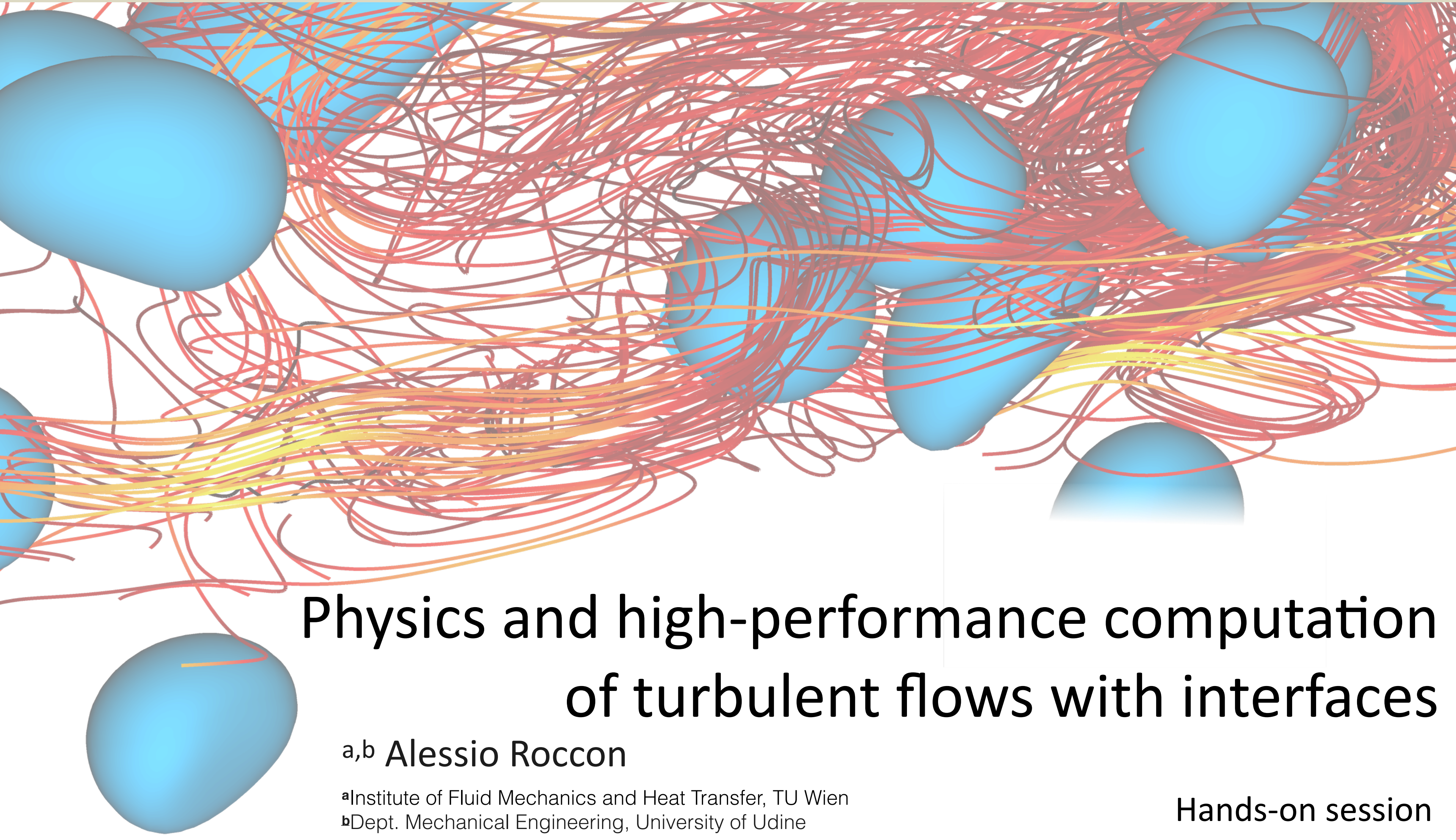
TECHNISCHE
UNIVERSITÄT
WIEN
Vienna | Austria



Physics and high-performance computation of turbulent flows with interfaces

[a,b] Alessio Roccon

[a]Institute of Fluid Mechanics and Heat Transfer, TU Wien
[b]Dept. Mechanical Engineering, University of Udine

Hands-on session

Training School ATS1
Vienna, 11-13 Feb. 2020

COMETE

Fil Rouge

1. Governing equations

2. Numerical method (flow)

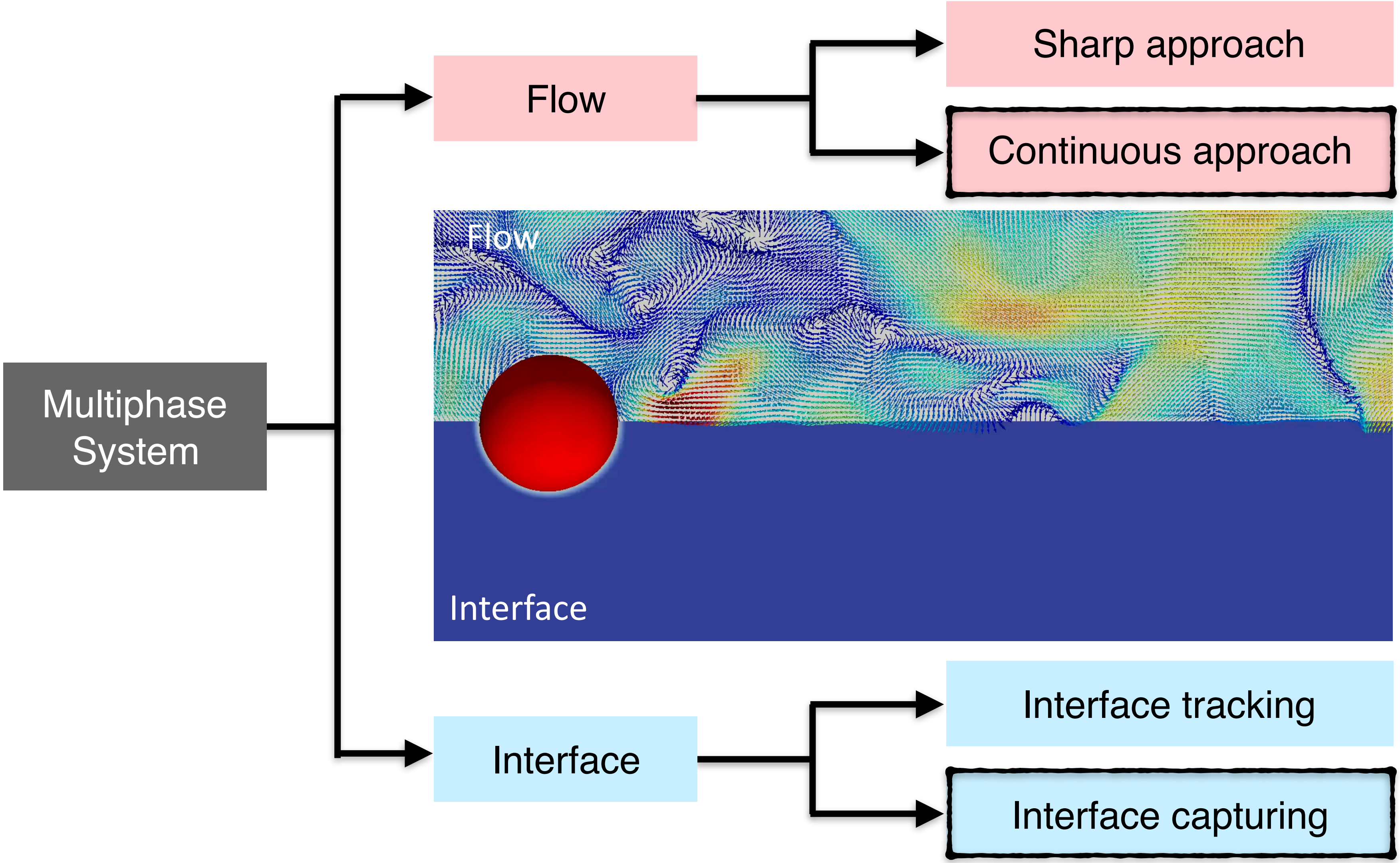
3. Solver parallelisation (flow)

4. Numerical method and parallelisation (phase-field)

5. Further challenges (parallelisation)

6. Hands-on

Description of a multiphase flow



Multiphase System

Flow

Sharp approach

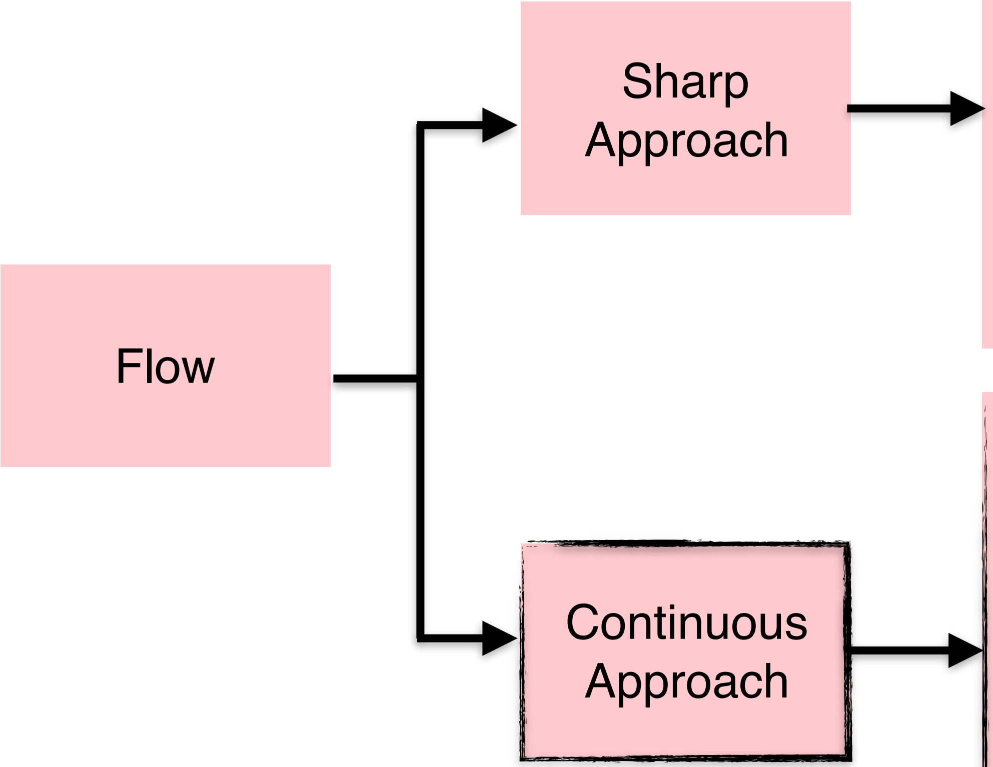
Continuous approach

Interface

Interface tracking

Interface capturing

Flow Solution

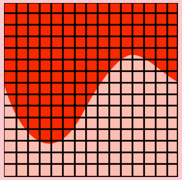


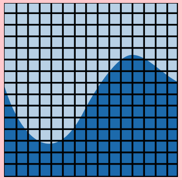
Ghost Fluid Method (GFM):

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re_\tau} \nabla^2 \mathbf{u}$$

$$p_A = p_B + \sigma \mathcal{K}$$

$$\mathbf{u}_A = \mathbf{u}_B + \Delta \mathbf{u}$$

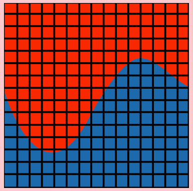




Continuous:

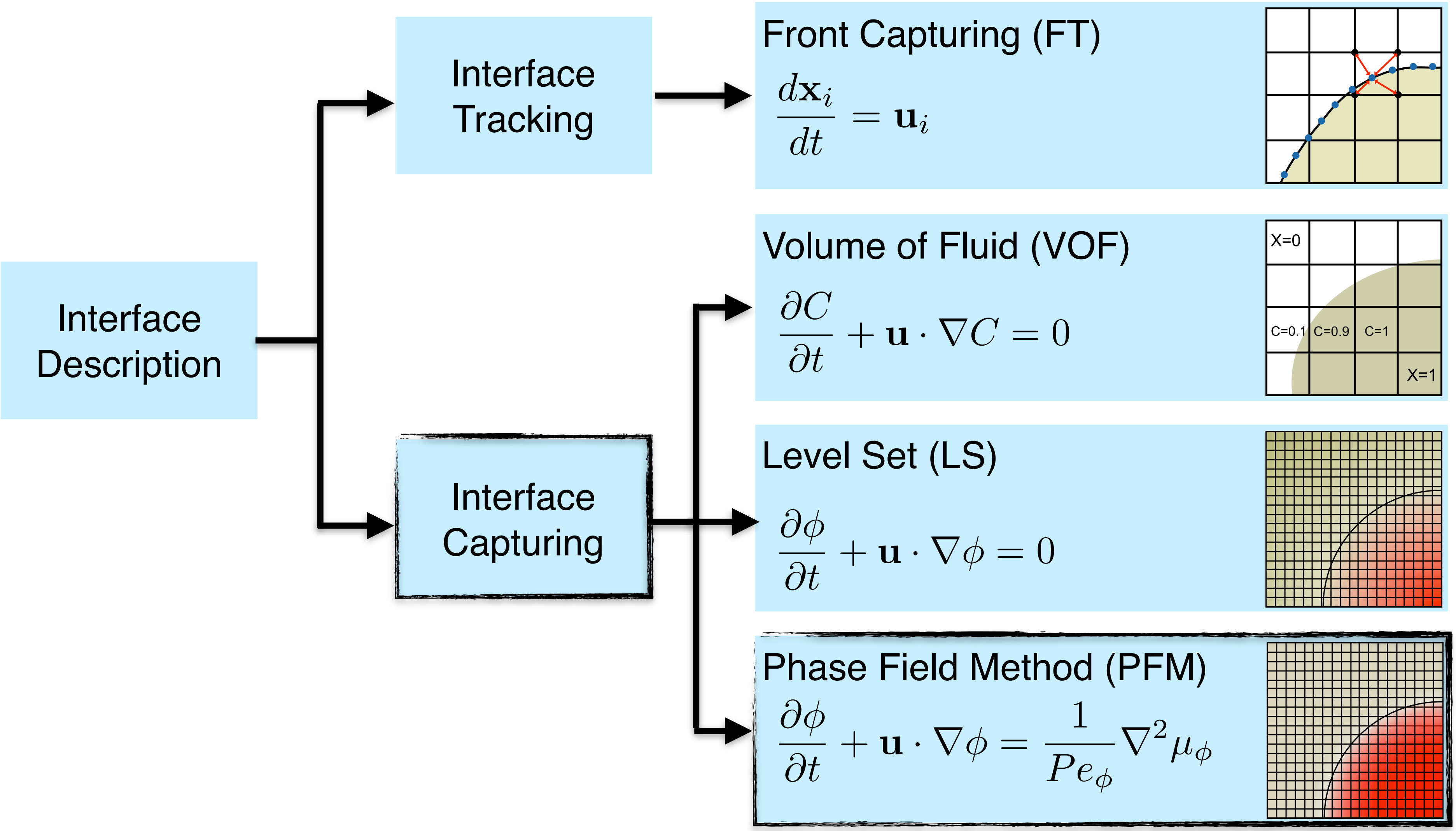
$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re_\tau} \nabla^2 \mathbf{u} + \frac{1}{We} \mathcal{K} \mathbf{n} \delta$$

Surface tension forces accounted using a delta function in the r.h.s.



Brackbill et al., A Continuum Method for Modeling Surface Tension, JCP (1992)
Scardovelli and Zaleski, Direct Numerical Simulation of free-surface and interfacial Flow, Ann. Rev. Fluid (1999)
Fedwik et al., A Non-oscillatory Eulerian approach to interfaces in multilateral flows (The Ghost Fluid Method), JSC (1999)
Lalanne et al., On the computation of viscous terms for incompressible two-phase flows with LS/GFM, JCP (2015)

Interface Description



Interface Description

Interface Tracking

Interface Capturing

Front Capturing (FT)

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{u}_i$$

Volume of Fluid (VOF)

$$\frac{\partial C}{\partial t} + \mathbf{u} \cdot \nabla C = 0$$

Level Set (LS)

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0$$

Phase Field Method (PFM)

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = \frac{1}{Pe_\phi} \nabla^2 \mu_\phi$$

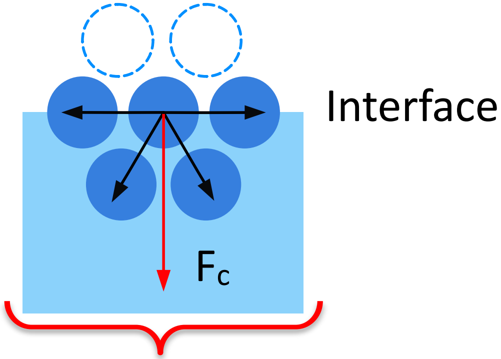
Unverdi and Tryggvason, A Front-Tracking Method for Viscous, Incompressible, Multi-fluid Flows, JCP (1991)
Hirt and Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, JCP (1981)
Sussmann et al., A level set approach for computing solutions to incompressible two-phase flow, JCP (1994)
Jacqmin, Calculation of Two-Phase Navier–Stokes Flows Using Phase-Field Modelling, JCP (1999)

Phase Field Method
Coupling with Navier-Stokes

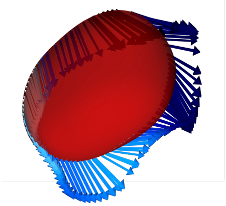
Assumptions:

- Matched densities
- Matched viscosities
- Constant surface tension

Surface tension forces:



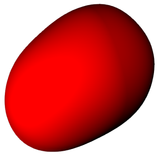
Flow:



$$\nabla \cdot \mathbf{u} = 0$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re_\tau}\nabla^2 \mathbf{u} + \frac{3}{\sqrt{8}}\frac{Ch}{We}\nabla \cdot \tau_c$$

Interface:



$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = \frac{1}{Pe_\phi}\nabla^2 \mu_\phi$$

Jacqmin, Calculation of Two-Phase Navier–Stokes Flows Using Phase-Field Modelling, JCP (1999)
Badalassi et al., Computation of multiphase systems with phase model, JCP (2003)

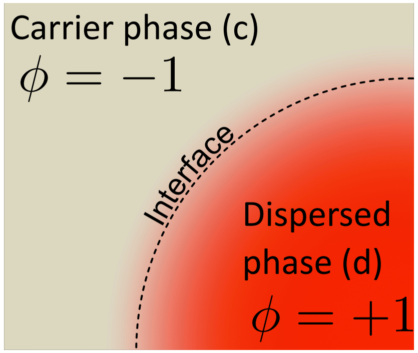
Phase Field Method
Treatment of different properties

<u>Assumptions:</u>

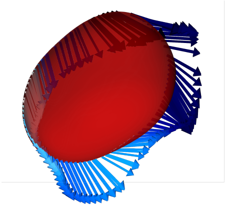
- Different densities
- Different viscosities
- Constant surface tension

Density ratio: $\gamma = \dfrac{\rho_d}{\rho_c}$

Viscosity ratio: $\lambda = \dfrac{\eta_d}{\eta_c}$



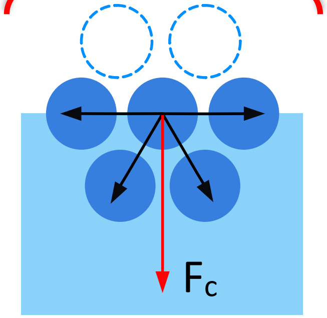
Flow:



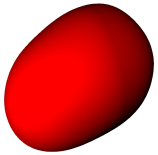
$$\nabla \cdot \mathbf{u} = 0$$

$$\underbrace{\rho(\phi)\left[\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right]}_{\rho(\phi) = 1 + \frac{\gamma - 1}{2}(\phi + 1)^{*}} = -\nabla p + \overbrace{\frac{1}{Re_\tau}\nabla\cdot\left[\eta(\phi)(\nabla\mathbf{u} + \nabla\mathbf{u}^T))\right]}^{\eta(\phi) = 1 + \frac{\lambda - 1}{2}(\phi + 1)^{*}} + \underbrace{\frac{3}{\sqrt{8}}\frac{Ch}{We}\nabla\cdot\tau_c}$$

Interface



Interface:



$$\frac{\partial\phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = \frac{1}{Pe_\phi}\nabla^2\mu_\phi$$

Jacqmin, Calculation of Two-Phase Navier–Stokes Flows Using Phase-Field Modelling, JCP (1999)
Badalassi et al., Computation of multiphase systems with phase model, JCP (2003)

*Dimensionless, normalised by the carrier phase density/viscosity.

Phase Field Method
Coupling with Navier-Stokes

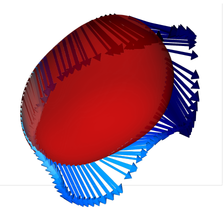
Assumptions:

- Matched densities
- Matched viscosities
- <u>Non-uniform surface tension</u>

Equation of state for surface tension:

$$f_\sigma(\psi) = \frac{\sigma(\psi)}{\sigma_0} = 1 + \beta_s \log(1-\psi)$$

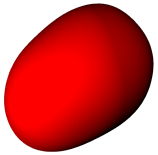
Flow:



$$\nabla \cdot \mathbf{u} = 0$$

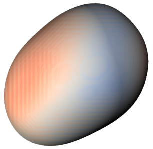
$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re_\tau} \nabla^2 \mathbf{u} + \frac{3}{\sqrt{8}} \frac{Ch}{We} \nabla \cdot [\tau_c f_\sigma(\psi)]$$

Interface:



$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = \frac{1}{Pe_\phi} \nabla^2 \mu_\phi \qquad \mu_\phi = \frac{\delta \mathcal{F}}{\delta \phi}$$

Surfactant:



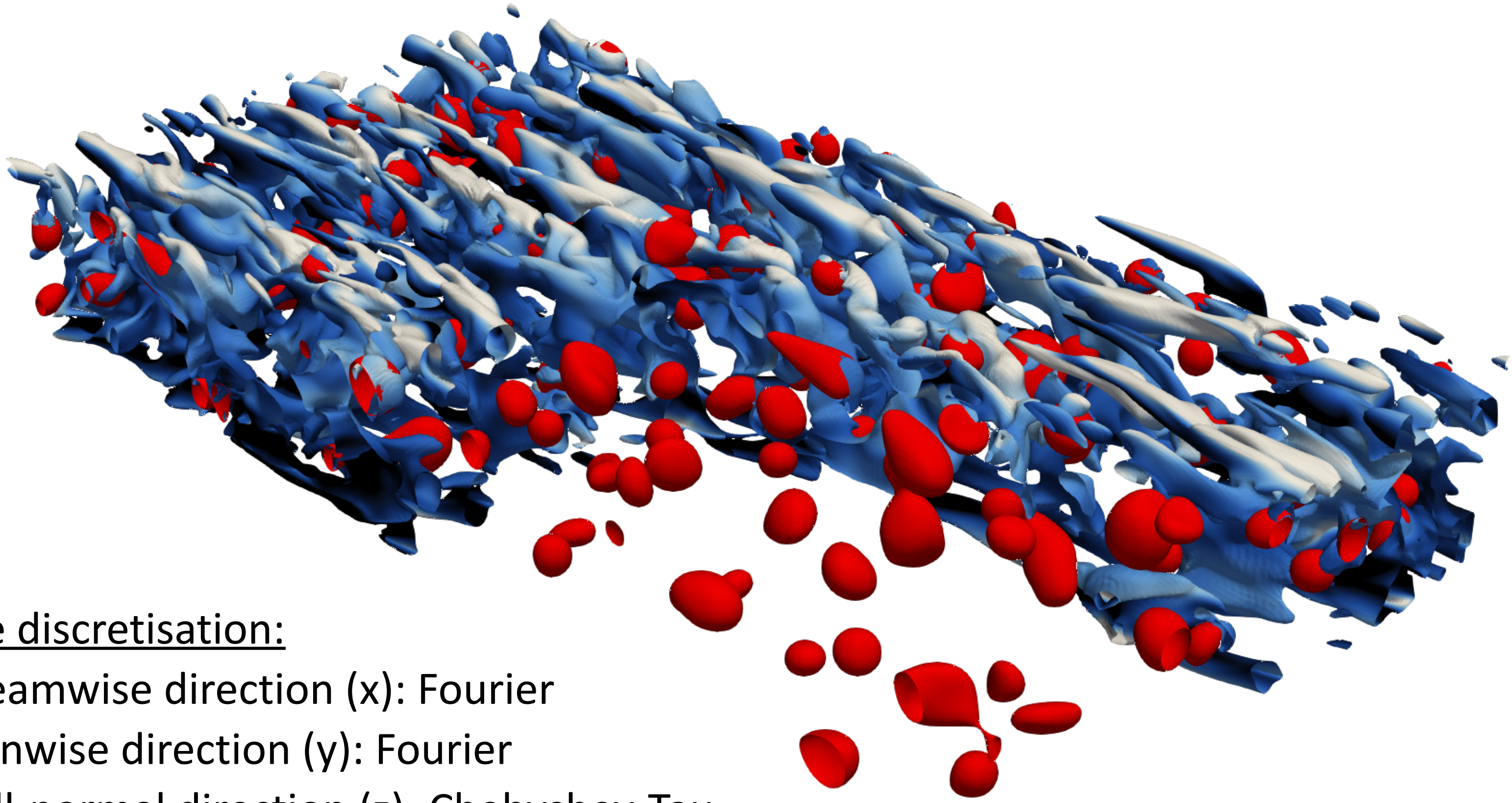
$$\frac{\partial \psi}{\partial t} + \mathbf{u} \cdot \nabla \psi = \frac{1}{Pe_\psi} \nabla \cdot (\mathcal{M}_\psi(\psi) \nabla \mu_\psi) \qquad \mu_\psi = \frac{\delta \mathcal{F}}{\delta \psi}$$

Fil Rouge

1. Governing equations
2. Numerical method (flow)
3. Solver parallelisation (flow)
4. Numerical method and parallelisation (phase-field)
5. Further challenges (parallelisation)
6. Hands-on

Numerical method
Turbulent channel flow

Turbulent channel flow:



Space discretisation:

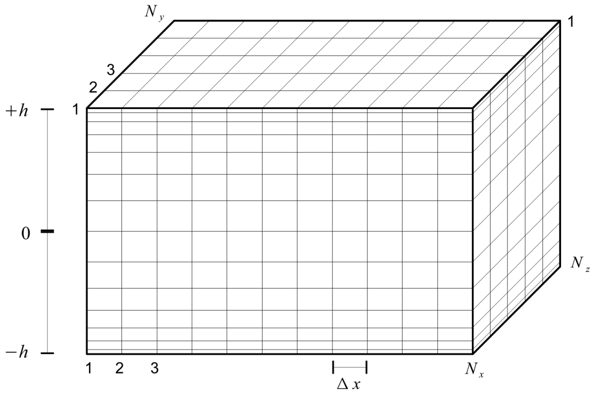
- Streamwise direction (x): Fourier
- Spanwise direction (y): Fourier
- Wall-normal direction (z): Chebyshev-Tau

Numerical method
Space discretisation

Physical space (collocation points):

Each variable in the physical space is defined on the following collocation points:

$$
\begin{aligned}
x_i &= (i-1)\frac{L_x}{N_x-1} & \qquad i &= 1,\dots,N_x \\
y_j &= (j-1)\frac{L_y}{N_y-1} & \qquad j &= 1,\dots,N_y \quad . \\
z_k &= \cos\left(\frac{k-1}{N_z-1}\pi\right) & \qquad k &= 1,\dots,N_z
\end{aligned}
$$



Spectral space (wavenumbers):

A generic variable can be represented in modal space as a function of Fourier wavenumbers and Chebyshev polynomials:

$$f(x,y,z,t) = \sum_{i=0}^{N_x/2} \sum_{j=-N_y/2+1}^{N_y/2} \sum_{k=0}^{N_z-1} \hat{f}(k_{x,i},k_{y,j},k,t)T_k(z)e^{\iota(k_{x,i}x+k_{y,j}y)}$$

Numerical method
Navier Stokes and continuity

Continuity and Navier-Stokes equations:

$$\nabla \cdot \mathbf{u} = 0$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{S} - \nabla p' + \frac{1}{Re_\tau} \nabla^2 \mathbf{u}$$

All the non-linear terms are collected here.

Navier-Stokes equations is solved using the so-called wall-normal velocity-vorticity formulation:

- Curl of NS (vorticity equation) and using $\nabla \times \nabla p' = 0$

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = \nabla \times \mathbf{S} + \frac{1}{Re_\tau} \nabla^2 \boldsymbol{\omega}$$

- Curl of (Curl of NS) and using $\nabla \times (\nabla \times \mathbf{A}) = \nabla(\nabla \cdot \mathbf{A}) - \nabla^2 \mathbf{A}$

$$\frac{\partial (\nabla^2 \mathbf{u})}{\partial t} = \nabla^2 \mathbf{S} - \nabla(\nabla \cdot \mathbf{S}) + \frac{1}{Re_\tau} \nabla^4 \mathbf{u}$$

Navier-Stokes
Temporal discretisation

Time discretisation: IMplicit-EXplicit scheme (IMEX):

- Crank-Nicolson for the linear terms (implicit)
- Adams-Bashforth for the non-linear terms (explicit)

Wall-normal vorticity:

$$\frac{\boldsymbol{\omega}^{n+1}-\boldsymbol{\omega}^{n}}{\Delta t}\cdot\mathbf{n}_z = \left(\underbrace{\frac{3\nabla\times\mathbf{S}^{n}-\nabla\times\mathbf{S}^{n-1}}{2}}_{\text{Adams-Bashforth}} + \underbrace{\frac{1}{Re_\tau}\,\frac{\nabla^2\boldsymbol{\omega}^{n+1}+\nabla^2\boldsymbol{\omega}^{n}}{2}}_{\text{Crank-Nicolson}}\right)\cdot\mathbf{n}_z \longleftarrow \text{Wall-normal unit vector}$$

Wall-normal velocity:

$$\begin{aligned}\frac{\nabla^2\mathbf{u}^{n+1}-\nabla^2\mathbf{u}^{n}}{\Delta t}\cdot\mathbf{n}_z = &\left[\overbrace{\frac{3[\nabla^2\mathbf{S}^{n}-\nabla(\nabla\cdot\mathbf{S}^{n})]-[\nabla^2\mathbf{S}^{n-1}-\nabla(\nabla\cdot\mathbf{S}^{n-1})]}{2}}^{\text{Adams-Bashforth}}+\right.\\ &\left.+\underbrace{\frac{1}{Re_\tau}\,\frac{\nabla^4\mathbf{u}^{n+1}+\nabla^4\mathbf{u}^{n}}{2}}_{\text{Crank-Nicolson}}\right]\cdot\mathbf{n}_z\end{aligned}$$

Navier-Stokes
Spatial discretisation

In the spectral space, the derivative can be easily computed as follows:

$$\frac{\partial f(x,y,z,t)}{\partial x} = \sum_{i=0}^{N_x/2} \sum_{j=-N_y/2+1}^{N_y/2} \sum_{k=0}^{N_z-1} \iota \underset{\big\uparrow}{k_{x,i}} \hat{f} T_k e^{\iota(k_{x,i}x+k_{y,j}y)}$$

1st order derivative along x

Streamwise wave-number

$$\frac{\partial f(x,y,z,t)}{\partial y} = \sum_{i=0}^{N_x/2} \sum_{j=-N_y/2+1}^{N_y/2} \sum_{k=0}^{N_z-1} \iota \underset{\big\uparrow}{k_{y,j}} \hat{f} T_k e^{\iota(k_{x,i}x+k_{y,j}y)}$$

1st order derivative along y

Spanwise wave-number

$$\frac{\partial f(x,y,z)}{\partial z} \longrightarrow \frac{\partial T_n(z)}{\partial z} = \frac{\partial T_{n-2}(z)}{\partial z} + 2nT_{n-1}$$

1st order derivative along z

Derivative of the Chebyshev polynomials

Navier-Stokes
Spatial and temporal discretisation

Wall-normal vorticity:

$$\frac{\hat{\omega}_z^{n+1}-\hat{\omega}_z^{n}}{\Delta t} = \frac{3}{2}\left(\iota k_{x,i}\hat{S}_y^n - \iota k_{y,j}\hat{S}_x^n\right) - \frac{1}{2}\left(\iota k_{x,i}\hat{S}_y^{n-1} - \iota k_{y,j}\hat{S}_x^{n-1}\right) + \\ + \frac{1}{2Re_\tau}\left(\frac{\partial^2\hat{\omega}_z^{n+1}}{\partial z^2} - k_{i,j}^2\hat{\omega}_z^{n+1} + \frac{\partial^2\hat{\omega}_z^{n}}{\partial z^2} - k_{i,j}^2\hat{\omega}_z^{n}\right)$$

Wall-normal velocity:

$$\frac{1}{\Delta t}\left(\frac{\partial^2\hat{w}^{n+1}}{\partial z^2} - k_{i,j}^2\hat{w}^{n+1} - \frac{\partial^2\hat{w}^{n}}{\partial z^2} + k_{i,j}^2\hat{w}^{n}\right) = \\ = \frac{3}{2}\left(-k_{i,j}^2\hat{S}_z^n - \iota k_{x,i}\frac{\partial\hat{S}_x^n}{\partial z} - \iota k_{y,j}\frac{\partial\hat{S}_y^n}{\partial z}\right) - \\ - \frac{1}{2}\left(-k_{i,j}^2\hat{S}_z^{n-1} - \iota k_{x,i}\frac{\partial\hat{S}_x^{n-1}}{\partial z} - \iota k_{y,j}\frac{\partial\hat{S}_y^{n-1}}{\partial z}\right) + \\ + \frac{1}{2Re_\tau}\left(k_{i,j}^4\hat{w}^{n+1} + \frac{\partial^4\hat{w}^{n+1}}{\partial z^4} - 2k_{i,j}^2\frac{\partial^2\hat{w}^{n+1}}{\partial z^2}\right) + \\ + \frac{1}{2Re_\tau}\left(k_{i,j}^4\hat{w}^{n} + \frac{\partial^4\hat{w}^{n}}{\partial z^4} - 2k_{i,j}^2\frac{\partial^2\hat{w}^{n}}{\partial z^2}\right)$$

Navier-Stokes
Spatial and temporal discretisation

After some algebraical manipulation and collecting the term at time n+1, we obtain a system of equations for each couple of spanwise-streamwise wavenumber:

Wall-normal vorticity:

$$\left(\frac{\partial^2}{\partial z^2} - \beta^2\right)\hat{\omega}_z^{n+1} = -\frac{\imath k_{x,i} H_y^n - \imath k_{y,j} H_x^n}{\gamma}$$

Historical terms

Coefficients:

$$\gamma = \frac{\Delta t}{2Re_\tau} \quad , \quad \beta^2 = \frac{1 + \gamma k_{i,j}^2}{\gamma}$$

Wall-normal velocity:

$$\left(\frac{\partial^2}{\partial z^2} - \beta^2\right)\left(\frac{\partial^2}{\partial z^2} - k_{i,j}^2\right)\hat{w}^{n+1} = \frac{H^n}{\gamma}$$

Splitting in two second order equations employing an auxiliary variable.

Each equation is solved using a Chebyshev-Tau method:

After the imposition of the boundary conditions (next slide), we obtain a tridiagonal matrix which can be solved using a Gauss-Jordan elimination procedure followed by a forward substitution step.

Discretisation
Boundary conditions

At the two walls of the turbulent channel flow (primitive variables):

$$\mathbf{u}(x,y,z=\pm 1)=\mathbf{0}$$

We also have:

$$\left.\begin{aligned}&\frac{\partial u}{\partial x}(x,y,z=\pm 1)=0\\ &\frac{\partial v}{\partial y}(x,y,z=\pm 1)=0\end{aligned}\right\}\quad \text{Using the continuity equation} \longrightarrow \frac{\partial w}{\partial z}(x,y,z=\pm 1)=0$$

$$\left.\begin{aligned}&\frac{\partial v}{\partial x}(x,y,z=\pm 1)=0\\ &\frac{\partial u}{\partial y}(x,y,z=\pm 1)=0\end{aligned}\right\}\quad \text{Using the vorticity definition} \longrightarrow \omega_z(x,y,z=\pm 1)=\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}=0$$

The full set of BCs is:

Wall-normal vorticity (2nd): $\omega_z(x,y,z=\pm 1)=0$

Wall-normal velocity (4th): $w(x,y,z=\pm 1)=0$ And $\dfrac{\partial w}{\partial z}(x,y,z=\pm 1)=0$

BCs on 2nd and 3rd derivative of w??

Discretisation
Influence matrix

Wall-normal vorticity transport equation: ✓

- 2nd order equation.
- BCs on the value at the two walls.

Wall-normal velocity transport equation:

- 4th order equation.
- BCs on the value and on the 1st derivative.
- BCs on the auxiliary variable?

Influence matrix method:

The solution for the velocity (w) and for the auxiliary variable are split in three contributions:

- A first contribution that does not necessarily verify the boundary conditions (w_1 and ϑ_1)
- Two contributions that verify the boundary conditions at one boundary (w_2, w_3 and ϑ_2, ϑ_3)

$$\begin{cases} \hat{w}^{n+1} = w_1 + Aw_2 + Bw_3 \\ \theta^{n+1} = \theta_1 + A\theta_2 + B\theta_3 \end{cases}$$

Imposing the BCs, the coefficients A and B can be calculated and thus the solution.

Fil Rouge

1. Governing equations
2. Numerical method (flow)
3. Solver parallelisation (flow)
4. Numerical method and parallelisation (phase-field)
5. Further challenges (parallelisation)
6. Hands-on

Parallelisation
Domain Decomposition

Code: FLOW36

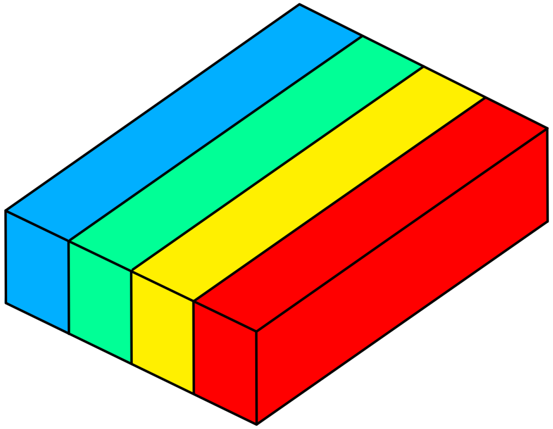
Programming language: Fortran 90

Parallelisation: Pure MPI.

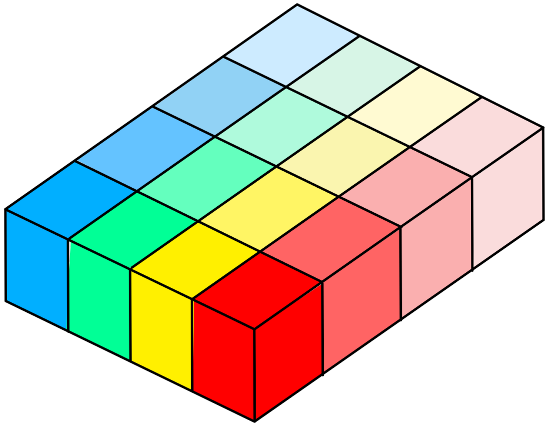
Main concept:

All the variables are Eulerian and defined on the same cartesian grid, we can use a 1D or a 2D domain decomposition.

1D Decomposition (slab)



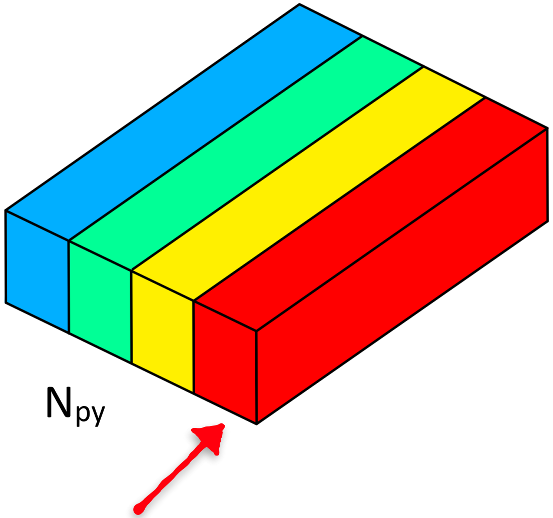
2D Decomposition (pencil)



Each colour represents a different task (MPI process) in physical space.

Parallelisation
1D vs 2D

1D Decomposition (slab)



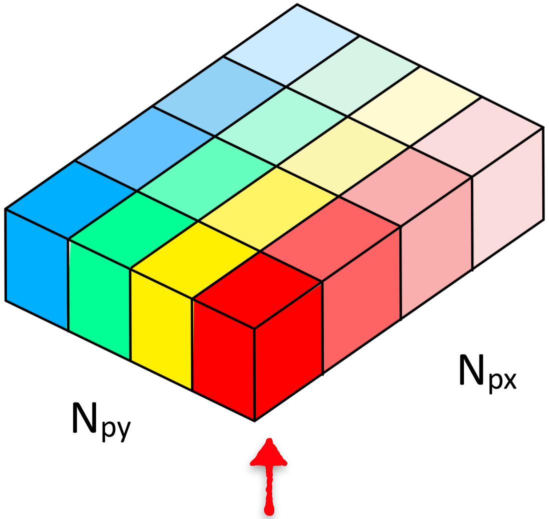
Each task (e.g. red) works on a fraction of the total number of nodes along the y direction:

1D domain decomposition:

Points x task = $(N_y / N_{py}) \times N_x \times N_z$

Maximum number of tasks: N_y

2D Decomposition (pencil)



Each task (e.g. red) works on a fraction of the total number of nodes along the x and y directions:

2D domain decomposition:

Points x task = $(N_x / N_{px}) \times (N_y / N_{py}) \times N_z$

Maximum number of tasks: $N_x \times N_y$

Is the 2D domain decomposition (larger number of processes) the best choice?

Parallelisation
Non-linear terms

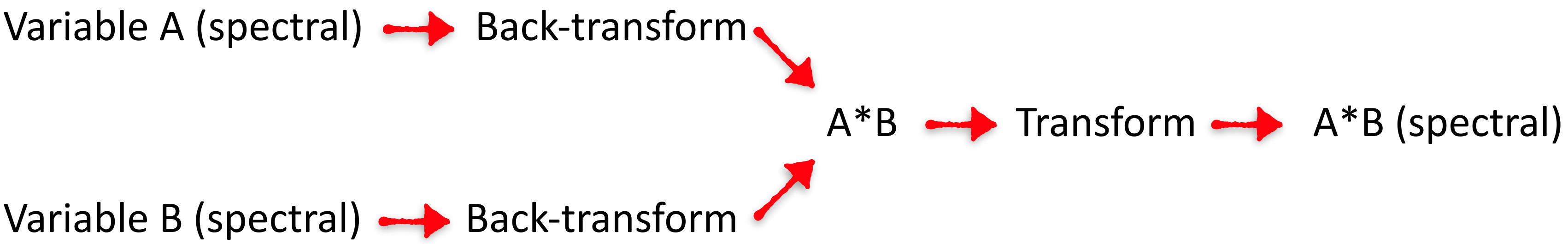
Do we need any MPI communications?

How are the non-linear terms computed?

Two options:

- Convolution, however it’s extremely expensive from a computational point of view (i.e. not feasible).
- Computation in the physical space, faster but require to back-transform each variable in the physical space and then back to the spectral space.

Computation of a product of two variables:



Variable A (spectral) → Back-transform

Variable B (spectral) → Back-transform

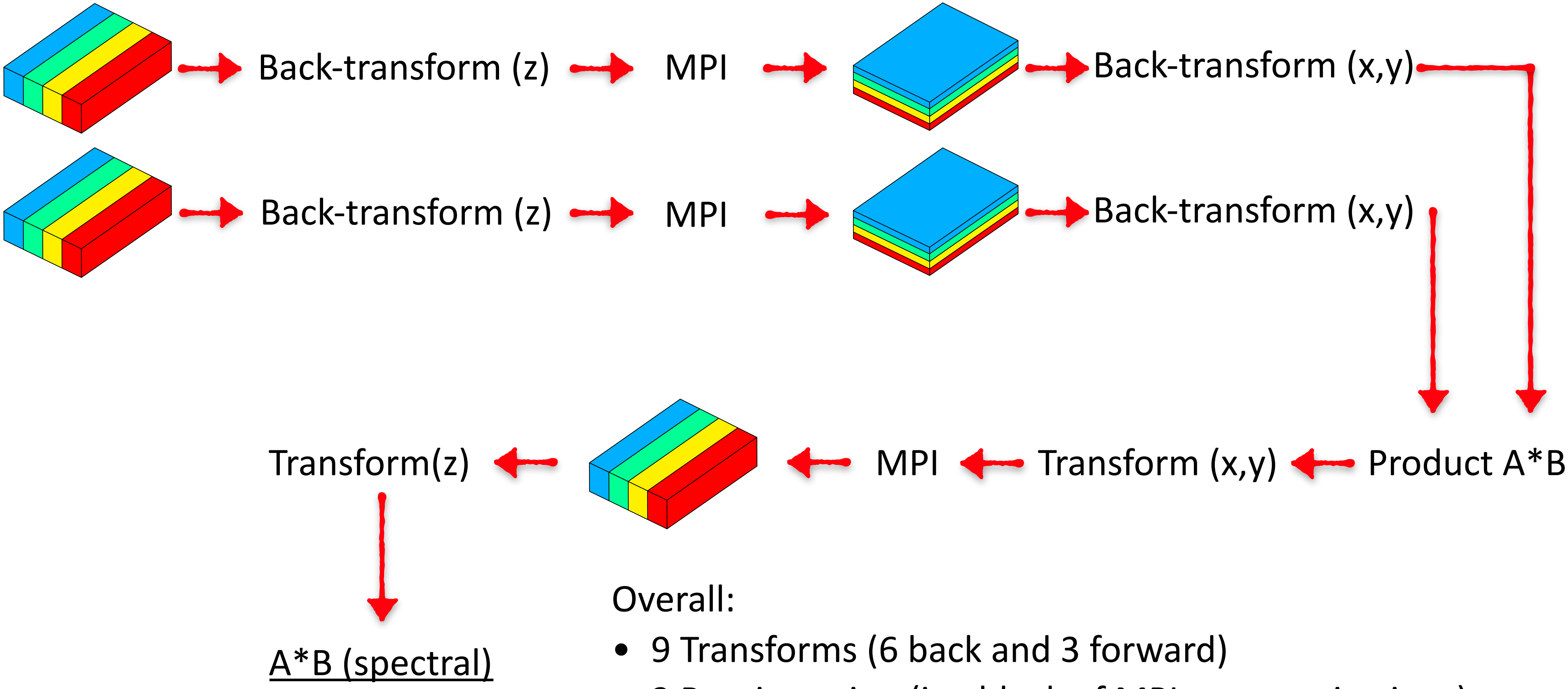
A*B → Transform → A*B (spectral)

Transform along a direction requires that the task handles all the points along that direction. To complete it (3 directions), we need to reorient the slab or pencil (i.e. MPI-communications)

Parallelisation
Non-linear terms - 1D

We consider the product A*B and a 1D domain decomposition:

A and B in spectral space:



Back-transform (z) → MPI → → Back-transform (x,y)

Back-transform (z) → MPI → → Back-transform (x,y)

Product A*B → Transform (x,y) → MPI → → Transform(z)

A*B (spectral)

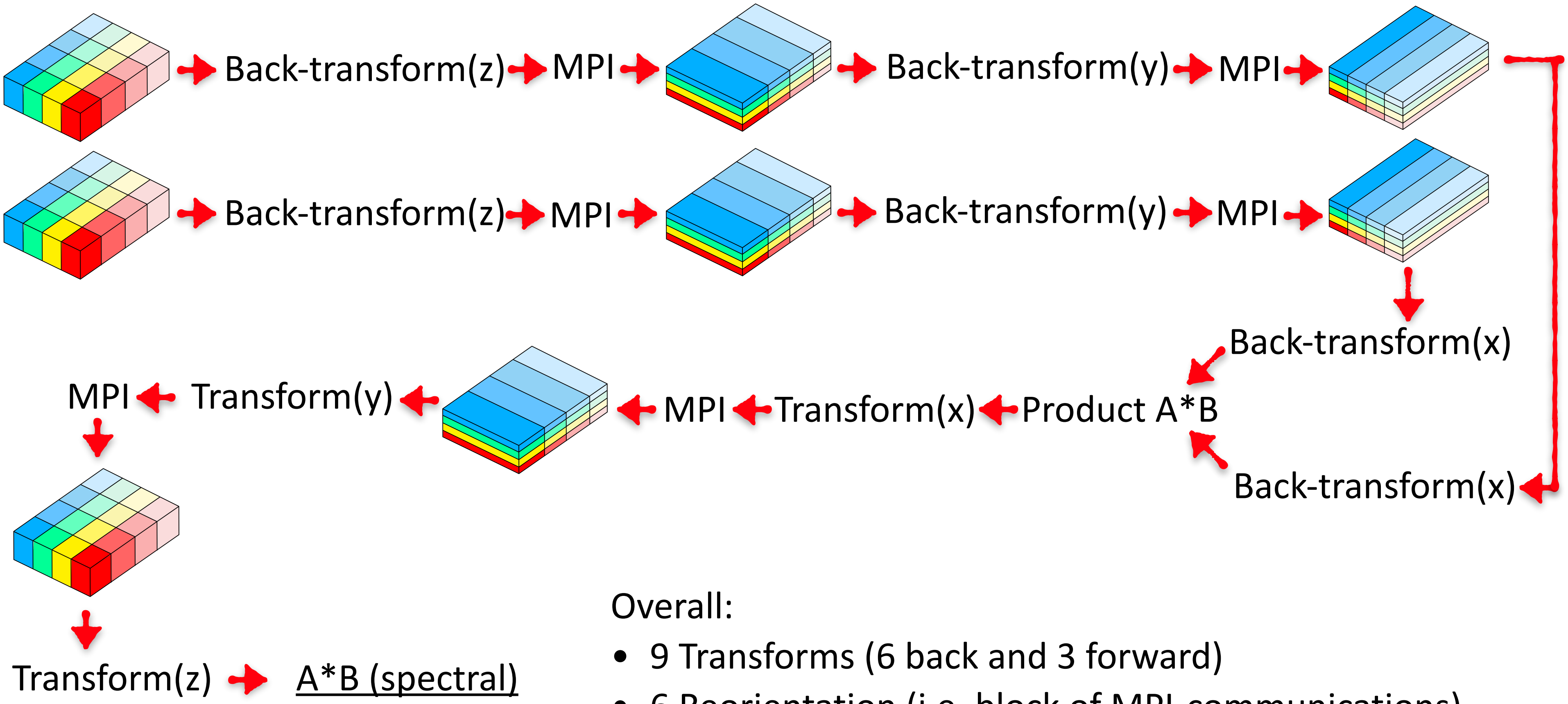
Overall:

- 9 Transforms (6 back and 3 forward)
- 3 Reorientation (i.e. block of MPI-communications)

Parallelisation
Non-linear terms - 2D

We consider the product A*B and a 2D domain decomposition:

A and B in spectral space:



Back-transform(z) → MPI → Back-transform(y) → MPI →

Back-transform(z) → MPI → Back-transform(y) → MPI →

Back-transform(x)

Back-transform(x)

Product A*B

Transform(x)

MPI

Transform(y)

MPI

Transform(z) → <u>A*B (spectral)</u>

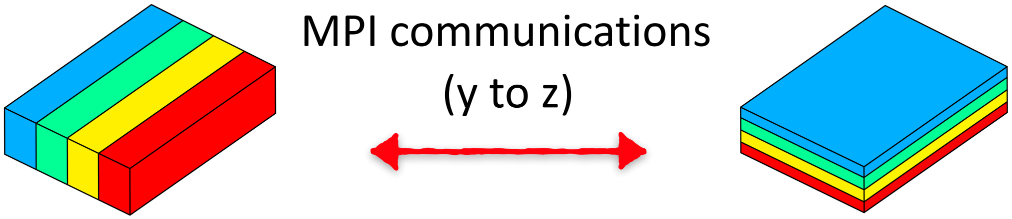
Overall:

- 9 Transforms (6 back and 3 forward)
- 6 Reorientation (i.e. block of MPI-communications)

Parallelisation
MPI implementation

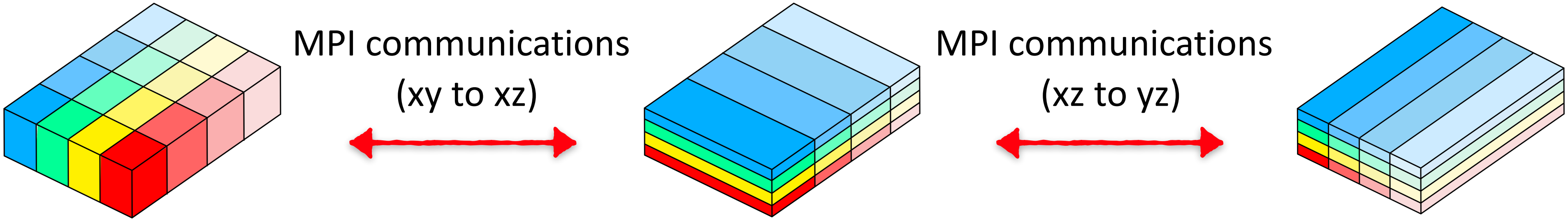
The reorientation are performed with MPI communications.

1D Decomposition:



MPI communications
(y to z)

2D Decomposition:



MPI communications
(xy to xz)

MPI communications
(xz to yz)

A Cartesian virtual topology is used to find the neighbouring tasks and to manage the communications among the different ranks.

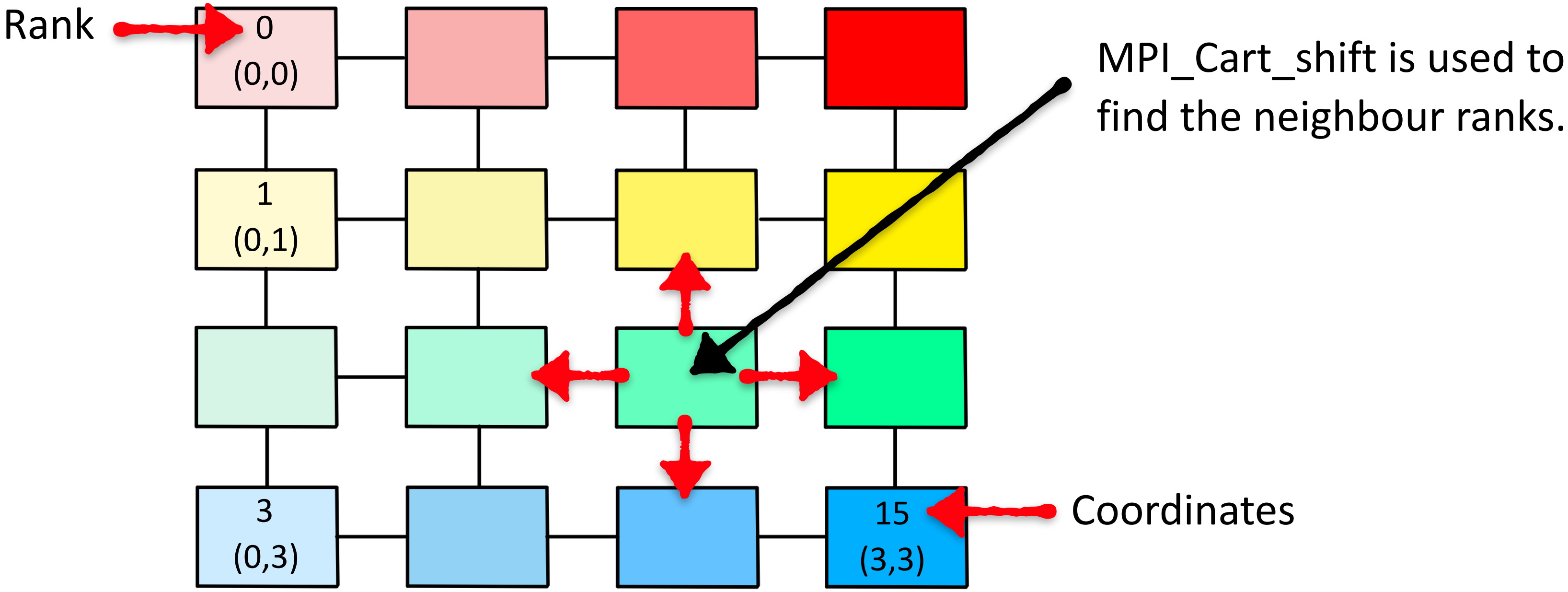
Parallelisation
MPI implementation

Virtual topology (e.g. spectral space):

- Convenient process naming
- Can allow MPI to optimise communications
- Mapping function
- Boundaries can be cyclic/periodic

Each process gets its own coordinates (MPI_Comm_rank and MPI_Cart_coords):

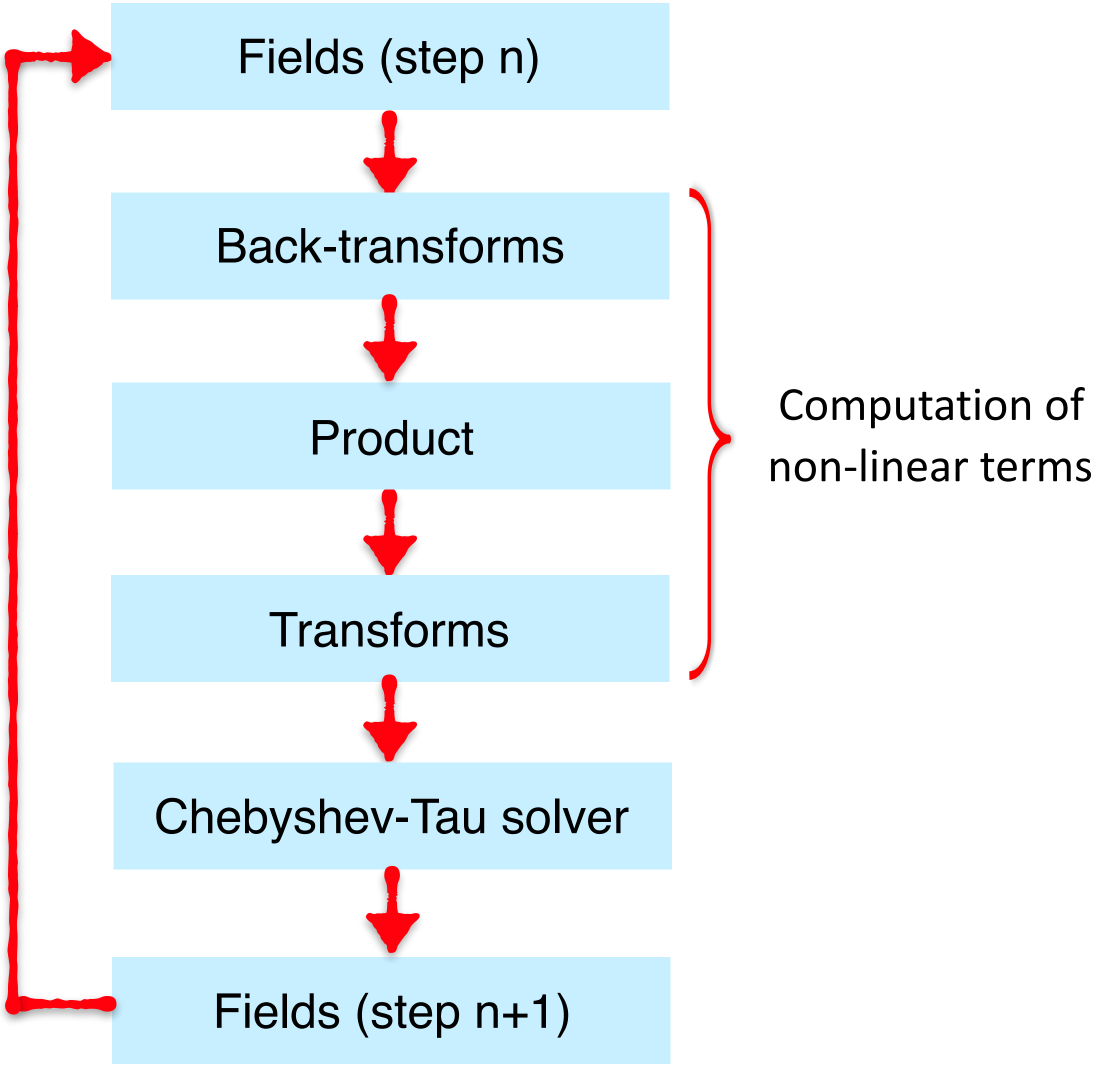
Virtual topology:



MPI_Cart_shift is used to find the neighbour ranks.

Parallelisation

General scheme of the solver



Parallelisation
Theoretical performance

The time employed for a single time step can be expressed as the sum of different contributions:

Time required for the operation that cannot be parallelised (negligible in this case)

Time employed for the Fourier and Chebyshev transforms

$$t = t_{SER} + t_{SOL} + t_{FFT} + t_{MPI}$$

Chebyshev-Tau solver

Time required for the MPI-communications
(data-transfer, etc.)

For a serial code, we have: $t_1 = t_{SOL} + t_{FFT}$ ← Function of the number of grid points

Using a 1D domain decomposition: $t_n = \dfrac{t_{SOL} + t_{FFT}}{N_p} + 3\, t_{MPI}$

Using a 2D domain decomposition: $t_n = \dfrac{t_{SOL} + t_{FFT}}{N_p} + 6\, t_{MPI}$

Parallelisation
Theoretical performance

The speed-up is the ratio:

$$\text{Speed-up} = \frac{t_1}{t_n}$$

Serial time

Parallel time with n tasks.

<u>Using a 1D domain decomposition:</u>

$$t_n = \frac{t_{SOL} + t_{FFT}}{\frac{t_{SOL}+t_{FFT}}{N_p} + 3\, t_{MPI}}$$

<u>Using a 2D domain decomposition:</u>

$$t_n = \frac{t_{SOL} + t_{FFT}}{\frac{t_{SOL}+t_{FFT}}{N_p} + 6\, t_{MPI}}$$

For t_{MPI}=0, the speed-up is ideal an equal to N_p

In order to compensate the larger number of MPI-communications for the 2D domain decomposition, the time required for the solver and for the FFT has to be larger (i.e. the 2D works better with larger grid resolutions).

Parallelisation
1D vs 2D Performance

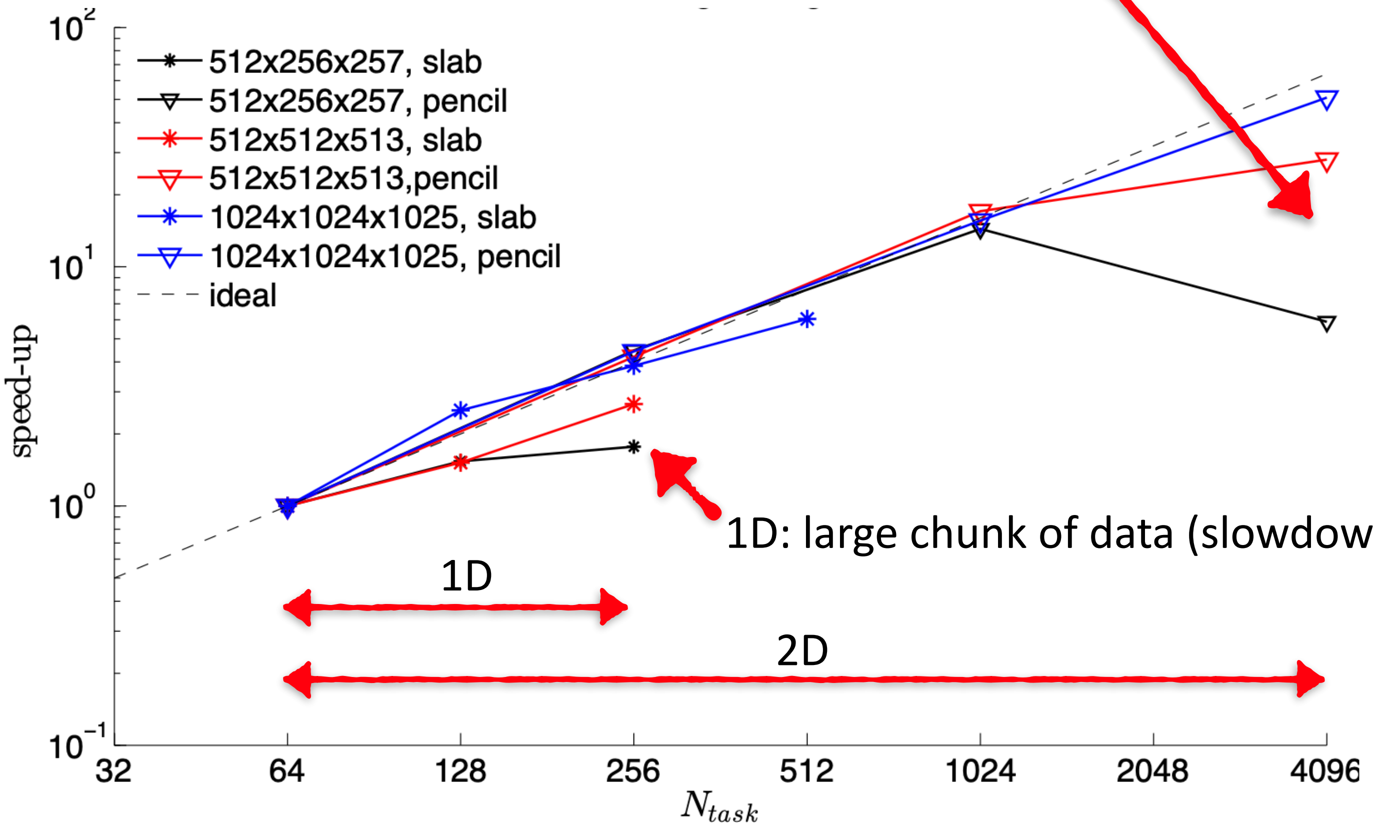
Strong scaling test performed on Marconi-A1 (Intel Broadwell):

Results are normalise by the 64 tasks result

Three different grid sizes (N_x x N_y x N_z):

- 512 x 256 x 257
- 512 x 512 x 513
- 1024 x 1024 x 1025

2D: t_{FFT} is too small (small chunk)



1D: large chunk of data (slowdown)

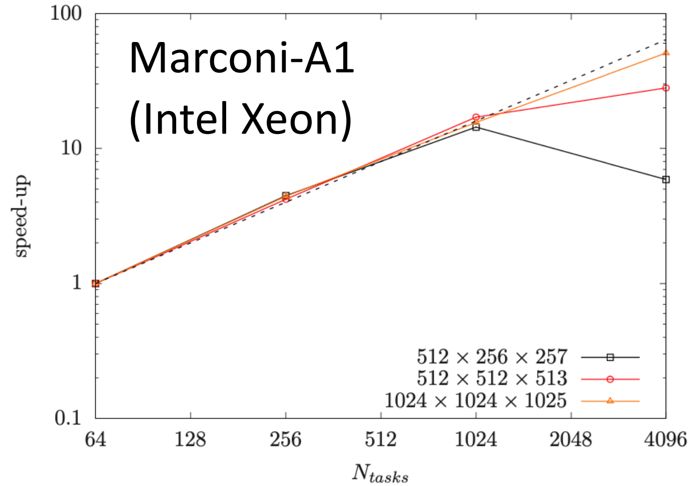
1D

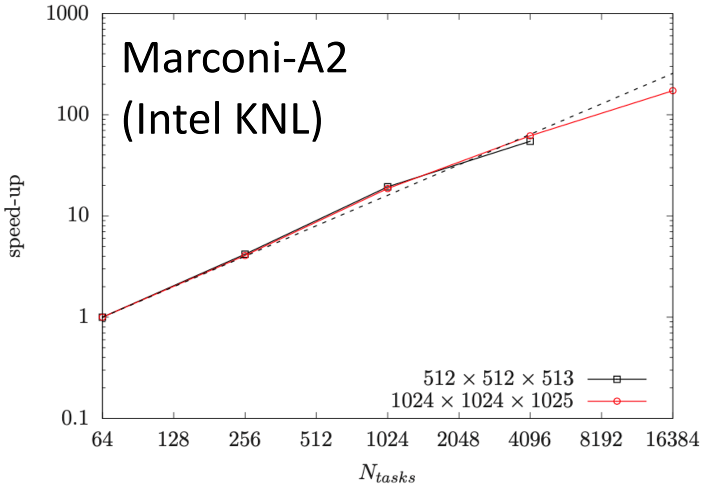
2D

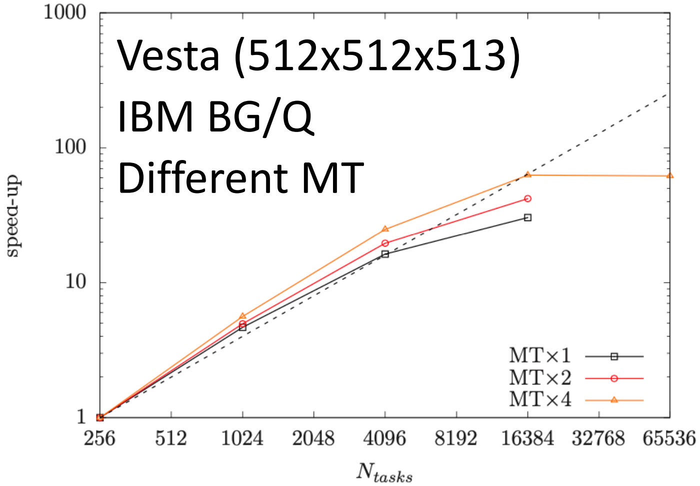
2D outperforms 1D (for the grids considered)

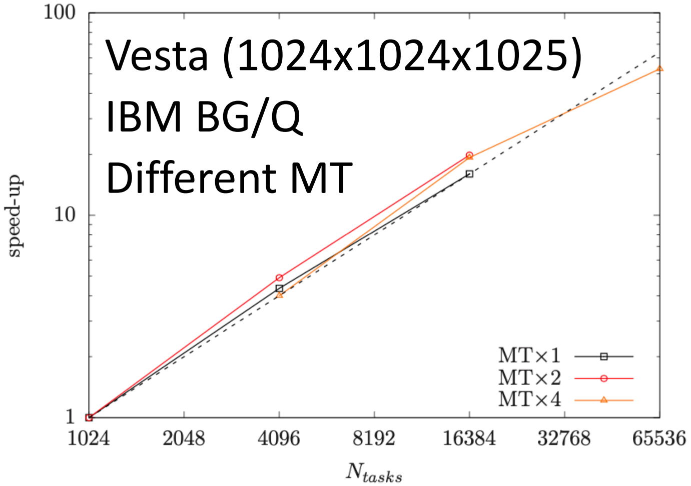
Parallelisation
1D vs 2D Performance

Strong scaling tests (2D domain decomposition) on other machines:


Marconi-A1 (Intel Xeon)

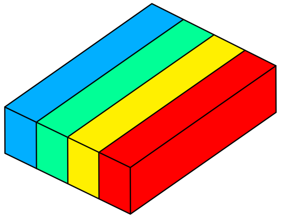

Marconi-A2 (Intel KNL)


Vesta (512x512x513)
IBM BG/Q
Different MT

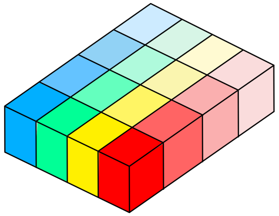

Vesta (1024x1024x1025)
IBM BG/Q
Different MT

Parallelisation
1D vs 2D Comparison

1D Decomposition (slab)

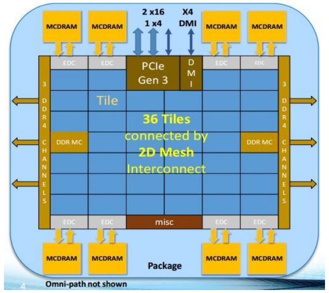


2D Decomposition (pencil)

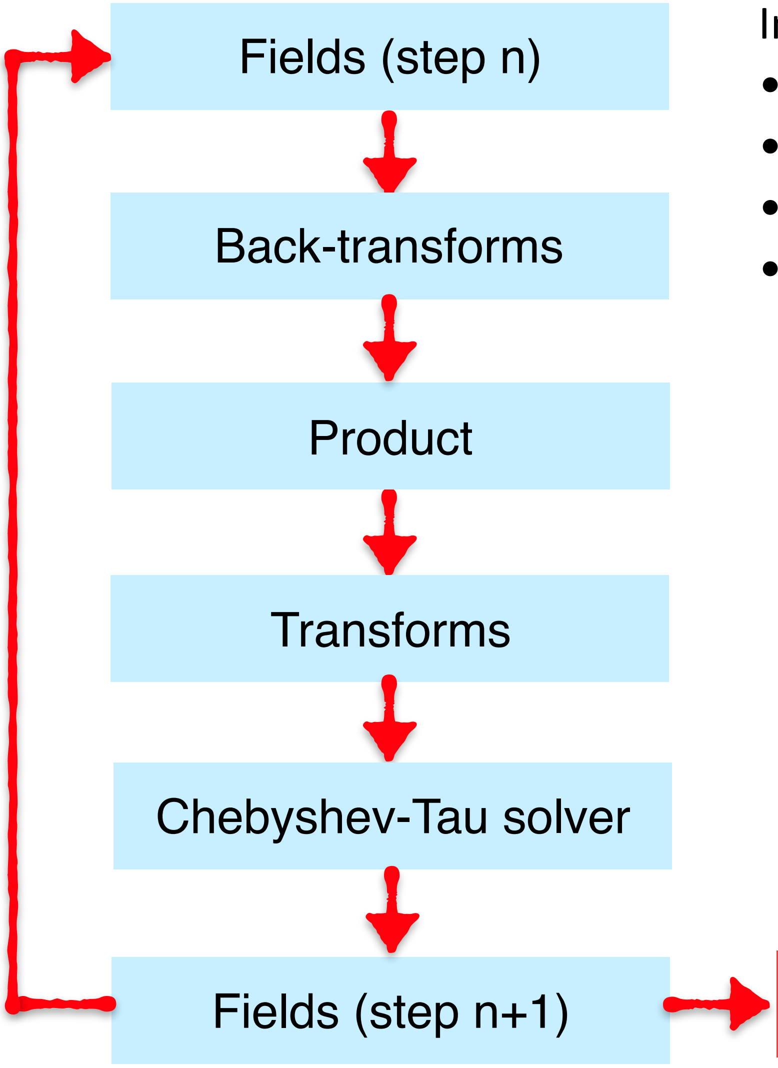


Taking into consideration also the reorientation and the MPI communications, we draw the following conclusions:

- Max number of ranks: min(N_y,N_z) for 1D and min(N_x x N_y, N_x x N_z, N_y x N_z) for 2D.
- For the grid commonly employed for turbulent flow, 2D outperforms 1D.
- The larger number of ranks that can be used (2D) lead to smaller wall clock time (total simulation time.
- In general, 2D has lower memory requirements, important for KNL and to optimise the use of the memory (integrated DCRAM).

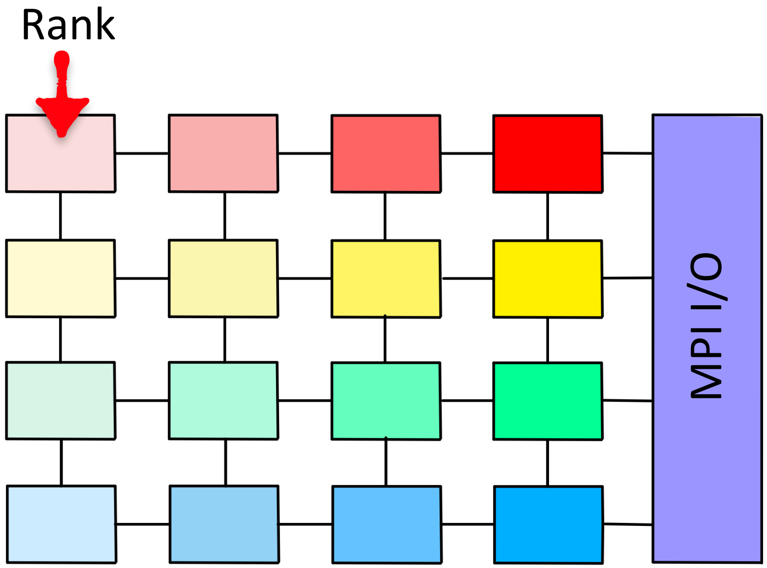


Parallelisation
Input and Output

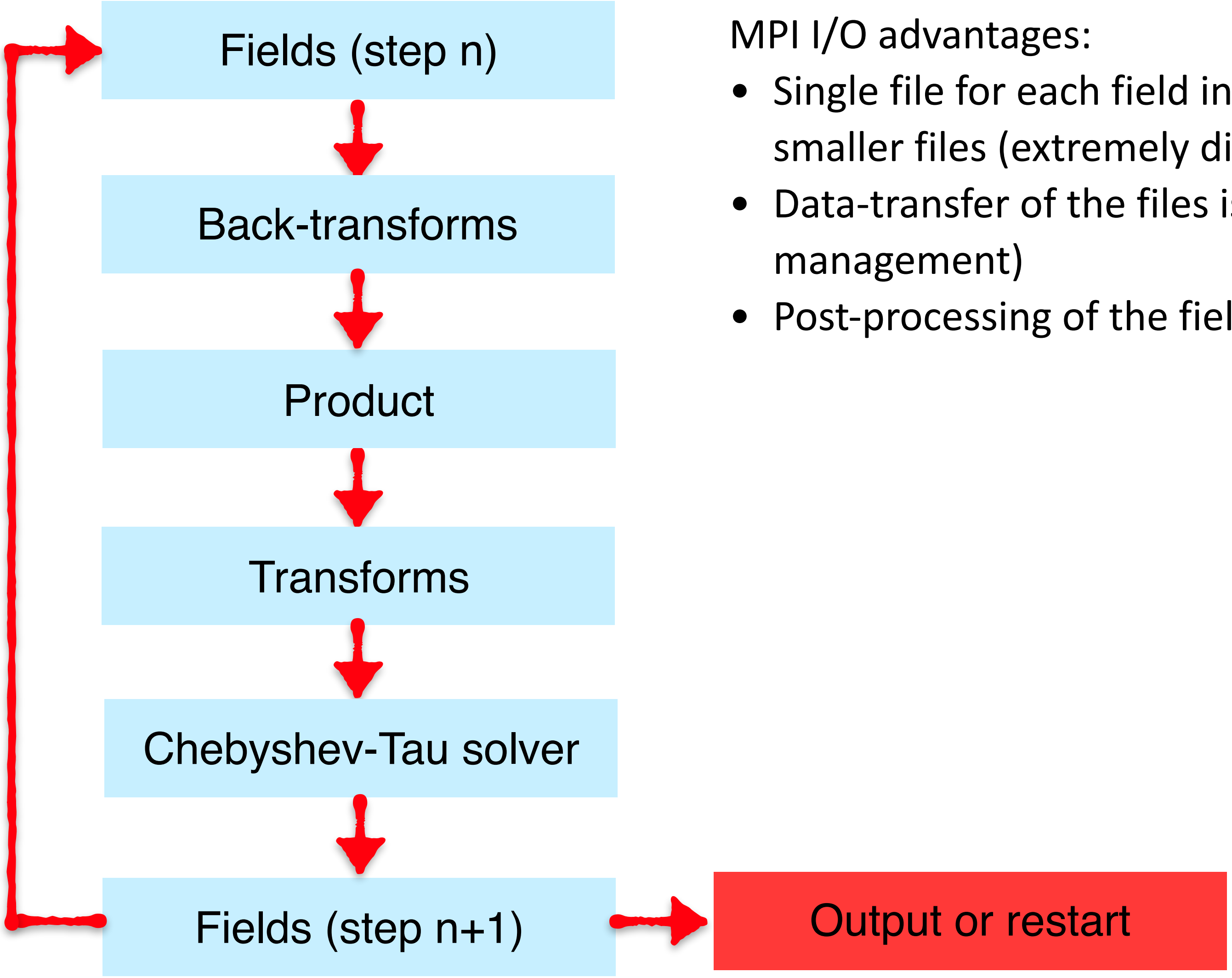


Input/output files:

- Input parameters (small files): Standard Fortran
- Fields (order of GB or more): MPI I/O
- Restart (order of GB or more): MPI I/O
- Fields written in binary format



Parallelisation
Input and Output



MPI I/O advantages:

- Single file for each field instead of thousand of smaller files (extremely difficult to manage)
- Data-transfer of the files is easier (and also the management)
- Post-processing of the fields is easier as well.

Parallelisation Conclusions

Conclusions on the flow solver and on its parallelisation:

- Parallelisation based on a pure MPI which leads to a fully parallel code (time required by the serial operations is negligible)
- Non-linear terms are computed in the physical space (i.e. pseudo-spectral); their computation requires MPI communications (reorientation of the slab/pencil).
- MPI virtual topology is used to better manage the MPI communications among the tasks.
- Different domain decomposition can be used: for the grid commonly employed, the 2D outperforms the 1D.
- Excellent strong scalability in different machines with different architecture (Xeon, Xeon Phi, IBM BG/Q).
- Input/output operations are managed with the MPI library.

Fil Rouge

1. Governing equations
2. Numerical method (flow)
3. Solver parallelisation (flow)
4. Numerical method and parallelisation (phase-field)
5. Further challenges (parallelisation)
6. Hands-on

Numerical method
Phase-field (1st order parameter)

Phase field transport equation:

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = \frac{1}{Pe_\phi} \nabla^2 (\phi^3 - \phi - Ch^2 \nabla^2 \phi)$$

In order to obtain a linear diffusive term, we can recast the equation as follows:

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = \frac{1}{Pe_\phi} \nabla^2 (\phi^3 - \phi - Ch^2 \nabla^2 \phi) + \frac{s}{Pe_\phi} \nabla^2 \phi - \frac{s}{Pe_\phi} \nabla^2 \phi$$

And collecting the non-linear terms we have:

$$\frac{\partial \phi}{\partial t} = \underset{\uparrow}{S_\phi} + \frac{s}{Pe_\phi} \nabla^2 \phi - \frac{Ch^2}{Pe_\phi} \nabla^4 \phi$$

All the non-linear terms are collected here.

$$S_\phi = -\mathbf{u} \cdot \nabla \phi + \frac{1}{Pe_\phi} [\nabla^2 \phi^3 - (1 + s) \nabla^2 \phi]$$

Phase-field

Temporal and spatial discretisation

Time discretisation: IMplicit-EXplicit scheme (IMEX):

- Euler for the linear terms (implicit)
- Adams-Bashforth for the non-linear terms (explicit)

Time discretised phase-field transport equation:

$$\frac{\phi^{n+1}-\phi^{n}}{\Delta t}=\underbrace{\frac{3S_{\phi}^{n}-S_{\phi}^{n-1}}{2}}_{\text{Adams-Bashforth}}+\underbrace{\frac{s}{Pe_{\phi}}\nabla^{2}\phi^{n+1}-\frac{Ch^{2}}{Pe_{\phi}}\nabla^{4}\phi^{n+1}}_{\text{Euler implicit}}$$

Introducing the spatial discretisation, we obtain:

$$\frac{\hat{\phi}^{n+1}-\hat{\phi}^{n}}{\Delta t}=\frac{3\hat{S}_{\phi}^{n}-\hat{S}_{\phi}^{n-1}}{2}+\frac{s}{Pe_{\phi}}\left(\frac{\partial^{2}\hat{\phi}^{n+1}}{\partial z^{2}}-k_{i,j}^{2}\hat{\phi}^{n+1}\right)-$$

$$-\frac{Ch^{2}}{Pe_{\phi}}\left(k_{i,j}^{4}\hat{\phi}^{n+1}+\frac{\partial^{4}\hat{\phi}^{n+1}}{\partial z^{4}}-2k_{i,j}^{2}\frac{\partial^{2}\hat{\phi}^{n+1}}{\partial z^{2}}\right)$$

Phase-field
Splitting coefficient

After some algebraical manipulation and collecting the term at time n+1, we obtain:

$$\underbrace{\left[\frac{1}{\gamma_\phi} - s\left(\frac{\partial^2}{\partial z^2} - k_{i,j}^2\right) + \left(\frac{\partial^2}{\partial z^2} - k_{i,j}^2\right)^2\right]}_{\left[\left(\frac{\partial^2}{\partial z^2} - k_{i,j}^2 - \lambda_1\right)\left(\frac{\partial^2}{\partial z^2} - k_{i,j}^2 - \lambda_2\right)\right]} \hat{\phi}^{n+1} = \frac{\hat{H}_\phi}{\gamma_\phi}$$

Historical term

Coefficients:

$$\gamma_\phi = \frac{Ch^2\Delta t}{Pe_\phi}$$

In order to split the equation (two 2nd order equations), we need to have the equation in this form

The characteristic equation is:

$$\gamma_\phi \lambda^2 - s\gamma_\phi \lambda + 1 = 0$$

With solutions:

$$\lambda_{1/2} = -\frac{s}{2} \pm \frac{\sqrt{s^2\gamma_\phi^2 - 4\gamma_\phi}}{2\gamma_\phi}$$

To have two real solutions and coincident (numerical stability), we choose:

$$s \geq \sqrt{\frac{4}{\gamma}} = \sqrt{\frac{4Pe}{\Delta t Ch^2}}$$

Phase-field
Boundary conditions

Finally, we obtain:

$$\left(\frac{\partial^2}{\partial z^2} - \beta_\phi^2\right)\left(\frac{\partial^2}{\partial z^2} - \beta_\phi^2\right)\hat{\phi}^{n+1} = \frac{H_\phi^n}{\gamma_\phi} \qquad \beta_\phi^2 = \frac{s}{2Ch^2} + k_{i,j}^2$$

Which can be split as follows:

$$\left(\frac{\partial^2}{\partial z^2} - \delta^2\right)\hat{\theta}_\phi = \frac{\hat{H}_\phi}{\gamma_\phi}$$

$$\left(\frac{\partial^2}{\partial z^2} - \delta^2\right)\hat{\phi}^{n+1} = \hat{\theta}_\phi$$

← Auxiliary variable

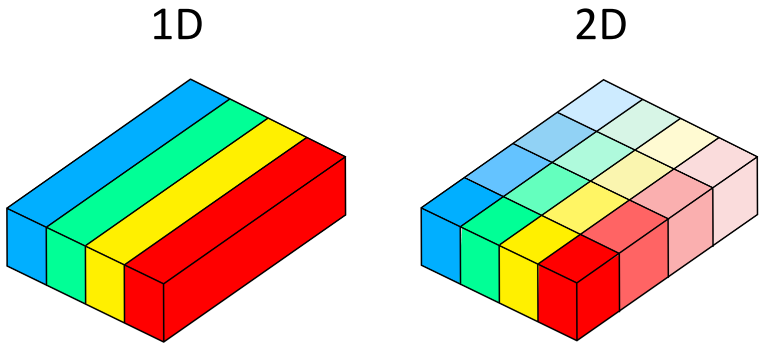
Both equations are solved using a Chebyshev-Tau method, with the following boundary conditions (no-flux BC):

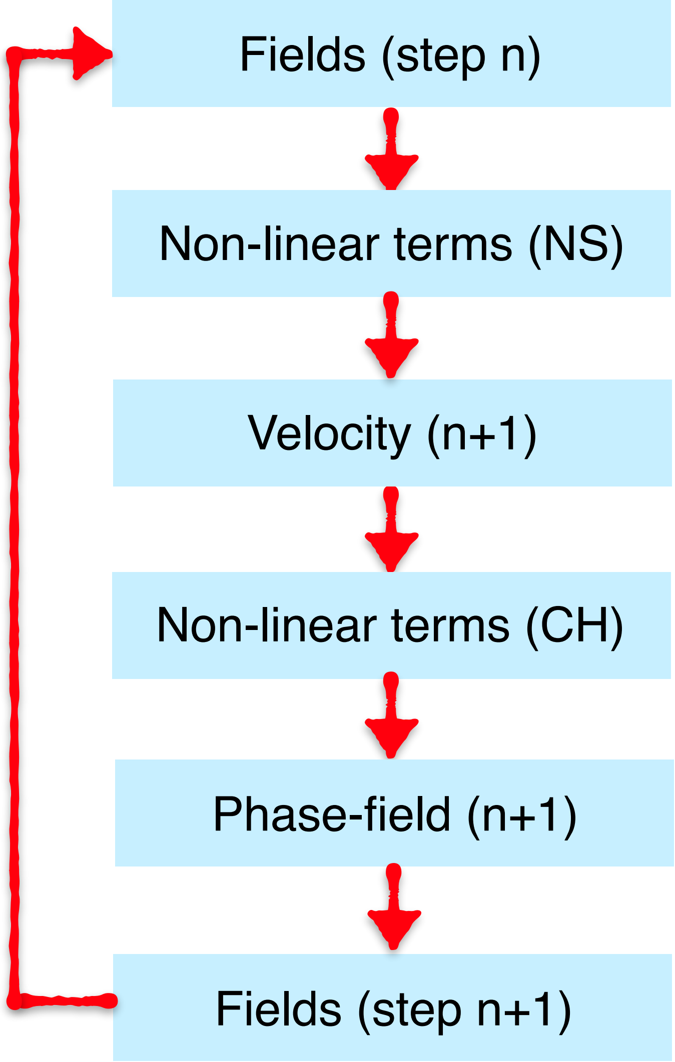
$$\frac{\partial \phi}{\partial z}(x, y, \pm 1) = 0 \qquad \frac{\partial^3 \phi}{\partial z^3}(x, y, \pm 1) = \frac{\partial \theta_\phi}{\partial z}(x, y, \pm 1) = 0$$

We have BCs for both the variables, no need to use the influence matrix method.

Phase-field Parallelisation

Both velocity fields and phase-field are defined on the same cartesian grid (same number of collocation points), we can use the same domain decomposition:





Numerical method
Surfactant (2nd order parameter)

Surfactant concentration transport equation:

$$\frac{\partial \psi}{\partial t} + \mathbf{u} \cdot \nabla \psi = \frac{1}{Pe_\psi} \nabla \cdot [\psi (1-\psi) \nabla \mu_\psi]$$

And the chemical potential is:

$$\mu_\psi = Pi \log \left(\frac{\psi}{1-\psi} \right) - \frac{(1-\phi^2)^2}{2} + \frac{1}{2E_x} \phi^2$$

And collecting the non-linear terms we have:

$$\frac{\partial \psi}{\partial t} = S_\psi + \frac{Pi}{Pe_\psi} \nabla^2 \psi$$

All the non-linear terms are collected here.

$$S_\psi = -\mathbf{u} \cdot \nabla \psi + \frac{1}{Pe_\psi} \nabla \cdot \left[\psi (1-\psi) \nabla \left(\frac{(1-\phi^2)^2}{2} + \frac{\phi^2}{2E_x} \right) \right]$$

Surfactant

Temporal and spatial discretisation

Time discretisation: IMplicit-EXplicit scheme (IMEX):

- Euler for the linear terms (implicit)
- Adams-Bashforth for the non-linear terms (explicit)

Time discretised surfactant concentration transport equation:

$$\frac{\psi^{n+1}-\psi^{n}}{\Delta t} = \underbrace{\frac{3S_\psi^{n} - S_\psi^{n-1}}{2}}_{\text{Adams-Bashforth}} + \underbrace{\frac{Pi}{Pe_\psi}\nabla^2\psi^{n+1}}_{\text{Euler}}$$

Introducing the spatial discretisation, we obtain:

$$\frac{\hat{\psi}^{n+1}-\hat{\psi}^{n}}{\Delta t} = \frac{3\hat{S}_\psi^{n} - \hat{S}_\psi^{n-1}}{2} + \frac{Pi}{Pe_\psi}\left(\frac{\partial^2\hat{\psi}^{n+1}}{\partial z^2} - k_{i,j}^2\hat{\psi}^{n+1}\right)$$

Surfactant
Boundary conditions

After some algebraical manipulation and collecting the term at time n+1, we obtain:

$$\left(\frac{\partial^2}{\partial z^2} - \beta_\psi^2\right)\hat{\psi}^{n+1} = -\frac{H_\psi^n}{\gamma_\psi}$$

← Historical term

Coefficients:

$$\gamma_\psi = \frac{Pi\Delta t}{Pe_\psi} \qquad \beta_\psi^2 = \frac{1+\gamma_\psi k_{i,j}^2}{\gamma_\psi}$$

The equation (2nd order) is directly solved using a Chebyshev-Tau method, with the following boundary conditions (no-flux BC):

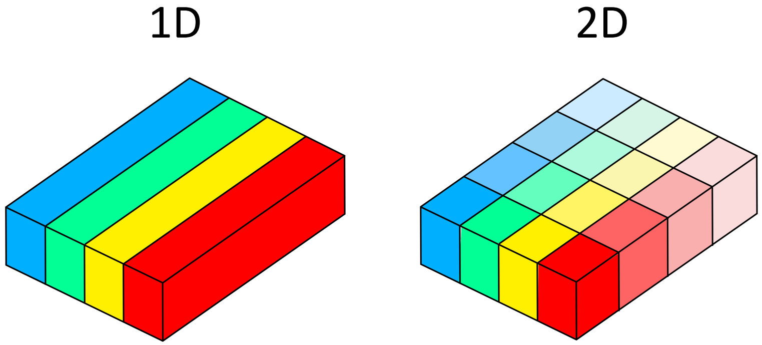
$$\frac{\partial\psi}{\partial z}(x,y,\pm1) = 0$$

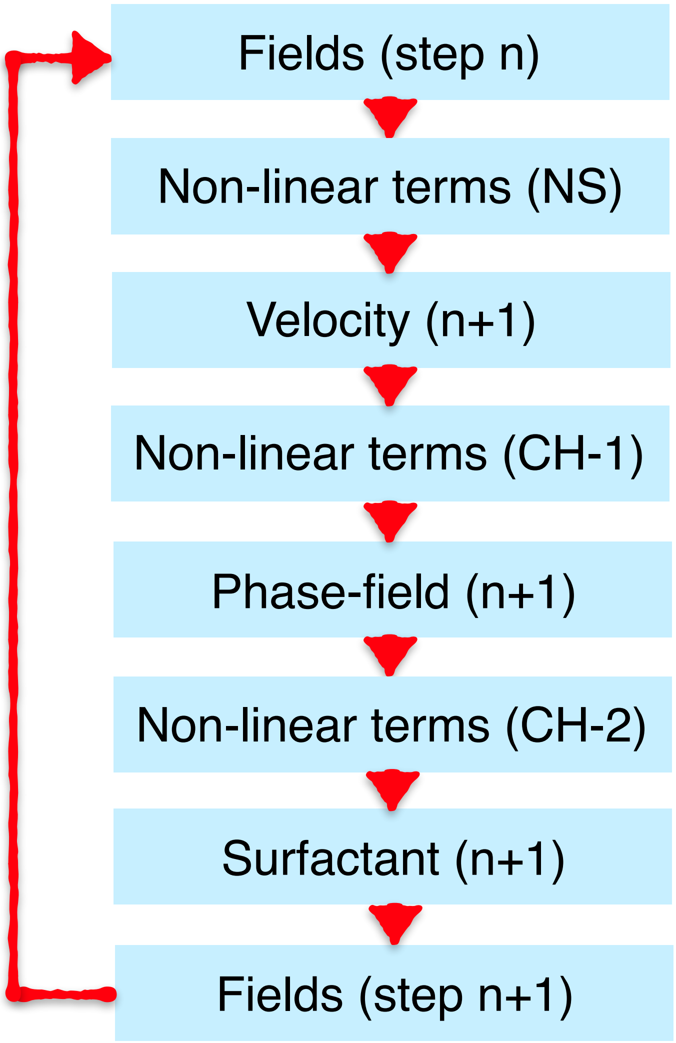
Also for the surfactant, we have BCs on the derivative, no influence matrix method needed. The set of BCs used lead to the conservation of the two order parameters (total mass and surfactant mass) over time:

$$\frac{\partial}{\partial t}\int_\Omega \phi d\Omega = 0 \qquad \frac{\partial}{\partial t}\int_\Omega \psi d\Omega = 0$$

Phase-field Parallelisation

Also the surfactant is defined on the same cartesian grid (same number of collocation points), we can use the same domain decomposition:




Fields (step n)

Non-linear terms (NS)

Velocity (n+1)

Non-linear terms (CH-1)

Phase-field (n+1)

Non-linear terms (CH-2)

Surfactant (n+1)

Fields (step n+1)

Fil Rouge

1. Governing equations
2. Numerical method (flow)
3. Solver parallelisation (flow)
4. Numerical method and parallelisation (phase-field)
5. Further challenges (parallelisation)
6. Hands-on

Further challenges
Particle Tracking

In many situations, it is of interest to track the motion of a certain number of point-wise particles (tracers or inertial) released inside the flow.
The motion of the particles can be described as follows:

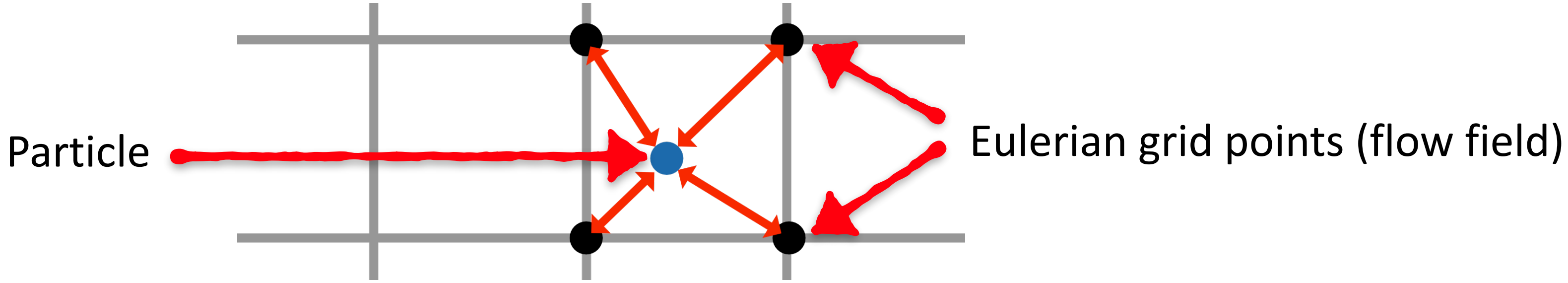
Particle position

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{u}_i \qquad\qquad \frac{d\mathbf{u}_i}{dt} = \underbrace{\mathbf{F}_i^d + \mathbf{F}_i^b + \mathbf{F}_i^*}$$

Particle velocity

Forces acting on the particle (e.g. drag, buoyancy, etc.)

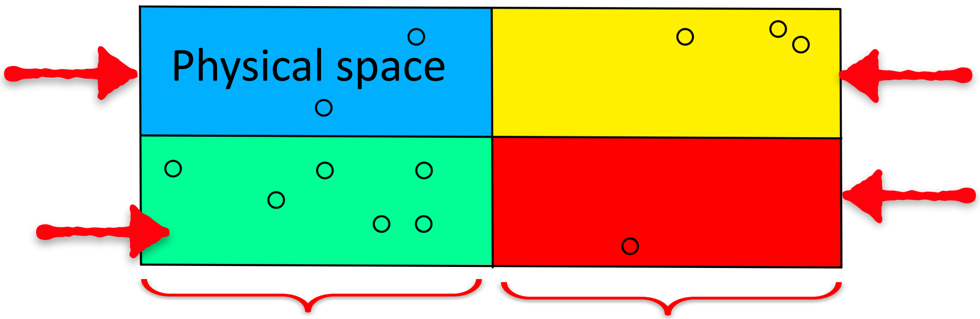
We have a set of ode to solve for each particle (feasible); however, if we consider drag forces, the fluid velocity has to be computed at the particle position. Thus, communications between the particles and the flow field are required:



Further challenges
Particle Tracking

We have two options:

- Each rank tracks the particles in its portion of domain, load balance?



Domain of rank #0

Domain of rank #1

Domain of rank #2

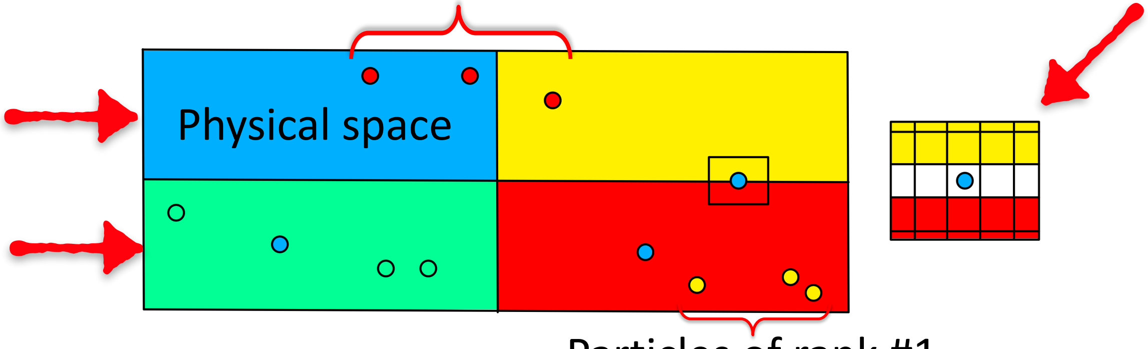
Domain of rank #3

Particles of rank #1 (green) Particles of rank #3 (green)

- Each rank tracks a pre-defined set of particles.

Particles of rank #3

Possible further complication



Domain of rank #0

Domain of rank #2

Particles of rank #1

We need MPI communications to obtain the velocity at the particle position.

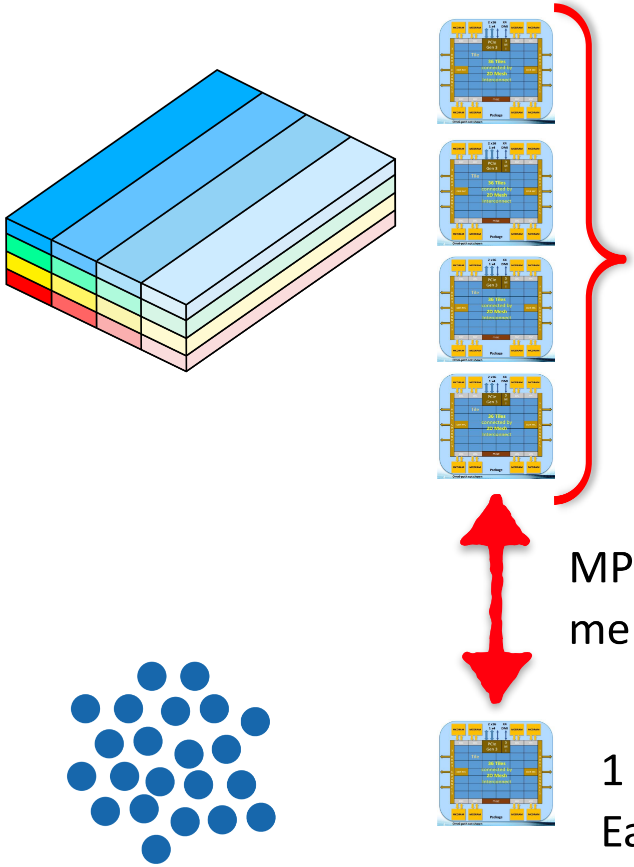
Particles are represented fictitiously enlarged!

Further challenges
Particle Tracking

Possible solution:

Splitting of the two solvers (Eulerian and tracking), we can use MPI shared for the particles.

We consider a run on a number of node, N_d, every node has a number of tasks, N_t.



N_d -1 nodes are used to solve the Eulerian fields (e.g. flow, phase-field, temperature, etc.): the solver is parallelised on a number N_d x N_t of total tasks.

MPI communications are used to transfer the entire fields to the memory of the “particles” node.

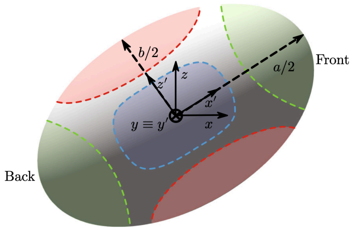
1 node track all the particles, the tracking is execute on N_t.
Each task can access the entire field (MPI shared).

Fil Rouge

1. Governing equations
2. Numerical method (flow)
3. Solver parallelisation (flow)
4. Numerical method and parallelisation (phase-field)
5. Further challenges (parallelisation)
6. Hands-on session

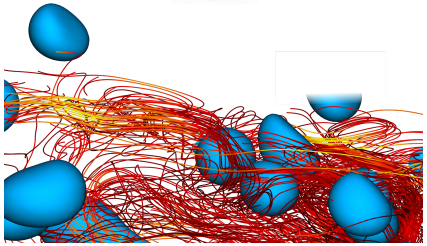
Hands-on

- Computation of the deformation of a droplet.



- Deformation of two interacting droplets.

- Data visualisation using paraview.



Software required: Matlab, Paraview.

Link to download the material: shorturl.at/mIJX4

Hands-on Session #1

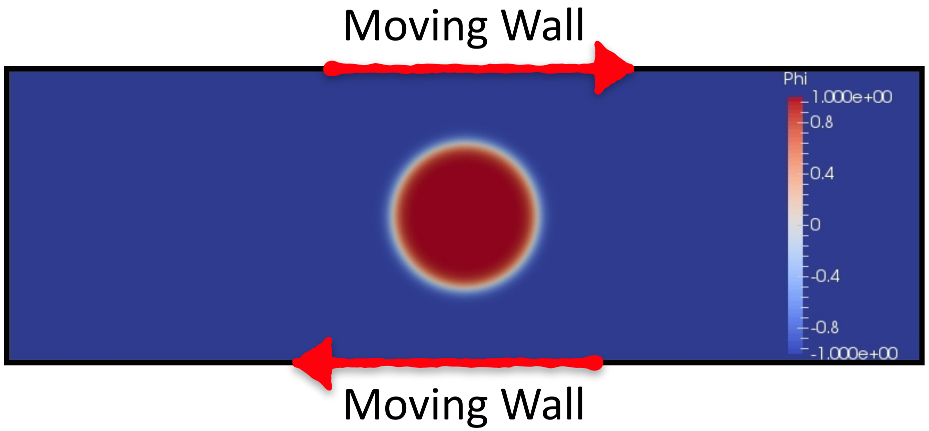
We consider a 2D droplet.

Data and parameters of the simulation:

N_x x N_z x N_y = 2 x 513 x 256

Capillary number: Ca= 0.187

Reynolds number: Re=0.1



Objective of this session:

- Load the phi_00*******.dat fields in Matlab (binary, little-endian): Matrix 2 x 513 x 256
- Compute the centre of mass
- Compute the major and minor axis of deformation
- Compute the deformation of the droplet
- Compute the orientation angle of the droplet
- Compare it with the theoretical results

Additional information:

Step from 0 to 3000 (every 100)

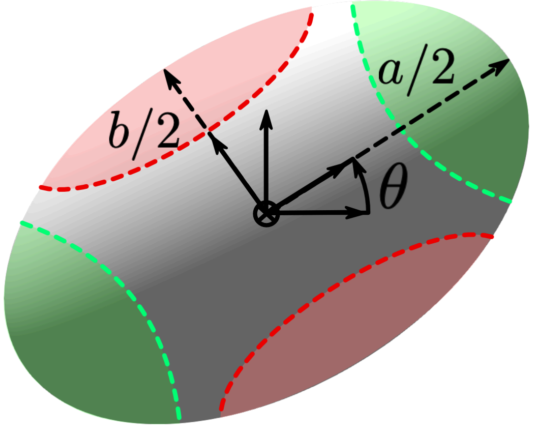
dt=5.e-4

Link to download the material: shorturl.at/mIJX4

Hypothesis:

The deformed shape of the droplet is an ellipsoid, we can compute the matrix of inertia of the droplet.

$$\begin{cases} I_{zz} = \int_A (y - y_g)^2 dA \\ I_{yy} = \int_A (z - z_g)^2 dA \\ I_{yz} = \int_A (y - y_g)(z - z_g) dA \end{cases}$$



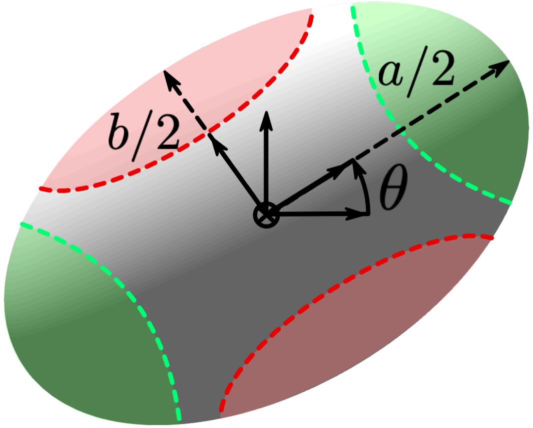
The eigenvalues of the matrix are the values of the principal axis of inertia:

$$\begin{cases} \lambda_1 = \dfrac{\pi a^3 b}{4} \\ \lambda_2 = \dfrac{\pi a b^3}{4} \end{cases}$$

We can solve for a and b.

Results:

$$\begin{cases} a = \left(\dfrac{16\lambda_1^3}{\pi^2\lambda_2}\right)^{(1/8)} \\ b = \dfrac{4\lambda_1}{\pi a^3} \\ \theta = \dfrac{1}{2} tan^{-1}\left(\dfrac{2I_{yz}}{I_{zz} - I_{yy}}\right) \end{cases}$$



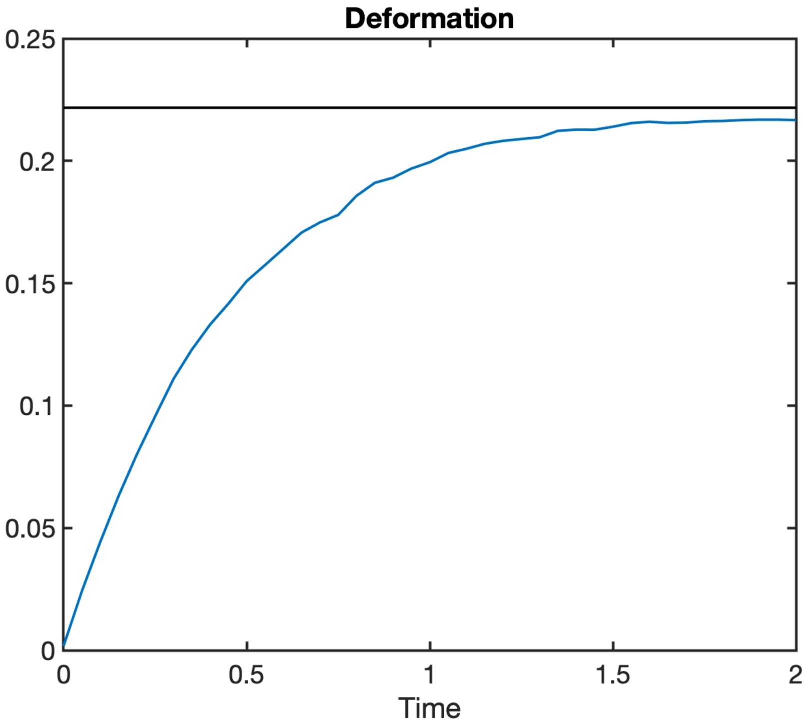
Theoretical results:

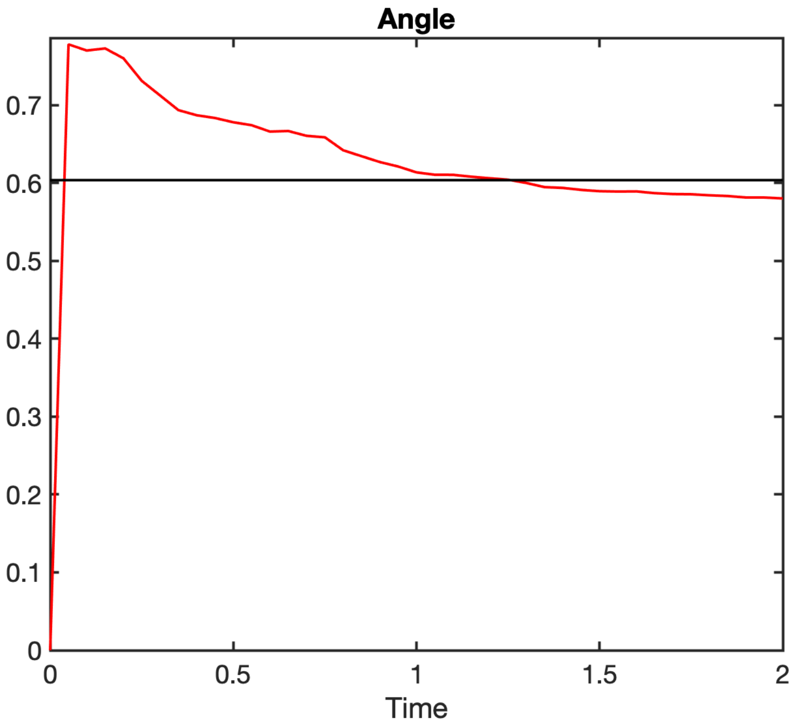
$$D = \frac{a-b}{a+b}$$

$$D = \frac{35}{32} Ca \left[1 + C_{SH} \frac{3.5}{2} \left(\frac{d}{4h}\right)^3\right]$$

$$\theta = \frac{\pi}{4} - \frac{(19\lambda + 16)(2\lambda + 3)}{80(1+\lambda)} Ca$$

Results:


Deformation
0.25
0.2
0.15
0.1
0.05
0
0
0.5
1
1.5
2
Time


Angle
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.5
1
1.5
2
Time

Hands-on Session #2

We consider the interaction between two droplets in shear-flow (a clean and a surf-laden case)

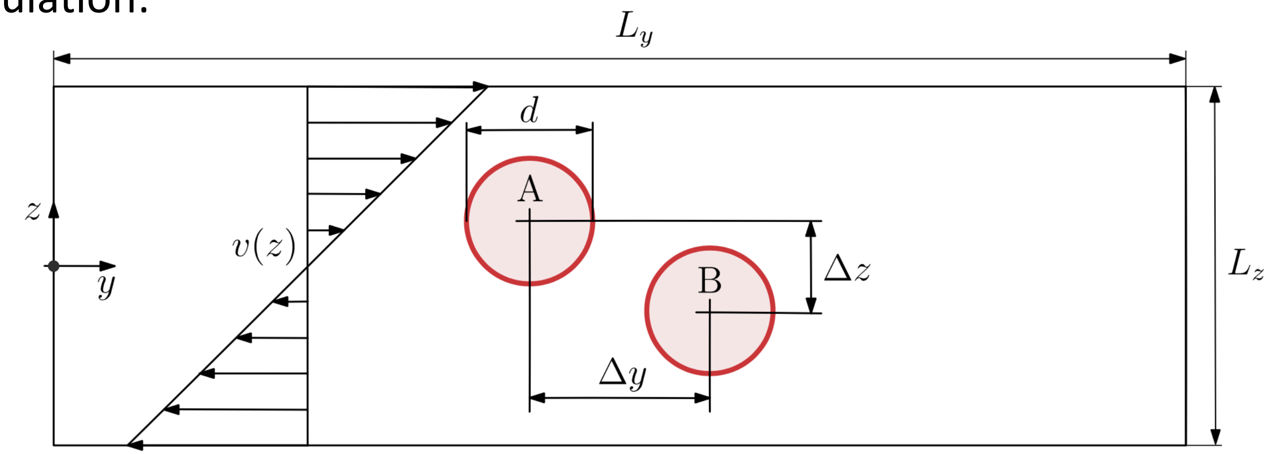
Data and parameters of the simulation:

N_x x N_z x N_y = 2 x 513 x 512

Capillary number: Ca= 0.1

Reynolds number: Re=0.5

Diameter = 0.7h



Objective of this session:

- Load the phi_00*******.dat fields in Matlab (binary, little-endian): Matrix 2 x 513 x 512
- Compute the centre of mass
- Compute the major and minor axis of deformation
- Compute the deformation parameter of the droplet (hint: same for both droplets)

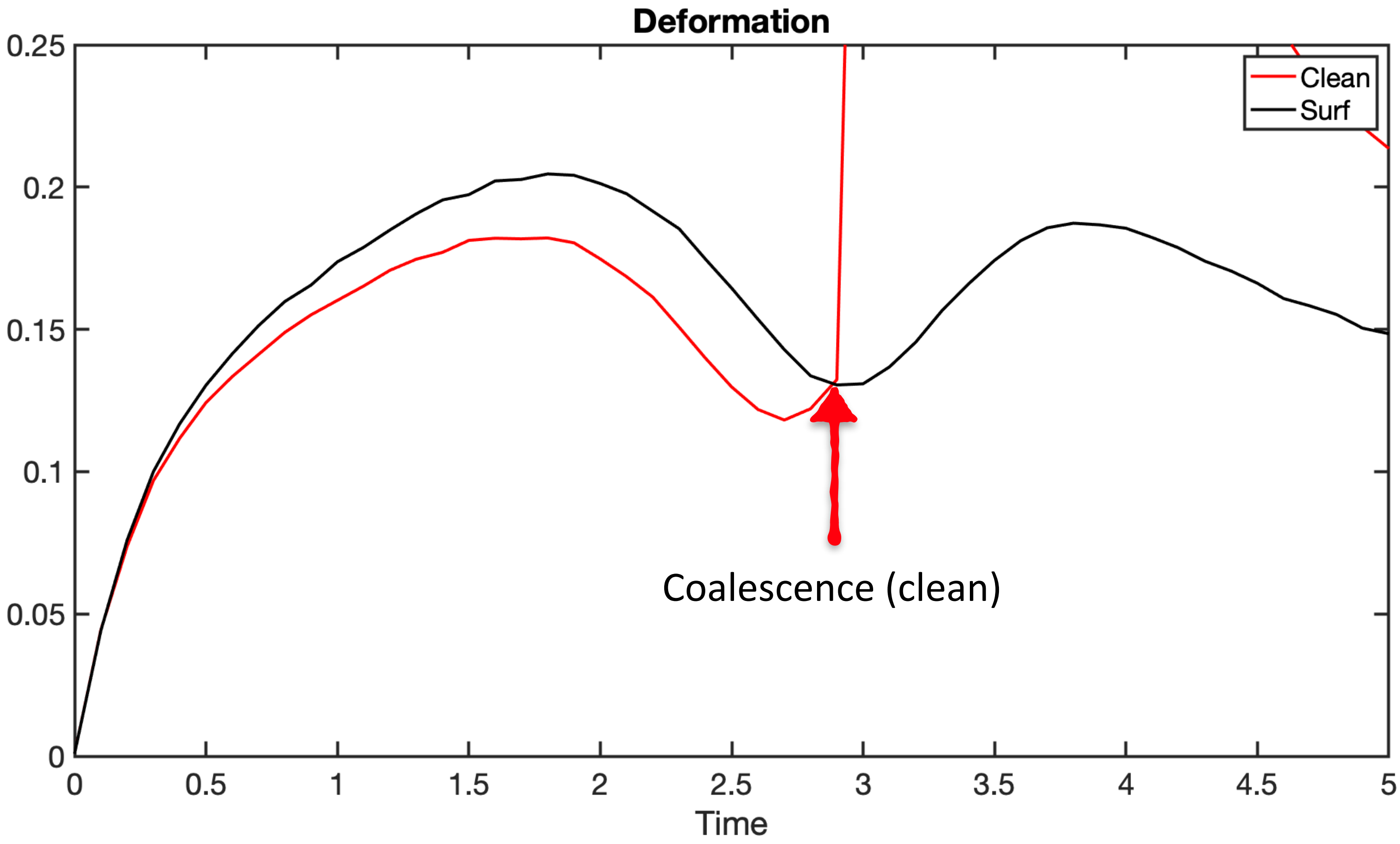
Additional information:

Step from 0 to 50000 (every 1000)

dt=1.e-4

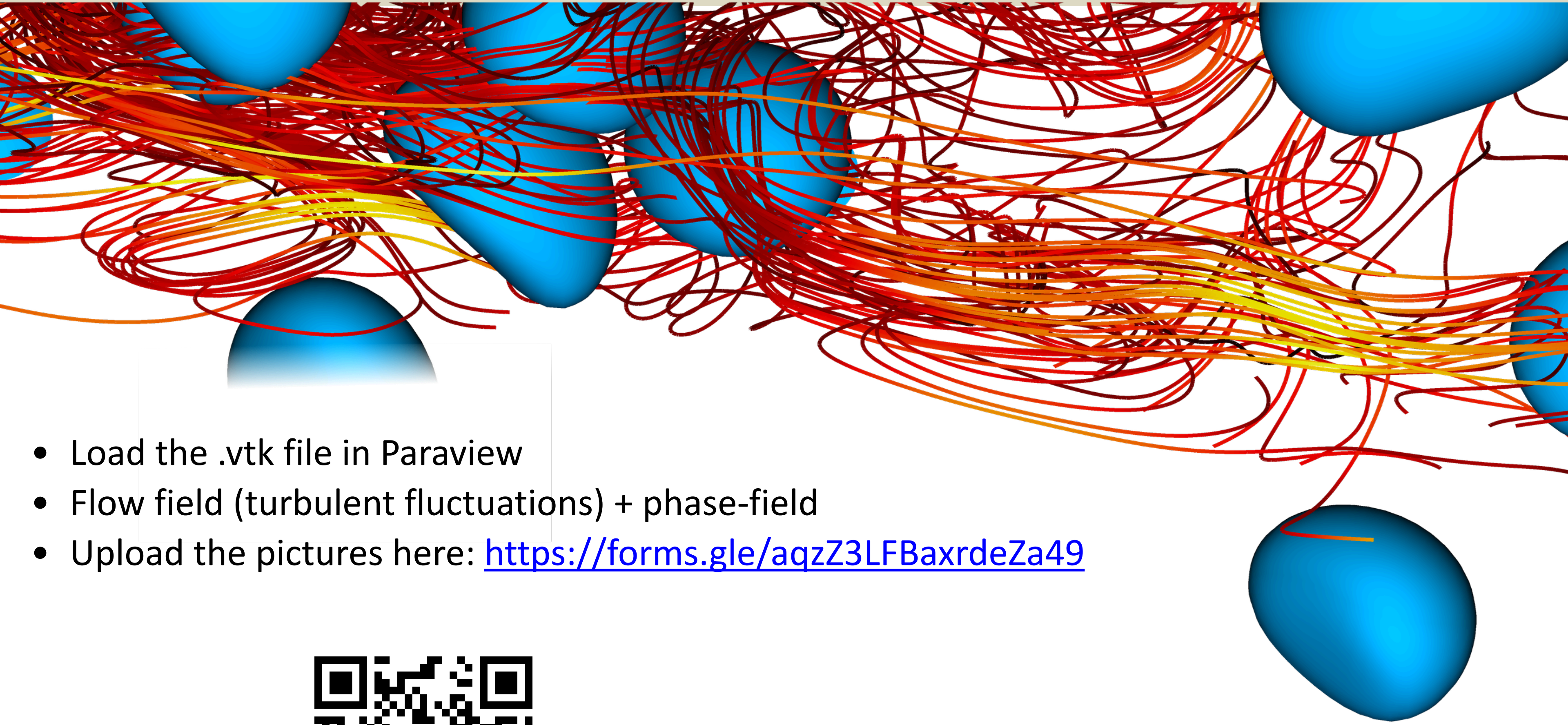
Link to download the material: shorturl.at/mIJX4

Results:



Coalescence (clean)

Hands-on Session #3



- Load the .vtk file in Paraview
- Flow field (turbulent fluctuations) + phase-field
- Upload the pictures here: https://forms.gle/aqzZ3LFBaxrdeZa49

Upload:

Backup
Influence Matrix-1

$$[P_1] = \begin{cases} \dfrac{\partial^2 w_1}{\partial z^2} - k_{i,j}^2 w_1 = \theta_1 & \text{in } \Omega \\ w_1 = w_\Gamma & \text{in } \Gamma \\ \dfrac{\partial^2 \theta_1}{\partial z^2} - \beta^2 \theta_1 = \dfrac{H^n}{\gamma} & \text{in } \Omega \\ \theta_1 = \theta_\Gamma & \text{in } \Gamma \end{cases}$$

$$[P_2] = \begin{cases} \dfrac{\partial^2 w_2}{\partial z^2} - k_{i,j}^2 w_2 = \theta_2 & & \text{in } \Omega \\ w_2 = 0 & & \text{in } \Gamma \\ \dfrac{\partial^2 \theta_2}{\partial z^2} - \beta^2 \theta_2 = 0 & & \text{in } \Omega \\ \theta_2(z=-1) = 1 & \theta_2(z=+1) = 0 & \end{cases}$$

$$[P_3] = \begin{cases} \dfrac{\partial^2 w_3}{\partial z^2} - k_{i,j}^2 w_3 = \theta_3 & & \text{in } \Omega \\ w_3 = 0 & & \text{in } \Gamma \\ \dfrac{\partial^2 \theta_3}{\partial z^2} - \beta^2 \theta_3 = 0 & & \text{in } \Omega \\ \theta_3(z=-1) = 0 & \theta_3(z=+1) = 1 & \end{cases}$$

Backup
Influence Matrix-2

General form of the BCs:

$$\begin{cases} p_1 \hat{w}^{n+1}(x,y,z=-1) + q_1 \left.\dfrac{\partial \hat{w}^{n+1}}{\partial z}\right|_{z=-1} = r_1 \\ p_2 \hat{w}^{n+1}(x,y,z=+1) + q_2 \left.\dfrac{\partial \hat{w}^{n+1}}{\partial z}\right|_{z=+1} = r_2 \end{cases} .$$

Replacing them in the different contributions:

$$\begin{bmatrix} p_1 w_2(-1) + q_1 \left.\dfrac{\partial w_2}{\partial z}\right|_{z=-1} & p_1 w_3(-1) + q_1 \left.\dfrac{\partial w_3}{\partial z}\right|_{z=-1} \\ p_2 w_2(-1) + q_2 \left.\dfrac{\partial w_2}{\partial z}\right|_{z=-1} & p_2 w_3(-1) + q_2 \left.\dfrac{\partial w_3}{\partial z}\right|_{z=-1} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} r_1 - p_1 w_1(-1) - q_1 \left.\dfrac{\partial w_1}{\partial z}\right|_{z=-1} \\ r_2 - p_2 w_1(-1) - q_2 \left.\dfrac{\partial w_1}{\partial z}\right|_{z=-1} \end{bmatrix}$$

We obtain the coefficients A and B.