A Parareal Algorithm for Optimal Control Problems

Felix Kwok

Hong Kong Baptist University

Joint work with M.J. Gander (Geneva) and J. Salomon (Paris-Dauphine)

Gene Golub Memorial Day
February 25, 2017

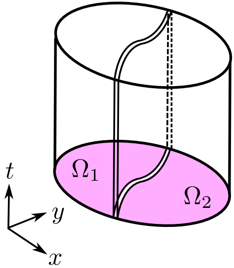
Outline

Outline

Parallel Solvers for Numerical PDEs

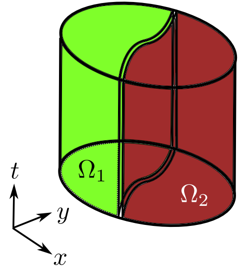
- **Goal:** Solve large, sparse (non)linear systems arising from discretized PDEs
- Exploit parallel architectures
- Use iterative methods/preconditioners in which each step has components that can be done in parallel

Domain Decomposition in Space



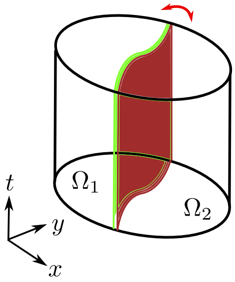
- Decompose domain into subdomains

Domain Decomposition in Space



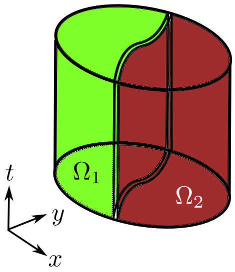
- Decompose domain into subdomains
- Iterate until convergence:
  1. Solve problem in each subdomain in parallel

Domain Decomposition in Space



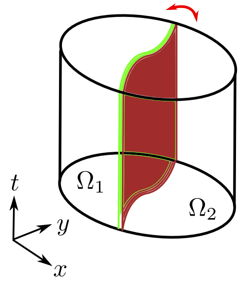
- Decompose domain into subdomains
- Iterate until convergence:
  1. Solve problem in each subdomain in parallel
  2. Exchange interface data

Domain Decomposition in Space



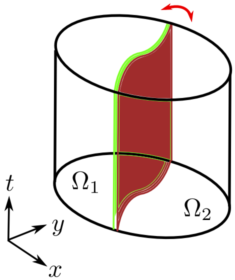
- Decompose domain into subdomains
- Iterate until convergence:
  1. Solve problem in each subdomain in parallel
  2. Exchange interface data

Domain Decomposition in Space



- Decompose domain into subdomains
- Iterate until convergence:
  1. Solve problem in each subdomain in parallel
  2. Exchange interface data

Domain Decomposition in Space

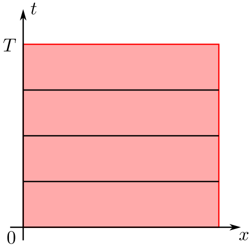


- Decompose domain into subdomains
- Iterate until convergence:
  1. Solve problem in each subdomain in parallel
  2. Exchange interface data
- Essentially a block Jacobi method/preconditioner

Parallelization in Time?

- Solve systems of ODEs ($\mathbf{y}' = f(t, \mathbf{y})$) or discretized PDEs ($\frac{\partial y}{\partial t} = \mathcal{L}y + f$)
- Block triangular system

$$\begin{bmatrix} A & & & \\ -I & A & & \\ & \ddots & \ddots & \\ & & -I & A \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix}$$



- Is it possible to do useful computation at future time steps, before earlier time steps are known?

Parareal for IVP (Lions, Maday & Turinici 2001)

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]

Résumé. On propose dans cette Note un schéma permettant de profiter d'une architecture parallèle pour la discrétisation en temps d'une équation d'évolution aux dérivées partielles. Cette méthode, basée sur un schéma d'Euler, combine des résolutions grossières et des résolutions fines et indépendantes en temps en s'inspirant de ce qui est classique en espace. La parallélisation qui en résulte se fait dans la direction temporelle ce qui est en revanche non classique. Elle a pour principale motivation les problèmes en temps réel, d'où la terminologie proposée de « pararéel ». © 2001 Académie des sciences/Éditions scientifiques et médicales Elsevier SAS

A "parareal" in time discretization of PDE's

Abstract. *The purpose of this Note is to propose a time discretization of a partial differential evolution equation that allows for parallel implementations. The method, based on an Euler scheme, combines coarse resolutions and independent fine resolutions in time in the same spirit as standard spacial approximations. The resulting parallel implementation is done in the non standard time direction. Its main goal concerns real time problems, hence the proposed terminology of "parareal" algorithm.* © 2001 Académie des sciences/Éditions scientifiques et médicales Elsevier SAS

Parareal for IVP (Lions, Maday & Turinici 2001)

- Unknowns: $Y_n \approx y(t_n)$
- Fine/coarse propagators: $P^{\delta t}(t_{n+1}; t_n, y_n)$, $P^{\Delta t}(t_{n+1}; t_n, y_n)$
- For $k = 1, 2, \dots$:
  1. Solve **fine** problems **in parallel**:

$$y_{n+1}^k(t_{n+1}) = P^{\delta t}(t_{n+1}; t_n, Y_n^k)$$

  2. Correct initial conditions using **coarse propagator**:

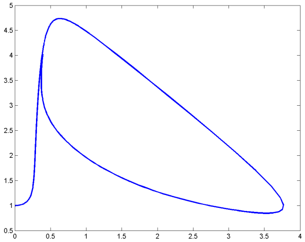
$$Y_{n+1}^{k+1} = y_{n+1}^k(t_{n+1}) + P^{\Delta t}(t_{n+1}; t_n, Y_n^{k+1}) - P^{\Delta t}(t_{n+1}; t_n, Y_n^k)$$

- Initial guesses for Y_n^0 obtained by coarse propagation

Example: Brusselator

- Model equation:

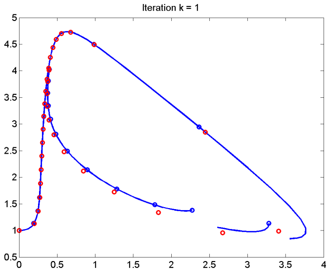
$$\frac{dx}{dt} = A + x^2y - (B+1)x$$
$$\frac{dy}{dt} = Bx - x^2y$$

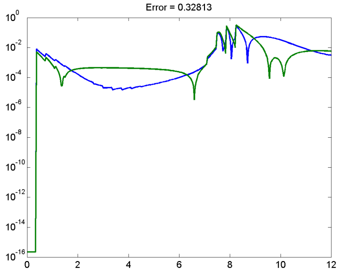


- Parameters: $A = 1$, $B = 3$, $x(0) = 0$, $y(0) = 1$
- Use 4th order Runge-Kutta with 32 coarse steps and 320 fine steps

香港浸會大學 HONG KONG BAPTIST UNIVERSITY

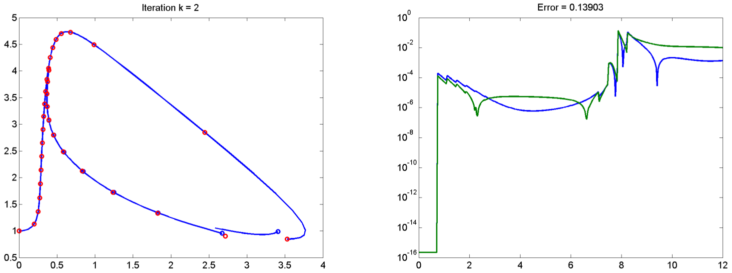
Example: Brusselator





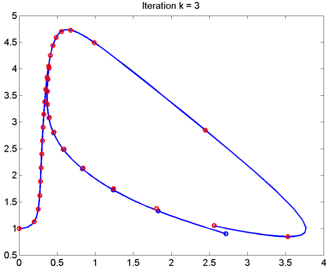
- Method is exact in the first k intervals after k iterations
- Parallel speedup $\leq N/K$, where $K =$#iters to convergence

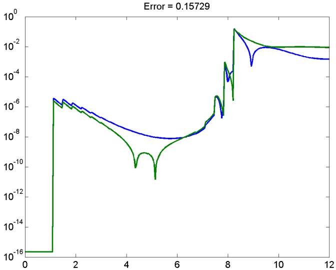
Example: Brusselator



- Method is exact in the first k intervals after k iterations
- Parallel speedup $\leq N/K$, where $K =$#iters to convergence

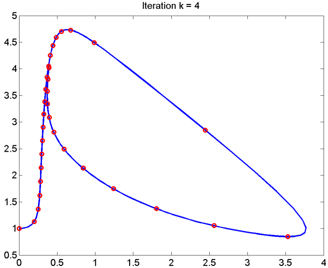
Example: Brusselator

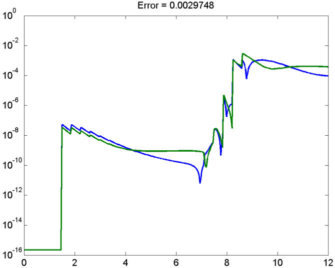




- Method is exact in the first k intervals after k iterations
- Parallel speedup $\leq N/K$, where $K =$#iters to convergence

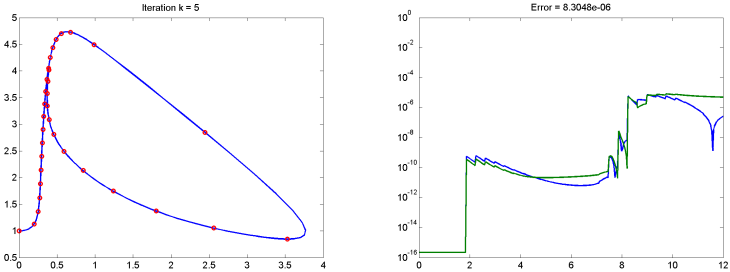
Example: Brusselator





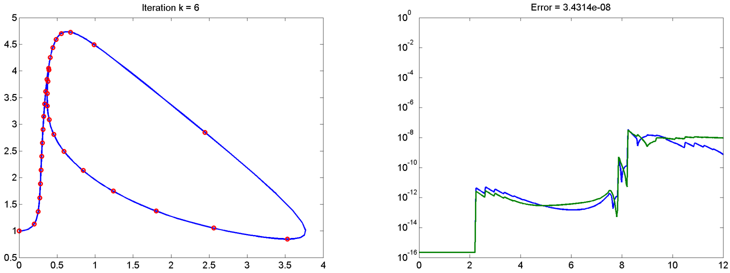
- Method is exact in the first k intervals after k iterations
- Parallel speedup $\leq N/K$, where $K =$#iters to convergence

Example: Brusselator



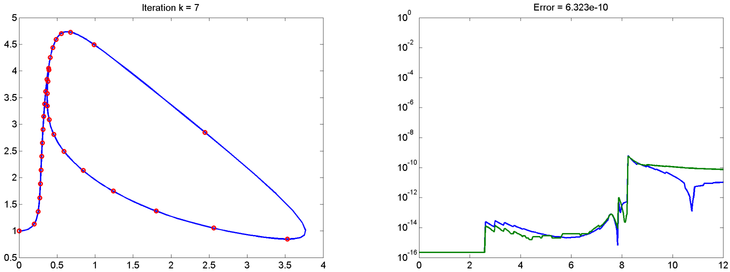
- Method is exact in the first k intervals after k iterations
- Parallel speedup $\leq N/K$, where $K =$#iters to convergence

Example: Brusselator



- Method is exact in the first k intervals after k iterations
- Parallel speedup $\leq N/K$, where $K =$#iters to convergence

Example: Brusselator



- Method is exact in the first k intervals after k iterations
- Parallel speedup $\leq N/K$, where $K =$#iters to convergence

Parareal = Inexact Newton (Gander & Vandewalle 2003)

- Define the **residual function**

$$F(Y) = \begin{pmatrix} Y_0 - y_{\text{init}} \\ Y_1 - P^{\delta t}(t_1; t_0, Y_0) \\ \vdots \\ Y_N - P^{\delta t}(t_N; t_{N-1}, Y_{N-1}) \end{pmatrix} = 0.$$

- Newton's method reads (for $n \geq 1$)

$$Y_n^{k+1} = P^{\delta t}(t_n; t_{n-1}, Y_n^k) + \frac{\partial P^{\delta t}}{\partial Y}(t_n; t_{n-1}, Y_{n-1}^k)(Y_n^{k+1} - Y_n^k).$$

- Parareal replaces the derivative with a finite difference:

$$\frac{\partial P^{\delta t}}{\partial Y}(\cdot)(Y_n^{k+1} - Y_n^k) \approx P^{\Delta t}(t_{n+1}; t_n, {\color{red}Y_n^{k+1}}) - P^{\Delta t}(t_{n+1}; t_n, {\color{red}Y_n^k})$$

Derivative Parareal (Gander & Hairer 2014)

- Newton's method:

$$Y_n^{k+1} = P^{\delta t}(t_n; t_{n-1}, Y_n^k) + \frac{\partial P^{\delta t}}{\partial Y}(t_n; t_{n-1}, Y_{n-1}^k)(Y_n^{k+1} - Y_n^k).$$

- Approximate derivatives cheaply using coarse propagator:

$$Y_n^{k+1} = P^{\delta t}(t_n; t_{n-1}, Y_n^k) + \frac{\partial P^{\Delta t}}{\partial Y}(t_n; t_{n-1}, Y_{n-1}^k)(Y_n^{k+1} - Y_n^k).$$

- Compare with Parareal:

$$Y_{n+1}^{k+1} = P^{\delta t}(t_n; t_{n-1}, Y_n^k) + P^{\Delta t}(t_{n+1}; t_n, Y_n^{k+1}) - P^{\Delta t}(t_{n+1}; t_n, Y_n^k)$$

Outline

Optimal Control Problem

- Optimal control problem: minimize

$$J[y,u] = \frac{1}{2}\|y(T) - y_{\text{target}}\|^2 + \frac{\alpha}{2}\int_0^T \|u(t)\|^2\,dt$$

subject to the (non)-linear ODE constraint

$$\dot{y}(t) = f(y(t)) + u(t), \qquad t \in (0,T).$$

- Assumptions:
  1. No state or control constraints
  2. Distributed control entering additively
- Includes cases where f is the discretization of a partial differential operator

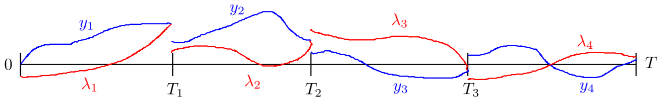
Optimality System

- Using Lagrange multipliers, we deduce the optimality conditions to be

$$
\begin{cases} \dot{y}(t) = f(y(t)) - \dfrac{1}{\alpha}\lambda(t), \\ y(0) = y_{\text{init}}, \end{cases} \qquad \begin{cases} \dot{\lambda}(t) = -(f'(y(t)))^T\lambda(t), \\ \lambda(T) = y(T) - y_{\text{target}}. \end{cases}
$$

- Coupled forward-backward problem!

Time-domain decomposition for control



- Divide time horizon $(0, T)$ into "subdomains" $I_i = (T_{i-1}, T_i)$
- Subdomain problem on I_i well defined when $y(T_{i-1})$ and $\lambda(T_i)$ are given
- Multiple shooting: solve for intermediate states $Y_i = y(T_i)$ and $\Lambda_i = \lambda(T_i)$

Parareal for Control

- Unknowns: state $Y_n \approx y(t_n)$ and adjoint $\Lambda_n \approx \lambda(t_n)$
- Fine/coarse propagators:

$$
\begin{aligned}
P^{\delta t}, P^{\Delta t}: &\quad (Y_n, \Lambda_{n+1}) \mapsto y(t_{n+1}),\\
Q^{\delta t}, Q^{\Delta t}: &\quad (Y_n, \Lambda_{n+1}) \mapsto \lambda(t_n)
\end{aligned}
$$

- Residual function:

$$
F(Y,\Lambda) = \begin{pmatrix}
Y_0 - y_{\text{init}} \\
Y_1 - P^{\delta t}(t_1; t_0, Y_0, \Lambda_1) \\
\vdots \\
Y_N - P^{\delta t}(t_N; t_{N-1}, Y_{N-1}, \Lambda_N) \\
\Lambda_1 - Q^{\delta t}(t_1; t_2, Y_1, \Lambda_2) \\
\vdots \\
\Lambda_{N-1} - Q^{\delta t}(t_{N-1}; t_N, Y_{N-1}, \Lambda_N) \\
\Lambda_N - y_N + y_{\text{target}}
\end{pmatrix} = 0
$$

香港浸會大學 HONG KONG BAPTIST UNIVERSITY

Parareal for Control

- Newton's method:

$$Y_n^{k+1} = P^{\delta t}(t_n; t_{n-1}, Y_{n-1}^k, \Lambda_n^k) + \frac{\partial P^{\delta t}}{\partial Y}(Y_{n-1}^{k+1} - Y_{n-1}^k) + \frac{\partial P^{\delta t}}{\partial \Lambda}(\Lambda_n^{k+1} - \Lambda_n^k)$$
$$\Lambda_n^{k+1} = Q^{\delta t}(t_n; t_{n+1}, Y_n^k, \Lambda_{n+1}^k) + \frac{\partial Q^{\delta t}}{\partial Y}(Y_n^{k+1} - Y_n^k) + \frac{\partial Q^{\delta t}}{\partial \Lambda}(\Lambda_{n+1}^{k+1} - \Lambda_{n+1}^k)$$

- Derivatives expensive to evaluate
- Use parareal approximation

Parareal for Control

- Derivative Parareal:

$$Y_n^{k+1} = P^{\delta t}(t_n; t_{n-1}, Y_{n-1}^k, \Lambda_n^k) + \frac{\partial P^{\Delta t}}{\partial Y}(Y_{n-1}^{k+1} - Y_{n-1}^k) + \frac{\partial P^{\Delta t}}{\partial \Lambda}(\Lambda_n^{k+1} - \Lambda_n^k)$$
$$\Lambda_n^{k+1} = Q^{\delta t}(t_n; t_{n+1}, Y_n^k, \Lambda_{n+1}^k) + \frac{\partial Q^{\Delta t}}{\partial Y}(Y_n^{k+1} - Y_n^k) + \frac{\partial Q^{\Delta t}}{\partial \Lambda}(\Lambda_{n+1}^{k+1} - \Lambda_{n+1}^k)$$

- Still coupled, must solve

$$\widetilde{F}'(Y^k, \Lambda^k)\begin{pmatrix} Y^{k+1} - Y^k \\ \Lambda^{k+1} - \Lambda^k \end{pmatrix} = -F(Y^k, \Lambda^k)$$

  using an iterative method, e.g., preconditioned GMRES
- $\widetilde{F}'$ = approximation of Jacobian matrix

Parareal for Control: Algorithm

- Unknowns: $Y_n \approx y(t_n)$, $\Lambda_n \approx \lambda(t_n)$
- For $k = 1, 2, \ldots$:
  1. Solve fine subdomain problems **in parallel**:
  $$y_n(t_{n+1}) = P^{\delta t}(Y_n^k, \Lambda_{n+1}^k), \quad \lambda_n(t_n) = Q^{\delta t}(Y_n^k, \Lambda_{n+1}^k)$$
  2. Solve $\tilde{F}'(Y^k, \Lambda^k) \begin{pmatrix} \Delta Y \\ \Delta \Lambda \end{pmatrix} = -F(Y^k, \Lambda^k)$ by GMRES:
     - Calculate P_Y, P_Λ, Q_Y, Q_Λ with coarse propagator on **each subdomain**
     - Matrix multiplication = **in parallel**
     - Preconditioning = backward/forward sweep
  3. Update: $Y^{k+1} = Y^k + \Delta Y$, $\Lambda^{k+1} = \Lambda^k + \Delta\Lambda$

Parallel Speedup

- Assume cost of solving problem $\propto$ # time steps
- Global fine problem (no parareal): $C_1 T/\delta t$
- Parareal with N subdomains:
  - Fine propagation: $C_1 T/(N\delta t)$ (parallel)
  - Coarse propagation: $C_1 T/(N\Delta t)$ (parallel)
  - Preconditioning: $2T/\Delta t$ (sequential)
- If K parareal iterations are needed for convergence, then

$$\text{Speedup} = \frac{C_1 T/\delta t}{K(C_1 T/(N\delta t) + C_1 T/(N\Delta t) + 2T/\Delta t)}$$

Parallel Speedup

- Assume cost of solving problem $\propto$ # time steps
- Global fine problem (no parareal): $C_1 T/\delta t$
- Parareal with N subdomains:
  - Fine propagation: $C_1 T/(N\delta t)$ (parallel)
  - Coarse propagation: $C_1 T/(N\Delta t)$ (parallel)
  - Preconditioning: $2T/\Delta t$ (sequential)
- If K parareal iterations are needed for convergence, then

$$\text{Speedup} = \frac{N}{K}\left(1 + \frac{\delta t}{\Delta t}(1 + 2N/C_1)\right)^{-1}$$

Convergence of Parareal for Control

- Method works for linear as well as nonlinear problems
- Convergence results for linear problems via eigenvalue analysis
- Results for implicit Euler:
  - Contraction factor: $\rho \leq C(\Delta t - \delta t)$
  - For dissipative problems, C can be chosen independent of σ
  - For very large α (heavy penalization against large control values), C can grow like $\log(\alpha)$ when the number of subdomains becomes large

Outline

Parareal

Parareal for Control

Numerical Experiments

Conclusion

Scalar example

- Minimize

$$J[y,u] = \frac{1}{2}|y(1) - y_{\text{target}}|^2 + \frac{1}{2}\int_0^1 u^2(t)\,dt$$

with $y_{\text{target}} = 1$, subject to

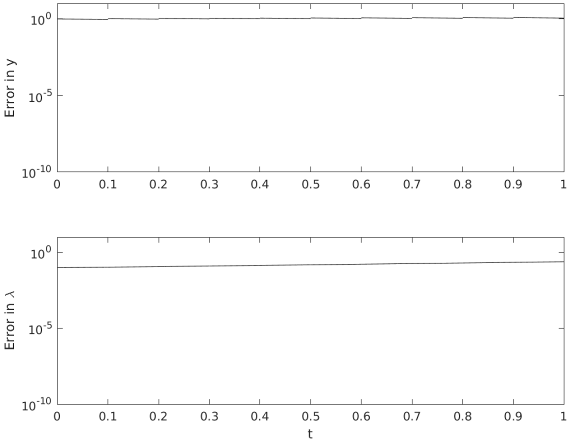
$$\frac{\partial y}{\partial t} = \sin(y) + u, \qquad t \in (0,1)$$

$$y(0) = 1$$

- Backward Euler, $\delta t = 10^{-6}$

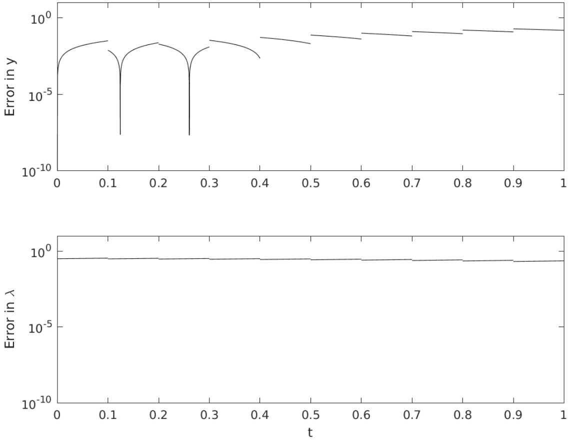
Scalar example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #1



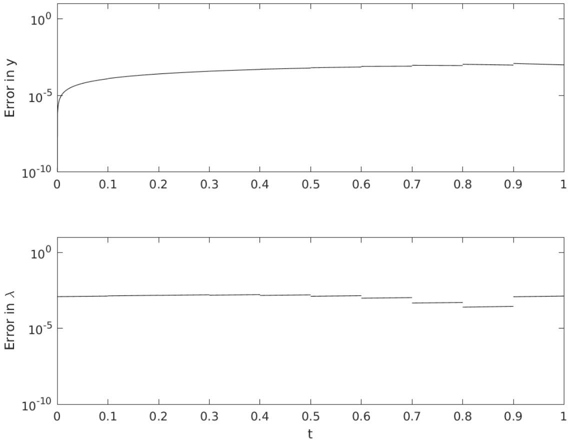
Scalar example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #2



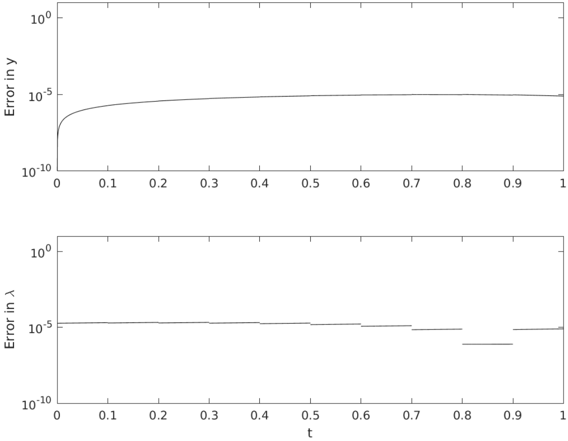
Scalar example - $N = 10$, $r = \delta t/\Delta t = 0.01$

Iteration #3



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

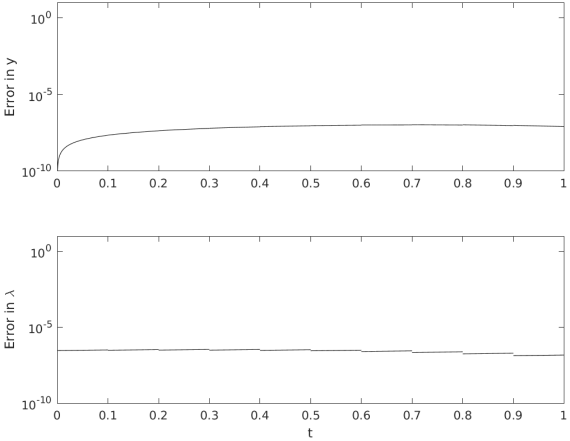
Iteration #4



香港浸會大學 HONG KONG BAPTIST UNIVERSITY

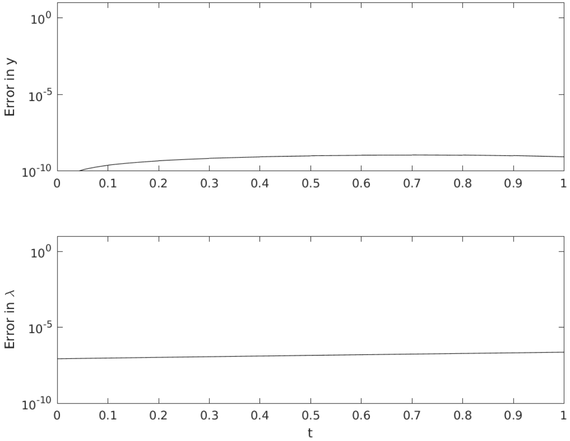
Scalar example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #5



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

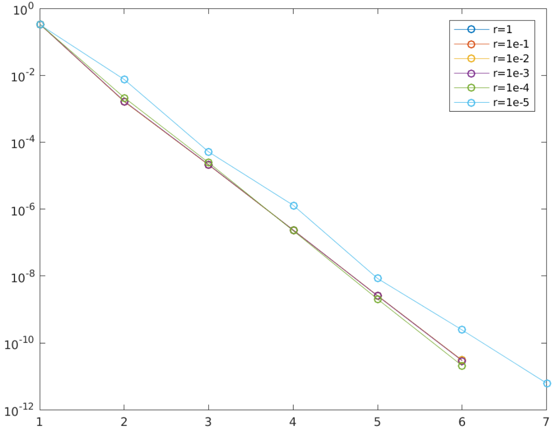
Iteration #6



香港浸會大學 HONG KONG BAPTIST UNIVERSITY

Scalar example

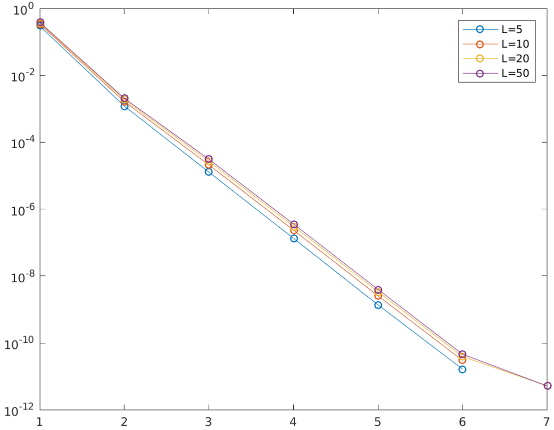
$N = 10$ subdomains, varying $r = \delta t / \Delta t$



香港浸會大學
HONG KONG BAPTIST UNIVERSITY

Scalar example

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



Vector example

- Minimize

$$J[y,u] = \frac{1}{2}|y(1) - y_{\text{target}}|^2 + \frac{1}{2}\int_0^1 |u(t)|^2\,dt$$

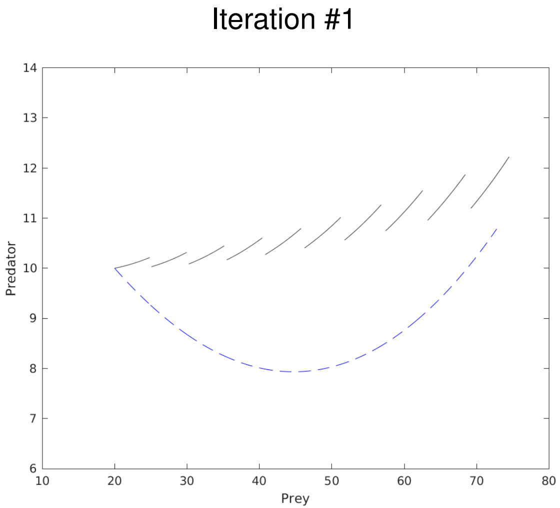
with $y_{\text{target}} = (100, 20)^T$, subject to the Lotka-Volterra equation

$$\dot{y}_1 = ay_1 - by_1y_2, \quad \dot{y}_2 = cy_1y_2 - dy_2$$

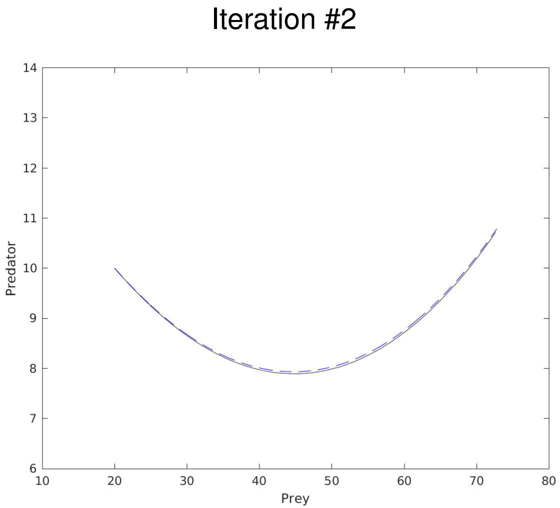
with initial conditions $y(0) = (20, 10)^T$

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

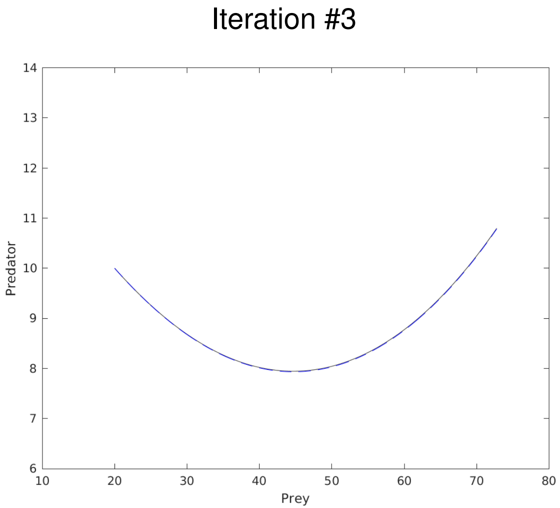
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$



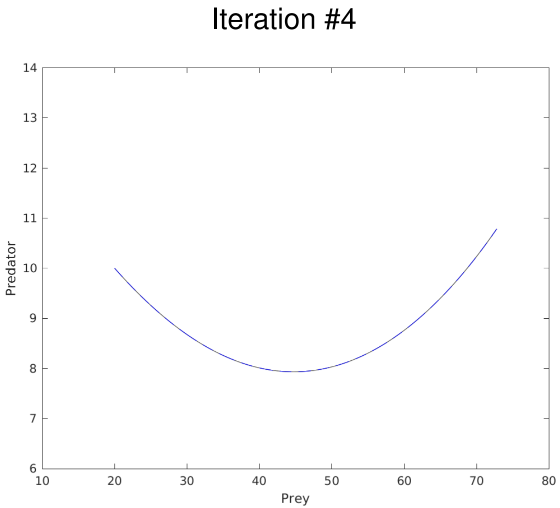
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$



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

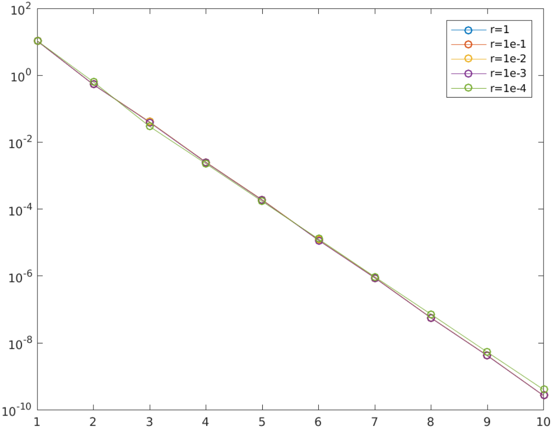


Vector example - $N = 10$, $r = \delta t/\Delta t = 0.01$



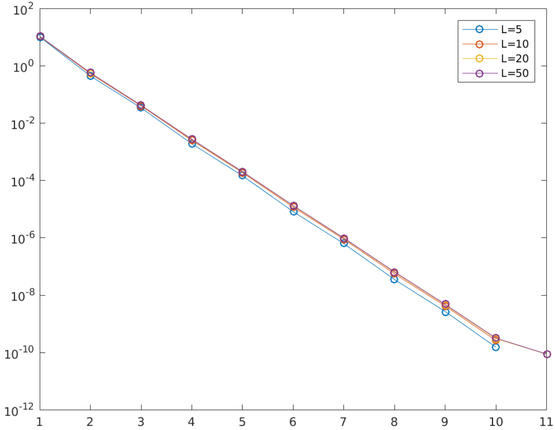
Vector example

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



Vector example

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



Vector example

- Minimize

$$J[y,u] = \frac{1}{2}|y(20) - y_{\text{target}}|^2 + \frac{1}{2}\int_0^{20} |u(t)|^2\,dt$$

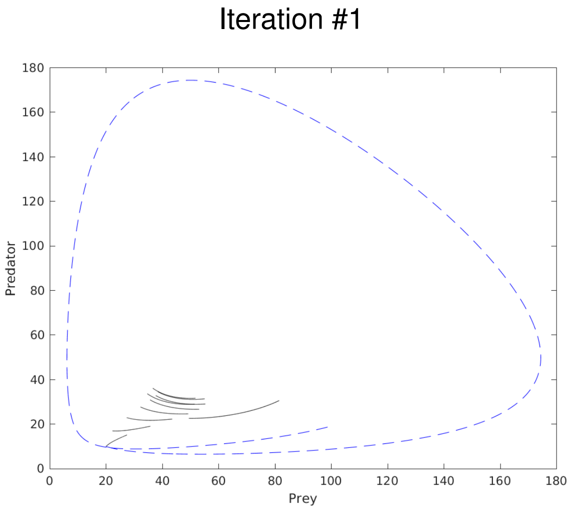
with $y_{\text{target}} = (100, 20)^T$, subject to the Lotka-Volterra equation

$$\dot{y}_1 = ay_1 - by_1y_2, \quad \dot{y}_2 = cy_1y_2 - dy_2$$

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

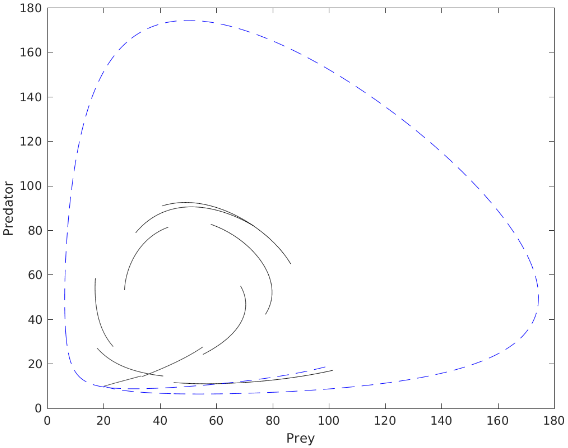
- Backward Euler, $\delta t = 20 \cdot 10^{-5}$

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

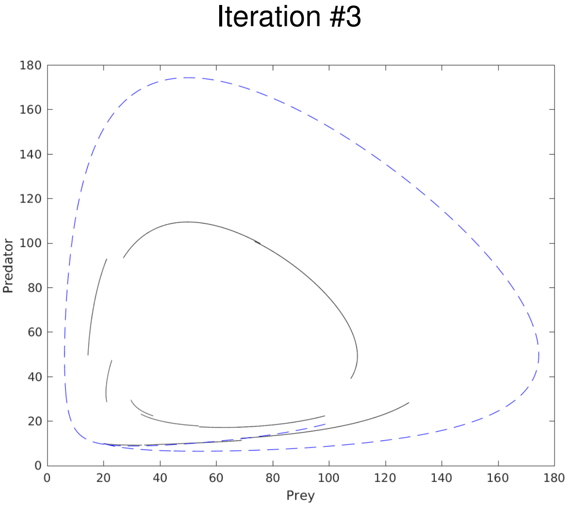


Vector example - $N = 10$, $r = \delta t/\Delta t = 0.01$

Iteration #2

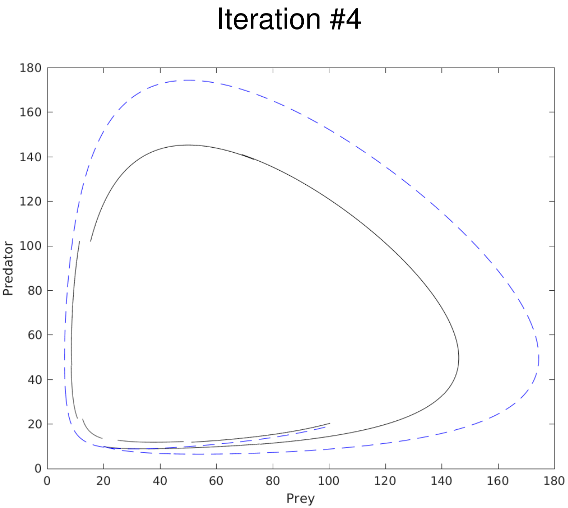


Vector example - $N = 10$, $r = \delta t/\Delta t = 0.01$



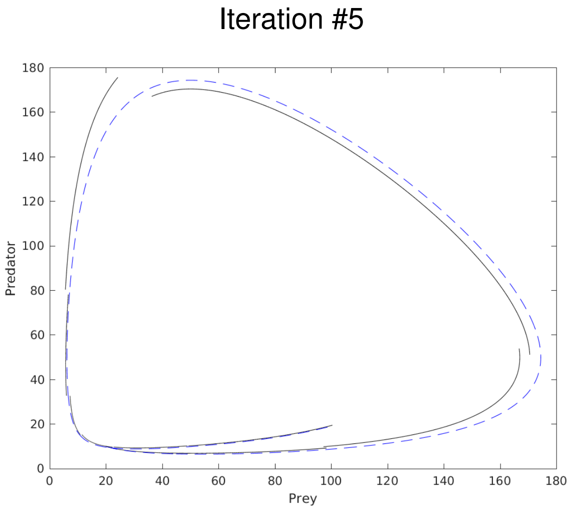
香港浸會大學 HONG KONG BAPTIST UNIVERSITY

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



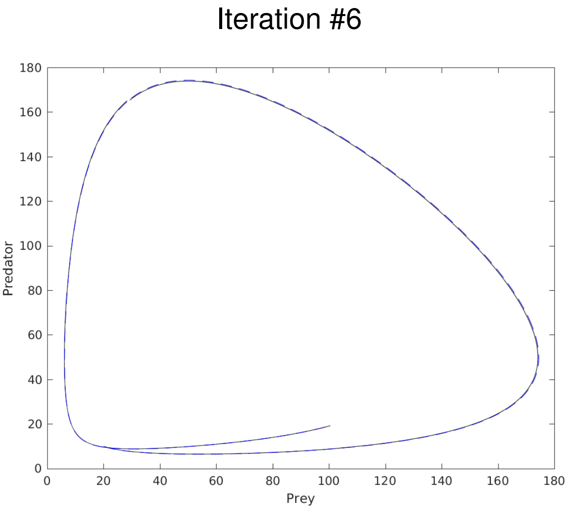
香港浸會大學 HONG KONG BAPTIST UNIVERSITY

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



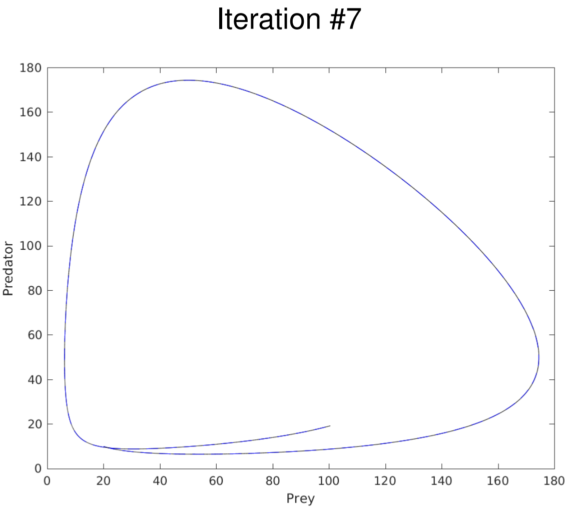
香港浸會大學 HONG KONG BAPTIST UNIVERSITY

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



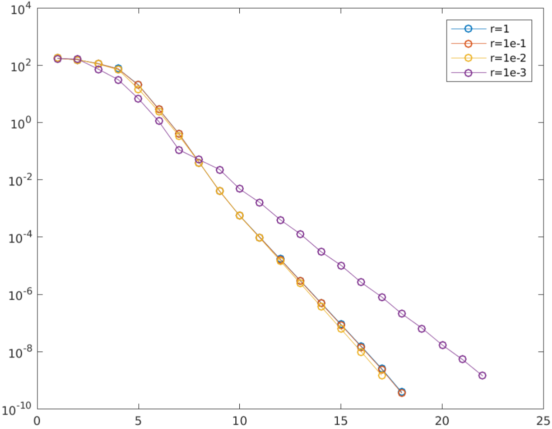
香港浸會大學 HONG KONG BAPTIST UNIVERSITY

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



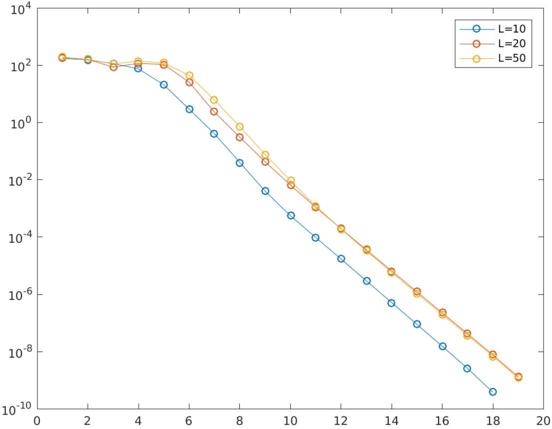
Vector example

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



Vector example

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



PDE example

- Minimize

$$J[y,u] = \frac{1}{2}\|y(x,1) - y_{\text{target}}(x)\|^2 + \frac{0.1}{2}\int_0^1 u^2(t)\,dt$$

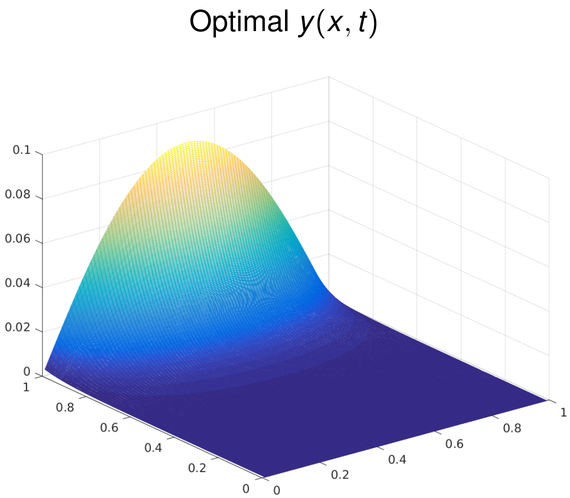
with $y_{\text{target}}(x) = x(1-x)$, subject to the minimal surface PDE

$$\frac{\partial y}{\partial t} = \frac{\partial}{\partial x}\left(\frac{y_x}{\sqrt{1+|y_x|^2}}\right) + u, \qquad (x,t) \in (0,1)^2$$

$$y(x,0) = 0, \qquad y(0,t) = y(1,t) = 0$$

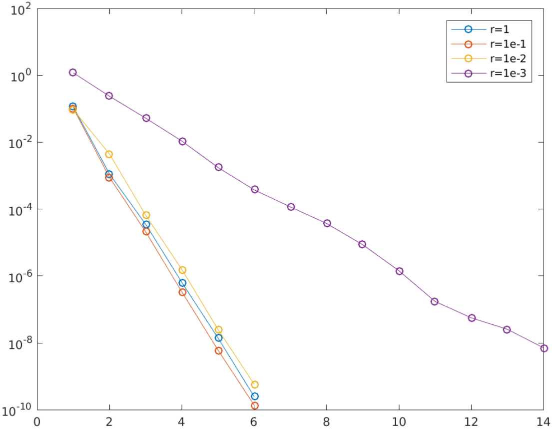
- Centered difference in space with $h = 1/100$
- Backward Euler, $\delta t = 10^{-4}$

PDE example



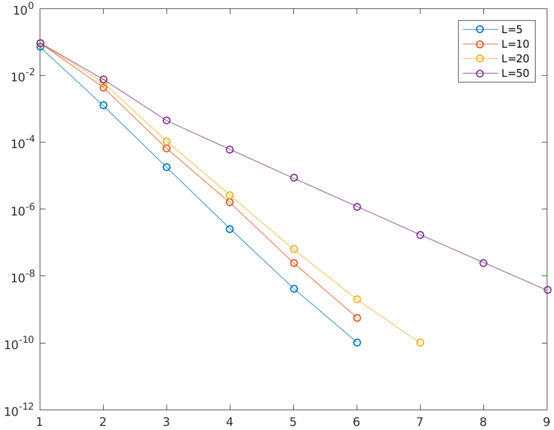
PDE example

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



PDE example

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



Outline

Conclusion

- Parareal for control problems involving nonlinear ODE/PDEs
- Derivatives easier to approximate than finite difference $\Longrightarrow$ derivative parareal variant
- Contraction factor proportional to coarse step sizes
- Ongoing work:
    - Control entering nonlinearly, or only over part of the domain
    - Spatial coarsening in the PDE case
    - Control constraints

Other approaches

DD on control problem:

- Heinkenschloss (JCAM 2005): block symmetric Gauss-Seidel preconditioning
- Maday, Salomon and Turinici (SINUM 2007): Quantum control, method of intermediate targets
- Maday, Riahi & Salomon (2013): Intermediate targets for parabolic problems
- Barker & Stoll (2015), Gander & Kwok (2016): Schwarz methods in time

Related:

- Apply parareal to forward and backward solves in shooting method (Mathew, Sarkis & Shaerer 2010, Ulbrich 2015)
- Multigrid methods (Hackbusch 1984, Borzì 2003)

Thank you!