

# Parallel in Time Algorithms for Multiscale Dynamical Systems using Interpolation and Neural Networks

Gopal Yalla  
Björn Engquist

University of Texas at Austin  
Institute for Computational Engineering and Sciences

April 17, 2018

# Outline

- 1 Parallel in Time Algorithms
- 2 Parareal Algorithm
- 3 Phase-accurate Coarse Solver
- 4 Numerical Examples
- 5 Future Work

# Parallel in Time Algorithms<sup>1</sup>

- Required for **massively parallel** simulations of systems governed by time-dependent dynamical systems, e.g., molecular dynamics.
- Challenge arises due to **causality** in time.

---

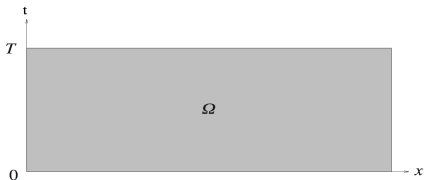
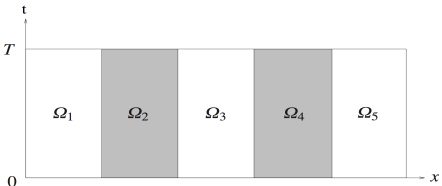
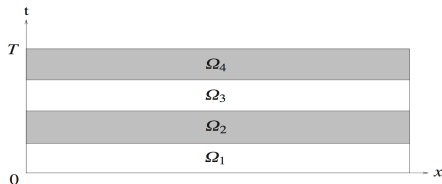
<sup>1</sup>Martin J Gander. “50 years of time parallel time integration”. In: *Multiple Shooting and Time Domain Decomposition Methods*. Springer, 2015, pp. 69–113.

# Parallel in Time Algorithms<sup>1</sup>

1 Multiple Shooting Methods

2 Domain Decomposition & Waveform Relaxation Methods

3 Multigrid Based Methods



<sup>1</sup>Martin J Gander. “50 years of time parallel time integration”. In: *Multiple Shooting and Time Domain Decomposition Methods*. Springer, 2015, pp. 69–113.

# The Parareal Algorithm

Consider the **dynamical system**:

$$\begin{aligned}u'(t) &= f(u(t)), & t \in (t_0, t_f) \\ u(t_0) &= u_0\end{aligned}\tag{1}$$

Divide the time domain  $(t_0, t_f)$  into  $N$  **equal subdomains**  $\Omega_n = (T_n, T_{n+1})$ .

$$\begin{aligned}u'_n(t) &= f(u_n(t)) & t \in (T_n, T_{n+1}) \\ u(T_n) &= U_n\end{aligned}\tag{2}$$

This is simply a **shooting method** in time, and it equivalent to solving:

$$\begin{pmatrix} U_0 - u_0 \\ U_1 - \phi_{\Delta T_0}(U_0) \\ \vdots \\ U_{N-1} - \phi_{\Delta T_{N-2}}(U_{N-2}) \end{pmatrix} = 0\tag{3}$$

where  $\phi_{\Delta T_n}(U_n)$  is solution of (1) with initial condition  $U_n$  after time  $\Delta T_n$ .

# The Parareal Algorithm

Consider the **dynamical system**:

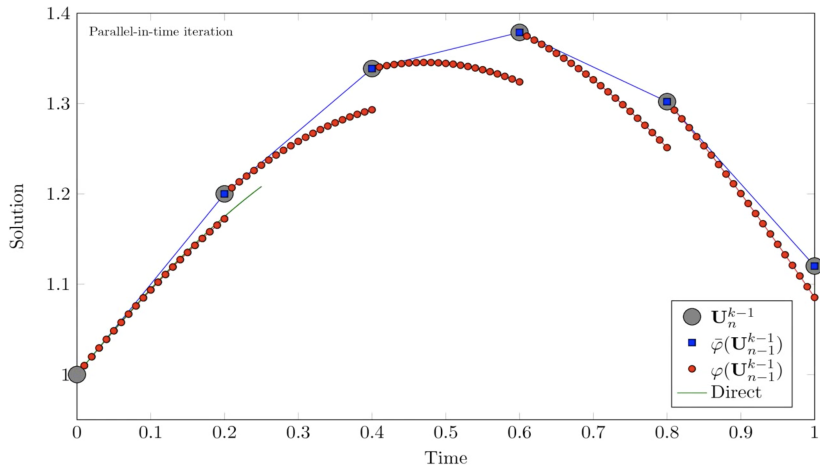
$$\begin{aligned}u'(t) &= f(u(t)), & t \in (t_0, t_f) \\u(t_0) &= u_0\end{aligned}\tag{1}$$

Divide the time domain  $(t_0, t_f)$  into  $N$  **equal subdomains**  $\Omega_n = (T_n, T_{n+1})$ .

## Parareal Algorithm

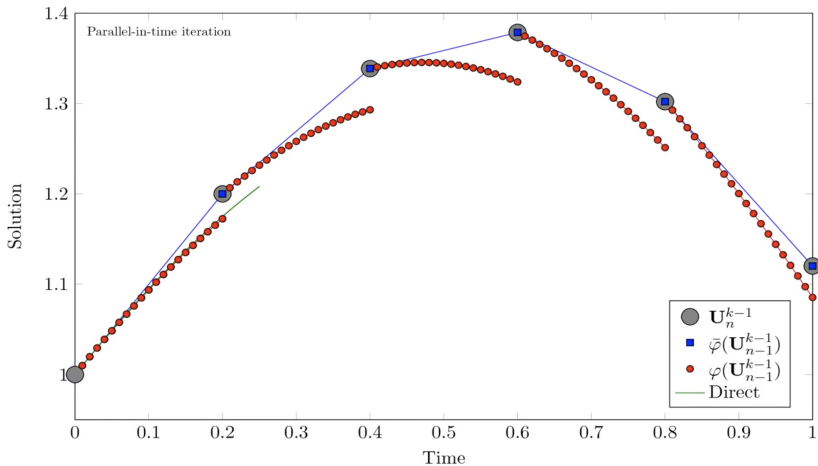
$$\begin{aligned}U_0^{k+1} &= u_0 \\U_n^{k+1} &= \mathcal{G}(U_{n-1}^{k+1}) + \left[ \mathcal{F}(U_{n-1}^k) - \mathcal{G}(U_{n-1}^k) \right]\end{aligned}$$

- $U_n^k$  is the solution  $u(T_n)$  on the  $k^{\text{th}}$  iteration of the algorithm.
- $\mathcal{G}$  denotes a **coarse solver**
- $\mathcal{F}$  denotes a **fine solver**



$$U_0^k = u_0 \quad U_n^k = \mathcal{G}(U_{n-1}^k) + \left[ \mathcal{F}(U_{n-1}^{k-1}) - \mathcal{G}(U_{n-1}^{k-1}) \right]$$

Video Source: <https://en.wikipedia.org/wiki/Parareal>



$$U_0^k = u_0 \quad U_n^k = \mathcal{G}(U_{n-1}^k) + \left[ \mathcal{F}(U_{n-1}^{k-1}) - \mathcal{G}(U_{n-1}^{k-1}) \right]$$

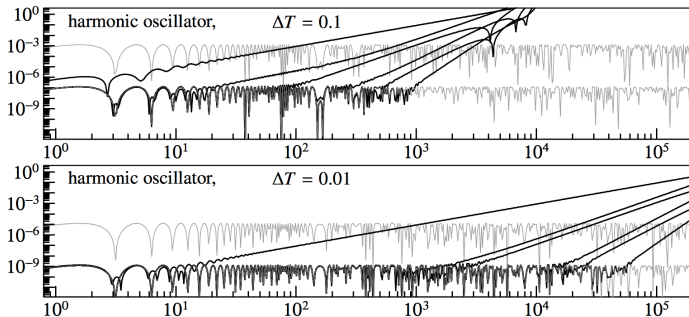
*Efficiency = Iterations for Convergence / Total Number of Steps*

Video Source: <https://en.wikipedia.org/wiki/Parareal>



# Drawbacks of the Parareal Algorithm

For Hamiltonian systems, **very high accuracy** is required for the coarse solver!<sup>2</sup> A good coarse approximation is needed.



Typical coarse integrators fail to predict phase correctly, leading to blow up of error during correction step with fine solver.

<sup>2</sup>Martin J Gander and Ernst Hairer. "Analysis for parareal algorithms applied to Hamiltonian differential equations". In: *Journal of Computational and Applied Mathematics* 259 (2014), pp. 2–13.

# Phase Plane Map

Consider a set of  $M$  initial conditions (**training points**)  $\{u_0^i\}_{i=1}^M$  and a set of  $M$  corresponding (**target points**)  $\{v^i\}_{i=1}^M$  defined by,

$$v^i = \mathcal{F}(u_0^i) \quad i = 1, \dots, M$$

The set  $\{u_0^i \rightarrow v^i\}_{i=1}^M$  acts as a type of **look-up table** for future initials conditions<sup>3</sup>.

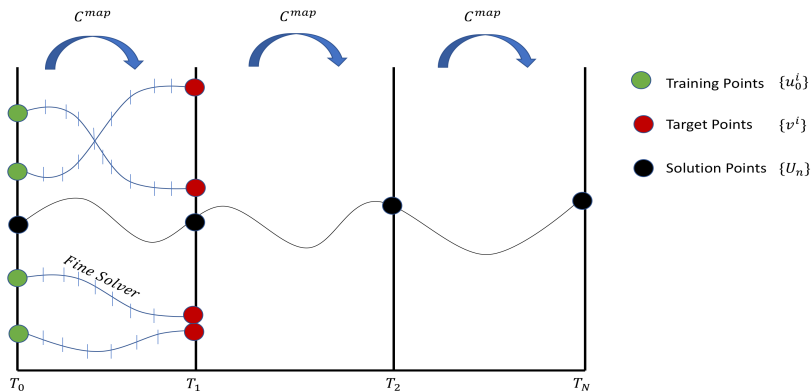
## Definition

Define a **phase plane map**  $\mathcal{G}^{map}(\{u_0^i \rightarrow v^i\}_{i=1}^M, U_n)$  to be a coarse solver that uses the information contained in  $\{u_0^i \rightarrow v^i\}_{i=1}^M$  and a point  $U_n$  at time  $T_n$ , to determine the solution  $U_{n+1}$  at time  $T_{n+1}$ .  $\mathcal{G}^{map}$  can be defined through, e.g., **Interpolation** or a **Neural Network**.

---

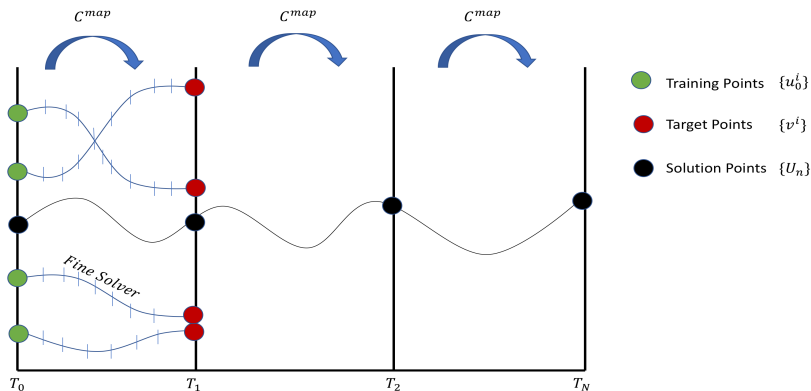
<sup>3</sup>Jürg Nievergelt. "Parallel methods for integrating ordinary differential equations". In: *Communications of the ACM* 7.12 (1964), pp. 731–733.

# Phase Plane Map



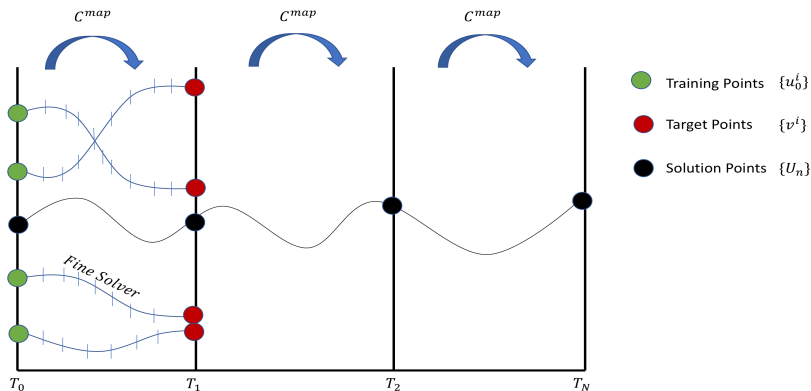
$$v^i = \mathcal{F}(u_0^i) \quad U_{n+1} \approx \mathcal{G}^{map}(\{u_0^i \rightarrow v^i\}_{i=1}^M, U_n)$$

# Phase Plane Map



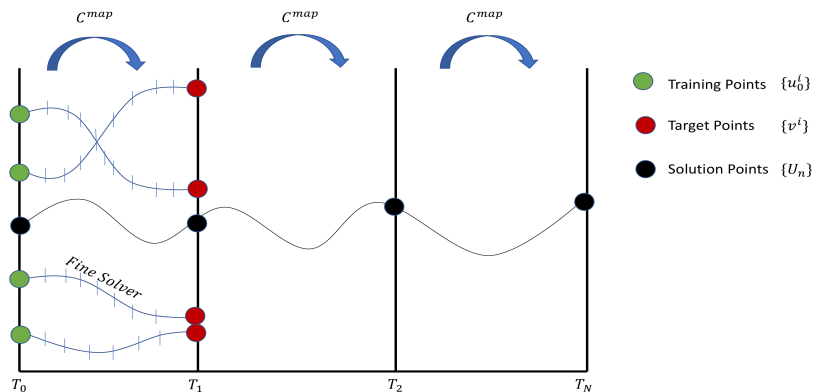
The computation of  $v^i = \mathcal{F}(u_0^i)$  is **embarrassingly parallel**.

# Phase Plane Map



For autonomous ODEs,  $\mathcal{G}^{map}$  can be applied over each subdomain  $\Omega_n$ .  
 Else, generate  $\{u_j^i \rightarrow v^i\}_{i=1}^M$  for each  $\Omega_j$ ,  $j = 0, \dots, N - 1$ .

# Phase Plane Map



- 1 How to select  $\{u_0^i\}_{i=1}^M$  ?
- 2 How to select  $M$  ?
- 3 Which method is best for  $\mathcal{G}^{map}$  ?

# Numerical Examples

- **Traditional** parareal algorithm  $\implies \mathcal{G} = \text{RK4}$  with step size  $T/N$
- **Modified** parareal algorithm  $\implies \mathcal{G} = \text{phase plane map}$  with step size  $T/N$
- For both, the fine solver is an **adaptive RK45 solver**.
- $T = \text{Total Simulation Time}$ .
- $N = \text{Number of time-subdomains } \Omega_n$ .

# Harmonic Oscillator

Consider a simple **harmonic oscillator**:

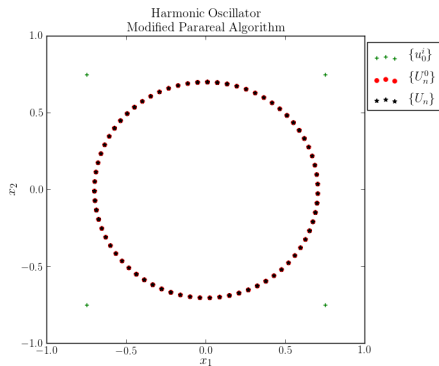
$$\begin{aligned} \dot{x}_1 &= \frac{1}{\varepsilon} x_2 \\ \dot{x}_2 &= \frac{-1}{\varepsilon} x_1 \end{aligned}$$

with

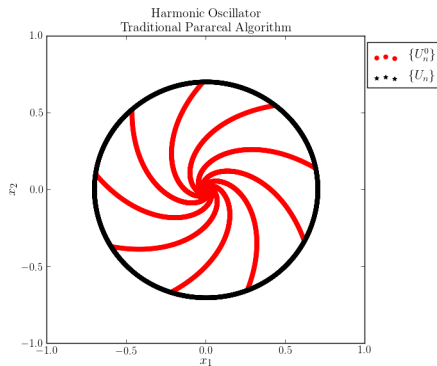
- $\varepsilon = 0.01$
- $T = 70$  (1000 revolutions)
- $\mathbf{x}(0) = [-2/9, -2/3]^T$



# Harmonic Oscillator



$N = 100, M = 2$   
Iterations: 1



$N = 10,000$   
Iterations: 42

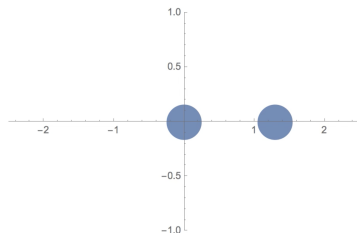
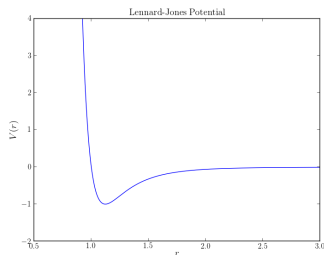
# Lennard-Jones System

$$r'' = \frac{2}{m}F(r)$$

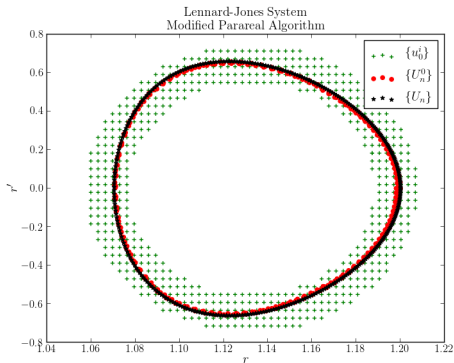
$$F(r) = -\nabla V(r)$$

$$V(r) = 4\varepsilon \left[ \left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6 \right]$$

- $\varepsilon$  = “well depth”
- $\sigma$  = zero distance
- $r_{\min} = 1.12$



# Lennard-Jones System



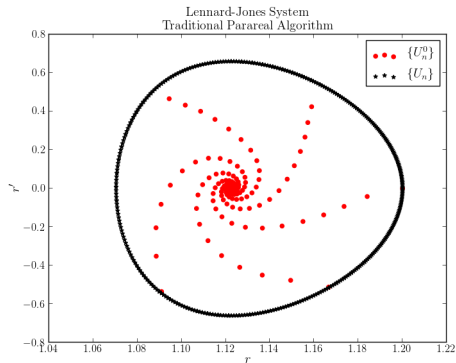
Iterations: 10

$$T = 50$$

$$N = 400$$

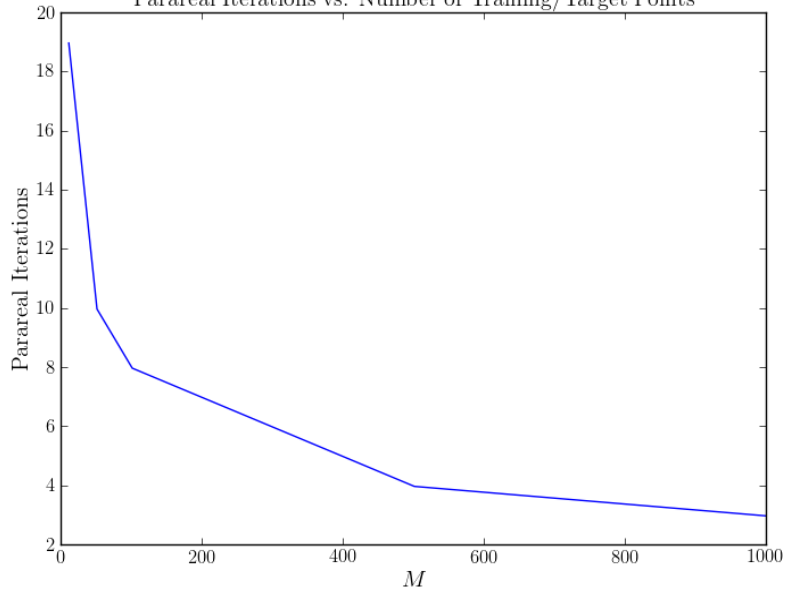
$$\mathbf{r}_0 = [1.2, 0]^T$$

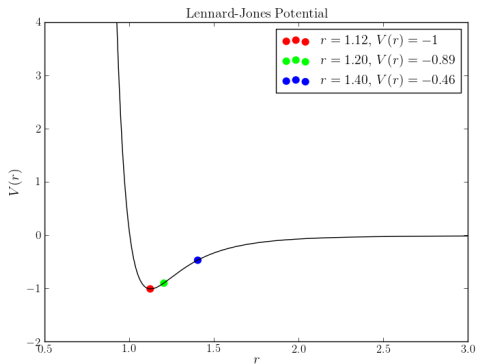
$$M=50$$



Iterations: 53

Parareal Iterations vs. Number of Training/Target Points





●  $r = 1.12$

M	Iterations
3	7
5	3
10	2
50	1

●  $r = 1.2$

M	Iterations
10	19
50	10
100	8
500	4

●  $r = 1.4$

M	Iterations
50	207
100	39
500	14
1000	9

# High Dimensional Harmonic Oscillator

$$\dot{\mathbf{x}} = \frac{1}{\varepsilon} A \mathbf{x}$$

where  $x = [x_1, v_1, x_2, v_2, x_3, v_3, x_4, v_4]^T$ ,  $\varepsilon = 0.01$ , and  $A \in \mathbb{R}^{8 \times 8}$  is defined as,

$$A = \begin{bmatrix} 0 & a & 0 & 0 & 0 & 0 & 0 & 0 \\ -a & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & b & 0 & 0 & 0 & 0 \\ 0 & 0 & -b & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c & 0 & 0 \\ 0 & 0 & 0 & 0 & -c & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & d \\ 0 & 0 & 0 & 0 & 0 & 0 & -d & 0 \end{bmatrix}$$

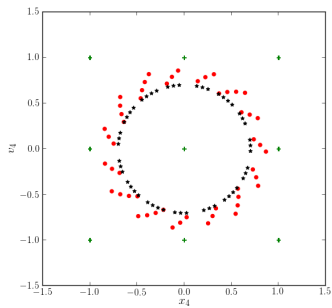
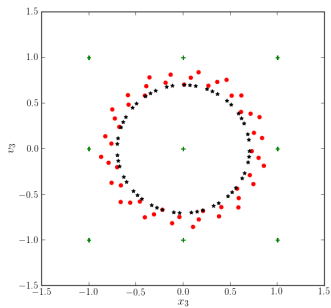
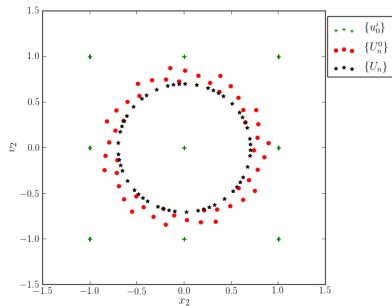
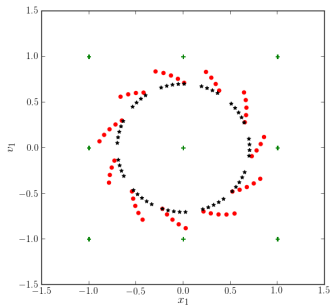
# High Dimensional Harmonic Oscillator

$$\dot{\mathbf{x}} = \frac{1}{\varepsilon} A \mathbf{x}$$

where  $x = [x_1, v_1, x_2, v_2, x_3, v_3, x_4, v_4]^T$ ,  $\varepsilon = 0.01$ , and  $A \in \mathbb{R}^{8 \times 8}$  is defined as,

$$A = \begin{bmatrix} 0 & a & 0 & 0 & 0 & 0 & 0 & 0 \\ -a & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & b & 0 & 0 & 0 & 0 \\ 0 & 0 & -b & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c & 0 & 0 \\ 0 & 0 & 0 & 0 & -c & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & d \\ 0 & 0 & 0 & 0 & 0 & 0 & -d & 0 \end{bmatrix}$$

*\*Neural network defined with logistic activation function, size  $1000 \times 1$ , with a limited memory BFGS solver for optimization.*





# Localized Multiscale Problem

- 1 **Type A:** Problems that contain **local** defects or singularities. Microscale model needed only near defects.
- 2 **Type B:** Microscale model needed **everywhere** as a supplement to macroscale model, e.g., resolve constitutive model.<sup>4</sup>

Type A Example: An **N-body Problem**:

$$m_i \ddot{\mathbf{q}}_i = \sum_{j \neq i}^n \frac{m_i m_j (\mathbf{q}_j - \mathbf{q}_i)}{\|\mathbf{q}_j - \mathbf{q}_i\|^3}$$

- $m_i$  denotes the **mass** of the  $i^{\text{th}}$  body.
- $\mathbf{q}_i$  denotes the **position** of the  $i^{\text{th}}$  body.

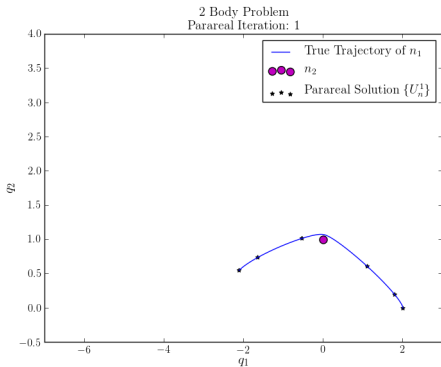
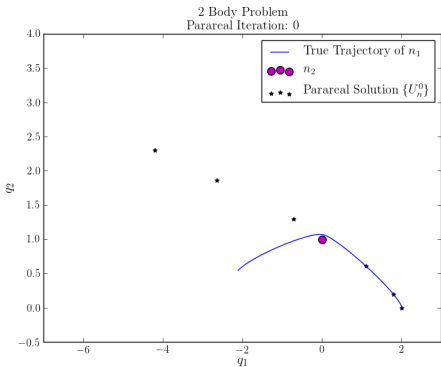
Let  $n = 2$  and assume  $m_2 \gg m_1$ .

---

<sup>4</sup>Weinan E. *Principles of multiscale modeling*. Cambridge University Press, 2011.

Apply a **traditional** parareal algorithm with

- RK45 as fine scale solver with adaptive step size.
- RK4 as coarse solver with step size  $T/N$ .








$T = 0.5$      $N = 5$     Iterations = 2

# Conclusion & Future Work

- Parallel in time algorithms needed for massively parallel simulations of time dependent dynamical systems.
- Results for parareal + phase plane map show promise. Demonstrate a proof of concept.
- Future Work
  - ▶ Potential Improvements include use of modern sparse grid and adaptive methods for interpolation or optimizing neural network approach.
  - ▶ More realistic examples in the realm of molecular dynamics.
  - ▶ Scaling/speedup results on modern supercomputing platforms.

## References

-  Weinan E. *Principles of multiscale modeling*. Cambridge University Press, 2011.
-  Martin J Gander. “50 years of time parallel time integration”. In: *Multiple Shooting and Time Domain Decomposition Methods*. Springer, 2015, pp. 69–113.
-  Martin J Gander and Ernst Hairer. “Analysis for parareal algorithms applied to Hamiltonian differential equations”. In: *Journal of Computational and Applied Mathematics* 259 (2014), pp. 2–13.
-  Jacques-Louis Lions, Yvon Maday, et al. “A parareal method in time discretization of pde’s”. In: *Comptes Rendus de l’Académie des Sciences-Series I-Mathematics* 332.7 (2001), pp. 661–668.
-  Jürg Nievergelt. “Parallel methods for integrating ordinary differential equations”. In: *Communications of the ACM* 7.12 (1964), pp. 731–733.

\*Code will be available at <https://github.com/gyalla/interpareal>