PySiPM

1.0

Generated by Doxygen 1.8.20

Chapter 1

pySiPM

Toolkit to simulate SiPM events

Link to docs

Chapter 2

Modules Index

2.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 3

Module Documentation

3.1 frandom Module Reference

F2PY extension for Python. Random arrays generation in Fortran.

Functions/Subroutines

- subroutine integer (out, sup, n)

  Generation of random integers in range [0 - sup].
- subroutine normal (out, mu, sigma, n)

  Generation of gaussian random values using Box-Muller transform.
- subroutine poisson (out, mu, n)

  Generation of poissonian random values.
- subroutine exponential (out, mu, n)

  Generation of exponential random values.

3.1.1 Detailed Description

F2PY extension for Python. Random arrays generation in Fortran.

Author

Edoardo Proserpio

3.1.2 Function/Subroutine Documentation

3.1.2.1 exponential()

```
subroutine frandom::exponential (
             real(8), dimension(n) out,
             real(8) mu,
             integer(8) n )
```

Generation of exponential random values.

Parameters

mu	Mean value of exponential distribution
n	Number of elements to generate
out	

Definition at line 151 of file FortranFunctions.f90.

3.1.2.2 integer()

```
subroutine frandom::integer (
             integer(8), dimension(n) out,
             integer(8) sup,
             integer(8) n )
```

Generation of random integers in range [0 - sup].

Parameters

sup	Superior limit for random generation
n	Number of elements to generate
out	

Definition at line 66 of file FortranFunctions.f90.

3.1.2.3 normal()

```
subroutine frandom::normal (
             real(8), dimension(n) out,
             real(8) mu,
             real(8) sigma,
             integer(8) n )
```

Generation of gaussian random values using Box-Muller transform.

Parameters

mu	Mean value of gaussian distribution
sigma	Standard deviation of gaussian distribution
n	Number of elements to generate
out	

Definition at line 87 of file FortranFunctions.f90.

3.1.2.4 poisson()

```
subroutine frandom::poisson (
             integer(8), dimension(n) out,
             real(8) mu,
             integer(8) n )
```

Generation of poissonian random values.

Parameters

mu	Mean value of poissonian distribution
n	Number of elements to generate
out	

Definition at line 116 of file FortranFunctions.f90.

3.2 lib Namespace Reference

Functions

- def addDCR (rate)

  Generation of dark count events.
- def HitCells (n)

  Generation of cell IDs.
- def addXT (times, idx, xt)

  Generation of optical crosstalk events.
- def addAP (times, h, ap)

  Generation of afterpulses.
- def SiPMEventAction (times, idx)

  Calculates relative signal height for each photoelectron.
- def SiPMSignalAction (times, sigH, snr=SNR, basespread=0)

  Generation of full SiPM signal.
- def PulseCPU (t, h)

  Generation of single cell signals.
- def signalAnalysis (signal, intstart, intgate, threshold)

  Features extraction from signals.
- def somestats (output, realpe=None)

  Function that displays histograms of generated events.
- def sigPlot (signal, npe, ndcr, dev)

  Plot each signal pulse produced in the simulation.
- def initializeRandomPool (seed=None)

  Function that initializes random seeds for each worker in the multiprocessing Pool.

Variables

- signalmodel = fsignal(0, TF, TR, 1, SIGPTS)

  Array containing the signal shape of the SiPM intended as the ideal signal generated by a signle photoelectron at time 0 ns.

3.2.1 Detailed Description

```
In this file I define all the functions I will use in the main file of simulation.
```

3.2.2 Function Documentation

3.2.2.1 addAP()

```
def lib.addAP (
              times,
              h,
              ap )
```

Generation of afterpulses.

Function that generates afterpulses(AP) events and adds them to the list of events to simulate. Each avalanche generates a poissonian number of AP events. APs are delayed from teyr main signal following a fast and slow exponential distributions. Theyr relative signal height is calculated consdering the cell recovery time as an RC circuit:

$$h = 1 - e^{-\frac{\Delta_t}{\tau}}$$

Parameters

times	List containing the time at which SiPM cells are fired, including DCR and XT events.
h	List containing the relative signal height of each fired cell.
ap	Value (probability) of AP events.

Returns

times: List containing the time at which SiPM cells are fired, including AP events.

h: List containing the relative signal height of each hitted cell.

nAp: Number of Ap events generated.

Definition at line 84 of file lib.py.

3.2.2.2 addDCR()

```
def lib.addDCR (
              rate )
```

Generation of dark count events.

Function that generates times for dark count events (DCR). Events are generated as an uniform poisson process, hence the time distance between consecutive DCR events follows an exponential distribution with a mean value of: $\tau = \frac{1}{DCR}$

Parameters

rate	Dark counts rate un Hz.

Returns

dcrTime: List containing times of DCR events in ns.

nDcr: Number of DCR events generated

Definition at line 7 of file lib.py.

3.2.2.3 addXT()

```
def lib.addXT (
              times,
              idx,
              xt )
```

Generation of optical crosstalk events.

Function that generates optical crosstalk events(XT). Each photoelectron has generates a poissonian number of XT events accordingly to the crosstalk probability. XT events are generated in the 8 neighbouring cells and have the same time of the main cell.
XT events can also generate other XT events in theyr neighbouring cells.

Parameters

times	List containing the time of each SiPM cell avalanche, including DCR events.
idx	List containing the ID of each fired cell.
xt	Value (probability) of XT events.

Returns

times: List containing the time at which SiPM cells are fired including XT events

idx: List containing the ID of the hitted cells including XT events.

nXt: Number of Xt events generated.

Definition at line 48 of file lib.py.

3.2.2.4 HitCells()

```
def lib.HitCells (
              n )
```

Generation of cell IDs.

Function that generated cell IDs for each photoelectron in the list.
Cell IDs are integers in range [0 - NCELLS] and are generated randomly. An ID can appear multiple times, meaning that the corresponding cell has generated an avalanche due to a photoelectron multiple times.

Parameters

n	Number of photoelectrons that have to be simulated.

Returns

idx: List containing the ID of each hitted cell.

Definition at line 32 of file lib.py.

3.2.2.5 initializeRandomPool()

```
def lib.initializeRandomPool (
              seed = None )
```

Function that initializes random seeds for each worker in the multiprocessing Pool.

This function extracts seeds using os.urandom, so from the operating system entropy pool, and uses the to set the seeds for the random generator and numpy random generator. Seeds need to be different for each worker, otherwise the random values are generate equally among workers.

Definition at line 440 of file lib.py.

3.2.2.6 PulseCPU()

```
def lib.PulseCPU (
              t,
              h )
```

Generation of single cell signals.

Function that generates the signal from a single SiPM cell. This is the "fast" version that uses a pre-computed signal shape and moves it in the "right" position in time.

Parameters

t	Time at which the cell is triggered.
h	The relative pulse height of the cell signal
gainvar	Value of cell to cell gain variation for this signal.

Returns

s: Array containing the generated cell signal

Definition at line 218 of file lib.py.

3.2.2.7 signalAnalysis()

```
def lib.signalAnalysis (
              signal,
              intstart,
              intgate,
              threshold )
```

Features extraction from signals.

Function that extracts some simple features from signals considering an integration gate. At the moment the features considered are:

- Peak: signal peak in the integration gate.
- Integral: sum of samples in the integration gate corrected by the sampling time.
- Time of Arrival: time position of the first sample above the threshold.
- Time over Threshold: Number fo samples above the threshold corrected by the sampling time.
- Time of Peak: time position of the sample in the peak. This list might be expanded in future with more complex features.

Parameters

signal	Array containing the digitized signal.
intstart	Integer value containing the index corresponding to the start of the integration gate.
intgate	Integer value containing the lenght of the integration gate in units of samples.
threshold	Threshold used to calculate the values described above. It is automatically set to 1.5 times the single photoelectron peak height.

Returns

Returns the features extracted. If thesignal is below the threshold all features are set to -1.

Definition at line 245 of file lib.py.

3.2.2.8 sigPlot()

```
def lib.sigPlot (
              signal,
              npe,
              ndcr,
              dev )
```

Plot each signal pulse produced in the simulation.

Function used for debug purposes only. It creates a window for each worker and draws the signals as they are generated.

Parameters

signal	Array containing the generated SiPM signal.
npe	Total number of hitted cells.
ndcr	Total number of DCR events.
dev	String that describes the device on which the signal is computed. (cpu / cpu-fast / gpu)

Definition at line 379 of file lib.py.

3.2.2.9 SiPMEventAction()

```
def lib.SiPMEventAction (
              times,
              idx )
```

Calculates relative signal height for each photoelectron.

Since a SiPM cell may be hitted multiple times, recovery time has to be considered. If all cell IDs are different all signal heights are left unchanged with a value of 1, else if an ID appears multiple times then the first (in time) hit gives a signal height of 1 and the following will have a lower relative height. The relative signal height, after a Δ_t time from the previous hit, is calculated supposing that each SiPM cell recovers as an RC circuit:

$$h(\Delta_t) = 1 - e^{-\frac{t}{\tau}}$$

Parameters

times	List containing the time of all events that have generated an avalanche in the SiPM cells.
idx	List containing the corresponding cell IDs of the events.

Returns

h: List containing the relative signal height of each avalanche.

Definition at line 127 of file lib.py.

3.2.2.10 SiPMSignalAction()

```
def lib.SiPMSignalAction (
              times,
              sigH,
              snr = SNR,
              basespread = 0 )
```

Generation of full SiPM signal.

Function that generates the full SiPM signal as the sum of the signals of each cell starting from gaussian noise.

Parameters

times	List containing the time at wich SiPM cells are fired, including DCR, XT and AP events
sigH	List containing the correspondin relative pulse height of each fired cell.
snr	Signal to noise ratio converted into the RMS of the gaussian noise.
basespread	Sigma of the value to add as baseline spread.

Returns

signal: Array containing the complete sigitized SiPM signal.

Definition at line 176 of file lib.py.

3.2.2.11 somestats()

```
def lib.somestats (
              output,
              realpe = None )
```

Function that displays histograms of generated events.

Parameters

output	Array containing the signal features.
realpe	Optional parameter. Array containing the real number of photoelectrons given to the simulation.

See also

signalAnalysis for a description of the signal features.

Definition at line 293 of file lib.py.

3.3 libCPU Namespace Reference

Functions

- def PulseCPU (t, h)

  Generation of single cell signals.

3.3.1 Detailed Description

```
In this file I define all the functions I will use in the main file of simulation.
```

3.3.2 Function Documentation

3.3.2.1 PulseCPU()

```
def libCPU.PulseCPU (
              t,
              h )
```

Generation of single cell signals.

Function that generates the signal from a single SiPM cell. This is the "full" version that calculates each signal considering a double exponential function.

Parameters

t	Time at which the cell is triggered.
h	The relative pulse height of the cell signal
gainvar	Value of cell to cell gain variation for this signal.

```
@return s: Array containing the generated cell signal
```

Definition at line 6 of file libCPU.py.

3.4 libGPU Namespace Reference

Functions

- def PulseGPU (t, h)

  Function that generates the signal from all SiPM cells at once.
- def SiPMSignalAction (times, sigH, snr, basespread)

  Generation of full SiPM signal.

3.4.1 Detailed Description

```
In this file I define all the functions I will use in the main file of simulation.
```

3.4.2 Function Documentation

3.4.2.1 PulseGPU()

```
def libGPU.PulseGPU (
              t,
              h )
```

Function that generates the signal from all SiPM cells at once.

This is the "full" version that computes the signal shape on GPU. The signals are generated at once using a CUDA kernel that generates the signals of each cell in a matrix (one cell per row) and then sums everything column-wise obtaining the complete SiPM signal.

Parameters

t	Array containing times at which each the cell is triggered
h	Array containing the relative pulse height of each cell signal

Returns

s: Array containing the generated SiPM signal

Definition at line 17 of file libGPU.py.

3.4.2.2 SiPMSignalAction()

```
def libGPU.SiPMSignalAction (
              times,
              sigH,
              snr,
              basespread )
```

Generation of full SiPM signal.

Function that generates the full SiPM signal as the sum of the signals of each cell starting from gaussian noise. If there are fewer hitted cells the signal is generated on CPU to avoid the overhead caused by moving data to GPU memory. Also if there are too many hitted cells the signal is generated on CPU to avoid an out-of-memory condition since usually VRAM is smaller than system memory. Those conditions are controlled by variables.CPUTHRESH↩ OLD and variables.GPUMAX and are set by default at 100 and 2000.

Parameters

times	List containing the time at wich SiPM cells are fired, including DCR, XT and AP events
sigH	List containing the correspondin relative pulse height of each fired cell.
snr	Signal to noise ratio converted into the RMS of the gaussian noise.
basespread	Sigma of the value to add as baseline spread.

Returns

signal: Array containing the complete sigitized SiPM signal.

Definition at line 47 of file libGPU.py.

3.5 main Namespace Reference

File containing the main function used to simulate SiPM events.

Functions

- def SiPM (times, other=None)

  @biref that calls all the procedures defined in libs to generate a complete SiPM event.

3.5.1 Detailed Description

File containing the main function used to simulate SiPM events.

Author: Edoardo Proserpio Email: eproserpio@studenti.uninsubria.it edoardo.proserpio@gmail.com

3.5.2 Function Documentation

3.5.2.1 SiPM()

```
def main.SiPM (
              times,
              other = None )
```

@biref that calls all the procedures defined in libs to generate a complete SiPM event.

This function is the main function used to simulate a complete SiPM event. It calls all the other methods with the correct parameters and user settings.

Parameters

times	This list contains the arriving time of each photon on the SiPM sensor surface. This list is the main input of the simulation.
other	This variable may contain other informations about the event generated. It can be the event id, the arriving time inserted in the simulation or the real number of photons inserted in the simulation. This tuple will be copied as it is in the output.

Returns

integral The integral of the signal calculated in the integration gate.

peak The height of the signal in the integration gate.

tstart The time of arrival of the signal in ns defined as the first sample over the threshld of 1.5

other The copy of the `other` variable given in the input.

signal If the options -W is enabled the complete SiPM signal will be passed in the output. Otherwise this output is "None".

Definition at line 9 of file main.py.

3.6 variables Namespace Reference

3.6.1 Detailed Description

```
In this file I define all the global variables that I will use in other files.
```

Index