NanoSim

Highly efficient «on-the-fly» data processing using the open-source library CPPPO

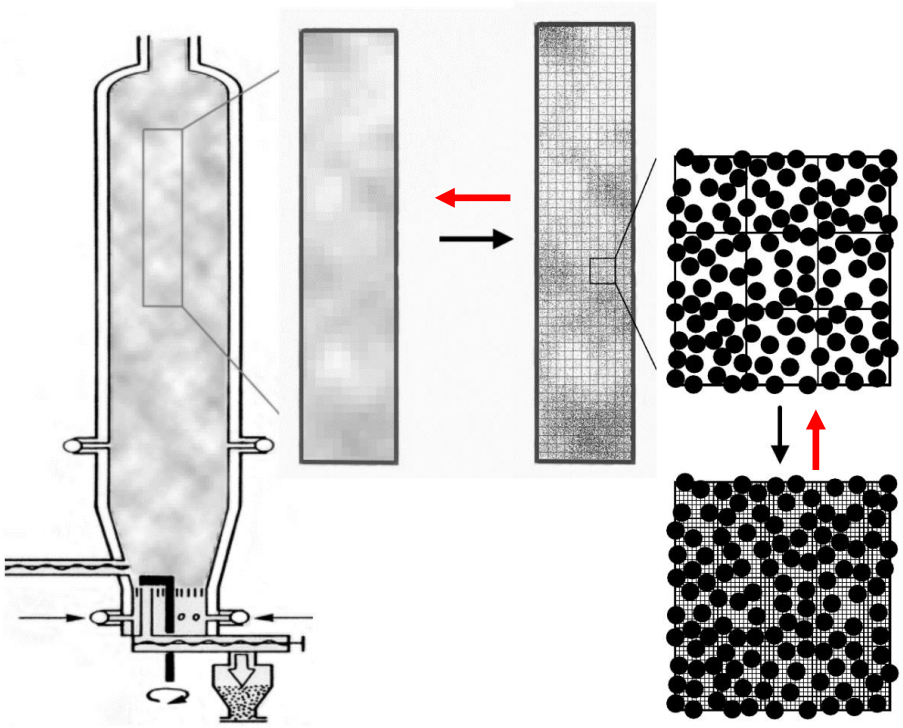
Graz University of Technology,
DCS Computing GmbH

Federico Municchi,
Stefan Radl,
Christoph Goniva

April 7 2016, Workshop HPC enabling of OpenFOAM®, CINECA, Casalecchio di Reno

- **Coarse graining and multiscale approach**
- CPPPO - overview
- CPPPO - algorithms and performance
- CPPPO - coupling to simulator
- Application to fluid-particle systems
- Conclusions

Coarse graining and multiscale approach



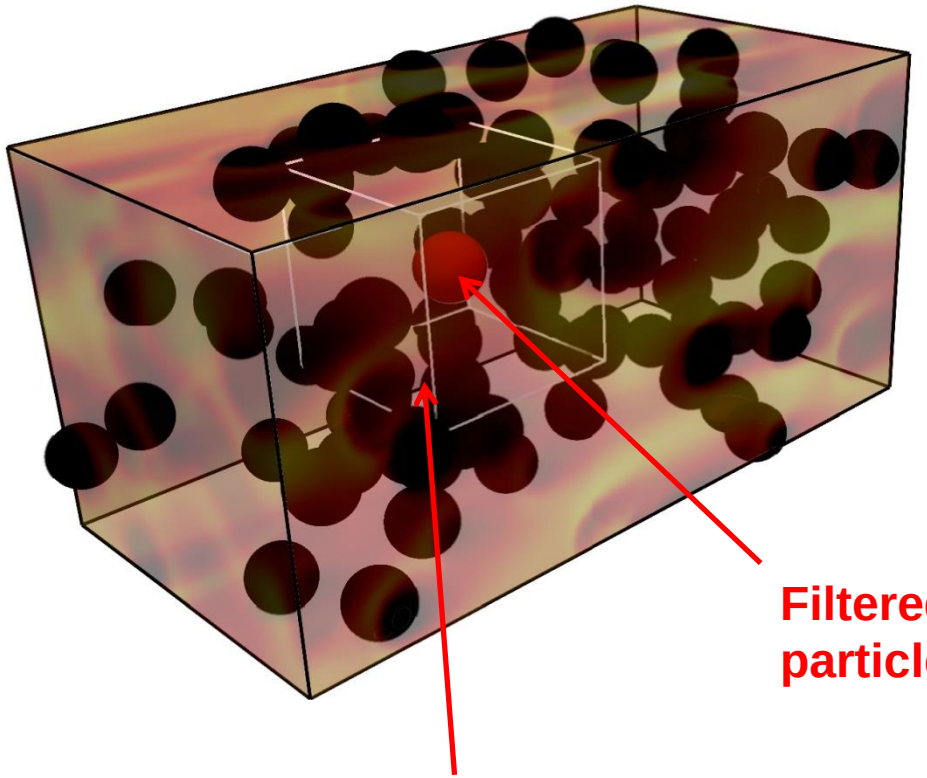
Van Der Hoef et al. 2006, Multiscale modeling of Gas-Fluidized beds, Advances in chemical engineering.

To solve the transport equations on (affordable) coarse grids we need to take into account transport phenomena occurring at sub-grid scales by mean of closure models («**material relations**»[1])

Closure models can be derived by **filtering «resolved» simulations**

[1]https://ec.europa.eu/research/industrial_technologies/pdf/review_of_materials_modelling_iv.pdf

Coarse graining and multiscale approach



Direct Numerical Simulation allows to calculate **particle-based** quantities like drag forces (f_p) and interphase transferred heat (Q_p) and the surrounding Eulerian fields (**not particle-based**)

Filtering allows to relate particle-based quantitities with Eulerian fields to evaluate **interphase exchange coefficients**.

Coarse graining and multiscale approach

Spatial filtering

$$\bar{\phi}(\boldsymbol{x},t) = \int_{\Omega} K(\boldsymbol{x} - \boldsymbol{x}', t)\phi(x',t)d\Omega'$$

$K(\boldsymbol{x} - \boldsymbol{x}', t)$ is the **Kernel function**

Top-Hat Kernel:

$$K(\boldsymbol{x} - \boldsymbol{x}', t) = \prod_i \frac{H\left(\frac{\Delta_i}{2} - |x_i - x_i'|\right)}{\Delta_i}\delta(t)$$

H : Heaviside step function

Δ_i : spatial filter cut-off lenght on the i direction

However, different Kernel functions can be found on literature.

- Coarse graining and multiscale approach
- **CPPPO - overview**
- CPPPO - algorithms and performance
- CPPPO - coupling to simulator
- Application to fluid-particle systems
- Conclusions

CPPPO - overview

Compilation of fluid/Particle Post Processing rOutines

CPPPO is a C++ library of **parallel** data processing functions.

It is a tool for "**offline scale bridging**", i.e., developing closures for coarse mesh models by **filtering** fine mesh data.

W. Holloway, PhD Thesis, 2012.

smokingdesigners.com

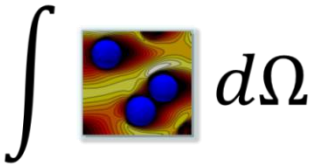
CPPPO - overview

Main features

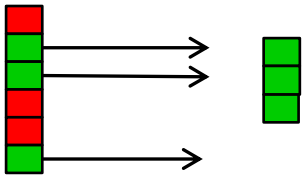
- It performs **on-line filtering** of data from particle/fluid flow simulators **running in parallel**.
- It provides a **number of tools** for drawing samples and performing statistical analysis.
- Can be linked to existing simulation software. Linking to **OpenFOAM®** is currently available as well as **CSV** interfaces for **ANSYS FLUENT®** or **Neptune CFD®**.
- It features **run-time specification of data operations** (no more coding to do fancy post-processing!).

CPPPO - overview

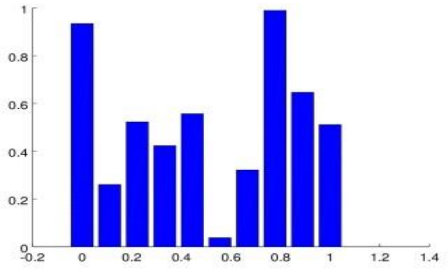
A typical CPPPO run consists of **three sets of operation** performed **on the fly** (i.e., while the solver is running).



Filtering of fluid and particle data, including **variance calculation**



Sampling of filtered data and their derivatives with **statistical biasing** (e.g. limiters)



Binning of sampled data using **running statistics**

CPPPO - overview

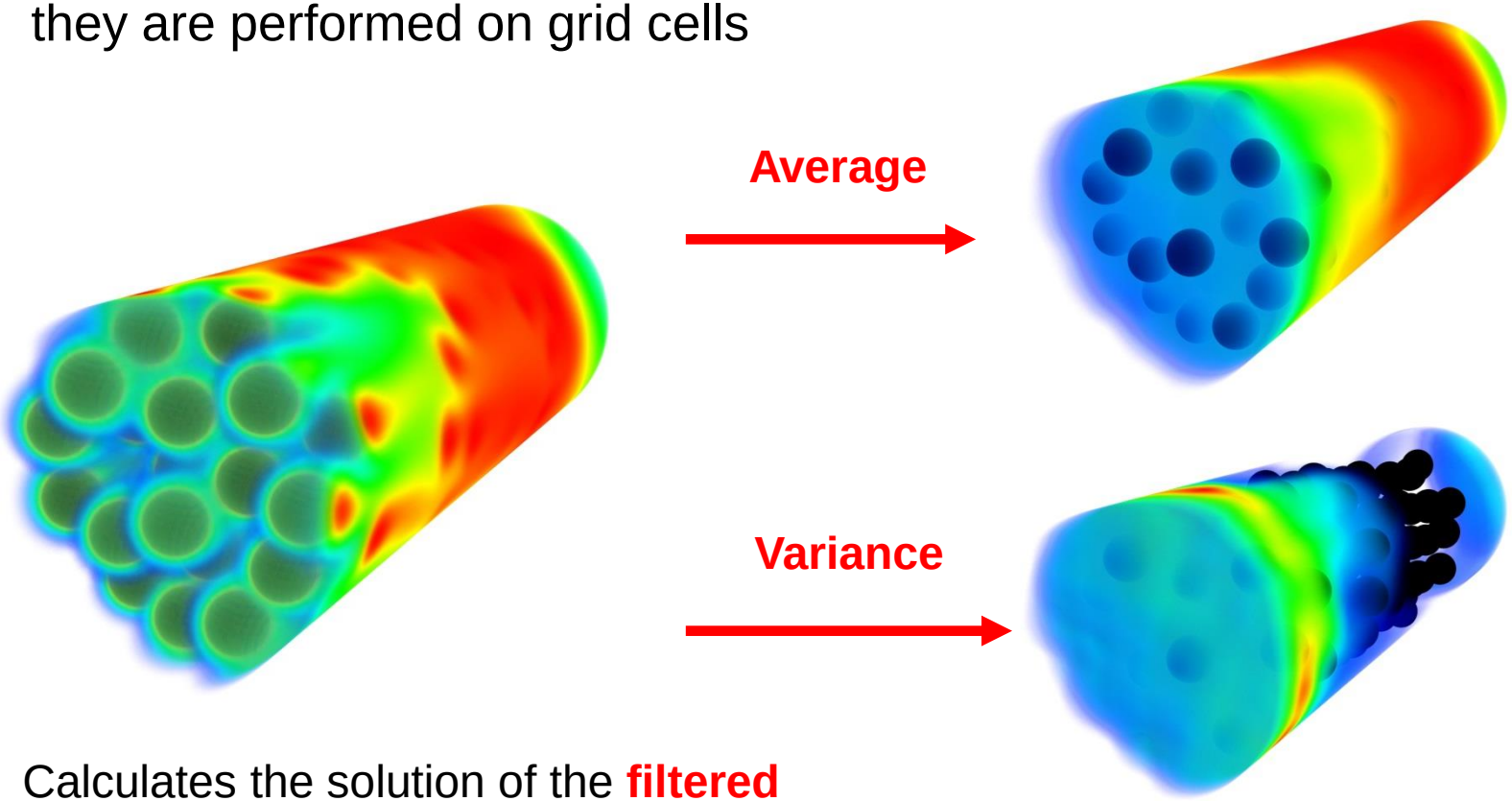
Filtering and Sampling operations are subdivided in:

- **Eulerian operations**: only Eulerian fields are involved
- **Lagrangian operations** : Lagrangian and Eulerian fields involved

Lagrangian fields can be read in the interface class or provided to the core library in *Json* format (i.e., when probing specific locations).

CPPPO - overview

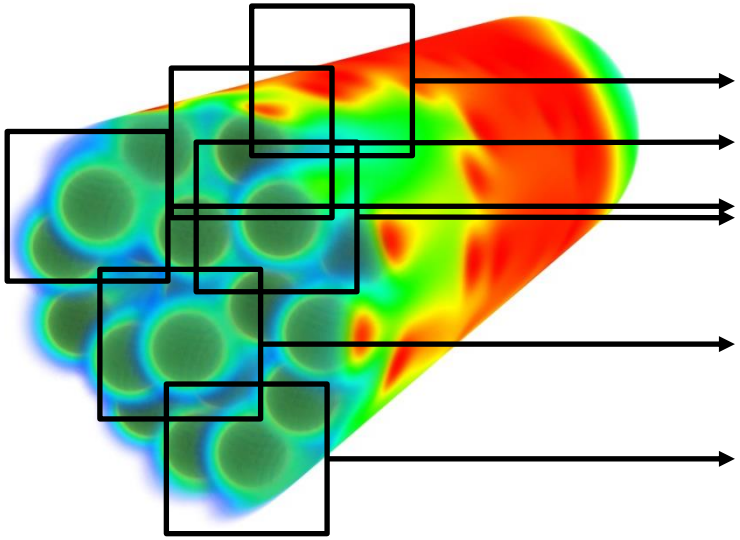
Eulerian filters and samples are **cell based,** i.e. they are performed on grid cells

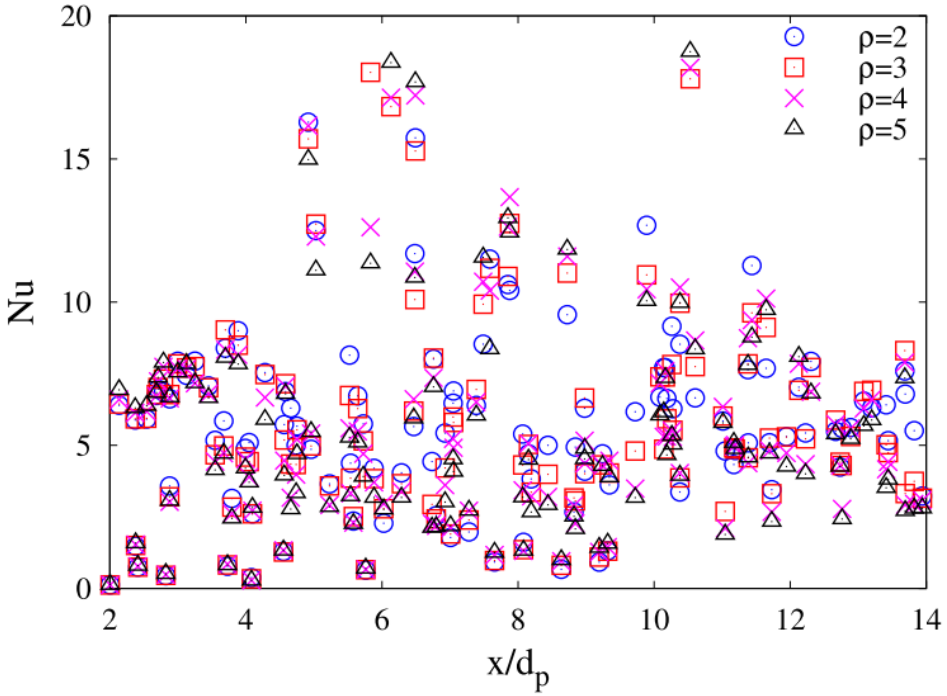


Calculates the solution of the **filtered equation**

CPPPO - overview

Lagrangian filters and samples are **particle based,** i.e. they are performed at specific user-defined locations

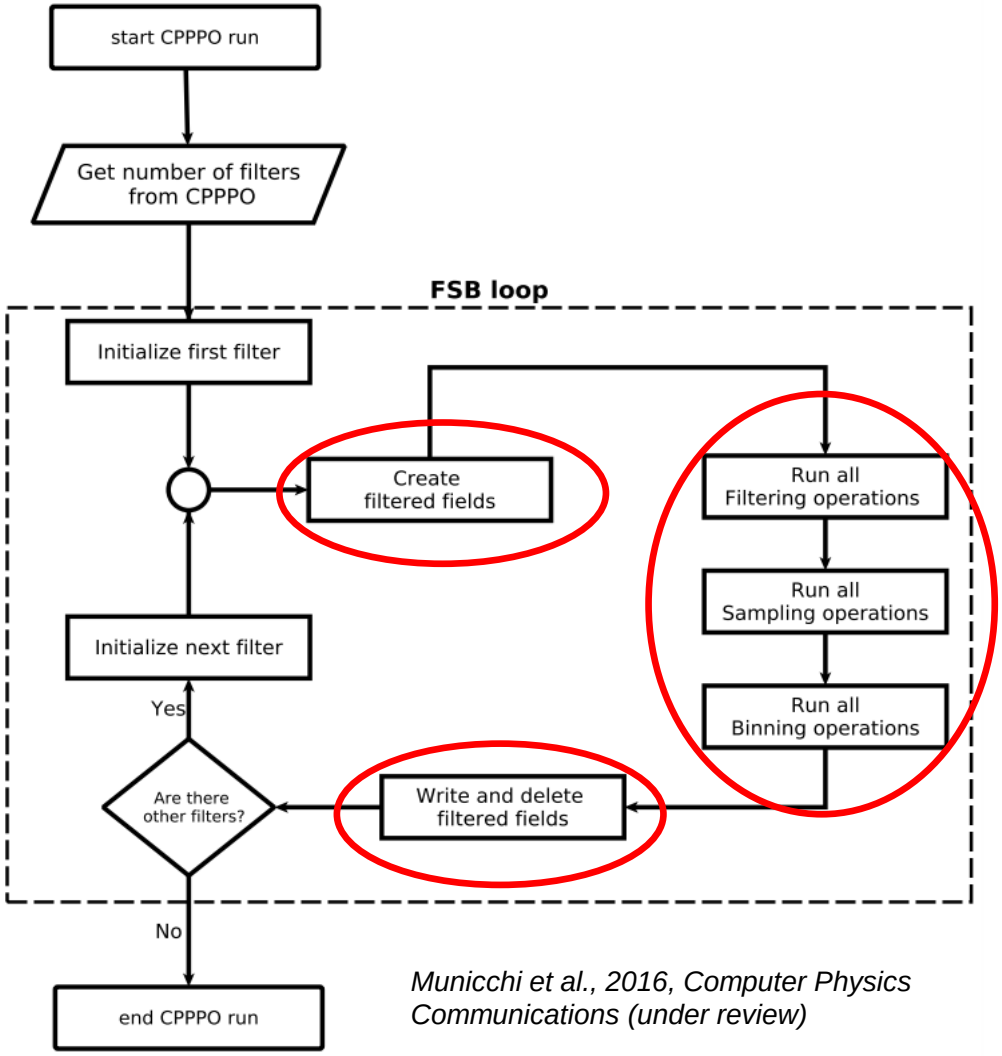




Municchi et al., 2016, Computer Physics Communications (under review)

- Coarse graining and multiscale approach
- CPPPO - overview
- **CPPPO - algorithms and performance**
- CPPPO - coupling to simulator
- Application to fluid-particle systems
- Conclusions

CPPPO – algorithms and performance



The Filtering-Sampling-Binning loop allows to **reduce** the amount of stored **memory** since different filters do not need to interact

Additional memory is allocated during operations

Municchi et al., 2016, Computer Physics Communications (under review)

CPPPO – algorithms and performance

Filtering operations are performed togheter with **Selectors.**

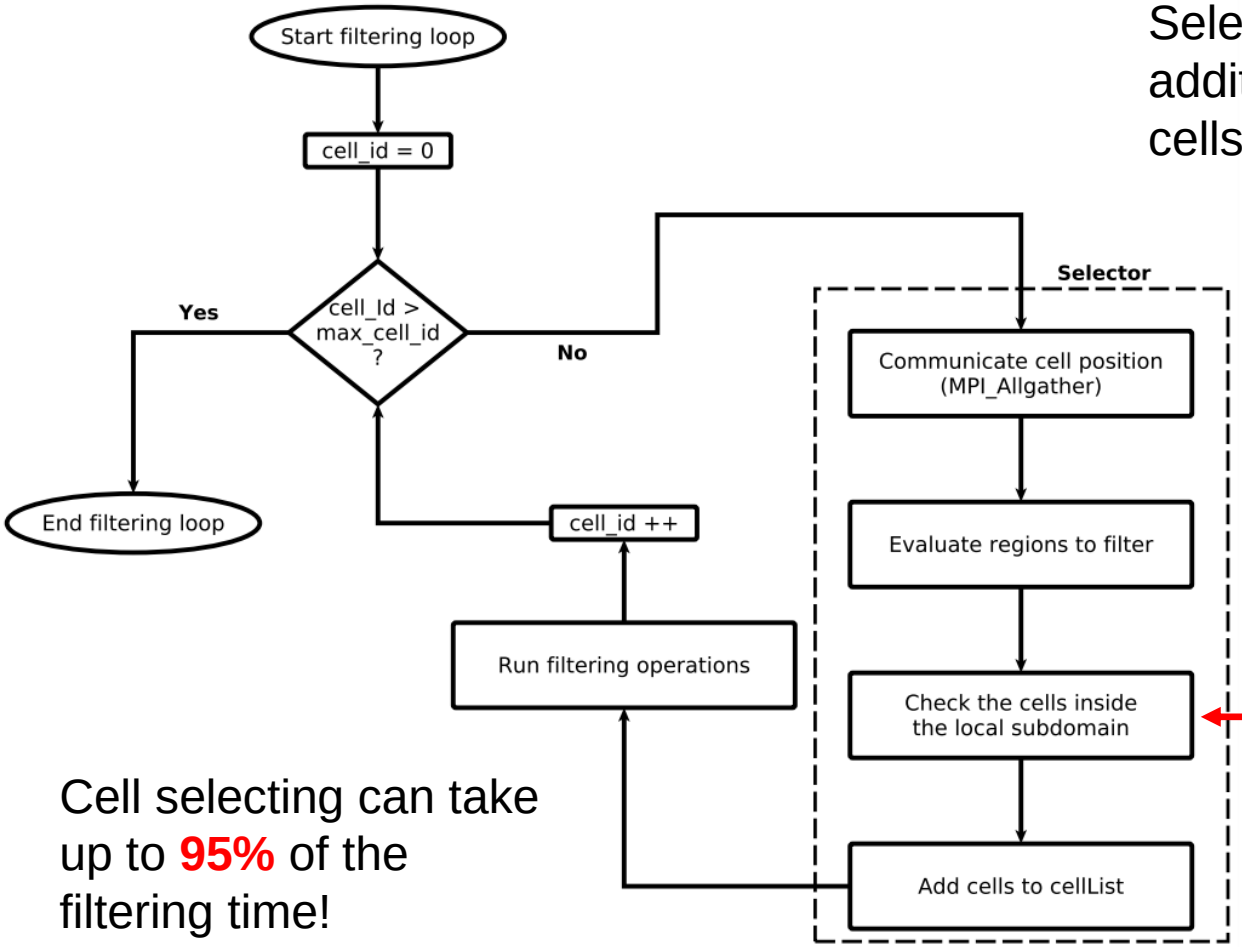
For each filtered cell, the selector evaluates the cells within the filter domain.

CPPPO features two selectors for filtering:

- **Cartesian** : for fully structured grids
- **Unstructured**: for general meshes

In addition, CPPPO features a **cell region selector** to evaluate zones of interest (e.g., bubbles) in the fluid domain.

CPPPO – algorithms and performance



Selectors require additional looping over cells.

Requires sub-looping

Cell selecting can take up to **95%** of the filtering time!

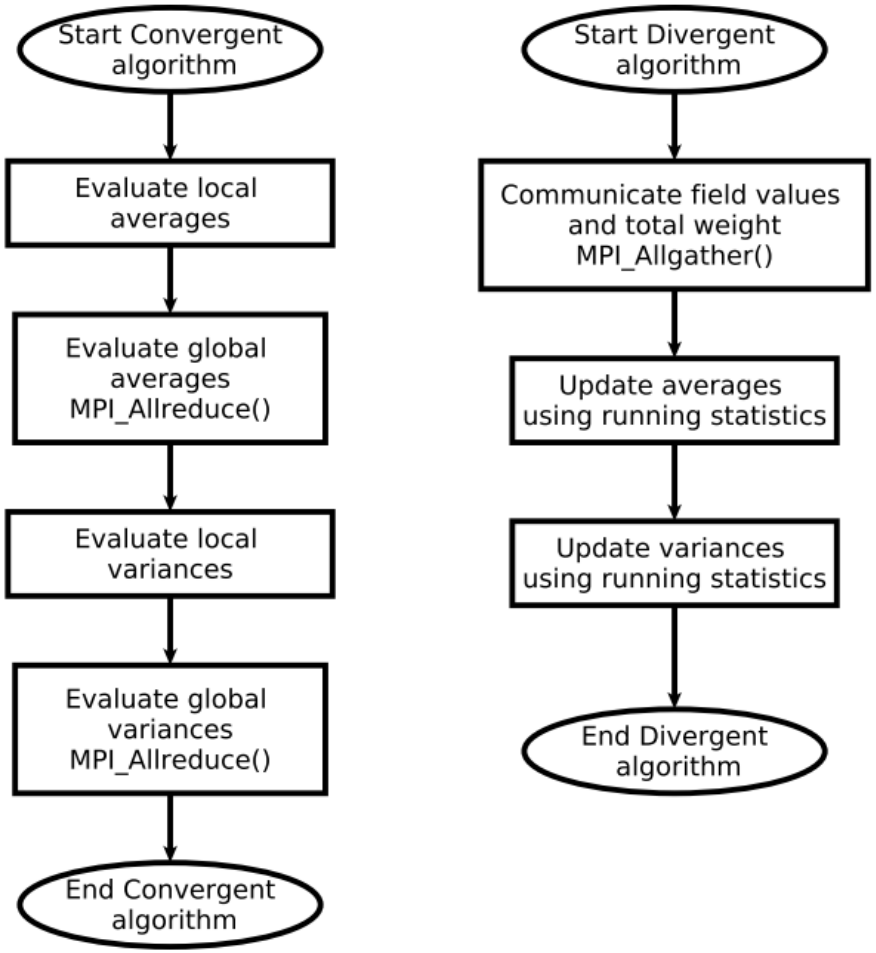
Municchi et al., 2016, Computer Physics Communications (under review)

CPPPO – algorithms and performance

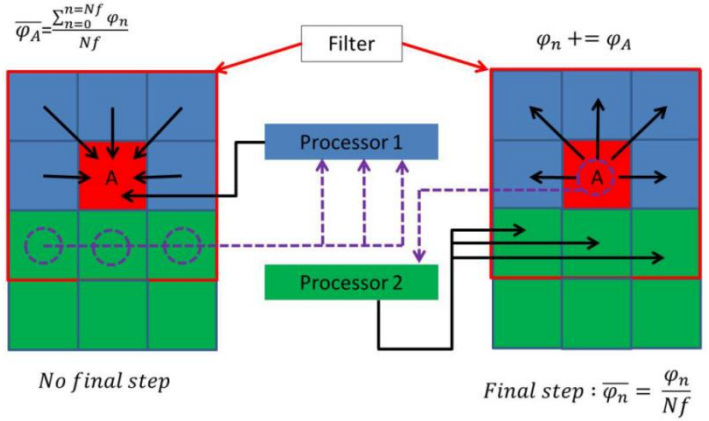
CPPPO features two different algorithms for filtering:

- **Convergent algorithm** : standard filtering algorithm where filtered fields are calculated «cell-by-cell»
- **Divergent algorithm** : all cells are filtered togheter and the filtered field is updated with successive steps using running statistics.

CPPPO – algorithms and performance



Municchi et al., 2016, Computer Physics Communications (under review)



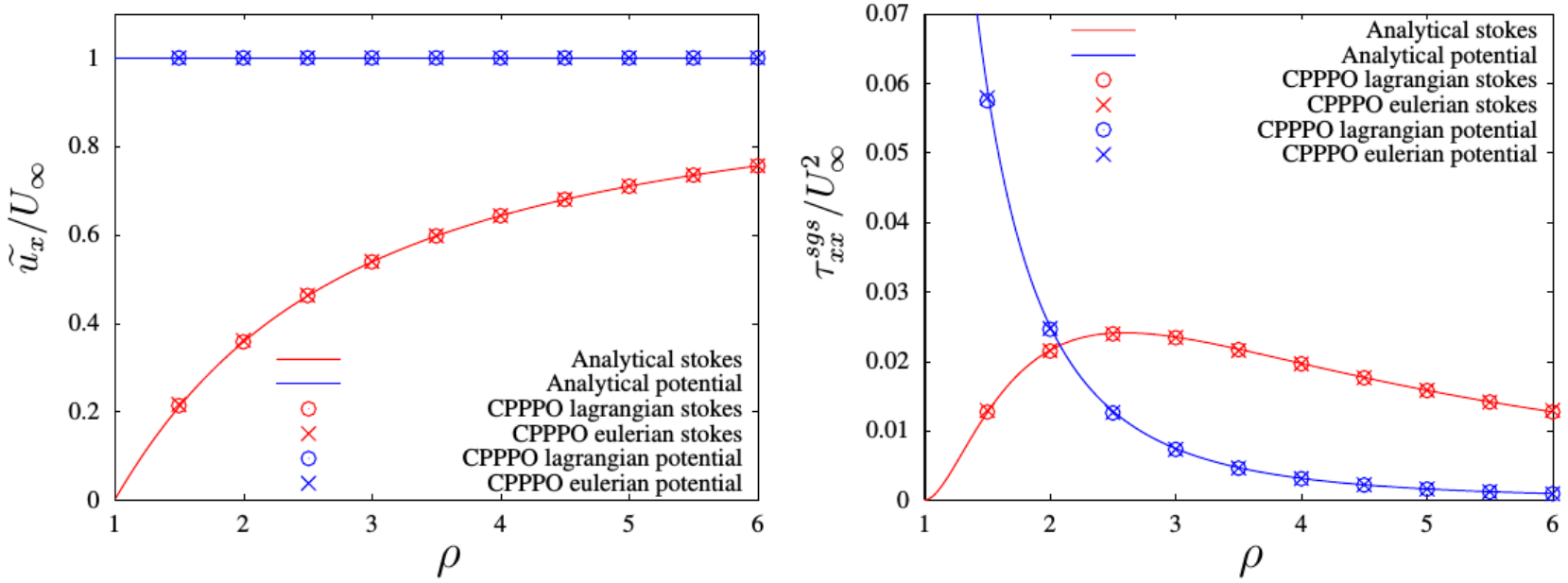
Municchi et al., 2016, Computer Physics Communications (under review)

In the **divergent algorithm** the data flow is directed to the neighboring processors. This leads to **improved performance when running in parallel**.

CPPPO – algorithms and performance

Test case: filtered flow at the center of a sphere

No significant difference in terms of accuracy between the two filtering algorithms when computing average and variance.



Municchi et al., 2016, Computer Physics Communications (under review)

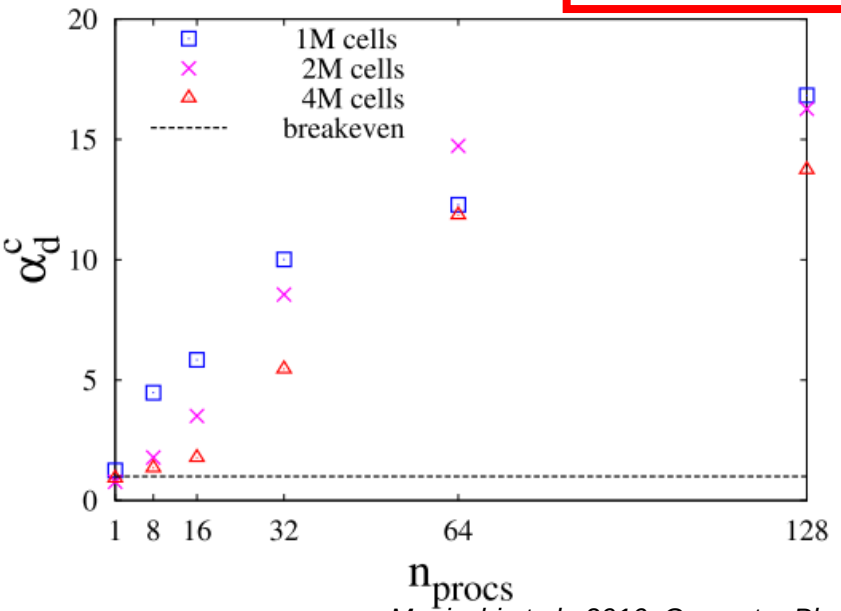
ρ is the dimesionless filter size

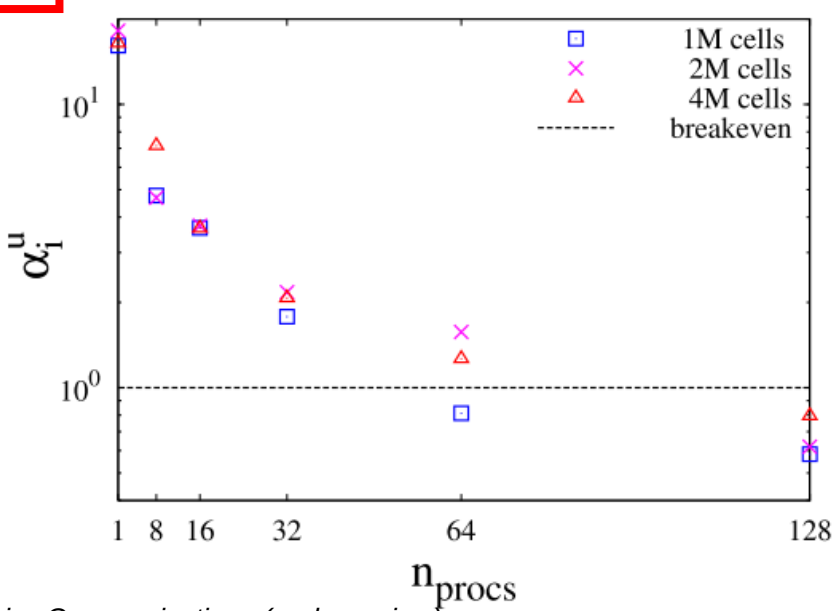
CPPPO – algorithms and performance

Advantage factor ->

$$\alpha_n^k = \frac{\tau_p|_k}{\tau_p|_n}$$

Is the ratio of the average time required by operations *k* and *n* when *p* processors are used





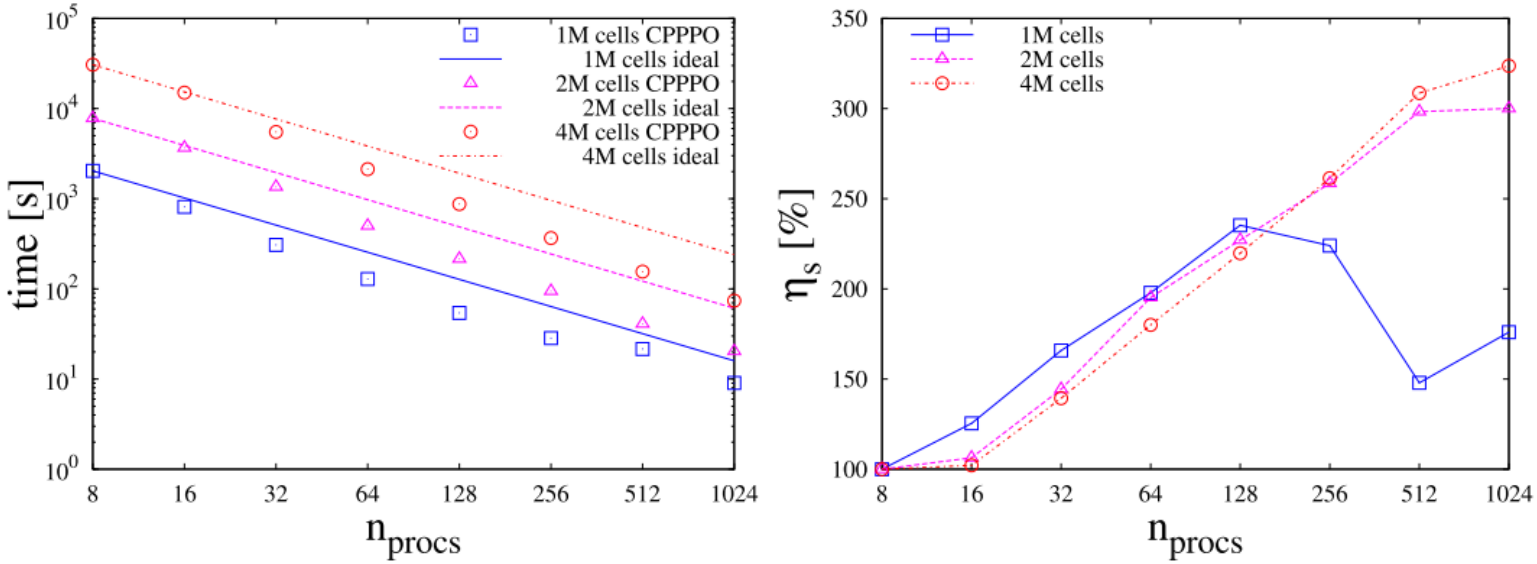
Municchi et al., 2016, Computer Physics Communications (under review)

convergent Vs divergent

unstructured Vs cartesian

With more than 64 cores, the unstructured selector performs better than the cartesian selector

CPPPO – algorithms and performance



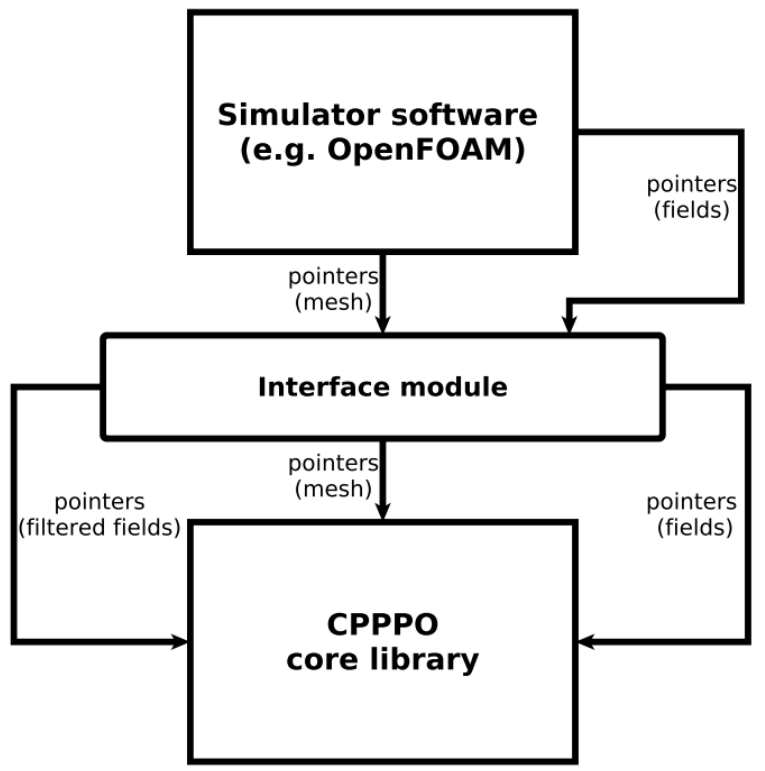
Municchi et al., 2016, Computer Physics Communications (under review)

Strong parallel efficiency (η_s) **well above 100%** on the VSC-3 (*Vienna Scientific Cluster* http://vsc.ac.at/)

The total time is **a small fraction** of the total computational time (**less than 2%** for flow and heat transfer in a particle bed)

- Coarse graining and multiscale approach
- CPPPO - overview
- CPPPO - algorithms and performance
- **CPPPO - coupling to simulator**
- Application to fluid-particle systems
- Conclusions

CPPPO – coupling to simulator



Municchi et al., 2016, Computer Physics Communications (under review)

The **CPPPO core library** is linked to the simulator software by mean of an **interface module**.

CPPPO only needs pointers to field and mesh quantities.

Additional heap memory is allocated in the interface class to create the filtered fields.

CPPPO – coupling to simulator

```
#include "c3po_OF_interface.H"
#include "argList.H"
#include "fvMesh.H"
```

Include the CPPPO-OpenFOAM interface in your solver

```
using namespace Foam;
using namespace C3PO_NS;
```

Use the **CPPPO namespace**

Create the c3poOFInterface object

Let CPPPO check your mesh

```
//Create C3PO
c3poOFInterface *myC3PO= new c3poOFInterface(mesh_,MPI_COMM_WORLD);
myC3PO->checkMyMesh();
```

Run

Delete

```
myC3PO->run();

MPI_Barrier(MPI_COMM_WORLD);
delete myC3PO;
Pout << "End of application." << endl;
```

CPPPO – coupling to simulator

CPPPO can be linked to existing OpenFOAM solver with a reduced amount of modifications.

The meshCheck() function can be called multiple times to account for **dynamic mesh**.

Simulation data are not modified by CPPPO. However, **filtered fields can be used in OpenFOAM for further calculations**.

- Coarse graining and multiscale approach
- CPPPO - overview
- CPPPO - algorithms and performance
- CPPPO - coupling to solver
- **Application to fluid-particle systems**
- Conclusions

Application to fluid-particle systems

We use **CFDEMCoupling®**[1] and **LIGGGHTS®** to simulate flow field and heat/mass transport in gas-particle suspensions by mean of **direct numerical simulation**.

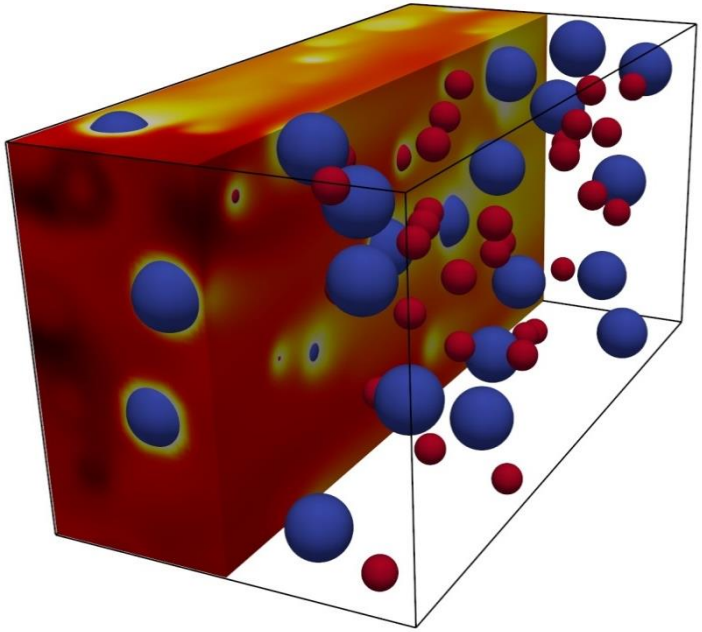
A novel **hybrid immersed-boundary/fictitious-domain method**[2] is adopted to impose Dirichlet boundary conditions at each particle surface.

The computational domain consists of a fully periodic box where an average flow field is induced by a pressure gradient.

We consider a **bi-disperse** particle population (i.e., two particle species with different diameters).

1) *Kloss, C., Goniva, C., Hager, A., Amberger, S., and Pirker, S. Models , algorithms and validation for opensource DEM and CFD-DEM. Progress in Computational Fluid Dynamics, 12:140–152, 2012.*
2)Municchi , F. Radl, S. Goniva, C. A Hybrid Fictitious Domain-Immersed Boundary Method for the Direct Simulation of Heat and Mass Transport in Fluid-Particle Systems, CFDEMCoupling LIGGGHTS user meeting 2016 (available on Researchgate.com)

Application to fluid-particle systems



Computational domain for a diluite bi-disperse particle population

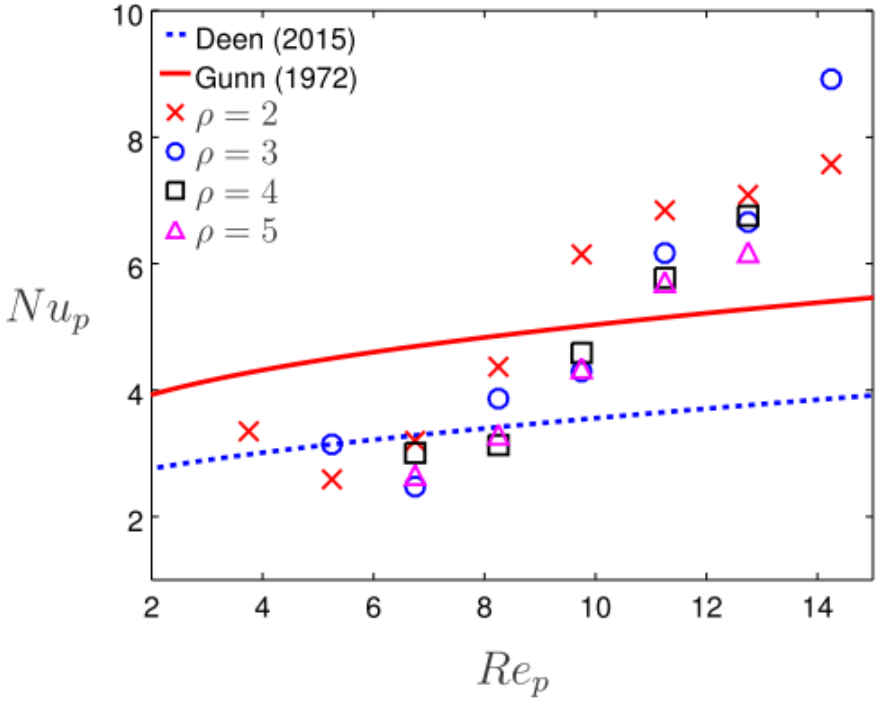
Particle-based **Nusselt number**:

$$Nu_p = \frac{Q_p^* Pr Re_p}{\theta_s^p - \theta_f^p}$$

Q_p^* : imensionless interphase heat
Pr : Prandtl number
Re_p: Particle-based Reynolds number
θ_s^p : Particle dimensionless surface temperature
θ_f^p : Particle dimensionless filtered temperature

Depend on the filtering Kernel!

Application to fluid-particle systems



Municchi et al., ICMF 2016

Nu_p shows a different dependence on Re_p with respect to mono-disperse correlations.

Also a slight dependence on the filter size is observed.

Further investigation is in progress...

Deen, N. G., Peters, E. a. J. F., Padding, J. T., and Kuipers, J.. M. Review of direct numerical simulation of fluid-particle mass, momentum and heat transfer in dense gas-solid flows. Chemical Engineering Science, 116:710–724, 2014.

Gunn, D. Transfer of heat or mass to particles in fixed and fluidised beds. International Journal of Heat and Mass Transfer, 21(4):467–476, 1978.

- Coarse graining and multiscale approach
- CPPPO - overview
- CPPPO - algorithms and performance
- CPPPO - coupling to solver
- Application to fluid-particle systems
- **Conclusions**

Conclusions

- CPPPO allows to develop closures for «coarse mesh» models from «resolved» simulations by mean of spatial filtering.
- CPPPO **significantly reduces the effort in post processing** of multiphase simulations, especially when running a large number of cases.
- CPPPO showed **outstanding parallel performance**.
- CPPPO is provided with a **full documentation**, **examples** and additional scripts.
- Input scripts in *Json* format are easy to **generate automatically** .

Conclusions

- CPPPO is an opensource library developed in the frame of the NanoSim project FP7 Grant agreement 604656
- CPPPO comes with **an interface class for OpenFOAM** and a standalone application to read CSV files
- CPPPO allows to easily **customize** the **filtering Kernel**
- Additional online documentation, screencasts and downloads are available at *http://www.tugraz.at/en/institute/ippt/downloads-software/*