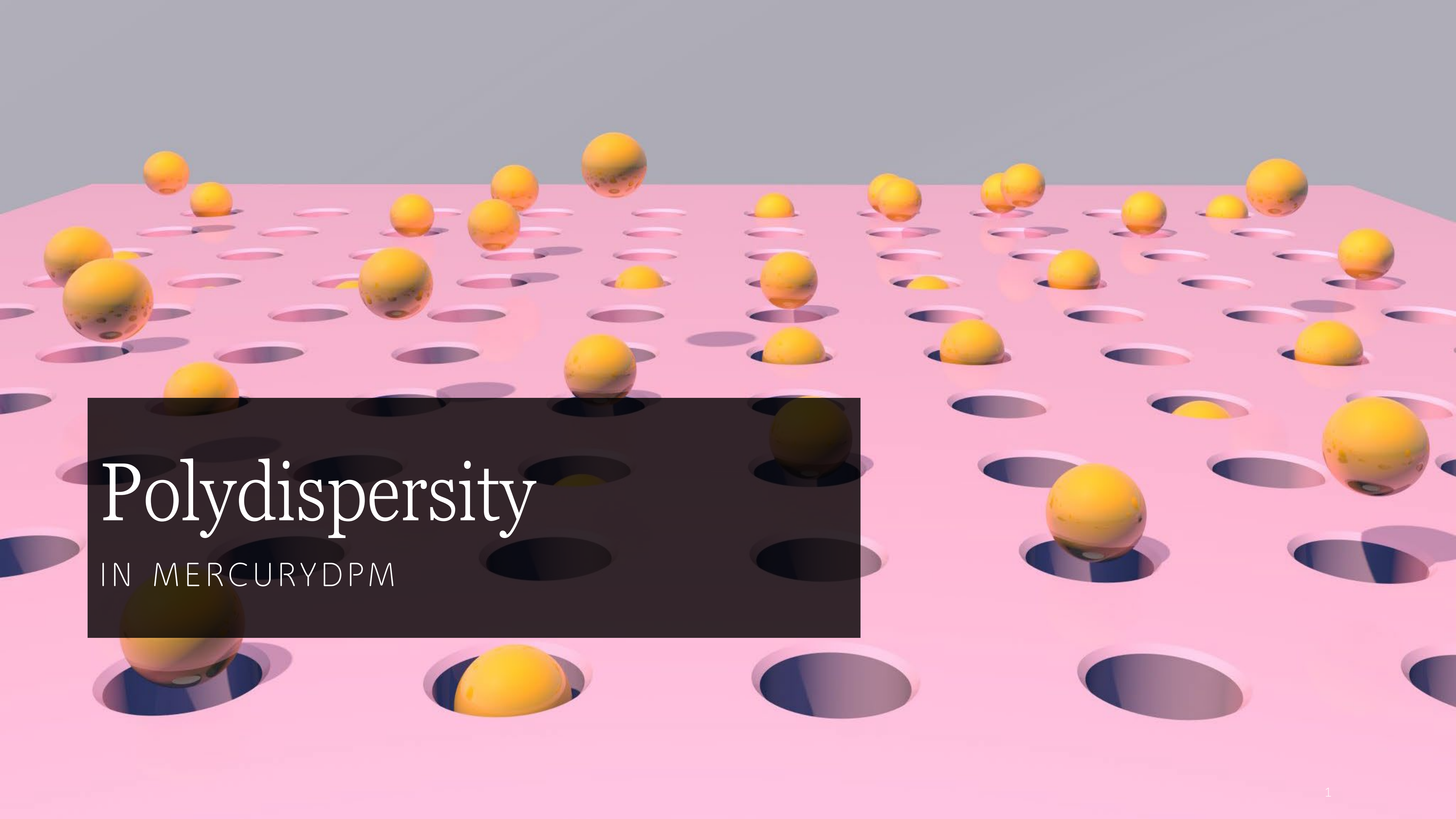


Polydispersity

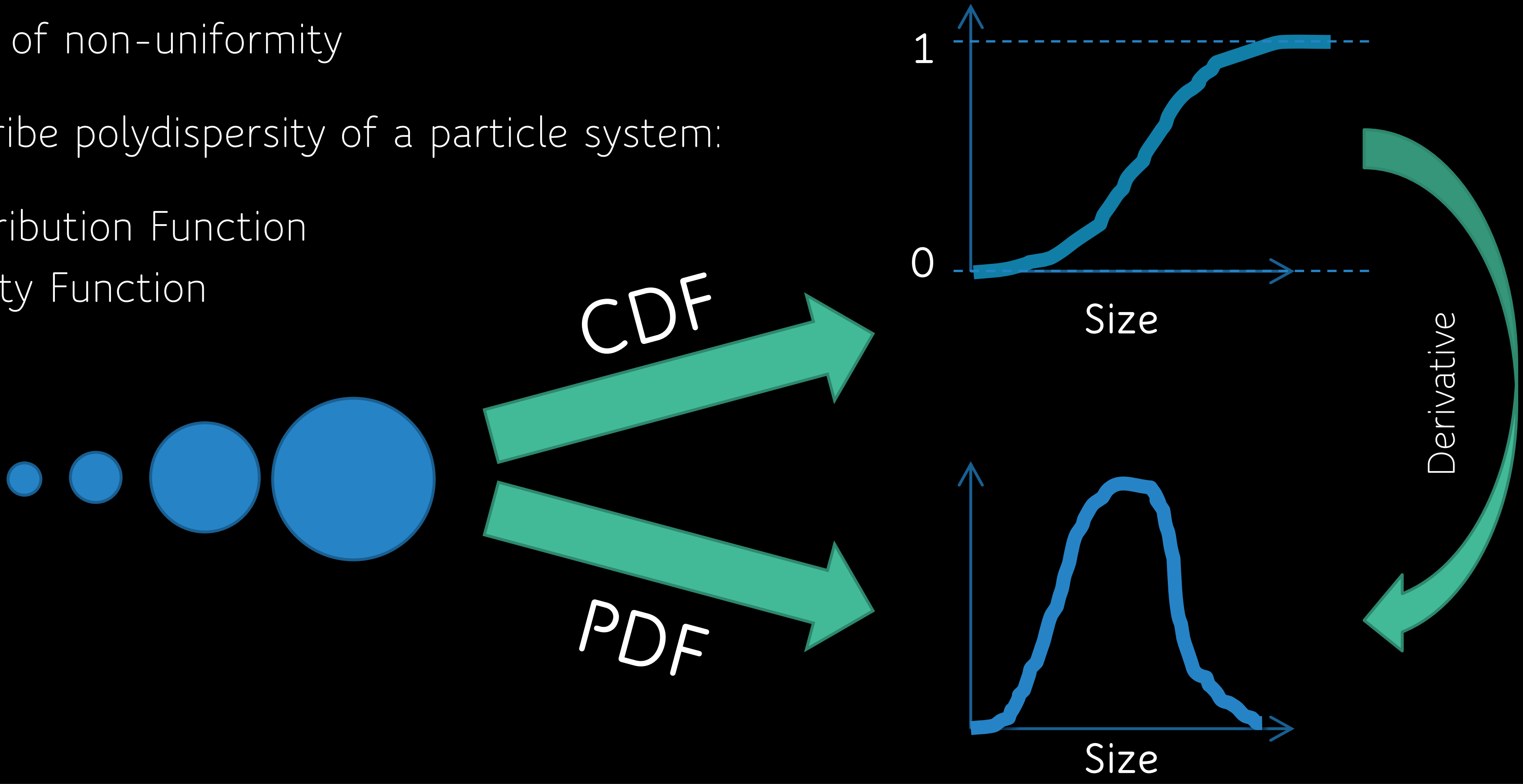
IN MERCURYDPM

Introduction

Polydispersity := Degree of non-uniformity

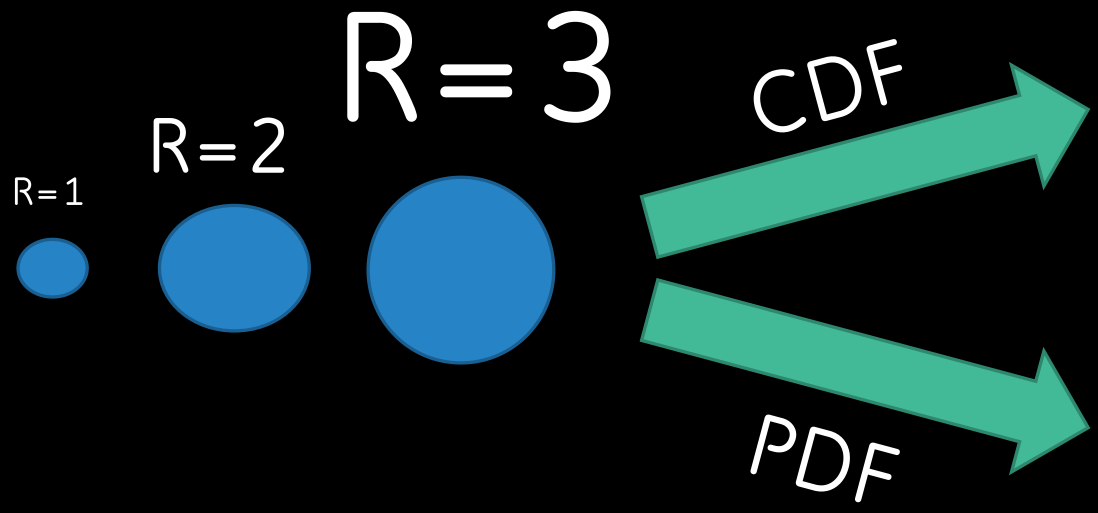
Two functions can describe polydispersity of a particle system:

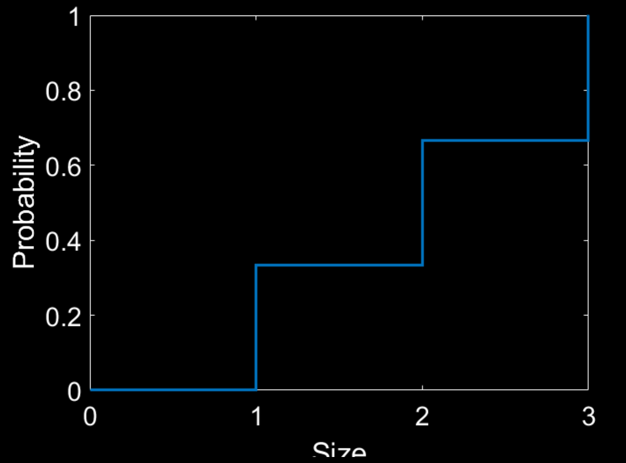
CDF := Cumulative Distribution Function
PDF := Probability Density Function

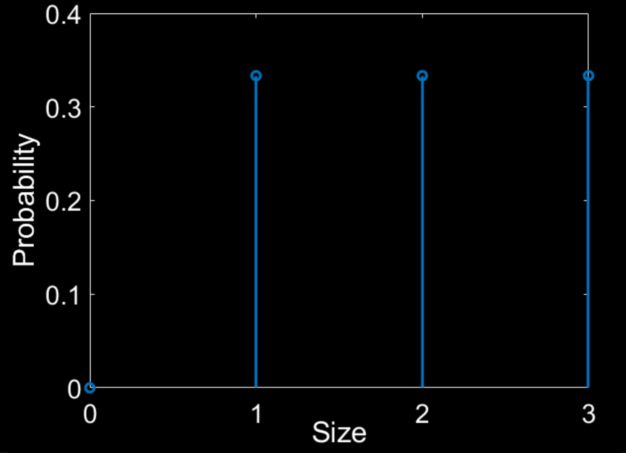


Distribution example

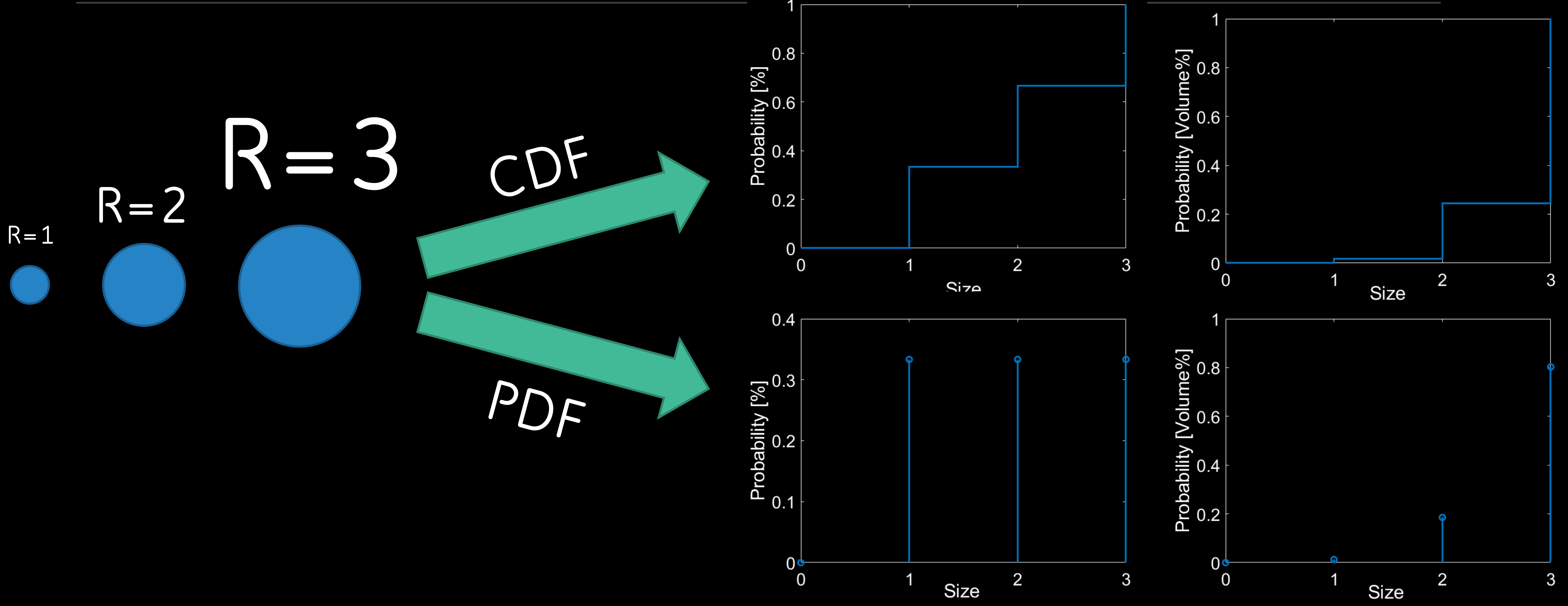
R = Radius





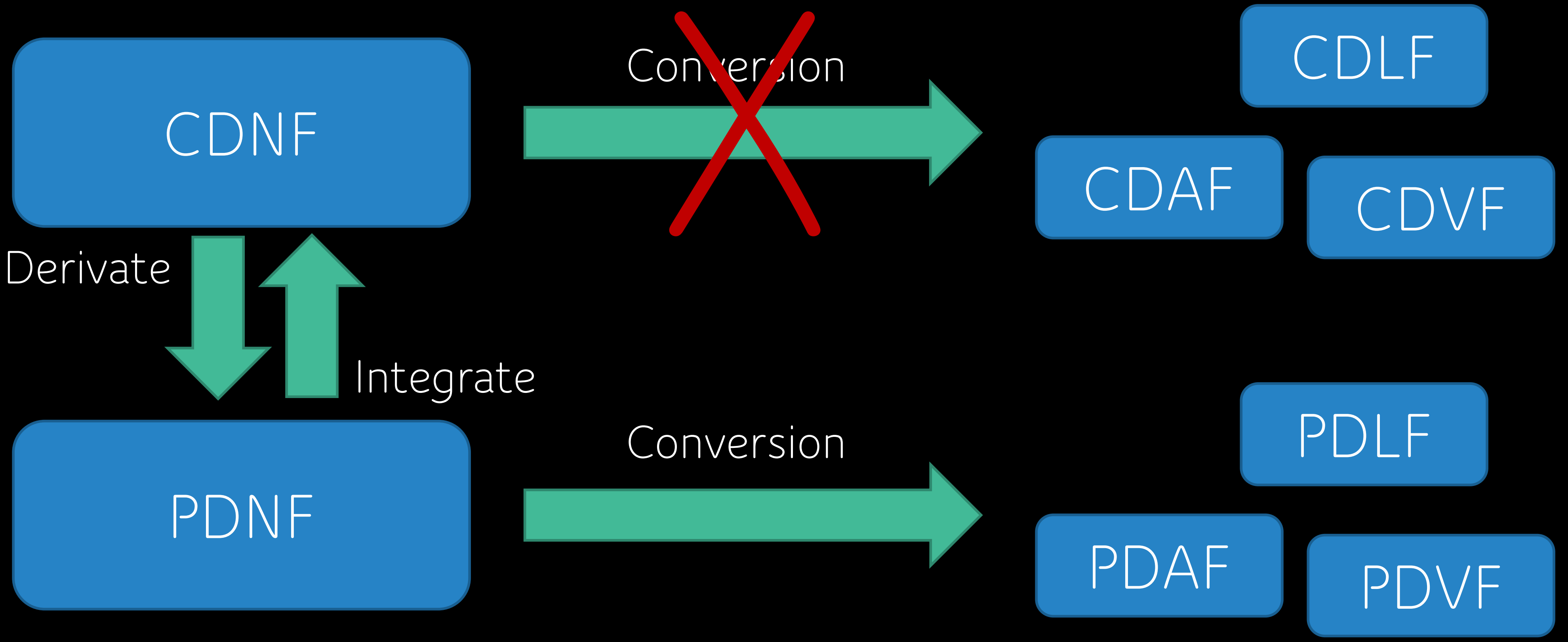


Distribution types



Distribution types

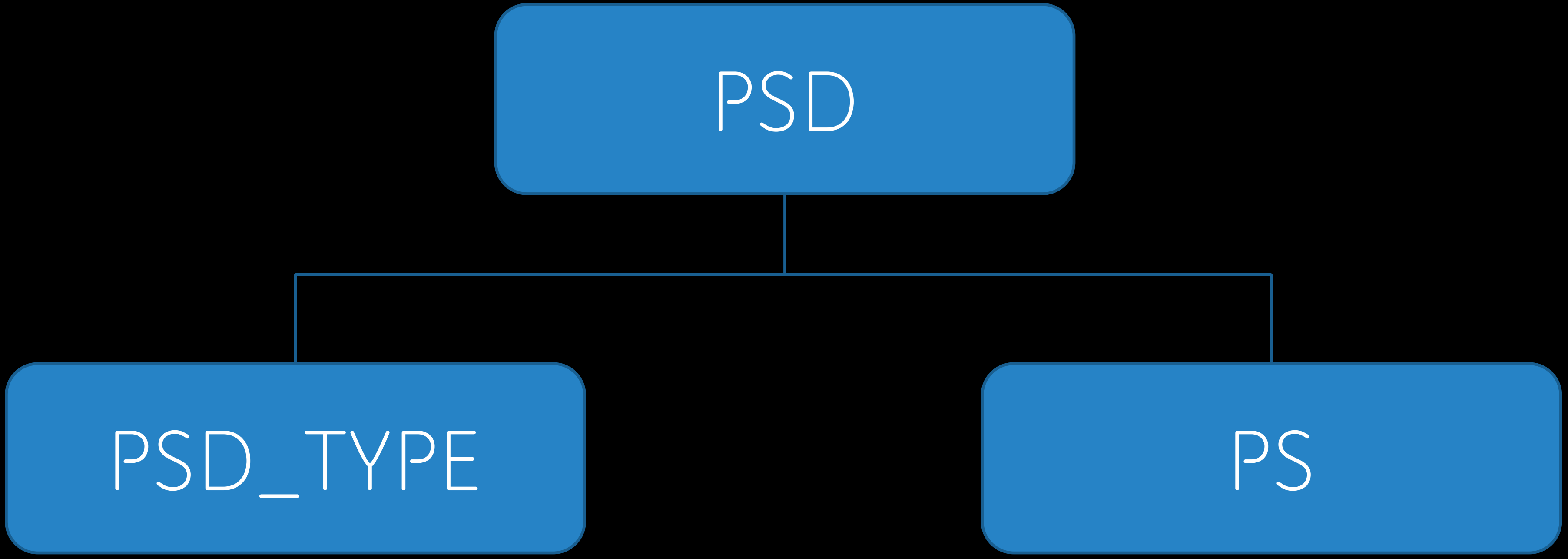
N = Number A = Area
L = Length V = Volume



The vision

1. A PSD class should be implemented which stores the PSD in a class object
2. PSDs should be read in via CSV files
3. Default distribution should be a Cumulative number distribution (CND)
4. The user has to know which distribution he is reading into Mercury
5. Statistical values should be derivable from the PSD class

The PSD class

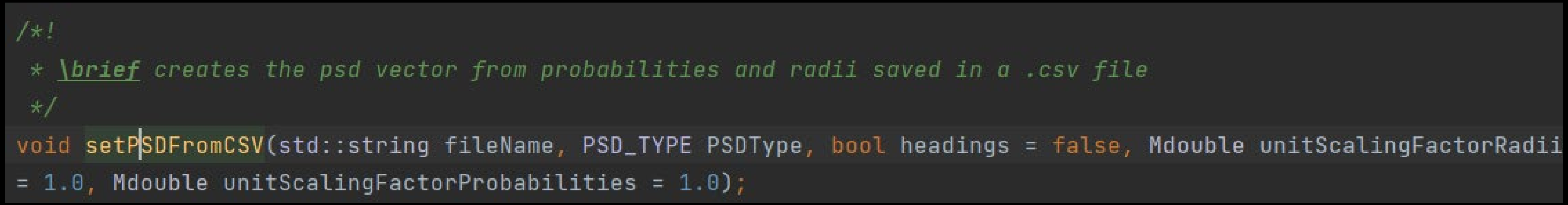


Stores type of PSD, used for conversion

Stores radius and probabilites

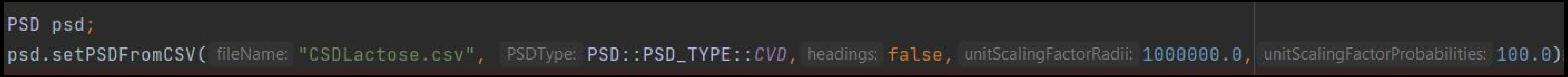
Setting a PSD

To set a PSD you need the following information:



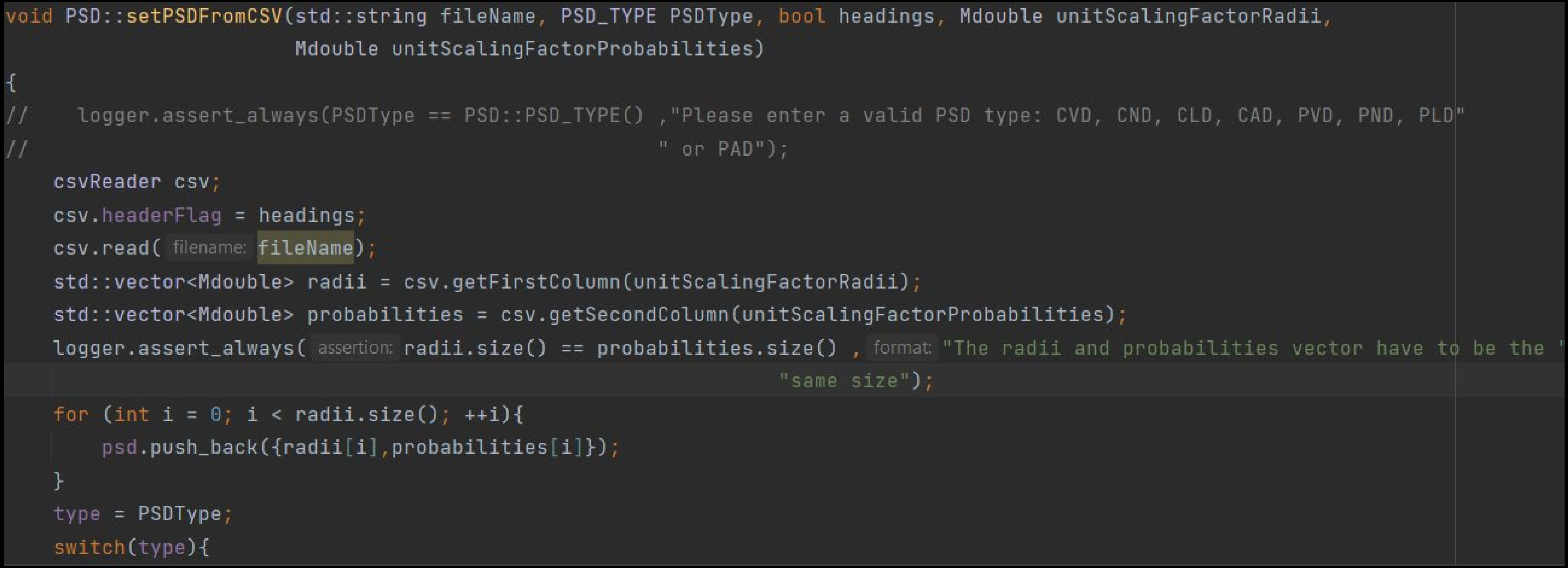
```
/*!
 * \brief creates the psd vector from probabilities and radii saved in a .csv file
 */
void setPSDFromCSV(std::string fileName, PSD_TYPE PSDType, bool headings = false, Mdouble unitScalingFactorRadii
= 1.0, Mdouble unitScalingFactorProbabilities = 1.0);
```

Example:



```
PSD psd;
psd.setPSDFromCSV( fileName: "CSDLactose.csv",  PSDType: PSD::PSD_TYPE::CVD, headings: false, unitScalingFactorRadii: 1000000.0, unitScalingFactorProbabilities: 100.0)
```

Setting a PSD details



```
void PSD::setPSDFromCSV(std::string fileName, PSD_TYPE PSDType, bool headings, Mdouble unitScalingFactorRadii,
                        Mdouble unitScalingFactorProbabilities)
{
//    logger.assert_always(PSDType == PSD::PSD_TYPE() ,"Please enter a valid PSD type: CVD, CND, CLD, CAD, PVD, PND, PLD"
//                                                     " or PAD");
    csvReader csv;
    csv.headerFlag = headings;
    csv.read( filename: fileName);
    std::vector<Mdouble> radii = csv.getFirstColumn(unitScalingFactorRadii);
    std::vector<Mdouble> probabilities = csv.getSecondColumn(unitScalingFactorProbabilities);
    logger.assert_always( assertion: radii.size() == probabilities.size() , format: "The radii and probabilities vector have to be the "
                                                                  "same size");
    for (int i = 0; i < radii.size(); ++i){
        psd.push_back({radii[i],probabilities[i]});
    }
    type = PSDType;
    switch(type){
```

A switch-statement ensures that every PSD_TYPE is converted to the default CND which can then be passed to the insertionboundary

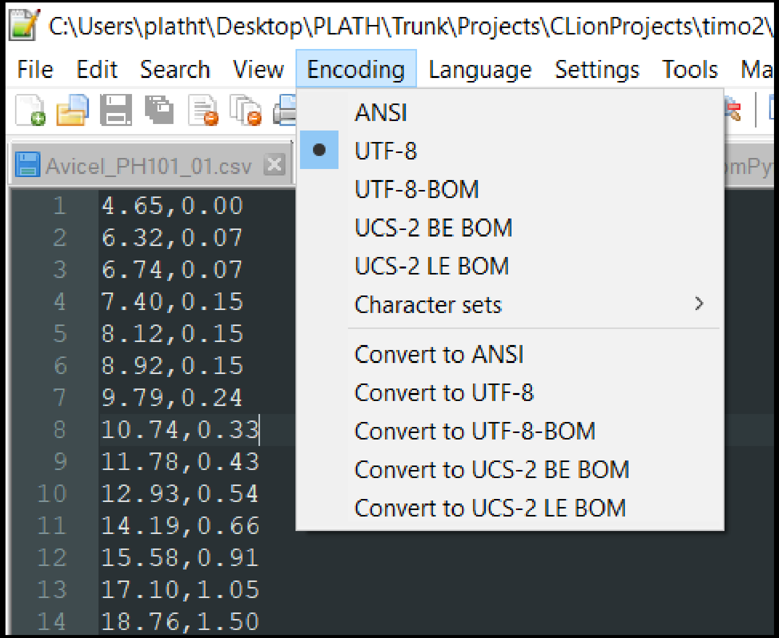
Encoding problem

Files are encoded by UTF-8-BOM

- -> BOM adds a Zero Width No-Break Space (ZWNBSP) at the first line of your CSV
  - -> The CSVReader will set the first line to zero

All files generated by CLion are generated with UTF-8-BOM by default

Errors will occur when calculating a minimal radius.

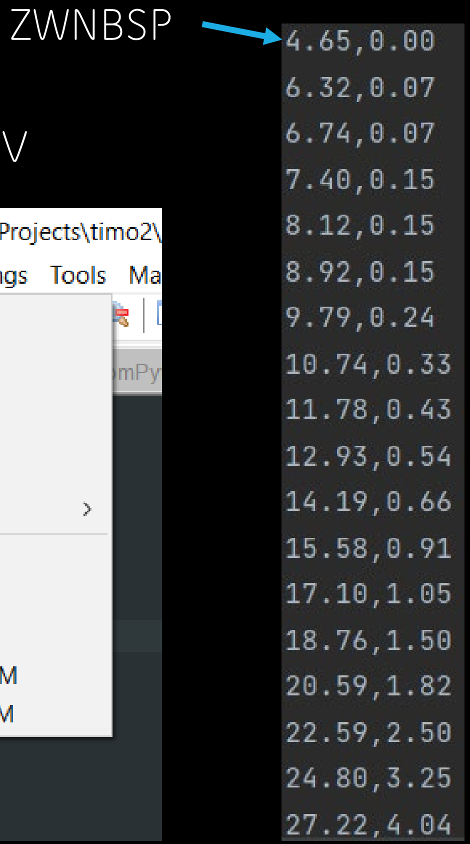


C:\Users\platht\Desktop\PLATH\Trunk\Projects\CLionProjects\timo2\

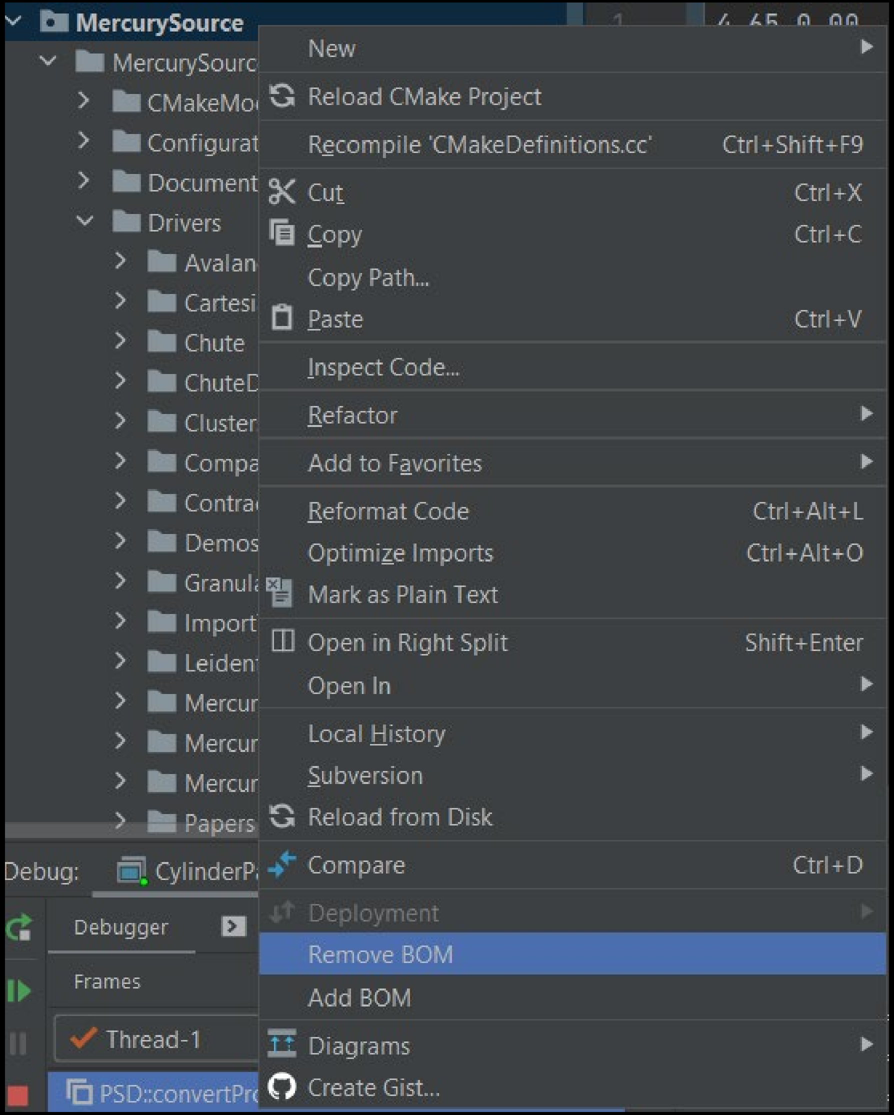
File Edit Search View Encoding Language Settings Tools Ma

Avicel_PH101_01.csv

```
4.65,0.00
6.32,0.07
6.74,0.07
7.40,0.15
8.12,0.15
8.92,0.15
9.79,0.24
10.74,0.33
11.78,0.43
12.93,0.54
14.19,0.66
15.58,0.91
17.10,1.05
18.76,1.50
```



```
4.65,0.00
6.32,0.07
6.74,0.07
7.40,0.15
8.12,0.15
8.92,0.15
9.79,0.24
10.74,0.33
11.78,0.43
12.93,0.54
14.19,0.66
15.58,0.91
17.10,1.05
18.76,1.50
20.59,1.82
22.59,2.50
24.80,3.25
27.22,4.04
```


MercurySource
MercurySourc
CMakeMo
Configurat
Document
Drivers
Avalan
Cartesi
Chute
ChuteD
Cluster
Compa
Contra
Demos
Granul
Import
Leiden
Mercur
Mercur
Mercur
Papers
Debug:
CylinderP
Debugger
Frames
Thread-1
PSD::convertPr
New
Reload CMake Project
Recompile 'CMakeDefinitions.cc' Ctrl+Shift+F9
Cut Ctrl+X
Copy Ctrl+C
Copy Path...
Paste Ctrl+V
Inspect Code...
Refactor
Add to Favorites
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Mark as Plain Text
Open in Right Split Shift+Enter
Open In
Local History
Subversion
Reload from Disk
Compare Ctrl+D
Deployment
Remove BOM
Add BOM
Diagrams
Create Gist...

Further implementations

PSD manipulating functions:

- Squeeze
- Cut-off

Statistical values:

- Dx, $x \in [0,100]$
- VolumetricMeanRadius (Radius where a monodisperse system has the same number of particles)
- MaxRadius
- MinRadius
- Moments? (standard deviation, mean, skewness, kurtosis, etc.)
- Sauter diameter, De Brouckere diameter? (i.e. D[p,q])
- Span?
- Mode?
- Median?