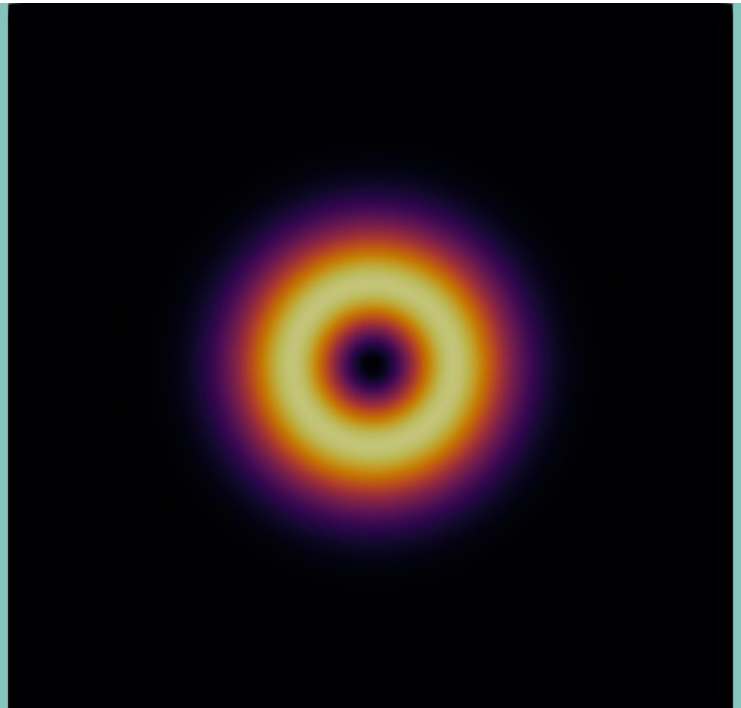
Solving plasma problems using adaptive multiphysics coupling

Simon Candelaresi, Michael Schlottke-Lakemper

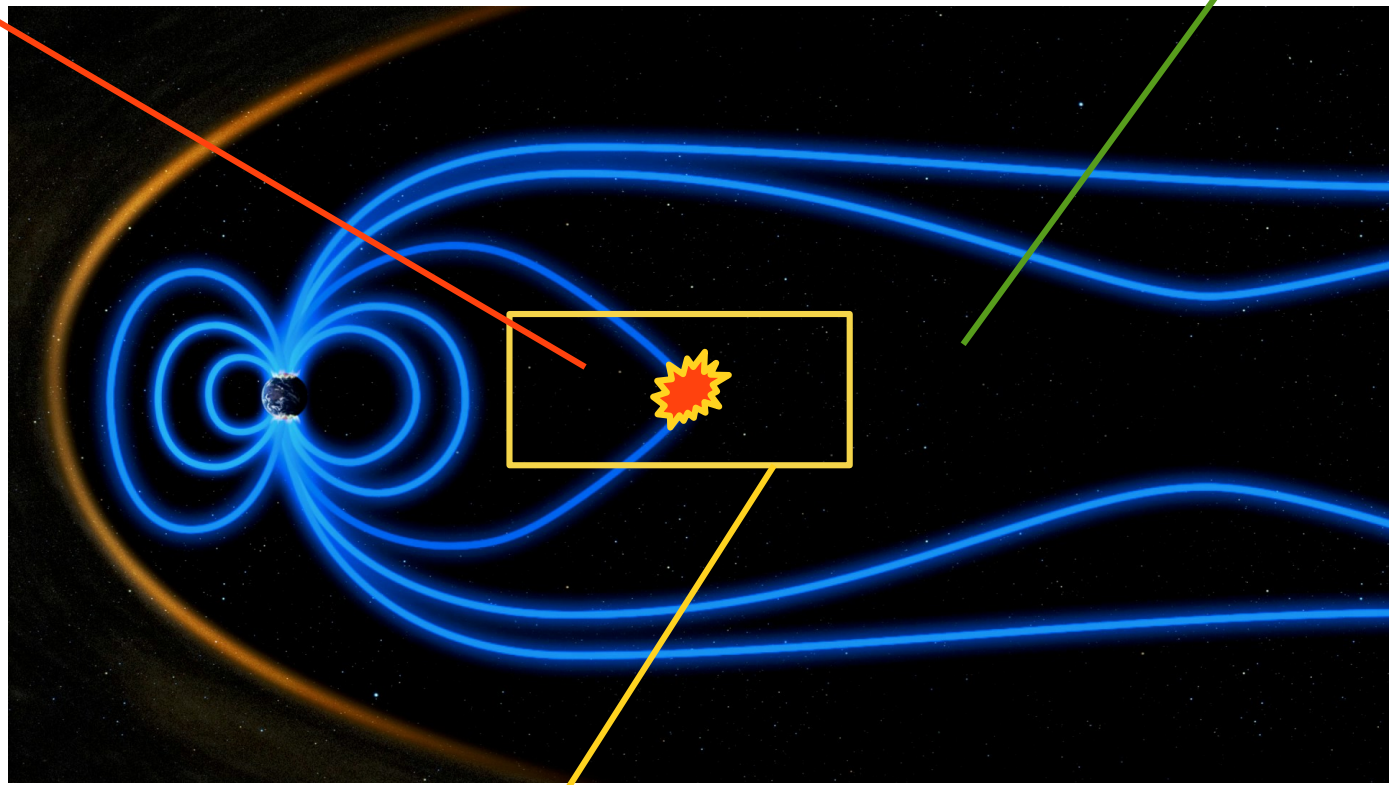


Universität
Augsburg
University

Modeling

kinetic model
(expensive)

magnetohydrodynamics (MHD)
(cheap)



(ESA)

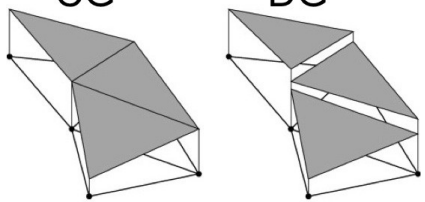
interface coupling

Trixi.jl

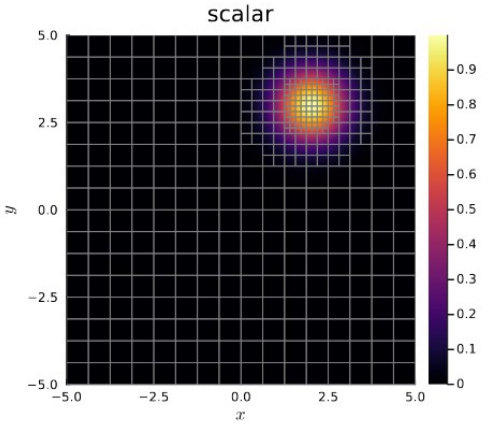
open source:
github.com/trixi-framework/Trixi.jl

Discontinuous Galerkin

CG DG

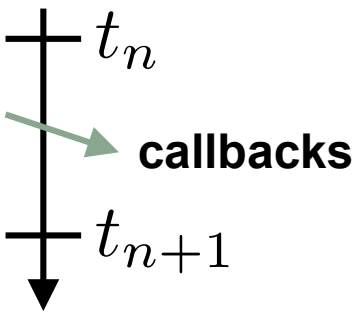


(Stack Overflow)



AMR

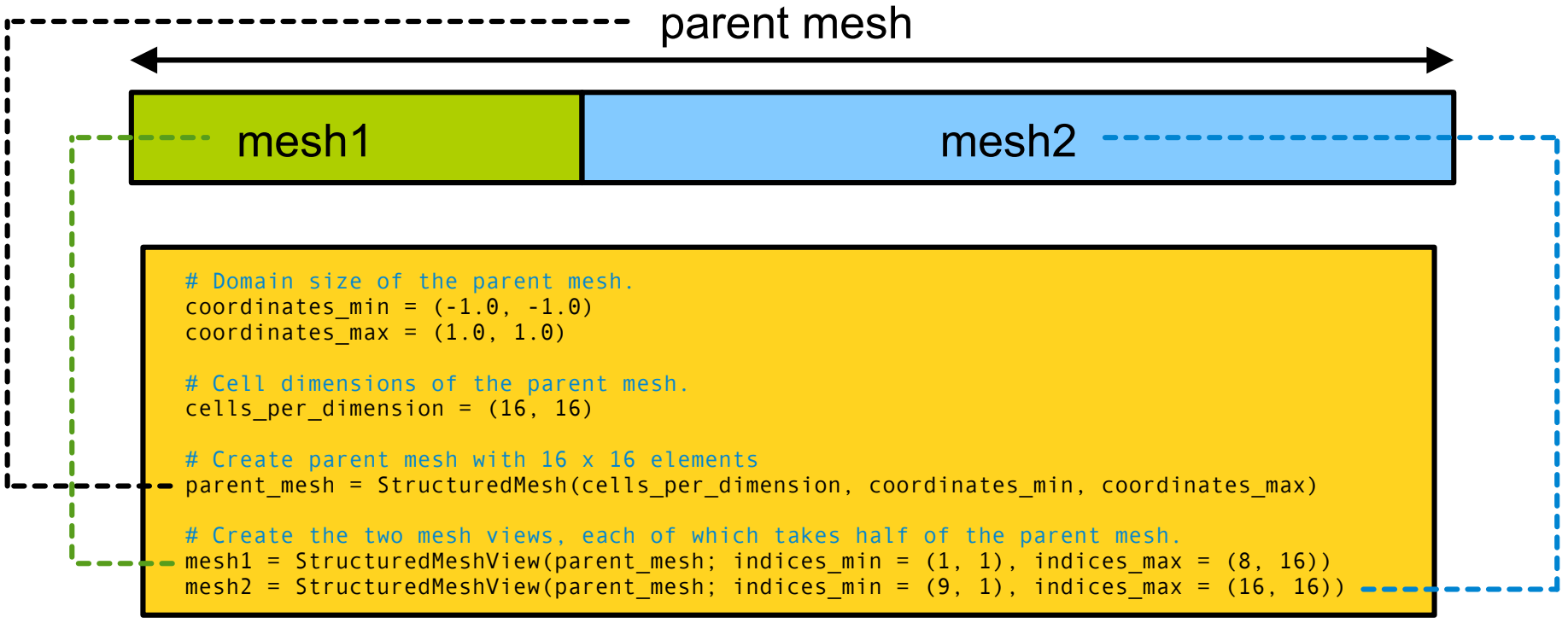
julia



OrdinaryDiffEq (SciML)

```
using Trixi

equations = LinearScalarAdvectionEquation2D((0.5, -0.3)
solver = DGSEM(polydeg = 3)
cells_per_dimension = (16, 16)

mesh = StructuredMesh(cells_per_dimension, (-1, -1), (1, 1))

semi = SemidiscretizationHyperbolic(mesh, equations,
                                    initial_condition, solver)
ode = semidiscretize(semi, (0.0, 1.0));

stepsize_callback = StepsizeCallback(cfl = 1.6)
callbacks = CallbackSet(stepsize_callback)

sol = solve(ode, CarpenterKennedy2N54(),
            dt = 1.0, callback = callbacks);
```

Mesh Views

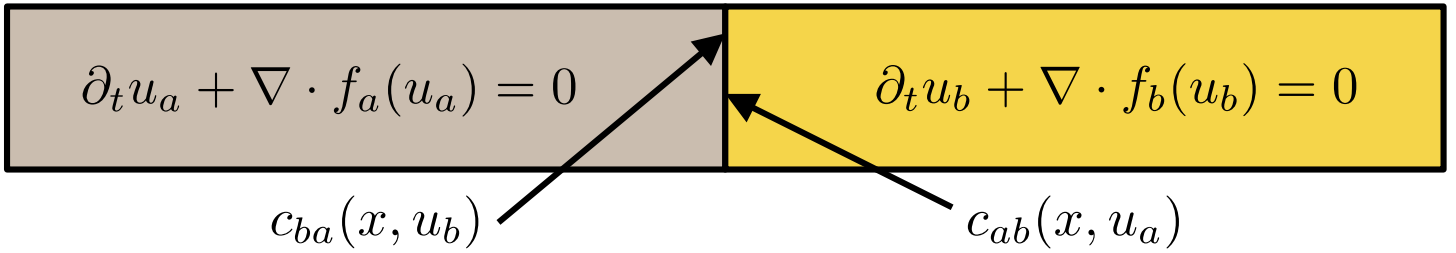


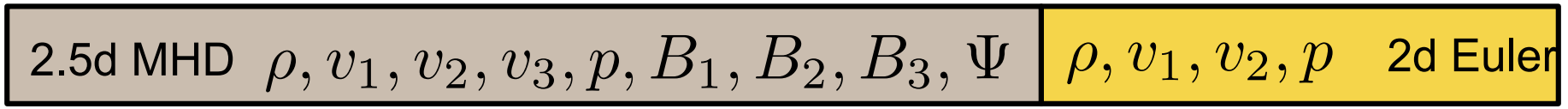

```
# Domain size of the parent mesh.
coordinates_min = (-1.0, -1.0)
coordinates_max = (1.0, 1.0)

# Cell dimensions of the parent mesh.
cells_per_dimension = (16, 16)

# Create parent mesh with 16 x 16 elements
parent_mesh = StructuredMesh(cells_per_dimension, coordinates_min, coordinates_max)

# Create the two mesh views, each of which takes half of the parent mesh.
mesh1 = StructuredMeshView(parent_mesh; indices_min = (1, 1), indices_max = (8, 16))
mesh2 = StructuredMeshView(parent_mesh; indices_min = (9, 1), indices_max = (16, 16))
```

- Mesh views appear like proper meshes for Trixi.jl.
- Flexible and potentially adaptive usage of sub-domains.

Coupling via Converter Functions

Two system with any number of shared variables, including 0:

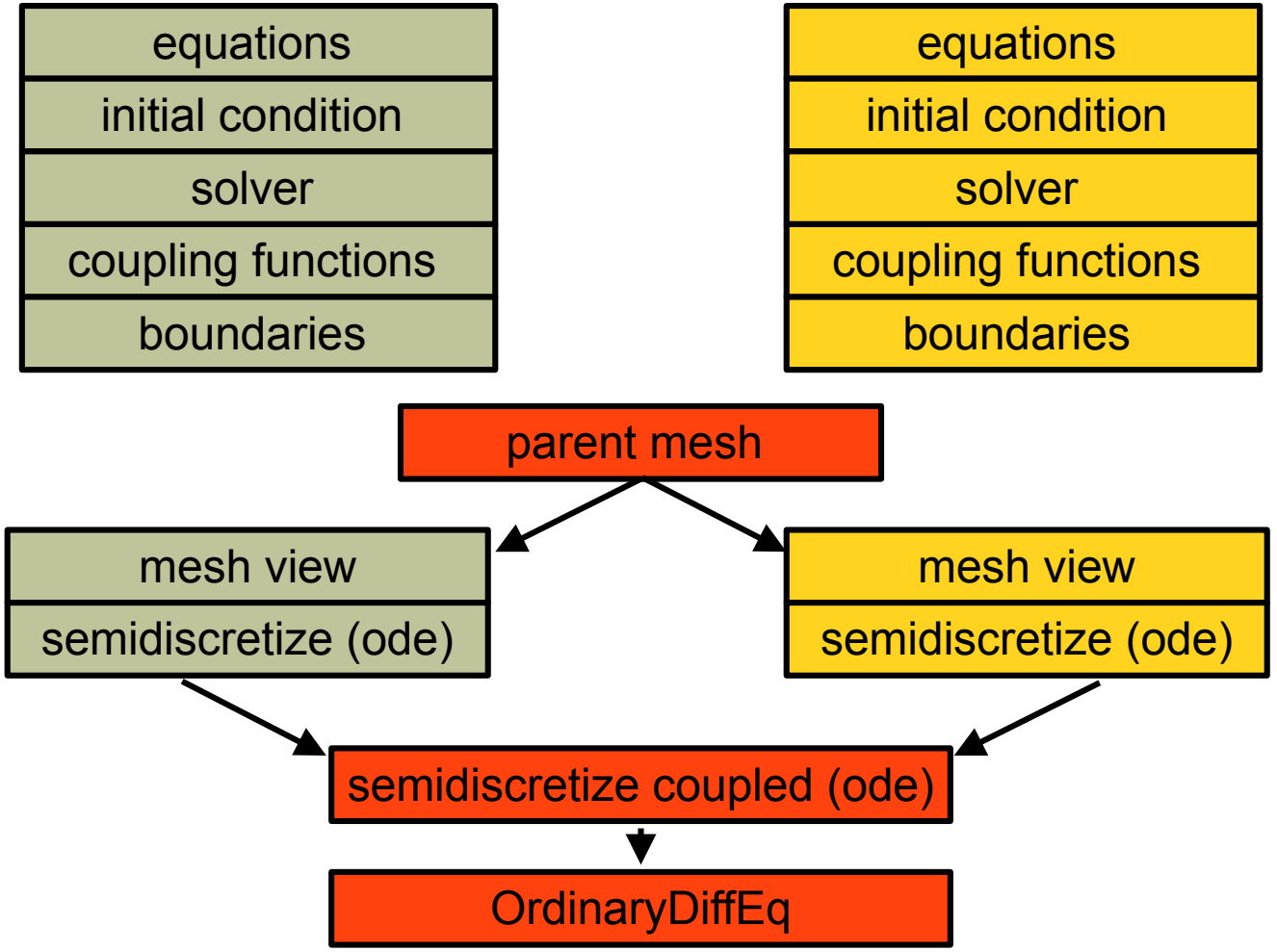





```
coupling_function12 = (x, u, equations_other, equations_own)
                      -> SVector(u[1], u[2], u[3], 0.0, u[4], 0.0, 0.0, 0.0, 0.0)
coupling_function21 = (x, u, equations_other, equations_own) -> SVector(u[1], u[2], u[3], u[5])
```

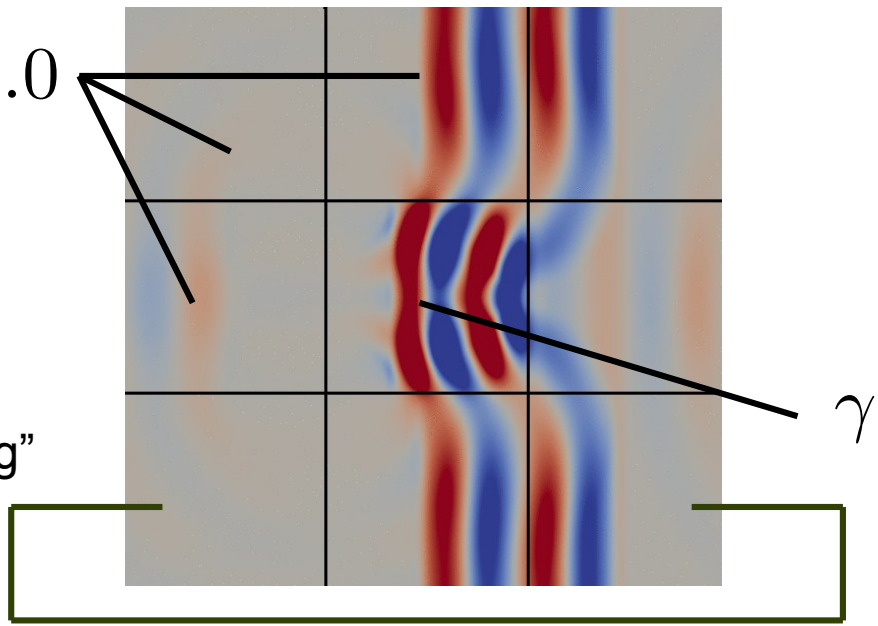
- User can define converter functions.
- Any pair of systems can be coupled.

Work Flow Coupling



Isothermal-Polytropic

$$\partial t \begin{pmatrix} \rho \\ \rho v_1 \\ \rho v_2 \end{pmatrix} + \partial x \begin{pmatrix} \rho v_1 \\ \rho v_1^2 + \rho^\gamma \\ \rho v_1 v_2 \end{pmatrix} + \partial y \begin{pmatrix} \rho v_2 \\ \rho v_1 v_2 \\ \rho v_2^2 + \rho^\gamma \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

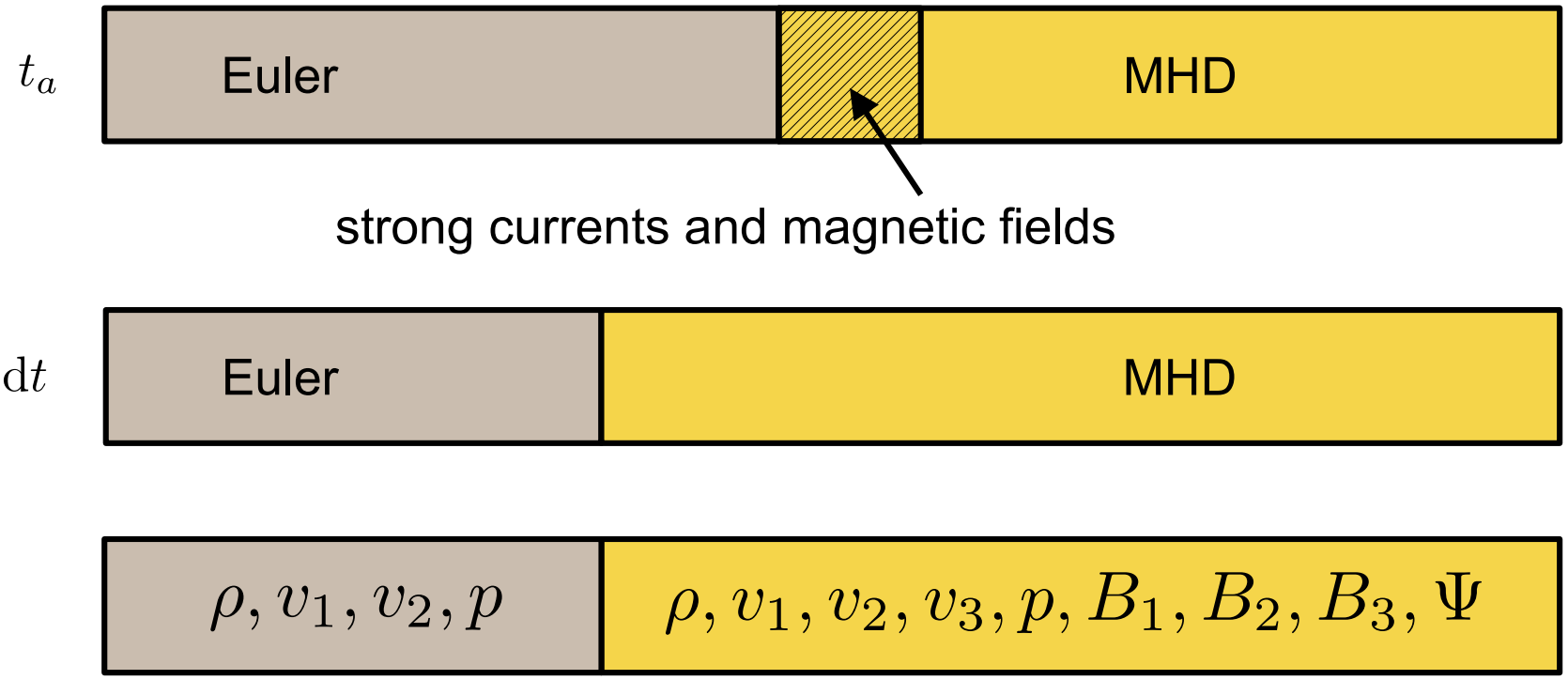


coupling functions:

$$c_{ab} = u^b$$

$$c_{ba} = u^a$$

Adaptive Coupling

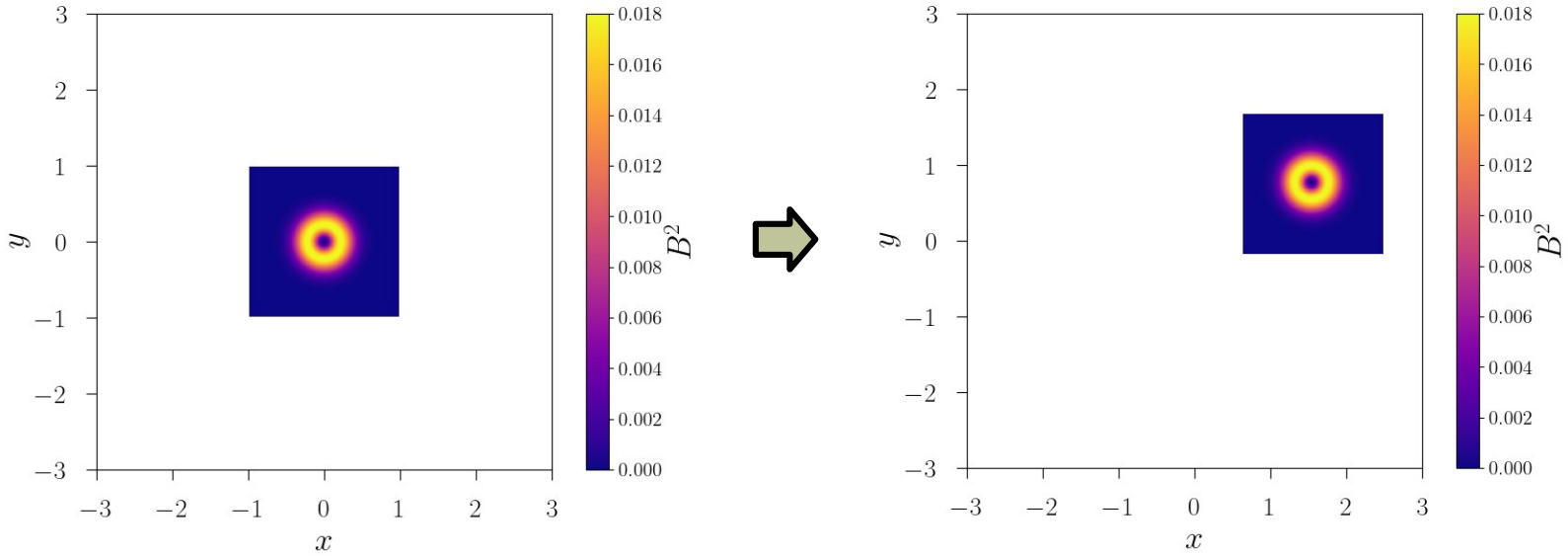


- Use callback functions to remesh.
- Use coupling functions to copy data.

Adaptive Model Selection

1. Generate the new grid (mesh views).
2. Write new u-solution vectors
3. Generate new ODE for OrdinaryDiffEq (integrator).
4. Reinitialize ODE integrator with new problem and new solution vector.

Euler and MHD



(movies)

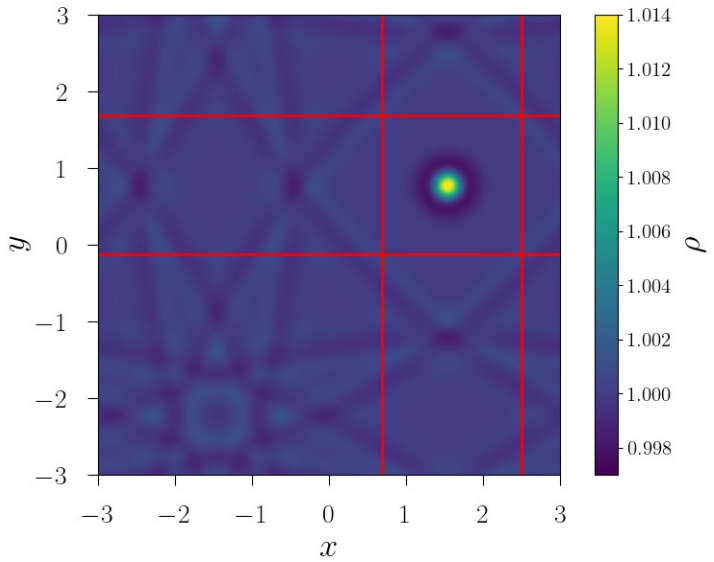
Time spent on coupled boundaries: 0.8%.

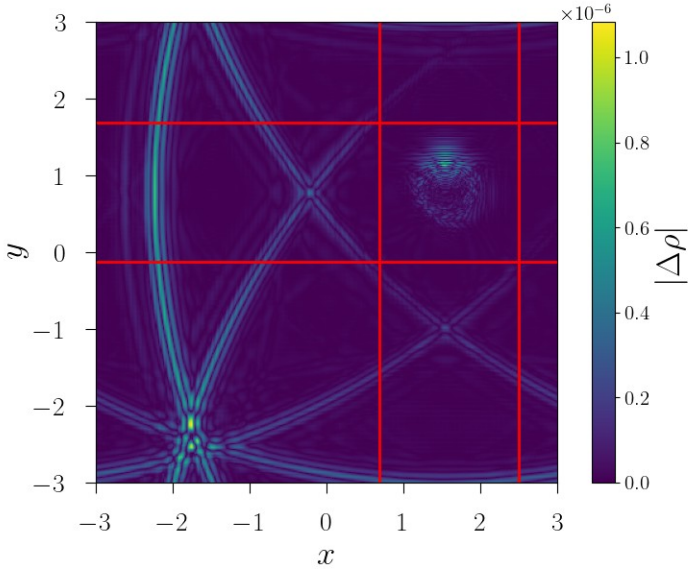
Time spent on model selection: 1.08s per model selection

Timings: coupled → 765.6s
full MHD → 1905.3s

Estimated maximum speed up: 3.31x; here: 2.49x

Density Evolution





full MHD vs. coupled

- No artificial discontinuities.
- Small differences to full MHD simulation.

Conclusion

- Flexible coupling through converter functions.
- Free domain definitions.
- Adaptive coupling with arbitrary criteria.
- Coupled hierarchy of models.

simon.candelaresi@uni-a.de, simon.candelaresi@gmail.com