Palabos Summer School 2021

Monday morning introductory class

- Introduction and overview
- How to develop new models in Palabos

Organizers

Christophe Coreixas is senior researcher at the University of Geneva. His current research focuses on advanced LB collision models and compressible fluid flow.

Jonas Latt is professor in computer science at the University of Geneva. He specializes in LB theory, LB applications in geosciences and computational biology, and in high performance computing.

Jonathan Lemus is PhD researcher at the University of Geneva. He specializes in computational geophysics, in particular numerical modeling of plume transport in volcanic eruptions.

Orestis Malaspinas is senior lecturer and researcher at the University of Applied Sciences and Arts of Western Switzerland who specializes in the theory and applications of the LB method.

Francesco Marson is PhD researcher at the University of Geneva specializing in biomedical flows and modeling of curved boundaries in the LB method.

Jose Pedro de Santana Neto is PhD researcher at the University of Geneva specializing in LB modeling of magmatic flows.

Rémy Petkantchin is PhD researcher at the University of Geneva with specialty in numerical modeling of biomedical processes.

Schedule

	9:00 - 10:30	**10:45-12:00**	**13:30 - 14:30**	**14:45-17:00**
Mon June 7	Theory: The Palabos library; overview and model development	Palabos hands-on	Theory: Partially saturated bounce-back	Palabos hands-on
Tue June 8	Theory: Use of curved boundary conditions in Palabos	Palabos hands-on	Theory: Advection-diffusion with sharp interfaces and couplings	Palabos hands-on
Wed June 9	Theory: Mesh refinement in Palabos	Palabos hands-on	Theory: Hybrid CPU-GPU simulations for blood flow with RBCs	Palabos hands-on

Organization

- Theory sessions will be recorded and will be publicly available after the class, whenever possible.
- You are welcome to ask questions during the theory sessions. No need to raise your hands ! These questions will be recorded, though.
- After the theory, there will be a non-recorded session for questions.
- Hands-on sessions are in three different Zoom meetings, as the participants are assigned to three groups.

Course Materials

All course materials are in the Google Drive folder:

https://tinyurl.com/PalabosSummerSchool2021

Recording starts

History of Palabos

Around 2005: Predecessors of Palabos: Vladymir, OpenLB

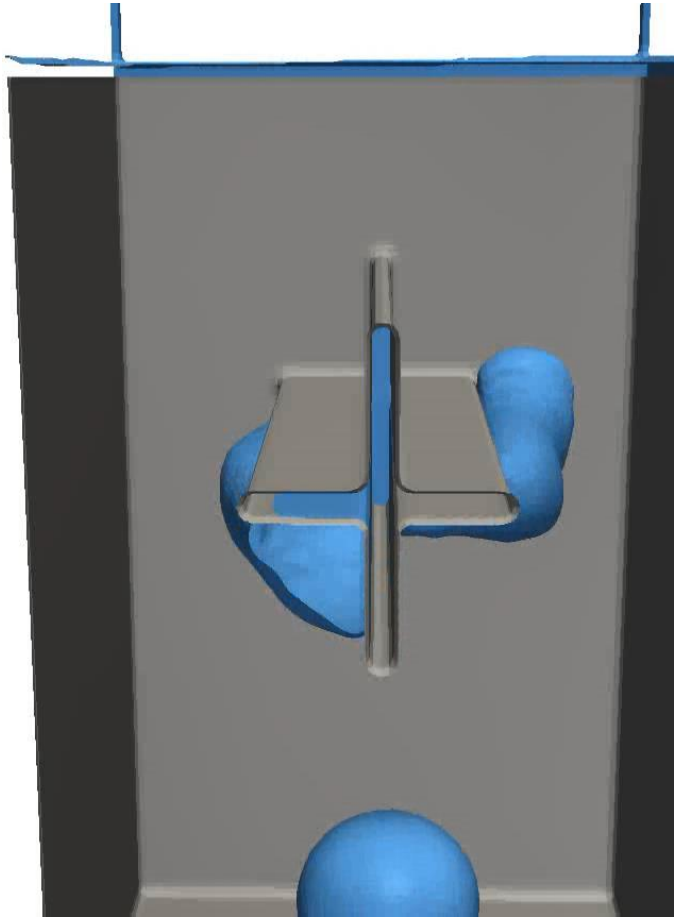
2010: First version of Palabos

2011: Joint development UniGe / FlowKit

2018: Numeca / FlowKit develop commercial Omnis/LB

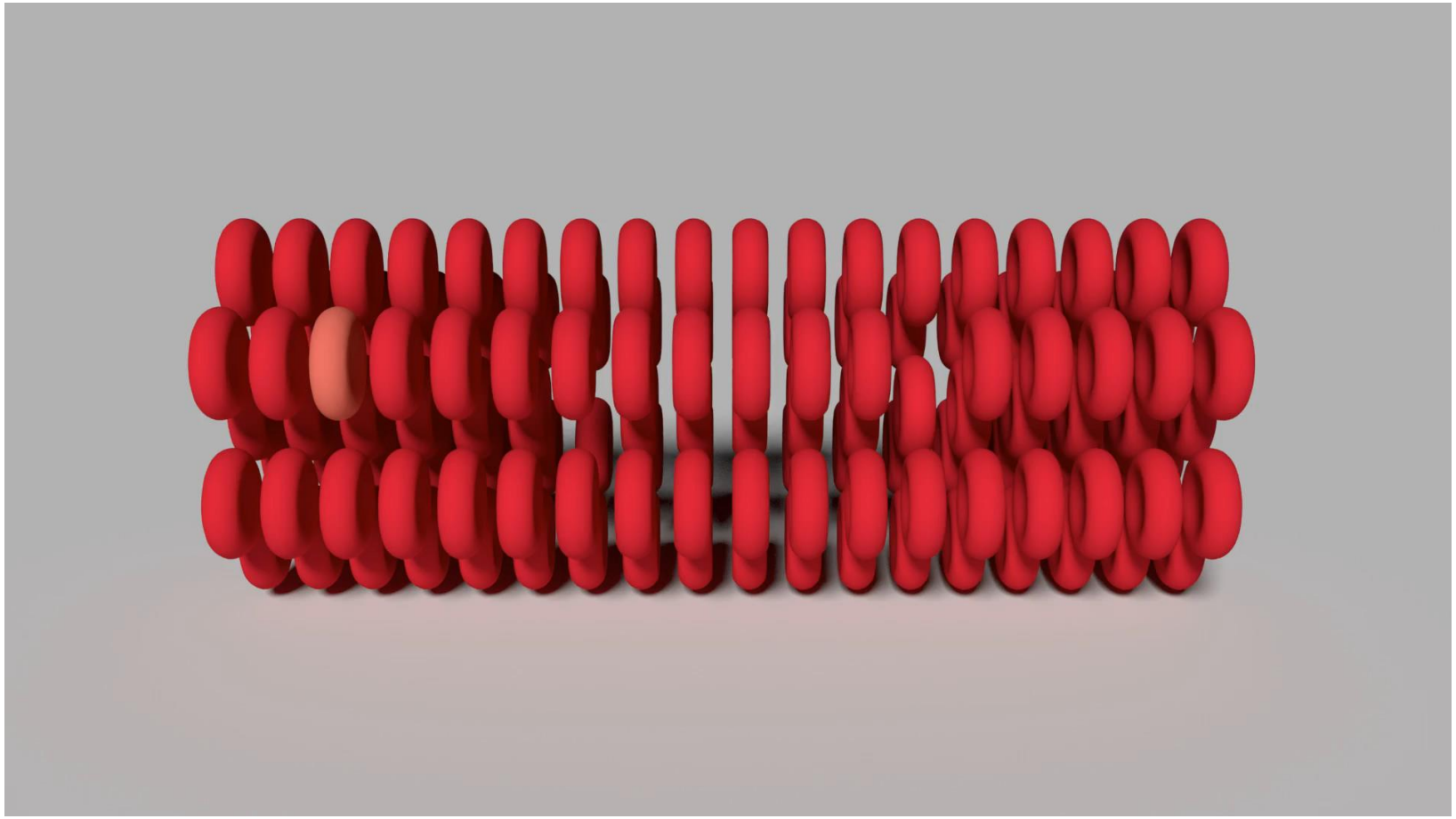
2018: UniGe and hepia continue to develop the open-source code Palabos

Palabos: Complex flows

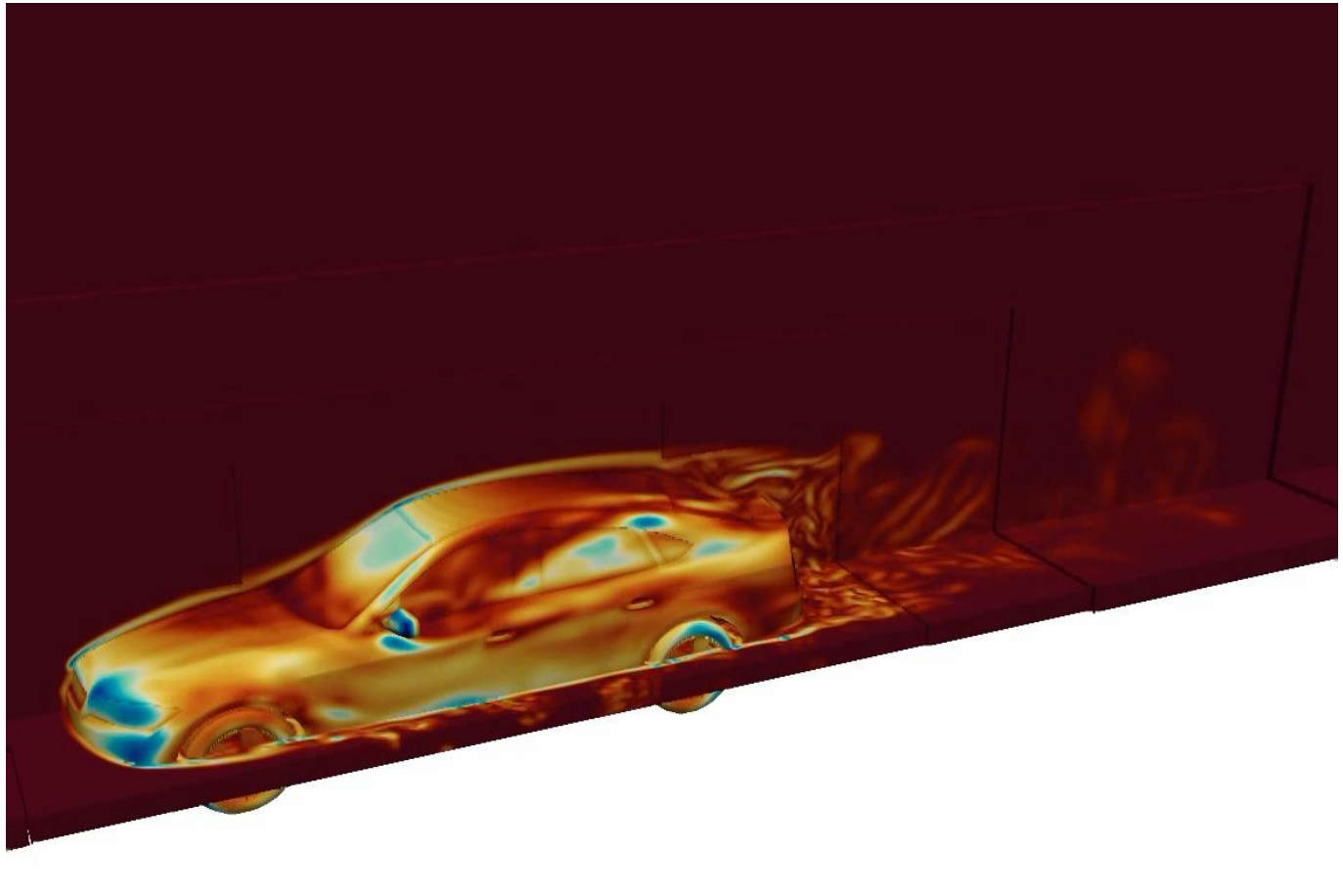


Free-surface model with
bubble pressure correction

Palabos: Coupled flows



Palabos: Refined meshes



Recent additions to Palabos

- A large variety of modern collision models (see this morning's exercise session)
- Partially saturated bounce-back model (presented this afternoon)
- Coupling with advection-dominated phenomena (Tuesday afternoon)
- Fluid-structure interaction (see Wednesday afternoon)

Discussion: current trends in lattice Boltzmann research

1. Automatic mesh refinement (AMR)
   - In principle well suited to LBM thanks to flexible block-structured refinement.
   - Not currently a direct development effort in Palabos, but an ongoing thought.

2. Exascale computing
   - How to run on machines with millions of cores ?
   - Use of hybrid parallelism (Palabos is MPI only)
   - Use of accelerators (GPUs)

GPU programming for LBM
The STLBM project

- While LBM on GPUs has been around for quite a while, the software development models have become much easier recently.
- One approach is C++ parallel algorithms, which uses built-in parallelism in the C++ language.
- Case study: the STLBM project (https://gitlab.com/unigehpfs/stlbm). Implements full LB applications on GPU with around 600 lines of code.
- Application of C++ parallel algorithms to GPU? Ongoing project *From CPU to GPU in 80 days* https://palabos.unige.ch/community/cpu-gpu-80-days/

Questions ?

Palabos Programming Model

Description of data layout: Lattice Descriptor

- Modifies the `MultiBlockLattice` with C++ templates.
- Prescribes number of populations (e.g. 19)
- Describes additional constants (e.g. discrete velocities)
- Adds additional variables to every cell (e.g. force term)

Definition of collision models: Dynamics objects

- Only applicable to local algorithms (limited to a cell)
- User defines collision and other functions (computation of macroscopic variables, computation of equilibrium, …)

Data Container: `MultiBlock`

- `MultiBlockLattice`
- `MultiScalarField`
- `MultiTensorField`

Parallelism is built-in

Definition of non-local algorithms or couplings between multi-blocks: Data Processors

- As for everything else, parallelism is implicit

Example: implementation of a multi-phase model

- Single-population pseudo-potential model for multi-phase flow
- A non-local interaction force enforces and maintains density gradients across multi-phase interfaces.
- A **lattice descriptor** is used to reserve space for the force term in each cell.
- A **collision model (dynamics object)** is written to implement the force through a Guo forcing term.
- A **data processor** is written to compute the non-local interaction and write it into every cell's reserved memory space for the force.

Shan, X., & Chen, H. (1993). *Lattice Boltzmann model for simulating flows with multiple phases and components.* Physical Review E, 47(3), 1815.

Guo, Z., Zheng, C., & Shi, B. (2002). *Discrete lattice effects on the forcing term in the lattice Boltzmann method.* Physical Review E, 65(4), 046308.

Use of a lattice descriptor with force term

Use a predefined descriptor for the D2Q9 lattice with reserves 9+2 variables (for the populations and for the force term).

```
typedef double T;
#define DESCRIPTOR descriptors::ForcedD2Q9Descriptor

MultiBlockLattice3D<T, DESCRIPTOR> lattice(nx, ny, new BGKdynamics(omega));
```

Instantiate the lattice with this descriptor. The mechanism is template-based and therefore efficiently carried out a precompile time.

Write a Data Processor to compute the interaction force

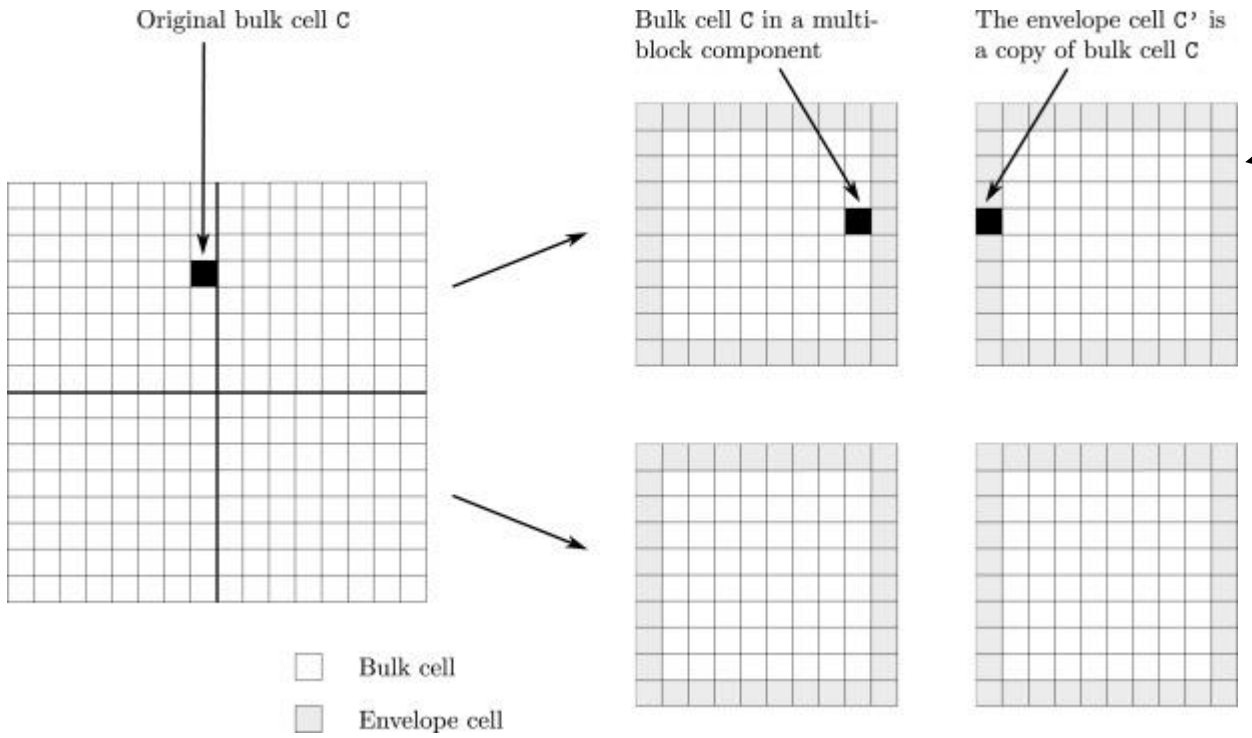
```
template<typename T, template<typename U> class Descriptor>
void ShanChenSingleComponentNoMomCorrProcessor2D<T,Descriptor>::process (
        Box2D domain, BlockLattice2D<T,Descriptor>& lattice )
{
    // ...

    // Compute the interparticle forces, and store them in the external force field.
    for (plint iX=domain.x0; iX<=domain.x1; ++iX) {
        for (plint iY=domain.y0; iY<=domain.y1; ++iY) {
            Array<T,D::d> rhoContribution;
            rhoContribution.resetToZero();
            // Compute the term \sum_i ( t_i psi(x+c_i,t) c_i )
            for (plint iPop = 0; iPop < D::q; ++iPop) {
                plint nextX = iX + D::c[iPop][0];
                plint nextY = iY + D::c[iPop][1];
                T psi = psiField.get(nextX-offsetX, nextY-offsetY);
                for (int iD = 0; iD < D::d; ++iD) {
                    // Access the neighbors to add a contribution to the interparticle force
                }
            }
            // Write the force into the cell variables reserved for this purpose.
        }
    }
}
```

Palabos executes the data processor multiple times, with different domain arguments

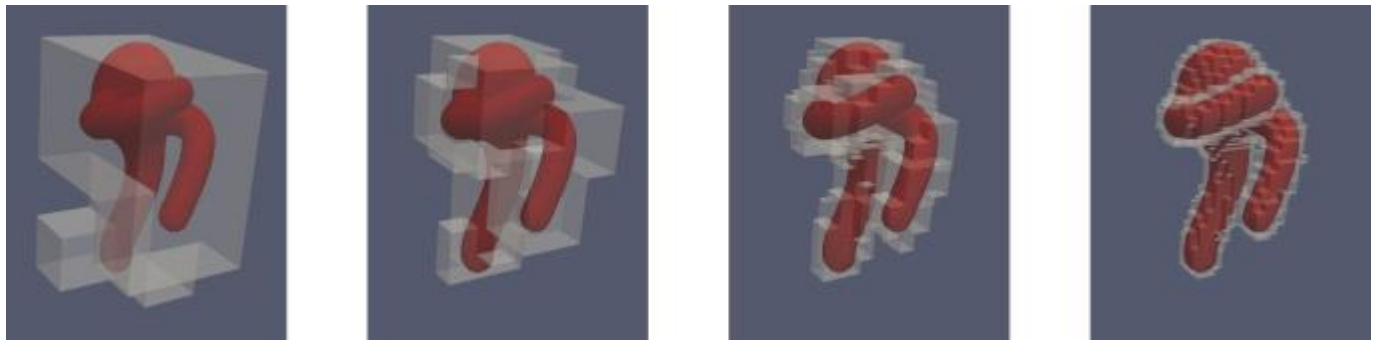
Data processor: a call-back function for Palabos

Data processors: principles of domain decomposition



This domain is provided as an argument to the data-processor callback.

Domain decomposition also works in 3D and can be sparse



Different block sizes which may be chosen to cover the simulation domain, and to decompose the main loop into parallelized piece-wise callbacks to data processors.

Questions ?

Dynamics classes: defining a collision model

```
template<typename T, template<typename U> class Descriptor>
void BGKdynamics<T,Descriptor>::collide (
        Cell<T,Descriptor>& cell, BlockStatistics& statistics )
{
    // ...
    for (int i = 0; i < Descriptor<T>::numPop; ++i) {
        cell[i] = (1 - omega) * cell[i] + omega * equilibrium(cell, i);
    }
    // ...
}
```

- Dynamics classes also define call-back functions
- These functions are however applied to a single cell only
- They are efficiently integrated into an overall collision-streaming cycle

Dynamics classes: add a force term

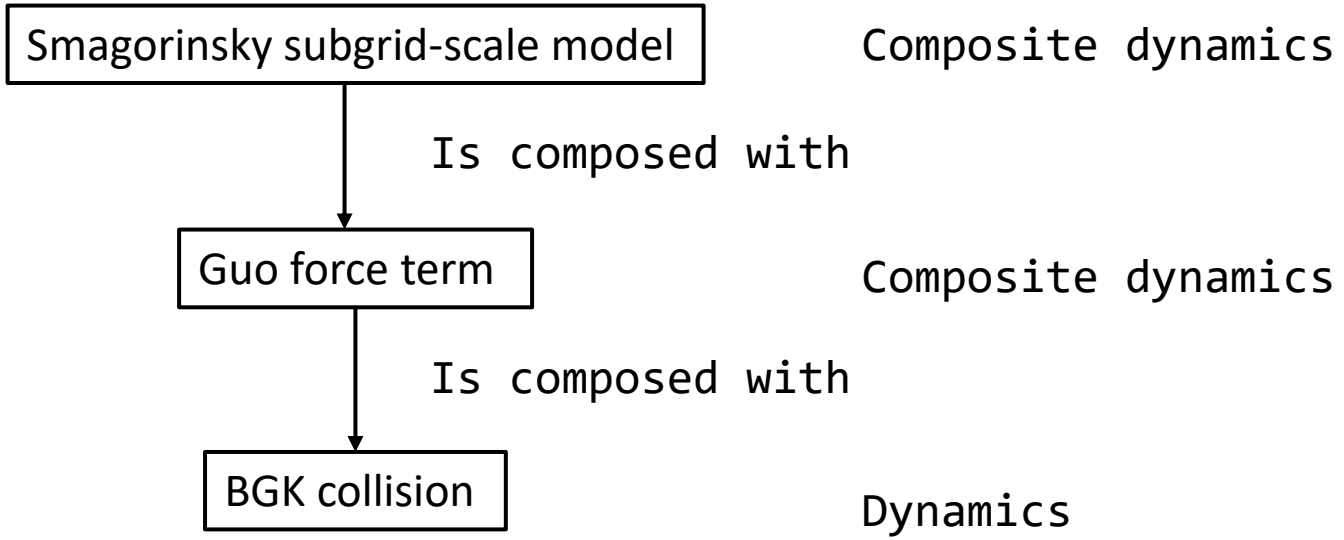
```
template<typename T, template<typename U> class Descriptor>
void BGKdynamics<T,Descriptor>::collide (
        Cell<T,Descriptor>& cell, BlockStatistics& statistics )
{
    // ...
    for (int i = 0; i < Descriptor<T>::numPop; ++i) {
        cell[i] = (1 - omega) * cell[i] + omega * equilibrium(cell, i);
    }
    force.from_cArray(cell.getExternal(Descriptor<T>::ExternalField::forceBeginsAt));
    externalForceTemplates<T,Descriptor>::addGuoForce(cell, …);
}
```

The lattice descriptor tells us in which variables the external force is stored.

Predefined Palabos code exists to implement the Guo force term.

Composite dynamics: chained collision terms

Example: BGK collision model with force term an subgrid-scale model



Advantage: each ingredient can be chained with different models

Back to our example: external force with many different collision models

```
template<typename T, template<typename U> class Descriptor>
void GuoCompositeDynamics<T,Descriptor>::collideExternal (
        Cell<T,Descriptor>& cell, T rhoBar,
        Array<T,Descriptor<T>::d> const& j, T thetaBar, BlockStatistics& stat )
{
    // … Compute macroscopic variables

    // Call base dynamics to execute collision model (with modified velocity, though).
    baseDynamics -> collideExternal(cell, rhoBar, j, thetaBar, stat);

    // Add external force term
    externalForceTemplates<T,Descriptor>::addGuoForce(cell, j, this->getOmega());
}
```

Questions ?