POLITECNICO DI MILANO

libEngine: a C++ object-oriented platform for intake and exhaust system simulation

Federico Piscaglia

Dipartimento di Energia, POLITECNICO DI MILANO

Acknowledgements

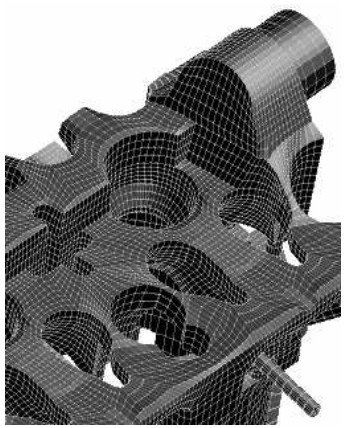
- Ph.D. student **Andrea Montorfano**, Politecnico di Milano
- Prof. **Hrvoje Jasak** - Wikki Ltd., University of Zagreb
- Prof. **Giancarlo Ferrari** Politecnico di Milano
  Prof. **Angelo Onorati** Politecnico di Milano
  Dr. **Tommaso Lucchini** Politecnico di Milano
  Dr. **Gianluca D'Errico** Politecnico di Milano
  Dr. **Gianluca Montenegro** Politecnico di Milano
  Mr. **Daniele Ettorre** Politecnico di Milano
- Prof. **Chris J. Rutland**, Prof. **David E. Foster** - ERC, University of Wisconsin-Madison (USA)
- Mr. **Luciano Spaggiari**, MV Agusta SpA (Italy)

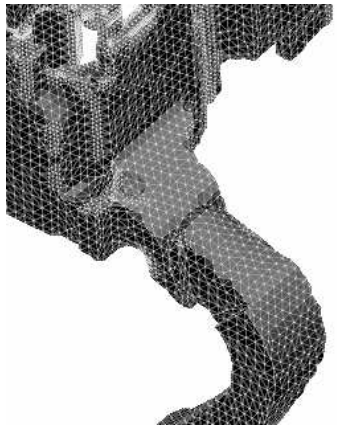
Summary of topics

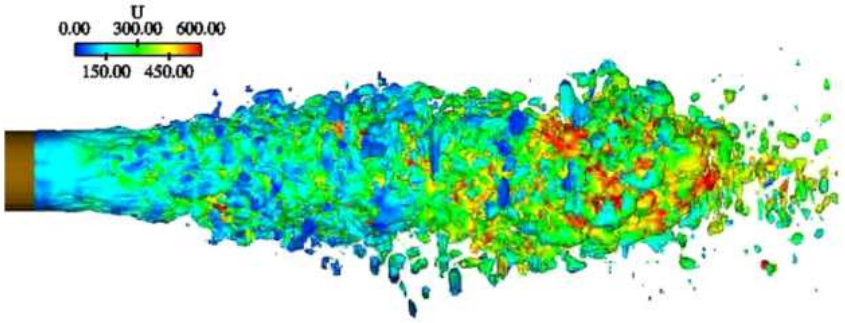
- The OpenFOAM® technology: an overview
- The **libEngine®** project: an overview
  - Diesel exhaust aftertreatment modeling
  - numerical methods for flows through porous media
  - automatic mesh generation of complex geometries
  - 1D-3D coupling
  - acoustics and silencers
- **Test cases** and **applications**

OpenFOAM®: an open-source CFD toolbox

OpenFOAM® is an **open source**, **freely available CFD Toolbox**, licensed under the GNU General Public Licence, written in highly efficient **C++ object-oriented programming**. OpenFOAM® uses the **finite volume approach** to solve systems of partial differential equations ascribed on **any 3D unstructured mesh of polyhedral cells**. **Domain decomposition parallelism is integrated at a low level** so that the implementation of parallel solvers can be performed without the need for any “parallel-specific” coding.







- **Objective**: open source implementation of existing knowledge and an object-oriented platform for easy and collaborative future development

  1. Completely open software platform using **object-oriented design**
  2. Extensive modelling capabilities in library form: **component re-use**
  3. Collaborative and project-driven model development

- This furthers the **research and collaboration** by removing proprietary software issues: source code and algorithmic details available to all

Top-Level Solver Code

Application Development in OpenFOAM®

- Custom-written top-level solvers are written for each class of physics
- Writing top-level code is very similar to manipulating the equations
- Ultimate user-coding capabilities: components can be re-used to handle most problems in computational continuum mechanics

Layered Development

- Design encourages code **re-use**: **developing shared tools**
- Classes and functional components developed and tested in isolation
  - Vectors, tensors and field algebra
  - Mesh handling, refinement, mesh motion, topological changes
  - Discretisation, boundary conditions
  - Matrices and linear solver technology
  - Physics by segment in library form
- Library level mesh, pre-, post- and data handling utilities
- Model-to-model interaction handled through common interfaces
- **New components do not disturb existing code: fewer bugs**

OpenFOAM® technology: pre-implemented capabilities

OpenFOAM® library:

- **Finite-volume discretization** with polyhedral cell support
- **Finite-element mesh motion** + topological changes
- **Lagrangian particle tracking** algorithm
- **Thermo-physical models** for liquid and gas
- **Detailed chemistry**

OpenFOAM® applications/solvers:

- **Compressible flow solvers:** RANS, LES, pressure-density based, density based, steady, unsteady
- **Combustion:** premixed or non-premixed combustion models
- **heatTransfer:** solvers for buoyancy-driven or Bousinnesq flows
- **Incompressible flows:** (steady, unsteady, viscid, inviscid, RANS, LES, . . .)

Third OpenFOAM® Workshop in Milan

The Internal Combustion Engine Group of Politecnico di Milano organized the **Third OpenFOAM Workshop**, held in Milan on July 10-11, 2008.

- 44 presentations in 7 sessions
- 250 participants from 30 countries
- 130 participants to the training course

WORKSHOP SESSIONS:

- **Automotive**, organized by Politecnico di Milano.
- **Turbomachinery**, organized by Hydro-Québec and Chalmers University of Technology.
- **Mesh handling and generation**, organized by ICON.
- **Heat transfer and fluid structure interaction**, organized by Univ. College Dublin
- **LES and hybrid LES/RANS models**, organized by the University of Exeter.
- **Multiphase and Free-Surface Flows**, organized by the Pennsylvania State Univ.

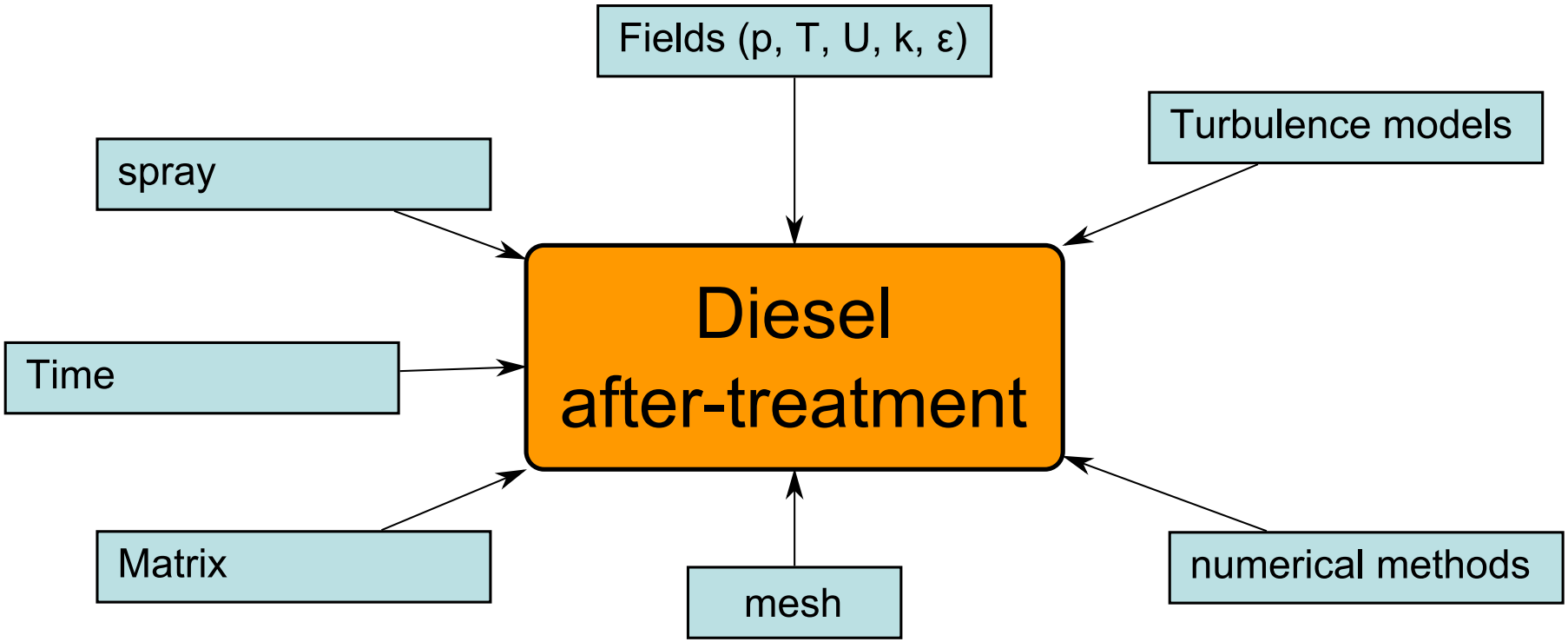
OpenFOAM®-related CFD projects

OpenFOAM®:
Official version developed and mantained by OpenCFD®

OpenFOAM®-dev:

- All the basic features of the official OpenFOAM versions;
- Advanced applications/libraries contributed by different groups:
  - Wikki Ltd. (Prof. H. Jasak);
  - Chalmers University of Technology (Prof. H. Nilsson);
  - **Politecnico di Milano (ICE PoliMi Group)**
  - University of Zagreb (Dr. Z. Tukovic)
  - Penn-State University (Prof. E. Patterson)
  - ICE (Dr. B. Gshaider)
  - . . .

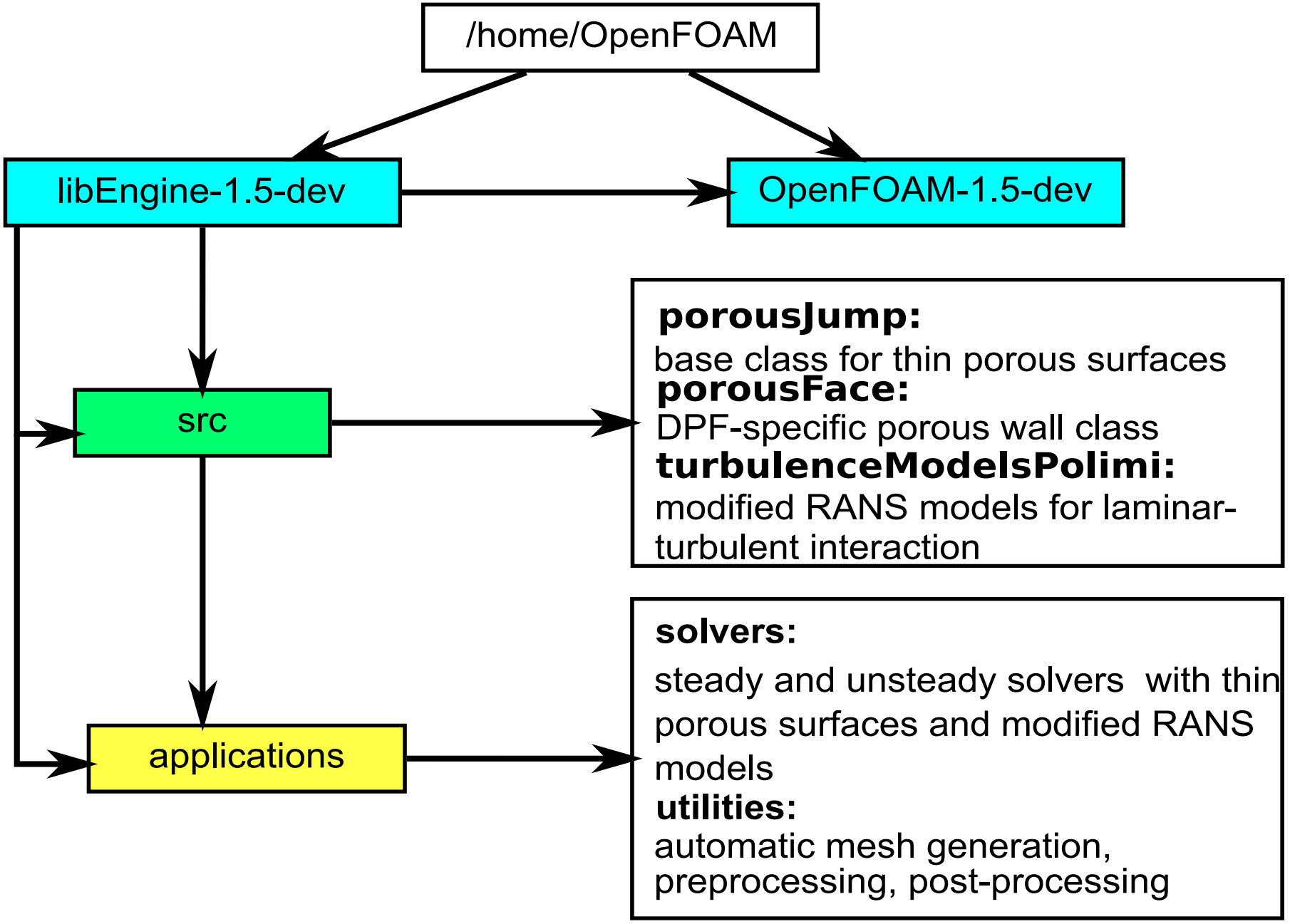
libEngine® for exhaust aftertreatment simulation



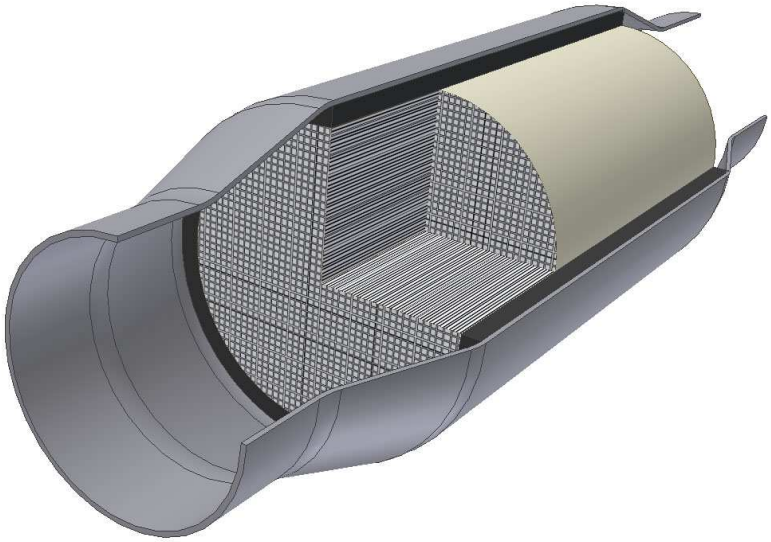
Introduce new data types (classes) appropriate for the problem:

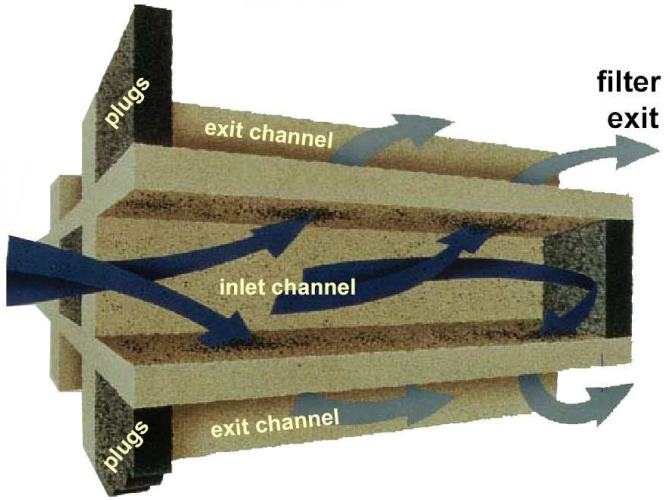
- **Class**: user defined type representing one part of the problem I have to solve (mesh, matrix, field, ...)
- **Library**: definition and implementation of related classes and functions (finite volume library, turbulence model library, mesh tools library...)
- **Applications**: collection of object of different classes interacting each others

libEngine®


/home/OpenFOAM
libEngine-1.5-dev
OpenFOAM-1.5-dev
src
applications
porousJump:
base class for thin porous surfaces
porousFace:
DPF-specific porous wall class
turbulenceModelsPolimi:
modified RANS models for laminar-turbulent interaction
solvers:
steady and unsteady solvers with thin porous surfaces and modified RANS models
utilities:
automatic mesh generation, preprocessing, post-processing

DPF modeling: motivation



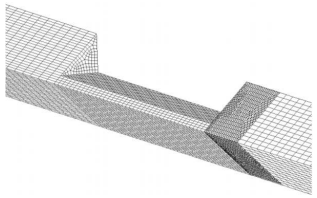


The availability of high performance CPUs at low cost, together with the improvements in the performance and stability of the message passing libraries for parallel computing, make full scale DPF simulations suitable for industrial applications.

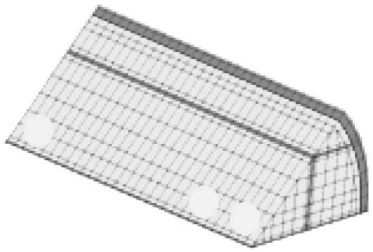
- **Need to have a predictive tool for real world Particulate Filter design** (segmented filters, non axisymmetric geometries)
- Channel-scale simulation cannot provide a detailed description of radial temperature gradients occurring during filter-regeneration, due to non-uniform conditions in the different filter channels (HEAT TRANSFER PROBLEM)

Hence, the development of a parallel **general purpose model** for the **multi-dimensional simulation of flows through porous media**, to simulate exhaust after-treatment systems of i.c. engines, is required.

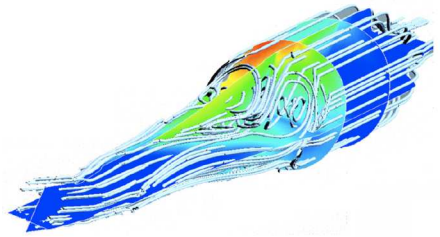
State of the Art in Full Scale DPF Modeling



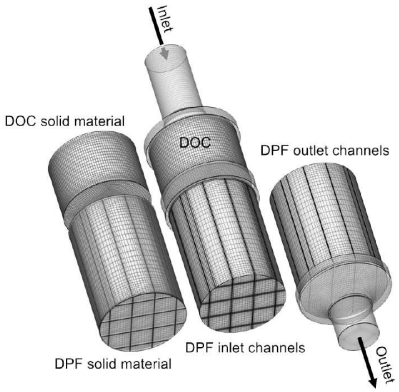
- **Modeling of the flow within the DPF as a macroscopic continuum**: simulations of a 2D section of an axisymmetric filter [6].



- **External 1D single channel model embedded within a 3D multi-channel model** plus an additional solid region to account for the heat transfer between channels [7].



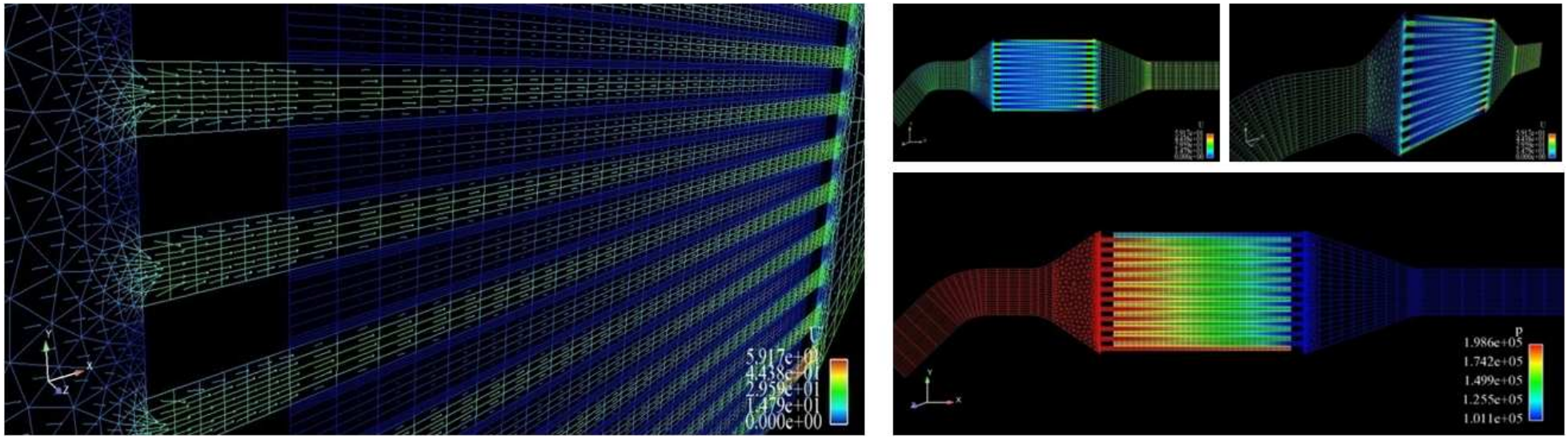
- **co-simulation methods**: the solution of the DPF region is delegated to an external solver, while the solution of the system is performed by a standard CFD solver. The inherently disconnected solution domains make this approach computationally less efficient [8].



- Diesel Particulate Filter is modelled **using anisotropic porosities with additional momentum sources**. A continuum model simultaneously computes the DPF and non-DPF flow within a single flow solver [9].

Porous Media Modeling by OpenFOAM®

Numerical solution for flows through porous media is achieved by adding a sink term to the NS equations in the differential form.

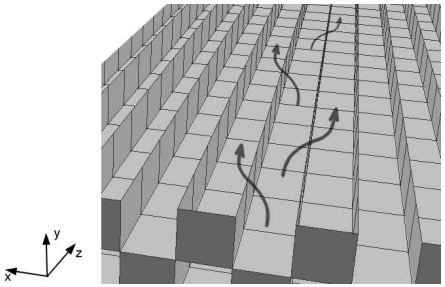


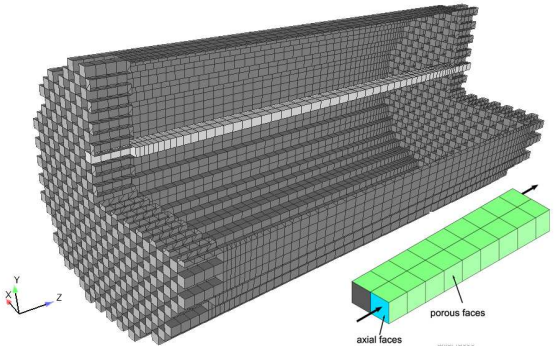
In the OpenFOAM® official distribution, the **sink term representing pressure drop through porous media** is expressed by an additional resistance (implicit or explicit) defined as a **VOLUME TERM** over the three main directions. Despite this **formulation** is very **fundamental** and it is reliable for most applications, it **does not represent the most convenient approach for DPF modeling**, because:

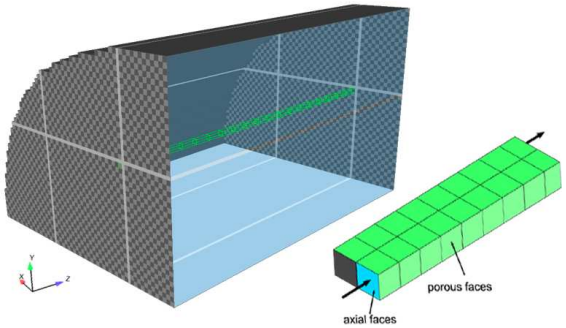
- Mesh generation and case setup for Diesel Particulate Filters look quite complicated
- **DPF grids** generated by this approach results to be **discretized in a high number of cells**, whose small size leads to very low timesteps
- At **typical DPF operating conditions**, **solvers are slow to to converge**, in particular when the filter axis is not parallel to the main reference frame

DPF Modeling by OpenFOAM®: Basic Assumptions

In Diesel Particulate Filters, porous walls are developed along the axial direction (**THIN POROUS LAYER**) and gas flows mainly along the **transverse direction (orthogonal to the porous surface)**. Hence, for ICE applications, the **volume of the porous medium may be neglected**.







Under these assumptions, **POROUS WALL CHARACTERISTICS** may be defined as a **CELL-FACE PROPERTY**. Porous walls in the filter monolith are grouped as face sets.

- **Pressure drop across the porous medium** is a **first order discontinuity** function. It is defined as a **cell-face value**, and it is approximated at the center of the cell face: no interpolation is allowed. Surface integral are approximated by the **midpoint rule**:

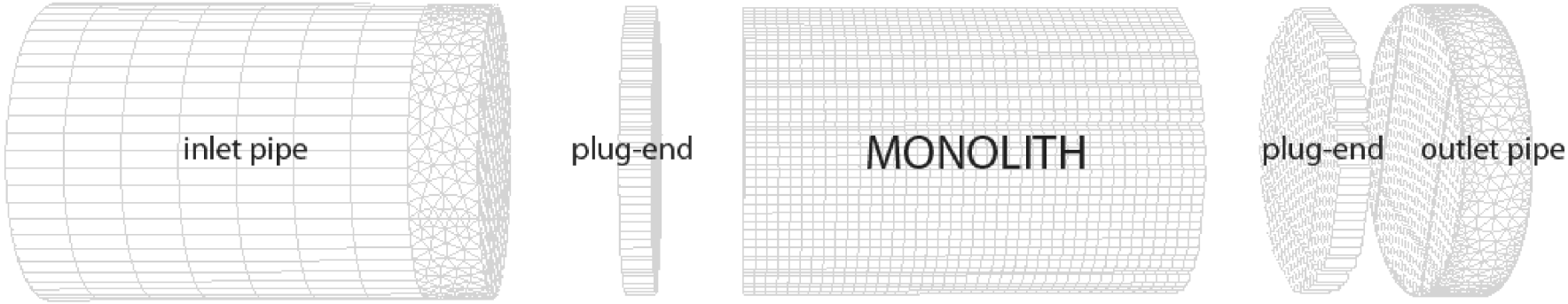
$$F_e = \int_{S_e} f dS = \bar{f}_e S_e \simeq f_e S_e$$

- **1D schematization of filter channels**: cell size over the transverse direction is taken equal to the channel size
  - **Straightforward mesh generation** of the filter, easier definition of the whole layout
  - **Faster simulations**, because of a more efficient numerical solver may be used

Developed libraries, solvers and applications in OpenFOAM®

DPF-SPECIFIC APPLICATIONS:

- New application createDpfPatch for fully automatic imposition of open-closed checkerboard pattern on filter inlet/outlet sections
- Automatic algorithm for DPF mesh generation



DEVELOPED FOR DPF USE BUT SUITABLE FOR OTHER APPLICATIONS:

- Class porousFace for simulation of **thin porous layers**:
  - Fully automatic setup of the DPF cases
  - Darcy-Forchheimer pressure drop and fluid-dynamic friction
  - Soot filtration and deposition model
- **New pseudo-staggered solvers for non-porous and porous components**, with or without soot transport:
  - rhoSimplePorousFaceFoam
  - rhoPISOTransientPorousFaceFoam

Developed libraries, solvers and applications in OpenFOAM®

GENERAL PURPOSE APPLICATIONS:

- Class zoneExt: extracts face- and cellZones for post-processing;
- Post-processing tools to visualize cellZones and faceZones and surface variables (e.g. face fluxes)

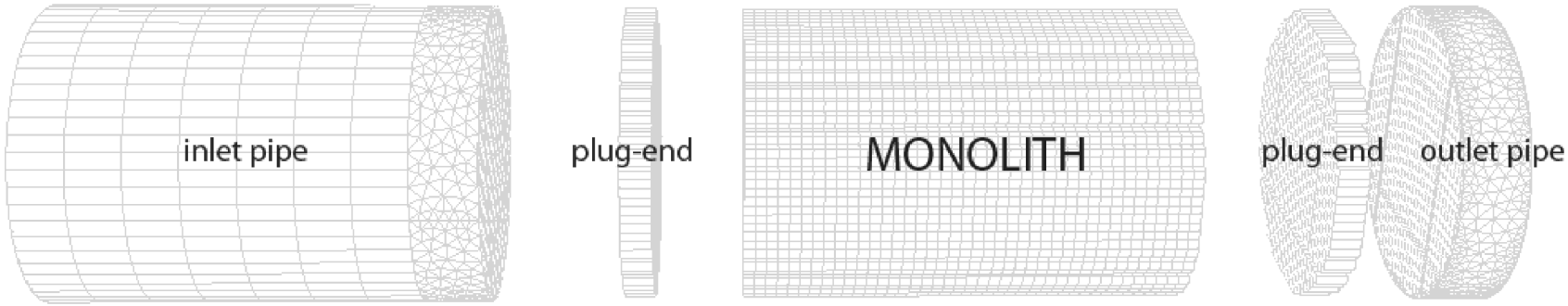
DPF-SPECIFIC APPLICATIONS:

- New application createDpfPatch for fully automatic imposition of open-closed checkerboard pattern on filter inlet/outlet sections
- Shell script for automatic DPF mesh generation.

DEVELOPED FOR DPF USE BUT SUITABLE FOR OTHER APPLICATIONS:

- Class porousFace for simulation of thin porous layers:
  - Fully automatic creation for DPF cases;
  - Darcy-Forchheimer pressure drop and fluid-dynamic friction;
  - Filtration model for soot.
- **New pseudo-staggered solvers for non-porous and porous components**, with or without soot transport:
  - rhoSimplePorousFaceFoam
  - rhoTransientSimplePorousFaceFoam
  - transient PISO solver for flows through porous media

Case Setup and Automatic Mesh Handling



The computational domain is partitioned into two regions, corresponding to different sub-domains:

FLUID REGION

- inlet pipe
- plug-ends
- filter channels
- outlet pipe

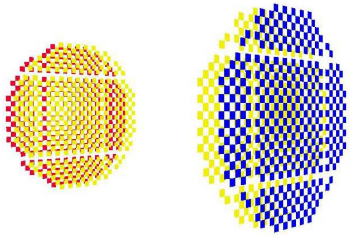
SOLID REGION

- filter segments

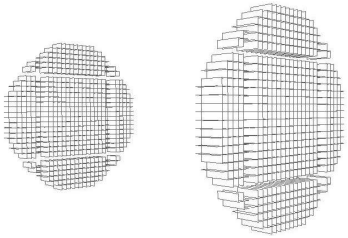
- **Any block** of the computational mesh is **generated separately** by a grid generator. During this first step, all the inner cells (and their faces) in the computational domain are defined as fluid
- **Blocks are merged** into one block
- Plug-ends, filter channels and closed-ends in the monoliths are set as cell face properties in the DPF by **AUTOMATIC ALGORITHMS** developed as applications in the OpenFOAM® technology

Case Setup

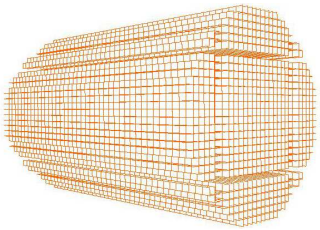
The **three-dimensional mesh geometry of the monolith** is generated from a 2D sketch; faceSets customization for DPF applications is performed by an automatic algorithm (**createDpfFaceSets**):



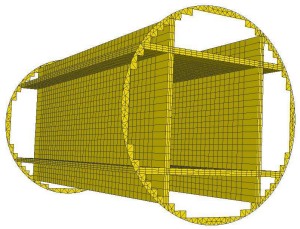
- **inlet and outlet ends** of the monolith show a typical “chessboard” arrangement, where channels are alternatively open and closed; cell-faces representing the **closed-ends** of filter channels are automatically set as ”**walls**”



- **plug ends** are automatically generated by extruding inlet and outlet ends of the monolith and then adding the resulting mesh to the original one; they have non-permeable walls, that are automatically grouped and set as ”walls”



- **porosity is modeled as a cell-face property**; porous walls dividing inlet and outlet channels of the monolith are grouped in a faceSet defined as ”porous”



- The **solid region** for the DPF material and the cement strips is used to **model the heat exchange to the surroundings**

Porous class: a class to calculate pressure jumps across surfaces

The resulting formulation of the **Navier Stokes equations for flows through porous media** is:

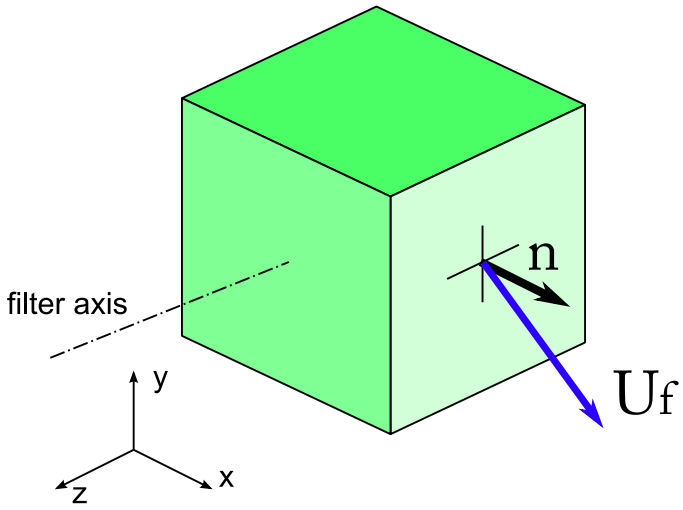
$$\int_\Omega \frac{\partial(\rho\vec{u})}{\partial t}\, d\Omega + \int_A \rho\vec{u}\vec{u}\cdot\vec{n}\, dA = -\int_\Omega \vec{\nabla}p\, d\Omega + \int_\Omega \vec{\nabla}\cdot\vec{\sigma}\, d\Omega + \int_\Omega \rho\vec{g}\, d\Omega + \vec{S}$$

where the sink term $\vec{S}$ is written as follows:

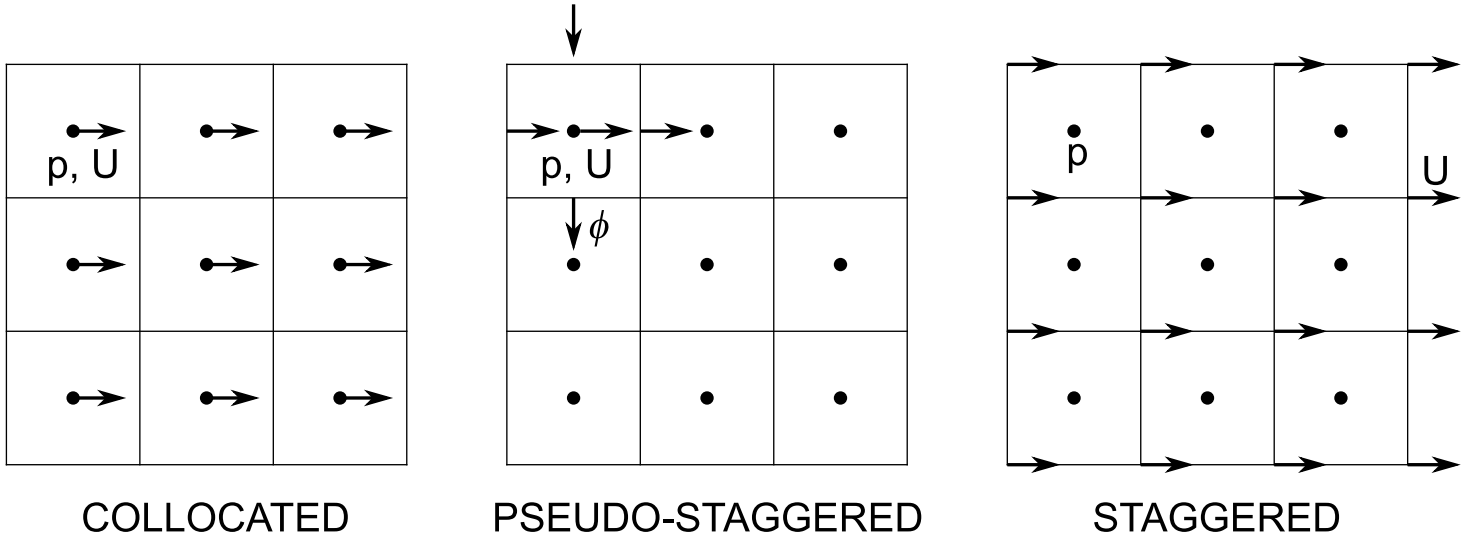
$$\vec{S} = \sum_{i=1}^{n_{por}} \left(\frac{\mu \cdot w_s}{k_p}\vec{u}_{w_i} + \frac{1}{2}\beta\rho\vec{u}_{w_i}^2 \right) A_{f_i} + \int_\Omega \left(\frac{F\cdot\mu}{a^2}\vec{u}_{ax} \right) d\Omega$$

being:

- μ = dynamic viscosity of the gas $\left[\frac{kg}{m\cdot s}\right]$
- $\vec{u}_w$ = wall-flow velocity (defined over the porous surface) $[m/s]$
- $\vec{u}_{ax}$ = axial velocity (defined at the cell center) $[m/s]$
- w_s = porous wall thickness $[m]$
- k_p = porous wall permeability along the surface-normal direction $[m^2]$
- F = friction coefficient for DPF channels $[-]$

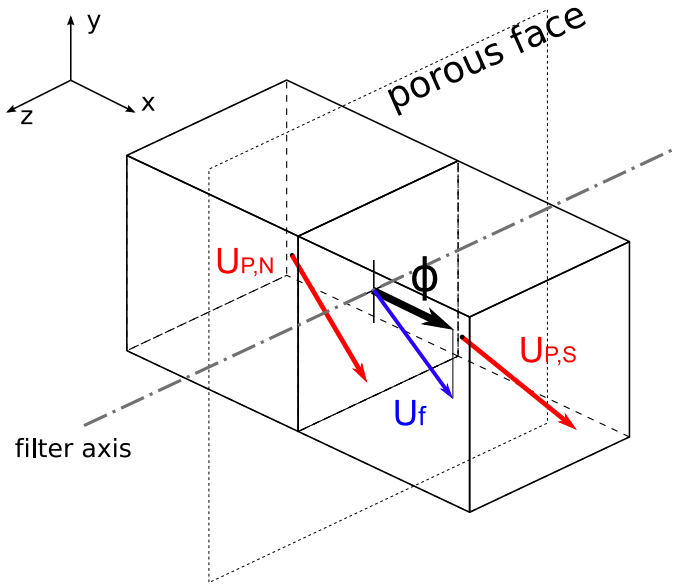


Variable Arrangement in OpenFOAM®



- **Discretization** in OpenFOAM® is based on the **collocated arrangement**, that allows significant advantages in complicated solution domains, especially when the boundaries have slope discontinuities or the boundary conditions are discontinuous.
- In the DPF mesh, velocity and pressures between neighbour cells (divided by the porous cell-face) may be very different; hence, the pressure-velocity handling of the collocated variable arrangement may cause that **mass is poorly conserved**.
- The problem does not occur if a **staggered variable arrangement** is used. A **pseudo-staggered approach** has been used to preserve sharp value changes in pressure and velocity field between DPF channels. The method **mimics the operation of a solution procedure devised for a staggered variable arrangement, but keeping the collocated variables**.

Solver Implementation



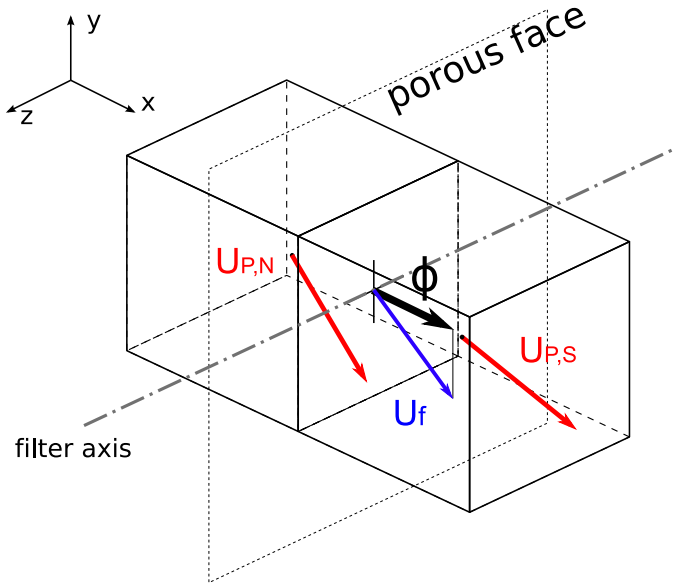
Momentum equation is solved for the face flux ϕ instead of cell-centered velocity U. Face velocities $\vec{u}_f$ are calculated from the corrected flux fields. The **cell-centred velocity** is regarded as a **secondary variable**, which is used in the construction of the momentum equation. **Pressure gradient** $\vec{\nabla} p$ is also calculated on cell faces.

$$\int_\Omega \vec{\nabla} \cdot \left(\rho \vec{u}^{n-1} \cdot \vec{u}^{n}\right) \, d\Omega = -\int_\Omega \nabla p \, d\Omega + \int_\Omega \vec{\nabla} \cdot \vec{\sigma} \, d\Omega + \vec{S}^{n-1}$$

where the sink term $\vec{S}$ is calculated EXPLICITLY and it is written as follows:

$$\vec{S} = \sum_{i=1}^{n_{por}} \left[\frac{\mu_{f_i}^{n-1} w_{th}}{k_{f_i}} \vec{u}_{f_i \perp}^{n-1} + \frac{1}{2} \rho_{f_i}^{n-1} \beta_{f_i} \vec{u}_{f_i \perp}^{n-1} \cdot \vec{u}_{f_i \perp}^{n-1} \right] \cdot A_f + \int_\Omega \frac{F \cdot {\mu_f}^{n-1}}{a^2} \vec{u}_{ax}^{n-1} \cdot \, d\Omega$$

Solver Implementation



Face orthogonal velocity $\vec{u}_f$ in the sink term is computed by interpolation of the cell-centered velocity $\vec{u}$ and it is calculated through the face flux ϕ:

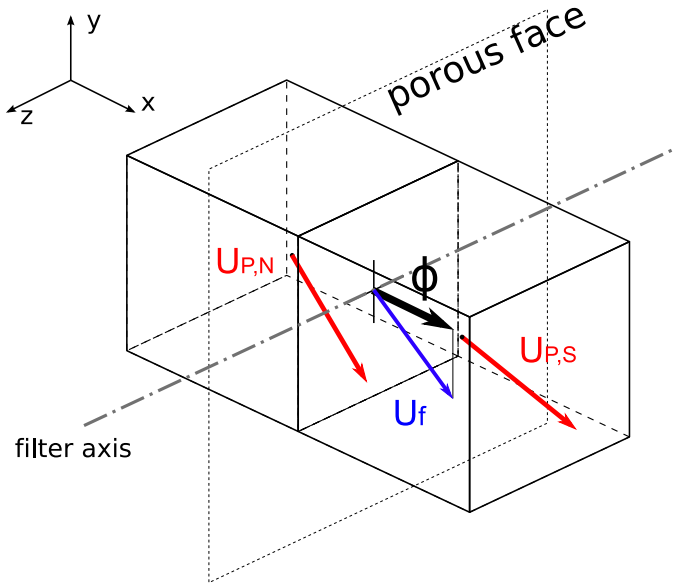
$$\vec{u}_f = \sum_i^{nbc} \alpha_{f_i} \vec{u}_i$$

$$\phi = \rho_f A_f (\vec{u}_f \cdot \vec{n}_f)$$

Hence:

$$\vec{u}_{f_\perp} = \frac{\phi}{\rho_f A_f} \vec{n}_f$$

Solver Implementation



The resulting formulation of the sink term $\vec{S}$ is:

$$\vec{S} = \sum_{i=1}^{n_{por}} \left[\frac{{\mu_f}^{n-1}\, w_{th}}{k_{f_i}} \frac{{\phi_f}^{n-1}}{{\rho_f}^{n-1} A_f} + \frac{1}{2}\beta \frac{\left(\phi_f^{n-1}\right)^2}{\rho_f^{n-1} {A_f}^2} \right] \cdot A_f + \int_\Omega \frac{F \cdot {\mu_f}^{n-1}}{4a} \vec{i}_3 \; \vec{i}_3^{\,T} \; \vec{u_{ax}} \; d\Omega$$

- Linear interpolation has been used to calculate over the cell face any other quantity (**gas density** ρ_f and **dynamic viscosity** μ_f for instance) originally defined over the cell volume.
- Code has been implemented in a **coordinate free formulation**

Solver Implementation

Governing equations are solved for steady flows by means of the **SIMPLE algorithm**.
Momentum equations are discretized as follows:

$$A_P^{u_i} u_{i,P}^n + \sum_l A_l^{u_i} u_{i,l}^n = Q_{u_i}^{n-1} - \left(\frac{\delta p}{\delta x_i}\right)_P^{n-1} - \left(\frac{\delta p_{por}}{\delta x_i}\right)_P^{EXP}$$

- **Convection term** is treated in a **semi-implicit** fashion;
- **Stress tensor** is treated in a **fully-implicit** fashion;
- **Other source terms** are treated **explicitly**.

- For each outer iteration (where the coefficients and source matrices are updated), the set of algebraic equations for each component of the momentum equations is solved in turn and the velocity components are the unknowns.
- Index P indicates the current cell (owner), while the index l indicates the neighboring cells. In the term Q (RHS of the linear system) sink terms depending on the velocity at the previous timestep $u_{i,P}^{n-1}$ are included
- **Gas density ρ is computed by the energy equation; it does not vary significantly, because gas flow is almost incompressible ($\mathrm{Ma} < 0.3$).**

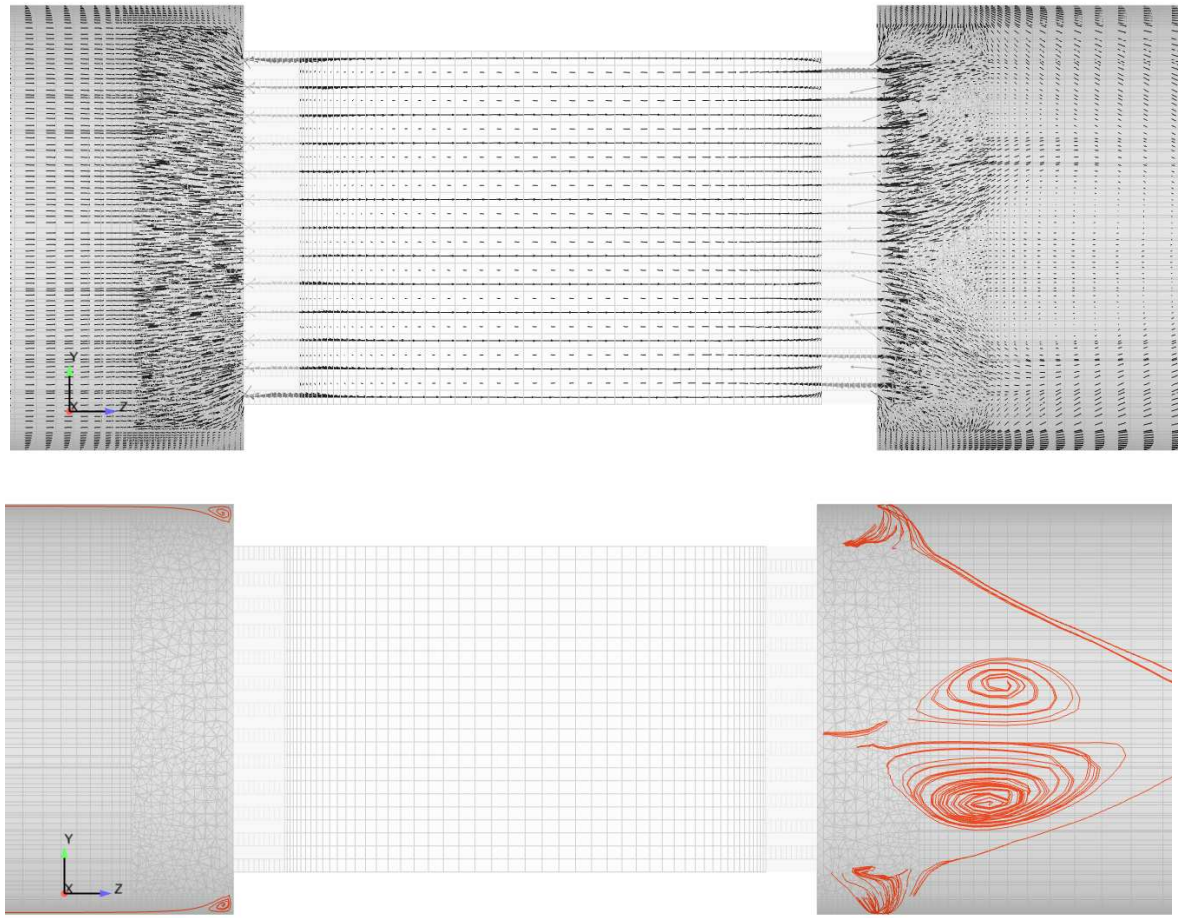
Top-level Solver Implementation in OpenFOAM®

The **POROUS CLASS** has been implemented accordingly to the o.o. structure of the OpenFOAM® CFD code, so that **partial differential equations** are expressed in their **natural language**:

```
label Eqn = solve
(
  fvm::div(phi,U)
  - fvm::Sp(fvc::div(phi), U)
  + turbulence->divRhoR(U)
  + porousWall.fvcFriction(U)
  ==
    stagPgrad
  + porousWall.fvcResistance(phi)
)
```

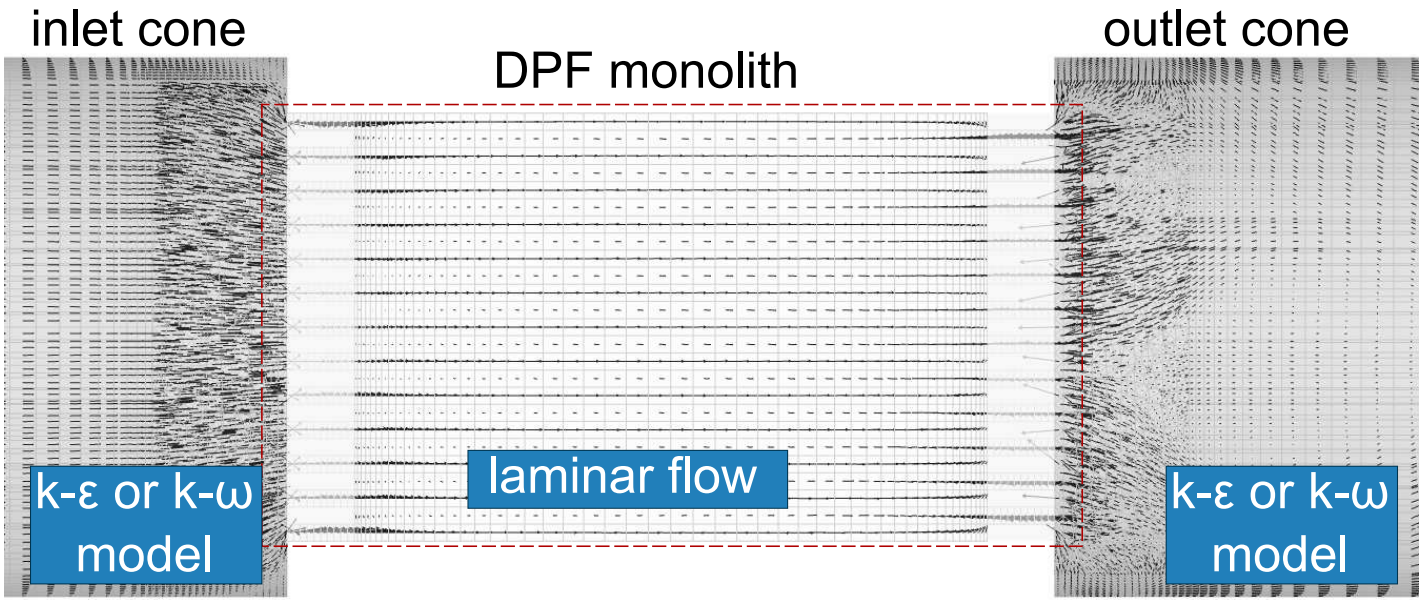
- **Correspondence** between the **implementation** and the **original equation** is clear
- Each model answers the interface of its class, but its implementation is separated and independent of other models
- Model implementation and inter-equation coupling can be examined independently from the related numerical issues
- New components do not disturb existing code

Interaction between laminar and turbulent flow regions in DPFs



- Turbulence class has been extended in order to allow for switching off turbulence source terms in the NS equations within a single cell region, for a given cellSet. In the case studied, turbulence is not calculated in the monolith channels, where the flow regime is laminar.
- simulations were carried out by a transient PISO solver, where the porous class was included

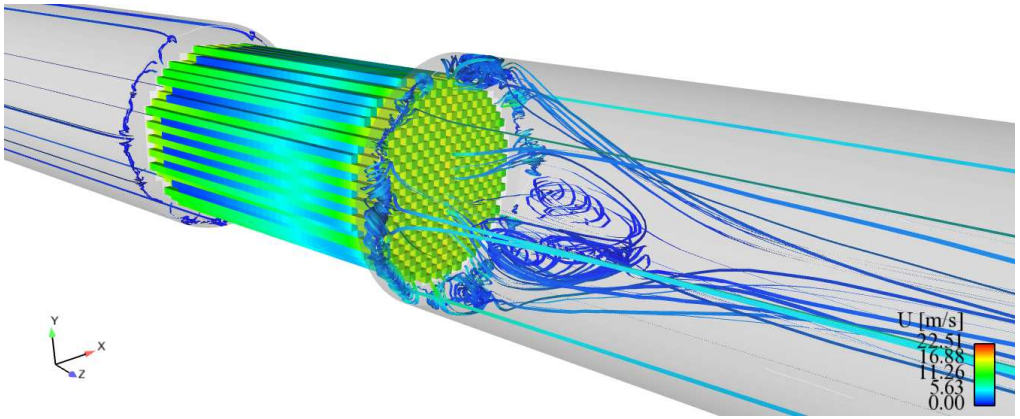
Turbulence models

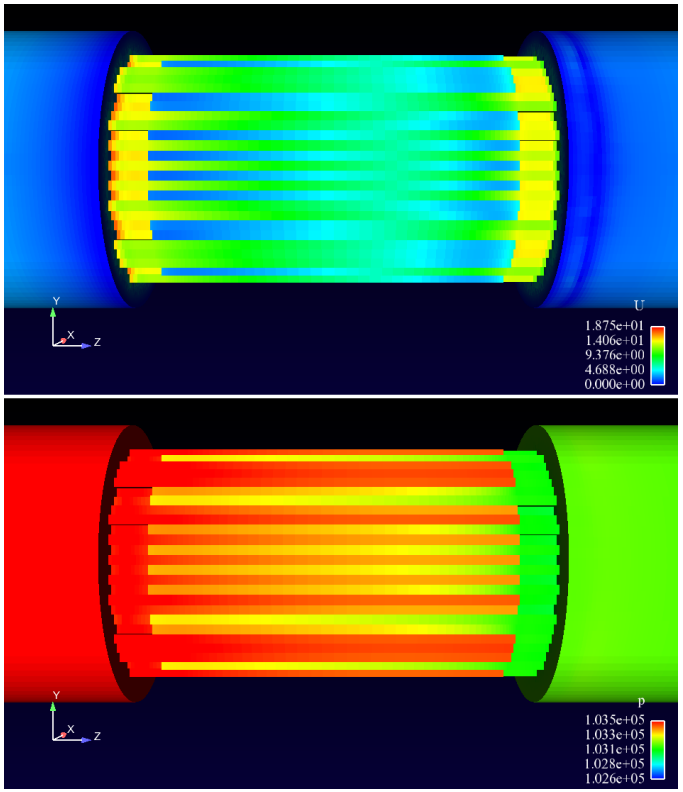


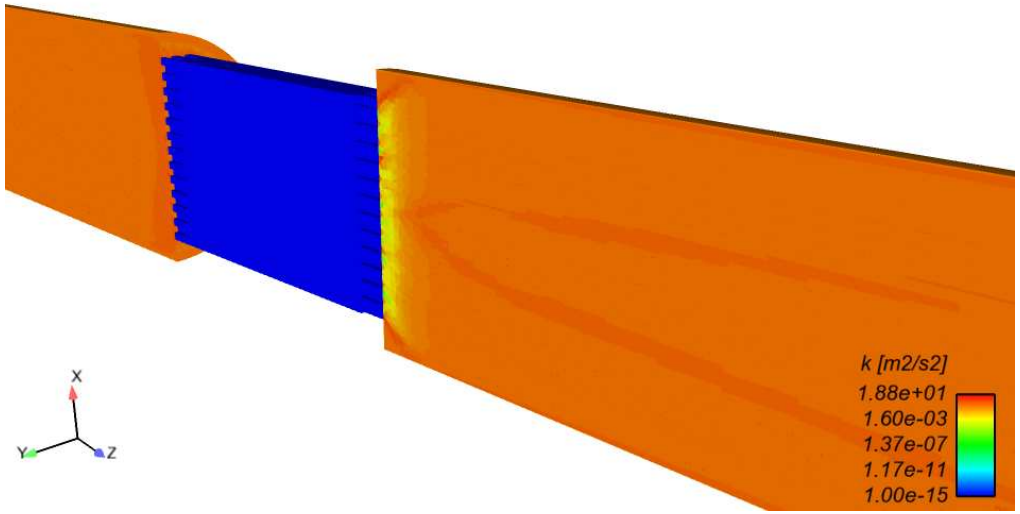
- **Flow conditions in after-treatment devices are characterized by very different Reynolds numbers:**
  - flow regime in outlet and inlet cones is fully turbulent;
  - flow into the filter monolith is laminar;
- **Standard turbulence class in OpenFOAM® requires modifications to be used with after-treatment components:**
  - turbulent viscosity is not calculated in monolith cells;
  - particular interpolation techniques are needed for turbulent fields (k, ε,ω);

Turbulence models

- **Modified RANS models to be used with DPF simulations:**
  - Modified k-ε: `kEpsilonDpf`
  - Modified k-ω SST: `kOmegaSSTDpf`
- **New turbulence class has been included into `libEngine` package:** no recompilation of OpenFOAM® standard libraries is required







Code validation

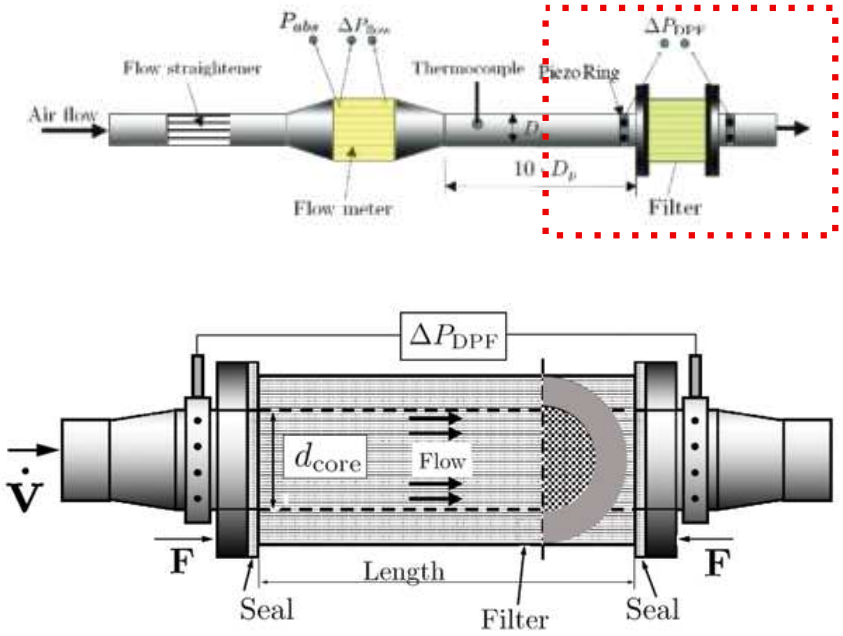


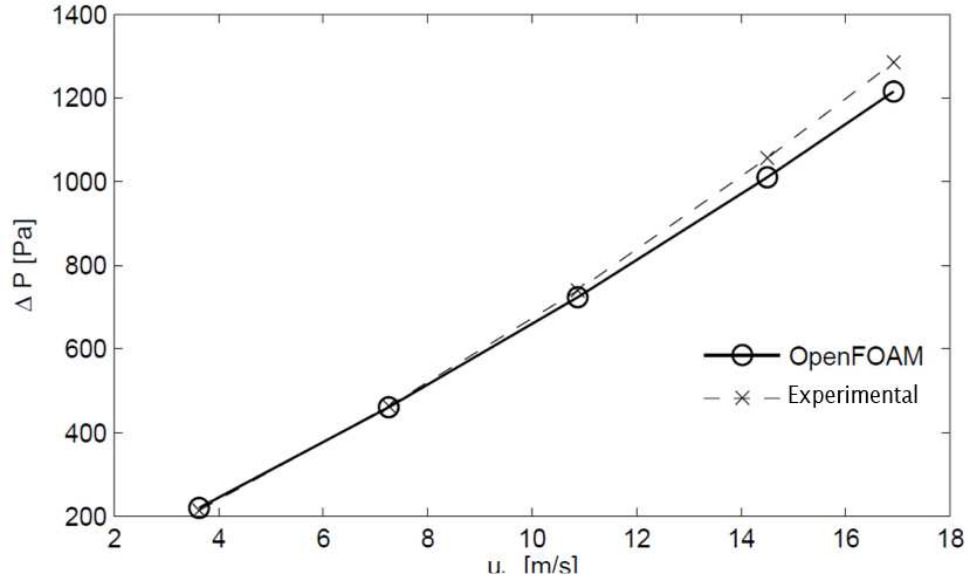
Table 1: Specification of the Filter Used in the Experiments and Simulations [10].

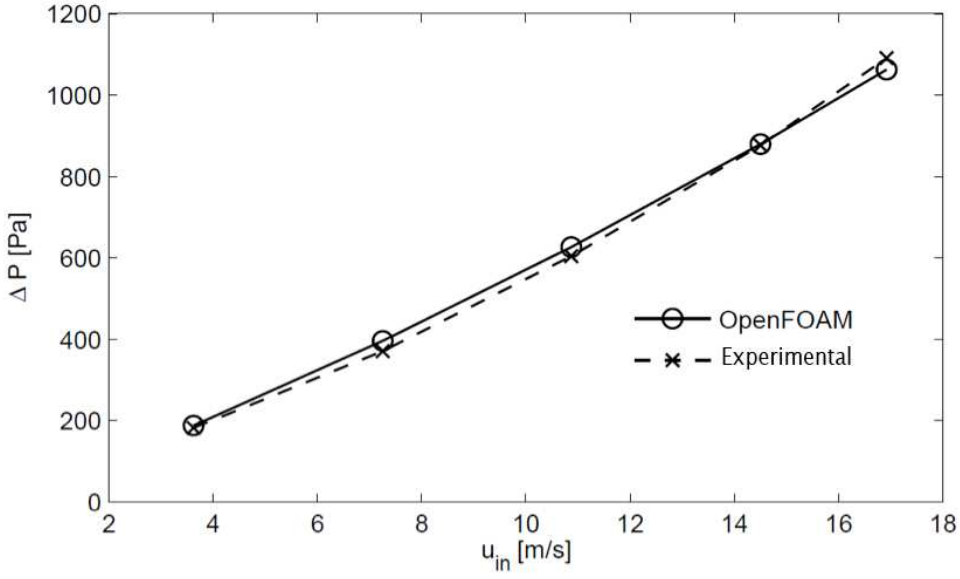
filter type	EX-80 100/17
channel length [mm]	114/165
plug length [mm]	10
channel width [mm]	2.28
porous wall thickness, w_s [mm]	0.432
cell density [CPSI]	100
porosity	48%
mean pore size [μm]	12

Numerical simulations were carried out on a grid having **199260 computational cells** (96462 hexahedra, 2186 pyramids, 100612 tetrahedra); calculation of each operating point required approximately 7.5 CPU hours by an Intel Core Duo 2.8 GHz. Simulation time was about 5 hours as simulations ran in parallel on 2 processors.

- **Flow conditions**: p_{out} = 101325 Pa, T = 300 K
- **Clean trap permeability** of the porous medium k_p= $8.2 \cdot 10^{-13}$ m^2, in order to match the experimental point at the lowest flow rate ($\dot{V}$= 15 Nm^3 /h)
- k_p **was kept constant** as the inlet flow rate was varied through the simulations

Code validation (clean trap)

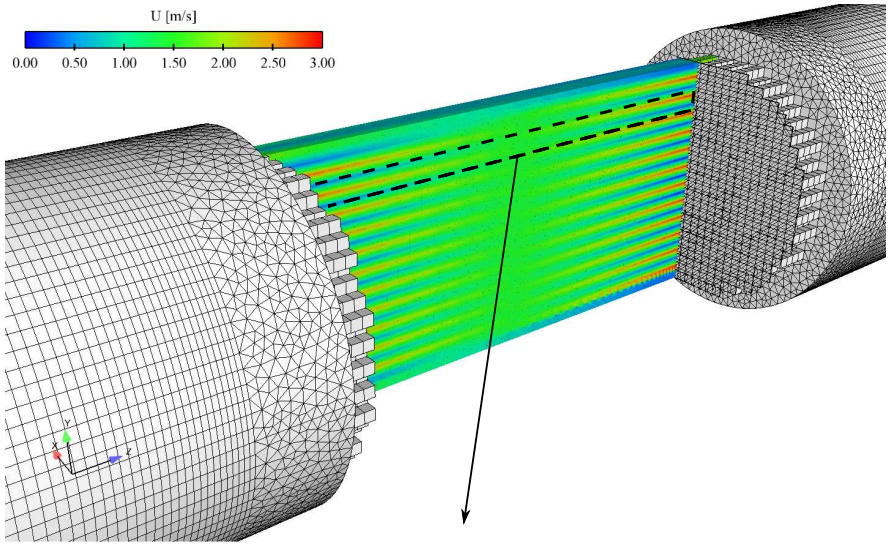




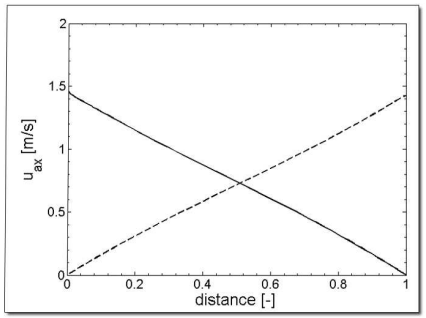
- Pressure drop across the filter monolith has been calculated for two different monolith lengths:
  - L=114 mm (plot on the left)
  - L=165 mm (plot on the right)
- porous medium permeability for the clean trap has been tuned to match the pressure drop at the lowest flow rate (L=114 mm)
- The agreement between simulations and experiments look quite satisfying both for low and high flow rates

Filter Hydrodynamics

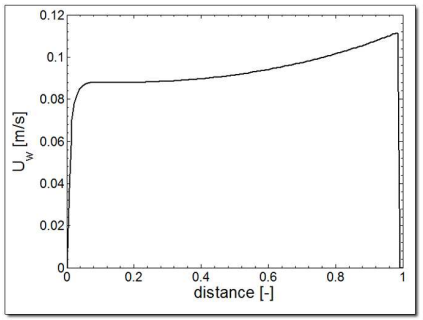
Velocity field



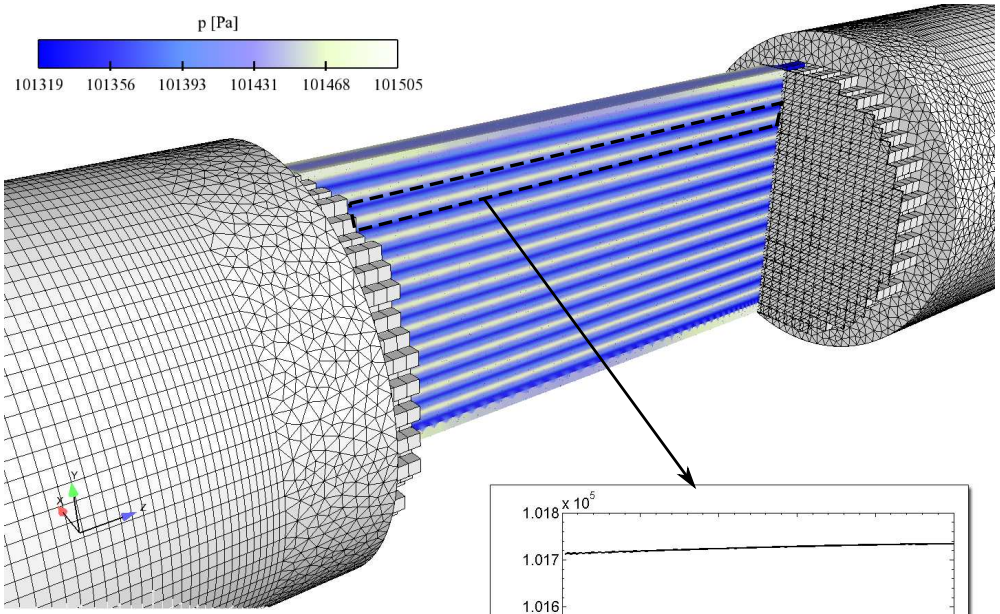
Axial Velocity

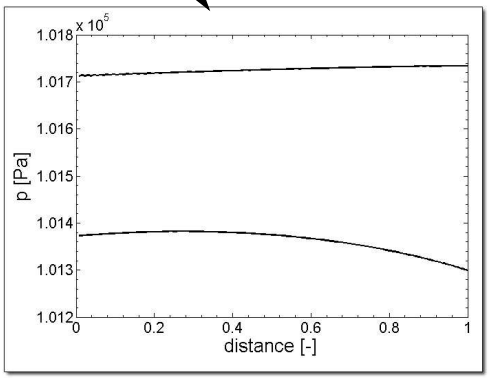


Wallflow Velocity

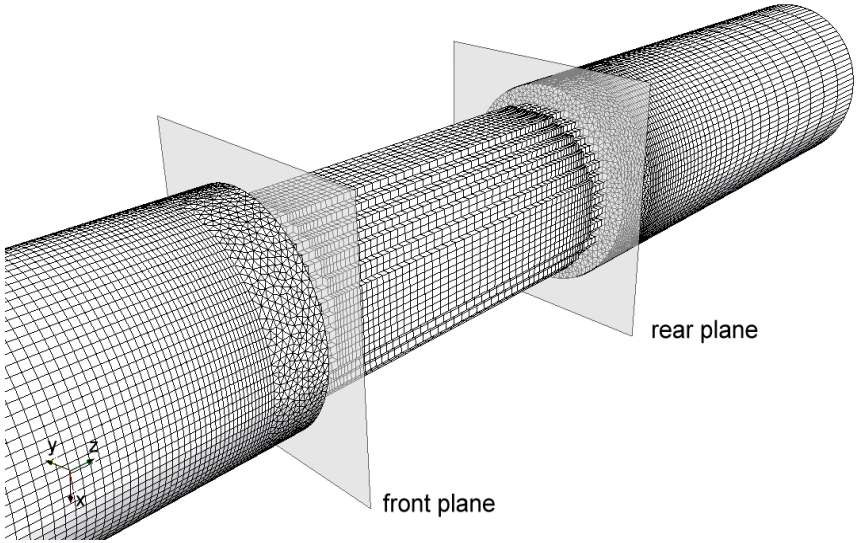


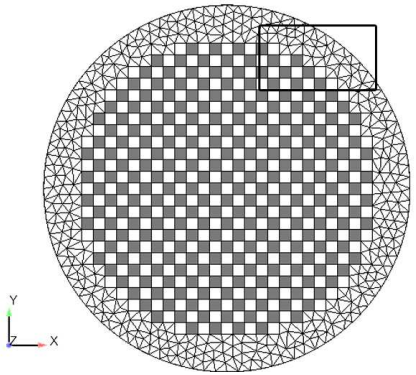
Pressure field

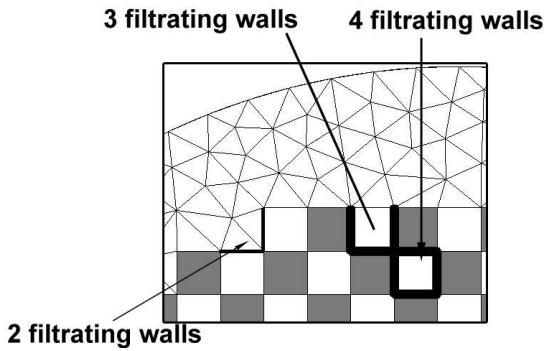




Filter Hydrodynamics







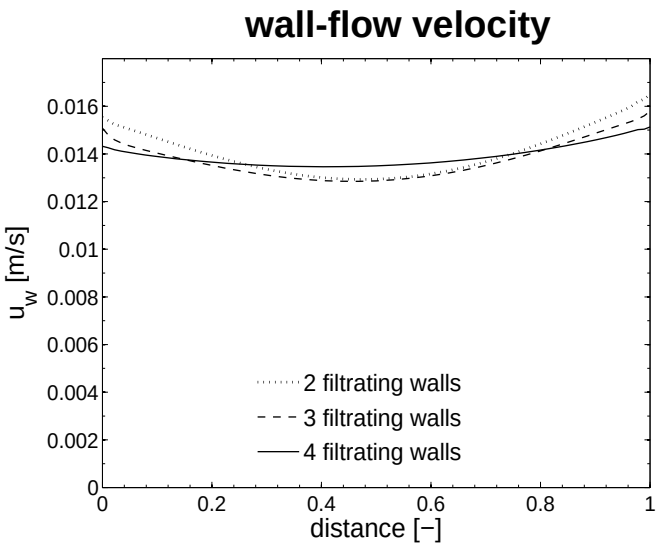
Filter channels may have a **different number of filtrating walls** :

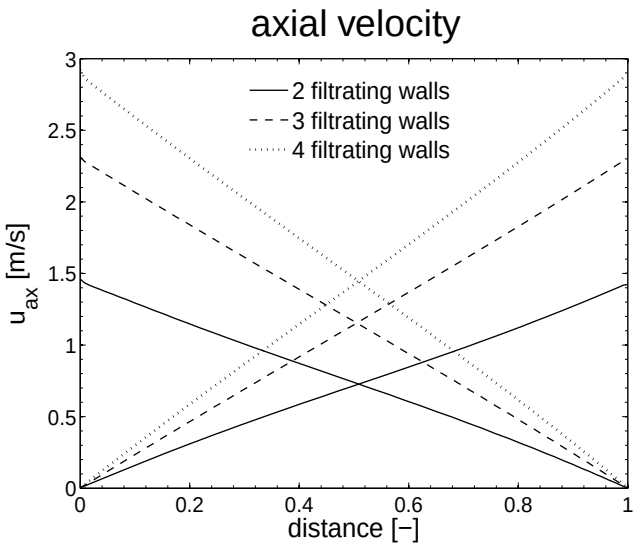
- internal channels have 4 filtrating walls
- channels at the boundary may have 2 or 3 filtrating walls

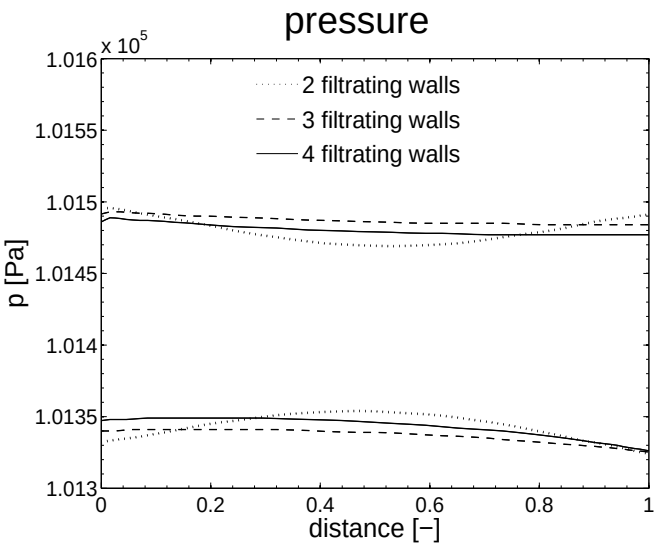
Flow velocity and channel pressure are expected to be different for channels having a different number of filtrating walls: **flow velocity and pressure field have been monitored for two different inlet flow rates: 15 Nm^3/h and 70 Nm^3/h**

Filter Hydrodynamics

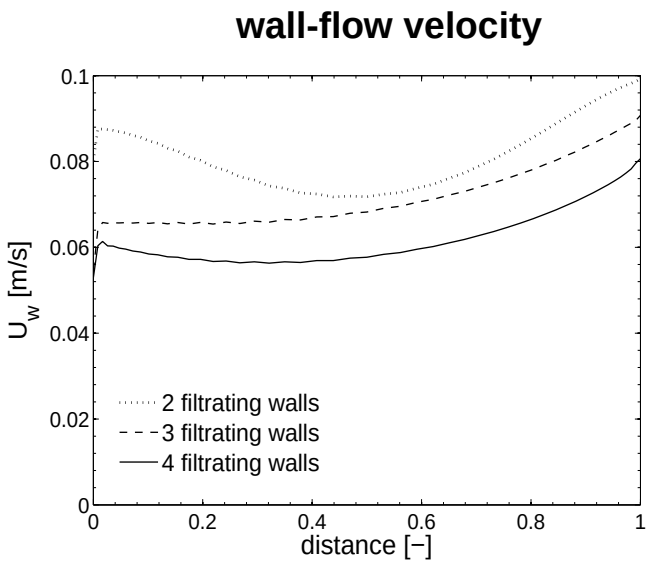
15 Nm^3/h

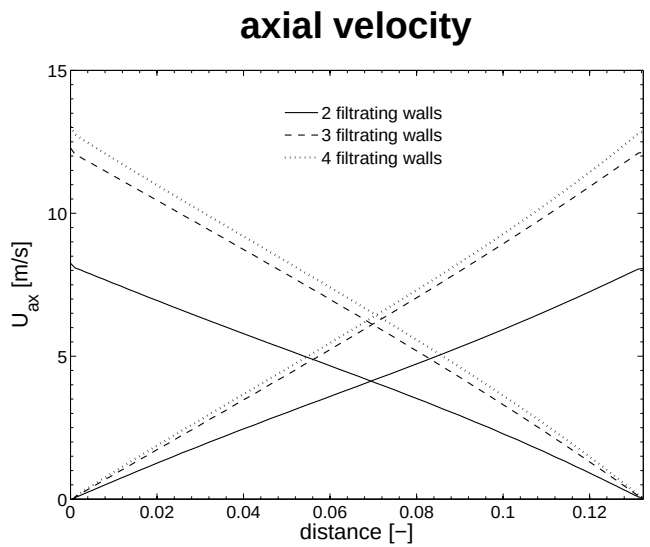


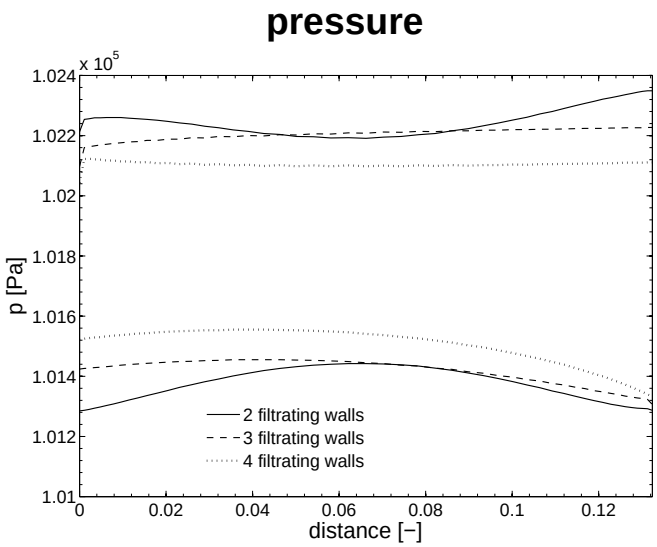




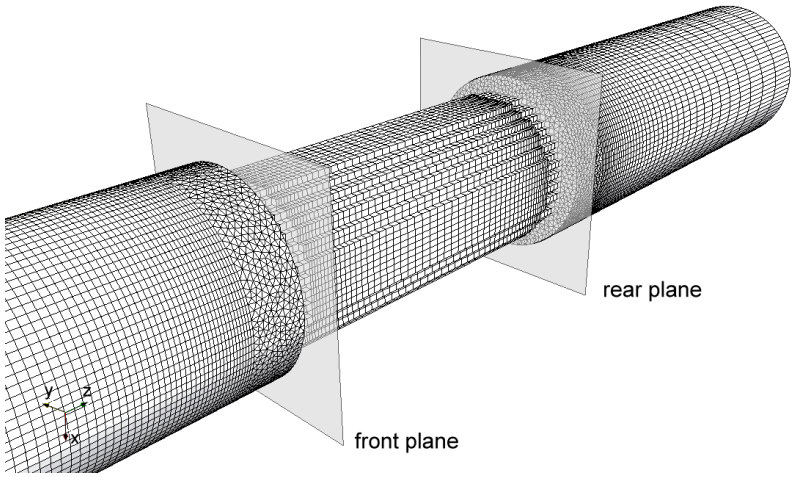
70 Nm^3/h

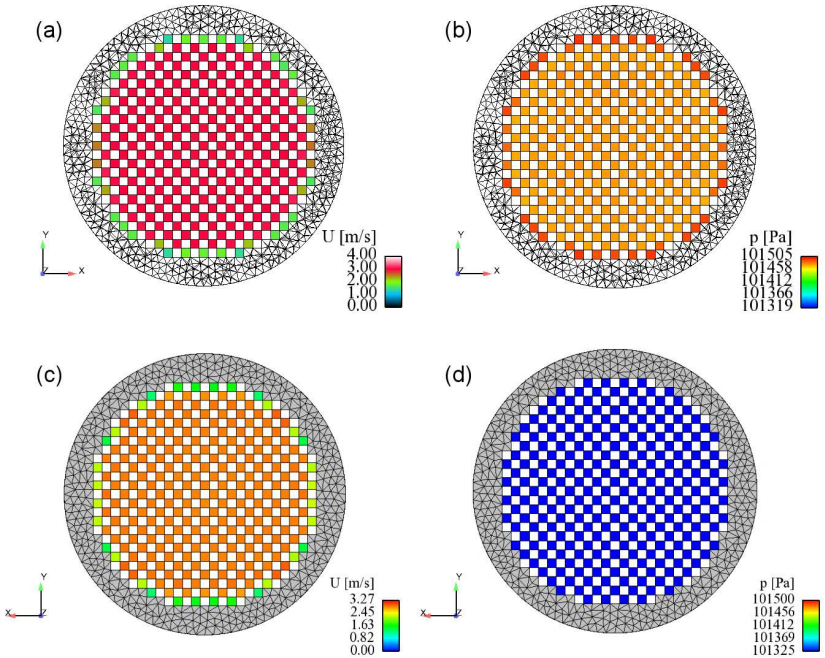






Non-uniform distribution of the flow at filter ends





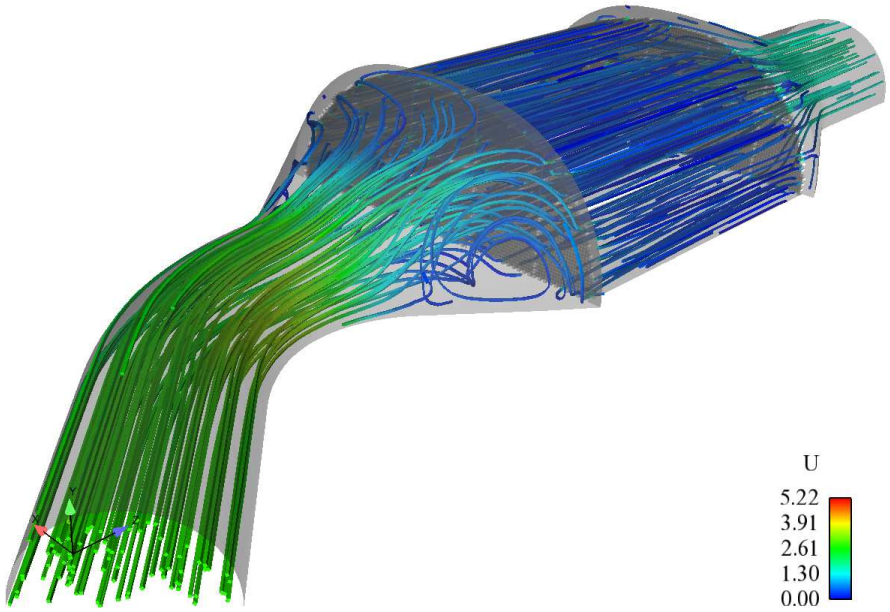
- 3D full-scale approach allows for prediction of flow non-uniformities at filter ends

- Flow non-uniformity may be estimated through the parameter Γ:

$$\Gamma = 1 - \frac{1}{2} \frac{\sum_i |u_i - u_{\text{mean}}| \cdot S_i}{\frac{\pi D^2}{4} u_{\text{mean}}}$$

- Despite the **uniformity index** can give an **overview of the uniformity of the flow at the frontal and rear transverse section of the filter**, it gives neither information about the location of the maximum velocity over the transverse section, nor about the ratio v_{ratio} = v_{max}/v_{mean}. These indicators may be important to define flow uniformity.

Non-uniform distribution of the flow at filter ends



- 3D full-scale approach allows for prediction of flow non-uniformities at filter ends
- Flow non-uniformity may be estimated through the parameter Γ:

$$\Gamma = 1 - \frac{1}{2} \frac{\sum_i |u_i - u_{\text{mean}}| \cdot S_i}{\frac{\pi D^2}{4} u_{\text{mean}}}$$

- Despite the **uniformity index** can give an **overview of the uniformity of the flow at the frontal and rear transverse section of the filter**, it gives neither information about the location of the maximum velocity over the transverse section, nor about the ratio v_{ratio} = v_{max}/v_{mean}. These indicators may be important to define flow uniformity.

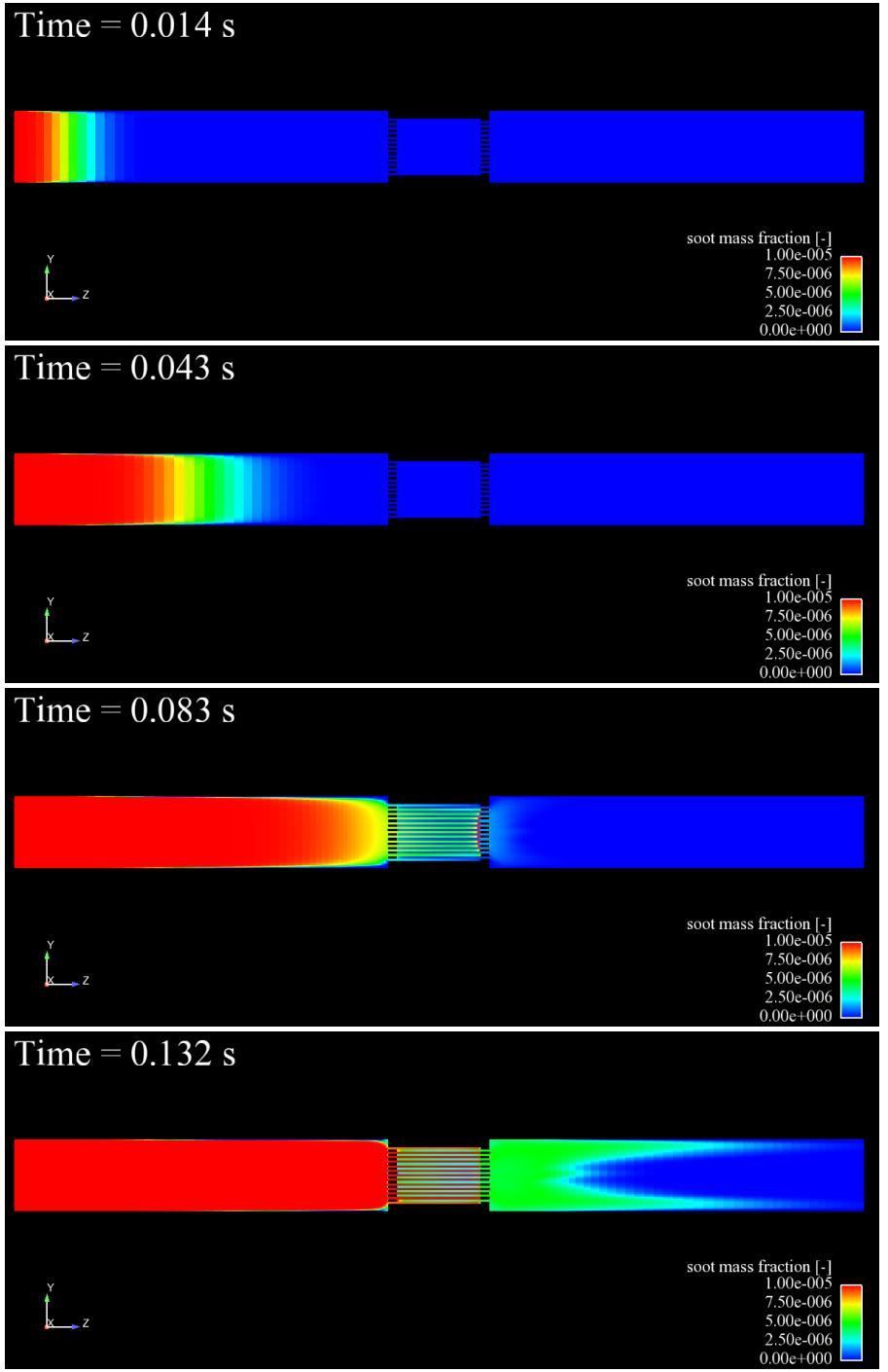
Soot transport and deposition

- **Soot is transported as a chemical specie**: previous studies [3, 2] indicates that Stokes number of soot particles in engine exhaust gas flow is very low ($\mathrm{St} < 10^{-4}$);

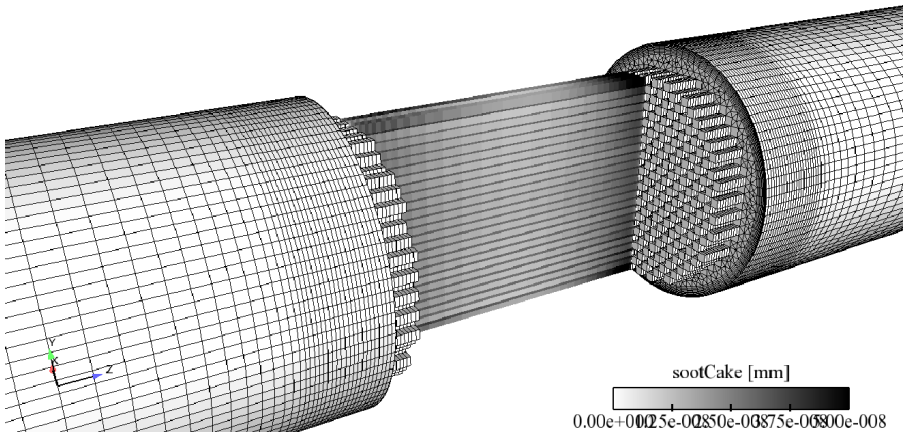
$$\frac{\partial\left(\rho\vec{Y}\right)}{\partial t} + \vec{\nabla}\cdot(\rho\vec{u}\vec{Y}) + \vec{D} = 0$$

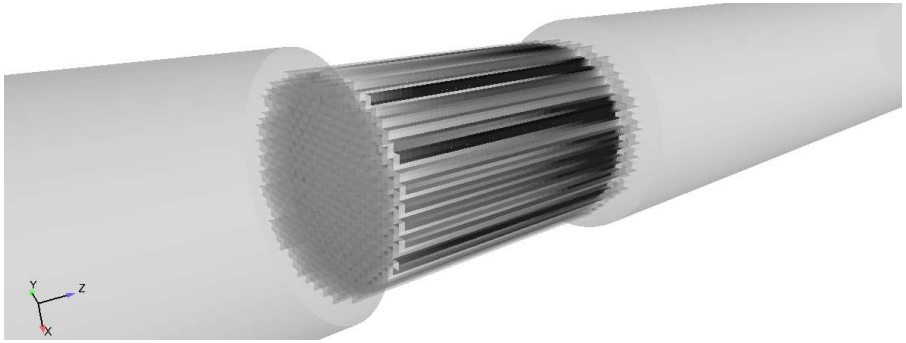
- **Soot is removed by an implicit sink term $\vec{D}$ applied on porous faces:** the amount of soot removal is determined by the filtration efficiency of that particular face;

- During **filter loading** the trapped soot particles change the following quantities, defined as porous-cell face properties:
  - soot cake thickness
  - collection efficiency
  - hydrodynamic resistance (Darcy coefficient)



Soot transport and deposition





- **The filtration sub-model used is based on the Konstandopoulos and Johnson's model [11]:** the particulate matter is trapped inside the porous media and it is also deposited on the filter substrate microstructure.

$$m_{\text{soot}} = \phi \cdot m_{\text{cake}} + (1-\phi) m_{\text{trapped}}$$

where the **partition coefficient** ϕ determines the amount of soot which that fills the porous medium or that builds up the soot cake:

$$\phi = \frac{d_c^2 - d_{c,0}^2}{(\psi b)^2 - d_{c,0}^2}$$

and it is a function of the **unit collector diameter** d_c defined as:

$$d_c = 2\left[\frac{3}{4\pi} \cdot \frac{m_c}{\rho_{soot,w}} + \frac{d_{c,0}}{2}\right]^{1/3}$$

Soot transport and deposition

- **Local wall porosity** ε change continuously as filtration proceeds:

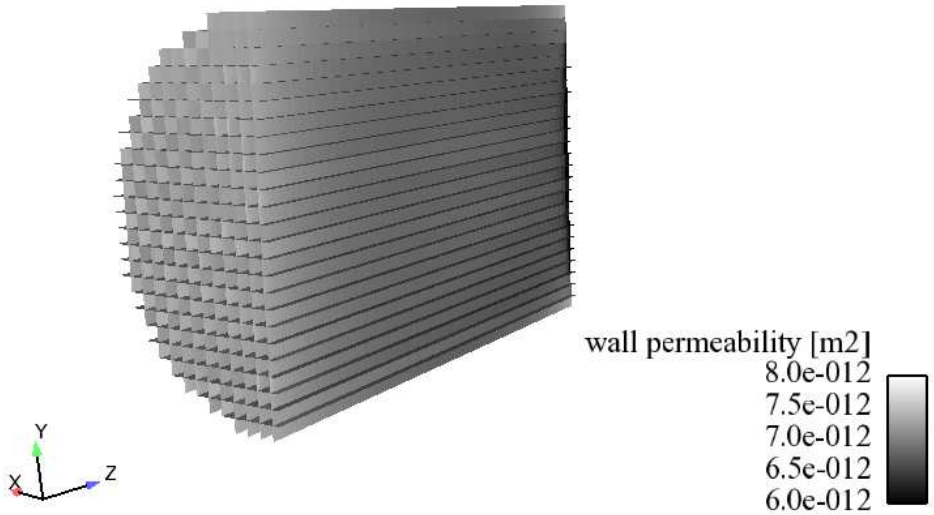
$$\varepsilon = 1 - \left(\frac{d_c}{d_{c,0}}\right)^3 (1 - \varepsilon_0)$$

- **Collection efficiency** is determined by two concurring mechanisms:
  - Direct interception by spherical collectors η_R
  - Brownian diffusion η_D

$$E = 1 - \exp\left[-\frac{3(1-\varepsilon)(\eta_D + \eta_R - \eta_D\eta_R)w_{\text{soot}}}{4\varepsilon d_c}\right]$$

- **Wall resistance** is given by the sum of porous wall and soot cake:

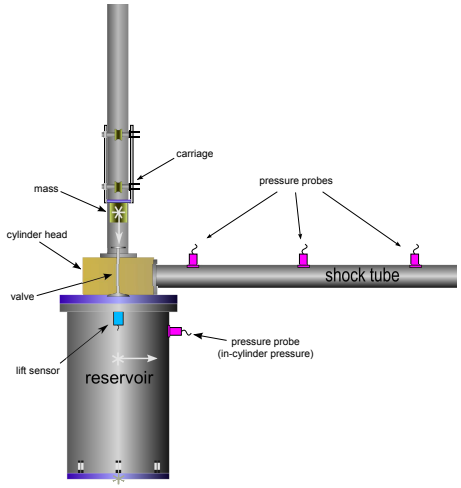
$$\Delta P = \left[\frac{w_{\text{wall}}}{k_{\text{wall}}} + \frac{w_{\text{soot}}}{k_{\text{soot}}}\right] \cdot \mu_w \cdot u_n$$

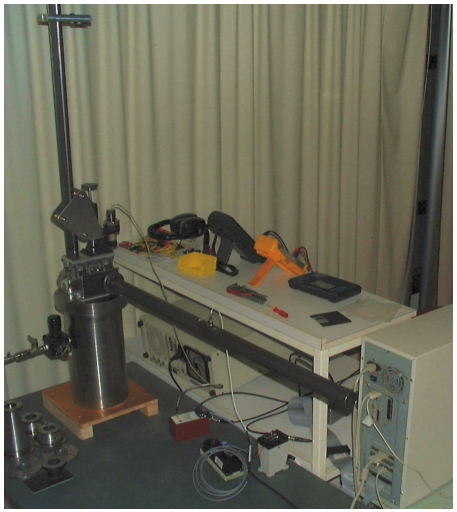


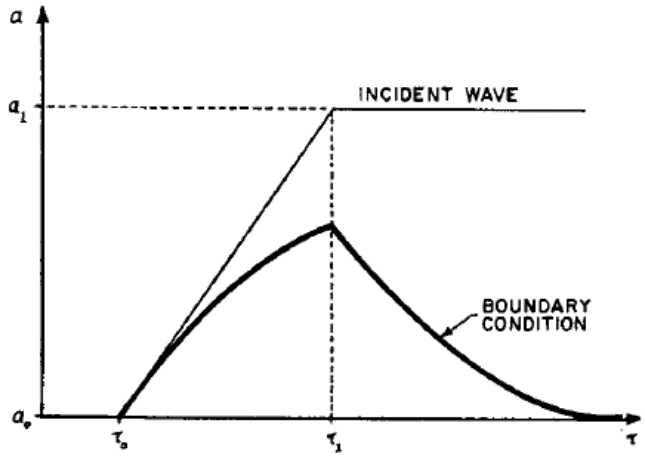
- **Wall permeability** changes accordingly to the wall porosity:

$$k_{\text{wall}} = k_{\text{wall},0} \left(\frac{d_c}{d_{c,0}}\right)^2 \frac{f(\varepsilon)}{f(\varepsilon_0)} \cdot \frac{1-\varepsilon_0}{1-\varepsilon}$$

Partially Reflecting Open End







When a shock wave reaches the open end of a pipe, a sudden increase of pressure from the undisturbed value p_0 to the value p occurs, so the values of $p(t) - p_0$ would be represented by a step function:

$$\begin{cases} p(t) - p(0) = 0 & \text{for} \quad t < 0 \\ p(t) - p(0) = p_1 & \text{for} \quad t > 0 \end{cases} \tag{1}$$

Fourier analysis must be carried out to decompose the function into its frequency components. Hence, the Fourier integral of a generic pressure wave must be used:

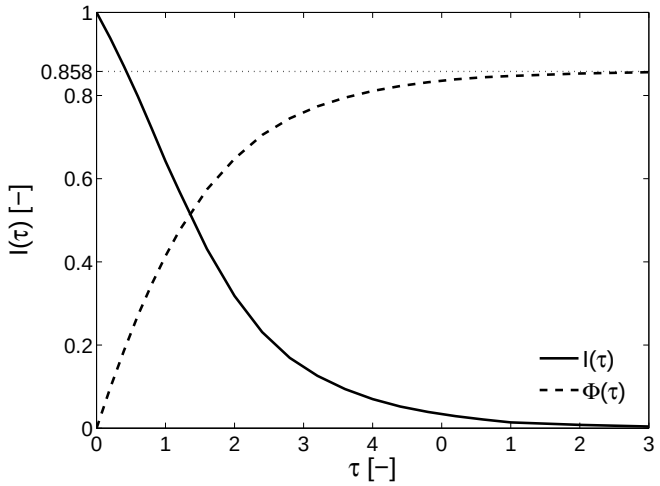
$$p - p_0 = \int_{-\infty}^{+\infty} A(\omega) \mathbf{e}^{i\omega(x - a_0 t)/a_0} d\omega \tag{2}$$

F. Piscaglia, A. Montorfano, A. Onorati, G. Ferrari. 'Modeling of Pressure Wave Reflection from Open-Ends in I.C.E. Duct Systems". SAE paper n. **2010-01-0XXX**, to be presented at the SAE World Congress 2010.

Partially Reflecting Open End

The Fourier's integral does not exist for a step function, because it does not satisfy the necessary condition for the existence of the Fourier integral. Step function must be therefore approximated by the modified function:

$$\begin{cases} p(t) - p(0) = 0 & \text{for} \quad t < 0 \\ p(t) - p(0) = p_1 \mathbf{e}^{-\beta t} & \text{for} \quad t > 0 \end{cases} \tag{3}$$



and

$$p - p_0 = \int_{-\infty}^{+\infty} A(\omega) \mathbf{e}^{i\omega(x - a_0 t)/a_0} d\omega \tag{4}$$

and the solution of the problem in the form of a Fourier integral is calculated as follows:

$$I(\tau) = \frac{p(\tau) - p_0}{p_1 - p_0} = 1 + \frac{1}{2\pi i} \int_{-\infty}^{+\infty} \frac{(1 - \theta + i\chi)\mathbf{e}^{-i\omega\tau}}{(1 + \theta - i\chi)(w + i\beta')} dw \tag{5}$$

Function $I(\tau)$ indicates the deviation of the exit pressure from the steady-flow value and it is expressed in terms of pressure rise through the incident wave. Since $I(\tau)$ is a function of τ only, the solution is independent of the wave strength or the gas properties. $I(\tau)$ **must be integrated numerically**.

Partially Reflecting Open End

The response of the pipe end in the time domain to a pressure step can be calculated as:

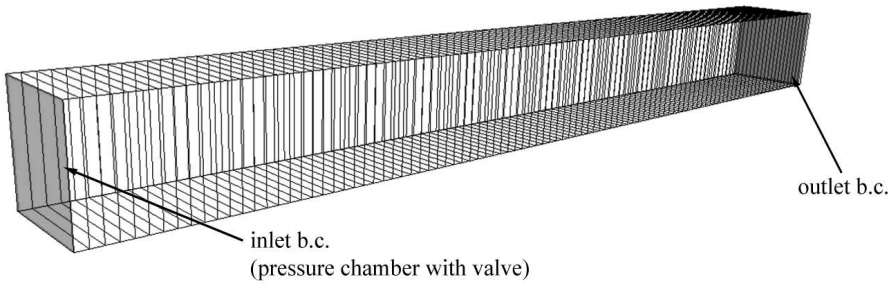
$$p_{end}(\tau) - p_0 = (p_i - p_0) I(\tau) \tag{6}$$

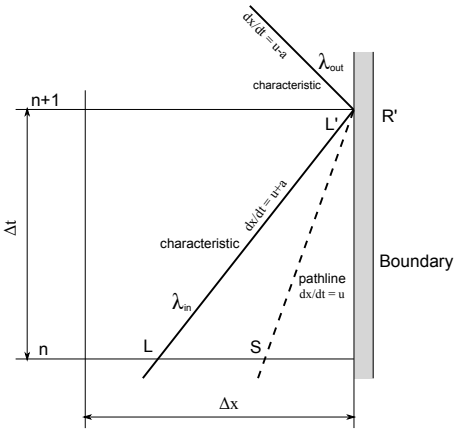
where $p_i - p_0$, that is the pressure rise due to the incident shock wave expressed as a function of $I(\tau)$. $I(\tau)$ represents the response of an arbitrary incident wave, that can be seen as the combination of an infinite number of pressure steps, having an infinitesimal duration; the response function of a complex could be written as:

$$p_{end}(\tau) - p_0 = F(\tau_0) I(\tau - \tau_0) + \int_{\tau_0}^{\tau} \frac{dF(\vartheta)}{d\vartheta} I(\tau - \vartheta) d\vartheta \tag{7}$$

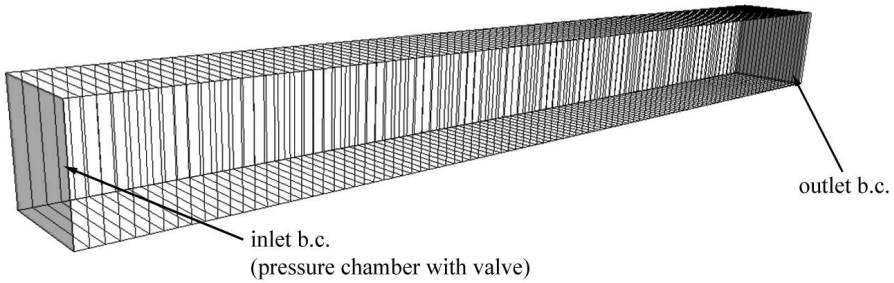
Discrete integration is performed by series of terms each of which represents the contribution at the outlet pressure from all the preceding elements of the incident wave:

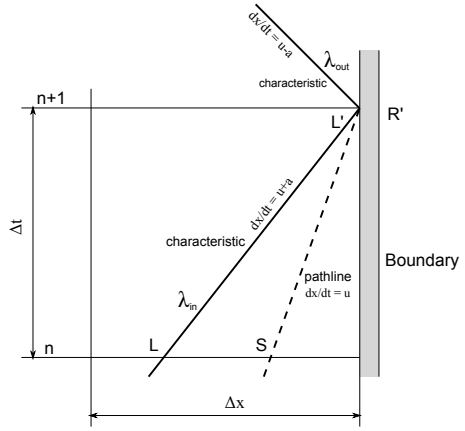
$$p_{end}(\tau_n) = p_{\text{amb}} + \sum_{j=1}^{n} \frac{p_{in}(\tau_j) - p_{end}(\tau_{j-1})}{\tau_j - \tau_{j-1}} \left[\Phi(\tau_n - \tau_{j-1}) - \Phi(\tau_n - \tau_j) \right] \tag{8}$$



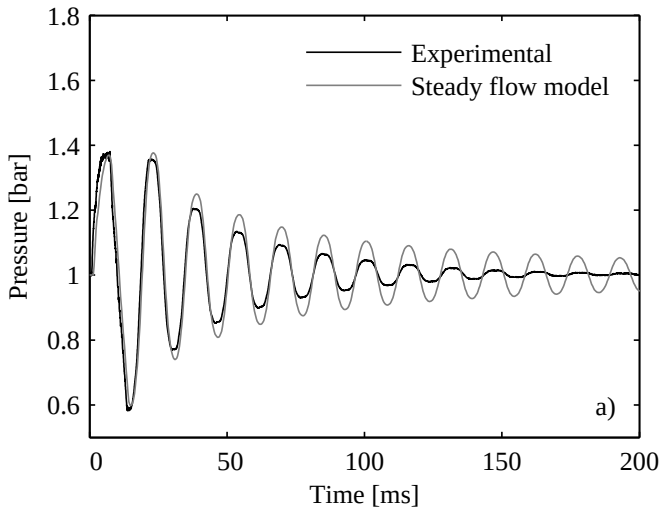


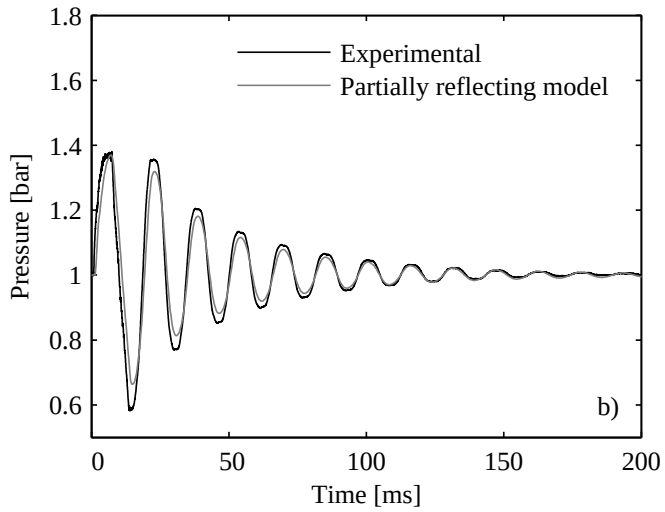
Partially Reflecting Open End VS const Pressure Model



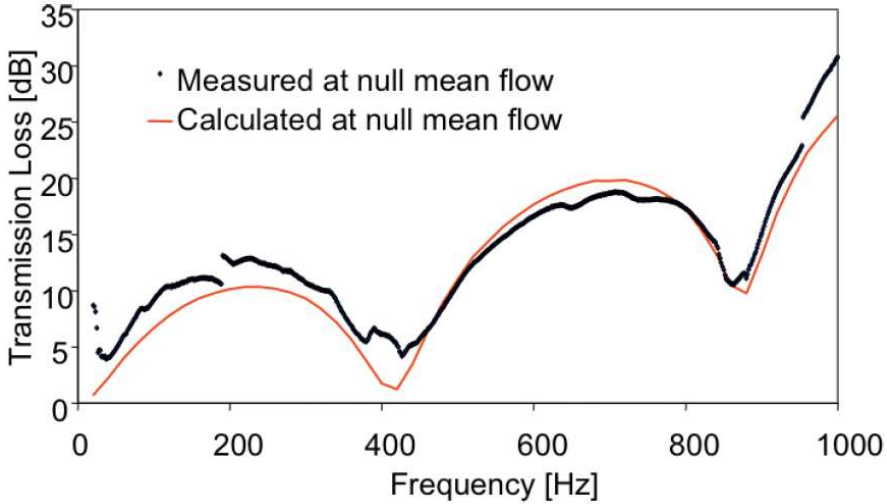


No reflected waves can propagate into the duct if the outflow velocity is sonic or supersonic; hence, calculation of function I(τ), which includes **information about the frequency dependence of the acoustic impedance of an open end, is not influenced by the conditions inside the duct**; so, it can be correctly applied also to subsonic flows to compute wave reflection in the surrounding gas



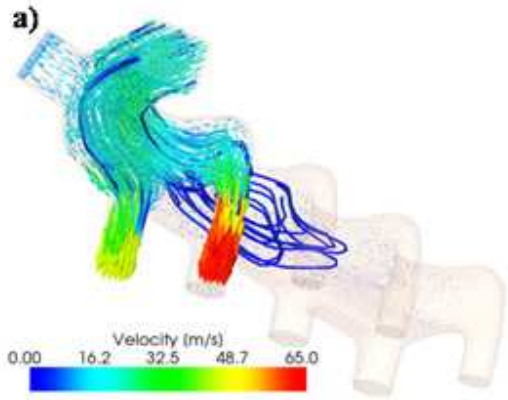


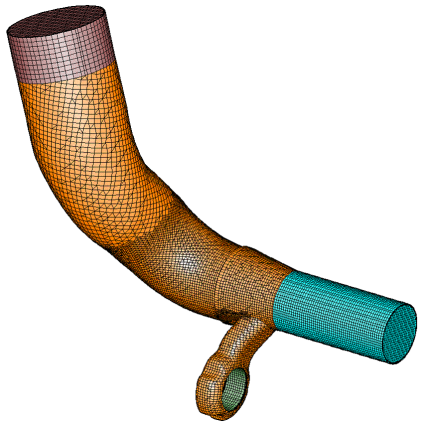
Current development: acoustic simulation



- **Development of the OpenFOAM® to perform <u>ACOUSTIC SIMULATIONS</u>**
  - gasdynamic noise radiated from silencers having complex geometries

Current development





- **Development of the code to perform simulation of SCR systems in OpenFOAM®**
- **Numerical Solvers**:
  - **Unsteady compressible solver** to calculate filter pressure drop during engine transients
  - **Fully implicit solver** (block-matrix solver): pressure drop across the porous cell face is written as a function of the velocity $\mathbf{u}^{n+1}$ at the current time-step, in order to improve stability and performance
- **Submodels for simulation of DPFs:**
  - THERMAL and CHEMICAL models implemented as OpenFOAM® classes for filter regeneration
- **Development of the OpenFOAM® to perform ACOUSTIC SIMULATIONS**
  - gasdynamic noise radiated from silencers having complex geometries
- **FUTURE WORK: LES modeling**

Thanks for your attention!

Federico Piscaglia, Ph.D.
Assistant Professor of Internal Combustion Engines

CONTACT INFORMATION

Address	Dipartimento di Energia, Politecnico di Milano (campus Bovisa) via Lambruschini 4, 20156 Milano (ITALY)
E-Mail:	federico.piscaglia@polimi.it
Phone:	(+39) 02 2399 8620
Fax:	(+39) 02 2399 3863
Web page:	http://www.engines.polimi.it/

References

[1] **F. Piscaglia, A. Montorfano, and A. Onorati**. Development of a Multi-Dimensional Parallel Solver for Full-Scale DPF Modeling in OpenFOAM. SAE paper n. 2009-01-1965, SAE 2009 International Powertrains, Fuels and Lubricants Meeting, June 15-17, 2009, Florence, Italy, 2009, 2009.

[2] **F. Piscaglia, A. Onorati, C. J. Rutland, and D. E. Foster**. "Multi-dimensional modeling of the soot deposition mechanism in diesel particulate filters". In SAE Transactions, Journal of Fuel & Lubricants. SAE paper n. 2008-01-0444, SAE 2008 Int. Congress & Exp. (Detroit, Michigan), 2008.

[3] **F. Piscaglia, C. J. Rutland, and D. E. Foster**. "Development of a CFD model to study the hydrodynamic characteristics and the soot deposition mechanism on the porous wall of a diesel particulate filter". SAE paper n. 2005-01-0963, SAE 2005 Int. Congress & Exp. (Detroit, Michigan), 2005.

[4] **G. Rudinger**. Wave Diagrams for Nonsteady Flow in Ducts. D. Van Nostrand Company (Canada), Ltd., 1955.

[5] **F. Piscaglia and G. Ferrari**. A novel 1d approach for the simulation of unsteady reacting flows in diesel exhaust after-treatment systems. Energy, 34 (12):2051–2062, 2009.

[6] **A. Konstandopoulos, M. Kostoglou, D. Zarvalis, P. Housiada, and N. Vlachos**. "Multichannel simulation of soot oxidation in diesel particulate filters". SAE paper n. 2003-01-0839, SAE 2003 Int. Congress & Exp. (Detroit, Michigan), 2003.

[7] **G. Koltsakis, O. Haralampous, N. Margaritis, Z. Samaras, C. Vogt, E. Ohara, Y. Watanabe, and T. Mizutani**. "3-Dimensional modeling of the regeneration in SiC particulate filters". SAE Paper n. 2005-01-0953, SAE 2005 Int. Congress & Exp. (Detroit, Michigan), 2005.

[8] AVL List GmbH. 2005.

[9] **C. Hinterberger, M. Olesen, and R. Kaiser**. "3D Simulation of Soot Loading and Regeneration of Diesel Particulate Filter Systems". 2007.

[10] **M. Masoudi, A. Heibel, and P. Then**. Predicting pressure drop of wall-flow diesel particulate filters – theory and experiment. SAE paper n. 2000-01-1084, SAE 2000 Int. Congress & Exp. (Detroit, Michigan), 2000.

[11] **A. G. Konstandopoulos and J. H. Johnson**. "Wall-flow diesel particulate filters - their pressure drop and collection efficiency". SAE paper n. 890405, SAE 1989 Int. Congress & Exp. (Detroit, Michigan), 1989.