Grid-refinement in Palabos

Palabos summer school - O. Malaspinas

June 9, 2021

Don't hesitate to interrupt!

General introduction

- Palabos is a **collaborative projet** (v2.3 currently) https://www.palabos.org.
- Merge requests are **encouraged** (19 contributors).
- Different means to communicate with us:
  - Palabos forum https://palabos-forum.unige.ch/
  - Discord server https://discord.com/invite/UEa9sEQ
  - Gitlab repository https://gitlab.com/unigespc/palabos
  - Twitter https://twitter.com/Palabos1
- Ressources:
  - Palabos online seminar series https://palabos.unige.ch/community/palabos-online-seminar-series/
  - YouTube channel: https://www.youtube.com/channel/UCO3qoJm3U8cu9D_lrqEHctQ
  - Live streams (follow me for announces: https://twitter.com/omalaspinas)

Planned additions

Drafts (in the coming weeks)

- Jonathan's WENO scheme.
- Francesco's and Irina's new boundary conditions.
- Christophe's general force models.

WIP (in the coming months)

- Jonas' GPU implementation.
- Orestis' adjoint method.

Somewhere in the cloud (in the coming century)

- Sébastien Leclair's color-gradient model.
- Adaptive grid refinement.
- Multi-phase grid refinement.

Grid-refinement generalities (1/2)

The need for variable resolution

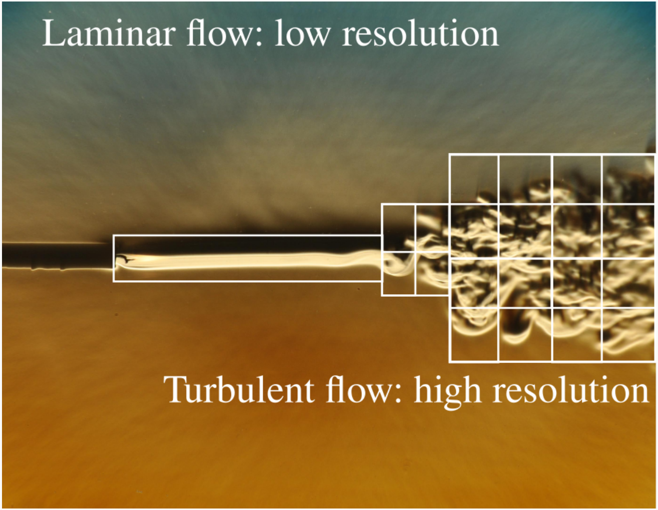


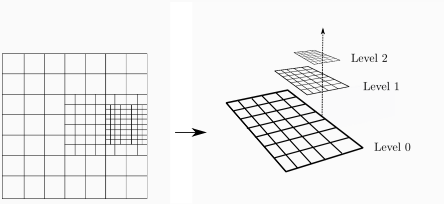
Figure 1: Source: Wikipedia, https://bit.ly/38l3Kor

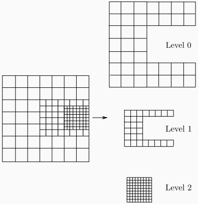
Grid-refinement generalities (2/2)

Basics of grid refinement

- Geometrical considerations[1]

Multi–grid | **Multi–domain**





- In Palabos: multi-domain[2]

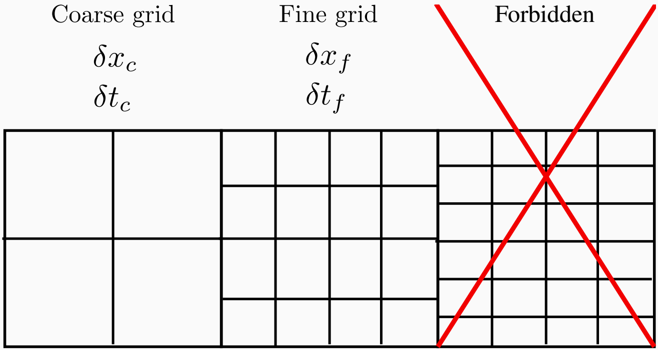
---

[1] P. Sagaut, et. al, Multiscale And Multiresolution Approaches in Turbulence, Imperial College Press, June 2006.

[2] D. Lagrava et al., Advances in multi-domain lattice Boltzmann grid refinement, J. Comp. Phys., 231, p. 4808-4822, (2012)

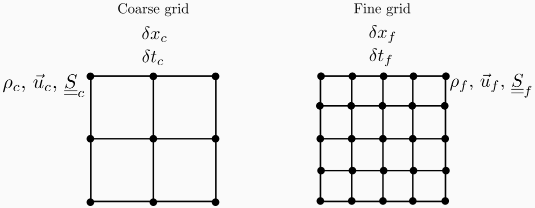
Basics of grid refinement (1/2)

- The discretization LBM performed over a regular grid
- Introduction of non–uniform structure
  - Discontinuity of the physical quantities
  - Quantities must be rescaled (use of LB units)
- Transitions are only powers of two



Basics of grid refinement (2/2)

- Coarse grid: p_c, $\vec{u}_c$, $\underline{\underline{S}}_c$, ...
- Fine grid: p_f, $\vec{u}_f$, $\underline{\underline{S}}_f$, ...



$f_{i,c}(t+1, \vec{x}+\vec{c}_i) = f_{i,c}(t,\vec{x}) - \frac{1}{\tau_c}\left(f_{i,c}(t,\vec{x}) - f_{i,c}^{(0)}(t,\vec{x})\right)$

$f_{i,f}(t+1, \vec{x}+\vec{c}_i) = f_{i,f}(t,\vec{x}) - \frac{1}{\tau_f}\left(f_{i,f}(t,\vec{x}) - f_{i,f}^{(0)}(t,\vec{x})\right)$

- Related via physical units: p, $\vec{u}$, $\underline{\underline{S}}$, ...

Pressure: $$p = \frac{\delta x_c^2}{\delta t_c^2} p_c = \frac{\delta x_f^2}{\delta t_f^2} p_f,$$

Velocity: $$\vec{u} = \frac{\delta x_c}{\delta t_c} \vec{u}_c = \frac{\delta x_f}{\delta t_f} \vec{u}_f,$$

Strain: $$\underline{\underline{S}} = \frac{1}{\delta t_c} \underline{\underline{S}}_c = \frac{1}{\delta t_f} \underline{\underline{S}}_f.$$

Questions?

Rescaling of macroscopic quantities[3] (1/2)

Density, pressure, and velocity

- We consider the convective scaling $\delta t \sim \delta x$.
  - Meaning $\delta x_c = \delta x_f/2$ then $\delta t_c = \delta t_f/2$.
  - There are **more** time-steps on the fine lattice.
  - p, $\vec{u}$, and ρ are continuous at the interface.
- Pressure and Density ($p = c_s^2\rho$):

$$\frac{\delta x_f^2}{\delta t_f^2} p_f = \frac{4\delta x_c^2}{4\delta t_c^2} p_f = \frac{\delta x_c^2}{\delta t_c^2} p_c,$$

$$p_f = p_c \Leftrightarrow \rho_f = \rho_c.$$

- Velocity:

$$\frac{\delta x_f}{\delta t_f} \vec{u}_f = \frac{2\delta x_c}{2\delta t_c} \vec{u}_f = \frac{\delta x_c}{\delta t_c} \vec{u}_c,$$

$$\vec{u}_f = \vec{u}_c.$$

[3] A. Dupuis, B. Chopard, Theory and applications of an alternative lattice Boltzmann grid refinement algorithm, Physical Review E, 67 (2003), p. 066707.

Rescaling of macroscopic quantities (2/2)

Rate of strain tensor

- We consider the convective scaling $\delta t \sim \delta x$.
  - Meaning $\delta x_c = \delta x_f/2$ then $\delta t_c = \delta t_f/2$.
  - p, $\vec{u}$, and ρ are continuous at the interface ($\vec{u}_f = \vec{u}_c, \dots$).
- Strain:

$$\frac{1}{\delta t_f}\underline{\underline{S}}_f = \frac{2}{\delta t_c}\underline{\underline{S}}_f = \frac{1}{\delta t_c}\underline{\underline{S}}_c,$$
$$\underline{\underline{S}}_f = \frac{1}{2}\underline{\underline{S}}_c.$$

Rescaling of populations (1/2)

Three different parts: f_i, $f_i^{(0)}$, and f_i^{neq}

- Populations are represented as $f_i = f_i^{(0)} + f_i^{\mathrm{neq}}$

$$f_i^{(0)} = w_i \rho \left(1 + \frac{\vec{c}_i \cdot \vec{u}}{c_s^2} + \frac{1}{2c_s^4} (\vec{c}_i \vec{c}_i - c_s^2 \underline{\underline{I}}) : \vec{u}\vec{u} \right).$$

- We know $\rho_c = \rho_f$, $\vec{u}_c = \vec{u}_f$.
- Equilibrium pop:

$$f_{i,c}^{(0)} = f_{i,f}^{(0)}.$$

- Non-equilibrium pop:
  - $f_i^{\mathrm{neq}} = f_i - f_i^{(0)}$ is not continuous.
  - $f_{i,c}^{\mathrm{neq}} = \alpha f_{i,f}^{\mathrm{neq}}$.

Rescaling of populations (2/2)

Let us start with the BE

BE equation with BGK approximation ($f(\vec{x}, \vec{c}, t)$)

$$(\partial_t + \vec{c} \cdot \vec{\nabla}_{\vec{x}}) f = -\frac{1}{\tau}(f - f^{(0)})$$

Velocity discretization

Finite velocity BE equation ($f(\vec{x}, \vec{c}_i, t) \equiv f_i(\vec{x}, t)$)

$$(\partial_t + \vec{c}_i \cdot \vec{\nabla}_{\vec{x}}) f_i = -\frac{1}{\tau}(f_i - f_i^{(0)})$$

We rewrite it

$$\frac{\mathrm{d} f_i}{\mathrm{d} t} = -\frac{1}{\tau}(f_i - f_i^{(0)})$$

A priori determination of α (1/2)

Space-time discretization

After numerical integration along characteristics

$$f_i^+ - f_i = -\frac{\delta t}{2\tau}\left(f_i^+ - {f_i^{(0)}}^+ + f_i - f_i^{(0)}\right),$$

"+" : function evaluated at position $\vec{x} + \vec{c}_i\delta t$ and time $t + \delta t$. With the change of variable

$$\bar{f}_i = f_i + \frac{\delta t}{2\tau}\left(f_i - f_i^{(0)}\right),$$

$$\bar{\tau} = \frac{2\tau + \delta t}{2\delta t},$$

we obtain

$$\bar{f}_i^+ = \bar{f}_i - \frac{1}{\bar{\tau}}\left(\bar{f}_i - f_i^{(0)}\right).$$

A priori determination of α (2/2)

Non-equilibrium distribution

Subtracting $f_i^{(0)}$ from $\bar{f}_i = f_i + \frac{\delta t}{2\tau}\left(f_i - f_i^{(0)}\right)$

$$\bar{f}_i^{\mathrm{neq}} = \left(\frac{2\tau + \delta t}{2\tau}\right) f_i^{\mathrm{neq}} \Leftrightarrow f_i^{\mathrm{neq}} = \left(\frac{2\tau}{2\tau + \delta t}\right) \bar{f}_i^{\mathrm{neq}},$$

Continuity of f_i^{neq}

Ensure continuity of “bare” quantities on coarse and fine grid

$$\left(\frac{2\tau}{2\tau + \delta t_c}\right) \bar{f}_{i,c}^{\mathrm{neq}} = \left(\frac{2\tau}{2\tau + \delta t_f}\right) \bar{f}_{i,f}^{\mathrm{neq}},$$

$$\frac{\delta t_c}{\bar{\tau}_c} \bar{f}_i^{\mathrm{neq,c}} = \frac{\delta t_f}{\bar{\tau}_f} \bar{f}_i^{\mathrm{neq,f}},$$

$$\alpha = \frac{\delta t_f}{\delta t_c} \frac{\bar{\tau}_c}{\bar{\tau}_f}$$

Rescaling in Palabos (1/4)

From coarse to fine

- Decompose: from $f_{i,c}$, compute ρ_c, $\vec{u}_c$, $f_{i,c}^{\mathrm{neq}}$.

$$\rho_c = \sum_i f_{i,c}, \quad \vec{u}_c = \sum_i f_{i,c}\vec{c}_i / rho_c, \quad f_{i,c}^{\mathrm{neq}} = f_{i,c} - f_{i,c}^{(0)}.$$

- Rescale:

$$\rho_f = \rho_c, \quad \vec{u}_f = \vec{u}_c, \quad f_{i,f}^{\mathrm{neq}} = f_{i,c}^{\mathrm{neq}}/\alpha.$$

- Recompose: from ρ_f, $\vec{u}_f$, $f_{i,f}^{\mathrm{neq}}$ compute $f_{i,f}$.

$$f_{i,f} = f_i^{(0)}(\rho_f, \vec{u}_f) + f_{i,f}^{\mathrm{neq}}.$$

Rescaling in Palabos (2/4)

From coarse to fine

- Decompose: from $f_{i,f}$, compute ρ_f, $\vec{u}_f$, $f_{i,f}^{\mathrm{neq}}$.

$$\rho_f = \sum_i f_{i,f}, \quad \vec{u}_f = \sum_i f_{i,f}\vec{c}_i/rho_f, \quad f_{i,f}^{\mathrm{neq}} = f_{i,f} - f_{i,f}^{(0)}.$$

- Rescale:

$$\rho_c = \rho_f, \quad \vec{u}_c = \vec{u}_f, \quad f_{i,c}^{\mathrm{neq}} = \alpha f_{i,f}^{\mathrm{neq}}.$$

- Recompose: from ρ_c, $\vec{u}_c$, $f_{i,c}^{\mathrm{neq}}$ compute $f_{i,c}$.

$$f_{i,c} = f_i^{(0)}(\rho_c, \vec{u}_c) + f_{i,c}^{\mathrm{neq}}.$$

Rescaling in Palabos (3/4)

Some code

```
class Rescaler {
  // Rescales rel. freq. (potentially many of them)
  virtual Array<T,Descriptor<T>::q> computeRescaledRelFreq(
    const Array<T,Descriptor<T>::q> &relFreq, T xDt) const;
  // Recale the decomposed quantities
  virtual void rescale(const Dynamics<T,Descriptor> &dyn,
    T xDt, std::vector<T> &rawData ) const = 0;

  // Decomposes a cell into rho, u, fneq, and rescales it
  virtual void decomposeAndRescale(
    Cell<T,Descriptor> const& cell, T xDt, plint order,
    std::vector<T> &decompAndRescaled) const;
  // Other things (constructor, ...)
}
```

For BGK (simplified)

```
class Rescaler {
  // xDt -> 2 other 1/2
  virtual void rescale(const Dynamics<T,D> &dyn,
    T xDt, std::vector<T> &rawData ) const {
      // rawData[0] = rho, rawData[1-3] = u,
      // rawData[4-q] = fneq
      Array<T, D<T>::q> resRelFreq =
        this->computeRescaledRelFreq(relFreq, xDt);
      for (plint iPop = 0;
            iPop < SymmetricTensorImpl<T,D<T>::d>::n; ++iPop) {
        plint iA = 1+D<T>::d+iPop;
        T prefactor = relFreq[iA] / resRelFreq[iA] * xDt;
        rawData[iA] *= prefactor;
      }
    }
}
```

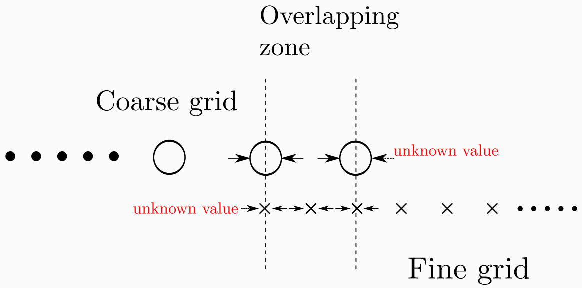
Questions on rescaling

Questions?

Overlapping zone

Coupling between refinement zones

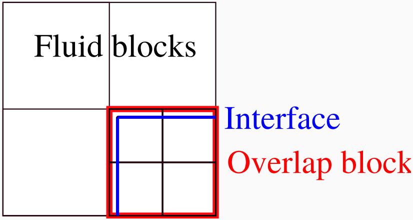
- Two way coupling to complete missing information.



- Need for a buffering zone: “overlap”.
- In Palabos the thickness is *one* coarse node.

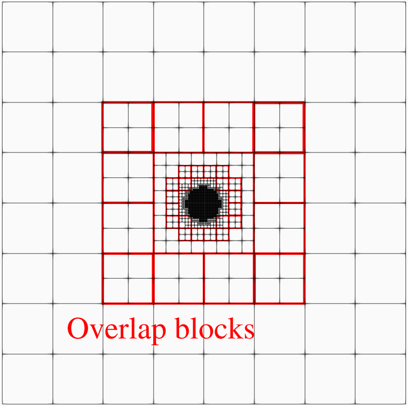
Overlapping in Palabos (1/3)

Overlap and interface



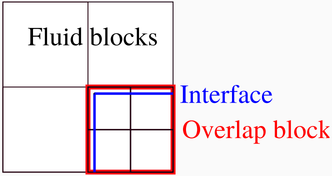
- Communication in overlaps
- Implemention: `NTensorFields`
- Containing `rawData`
- Efficiency vs "ease" of impl.

Complex overlap



Overlapping in Palabos (2/3)

Overlap and interface



- Complex data struct.
- Sparse data struct.
- Complex interface geom.

MultiLevel3D

Each grid level contains

```
// Lattice at that level
MultiBlockLattice3D<T, D> *lattice;
// Buffer zones
MultiNTensorField3D<T> *decomp_t0,
                       *decomp_t12,
                       *decomp_t1,
                       *decomp_fine;
```

Coupling involves different operations on the interface.

Overlapping in Palabos (3/3)

MultiLevel3D

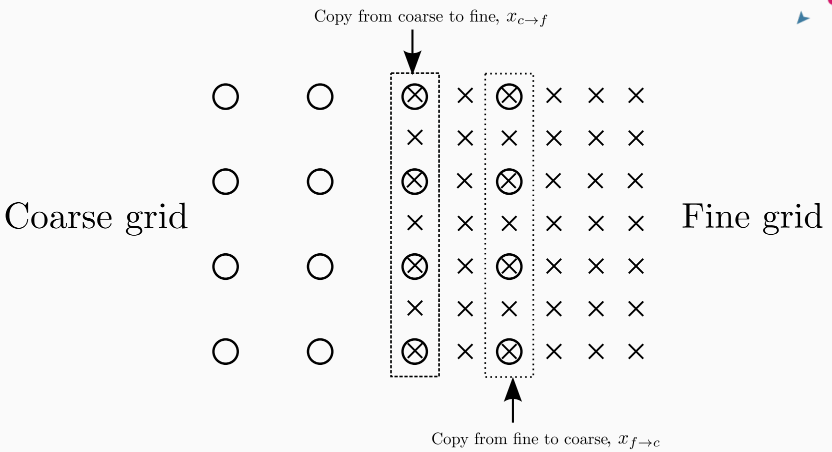
- `DecomposeAndRescaleFunctional3D`

```
Cell<T,Descriptor> &cell = lattice.get(iX,iY,iZ);
egine.decomposeAndRescale( cell, xDt, order,
  decompAndRescaled);
for (iA = 0; iA < decompAndRescaled.size(); ++iA) {
  tensor.get(oX,oY,oZ)[iA] = decompAndRescaled[iA];
}
```

- `RecomposeFunctional3D`

```
Cell<T,Descriptor> &cell = lattice.get(iX,iY,iZ);
for (iA = 0; iA < nDim; ++iA) {
  decomposed[iA] =
    tensor.get(oX,oY,oZ)[iA];
}
cell.getDynamics().
  recompose(cell, decomposed, order );
```

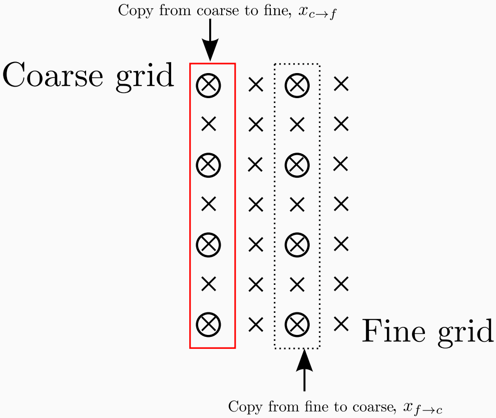
Two-dimensional interface



Coarse to fine coupling

Missing information

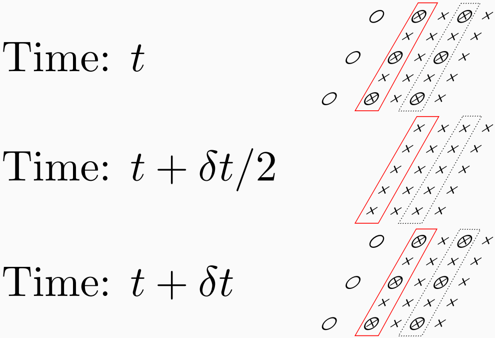
- One must "increase" the information.



- Copy on superposed nodes.
- Interpolate missing info.
- Temporal **and** spatial interpolations.

Coarse to fine coupling: temporal interpolation

Need time interpolation for $t = t + \delta t/2$.



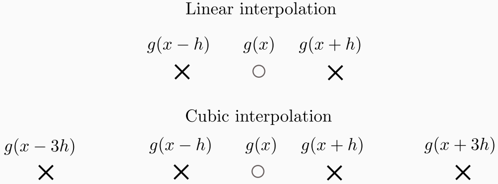
Linear time interpolation (second order)

$$f_i(\vec{x}, t + \delta t/2) = \frac{f_i(\vec{x}, t + \delta t) + f_i(\vec{x}, t)}{2}.$$

Coarse to fine coupling: spatial interpolation

What interpolation ?

- Linear interpolation (second order)
- Cubic interpolation (fourth order)



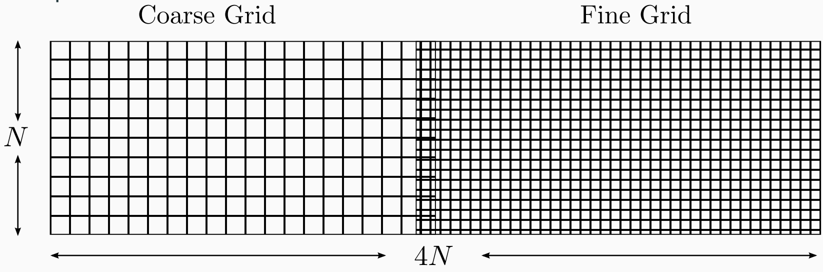
- Which one to chose?

  Cubic interpolation.

Importance of the spatial interpolation (1/2)

Second order interpolation is not enough

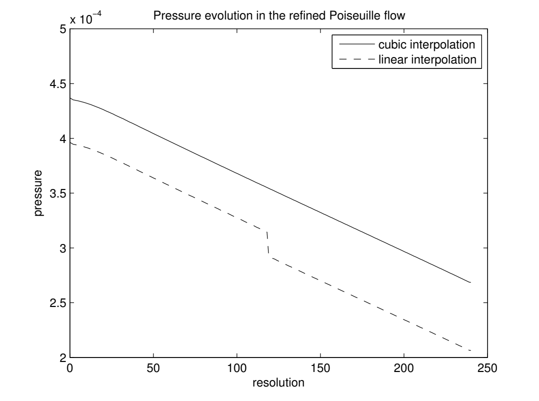
- Numerical proof using a simple 2D Poiseuille flow
- Compare linear spatial interpolation and cubic spatial interpolation
- Setup of the simulation



Importance of the spatial interpolation (2/2)

Results

- A linear pressure gradient is expected.
- The pressure gradient of the simulation (both interpolations)



- There is a loss of mass on the interface (when $\tau \rightarrow 1/2$, or high Re)!
- No more second order accuracy

Questions on interpolations

Questions?

Coarse to fine coupling in Palabos

Processing functionals

- Couplings is done through Processing Functionals.
- Ordering through negative levels (executed explicitly).

Spatial processing functionals

- From coarse `MultiNTensorField3D` to fine `MultiNTensorField3D`

```
CopyAndSpatialInterpolationPlaneFunctional3D
CopyAndSpatialInterpolationEdgeFunctional3D
CopyAndSpatialInterpolationCornerFunctional3D
```

Temporal functionals

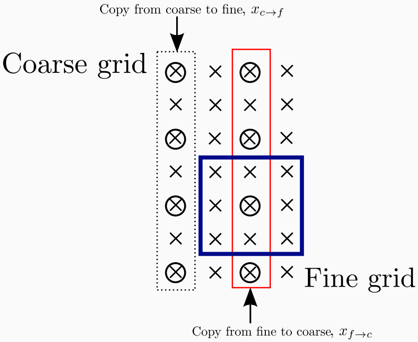
- From fine `MultiNTensorField3D` to fine `MultiNTensorField3D`

```
TemporalInterpolationFunctional3D
```

Fine to coarse coupling

Filtering

The fine grid has “too much” information



Averaging over all lattice directions

$$f_{i,f}^{\mathrm{neq}}(\vec{x}_{f\to c}^{c}, t) = \frac{1}{q}\sum_{i=0}^{q-1} f_{i,f}^{\mathrm{neq}}(\vec{x}_{f\to c}^{c} + \vec{c}_i, t)$$

Fine to coarse coupling in Palabos

Filtering functional

```
for (plint iA = 0; iA < minIndex; ++iA) {
  // rho and u may not be filtered (only copied)
  cTensor.get(iX,iY,iZ)[iA] = fTensor.get(fX,fY,fZ)[iA];
}
for (plint iA = minIndex; // only fneq is
     iA < nDim-D<T>::ExternalField::numScalars; ++iA) {
  cTensor.get(iX,iY,iZ)[iA] = fTensor.get(fX,fY,fZ)[iA];
  for (plint iPop = 1; iPop < q; ++iPop) {
    plint nextX = fX+c[iPop][0];
    plint nextY = fY+c[iPop][1];
    plint nextZ = fZ+c[iPop][2];
    cTensor.get(iX,iY,iZ)[iA] +=
      fTensor.get(nextX,nextY,nextZ)[iA];
  }
  cTensor.get(iX,iY,iZ)[iA] /= (T)q;
}
```

Algorithm

One time step: $t \to t + \delta t$

- CS $t \to t + \delta t$.

- CS $t \to t + \frac{\delta t}{2}$.
- CS $t + \frac{\delta t}{2} \to t + \delta t$.

- Time interpolation.
- Space interpolation.
- Complete fine.
- Space interpolation.
- Complete fine.
- Filter.
- Complete coarse.

The algorithm in Palabos

Recursive algorithm

```
void collideAndStream(plint iL) {
  lattice[iL].collideAndStream();  // collision coarse
  if (iL < (plint)(gridLevels.size()-1)) {
    // coarse to fine coupling
    lattice[iL].decomposeAndRescale(); // t+1, resc f_i in fine NTensor
    lattice[iL].timeInterp(); // interp at time t + 1/2
    collideAndStream(iL+1);  // collision fine t->t+1/2
    lattice[iL].spatialInterp(); // at time t+1/2
    lattice[iL+1].recompose() // fine lattice recomposed at t+1/2
    lattice[iL+1].executeProcessors(); // BC, stats, ...
    collideAndStream(iL+1);  // collision fine t+1/2->t+1
    lattice[iL].spatialInterp(); // at time t+1
    lattice[iL+1].recompose() // fine is OK
    lattice[iL+1].executeProcessors(); // BC, stats, ...
    // fine to coarse coupling
    lattice[iL+1].decomposeAndRescale(); // t+1, resc f_i in coarse NTensor
    lattice[iL].filter();
    lattice[iL].recompose(); // coarse is OK
  }
}
```

Questions on the algorithm

Questions?

Data analysis in Palabos

MultiLevelScalar/Tensor fields

- Must have the same grid structure than `MultiLevel3D`.
- The interface is similar to `MultiBlock3D`.

Processing functionals

- `Integrate/Apply` must specify grid level.
- `Reductive` must specify grid level and provide “container” for the result:
  - More data is generated for fine level than coarse (twice more).

Grid generation (1/2)

Spoiler

- Grid generation is a very complex topic.

Grid density

- Simplification: offload to external tool.
- Grid density: Scalar field $\in [0, 1]$.
- Generated manually or by analyzing “coarse” simulations:
  - Typically more points where the are “large” gradients.

Grid generation (2/2)

Octree grid

- Start with cuboid.
- If grid density $>$ threshold divide.
- Continue as long as the max number levels has not been reached.
- Balance the load on multiple processors.
- There are strong constraints:
  - Only factor of two at each interface.
  - Overlap blocks must be on the same processor as finer blocks.
  - Remove useless blocks (inside geometry).

The end

Questions?

Live stream

Sorry I lied

Two live demos:

- Naive and non-naive grid generation for the cavity.
- Implemention of the grid-refined-3d-cavity.

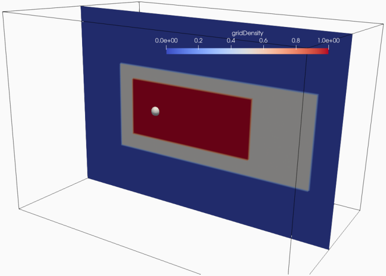
Exercises

- Three different exercises
  1. Familiarization with grid density.
  2. Generate alternative grid densities.
  3. Add functionalities for external flow simulations.
- Inspire yourself from existing code to add novel functionalities:
  - Read the code and understand it.
  - Develop new functionalities.
- Look for `Exercise` comments to find where to add functionalities.

Exercises: Grid density (1/2)

Boxes

- Add/remove boxes to see how grid densities are built.
- Visualize grid density fields.

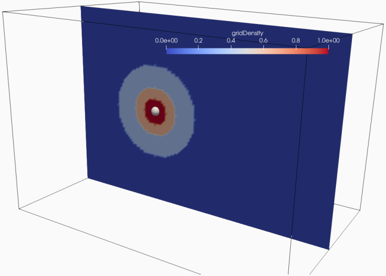


- Modify the `simpleSphere_exercise.xml` to add boxes.

Exercises: Grid density (2/2)

Spheres

- Inspired by the Boxes code create spherical grid densities.
- Look for `Exercise` comments to find the relevant places to add code.



- Modify the `generateGridDensityFromSpheres_exercise.cpp` to add spherical shapes grid density.

Exercises: flow past a sphere (1/2)

Reynolds stress definition

- Reynolds decomposition

$$\vec{u} = \bar{\vec{u}} + \vec{u}',$$

  $\bar{\vec{u}}$ and $\vec{u}'$ mean and fluctuating part.
- Reynolds stress tensor

$$\underline{\underline{T}} = \overline{\vec{u}'\vec{u}'}.$$

Add Reynolds stress

- Compute $\bar{\vec{u}}$ by averaging $\vec{u}$ over time.
- Once $\bar{\vec{u}}$ has converged:
  - Compute $\vec{u}' = \vec{u} - \bar{\vec{u}}$.
  - Compute $\vec{u}'\vec{u}'$.
  - Average $\vec{u}'\vec{u}'$ over time to get $\underline{\underline{T}}$.
- Need `integrateProcessingFunctional`.

Add probes

- Want to measure values on special places.
- Use existing probes of Palabos: reductions.
- Input: position. Output: velocity, pressure, vorticity, . . .
- Difficulty, one must decide on which level to apply the probes and store them.

The real end

Questions?