TECHNICAL UNIVERSITY MUNICH

PROJECT DOCUMENTATION

GPU-Accelerated Multi-Species Simulation

Group D
Dominik Huber
Josef Stumpfegger
Sooraj Raj

Advisor
Gerasimos Chourdakis

Code submission: https://gitlab.lrz.de/cfdlabgroupd/cfdlabcodeskeleton

Master's Lab-Course: CFD

July 14, 2020

1. Project Setup and Requirements

1.1 Requirements

For compilation:

- Installed Vulkan-SDK (*https://www.lunarg.com/vulkan-sdk/*)

For GPU usage:

- Vulkan supporting device

1.2 Build and Run

The program can either be run as multi-threaded CPU version or as GPU version. To run the CPU version specify in the .dat file **GpuIndex as negative** and iproc, jproc accordingly, then use the command:

mpi_exec -n [num_threads] sim [problem_name]

To use GPU-acceleration, in the according .dat file the **GpuIndex has to be set to the index of the GPU to be used**. Then use the command

./sim [problem_name]

Currently the following problem_names can be used: *death_valley, indoor_fire, chimney_small* or create your own test cases.

2. Project Documentation

We developed a GPU-accelerated multi-species simulation for 2-dimensional domains.

Our implementation is based on the CFD-Lab code skeleton and supports all simulation cases from the course. The main features that were built on top of the skeleton are listed in section 2.1.

The application is capable of simulating incompressible multi-species flow, insertion of specific species into our domain (e.g. chimneys) as well as conversions between species (e.g. CO2 to O2 conversion by trees). To enable the simulation on large physical domains, we implemented a GPU-accelerated version of the code which allows us to use grids with up to few million cells which would be infeasible with the CPU-only version.

2.1 Features

- Simulation of incompressible, non-separating multi-species flow
- Up to four different multi-species inserter types
- Different multi-species-converter types
- Constant and linear conversion rates
- Significant speedup through GPU-acceleration

2.2 Physical Modeling

Species Concentrations

For each species the concentration is defined on the grid. We use one velocity field for our domain and model the species transport by convection due to the velocity and diffusion from the concentration gradients:

$$\frac{\partial c}{\partial t} + \frac{\partial}{\partial x_i}(u_i c) = \frac{\partial}{\partial x_i}(D\frac{\partial c}{\partial x_i}) + S$$

where c is the concentration, u the velocity, D the diffusion coefficient and S a source term [1].

Species Insertion

For the insertion of species we use a special kind of inflow boundary condition which enforces the defined species concentrations, temperature and velocity at the inserter cell.

Species Conversion

The species conversion is described by reductions and additions of mass of certain species. The defining properties of the conversion process are the rate of reduction of each species ($\frac{g}{m^3 s}$) and the portions of these quantities to be converted into other species. Further, the conversion process can produce heating/cooling effects.

2.3 Implementation Details

2.3.1 Overall program flow

For the CPU-only implementation our main loop follows the pattern of the code skeleton. The species insertion and temperature change due to conversion is done by setting boundary values in:

- setAdiabaticMultispecies(SimulationContext& sim_ctx, Grid& grid, **int** i, **int** j)
- set_conversion_temp(**int** model, SimulationContext &sim_ctx, Grid &grid, **int** converter_index, **int** i, **int** j)

Further, we added the species transport, species conversion and concentration normalization between the temperature- and F/G-calculations. The according functions are implemented in the uvp.cpp file:

- **void** calculate_matrix(simulationContext& sim_ctx, Grid& grid, matrix2d<**double**>& M)
- **void** normalize_species_ratios(SimulationContext& sim_ctx, Grid& grid)
- **void** convert_species(SimulationContext& sim_ctx, Grid& grid, **int** i, **int** j, **int** c)

Information about the GPU-accelerated version is given in Sec. 2.3.4.

2.3.2 Encoding of domain

To support multiple species the domain specifications in the .dat and .pgm files had to be adjusted. In the pgm file inserters are encoded by numbers 5-9 and converters by numbers >= 10. The properties of the species, inserters and converters are specified in the .dat file as shown in the table below. These values are then stored in according structs to be used during the simulation.

Species	[species name]	[molar mass]	$[c_{init}]$	
MultiSpeciesInserter	$[[c_1]$ ... $[c_n]]$	[temp]	$[u]$	$[v]$
MultiSpeciesConverter	$[[r_1]$ ... $[r_n]]$	$[[[a_{1,1}]$ ... $[a_{1,n}]]$ ... $[[a_{n,1}]$ ... $[a_{n,n}]]]$	[temp]	

TABLE 1: **Specification of species, inserters and converters in the .dat file.** C_{init} is the initial concentration of the species in the fluid cells. C_i refers to the concentration of species i enforced by the inserter, where i refers to the i-th species defined in the .dat file. Symbol r_i refers to the reduction mass of species i defined as $\frac{g}{m^3 s}$. Symbol $a_{i,j}$ refers to the fraction of the reduced mass from species j that is added to species i. Temperature can be defined if conversion produces temporal effects, or should be *nan* otherwise.

2.3.3 Multi-species Inserter

The inserters are handled as inflow boundary cells. We set the species concentrations, temperature and velocity according to the values specified in the .dat file.

2.3.4 Multi-species Converter

The multi-species converter cells are generally treated as fluid cells. The conversion is done after the species transport during normalization of the concentrations. We first calculate the current masses of the different species in the converter cell as:

$$m_{species_i} = c_i \times molarMass_i \times p_{init} \times \frac{1}{const_{gas} * T} \times dx \times dy$$

Then the specified amount of the species (r_i) is subtracted. R_i can be adjusted by plugging it into specified functions of e.g concentration or temperature and is of course scaled according to size of the cell and the time step size. The specified fractions of these reduced masses are then added to the according species masses. Finally the concentrations are recalculated with the above function and normalized to sum up to 1.

2.3.5 GPU implementation

To reduce the exection time, especially for larger problems, we use an implementation for graphics cards using the graphics API *Vulkan*. Every method (calc_dt , etc.) is a pipeline. Every matrix (velocity, temperature, concentrations etc.) are treated as 2d images. The pipelines are recorded into a *CommandBuffer*, in the order as they are executed in the CPU implementation (see Fig. 1). Some difficulties had to be tackled for instance conflicting read-/write operations on 1 cell:

- **Boundary values:** each cell only writes to itself and only reads from neighboring cells,
- **Pressure calculation:** Use of checkerboard pattern,

or the conditional loops as the sor iteration and global min/max calculation. Finally, we achieve speedups up to 10× compared to the CPU implementation fo large grids..

3. Investigated Test Cases

We investigated two specific test cases specified in the .dat and .pgm files named *indoor_fire* and *death_valley*. For these cases we used special conversion models which are currently hardcoded and only used for these simulations. We tried to use values as realistic as possible.

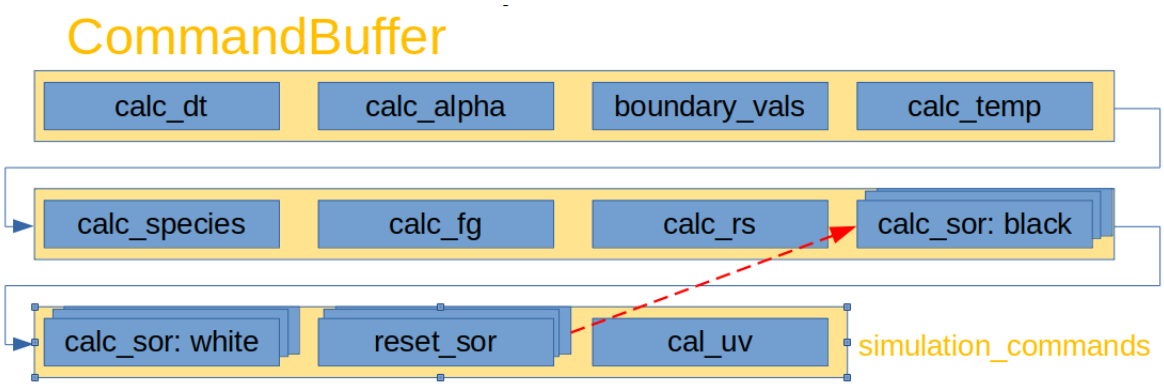
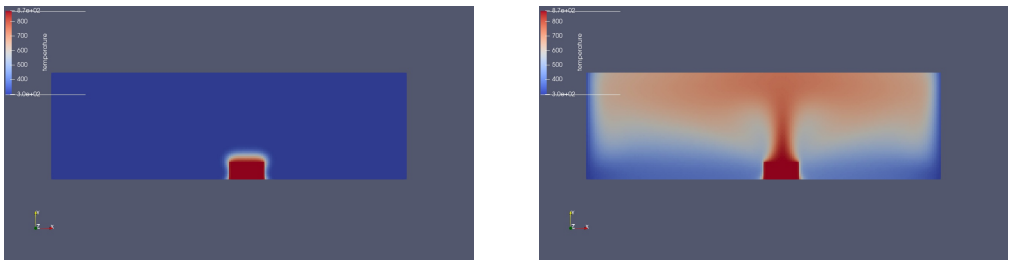


FIGURE 1: **Command Buffer** Program structure of GPU-accelerated version. Each method is a single pipeline.



(A) label 1 (B) label 2

FIGURE 2: **Results for *indoor_fire* simulation (temperature)**. Left: after $\approx$ 19 seconds, right after 1000 seconds. O2 concentration is still above 16% so the fire is still burning

3.1 CO2 exposure from indoor fire place

The test case *indoor_fire* simulates a fire inside a closed domain. The fire is modeled by a species converter, converting O2 to CO2, thereby producing heat. This conversion does only occur as long as O2 concentration in the cell is above 16 percent. Figure 2 shows the results.

3.2 Influence of tree planting on CO2 pollution from power stations

The test case *death_valley* simulates a coal-fired power station, which is located on a hill and produces CO2. For this we use a species inserter. In our domain we have a constant wind from left to right. On the right slope of the hill several trees are modeled by species converters, converting CO2 to O2. On the right side of the domain some "houses" are located (small village). With this setup we want to evaluate the influence of tree planting on the CO2 exposure in the village. For the conversion rate we use a linear function for concentrations between 0.1% and 1.2% (based on [2]). Figure 3 shows the result after 100 seconds with trees. Unfortunately the trees are not able to compensate the CO2 much.

4. Limitations and possible extensions

Although our application is powerful in simulating simple multi-species flow, there are amongst others the following limitations:

- Only 2 dimensional domains

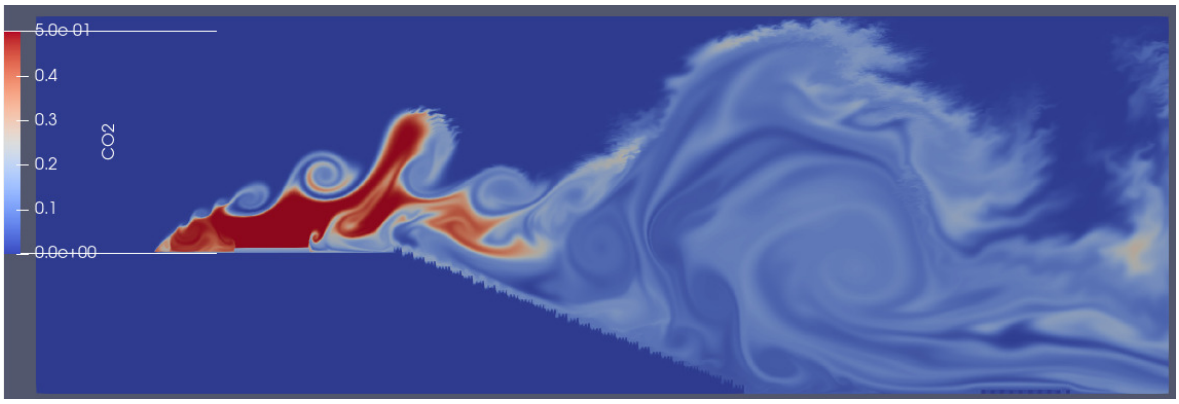


FIGURE 3: Death Valley simulation with trees after 100 seconds. Overall simulation duration with GPU-acceleration was 2.25 hours.

- No separation of species
- Only simple model of species diffusion and conversion
- Incompressability

Possible extensions of our code could include:

- User specified conversion functions
- User specified diffusion models

5. Bibliography

[1] Bakker, A.: Modeling Material/Species Flow reacting flows - lecture 8. http://www.bakker.org/dartmouth06/engs199/08-spcs.pdf (2006), accessed: 2020-07-13

[2] Luettge, U.: Botanik - Die einführende Biologie der Pflanzen. Wiley-VCH, 6. aktualisierte auflage edn. (2012)