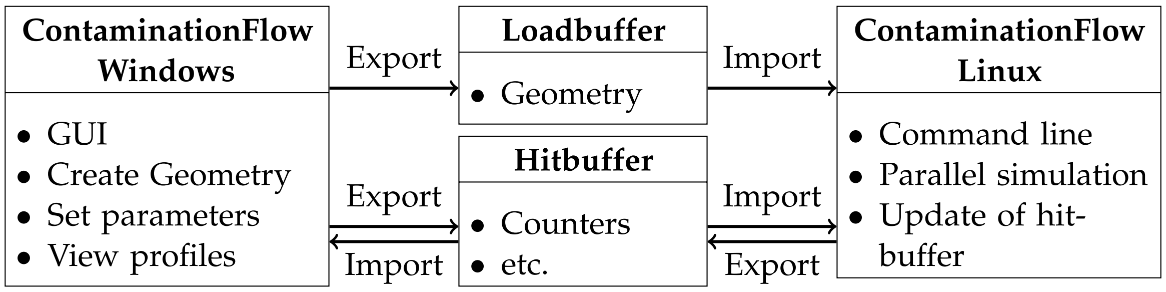
Documentation

ContaminationFlow on Linux and Windows

Hoai My Van

Contents

1. General Structure



General structure for ContaminationFlow simulation

- Code adapted from Molflow
- ContaminationFlow Windows primary used to create Geometry and to view simulation results through the GUI
- ContaminationFlow Linux primary 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 ContaminationFlow Windows and ContaminationFlow Linux
- Export of simulationHistory for ContaminationFlow Linux

2. ContaminationFlow Linux

- Parallel simulation on several sub processes
- Processing and control of data in main process
- Update and accumulation of hit counters and other information such as profiles
- SimulationHistory, final hitbuffer and used parameters exported to results folder

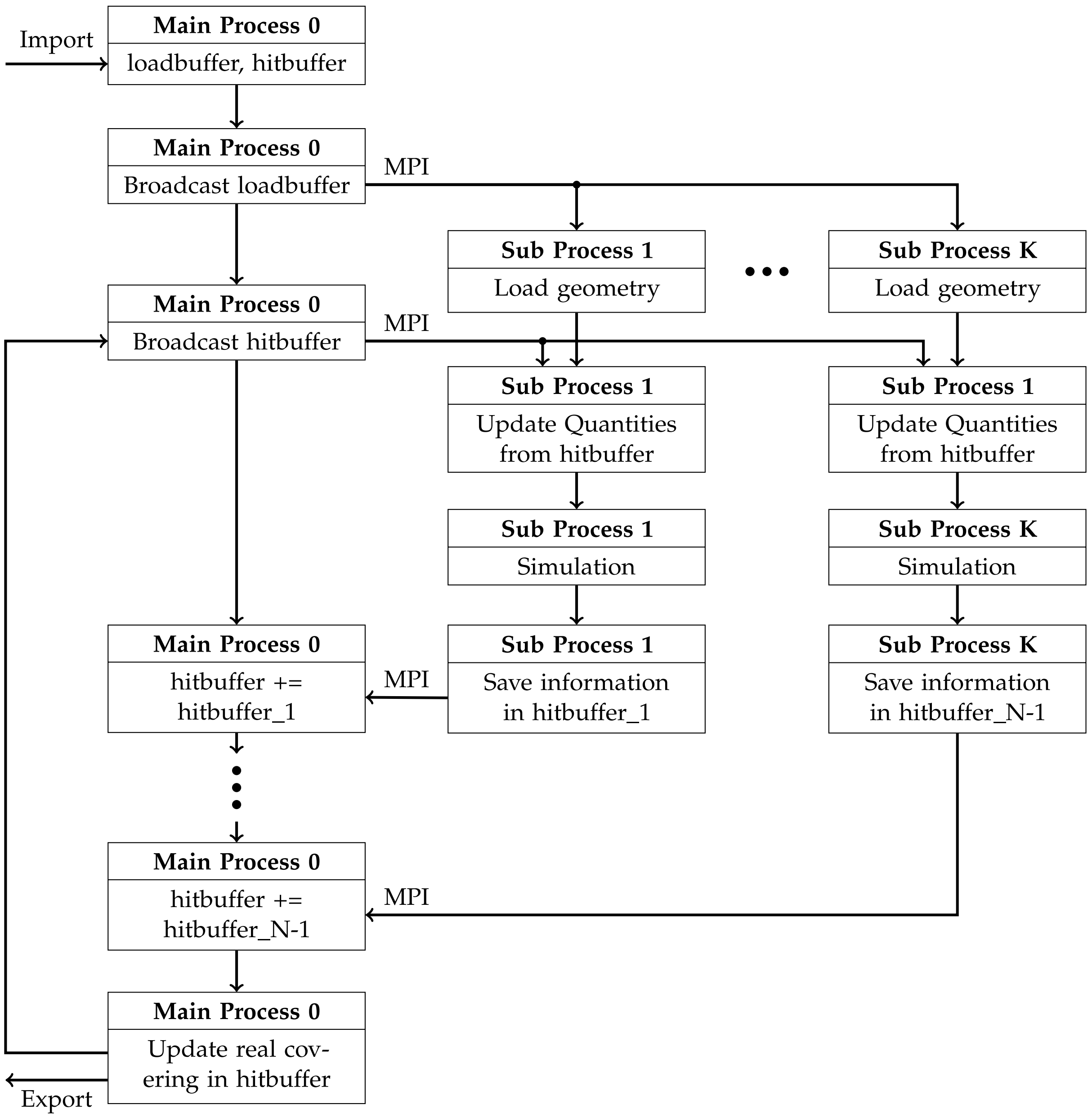


Figure 2.1.: Processing of data in main and sub processes

2.1. Call of Application from Command line

New class ProblemDef

- Defines parameters used for simulation
- Possible adaptation of default paramaters through input file
- Conducts some intern conversions
- Creates result folder for simulation if desired
  - Final resultbuffer
  - Final covering, input file and console output as text files

Application with custom parameters using input file

Call of ContaminationFlow Linux application in the command line:

```
$ module load mpi
$ mpirun -n N MolflowLinux inputfile save
```

with the following command line parameters:

- `N` : desired number of worker processes; simulation on K=N-1 worker processes
- `MolflowLinux` : path to application, e.g. `~/MolflowLinux/Debug/MolflowLinux`
- `inputfile` : path to file that defines simulation parameters
- `save` : determines whether result directory is created (1: true, 0:false); default: `1`

and the input file defining the following parameters:

- $loadbufferPath$: path to loadbuffer file, contains geometry, e.g. `~/loadbuffer`
- $hitbufferPath$: path to hitbuffer file, contains counters, etc., e.g. `~/hitbuffer`
- $simulationTime$: simulation time per iteration step; default: `10.0`
- $unit$: simulation time unit; default: `s`
- $maxTime$: maximum simulation time; default: `10.0`
- $maxUnit$: maximum simulation time unit; default: `y`
- $iterationNumber$: number of iterations; default: 43200
- d : power for base of coverage used for calculation of desorption; default: `1`
- E_{de} : maximum energy used for calculation of desorption; default: `1E-21`
- H_{vap} : minimum energy used for calculation of desorption; default: `1E-21`
- W_{tr} : window width used for calculation of desorption; default: `1E-21`
- $sticking$: constant sticking coefficient for all facets; default: `0.8E-19`
- $coveringLimit$: threshold for zero desoprtion; default: `0`
- $targetPaticles$: Minimum number of desorbed particles per iteration; default: `1000`
- $targetError$: Maximum error per iteration; default: `0.001`
- $hitRatioLimit$: Ratio at which hits are ignored, default: `1E-5`
- $Tmin$: minimum time for step size; default: `1E-4`

Terminology

- Simulation time: desired computation time until check if target is reached for iteration
- Maximum simulation time: desired total simulated time
- Step size: desired simulated time per particle for iteration

2.2. Communication

Import and export of buffer files

- New Databuff struct that replaces Dataport struct from MolFlow Windows

  ```
  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
- New functions `checkReadable(·)` and `checkWriteable(·)` to check if file is readable or writeable

Communication between worker processes via MPI

- Main process 0 sends Databuff struct containing loadbuffer and Databuff struct containing hitbuffer and required simulationHistory values to sub processes using `MPI_Bcast(·)`
- Sub processes send updated Databuff struct containing hitbuffer and required simulationHistory values to main process 0 using `MPI_Send(·)` and `MPI_Recv(·)`

2.3. Usage of *boost* Library

Multiprecision

- Increase precision for variables if required
- Avoid underflow for integer and underflow for floating point numbers

2.4. New Quantities

New counter `covering`

- Number of carbon equivalent particles on facet
- Added covering counter to hitbuffer
- Extracted from hitbuffer from `Simulationcalc.cpp` file in `getCovering(·)`
- Covering increases with adsorption, decreases with desorption

Coverage

- Number of carbon equivalent particles per monolayer on facet
- Calculated from covering, gas mass and facet area
- Coverage computed from `Simulationcalc.cpp` file in `calcCoverage(·)`

Sticking factor

- Ratio adsorbed particles to impinging particles
- Set to 0, can be adapted for all facets through input file

Binding energy

- Calculated from E_{de}, H_{vap} and W_{tr}
- Desorption computed from `Simulationcalc.cpp` file in `calcEnergy(·)`

Desorption [1/s]

- Calculated from binding energy, covering and temperature
- Desorption computed from `Simulationcalc.cpp` file in `calcDesorption(·)`

Desorption Rate [**Pa m^3/s**]

- Calculated from desorption, gas mass and facet area
- Desorption rate computed from `Simulationcalc.cpp` file in `calcDesorptionRate(·)`
- Used to determine starting point for new particle
- Updated and fixed before each interation

Outgassing Rate

- Calculated from facet outgassing and temerature defined in sHandle
- Outgassing rate computed from `Worker.cpp` file in `CalcTotalOutgassingWorker()`

K$_{\text{real/virtual}}$

- Number of real particles represented by test particles
- Calculated from desorption rate, outgassing rate and number of desorbed molecules
- K$_{\text{real/virtual}}$ computed from `Simulationcalc.cpp` file in `GetMoleculesPerTP(·)`

2.5. Iterative Algorithm

2.5.1. Initialization of simulation

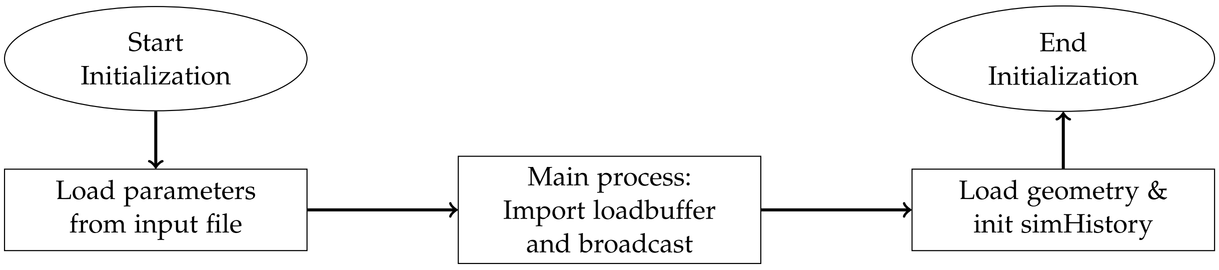

Start Initialization
Load parameters from input file
Main process: Import loadbuffer and broadcast
Load geometry & init simHistory
End Initialization

Figure 2.2.: Overview: Initialize simulation

New class to store Simulation History

- SimulationHistory class

```
class SimulationHistory {
public:
 SimulationHistory();
 SimulationHistory(Databuff *hitbuffer);

 HistoryList<llong> coveringList;
 HistoryList<llong> desorbedList;
 HistoryList<double> hitList;
 HistoryList<double> errorList;

 double lastTime;
 int currentStep;
};
```

```
template <typename T> class HistoryList {
public:
 HistoryList();
 std::vector<std::pair<double,std::vector<T>>> pointInTimeList;
 std::vector<T> currentList;
};
```

- In `SimulationLinux.h` and `SimulationLinux.cpp` file
- SimulationHistory updated after each iteration in `UpdateCovering(·)` from `UpdateMainProcess.cpp` file
- Currently recorded quantities: covering and error for each facet for each iteration, total hits and desorbed particles for each facet
- Time: simulated time (accumulated time steps) instead of computation time

2.5.2. Simulation on subprocesses

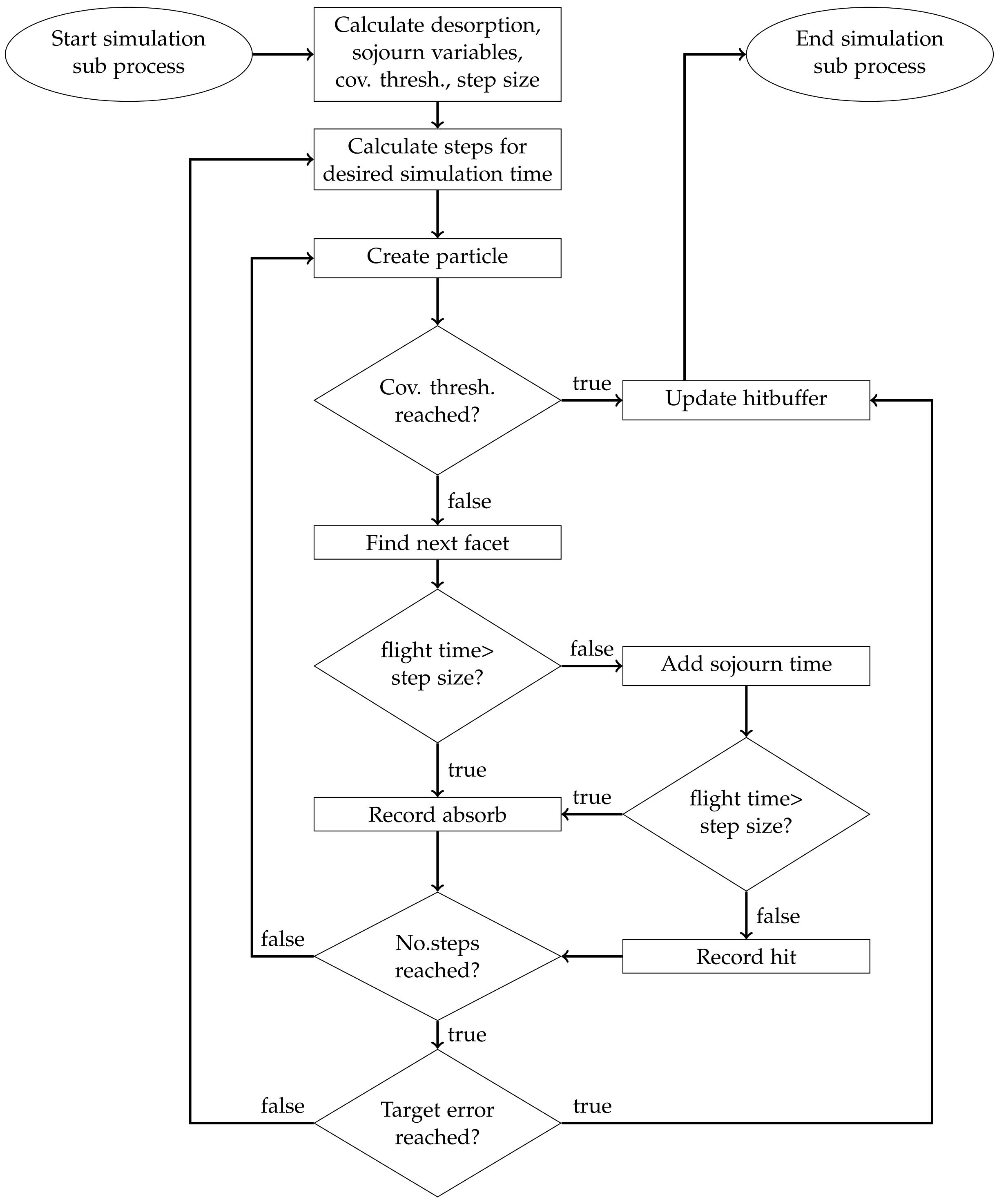


Figure 2.3.: Overview: simulation on sub processes

Set Covering Threshold

- Set lower threshold for covering for each facet to prevent covering getting negative
- Stop simulation once threshold is reached
- Threshold set in `setCoveringThreshold(·)` from `Iteration.cpp` file

Sojourn

- Sojourn time of bounce calculated from energy and frequency
- Sojourn energy equal to binding energy
- Sojourn frequency calculated from temperature

2.6. Update main buffer

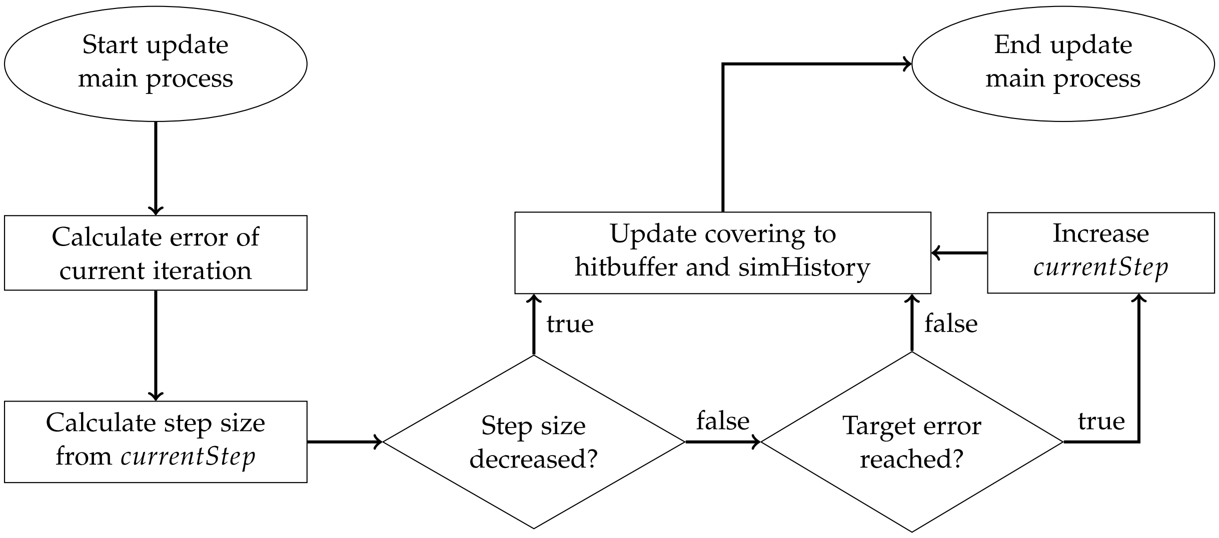

Start update main process
Calculate error of current iteration
Calculate step size from *currentStep*
Step size decreased?
true
Update covering to hitbuffer and simHistory
End update main process
false
Target error reached?
false
true
Increase *currentStep*

Figure 2.4.: Overview: update of covering in hitbuffer

Error Calculation

- Calculate statistical error
- Facet error calculated from hits of facet and total hits, hits weighted with opacity of facet
- Total error calculated from summing facet error weighted with facet area

Calculate Step Size

- Use `SimulationHistory::currentStep` to calculate logarithmic step size
- Duration between outgassing/desorption and adsorption

Management of Step Size

- Check whether current step size would cause desorption to be larger than covering
- Adapt step size if required
- Increment `SimulationHistory::currentStep` if step size not decreased and `ProblemDef::targetError` reached
- Management in `UpdateMainProcess.cpp` file in `manageStepSize(·)`

2.7. Summary

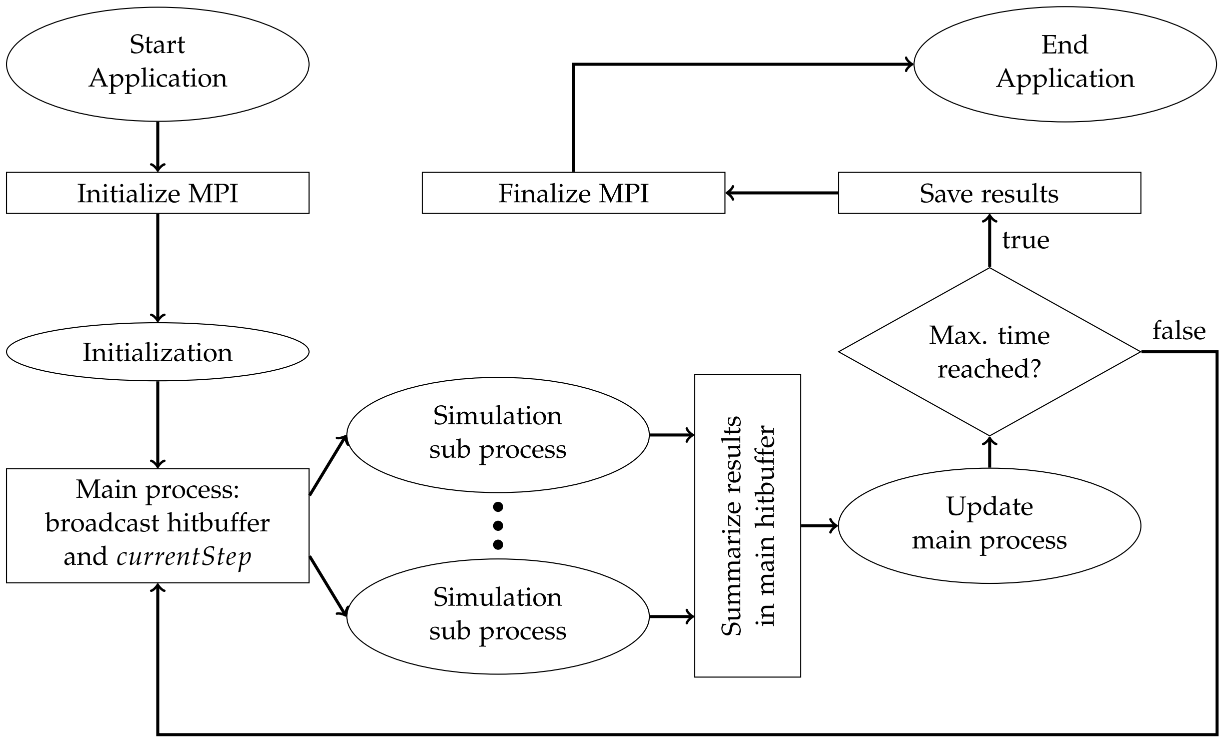


Figure 2.5.: Overview: ContaminationFlow application

General Pipeline

- Initialize MPI, `ProblemDef p` and `SimulationHistory simHistory`
- Load geometry into `Simulation sHandle` using `LoadSimulation()`
- Iteration until desired maximum simulation time is reached:
    - Reset hitbuffer counters using `initbufftotero(·)`, broadcast using `MPI_Bcast(·)`
    - Simulation in sub processes
        - Simulate until `targetParticles` and `targetError` or `covthresh` reached
        - Update hitbuffer of sub processes from `sHandle` using `UpdateSubHits(·)` from `UpdateSubProcess.cpp`
    - Update Main process:
        - Send hitbuffer to main process using `MPI_Send(·)` and `MPI_Recv(·)`
        - Update of hitbuffer in `UpdateMainHits(·)` from `UpdateMainProcess.cpp`
    - Update error of iteration using `UpdateErrorMain(·)` from `UpdateMainProcess.cpp`
    - Calculate real covering in main process using $K_{real/virtual}$ and simulated step size in `UpdateCovering(·)` from `UpdateMainProcess.cpp`, save in `simHistory`
    - Update real covering in hitbuffer of main process in `UpdateCoveringphys(·)` from `UpdateMainProcess.cpp`
- Export final results (hitbuffer and simulationHistory) to results folder
- Close MPI

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

New GUI elements

- "Particles out" renamed to Contamination level
  - Text field for covering
  - Text field for coverage
- New facet properties
  - Effective surface factor
  - Facet depth and facet volume
  - Diffusion coefficient
  - Concentration and gas mass
- Window for CoveringHistory (reworked to SimulationHistory in Contamination-Flow Linux)
- PressureEvolution window expanded
  - Added list that contains information of graph
  - Option to show only selected facets or all
  - List exportable

3.2. 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`

3.3. 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

New facet property `effetiveSurfaceFactor`

- Defines increase of facet area due to texture

New facet property `facetDepth`

- Defines depth of facet

New facet property `diffusionCoefficient`

- Defines diffusion coefficient

New facet property `concentration`

- Defines concentration = mass of particles in volume

Removal of irrelevant quantities

- Sticking factor and pumping speed removed from GUI
- `calcSticking()` and `calcFlow()` in `Molflow.cpp` file not used anymore
- Flow not needed for iterative Algorithm

3.4. 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

Constants

$$\begin{aligned} carbondiameter &= 2 \cdot 76\text{E} - 12 \\ K_b &= 1.38\text{E} - 23 \\ h &= 6.626\text{E} - 34 \end{aligned} \tag{A.1}$$

Number of carbon equivalent particles of one monolayer

$$N_{mono} = \frac{\text{Area of Facet } [\text{m}^2]}{carbondiameter^2} \tag{A.2}$$

Carbon equivalent relative mass factor

$$\Delta N_{surf} = \frac{\text{carbon equivalent gas mass}}{12.011} \tag{A.3}$$

Covering θ^*

$$\theta^* = N_{\text{particles on facet}} \tag{A.4}$$

Coverage θ

$$\theta = \frac{\theta^*}{N_{mono}/\Delta N_{surf}} \tag{A.5}$$

Energy E

$$E = \frac{E_{de} - H_{vap}}{2} \cdot \tanh\Big((1 - coverage) \cdot \frac{5.4}{W_{tr}}\Big) + \frac{E_{de} + H_{vap}}{2} \tag{A.6}$$

Sojourn

$$\begin{aligned} Frequency &= \frac{K_b T}{h} \\ Energy &= E \end{aligned} \tag{A.7}$$

Desorption rate *des*

$$\begin{gathered} \tau = \frac{h}{K_b T} \\ des = \begin{cases} \frac{1}{\tau}\, \theta^d \;\exp\left(-\frac{E}{K_b T}\right) \cdot \frac{N_{mono}}{\Delta N_{surf}} \cdot K_b T, & \text{if } \theta^* \geq \text{covering limit} \\ 0, & \text{otherwise} \end{cases} \end{gathered} \tag{A.8}$$

Outgassing rate *out*

$$out = \frac{\text{Facet outgassing}}{K_b T} \tag{A.9}$$

K$_{\textbf{real/virtual}}$

$$\mathrm{K}_{\text{real/virtual}} = \frac{\sum\limits_{\text{facets}} \left(out + \frac{des}{K_b T} \right)}{\text{number of total desorbed molecules}} \tag{A.10}$$

Step Size T_{step}

$$\begin{aligned} T_{min} &= \text{Tmin} \\ T_i &= T_{min} \cdot \exp\Big(\text{i} \cdot \ln(\text{max. simulation time}/T_{min})/\text{max. \# of steps})\Big) \\ T_{step} &= T_{currentStep+1} - T_{currentStep} \end{aligned} \tag{A.11}$$

Error

$$error = \left(\frac{1}{\text{(hits + desorbed) on facet}} \cdot \frac{1 - \text{(hits + desorbed) on facet}}{\text{total (hits + desorbed)}} \right)^{0.5} \tag{A.12}$$

B. Overview of new Classes and Functions

B.1. New Classes

SimulationHistory	
coveringList	of class HistoryList, stores covering history
errorList	of class HistoryList, stores error history
hitList	of class HistoryList, stores hits for each facet
desorbedList	of class HistoryList, stores desorbed particles for each facet
numFacet	number of Facets
numSubProcess	number of sub processes used for simulation
nbDesorbed_old	number of total desorbed molecules of previous iteration ⇒ To calculate difference between consecutive iterations
flightTime	number of Facets
nParticles	Simulated flight time for iteration
numFacet	Simulated particles for iteration
lastTime	Total simulated time = last time in Lists
currentStep	step of logarithmic time step calculation in `getStepSize()`
updateHistory()	Reset and update from hitbuffer
appendList()	Updates coveringList from hitbuffer
print()	Print to terminal
write()	Write to file

HistoryList	
pointInTimeList	list containing history respective facet values
currentList	list containing facet values at current step
appendCurrent()	Appends currentList to pointInTimeList
appendList()	Append input list to pointInTimeList
convertTime()	Converts time for better clarity
print()	Print pointInTimeList as table to terminal, optional message
printCurrent()	Print currentList as table to terminal, optional message
write(), read()	Write to file, read from file
setCurrentList()	Set value of desired facet in currentList
getCurrent()	Get value of desired facet from currentList
setLast()	Set value of desired facet from pointInTimeList
getLast()	Get value of desired facet from pointInTimeList

ProblemDef	
resultpath	Path of result folder
outFile	Path of file that contains terminal output
loadbufferPath	Path of loadbuffer file
hitbufferPath	Path of hitbuffer file
simulationTime, unit $\Rightarrow$simulationTimeMS	Computation time of each iteration in milliseconds
maxTime, maxUnit $\Rightarrow$maxTimeS	Maximal total simulated time in seconds
iterationNumber	Number of iterations
d	Parameter to calculate desorption rate, see equation A.8
E_de, H_vap, W_tr	Parameters to calculate binding energy, see equation A.6
sticking	Sticking factor for all facets
coveringLimit	Covering limit for zero desorption
targetParticles/-Error	Target values for each iteration
hitRatioLimit	Limit of hitratio to ignore hits
Tmin	Minimum step size
readInputfile()	Initialization from input file
printInputfile()	Print to terminal

B.2. New Functions

B.2.1. molflowlinux_main.cpp

Preprocessing	
parametercheck()	Checks validity of input parameters from input file Defines values for ProblemDef object `p`
importBuff()	Import load- and hitbuffer to main process
MPI_Bcast()	Send loadbuffer to sub processes
LoadSimulation()	Load geometry from loadbuffer
initCoveringThresh()	Initialize covering threshold
`simHistory`	Initialize SimulationHistory object

Simulation Loop	
initbufftozero()	Reset all hitbuffer counters except covering
MPI_Bcast()	Send hitbuffer and `simHistory→currentStep` to sub processes
setCoveringThreshold()	Sets covering threshold for each facet
UpdateDesorptionRate()	Sets desorption for each facet
UpdateSojourn()	Sets sojourn variables for each facet
simulateSub()	Simulation on sub processes
MPI_Send(), MPI_Recv()	Send sub hitbuffer to main process
UpdateMCMainHits()	Add simulation results from sub hitbuffer to main hitbuffer
UpdateErrorMain()	Calculate and save error of iteration to simHistory
UpdateCovering()	Calculate and save new covering to simHistory
UpdateCoveringphys()	Saves current covering to hitbuffer
End simulation if maximum simulation time is reached	

Postprocessing	
exportBuff()	Export final hitbuffer
simHistory→write()	Export simulation history

B.2.2. SimulationLinux.cpp

simulateSub()	
targetParticles, targetError	Calculate target values from overall target and number sub processes
`simHistory->updateHistory()`	Reset and Update SimulationHistory object from hitbuffer
SimulationRun()	Simulate for desired simulation time
UpdateError()	Calculate current error of sub process End simulation if targets or covthresh reached
UpdateMCSubHits()	Save simulation results to hitbuffer

B.2.3. Calculations in SimulationCalc.cpp etc.

SimulationCalc.cpp	
getCovering()	Get covering from hitbuffer
getHits()	Get number of hits from hitbuffer
getnbDesorbed()	Get number of total desorbed molecules from hitbuffer
calcNmono()	see equation A.2
calcdNsurf()	see equation A.3
calcCoverage()	see equation A.5
calcEnergy()	see equation A.6
calcStickingnew()	sets sticking coefficient to p$\rightarrow$sticking
calcDesorption(), calcDesorptionRate()	see equation A.8
GetMoleculesPerTP()	see equation A.10
calctotalDesorption	calculates desorption for `startFromSource()`
calcPressure(), calcParticleDensity()	TODO has to be verified

worker.cpp	
CalcTotalOutgassingWorker()	see equation A.9, calculates outgassing for `startFromSource()`

SimulationLinux.cpp	
covertunit()	Converts simutime*unit to milliseconds

B.2.4. UpdateSubProcess.cpp

Update sHandle paramters from hitbuffer	
UpdateSticking()	Updates sticking
UpdateDesorptionRate()	Updates desorption rate
UpdateSojourn()	Updates sojourn freqency and energy
UpdateError()	Updates error for current iteration, see equation A.12

Update hitbuffer	
initbufftozero()	Sets hitbuffer except covering to zero
UpdateMCSubHits()	Saves simulation results from sHandle into hitbuffer

B.2.5. UpdateMainProcess.cpp

Update main hitbuffer from sub hitbuffer	
UpdateMCMainHits()	Add simulation results from sub hitbuffer to main hitbuffer

Update real covering in hitbuffer	
getStepSize()	Calculates step size for current step, see equation A.11
manageStepSize()	Adapts step size if desorptionrate · step size > than covering
UpdateCovering()	Uses step size and Krealvirt to calculate new covering Saved to `simHistory→coveringList`
UpdateCoveringphys()	Saves current real covering to hitbuffer
UpdateErrorMain()	Calculates total error, see equation A.12 Saved to `simHistory→errorList`
CalcPerIteration()	Calculates total error and covering over all facets per iteration

B.2.6. Iteration.cpp

Set Covering Threshold to avoid negative covering	
initCoveringThresh()	Initializes size of covering threshold vector
setCoveringThreshold()	Sets covering threshold for each facet

B.2.7. Buffer.cpp

Buffer functions	
Databuff struct()	signed int size BYTE *buff
checkReadable()	Checks if file can be opened for reading
checkWriteable()	Checks if file can be openend or created for writing
importBuff()	Imports buffer file to Databuff struct
exportBuff()	Exports Databuff struct to buffer file