GeoEfficiency.jl

Mohamed E. Krar

April 13, 2019

Contents

Part I

Home

Chapter 1

GeoEfficiency: Accurate Geometrical Efficiency Calculator

An officially registered Julia program that 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 E. Krar (DrKrar@gmail.com)
Repository	GitHub.com
Documentation	GitHub.io
Current version	v"0.9.3"
First Created	Fri Aug 14 20:12:01 2015

This documentation is also available in pfd format.

1.1 Current and Planed Features

The following list show the state of current feature and planed feature. the checked items represent already 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 Installation

the package is registered officially and so it can be installed through the Julia package management system by typing the following into the REPL prompt.

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

1.4 Quick Usage

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

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

1.5 Unit Test

For scientific calculation accuracy in calculation and being error free is a highly demanded objective. Thus, the package is extensively tested method-wise in each supported operating system. Operating system fully supported include Windows, Linus, Apple OSx.

After installing the package can be tested in your own system by typing the following into the REPL prompt.

```
julia> using Test, Pkg
julia> Pkg.test("GeoEfficiency")
```

1.6 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 source anchoring point the source position 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` ; `π/2` ; `exp(2)` ; `1E-2` ; `5.2/3` ; `sin(1)` ; `sin(1)^2` are all valid expressions.

1.7 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 directory.

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 (`a detector per line`); 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

Summery

`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" - (0 days old master)
Documentation: https://GeoEfficiency.GitHub.io/index.html
PDF_Manual:    https://GeoEfficiency.GitHub.io/pdf/GeoEfficiency.pdf



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

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

source

`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/GeoEfficiency`** folder in batch mode.

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

`source`

Chapter 3

Physics Model

Geometrical efficiency of radioactive source measurement is a type of detection efficiency. A fully describe a radioactive source measurement at the most basic level three component should be provided.

- radioactive detector description
- radiation source description
- relative position of the source to detector.

this section will discus how to instruct the program to construct each of the aforementioned component.

Chapter 4

Detector

Currently, only cylindrical-like types of detectors are supported.

4.1 Cylindrical Detector

`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.CylDetector` – Method.

```
CylDetector(CryRadius::Real)
```

construct and return a `cylindrical` (really `disk`) detector with crystal length equal to **`zero`**.

see also: `CylDetector(CryRadius::Real, CryLength::Real)`.

source

`GeoEfficiency.CylDetector` – Method.

```
CylDetector()
```

construct and return a `cylindrical` detector according to the input from the `console`.

see also: `CylDetector(CryRadius::Real, CryLength::Real)`.

source

Note
the positon of the source is reported relative to the detector anchoring point, for a cylinder detector it is taking as a point in the plain surface nearest to the source which lies on the detector axis of symmetry.

4.2 Bore-hole Detector

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.BoreDetector – Method.

```
BoreDetector()
```

construct and return a bore-hole detector according to the input from the console.

see also: BoreDetector(CryRadius::Real, CryLength::Real, HoleRadius::Real).

source

Note

the positon of the source is reported relative to the detector anchoring point, for a bore-hole detector it is taking as the middle point of its axis of symmetry.

4.3 Well-type Detector

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.WellDetector – Method.

```
WellDetector()
```

construct and return a Well-Type detector according to the input from the `console`.

see also: `WellDetector(CryRadius::Real, CryLength::Real, HoleRadius::Real, HoleDepth::Real)`.

`source`

Note

the positon of the source is reported relative to the detector anchoring point, for well-type detector it is taking as the point detector hole surface that lies on the detector axis of symmetry.

Chapter 5

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 source type set to point source, both `SrcRadius` and `SrcLength` are set to zero.
for more information **see also:** [`typeofSrc()`](@ref) and [`typeofSrc(x::Int)`](@ref).
```

`source`

Chapter 6

Source Anchoring Point

GeoEfficiency.Point – Type.

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

construct and return a `Point` source. The `Point` 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.Point – Method.

```
Point(Height::Real)
```

construct and return an `axial point`.

see also: `Point(Height::Real, Rho::Real)`.

source

GeoEfficiency.Point – Method.

```
Point()
```

construct and return a `point`. prompt to input information via the `console`.

see also: `Point(Height::Real, Rho::Real)`.

source

GeoEfficiency.Point – Method.

```
Point(xHeight::Real, aPnt::Point)
```

construct and return a point that has the same off-axis distance as aPnt but of new height xHeight.

see also: Point(Height::Real, Rho::Real)

source

GeoEfficiency.Point – Method.

```
Point(aPnt::Point, xRho::Real)
```

construct and return a point that has the same height as aPnt but of new off-axis distance Rho.

see also: Point(Height::Real, Rho::Real).

source

Chapter 7

Calculations

calculation of the geometrical efficiency can be done via a call to the function `geoEff`.

`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`.

Arguments

- `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 `InValidGeometry` if the point location is invalide.
- an `NotImplementedError` if source-to-detector geometry not supported yet.

  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

This function has another method `geoEff()` that prompt the user to input a source and a detector via the `console`.

Chapter 8

Output Interface

Calculation of the geometrical efficiency can be run in one of two modes aside from using `geoEff`, the interactive mode and the batch mode.

8.1 Interactive Mode

`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 `inValidGeometry` 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`

for repeated calculations.

`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`

8.2 Batch Mode

`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/results`**, 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

The batch calculation controlled by CSV files. the following refer to information on the CSV files structure and location.

`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

The result of the batch calculation is also displayed in the `console`. the function `max_batch(n::Real)` can be used to give a hint (thus it may or may not apply) to the program to limit displayed results.

`GeoEfficiency.max_batch` – Method.

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

set the value of `_max_batch` which give a hint to the program on maxumam number of entries per detector displayed on the `console` in btach mode. This function `do not` affect the saving of the batch calculation.

> **Note**

```
Negative value will display prevent batch results from printed to the `console`.
while `Inf` will print all  batch results to the `console`.
```

see also: **`max_batch()`**

source

also the without arguments `max_batch()` restore back the default vaule.

`GeoEfficiency.max_batch` – Method.

```
max_batch()
```

set the value of `_max_batch` which give a hint to the program on maxumam number of entries per detector displayed on the `console` in btach mode. to its default value set by the contant `max_display`.

see also: **`max_batch(n::Real)`**

source

Before the batch mode start the user is asked to decide the source type. once the calculation is done the user can check the current seting for the source or modifiy it. for details see the next section.

Chapter 9

Batch Mode 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`

Warnning

Currently, the source type has no effect but to decide if the source is a point source or a higher dimention source.

Part III

Development

Chapter 10

Introduction

This section is provided for developer who are interested in extending the functionality of the `GeoEfficiency` package or just make use of some of its functionality. this software is licensed under the MIT license.

```
MIT "Expat" License

 Copyright (c) 2019: Mohamed Krar.

 Permission is hereby granted, free of charge, to any person obtaining a copy
 of this software and associated documentation files (the "Software"), to deal
 in the Software without restriction, including without limitation the rights
 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 copies of the Software, and to permit persons to whom the Software is
 furnished to do so, subject to the following conditions:

 The above copyright notice and this permission notice shall be included in all
 copies or substantial portions of the Software with an appropriate reference to
 the original work.

 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 SOFTWARE.
```

Chapter 11

Configuration

The package contain many parameters that can be set within the program sourcecode. they can be found in the source file `Config.jl`

parameter	description	default value
dataFolder	name of the root directory	"GeoEfficiency"
dataDir	root directory	joinpath(homedir(), dataFolder)
integrate	use the package `QuadGK` to perform integration	begin using QuadGK; QuadGK.quadgk; end
relativeError		1.0E-4
absoluteError		eps(1.0)
resultsFolder	name of the result directory inside the root directory	"results"
max_display	define the default for maximum number of entries shown in the `console` in batch mode	20 **see** `max_batch`

Chapter 12

Error System

`GeoEfficiency.GeoException` – Type.

custom abstract `exception` that is the parent of all exception in the `GeoEfficiency` package

source

`GeoEfficiency.InValidDetectorDim` – Type.

custom `exception` indicating invalid radiation detector dimensions

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.InValidGeometry` – Type.

custom `exception` indicating a not valid source to detector geometry

source

`GeoEfficiency.@inValidGeometry` – Macro.

custom macro to throw `NotImplementedError` `exception`

source

`GeoEfficiency.NotImplementedError` – Type.

custom `exception` indicating a source to detector geometry which may be valid but not implemented yet

source

`GeoEfficiency.@notImplementedError` – Macro.

custom macro to throw `NotImplementedError` exception

source

Chapter 13

Console Input

Julia language is quite reach language but it seems a good idea thought to collect repeated tasks involving input from console in compact and customized to the need function. this section provide two essential functions to deal with inputs from the console. the first:

`GeoEfficiency.input` – Function.

UnExported

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

return a string represent the user respond delimited by new line excluding the new line. prompt the user with the massage `prompt` defaults to ?. wait until the user type its respond and press return. `incolor` specify the prompt text color, default to : $green$ may take any of the values `:black`, `:blue`, `:cyan`, `:green`, `:light_black`, `:light_blue`, `:light_cyan`, `:light_green`, `:light_magenta`, `:light_red`, `:light_yellow`, `:magenta`, `:red`, `:white`, or `:yellow`.

Color

```
    The effect of color is not allways respected in all teriminals as some color may be
simplly
    ignored by some teriminals.
```

source

while the second is a more complex function:

`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. check that the numerical value is in interval [`from`, `to`[by default [-∞, ∞[before returning it as a `Float64`. throws `ArgumentError` when the given interval is not valid. if the numerical expression fail to evaluated to numerical value or the numerical value is not in the valid interval the function will warn the user and **reprompt** him to give a valid expresion.

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.
    expression like ``5/2`` ; ``5//2`` ; ``pi`` ; π``/2`` ; ``exp(2)`` ; ``1E-2 `` ;
``5.2/3`` ;
    ``sin(1)`` ;  ``sin(1)^2`` are all valid 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:", value="sin(1)^2")
0.7080734182735712

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

source

Those function are not exported that is normally the user will not need to use them but they are documented here to allow a developer ranked user to make use of them.

Chapter 14

Physics Model

Two abstract detector types defined in the package to classify the detectors, the top most super type,

`GeoEfficiency.RadiationDetector` – Type.

abstract super-supertype of all detectors types

source

any future detector definition should inherit from `RadiationDetector`. The second abstract detector `Detector` is also a sub-type of `RadiationDetector` but it only accommodates cylindrical type only.

`GeoEfficiency.Detector` – Type.

```
Detector
```

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

source

can be used to construct leaf detector.

`GeoEfficiency.Detector` – Method.

```
Detector()
```

construct and return an object of the `Detector` leaf types (`CylDetector`, `BoreDetector` or `WellDetector`) according to the input from the console.

Note

all required information is acquired from the `console` and would warn user on invalid data.

source

also it can be used to construct a concrete detector depend on the provided arguments.

`GeoEfficiency.Detector` – Method.

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

construct and return `well-type`, `bore-hole` or `cylindrical` detector according to the arguments. it inspect the arguments and call the appropriate leaf type constructor.

Note

if the value(s) of the last argument(s) is\are `zero`, it acts as a missing argument(s).

see also: `CylDetector`, `BoreDetector`, `WellDetector`.

source

`GeoEfficiency.Detector` – Method.

```
Detector(CryRadius::Real)
```

same as `CylDetector(CryRadius::Real)`.

source

`GeoEfficiency.Detector` – Method.

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

same as `CylDetector(CryRadius::Real, CryLength::Real)`.

source

`GeoEfficiency.Detector` – Method.

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

same as `BoreDetector(CryRadius::Real, CryLength::Real, HoleRadius::Real)` except when `HoleRadius = 0.0` it acts as `CylDetector(CryRadius::Real, CryLength::Real)`.

source

Chapter 15

Batch Mode Input

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 16

Output Interface

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.

Global variable that give a hint to the program on maxumam number of entries per detector displayed on the `console` in btach mode.

Note

```
Negative value will display prevent batch results from printed to the `console`.
while `Inf` will print all  batch results to the `console`.
```

source

GeoEfficiency.max_display – Constant.

set the default value for the global variable `_max_batch`

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 IV

Index

Chapter 17

Index