Complex fluids in the GPU era

Algorithms and simulations

Raúl P. Peláez

Universidad Autónoma de Madrid

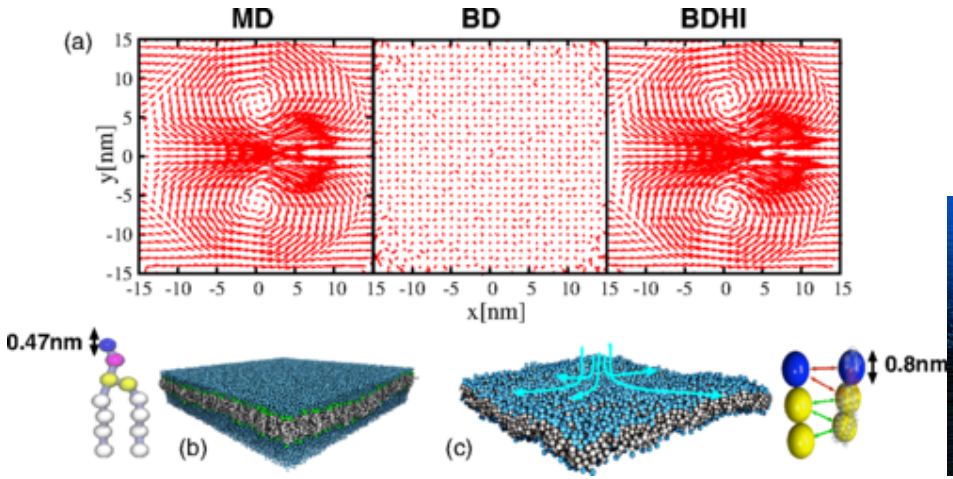
May 9, 2022

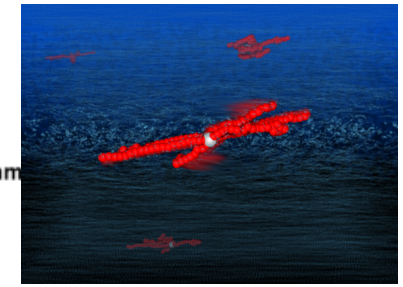
Supervisor: Rafael Delgado-Buscalioni

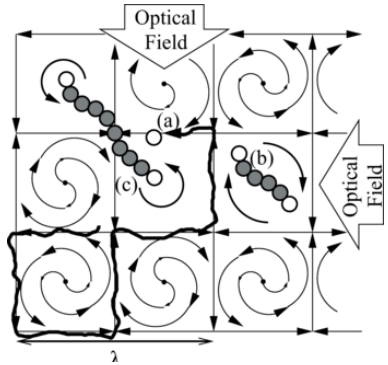
Universidad Autónoma
de Madrid

Talk outline

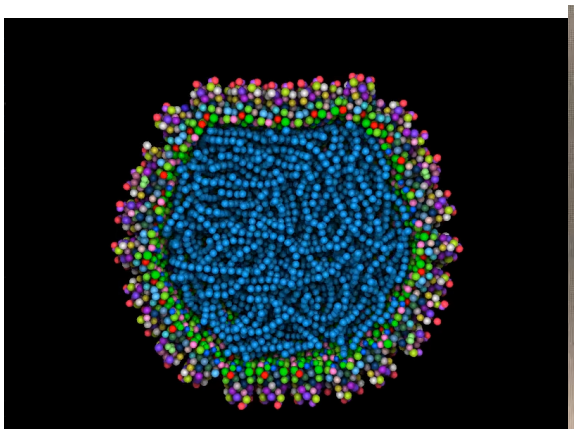
The coexistence between a liquid and solid phase

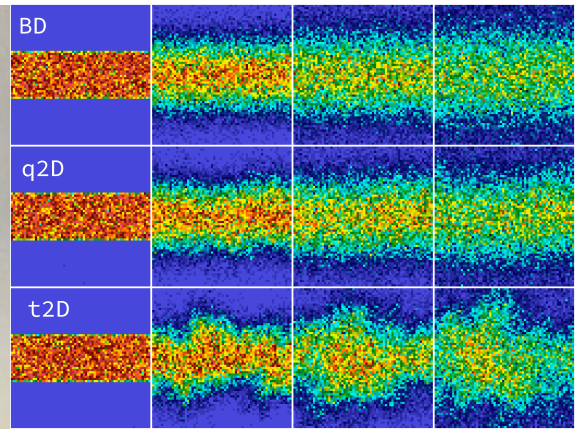






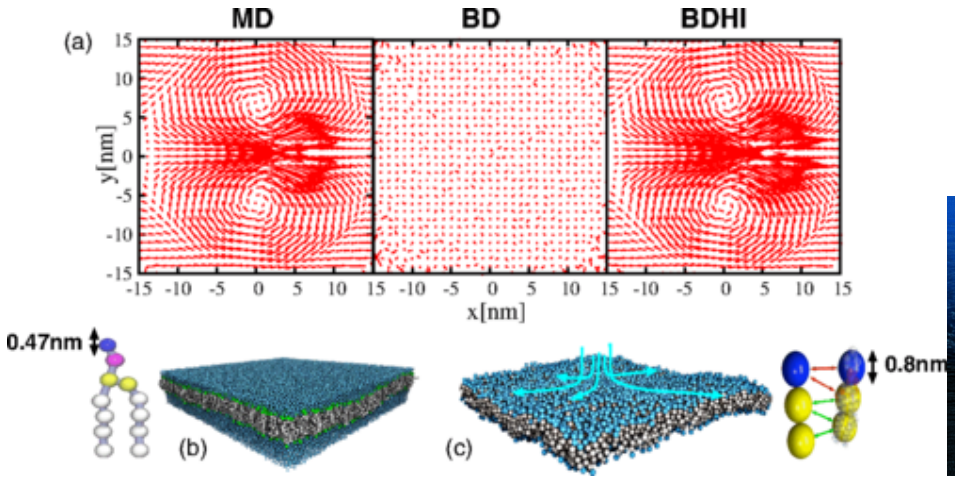
[**Panzuela, S. and Delgado-Buscalioni, R.** PRL 2018.] - [**Raul P. Pelaez and Delgado-Buscalioni, R.** Macromolecules 2020] - [**Meléndez, M.** et. al. PRE 2019.]

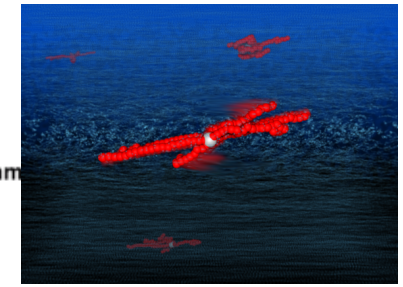


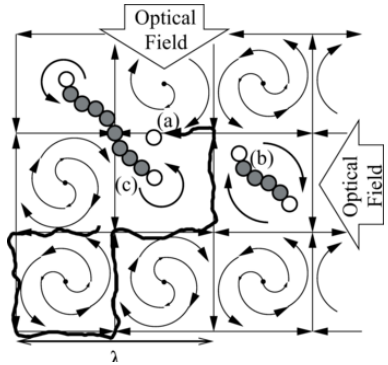


[Courtesy of Pablo Ibañez] ——— [Courtesy of Berta Tinao] ——— [**Raul P. Pelaez** et. al. JSTAT 2018.]

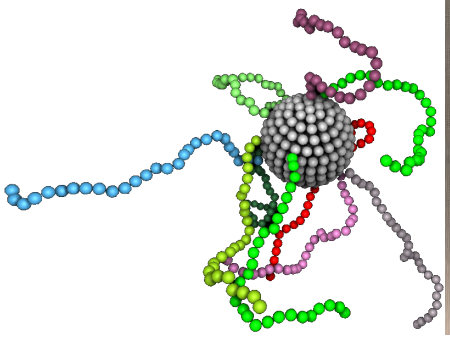
The coexistence between a liquid and solid phase

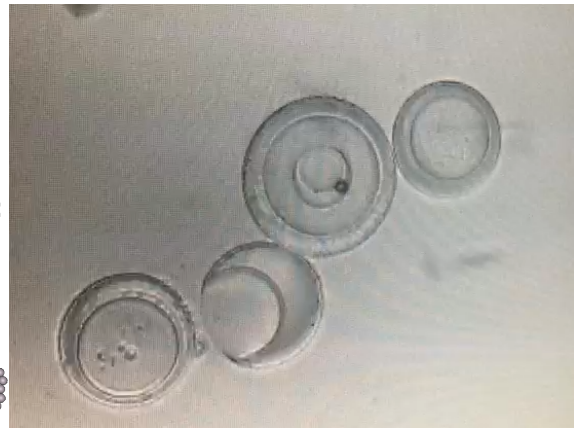


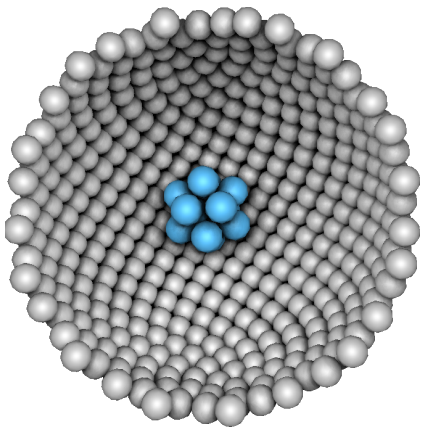




[**Panzuela, S. and Delgado-Buscalioni, R.** PRL 2018.] - [**Raul P. Pelaez and Delgado-Buscalioni, R.** Macromolecules 2020] - [**Meléndez, M.** et. al. PRE 2019.]







[Courtesy of Pablo Palacios] [Courtesy of Berta Tinao] [Courtesy of Pablo Palacios]

Complex fluids

The spatio-temporal landscape of numerical techniques.

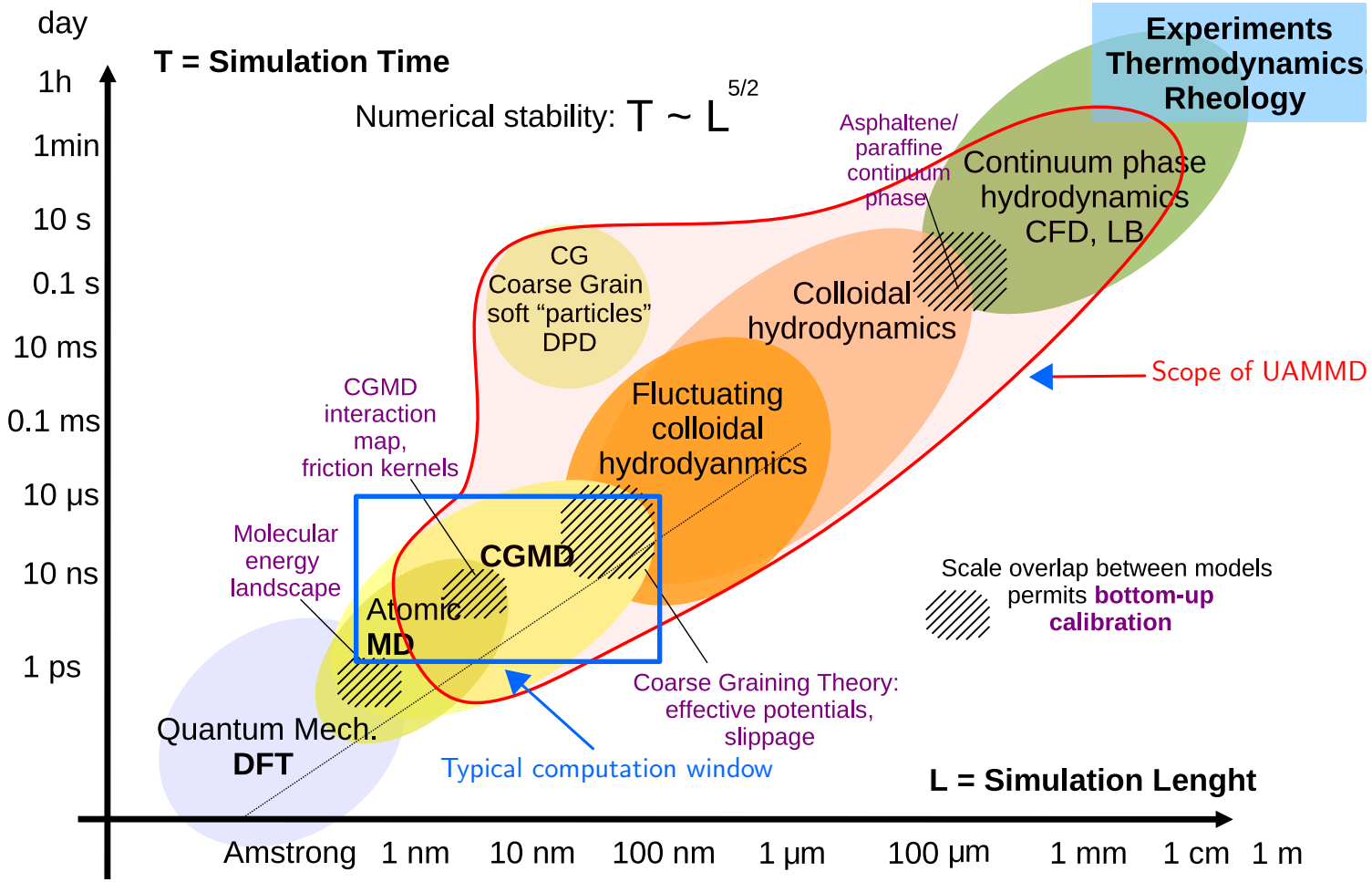


Figure: Figure courtesy of Rafael Delgado-Buscalioni.

Complex fluids

The spatio-temporal landscape of numerical techniques.

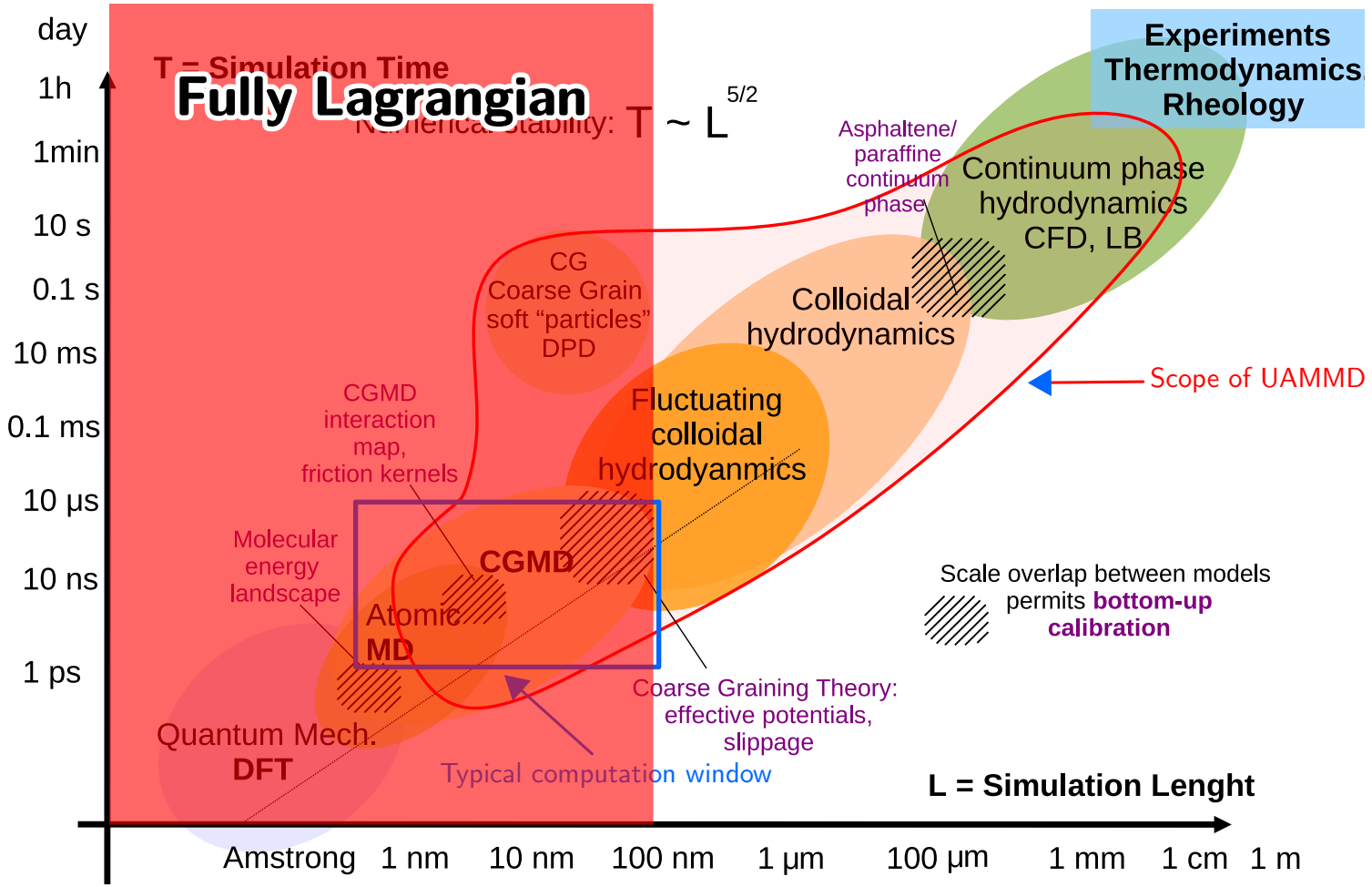


Figure: Figure courtesy of Rafael Delgado-Buscalioni.

Complex fluids

The spatio-temporal landscape of numerical techniques.

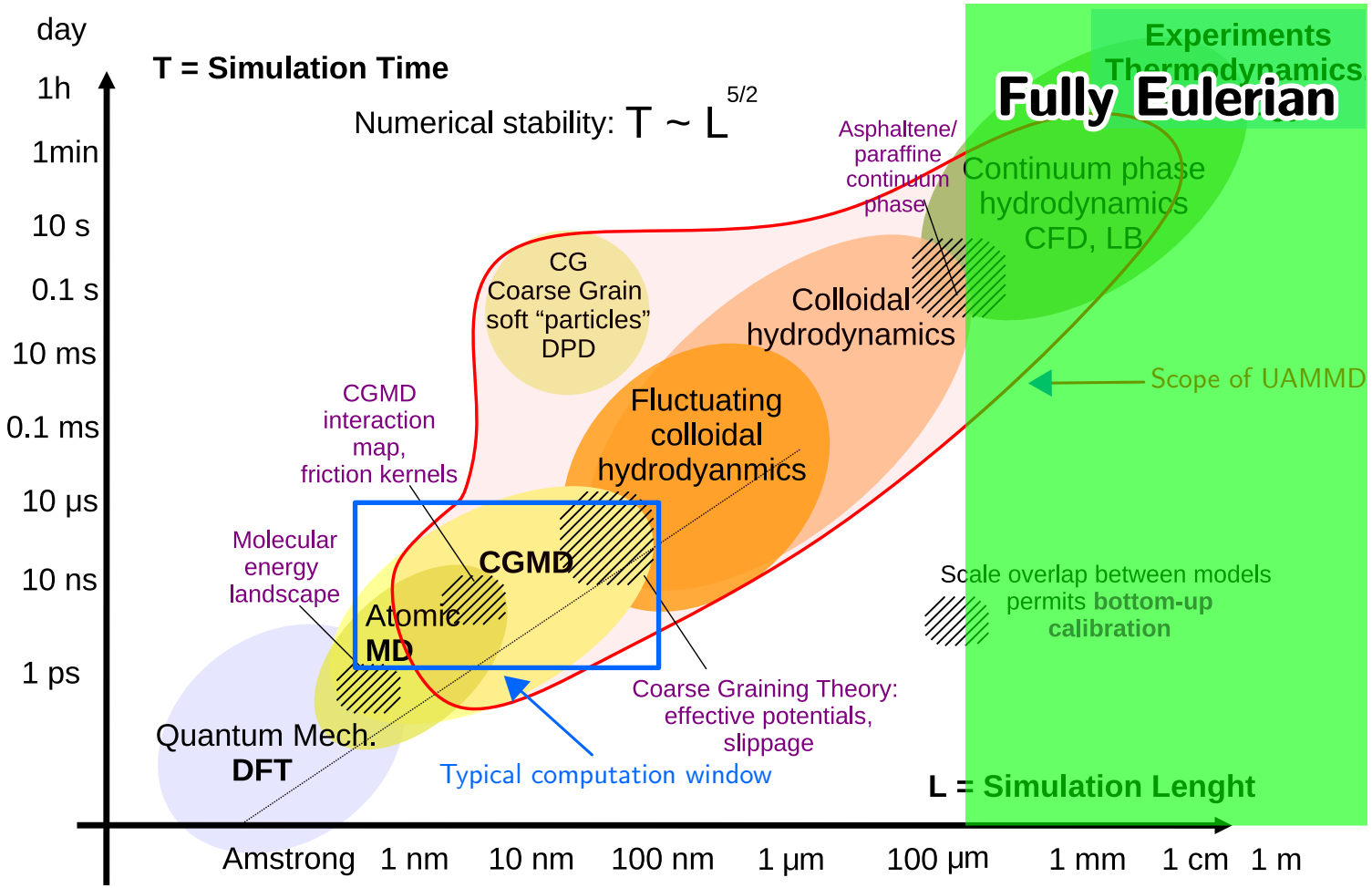


Figure: Figure courtesy of Rafael Delgado-Buscalioni.

Complex fluids

The spatio-temporal landscape of numerical techniques.

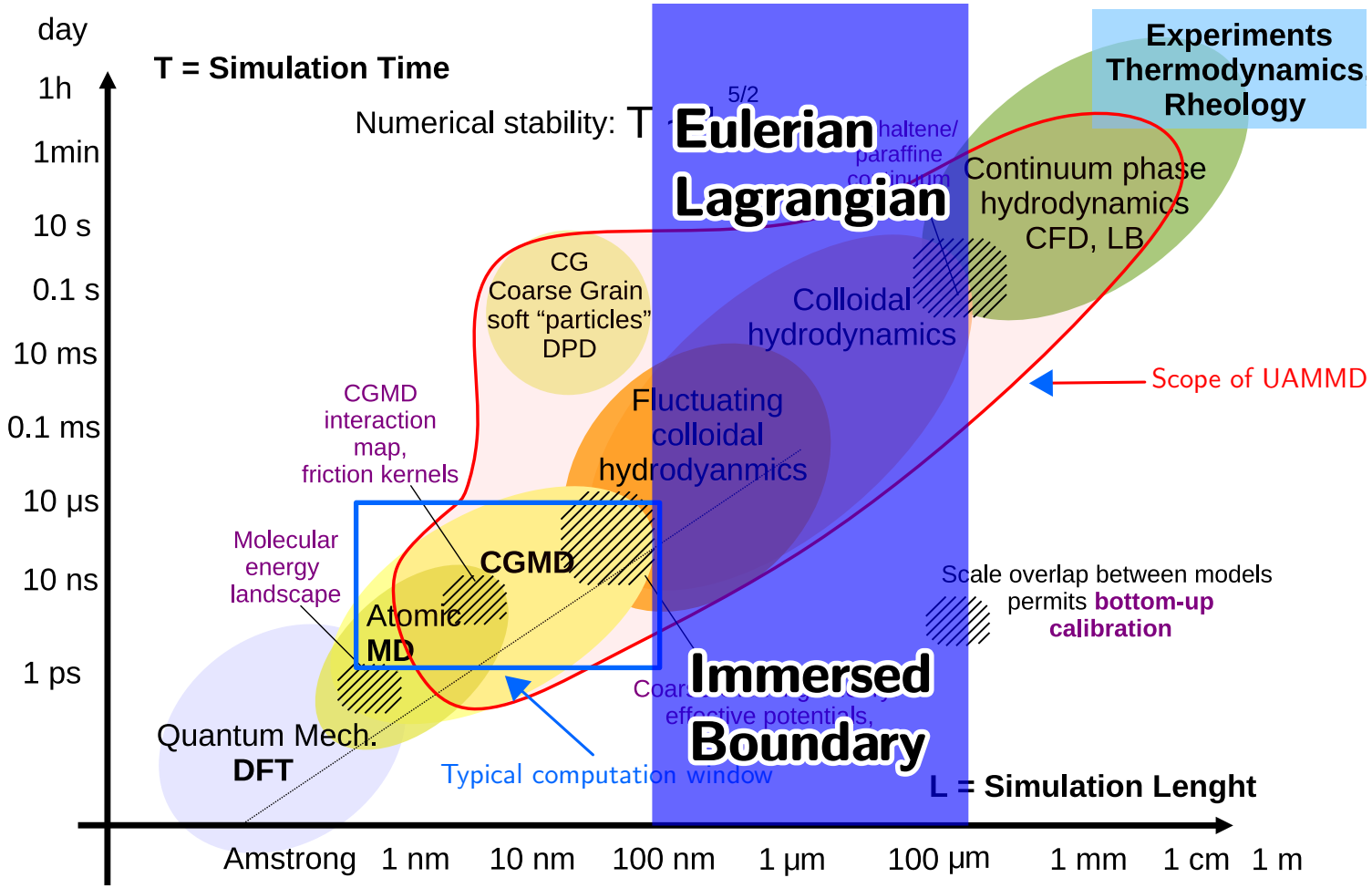


Figure: Figure courtesy of Rafael Delgado-Buscalioni.

Complex fluids

The spatio-temporal landscape of numerical techniques.

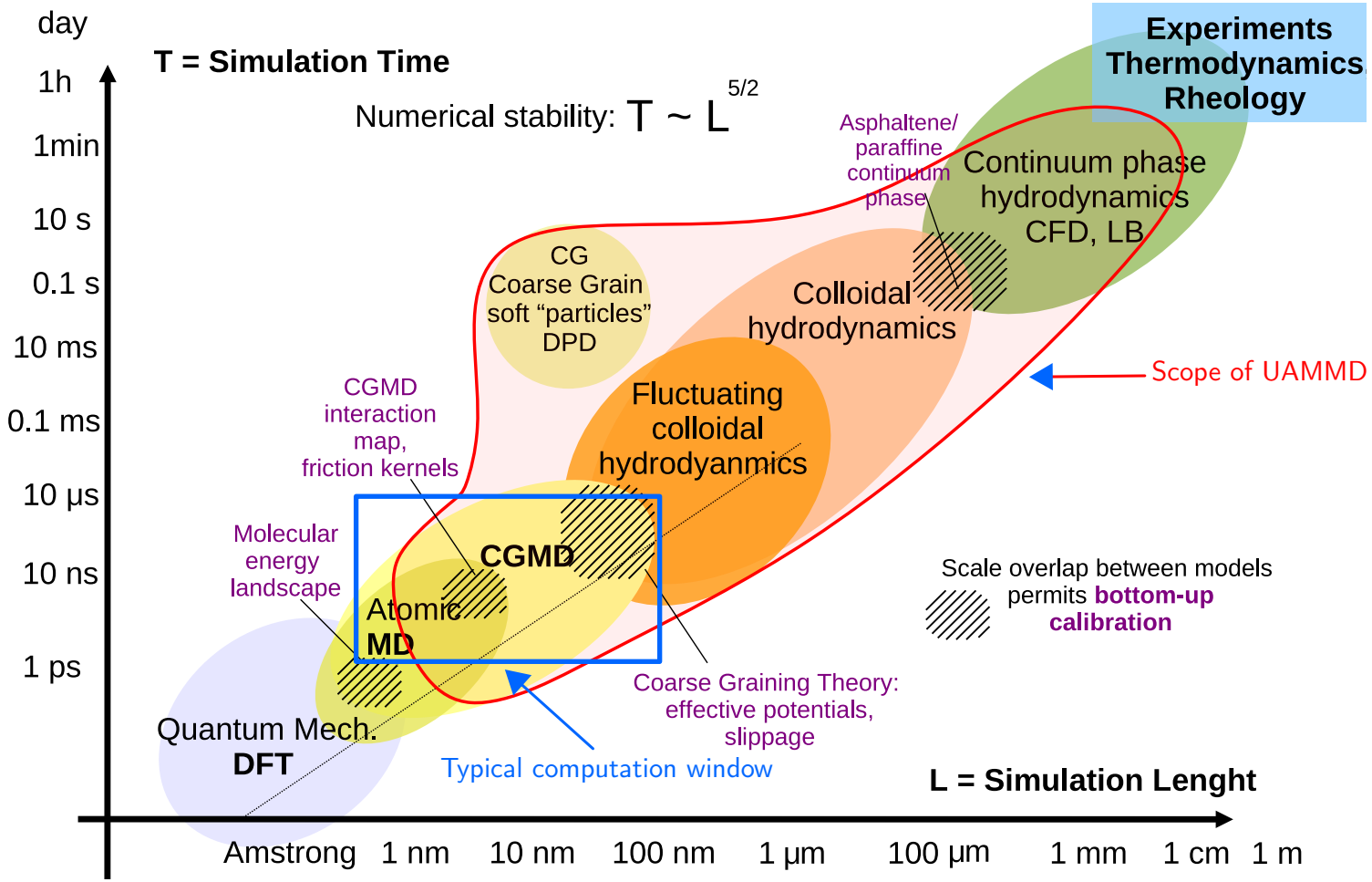
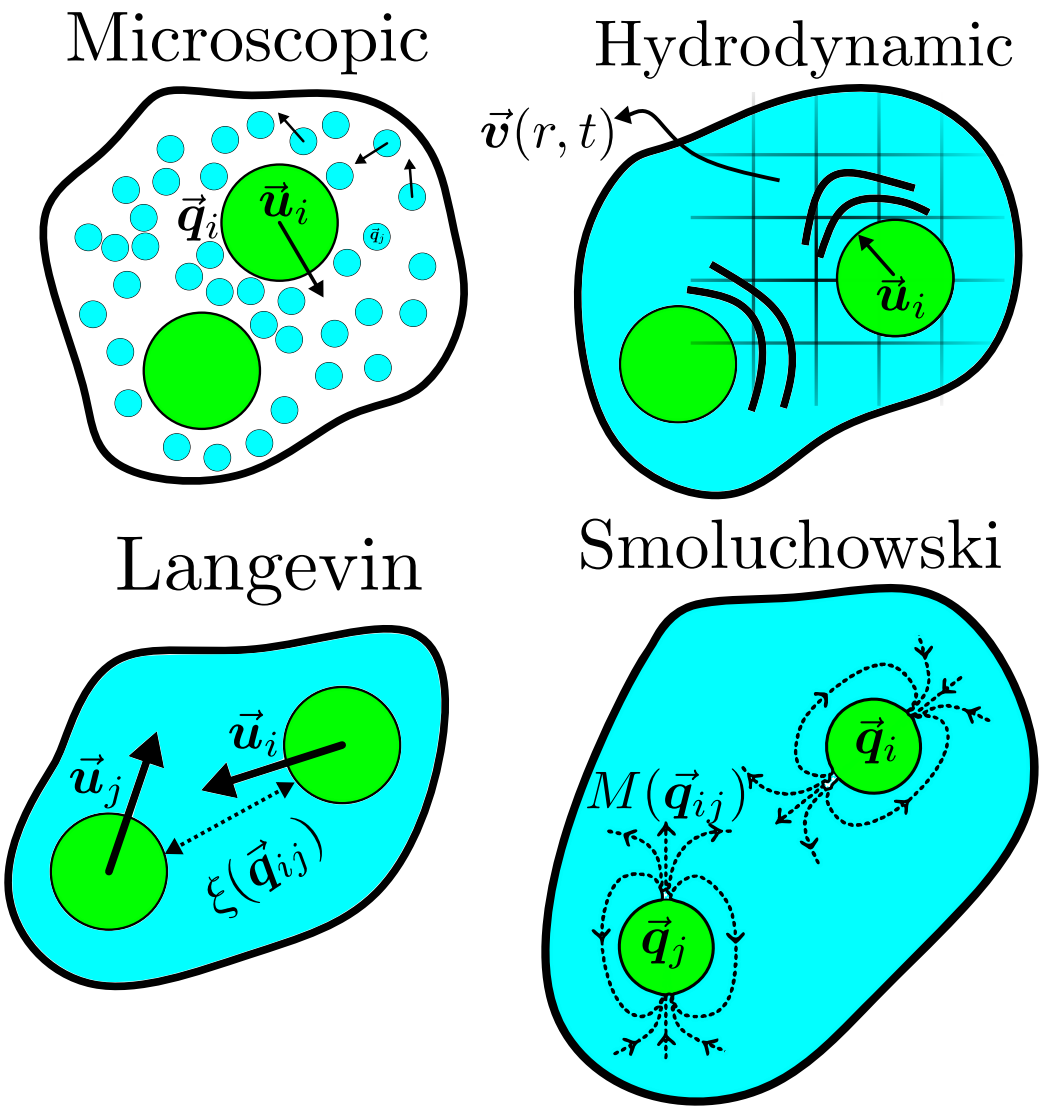


Figure: Figure courtesy of Rafael Delgado-Buscalioni.

Complex fluids

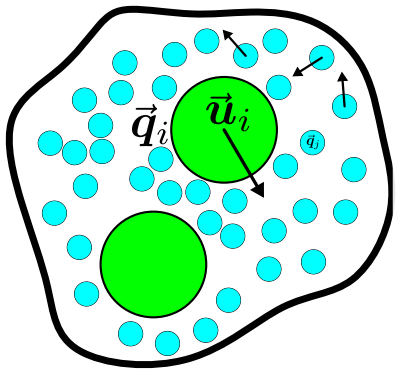
Usual levels of coarse-grained description



Complex fluids

Usual levels of coarse-grained description

Microscopic



Hydrodynamic

Langevin

Smoluchowski

Relevant variables:

- $\boldsymbol{q}_i$: Position of particle i.
- $\boldsymbol{u}_i$: Velocity of particle i.

Molecular Dynamics (MD)

$$m\ddot{\boldsymbol{q}} = \boldsymbol{F}$$

$$\boldsymbol{u} = \dot{\boldsymbol{q}}$$

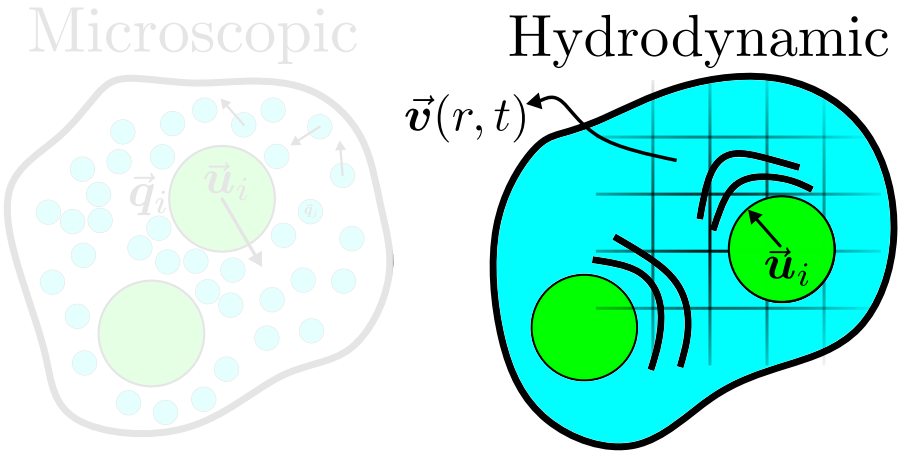
Typical timescale:

$$\tau \sim 10^{-12} s$$

Supervector notation $\boldsymbol{q} := \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_N\}$

Complex fluids

Usual levels of coarse-grained description



Relevant variables:

- $\boldsymbol{q}_i$: Position of particle i.
- $\boldsymbol{u}_i$: Velocity of particle i.
- $\boldsymbol{v}(\boldsymbol{r},t)$: Fluid velocity field.

Fluctuating Hydrodynamics

$$\rho\partial_t \boldsymbol{v} = -\boldsymbol{\partial}_{\boldsymbol{r}} \cdot \boldsymbol{\sigma} + \boldsymbol{f} + \text{fluct}$$

$$\int_{V_p} \boldsymbol{f} d\boldsymbol{r} = \boldsymbol{F}_i \quad \text{and} \quad \boldsymbol{u}_i = \int_{V_p} \boldsymbol{v} d\boldsymbol{r}$$

Typical timescale:

$$\tau \sim 10^{-\{9-6\}} s$$

$$\partial_t := \frac{\partial}{\partial t} \rightarrow \boldsymbol{\partial}_{\boldsymbol{r}} := \nabla := (\partial_x, \partial_y, \partial_z)$$

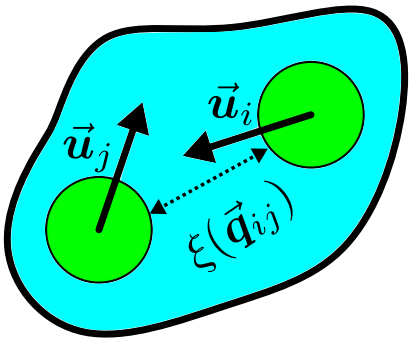
Complex fluids

Usual levels of coarse-grained description

Langevin Dynamics (LD)

$$m d\boldsymbol{u} = \boldsymbol{F} dt - \xi \boldsymbol{u} dt + \sqrt{2\xi k_B T}\widetilde{\boldsymbol{W}}$$

Langevin



Relevant variables:

- $\boldsymbol{q}_i$: Position of particle i.
- $\boldsymbol{u}_i$: Velocity of particle i.
- $\xi(\boldsymbol{q}_{ij})$: Friction kernel.

Typical timescale:

$$\tau \sim 10^{-5} s$$

$\widetilde{\boldsymbol{W}}$: Wienner increments, $\mathcal{N}(0, dt)$

Complex fluids

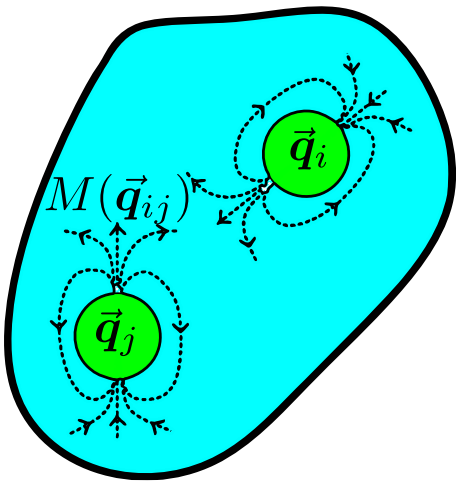
Usual levels of coarse-grained description

Brownian Dynamics (BD)

$$d\boldsymbol{q} = \boldsymbol{\mathcal{M}}\boldsymbol{F}dt + \sqrt{2k_BT\boldsymbol{\mathcal{M}}}d\widetilde{\boldsymbol{W}} + k_BT\boldsymbol{\partial_q}\cdot\boldsymbol{\mathcal{M}}dt.$$

Langevin

Smoluchowski



Relevant variables:

- $\boldsymbol{q}_i$: Position of particle i.
- $\boldsymbol{\mathcal{M}}(\boldsymbol{q}_{ij})$: Mobility tensor.

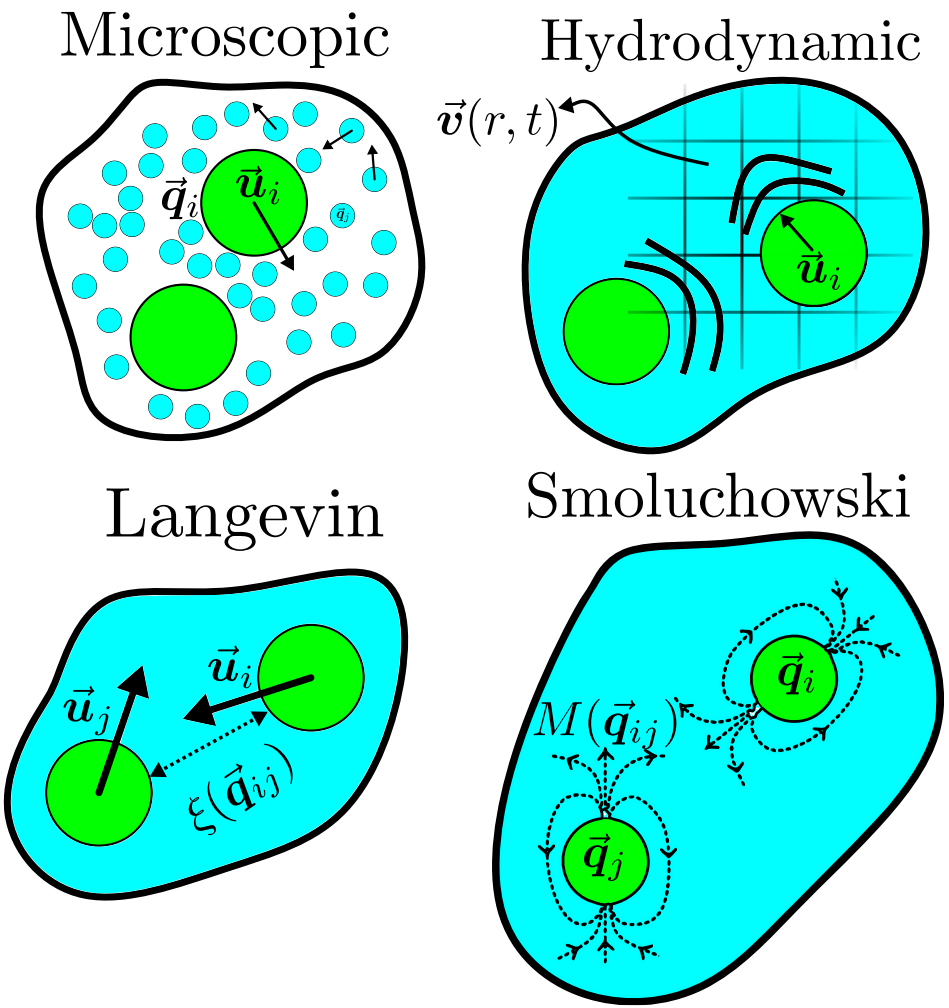
Typical timescale:

$$\tau \sim 10^{-3} s$$

$d\widetilde{\boldsymbol{W}}$: Wienner increments, $\mathcal{N}(0, dt)$

Complex fluids

Usual levels of coarse-grained description



Relevant variables:

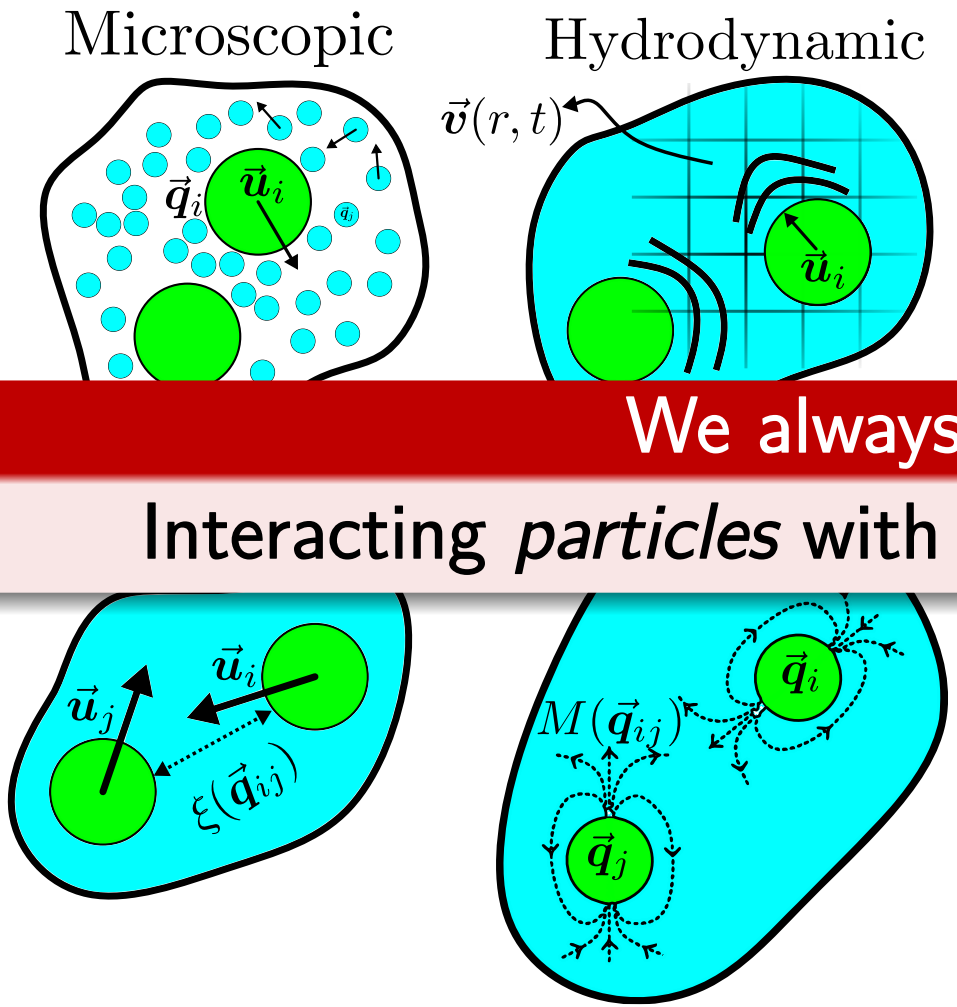
- $\boldsymbol{q}_i$: Position of particle i.
- $\boldsymbol{u}_i$: Velocity of particle i.
- $\xi(\boldsymbol{q}_{ij})$: Friction kernel.
- $\boldsymbol{v}(\boldsymbol{r},t)$: Fluid velocity field.
- $\boldsymbol{\mathcal{M}}(\boldsymbol{q}_{ij})$: Mobility tensor.

Timescale range:

$$\tau \sim [10^{-12}, 10^{-3}]s$$

Complex fluids

Usual levels of coarse-grained description



Relevant variables:

- $\boldsymbol{q}_i$: Position of particle i.
- $\boldsymbol{u}_i$: Velocity of particle i.

We always have

Interacting *particles* with a *state* that *evolves*.

- $\boldsymbol{\mathcal{M}}(\boldsymbol{q}_{ij})$: Mobility tensor.

Timescale range:

$\tau \sim [10^{-12}, 10^{-3}]s$

Computational challenges

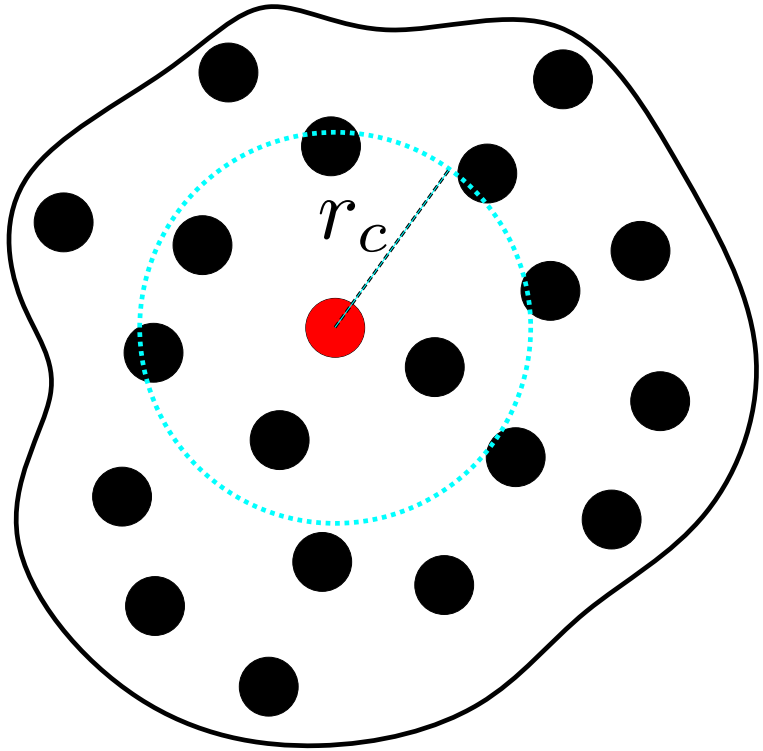
1. Short range interactions
2. Particle-grid coupling

Computational challenges

1. Short range interactions
2. Particle-grid coupling

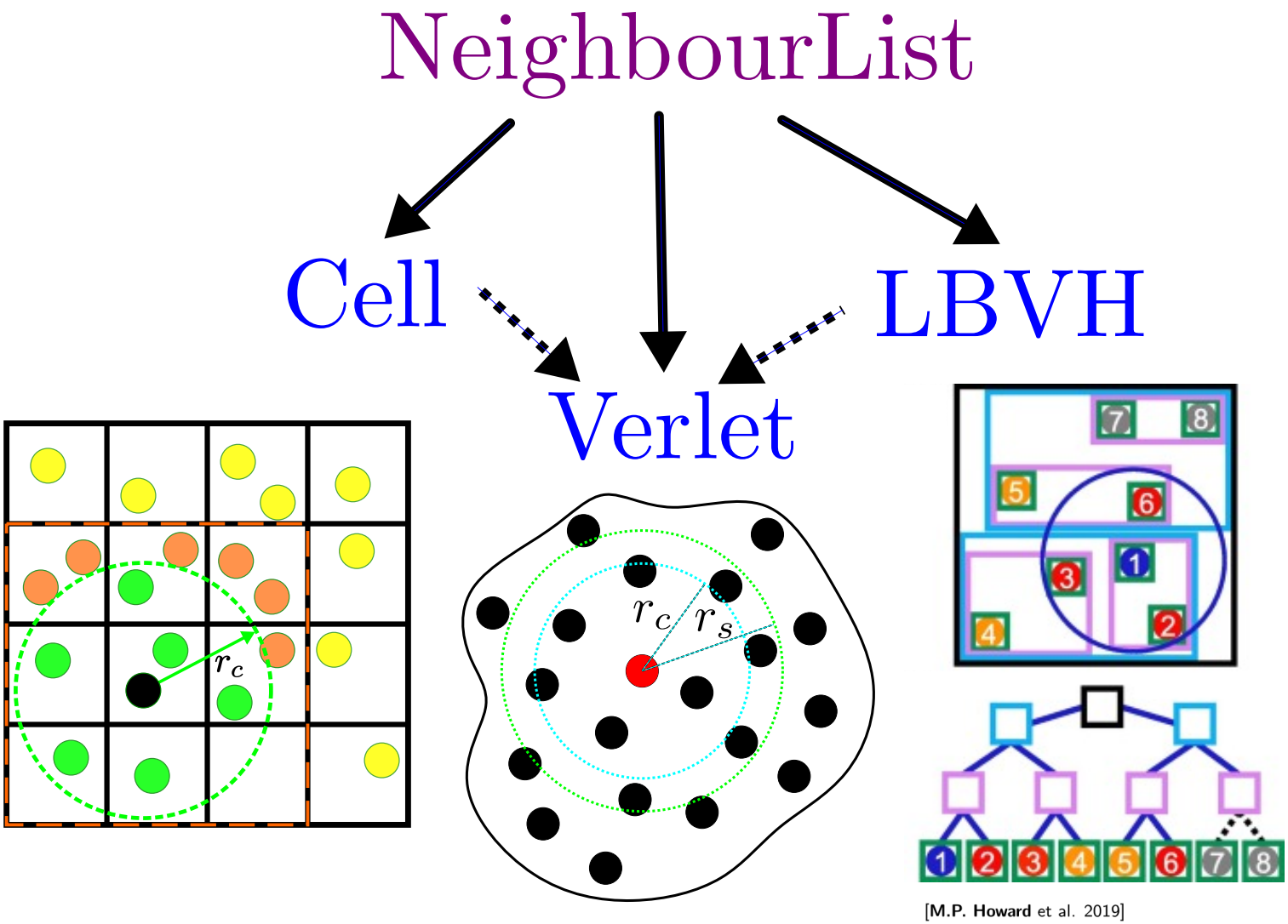
   Example: Lennard-Jones potential

$$U_{LJ}(r) = \begin{cases} 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] & r < r_c \\ 0 & r \geq r_c \end{cases}$$



Short range interactions

Neighbour lists



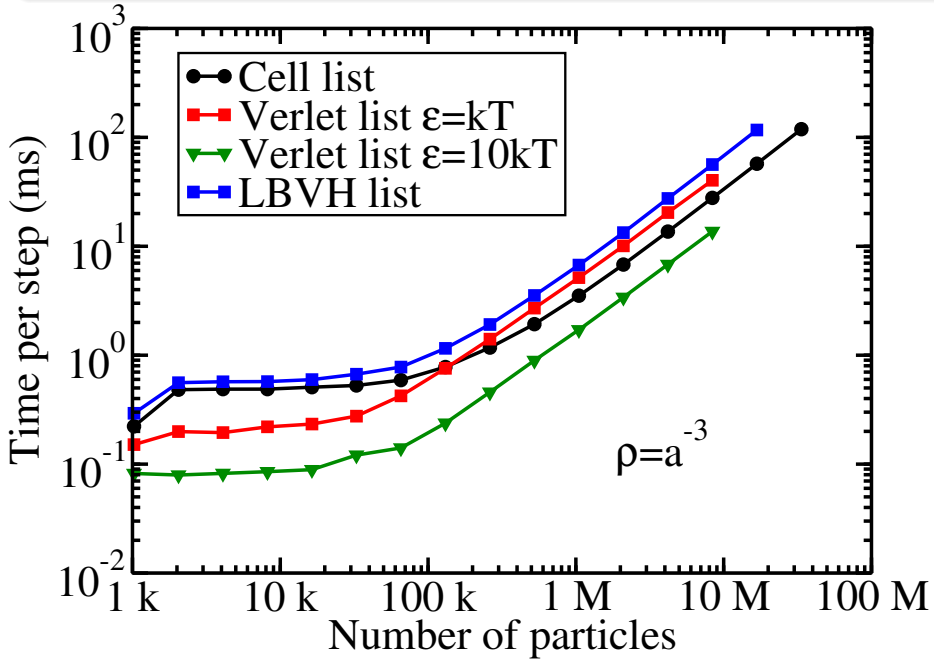
[M.P. Howard et al. 2019]

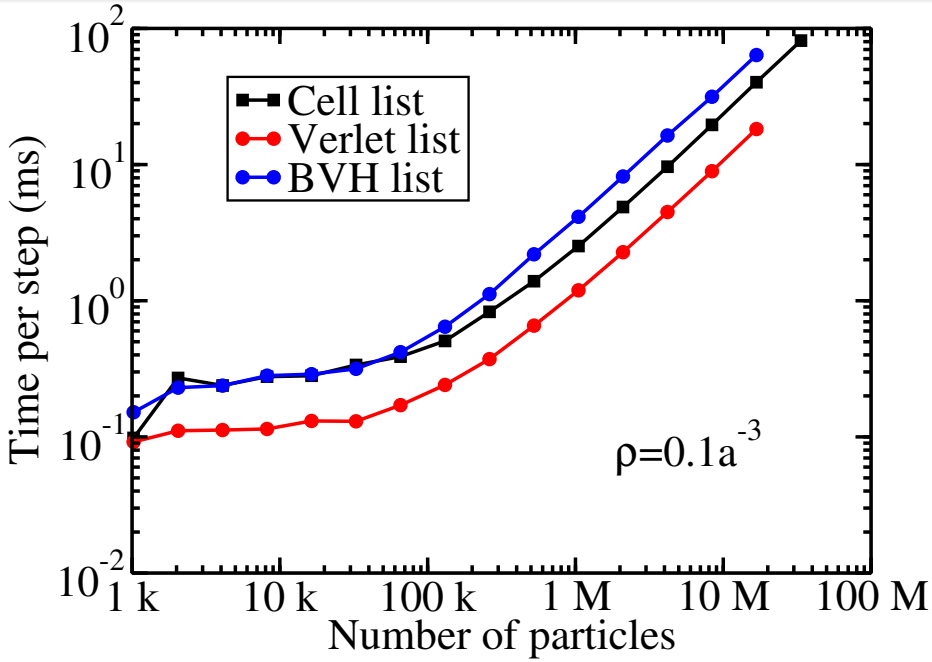
Short range interactions

Neighbour lists: Performance

Example: Lennard-Jones potential, $r_c = 2.5\sigma$ and $\varepsilon = kT$.

$$U_{LJ}(r) = \begin{cases} 4\varepsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right] & r < r_c \\ 0 & r \geq r_c \end{cases}$$





Ran with a single RTX2080Ti GPU.

Computational challenges

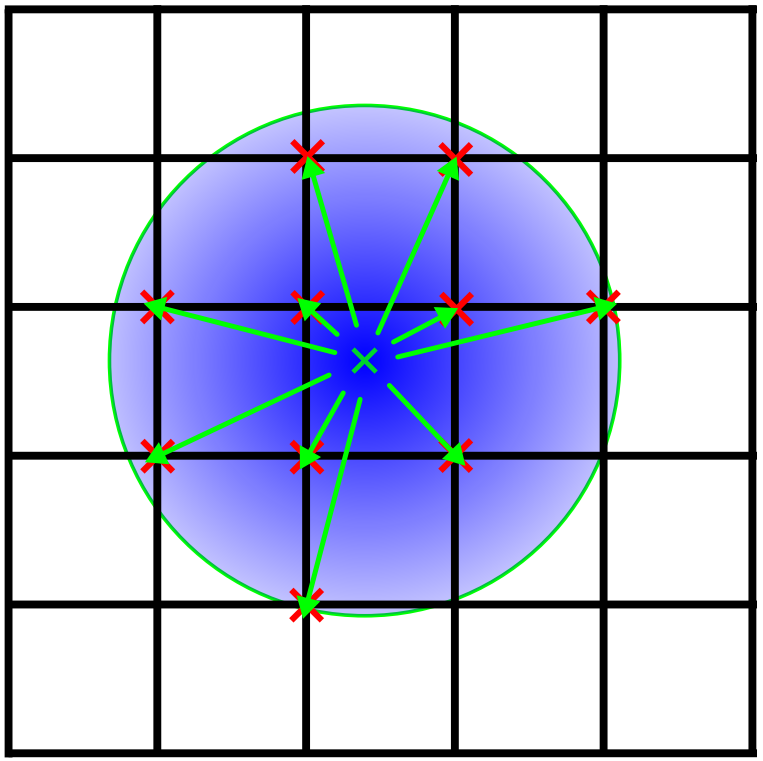
1. Short range interactions
2. Particle-grid coupling

Computational challenges

1. Short range interactions
2. Particle-grid coupling

Spreading ($\mathcal{S}$):

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F} := \sum_i \boldsymbol{F}_i \delta_a(\boldsymbol{r} - \boldsymbol{q}_i)$$



Supervector notation $\boldsymbol{F} := \{\boldsymbol{F}_1, \ldots, \boldsymbol{F}_N\}$

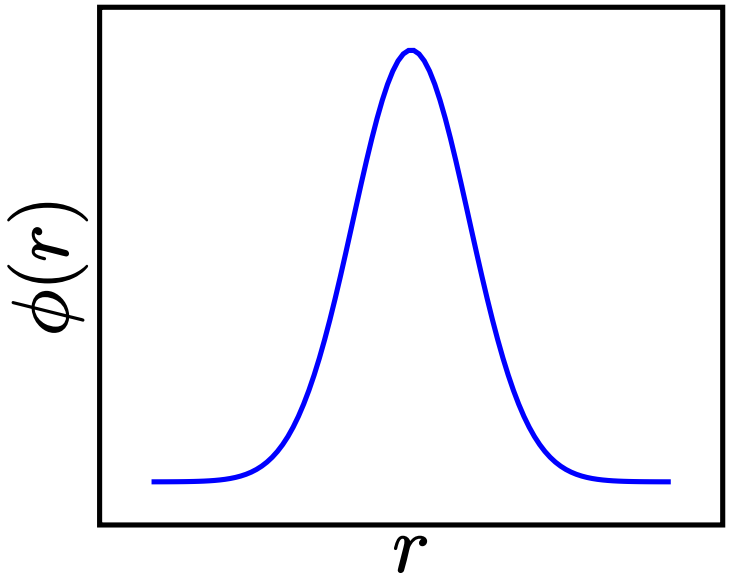
Computational challenges

1. Short range interactions
2. Particle-grid coupling

Spreading ($\mathcal{S}$):

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F} := \sum_i \boldsymbol{F}_i \delta_a(\boldsymbol{r} - \boldsymbol{q}_i)$$

$\delta_a(\boldsymbol{r}) := \phi(r_x)\phi(r_y)\phi(r_z) \rightarrow$ Smeared delta



Example: $\phi(r) \propto e^{-\frac{r^2}{2\sigma^2}}$

Computational challenges

1. Short range interactions
2. Particle-grid coupling

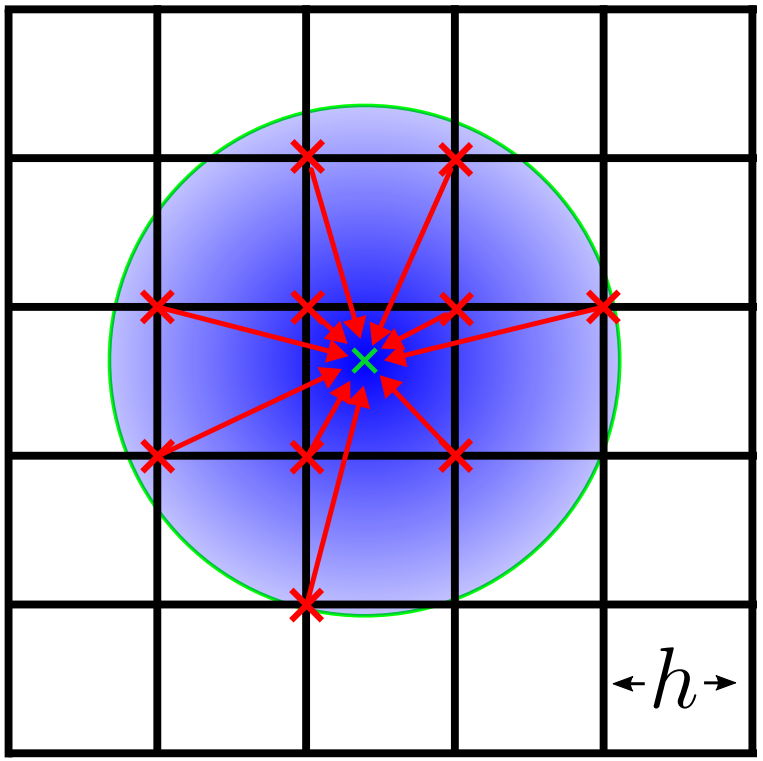
Spreading ($\mathcal{S}$):

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F} := \sum_i \boldsymbol{F}_i \delta_a(\boldsymbol{r} - \boldsymbol{q}_i)$$

$\delta_a(\boldsymbol{r}) := \phi(r_x)\phi(r_y)\phi(r_z) \rightarrow$ Smeared delta

Interpolation ($\mathcal{J}$):

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r}) = \int \boldsymbol{v}(\boldsymbol{r})\delta_a(\boldsymbol{r} - \boldsymbol{q}_i)d\boldsymbol{r}$$



Computational challenges

1. Short range interactions
2. Particle-grid coupling

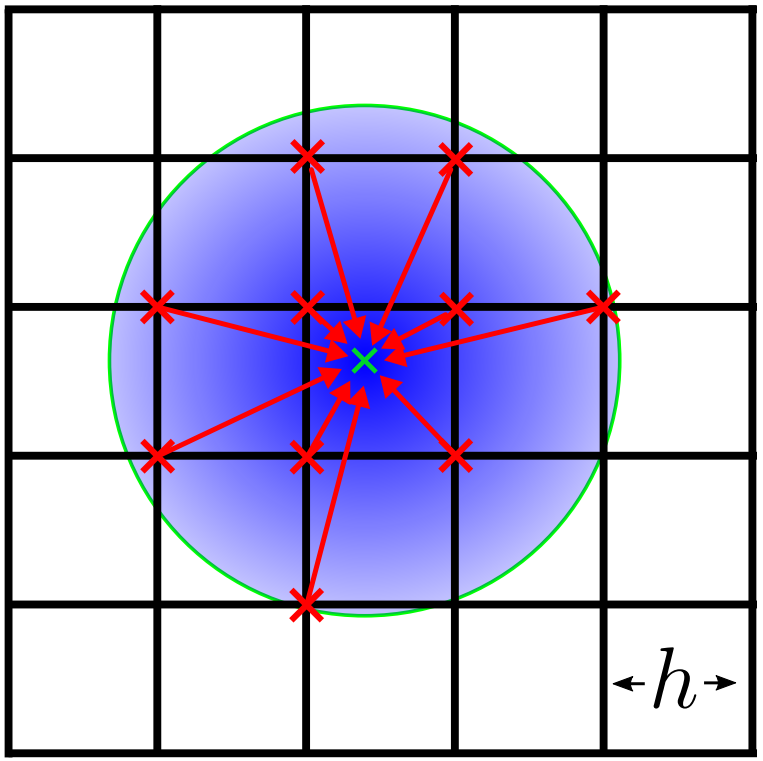
Spreading ($\mathcal{S}$):

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F} := \sum_i \boldsymbol{F}_i \delta_a(\boldsymbol{r} - \boldsymbol{q}_i)$$

$\delta_a(\boldsymbol{r}) := \phi(r_x)\phi(r_y)\phi(r_z) \rightarrow$ Smeared delta

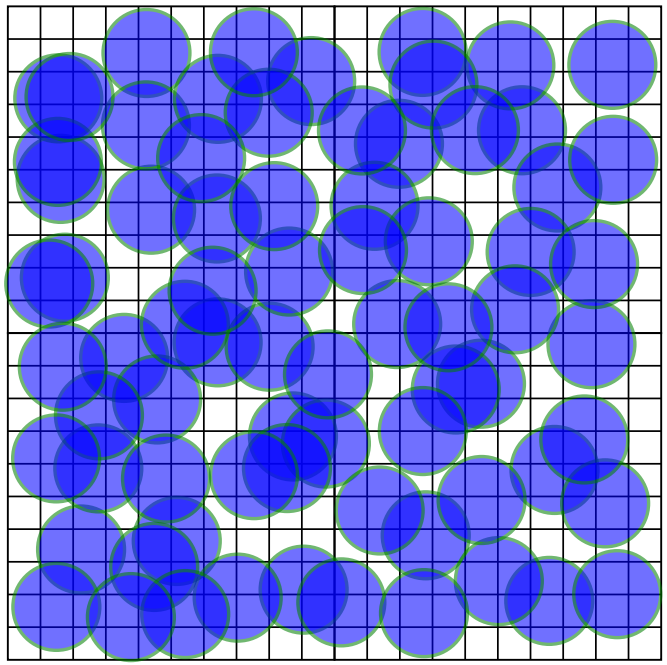
Interpolation ($\mathcal{J}$):

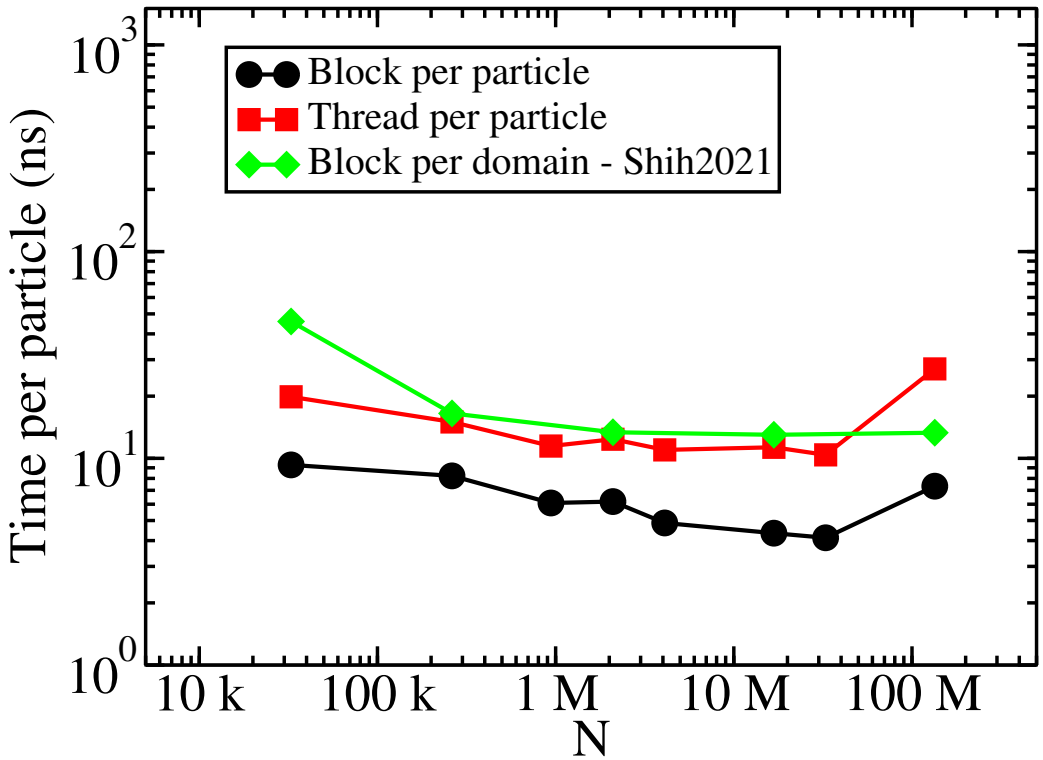
$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r}) \approx \sum_n \boldsymbol{v}_n \delta_a(\boldsymbol{r}_n - \boldsymbol{q}_i) h^3$$



Particle-grid coupling

Performance



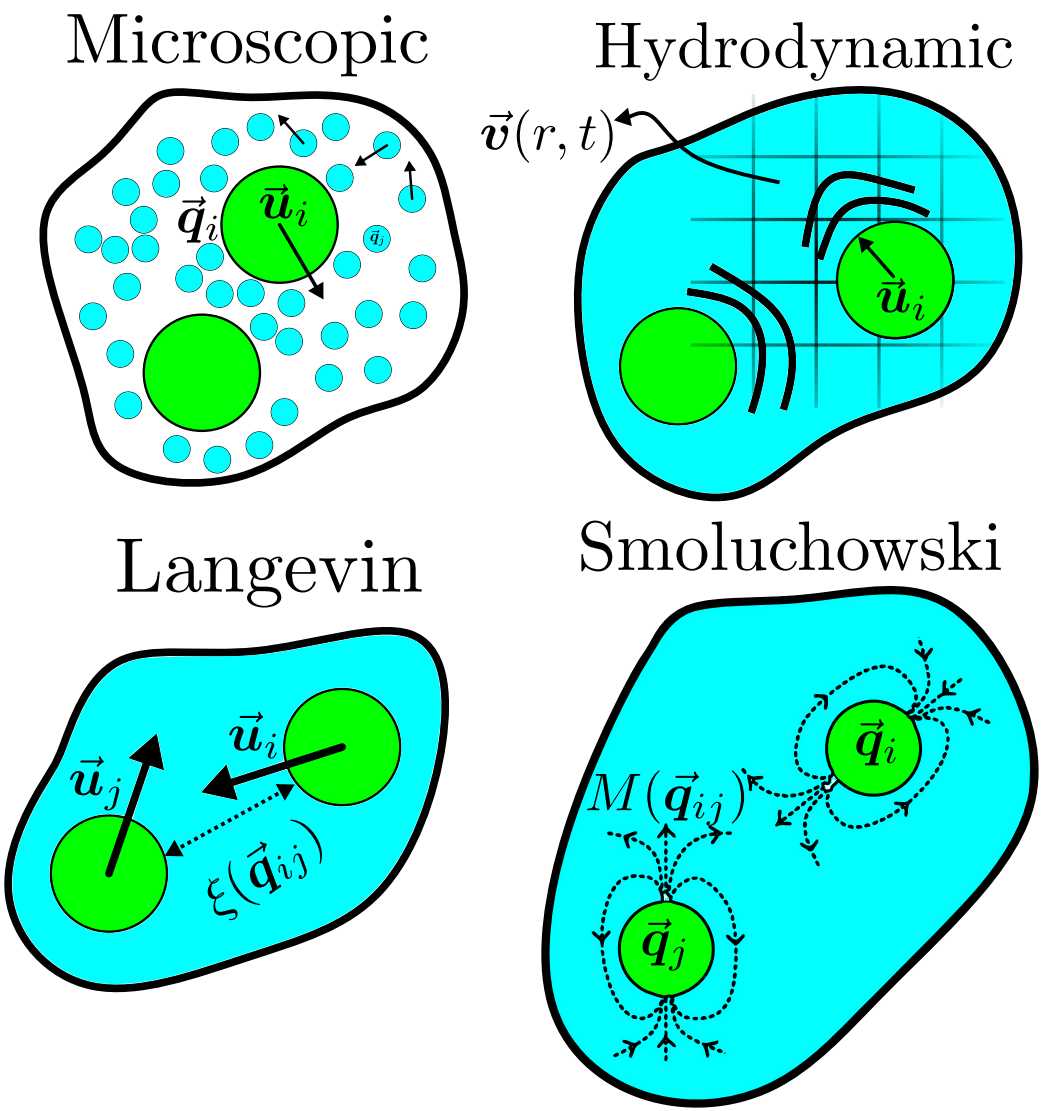


3D random distribution. Ran with a single RTX2080Ti GPU.

Talk outline

Hydrodynamics

Algorithms for Brownian Dynamics with Hydrodynamic Interactions (BDHI)



Complex fluids

The spatio-temporal landscape of numerical techniques.

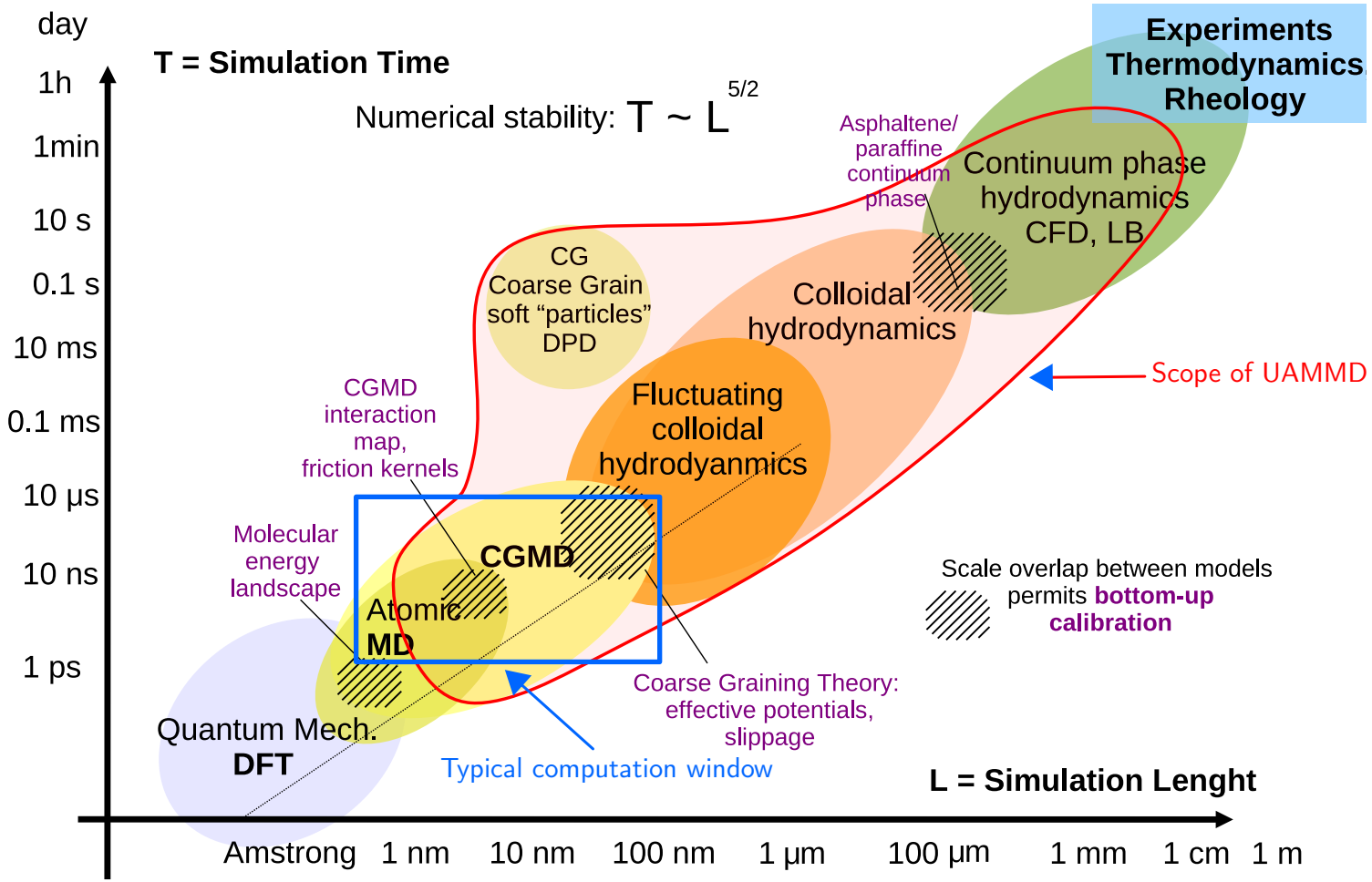


Figure: Figure courtesy of Rafael Delgado-Buscalioni.

Complex fluids

The spatio-temporal landscape of numerical techniques.

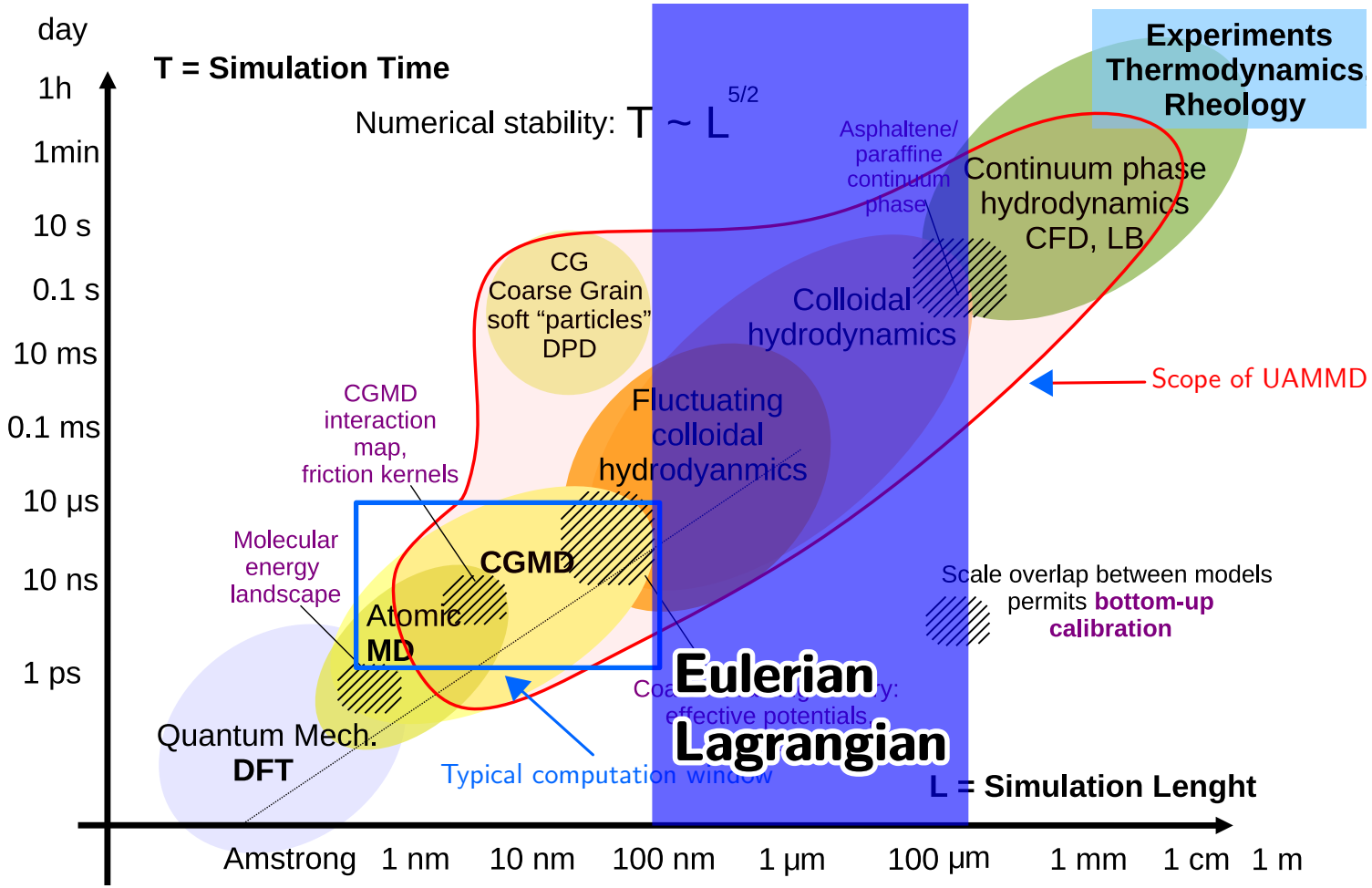


Figure: Figure courtesy of Rafael Delgado-Buscalioni.

$$d\boldsymbol{q} = \boldsymbol{\mathcal{M}}\boldsymbol{F}dt + \sqrt{2k_BT\boldsymbol{\mathcal{M}}}d\widetilde{\boldsymbol{W}} + k_BT\boldsymbol{\partial}_{\boldsymbol{q}} \cdot \boldsymbol{\mathcal{M}}dt.$$

- $\boldsymbol{\mathcal{M}}(\boldsymbol{q})$:
  Mobility tensor. Configuration- and BC-dependent

Smoluchowski level

Brownian Dynamics with Hydrodynamics Interactions (BDHI)

$$d\boldsymbol{q} = \boldsymbol{\mathcal{M}}\boldsymbol{F}dt + \sqrt{2k_BT\boldsymbol{\mathcal{M}}}d\widetilde{\boldsymbol{W}} + k_BT\boldsymbol{\partial_q}\cdot\boldsymbol{\mathcal{M}}dt.$$

- $\boldsymbol{\mathcal{M}}(\boldsymbol{q})$:
  Mobility tensor. Configuration- and BC-dependent
- $d\widetilde{\boldsymbol{W}}$: Brownian motion

$$d\boldsymbol{q} = \boldsymbol{\mathcal{M}}\boldsymbol{F}dt + \sqrt{2k_BT\boldsymbol{\mathcal{M}}}d\widetilde{\boldsymbol{W}} + k_BT\boldsymbol{\partial_q}\cdot\boldsymbol{\mathcal{M}}dt.$$

- $\boldsymbol{\mathcal{M}}(\boldsymbol{q})$:
  Mobility tensor. Configuration- and BC-dependent
- $d\widetilde{\boldsymbol{W}}$: Brownian motion
- $k_BT\boldsymbol{\partial_q}\cdot\boldsymbol{\mathcal{M}}dt$:
  Thermal drift. Zero depending on the BCs

$$d\boldsymbol{q} = \boldsymbol{\mathcal{M}}\boldsymbol{F}dt + \sqrt{2k_BT\boldsymbol{\mathcal{M}}}d\widetilde{\boldsymbol{W}} + k_BT\boldsymbol{\partial}_{\boldsymbol{q}} \cdot \boldsymbol{\mathcal{M}}dt.$$

Problems

$$d\boldsymbol{q} = \boldsymbol{\mathcal{M}}\boldsymbol{F}dt + \sqrt{2k_BT\boldsymbol{\mathcal{M}}}d\widetilde{\boldsymbol{W}} + k_BT\boldsymbol{\partial_q}\cdot\boldsymbol{\mathcal{M}}dt.$$

Problems

- $\boldsymbol{\mathcal{M}}\boldsymbol{F}$: Matrix-vector multiplication is $O(N^2)$

$$d\boldsymbol{q} = \boldsymbol{\mathcal{M}}\boldsymbol{F}dt + \sqrt{2k_BT\boldsymbol{\mathcal{M}}}d\widetilde{\boldsymbol{W}} + k_BT\boldsymbol{\partial_q}\cdot\boldsymbol{\mathcal{M}}dt.$$

Problems

- $\boldsymbol{\mathcal{M}}\boldsymbol{F}$: Matrix-vector multiplication is $O(N^2)$
- $\sqrt{\boldsymbol{\mathcal{M}}}$: Quite expensive in general. $O(N^{[2.25-3]})$

$$d\boldsymbol{q} = \boldsymbol{\mathcal{M}}\boldsymbol{F}dt + \sqrt{2k_BT\boldsymbol{\mathcal{M}}}d\widetilde{\boldsymbol{W}} + k_BT\boldsymbol{\partial_q}\cdot\boldsymbol{\mathcal{M}}dt.$$

Problems

- $\boldsymbol{\mathcal{M}}\boldsymbol{F}$: Matrix-vector multiplication is $O(N^2)$
- $\sqrt{\boldsymbol{\mathcal{M}}}$: Quite expensive in general. $O(N^{[2.25-3]})$
- $\boldsymbol{\mathcal{M}}$ might be unknown/not analytical.

Stokes equations

$$\nabla\pi - \eta\nabla^2\boldsymbol{v} = \boldsymbol{f} + \nabla\cdot\mathcal{Z}$$
$$\nabla\cdot\boldsymbol{v} = 0$$

Stokes equations

$$\nabla\pi - \eta\nabla^2\boldsymbol{v} = \boldsymbol{f} + \nabla\cdot\mathcal{Z}$$
$$\nabla\cdot\boldsymbol{v} = 0$$

Fluid forcing

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F}$$

Particle velocity

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r})$$

Stokes equations

$$\nabla\pi - \eta\nabla^2\boldsymbol{v} = \boldsymbol{f} + \nabla\cdot\mathcal{Z}$$
$$\nabla\cdot\boldsymbol{v} = 0$$

Fluid forcing

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F}$$

Particle velocity

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r})$$

Green formalism

$$\boldsymbol{v}(\boldsymbol{r}) = \mathcal{G}\tilde{\boldsymbol{f}}$$

Stokes equations

$$\nabla\pi - \eta\nabla^2\boldsymbol{v} = \boldsymbol{f} + \nabla\cdot\mathcal{Z}$$
$$\nabla\cdot\boldsymbol{v} = 0$$

Fluid forcing

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F}$$

Particle velocity

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r})$$

Green formalism

$$\boldsymbol{v}(\boldsymbol{r}) = \mathcal{G}\tilde{\boldsymbol{f}} := \int \mathcal{G}(\boldsymbol{r},\boldsymbol{r}')\tilde{\boldsymbol{f}}(\boldsymbol{r}')d\boldsymbol{r}'$$

Stokes equations

$$\nabla\pi - \eta\nabla^2\boldsymbol{v} = \boldsymbol{f} + \nabla\cdot\mathcal{Z}$$
$$\nabla\cdot\boldsymbol{v} = 0$$

Fluid forcing

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F}$$

Particle velocity

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r})$$

Green formalism

$$\boldsymbol{v}(\boldsymbol{r}) = \mathcal{G}\tilde{\boldsymbol{f}} := \int \mathcal{G}(\boldsymbol{r},\boldsymbol{r}')\tilde{\boldsymbol{f}}(\boldsymbol{r}')d\boldsymbol{r}'$$

Force Coupling Method (FCM)

$$\frac{d\boldsymbol{q}_i}{dt} = \boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\mathcal{G}(\mathcal{S}\boldsymbol{F} + \nabla\cdot\mathcal{Z})$$

Stokes equations

$$\nabla\pi - \eta\nabla^2\boldsymbol{v} = \boldsymbol{f} + \nabla\cdot\mathcal{Z}$$
$$\nabla\cdot\boldsymbol{v} = 0$$

Fluid forcing

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F}$$

Particle velocity

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r})$$

Green formalism

$$\boldsymbol{v}(\boldsymbol{r}) = \mathcal{G}\tilde{\boldsymbol{f}} := \int \mathcal{G}(\boldsymbol{r},\boldsymbol{r}')\tilde{\boldsymbol{f}}(\boldsymbol{r}')d\boldsymbol{r}'$$

Force Coupling Method (FCM)

$$\frac{d\boldsymbol{q}_i}{dt} = \boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\mathcal{G}(\mathcal{S}\boldsymbol{F} + \nabla\cdot\mathcal{Z})$$

$$\begin{aligned}
\boldsymbol{\mathcal{M}} &= \mathcal{J}\mathcal{G}\mathcal{S} \\
\boldsymbol{\mathcal{M}}^{1/2} &= \mathcal{J}\mathcal{G}\nabla\cdot \\
\boldsymbol{\partial}_{\boldsymbol{q}}\cdot\boldsymbol{\mathcal{M}} &= 0 \leftarrow \text{Incompressible}
\end{aligned}$$

Stokes equations

$$\nabla\pi - \eta\nabla^2\boldsymbol{v} = \boldsymbol{f} + \nabla\cdot\mathcal{Z}$$
$$\nabla\cdot\boldsymbol{v} = 0$$

Fluid forcing

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F}$$

Particle velocity

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r})$$

Green formalism

$$\boldsymbol{v}(\boldsymbol{r}) = \mathcal{G}\tilde{\boldsymbol{f}} := \int \mathcal{G}(\boldsymbol{r},\boldsymbol{r}')\tilde{\boldsymbol{f}}(\boldsymbol{r}')d\boldsymbol{r}'$$

Force Coupling Method (FCM)

$$\frac{d\boldsymbol{q}_i}{dt} = \boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\mathcal{G}(\mathcal{S}\boldsymbol{F} + \nabla\cdot\mathcal{Z})$$

$$\boldsymbol{\mathcal{M}}_{ij} = \iint \delta_a(\boldsymbol{r}-\boldsymbol{q}_j)\mathcal{G}(\boldsymbol{r},\boldsymbol{r}')\delta_a(\boldsymbol{q}_i-\boldsymbol{r}')d\boldsymbol{r}'d\boldsymbol{r}$$

Stokes equations

$$\nabla\pi - \eta\nabla^2\boldsymbol{v} = \boldsymbol{f} + \nabla\cdot\mathcal{Z}$$
$$\nabla\cdot\boldsymbol{v} = 0$$

Fluid forcing

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F}$$

Particle velocity

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r})$$

Green formalism

$$\boldsymbol{v}(\boldsymbol{r}) = \mathcal{G}\tilde{\boldsymbol{f}} := \int \mathcal{G}(\boldsymbol{r},\boldsymbol{r}')\tilde{\boldsymbol{f}}(\boldsymbol{r}')d\boldsymbol{r}'$$

Force Coupling Method (FCM)

$$\frac{d\boldsymbol{q}_i}{dt} = \boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\mathcal{G}(\mathcal{S}\boldsymbol{F} + \nabla\cdot\mathcal{Z})$$

$$\mathcal{G} := -\eta^{-1}\nabla^{-2}\left(\mathbb{I} - \nabla\nabla^{-2}\nabla\right)$$

Stokes equations

$$\nabla\pi - \eta\nabla^2\boldsymbol{v} = \boldsymbol{f} + \nabla\cdot\mathcal{Z}$$
$$\nabla\cdot\boldsymbol{v} = 0$$

Fluid forcing

$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F}$$

Particle velocity

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r})$$

Green formalism

$$\boldsymbol{v}(\boldsymbol{r}) = \mathcal{G}\tilde{\boldsymbol{f}} := \int \mathcal{G}(\boldsymbol{r},\boldsymbol{r}')\tilde{\boldsymbol{f}}(\boldsymbol{r}')d\boldsymbol{r}'$$

Force Coupling Method (FCM)

$$\frac{d\boldsymbol{q}_i}{dt} = \boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\mathcal{G}(\mathcal{S}\boldsymbol{F} + \nabla\cdot\mathcal{Z})$$

$$\mathcal{G} := -\eta^{-1}\nabla^{-2}\left(\mathbb{I} - \nabla\nabla^{-2}\nabla\right)$$

Open boundaries

$$\mathcal{G} \to \boldsymbol{\mathcal{O}}(\boldsymbol{r}) = \frac{1}{8\pi\eta r}\left(\mathbb{I} - \frac{\boldsymbol{r}\otimes\boldsymbol{r}}{r^2}\right)$$

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i} \mathcal{G} (\mathcal{S}\boldsymbol{F} + \nabla \cdot \mathcal{Z})$$

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\mathcal{G}(\mathcal{S}\boldsymbol{F} + \nabla \cdot \mathcal{Z})$$

$\mathfrak{F}$: Fourier transform

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i} \quad \mathcal{G}(\ \mathcal{S}\boldsymbol{F} + \nabla \ \cdot \mathcal{Z} \ )$$

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i} \mathcal{G} (\mathcal{S}\boldsymbol{F} + \nabla \cdot \mathcal{Z})$$

$\mathfrak{F}$: Fourier transform

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i} \mathfrak{F}^{-1} \widehat{\mathcal{G}} (\mathfrak{F}\mathcal{S}\boldsymbol{F} + \quad \boldsymbol{k} \cdot \mathcal{Z}_{\boldsymbol{k}})$$

Force Coupling Method

$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i} \mathfrak{F}^{-1} \widehat{\mathcal{G}} (\mathfrak{F} \mathcal{S} \boldsymbol{F} + \boldsymbol{k} \cdot \mathcal{Z}_{\boldsymbol{k}})$$

Triply Periodic

$$\widehat{\mathcal{G}}_{\mathsf{3D}} := \eta^{-1} k^{-2} \left(\mathbb{I} - \frac{\boldsymbol{k} \otimes \boldsymbol{k}}{k^2} \right)$$

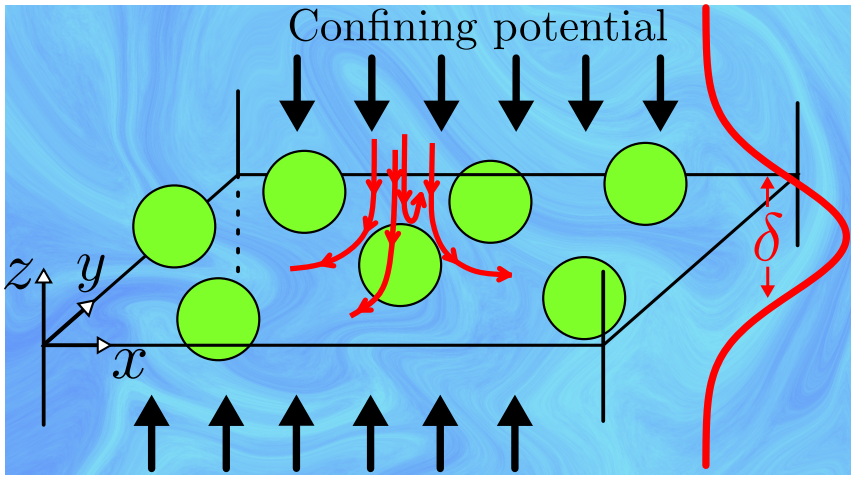
Hydrodynamics in confined geometries

New algorithms for Brownian Dynamics with Hydrodynamic Interactions (BDHI)

Force Coupling Method

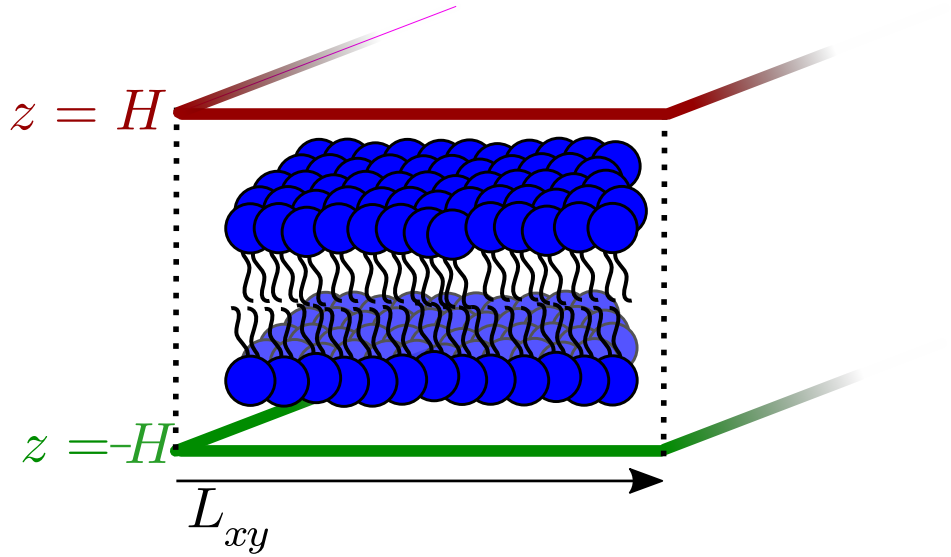
$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i} \mathfrak{F}^{-1} \hat{\mathcal{G}}_{\text{3D}} \left(\mathfrak{F} \mathcal{S} \boldsymbol{F} + \boldsymbol{k} \cdot \mathcal{Z} \right)$$

- Quasi two-dimensional (q2D)



q2D: $\delta \to 0$.
3D fluid, 2D particles

- Doubly Periodic Stokes (DPStokes)



[**Raul P. Pelaez** et. al. JSTAT 2018.] [O. Maxian, **Raul P. Pelaez** et. al. J.

Triply Periodic Electrostatics

Force Coupling Method for the Poisson equation

Poisson

$$\varepsilon_0 \nabla^2 \phi = -f(\boldsymbol{r}) = -\mathcal{S}(\boldsymbol{r})\boldsymbol{Q}$$

Spreading

$$\mathcal{S}(\boldsymbol{r})\boldsymbol{Q} = \sum_i Q_i \delta_a(\boldsymbol{q}_i - \boldsymbol{r})$$

$$\boldsymbol{E}(\boldsymbol{r}) = \partial_{\boldsymbol{r}}\phi \rightarrow \widehat{\boldsymbol{E}} = i\boldsymbol{k}\widehat{\phi}$$

$$\boldsymbol{F}_i = Q_i \mathcal{J}_{\boldsymbol{q}_i} \boldsymbol{E}(\boldsymbol{r})$$

Gaussian sources

$$\delta_a(\boldsymbol{r}) \propto e^{-r^2/(2a^2)}$$

Charges to forces

$$\boldsymbol{F}_i = Q_i \mathcal{J}_{\boldsymbol{q}_i} \mathfrak{F}^{-1} i\boldsymbol{k} \hat{\mathcal{G}}_P \mathfrak{F} \mathcal{S} \boldsymbol{Q}$$

Supervector notation. Charges $\boldsymbol{Q} := \{Q_1, \ldots, Q_N\}$

$$\hat{\mathcal{G}}_P = \left(\varepsilon_0 k^2\right)^{-1}$$

[O. Maxian, **Raul P. Pelaez** et. al. J. Chem. Phys. 2021.]

Talk outline

UAMMD: A CUDA/C++ library for complex fluids.

- Header only and library-like

UAMMD: A CUDA/C++ library for complex fluids.

- Header only and library-like
- Lightweight with minimal dependencies

UAMMD: A CUDA/C++ library for complex fluids.

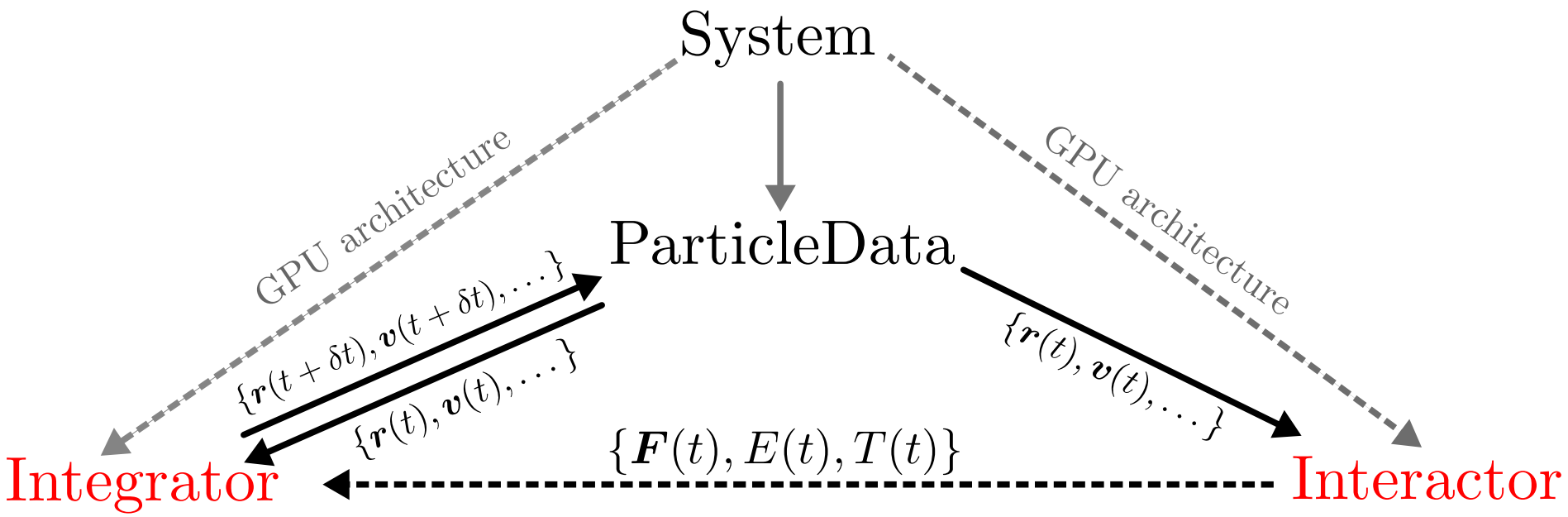
- Header only and library-like
- Lightweight with minimal dependencies
- Hackable

UAMMD: A CUDA/C++ library for complex fluids.

- Header only and library-like
- Lightweight with minimal dependencies
- Hackable
- Focus on hydrodynamics

UAMMD

Conceptual hierarchy

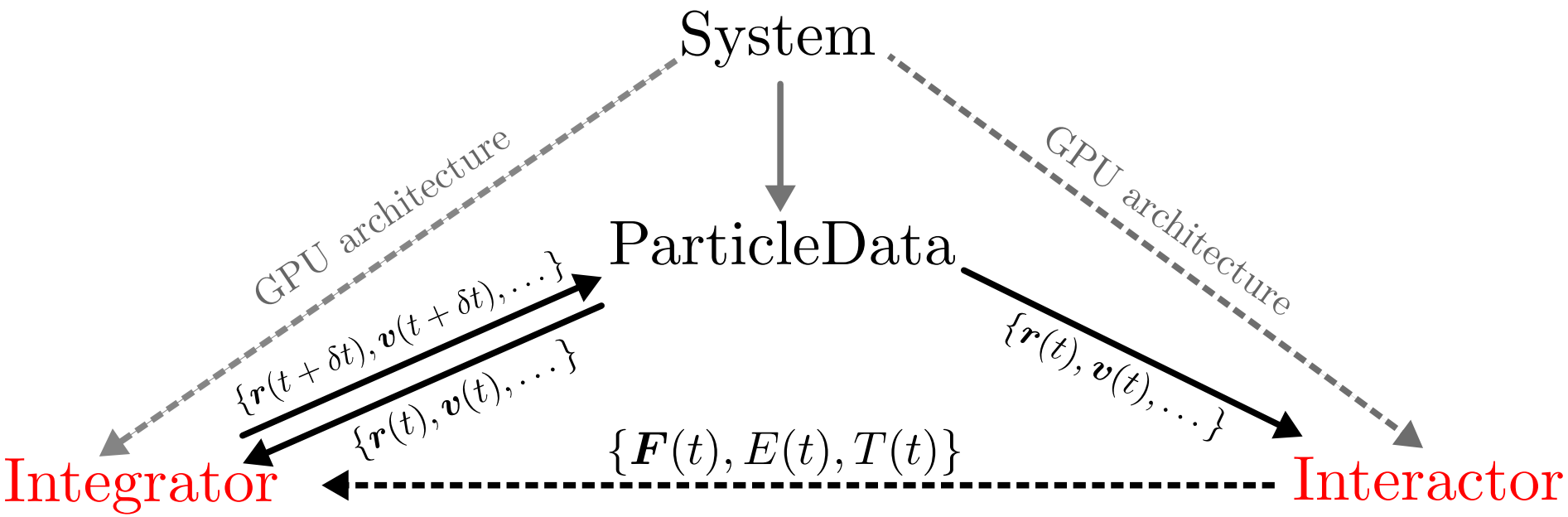


Remember

"Interacting *particles* with a state that *evolves*"

UAMMD

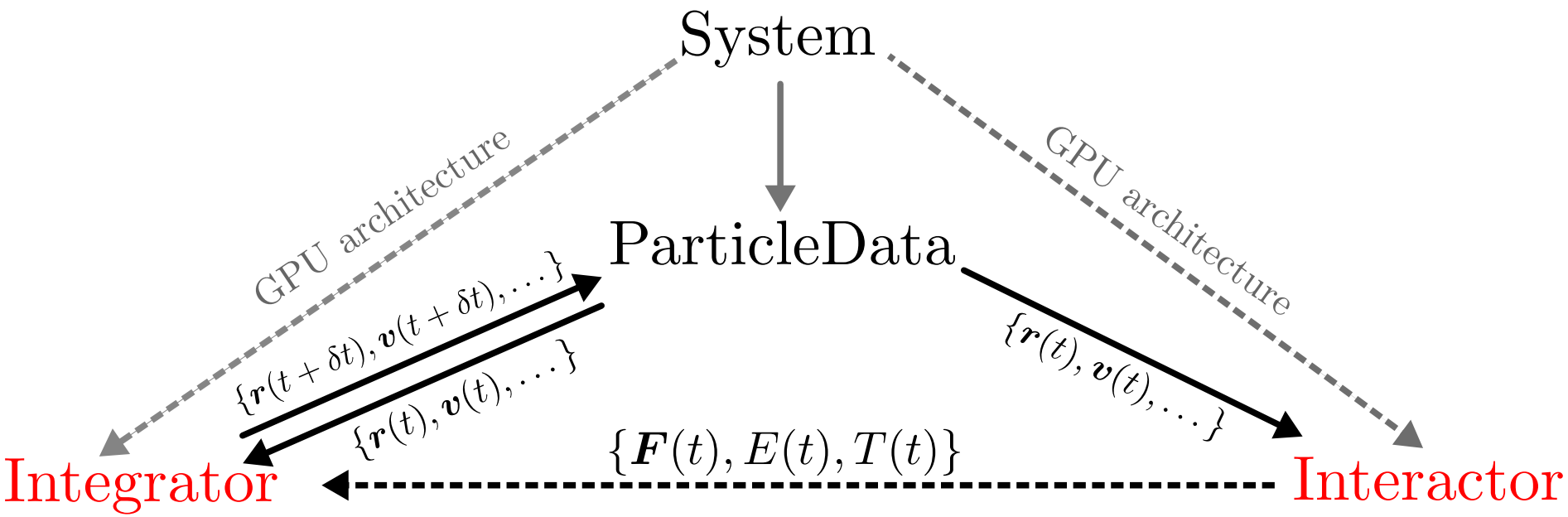
Conceptual hierarchy




```
#include <uammd.cuh>
int main(int argc, char* argv[]){
  auto sys = std::make_shared<System>(argc, argv);
  ...
```

UAMMD

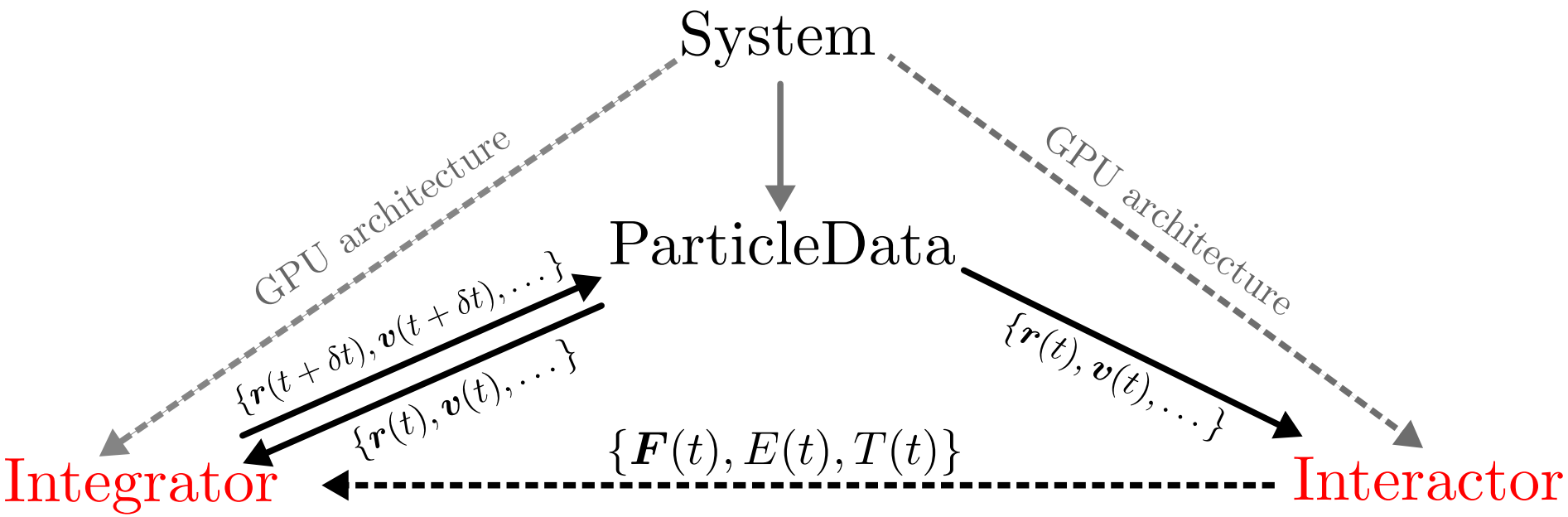
Conceptual hierarchy




```
#include <uammd.cuh>
int main(int argc, char* argv[]){
  auto sys = std::make_shared<System>(argc, argv);
  const int nP = 1e6; //Number of particles
  auto pd = std::make_shared<ParticleData>(nP, sys);
  ...
```

UAMMD

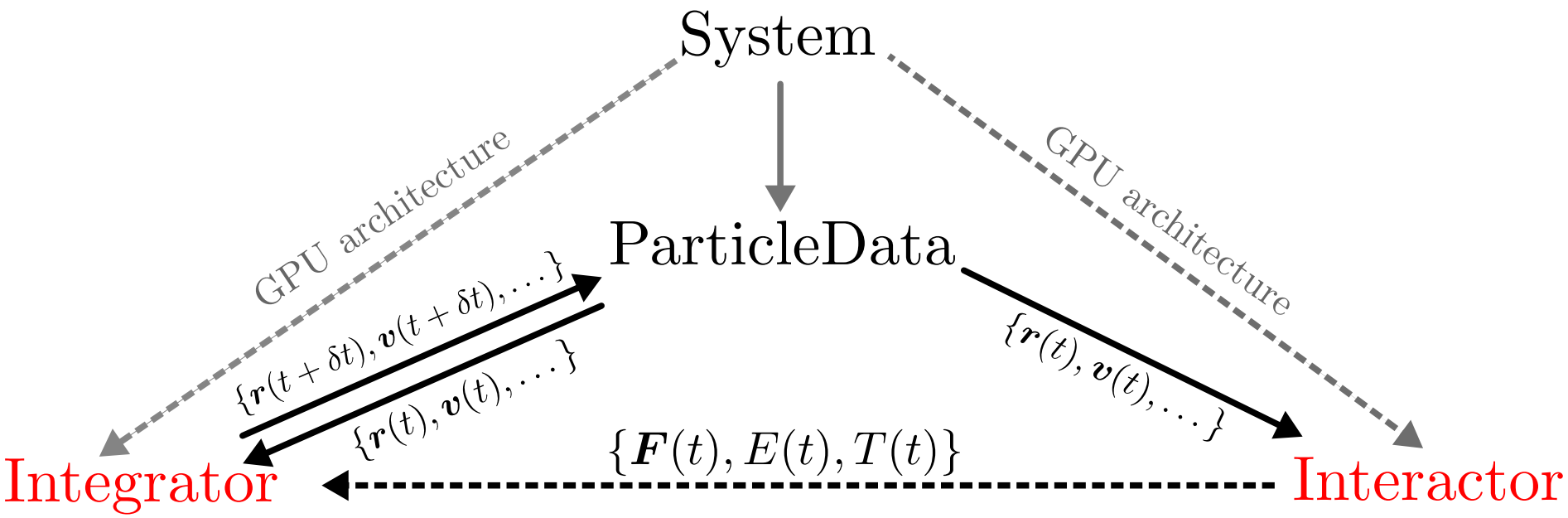
Conceptual hierarchy




```
...
auto pos = pd->getPos(access::cpu, access::write);
pos[0] = {1,1,1,0}; // x,y,x, color (type)
...
```

UAMMD

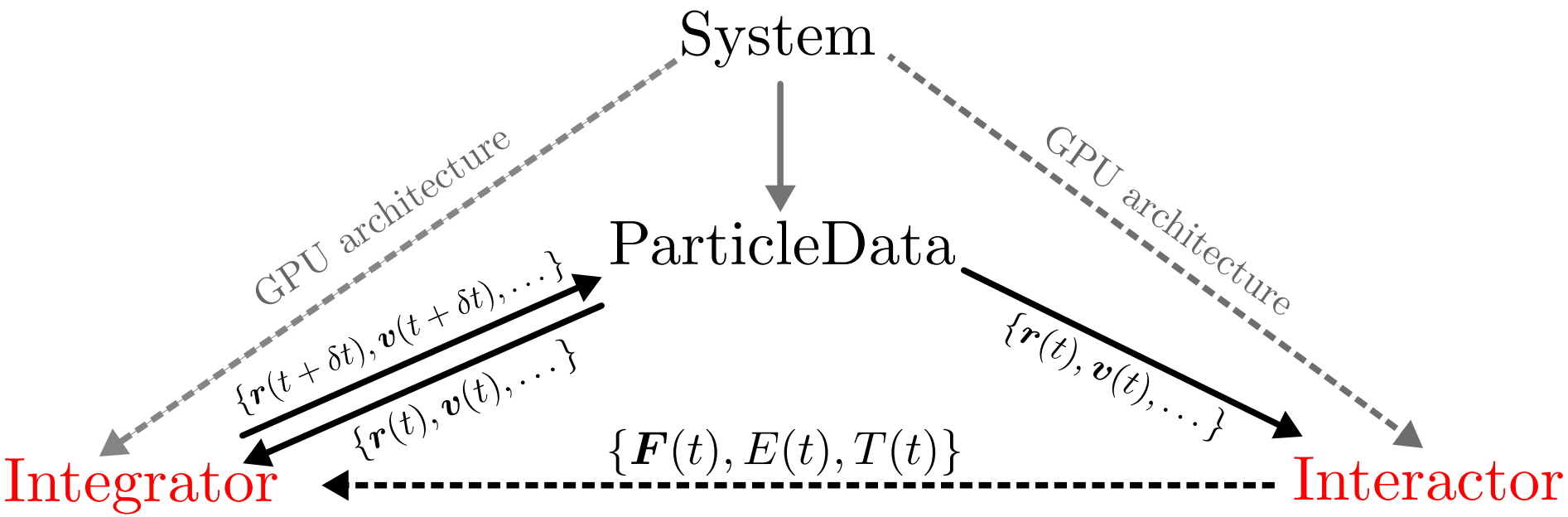
Conceptual hierarchy




```
...
auto bd = std::make_shared<BDHI::FCM>(pd, /*Parameters*/);
bd->forwardTime(); //Exposed by every Integrator
...
```

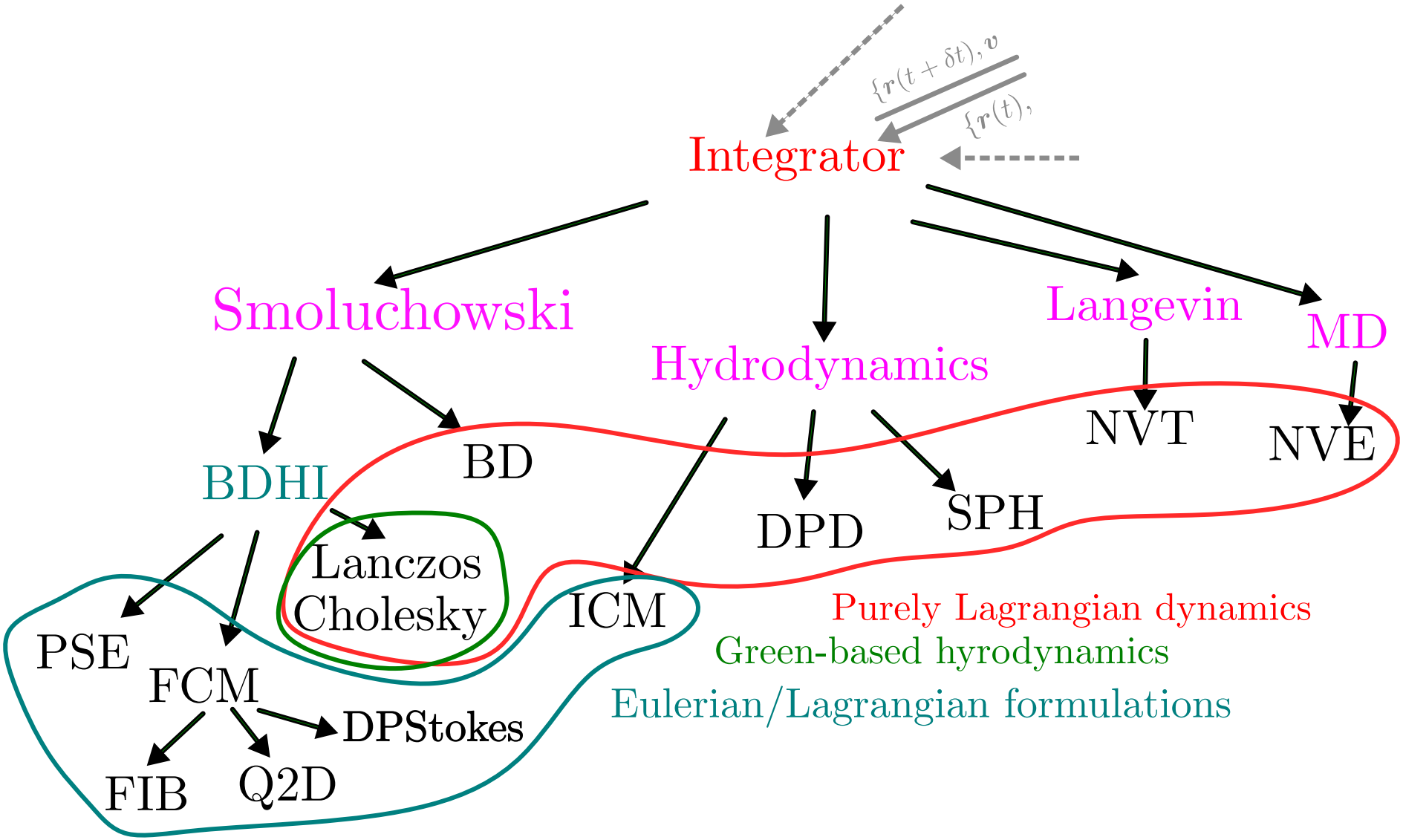
UAMMD

Conceptual hierarchy

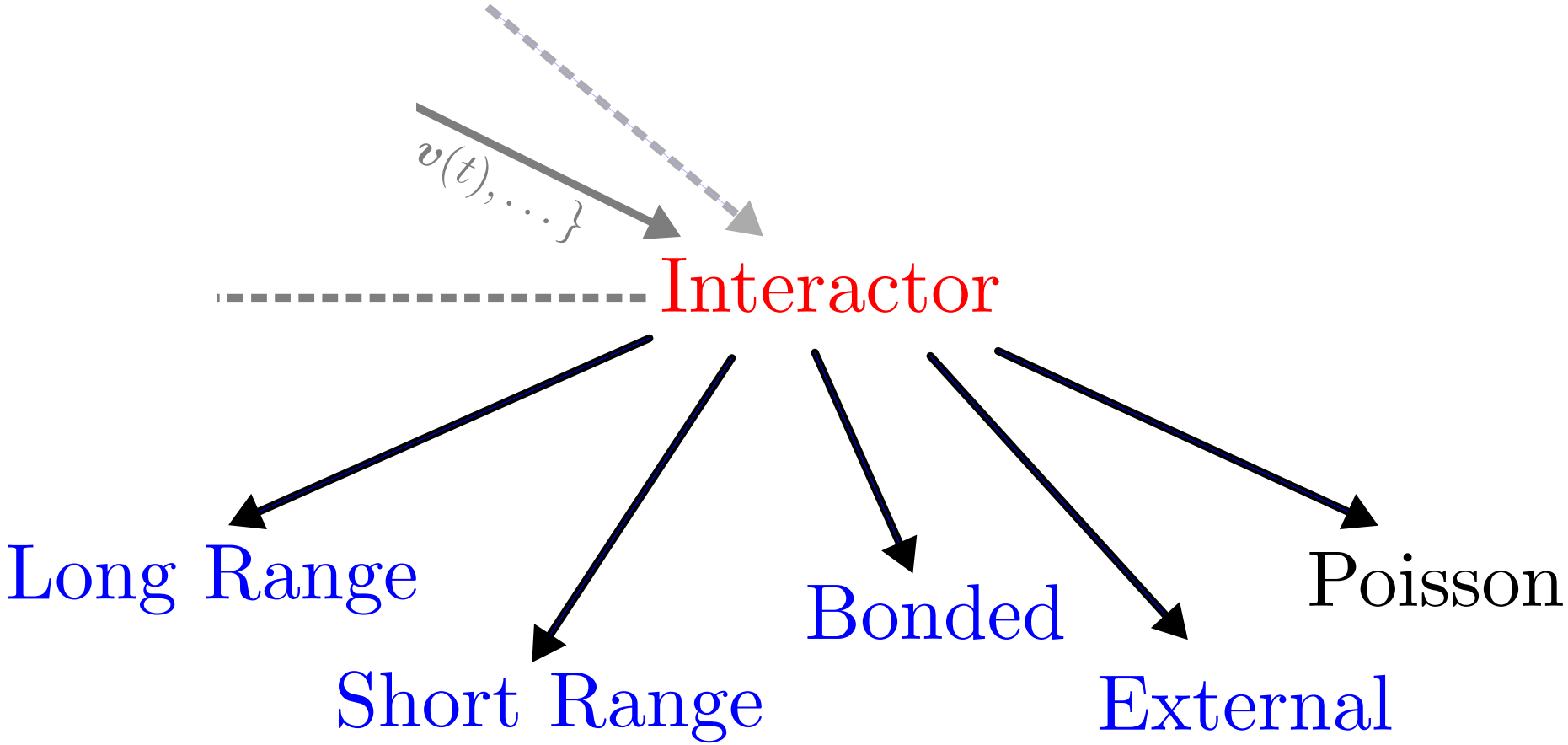



```
...
auto elec = std::make_shared<Poisson>(pd,/*Parameters*/);
elec->sum({.force=true, .energy=false, .virial=false});
//Exposed by every Integrator
bd->addInteractor(elec);
bd->forwardTime();
...
```

Available solvers



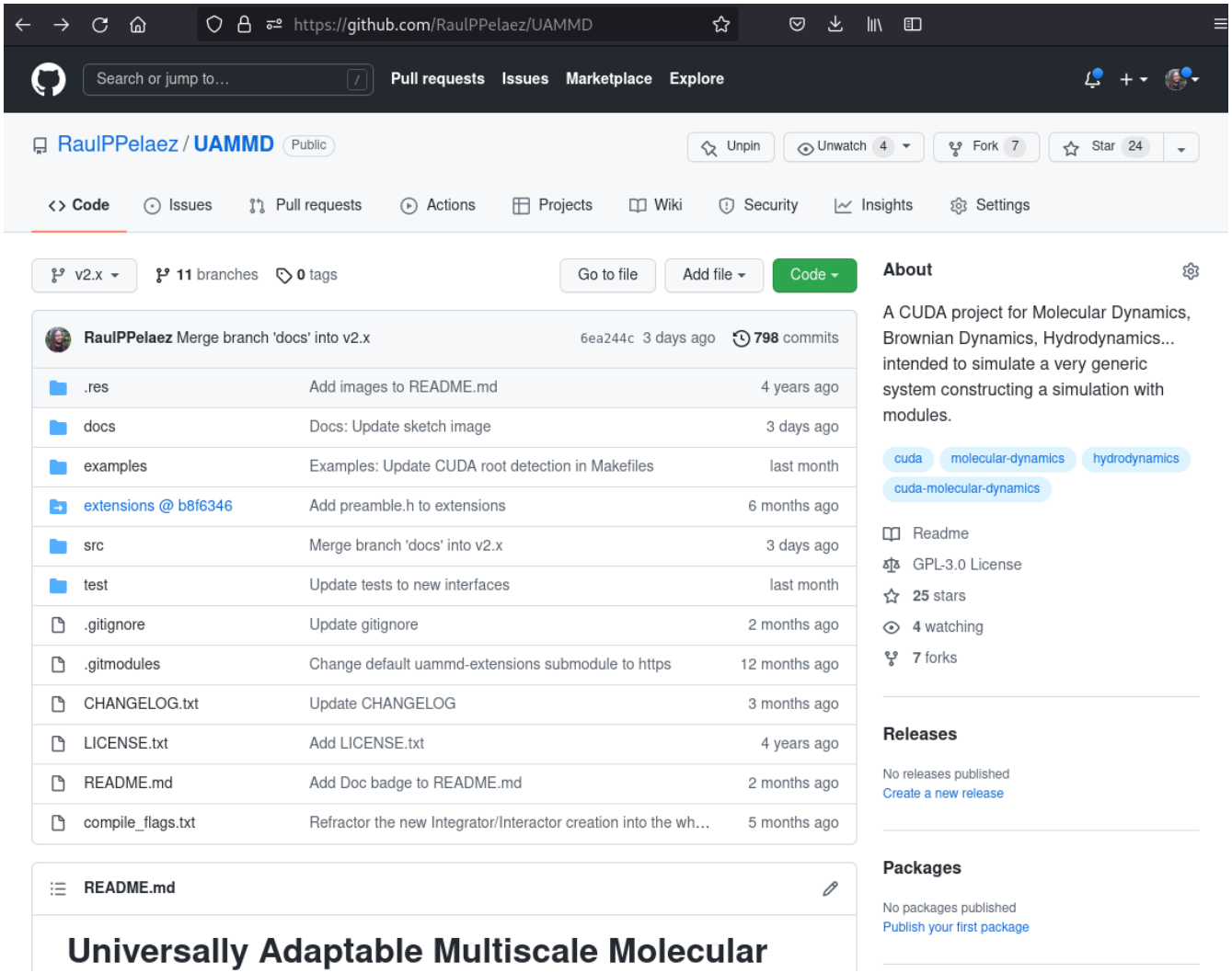
Available interactions



UAMMD's online presence

Git repository

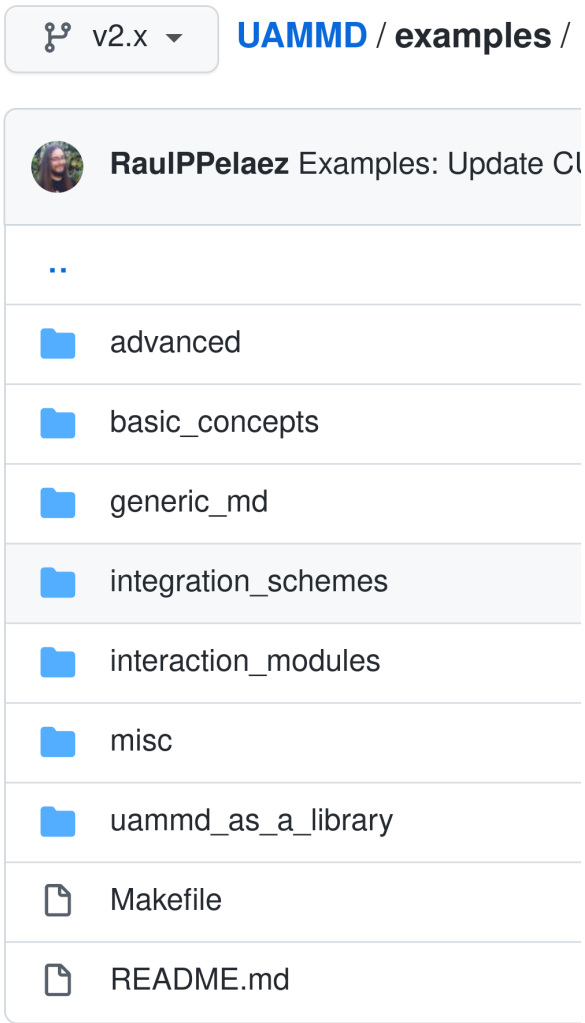
https://github.com/RaulPPelaez/uammd



UAMMD's online presence

Git repository

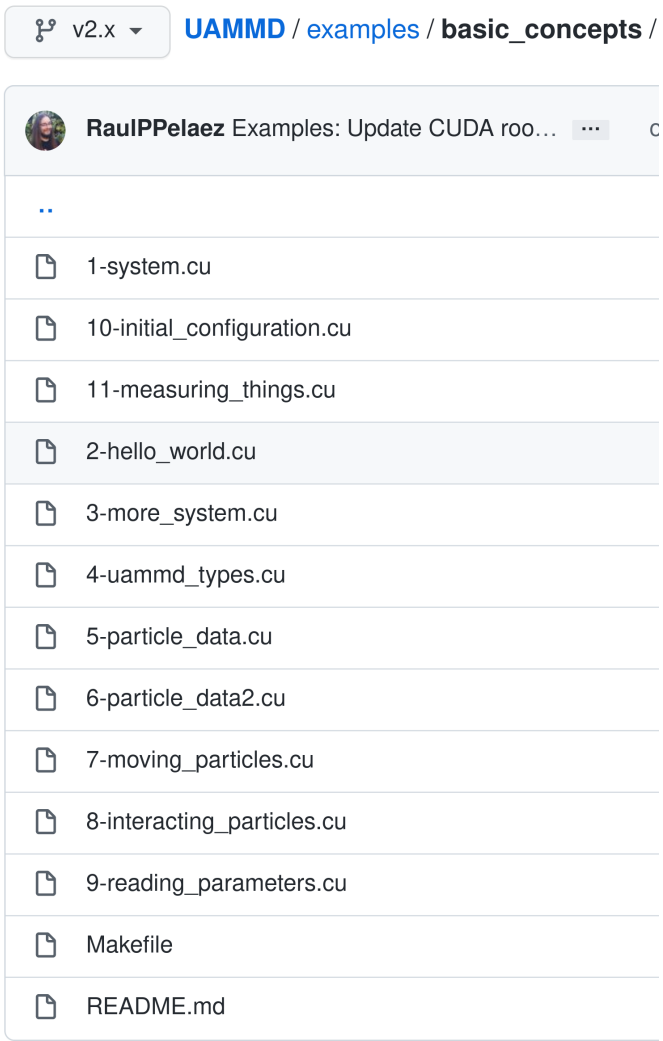
https://github.com/RaulPPelaez/uammd

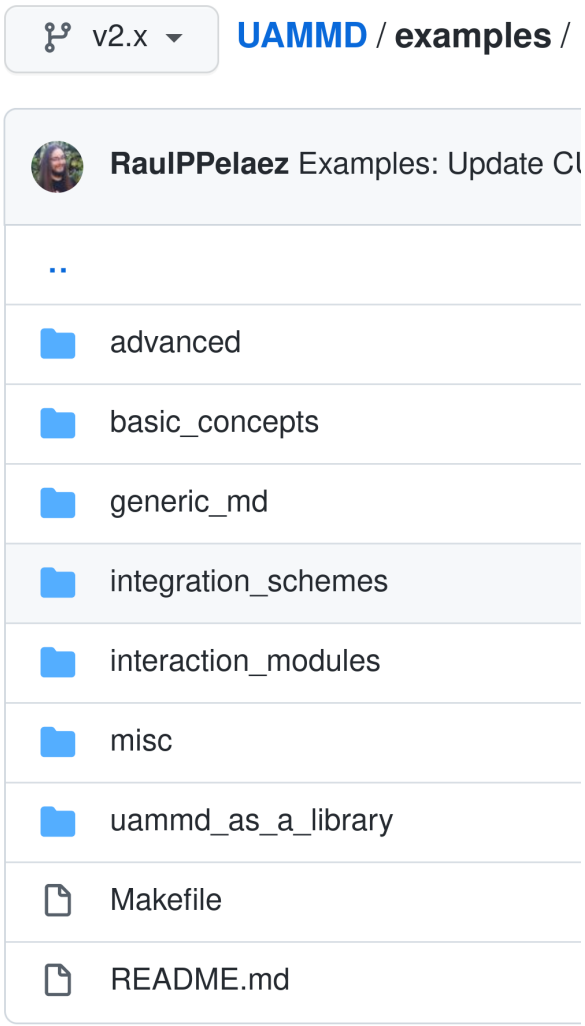


UAMMD's online presence

Git repository

https://github.com/RaulPPelaez/uammd





UAMMD's online presence

Git repository

https://github.com/RaulPPelaez/uammd

v2.x **UAMMD** / examples / **basic_concepts** /

RaulPPelaez Examples: Update CUDA roo…

- ..
- 1-system.cu
- 10-initial_configuration.cu
- 11-measuring_things.cu
- 2-hello_world.cu
- 3-more_system.cu
- 4-uammd_types.cu
- 5-particle_data.cu
- 6-particle_data2.cu
- 7-moving_particles.cu
- 8-interacting_particles.cu
- 9-reading_parameters.cu
- Makefile
- README.md

v2.x

UAMMD / examples / basic_concepts / **2-hello_world.cu**

RaulPPelaez Update examples — **History**

1 contributor

36 lines (35 sloc) | 2.2 KB

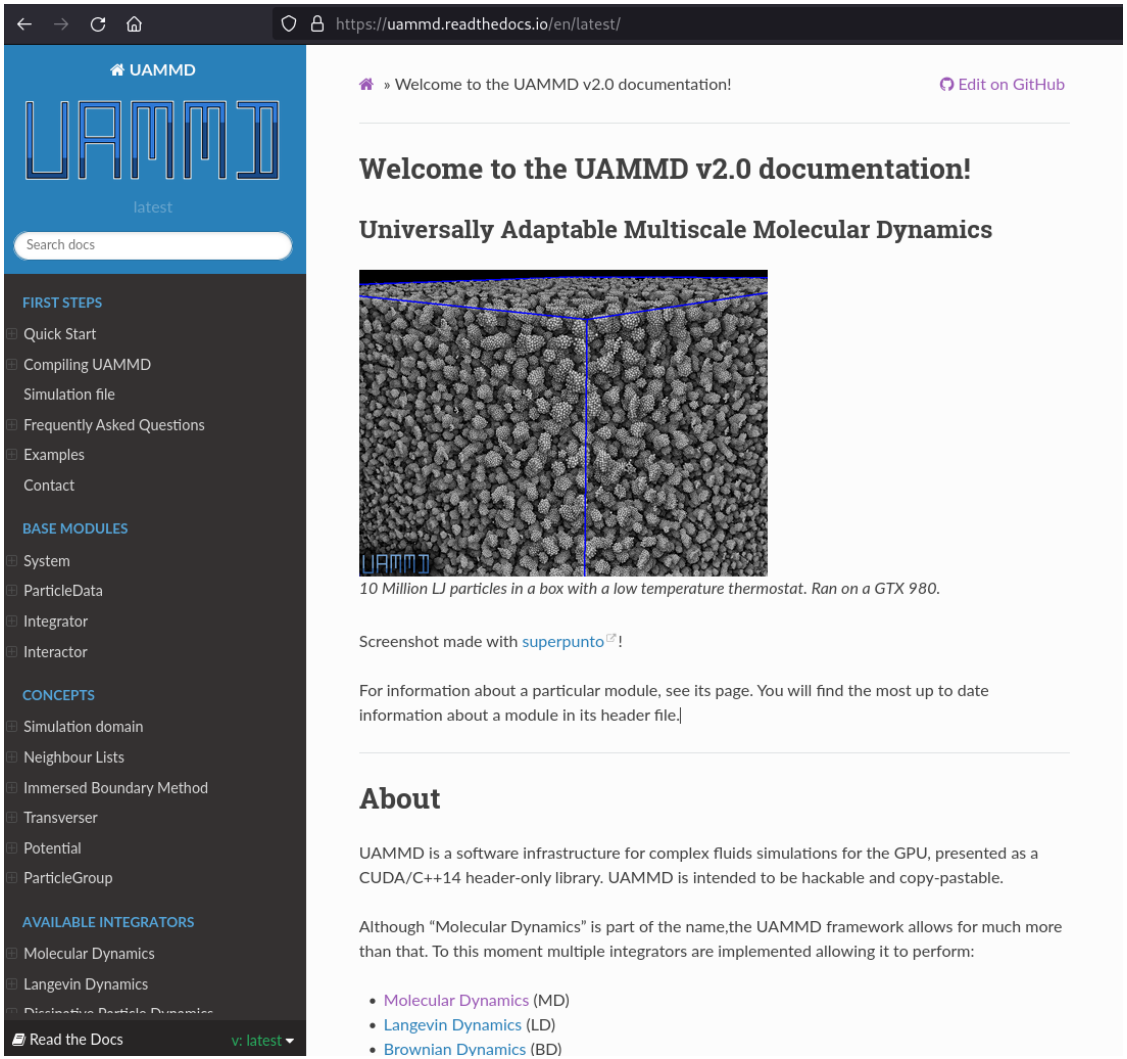
```
/* Raul P. Pelaez 2021
   Hello world with UAMMD
   We will start from the previous example and see the first utility provided by System, th
   System::log allows to, well, log messages with several levels of priority.
   Internally UAMMD modules use this utility to print everything ranging from informational
   In this example we will use System::log to print a simple message along with the date.
 */

//uammd.cuh is the basic uammd include containing, among other things, the System struct.
#include<uammd.cuh>
#include<ctime> //For time and ctime
using namespace uammd;
//The main function will initialize an UAMMD environment, print a message, then destroy it
int main(int argc, char* argv[]){
  //Initialize System
  auto sys = std::make_shared<System>(argc, argv);
  //Unles something goes wrong System creation logs messages using level MESSAGE, which pri
  //There are a lot more levels to choose from, each associated with a number. From highest
  //CRITICAL=0, ERROR, EXCEPTION, WARNING, MESSAGE, STDERR, STDOUT,DEBUG, DEBUG1, DEBUG2, D
  //For example, MESSAGE is associated with log level number 5.
  //Lets print something using the MESSAGE level:
  sys->log<System::MESSAGE>("Hello from UAMMD");
  //Lets also print todays date, this time as a WARNING:
  auto currentTime = time(nullptr);
  //Notice that System::log works as C's printf, with a format string and then arguments.
  sys->log<System::WARNING>("Current time is: %s", ctime(&currentTime));
  //The maximum log level printed can be controled through the MAXLOGLEVEL compile macro (w
  //The default is 5, which will print up to MESSAGE.
  //The special level CRITICAL will terminate the execution of the program with an error co
  //Unless the log level STDOUT is used, all messages will be issued to stderr.
  //Since it is known at compile time, any log calls with levels above the maximum one will
  // no performance penalty
  //Destroy the UAMMD environment and exit
  sys->finish();
  return 0;
}
```

UAMMD's online presence

Documentation

https://uammd.readthedocs.io

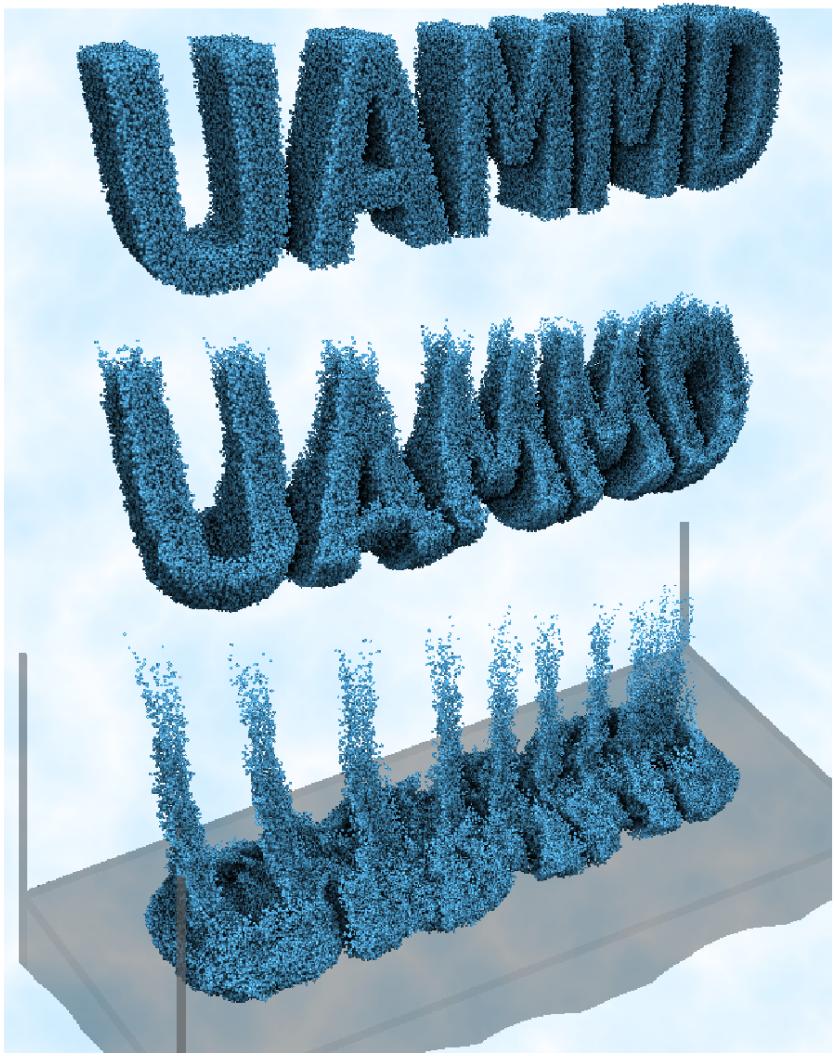


Talk outline

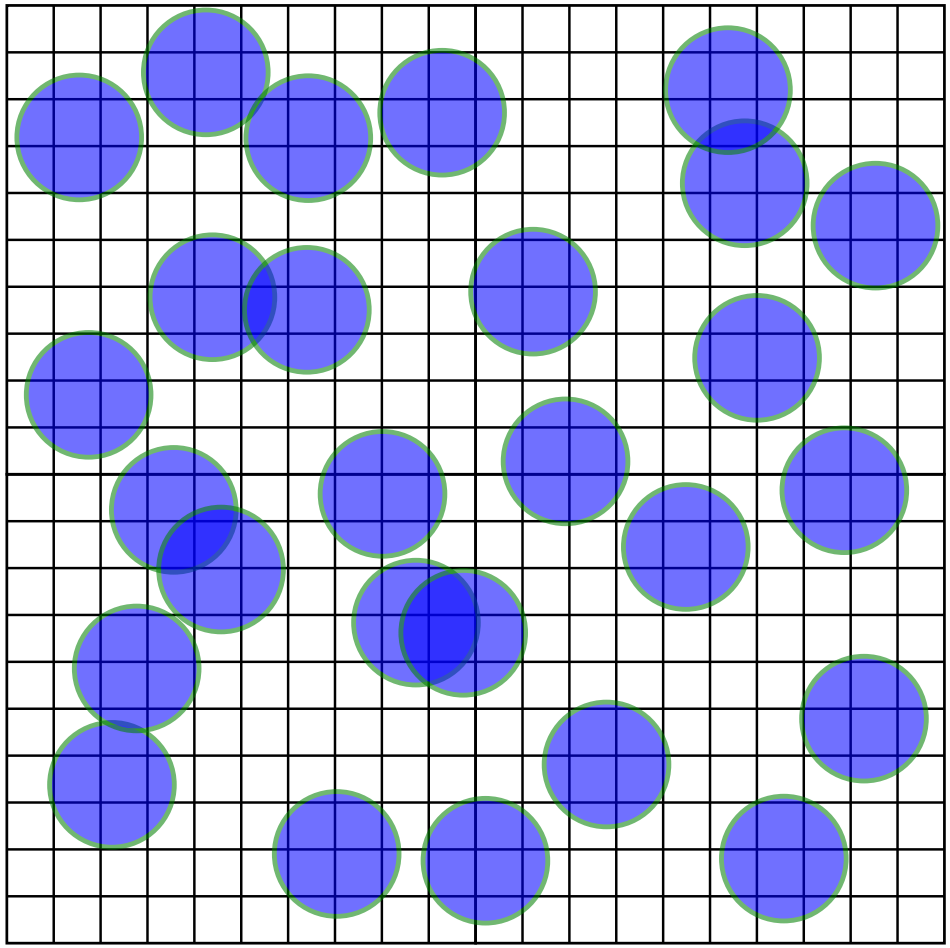
Lets clone

```
$ git clone git@github.com:RaulPPelaez/2022Bilbao
```

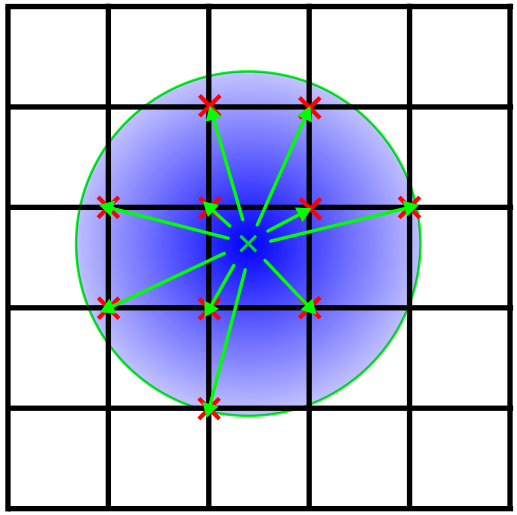
The falling logo



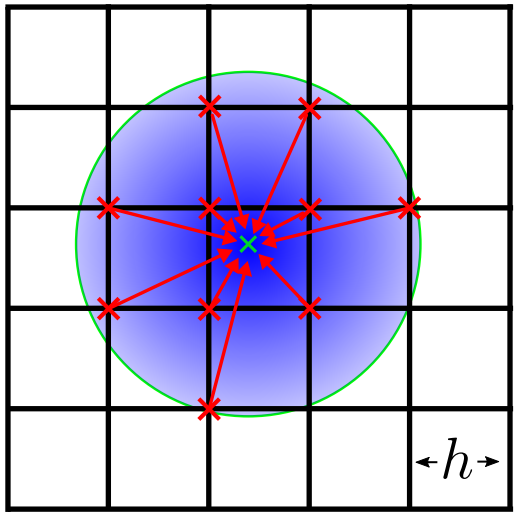
Particle-grid coupling



$$\boldsymbol{f}(\boldsymbol{r}) = \mathcal{S}(\boldsymbol{r})\boldsymbol{F}$$



$$\boldsymbol{u}_i = \mathcal{J}_{\boldsymbol{q}_i}\boldsymbol{v}(\boldsymbol{r})$$



Thanks for your attention!