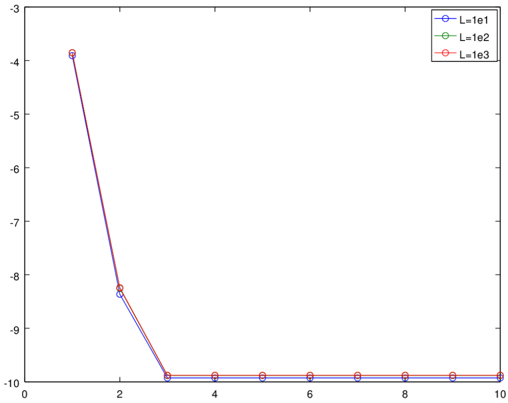


Convergence for various with respect to the number of iteration for various number of subintervals.



$$\dot{y}(t) = \sigma y(t) + c(t).$$

Convergence for various values of $r = \delta t / \Delta t$ for fixed $\delta t = \delta t_0$.



Convergence for various with respect to the number of iteration for various number of subintervals.

- Minimize

$$J(c) = \frac{1}{2}|y(1) - y_{\mathsf{target}}|^2 + \frac{1}{2}\int_0^1 |c(t)|^2 \, dt$$

with $y_{\mathsf{target}} = (100, 20)^T$, subject to the Lotka-Volterra equation

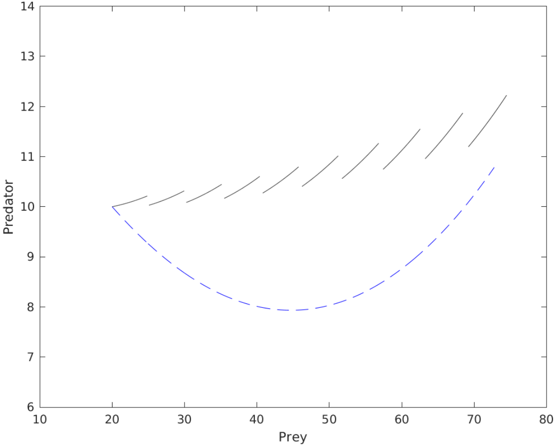
$$\dot{y}_1 = a_1 y_1 - b_1 y_1 y_2 + c_1, \quad \dot{y}_2 = a_2 y_1 y_2 - b_2 y_2 + c_2$$

with initial conditions $y(0) = (20, 10)^T$

- Backward Euler, $\delta t = 10^{-5}$

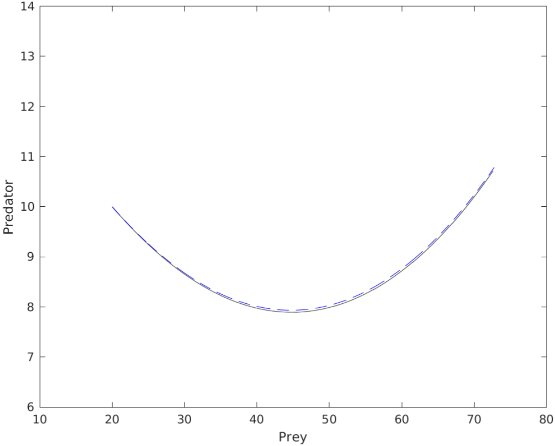
Vector example - $N = 10$, $r = \delta t/\Delta t = 0.01$

Iteration #1



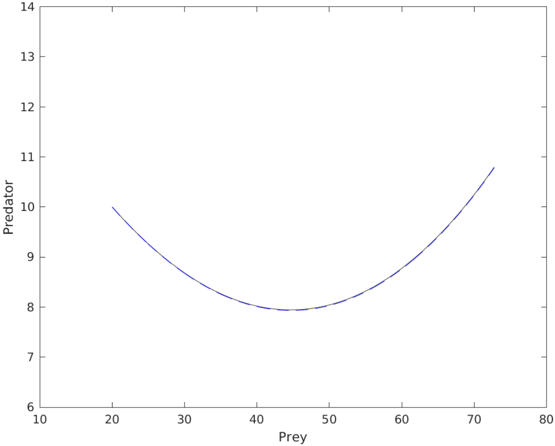
Vector example - $N = 10$, $r = \delta t/\Delta t = 0.01$

Iteration #2



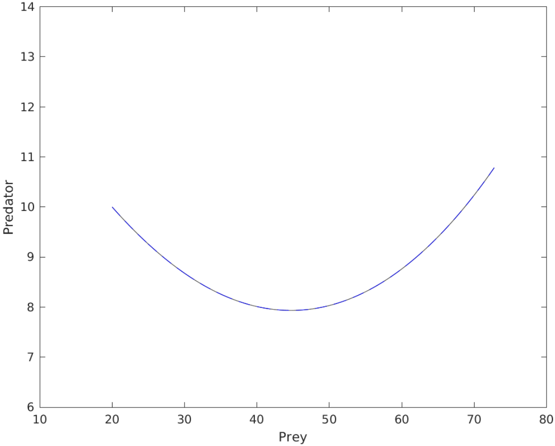
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #3



Vector example - $N = 10$, $r = \delta t/\Delta t = 0.01$

Iteration #4



- Minimize

$$J(c) = \frac{1}{2}|y(20) - y_{\mathsf{target}}|^2 + \frac{1}{2}\int_0^{20} |c(t)|^2 \, dt$$

with $y_{\mathsf{target}} = (100, 20)^T$, subject to the Lotka-Volterra equation

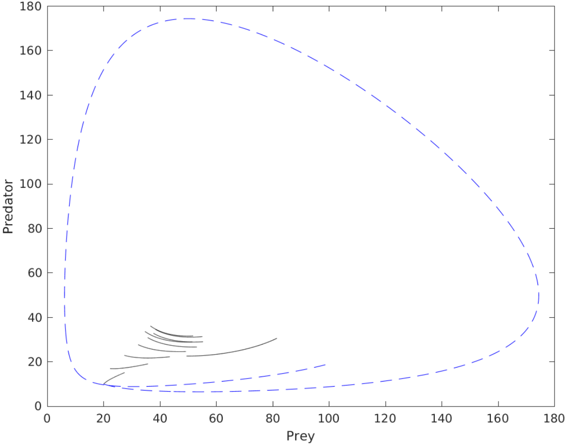
$$\dot{y}_1 = a_1 y_1 - b_1 y_1 y_2 + c_1, \quad \dot{y}_2 = a_2 y_1 y_2 - b_2 y_2 + c_2$$

with initial conditions $y(0) = (20, 10)^T$

- Backward Euler, $\delta t = 20 \cdot 10^{-5}$

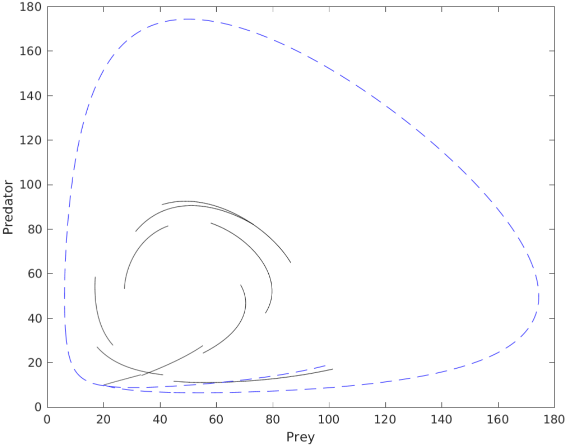
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #1



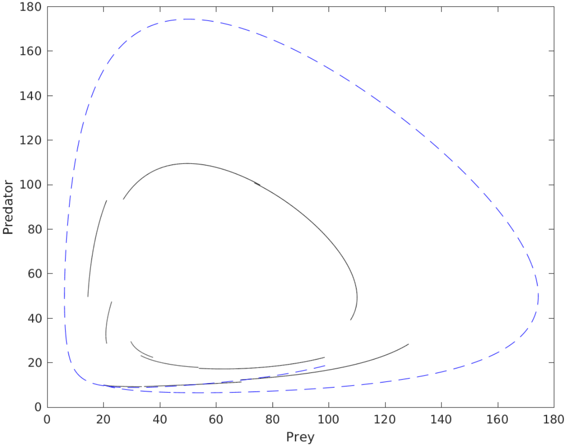
Vector example - $N = 10$, $r = \delta t/\Delta t = 0.01$

Iteration #2



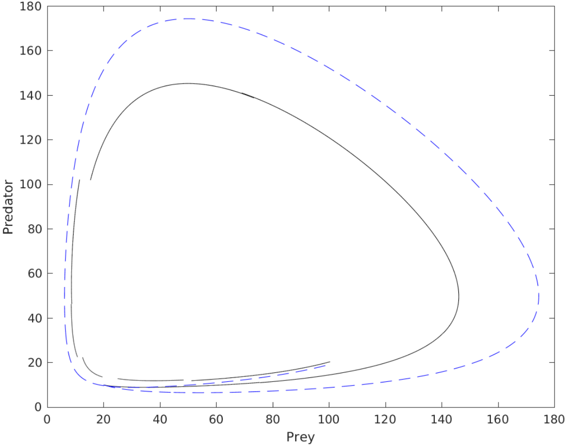
Vector example - $N = 10$, $r = \delta t/\Delta t = 0.01$

Iteration #3



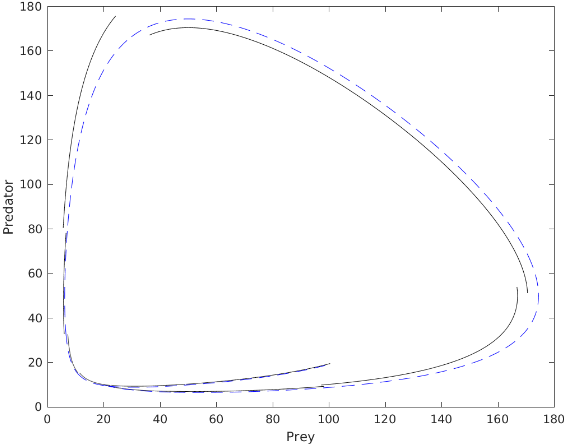
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #4



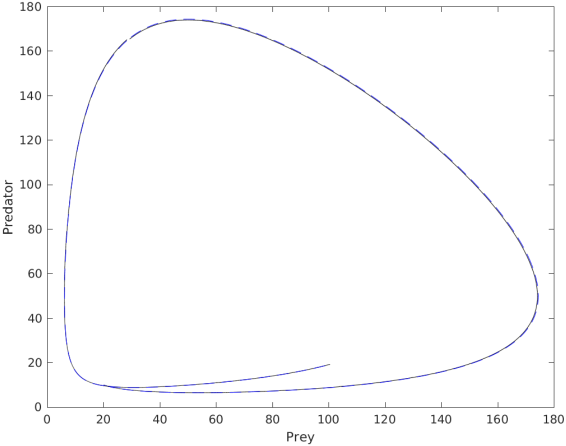
Vector example - $N = 10$, $r = \delta t/\Delta t = 0.01$

Iteration #5



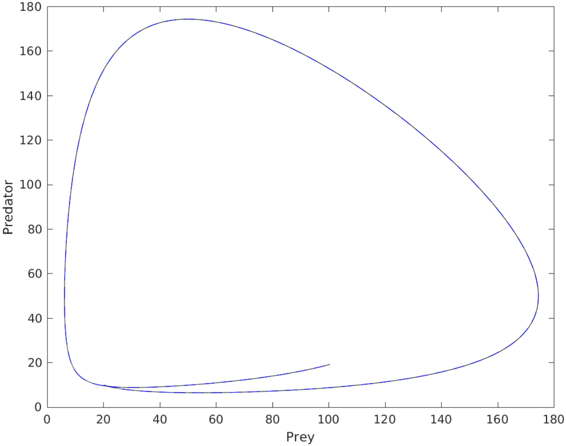
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #6



Vector example - $N = 10$, $r = \delta t/\Delta t = 0.01$

Iteration #7



Trick : Derivative Evaluation by **Gauss-Newton**

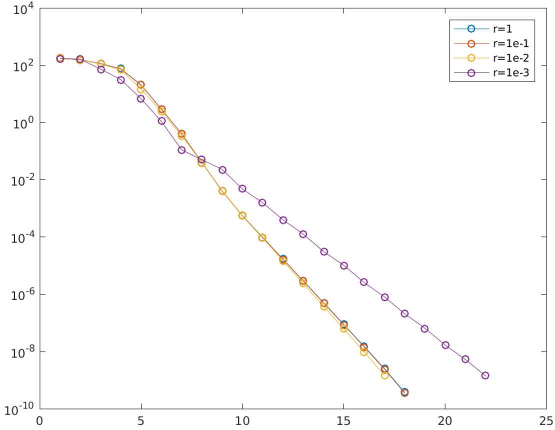
- Approximation : neglect 2nd derivatives

$$\frac{dy'}{dt} = f'(y)y' - \frac{\lambda'}{\alpha}, \qquad y'(0) = Y_n^{k+1} - Y_n^k,$$

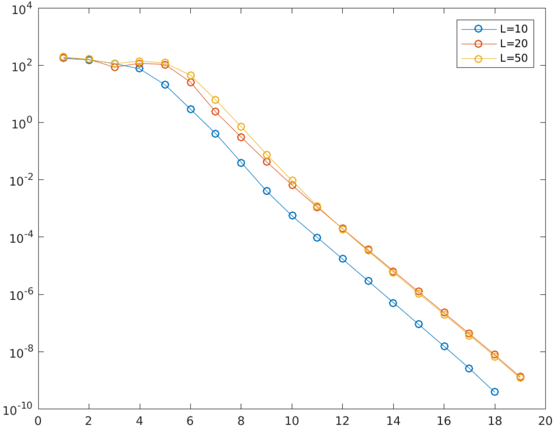
$$\frac{d\lambda'}{dt} = -(f'(y))^T \lambda' - \cancel{(f''(y, y'))^T \lambda}, \quad \lambda'(T) = \Lambda_{n+1}^{k+1} - \Lambda_{n+1}^k.$$

- Simplified ODE for λ' independent of y'
- Approximate derivatives in one backward-forward sweep !

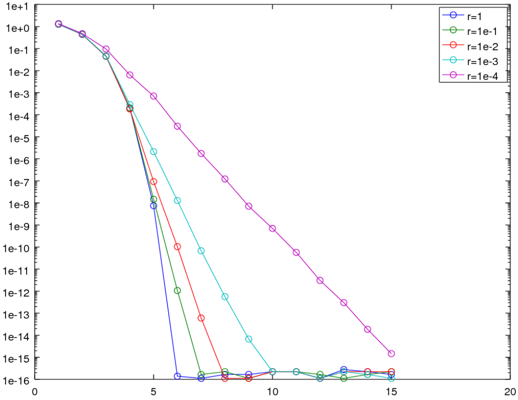
$N = 10$ subdomains, varying $r = \delta t / \Delta t$



$\delta t/\Delta t = 0.01$, varying # subdomains



True Newton :


1e+1
1e+0
1e-1
1e-2
1e-3
1e-4
1e-5
1e-6
1e-7
1e-8
1e-9
1e-10
1e-11
1e-12
1e-13
1e-14
1e-15
1e-16
0
5
10
15
20
r=1
r=1e-1
r=1e-2
r=1e-3
r=1e-4

Given a (linear) dynamic

$$\dot{x}(t) = Ax(t) + Bu(t)$$

whose initial condition **is NOT known**, and an output

$$y(t) = Cx(t),$$

which **is known** : data to be assimilated.
→ **Solver :** Luenberger observer

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L\left(C\hat{x}(t) - y(t)\right).$$

In general :

$$\hat{x}(t_0) \neq x(t_0).$$

Theoretical result : Assume the observability condition

$$rank \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{pmatrix} = n,$$

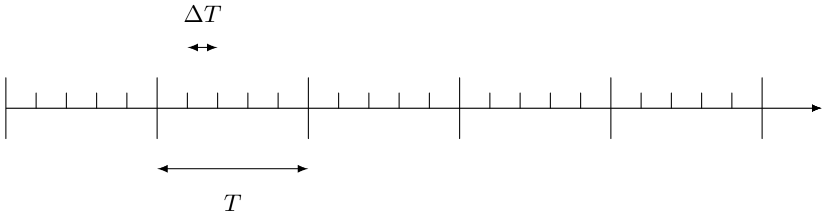
then **there exists** L such that

$$\rho\left(\exp(A - LC)\right) \leq 1 \;\Rightarrow \|x(t) - \hat{x}(t)\| \leq \kappa \; e^{-\lambda t} \|x(0) - \hat{x}(0)\|,$$

with $\lambda = \min_{\alpha \in \mathrm{spec}(A-LC)} |\alpha|$, $\kappa = \mathrm{Cond}(A - LC)$.

$\rightarrow$ *Standard algorithms to design L : Routh's or Hurwitz criterion, Ackermann's formula, LQ theory...*

Idea : In order to accelerate the assimilation, simulate the observer using time-parallelization on **Windows**.



Consider the **Parareal algorithm** and introduce

- **Windows :** interval of length T on which are applied k_ℓ iterations of parareal algorithm.
- **Subintervals :** set of N intervals of length ΔT that make up the decomposition on which the iterations of the algorithm are based.
- **Two other time steps :** Δt and δt used in the coarse and the fine solver respectively.

Mandatory :

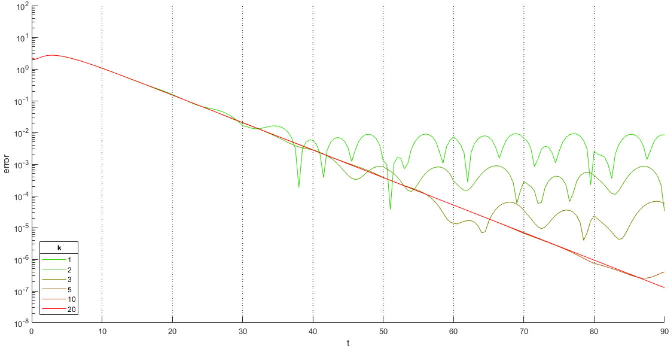
$$k_\ell << N.$$

We proceed as follows : Suppose we are on the window ℓ

$$W_\ell := [t_\ell, t_{\ell+1} = t_\ell + T],$$

1. Consider an approximation $\hat{x}_\parallel(t_\ell)$ of $\hat{x}(t_\ell)$.
2. Apply k_ℓ iterations of parareal algorithm to get an approximation of $\hat{x}$ on W_ℓ.
3. Let the final state $\hat{x}_\parallel(t_{\ell+1})$ be an initial point for the next window.

What happens when $k_\ell = k_{\max}$ is fixed for all windows ?



This is not surprising : k_{max} parareal iterations introduce a constant error.

Outline

Lemma : Denote by p_ℓ^n the **jump** of the fine (discontinuous) trajectory $\hat{x}_\parallel(t)$ at time $t_\ell + n\Delta T$.
Suppose that

$$\forall 1 \leq n \leq N, \lim_{\ell \to +\infty} p_\ell^n \to 0. \qquad (\star)$$

Then

$$\lim_{t \to +\infty} \hat{x}_\parallel(t) - x(t) \to 0.$$

$\to$ *Condition* $(\star)$ *automatically holds if* $k_\ell \to N$.

Proof : Define $\varepsilon_{\|}(t) = \hat{x}_{\|}(t) - x(t)$. We have

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ \dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L\left(C\hat{x}(t) - y(t)\right) + \delta p(t), \end{cases}$$

Substracting, we get :

$$\dot{\varepsilon}_{\|}(t) = (A - LC)\varepsilon_{\|}(t) - \delta p(t),$$

so that integrating over $[t_\ell + n\Delta T, t_\ell + (n+1)\Delta T]$ gives :

$$\begin{aligned} \varepsilon_{\|}(t_\ell + (n+1)\Delta T) = & \exp\left((A - LC)\Delta T\right)\varepsilon_{\|}(t_\ell + n\Delta T) \\ & + \exp\left((A - LC)\Delta T\right)p_\ell^n - p_\ell^{n+1} \end{aligned}$$

Define $s_n = \varepsilon_{\|}(t_\ell + n\Delta T) + p_\ell^n$:

$$\begin{aligned} \Rightarrow s_{n+1} &= \exp\left((A - LC)\Delta T\right)s_n \\ \|s_{n+1}\| &\leq \kappa e^{-\lambda\Delta T}\|s_n\|. \end{aligned}$$

Strategy : From

$$s_{n+1} = \exp\left((A - LC)\Delta T\right) s_n,$$

we get

$$s_{N.\ell+n} = \exp\left((N.\ell + n)(A - LC)\Delta T\right) s_0,$$

hence :

$$\varepsilon_{\|}(t_\ell + n\Delta T) = \exp\left((N.\ell + n)(A - LC)\Delta T\right) \varepsilon_{\|}(t_0) - p_\ell^n.$$

$\rightarrow$ If we want to **keep Luenberger's observer rate of convergence**, we need to impose :

$$\|p_\ell^n\| \leq \widetilde{\kappa} e^{-(N.\ell+n)\lambda\Delta T} \|p_0^0\|. \qquad (\star\star)$$

$\rightarrow$ On each window, define k_ℓ as the minimal integer such that $(\star\star)$ holds.

F. Kwok, S. Reyes-Riffo, J. Salomon, to appear ...

Example : $N = 20$.

$$A = \begin{pmatrix} 0 & 1 \\ -1 & -2 \end{pmatrix}$$

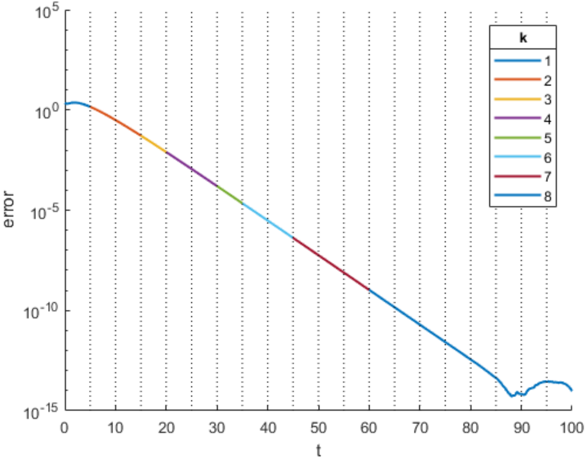
$$B = \begin{pmatrix} 0 \\ 1 \end{pmatrix},\ C = \begin{pmatrix} 0 & 1 \end{pmatrix},\ L = \begin{pmatrix} 0.8 \\ -1.1 \end{pmatrix}$$

$$u(t) = 3 + 0.5\sin(0.75t)$$

$$T = 5,\ \Delta T = \frac{T}{N} = 0.25.$$

$$\Delta t = \Delta T,\ \delta t = \frac{\Delta t}{25}.$$

Example : $N = 20$.


k
1
2
3
4
5
6
7
8
error
t
10^5
10^0
10^-5
10^-10
10^-15
0
10
20
30
40
50
60
70
80
90
100

Efficiency : CPU time to reach $\|\varepsilon_{\|}\| = \|x(t) - \hat{x}_{\|}(t)\| \leq 10^{-12}$.

- $CPU_{\|}$: 0.2363
- CPU_{seq} : 0.8361
- Ratio : 0.2826
- Efficiency : 17%

Trugarez-vras !