The implementation of the Full Lagrangian Approach into ANSYS Fluent

T. Zaripov[1]

[1]School of Computing, Engineering and Mathematics
University of Brighton

Modelling of droplets and flames, 2015

Topics

Full Lagrangian Method

Implementation into ANSYS Fluent UDF

Flow around cylinder

Description of experiment

Full Lagrangian Method

Modified Lagrangian method for calculating the particle concentration in dusty-gas flows with intersecting particle trajectories
by A.N. Osiptsov (1998).

Full Lagrangian methods for calculating particle concentration fields in dilute gas-particle flows
by D.P. Healy and J.B. Young (2005).

Full Lagrangian Method: idea

Equations of particle motion for Stokes drag law:

$$\frac{\partial x_i}{\partial t} = v_i, \qquad \frac{\partial v_i}{\partial t} = \beta(u_i - v_i)$$

$i = 1, 2$ in 2D case
$i = 1, 2, 3$ in 3D case
u_i and v_i – flow and particle velocity components

Full Lagrangian Method: idea

Equations of particle motion for Stokes drag law:

$$\frac{\partial x_i}{\partial t} = v_i,$$
$$\frac{\partial v_i}{\partial t} = \beta(u_i - v_i)$$

$i = 1, 2$ in 2D case
$i = 1, 2, 3$ in 3D case
u_i and v_i – flow and particle velocity components

Mass conservation along trajectory:

$$n = \frac{n_0}{|J|},$$

n – particle concentration
n_0 – concentration at start of trajectory
J – Jacobian of Eulerian–Lagrangian transformation

How do we find J components?

$$J_{ij} = \left(\frac{\partial x_i}{\partial x_{j0}}\right)_{p0,t}$$

t – time

$p0 = \{x_{i0}\}$ – defines trajectory by it's starting point

How do we find J components?

$$J_{ij} = \left(\frac{\partial x_i}{\partial x_{j0}} \right)_{p0,t}$$

t – time
$p0 = \{x_{i0}\}$ – defines trajectory by it's starting point

Lets introduce new variables:

$$\omega_{ij} = \frac{\partial J_{ij}}{\partial t} = \frac{\partial v_i}{\partial x_{j0}}$$

Differentiating with respect to t and considering equations of motion:

$$\frac{\partial \omega_{ij}}{\partial t} = \beta \left(\frac{\partial u_i}{\partial x_{k0}} - \omega_{ij} \right)$$

Finally we get

Equations for components of the Jacobian:

$$\frac{\partial J_{ij}}{\partial t} = \omega_{ij}$$
$$\frac{\partial \omega_{ij}}{\partial t} = \beta\left(\sum_k \left(J_{kj}\frac{\partial u_i}{\partial x_k}\right) - \omega_{ij}\right)$$

8 equations in 2D case
18 equations in 3D case

Finally we get

Equations for components of the Jacobian:

$$\frac{\partial J_{ij}}{\partial t} = \omega_{ij}$$
$$\frac{\partial \omega_{ij}}{\partial t} = \beta \left(\sum_k \left(J_{kj} \frac{\partial u_i}{\partial x_k} \right) - \omega_{ij} \right)$$

8 equations in 2D case
18 equations in 3D case

Initial conditions:

$$x_i = x_{i0},\; v_i = v_{i0},$$
$$c = c_0$$
$$J_{ii} = 1,\; J_{ij}(i \neq j) = 0$$
$$\omega_{ij} = 0$$

Implementation into ANSYS Fluent: motivation

ANSYS Fluent provides tools to work with

- arbitrary 2D or 3D geometry,
- steady and transient flows,
- variety of turbulence models.

Fluent takes care of the carrier phase flow. We can focus on particles concentration.

Implementation into ANSYS Fluent: UDF

Equations of particles motion are solved by Fluent.

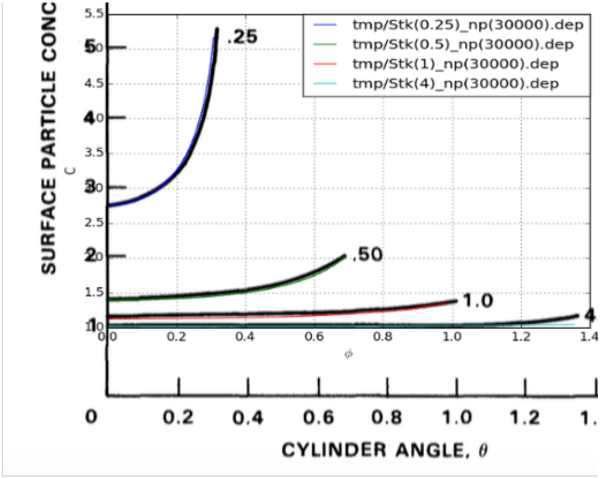
Equations for J_{ij} and ω_{ij} are solved using fourth order Runge–Kutta method at each point of trajectory at each time step.

DEFINE_DPM_UPDATE_SCALAR macros is used to execute in-house code inside the Fluent.

DEFINE_DPM_INJECTION_INIT macros is used to set initial conditions.

Verification of implementation: potential flow

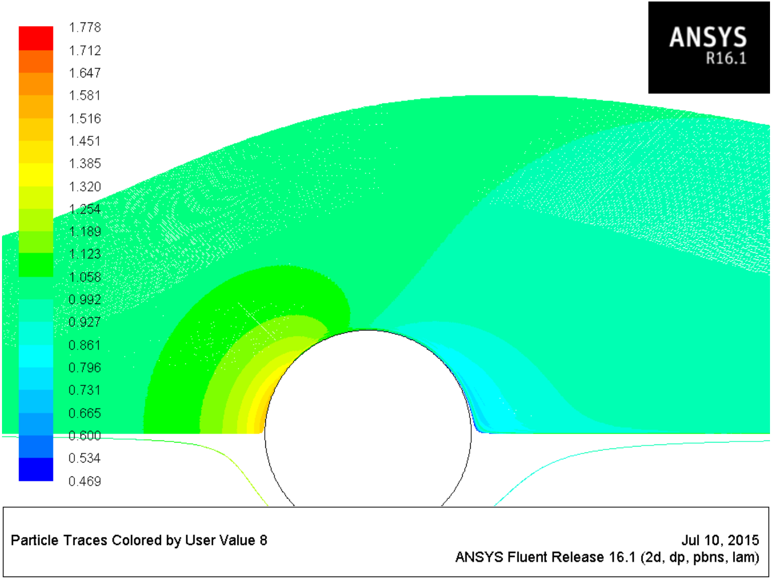
Generalized Correlations for Inertial Impaction of Particles on a Circular Cylinder
by R.A. Wessel & J. Righi (1988)



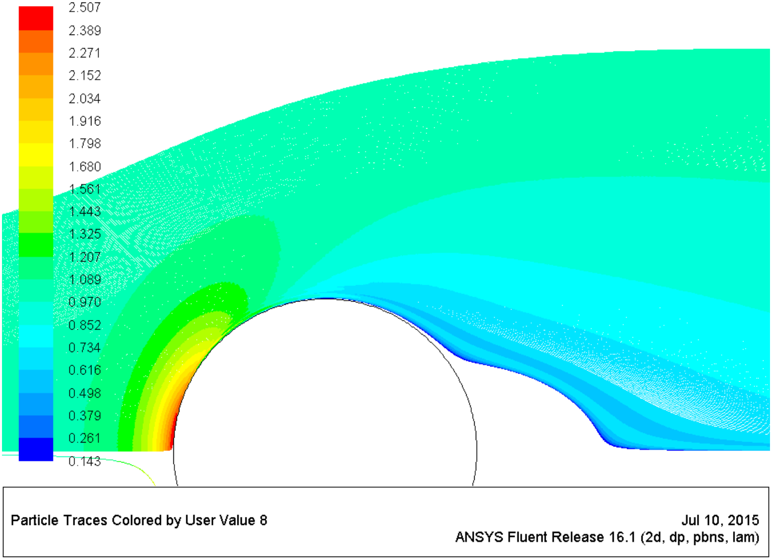
Flow around cylinder

- Viscous laminar flow
- One way interaction between phases
- Stokes drag law
- Lagrangian vs Eulerian: $\mathrm{Stk} = 0.1$
- Steady and unsteady: $\mathrm{Re} = 1,\ 10,\ 100,\ 200$

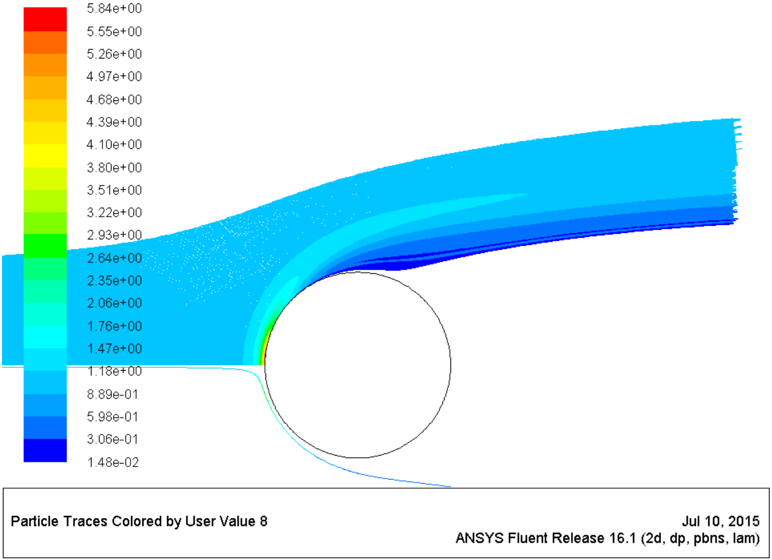
Typical calculation result: steady flow ($\mathrm{Re} = 1$, $\mathrm{Stk} = 0.1$)



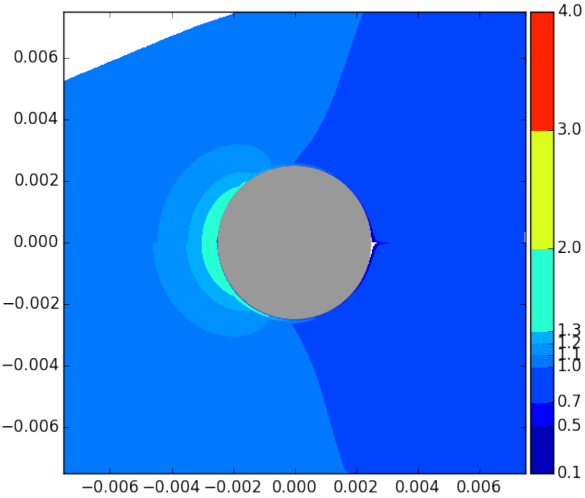
Typical calculation result: steady flow ($\mathrm{Re} = 10$, $\mathrm{Stk} = 0.1$)



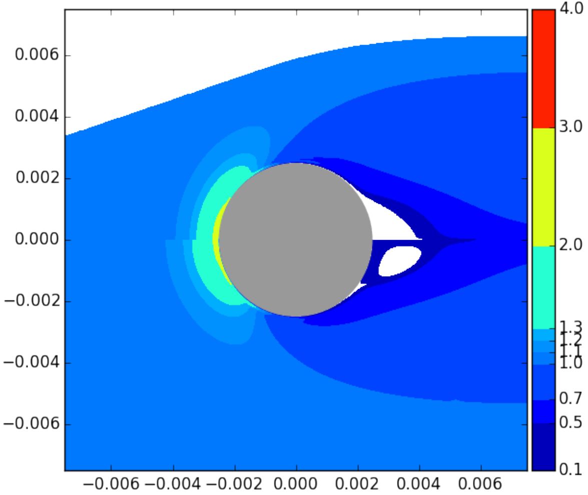
Typical calculation result: steady flow ($\mathrm{Re} = 100$, $\mathrm{Stk} = 0.1$)



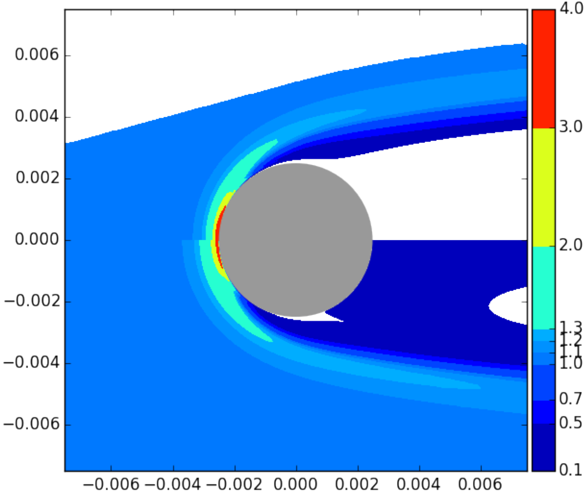
Lagrangian vs Eulerian: steady flow ($\mathrm{Re} = 1$, $\mathrm{Stk} = 0.1$)



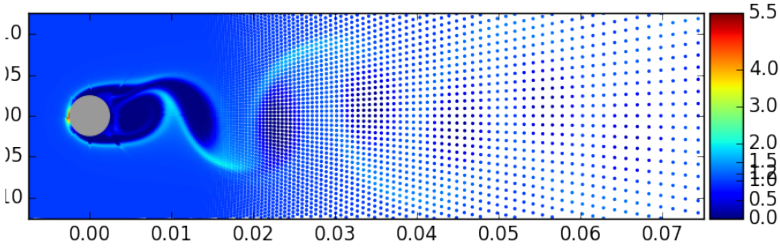
Lagrangian vs Eulerian: steady flow ($\mathrm{Re} = 10$, $\mathrm{Stk} = 0.1$)

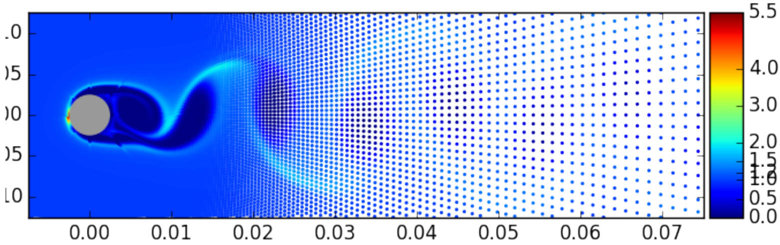


Lagrangian vs Eulerian: steady flow $(\mathrm{Re} = 100, \mathrm{Stk} = 0.1)$

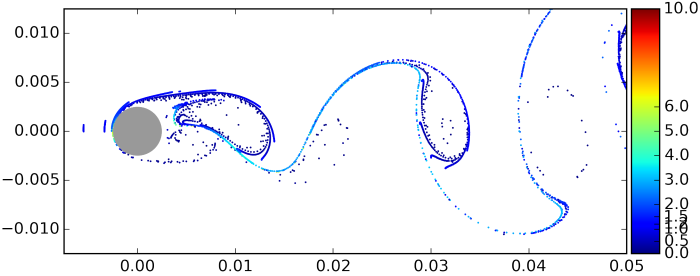


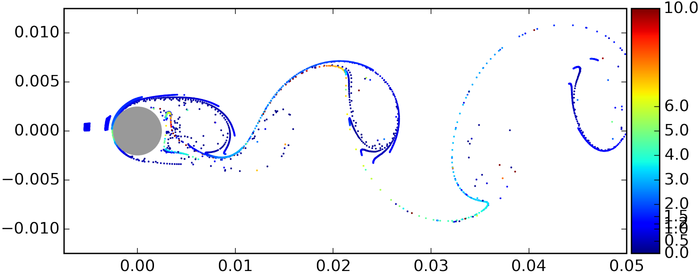
Unsteady flow: Eulerian ($\mathrm{Re} = 200$)





Unsteady flow: Lagrangian ($\mathrm{Re} = 200$)





Show the video.

Description of experiment

Dr Steven Begg
School of Computing, Engineering and Mathematics
University of Brighton
http://about.brighton.ac.uk/cem/contact/details.php?uid=smb