`puritymonitor`

Python package

Version 0.1.0 (2024-09-27) unreleased

Manual rev. 2 (2024-09-27) unreleased

Bastien Voirin

Contents

1. Introduction and motivation

Noble liquids such as liquid argon (LAr) and liquid xenon (LXe) are chemically inert, dense, scintillating, and transparent to their own scintillation light and ionization electrons.

Successful operation of LArTPCs experiments with long drift lengths (e.g. DUNE) relies on extremely low concentrations of electronegative impurities (O_2, H_2O, CO_2, N_2O…) as they hinder the free drifting of ionization electrons in the LAr volume.

2. Conventions

2.1. Naming

- `puritymonitor` (no dash, no underscore)
- PascalCase (a.k.a. UpperCamelCase) for `class`es (including `Exception`s with the suffix "`Error`") and `type`s
- SCREAMING_SNAKE_CASE for constants
- mixedCase (a.k.a. lowerCamelCase or medial capitals) for everything else, contrary to the PEP 8 – Style Guide for Python Code

2.2. Python type hints or annotations

[1]

3. Package scope

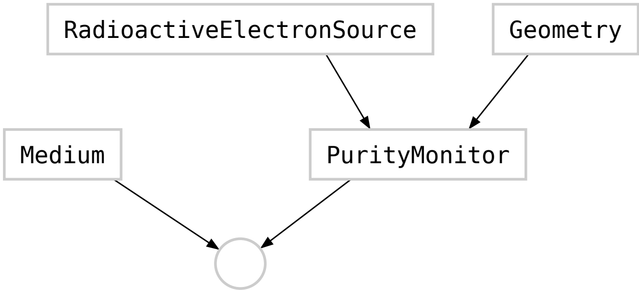
The `puritymonitor` package was initially intended for R&D about a ^{207}Bi LAr PM with a cylindrical TPC geometry with concentric inner disk and outer ring anodes. However, the package was designed with the intent of maximum orthogonality and flexibility, including

- other TPC geometries than a cylinder with concentric inner disk anode and outer ring anode, e.g. cuboids;
- other radioactive electron sources than ^{207}Bi;
- other noble liquids than liquid argon, e.g. liquid xenon;
- other Monte Carlo simulation algorithms.

To achieve this extensibility, abstract base classes for TPC geometries, radioactive electron sources, noble liquids, or Monte Carlo simulation provide a template for derived classes to implement a particular setup.

[1] See https://mypy.readthedocs.io/en/stable/cheat_sheet_py3.html.

4. Architecture



5. Package installation

Once installed, the `puritymonitor` package can be imported in Python files using

```
import puritymonitor
```

Its command line interface (CLI) can be called from any terminal using

```
python -m puritymonitor
```

```
python -m puritymonitor -h # help
```

5.1. Installation from PyPI or conda-forge (not implemented)

The `puritymonitor` package can be installed from PyPI using either `pip`

```
pip install ...
```

or `conda`

```
conda install ...
```

5.2. Development (editable) installation

The `puritymonitor` package can be installed such as to edit its source code using `git`

```
# Clone the package repository locally without the entire git history thanks to
# --depth 1
git clone --depth 1 https://github.com/bastienvoirin/puritymonitor.git
```

```
# Enter the package directory
cd puritymonitor
```

and `pip`

```
# Install the package found in the current directory in editable mode using -e
pip install -e .
```

6. Purity monitor Monte Carlo simulation

The physical model used in the `puritymonitor` Python package to account for the free electron drift relies on 5 intertwined parameters,

- $\mu(T)$, the electron mobility at temperature T in $(\mathrm{cm} \cdot \mathrm{\mu s}^{-1})/(\mathrm{V} \cdot \mathrm{cm}^{-1}) = \mathrm{cm}^2 \cdot \mathrm{\mu s}^{-1} \cdot \mathrm{V}^{-1}$
- E, the constant electric field in $\mathrm{V} \cdot \mathrm{cm}^{-1}$
- v_{drift}, the electron drift velocity in $\mathrm{cm} \cdot \mathrm{\mu s}^{-1}$
- d_{atten}, the mean electron attenuation distance in cm
- τ_e, the mean electron lifetime in $\mathrm{\mu s}$

which satisfy 2 independent equations,

- $(1)\ v_{\text{drift}} = \mu(T) \cdot E$
- $(2)\ d_{\text{atten}} = v_{\text{drift}} \cdot \tau_e$

This amounts to 2 or 3 independent parameters,

- $\mu(T)$ which has a sensible default value taken from the scientific literature
- either E or v_{drift}, the other one is computed from $\mu(T)$ and equation (1)
- either τ_e or d_{atten} , the other one is computed from v_{drift} and equation (2)

7. Purity monitor

7.1. `class PurityMonitor`

Abstract base `class`. As of version 0.1.0 of this package,

- `class PurityMonitorInitDecay(PurityMonitor)` is implemented,
- `class PurityMonitorInitDecayTimed(PurityMonitor)` is not implemented,
- `class PurityMonitorFullDecay(PurityMonitor)` is not implemented,
- `class PurityMonitorFullDecayTimed(PurityMonitor)` is not implemented.

However, some indications toward an implementation of the last three Monte Carlo simulations in future versions are provided in the corresponding sections.

7.1.1. Constructor

```
PurityMonitor(
  radioactiveElectronSource,
  geometry
)
```

- `radioactiveElectronSource: RadioactiveElectronSource`
  Radioactive electron source (see Section 10).

- `geometry: Geometry`
  Purity monitor TPC geometry (see Section 8).

Example

```
purityMonitor = PurityMonitor(
  radioactiveElectronSource = Bi207InLAr(),
  geometry = CylinderConcentricTwoPartAnode(
    innerRadius = 10,
    outerRadius = 20,
    driftLength = 30
  )
)
```

7.1.2. `__str__()`

Readable representation of the `PurityMonitor` instance.

```
str(purityMonitor)
```

7.1.3. `__repr__()`

Unambiguous, explicit representation of the `PurityMonitor` instance.

```
purityMonitor.__repr__()
```

7.1.4. `draw(ax)`

Draw the TPC geometry.

```
fig, ax = plt.subplots()
purityMonitor.draw(ax)
```

7.2. `class PurityMonitorInitDecay(PurityMonitor)`

Only consider the electron coming from the initial de-excitation of the ^{207}Bi nucleus to excited ^{207}Pb. The Monte Carlo simulation thus consists in a single draw of gamma photon or internal conversion electron emitted by the initial ^{207}Bi atom and ignores the subsequent de-excitations of the unstable ^{207}Pb nucleus.

7.3. `class PurityMonitorInitDecayTimed(PurityMonitor)` (not implemented)

The arrival times of the electrons at the anode(s) are taken into account to be able to veto overlapping signals with respect to a given time delta threshold.

7.4. `class PurityMonitorFullDecay(PurityMonitor)` (not implemented)

The Monte Carlo simulation accounts for the full decay chain of ^{207}Bi up to ground state ^{207}Pb.

7.5. `class PurityMonitorFullDecayTimed(PurityMonitor)` (not implemented)

The arrival times of the electrons at the anode(s) are taken into account to be able to veto overlapping signals with respect to a given time delta threshold.

8. Time projection chamber geometry

8.1. `class Geometry`

Any TPC geometry must inherit from the `Geometry` abstract base class and implement the `draw(ax)` method, like the built-in `class CylinderConcentricTwoPartAnode(Geometry)` which can be used as an inspiration.

As of version 0.1.0 of the `puritymonitor` package, `CylinderConcentricTwoPartAnode` is the only built-in TPC geometry and is detailed below.

8.1.1. `__str__()`

Readable representation of the `Geometry` instance.

```
str(geometry)
```

8.1.2. `__repr__()`

Unambiguous, explicit representation of the `Geometry` instance.

```
geometry.__repr__()
```

8.1.3. `draw(ax)`

Draw the TPC geometry.

```
fig, ax = plt.subplots()
geometry.draw(ax)
```

As `Geometry` itself is an abstract base `class`, attempting to call `draw(ax)` on an instance of `Geometry` instead of a class *derived* from `Geometry` will `raise` a `NotImplementedError`.

8.2. `class CylinderConcentricTwoPartAnode(Geometry)`

8.2.1. Constructor

```
CylinderConcentricTwoPartAnode(
  innerRadius,
  outerRadius,
  driftLength
)
```

- `innerRadius: float`
  Radius of the inner disk anode (or inner radius of the outer ring anode) in cm.
- `outerRadius: float`
  Outer radius of the outer ring anode in cm.
- `driftLength: float`
  Maximum drift length, i.e. distance between the radioactive electron source and the anode plane, in cm.

9. TPC medium

9.1. class LAr

9.1.1. Constructor

```
LAr()
```

10. Radioactive electron source

10.1. class RadioactiveElectronSource

Abstract base class.

10.1.1. Constructor

```
RadioactiveElectronSource(
  electronEnergy: list,
  gammaEnergy: list,
  electronProba: list,
  gammaProba: list,
  activity: float,
  description: str
)
```

- `electronEnergy: list = []`
  List of possible energies of emitted (internal conversion) electrons in MeV.
- `gammaEnergy: list = []`
  List of possible energies of emitted gamma photons in MeV.
- `electronProba: list = []`
  List of relative emission probabilities of (internal conversion) electrons.
- `gammaProba: list = []`
  List of relative emission probabilities of gamma photons.
- `activity: float = float("NaN")`
  Activity of the radioactive electron source in Bq. Only used in timed Monte Carlo simulations which are not implemented yet.
- `description: str = "Unspecified radioactive electron source"`

10.1.2. decay(nEvents)

Generator function for high-energy decay products (gamma rays or internal conversion electrons).

10.1.2.1. Example

```
for energy, isElectron in radioactiveElectronSource.decay(nEvents = 1000000):
  pass
```

10.2. class Bi207InLAr(RadioactiveElectronSource)

class derived from RadioactiveElectronSource defining the decay of ^{207}Bi in LAr.

10.2.1. Constructor

```
Bi207InLAr(
  electronEnergy: list,
  gammaEnergy: list,
  electronProba: list,
  gammaProba: list,
  activity: float,
  description: str
)
```

- `electronEnergy: list = []`
  List of possible energies of emitted (internal conversion) electrons in MeV.
- `gammaEnergy: list = []`
  List of possible energies of emitted gamma photons in MeV.
- `electronProba: list = []`
  List of relative emission probabilities of (internal conversion) electrons.
- `gammaProba: list = []`
  List of relative emission probabilities of gamma photons.
- `activity: float = float("NaN")`
  Activity of the radioactive electron source in Bq. Only used in timed Monte Carlo simulations which are not implemented yet.
- `description: str = "Bi-207 radioactive electron source in LAr"`

10.2.2. decay(nEvents)

Generator function for high-energy decay products (gamma rays or internal conversion electrons).

10.2.2.1. Example

```
for energy, isElectron in bismuth207.decay(nEvents = 1000000):
  pass
```

11. Command-line interface

11.1. Data file format unification

A unified file format decouples the Monte Carlo simulation output and experimental data analysis input from the particular experimental setup used. Data exchange and collaboration as well as custom figure generation are all made easier by such a unified data file format, at the cost of converting experimental data coming from the acquisition apparatus first.

11.1.1. Data file format description

The `puritymonitor` package expects to read and writes CSV files (i.e. comma-separated values over multiple lines) of electron energy spectra whose first column gives the lower energy bounds of the bins (i.e. rows) and header line consists in comma-separated labels for the energy column and for each spectrum column.

Example

```
Energy (MeV),Inner disk anode,Outer ring anode
0.0,0.0,0.0
0.01,0.0,0.0
0.02,0.0,0.0
0.03,0.0,0.0
0.04,0.0,0.0
...
0.98,80.0,80.0
0.99,90.0,90.0
1.00,100.0,100.0
1.01,90.0,90.0
1.02,80.0,80.0
...
1.96,0.0,0.0
1.97,0.0,0.0
1.98,0.0,0.0
1.99,0.0,0.0
2.0,0.0,0.0
```

11.1.2. Data file format conversion from the oscilloscope used at B182 at CERN

```
puritymonitor.convert_b182_osc(*args)
```

```
python -m puritymonitor.convert_b182_osc *args
```

Short	Long	Type	Description
`-d`	`--input-dir`	`path`	Data files directory
`-is`	`--inner-short`	`path`	Short PM inner anode data filename
`-il`	`--inner-long`	`path`	Long PM inner anode data filename
`-os`	`--outer-short`	`path`	Short PM outer anode data filename
`-ol`	`--outer-long`	`path`	Long PM outer anode data filename
`-s`	`--short`	`path`	Output filename for short PM data
`-l`	`--long`	`path`	Output filename for long PM data

Table 1: Arguments to the `python -m puritymonitor.convert_b182_osc` command.

11.1.3. Data file format conversion from the multichannel analyzer used at B182 at CERN

```
puritymonitor.convert_b182_mca(*args)
```

```
python -m puritymonitor.convert_b182_mca *args
```

Short	Long	Type	Description
`-d`	`--input-dir`	`path`	Data files directory
`-is`	`--inner-short`	`path`	Short PM inner anode data filename
`-il`	`--inner-long`	`path`	Long PM inner anode data filename
`-os`	`--outer-short`	`path`	Short PM outer anode data filename
`-ol`	`--outer-long`	`path`	Long PM outer anode data filename
`-s`	`--short`	`path`	Output filename for short PM data
`-l`	`--long`	`path`	Output filename for long PM data

Table 2: Arguments to the `python -m puritymonitor.convert_b182_mca` command.

11.2. Main command

11.2.1. `CylinderConcentricTwoPartAnode` geometry

```
python -m puritymonitor.cctpa *args
```

Short	Long	Type	Description	Unit
`-e`	`--events`	`int`	Number of events	
`-f`	`--field`	`float`	Electric field	V
`-a`	`--atten`	`float`	Electron attenuation distance	cm
`-l`	`--length`	`float`	Drift length between cathode and anode planes	cm
`-ir`	`--inner-radius`	`float`	Inner disk anode radius	cm
`-or`	`--outer-radius`	`float`	Outer ring anode radius	cm
`-r`	`--relative-scale`	`list[float]`	Relative scaling between the inner disk anode spectrum and outer ring anode spectrum.	
`-g`	`--geom`	Flag	TPC geometry visualization	
`-d`	`--data`	Flag	Experimental data	
`-s`	`--simu`	Flag	Monte Carlo simulation	

Table 3: Arguments to the `python -m puritymonitor.cctpa` command.

12. Package structure summary

```
class PurityMonitor
  class PurityMonitorInitDecay(PurityMonitor)
  class PurityMonitorInitDecayTimed(PurityMonitor)
  class PurityMonitorFullDecay(PurityMonitor)
  class PurityMonitorFullDecayTimed(PurityMonitor)
```

```
class RadioactiveElectronSource
  class Bi207InLAr(RadioactiveElectronSource)
```

```
class Geometry
  class CylinderConcentricTwoPartAnode(Geometry)
```

```
class LAr
```

```
class EnergySpectra
```

13. Example

Dual cylindrical ^{207}Bi LAr purity monitor (one of 6 cm, one of 18 cm drift length) with two concentric anodes like the one developed at CERN and INFN Padova.

13.1. Python code

```python
from puritymonitor import (PurityMonitor, Bi207InLAr, CylindricConcentricTwoPartAnode)

purityMonitor = PurityMonitor(
  radioactiveElectronSource = Bi207InLAr(),
  geometry = CylinderConcentricTwoPartAnode(
    innerRadius = 10,
    outerRadius = 20,
    driftLength = 30
  )
)

fig, (axGeom, axSimu, axData) = plt.subplots(3)

purityMonitor.draw(ax = axGeom)

purityMonitor
  .simulate(events = 1000000)
  .plot(ax = axSimu, minEnergy = 0.0, maxEnergy = 2.0)
```

13.2. Command line interface

First convert experimental data to unified file format by running

```
python
  -m puritymonitor.convert_b182_osc
  -d INPUT_FILES_PARENT_DIRECTORY
  -is FILENAME_INNER_S_FROM_OSCILLOSCOPE
  -il FILENAME_INNER_L_FROM_OSCILLOSCOPE
  -os FILENAME_OUTER_S_FROM_OSCILLOSCOPE
  -ol FILENAME_OUTER_L_FROM_OSCILLOSCOPE
  -s OUTPUT_FILENAME_INNER_OUTER_S
  -l OUTPUT_FILENAME_INNER_OUTER_L
```

for data coming from the oscilloscope, or

```
python
  -m puritymonitor.convert_b182_mca
  -d INPUT_FILES_PARENT_DIRECTORY
  -is FILENAME_INNER_S_FROM_ANALYZER
  -il FILENAME_INNER_L_FROM_ANALYZER
  -os FILENAME_OUTER_S_FROM_ANALYZER
  -ol FILENAME_OUTER_L_FROM_ANALYZER
  -s OUTPUT_FILENAME_INNER_OUTER_S
  -l OUTPUT_FILENAME_INNER_OUTER_L
```

for data coming from the multichannel analyzer.

Then

```
python
  -m puritymonitor.cctpa
  -e 1000000 # Number of events
  -f 1000 # Electric field in V/cm
  -ir 1.5
  -or 3.0
  -l 6.5 18.5

  -g # Draw PM TPC geometries
  -s # Plot simulation
```

```
python
  --module puritymonitor.cctpa
  --events 1000000 # Number of events
  --field 1000 # Electric field in V/cm
  --inner-radius 1.5
  --outer-radius 3.0
  --drift-length 6.5 18.5

  --geom # Draw PM TPC geometries
  --simu # Plot simulation
```

14. Contribution

This package is still under migration/development from its initial (legacy) unpublished version of August.

15. Acknowlegments

The `puritymonitor` package has been developed as part of the **CERN Summer Student Programme 2024** under the supervision of **Francesco Pietropaolo** (CERN) from his Monte Carlo simulation script, data analysis scripts and experimental data from **Robert Gan** (Boston University), and experimental data from **Gajendra Gurung** (University of Texas at Arlington) with the intent of contributing to the ongoing R&D on a novel ^{207}Bi-based LAr purity monitor at CERN and INFN Padova.