Fully Lagrangian Approach in OpenFOAM: implementation and preliminary results

T.S. Zaripov

Vortex rings and related problems
24th August, 2018
University of Brighton

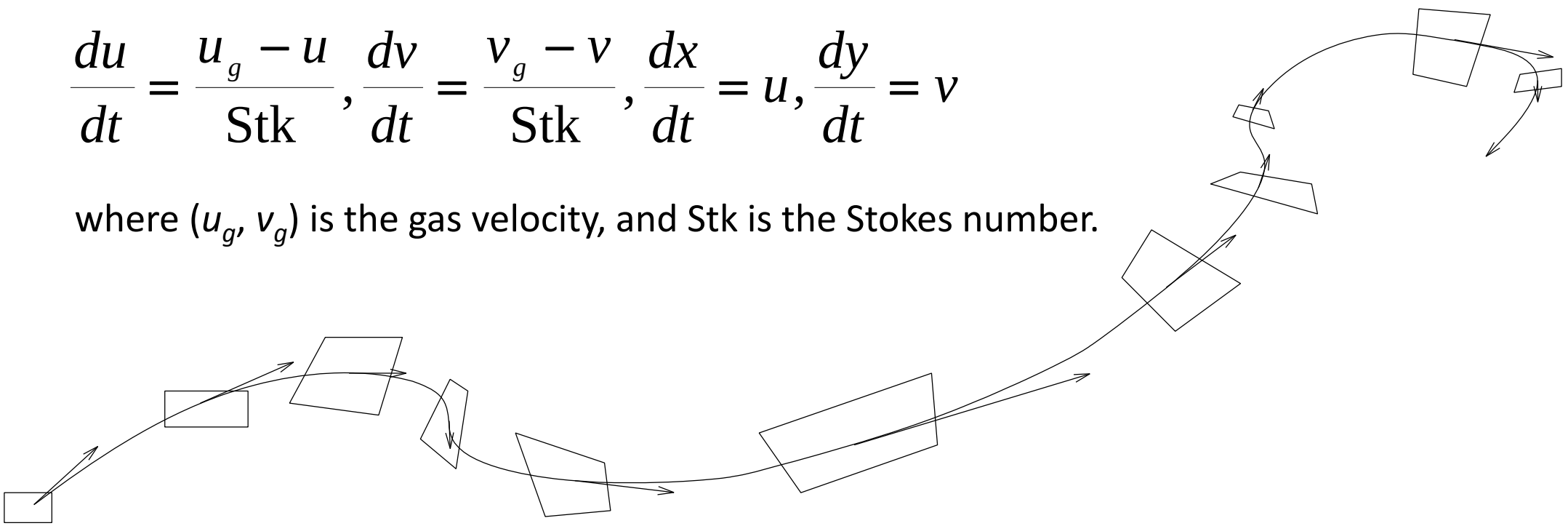
Overview

- Fully Lagrangian approach (FLA)
- FLA in ANSYS Fluent and why OpenFOAM?
- Overview of FLA implementation into OpenFOAM Lagrangian subsystem
- Test case: number density of particles in a flow around cylinder

Fully Lagrangian approach (FLA)

$$\frac{du}{dt} = \frac{u_g - u}{\mathrm{Stk}}, \frac{dv}{dt} = \frac{v_g - v}{\mathrm{Stk}}, \frac{dx}{dt} = u, \frac{dy}{dt} = v$$

where (u_g, v_g) is the gas velocity, and Stk is the Stokes number.



$$n(x_0, y_0, t) = \det\|J\| n(x_0, y_0, 0)$$

Fully Lagrangian approach (FLA)

$$J_{11} = \frac{\partial x}{\partial x_0}, J_{12} = \frac{\partial x}{\partial y_0}, J_{21} = \frac{\partial y}{\partial x_0}, J_{22} = \frac{\partial y}{\partial y_0}$$

$$\frac{dJ_{11}}{dt} = w_{11}, \frac{dJ_{12}}{dt} = w_{12}, \frac{dJ_{21}}{dt} = w_{21}, \frac{dJ_{22}}{dt} = w_{22}$$

$$\frac{dw_{11}}{dt} = \frac{1}{\mathrm{Stk}}\left(J_{11}\frac{\partial u_g}{\partial x} + J_{21}\frac{\partial u_g}{\partial y} - w_{11} \right), \quad \frac{dw_{12}}{dt} = \frac{1}{\mathrm{Stk}}\left(J_{12}\frac{\partial u_g}{\partial x} + J_{22}\frac{\partial u_g}{\partial y} - w_{12} \right), \qquad (1)$$

$$\frac{dw_{21}}{dt} = \frac{1}{\mathrm{Stk}}\left(J_{11}\frac{\partial v_g}{\partial x} + J_{21}\frac{\partial v_g}{\partial y} - w_{21} \right), \quad \frac{dw_{22}}{dt} = \frac{1}{\mathrm{Stk}}\left(J_{12}\frac{\partial v_g}{\partial x} + J_{22}\frac{\partial v_g}{\partial y} - w_{22} \right).$$

FLA in ANSYS Fluent CFD

International Communications in Heat and Mass Transfer 97 (2018) 85–91

Contents lists available at ScienceDirect

International Communications in Heat and Mass Transfer

journal homepage: www.elsevier.com/locate/ichmt

A model for heating and evaporation of a droplet cloud and its implementation into ANSYS Fluent

Timur S. Zaripov[a,b,*], Oyuna Rybdylova[b], Sergei S. Sazhin[b]

Atomization and Sprays, 27(6): 493–510 (2017)

FLA implemented as dll extension using User Defined Functions

THE FULLY LAGRANGIAN APPROACH TO THE ANALYSIS OF PARTICLE/DROPLET DYNAMICS: IMPLEMENTATION INTO ANSYS FLUENT AND APPLICATION TO GASOLINE SPRAYS

Timur S. Zaripov,[1,2,] Artur K. Gilfanov,[2] Steven M. Begg,[1] Oyuna Rybdylova,[1] Sergei S. Sazhin,[1] & Morgan R. Heikal[1]*

Why not ANSYS Fluent?

- Low control and flexibility (closed source)
- Two-way coupling is tricky
- Poor reproducibility (manual case setup in GUI, can’t easily compare parameters of cases)

Why OpenFOAM?

- Full control (Open Source)
- Great reproducibility (case setting are defined in plaintext files)

FLA in OpenFOAM: implementation strategy for steady-state two-way coupled solver

- **Case** level:
  - Let user choose initial conditions for $\boldsymbol{J}$ and $\boldsymbol{w}$ (Omega)
  - Handle IO (saving results, restarting calculations)
  - Integrate with existing postprocessing tools (ParaFoam)
- **Solver** level (high level combination of existing building blocks):
  - Iterate between calculations for carrier and dispersed phases
  - Calculate **grad(U)** and pass it to Lagrangian particle tracker
- **Lagrangian library** level (building block):
  - Add $\boldsymbol{J}$ and $\boldsymbol{w}$ (Omega) as particle properties
  - Integrate $\boldsymbol{J}$ and $\boldsymbol{w}$ (Omega) along the particle trajectory
  - Update appropriate source terms in the cell containing the particle

Solver level: modify existing SIMPLE solver

```
basicKinematicCloud\
    parcels("kinematicCloud", rhoInf, U, mu, GradU, g);
while (simple.loop(runTime))
{
    parcels.evolve();

    // --- Pressure-velocity SIMPLE corrector
    {
        #include "UEqn.H"
        #include "pEqn.H"
    }
    // @tz need this for FLA, is used by parcels
    GradU = fvc::grad(U);

    laminarTransport.correct();
    turbulence->correct();
    runTime.write();
}
```

Lagrangian library level

- **Cloud** is a collection of parcels with different physical properties.
- **Parcel** is a collection of particles with the same physical properties.
- **Particle** is responsible for tracking position in the mesh.

Actual physics is in the parcel.

Relevant files are located at:
OpenFOAM/src/lagrangian/intermediate/parcels/Templates/KinematicParcel/

KinematicParcel.H: J, Omega

```
class KinematicParcel
{
    // @tz components of the
    Jacobian and auxilary variables
    tensor J_;
    tensor Omega_;

    inline const tensor& J() const;
    inline const tensor& Omega()
    const;
    inline tensor& J();
    inline tensor& Omega();
}
```

Access to carrier phase properties in the cell:

```
class trackingData
{
    // @tz We need the continuous phase
    velocity gradient in the cell
    tensor GradUc_;
    inline const interpolation<tensor>&
    GradUInterp() const;
    inline const tensor& GradUc()
    const;
    inline tensor& GradUc();
}
```

KinematicParcel.C: KinematicParcel::calc

```
void Foam::KinematicParcel<ParcelType>::calc
{
    // Linearised momentum source coefficient
    scalar Spu = 0.0;
    this->U_ = calcVelocity(cloud, td, dt, Re, td.muc(), mass0, Su,
    dUTrans, Spu);

    this->Omega_.xx() = this->Omega_.xx() + Spu*(
            this->J_.xx()*td.GradUc().xx()
            + this->J_.yx()*td.GradUc().xy()
             - this->Omega_.xx());
    // repeat for all components of Omega

    this->J_ = this->J_ + this->Omega_*dt;
    // Accumulate carrier phase source terms
    // {…}
}
```

KinematicParcel.C: KinematicParcel::calc

```
// Accumulate carrier phase source terms
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
if (cloud.solution().coupled())
{
    scalar n_d = 1./ fabs(det(this->J_));

    // Update momentum transfer
    cloud.UTrans()[this->cell()] += n_d*np0*dUTrans;

    // Update momentum transfer coefficient
    cloud.UCoeff()[this->cell()] += n_d*np0*Spu;
}
```

KinematicParcel: Constructors + IO

- **KinematicParcelI.H**
  Modify all constructors to account for J, Omega
  Implement access functions for J, Omega

- **KinematicParcelIO.H**
  Modify all input-output functions to account for J, Omega

- **KinematicParcelTrackingDataI.H**
  Modify constructor to account for GradUInterp and GradUc
  Implement access functions for GradUInterp and GradUc

FLA injection model: J0, Omega0

Located at:
lagrangian/intermediate/submodels/Kinematic/InjectionModel/PatchInjection/

PatchInjection.H

```
class PatchInjection
{
    //- @tz
    const tensor J0_;
    const tensor Omega0_;
}
```

PatchInjection.C

```
Foam::PatchInjection<CloudType>::PatchInjection
:
    J0_(this->coeffDict().lookup("J0")),
    Omega0_(this->coeffDict().lookup("Omega0"))

void
Foam::PatchInjection<CloudType>::setProperties
{
    parcel.J() = J0_;
    parcel.Omega() = Omega0_;
}
```

What does this give us?

- Two-way coupled steady state solver for kinematic particles in fluid flow
- OpenFOAM takes care of under-relaxation of fluid flow equations (don't have to worry about source terms being too big)
- Library code can be reused as it is to build transient solvers
- Integration with OpenFOAM case structure (no hardcoded parameters that require recompilation to change)
- Integration with ParaFoam (postprocessing)

Ongoing work

- Validation and verification
- Equations for Omega can be rewritten in more general tensor form
- More general approach to setting J/Omega integration schemes (currently there is only hardcoded Euler scheme)

Flow around cylinder

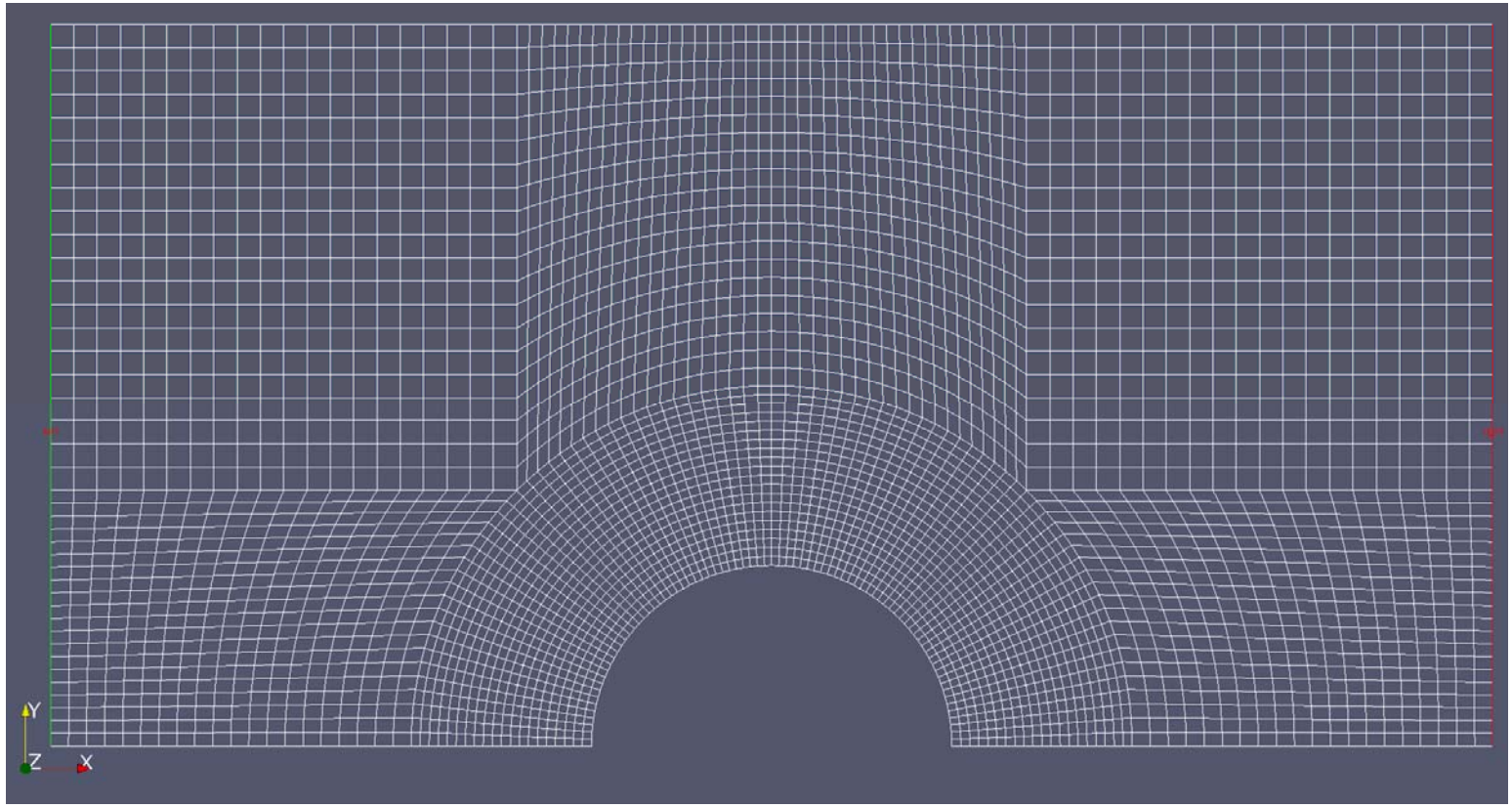
Cylinder radius:
R = 0.05 m

Domain height 0.2 m
Domain width 0.4 m

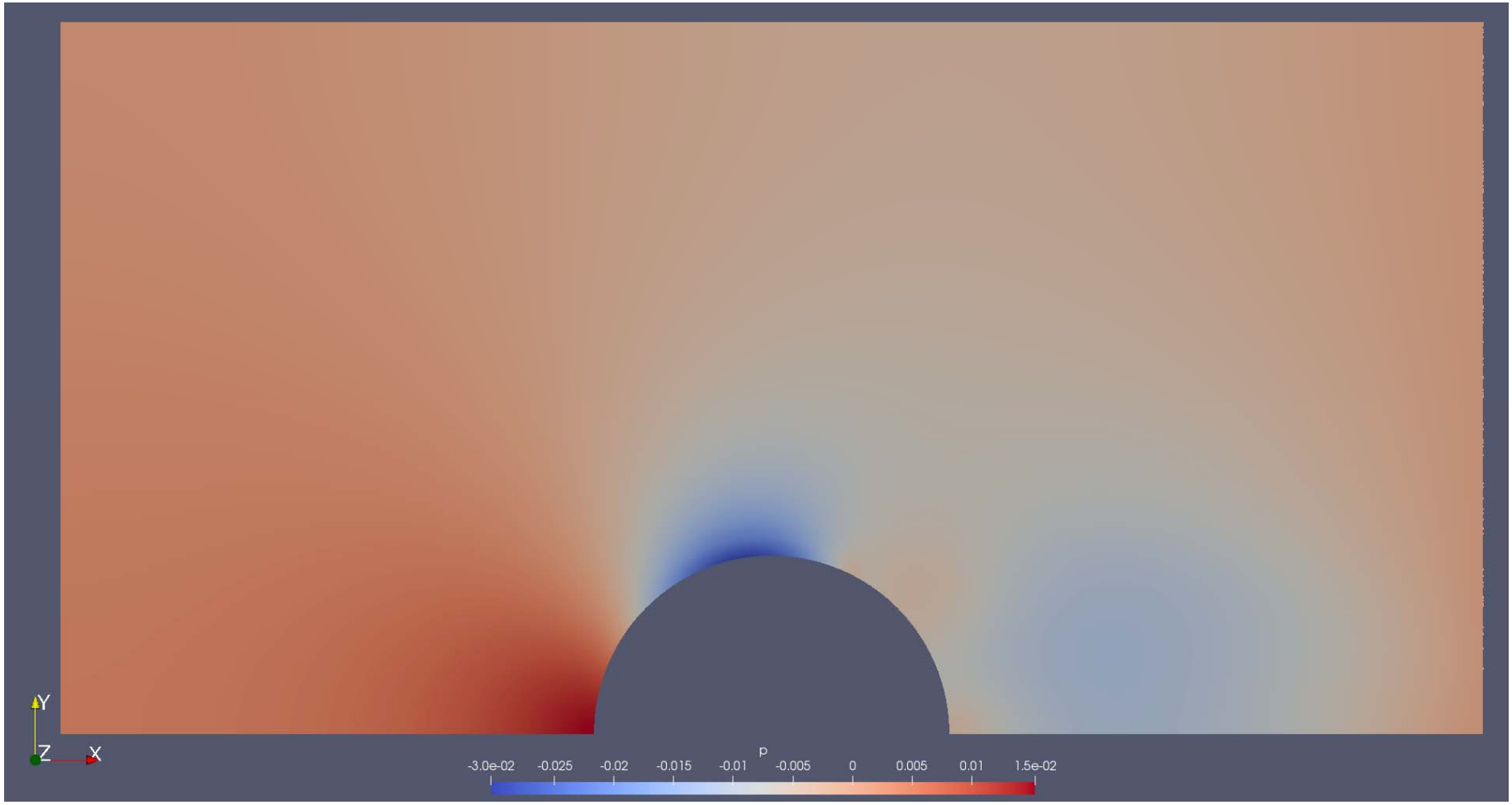
left (green):
U0 = 0.15 m/s

right (red):
P = 0

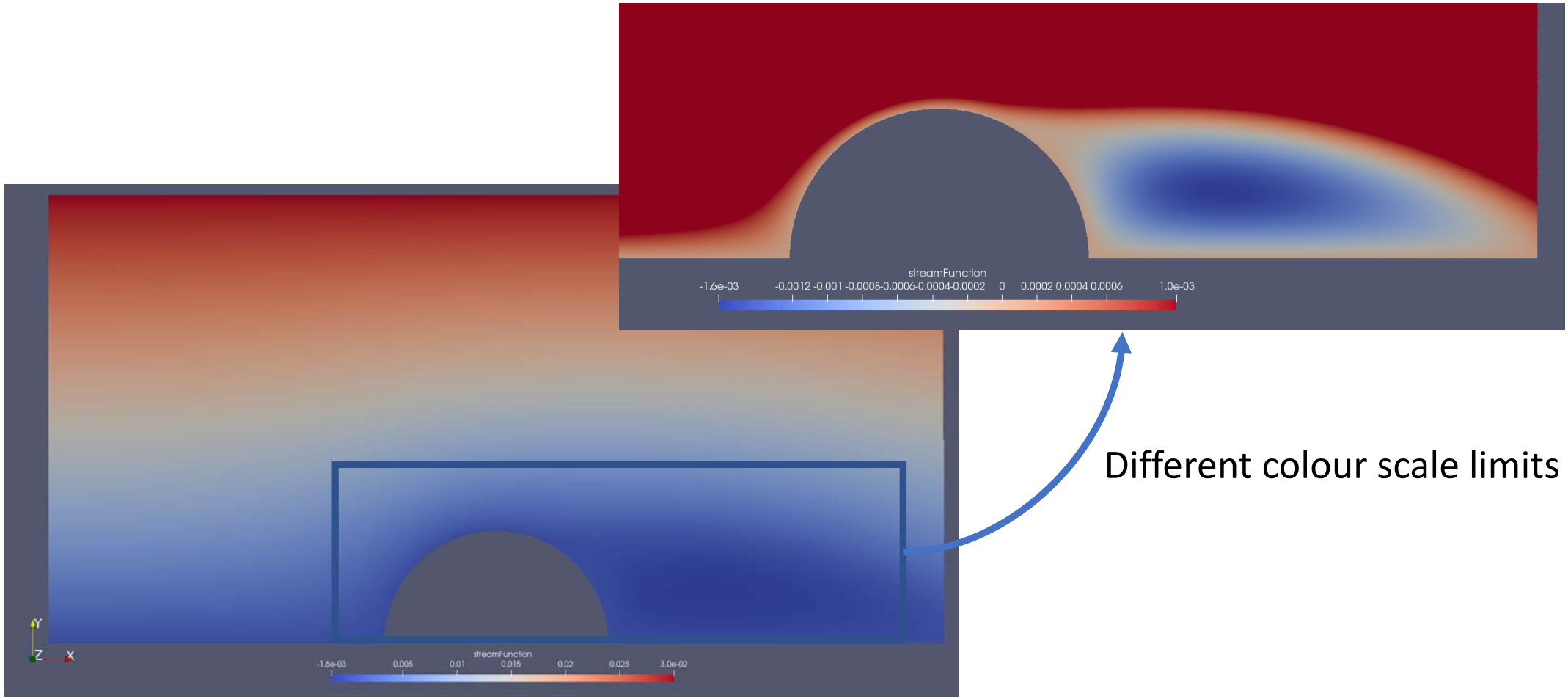
Symmetry at all other boundaries.



Pressure contour



Stream function contour



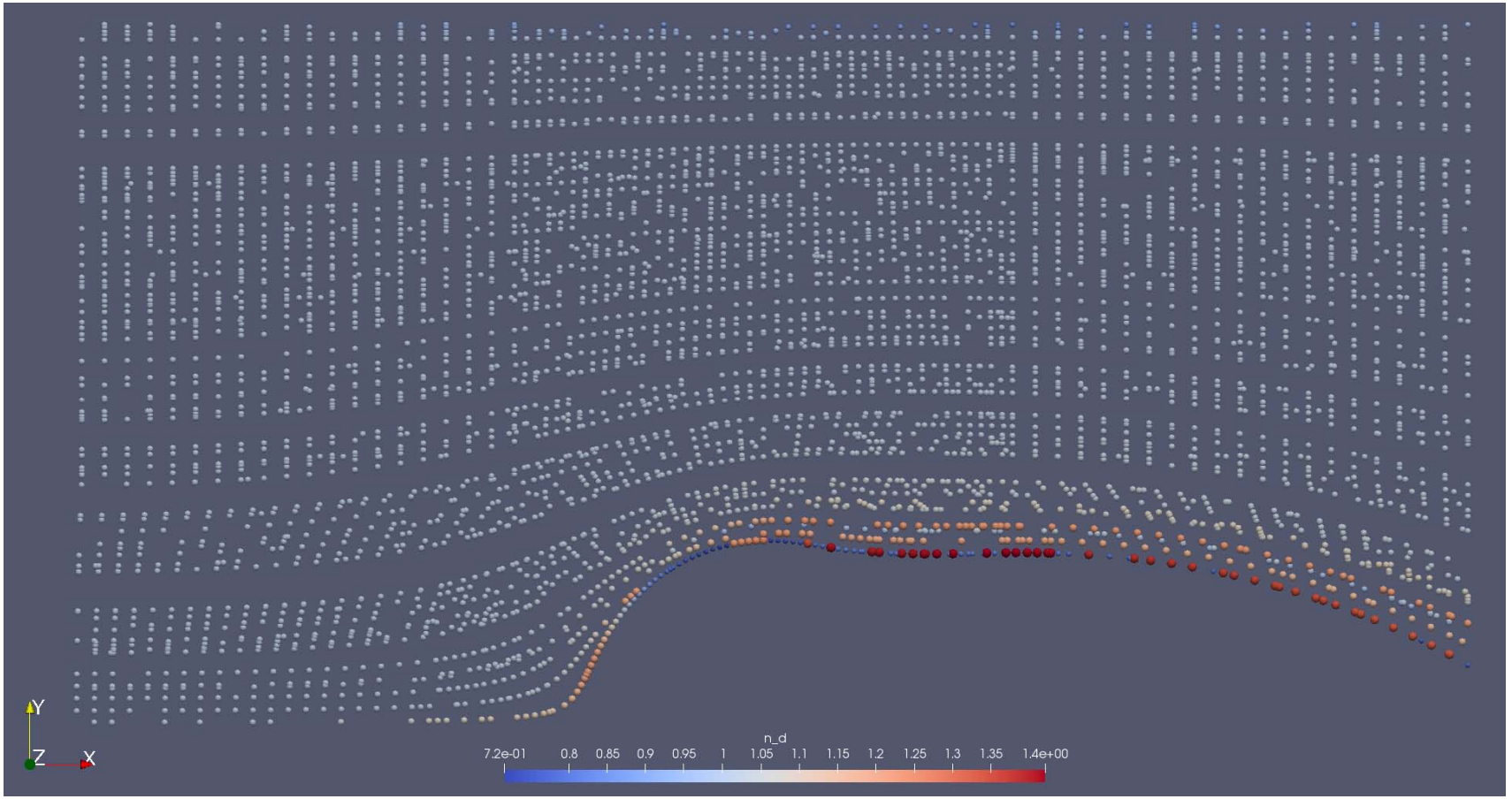
Different colour scale limits

Particle trajectories coloured by number density

Starting at the left boundary with
V0 = 0.15 m/s

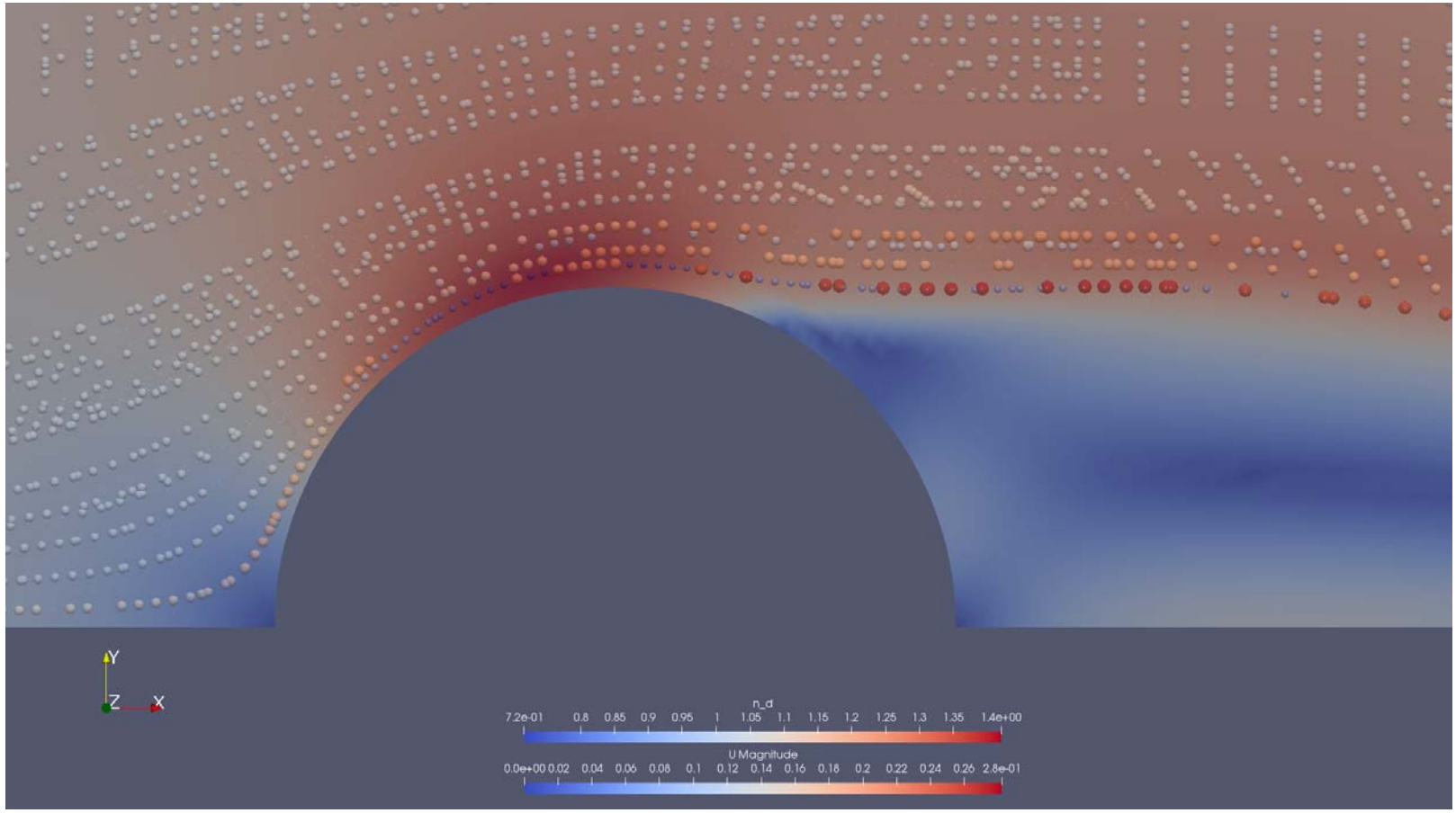
Material: water

Diameter: 5e-5 m



Particle trajectories on top of flow velocity magnitude contour

Number density represented by size and colour of the spheres



Acknowledgments

The author is grateful to the EPSRC, UK (Grant no. EP/R012024/1) for their financial support.

Thank you!