The parareal in time algorithm for fast simulations of time dependent PDE's: Basics and new developments and directions

Y. Maday[1]

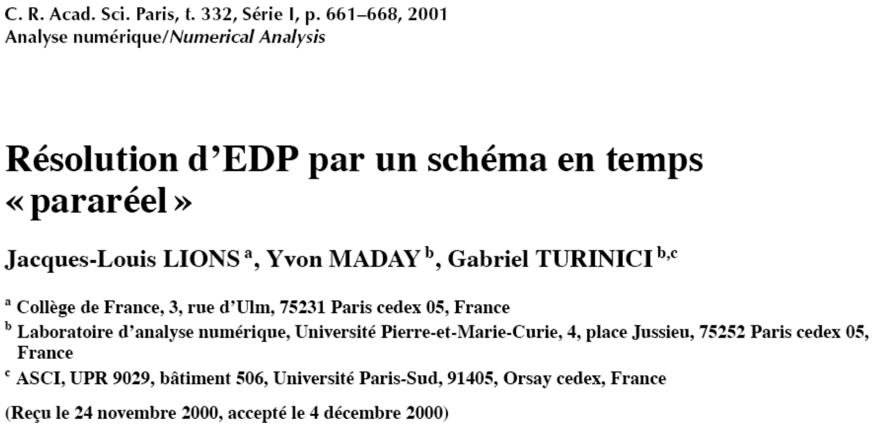
[1]Laboratoire Jacques-Louis Lions
Université Pierre et Marie Curie, Paris, France, and
Division of Applied Maths, Brown University, RI

ORNL

Collaboration

- Leonado Baffico
- Guillaume Bal
- Paul Fischer
- Frederic Hecht
- Sidi M. Kaber
- Julien Salomon
- Gabriel Turinici
- Gilles Zerah

Collaboration



C. R. Acad. Sci. Paris, t. 332, Série I, p. 661–668, 2001
Analyse numérique/*Numerical Analysis*

Résolution d'EDP par un schéma en temps « pararéel »

Jacques-Louis LIONS [a], Yvon MADAY [b], Gabriel TURINICI [b,c]

[a] Collège de France, 3, rue d'Ulm, 75231 Paris cedex 05, France
[b] Laboratoire d'analyse numérique, Université Pierre-et-Marie-Curie, 4, place Jussieu, 75252 Paris cedex 05, France
[c] ASCI, UPR 9029, bâtiment 506, Université Paris-Sud, 91405, Orsay cedex, France

(Reçu le 24 novembre 2000, accepté le 4 décembre 2000)

Basics on the algorithm.

- Domain decomposition method is the general technique for the parallelisation of problems modeled through Partial Differential Equations.
- The new generation of parallel computers provides more processors than you can fill up efficiently with current algorithms.
- A new type of parallelisation has to be added to this sole argument.
- The time directions has not received much attention right now for large parallisation, mainly due to the intrinsic sequential nature of this direction.

Basics on the algorithm.

- Domain decomposition method is the general technique for the parallelisation of problems modeled through Partial Differential Equations.
- The new generation of parallel computers provides more processors than you can fill up efficiently with current algorithms.
- A new type of parallelisation has to be added to this sole argument.
- The time directions has not received much attention right now for large parallisation, mainly due to the intrinsic sequential nature of this direction.

Basics on the algorithm.

- Consider the following time dependant problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = y^0$$

where, for the sake of simplicity, $\underline{\mathcal{A}\text{ does not depend on time.}}$

- We introduce the propagator $\mathcal{E}$ such that, $\mathcal{E}_\tau(\mu)$ is the solution, at time τ of the problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = \mu \longrightarrow \mathcal{E}_\tau(\mu) = y(\tau)$$

- Due to time invariance, it is well known that

$$\forall \tau' < \tau, \quad \mathcal{E}_\tau = \mathcal{E}_{\tau-\tau'} \circ \mathcal{E}_{\tau'}$$

- For instance, let $0 = T_0 < T_1 < \ldots < T_n < \ldots < T_N = T$ be special times at which we are interested to consider snapshots of the solution $y(T_n)$, then we have

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

Basics on the algorithm.

- Consider the following time dependant problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = y^0$$

where, for the sake of simplicity, $\mathcal{A}$ does not depend on time.

- We introduce the propagator $\mathcal{E}$ such that, $\mathcal{E}_\tau(\mu)$ is the solution, at time τ of the problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = \mu \longrightarrow \mathcal{E}_\tau(\mu) = y(\tau)$$

- Due to time invariance, it is well known that

$$\forall \tau' < \tau, \quad \mathcal{E}_\tau = \mathcal{E}_{\tau - \tau'} \circ \mathcal{E}_{\tau'}$$

- For instance, let $0 = T_0 < T_1 < ... < T_n < ... < T_N = T$ be special times at which we are interested to consider snapshots of the solution $y(T_n)$, then we have

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

Basics on the algorithm.

- Consider the following time dependant problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = y^0$$

where, for the sake of simplicity, $\underline{\mathcal{A}\text{ does not depend on time.}}$

- We introduce the propagator $\mathcal{E}$ such that, $\mathcal{E}_\tau(\mu)$ is the solution, at time τ of the problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = \mu \longrightarrow \mathcal{E}_\tau(\mu) = y(\tau)$$

- Due to time invariance, it is well known that

$$\forall \tau' < \tau, \quad \mathcal{E}_\tau = \mathcal{E}_{\tau-\tau'} \circ \mathcal{E}_{\tau'}$$

- For instance, let $0 = T_0 < T_1 < ... < T_n < ... < T_N = T$ be special times at which we are interested to consider snapshots of the solution $y(T_n)$, then we have

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

Basics on the algorithm.

- Consider the following time dependant problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = y^0$$

  where, for the sake of simplicity, $\underline{\mathcal{A}\text{ does not depend on time.}}$

- We introduce the propagator $\mathcal{E}$ such that, $\mathcal{E}_\tau(\mu)$ is the solution, at time τ of the problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = \mu \longrightarrow \mathcal{E}_\tau(\mu) = y(\tau)$$

- Due to time invariance, it is well known that

$$\forall \tau' < \tau, \quad \mathcal{E}_\tau = \mathcal{E}_{\tau-\tau'} \circ \mathcal{E}_{\tau'}$$

- For instance, let $0 = T_0 < T_1 < ... < T_n < ... < T_N = T$ be special times at which we are interested to consider snapshots of the solution $y(T_n)$, then we have

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

Basics on the algorithm.

- Consider the following time dependant problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = y^0$$

where, for the sake of simplicity, $\underline{\mathcal{A}\text{ does not depend on time.}}$

- We introduce the propagator $\mathcal{E}$ such that, $\mathcal{E}_\tau(\mu)$ is the solution, at time τ of the problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = \mu \longrightarrow \mathcal{E}_\tau(\mu) = y(\tau)$$

- Due to time invariance, it is well known that

$$\forall \tau' < \tau, \quad \mathcal{E}_\tau = \mathcal{E}_{\tau-\tau'} \circ \mathcal{E}_{\tau'}$$

- For instance, let $0 = T_0 < T_1 < ... < T_n < ... < T_N = T$ be special times at which we are interested to consider snapshots of the solution $y(T_n)$, then we have

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

Basics on the algorithm.

- Consider the following time dependant problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = y^0$$

where, for the sake of simplicity, $\underline{\mathcal{A}\text{ does not depend on time.}}$

- We introduce the propagator $\mathcal{E}$ such that, $\mathcal{E}_\tau(\mu)$ is the solution, at time τ of the problem

$$\frac{\partial y}{\partial t} + \mathcal{A}y = 0, \qquad y(0) = \mu \longrightarrow \mathcal{E}_\tau(\mu) = y(\tau)$$

- Due to time invariance, it is well known that

$$\forall \tau' < \tau, \quad \mathcal{E}_\tau = \mathcal{E}_{\tau-\tau'} \circ \mathcal{E}_{\tau'}$$

- For instance, let $0 = T_0 < T_1 < ... < T_n < ... < T_N = T$ be special times at which we are interested to consider snapshots of the solution $y(T_n)$, then we have

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

Basics on the algorithm.

- In most cases $\mathcal{E}$ is not achievable but only approximations based on time discretization and the use of Euler or more involved schemes.
- For instance we can introduce a fine and precise approximated propagator $\mathcal{F}$ defined through the resolution of

$$\frac{y^{m+1} - y^m}{\delta t} + \mathcal{A} y^{m(+1)} = 0 \qquad \text{for an explicit implicit scheme}$$

for any time $T = M\delta t$ the approximated propagator $\mathcal{F}_T$ involves the iterative resolution of M problems as above.

Basics on the algorithm.

- In most cases $\mathcal{E}$ is not achievable but only approximations based on time discretization and the use of Euler or more involved schemes.
- For instance we can introduce a fine and precise approximated propagator $\mathcal{F}$ defined through the resolution of

$$\frac{y^{m+1} - y^m}{\delta t} + \mathcal{A} y^{m(+1)} = 0$$ for an explicit implicit scheme

for any time $T = M\delta t$ the approximated propagator $\mathcal{F}_T$ involves the iterative resolution of M problems as above.

Basics on the algorithm.

- In most cases $\mathcal{E}$ is not achievable but only approximations based on time discretization and the use of Euler or more involved schemes.
- For instance we can introduce a fine and precise approximated propagator $\mathcal{F}$ defined through the resolution of

$$\frac{y^{m+1}-y^{m}}{\delta t}+\mathcal{A}y^{m(+1)}=0 \qquad \text{for an explicit implicit scheme}$$

for any time $T = M\delta t$ the approximated propagator $\mathcal{F}_T$ involves the iterative resolution of M problems as above.

Basics on the algorithm.

- Similarly as for the continuous solution,

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

- we have the approximations Y_n of $y(T_n)$ given by

$$Y_{n+1} = \mathcal{F}_{T_{n+1}}(y^0) = \mathcal{F}_{T_{n+1}-T_n}(Y_n)$$

- Assuming, for the sake of simplicity that $T_{n+1} - T_n$ is constant $(= \Delta T >> \delta t)$, then this reads

$$Y_{n+1} = \mathcal{F}_{\Delta T}(Y_n)$$

  where it appears that the approximated solution process is sequential, which, a priori, prevents from a parallelization.

- In what follows we propose an algorithm $Y_n^k \longrightarrow Y_n$ as k goes to infinity.

Basics on the algorithm.

- Similarly as for the continuous solution,

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

- we have the approximations Y_n of $y(T_n)$ given by

$$Y_{n+1} = \mathcal{F}_{T_{n+1}}(y^0) = \mathcal{F}_{T_{n+1}-T_n}(Y_n)$$

- Assuming, for the sake of simplicity that $T_{n+1} - T_n$ is constant $(= \Delta T >> \delta t)$, then this reads

$$Y_{n+1} = \mathcal{F}_{\Delta T}(Y_n)$$

where it appears that the approximated solution process is sequential, which, a priori, prevents from a parallelization.

- In what follows we propose an algorithm $Y_n^k \longrightarrow Y_n$ as k goes to infinity.

Basics on the algorithm.

- Similarly as for the continuous solution,

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

- we have the approximations Y_n of $y(T_n)$ given by

$$Y_{n+1} = \mathcal{F}_{T_{n+1}}(y^0) = \mathcal{F}_{T_{n+1}-T_n}(Y_n)$$

- Assuming, for the sake of simplicity that $T_{n+1} - T_n$ is constant $(= \Delta T >> \delta t)$, then this reads

$$Y_{n+1} = \mathcal{F}_{\Delta T}(Y_n)$$

where it appears that the approximated solution process is sequential, which, a priori, prevents from a parallelization.

- In what follows we propose an algorithm $Y_n^k \longrightarrow Y_n$ as k goes to infinity.

Basics on the algorithm.

- Similarly as for the continuous solution,

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

- we have the approximations Y_n of $y(T_n)$ given by

$$Y_{n+1} = \mathcal{F}_{T_{n+1}}(y^0) = \mathcal{F}_{T_{n+1}-T_n}(Y_n)$$

- Assuming, for the sake of simplicity that $T_{n+1} - T_n$ is constant ($= \Delta T >> \delta t$), then this reads

$$Y_{n+1} = \mathcal{F}_{\Delta T}(Y_n)$$

  where it appears that the approximated solution process is sequential, which, a priori, prevents from a parallelization.

- In what follows we propose an algorithm $Y_n^k \longrightarrow Y_n$ as k goes to infinity.

Basics on the algorithm.

- Similarly as for the continuous solution,

$$y(T_{n+1}) = \mathcal{E}_{T_{n+1}}(y^0) = \mathcal{E}_{T_{n+1}-T_n}(y(T_n))$$

- we have the approximations Y_n of $y(T_n)$ given by

$$Y_{n+1} = \mathcal{F}_{T_{n+1}}(y^0) = \mathcal{F}_{T_{n+1}-T_n}(Y_n)$$

- Assuming, for the sake of simplicity that $T_{n+1} - T_n$ is constant $(= \Delta T >> \delta t)$, then this reads

$$Y_{n+1} = \mathcal{F}_{\Delta T}(Y_n)$$

  where it appears that the approximated solution process is sequential, which, a priori, prevents from a parallelization.
- In what follows we propose an algorithm $Y_n^k \longrightarrow Y_n$ as k goes to infinity.

Basics on the algorithm.

- For this we assume that another propagator is achievable. It is denoted as $\mathcal{G}$ and is assumed to be cheap but inaccurate.
- One can think about $\mathcal{F}$ based on an Euler scheme with a very small time step δt and $\mathcal{G}$ based on an Euler scheme with the larger time step ΔT.
- But other possibility are offered as e.g. $\mathcal{F}$ carries all the physics of the phenomenon but $\mathcal{G}$ is based on a simplified physics (see latter).
- The iterative process is

$$Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)$$

Basics on the algorithm.

- For this we assume that another propagator is achievable. It is denoted as $\mathcal{G}$ and is assumed to be cheap but inaccurate.
- One can think about $\mathcal{F}$ based on an Euler scheme with a very small time step δt and $\mathcal{G}$ based on an Euler scheme with the larger time step ΔT.
- But other possibility are offered as e.g. $\mathcal{F}$ carries all the physics of the phenomenon but $\mathcal{G}$ is based on a simplified physics (see latter).
- The iterative process is

$$Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)$$

Basics on the algorithm.

- For this we assume that another propagator is achievable. It is denoted as $\mathcal{G}$ and is assumed to be cheap but inaccurate.
- One can think about $\mathcal{F}$ based on an Euler scheme with a very small time step δt and $\mathcal{G}$ based on an Euler scheme with the larger time step ΔT.
- But other possibility are offered as e.g. $\mathcal{F}$ carries all the physics of the phenomenon but $\mathcal{G}$ is based on a simplified physics (see latter).
- The iterative process is

$$Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)$$

Basics on the algorithm.

- For this we assume that another propagator is achievable. It is denoted as $\mathcal{G}$ and is assumed to be cheap but inaccurate.
- One can think about $\mathcal{F}$ based on an Euler scheme with a very small time step δt and $\mathcal{G}$ based on an Euler scheme with the larger time step ΔT.
- But other possibility are offered as e.g. $\mathcal{F}$ carries all the physics of the phenomenon but $\mathcal{G}$ is based on a simplified physics (see latter).
- The iterative process is

$$Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)$$

Basics on the algorithm.

- For this we assume that another propagator is achievable. It is denoted as $\mathcal{G}$ and is assumed to be cheap but inaccurate.
- One can think about $\mathcal{F}$ based on an Euler scheme with a very small time step δt and $\mathcal{G}$ based on an Euler scheme with the larger time step ΔT.
- But other possibility are offered as e.g. $\mathcal{F}$ carries all the physics of the phenomenon but $\mathcal{G}$ is based on a simplified physics (see latter).
- The iterative process is

$$Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)$$

Basics on the algorithm.

- The iterative process

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

provides a converging sequence, in the sense that

if $|\mathcal{E}_T - \mathcal{F}_T| \simeq \delta t$ and if $|\mathcal{G}_{\Delta T} - \mathcal{F}_{\Delta T}| \simeq \varepsilon \Delta T$

after k iterations the error between Y_n^k and $y(T_n)$ is $\simeq \varepsilon^k + \delta t$

- Allows to envision real time simulations

Basics on the algorithm.

- The iterative process

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

provides a converging sequence, in the sense that

if $\quad |\mathcal{E}_T - \mathcal{F}_T| \simeq \delta t$ and if $\quad |\mathcal{G}_{\Delta T} - \mathcal{F}_{\Delta T}| \simeq \varepsilon \Delta T$

after k iterations the error between Y_n^k and $y(T_n)$ is $\simeq \varepsilon^k + \delta t$

- Allows to envision real time simulations

Basics on the algorithm.

- The iterative process

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

provides a converging sequence, in the sense that

if $\quad |\mathcal{E}_T - \mathcal{F}_T| \simeq \delta t$ and if $\quad |\mathcal{G}_{\Delta T} - \mathcal{F}_{\Delta T}| \simeq \varepsilon \Delta T$

after k iterations the error between Y_n^k and $y(T_n)$ is $\simeq \varepsilon^k + \delta t$

- Allows to envision real time simulations

Basics on the algorithm.

- The iterative process

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

provides a <u>converging sequence</u>, in the sense that

$$\text{if} \qquad |\mathcal{E}_T - \mathcal{F}_T| \simeq \delta t \text{ and if} \qquad |\mathcal{G}_{\Delta T} - \mathcal{F}_{\Delta T}| \simeq \varepsilon \Delta T$$

after k iterations the error between Y_n^k and $y(T_n)$ is $\simeq \varepsilon^k + \delta t$

- Allows to envision real time simulations

An example

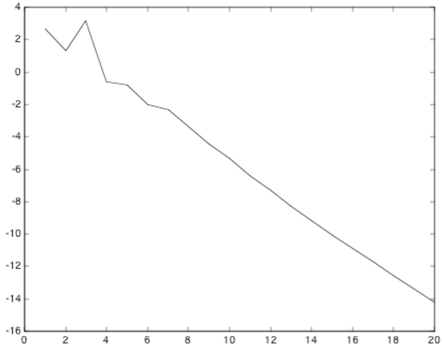


Figure: log of the error on $\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} - \Delta u + 5u^3 = 5\sin(2t)$

$\Delta T = 0.1$ and $\delta t = 810^{-4}$, speed-up factor of 14

An example

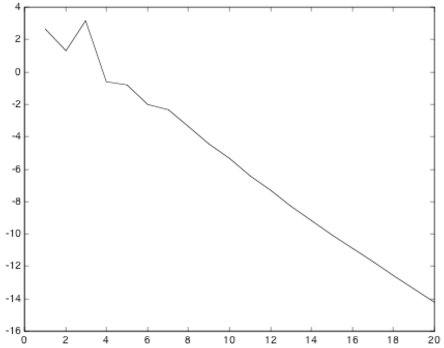


Figure: log of the error on $\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} - \Delta u + 5u^3 = 5\sin(2t)$

$\Delta T = 0.1$ and $\delta t = 8 10^{-4}$, speed-up factor of 14

Another example... chaotic behavior

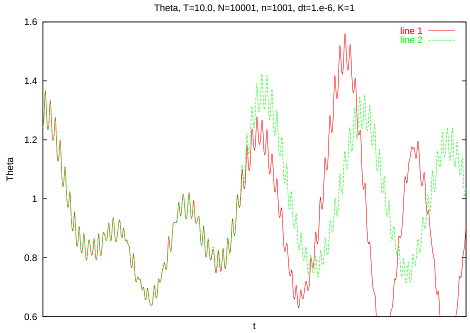


Figure: variation of the angle of a molecule A-A-B, after iteration 1, with Leonado Baffico and Gilles Zerah

Another example... chaotic behavior

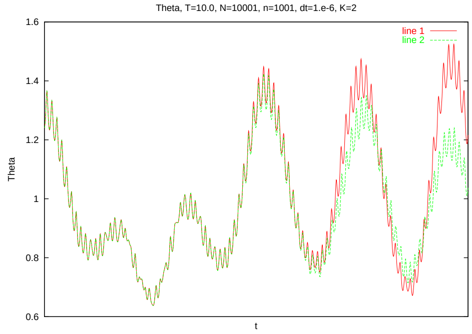


Figure: variation of the angle of a molecule A-A-B, after iteration 2, with Leonado Baffico and Gilles Zerah

Another example... chaotic behavior

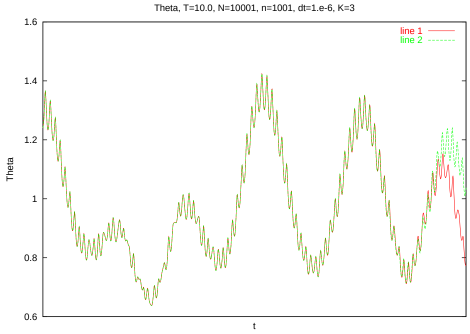


Figure: variation of the angle of a molecule A-A-B, after iteration 3, with Leonado Baffico and Gilles Zerah

Another example... chaotic behavior

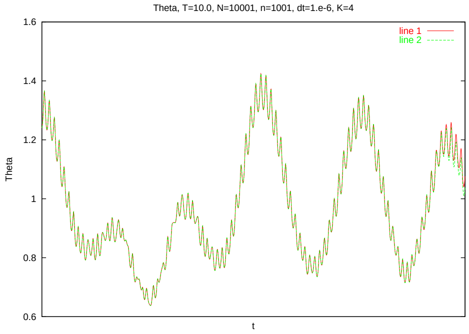


Figure: variation of the angle of a molecule A-A-B, after iteration 4, with Leonado Baffico and Gilles Zerah

Another example... chaotic behavior

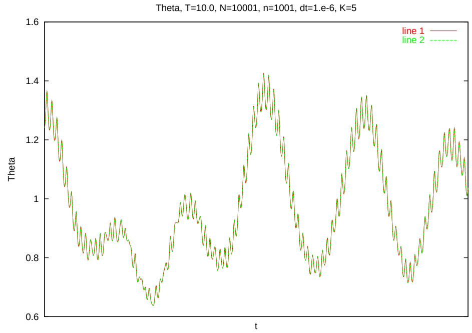


Figure: variation of the angle of a molecule A-A-B, after iteration 5, with Leonado Baffico and Gilles Zerah

Another example... molecular dynamics

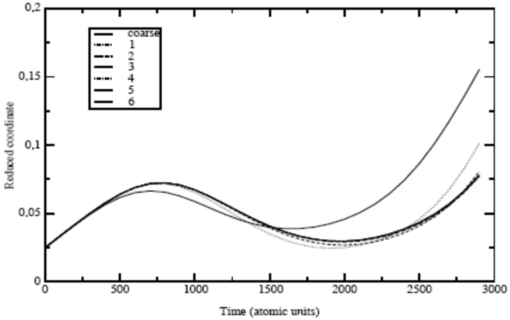


Figure: 4 aluminium atoms in a liquid state, periodic BC, convergence after 4 iterations, with Leonado Baffico and Gilles Zerah

Another example... molecular dynamics

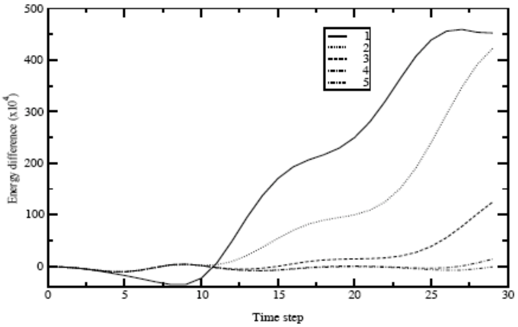


Figure: 4 aluminium atoms in a liquid state, periodic BC, convergence after .... 5 iterations, with Leonado Baffico and Gilles Zerah

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Summary

$$\boxed{Y_{n+1}^{k+1} = \mathcal{G}_{\Delta T}(Y_n^{k+1}) + \mathcal{F}_{\Delta T}(Y_n^k) - \mathcal{G}_{\Delta T}(Y_n^k)}$$

- Initialization : $Y_{n+1}^1 = \mathcal{G}_{\Delta T}(Y_n^1)$ (sequential)
  Assume every $(Y_n^k)_n$ is known at step k
- Resolution over each $]T_n, T_{n+1}[$: $\mathcal{F}_{\Delta T}(Y_n^k)$ (parallel)
- Resolution of the coarse predictor : $\mathcal{G}_{\Delta T}(Y_n^{k+1})$ (sequential)
- Correction at each initial time T_n (sequential)
- until convergence

Different concepts of coarse propagators

$\mathcal{G}$ can also differ in other ways ....

- First, we can choose a coarser mesh for the spacial discretization of the PDE.
  Actually, this is quite consistant with the use of a coarser time step for stability conditions.
- As a first example, the viscous Burgers equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

$$u(t,-1) = u(t,1), u_x(t,-1) = u_x(t,1)$$

$$u(0,x) = u_0(x) = -sin\pi x$$

with $\nu = 0.02$, $\delta t = 2.510^{-3}$, $\Delta T = 510^{-2}$ Runge Kutta Scheme
In space : Legendre spectral based on polynomial order 55. The coarse propagation is based on polynomial orders 20 or 30.

Different concepts of coarse propagators

$\mathcal{G}$ can also differ in other ways ....

- First, we can choose a coarser mesh for the spacial discretization of the PDE.
  Actually, this is quite consistant with the use of a coarser time step for stability conditions.
- As a first example, the viscous Burgers equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}$$

$$u(t,-1) = u(t,1), u_x(t,-1) = u_x(t,1)$$

$$u(0,x) = u_0(x) = -sin\pi x$$

with $\nu = 0.02$, $\delta t = 2.510^{-3}$, $\Delta T = 510^{-2}$ Runge Kutta Scheme
In space : Legendre spectral based on polynomial order 55. The coarse propagation is based on polynomial orders 20 or 30.

Different concepts of coarse propagators

$\mathcal{G}$ can also differ in other ways ....

- First, we can choose a coarser mesh for the spacial discretization of the PDE.
  Actually, this is quite consistant with the use of a coarser time step for stability conditions.
- As a first example, the viscous Burgers equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}$$

$$u(t,-1) = u(t,1), u_x(t,-1) = u_x(t,1)$$

$$u(0,x) = u_0(x) = -sin\pi x$$

with $\nu = 0.02$, $\delta t = 2.510^{-3}$, $\Delta T = 510^{-2}$ Runge Kutta Scheme
In space : Legendre spectral based on polynomial order 55. The coarse propagation is based on polynomial orders 20 or 30.

Different concepts of coarse propagators

$\mathcal{G}$ can also differ in other ways ....

- First, we can choose a coarser mesh for the spacial discretization of the PDE.
  Actually, this is quite consistant with the use of a coarser time step for stability conditions.
- As a first example, the viscous Burgers equation

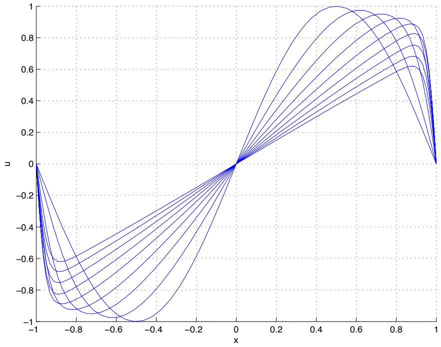
$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}$$

$$u(t,-1) = u(t,1), u_x(t,-1) = u_x(t,1)$$

$$u(0,x) = u_0(x) = -sin \pi x$$

with $\nu = 0.02$, $\delta t = 2.510^{-3}$, $\Delta T = 510^{-2}$ Runge Kutta Scheme
In space : Legendre spectral based on polynomial order 55. The coarse propagation is based on polynomial orders 20 or 30.

Different concepts of coarse propagators



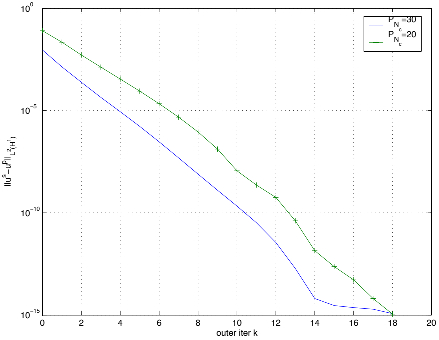


Figure: The left plot shows the numerical solution of the viscous Burger's equation at different times. The right plot shows the convergence of the parareal-in-time algorithm using a coarse propagator with a time step $\Delta T = 510^{-2}$ and a lower order polynomial space (N=20, or 30).

Different concepts of coarse propagators

With Fischer and Hecht, in 2004, we have also implemented this method for the simulations of the Navier Stokes equation

$$\mathbf{U}_{n+1}^{k+1} = \Pi_H^h \mathcal{G}_{\Delta T}(\Pi_h^H \mathbf{U}_n^{k+1}) + \mathcal{F}_{\Delta T}(\mathbf{U}_n^k) - \Pi_H^h \mathcal{G}_{\Delta T}(\Pi_h^H \mathbf{U}_n^k),$$

e.g. spectral approximation, for the simulation of the growth of a small-amplitude (10^{-5}) Tollmien-Schlichting wave, superimposed on plane Poiseuille chanel flow at $Re = 7500$, compared with linear theory.

Different concepts of coarse propagators

With Fischer and Hecht, in 2004, we have also implemented this method for the simulations of the Navier Stokes equation

$$\mathbf{U}_{n+1}^{k+1} = \Pi_H^h \mathcal{G}_{\Delta T}(\Pi_h^H \mathbf{U}_n^{k+1}) + \mathcal{F}_{\Delta T}(\mathbf{U}_n^k) - \Pi_H^h \mathcal{G}_{\Delta T}(\Pi_h^H \mathbf{U}_n^k),$$

e.g. spectral approximation, for the simulation of the growth of a small-amplitude (10^{-5}) Tollmien-Schlichting wave, superimposed on plane Poiseuille chanel flow at $Re = 7500$, compared with linear theory.

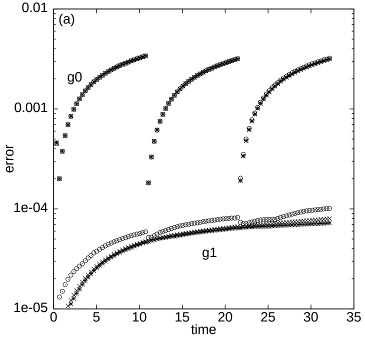
Different concepts of coarse propagators

With Fischer and Hecht, in 2004, we have also implemented this method for the simulations of the Navier Stokes equation

$$\mathbf{U}_{n+1}^{k+1} = \Pi_H^h \mathcal{G}_{\Delta T}(\Pi_h^H \mathbf{U}_n^{k+1}) + \mathcal{F}_{\Delta T}(\mathbf{U}_n^k) - \Pi_H^h \mathcal{G}_{\Delta T}(\Pi_h^H \mathbf{U}_n^k),$$

e.g. spectral approximation, for the simulation of the growth of a small-amplitude (10^{-5}) Tollmien-Schlichting wave, superimposed on plane Poiseuille chanel flow at $Re = 7500$, compared with linear theory.

Different concepts of coarse propagators



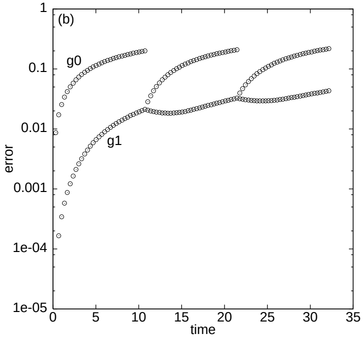


Figure: Error histories in the y-component of velocity versus time : 15 spectral elements with degree =15, $\delta t = .005$ and $\Delta t = .333$ using varying fine/coarse approximation orders (M,$\tilde{M}$): $\circ$ = (13,13), $\times$ = (15,13), $*$ = (15,15) with 3 restarts..... .

Different concepts of coarse propagators

$\mathcal{G}$ can also differ in other ways ....

Another choice is the replacement of the model, by a coarser one based on simpler physics.

A first example is given by the reduction procedure to help in the solution procedure of stiff molecular kinetic reactions.
The problem of interest is written in a matricial system

$$\frac{d\mathbf{y}}{dt} = J\mathbf{y}$$

where J is a kinetic matrix, and the system is stiff due to different speeds in the reactions

Different concepts of coarse propagators

$\mathcal{G}$ can also differ in other ways ....

Another choice is the replacement of the model, by a coarser one based on simpler physics.

A first example is given by the reduction procedure to help in the solution procedure of stiff molecular kinetic reactions.
The problem of interest is written in a matricial system

$$\frac{d\mathbf{y}}{dt} = J\mathbf{y}$$

where J is a kinetic matrix, and the system is stiff due to different speeds in the reactions

Different concepts of coarse propagators

$\mathcal{G}$ can also differ in other ways ....

Another choice is the replacement of the model, by a coarser one based on simpler physics.

A first example is given by the reduction procedure to help in the solution procedure of stiff molecular kinetic reactions.
The problem of interest is written in a matricial system

$$\frac{d\mathbf{y}}{dt} = J\mathbf{y}$$

where J is a kinetic matrix, and the system is stiff due to different speeds in the reactions

Different concepts of coarse propagators

The reduction process

- Rank the eigenvalues in increasing order of real part

$$Re(\lambda_1) \leq Re(\lambda_2) \leq \cdots \leq Re(\lambda_p) = Re(\lambda_{p+1}) = Re(\lambda_N) = 0$$

- Get rid one after the other of the largest eigenvalues (in absolute value) and corresponding eigenfunctions
- Solve $\frac{dy^R}{dt} = J^R y^R$
- Reconstruct recursively the lacking eigenfunctions
- If the propagated reduced species are such that $y_j^R(t_0) - y_j(t_0) \leq \varepsilon$, then, the non propagated species satisfy $y_j^R(t_0) - y_j(t_0) \leq \varepsilon + [\mathcal{R}e(\lambda_j)]^{-1}$ if $t_0 \simeq \mathcal{O}(\mathcal{R}e(\lambda_j)^{-1} \ln[\mathcal{R}e(\lambda_j])$

The idea is then to use the reduced model as a propagator for the coarse system and solve the exact problem on short time intervals in parallel

Different concepts of coarse propagators
The reduction process

- Rank the eigenvalues in increasing order of real part

$$Re(\lambda_1) \leq Re(\lambda_2) \leq \cdots \leq Re(\lambda_p) = Re(\lambda_{p+1}) = Re(\lambda_N) = 0$$

- Get rid one after the other of the largest eigenvalues (in absolute value) and corresponding eigenfunctions
- Solve $\frac{dy^R}{dt} = J^R y^R$
- Reconstruct recursively the lacking eigenfunctions
- If the propagated reduced species are such that $y_j^R(t_0) - y_j(t_0) \leq \varepsilon$, then, the non propagated species satisfy $y_j^R(t_0) - y_j(t_0) \leq \varepsilon + [\mathcal{R}e(\lambda_j)]^{-1}$ if $t_0 \simeq \mathcal{O}(\mathcal{R}e(\lambda_j)^{-1} \ln[\mathcal{R}e(\lambda_j]))$

The idea is then to use the reduced model as a propagator for the coarse system and solve the exact problem on short time intervals in parallel

Different concepts of coarse propagators
The reduction process

- Rank the eigenvalues in increasing order of real part

$$Re(\lambda_1) \le Re(\lambda_2) \le \cdots \le Re(\lambda_p) = Re(\lambda_{p+1}) = Re(\lambda_N) = 0$$

- Get rid one after the other of the largest eigenvalues (in absolute value) and corresponding eigenfunctions
- Solve $\frac{dy^R}{dt} = J^R y^R$
- Reconstruct recursively the lacking eigenfunctions
- If the propagated reduced species are such that $y_j^R(t_0) - y_j(t_0) \le \varepsilon$, then, the non propagated species satisfy $y_j^R(t_0) - y_j(t_0) \le \varepsilon + [\mathcal{R}e(\lambda_j)]^{-1}$ if $t_0 \simeq \mathcal{O}(\mathcal{R}e(\lambda_j)^{-1} \ln[\mathcal{R}e(\lambda_j]))$

The idea is then to use the reduced model as a propagator for the coarse system and solve the exact problem on short time intervals in parallel

Different concepts of coarse propagators

The reduction process

- Rank the eigenvalues in increasing order of real part

$$Re(\lambda_1) \leq Re(\lambda_2) \leq \cdots \leq Re(\lambda_p) = Re(\lambda_{p+1}) = Re(\lambda_N) = 0$$

- Get rid one after the other of the largest eigenvalues (in absolute value) and corresponding eigenfunctions
- Solve $\frac{dy^R}{dt} = J^R y^R$
- Reconstruct recursively the lacking eigenfunctions
- If the propagated reduced species are such that $y_j^R(t_0) - y_j(t_0) \leq \varepsilon$, then, the non propagated species satisfy $y_j^R(t_0) - y_j(t_0) \leq \varepsilon + [\mathcal{R}e(\lambda_j)]^{-1}$ if $t_0 \simeq \mathcal{O}(\mathcal{R}e(\lambda_j)^{-1} \ln[\mathcal{R}e(\lambda_j])$

The idea is then to use the reduced model as a propagator for the coarse system and solve the exact problem on short time intervals in parallel

Different concepts of coarse propagators
The reduction process

- Rank the eigenvalues in increasing order of real part

$$Re(\lambda_1) \leq Re(\lambda_2) \leq \cdots \leq Re(\lambda_p) = Re(\lambda_{p+1}) = Re(\lambda_N) = 0$$

- Get rid one after the other of the largest eigenvalues (in absolute value) and corresponding eigenfunctions
- Solve $\frac{dy^R}{dt} = J^R y^R$
- Reconstruct recursively the lacking eigenfunctions
- If the propagated reduced species are such that $y_j^R(t_0) - y_j(t_0) \leq \varepsilon$, then, the non propagated species satisfy $y_j^R(t_0) - y_j(t_0) \leq \varepsilon + [\mathcal{R}e(\lambda_j)]^{-1}$ if $t_0 \simeq \mathcal{O}(\mathcal{R}e(\lambda_j)^{-1} \ln[\mathcal{R}e(\lambda_j)])$

The idea is then to use the reduced model as a propagator for the coarse system and solve the exact problem on short time intervals in parallel

Different concepts of coarse propagators

The reduction process

We test this on the evolution in case of source terms
Done on the simple case

$$J = \begin{pmatrix} -310 & 100 & 0 \\ 300 & -100 & 0 \\ 10 & 0 & 0 \end{pmatrix}$$

in what follows, we have incorporated sources terms at times : 0.4, 0.8, 1.2, ...

Different concepts of coarse propagators

More interesting.... exact solution

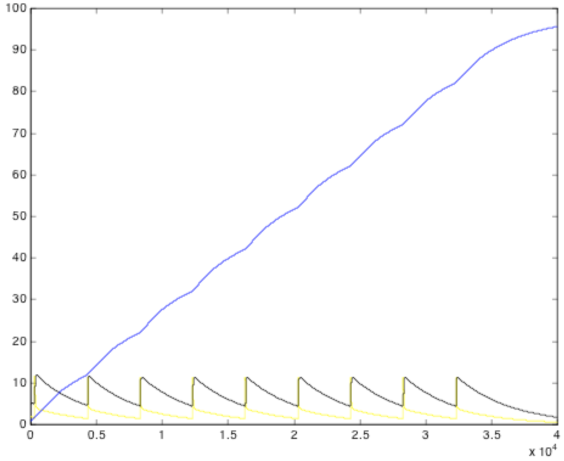


Figure: exact solution .

Different concepts of coarse propagators

More interesting

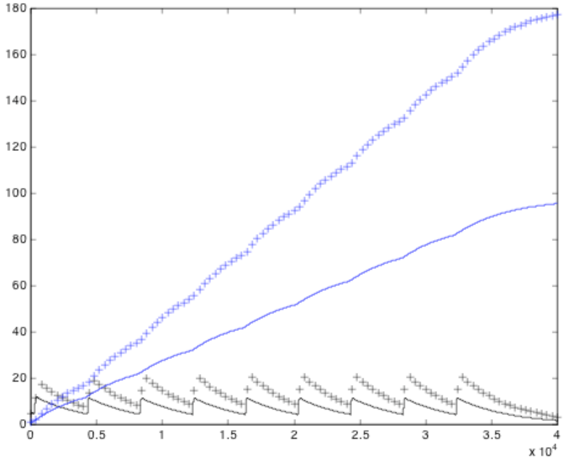


Figure: effect of the reduced model, iteration 1 .

Different concepts of coarse propagators

More interesting

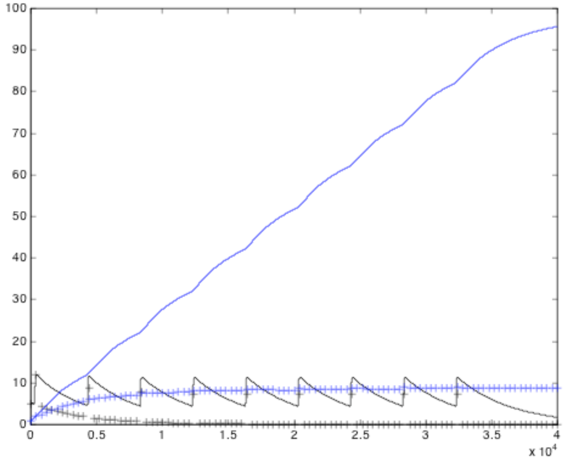


Figure: effect of the reduced model, iteration 2 .

Different concepts of coarse propagators

More interesting

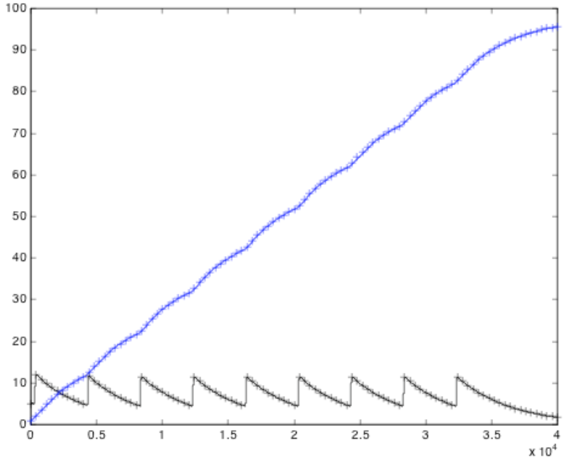


Figure: effect of the reduced model, iteration 3... Bingo!!

Different concepts of coarse propagators

Even more interesting

Now on a more complex system of Thyroid metabolism

$$J = \begin{pmatrix} -5.1 & .01 & 0 & 0 & .06 & 0 \\ 0 & -2.516 & 0 & 0 & 0 & .0008 \\ 0 & 0 & -1.3 & .001 & .0003 & 0 \\ 0 & 0 & 0 & -1.091 & 0 & .00008 \\ 5 & 0 & 1 & 0 & -.0603 & 0 \\ 0 & 2.5 & 0 & 1 & 0 & -.00088 \end{pmatrix}$$

where 3 levels of reduction are required and the evolution is done for only two species

Different concepts of coarse propagators

Even more interesting

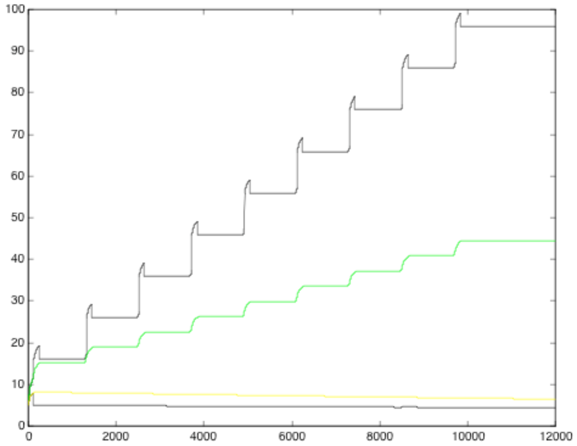


Figure: effect of the reduced model, iteration 1... not too bad .

Different concepts of coarse propagators

Even more interesting

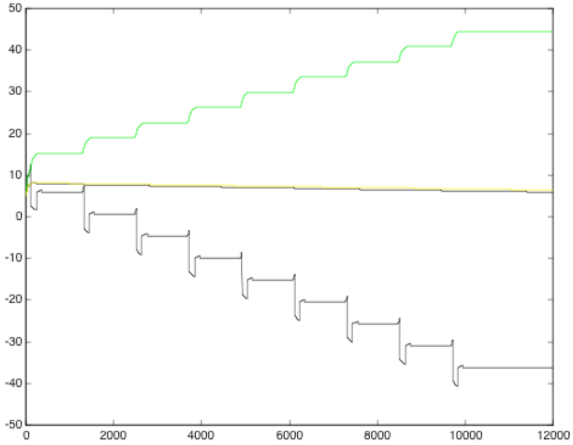


Figure: effect of the reduced model, iteration 2... oups

Different concepts of coarse propagators
Even more interesting

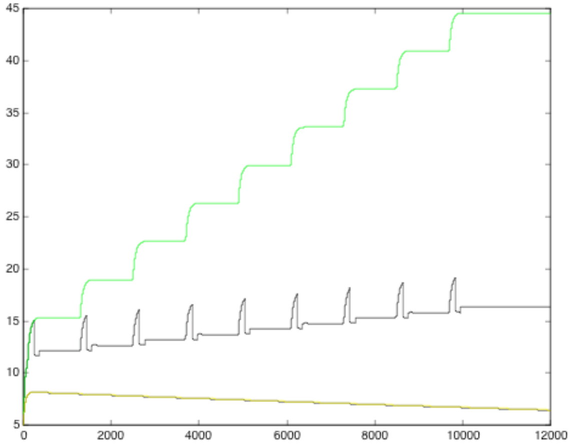


Figure: effect of the reduced model, iteration 3.... still far away???

Different concepts of coarse propagators

Even more interesting

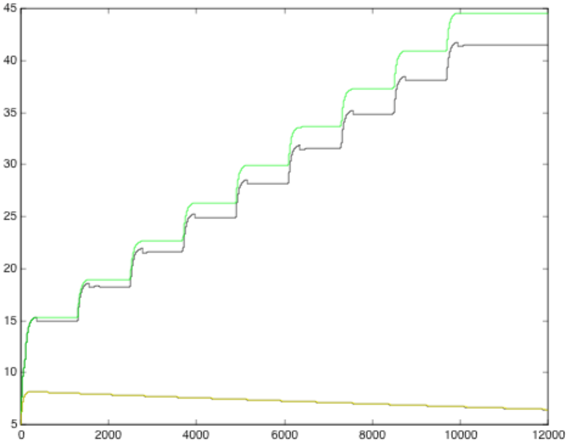


Figure: effect of the reduced model, iteration 4... wait .

Different concepts of coarse propagators

Even more interesting

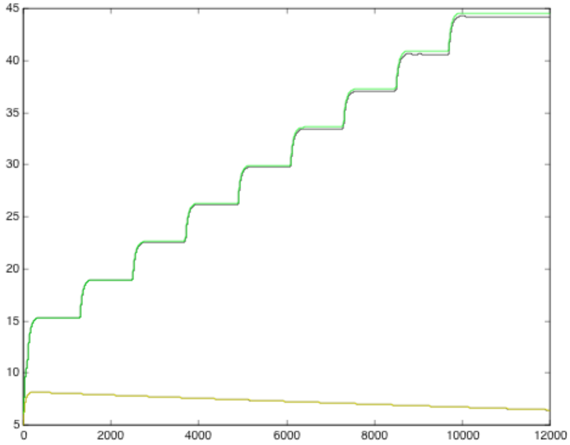


Figure: effect of the reduced model, iteration 5... it took more but Bingo again

Another example... the KdVB equation

Let us consider the following problem : Find $u(x, t)$, $x \in \mathbb{R}, t > 0$ such that

$$\partial_t u + u\partial_x u = \partial_x (\nu \partial_x u) + M\partial^3_{xxx} u \tag{1}$$

with boundary conditions

$$u(-\infty) = U, \quad u(+\infty) = 0, \quad \partial_x u(-\infty) = 0 \tag{2}$$

and initial data

$$u(x, 0) = u_0(x). \tag{3}$$

where the viscosity parameter ν is very small.

Another example

It is known that, if $\nu < \nu_0$ the solution evolves to a very oscillatory profile

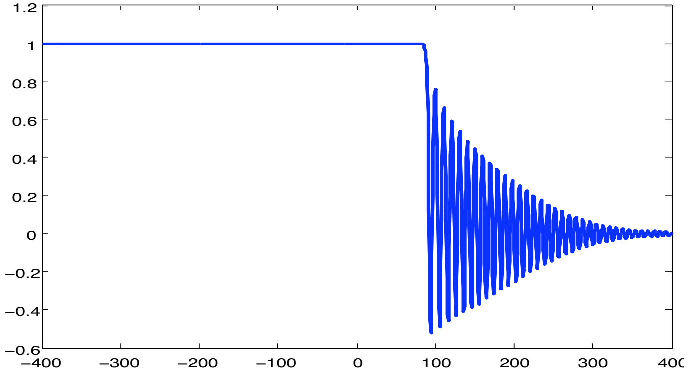


Figure: evolution for $\nu < \nu_0$.

Another example

The problem is numerically difficult to solve and following the work of Chorin and Barenblatt, the idea is to define the spacial averaging operator

$$\mathcal{M}_\lambda(\varphi)(x) = \frac{1}{2\lambda}\int_{x-\lambda}^{x+\lambda} \varphi(z)dz$$

and set

$$\bar{u}_\lambda(.,t) = \mathcal{M}_\lambda(u(.,t))$$

Averaging equation yields to an effective equation

$$\partial_t \bar{u}_\lambda + \bar{u}_\lambda \partial_x \bar{u}_\lambda = \nu_{eff} \partial^2_{xx} \bar{u}_\lambda \tag{4}$$

with

$$\nu_{eff} = R^{3/4}\psi(\ell)$$

and $R == \frac{\sqrt{M\,U}}{\nu}$, resulting in $\nu_{eff} >> \nu$ so that the effective equation is more simple to be solved

Another example

We can overlap the two solutions

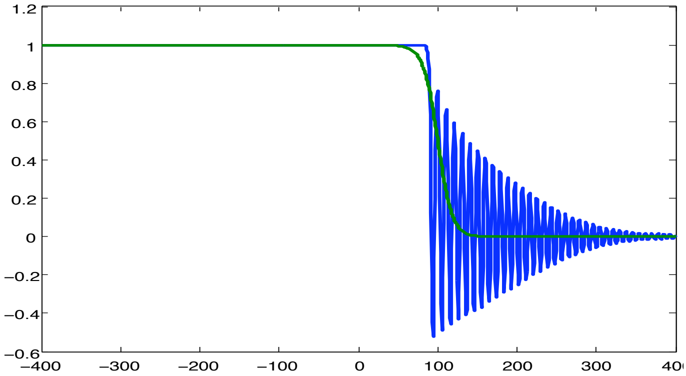


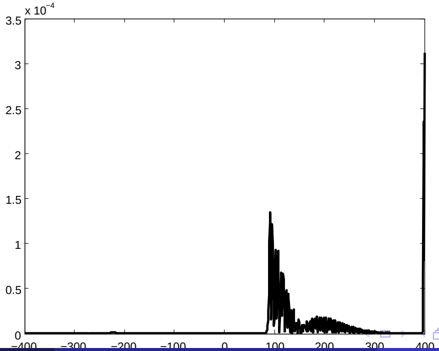
Figure: evolution for $\nu < \nu_0$ and the effective equation.

Another example

- Our idea in this context is to use the effective equation for the coarse solver and the complete model for the fine solver... the non linearity does the remaining part
- The algorithm (preliminary computations with only one grid and different time steps) gives a convergence after only 3 iterations of the parareal algorithm for 50 parareal intervals.

Another example

This plot represents the error (absolute value) beetwen the parareal solution and the direct solution after 3 iterations with 50 time subintervals.



A control problem

Let A be a linear (or nonlinear) operator and let us consider the following state equation:

$$\frac{\partial y}{\partial t} + Ay = Bv$$

with initial condition : $y(0) = y^0$ and where the control v (boundary or distributed) belongs to some space $\mathcal{U}$ and B is some appropriate operator.
We assume that for any given v, this problem is well posed.
We complement this problem with a cost functional to be minimized

$$\mathcal{J}(v) = \frac{1}{2}\int_0^T \|v\|_{\mathcal{U}}^2 + \frac{\alpha}{2}\|y(T) - y^T\|^2, \qquad (2)$$

where $\alpha > 0$, y^T is a target and the norm is the H norm if, for instance $V \subset H \subset V'$.

A control problem

Let A be a linear (or nonlinear) operator and let us consider the following state equation:

$$\frac{\partial y}{\partial t} + Ay = Bv$$

with initial condition : $y(0) = y^0$ and where the control v (boundary or distributed) belongs to some space $\mathcal{U}$ and B is some appropriate operator.
We assume that for any given v, this problem is well posed.
We complement this problem with a cost functional to be minimized

$$\mathcal{J}(v) = \frac{1}{2}\int_0^T \|v\|^2_{\mathcal{U}} + \frac{\alpha}{2}\|y(T) - y^T\|^2, \tag{2}$$

where $\alpha > 0$, y^T is a target and the norm is the H norm if, for instance $V \subset H \subset V'$.

A control problem

Let A be a linear (or nonlinear) operator and let us consider the following state equation:

$$\frac{\partial y}{\partial t} + Ay = Bv$$

with initial condition : $y(0) = y^0$ and where the control v (boundary or distributed) belongs to some space $\mathcal{U}$ and B is some appropriate operator.
We assume that for any given v, this problem is well posed.
We complement this problem with a cost functional to be minimized

$$\mathcal{J}(v) = \frac{1}{2}\int_0^T \|v\|^2_{\mathcal{U}} + \frac{\alpha}{2}\|y(T) - y^T\|^2, \qquad (2)$$

where $\alpha > 0$, y^T is a target and the norm is the H norm if, for instance $V \subset H \subset V'$.

A control problem decomposed over sub-intervals

Let us decompose the time interval again

$$0 = T_0 < T_1 < ... < T_n < T_{n+1} < ... < T_N = T.$$

Then we define $\{y_0, y_1, ..., y_n, ..., y_{N-1}\}$ solutions of

$$\frac{\partial y_n}{\partial t} + Ay_n = Bv_n, \quad \text{over } (T_n, T_{n+1})$$

$$y_n(T_n^+) = \lambda_n,$$

The collection of solutions $\{y_0, y_1, ..., y_n, ..., y_{N-1}\}$ satisfies $y_n = y_{|(T_n,T_{n+1})}$) if and only if,

$$\forall n, \quad v_n = v_{|(T_n,T_{n+1})} \quad \text{and} \quad \lambda_n = y(T_n),$$

or again $\qquad \forall n, \quad v_n = v_{|(T_n,T_{n+1})} \quad \text{and} \quad \lambda_n = y_{n-1}(T_n^-).$

This way, λ_n can be interpreted as a "virtual" control (*à la Lions*) that leads to the following development.

A control problem decomposed over sub-intervals

Let us decompose the time interval again

$$0 = T_0 < T_1 < ... < T_n < T_{n+1} < ... < T_N = T.$$

Then we define $\{y_0, y_1, ..., y_n, ..., y_{N-1}\}$ solutions of

$$\frac{\partial y_n}{\partial t} + Ay_n = Bv_n, \quad \text{over } (T_n, T_{n+1})$$

$$y_n(T_n^+) = \lambda_n,$$

The collection of solutions $\{y_0, y_1, ..., y_n, ..., y_{N-1}\}$ satisfies $y_n = y_{|(T_n,T_{n+1})}$) if and only if,

$$\forall n, \quad v_n = v_{|(T_n,T_{n+1})} \quad \text{and} \quad \lambda_n = y(T_n),$$

or again $\forall n, \quad v_n = v_{|(T_n,T_{n+1})} \quad \text{and} \quad \lambda_n = y_{n-1}(T_n^-).$

This way, λ_n can be interpreted as a "virtual" control (*à la Lions*) that leads to the following development.

A control problem decomposed over sub-intervals

Let us decompose the time interval again

$$0 = T_0 < T_1 < ... < T_n < T_{n+1} < ... < T_N = T.$$

Then we define $\{y_0, y_1, ..., y_n, ..., y_{N-1}\}$ solutions of

$$\frac{\partial y_n}{\partial t} + Ay_n = Bv_n, \quad \text{over } (T_n, T_{n+1})$$

$$y_n(T_n^+) = \lambda_n,$$

The collection of solutions $\{y_0, y_1, ..., y_n, ..., y_{N-1}\}$ satisfies $y_n = y_{|(T_n,T_{n+1})})$ if and only if,

$$\forall n, \quad v_n = v_{|(T_n,T_{n+1})} \quad \text{and} \quad \lambda_n = y(T_n),$$

or again $\forall n, \quad v_n = v_{|(T_n,T_{n+1})} \quad \text{and} \quad \lambda_n = y_{n-1}(T_n^-).$
This way, λ_n can be interpreted as a "virtual" control (*à la Lions*) that leads to the following development.

A control problem decomposed over sub-intervals

Let us decompose the time interval again

$$0 = T_0 < T_1 < ... < T_n < T_{n+1} < ... < T_N = T.$$

Then we define $\{y_0, y_1, ..., y_n, ..., y_{N-1}\}$ solutions of

$$\frac{\partial y_n}{\partial t} + Ay_n = Bv_n, \quad \text{over } (T_n, T_{n+1})$$

$$y_n(T_n^+) = \lambda_n,$$

The collection of solutions $\{y_0, y_1, ..., y_n, ..., y_{N-1}\}$ satisfies $y_n = y_{|(T_n,T_{n+1})})$ if and only if,

$$\forall n, \quad v_n = v_{|(T_n,T_{n+1})} \quad \text{and} \quad \lambda_n = y(T_n),$$

or again $\forall n, \quad v_n = v_{|(T_n,T_{n+1})} \quad \text{and} \quad \lambda_n = y_{n-1}(T_n^-).$

This way, λ_n can be interpreted as a “virtual” control (*à la Lions*) that leads to the following development.

A control problem decomposed over sub-intervals

Let us decompose the time interval again

$$0 = T_0 < T_1 < ... < T_n < T_{n+1} < ... < T_N = T.$$

Then we define $\{y_0, y_1, ..., y_n, ..., y_{N-1}\}$ solutions of

$$\frac{\partial y_n}{\partial t} + Ay_n = Bv_n, \quad \text{over } (T_n, T_{n+1})$$

$$y_n(T_n^+) = \lambda_n,$$

The collection of solutions $\{y_0, y_1, ..., y_n, ..., y_{N-1}\}$ satisfies $y_n = y_{|(T_n,T_{n+1})})$ if and only if,

$$\forall n, \quad v_n = v_{|(T_n,T_{n+1})} \quad \text{and} \quad \lambda_n = y(T_n),$$

or again $\qquad \forall n, \quad v_n = v_{|(T_n,T_{n+1})} \quad$ and $\quad \lambda_n = y_{n-1}(T_n^-)$.
This way, λ_n can be interpreted as a “virtual” control (*à la Lions*) that leads to the following development.

A control problem decomposed over sub-intervals

We then propose to modify slightly the cost functional $\mathcal{J}$ as follows

$$\mathcal{J}_\varepsilon(v,\Lambda) = \frac{1}{2}\int_0^T \|v\|_{\mathcal{U}}^2 + \frac{\alpha}{2}\|y_{N-1}(T) - y^T\|^2 + \frac{1}{2\varepsilon\Delta T}\sum_{n=1}^{N-1}\|y_{n-1}(T_n^-) - \lambda_n\|^2,$$

where $\Lambda = \{\lambda_1, ..., \lambda_n, ..., \lambda_{N-1}\}$ and $\varepsilon > 0$ is small. In order to solve this minimization problem, we compute the derivative of $\mathcal{J}_\varepsilon(v,\Lambda)$

$$\delta\mathcal{J}_\varepsilon(v,\Lambda)(\delta v,\delta\Lambda) = \sum_{n=0}^{N-1}\int_{T_n}^{T_{n+1}}(v_n,\delta v_n)_{\mathcal{U}} + \alpha(y_{N-1}(T) - y^T, \delta y_{N-1}(T))$$

$$+\frac{1}{\varepsilon\Delta T}\sum_{n=1}^{N-1}(y_{n-1}(T_n^-) - \lambda_n, \delta y_{n-1}(T_n^-) - \delta\lambda_n$$

A control problem decomposed over sub-intervals

We then propose to modify slightly the cost functional $\mathcal{J}$ as follows

$$\mathcal{J}_\varepsilon(v,\Lambda) = \frac{1}{2}\int_0^T \|v\|^2_{\mathcal{U}} + \frac{\alpha}{2}\|y_{N-1}(T) - y^T\|^2 + \frac{1}{2\varepsilon\Delta T}\sum_{n=1}^{N-1}\|y_{n-1}(T_n^-) - \lambda_n\|^2,$$

where $\Lambda = \{\lambda_1, ..., \lambda_n, ..., \lambda_{N-1}\}$ and $\varepsilon > 0$ is small. In order to solve this minimization problem, we compute the derivative of $\mathcal{J}_\varepsilon(v,\Lambda)$

$$\delta\mathcal{J}_\varepsilon(v,\Lambda)(\delta v, \delta\Lambda) = \sum_{n=0}^{N-1}\int_{T_n}^{T_{n+1}}(v_n, \delta v_n)_{\mathcal{U}} + \alpha(y_{N-1}(T) - y^T, \delta y_{N-1}(T))$$

$$+\frac{1}{\varepsilon\Delta T}\sum_{n=1}^{N-1}(y_{n-1}(T_n^-) - \lambda_n, \delta y_{n-1}(T_n^-) - \delta\lambda_n$$

A control problem decomposed over sub-intervals

We then propose to modify slightly the cost functional $\mathcal{J}$ as follows

$$\mathcal{J}_\varepsilon(v,\Lambda) = \frac{1}{2}\int_0^T \|v\|_{\mathcal{U}}^2 + \frac{\alpha}{2}\|y_{N-1}(T) - y^T\|^2 + \frac{1}{2\varepsilon\Delta T}\sum_{n=1}^{N-1}\|y_{n-1}(T_n^-) - \lambda_n\|^2,$$

where $\Lambda = \{\lambda_1, ..., \lambda_n, ..., \lambda_{N-1}\}$ and $\varepsilon > 0$ is small. In order to solve this minimization problem, we compute the derivative of $\mathcal{J}_\varepsilon(v,\Lambda)$

$$\delta\mathcal{J}_\varepsilon(v,\Lambda)(\delta v, \delta\Lambda) = \sum_{n=0}^{N-1}\int_{T_n}^{T_{n+1}} (v_n, \delta v_n)_{\mathcal{U}} + \alpha(y_{N-1}(T) - y^T, \delta y_{N-1}(T))$$

$$+ \frac{1}{\varepsilon\Delta T}\sum_{n=1}^{N-1}(y_{n-1}(T_n^-) - \lambda_n, \delta y_{n-1}(T_n^-) - \delta\lambda_n$$

Minimization of the control problem

In order to get an expression of this derivative we introduce the adjoint states

First p_{N-1} be the solution over (T_{N-1}, T_N) of

$$\frac{\partial p_{N-1}}{\partial t} + A^* p_{N-1} = 0 \quad \text{over } (T_{N-1}, T_N)$$
$$p_{N-1}(T) = \alpha(y_{N-1}(T) - y^T)$$

and the collection p_n, $n = N-2, N-1, ..., 0$ of solutions of

$$\frac{\partial p_n}{\partial t} + A^* p_n = 0 \quad \text{over } (T_n, T_{n+1}$$
$$p_n(T_{n+1}^-) = \frac{1}{\varepsilon \Delta T}(y_n(T_{n+1}^-) - \lambda_{n+1}).$$

Minimization of the control problem

In order to get an expression of this derivative we introduce the adjoint states

First p_{N-1} be the solution over (T_{N-1}, T_N) of

$$\frac{\partial p_{N-1}}{\partial t} + A^* p_{N-1} = 0 \quad \text{over } (T_{N-1}, T_N)$$
$$p_{N-1}(T) = \alpha(y_{N-1}(T) - y^T)$$

and the collection p_n, $n = N-2, N-1, ..., 0$ of solutions of

$$\frac{\partial p_n}{\partial t} + A^* p_n = 0 \quad \text{over } (T_n, T_{n+1}$$
$$p_n(T_{n+1}^-) = \frac{1}{\varepsilon \Delta T}(y_n(T_{n+1}^-) - \lambda_{n+1}).$$

Minimization of the control problem

In order to get an expression of this derivative we introduce the adjoint states
First p_{N-1} be the solution over (T_{N-1}, T_N) of

$$\frac{\partial p_{N-1}}{\partial t} + A^* p_{N-1} = 0 \quad \text{over } (T_{N-1}, T_N)$$
$$p_{N-1}(T) = \alpha(y_{N-1}(T) - y^T)$$

and the collection p_n, $n = N-2, N-1, ..., 0$ of solutions of

$$\frac{\partial p_n}{\partial t} + A^* p_n = 0 \quad \text{over } (T_n, T_{n+1}$$
$$p_n(T_{n+1}^-) = \frac{1}{\varepsilon \Delta T}(y_n(T_{n+1}^-) - \lambda_{n+1}).$$

we get

$$\delta\mathcal{J}_\varepsilon(v,\Lambda)(\delta v,\delta\Lambda) = \sum_{n=0}^{N-1}\int_{T_n}^{T_{n+1}}(v_n + B^* p_n, \delta v_n)_{\mathcal{U}}$$

$$+\sum_{n=0}^{N-1}(p_n(T_n^+),\delta y_n(T_n^+)) - \sum_{n=1}^{N-1}(p_{n-1}(T_n^-),\delta\lambda_n),$$

$$= \sum_{n=0}^{N-1}\int_{T_n}^{T_{n+1}}(v_n + B^* p_n, \delta v_n)_{\mathcal{U}} + \sum_{n=1}^{N-1}(p_n(T_n^+) - p_{n-1}(T_n^-),\delta\lambda_n)$$

since $\delta\lambda_0 = 0$.

From these computations, we can implement a gradient method :
Assume $y_n^k, p_n^k, v_n^k, \lambda_n^k$ are known, then

$$v_n^{k+1} = v_n^k - \rho(v_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1})$$
$$\lambda_n^{k+1} = \lambda_n^k - \rho(p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1.$$

It is quite easy to realize that the speed of convergence of the latter algorithm depends on the number N, of time steps T_n. Indeed, the transfer of information between time 0 and time T for y and between time T and time 0 for p can only be done by successive iteration through the subintervals (T_n, T_{n+1}), and requires at least N steps.

In order to understand what kind of preconditioner can be added to the previous iterative algorithm, we shall investigate in the next section the (only) virtual control. This is done by letting $B = 0$ and $\alpha = 0$.

From these computations, we can implement a gradient method :
Assume $y_n^k, p_n^k, v_n^k, \lambda_n^k$ are known, then

$$\begin{aligned} v_n^{k+1} &= v_n^k - \rho(v_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1}) \\ \lambda_n^{k+1} &= \lambda_n^k - \rho(p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1. \end{aligned}$$

It is quite easy to realize that the speed of convergence of the latter algorithm depends on the number N, of time steps T_n. Indeed, the transfer of information between time 0 and time T for y and between time T and time 0 for p can only be done by successive iteration through the subintervals (T_n, T_{n+1}), and requires at least N steps.

In order to understand what kind of preconditioner can be added to the previous iterative algorithm, we shall investigate in the next section the (only) virtual control. This is done by letting $B = 0$ and $\alpha = 0$.

From these computations, we can implement a gradient method :
Assume $y_n^k, p_n^k, v_n^k, \lambda_n^k$ are known, then

$$v_n^{k+1} = v_n^k - \rho(v_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1})$$
$$\lambda_n^{k+1} = \lambda_n^k - \rho(p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1.$$

It is quite easy to realize that the speed of convergence of the latter algorithm depends on the number N, of time steps T_n. Indeed, the transfer of information between time 0 and time T for y and between time T and time 0 for p can only be done by successive iteration through the subintervals (T_n, T_{n+1}), and requires at least N steps.

In order to understand what kind of preconditioner can be added to the previous iterative algorithm, we shall investigate in the next section the (only) virtual control. This is done by letting $B = 0$ and $\alpha = 0$.

From these computations, we can implement a gradient method :
Assume $y_n^k, p_n^k, v_n^k, \lambda_n^k$ are known, then

$$\begin{aligned} v_n^{k+1} &= v_n^k - \rho(v_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1}) \\ \lambda_n^{k+1} &= \lambda_n^k - \rho(p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1. \end{aligned}$$

It is quite easy to realize that the speed of convergence of the latter algorithm depends on the number N, of time steps T_n. Indeed, the transfer of information between time 0 and time T for y and between time T and time 0 for p can only be done by successive iteration through the subintervals (T_n, T_{n+1}), and requires at least N steps.

In order to understand what kind of preconditioner can be added to the previous iterative algorithm, we shall investigate in the next section the (only) virtual control. This is done by letting $B = 0$ and $\alpha = 0$.

The parareal scheme revisited

What we want to solve here is thus simply

$$\frac{\partial y}{\partial t} + Ay = 0$$
$$y(0) = y^0$$

over the time interval $(0, T)$.
The method of resolution through the virtual control involves a decomposition of the time interval. It is interesting to note that the cost functional becomes a function of Λ only (up to a multiplicative factor)

$$\widetilde{\mathcal{J}(\Lambda)} = \sum_{n=1}^{N-1} \|y_{n-1}(T_n^-) - \lambda_n\|^2,$$

The minimum of $\widetilde{\mathcal{J}}$ is zero and is obtained by the choice $\lambda_n = y(T_n)$.

The parareal scheme revisited

What we want to solve here is thus simply

$$\frac{\partial y}{\partial t} + Ay = 0$$
$$y(0) = y^0$$

over the time interval $(0, T)$.
The method of resolution through the virtual control involves a decomposition of the time interval. It is interesting to note that the cost functional becomes a function of Λ only (up to a multiplicative factor)

$$\widetilde{\mathcal{J}(\Lambda)} = \sum_{n=1}^{N-1} \|y_{n-1}(T_n^-) - \lambda_n\|^2,$$

The minimum of $\widetilde{\mathcal{J}}$ is zero and is obtained by the choice $\lambda_n = y(T_n)$.

The parareal scheme revisited

What we want to solve here is thus simply

$$\frac{\partial y}{\partial t} + Ay = 0$$
$$y(0) = y^0$$

over the time interval $(0, T)$.
The method of resolution through the virtual control involves a decomposition of the time interval. It is interesting to note that the cost functional becomes a function of Λ only (up to a multiplicative factor)

$$\widetilde{\mathcal{J}(\Lambda)} = \sum_{n=1}^{N-1} \|y_{n-1}(T_n^-) - \lambda_n\|^2,$$

The minimum of $\widetilde{\mathcal{J}}$ is zero and is obtained by the choice $\lambda_n = y(T_n)$.

The parareal scheme revisited

What we want to solve here is thus simply

$$\frac{\partial y}{\partial t} + Ay = 0$$
$$y(0) = y^0$$

over the time interval $(0, T)$.
The method of resolution through the virtual control involves a decomposition of the time interval. It is interesting to note that the cost functional becomes a function of Λ only (up to a multiplicative factor)

$$\widetilde{\mathcal{J}(\Lambda)} = \sum_{n=1}^{N-1} \|y_{n-1}(T_n^-) - \lambda_n\|^2,$$

The minimum of $\widetilde{\mathcal{J}}$ is zero and is obtained by the choice $\lambda_n = y(T_n)$.

The parareal scheme revisited : algebraic parareal

The succession of resolution of problems (3) is equivalent to the resolution of the initial problem if and only if $\lambda_n = \mathcal{F}_{\Delta T}(\lambda_{n-1})$ as noted in (4), or again, in a matricial form

$$\begin{pmatrix} Id & 0 & \ldots & 0 \\ -\mathcal{F}_{\Delta T} & Id & 0 & \ldots \\ 0 & -\mathcal{F}_{\Delta T} & Id & .. \\ 0 & 0 & -\mathcal{F}_{\Delta T} & Id \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ .. \\ \lambda_{N-1} \end{pmatrix} = \begin{pmatrix} \lambda_0 \\ 0 \\ .. \\ 0 \end{pmatrix} \quad (5)$$

that can also be written, with obvious notations

$$M \quad \Lambda \quad = \quad F$$

The standard inversion of this triangular system involves $\mathcal{O}(N)$ resolutions

In order to accelerate, we shall use now the formalism of the parareal in time scheme

The parareal scheme revisited : algebraic parareal

The succession of resolution of problems (3) is equivalent to the resolution of the initial problem if and only if $\lambda_n = \mathcal{F}_{\Delta T}(\lambda_{n-1})$ as noted in (4), or again, in a matricial form

$$\begin{pmatrix} Id & 0 & \dots & 0 \\ -\mathcal{F}_{\Delta T} & Id & 0 & \dots \\ 0 & -\mathcal{F}_{\Delta T} & Id & .. \\ 0 & 0 & -\mathcal{F}_{\Delta T} & Id \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ .. \\ \lambda_{N-1} \end{pmatrix} = \begin{pmatrix} \lambda_0 \\ 0 \\ .. \\ 0 \end{pmatrix} \qquad (5)$$

that can also be written, with obvious notations

$$M \quad \Lambda \quad = \quad F$$

The standard inversion of this triangular system involves $\mathcal{O}(N)$ resolutions

In order to accelerate, we shall use now the formalism of the parareal in time scheme

The parareal scheme revisited : algebraic parareal

The succession of resolution of problems (3) is equivalent to the resolution of the initial problem if and only if $\lambda_n = \mathcal{F}_{\Delta T}(\lambda_{n-1})$ as noted in (4), or again, in a matricial form

$$\begin{pmatrix} Id & 0 & \dots & 0 \\ -\mathcal{F}_{\Delta T} & Id & 0 & \dots \\ 0 & -\mathcal{F}_{\Delta T} & Id & .. \\ 0 & 0 & -\mathcal{F}_{\Delta T} & Id \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ .. \\ \lambda_{N-1} \end{pmatrix} = \begin{pmatrix} \lambda_0 \\ 0 \\ .. \\ 0 \end{pmatrix} \tag{5}$$

that can also be written, with obvious notations

$$M \quad \Lambda \quad = \quad F$$

The standard inversion of this triangular system involves $\mathcal{O}(N)$ resolutions

In order to accelerate, we shall use now the formalism of the parareal in time scheme

The parareal scheme revisited : algebraic parareal

By introducing the matrix

$$\widetilde{M} = \begin{pmatrix} Id & 0 & \dots & 0 \\ -\mathcal{G}_{\Delta T} & Id & 0 & \dots \\ 0 & -\mathcal{G}_{\Delta T} & Id & .. \\ 0 & 0 & -\mathcal{G}_{\Delta T} & Id \end{pmatrix} \qquad (6)$$

the parareal in time scheme takes the matricial form

$$\Lambda^{k+1} \quad = \quad \Lambda^k \quad + \quad \widetilde{M}^{-1} \quad Res^k$$

where the residual Res^k is defined by $Res^k = F - M\Lambda^k$.
Since this method converges rapidly, independantly of N, whenever the governing part in A is linear positive definite. It results that $\widetilde{M}^{-1}$ can be considered as close to M^{-1}, in the sense that the amplification matrix $\widetilde{M}^{-1}M$ is close to Identity.

The parareal scheme revisited : algebraic parareal

By introducing the matrix

$$\widetilde{M} = \begin{pmatrix} Id & 0 & \dots & 0 \\ -\mathcal{G}_{\Delta T} & Id & 0 & \dots \\ 0 & -\mathcal{G}_{\Delta T} & Id & .. \\ 0 & 0 & -\mathcal{G}_{\Delta T} & Id \end{pmatrix} \tag{6}$$

the parareal in time scheme takes the matricial form

$$\Lambda^{k+1} \quad = \quad \Lambda^k \quad + \quad \widetilde{M}^{-1} \quad Res^k$$

where the residual Res^k is defined by $Res^k = F - M\Lambda^k$.
Since this method converges rapidly, independantly of N, whenever the governing part in A is linear positive definite. It results that $\widetilde{M}^{-1}$ can be considered as close to M^{-1}, in the sense that the amplification matrix $\widetilde{M}^{-1}M$ is close to Identity.

Back to the extended control problem

First we note that the resolution of $\delta\widetilde{\mathcal{J}}(\Lambda) = 0$ can also be writen as

$$M^* \quad M \quad \Lambda \quad = \quad M^* \quad F$$

Indeed, the vector of jumps $p_n^k(T_n^+) - p_{n-1}^k(T_n^-)$ in the dual state is exactely equal to the vector $M^* Res^k$.
The reason why the original gradient scheme is slow comes from the fact that the conditionning of M is $\mathcal{O}(N)$. The fact that we have produced a good preconditioner for M allows to forsee that $\widetilde{M}^{-1}(\widetilde{M}^{-1})^*$ may be a good preconditionner for $M^* \; M$ so that, going back to the original control problem, we propose the following preconditionned gradient method

$$v_n^{k+1} = v_n^k - \rho(v_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1})$$

$$\lambda_n^{k+1} = \lambda_n^k - \rho[\widetilde{M}^{-1}(\widetilde{M}^{-1})^*](p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1$$

Back to the extended control problem

First we note that the resolution of $\delta\widetilde{\mathcal{J}}(\Lambda) = 0$ can also be writen as

$$M^* \quad M \quad \Lambda \quad = \quad M^* \quad F$$

Indeed, the vector of jumps $p_n^k(T_n^+) - p_{n-1}^k(T_n^-)$ in the dual state is exactely equal to the vector $M^* Res^k$.

The reason why the original gradient scheme is slow comes from the fact that the conditionning of M is $\mathcal{O}(N)$. The fact that we have produced a good preconditioner for M allows to forsee that $\widetilde{M}^{-1}(\widetilde{M}^{-1})^*$ may be a good preconditionner for M^* M so that, going back to the original control problem, we propose the following preconditionned gradient method

$$v_n^{k+1} = v_n^k - \rho(v_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1})$$

$$\lambda_n^{k+1} = \lambda_n^k - \rho[\widetilde{M}^{-1}(\widetilde{M}^{-1})^*](p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1$$

Back to the extended control problem

First we note that the resolution of $\delta\widetilde{\mathcal{J}}(\Lambda) = 0$ can also be writen as

$$M^* \quad M \quad \Lambda \quad = \quad M^* \quad F$$

Indeed, the vector of jumps $p_n^k(T_n^+) - p_{n-1}^k(T_n^-)$ in the dual state is exactely equal to the vector $M^* Res^k$.
The reason why the original gradient scheme is slow comes from the fact that the conditionning of M is $\mathcal{O}(N)$. The fact that we have produced a good preconditioner for M allows to forsee that $\widetilde{M}^{-1}(\widetilde{M}^{-1})^*$ may be a good preconditionner for $M^* \ M$ so that, going back to the original control problem, we propose the following preconditionned gradient method

$$v_n^{k+1} = v_n^k - \rho(v_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1})$$
$$\lambda_n^{k+1} = \lambda_n^k - \rho[\widetilde{M}^{-1}(\widetilde{M}^{-1})^*](p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1$$

Back to the extended control problem

First we note that the resolution of $\delta\widetilde{\mathcal{J}}(\Lambda) = 0$ can also be writen as

$$M^* \quad M \quad \Lambda \quad = \quad M^* \quad F$$

Indeed, the vector of jumps $p_n^k(T_n^+) - p_{n-1}^k(T_n^-)$ in the dual state is exactely equal to the vector $M^* Res^k$.
The reason why the original gradient scheme is slow comes from the fact that the conditionning of M is $\mathcal{O}(N)$. The fact that we have produced a good preconditioner for M allows to forsee that $\widetilde{M}^{-1}(\widetilde{M}^{-1})^*$ may be a good preconditionner for $M^* \ M$ so that, going back to the original control problem, we propose the following preconditionned gradient method

$$v_n^{k+1} = v_n^k - \rho(v_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1})$$

$$\lambda_n^{k+1} = \lambda_n^k - \rho[\widetilde{M}^{-1}(\widetilde{M}^{-1})^*](p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1$$

Back to the extended control problem

First we note that the resolution of $\delta\widetilde{\mathcal{J}}(\Lambda) = 0$ can also be writen as

$$M^* \quad M \quad \Lambda \quad = \quad M^* \quad F$$

Indeed, the vector of jumps $p_n^k(T_n^+) - p_{n-1}^k(T_n^-)$ in the dual state is exactely equal to the vector $M^* Res^k$.
The reason why the original gradient scheme is slow comes from the fact that the conditionning of M is $\mathcal{O}(N)$. The fact that we have produced a good preconditioner for M allows to forsee that $\widetilde{M}^{-1}(\widetilde{M}^{-1})^*$ may be a good preconditionner for $M^*\ M$ so that, going back to the original control problem, we propose the following preconditionned gradient method

$$\begin{aligned} v_n^{k+1} &= v_n^k - \rho(v_n^k + B^* p_n) \qquad \text{in } (T_n, T_{n+1}) \\ \lambda_n^{k+1} &= \lambda_n^k - \rho[\widetilde{M}^{-1}(\widetilde{M}^{-1})^*](p_n^k(T_n^+) - p_{n-1}^k(T_n^-)), \qquad n = 1, ..., N-1 \end{aligned}$$

A simple numerical example

Find y such that

$$\frac{\partial y}{\partial t} - y'' = v\chi \quad \text{over }]0,1[$$

where v is the control and χ is the indicator of $]1/2, 2/3[$. We have simulated this equation over the time interval $]0, 100[$ from the initial condition $y^0 = 10x(1-x)$ so as to drive it to the target $y^T = sin(2\pi x)$. The fine simulations (corresponding to $\mathcal{F}_{\Delta T}$ are performed with the time step $2.\ 10^{-2}$ either without decomposition in time or by using the preconditionned controled problem with $\Delta T = 1$.

It is impressive (and not totally understood) to obtain that after 25 iterations of the preconditionned scheme, the cost function is about the same as after 100 iterations of the plain control procedure. We have used a gradient method with optimal step. The parareal scheme thus achieves a speedup of about 4 (the time restitution is divided by 400!)

A simple numerical example

Find y such that

$$\frac{\partial y}{\partial t} - y'' = v\chi \quad \text{over }]0,1[$$

where v is the control and χ is the indicator of $]1/2, 2/3[$. We have simulated this equation over the time interval $]0, 100[$ from the initial condition $y^0 = 10x(1-x)$ so as to drive it to the target $y^T = sin(2\pi x)$. The fine simulations (corresponding to $\mathcal{F}_{\Delta T}$ are performed with the time step $2.\ 10^{-2}$ either without decomposition in time or by using the preconditionned controled problem with $\Delta T = 1$.

It is impressive (and not totally understood) to obtain that after 25 iterations of the preconditionned scheme, the cost function is about the same as after 100 iterations of the plain control procedure. We have used a gradient method with optimal step. The parareal scheme thus achieves a speedup of about 4 (the time restitution is divided by 400!)

A simple numerical example

Find y such that

$$\frac{\partial y}{\partial t} - y'' = v\chi \quad \text{over }]0,1[$$

where v is the control and χ is the indicator of $]1/2, 2/3[$. We have simulated this equation over the time interval $]0, 100[$ from the initial condition $y^0 = 10x(1-x)$ so as to drive it to the target $y^T = sin(2\pi x)$. The fine simulations (corresponding to $\mathcal{F}_{\Delta T}$ are performed with the time step 2. 10^{-2} either without decomposition in time or by using the preconditionned controled problem with $\Delta T = 1$.

It is impressive (and not totally understood) to obtain that after 25 iterations of the preconditionned scheme, the cost function is about the same as after 100 iterations of the plain control procedure. We have used a gradient method with optimal step. The parareal scheme thus achieves a speedup of about 4 (the time restitution is divided by 400!)

Problem in laser chemistry control

The Schrödinger equation is to be solved and controled

$$i\hbar \frac{\partial}{\partial t}\psi(x,t) = H_0\psi(x,t) - \varepsilon(t)\mu\psi(x,t)$$

$$\psi(x, t=0) = \psi_0(x)$$

the control variable is $\varepsilon(t)$ and the target is an observable to be optimized.

$$J(\varepsilon) = \|\psi(T) - \psi_{target}\|_{L^2} + \int_0^T \alpha(t)\varepsilon^2(t)dt$$

With G. Turinici, we have generalized a former monotonically convergent algorithms for the iterative solution of this problem proposed by Zhu-Rabitz or Tannor have proposed . The sequence that is computed from their algorithm satisfies

$$J(\varepsilon^{k+1}) \geq J(\varepsilon^k)$$

Problem in laser chemistry control

The Schrödinger equation is to be solved and controled

$$i\hbar\frac{\partial}{\partial t}\psi(x,t) = H_0\psi(x,t) - \varepsilon(t)\mu\psi(x,t)$$

$$\psi(x,t=0) = \psi_0(x)$$

the control variable is $\varepsilon(t)$ and the target is an observable to be optimized.

$$J(\varepsilon) = \|\psi(T) - \psi_{target}\|_{L^2} + \int_0^T \alpha(t)\varepsilon^2(t)dt$$

With G. Turinici, we have generalized a former monotonically convergent algorithms for the iterative solution of this problem proposed by Zhu-Rabitz or Tannor have proposed . The sequence that is computed from their algorithm satisfies

$$J(\varepsilon^{k+1}) \geq J(\varepsilon^k)$$

Problem in laser chemistry control

The Schrödinger equation is to be solved and controled

$$i\hbar \frac{\partial}{\partial t}\psi(x,t) = H_0\psi(x,t) - \varepsilon(t)\mu\psi(x,t)$$

$$\psi(x, t=0) = \psi_0(x)$$

the control variable is $\varepsilon(t)$ and the target is an observable to be optimized.

$$J(\varepsilon) = \|\psi(T) - \psi_{target}\|_{L^2} + \int_0^T \alpha(t)\varepsilon^2(t)dt$$

With G. Turinici, we have generalized a former monotonically convergent algorithms for the iterative solution of this problem proposed by Zhu-Rabitz or Tannor have proposed . The sequence that is computed from their algorithm satisfies

$$J(\varepsilon^{k+1}) \geq J(\varepsilon^k)$$

Problem in laser chemistry control

The optimality system of equations

$$i\hbar\frac{\partial}{\partial t}\psi(x,t) = H_0\psi(x,t) - \varepsilon(t)\mu\psi(x,t)$$

$$\psi(x,t=0) = \psi_0(x)$$

$$i\hbar\frac{\partial}{\partial t}\chi(x,t) = H_0\chi(x,t) - \varepsilon(t)\mu\chi(x,t)$$

$$\chi(x,t=T) = \psi_{target}(x)$$

the control variable satisfies

$$\alpha(t)\varepsilon(t) = -Im(\chi(t)|\mu|\psi(t))$$

Problem in laser chemistry control

The optimality system of equations

$$i\hbar\frac{\partial}{\partial t}\psi(x,t) = H_0\psi(x,t) - \varepsilon(t)\mu\psi(x,t)$$

$$\psi(x,t=0) = \psi_0(x)$$

$$i\hbar\frac{\partial}{\partial t}\chi(x,t) = H_0\chi(x,t) - \varepsilon(t)\mu\chi(x,t)$$

$$\chi(x,t=T) = \psi_{target}(x)$$

the control variable satisfies

$$\alpha(t)\varepsilon(t) = -Im(\chi(t)|\mu|\psi(t))$$

parareal simulation.. with Salomon and Turinici

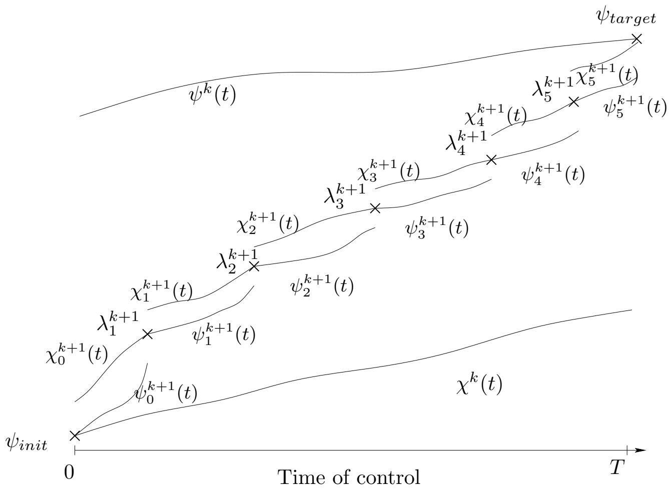
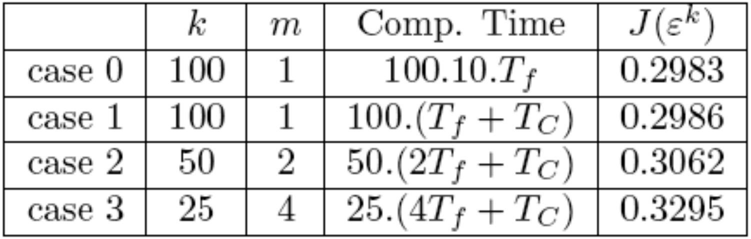


Figure: the parareal procedure .

parareal simulation.. with Salomon and Turinici



	k	m	Comp. Time	$J(\varepsilon^k)$
case 0	100	1	$100.10.T_f$	0.2983
case 1	100	1	$100.(T_f + T_C)$	0.2986
case 2	50	2	$50.(2T_f + T_C)$	0.3062
case 3	25	4	$25.(4T_f + T_C)$	0.3295

Figure: the parareal procedure .

parareal simulation.. with Salomon and Turinici

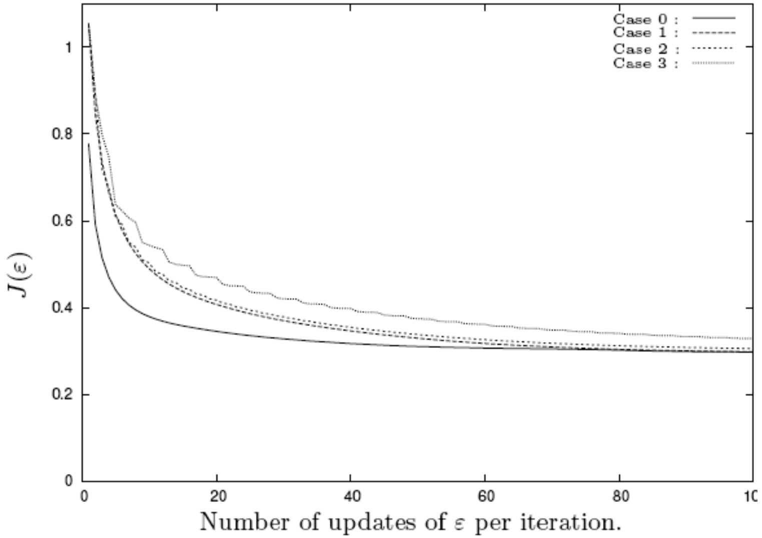


Figure: the parareal procedure .

CONCLUSIONS and perspectives

- The parareal algorithm, a predictor corrector parallel algorithm is an efficient way to get parallelism in the time direction
- We have presented a matricial formulation that allows for new ideas
- It can be combined with other iterative process, here the control, in other context with domain decomposition
- The coarse propagator can be optimized as well

CONCLUSIONS and perspectives

- The parareal algorithm, a predictor corrector parallel algorithm is an efficient way to get parallelism in the time direction
- We have presented a matricial formulation that allows for new ideas
- It can be combined with other iterative process, here the control, in other context with domain decomposition
- The coarse propagator can be optimized as well

CONCLUSIONS and perspectives

- The parareal algorithm, a predictor corrector parallel algorithm is an efficient way to get parallelism in the time direction
- We have presented a matricial formulation that allows for new ideas
- It can be combined with other iterative process, here the control, in other context with domain decomposition
- The coarse propagator can be optimized as well

CONCLUSIONS and perspectives

- The parareal algorithm, a predictor corrector parallel algorithm is an efficient way to get parallelism in the time direction
- We have presented a matricial formulation that allows for new ideas
- It can be combined with other iterative process, here the control, in other context with domain decomposition
- The coarse propagator can be optimized as well