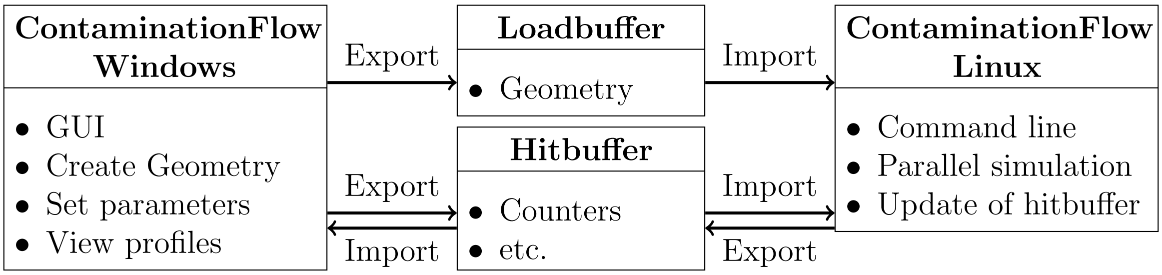
Documentation

ContaminationFlow on Linux and Windows

Hoai My Van, Rudolf Schönmann

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

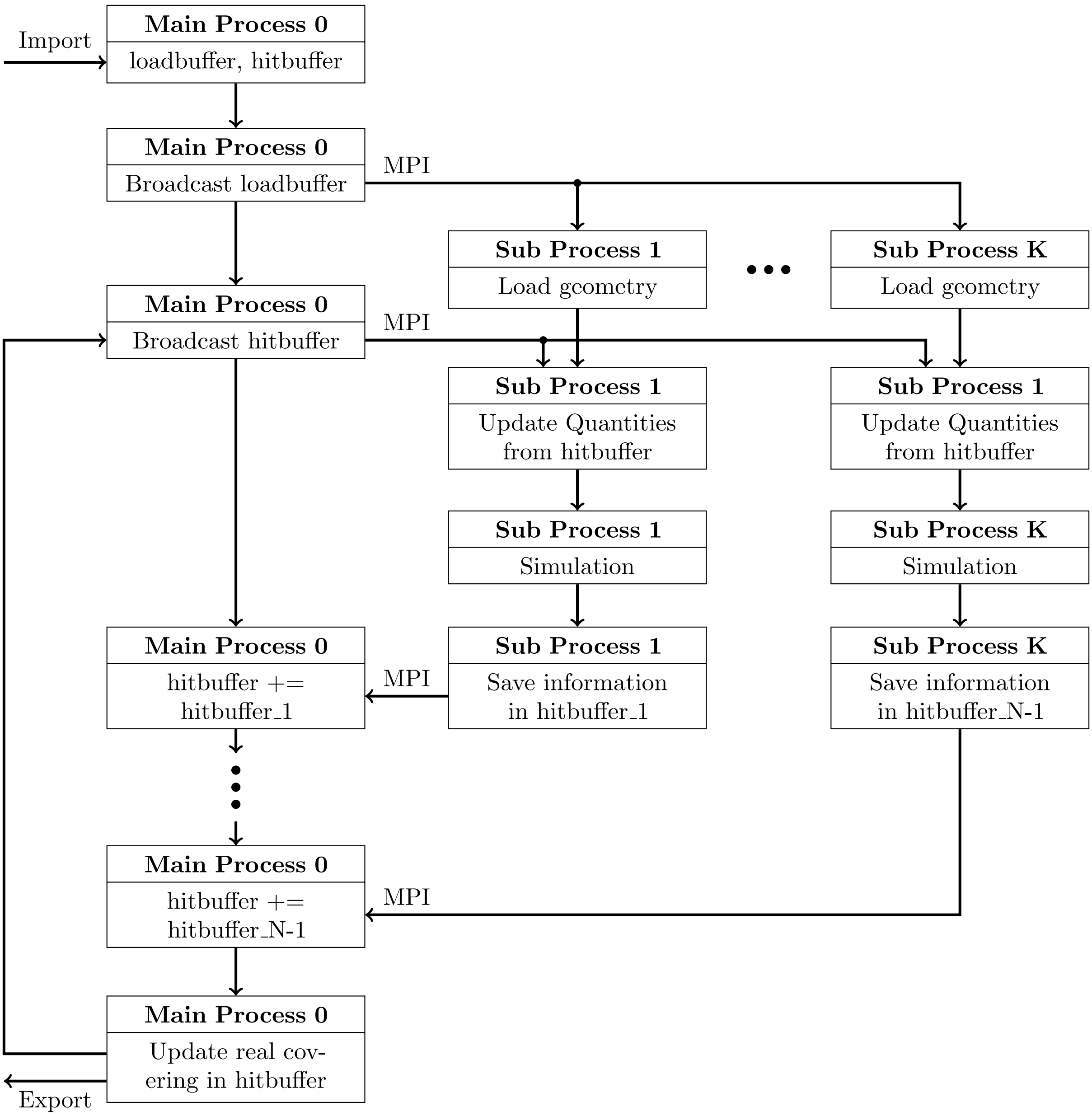


Figure 2.1.: Processing of data in main and sub processes

2.1. Call of Application from Command line

New class ProblemDef

- Defines parameters used for simulation
- Possible adaptation of default paramaters through input file
- 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
- $particleDia$: diameter of particles; default: `2·76E-12`
- E_{de}: binding energy of a particle on pure substrate; default: `1E-21`
- H_{vap}: vaporization enthalpy of a particle in case of multilayer contamination; default: `0.8E-19`
- W_{tr}: transition width between monolayer and multilayer properties; default: `1`
- $sticking$: constant sticking coefficient for all facets, set to zero, not used at the moment; default: `0`
- $targetPaticles$: minimum number of desorbed particles per iter.; default: `1000`
- $targetError$: average statistical uncertainty (error) to be achieved for each iteration, calculated as the average (weighted with the facets area) of the normalized standard deviation of events per facet; default: `0.001`
- $hitRatioLimit$: Ratio at which hits are ignored, default: `0`
- $Tmin$ or t_min: minimum time for step size; default: `1E-4`
- $maxStepSize$ or t_max: maximum time for step size; default: `max`
- $maxSimPerIt$: maximum simulation steps per iteration; default: `max`
- $coveringMinThresh$: minimum covering (through multiplication); default: `10000`

- *histsize* : Size of history lists; default: `max`
- *vipFacets* : very important facets: facets with have their own target error. input in inputfile as alternating sequence of facet numbers and respective target errors seperated via blanks; default: `[]`

Terminology

- Simulation time: desired computation time until check if target is reached for iteration
- Simulated time: physical time in the simulated system, e.g. flight time or residence time of a particle
- 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 overflow for integer and underflow for floating point numbers

2.4. New Quantities

New counter `covering`

- Number of carbon equivalent particles on facet
- Increases with adsorption, decreases with desorption
- Extracted from new hitbuffer counter from `Simulationcalc.cpp` file in `getCovering(·)`

Coverage

- Number of monolayers of adsorbed particles
- Calculated from covering, particle diamieter (previously 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

- Either E_{de} or H_{vap}
- Depending on the how many layers of particles are adsorbed.
- If coverage is smaller than a monolayer, it will be decided at random.

Desorption

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

Outgassing

- Number of particles from outgassing
- Calculated from facet outgassing and temperature defined in sHandle =¿ Um die genaue Berechnung vom Outgassing muessen wir uns noch kuemmern.
- Outgassing computed from `Worker.cpp` file in `CalcTotalOutgassingWorker()`

$K_{real/virtual}$

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

Statistical Error

- Event error: calculated from hits and desorbed particles (of facet and total)
- Covering error: calculated from adsorbed and desorbed particles (of facet and total)
- Used to determine significance of simulation results of iteration

Step size

- Minimum time between adsorption and desorption
- Step size computed from `UpdateMainProcess.cpp` file in `getStepSize()`

2.5. Iterative Algorithm

2.5.1. Initialization of simulation

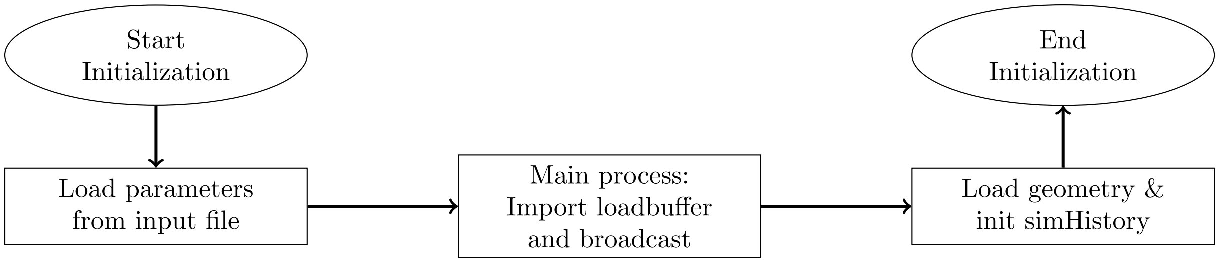


Figure 2.2.: Overview: Initialize simulation

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_event;
 HistoryList<double> errorList_covering;

 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 (event and covering) for each facet for each iteration, total hits and desorbed particles for each facet
- lastTime: simulated time (accumulated time steps) instead of computation time

2.5.2. Simulation on subprocesses

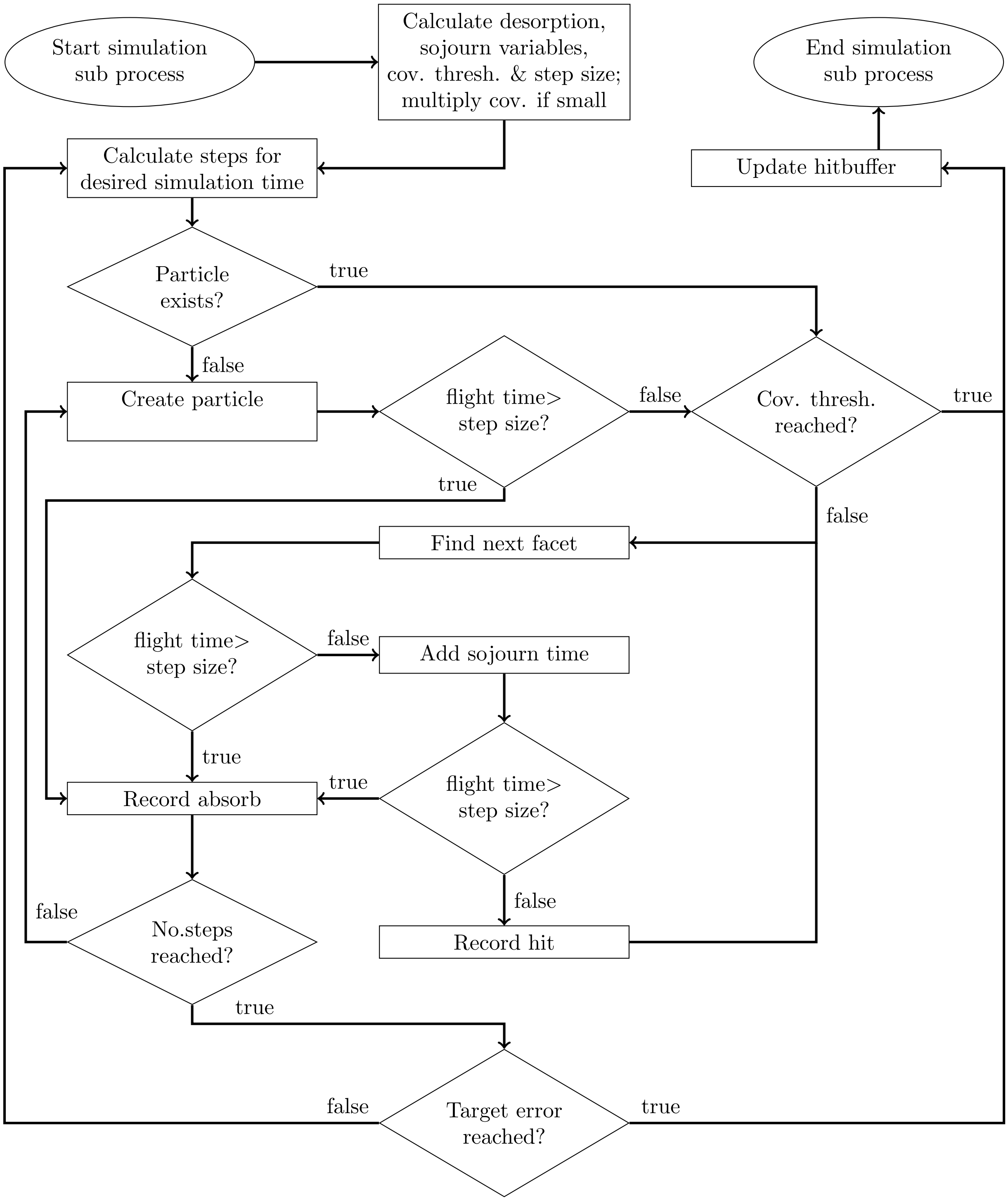


Figure 2.3.: Overview: simulation on sub processes

Calculate Step Size

- Use `simHistory→currentStep` to calculate logarithmic step size
- Calculation in `UpdateMainProcess.cpp` file in `getStepSize()`

Calculate 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

Multiply small covering

- Multiply covering so that smallest covering ≥ `ProblemDef::coveringThreshMin`
- Multiply covering threshold with same factor
- Calculation in `checkSmallCovering(·)` from `SimulationLinux.cpp` file

Calculate desorption

- Desorption calculated from current covering values
- Calculation in `UpdateDesorption(·)` from `UpdateSubProcess.cpp` file

Calculate residence time and binding energy

- Residence time of a particle on a surface calculated from binding energy, coverage and thermal oscillation frequency
- Binding energy calculated using binding energy on pure substrate, vaporization enthalpy and coverage
- Thermal oscillation frequency calculated using temperature, Boltzmann constant and Planck's constant
- Calculation in `UpdateSojourn(·)` from `UpdateSubProcess.cpp` file

Target error reached?

- Calculate statistical error in `UpdateError()` from `UpdateSubProcess.cpp` file
- Error to check can be either covering or event error (currently covering)
  - Check if vip facets reached their own target error
  - Check if normal facets total error reached target error
  - Currently vip facets not included in normal facets
- Total error calculated from summing facet error weighted with facet area
- Set error of facets that reached `ProblemDef::hitRatioLimit` to inf
- Set error of facets with no hits and desorption to inf
- Facets with error=inf are not considered
  - if vip facet: target automatically reached
  - if normal facet: facet error and area not used for calculation
- Check in `checkErrorSub(·)` from `UpdateSubProcess.cpp` file

2.6. Update main buffer

Before summation of subprocesses

- Calculate step size in `UpdateMainProcess.cpp` file in `getStepSize()` depending on, if the target error was reached
- Multiply covering in `hitbuffer` of main process in `checkSmallCovering(·)` from `SimulationLinux.cpp` file if covering is multiplied in sub processes

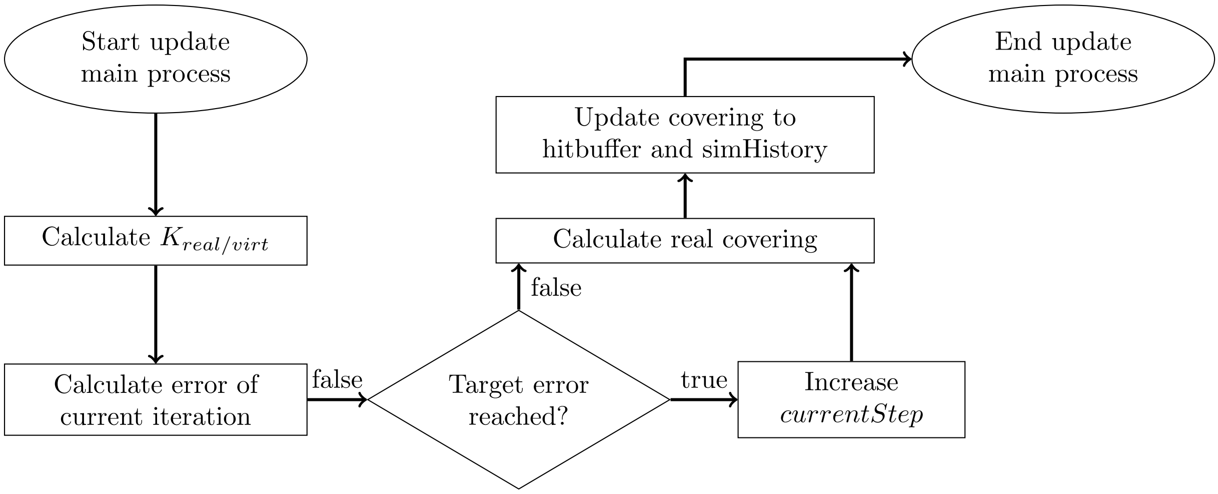


Figure 2.4.: Overview: update of covering in hitbuffer

Error Calculation

- Calculate statistical error of normal facets
- Error to can be either covering or event error (currently covering)
  - Covering: adsorbed and desorbed particles, weighted with opacity
  - Events: hits and desorbed particles, weighted with opacity
- Total error calculated from summing facet error weighted with facet area
- Set error of facets that reached `ProblemDef::hitRatioLimit` to inf
- Set error of facets with no hits and desorption to inf
- Facets with error=inf are not used for calculation of total error
  - if vip facet: target automatically reached
  - if normal facet: facet error and area not used for calculation
- Save error in `simHistory→errorList`
- Management in `UpdateErrorMain(·)` from `UpdateMainProcess.cpp` file
  ⇒ Increase `simHistory→currentStep` if target error reached

Calculate & Update Covering

- $K_{real/virtual}$ computed from `Simulationcalc.cpp` file in `GetMoleculesPerTP(·)`
- Divide covering in `hitbuffer` if previously multiplied
- Use $K_{real/virt}$ to calculate new covering
- Save new covering in `simHistory→coveringList` and hitbuffer
- Calculation in `UpdateCovering(·)` from `UpdateMainProcess.cpp` file
- Update buffers in `UpdateCoveringPhys(·)` from `UpdateMainProcess.cpp` file

2.7. Summary

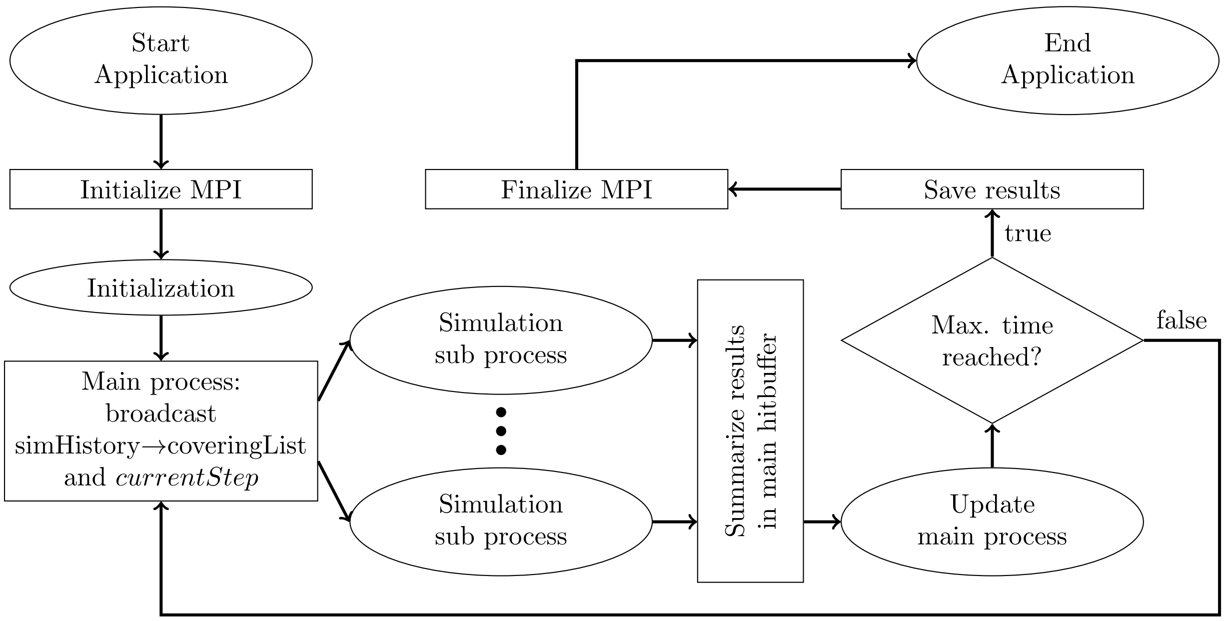


Figure 2.5.: Overview: ContaminationFlow application

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 `simHistory→coveringList` using `MPI_Bcast(·)`
  - Simulation in sub processes
    - Simulate until `targetParticles` and `targetError` or `covthresh` reached
    - Update hitbuffers 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 $K_{real/virtual}$ 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 initial coverage and temperature

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 ContaminationFlow 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

$$
\begin{aligned}
k_b &= 1.38\,10^{-23} \\
h &= 6.626\,10^{-34}
\end{aligned}
\tag{A.1}
$$

Variables

$$T = \text{Facet temperature} \tag{A.2}$$

Number of carbon equivalent particles of one monolayer

$$N_{mono} = \frac{\text{Area of Facet [m}^2\text{]}}{\text{ProblemDef::particleDia}^2\ \text{[m}^2\text{]}} \tag{A.3}$$

Carbon equivalent relative mass factor

$$\Delta N_{surf} = \frac{\text{carbon equivalent gas mass}}{12.011} \tag{A.4}$$

Covering θ^*

$$\theta^* = N_{\text{particles on facet}} \tag{A.5}$$

Coverage θ

$$\theta = \frac{\theta^*}{N_{mono}/\Delta N_{surf}} \tag{A.6}$$

Binding Energy E

$$E = \begin{cases} E_{de}, & \text{if particle binds with substrate} \\ H_{vap}, & \text{if particle binds with adsorbate} \end{cases} \tag{A.7}$$

Residence Time τ

$$
\begin{aligned}
A &= \exp\big(-E/(k_b T)\big) \\
\tau_0 &= \frac{k_b T}{h} \\
\tau &= \frac{-\ln(rnd)\cdot\tau_0}{A}
\end{aligned}
\tag{A.8}
$$

Step Size t_{step}

$$
\begin{aligned}
t_{min} &= \text{ProblemDef::t\_min} \\
t_i &= t_{min}\cdot\exp\big(\text{i}\cdot\ln(\text{ProblemDef::maxTimeS}/T_{min})/\text{ProblemDef::iterationNumber}\big) \\
t_{step} &= \min\big(t_{currentStep+1} - t_{currentStep}, \text{ProblemDef::t\_max}\big)
\end{aligned}
\tag{A.9}
$$

Desorption des

$$
\begin{aligned}
\tau_0 &= \frac{h}{k_b T} \\
\tau_{subst} &= \tau_0 \cdot \exp\left(\frac{E_{de}}{k_b T}\right) \\
\tau_{ads} &= \tau_0 \cdot \exp\left(\frac{H_{vap}}{k_b T}\right) \\
t_{ads} &= \tau_{ads} \cdot (\theta - 1)
\end{aligned}
\tag{A.10}
$$

$$
des = \begin{cases}
0, & \text{if } \theta = 0 \text{ or } T = 0 \\
\theta \cdot \big(1 - \exp(-t_{step}/\tau)\big), & \text{else if } \theta \leq 1 \\
t_{step}/\tau_{ads}, & \text{else if } \theta - 1 \geq t_{step}/\tau_{ads} \\
\theta - 1 + \big(1 - \exp(-(t_{step} - t_{ads}/\tau))\big), & \text{else if } \theta - 1 < t_{step}/\tau_{ads}
\end{cases}
$$

Outgassing out

$$
out = \frac{\text{Facet outgassing}}{k_b T} \tag{A.11}
$$

Small covering factor

$$
\begin{aligned}
mincov &= \text{Smallest covering on a single facet that desorbs} \\
\text{small covering factor} &= \begin{cases}
1, & \text{if } mincov \geq \text{ProblemDef::coveringMinThresh} \\
1 + 1.1 \cdot (\text{ProblemDef::coveringMinThresh}/mincov), & \text{otherwise}
\end{cases}
\end{aligned}
\tag{A.12}
$$

$K_{real/virtual}$

$$
\mathrm{K}_{\mathrm{real/virtual}} = \frac{\sum_{\text{facets}} \Big(out + des\Big)}{\text{number of total desorbed molecules/small covering factor}} \tag{A.13}
$$

Error

$$
\mathrm{error}(counter) = \begin{cases}
inf & \text{, if } (counter) \text{ on facet } = 0 \\
\Big(\frac{1}{(counter) \text{ on facet}} \cdot \frac{1 - (counter) \text{ on facet}}{\text{total } (counter)}\Big)^{0.5} & \text{, else}
\end{cases}
\tag{A.14}
$$

$$
\begin{aligned}
error\_covering &= \mathrm{error}(\text{adsorbed particles + desorbed particles}) \\
error\_event &= \mathrm{error}(\text{hits + desorbed particles})
\end{aligned}
$$

B. Datatypes

B.1. Class Members

Name	Datatype	Alias
SimulationHistory::coveringList	boost::multiprecision::uint_128t	covBoost
FacetHitBuffer::covering	llong	covLlong
FacetProperties::desorption	boost::multiprecision::float128	desBoost
Simulation::coveringThreshold	llong	

B.2. Functions

Function	Output Datatype	Relevant Input
getCovering()	boost::multiprecision::float128	covBoost
getCovering()	llong	covLlong
calcCoverage()	boost::multiprecision::float128 or llong	getCovering()
calcDesorption()	boost::multiprecision::float128	calcCoverage()
calctotalDesorption()	boost::multiprecision::float128	desBoost
GetMoleculesPerTP()	boost::multiprecision::float128	desBoost

C. Overview of new Classes and Functions

C.1. New Classes

SimulationHistory	
coveringList	of class HistoryList, stores covering history
errorList_event	of class HistoryList, stores error history for events
errorList_covering	of class HistoryList, stores error history for covering
hitList	of class HistoryList, stores hits for each facet
desorbedList	of class HistoryList, stores desorbed particles for each facet
startNewParticle	Determines wether to create a new particle for next iteration
numFacet	number of Facets
numSubProcess	number of sub processes used for simulation
nbDesorbed_old	number of total desorbed molecules of previous iteration $\Rightarrow$ To calculate difference between consecutive iterations
flightTime	Simulated flight time for iteration
nParticles	Simulated particles for iteration
lastTime	Total simulated time = last time in Lists
currentStep	step of logarithmic time step calculation in `getStepSize()`
stepSize	current step size
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
currIt	current iteration number
appendCurrent()	Appends currentList to pointInTimeList
appendList()	Append input list to pointInTimeList
convertTime()	Converts time for better clarity
printCurrent()	Print currentList as table to terminal, optional message
print()	Print pointInTimeList as table to terminal, optional msg
write(), read()	Write to file, read from file
set/getCurrent()	Set/get value of desired facet in currentList
setLast(), getLast()	Set/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 ⇒simulationTimeMS	Computation time of each iteration in milliseconds
maxTime, maxUnit ⇒maxTimeS	Maximal total simulated time in seconds
iterationNumber	Number of iterations
particleDia	Diameter of particles
E_de, H_vap	Parameters to calculate binding energy, see equation A.7
sticking	Sticking factor for all facets
targetParticles/-Error	Target values for each iteration
hitRatioLimit	threshold of hitratio at which hits are ignored
coveringMinThresh	Minimum covering, multiplication to this if covering low
t_min, t_max	Minimum/ Maximum step size
maxSimPerIt	Maximun simulation steps per iteration
histSize	Size of history lists (most recent values in memory)
vipFacets	alterning: vip facet and target error, e.g. 1 0.001 3 0.002
readInputfile()	Initialization from input file
printInputfile()	Print to terminal

C.2. New Functions

C.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 `simHistory→coveringList` and `simHistory→currentStep` to sub processes
setCoveringThreshold()	Sets covering threshold for each facet
UpdateSojourn()	Sets sojourn variables for each facet
UpdateDesorptionRate()	Sets desorption for each facet, ends simulation if 0
checkSmallCovering()	multiplies covering to reach threshold if covering small
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
`simHistory→coveringList` `simHistory→errorList`	Adapt size to `p→histSize` if necessary
End simulation if maximum simulation time is reached	

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

C.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 `sHandle`
smallCoveringFactor	If covering is small: Covering is multiplied by smallCoveringFactor to be able to have statistics without overflow of the covering variable
SimulationRun()	Simulate for desired simulation time
UpdateError()	Calculate current error of sub process
CheckErrorSub()	Checks if normal facets reached targetError and if vip facets reached own target
UpdateMCSubHits()	Save simulation results to hitbuffer

Small covering	
CheckSmallCovering()	If covering is small, find smallCoveringFactor to reach `p→coveringMinThresh`
Undo multiplication	In `UpdateCovering()`

C.2.3. Iteration.cpp

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

C.2.4. 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

C.2.5. Calculations in SimulationCalc.cpp etc.

SimulationCalc.cpp	
getCovering()	Get covering from hitbuffer or `simHistory`
getHits()	Get number of hits from hitbuffer
getnbDesorbed()	Get number of total desorbed molecules from hitbuffer
getnbAdsorbed()	Get number of total adsorbed molecules from hitbuffer
calcNmono()	see equation A.3
calcdNsurf()	see equation A.4
calcCoverage()	see equation A.6
calcEnergy()	see equation A.7
calcStickingnew()	sets sticking coefficient to p→sticking
calcDesorption()	see equation A.10
GetMoleculesPerTP()	see equation A.13
calctotalDesorption	calculates desorption for `startFromSource()`
calcPressure()	TODO has to be verified
calcParticleDensity()	TODO has to be verified

worker.cpp	
CalcTotalOutgassingWorker()	see equation A.11, calculates outgassing for `startFromSource()`

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

C.2.6. UpdateSubProcess.cpp

Update sHandle paramters from hitbuffer	
UpdateSticking()	Updates sticking
UpdateDesorptionRate()	Updates desorption
UpdateSojourn()	Enables sojourn time =¿ Verstehe ich nicht

Error calculations	
UpdateErrorSub()	Calculates error per facet, see equation A.14 Saves to `simHistory→ errorList_covering` and `simHistory→ errorList_event`
UpdateError()	Sums up error of normal facets & weights by facet area Currently covering error instead of event error
CheckErrorSub()	Checks if normal & vip facets reached respective target

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

C.2.7. 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.9
UpdateCovering()	Uses Krealvirt to calculate new covering Saved to `simHistory→coveringList`
UpdateCoveringphys()	Saves current real covering to hitbuffer
UpdateErrorMain()	Calculates total error for each facet, see equation A.14 Saves to `simHistory→errorList_event` and `simHistory→errorList_covering`
CalcPerIteration()	Calculates total error (covering and event) and covering over all facets per iteration