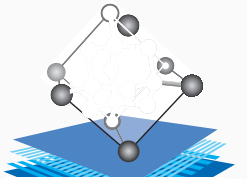


Adaptive parareal algorithms for molecular dynamics problems

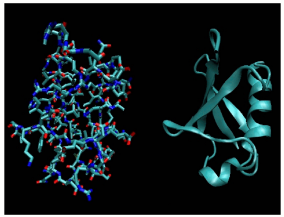
Olga Gorynina, Frédéric Legoll, Tony Lelièvre

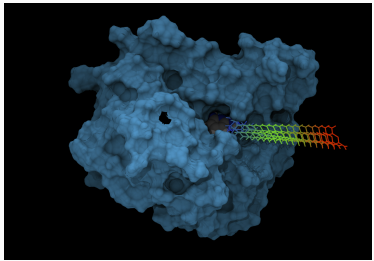
24/06/2021

SMAI 2021

Motivation

This work is motivated by molecular simulation, where we often have to simulate long trajectories of complex systems.





Typical dynamics: the Langevin equation

$$dq_t = p_t\,dt, \qquad dp_t = -\nabla V(q_t)\,dt - \gamma\,p_t\,dt + \sqrt{2\gamma\beta^{-1}}\,dW_t$$

Motivation

- Since we have to simulate long-time trajectories, it seems attractive to use the parareal algorithm, which solves initial value problems by parallel-in-time computations (domain-decomposition fashion)
- It turns out that this algorithm is not stable for MD problems when the time horizon is too large
- We therefore work with adaptive parareal algorithm, which performs simulations on shorter time slabs and paste them together, thereby allowing for a significant CPU gain (Legoll, Lelièvre and Sharma, HAL preprint 03189428, 2021)

Our goal is to apply the approach for realistic physical systems

Motivation

- Since we have to simulate long-time trajectories, it seems attractive to use the parareal algorithm, which solves initial value problems by parallel-in-time computations (domain-decomposition fashion)
- It turns out that this algorithm is not stable for MD problems when the time horizon is too large
- We therefore work with adaptive parareal algorithm, which performs simulations on shorter time slabs and paste them together, thereby allowing for a significant CPU gain (Legoll, Lelièvre and Sharma, HAL preprint 03189428, 2021)

Our goal is to apply the approach for realistic physical systems

Motivation

- Since we have to simulate long-time trajectories, it seems attractive to use the parareal algorithm, which solves initial value problems by parallel-in-time computations (domain-decomposition fashion)
- It turns out that this algorithm is not stable for MD problems when the time horizon is too large
- We therefore work with adaptive parareal algorithm, which performs simulations on shorter time slabs and paste them together, thereby allowing for a significant CPU gain (Legoll, Lelièvre and Sharma, HAL preprint 03189428, 2021)

Our goal is to apply the approach for realistic physical systems

Parallel in time algorithm for ODEs

$$\frac{dx}{dt} = f(x), \qquad x \in \mathbb{R}^d$$

The parareal algorithm (Lions, Maday and Turinici, 2001) is based upon two integrators to propagate the system over a time ΔT:

- a fine, accurate integrator $\mathcal{F}_{\Delta T}$
- a coarse, cheap integrator $\mathcal{C}_{\Delta T}$

For instance,

$$\mathcal{F}_{\Delta T} = (\Phi_{\delta t_F})^{\Delta T/\delta t_F} \quad \text{and} \quad \mathcal{C}_{\Delta T} = (\Phi_{\delta t_C})^{\Delta T/\delta t_C} \quad \text{with} \quad \delta t_F \ll \delta t_C$$

where $\Phi_{\delta t}$ is a one time step propagator

Parallel in time algorithm for ODEs

$$\frac{dx}{dt} = f(x), \qquad x \in \mathbb{R}^d$$

The parareal algorithm (Lions, Maday and Turinici, 2001) is based upon two integrators to propagate the system over a time ΔT:

- a fine, accurate integrator $\mathcal{F}_{\Delta T}$
- a coarse, cheap integrator $\mathcal{C}_{\Delta T}$

For instance,

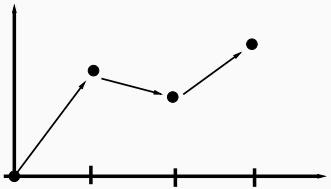
$$\mathcal{F}_{\Delta T} = (\Phi_{\delta t_F})^{\Delta T/\delta t_F} \quad \text{and} \quad \mathcal{C}_{\Delta T} = (\Phi_{\delta t_C})^{\Delta T/\delta t_C} \quad \text{with} \quad \delta t_F \ll \delta t_C$$

where $\Phi_{\delta t}$ is a one time step propagator

The parareal iterative procedure

- Initialization: coarse propagation that yields $\left\{x_n^{k=0}\right\}_n$:

$$\forall n, \quad x_{n+1}^{k=0} = \mathcal{C}_{\Delta T}(x_n^{k=0})$$

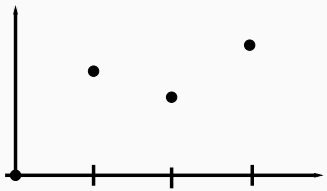


The parareal iterative procedure

- Initialization: coarse propagation that yields $\left\{x_n^{k=0}\right\}_n$:

$$\forall n, \quad x_{n+1}^{k=0} = \mathcal{C}_{\Delta T}(x_n^{k=0})$$

- Iterate over $k \geq 0$:

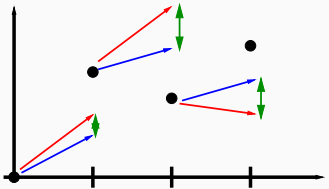


The parareal iterative procedure

- Initialization: coarse propagation that yields $\left\{x_n^{k=0}\right\}_n$:

$$\forall n, \quad x_{n+1}^{k=0} = \mathcal{C}_{\Delta T}(x_n^{k=0})$$

- Iterate over $k \geq 0$:
  - compute jumps (in parallel): $J_n^k = \mathcal{F}_{\Delta T}(x_n^k) - \mathcal{C}_{\Delta T}(x_n^k)$

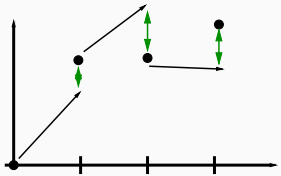


The parareal iterative procedure

- Initialization: coarse propagation that yields $\left\{x_n^{k=0}\right\}_n$:

$$\forall n, \quad x_{n+1}^{k=0} = \mathcal{C}_{\Delta T}(x_n^{k=0})$$

- Iterate over $k \geq 0$:
  - compute jumps (in parallel): $J_n^k = \mathcal{F}_{\Delta T}(x_n^k) - \mathcal{C}_{\Delta T}(x_n^k)$
  - sequential update to obtain $\left\{x_n^{k+1}\right\}_n$: $\forall n, \quad x_{n+1}^{k+1} = \mathcal{C}_{\Delta T}(x_n^{k+1}) + J_n^k$



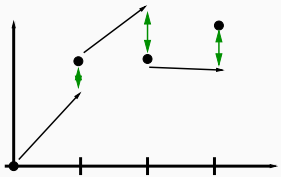
! The fine solver is called only in the parallel part of the algorithm !

The parareal iterative procedure

- Initialization: coarse propagation that yields $\left\{x_n^{k=0}\right\}_n$:

$$\forall n, \quad x_{n+1}^{k=0} = \mathcal{C}_{\Delta T}(x_n^{k=0})$$

- Iterate over $k \geq 0$:
  - compute jumps (in parallel): $J_n^k = \mathcal{F}_{\Delta T}(x_n^k) - \mathcal{C}_{\Delta T}(x_n^k)$
  - sequential update to obtain $\left\{x_n^{k+1}\right\}_n$: $\forall n, \quad x_{n+1}^{k+1} = \mathcal{C}_{\Delta T}(x_n^{k+1}) + J_n^k$



! The fine solver is called only in the parallel part of the algorithm !

Parareal algorithm for MD simulations

- Fundamental property : $x_n^k = \mathcal{F}_{\Delta T}^n(x_0)$ for any $n \leq k$. In practice, convergence is observed in many cases over long times in a few iterations.
- In MD, we often run simulations with time steps chosen just below the stability limit (this often provides sufficient accuracy on the quantities of interest). There is hence no room for choosing $\delta t_C \gg \delta t_F$

- We thus turn to a different paradigm where $\mathcal{C}_{\Delta T}$ integrates a simpler dynamics than $\mathcal{F}_{\Delta T}$ (say with the same time step):
  - $\mathcal{F}_{\Delta T}$ integrates the original Langevin dynamics (with the reference potential V_f)
  - $\mathcal{C}_{\Delta T}$ integrates a Langevin dynamics run on a simplified, cheaper to compute potential V_c.

Parareal algorithm for MD simulations

- Fundamental property : $x_n^k = \mathcal{F}_{\Delta T}^n(x_0)$ for any $n \leq k$. In practice, convergence is observed in many cases over long times in a few iterations.
- In MD, we often run simulations with time steps chosen just below the stability limit (this often provides sufficient accuracy on the quantities of interest). There is hence no room for choosing $\delta t_C \gg \delta t_F$

- We thus turn to a different paradigm where $\mathcal{C}_{\Delta T}$ integrates a simpler dynamics than $\mathcal{F}_{\Delta T}$ (say with the same time step):
  - $\mathcal{F}_{\Delta T}$ integrates the original Langevin dynamics (with the reference potential V_f)
  - $\mathcal{C}_{\Delta T}$ integrates a Langevin dynamics run on a simplified, cheaper to compute potential V_c.

Convergence criteria

- Relative error between consecutive parareal trajectories:

$$E(k,N) = \frac{\sum_{n=1}^{N} |q_n^k - q_n^{k-1}|}{\sum_{n=1}^{N} |q_n^{k-1}|}.$$

- We stop the algorithm at the first parareal iteration $\overline{k}$ for which

$$E(\overline{k},N) < \delta_{\text{conv}}$$

- Theoretical gain $\Gamma = \frac{N}{\overline{k}} = \frac{\text{\# fine propagations for a sequential algorithm}}{\text{\# fine propagations for the parareal algorithm}}$

Convergence criteria

- Relative error between consecutive parareal trajectories:

$$E(k,N) = \frac{\sum_{n=1}^{N} |q_n^k - q_n^{k-1}|}{\sum_{n=1}^{N} |q_n^{k-1}|}.$$

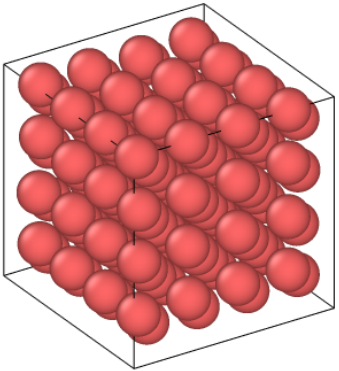
- We stop the algorithm at the first parareal iteration $\overline{k}$ for which

$$E(\overline{k},N) < \delta_{\text{conv}}$$

- Theoretical gain $\Gamma = \frac{N}{\overline{k}} = \frac{\text{\# fine propagations for a sequential algorithm}}{\text{\# fine propagations for the parareal algorithm}}$

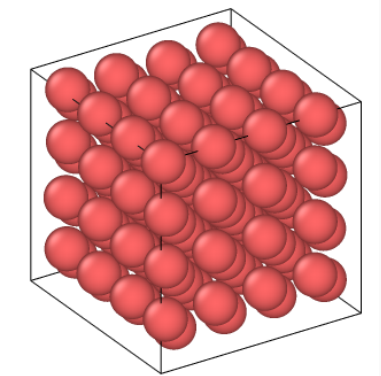
Settings

- 128 tungsten atoms
- BCC lattice + periodic boundary conditions
- Langevin dynamics
- $\beta^{-1} = T = 300$ Kelvin (+ we have to revisit the time integrator)
- $V_{\mathcal{F}}$ - fine machine-learning interatomic potentials SNAP-56 (spectral neighbor analysis potentials): Wood, Cusentino, Wirth, Thompson, Physical Review B (2019)
- $V_{\mathcal{C}}$ - coarse SNAP-15 potential



Settings

- 128 tungsten atoms
- BCC lattice + periodic boundary conditions
- Langevin dynamics
- $\beta^{-1} = T = 300$ Kelvin (+ we have to revisit the time integrator)
- $V_{\mathcal{F}}$ - fine machine-learning interatomic potentials SNAP-56 (spectral neighbor analysis potentials): Wood, Cusentino, Wirth, Thompson, Physical Review B (2019)
- $V_{\mathcal{C}}$ - coarse SNAP-15 potential



LAMMPS

We use LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) - a popular molecular dynamics software with a focus on materials modeling.

The parareal algorithm is written in Python and we use the following three LAMMPS include files

- **in.snap.WBe.simulation.box** - defines units, MD parameters, initial configuration of the atoms and simulation cell
- **in.snap.WBe.fine** - defines fine potential
- **in.snap.WBe.coarse** - defines coarse potential

It is important to provide the same white noise for both potentials

LAMMPS

We use LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) - a popular molecular dynamics software with a focus on materials modeling.

The parareal algorithm is written in Python and we use the following three LAMMPS include files

- **in.snap.WBe.simulation.box** - defines units, MD parameters, initial configuration of the atoms and simulation cell
- **in.snap.WBe.fine** - defines fine potential
- **in.snap.WBe.coarse** - defines coarse potential

It is important to provide the same white noise for both potentials

LAMMPS

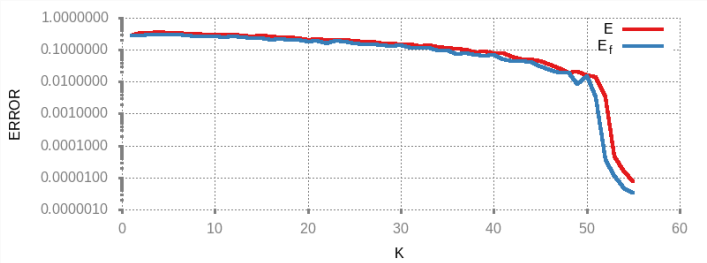
We use LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) - a popular molecular dynamics software with a focus on materials modeling.

The parareal algorithm is written in Python and we use the following three LAMMPS include files

- **in.snap.WBe.simulation.box** - defines units, MD parameters, initial configuration of the atoms and simulation cell
- **in.snap.WBe.fine** - defines fine potential
- **in.snap.WBe.coarse** - defines coarse potential

It is important to provide the same white noise for both potentials

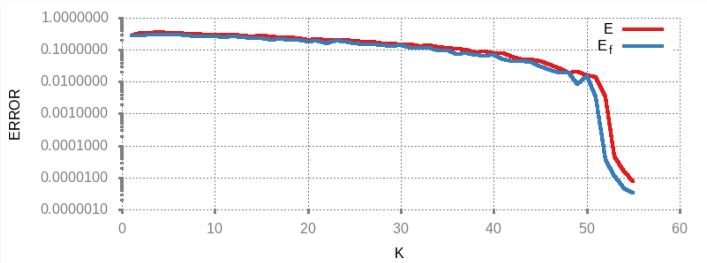
Plot of consecutive error for $N = 297$, $\Delta T = 0.005$



The gain $\Gamma = 5.4$. But for $N \geq 298$ it is not possible to work on the complete interval $[0, N]$ since, along the parareal iterations, the trajectory blows up

Remedy? Adaptive algorithm

Plot of consecutive error for $N = 297$, $\Delta T = 0.005$



The gain $\Gamma = 5.4$. But for $N \geq 298$ it is not possible to work on the complete interval $[0, N]$ since, along the parareal iterations, the trajectory blows up

Remedy? Adaptive algorithm

Adaptive algorithm

- On the time-slab $[0, N]$, we run the parareal algorithm until E is
  - either smaller than the convergence threshold δ_{conv}
  - or larger than an explosion threshold δ_{expl} (attained at parareal iteration # k_{cur})
- In the blow-up case, for the iteration k_{cur}, we find the first time iteration $1 + \widetilde{m}_1 \leq N$ for which E exceeds δ_{expl}, and we shorten the slab to $[0, \widetilde{m}_1]$.
- We then proceed with the parareal on the slab $[0, \widetilde{m}_1]$, that we possibly further shorten, until the relative error (on $[0, \widetilde{m}_1]$) is smaller than δ_{conv}.
- Once we have converged on $[0, \widetilde{m}_1]$ (δ_{conv}), we proceed with the next part of the time range and define the new (tentative) time-slab as $[\widetilde{m}_1, N]$.

Adaptive algorithm

- On the time-slab $[0, N]$, we run the parareal algorithm until E is
  - either smaller than the convergence threshold δ_{conv}
  - or larger than an explosion threshold δ_{expl} (attained at parareal iteration # k_{cur})

- In the blow-up case, for the iteration k_{cur}, we find the first time iteration $1 + \widetilde{m}_1 \leq N$ for which E exceeds δ_{expl}, and we shorten the slab to $[0, \widetilde{m}_1]$.

- We then proceed with the parareal on the slab $[0, \widetilde{m}_1]$, that we possibly further shorten, until the relative error (on $[0, \widetilde{m}_1]$) is smaller than δ_{conv}.

- Once we have converged on $[0, \widetilde{m}_1]$ (δ_{conv}), we proceed with the next part of the time range and define the new (tentative) time-slab as $[\widetilde{m}_1, N]$.

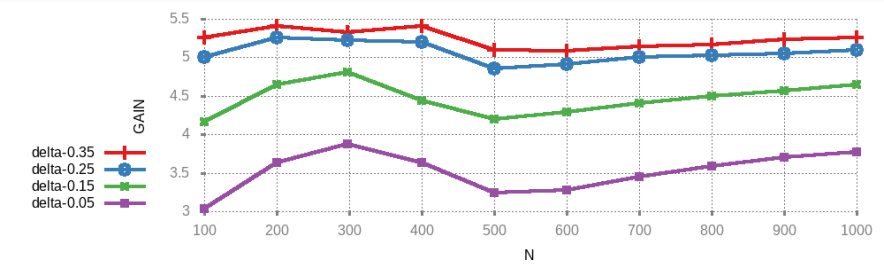
Adaptive algorithm

- On the time-slab $[0, N]$, we run the parareal algorithm until E is
  - either smaller than the convergence threshold δ_{conv}
  - or larger than an explosion threshold δ_{expl} (attained at parareal iteration # k_{cur})
- In the blow-up case, for the iteration k_{cur}, we find the first time iteration $1 + \widetilde{m}_1 \leq N$ for which E exceeds δ_{expl}, and we shorten the slab to $[0, \widetilde{m}_1]$.
- We then proceed with the parareal on the slab $[0, \widetilde{m}_1]$, that we possibly further shorten, until the relative error (on $[0, \widetilde{m}_1]$) is smaller than δ_{conv}.
- Once we have converged on $[0, \widetilde{m}_1]$ (δ_{conv}), we proceed with the next part of the time range and define the new (tentative) time-slab as $[\widetilde{m}_1, N]$.

Adaptive algorithm

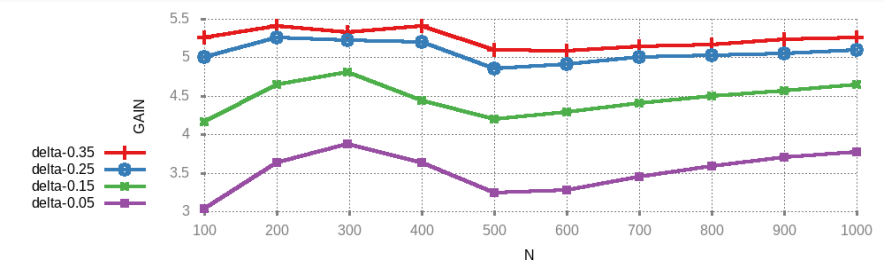
- On the time-slab $[0, N]$, we run the parareal algorithm until E is
  - either smaller than the convergence threshold δ_{conv}
  - or larger than an explosion threshold δ_{expl} (attained at parareal iteration # k_{cur})
- In the blow-up case, for the iteration k_{cur}, we find the first time iteration $1 + \widetilde{m}_1 \leq N$ for which E exceeds δ_{expl}, and we shorten the slab to $[0, \widetilde{m}_1]$.
- We then proceed with the parareal on the slab $[0, \widetilde{m}_1]$, that we possibly further shorten, until the relative error (on $[0, \widetilde{m}_1]$) is smaller than δ_{conv}.
- Once we have converged on $[0, \widetilde{m}_1]$ (δ_{conv}), we proceed with the next part of the time range and define the new (tentative) time-slab as $[\widetilde{m}_1, N]$.

Gain for adaptive parareal



- We remind that the classical parareal algorithm blows up when $N \geq 298$.
- For $N > 500$, the gain seems to be almost independent of N

Gain for adaptive parareal



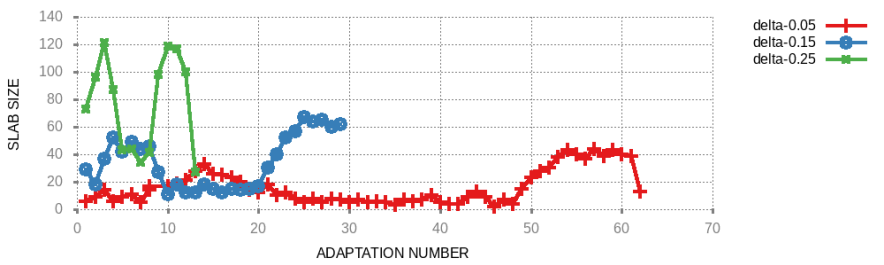
- We remind that the classical parareal algorithm blows up when $N \geq 298$.
- For $N > 500$, the gain seems to be almost independent of N

Explosion threshold

The slab sizes are such that $E \leq \delta_{\text{expl}}$:

- if δ_{expl} is chosen large, there is no adaptation: classical parareal
- if δ_{expl} is chosen small, the slabs are short: no parallelism anymore
- the optimal choice of δ_{expl} is somewhere in-between

List of the sizes of the time-slabs found by the algorithm for $N = 1000$:



Perspectives (all in progress)

Investigate the feasibility of the adaptive algorithm for realistic problems in the high performance computing context:

- Efficiency of the algorithm for computation of material properties:
  - elastic constants
  - dynamical quantities (diffusion coefficients of vacancies)
- Further comparison of the adaptive algorithm and the classical parareal
- Computational gain using different couples of SNAP potentials