Domain decomposition and optimal control

Julien Salomon

Laboratoire J.-L. Lions, Sorbonne Université
&
ANGE project-team, INRIA

Problem: control on a fixed, bounded interval $[0, T]$

Given $T > 0$, consider the optimal control problem associated with the cost functional

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

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

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

with initial condition $y(0) = y_{init}$.

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

Optimality system (1/2)

How to characterize the optima ?

$\rightarrow$ Solve the *Euler-Lagrange* equations !

- Define the *Lagrange operator*:

$$\mathcal{L}(y, p, c) = J(c) - \int_0^T p(t) \cdot (\dot{y}(t) - f(y(t), c(t)))dt$$

- Compute its partial derivatives.
- Cancel them !

Optimality system (2/2)

The optima are characterized by the Euler-Lagrange equations:

$$
\begin{aligned}
\dot{y}(t) &= f(y(t), c(t)) \\
y(t=0) &= y_0 \\
\dot{p}(t) &= -\left[\partial_y f(y(t), c(t))\right] p(t) \\
p(t=T) &= y(T) - y_0 \\
\alpha c(t) &= -\, p(t) \cdot \partial_c f(y(t), c(t))
\end{aligned}
$$

"Non-linear control" or "Bilinear control"

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

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

Outline

1. An intermediate states method to parallelize

2. Linear Control
   - Time sub-intervals decomposition
   - Use of a coarse solver
   - Numerical examples

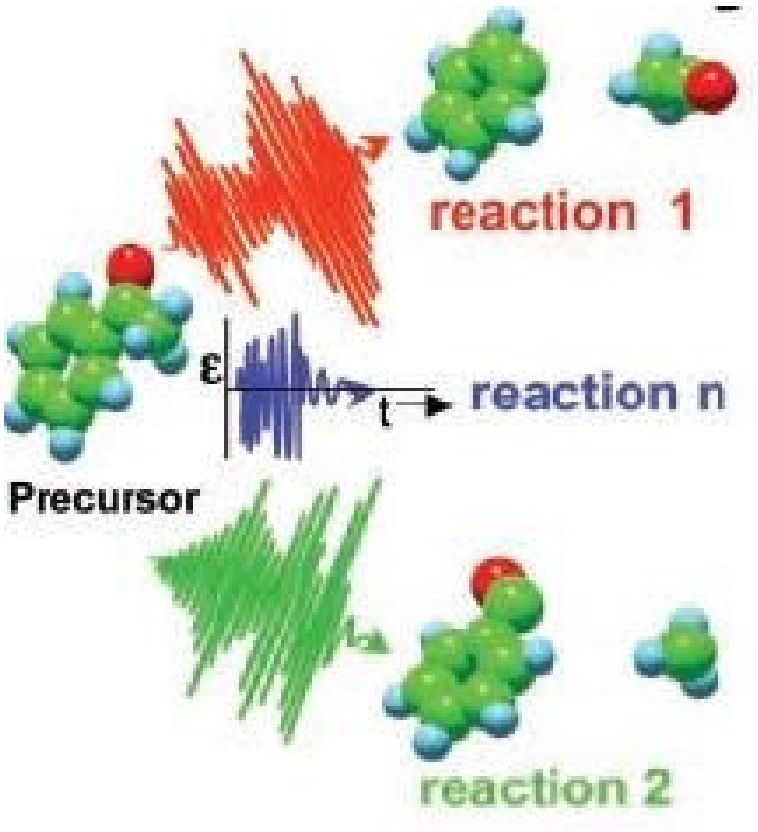
Outline

1. An intermediate states method to parallelize

2. Linear Control
   - Time sub-intervals decomposition
   - Use of a coarse solver
   - Numerical examples

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

Example: control physicochemical phenomena with laser pulses



R.J. Levis, G.M. Menkir, H. Rabitz, *Science*, 292, pp. 709-713, 2001.

Mathematical model : The Schrödinger equation

$$i\frac{\partial\psi(x,t)}{\partial t} = [\Delta + V(x) - \mu(x)\varepsilon(t)]\psi(x,t)$$

- $\psi(\cdot,t) \leftrightarrow y(t)$
- $\varepsilon(t) \leftrightarrow c(t)$
- $[\Delta + V(x) - \mu(x)\varepsilon(t)]\psi(x,t) \leftrightarrow f(y(t),c(t))$

Similar issue: Nuclear Magnetic Resonance

Aim : control spin using Magnetic fields.

Applications :

- Medical imaging
- Quantum computing
- Porous media identification
- ...

$$\frac{\partial M}{\partial t}(t,\omega) = \Big[u(t)e_1 + v(t)e_2 + \omega e_3\Big] \wedge M(t,\omega), \quad \omega \in (\omega_*, \omega^*)$$

Optimal control problems, usual requirements:

- Real-time constraints,
- Requires many propagations over the considered time interval.

Optimal control problems, usual requirements:

- Real-time constraints,
- Requires many propagations over the considered time interval.

$\rightarrow$ use parallelization to accelerate the computations.

Optimal control problems, usual requirements:

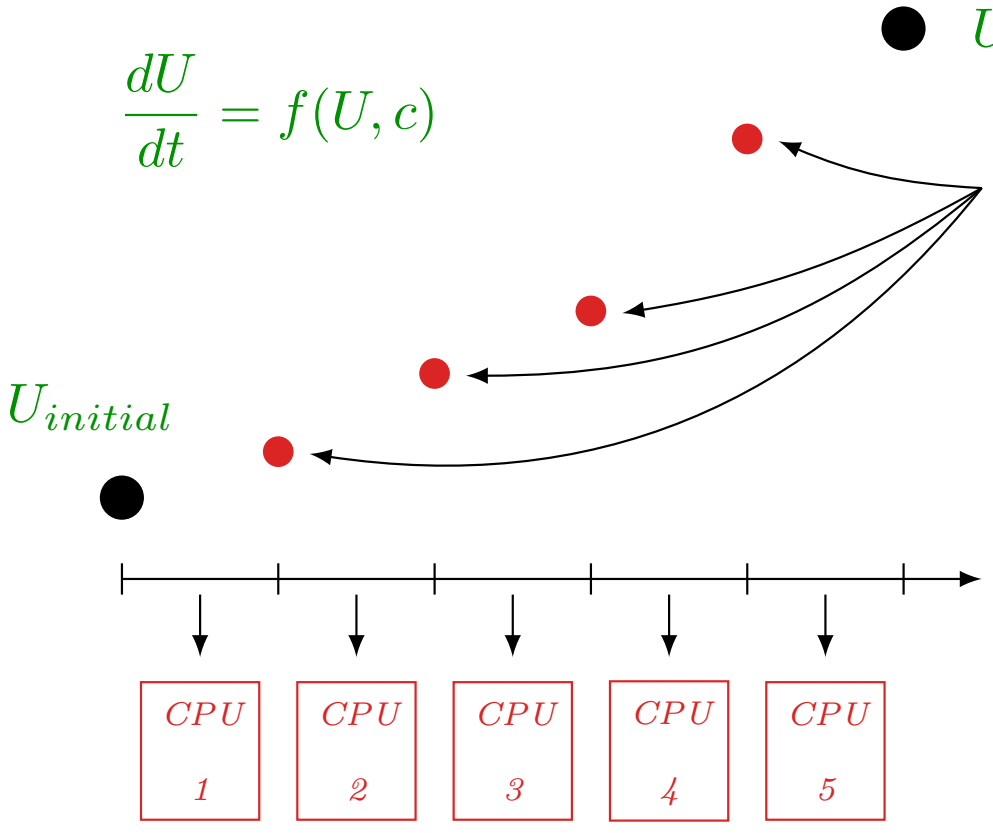
- Real-time constraints,
- Requires many propagations over the considered time interval.

$\rightarrow$ use parallelization to accelerate the computations.

Question: *How to reach full efficiency ?*

An intermediate states method

The idea behind the algorithm



Disclaimer: not a parareal algorithm.

And it follows:

- Independent sub-problems

$$J \to (J_j)_{j=1,\cdots,N} ,$$

- Need for an update formula for the intermediate states

$$\Lambda = (\lambda_j)_{j=1,\cdots,N} .$$

Algorithm:

Given c^k, Λ^k (intermediate targets) at step k:

1. *solve in parallel on* $[T_j, T_{j+1}]$

$$\max_{c_j} J_j(c_j) \to c_j^{k+1},$$

2. *define* c^{k+1} *as the concatenation of* c_j^{k+1},
3. *define* Λ^{k+1} *in a "relevant way" with* c^{k+1}, *so that the consistency lemma holds.*

Parallelization setting:

Define $\lambda_0 = \psi_0$, $\lambda_N = \psi_{target}$, $c_j = c_{|[T_j, T_{j+1}]}$ and $\beta_j = \frac{T}{T_{j+1} - T_j}$.

$$J_{\|}(c, \Lambda) = \sum_{j=0}^{N-1} \beta_j J_j(c_j, \lambda_j, \lambda_{j+1})$$

where J_j are the *parareal cost functionals*:

$$J_j(c_j, \lambda_j, \lambda_{j+1}) = \|\psi_j(T_{j+1}^-) - \lambda_{j+1}\|_{L^2}^2 + \int_{T_j}^{T_{j+1}} \alpha_j'(t) c_j(t)^2 dt,$$

$$\alpha_j'(t) = \frac{\alpha(t)}{\beta_j}, \ \psi_j(T_j^+) = \lambda_j.$$

Theorem

Given c, with the previous notations, let us define $\Lambda^c = (\lambda_j^c)_{j=1,\dots,N-1}$ by:

$$\lambda_j^c = (1-\gamma_j)\psi(T_j) + \gamma_j \chi(T_j),$$

where $\gamma_j = \frac{T_j}{T}$.
Then:

$$\Lambda^c = \operatorname{argmin}_\Lambda \left(J_{\|}(c,\Lambda)\right).$$

Moreover we have:

$$J_{\|}(c,\Lambda^c) = J(c).$$

Convergence?

$$J_{\|}(c^{k+1},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k}) = \begin{array}{l} J_{\|}(c^{k+1},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k+1}) \\ J_{\|}(c^{k},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k}) \end{array}$$

Convergence?

$$\begin{aligned} J_{\|}(c^{k+1},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k}) &= J_{\|}(c^{k+1},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k+1}) \\ & \quad J_{\|}(c^{k},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k}) \\ &\geq J_{\|}(c^{k+1},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k+1}). \end{aligned}$$

Convergence?

$$\begin{aligned} J_{\|}(c^{k+1},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k}) &= J_{\|}(c^{k+1},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k+1}) \\ & \quad J_{\|}(c^{k},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k}) \\ &\geq J_{\|}(c^{k+1},\Lambda^{k+1}) - J_{\|}(c^{k},\Lambda^{k+1}). \end{aligned}$$

$\rightarrow$Monotonicity preserved :
The proof of convergence is reduced to the one of the optimization solver.

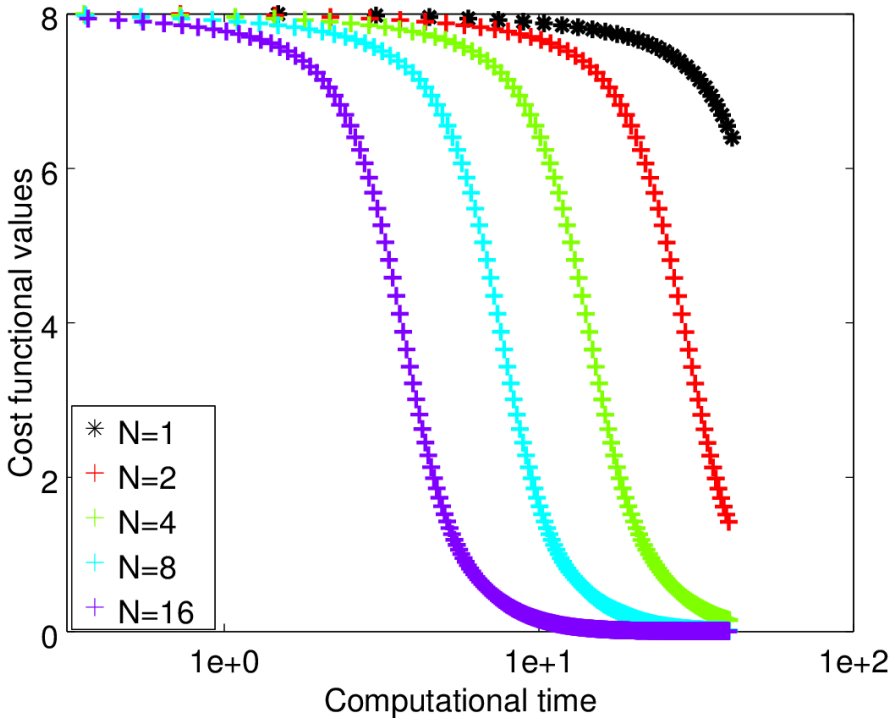
What about the optimization solver ?
We can show that:

$$\nabla J(c)_{|[T_j,T_{j+1}]} = \frac{T_{j+1}-T_j}{T} \nabla J_j(c_{|[T_j,T_{j+1}]}).$$

FOR EVERY c !

$\Rightarrow$ **A new interpretation of the method:**
the intermediate target method provides a **decomposition of the gradient** that enables parallelization.

Constant step gradient method



$\Rightarrow$ Full efficiency !

Newton

Here, our parallelization method not only improves the Newton convergence makes it possible .

N	$N \cdot Time_{\|\|}$
1	-
2	-
4	33.722
10	3.2544
20	0.72559

Full efficiency ?

The optimization is achieved in parallel, but $\psi(t)$ and $\chi(t)$ seem to require solving on $[0, T]$ (full propagation) ?

Full efficiency ?
The optimization is achieved in parallel, but $\psi(t)$ and $\chi(t)$ seem to require solving on $[0, T]$ (full propagation) ?

NO !!! $\rightarrow$ only $\psi(t_j)$ and $\chi(t_j)$ are required.

$\Rightarrow$ For low dimensional systems, the propagators $t_j \rightarrow t_{j+1}$ can be computed in parallel, when computing the gradient !

A nonlinear model: Gross-Pitaevskii equation

$$J(c) = \frac{1}{2}\|\psi_{target} - \psi(.,T)\|^2_{L^2}.$$

...to be minimized.

Constraint:

$$i\frac{\partial\psi(x,t)}{\partial t} = [-\frac{\hbar}{2m}\Delta + V(x,c(t)) + g|\psi(x,t)|^2]\psi(x,t),$$

with:

$$V(x,c) = \begin{cases} \frac{1}{2}\left(|x| - \frac{d.c}{2}\right)^2, & |x| > \frac{d.c}{4} \\ \frac{1}{2}\left(\frac{(d.c)^2}{8} - x^2\right), & otherwise. \end{cases}$$

U. Hohenester, P.K. Rekdal, A. Borzì, J. Schmiedmayer, Optimal quantum control of Bose-Einstein condensates in magnetic microtraps, Phys. Rev. A 75, 023602 (2007).

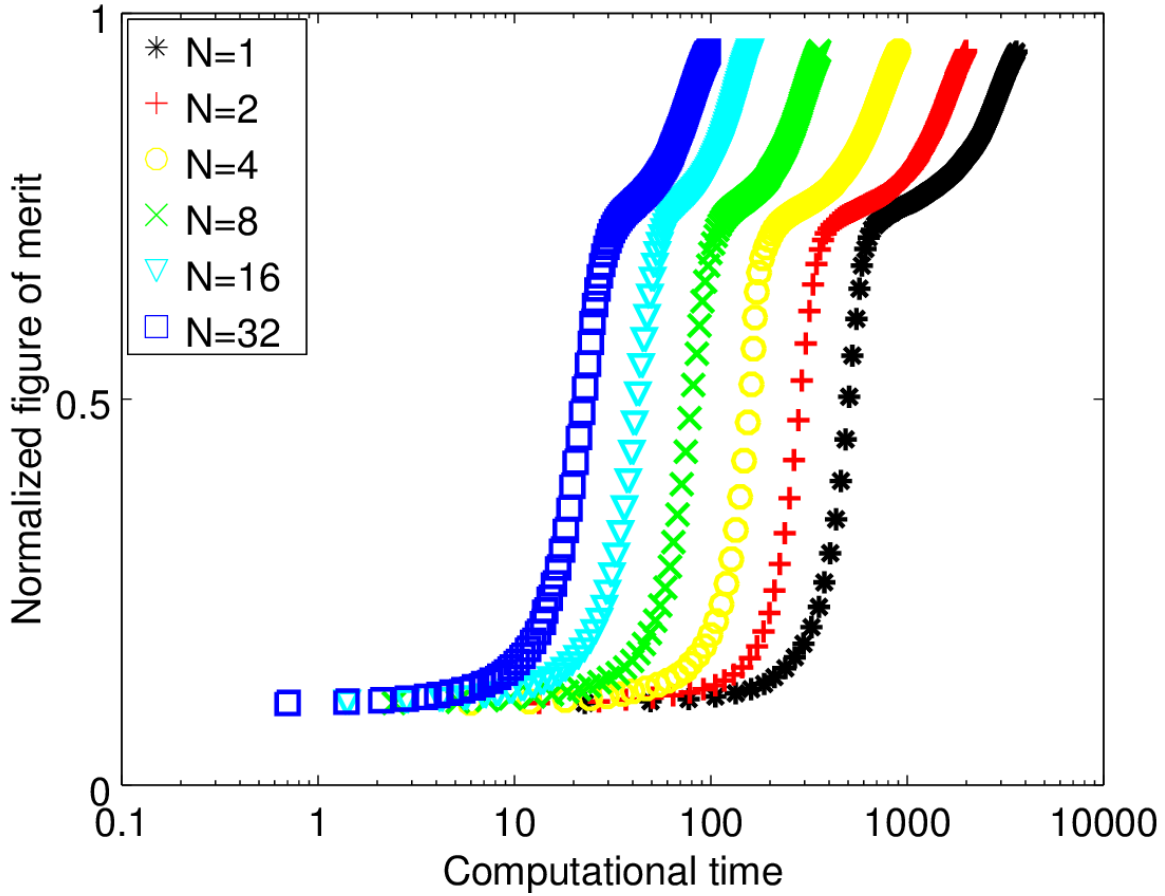
A nonlinear model: Adaptations required

- To keep the gradient property: splitting of initial states and targets.
- To approximate full-efficiency:

$$i\frac{\partial \psi^{k+1}(x,t)}{\partial t} = [-\frac{\hbar}{2m}\Delta + V(x,c(t)) + g|\psi^{k}(x,t)|^2]\psi^{k+1}(x,t).$$

$\Rightarrow$ Theoretical drawback: loss of the alternating optimization interpretation of the method.

A nonlinear model: numerical results



Outline

1 An intermediate states method to parallelize

2 Linear Control
- Time sub-intervals decomposition
- Use of a coarse solver
- Numerical examples

"Linear control"

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

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

The optimality condition then reads

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

$\rightarrow$ **Elimination of c:**

$$\begin{cases} \dot{y} = f(y) - \dfrac{\lambda}{\alpha}, \\ \\ \dot{\lambda} = -(f(y)')^T \lambda, \end{cases}$$

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

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

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

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

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

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

The optimality system is enriched:

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

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

Collecting the unknowns in the vector

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

we obtain the nonlinear system

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

Newton's method:

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

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

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

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

Third idea: coarse approximation of the Jacobian

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

Which concretely corresponds to:

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

$\rightarrow$ *Inspiration from the Parareal algorithm:*
J.-L. Lions, Y. Maday, and G. Turinici. A "parareal" in time disretization of pde's. Comptes Rendus de l'Acad. des Sciences, 2001.
$\rightarrow$ *and its interpretation:*
M. Gander, S. Vandewalle, SISC 2003.

Partial summary:

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

Example : linear dynamics

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

Discretizing and setting:

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

we get:

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

Example : linear dynamics

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

Discretizing and setting:

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

we get:

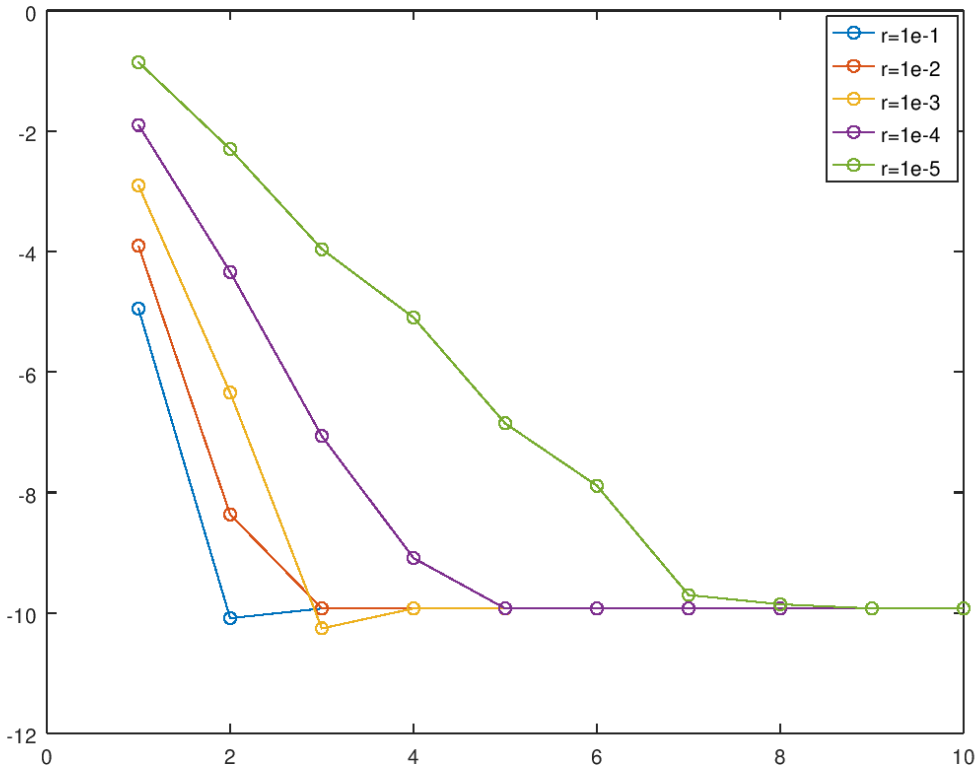
$$X^{k+1} = \left(Id - M_{\Delta t}^{-1} M_{\delta t}\right) X^k + M_{\Delta t}^{-1} b.$$

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

Results for implicit Euler:

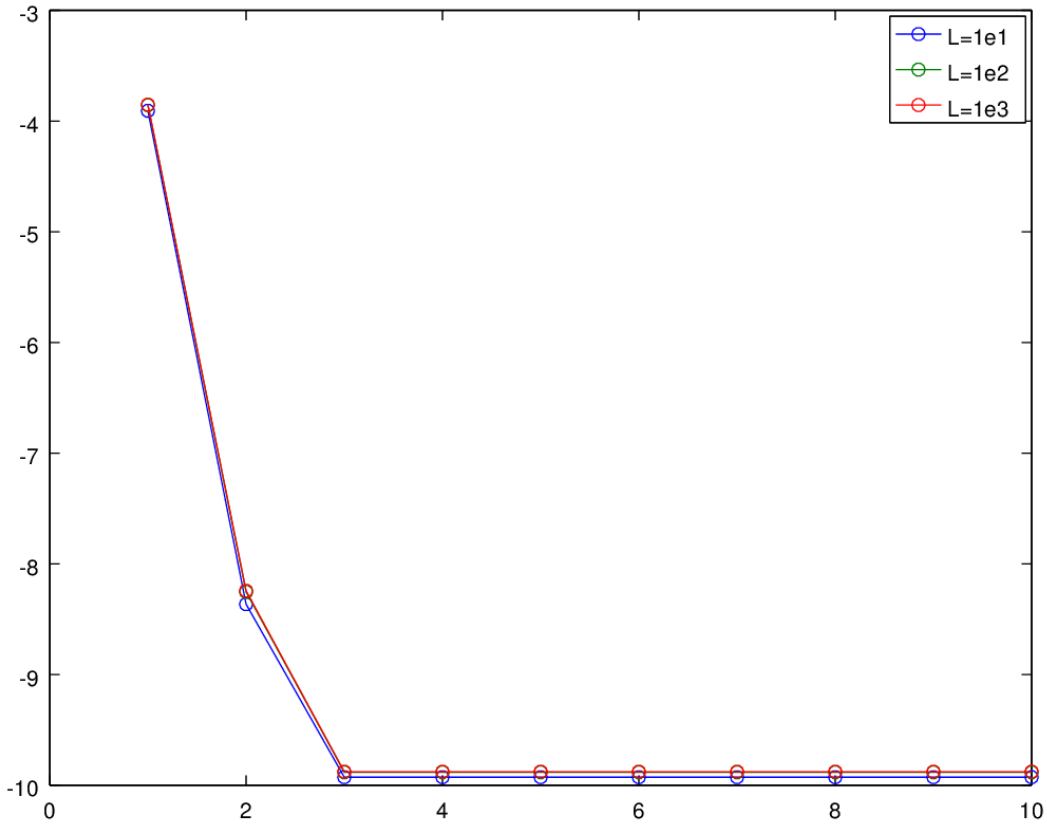
- Contraction factor: $\rho \leq C(\Delta t - \delta t)$
- For $\sigma < 0$, C can be chosen independent of σ
- For very large α, C can grow like $\log(\alpha)$ when the number of subdomains becomes large

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

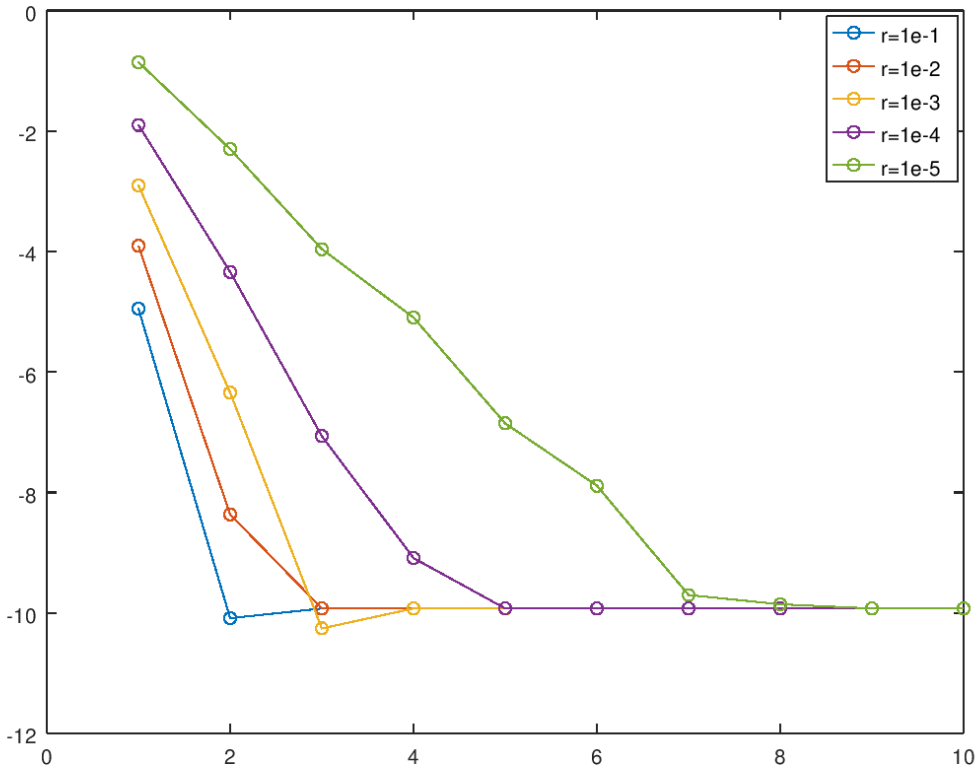


$$\dot{y}(t) = \sigma y(t) + c(t).$$

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

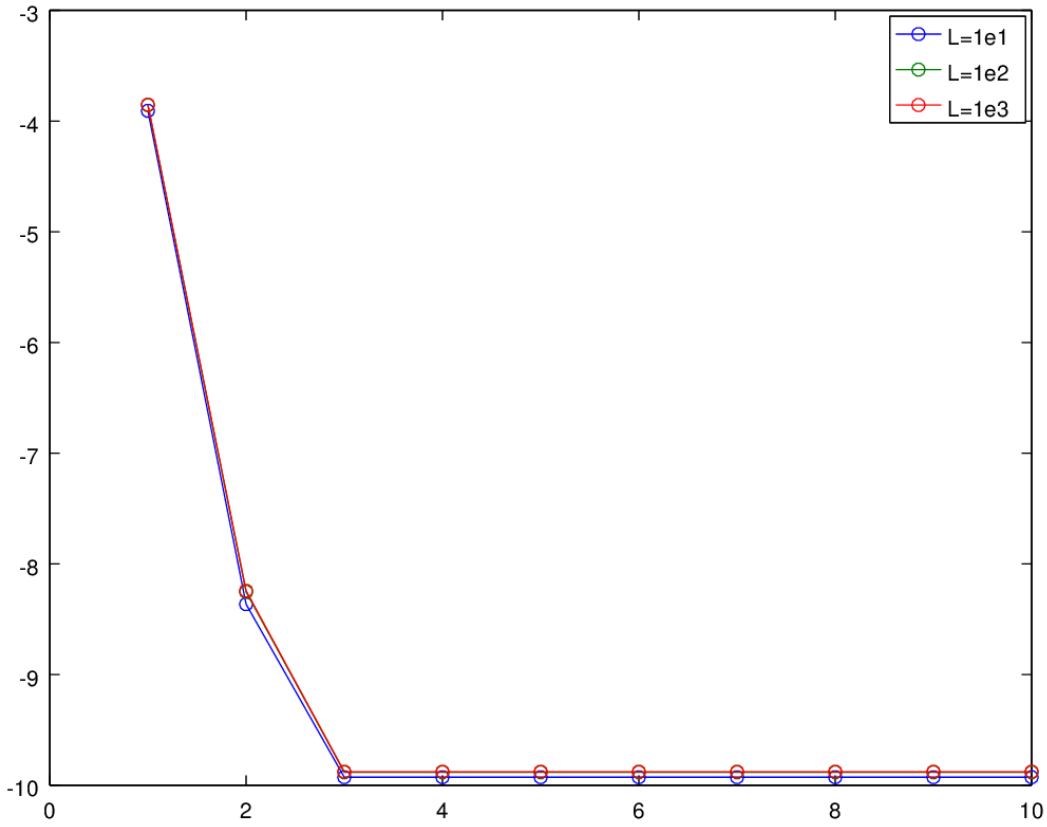


Convergence for various with respect to the number of iteration for various number of subintervals.



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

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



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

- Minimize

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

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

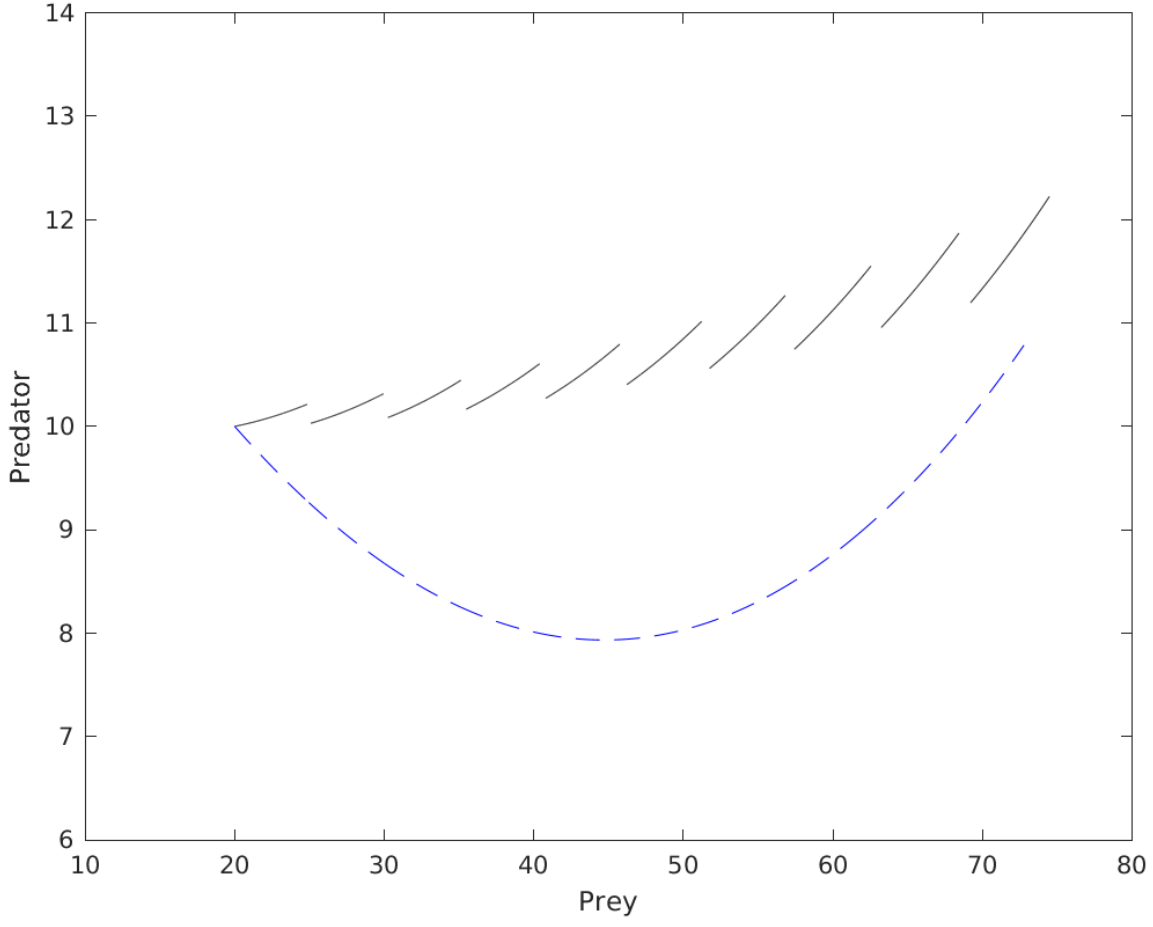
$$\dot{y}_1 = a_1 y_1 - b_1 y_1 y_2 + c_1, \quad \dot{y}_2 = a_2 y_1 y_2 - b_2 y_2 + c_2$$

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

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

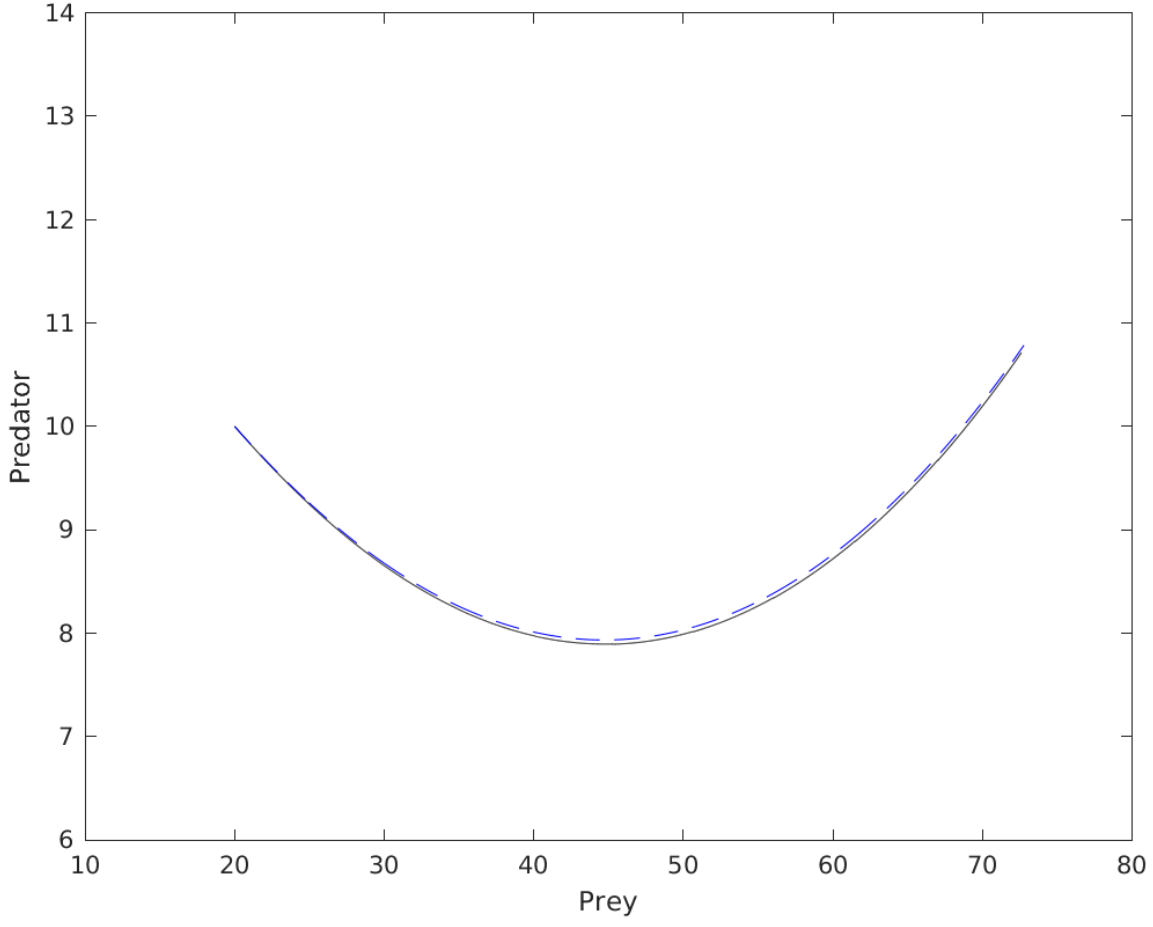
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #1



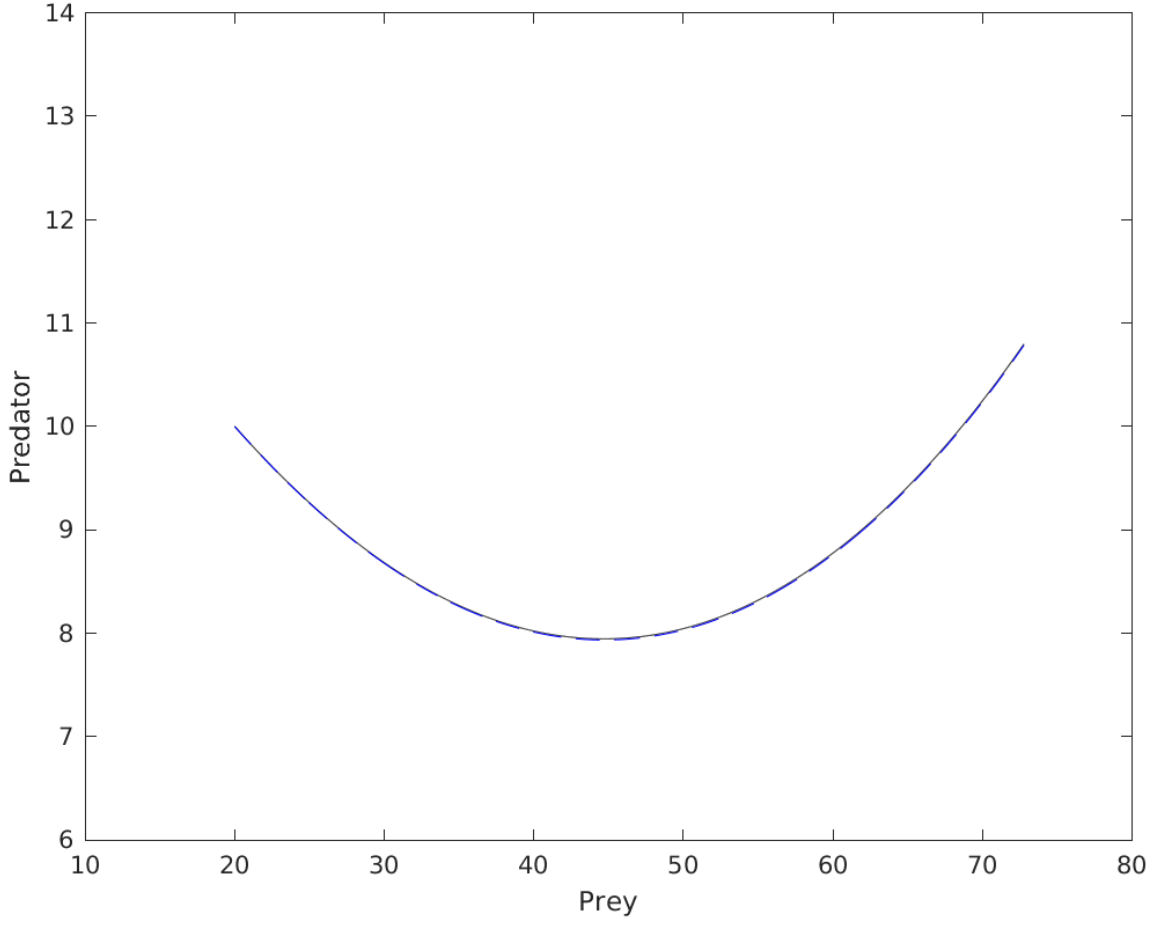
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #2



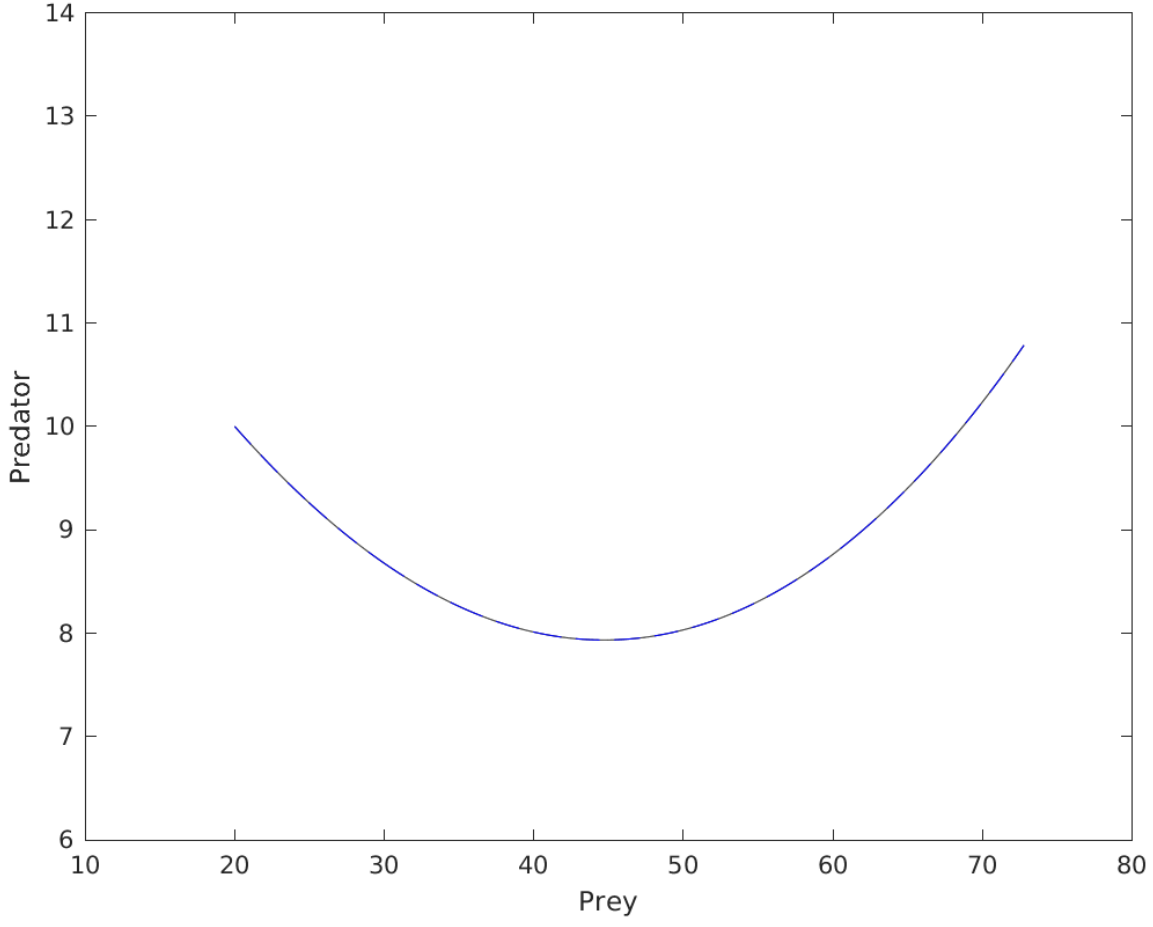
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #3



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

Iteration #4



- Minimize

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

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

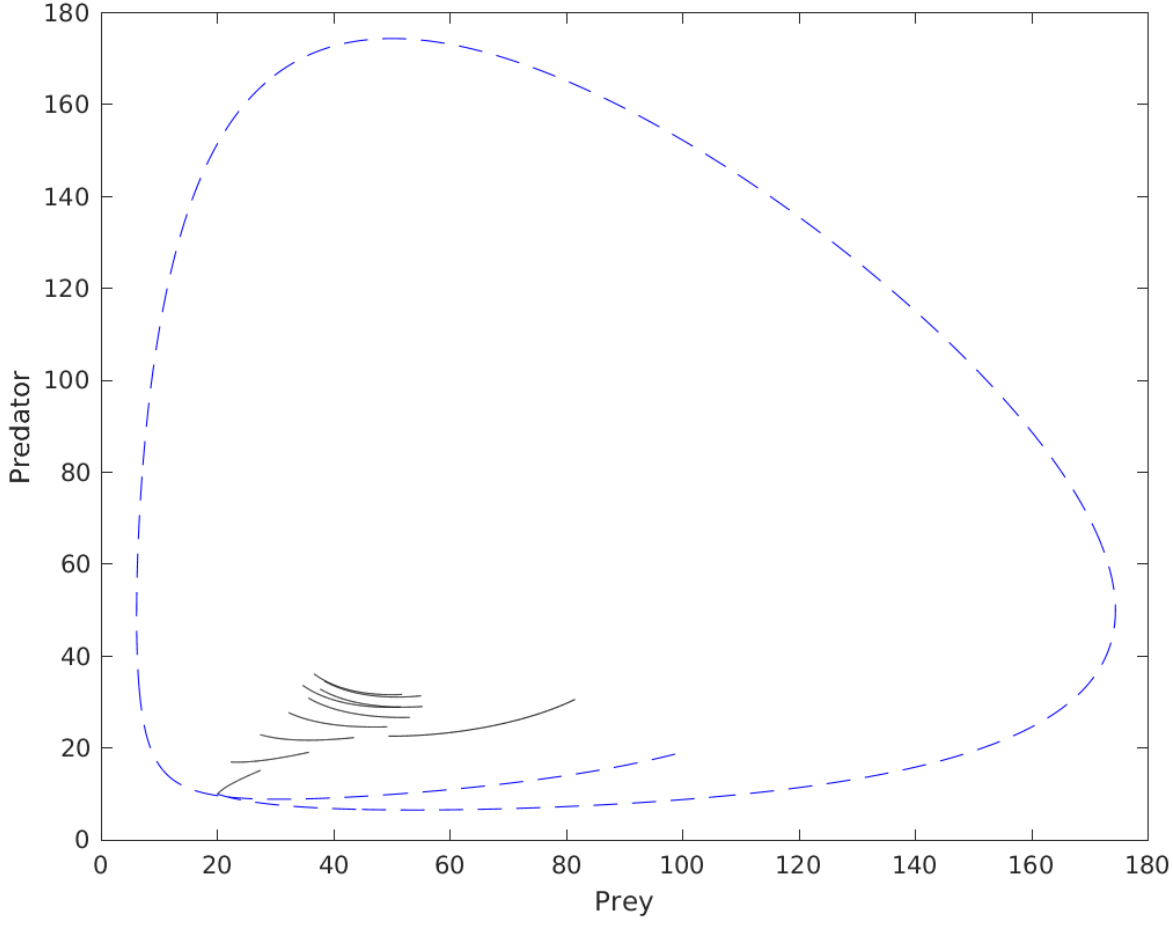
$$\dot{y}_1 = a_1 y_1 - b_1 y_1 y_2 + c_1, \quad \dot{y}_2 = a_2 y_1 y_2 - b_2 y_2 + c_2$$

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

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

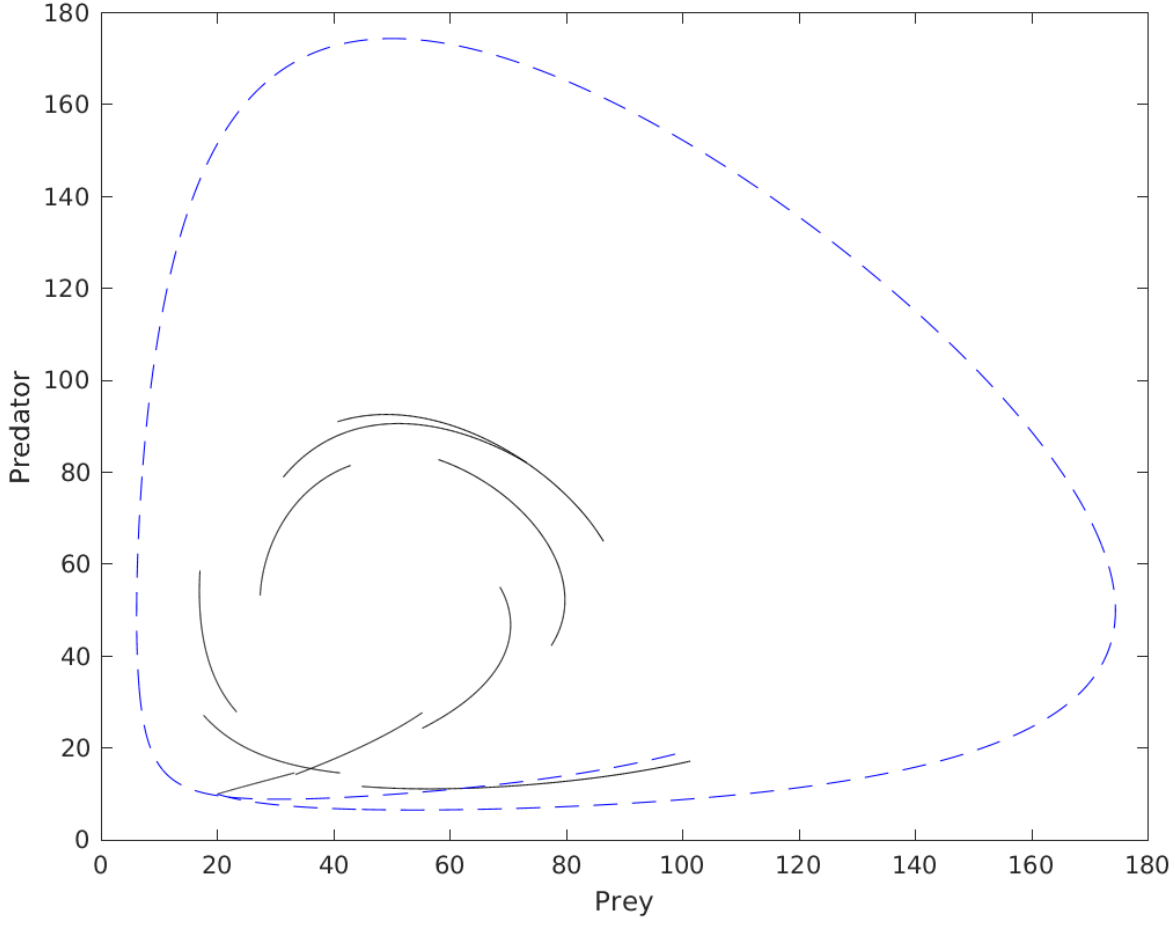
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #1



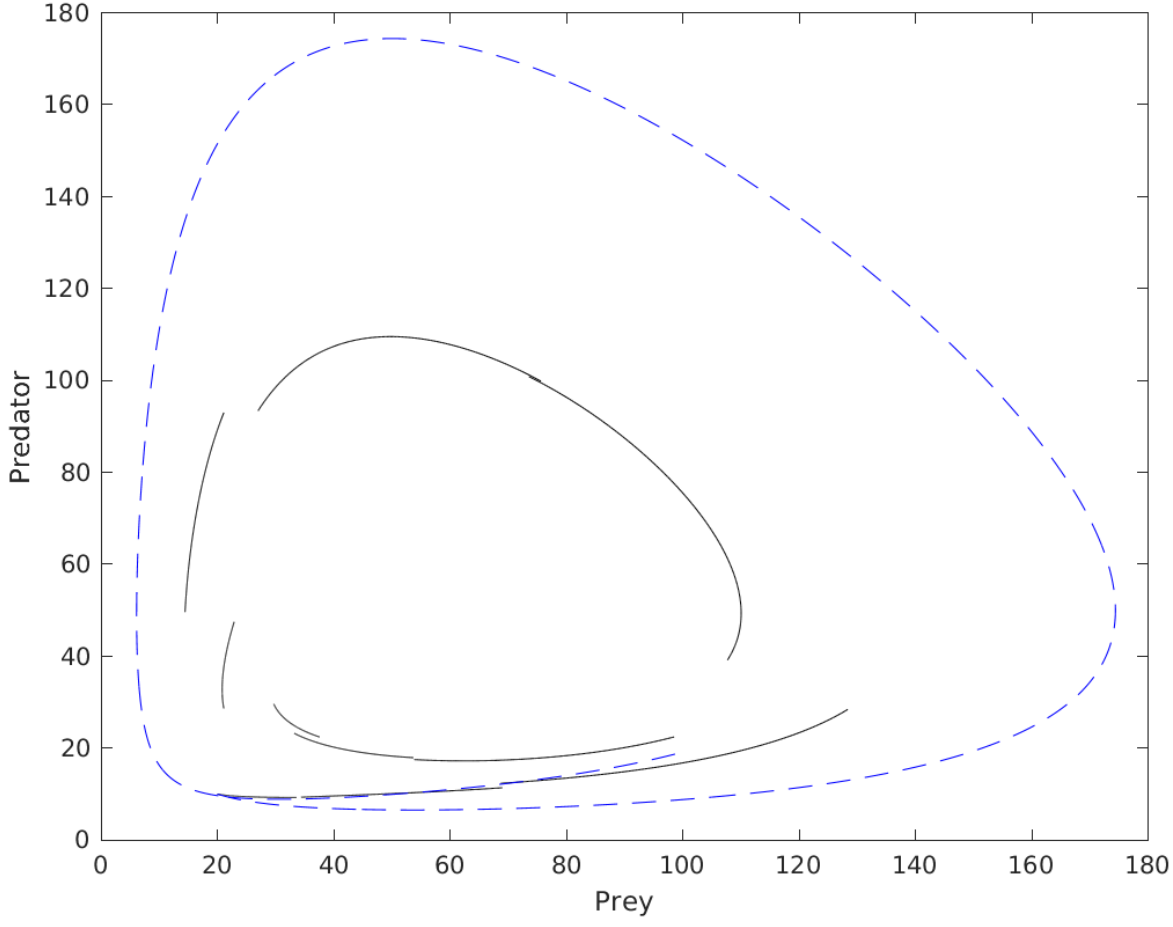
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #2



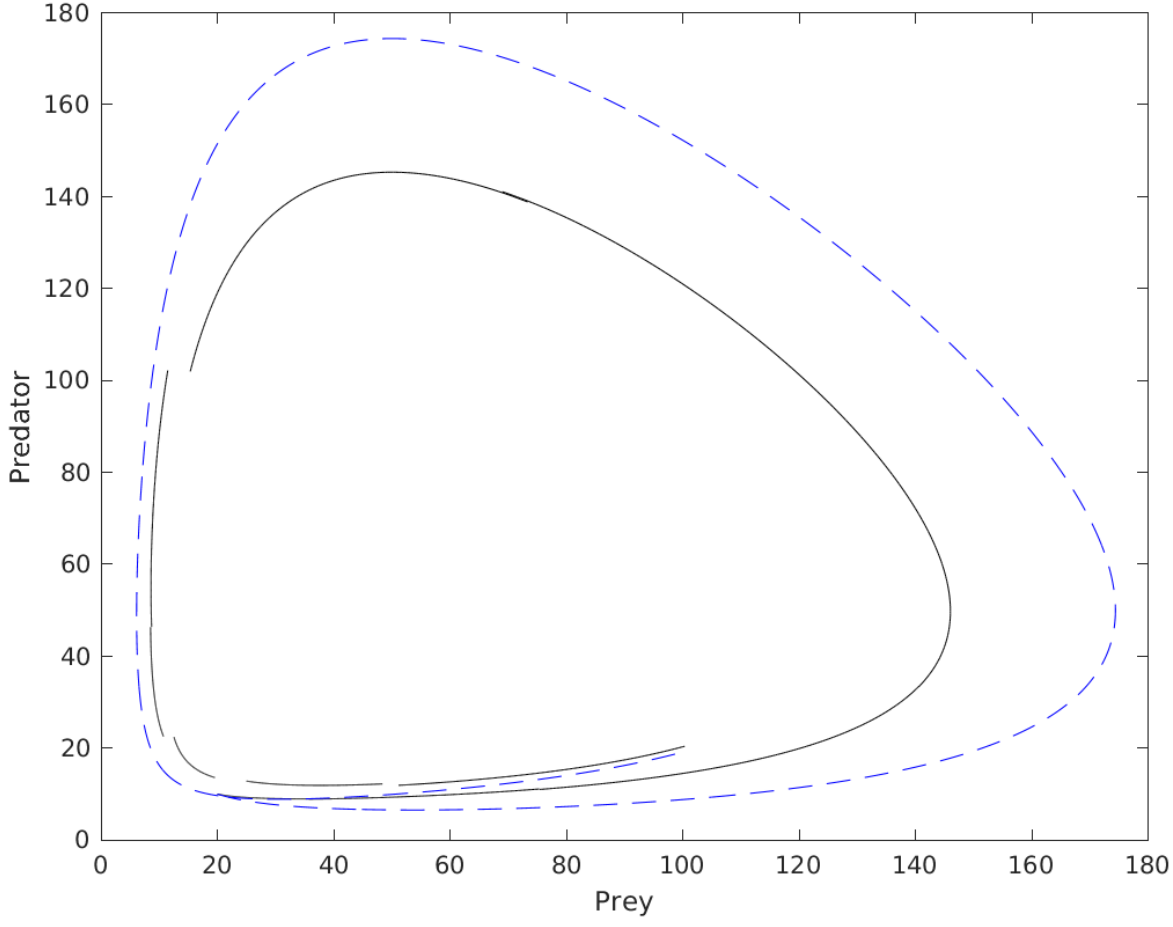
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #3



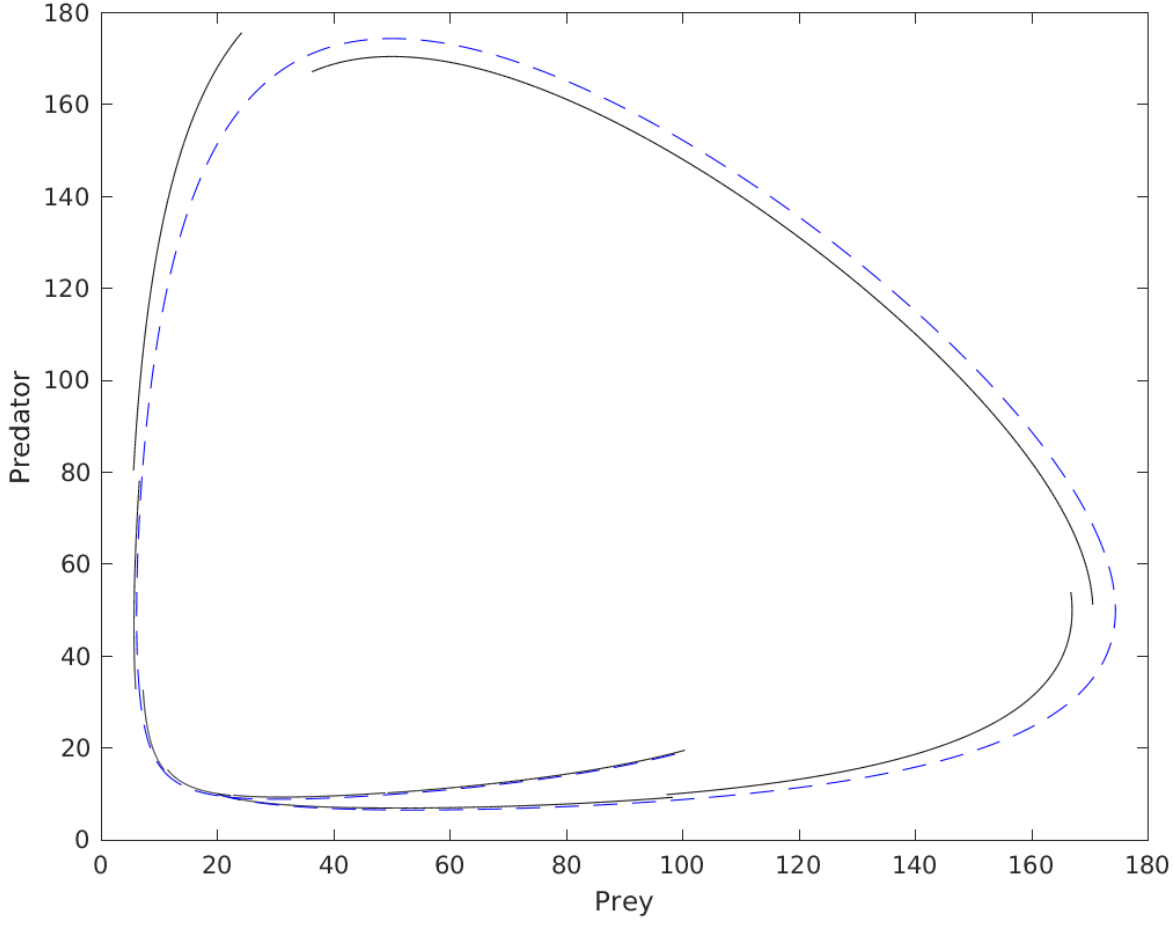
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #4



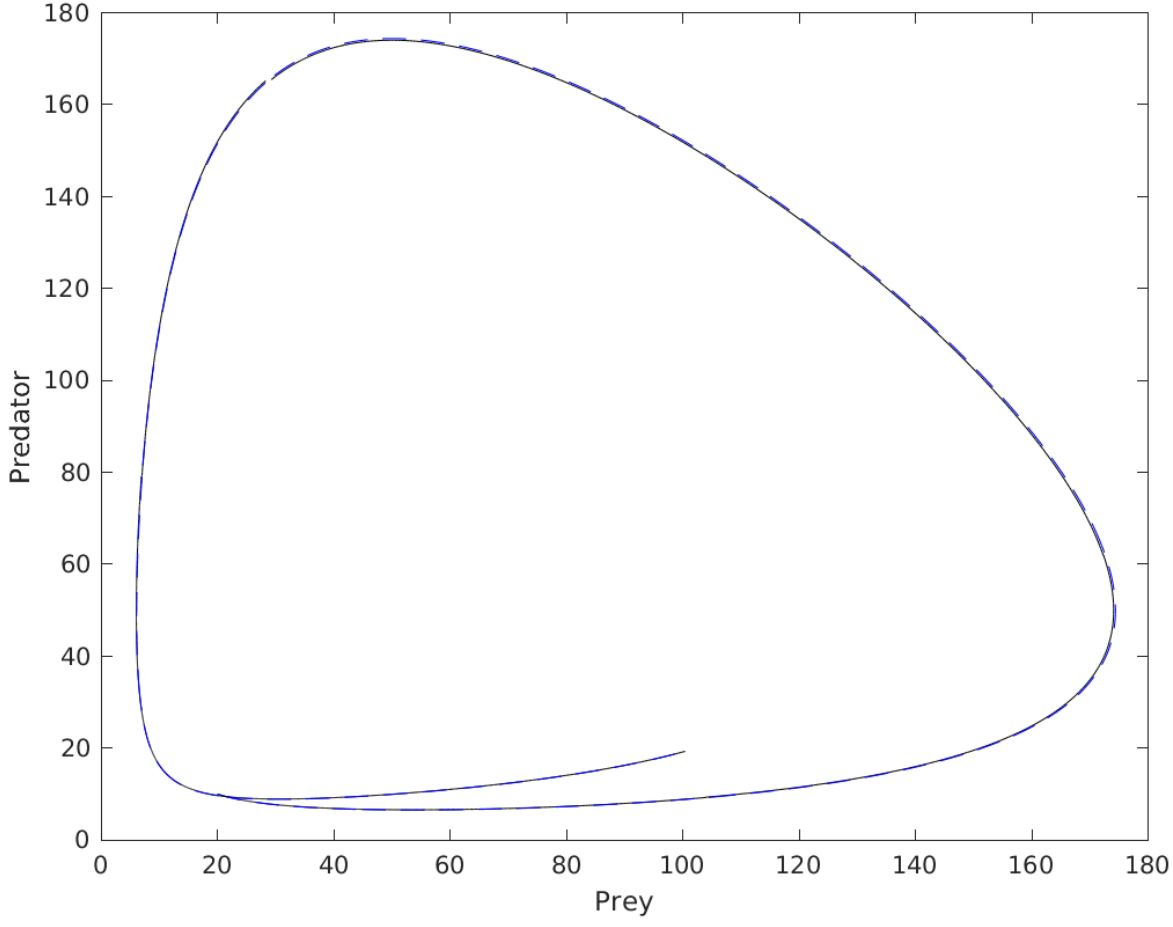
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #5



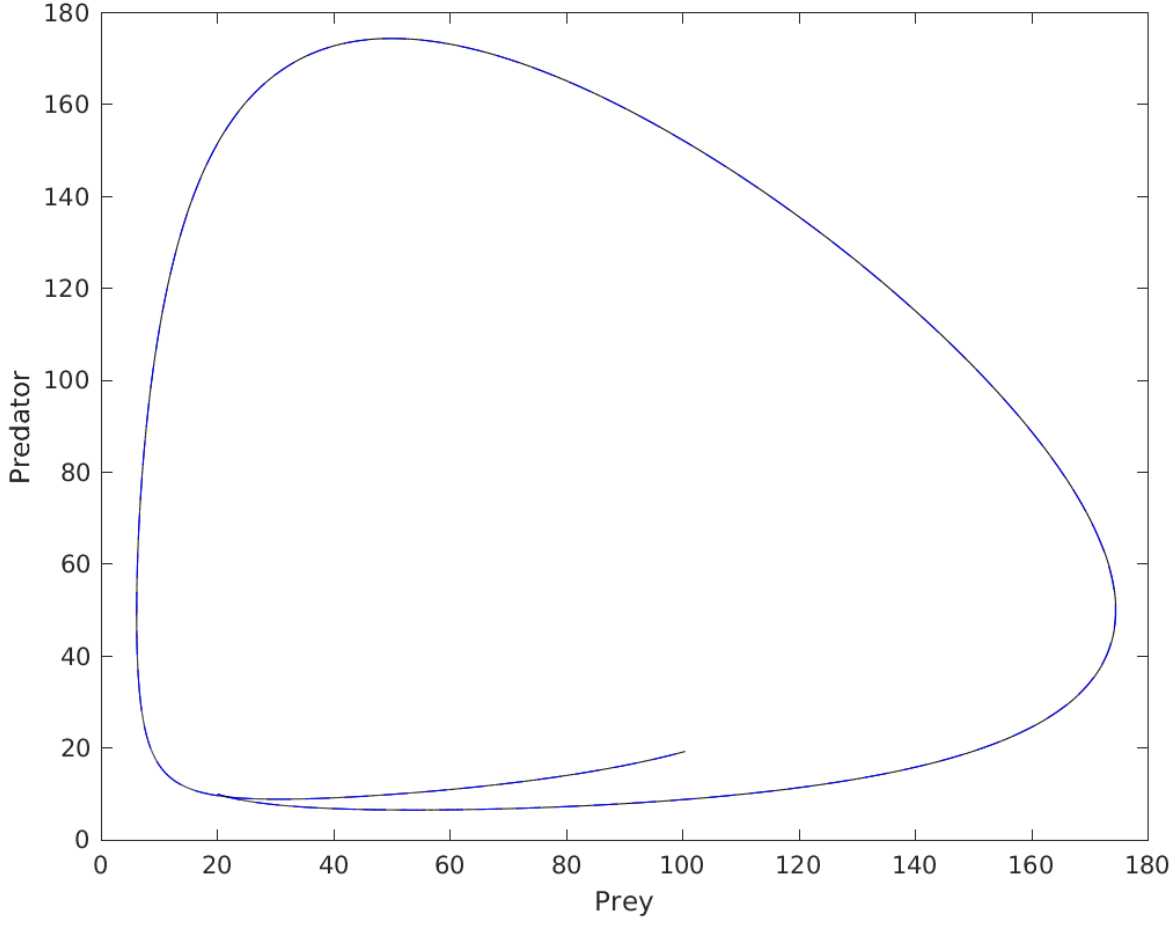
Vector example - $N = 10$, $r = \delta t / \Delta t = 0.01$

Iteration #6



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

Iteration #7



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

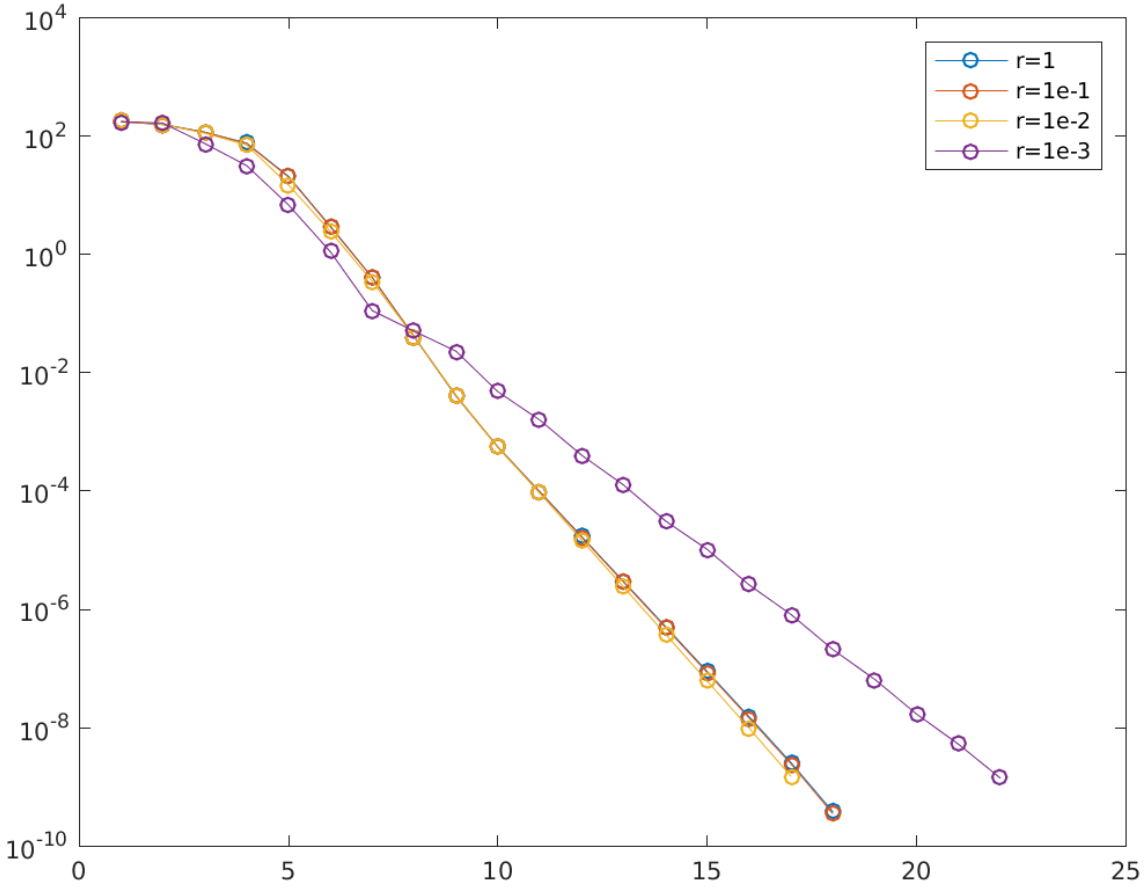
- Approximation: neglect 2nd derivatives

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

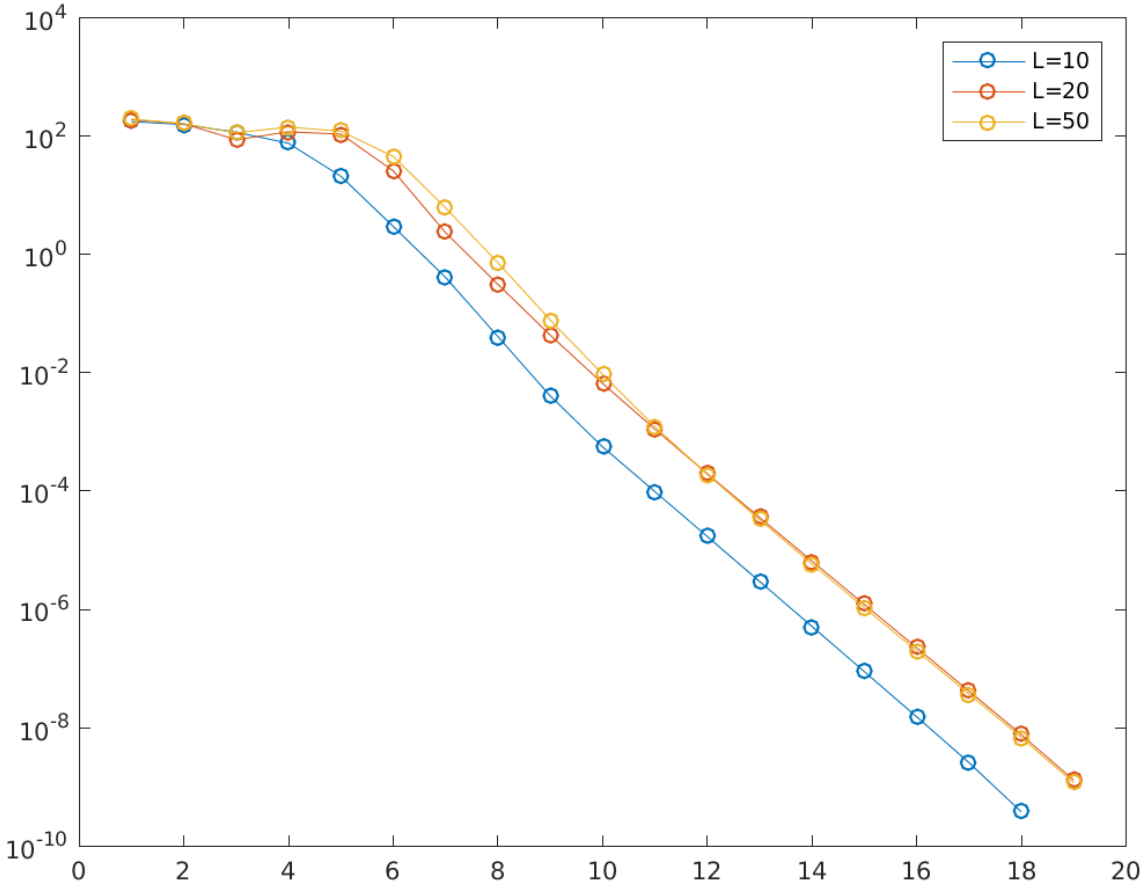
$$\frac{d\lambda'}{dt} = -(f'(y))^T\lambda' - \cancel{(f''(y,y'))^T\lambda}, \quad \lambda'(T) = \Lambda_{n+1}^{k+1} - \Lambda_{n+1}^k.$$

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

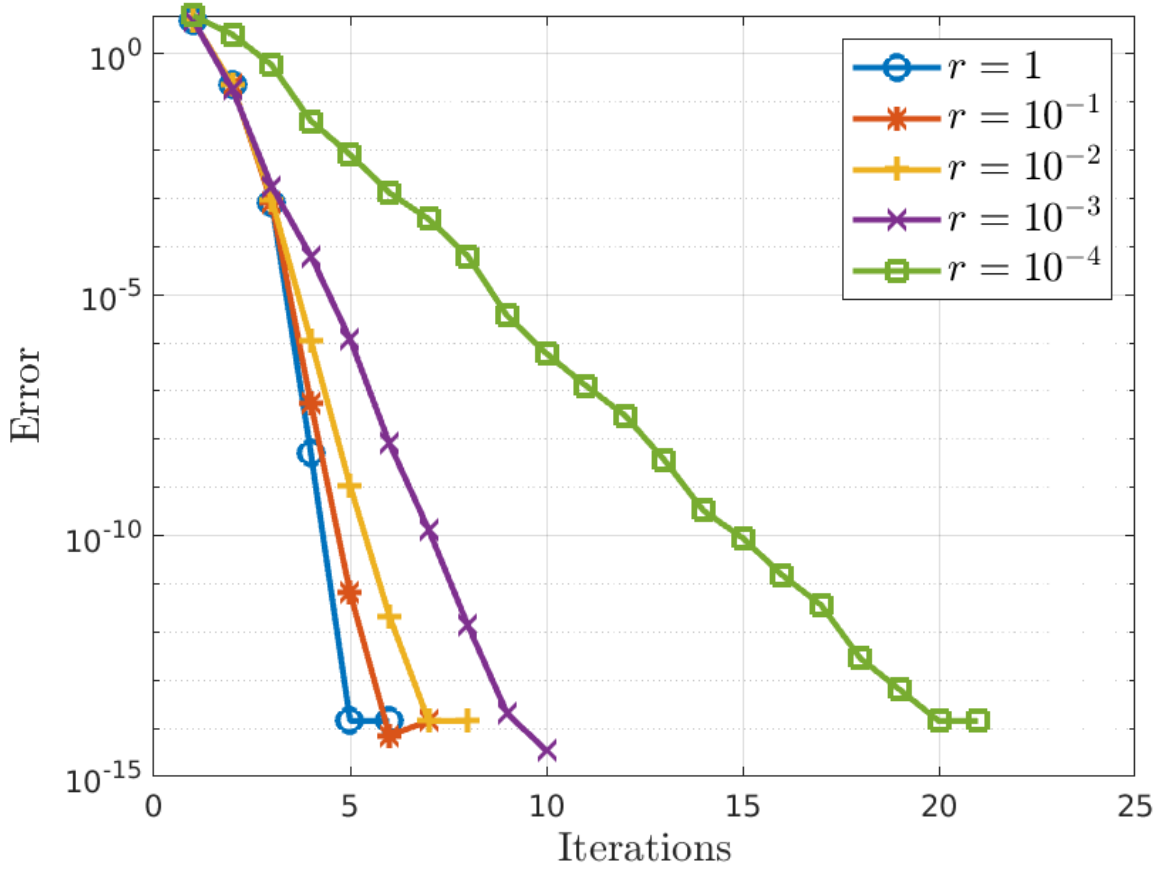
$N = 10$ subdomains, varying $r = \delta t / \Delta t$



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



True Newton:


10^{0}
10^{-5}
10^{-10}
10^{-15}
Error
0
5
10
15
20
25
Iterations
$r = 1$
$r = 10^{-1}$
$r = 10^{-2}$
$r = 10^{-3}$
$r = 10^{-4}$