BOUNDARY CONDITIONS AND PARTICLES
Why boundary conditions are important and difficult to treat
Why boundary conditions are important and difficult to treat
Boundary conditions and particle methods: a non-standard issue, because particles make sense only as a collection of overlapping points.

A single particle on a boundary not enough to enforce a given boundary condition at that point.
IMPORTANT to allow the flow to:

Boundary conditions and particle methods: a non-standard issue, because particles make sense only as a collection of overlapping points.

A single particle on a boundary not enough to enforce a given boundary condition at that point.
IMPORTANT to allow the flow to:

• leave the computational box without non-physical artifacts (numerical issue)

Boundary conditions and particle methods: a non-standard issue, because particles make sense only as a collection of overlapping points.

A single particle on a boundary not enough to enforce a given boundary condition at that point.
**IMPORTANT** to allow the flow to:

- leave the computational box without non-physical artifacts (numerical issue)
- inject material in computational box

Boundary conditions and particle methods: a non-standard issue, because particles make sense only as a collection of overlapping points.

A single particle on a boundary not enough to enforce a given boundary condition at that point.
IMPORTANT to allow the flow to:

• leave the computational box without non-physical artifacts (numerical issue)
• inject material in computational box
• generate vorticity and to impart forces at interfaces/solid boundaries (fluid-structure interaction, physical and numerical issue)

Boundary conditions and particle methods: a non-standard issue, because particles make sense only as a collection of overlapping points.

A single particle on a boundary not enough to enforce a given boundary condition at that point.
SPH methods often use the concept of ghost particles to overcome this difficulty.

Example where one wants to enforce zero flow at the boundary by using ghost particles:

\[ h \]

\[ h \] is the radius of the regularization kernel which allows to recover local values from particle strengths.
BCs using **Ghost Particles**

SPH methods often use the concept of ghost particles to overcome this difficulty.

Example where one wants to enforce zero flow at the boundary by using ghost particles:

- **DIFFICULTY:** accurate definitions of ghost particles need local mappings around interface onto half-space geometries

$h$ is the radius of the regularization kernel which allows to recover local values from particle strengths.
The case of vortex methods for incompressible flows even more delicate, because vorticity boundary values in general not known.
The case of vortex methods for incompressible flows even more delicate, because \textit{vorticity boundary values in general not known}.

Like for all numerical methods, \textit{can deal with boundary conditions in two ways:}
The case of vortex methods for incompressible flows even more delicate, because vorticity boundary values in general not known.

Like for all numerical methods, can deal with boundary conditions in two ways:

- using body-fitted grids (the boundary is made of specific points of a given grid used to solve for the flow)
The case of vortex methods for incompressible flows even more delicate, because vorticity boundary values in general not known.

Like for all numerical methods, can deal with boundary conditions in two ways:

• using body-fitted grids (the boundary is made of specific points of a given grid used to solve for the flow)
• seeing the boundary as an immersed boundary
Vorticity BCs for No-slip Incompressible Flows

Boundary conditions appear at two levels:

- **KINEMATICS**: velocity from vorticity: \( \text{div } \mathbf{u} = 0 \), \( \text{curl } \mathbf{u} = \omega \) in \( \Omega \) and \( \mathbf{u} \cdot \mathbf{n} = 0 \) on \( \partial \Omega \) (no-through condition)

- **DYNAMICS**: advection-diffusion equation for vorticity:
  \[
  \frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla) \omega = (\omega \cdot \nabla) \mathbf{u} + \nu \Delta \omega \quad \text{in } \Omega , \quad \omega = ? \; \text{on } \partial \Omega
  \]
Kinematic Boundary Condition

Classical way to deal with the first boundary condition ($u \cdot n = 0$ on $\partial \Omega$) is to look for a decomposition of the velocity field into a rotational and a potential part:

$$u = \nabla \phi + \nabla \times \psi$$

**IN PRACTICE**: compute first $\omega$, without bothering about boundary conditions, then fix boundary conditions with $\phi$
Kinematic Boundary Condition

Classical way to deal with the first boundary condition \((\mathbf{u} \cdot \mathbf{n} = 0 \text{ on } \partial \Omega)\) is to look for a decomposition of the velocity field into a rotational and a potential part:

\[
\mathbf{u} = \nabla \varphi + \nabla \times \psi
\]

\[
\text{div } \mathbf{u} = 0 \Rightarrow \Delta \varphi = 0 \text{ in } \Omega
\]

IN PRACTICE: compute first \(\omega\), without bothering about boundary conditions, then fix boundary conditions with \(\varphi\)
Kinematic Boundary Condition

Classical way to deal with the first boundary condition \((\mathbf{u}.\mathbf{n} = 0\) on \(\partial\Omega\)) is to look for a decomposition of the velocity field into a rotational and a potential part:

\[
\mathbf{u} = \nabla \varphi + \nabla \times \psi
\]

\[
\text{div} \mathbf{u} = 0 \Rightarrow \Delta \varphi = 0 \text{ in } \Omega
\]

\[
\nabla \times \mathbf{u} = \omega \Rightarrow \Delta \psi = \omega, \text{ div } \psi = 0 \text{ in } \Omega.
\]

**IN PRACTICE:** compute first \(\omega\), without bothering about boundary conditions, then fix boundary conditions with \(\varphi\).
Kinematic Boundary Condition

Classical way to deal with the first boundary condition \((u \cdot n = 0 \text{ on } \partial \Omega)\) is to look for a decomposition of the velocity field into a rotational and a potential part:

\[
    u = \nabla \phi + \nabla \times \psi
\]

\[
    \text{div } u = 0 \Rightarrow \Delta \phi = 0 \text{ in } \Omega
\]

\[
    \nabla \times u = \omega \Rightarrow \Delta \psi = \omega , \text{div } \psi = 0 \text{ in } \Omega.
\]

**Boundary Condition** 
\[u \cdot n = 0\] gives \(\frac{\partial \phi}{\partial n} = -\frac{\partial (\nabla \times \psi)}{\partial n}\) on \(\partial \Omega\)

**IN PRACTICE**: compute first \(\omega\), without bothering about boundary conditions, then fix boundary conditions with \(\phi\)
Kinematic Boundary Condition

Classical way to deal with the first boundary condition \((\mathbf{u}.\mathbf{n} = 0\) on \(\partial \Omega\)) is to look for a decomposition of the velocity field into a rotational and a potential part:

\[
\mathbf{u} = \nabla \varphi + \nabla \times \psi
\]

\[\text{div } \mathbf{u} = 0 \Rightarrow \Delta \varphi = 0 \text{ in } \Omega\]

\[\nabla \times \mathbf{u} = \mathbf{\omega} \Rightarrow \Delta \psi = \mathbf{\omega} , \text{div } \psi = 0 \text{ in } \Omega.\]

**Boundary Condition** \(\mathbf{u}.\mathbf{n} = 0\) gives \(\partial \varphi/\partial n = -\partial (\nabla \times \psi)/\partial n\) on \(\partial \Omega\)

**IN PRACTICE**: compute first \(\mathbf{\omega}\), without bothering about boundary conditions, then fix boundary conditions with \(\varphi\)

In a grid-free vortex method, this results in:

\[
\mathbf{u}(\mathbf{x}) = \int_{\Omega_f} \mathbf{K}(\mathbf{x} - \mathbf{y})\mathbf{\omega}(\mathbf{y}) \, d\mathbf{y} + \int_{\partial \Omega} \nabla G(\mathbf{x} - \mathbf{y})q(\mathbf{y}) \, d\mathbf{y}
\]

where \(q\) is a potential to be determined from an integral equation on \(\partial \Omega\)
DYNAMICS - No-slip Condition

Next, enforce that *tangential velocities are also zero* at the boundary.

*Traditional* numerical recipe for vortex methods *mimics the physical mechanism*: vorticity produced at the boundary to prevent any slip velocity at the boundary.
DYNAMICS - No-slip Condition

Next, enforce that *tangential velocities are also zero* at the boundary

**traditional** numerical recipe for vortex methods *mimics the physical mechanism*: vorticity produced at the boundary to prevent any slip velocity at the boundary

**FRACTIONAL STEP ALGORITHM** [Chorin 1978]:
Next, enforce that \textit{tangential velocities are also zero} at the boundary.

\textbf{traditional} numerical recipe for vortex methods \textit{mimics the physical mechanism}: vorticity produced at the boundary to prevent any slip velocity at the boundary.

\textbf{FRACTIONAL STEP ALGORITHM} \cite{Chorin 1978}:

1) first substep without vorticity creation.
Next, enforce that *tangential velocities are also zero* at the boundary.

**Traditional** numerical recipe for vortex methods *mimics the physical mechanism*: vorticity produced at the boundary to prevent any slip velocity at the boundary.

**FRACTIONAL STEP ALGORITHM** [Chorin 1978]:
1) first substep without vorticity creation
2) compute resulting slip
Next, enforce that *tangential velocities are also zero* at the boundary.

**Traditional** numerical recipe for vortex methods *mimics the physical mechanism*: vorticity produced at the boundary to prevent any slip velocity at the boundary.

**FRACTIONAL STEP ALGORITHM** [Chorin 1978]:
1) first substep without vorticity creation
2) compute resulting slip
3) remove this slip by injecting in the flow the appropriate sheet of vorticity
Next, enforce that *tangential* velocity are also zero at the boundary.

*Traditional* numerical recipe for vortex methods *mimics the physical mechanism*: vorticity produced at the boundary to prevent any slip velocity at the boundary.

[Koumoutsakos-Leonard 1992]
Next, enforce that \textit{tangential} velocity are also zero at the boundary. 

\textbf{traditional} numerical recipe for vortex methods \textit{mimics the physical mechanism}: vorticity produced at the boundary to prevent any slip velocity at the boundary.

\[
\begin{align*}
\frac{\partial \omega}{\partial t} - \nu \Delta \omega &= 0 \\
\nu \frac{\partial \omega}{\partial n} &= - \frac{1}{\Delta t} u \cdot \tau
\end{align*}
\]

[Koumoutsakos-Leonard 1992]
In 3D, need boundary conditions for 3 vorticity components

After advection step computation of slip

Vorticity flux onto flow particles

\[
\begin{align*}
\frac{\partial \omega}{\partial t} - \nu \Delta \omega &= 0 \\
\omega(t_0) &= 0 \\
\nu \frac{\partial \omega_z}{\partial n} &= - \frac{\partial u_\theta}{\partial t} \\
\nu \left( \frac{\partial \omega_\theta}{\partial n} + \kappa \omega_0 \right) &= - \frac{\partial u_z}{\partial t} \\
\omega_r &= 0
\end{align*}
\]

sur $\Omega \times ]t_0, t_0 + \delta t[$

sur $\Omega$

sur $\partial \Omega \times ]t_0, t_0 + \delta t[$

sur $\partial \Omega \times ]t_0, t_0 + \delta t[$

case of flow past a cylinder

[Cottet-Poncet 2003]
3D Vorticity Flux BCs

In 3D, need boundary conditions for 3 vorticity components

\[ \frac{\partial \omega}{\partial t} - \nu \Delta \omega = 0 \quad \text{sur } \Omega \times [t_0, t_0 + \delta t] \]

\[ \omega(t_0) = 0 \quad \text{sur } \Omega \]

\[ \nu \frac{\partial \omega_z}{\partial n} = - \frac{\partial u_\theta}{\partial t} \quad \text{sur } \partial \Omega \times [t_0, t_0 + \delta t] \]

\[ \nu \left( \frac{\partial \omega_0}{\partial n} + \kappa \omega_0 \right) = - \frac{\partial u_z}{\partial t} \quad \text{sur } \partial \Omega \times [t_0, t_0 + \delta t] \]

\[ \omega_r = 0 \quad \text{sur } \partial \Omega \times [t_0, t_0 + \delta t] \]

difficulty: requires local coordinate system on the surface

*case of flow past a cylinder [Cottet-Poncet 2003]*
Obstacles, walls, objects .. are part of the flow, with specific rheology

The case of a rigid body with prescribed velocity
IMMERSED BOUNDARIES

Obstacles, walls, objects .. are part of the flow, with specific rheology

- rigid body with prescribed velocity or interacting with flow

The case of a rigid body with prescribed velocity
Obstacles, walls, objects .. are part of the flow, with **specific rheology**

- rigid body with prescribed velocity or interacting with flow
- elastic membrane

The case of a rigid body with prescribed velocity
Obstacles, walls, objects .. are part of the flow, with specific rheology

- rigid body with prescribed velocity or interacting with flow
- elastic membrane
- visco-elastic body ..

The case of a rigid body with prescribed velocity
Obstacles, walls, objects .. are part of the flow, with specific rheology

- rigid body with prescribed velocity or interacting with flow
- elastic membrane
- visco-elastic body ..

The case of a rigid body with prescribed velocity

grid where particles are initialized/created/remeshed
Boundary Conditions = Coupling Dynamics

Boundary Conditions
Boundary Conditions = Coupling Dynamics
Boundary Conditions = Coupling

- Coupling via a Boundary Force
Enforce boundary velocity by a bodyforce $f$ in Momentum Equation

$$
\rho \frac{Du}{Dt} = -\nabla p + \nabla \cdot \tau + f
$$

$$
= f_{i,\text{part}} + f_{i,\text{boundary}}
$$
I. IMMERSED BOUNDARY METHOD for SPH

• Enforce boundary velocity by a bodyforce \( f \) in Momentum Equation
  \[
  \rho \frac{Du}{Dt} = -\nabla p + \nabla \cdot \tau + f
  \]

• Approximate Material Derivative at time step \( i \) and solve for \( f \)
  \[
  \rho_i \left( \frac{u_{i+1} - u_i}{\Delta t} \right) = -\nabla p_i + \nabla \cdot \tau_i + f_i \Rightarrow f_i = \rho_i \frac{u_{i+1} - u_i}{\Delta t} - (-\nabla p_i + \nabla \cdot \tau_i)
  \]

• Desired Velocity field on the boundary
  \[
  u_{i+1} = u_{desired}
  \]
  \[
  u_{i+1} = u_{desired} \Rightarrow f_i = \rho_i \frac{u_{desired} - u_i}{\Delta t} - (-\nabla p_i + \nabla \cdot \tau_i)
  \]
  \[
  = f_{i,part} + f_{i,boundary}
  \]
Boundary Conditions: A Particle-Mesh Operation

- Compute part of forcing term on the particles

\[ f_{i,\text{part}} = \rho_i \frac{-u_i}{\Delta t} - (-\nabla p_i + \nabla \tau_i) \]
Boundary Conditions: A Particle-Mesh Operation

- Compute part of forcing term on the particles
  \[
  f_{i,\text{part}} = \rho_i \frac{-u_i}{\Delta t} - (-\nabla p_i + \nabla \tau_i)
  \]

- Particles to Boundary (Particle to Mesh Interpolation)
  \[
  f_{i,\text{boundary}} = \rho_i \frac{u_{\text{desired}}}{\Delta t} + f_{i,\text{part},\text{interpolated}}
  \]
Boundary Conditions: A Particle-Mesh Operation

- Compute part of forcing term on the particles

\[ f_{i,\text{part}} = \rho_i \frac{-u_i}{\Delta t} - (\nabla p_i + \nabla \tau_i) \]

- Particles to Boundary (Particle to Mesh Interpolation)

\[ f_{i,\text{boundary}} = \rho_i \frac{u_{\text{desired}}}{\Delta t} + f_{i,\text{part,interpolated}} \]

- Force -> Boundary to Particles (Mesh - Particle Interpolation)
Boundary Conditions: A Particle-Mesh Operation

- **Compute part of forcing term on the particles**

\[
 f_{i, \text{part}} = \rho_i \frac{-u_i}{\Delta t} - (-\nabla p_i + \nabla \tau_i )
\]

- **Particles to Boundary** *(Particle to Mesh Interpolation)*

\[
 f_{i, \text{boundary}} = \rho_i \frac{u_{\text{desired}}}{\Delta t} + f_{i, \text{part, interpolated}}
\]

- **Force -> Boundary to Particles** *(Mesh - Particle Interpolation)*

Lattice Boltzmann and Impulsively Started Cylinders

Results on Swimming

Finite Volume  (Kern & Koumoutsakos, J. Exp. Biology, 2007)

Particle + IBM  (Hieber & Koumoutsakos, JCP, 2008)

Longitudinal and lateral velocity
Results on Swimming

Finite Volume (Kern & Koumoutsakos, J. Exp. Biology, 2007)

Particle + IBM (Hieber & Koumoutsakos, JCP, 2008)

Longitudinal and lateral velocity
One way to view/derive immersed boundary techniques based on a penalized flow equation [Angot-Bruneau-Fabrie 1999]

\[
\left( \frac{\partial u}{\partial t} + (u \cdot \nabla) u \right) - \nu \Delta u + \nabla p = \lambda \rho \chi_S (\bar{u} - u),
\]

In a vorticity formulation, leads to

\[
\frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega - (\omega \cdot \nabla) u - \nu \Delta \omega = + \lambda \nabla \times \chi_S (\bar{u} - u).
\]

complemented by the usual system giving \( u \) from \( \omega \)
II. IMMERSED BOUNDARY via PENALISATION

One way to view/derive immersed boundary techniques based on a \textit{penalized} flow equation [Angot-Bruneau-Fabrie 1999]

\[
\left( \frac{\partial u}{\partial t} + (u \cdot \nabla) u \right) - \nu \Delta u + \nabla p = \lambda \rho \chi_s (\bar{u} - u),
\]

- $\bar{u}$ = prescribed body velocity,
- $\chi_s = 1$ in the body, 0 outside and
- $\lambda$ a (large) penalization parameter.

In a vorticity formulation, leads to

\[
\frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega - (\omega \cdot \nabla) u - \nu \Delta \omega = + \lambda \nabla \times \chi_s (\bar{u} - u).
\]

complemented by the usual system giving $u$ from $\omega$
Vorticity form of PENALISATION

to understand the role of the additional term in the flow equation, useful to extend it

\[
\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega - (\omega \cdot \nabla)u - \nu \Delta \omega = \lambda \chi_S (\bar{\omega} - \omega) + \lambda \delta_\Sigma n \times (\bar{u} - u).
\]

\begin{align*}
\uparrow & \quad \text{drives vorticity back to the correct body vorticity} \\
\uparrow & \quad \text{creates a vortex sheet on the body surface}
\end{align*}
Vorticity form of PENALISATION

to understand the role of the additional term in the flow equation, useful to extend it

$$\frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega - (\omega \cdot \nabla) u - \nu \Delta \omega = \lambda \chi_S (\bar{\omega} - \omega) + \lambda \delta_{\Sigma} n \times (\bar{u} - u).$$

↑

drives vorticity back to the correct body vorticity

↑

creates a vortex sheet on the body surface

Main differences with previous approach:
Vorticity form of PENALISATION

to understand the role of the additional term in the flow equation, useful to extend it

\[ \frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega - (\omega \cdot \nabla) u - \nu \Delta \omega = \lambda \chi_S (\bar{\omega} - \omega) + \lambda \delta \Sigma \ n \times (\bar{u} - u). \]

- drives vorticity back to the correct body vorticity
- creates a vortex sheet on the body surface

Main differences with previous approach:
• more clear-cut
Vorticity form of PENALISATION

to understand the role of the additional term in the flow equation, useful to extend it

\[ \frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega - (\omega \cdot \nabla) u - \nu \Delta \omega = \lambda \chi_S (\bar{\omega} - \omega) + \lambda \delta_\Sigma n \times (\bar{u} - u). \]

- drives vorticity back to the correct body vorticity
- creates a vortex sheet on the body surface

Main differences with previous approach:
• more clear-cut
• both normal and tangential components taken care of in the same equation
Vorticity form of PENALISATION

to understand the role of the additional term in the flow equation, useful to extend it

$$\frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega - (\omega \cdot \nabla) u - \nu \Delta \omega = \lambda \chi_S (\bar{\omega} - \omega) + \lambda \delta_{\Sigma} n \times (\bar{u} - u).$$

Main differences with previous approach:
• more clear-cut
• both normal and tangential components taken care of in the same equation
• simpler and cheaper to implement (both in grid-free and particle-grid codes)
\[
\frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega = (\omega \cdot \nabla) u + \nu \Delta \omega - \nabla p \times \nabla \left( \frac{1}{\rho} \right) + \lambda \nabla \times \chi_S (\bar{u} - u).
\]

+ elastic stresses

additional vorticity generator at the fluid/solid interface
• fluid-solid system considered as a variable-density flow

\[
\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega = (\omega \cdot \nabla)u + \nu \Delta \omega - \nabla p \times \nabla \left( \frac{1}{\rho} \right) + \lambda \nabla \times \chi_S (\overline{u} - u). 
\]

+ elastic stresses
Extension to Flow-Structure Interaction

- fluid-solid system considered as a variable-density flow
- body velocity results from flow forces

\[
\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega = (\omega \cdot \nabla)u + \nu \Delta \omega - \nabla p \times \nabla \left(\frac{1}{\rho}\right) + \lambda \nabla \times \chi_S (\bar{u} - u).
\]

+ elastic stresses
Extension to Flow-Structure Interaction

- fluid-solid system considered as a variable-density flow
- body velocity results from flow forces
- velocity continuity obtained by penalization

\[
\frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega = (\omega \cdot \nabla) u + \nu \Delta \omega - \nabla p \times \nabla \left( \frac{1}{\rho} \right) + \lambda \nabla \times \chi_S (\bar{u} - u). \]

+ elastic stresses
Extension to Flow-Structure Interaction

- fluid-solid system considered as a variable-density flow
- body velocity results from flow forces
- velocity continuity obtained by penalization
- contact between solid and/or elastic behavior through a level set model

\[
\frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega = (\omega \cdot \nabla) u + \nu \Delta \omega - \nabla p \times \nabla \left( \frac{1}{\rho} \right) + \lambda \nabla \times \chi_s (\bar{u} - u) + \text{elastic stresses}
\]

additional vorticity generator at the fluid/solid interface
Extension to Flow-Structure Interaction

- fluid-solid system considered as a variable-density flow
- body velocity results from flow forces
- velocity continuity obtained by penalization
- contact between solid and/or elastic behavior through a level set model

\[
\frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega = (\omega \cdot \nabla) u + \nu \Delta \omega - \nabla p \times \nabla \left( \frac{1}{\rho} \right) + \lambda \nabla \times \chi_S (\bar{u} - u). + \text{elastic stresses}
\]

**NEED TO:**
Extension to Flow-Structure Interaction

• fluid-solid system considered as a variable-density flow
• body velocity results from flow forces
• velocity continuity obtained by penalization
• contact between solid and/or elastic behavior through a level set model

\[ \frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega = (\omega \cdot \nabla) u + \nu \Delta \omega - \nabla p \times \nabla \left( \frac{1}{\rho} \right) + \lambda \nabla \times \chi_S (\bar{u} - u). \]

NEED TO:
1. compute pressure
Extension to Flow-Structure Interaction

- fluid-solid system considered as a variable-density flow
- body velocity results from flow forces
- velocity continuity obtained by penalization
- contact between solid and/or elastic behavior through a level set model

\[
\frac{\partial \omega}{\partial t} + (u \cdot \nabla) \omega = (\omega \cdot \nabla) u + \nu \Delta \omega - \nabla p \times \nabla \left( \frac{1}{\rho} \right) + \lambda \nabla \times \chi_S (\bar{u} - u).
\]

+ elastic stresses

NEED TO:
1. compute pressure
2. compute solid velocity(ies)
Extension to Flow-Structure Interaction

- fluid-solid system considered as a variable-density flow
- body velocity results from flow forces
- velocity continuity obtained by penalization
- contact between solid and/or elastic behavior through a level set model

\[ \frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega = (\omega \cdot \nabla)u + \nu \Delta \omega - \nabla p \times \nabla \left( \frac{1}{\rho} \right) + \lambda \nabla \times \chi_S(\bar{u} - u). \]

additional vorticity generator at the fluid/solid interface

NEED TO:
1. compute pressure
2. compute solid velocity(ies)
3. model/compute elastic stresses
Pressure in a Vorticity Formulation?

Several possible ways in a vorticity formulation

In an hybrid particle-grid method, the simplest is to go back to the velocity-pressure formulation:

\[
\rho \left( \frac{\partial u}{\partial t} + (u \cdot \nabla) u \right) - \nu \Delta u + \nabla p = \rho g + \lambda \rho \chi_S (\bar{u} - u).
\]

\[\nabla p\text{ in terms of } u \text{ and its derivatives.}\]

Evaluated by finite differences on the underlying grid form velocity values at the beginning of the step and at previous time-step.
Projection of flow velocity onto rigid body velocity:

\[
\bar{u} = \frac{1}{|S|} \int_K \chi_S u \, dx + \left( J^{-1} \int_K \chi_S u \times (x - x_G) \, dx \right) \times (x - x_G)
\]

where \( J \) is the inertia matrix of the body \( K \), and \( x_G \) is its mass center.
Projection of flow velocity onto rigid body velocity:

\[
\overline{u} = \frac{1}{|S|} \int_K \chi_S u \, dx + \left( J^{-1} \int_K \chi_S u \times (x - x_G) \, dx \right) \times (x - x_G)
\]

where \( J \) is the inertia matrix of the body \( K \), and \( x_G \) is its mass center.
Solid Velocities

Projection of flow velocity onto rigid body velocity:

\[
\bar{u} = \frac{1}{|S|} \int_K \chi_S u \, dx + \left( J^{-1} \int_K \chi_S u \times (x - x_G) \, dx \right) \times (x - x_G)
\]

where \( J \) is the inertia matrix of the body \( K \), and \( x_G \) is its mass center.

• \( K \) is captured by a level set function, advected with the rigid motion \( \bar{u} \):

\[
\frac{\partial \phi}{\partial t} + (\bar{u} \cdot \nabla) \phi = 0
\]
Solid Velocities

Projection of flow velocity onto rigid body velocity:

\[
\bar{u} = \frac{1}{|S|} \int_K \chi_S u \, dx + \left( J^{-1} \int_K \chi_S u \times (x - x_G) \, dx \right) \times (x - x_G)
\]

where \( J \) is the inertia matrix of the body \( K \), and \( x_G \) is its mass center.

\( \cdot \)

\( K \) is captured by a level set function, advected with the rigid motion \( \bar{u} \):

\[
\frac{\partial \phi}{\partial t} + (\bar{u} \cdot \nabla)\phi = 0
\]

\( \cdot \)

*Accumulating* rigid motions step by step allows to advect exactly \( K \) from its initial stage (the advection equation is not solved - Prevents numerical errors to slightly deform \( K \))
Solid body Collisions

contact/collision model based on dynamical repulsive model, distributed on the skin of the bodies through a level set technique

1D dynamical system for a collision on a range $O(v^2/\kappa)$

$$\ddot{x} = \frac{\kappa}{\epsilon} \exp(-x/\epsilon),$$

translation into an interface-distributed contact force:

$$f_{\text{col}}(x) = -\rho \sum_{ij} \frac{K_{ij}}{\epsilon} \zeta \left( \frac{\phi_i(x)}{\epsilon} \right) \frac{\nabla \phi_j(x)}{\phi_j(x)} \exp(-\phi_j(x)/\epsilon).$$
A hybrid particle-grid method for flows interacting with rigid bodies - [Coquerelle-Cottet 2008]
**time 0**: initialize flow and assign level set signed-distance $\bar{u}$ functions to each body
**time 0:** initialize flow and assign level set signed-distance $\bar{u}$ functions to each body

**step iterations** (black on grid, red on particles):
A hybrid particle-grid method for flows interacting with rigid bodies - [Coquerelle-Cottet 2008]

**time 0**: initialize flow and assign level set signed-distance $\tilde{u}$ functions to each body

**step iterations** (black on grid, red on particles):

$$(u^n, \tilde{u}^n_i, \phi^n_i, \omega^n) \rightarrow (u^{n+1}, \tilde{u}^{n+1}_i, \phi^{n+1}_i, \omega^{n+1})$$
time 0: initialize flow and assign level set signed-distance $\bar{u}$ functions to each body

step iterations (black on grid, red on particles):

$(u^n, \bar{u}_i^n, \phi_i^n, \omega^n) \rightarrow (u^{n+1}, \bar{u}_i^{n+1}, \phi_i^{n+1}, \omega^{n+1})$

• advance exactly each level set function $\phi_i^n$ with the corresponding rigid motion $\bar{u}_i^n \rightarrow \phi_i^{n+1}$
time 0: initialize flow and assign level set signed-distance \( \bar{u} \) functions to each body

step iterations (black on grid, red on particles):

\[(u^n, \bar{u}_i^n, \phi_i^n, \omega^n) \rightarrow (u^{n+1}, \bar{u}_i^{n+1}, \phi_i^{n+1}, \omega^{n+1})\]

- advance exactly each level set function \( \phi_i^n \) with the corresponding rigid motion \( \bar{u}_i^n \rightarrow \phi_i^{n+1} \)

- compute collision forces
A hybrid particle-grid method for flows interacting with rigid bodies - [Coquerelle-Cottet 2008]

**time 0**: initialize flow and assign level set signed-distance $\tilde{u}$ functions to each body

**step iterations** (black on grid, red on particles):

$(u^n, \tilde{u}_i^n, \phi_i^n, \omega^n) \rightarrow (u^{n+1}, \tilde{u}_i^{n+1}, \phi_i^{n+1}, \omega^{n+1})$

- advance *exactly* each level set function $\phi_i^n$ with the corresponding rigid motion $\tilde{u}_i^n \rightarrow \phi_i^{n+1}$

- compute collision forces

\[ f_{\text{col}}(x) = -\rho \sum_u \frac{k_u}{\epsilon} \zeta \left( \frac{\phi_i(x)}{\epsilon} \right) \frac{V\phi_j(x)}{\phi_j(x)} \exp(-\phi_j(x)/\epsilon). \]
A hybrid particle-grid method for flows interacting with rigid bodies - [Coquerelle-Cottet 2008]

**time 0**: initialize flow and assign level set signed-distance $\bar{\mathbf{u}}$ functions to each body

**step iterations** (black on grid, red on particles):

$$(u^n, \bar{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \bar{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})$$

- advance *exactly* each level set function $\varphi_i^n$ with the corresponding rigid motion $\bar{u}_i^n: \rightarrow \varphi_i^{n+1}$

- compute collision forces
A hybrid particle-grid method for flows interacting with rigid bodies - [Coquerelle-Cottet 2008]

**time 0**: initialize flow and assign level set signed-distance $\tilde{u}$ functions to each body

**step iterations** (black on grid, red on particles):

$$(u^n, \tilde{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \tilde{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})$$

- advance *exactly* each level set function $\varphi_i^n$ with the corresponding rigid motion $\tilde{u}_i^n$:

- compute collision forces

- update rigid body motions (no vorticity of collision force needed)
A hybrid particle-grid method for flows interacting with rigid bodies - [Coquerelle-Cottet 2008]

**time 0**: initialize flow and assign level set signed-distance $\bar{u}$ functions to each body

**step iterations** (black on grid, red on particles):

$(u^n, \bar{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \bar{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})$

- advance *exactly* each level set function $\varphi_i^n$ with the corresponding rigid motion $\bar{u}_i^n \rightarrow \varphi_i^{n+1}$

- compute collision forces

- update rigid body motions (no vorticity of collision force needed)

- update vorticity on the grid with
A hybrid particle-grid method for flows interacting with rigid bodies - [Coquerelle-Cottet 2008]

**time 0**: initialize flow and assign level set signed-distance $\tilde{u}$ functions to each body

**step iterations** (black on grid, red on particles):

$$(u^n, \tilde{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \tilde{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})$$

- advance *exactly* each level set function $\varphi_i^n$ with the corresponding rigid motion $\tilde{u}_i^n \rightarrow \varphi_i^{n+1}$

- compute collision forces

- update rigid body motions (no vorticity of collision force needed)

- update vorticity on the grid with
  
  - penalization force $\nabla \times (\sum_i \chi_i (u-\tilde{u}_i^n))$
**time 0**: initialize flow and assign level set signed-distance $\tilde{u}$ functions to each body

**step iterations** (black on grid, red on particles):

$$(u^n, \tilde{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \tilde{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})$$

- advance *exactly* each level set function $\varphi_i^n$ with the corresponding rigid motion $\tilde{u}_i^n \rightarrow \varphi_i^{n+1}$
- compute collision forces
- update rigid body motions (no vorticity of collision force needed)
- update vorticity on the grid with
  - penalization force $\nabla \times (\sum_i \chi_i (u - \tilde{u}_i^n))$
  - pressure gradients
time 0: initialize flow and assign level set signed-distance $\bar{u}$ functions to each body

step iterations (black on grid, red on particles):

$(u^n, \bar{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \bar{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})$

- advance exactly each level set function $\varphi_i^n$ with the corresponding rigid motion $\bar{u}_i^n : \rightarrow \varphi_i^{n+1}$

- compute collision forces

- update rigid body motions (no vorticity of collision force needed)

- update vorticity on the grid with
  - penalization force $\nabla \times (\sum_i \chi_i (u-\bar{u}_i^n))$
  - pressure gradients

- push particles (transport-stretching-diffusion of vorticity)
time 0: initialize flow and assign level set signed-distance $\bar{u}$ functions to each body

step iterations (black on grid, red on particles):

$(u^n, \bar{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \bar{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})$

• advance exactly each level set function $\varphi_i^n$ with the corresponding rigid motion $\bar{u}_i^n \rightarrow \varphi_i^{n+1}$

• compute collision forces

• update rigid body motions (no vorticity of collision force needed)

• update vorticity on the grid with
  \[ \frac{d\varphi_i^n}{dt} = \nabla \times (\sum \chi_i (u-\bar{u}_i^n)) \]
  \[ \frac{d\omega_p}{dt} = [\nabla u(x_p, t)] \omega_p + \nu \Delta \omega(x_p). \]

• push particles (transport-stretching-diffusion of vorticity)
**Time 0**: initialize flow and assign level set signed-distance $\bar{u}$ functions to each body

**Step Iterations** (black on grid, red on particles):

\[(u^n, \bar{u}i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \bar{u}i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})\]

- advance *exactly* each level set function $\varphi_i^n$ with the corresponding rigid motion $\bar{u}_i^n: \rightarrow \varphi_i^{n+1}$

- compute collision forces

- update rigid body motions (no vorticity of collision force needed)

- update vorticity on the grid with
  - penalization force $\nabla \times (\sum_i \chi_i (u-\bar{u}_i^n))$
  - pressure gradients

- push particles (transport-stretching-diffusion of vorticity)
**time 0:** initialize flow and assign level set signed-distance $\tilde{u}$ functions to each body

**step iterations** (black on grid, red on particles):

$$(u^n, \tilde{u}_i^n, \phi_i^n, \omega^n) \rightarrow (u^{n+1}, \tilde{u}_i^{n+1}, \phi_i^{n+1}, \omega^{n+1})$$

- advance *exactly* each level set function $\phi_i^n$ with the corresponding rigid motion $\tilde{u}_i^n: \rightarrow \phi_i^{n+1}$

- compute collision forces

- update rigid body motions (no vorticity of collision force needed)

- update vorticity on the grid with
  - penalization force $\nabla \times (\sum_i \chi_i (u-\tilde{u}_i^n))$
  - pressure gradients

- push particles (transport-stretching-diffusion of vorticity)

- remesh particles $\rightarrow \omega^{n+1}$
**A hybrid particle-grid method for flows interacting with rigid bodies - [Coquerelle-Cottet 2008]**

**time 0**: initialize flow and assign level set signed-distance $\tilde{u}$ functions to each body

**step iterations** (black on grid, red on particles):

$(u^n, \tilde{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \tilde{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})$

- advance *exactly* each level set function $\varphi_i^n$ with the corresponding rigid motion $\tilde{u}_i^n \rightarrow \varphi_i^{n+1}$

- compute collision forces

- update rigid body motions (no vorticity of collision force needed)

- update vorticity on the grid with
  - $\nabla \times (\sum_i \chi_i (u-\tilde{u}_i^n))$
  - pressure gradients

- push particles (transport-stretching-diffusion of vorticity)

- remesh particles $\rightarrow \omega^{n+1}$

- compute flow velocity with a fast FFT-based Poisson solver $\rightarrow u^{n+1}$
time 0: initialize flow and assign level set signed-distance \( \tilde{u} \) functions to each body

step iterations (black on grid, red on particles):

\[
(u^n, \tilde{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \tilde{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})
\]

• advance exactly each level set function \( \varphi_i^n \) with the corresponding rigid motion \( \tilde{u}_i^n \rightarrow \varphi_i^{n+1} \)

• compute collision forces

• update rigid body motions (no vorticity of collision force needed)

• update vorticity on the grid with
  • penalization force \( \nabla \times (\sum_i \chi_i (u-\tilde{u}_i^n)) \)
  • pressure gradients

• push particles (transport-stretching-diffusion of vorticity)

• remesh particles \( \rightarrow \omega^{n+1} \)

• compute flow velocity with a fast FFT-based Poisson solver \( \rightarrow u^{n+1} \)

• compute rigid body motions
A hybrid particle-grid method for flows interacting with rigid bodies - [Coquerelle-Cottet 2008]

**time 0**: initialize flow and assign level set signed-distance $\bar{u}$ functions to each body

**step iterations** (black on grid, red on particles): $(u^n, \bar{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \bar{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})$

- advance *exactly* each level set function $\varphi_i^n$ with the corresponding rigid motion $\bar{u}_i^n \rightarrow \varphi_i^{n+1}$
- compute collision forces
- update rigid body motions (no vorticity of collision force needed)
- update vorticity on the grid with
  - penalization force $\nabla \times (\sum_i \chi_i (u-\bar{u}_i^n))$
  - pressure gradients
- push particles (transport-stretching-diffusion of vorticity)
- remesh particles $\rightarrow \omega^{n+1}$
- compute flow velocity with a fast FFT-based Poisson solver $\rightarrow u^{n+1}$
- compute rigid body motions

\[
\bar{u} = \frac{1}{|S|} \int_K \chi_S u \, dx + \left( J^{-1} \int_K \chi_S \nabla \times (x - x_G) \, dx \right) \times (x - x_G)
\]
A hybrid particle-grid method for flows interacting with rigid bodies - [Coquerelle-Cottet 2008]

**Time 0:** initialize flow and assign level set signed-distance $\tilde{u}$ functions to each body

**Step Iterations** (black on grid, red on particles):

$(u^n, \tilde{u}_i^n, \varphi_i^n, \omega^n) \rightarrow (u^{n+1}, \tilde{u}_i^{n+1}, \varphi_i^{n+1}, \omega^{n+1})$

- advance *exactly* each level set function $\varphi_i^n$ with the corresponding rigid motion $\tilde{u}_i^n: \rightarrow \varphi_i^{n+1}$

- compute collision forces

- update rigid body motions (no vorticity of collision force needed)

- update vorticity on the grid with
  - penalization force $\nabla \times (\sum_i \chi_i (u - \tilde{u}_i^n))$
  - pressure gradients

- push particles (transport-stretching-diffusion of vorticity)

- remesh particles $\rightarrow \omega^{n+1}$

- compute flow velocity with a fast FFT-based Poisson solver $\rightarrow u^{n+1}$

- compute rigid body motions
Final remarks:
✓ close to the projection method in [Sharma-Patankar 2005], but allows better fit of rigid body by implicit treatment on penalization
✓ can also add to collision force
  • free surfaces, with or without surface tension, captured by additional level set functions
  • elastic fluid-structure interaction of the form $F(\varphi, \nabla \varphi)$.

In these cases the level set equations must be solved (preferably with particles)
SUMMARY

• on the GRID Poisson solver and finite-differences

Final remarks:
✓ close to the projection method in [Sharma-Patankar 2005], but allows better fit of rigid body by implicit treatment on penalization
✓ can also add to collision force
  • free surfaces, with or without surface tension, captured by additional level set functions
  • elastic fluid-structure interaction of the form $F(\varphi, \nabla \varphi)$.  

In these cases the level set equations must be solved (preferably with particles)
SUMMARY

• on the GRID Poisson solver and finite-differences
• on the PARTICLES: time-stepping of flow

Final remarks:
✓ close to the projection method in [Sharma-Patankar 2005], but allows better fit of rigid body by implicit treatment on penalization
✓ can also add to collision force
  • free surfaces, with or without surface tension, captured by additional level set functions
  • elastic fluid-structure interaction of the form F(\varphi, \nabla \varphi).

In these cases the level set equations must be solved (preferably with particles)
SUMMARY

• on the GRID Poisson solver and finite-differences
• on the PARTICLES: time-stepping of flow

✓ most of the CPU time on remeshing

Final remarks:
✓ close to the projection method in [Sharma-Patankar 2005], but allows better fit of rigid body by implicit treatment on penalization
✓ can also add to collision force
  • free surfaces, with or without surface tension, captured by additional level set functions
  • elastic fluid-structure interaction of the form $F(\varphi, \nabla \varphi)$.

In these cases the level set equations must be solved (preferably with particles)
SUMMARY

• on the GRID Poisson solver and finite-differences
• on the PARTICLES: time-stepping of flow

✓ most of the CPU time on remeshing
✓ gain over grid-based methods through large time-steps and localization of computational effort (vorticity support) on interfaces.

Final remarks:
✓ close to the projection method in [Sharma-Patankar 2005], but allows better fit of rigid body by implicit treatment on penalization
✓ can also add to collision force
  • free surfaces, with or without surface tension, captured by additional level set functions
  • elastic fluid-structure interaction of the form $F(\varphi, \nabla \varphi)$.

In these cases the level set equations must be solved (preferably with particles)