GeoEfficiency.jl

Mohamed E. Krar

April 3, 2019

Contents

Part I

Home

Chapter 1

GeoEfficiency: Accurate Geometrical Efficiency Calculator

an officially registered Julia program, provides a set of tools to calculate the geometrical efficiency in a fast and accurate way. The Package models a radiation detector irradiated by a radioactive source. The Package relay directly on numerical evaluation of closed form analytical formula describing the geometrical efficiency.

Author	Mohamed Krar (DrKrar@gmail.com)
Repository	GitHub.com
Documentation	GitHub.io
Current version	v"0.9.3-dev"
First Created	Fri Aug 14 20:12:01 2015

this documentation is also available in pfd formate.

1.1 The following list show the current and planed features:-

the checked items represent allready present feature.

- x support of widely used detector geometries.
    - x `cylinder` detectors.
    - x `bore-hole` detectors.
    - x `well-type` detectors.
- support of specialized detector geometries.
- x support of isotropic radioactive sources.
    - x `point` sources.
    - x `disc` sources.
    - x `cylinder` sources.
- support of anisotropic radioactive sources.
    - `point` sources.

consider more details of the measurement setup.

the detector effect.

the end cap effect.

the medium and absorber effect.

combine the effect of the source geometry and composition.

1.2 Requirements

- Julia 0.6 or above.
- QuadGK 0.3.0 or above, will be installed automatically during the package Installation.
- Compat 0.63.0 or above, will be installed automatically during the package Installation.

1.3 Download and Install the Package

The package is registered and so can be installed through the Julia package system by running

```
julia> using Pkg
julia> Pkg.add("GeoEfficiency")
```

1.4 Quick Usage

```
julia> using GeoEfficiency
julia> calc()
```

see also: **`geoEff()`**, **`calcN()`**, **`batch()`**

1.5 Package Overview

The following constructor can be used to construct a specific type of detector

- `CylDetector` for cylindrical detector,
- `BoreDetector` for bore hole,
- `WellDetector` for well type detector.

While the function `Detector` can be used to construct any of the above types. You may try also `getDetectors`.

`Point` constructor is used to construct an anchoring point of a source relative to it its position to the detector is specified. For a point source, the anchoring point is the source itself. The `source()` method take input from the 'console' and return a tuple describing the source.

The efficiency calculation can be done by one of the functions:

- `geoEff` used with or without argument(s),
- `calc` ask for the required information from the 'console',

- `calcN` just a repeat of the `calc` function
- `batch()` which try to take required information from csv files located in the home directory inside a folder called `GeoEfficiency`.

For more on the function and its methods prefix the name of the function by ?.

> **Note**
>
> Input from the 'console' can be numerical expression not just a number.
>
> **5/2, 5//2, pi, exp(2) , 1E-2, 5.2/3, sin(1), pi/2/3 All are valid expressions.**

1.6 Batch Calculation

The package can be used to perform batch calculations by calling one of the methods of the function `batch`. The output results of batch calculations is found by default in `GeoEfficiency\results` folder inside the user home.

For example `c:\users\yourusername\GeoEfficiency\results\`.

The function `batch()` can be called with or without arrangement(s). The without argument version relay on previously prepared Comma Saved Values [CSV] files, that can be easily edit by Microsoft Excel, located by default in the `GeoEfficiency` folder.

Those Comma Saved Values [CSV] files are:-

- `Detectors.csv` contains the detectors description; The line format is:

```
        Crystal_Radius | Crystal_Length | Hole_Radius | Hole_Depth |
        ---------------| ---------------|-------------|----------- |
```

- `srcHeights.csv` contains the source heights;

```
            Source_Heights |
        ---------------|
```

- `srcRhos.csv` contains the source off-axis distances;

```
        Source_Rhos |
        ------------|
```

- `srcRadii.csv` contains the source radii for disc and cylindrical sources;

```
        Source_Radii|
        ------------|
```

- `srcLengths.csv` contains the source length for cylindrical sources;

```
    Source_Lengths|
    --------------|
```

Note

for Comma Saved Values [CSV] files each line represent an entry, the first line is always treated as the header.

Warning

the program expect each line to contain one number for all CSV files except for $Detectors.csv$ each line should contain at least one number or at most four separated numbers

Part II

Manual

Chapter 2

GeoEfficiency

GeoEfficiency.GeoEfficiency – Module.

GeoEfficiency Package

introduce a fast and flexible tool to calculate in batch or individually the `geometrical efficiency` for a set of common radiation detectors shapes (cylindrical,Bore-hole, Well-type) as seen form a source. The source can be a point, a disc or even a cylinder.

Quick Usage

- geoEff() : Calculate the geometrical efficiency for one geometrical setup return only the value of the geometrical efficiency.
- calc() : Calculate the geometrical efficiency for one geometrical setup and display full information on the console.
- calcN() : Calculate the geometrical efficiency for geometrical setup(s) and display full information on the console until the user quit.
- batch() : Calculate the geometrical efficiency using data in the **`/home/travis/GeoEfficiency`** folder in batch mode.

for more information and updates refer to the repository at `GitHub.com`

source

GeoEfficiency.about – Function.

```
 **************************************************
 **            -=) GeoEfficiency (=-            **
 **  Accurate Geometrical Efficiency Calculator **
 **   First Created on Fri Aug 14 20:12:01 2015 **
 **************************************************

 Author:        Mohamed E. Krar,  @e-mail: DrKrar@gmail.com
 Auth_Profile:  https://www.researchgate.net/profile/Mohamed_Krar3
 Repository:    https://github.com/DrKrar/GeoEfficiency.jl/
 Version:       v"0.9.3-DEV" - (4 days old master)
 Documentation: http://geoefficiencyjl.readthedocs.org



Batch mode
```

```
-  read files by defaul from directory `/home/travis/GeoEfficiency`
-  save results by default to directory `/home/travis/GeoEfficiency/results`

for more information see `batch`, `batchInfo`.
```

source

Chapter 3

Error

GeoEfficiency.GeoException – Type.

coustom abstract Exception that is the parent of all exception in the GeoEfficiency package

source

GeoEfficiency.InValidDetectorDim – Type.

coustom Exception indicating inValid radiation detector dimentions

source

GeoEfficiency.@validateDetector – Macro.

```
@validateDetector cond [text]
```

Throw an InValidDetectorDim if cond is false. Message text is optionally displayed upon validation failure.

Examples

```
julia> @validateDetector iseven(3) "3 is an odd number!"
ERROR: InValidDetectorDim: 3 is an odd number!

julia> @validateDetector isodd(3) "What even are numbers?"
```

source

GeoEfficiency.NotImplementedError – Type.

coustom Exception source to detector condation which may be valid but not implemented yet

source

GeoEfficiency.@notImplementedError – Macro.

coustom macro to throw NotImplementedError Exception

source

Chapter 4

Console Input

`GeoEfficiency.input` – Function.

UnExported

```
input(prompt::AbstractString = "? ", incolor::Symbol = :green)
```

return a string delimited by new line excluding the new line. prompt the user with the massage `prompt` defaults to ?. `incolor` specify the prompt text color, default to $green$.

source

`GeoEfficiency.getfloat` – Function.

UnExported

```
getfloat(prompt::AbstractString = "? ", from::Real = -Inf, to::Real = Inf; KW...)::Float64
```

prompts the user with the massage `prompt` defaults to `?` to input a numerical expression evaluate to a numerical value and asserts that the value is by default in the semi open interval [`from`, `to`[before returning it as a `Float64`. throws `ArgumentError` when the given interval is not valid.

KW arguments

- value::AbstractString=`"nothing"` : if provided the function will not ask for input from the `console` and take it as if it where inputted from the `console` [`for test propose mainly`].
- lower::Bool=`true` : whether or not to inculde `from` as accepted value.
- upper::Bool=`false` : whether or not to inculde `to` as accepted value.

  Note

  - a blank input (i.e just a return) is considered as being `0.0`.
  - input from the `console` can be numerical expression not just a number.
  - All `5/2`, `5//2`, `exp(2)`, `pi`, `1E-2`, `5.2/3`, `sin(1)`, `pi/2/3` are valid mathematical expressions.

Examples

```
julia> getfloat("input a number:", value="3")
3.0

julia> getfloat("input a number:", value="")
0.0
```

```
julia> getfloat("input a number:", value="5/2")
2.5

julia> getfloat("input a number:", value="5//2")
2.5

julia> getfloat("input a number:", value="pi")
3.141592653589793

julia> getfloat("input a number:", value="-2")
-2.0

julia> getfloat("input a number:", 1, 5, value="5", upper=true)
5.0
```

source

Chapter 5

Physics Model

`GeoEfficiency.Point` – Type.

```
Point(Height::Real, Rho::Real)
```

construct and return a `Point` source that can be used as either a source by itself or an `anchor point` of a higher dimension source.

- `Height` : point height relative to the detector surface.
- Rho : point off-axis relative to the detector axis of symmetry.

Note

Each detector type give different interpretation to the `height` as follow:-

- for `CylDetector` the point source `height` is consider to be measured from the detector `face surface`.
- for `BoreDetector` the point source `height` is consider to be measured from the `detector middle`, +ve value are above the detector center while -ve are below.
- for `WellDetector` the point source `height` is considered to be measured from the detector `hole surface`.

source

`GeoEfficiency.source` – Function.

```
source(anchorPnt::Point = Point())
```

return a tuple that describe the source (`anchorPnt`, `SrcRadius`, `SrcLength`) according to the input from the `console`.

- `anchorPnt` : the source anchoring point. if it is missing the user is prompt to input it via the `console`.
- `SrcRadius` : source radius.
- `SrcLength` : source length.

Warning

If the global variable `srcType` is set to $srcPoint$, both `SrcRadius` and `SrcLength` are set to zero.

source

`GeoEfficiency.CylDetector` – Type.

```
CylDetector(CryRadius::Real, CryLength::Real)
```

construct and return a `cylindrical` detector of the given crystal dimensions:-

- `CryRadius` : the detector crystal radius.
- `CryLength` : the detector crystal length.

> **Warning**
> both `CryRadius` and `CryLength` should be `positive`, while `CryLength` can also be set to **`zero`**.

source

`GeoEfficiency.BoreDetector` – Type.

```
BoreDetector(CryRadius::Real, CryLength::Real, HoleRadius::Real)
```

construct and return a `bore-hole` detector of the given crystal dimensions:-

- `CryRadius` : the detector crystal radius.
- `CryLength` : the detector crystal length.
- `HoleRadius` : the detector hole radius.

> **Warning**
> `CryRadius` and `CryLength`, `HoleRadius` should be `positive` numbers, also `CryRadius` should be greater than `HoleRadius`.

source

`GeoEfficiency.WellDetector` – Type.

```
WellDetector(CryRadius::Real, CryLength::Real, HoleRadius::Real, HoleDepth::Real)
```

construct and return a `Well-Type` detector of the given crystal dimensions:-

- `CryRadius` : the detector crystal radius.
- `CryLength` : the detector crystal length.
- `HoleRadius` : the detector hole radius.
- `HoleDepth` : the detector hole length.

> **Warning**
> all arguments should be `positive` numbers, also `CryRadius` should be greater than `HoleRadius` and `CryLength` should be greater than `HoleDepth`.

source

`GeoEfficiency.RadiationDetector` – Type.

abstract super-supertype of all detectors types

source

`GeoEfficiency.Detector` – Type.

```
Detector
```

abstract supertype of all detectors types of cylidericalish shapes. also can be used to construct any leaf type.

source

Chapter 6

Batch Input

`GeoEfficiency.typeofSrc` – Function.

```
typeofSrc()::SrcType
```

return the current value of the global `GeoEfficiency.srcType`.

source

```
typeofSrc(x::Int)::SrcType
```

set and return the value of the global `GeoEfficiency.srcType` corresponding to x.

- srcUnknown = -1 also any negative integer treated as so,
- srcPoint = 0,
- srcLine = 1,
- srcDisk = 2,
- srcVolume = 3,
- srcNotPoint = 4 also any greater than 4 integer treated as so.

source

`GeoEfficiency.setSrcToPoint` – Function.

```
setSrcToPoint()::Bool
```

return whether the source type is a point or not.

source

```
setSrcToPoint(yes::Bool)::Bool
```

return whether the source type is a point or not after setting `srcType` to `srcPoint` if `yes = true` else if `yes = false` setting it to `srcNotPoint` if it was not already set to other non-point type (`srcDisk`, `srcLine`, `srcVolume`).

Note

- The user can use this function to change the source type any time.
- The source type is set the fist time asked for source.

see also: `typeofSrc(::Int)`.

source

```
setSrcToPoint(prompt::AbstractString)::Bool
```

return whether the source type is a point or not. only prompt the user to set the source type if it were not already set before.

see also: `typeofSrc(::Int)`, `setSrcToPoint(::Bool)`.

source

`GeoEfficiency.detector_info_from_csvFile` – Function.

UnExported

```
 detector_info_from_csvFile(detectors::AbstractString = Detectors,
                                  datadir::AbstractString = dataDir)
```

return a vector{Detector} based on information in the file of name `detectors` found in the directory `datadir`.

Note

- if no path is given the second argument `datadir` is default to `/home/travis/GeoEfficiency` as set by the constant `dataDir`.
- if no file name is specified the name of the predefined file `Detectors.csv` as set by the constant `Detectors`.
- the no argument method is the most useful; other methods are mainly for `test propose`.

source

`GeoEfficiency.read_from_csvFile` – Function.

UnExported

```
read_from_csvFile(csv_data::AbstractString,
                   datadir::AbstractString = dataDir)::Vector{Float64}
```

return Vector{Float64} based on data in csv file named `csv_data`. directory `datadir` point to where the file is located default to $/home/travis/GeoEfficiency$ as set by the constant `dataDir`.

source

`GeoEfficiency.read_batch_info` – Function.

UnExported

```
read_batch_info()
```

read `detectors` and `sources` parameters from the predefined csv files.

Return a tuple (detectors*array, srcHeights*array, srcRhos*array, srcRadii*array, srcLengths*array, GeoEfficiency*isPoint)

source

UnExported

```
read_batch_info(datadir::AbstractString,
              detectors::AbstractString,
             srcHeights::AbstractString,
                srcRhos::AbstractString,
               srcRadii::AbstractString,
             srcLengths::AbstractString)
```

read detectors and sources parameters from the location given in the argument list.

Return a tuple

```
(detectors_array,
     srcHeights_array,
     srcRhos_array,
     srcRadii_array,
     srcLengths_array,
     isPoint)
```

source

GeoEfficiency.getDetectors – Function.

```
getDetectors(detectors_array::Vector{<:Detector} = Detector[])::Vector{Detector}
```

return the detectors_array as Vector{Detector} extended by the entered detectors and sorted according to the detector volume. prompt the user to input detector parameters from the console.

Note

If no array received in the input an empty array will be created to receive the converted detectors.

source

```
getDetectors(detector_info_array::Matrix{<:Real},
                                 detectors_array::Vector{<:Detector} = Detector[];
                                                                                  console_FB
    =true)::Vector{Detector}
```

return detectors_array as Vector{Detector}, after extending it with the successfully converted detectors. while, attempt to convert detectors from the information in detector_info_array.

Note

if console_FB argument is set to true , the function will call getDetectors() to take input from the console if the detector_info_array is empty or contain no numerical element.

source

Chapter 7

Calculations

`GeoEfficiency.geoEff` – Function.

```
geoEff(detector::Detector, aPnt::Point, SrcRadius::Real = 0.0, SrcLength::Real = 0.0)::
    Float64
```

return the `geometrical efficiency` for a source (`point`, `disk` or `cylinder`) with the detector `detector`. `detector` can be any of the leaf detectors types (`CylDetector`, `BoreDetector`, `WellDetector`).

- `aPNT`: a point represent the anchoring point of the source.
- `SrcRadius`: Radius of the source.
- `srcHeight`: the height of an upright cylinder source.

`Throw` an Error if the source location is inappropriate.

Warning

the point height of `aPnt` is measured differently for different detectors types. for the details, please refer to each detector entry.

Note

- if `SrcLength` equal to `zero`; the method return Geometrical Efficiency of a disc source of Radius = `SrcRadius` and center at the point `aPNT`.
- if both `SrcRadius` and `SrcLength` equal to `zero`; the method returns the Geometrical Efficiency of a point source at the anchoring point.

Example

- to obtain the efficiency of a `cylindrical` detector of crystal radius `2.0` cm for axial source cylinder of radius `1.0` cm and height `2.5` cm on the detector surface.

```
julia> using GeoEfficiency

julia> geoEff(CylDetector(2.0), Point(0.0), 1.0, 2.5)
0.2923777934922748
```

- to obtain the efficiency for a `bore-hole` detector of crystal radius of `2.0` and height of `3.0` with hole radius of `1.5` cm for axial source cylinder of radius `1.0` cm and height `2.5` cm starting from detector center.

```
julia> using GeoEfficiency

julia> newDet = BoreDetector(2.0, 3.0, 1.5);

julia> geoEff(newDet, Point(0.0), 1.0, 2.5)
0.5678174038944723
```

- to obtain the efficiency for a `well-type` detector of crystal radius of `2.0` cm and height `3.0` cm with hole radius of `1.5` cm and depth of `1.0` cm for axial source cylinder of radius `1.0` cm and height `2.5` cm at the hole surface.

```
julia> using GeoEfficiency

julia> newDet = WellDetector(2.0, 3.0, 1.5, 1.0);

julia> geoEff(newDet, Point(0.0), 1.0, 2.5)
0.4669614527701105
```

source

`GeoEfficiency.GeoEff_Pnt` – Function.

unexported

```
GeoEff_Pnt(detector::CylDetector, aPnt::Point)::Float64
```

return the `geometrical efficiency` for the point source `aPnt` located on front of the cylindrical detector `detector` face.

`Throw` an Error if the point is out of the cylindrical detector `detector` face.

> **Note**
>
> this is the base function that all other functions call directly or indirectly to calculate `geometrical efficiency` of the cylindrical-ish detector family.

source

`GeoEfficiency.GeoEff_Disk` – Function.

unexported

```
GeoEff_Disk(detector::CylDetector, SurfacePnt::Point, SrcRadius::Real)::Float64
```

return the `geometrical efficiency` for a `disk` source. The `disk` center is the `SurfacePnt` and its radius is `SrcRadius` on front of the cylindrical detector `detector` face.

produce a warning if the disk is out of the cylindrical detector face.

source

Chapter 8

Output Interface

GeoEfficiency.calc – Function.

```
calc(detector::Detector = Detector(), aSource::Tuple{Point, Float64, Float64,} = source())
```

calculate and display on the `console` the `geometrical efficiency` of the detector `detector` for the tuple `aSource` describing the source.

`Throw` an Error if the source location is inappropriate.

see also: `geoEff(::Detector, ::Tuple{Point, Float64, Float64})`

Note

if source description `aSource` alone or even both source description and detector `detect` are missing, the method prompt the user to complete the missing data via the `console`.

source

GeoEfficiency.calcN – Function.

```
calcN()
```

calculate and display the `geometrical efficiency` repeatedly. Prompt the user to input a `detector` and a `source` from the `console`. Prompt the user `repeatedly` until it exit (give a choice to use the same detector or a new detector).

source

GeoEfficiency.batch – Function.

```
batch()
```

provide batch calculation of the `geometrical efficiency` based on the information provided by the **CSV** files by default located in **`/home/travis/GeoEfficiency`**.

results are saved on a **CSV** file(s) named after the detector(s). the **CSV** file(s) by default found in **`/home/travis/GeoEfficiency/resu`** also a log of the results are displayed on the `console`.

for more information on batch refer to `batchInfo`.

source

```
batch(
        detector::Detector,
        srcHeights_array::Vector{S},
        srcRhos_array::Vector{S}=[0.0],
        srcRadii_array::Vector{S}=[0.0],
        srcLengths_array::Vector{S}=[0.0],
        ispoint::Bool=true
        )::String       where S <: Real
```

provide batch calculation of the `geometrical efficiency` for the detector `detector`. results are saved on a **CSV** file named after the detector. the **CSV** file by default found in **`/home/travis/GeoEfficiency/results`**. this method return the actual path to the **CSV** file. also a log of the results are displayed on the `console`.

- `srcHeights_array`: list of source heights to feed to batch.
- `srcRhos_array`: list of source off-axis distances to feed to batch.
- `srcRadii_array`: list of source radii to feed to batch.
- `srcLengths_array`: list of source lengths to feed to batch.

A set of sources is constructed of every valid **combination** of parameter in the `srcRhos_array`, `srcRadii_array` and `srcLengths_array` arrays with conjunction with `ispoint`.

> **Warning**
>
> - If `ispoint` is `true` the source type is a point source and the parameters in `srcRadii_array` and `srcLengths_array` arrays is completely ignored.
> - If `ispoint` is `false` the parameters in srcRhos_array is completely ignored.

source

```
batch(
        detectors_array::Vector{<: Detector},
    srcHeights_array::Vector{S},
    srcRhos_array::Vector{S}=[0.0],
    srcRadii_array::Vector{S}=[0.0],
    srcLengths_array::Vector{S}=[0.0],
        ispoint::Bool=true
        )::Vector{String} where S <: Real
```

same as **`batch(::Detector, ::Vector{Real},::Vector{Real},::Vector{Real},::Vector{Real},::Bool)`** but accept a list of detectors `detectors_array`. return a list of paths to the **CSV** of files (file for each detector) storing the results.

source

```
batch(
        detector_info_array::Matrix{S},
        srcHeights_array::Vector{S},
        srcRhos_array::Vector{S}=[0.0],
        srcRadii_array::Vector{S}=[0.0],
        srcLengths_array::Vector{S}=[0.0],
        ispoint::Bool=true
        )::Vector{String}       where S <: Real
```

same as **`batch(::Vector{Detector}, ::Vector{Real},::Vector{Real},::Vector{Real},::Vector{Real},::Bool)`** but provide batch calculation of the `geometrical efficiency` for the detector in the `detector_info_array` after applying `getDetectors`. return a list of paths to the **CSV** of files (file for each detector) storing the results.

source

GeoEfficiency.batchInfo – Constant.

The function batch() can be called with or without arrangement(s). The without argument version relay on previously prepared Comma Saved Values [CSV] files, that can be easily edit by Microsoft Excel, by default located in the directory **/home/travis/GeoEfficiency** .

results of batch calculation are saved on a **CSV** file(s) named after the detector(s). the **CSV** file by default found in **/home/travis/GeoEfficiency/results**.

CSV input files

- Detectors.csv contains the detectors description; The line format is:

```
    Crystal_Radius | Crystal_Length | Hole_Radius | Hole_Depth |
    ---------------| ---------------|-------------|----------- |
```

- srcHeights.csv contains the source heights;

```
    Source_Heights |
    ---------------|
```

- srcRhos.csv contains the source off-axis distances;

```
    Source_Rhos |
    ------------|
```

- srcRadii.csv contains the source radii for disc and cylindrical sources;

```
    Source_Radii|
    ------------|
```

- srcLengths.csv contains the source length for cylindrical sources;

```
    Source_Lengths|
    --------------|
```

CSV results files

CSV file containing the results has columns of headers AnchorHeight, AnchorRho, srcRadius, srcLength, GeoEfficiency for non-point sources and columns of headers Height, Rho, GeoEfficiency for point sources.

> **Note**
>
> for Comma Saved Values [CSV] files each line represent an entry, the first line is always treated as the header.

> **Warning**
>
> the program expect each line to contain one number for all CSV files except for Detectors.csv each line should contain at least one number or at most four separated numbers.

source

`GeoEfficiency.checkResultsDirs` – Function.

UnExported

```
checkResultsDirs()
```

make sure that directories for saving the results are already exist or create them if necessary.

source

`GeoEfficiency.writecsv_head` – Function.

unexported

```
writecsv_head(filename::AbstractString, content::VecOrMat{<:Union{Int,Float64}}, head=[])
```

Write `content` to the comma delimited values file `filename`. optionally with header `head`.

source

`GeoEfficiency._max_batch` – Constant.

-ve value will display all batch results on

source

`GeoEfficiency.max_batch` – Function.

```
max_batch(n<:Real)
```

set the value of '*max*batch' which default to 20 which control the maxumam number of entries per detector that permit the detector efficiency calculation to be displayed on console. this function `do not` affect the saving of the batch calculation.

-ve value of n result in displaying all batch calculation results on the console.

see also: `max_batch()`

source

```
max_batch()
```

set the value of '*max*batch' to its default value.

see also: `max_batch(::Integer)`

source

`GeoEfficiency._batch` – Function.

UnExported

```
_batch(
        ::Val{true},
        detector::Detector,
        srcHeights_array::Vector{Float64},
        srcRhos_array::Vector{Float64},
        srcRadii_array::Vector{Float64},
        srcLengths_array::Vector{Float64}
        )
```

batch calculation for specialized for **`point`** sources. return a tuple of three arrays the `detector`, the `results`and the path of the **CSV** file containing results.

The `results` has columns of headers `Height`, `Rho`, `GeoEfficiency`.

> **Note**
>
> for all arrays `srcHeights_array`, `srcRhos_array`, `srcRadii_array` and `srcLengths_array` element type should be `Float64`. if any of them have other numerical element type it should converted to `Float64` using `float` before passing it to this method.

> **Warning**
>
> both `srcRadii_array`, `srcLengths_array` are completely ignored as this method is for point sources.

source

UnExported

```
_batch(
        ::Val{false},
        detector::Detector,
        srcHeights_array::Vector{Float64},
        srcRhos_array::Vector{Float64},
        srcRadii_array::Vector{Float64},
        srcLengths_array::Vector{Float64},
        )
```

batch calculation for specialized for **`non-point`** sources. return a tuple of three arrays the `detector`, the `results`and the path of the **CSV** file containing results.

The `results` has columns of headers `AnchorHeight`, `AnchorRho`, `srcRadius`, `srcLength`, `GeoEfficiency`.

> **Note**
>
> for all arrays `srcHeights_array`, `srcRhos_array`, `srcRadii_array` and `srcLengths_array` element type should be `Float64`. if any of them have other numerical element type it should converted to `Float64` using `float` before passing it to this method.

source

Part III

Index

Chapter 9

Index