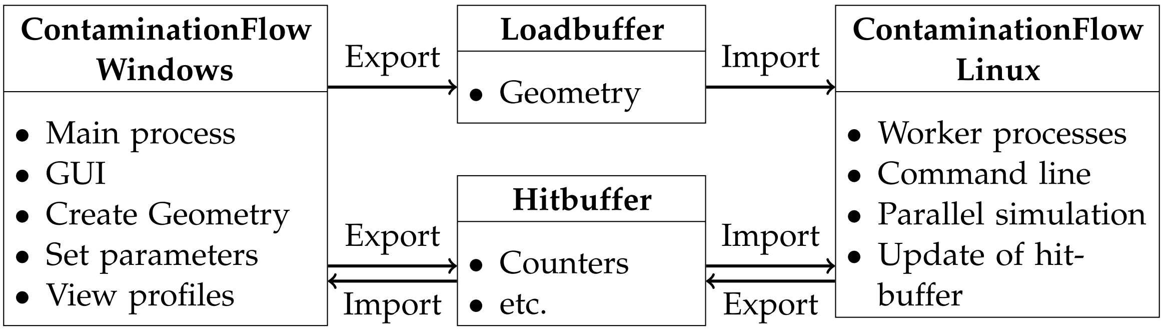
Documentation

ContaminationFlow
on Linux and Windows

Hoai My Van

Contents

1. General Structure

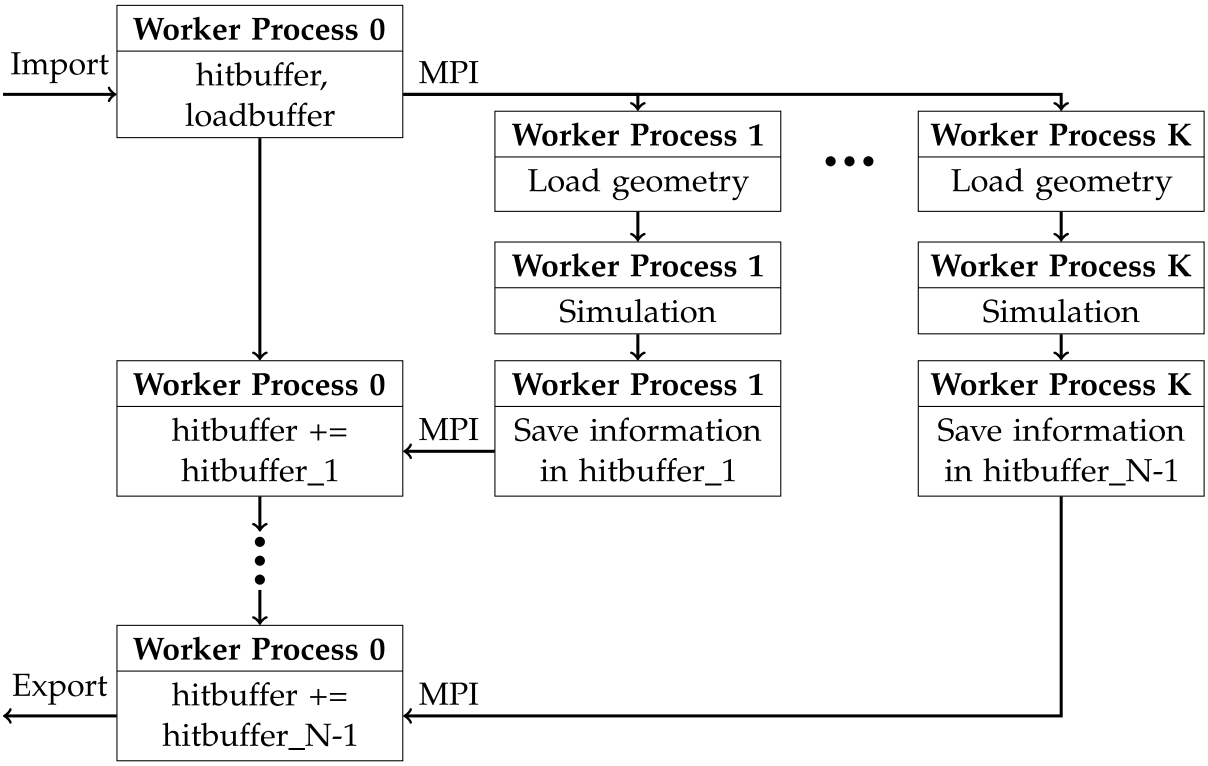


General structure for ContaminationFlow simulation

- Code adapted from Molflow
- ContaminationFlow Windows used to create Geometry
- ContaminationFlow Linux used for simulation and calculation of counters, profiles, etc.
- Loadbuffer contains information of geometry
- Hitbuffer contains information such as hit counters, profiles, etc.
- Import and export of buffer files for communication between Molflow Windows and Molflow Linux
- Import and export of covering history for both linux and windows

2. ContaminationFlow Linux

- Parallel simulation on several worker processes
- Update and accumulation of hit counters and other information such as profiles



2.1. Call of Application from Command line

Commands to call the Molflow Linux application in the command line:

```
$ module load mpi
$ mpirun -n N MolflowLinux loadbuffer hitbuffer resultbuffer
simulationtime unit
```

with the following parameters:

- N: desired number of worker processes; simulation on K=N-1 worker processes
- MolflowLinux: path to application, e.g. `~/MolflowLinux/Debug/MolflowLinux`
- loadbuffer: path to loadbuffer file, that contains geometry, e.g. `~/loadbuffer`
- hitbuffer: path to hitbuffer file, that contains counters, etc., e.g. `~/hitbuffer`
- resultbuffer: path to resultbuffer file, where the final hitbuffer is exported to, e.g. `~/resultbuffer`
- simulationtime: floatingpoint number, simulation time, e.g. `2.5`
- unit (optional): simulation time unit, e.g. `min`; default: `s`

2.2. Application

2.2.1. General Changes

Replacement/removal of Windows libraries/functions

- E.g. Databuff struct with import/export instead of using Dataports
- E.g. replace `#include <time.h>` with `#include <sys/time.h>`

Removal of functions used in `AC_MODE`

- Only `MC_MODE` used
- Removal of `AC_MODE` cases and functions

2.2.2. Communication

Import and export of buffer files

- New Databuff struct

```
typedef unsigned char BYTE;
typedef struct {
  signed int size;
  BYTE *buff;
} Databuff;
```

- New functions `importBuff(·)` and `exportBuff(·)` for import of buffer files and export of Databuff struct

Communication between worker processes via MPI

- Process 0 sends load Databuff struct and hit Databuff struct to other processes using `MPI_Bcast(·)`
- All processes send updated hit Databuff struct to process 0 using `MPI_Send(·)` and `MPI_Recv(·)`

2.2.3. New Quantities

New counter `covering`

- Covering computed in `Simulationcalc.cpp` file in `calcCoveringUpdate(·)`
- Covering increases with adsorption, decreases with desorption
- Added covering counter to hitbuffer

Sticking factor

- Dependent on covering and temperature
- Sticking factor computed in `Simulationcalc.cpp` file in `calcStickingnew(·)`
- Updated after/before each interation

Desorption

- Dependent on covering and temperature
- Desorption computed in `Simulationcalc.cpp` file in `calcDesorption(·)`
- Used to determine starting point for new particle

`Worker` **class for Worker processes**

- Reduced `Worker` class
- Only use of functions `GetCDFId(·)`, `GenerateNewCDF(·)`, `Generate_CDF(·)`, `CalcTotalOutgassing()`

2.2.4. Iterative algorithm

Serialization of Simulation on worker processes

- `InitSimulation()` to create simulation handle
- `LoadSimulation()` to load geometry in simulation handle
- `StartSimulation()` to create first particle for simulation
- `StartFronSource()` for initial values for new particle, adapted to include desorption rate
- `SimulationRun()` repeatedly for desired time step (default: $1s$) until desired simulation time reached, simulates particle at a time until it desorbs or adsorbs, saves information of hits in simulation handle

Update of hitbuffer after simulation finished

- `UpdateSubHits(·)` and `UpdateSubMCHits(·)` to save information from simulation handle into hit Databuff struct (no accumulation here)
- Process 0 adds hit Databuffs struct from subprocesses to original hit Databuff struct using `UpdateMainHits(·)` and `UpdateMCmainHits(·)`

Estimation of T_{min}

- Determines minimum timestep for simulation, average time between outgassing/desorption and adsorption
- T_{min} computed in `Iteration.cpp` file in `EstimateTmin()`

New class to store covering for all facets at any time

- TimeTest class

```
class TimeTest {
public:
  TimeTest();
  std::vector< std::pair< double,std::vector<double> > >
pointintime_list;

  void appendList(double time);
  void print();
  void write(std::string filename);
  void read(std::string filename);
};
```

- In `SimulationLinux.h` and `Iteration.cpp` file
- After each simulation step, list is appended with point in time and covering for all facets
- Used for extrapolation in future

3. ContaminationFlow Windows

- Create Geometry and set parameters such as pumping speed or sticking
- Evaluate profiles such as pressure profile
- Simulation also possible for testing, but mostly done on Linux

3.1. Graphical User Interface

Add screenshot of GUI

3.2. Application

3.2.1. Communication

Import and export of buffer files via GUI

- New Databuff struct

```
typedef unsigned char BYTE;
typedef struct
  signed int size;
  BYTE *buff;
Databuff;
```

- New functions `importBuff(·)` and `exportBuff(·)` for import and export of buffer files/Databuff struct
- New options in file menu: `Export buffer` and `Import buffer`

3.2.2. New Quantities

New counter `covering`

- Covering computed in `SimulationMC.cpp` file in `updatecovering(·)`
- Added covering counter to hitbuffer
- Added covering to GUI, can be defined through textfield

Compute sticking factor based on covering

- Sticking factor computed in `Molflow.cpp` file in `calcStickingnew()`
- Updated automatically whenever covering is changed

Removal of Flow-Sticking dependency

- `calcSticking()` and `calcFlow()` in `Molflow.cpp` file not used anymore
- Flow not needed for iterative Algorithm

3.2.3. Iterative algorithm

New class to store covering for all facets at any time

- In `HistoryWin.cpp` and `HistoryWin.h` file
- `std::vector< std::pair< double,std::vector<double> > > pointintime_list` to store points in time and respective covering for all facets
- New GUI option to add and remove entries for `pointintime_list`
- New GUI option to export or import a complete list

A. Formulas for new Quantities

Covering θ

$$\begin{aligned}
\Delta N_{surf} &= \frac{m}{12.011} \\
N_{surf} &= \sum_{\substack{\text{adsorbed}\\\text{particles}}} \Delta N_{surf} \\
N_{mono} &= \frac{\text{Area of Facet}[\mathrm{m}^2]}{(76 \cdot 10^{-12}\mathrm{m})^2} \\
\theta &= \frac{N_{surf}}{N_{mono}}
\end{aligned} \tag{A.1}$$

Sticking factor sc

$$sc(\theta) = \begin{cases} (s_1(1-\theta) + s_2\theta) \cdot (1 - \exp(-\frac{E_{ad}}{K_b T})), & \text{if } \theta < 1 \\ s_2(1 - \exp(-\frac{E_{ad}}{K_b T})), & \text{otherwise.} \end{cases} \tag{A.2}$$

Desorption rate des

$$des = \frac{1}{\tau}\, \theta^d \; \exp(-\frac{E_{de}}{K_b T}) \cdot \frac{N_{mono}}{\Delta N_{surf}}. \tag{A.3}$$