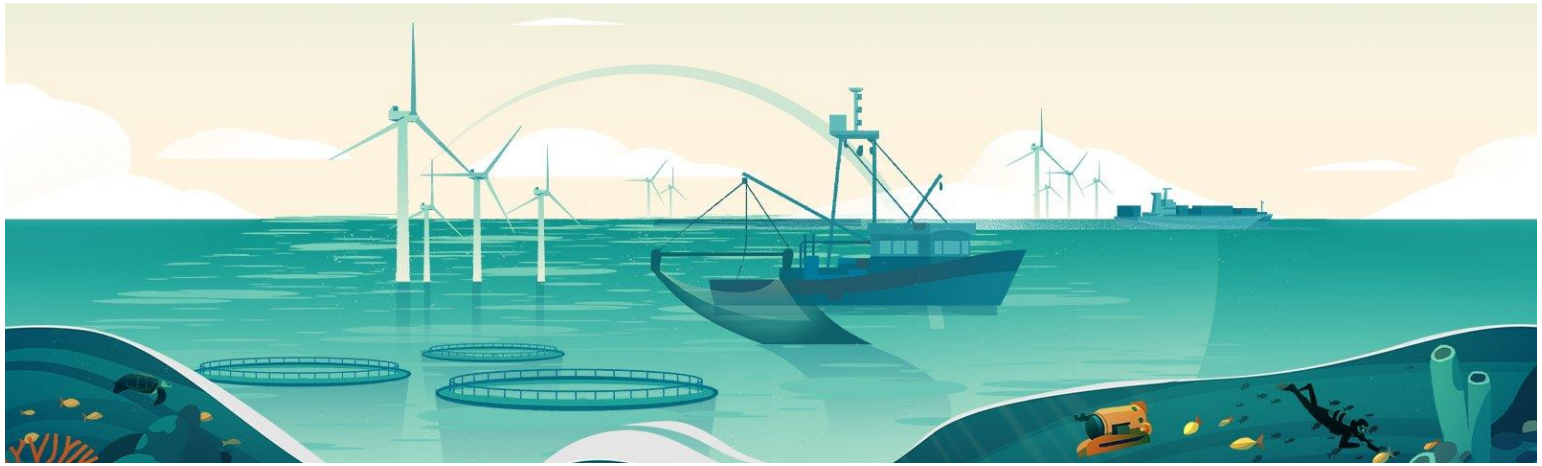
Exploiting the **Gridap.jl** ecosystem to solve complex PDEs for engineering problems



Oriol Colomés

27th March 2026

Many people involved...

CMOE

Computational Multiphysics in Offshore Engineering group

Lisa van der Linde

Shreyas Prashanth

Marta Amorós

Pau Manyer

Shagun Agarwal

Jan Modderman

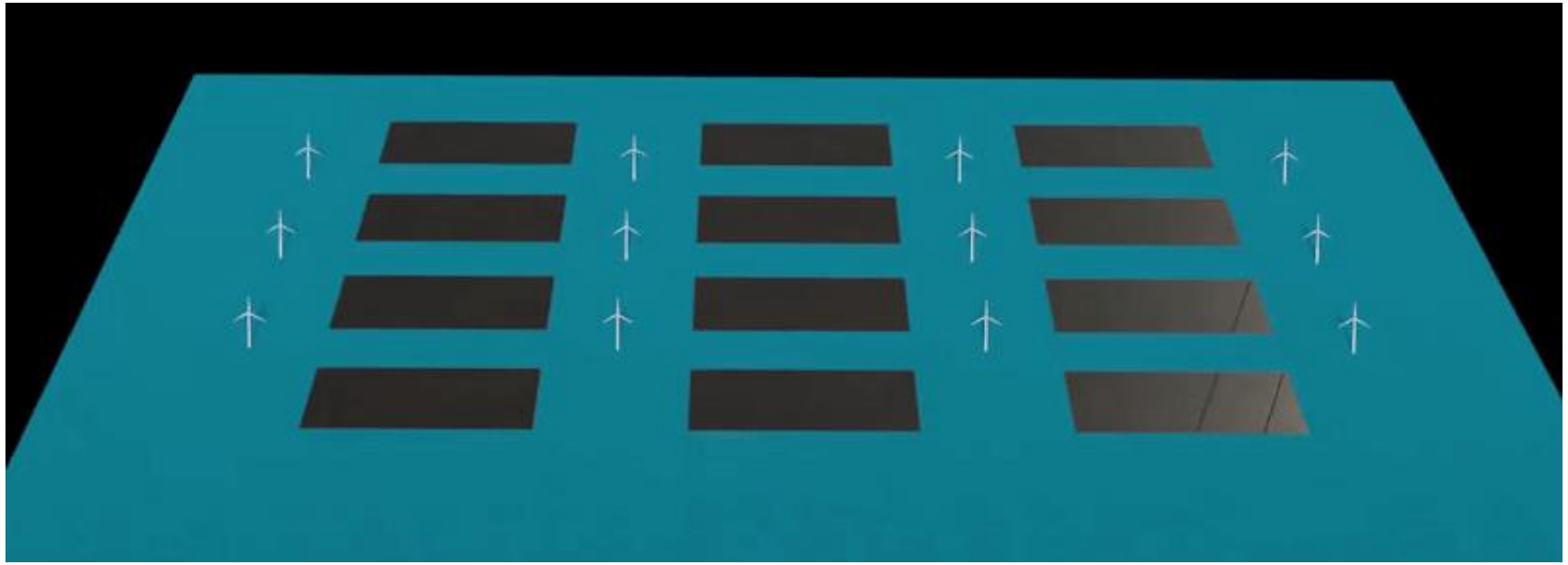
Nicholas Mueller

TU Delft

CMOE

What are we interested in?

The **CMOE group** specializes in **interdisciplinary research and education** bridging **scientific computing**, **fluid dynamics**, and **offshore engineering**.



What do we need?

- Be able to use the **same code for research and teaching**
- **Reduce learning time** for new members (engineering students background)
- Have a larger impact through **open source** and active community
- **Application-oriented** research

Why Julia and Gridap.jl?

Gridap

- Be able to use the **same code for research and teaching**

Implemented weak form as written in paper

- **Reduce learning time** for new members (engineering students background)

Single language, no compilation, python-like language

- Have a larger impact through **open source** and active community

Very active (growing?) community

- **Application-oriented** research

Easy-to-use for non-experts

How do we use Gridap.jl ecosystem?

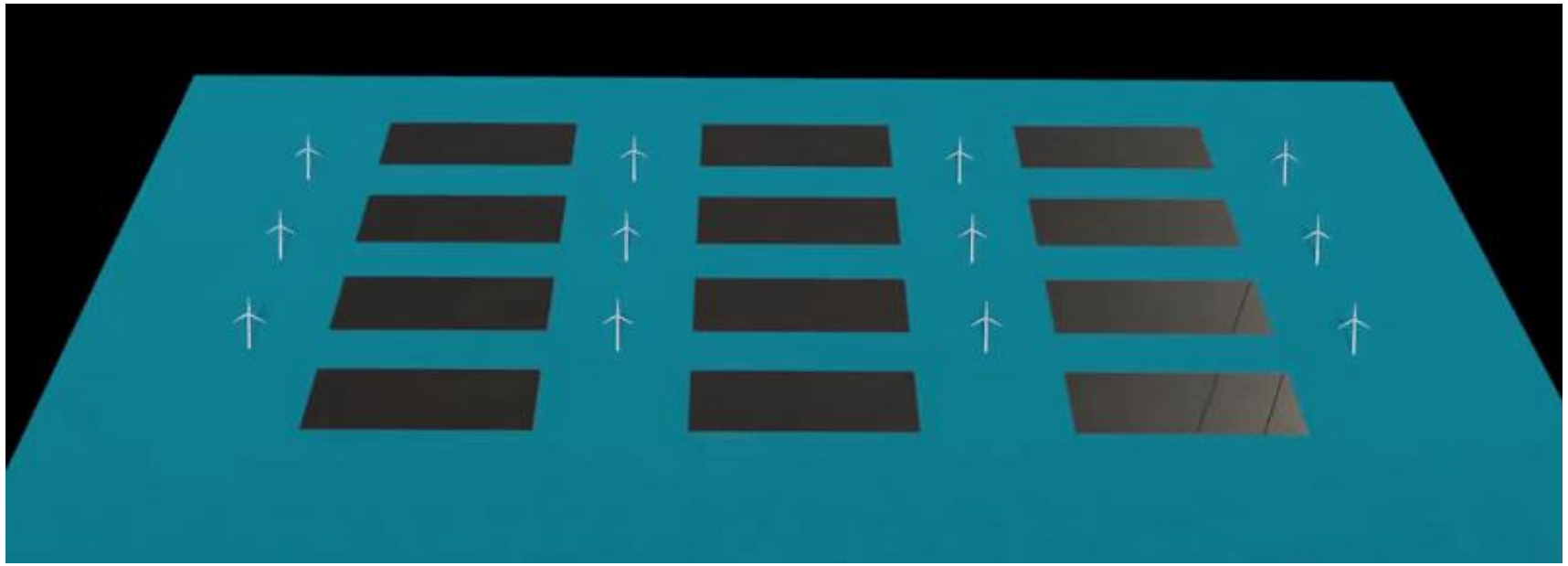
- Solution of mixed-dimensional PDEs
- Solution of PDEs in complex geometries
- Solution of highly nonlinear PDEs

How do we use Gridap.jl ecosystem?

- **Solution of mixed-dimensional PDEs**
- Solution of PDEs in complex geometries
- Solution of highly nonlinear PDEs

Solution of mixed-dimensional PDEs

Interested in the **hydroelastic** analysis of **floating flexible structures**



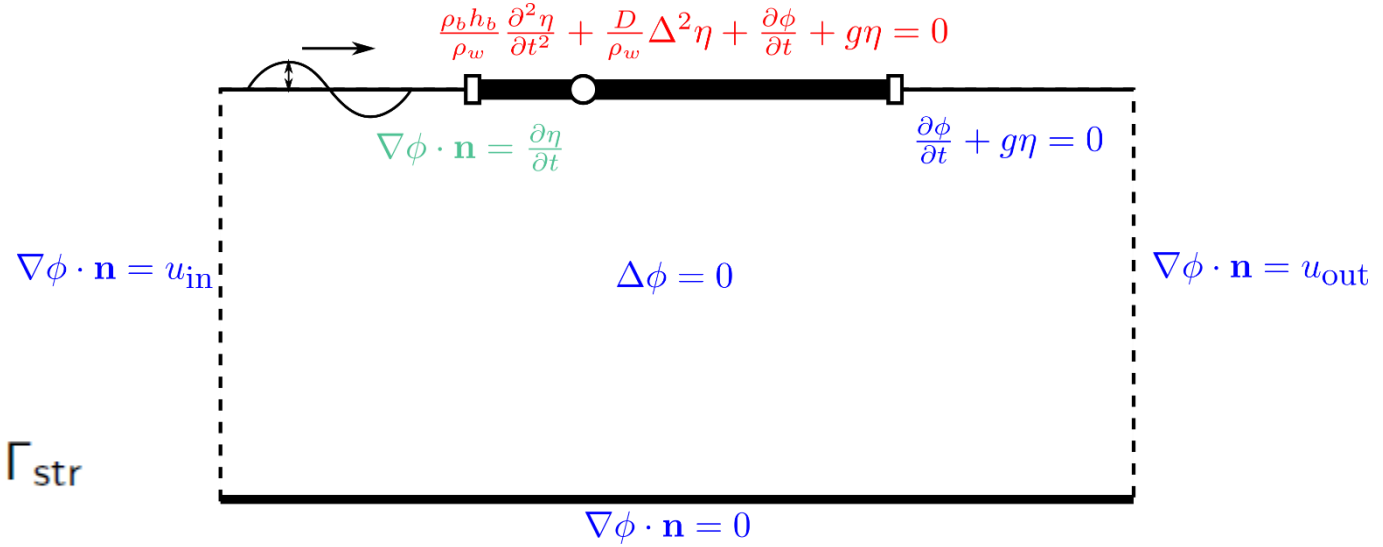
Problem setting

After many assumptions, we just need to determine 2 unknowns

1. Velocity potential (ϕ)
2. Surface elevation (η)

Final set of equations:

$$\begin{cases} -\Delta\phi = 0 \quad \text{in } \Omega \\ \dfrac{\partial\phi}{\partial t} + g\eta = 0 \quad \text{on } \Gamma_{\text{fs}} \\ \dfrac{\rho_b h_b}{\rho_w}\dfrac{\partial^2\eta}{\partial t^2} + D\Delta^2\eta + \dfrac{\partial\phi}{\partial t} + g\eta = 0 \quad \text{on } \Gamma_{\text{str}} \\ \nabla\phi\cdot\mathbf{n} = \dfrac{\partial\eta}{\partial t} \quad \text{on } \Gamma_{\text{fs}}\cup\Gamma_{\text{str}} \end{cases}$$



Solution strategy

There are a few approaches we could follow:

- **Analytical solution**. Only for regular shapes and simplifications at the boundary
- **Boundary Element method**. Limitations when extending this framework to more complex scenarios
- **Finite Element method**. General framework, can handle arbitrary shapes, nonlinear case…

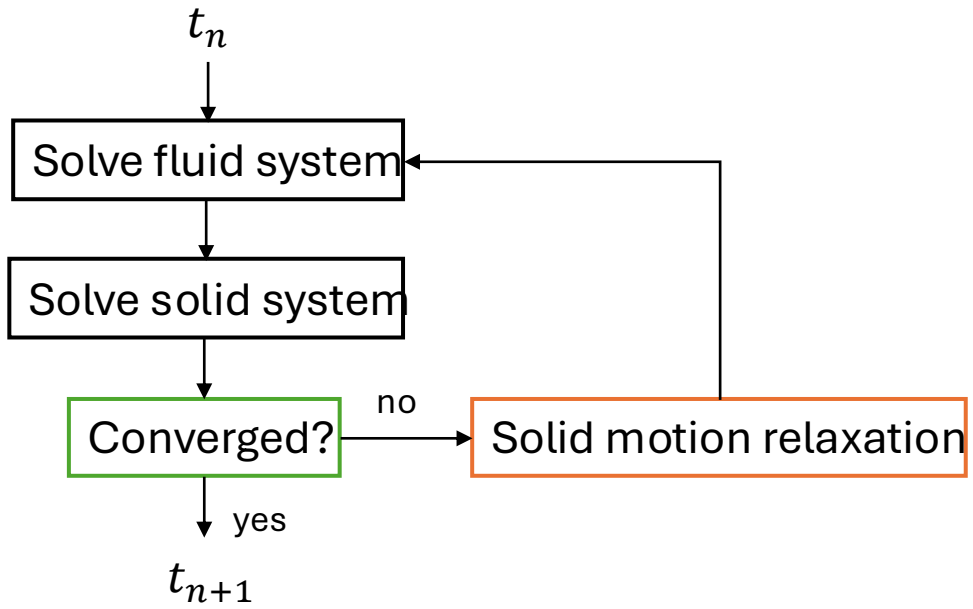
Additional challenges:

- Coupling between fluid and structure can lead to instabilities => **monolithic strategy**
- A monolithic formulation leads to a **mixed-dimensional PDE**

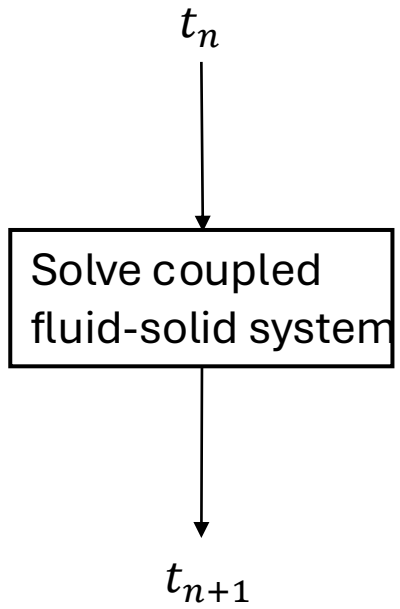
Why monolithic approach?

How can we solve coupled problems?

Partitioned approach



Monolithic approach



Why monolithic approach?

Partitioned scheme:

- Solution of two smaller problems (faster)
- Iterative process

Lack of convergence in partitioned schemes for Fluid-structure interaction problems: **Added mass effect**

Why monolithic approach?

When can we have problems with the added mass?

- Solid density approx. equal or less than fluid density
- Thin structures
- Highly flexible structures

We want to simulate **flexible floating thin structures**

Why monolithic approach?

When can we have problems with the added mass?

- Solid density approx. equal or less than fluid density ✓
- Thin structures
- Highly flexible structures

We want to simulate **flexible floating thin structures**

Why monolithic approach?

When can we have problems with the added mass?

- Solid density approx. equal or less than fluid density ✓
- Thin structures ✓
- Highly flexible structures

We want to simulate **flexible floating thin structures**

Why monolithic approach?

When can we have problems with the added mass?

- Solid density approx. equal or less than fluid density ✓
- Thin structures ✓
- Highly flexible structures ✓

We want to simulate **flexible floating thin structures**

Avoid partitioned scheme!!!

A note on the implementation

Final semi-discrete form:

$$
\begin{aligned}
&(\nabla\phi_h, \nabla w_h)_{\Omega_h} - (\kappa_{h,t}, w_h)_{\Gamma_{\mathrm{fs}}} - (\eta_{h,t}, w_h)_{\Gamma_{\mathrm{str}}} + \beta(\phi_{h,t} + g\kappa_h, \alpha\, w_h + v_h)_{\Gamma_{\mathrm{fs}}} \\
&+ (d_0\eta_{h,tt} + \phi_{h,t} + g\eta_h, u_h)_{\Gamma_{\mathrm{str}}} + (D_\rho\Delta\eta_h, \Delta u_h)_{\Gamma_{\mathrm{str}}} \\
&- (\langle D_\rho\Delta\eta_h\rangle, [\![\nabla u_h \cdot \mathbf{n}_\Lambda]\!])_{\Lambda_{\mathrm{str}}} - ([\![\nabla\eta_h \cdot \mathbf{n}_\Lambda]\!], \langle D_\rho\Delta u_h\rangle)_{\Lambda_{\mathrm{str}}} \\
&+ \frac{\gamma D_\rho}{h}([\![\nabla\eta_h \cdot \mathbf{n}_\Lambda]\!], [\![\nabla u_h \cdot \mathbf{n}_\Lambda]\!])_{\Lambda_{\mathrm{str}}} = (v_{\mathrm{in}}, w_h)_{\Gamma_{\mathrm{in}}} + (v_{\mathrm{out}}, w_h)_{\Gamma_{\mathrm{out}}}
\end{aligned}
$$

A note on the implementation

Final semi-discrete form:

$$
\begin{aligned}
&(\nabla\phi_h, \nabla w_h)_{\Omega_h} - (\kappa_{h,t}, w_h)_{\Gamma_{\text{fs}}} - (\eta_{h,t}, w_h)_{\Gamma_{\text{str}}} + \beta(\phi_{h,t} + g\kappa_h, \alpha\, w_h + v_h)_{\Gamma_{\text{fs}}} \\
&+ (d_0\eta_{h,tt} + \phi_{h,t} + g\eta_h, u_h)_{\Gamma_{\text{str}}} + (D_\rho\Delta\eta_h, \Delta u_h)_{\Gamma_{\text{str}}} \\
&- (\langle D_\rho\Delta\eta_h\rangle, [\![\nabla u_h \cdot \mathbf{n}_\Lambda]\!])_{\Lambda_{\text{str}}} - ([\![\nabla\eta_h \cdot \mathbf{n}_\Lambda]\!], \langle D_\rho\Delta u_h\rangle)_{\Lambda_{\text{str}}} \\
&+ \frac{\gamma D_\rho}{h}([\![\nabla\eta_h \cdot \mathbf{n}_\Lambda]\!], [\![\nabla u_h \cdot \mathbf{n}_\Lambda]\!])_{\Lambda_{\text{str}}} = (v_{\text{in}}, w_h)_{\Gamma_{\text{in}}} + (v_{\text{out}}, w_h)_{\Gamma_{\text{out}}}
\end{aligned}
$$

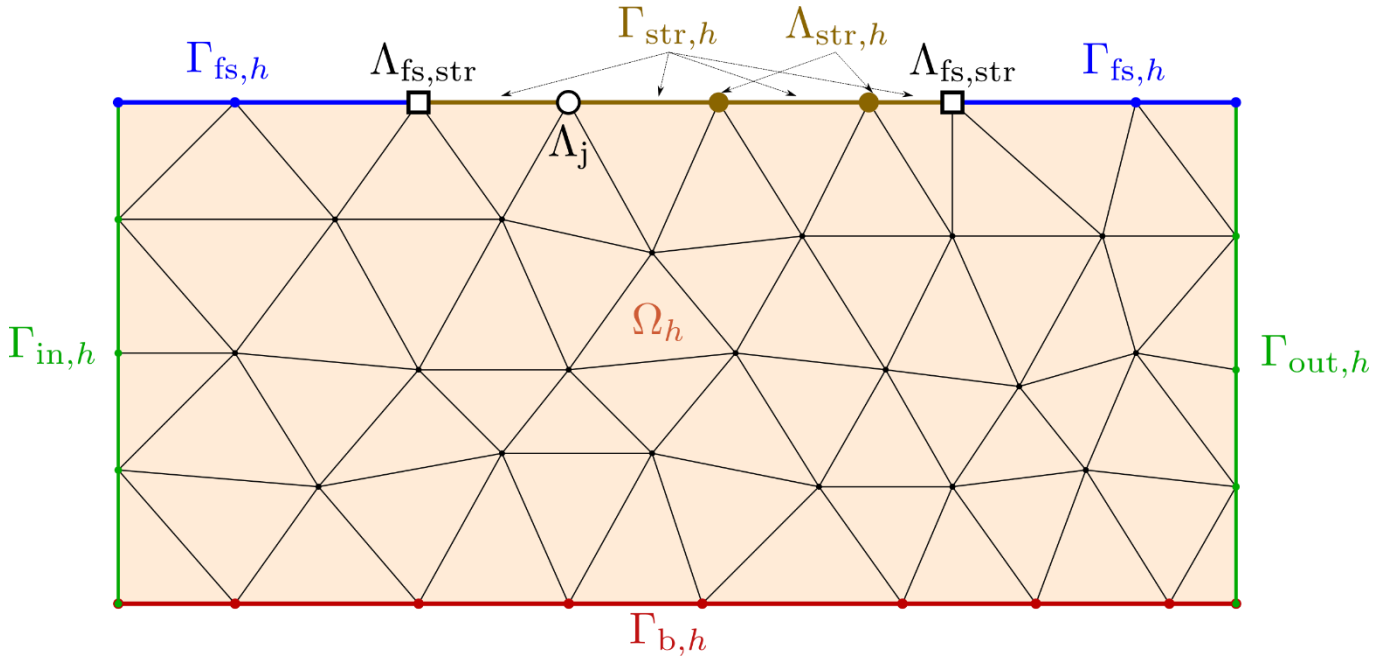
We have to define FE spaces and integrate on $\mathbb{R}^{\boldsymbol{d}}$, $\mathbb{R}^{\boldsymbol{d-1}}$ and $\mathbb{R}^{\boldsymbol{d-2}}$ with $\boldsymbol{d = 2, 3}$

A note on the implementation

Final semi-discrete form:

$$
\begin{aligned}
&(\nabla\phi_h, \nabla w_h)_{\Omega_h} - (\kappa_{h,t}, w_h)_{\Gamma_{\text{fs}}} - (\eta_{h,t}, w_h)_{\Gamma_{\text{str}}} + \beta(\phi_{h,t} + g\kappa_h, \alpha\, w_h + v_h)_{\Gamma_{\text{fs}}} \\
&+ (d_0\eta_{h,tt} + \phi_{h,t} + g\eta_h, u_h)_{\Gamma_{\text{str}}} + (D_\rho\Delta\eta_h, \Delta u_h)_{\Gamma_{\text{str}}} \\
&- (\langle D_\rho\Delta\eta_h\rangle, [\![\nabla u_h \cdot \mathbf{n}_\Lambda]\!])_{\Lambda_{\text{str}}} - ([\![\nabla\eta_h \cdot \mathbf{n}_\Lambda]\!], \langle D_\rho\Delta u_h\rangle)_{\Lambda_{\text{str}}} \\
&+ \frac{\gamma D_\rho}{h}([\![\nabla\eta_h \cdot \mathbf{n}_\Lambda]\!], [\![\nabla u_h \cdot \mathbf{n}_\Lambda]\!])_{\Lambda_{\text{str}}} = (v_{\text{in}}, w_h)_{\Gamma_{\text{in}}} + (v_{\text{out}}, w_h)_{\Gamma_{\text{out}}}
\end{aligned}
$$

We have to define FE spaces and integrate on $\mathbb{R}^{\boldsymbol{d}}$, $\mathbb{R}^{\boldsymbol{d-1}}$ and $\mathbb{R}^{\boldsymbol{d-2}}$ with $\boldsymbol{d = 2, 3}$



A note on the implementation

Final semi-discrete form:

$$
\begin{aligned}
&(\nabla\phi_h, \nabla w_h)_{\Omega_h} - (\kappa_{h,t}, w_h)_{\Gamma_{\text{fs}}} - (\eta_{h,t}, w_h)_{\Gamma_{\text{str}}} + \beta(\phi_{h,t} + g\kappa_h, \alpha\, w_h + v_h)_{\Gamma_{\text{fs}}} \\
&+ (d_0\eta_{h,tt} + \phi_{h,t} + g\eta_h, u_h)_{\Gamma_{\text{str}}} + (D_\rho\Delta\eta_h, \Delta u_h)_{\Gamma_{\text{str}}} \\
&- (\langle D_\rho\Delta\eta_h\rangle, [\![\nabla u_h \cdot \mathbf{n}_\Lambda]\!])_{\Lambda_{\text{str}}} - ([\![\nabla\eta_h \cdot \mathbf{n}_\Lambda]\!], \langle D_\rho\Delta u_h\rangle)_{\Lambda_{\text{str}}} \\
&+ \frac{\gamma D_\rho}{h}([\![\nabla\eta_h \cdot \mathbf{n}_\Lambda]\!], [\![\nabla u_h \cdot \mathbf{n}_\Lambda]\!])_{\Lambda_{\text{str}}} = (v_{\text{in}}, w_h)_{\Gamma_{\text{in}}} + (v_{\text{out}}, w_h)_{\Gamma_{\text{out}}}
\end{aligned}
$$

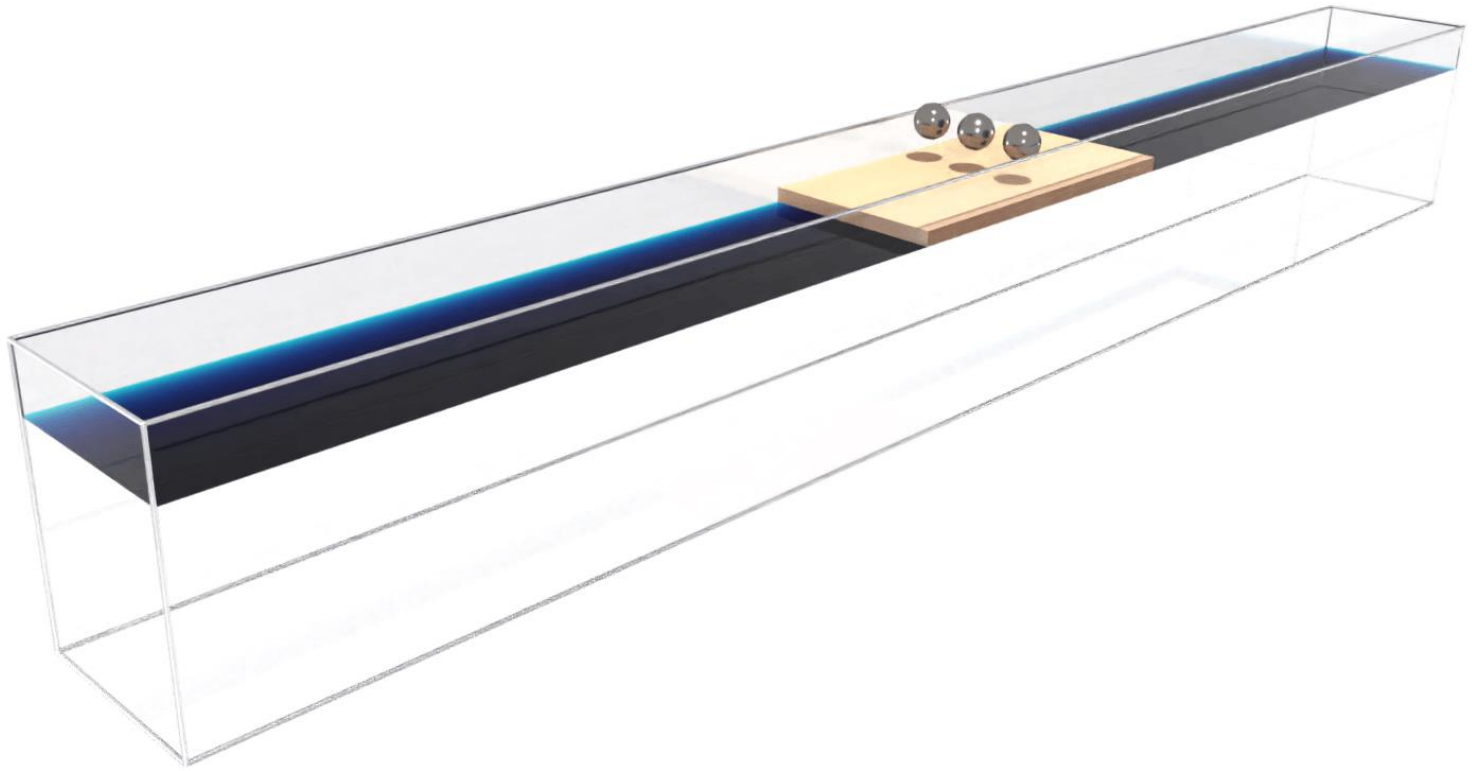
This has been implemented in **Gridap.jl**, a pure Julia FE library

```
residual(t,(ϕ,κ,η),(w,u,v)) = ∫( ∇(w)⋅∇(ϕ) )dΩ - ∫( ∂t(κ)*w )dΓfs - ∫( ∂t(η)*w )dΓb + ∫( β * (∂t(ϕ) + g*κ) * (u + α*w) )dΓfs
                              ∫( (d₀*∂tt(η) + ∂t(ϕ) + g*η) * v )dΓb + ∫( D*Δ(v)*Δ(η) )dΓb +
                              ∫( D * ( - jump(∇(v)⋅nΛb) * mean(Δ(η)) - mean(Δ(v)) * jump(∇(η)⋅nΛb) ) )dΛb +
                              ∫( D*γ/h * ( jump(∇(v)⋅nΛb) * jump(∇(η)⋅nΛb) ) )dΛb
op = TransientFEOperator(residual, X, Y)
(ϕh,κh,ηh) = solve(ODE_solver,x₀,t₀,tf)
```

Gridap

Solution of mixed-dimensional PDEs

Not only d-1 structures, also **d-2 fluid-structure-structure** interaction!

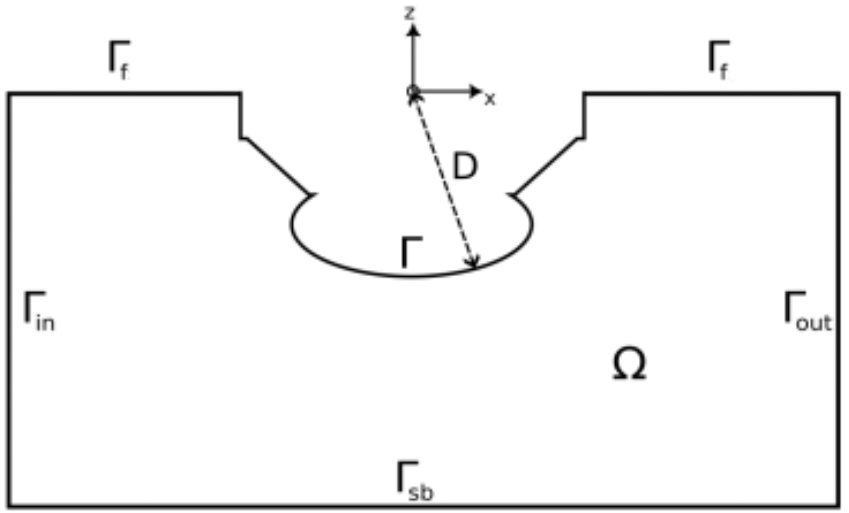


How do we use Gridap.jl ecosystem?

- Solution of mixed-dimensional PDEs
- **Solution of PDEs in complex geometries**
- Solution of highly nonlinear PDEs

Hydrodynamics of floating structures

$$
\begin{aligned}
\Delta\hat{\phi}_k &= 0 && \text{in} \quad \Omega, \\
-\omega_k^2\hat{\phi}_k + g\nabla\hat{\phi}_k \cdot \boldsymbol{n}_{\Gamma_f} &= 0 && \text{on} \quad \Gamma_f, \\
\nabla\hat{\phi}_k \cdot \boldsymbol{n}_\Gamma + i\omega_k\hat{\boldsymbol{u}}_k \cdot \boldsymbol{n}_{\Gamma,j} &= 0 && \text{on} \quad \Gamma, \\
-\omega_k^2\boldsymbol{M}_\rho\hat{\boldsymbol{u}}_k + \int_\Gamma \left(-i\omega\hat{\phi}_k + g\hat{\boldsymbol{u}}_k \cdot \boldsymbol{n}_z\right)\boldsymbol{n}_{\Gamma,j}d\Gamma &= \boldsymbol{0} && \text{on} \quad \Gamma.
\end{aligned}
$$

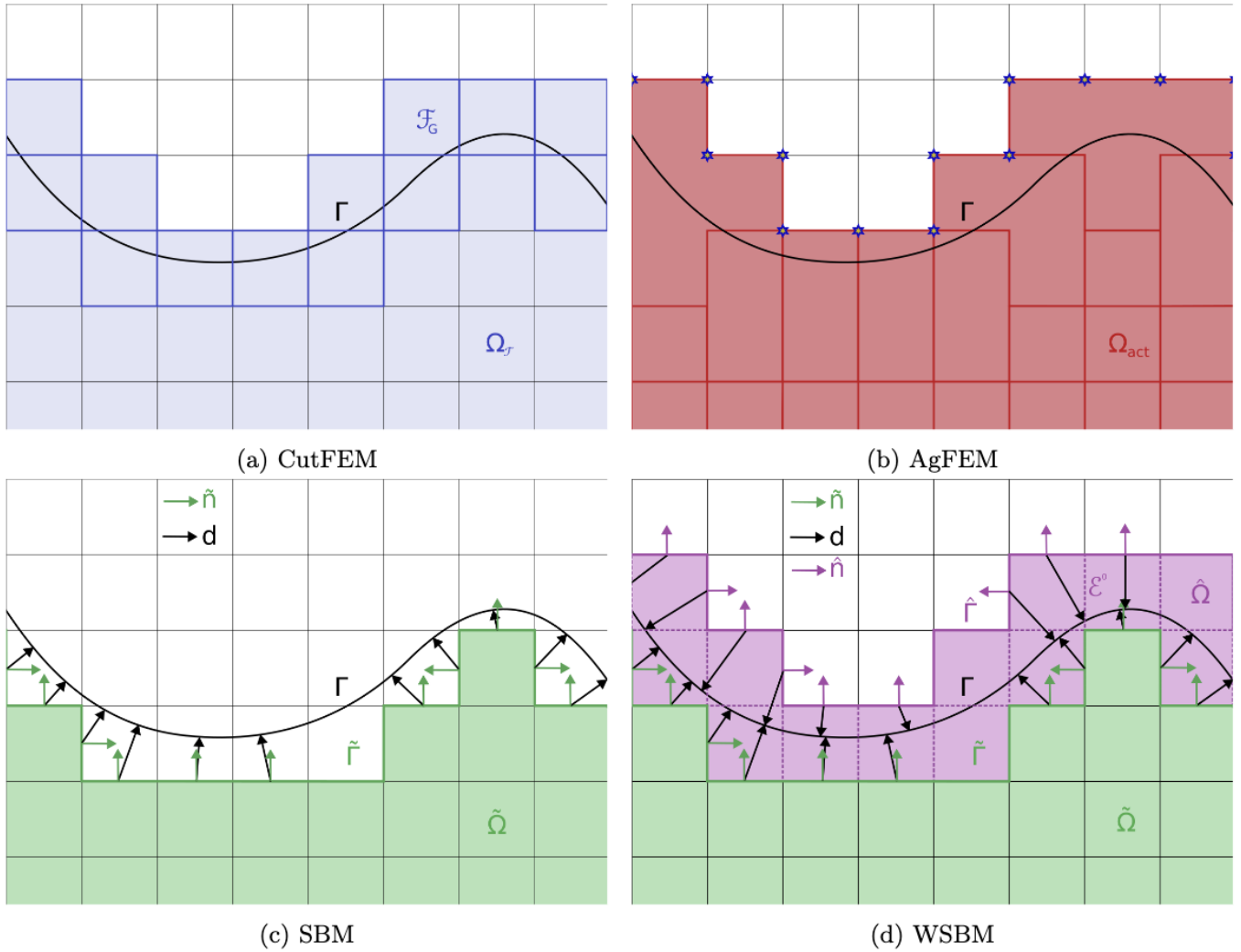


Weak form of the problem:

$$
\begin{aligned}
&(\nabla w_h, \nabla\phi_h)_\Omega - \frac{\omega^2}{g}(w_h, \phi_h)_{\Gamma_f} + i\omega\left(w_h, \boldsymbol{u}_h \cdot \boldsymbol{n}_{\Gamma,j}\right)_\Gamma + \\
&-i\omega\left(\boldsymbol{v}_h, \phi_h\boldsymbol{n}_{\Gamma,j}\right)_\Gamma - \omega^2\frac{M_\rho}{|\Gamma|}(\boldsymbol{v}_h, \boldsymbol{u}_h)_\Gamma + g\left(\boldsymbol{v}_h, (\boldsymbol{n}_z \cdot \boldsymbol{u}_h)\,\boldsymbol{n}_{\Gamma,j}\right)_\Gamma,
\end{aligned}
$$

Hydrodynamics of floating structures

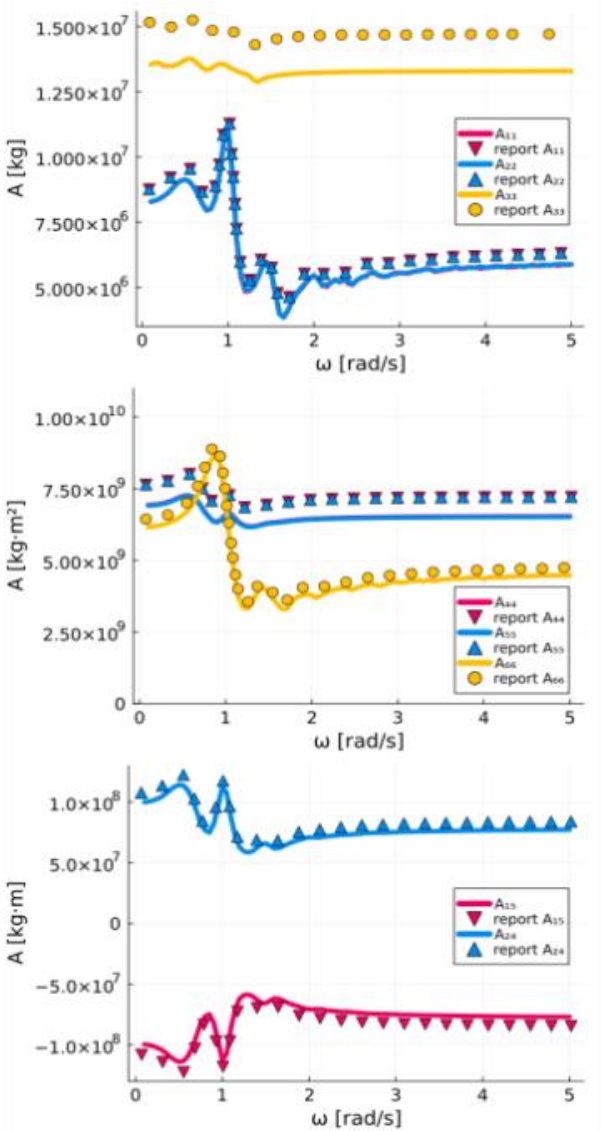
We use **Unfitted Finite Element** methods:

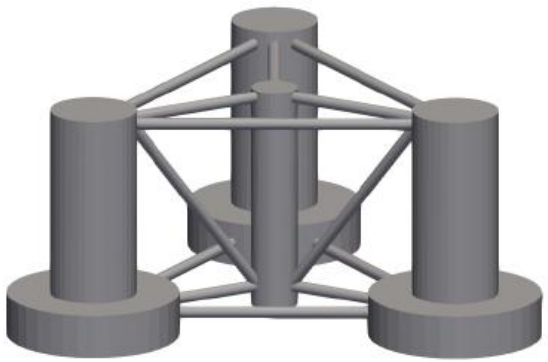


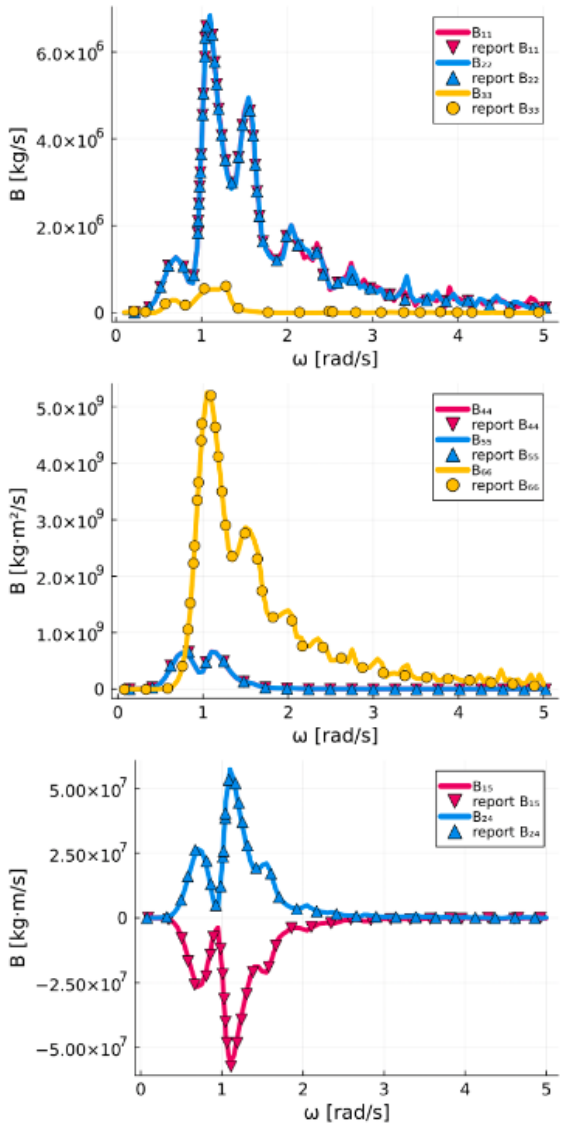
(a) CutFEM (b) AgFEM

(c) SBM (d) WSBM

Hydrodynamics of floating structures



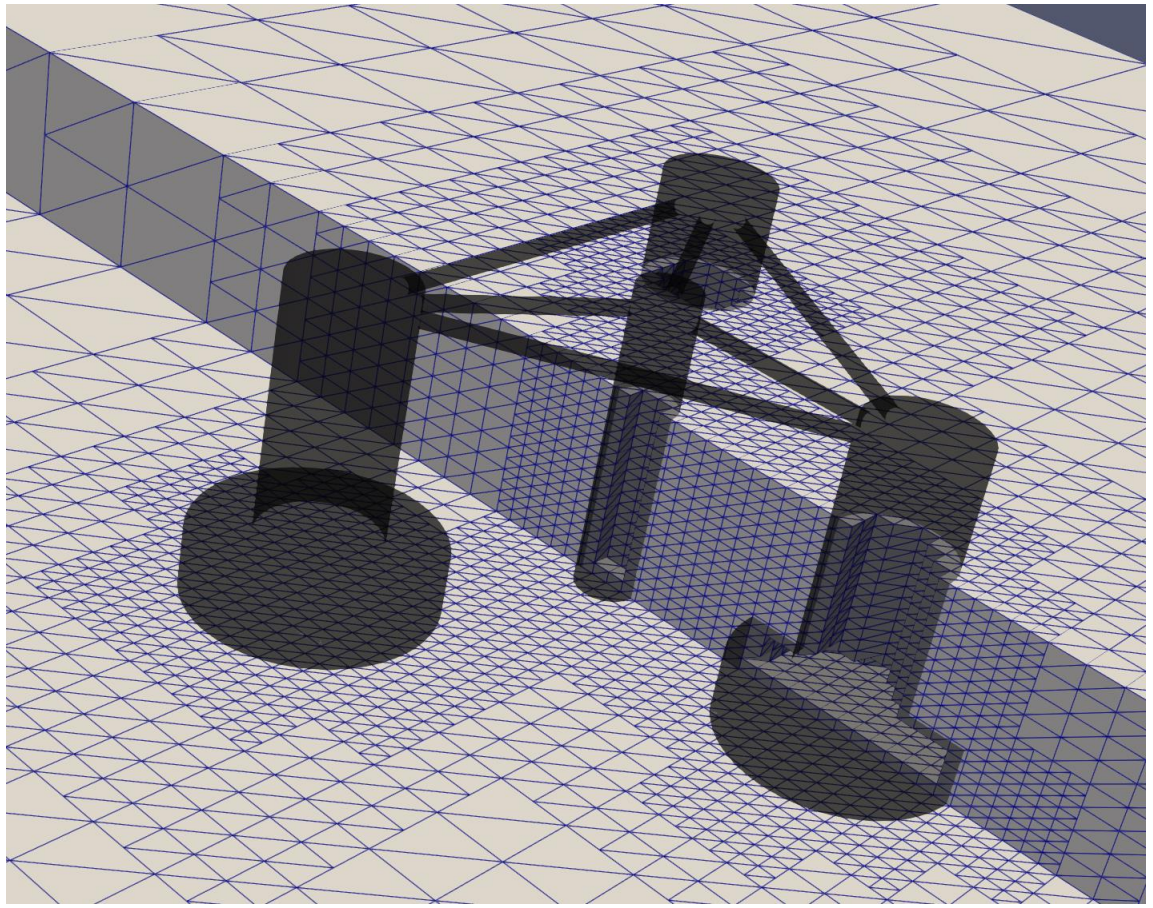




Hydrodynamics of floating structures

Further work on:

- h-adaptive refinement
- parallel aggregation in octrees
- multi-body simulation

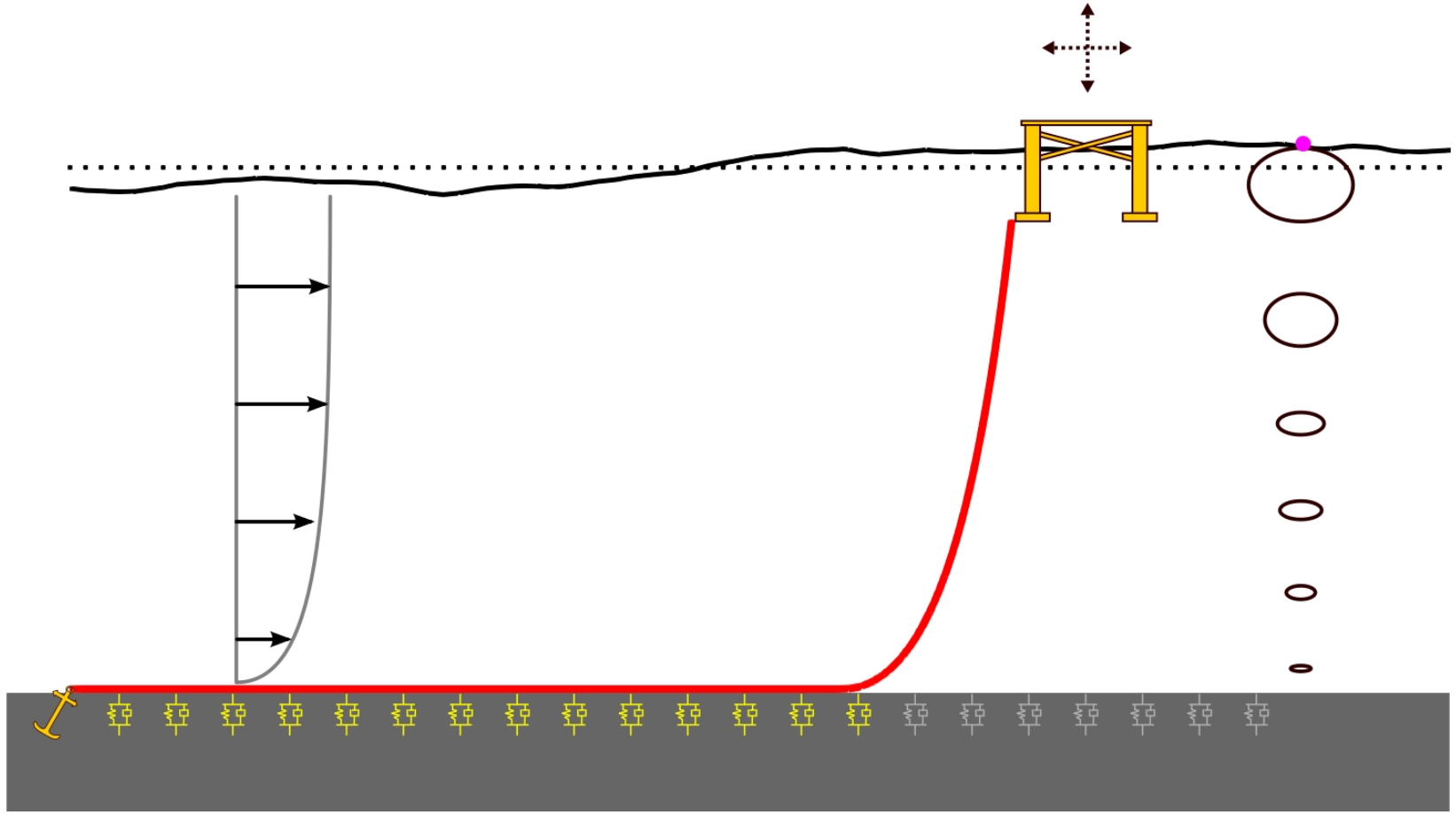


How do we use Gridap.jl ecosystem?

- Solution of mixed-dimensional PDEs
- Solution of PDEs in complex geometries
- **Solution of highly nonlinear PDEs**

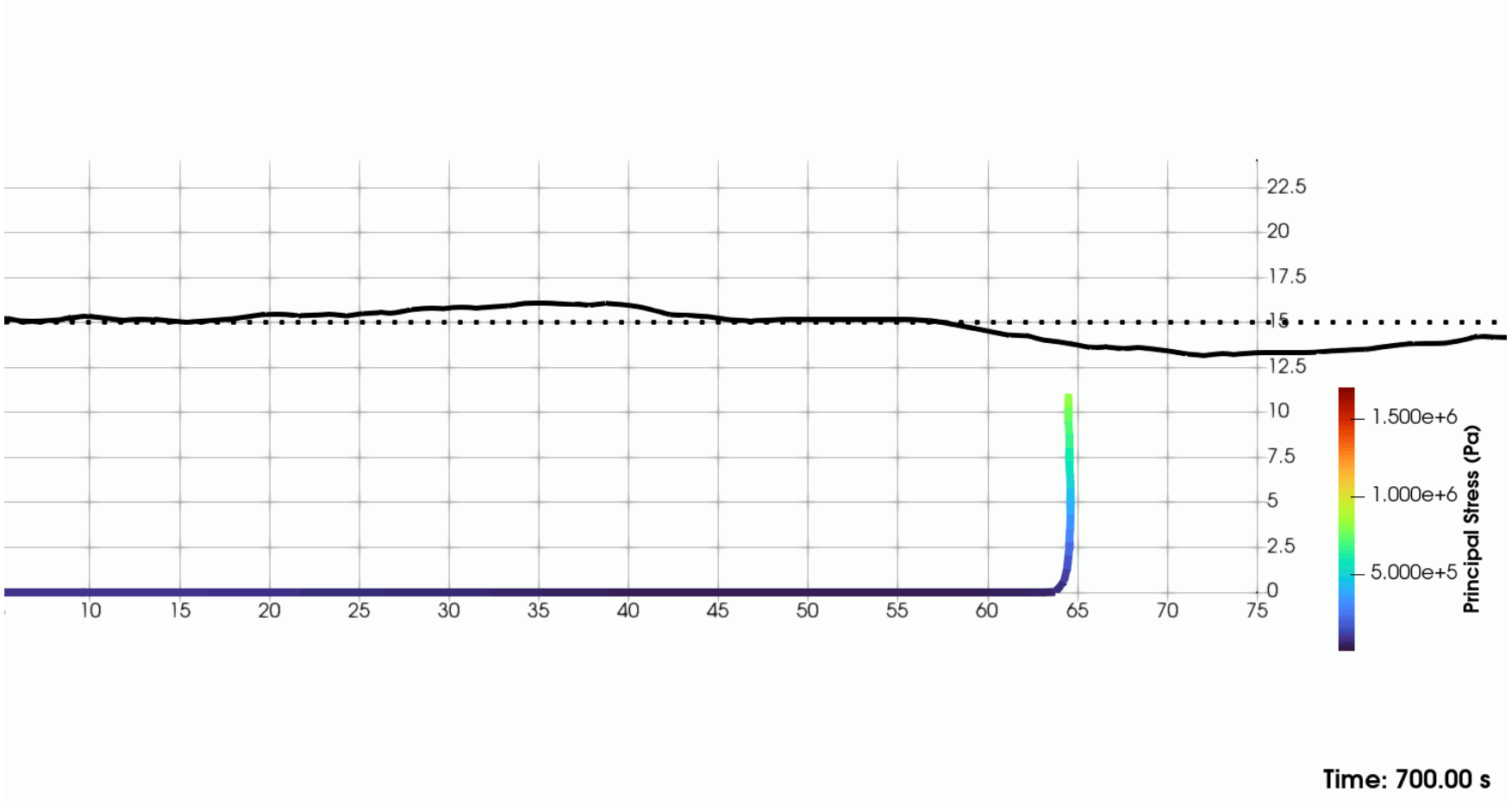
Solution of highly nonlinear PDEs

Dynamic mooring lines with nonlinear viscoelastic materials



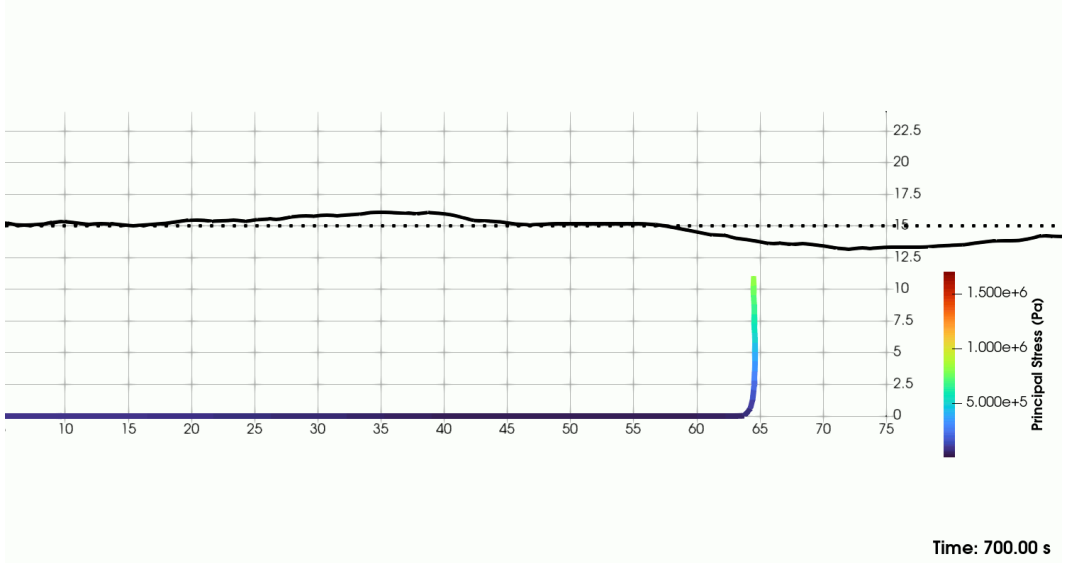
Solution of highly nonlinear PDEs

Dynamic mooring lines with nonlinear viscoelastic materials



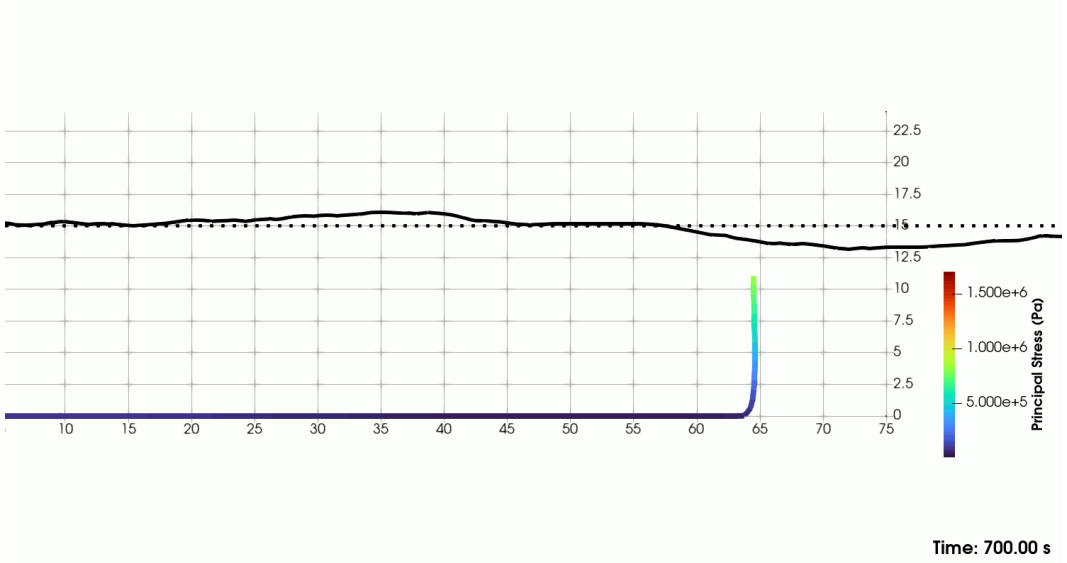
What are the issues here?

- Geometrical nonlinearity (large displacements)
- 1D structure in a 3D domain
- Dynamic behavior
- Snap loads (shocks)
- (nonlinear) viscoelastic material

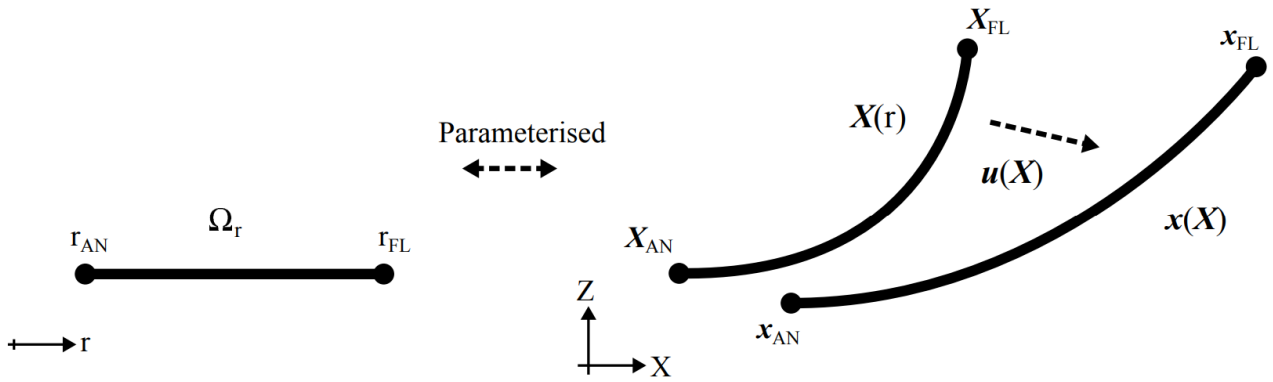


How does Gridap.jl help?

- **Geometrical nonlinearity (large displacements)**
- **1D structure in a 3D domain**
- Dynamic behavior
- Snap loads (shocks)
- (nonlinear) viscoelastic material



Tangential Differential Calculus



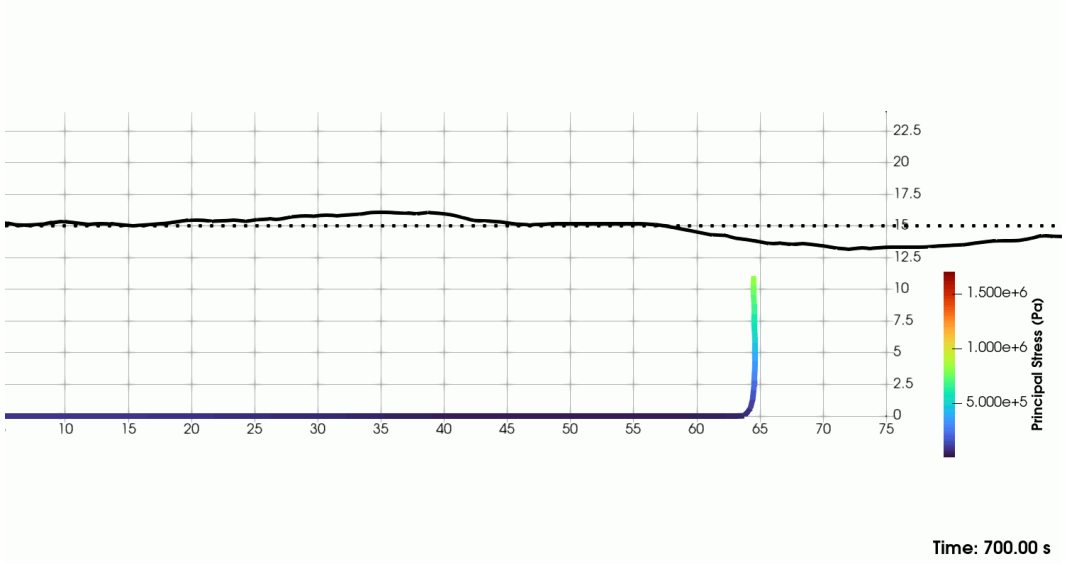
$$\rho\,\ddot{\boldsymbol{u}} - \mathrm{div}_\Gamma\,\boldsymbol{\sigma}(\boldsymbol{u}) - \boldsymbol{b}(\boldsymbol{x}) = 0 \quad \forall \boldsymbol{x} \in \Gamma_{\boldsymbol{x}}$$

$$\rho_0\,\ddot{\boldsymbol{u}}(\boldsymbol{X}) - \mathrm{Div}_\Gamma\,\mathbf{K}(\boldsymbol{u}) - \boldsymbol{B}(\boldsymbol{X}) = 0 \quad \forall \boldsymbol{X} \in \Gamma_{\boldsymbol{X}},$$

- $\boldsymbol{r} \in [\boldsymbol{0}, \boldsymbol{L_0}]$
- $\boldsymbol{X_{AN}} = \boldsymbol{X}(\boldsymbol{r} = \boldsymbol{0})$ Anchor position
- $\boldsymbol{X_{FL}} = \boldsymbol{X}(\boldsymbol{r} = \boldsymbol{L_0})$ Fairlead position

How does Gridap.jl help?

- Geometrical nonlinearity (large displacements)
- 1D structure in a 3D domain
- **Dynamic behavior**
- **Snap loads (shocks)**
- (nonlinear) viscoelastic material

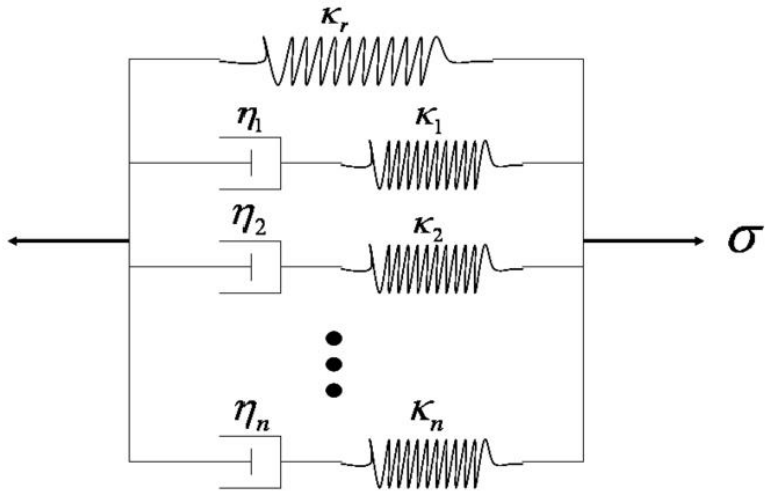


Implicit ODE solvers with **Automatic Differentiation**

```
U = ... # Trial FE Space on r∈[0,1]
V = ... # Test FE Space on r∈[0,1]
ξ = ... # Cell-wise state variables evaluated at quadrature points
ℛ(u,v) = ∫( ( v⋅(ρ∗∂tt(u)) + (∇(v)' ⋅ Qᵀ) ⊙ (K∘(Qᵀ, P, ∇(u)), ) ) ∗ J )dΩ  - ∫( ( v ⋅ Fρ )∗J )dΩ
op_S = TransientFEOperator(ℛ, U, V)
ode_solver = ... # ODE solver for time integration
uₕ = solve(ode_solver, op_D, t0, T, (u₀,u₀ₜ))
```

How does Gridap.jl help?

- Geometrical nonlinearity (large displacements)
- 1D structure in a 3D domain
- Dynamic behavior
- Snap loads (shocks)
- **(nonlinear) viscoelastic material**



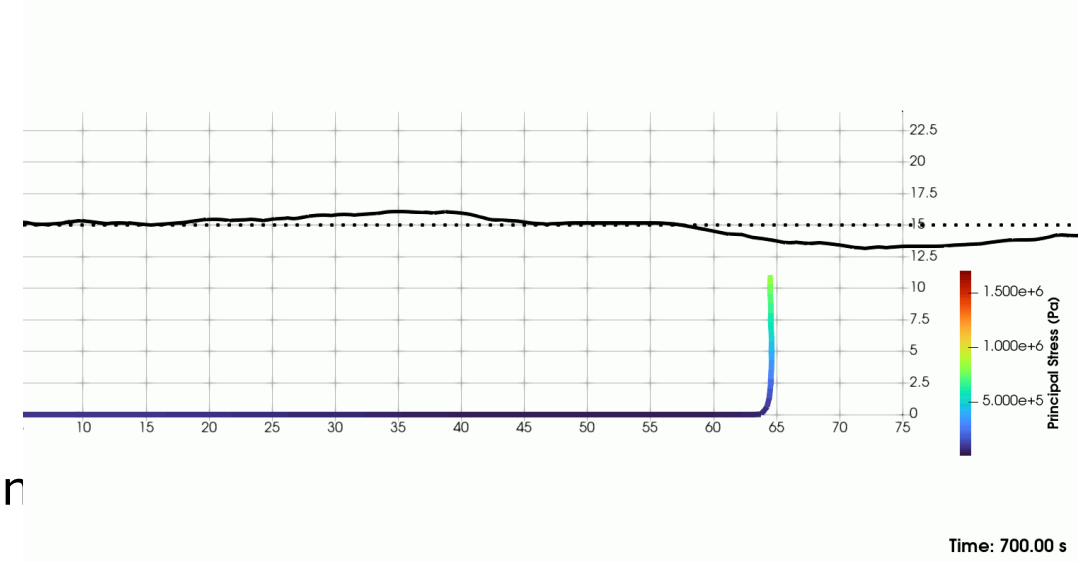
$$\varepsilon_{ve} = g_0(\sigma)D_0\sigma(t) + g_1(\sigma)\int_0^t \Delta D\left(\psi(t) - \psi(\tau)\right)\frac{d(g_2(\sigma)\sigma)}{dt}\,d\tau$$

Implicit ODE solvers with **Automatic Differentiation**

```
U = ... # Trial FE Space on r∈[0,1]
V = ... # Test FE Space on r∈[0,1]
ξ = ... # Cell-wise state variables evaluated at quadrature points
ℛ(u,v) = ∫( ( v·(ρ*∂tt(u)) + (∇(v)' · Qᵀ) ⊙ (K∘(Qᵀ, P, ∇(u)), ξ) ) * J )dΩ  - ∫( ( v · Fρ )*J )dΩ
op_S = TransientFEOperator(ℛ, U, V)
ode_solver = ... # ODE solver for time integration
uₕ = solve(ode_solver, op_D, t0, T, (u₀,u₀ₜ))
```

How does Gridap.jl help?

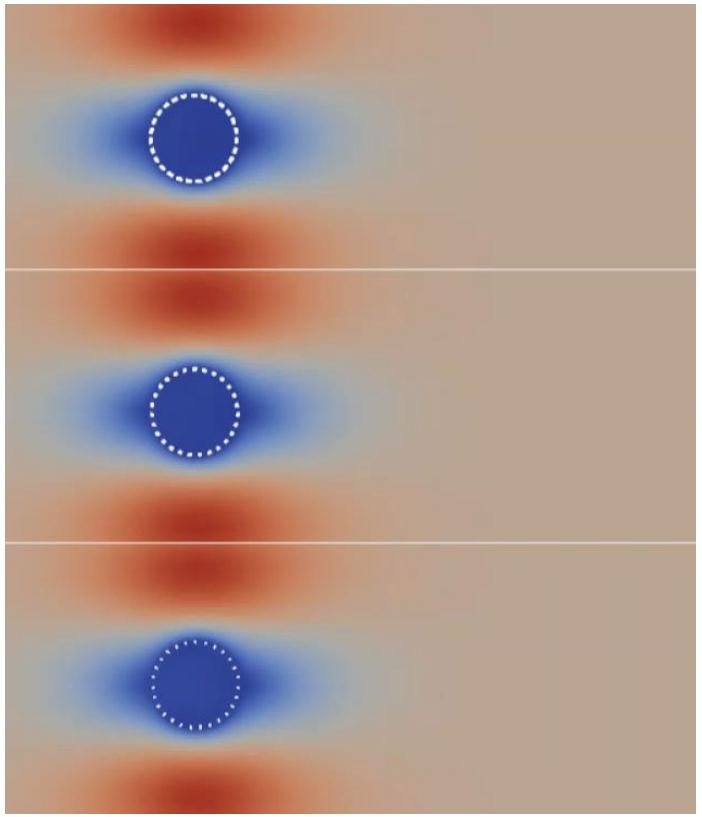
- Dynamic behavior -> Transient PDEs
- Snap loads (shocks) -> Implicit solvers with AD
- Geometrical nonlinearity (large displacements)
- 1D structure in a 3D domain -> Tangential Differen
- (nonlinear) viscoelastic material -> Handling of state variables + AD



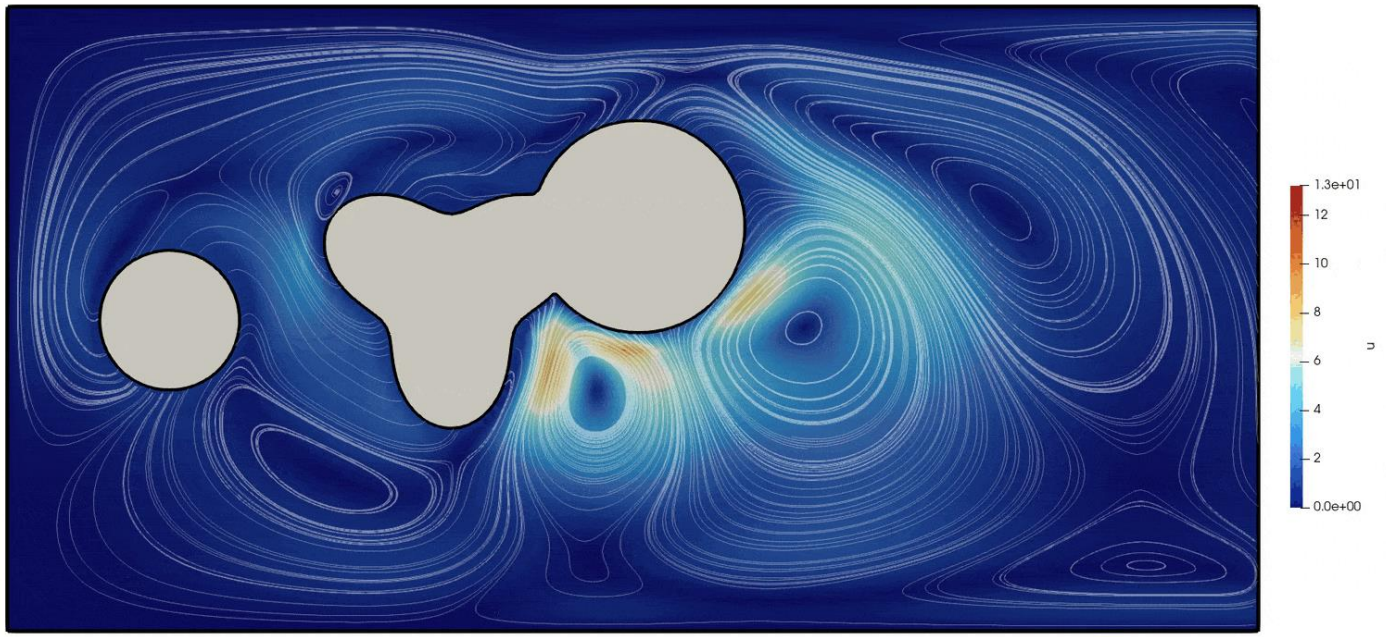
How do we use Gridap.jl ecosystem?

- Solution of mixed-dimensional PDEs
- Solution of PDEs in complex geometries
- Solution of highly nonlinear PDEs
- Many more...

How do we use Gridap.jl ecosystem?



HPC for turbulent flows
(GridapDistributed.jl)



Advanced FEs for moving domains with topology changes
(GridapEmbedded + GridapTopOpt.jl ?)

GridapSolvers.jl
GridapROMS.jl
...

Want to know more?

j.o.colomesgene@tudelft

https://www.linkedin.com/in/oriolcolomes/

https://tudelftcmoe.super.site/

Thank you!