Application of Object-Oriented Principles and Design Patterns in the architecture of a Monte-Carlo simulation of liquids

February 17, 2015 - November 29, 2016

### Abstract

A classical Monte-Carlo simulation of liquids has a simple and well-defined structure: 1. Intialisation, 2. Iterations of move trial, 3. Results calculation. However, some details may vary such as the geometry (e.g. Bulk / Slab), the thermodynamic ensemble (e.g. Canonical / Grand canonical / Isobaric) or the particles type and interactions (e.g. Hard or Soft Spheres / Apolar or Dipolar Spheres). This tension between constance and variability may be addressed by the Object-Oriented paradigm (e.g. in Fortran, no kidding). By following simple OOP principles, we notice the natural emergence of Design Patterns in the architecture of our program. Those patterns ensure that our simulation program will be robust yet flexible to a certain extent.

However, compared to a procedural programming approach, object-oriented programming may have more pitfalls. For instance, the implementation is likely to be hindered if the overall design is not well-defined in advance. Thus an effort must be made upstream.

The notion of *interface* will be fundamental. It will define the general boundaries of the program. And it will be the key to switch between local alternatives.

## Notations:

- Type[:, ..., :] is an array of Type. The special case Type[:, :]<sub>sym</sub> means it's a 2D symmetric array. Hence, only the (upper or lower) triangular values need to be stored.
- Among the arguments of a ClassA :: construct(...), ♦ object: ClassB and ♦ object: ClassB are shorthands for aggregation and composition respectively. They are mainly used when ClassB is defined outside the namespace of ClassA, cf. Import. Otherwise they merely simplify the diagram.

# Contents

1	Core	3
	1.1 Common	3
	1.2 Common utilities & Random	3
2	Physical model	4
	2.1 Environment	4
	2.2 Particles	
	2.3 Interactions	6
3	Markov chain generator	9
	Markov chain generator  3.1 Changes	9
	3.2 Generating algorithms	
4	Markov chain explorer	11
	Markov chain explorer 4.1 Exploring algorithms	11
5	Observables	<b>12</b>
6	Input / Output	13

# 1 Core

# 1.1 Common

### PhysicalModel

environment: Environment

mixture: Mixture

shortInteractions: ShortInteractions

dipolar Interactions Dynamic:

DipolarInteractionsDynamic[numBoxes]

dipolar Interactions Static:

DipolarInteractionsStatic[numBoxes]

dipolarInteractionsFacades:

DipolarInteractionsFacade[numBoxes]

### MarkovChainGenerator

changes: Changes generatingAlgorithms:

Generating Algorithm [numAlgorithms]
plmcPropagator: PLMCpropagator

## GeneratingIO

data: JSONfile

report: GeneratingJSONreport

readers: Readers

writers: GeneratingWriters

# <<abstract>> PLMCpropagator

construct(◊ generatingAlgorithms:
GeneratingAlgorithm[:], ♦ selector:

TowerSampler)
destroy()
reset()

try(observables: GeneratingObservables)

#### MarkovChainExplorer

maximumBoxesCompressionExplorer: MaximumBoxCompressionExplorer[numBoxes

changedBoxesSizeRatio:
ChangedBoxSizeRatio[numBoxes]
volumeChangeMethod: VolumeChangeMethod

particleInsertionDomains:
ParallelepipedDomain[numBoxes]

particleInsertionMethod: ParticleInsertionMethod dipolarNeighbourhoodsVisitors:

dipolarNeighbourhoodsVisitors:
DipolarNeighbourhoodsVisitor[numBoxes]

### ExploringIO

data: JSONfile report: ExploringJSONreport readers: Readers writers: ExploringWriters

Figure 1: Common

Import: JSON :: JSONfile RANDOM :: TowerSampler

Environment, Paral-

lelepipedDomain

MIXTURE :: Component, Mixture SHORTINTERACTIONS :: ShortInteractions DIPOLARINTERACTIONS :: DipolarInteractionsDynamic, DipolarInteractionsStatic, Dipo-

larInteractionsFacade
CHANGES:: Changes, ChangedBoxSizeRatio
GENERATINGALGORITHM:: GeneratingAlgo-

rithm

EXPLORINGALGORITHMS:: MaximumBox-CompressionExplorer, VolumeChangeMethod ParticleInsertionMethod, DipolarNeighbour-

hoods Visitor

 $OBSERVABLES :: \ GeneratingObservables$ 

Readers :: Readers

 $W_{RITERS}:: Generating JSON report, \ Exploring JSON report, \ Generating Writers, \ Exploring Writers$ 

<<abstract>>
HeteroCouples

PotentialDomain

# 1.2 Common utilities & Random

# <abstract>> HeteroCouples construct(numPartners: int) destroy() getNum(): int get(index: int): int[2]

PotentialDomain
min: real
maxOverBoxEdge: real
max: real
delta: real

(a) COMMON: utilities

<<abstract>>
TowerSampler

construct(numCandidates: int)
destroy()
reset(numSCandidates: int[:])
getNumChoices(): int
get(): int

<<abstract>>
RandomCoordinates

construct(...)
destroy()
get(iComponent: int): real[3]

(b) Random

Figure 2: Common utilities & Random

# 2 Physical model

# 2.1 Environment

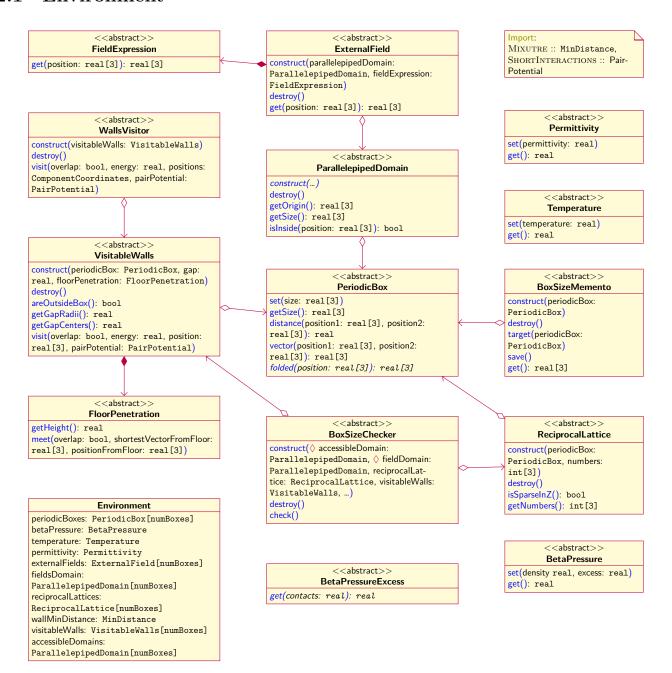


Figure 3: Environment

## 2.2 Particles

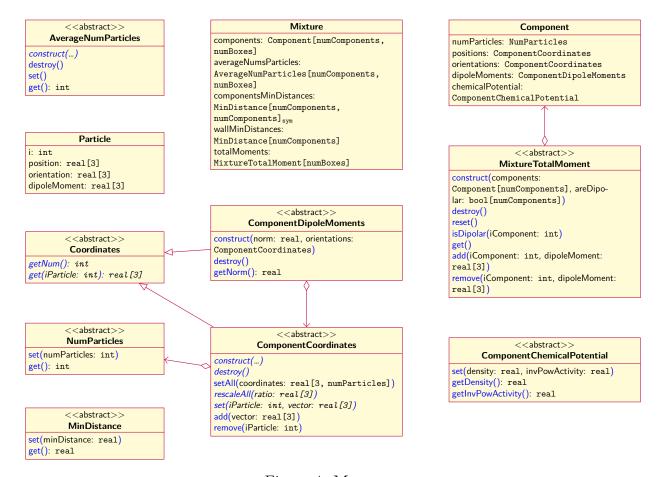


Figure 4: MIXTURE

## 2.3 Interactions

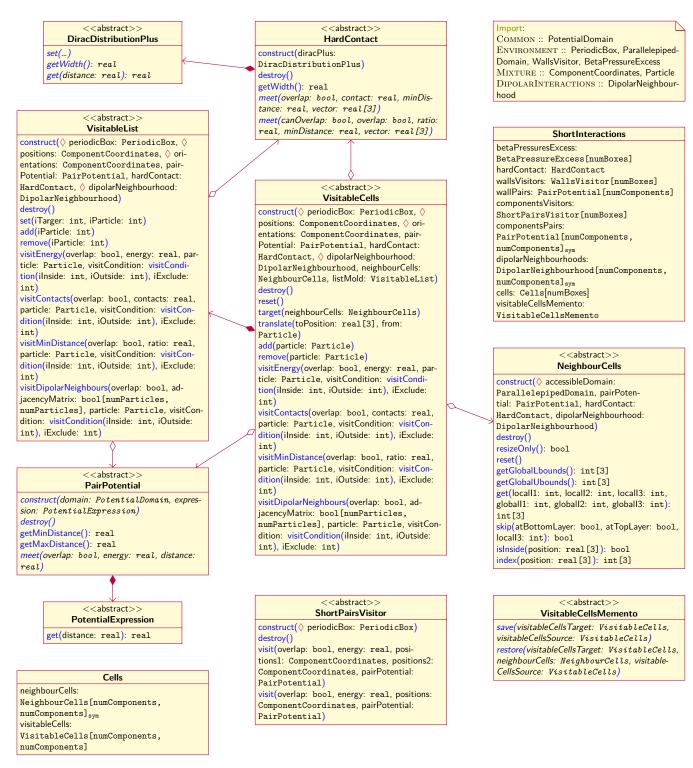


Figure 5: SHORT INTERACTIONS

#### <<abstract>> <<abstract>> **DESrealComponent** DESselfComponent COMMON :: PotentialDomain Environment :: PeriodicBox, BoxSizeMe construct(♦ periodicBox: PeriodicBox, ♦ construct(♦ periodicBox: PeriodicBox, mento, Permittivity boxSizeMemento: BoxSizeMemento, ◊ posipermittivity: Permittivity, \( \rightarrow \text{dipoleMo-} MIXTURE :: Component, MixtureTotalMotions: ComponentCoordinates, \( \Q \) dipoleMoments: ComponentDipoleMoments, alpha: ment, ComponentCoordinates, Componentments: ComponentDipoleMoments, realPair: DESconvergenceParameter) DipoleMoments, Particle DESrealPair) destroy() destroy() target(boxSizeMemento: BoxSizeMemento, remeet(dipoleMoment: rea[3]): real alPair: DESrealPair) <<abstract>> visit(energy: real, particle: Particle, iEx-DipolarInteractionsFacade clude: int) construct(..., ♦ components: <<abstract>> Component [numComponents], **DESrealPair** dipolarInteractionsDynamic: DipolarInteractionsDynamic, dipolarInterconstruct(♦ boxSizeMemento: <<abstract>> actionsStatic: DipolarInteractionsStatic) BoxSizeMemento, permittiv-**DESsurfMixture** itv: Permittivity, alpha: construct(♦ periodicBox: PeriodicBox, persave(dipolarInteractionsStatic: DESconvergenceParameter, domain: DipolarInteractionsStatic, newBoxVolmittivity: Permittivity, $\Diamond$ totalMoment: Potential Domain) ume: real) MixtureTotalMoment) destroy() restore(dipolarInteractionsStatic: target(boxSizeMemento: BoxSizeMemento) destroy() visit(): real visitTransmutation(ijComponents int[2], DipolarInteractionsStatic) reset() meet(vectorIJ: real[3], momentI: real[3], visit(newEnergies: real[numComponents, dipoleMoment2: real[3], dipoleMoment1: momentJ: real[3]) numComponents] sym, newSharedEnergy: real[3]): real visitRotation(iComponent: int, dipoleMo-ment2: real[3], dipoleMoment1: real[3]): real, boxVolumeRatio: real, energies: real[numComponents, numComponents]sym, sharedEnergy: real) real visitAdd(iComponent: int, dipoleMoment: **DipolarInteractionsStatic** real[3]): real boxSizeMementoReal: BoxSizeMemento visitRemove(iComponent: int, dipoleMoment: realPair: DESrealPair real[3]): real boxSizeMementoReci: BoxSizeMemento DipolarInteractionsDynamic reciWeight: DESreciWeight alpha: DESconvergenceParameter reciStructure: DESreciStructure dlcWeight: DLCweight realComponents: DESrealComponent[numComponents, dlcStructures: DLCstructures <<abstract>> numComponents, numBoxes] **DESconvergenceParameter** reciVisitors: DESreciVisitor[numBoxes] set(alphaXboxEdge: real) selfComponents: getTimesBoxEdge(): real DESselfComponent[numComponents, numBoxesl surfMixture: DESsurfMixture[numBoxes] dlcVisitors: DLCvisitor[numBoxes] <abstract>> <<abstract>> <<abstract>> **DESreciWeight** StructureVisitor **DLCweight** <<abstract>> DipolarNeighbourhood <<abstract>> <<abstract>> set(maxDistance: real) **DESreciVisitor DLC**visitor getMaxDistance(): real meet(overlap: bool, ijAreNeighbour: bool,

Figure 6: DIPOLAR INTERACTIONS

<<abstract>>

**DLCstructures** 

<<abstract>>

StructureFactor

<<abstract>>

**DESreciStructure** 

minDistance: real, vectorIJ: real[3], orien-

tationI: real[3], orientationJ: real[3])

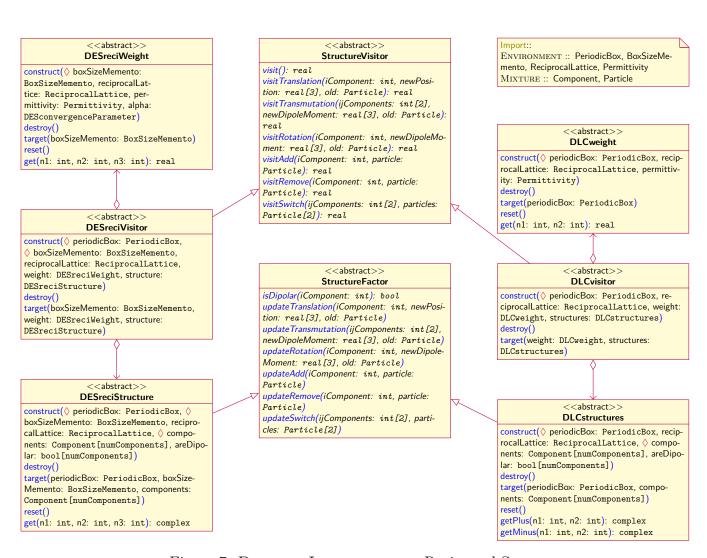


Figure 7: DIPOLAR INTERACTIONS: Reciprocal Space

# 3 Markov chain generator

# 3.1 Changes

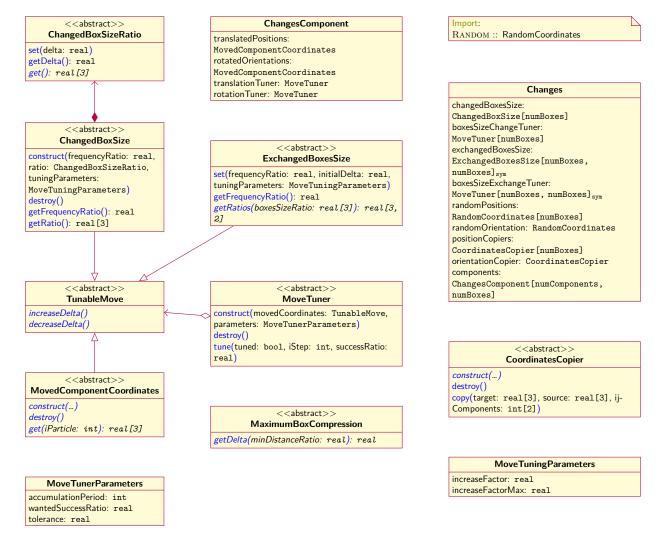


Figure 8: Changes

# 3.2 Generating algorithms

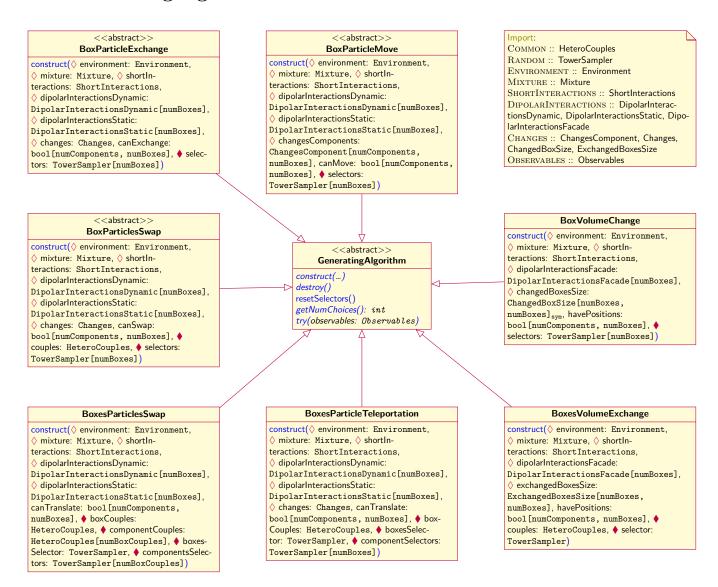


Figure 9: Generating Algorithms

# 4 Markov chain explorer

# 4.1 Exploring algorithms

### 

# <abstract>> ParticleInsertionMethod construct(◊ environment: Environment, ↑ numsParticles: NumParticles [numComponents], ◊ components: Component[numComponents], ◊ shortInteractions: ShortInteractions, ◊ dipolarInteractionsDynamic: DipolarInteractionsDynamic, ↑ randomPosition: RandomCoordinates, ↑ randomOrientation: RandomCoordinates) destroy() try(observables: ExploringObservables)

```
Import:
RANDOM :: RandomCoordinates
ENVIRONMENT :: PeriodicBox, Environment
MIXTURE :: NumParticles, Component
SHORTINTERACTIONS :: PairPotential, VisitableCells, ShortInteractions
DIPOLARINTERACTIONS :: DipolarInteractionsDynamic, DipolarInteractionsStatic, DipolarInteractionsFacade
CHANGES :: ChangedBoxSizeRatio, Maximum-BoxCompression
OBSERVABLES :: ExploringObservables
```

# <<abstract>> VolumeChangeMethod construct(◊ environment: Environment, ◊ components: Component[numComponents], ◊ shortInteractions: ShortInteractions, ◊ dipolarInteractionsFacade: DipolarInteractionsFacade, ◊ changed-BoxSizeRatio: ChangedBoxSizeRatio) destroy() try(observables: ExploringObservables)

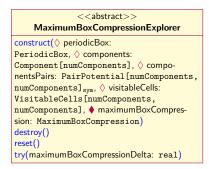


Figure 10: EXPLORING ALGORITHMS

# 5 Observables

## GeneratingObservables

accessibleDomainsSize: real[3, numBoxes] volumesChangeCounter: ChangeCounter[numBoxes]  $volumes Change Success: \verb|real[numBoxes]||$ volumes Exchange Counter:ChangeCounter[numBoxes, numBoxes] sym  $volumes {\sf ExchangeSuccess: real[numBoxes,}$ numBoxes]<sub>sym</sub> teleportationsCounters: ChangeCounter[numComponents, numBoxes, numBoxes] teleportationsSuccesses:  $\verb"real[numComponents", numBoxes","\\$ numBoxesl switchesCounters: ChangeCounter[numComponents, numComponents, numBoxes, numBoxes] switchesSuccesses: real[numComponents, numComponents, numBoxes, numBoxes] numsParticles: int[numComponents, numBoxesl energies: ObservablesEnergies[numBoxes] changes: ObservablesChanges[numBoxes]

### ObservablesEnergies

wallsEnergies: real[numComponents] fieldEnergies: real[numComponents] shortEnergies: real[numComponents, numComponents] sym dipolarEnergies: real[numComponents, numComponents] sym dipolarMixtureEnergy: real

### ChangesCounter

translation: ChangeCounter rotation: ChangeCounter exchange: ChangeCounter

### ChangesSuccess

translation: real rotation: real exchange: real

### ChangeCounter

numHits: int numSuccesses: int

### **ExploringObservables**

maximumBoxesCompressionDelta:
real[numBoxes]
betaPressuresExcess: real[numBoxes]
invPowActivities: real[numComponents,
numBoxes]
energies: ObservablesEnergies[numBoxes]
insertionCounters:
ChangeCounter[numComponents, numBoxes]
insertionSuccesses: real[numComponents,
numBoxes]
adjacencyMatrices: bool[numParticlesI,

numParticlesJ][numComponents,

 ${\tt numComponents, numBoxes]}$ 

### ObservablesChanges

changesCounters:
ChangesCounter[numComponents]
changesSuccesses:
ChangesSuccesse[numComponents]
switchesCounters:
ChangeCounter [numComponents,
numComponents]
sym
switchesSuccesses: real[numComponents,
numComponents]
transmutationsCounters:
ChangeCounter [numComponents,
numComponents]
transmutationsSuccesses:
real[numComponents]

Figure 11: Observables

# 6 Input / Output

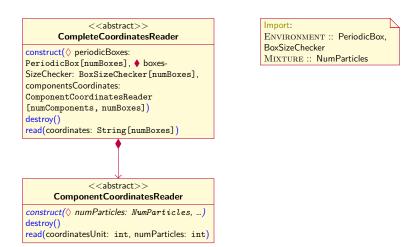


Figure 12: Readers

## 

# construct(iComponent: int, ◊ positions: ComponentCoordinates, ◊ orientations: ComponentCoordinates, coordinatesSelector: ComponentCoordinatesWriterSelector) destroy() getNum(): int write(coordinatesUnit: int)

# <<abstract>> ChangesSuccessWriter construct(filename: String, selector: ChangesSuccessWriterSelector) destroy() write(iStep: int, changesSuccess:

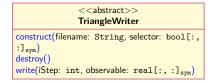
ChangesSuccess)

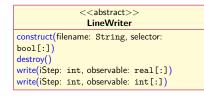
# ExploringWriters

maximumBoxesCompressionDelta:
RealWriter[numBoxes]
betaPressuresExcess: RealWriter[numBoxes]
energies: EnergiesWriters[numBoxes]
invPowActivities: LineWriter[numBoxes]
insertionSuccesses: LineWriter[numBoxes]

# ExploringJSONreport root: JSONvalue randomSeed: JSONvalue

# 





**GeneratingWriters** 

# accessibleDomainsSize: RealWriter[numBoxes] volumesChangeSuccess: LineWriter volumesExchangeSuccess: TriangleWriter teleportationsSuccesses: LineWriter[numBoxes, numBoxes] swapsSuccesses: RectangleWriter[numBoxes, numBoxes] numsParticles: LineWriter[numBoxes] completeCoordinates: CompleteCoordinatesWriter energies: EnergiesWriters[numBoxes] componentsChanges: ChangesSuccessWriter[numComponents, numBoxes]

completeCoordinates:
CompleteCoordinatesWriter
energies: EnergiesWriters[numBoxes]
componentsChanges:
ChangesSuccessWriter[numComponents,
numBoxes]
switchesSuccesses:
TriangleWriter[numBoxes]
transmutationsSuccesses:
RectangleWriter[numBoxes]

Figure 13: Writers

Import:
JSON :: JSONvalue
ENVIRONMENT :: PeriodicBox
MIXTURE :: ComponentCoordinates
OBSERVABLES :: ChangesSuccess

# <<abstract>> RealWriter construct(filename: String) destroy() write(iStep: int, observable: real) write(iStep: int, observable: real[:])

# ComponentCoordinatesWriterSelector writePositions: bool writeOrientations: bool

# ChangesSuccessWriterSelector writeTranslations: bool writeRotations: bool writeExchanges: bool

EnergiesWriters

wallsEnergies: LineWriter
fieldEnergies: LineWriter
shortEnergies: TriangleWriter
dipolarEnergies:
TriangleWriter
dipolarMixtureEnergy:
RealWriter

GeneratingJSONreport
root: JS0Nvalue
randomSeed: JS0Nvalue
algorithmsWeight: JS0Nvalue

# References

- [1] Eric Freeman et al. *Head First Design Patterns: A Brain-Friendly Guide*. 1st edition. Sebastopol, CA: O'Reilly Media, Oct. 2004. 694 pp. ISBN: 978-0-596-00712-6.
- [2] Daan Frenkel and Berend Smit. *Understanding Molecular Simulation*. 2nd. Orlando, FL, USA: Academic Press, Inc., 2001. ISBN: 978-0-12-267351-1.