

DOS extraction

Generated by Doxygen 1.8.6

Thu Sep 11 2014 17:42:35



# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Index</b>                            | <b>1</b>  |
| 1.1      | Introduction . . . . .                  | 1         |
| 1.2      | Dependancies: . . . . .                 | 1         |
| 1.3      | Compile . . . . .                       | 1         |
| 1.4      | Set up the configuration file . . . . . | 2         |
| 1.5      | Run! . . . . .                          | 2         |
| <b>2</b> | <b>Namespace Index</b>                  | <b>3</b>  |
| 2.1      | Namespace List . . . . .                | 3         |
| <b>3</b> | <b>Hierarchical Index</b>               | <b>5</b>  |
| 3.1      | Class Hierarchy . . . . .               | 5         |
| <b>4</b> | <b>Class Index</b>                      | <b>7</b>  |
| 4.1      | Class List . . . . .                    | 7         |
| <b>5</b> | <b>File Index</b>                       | <b>9</b>  |
| 5.1      | File List . . . . .                     | 9         |
| <b>6</b> | <b>Namespace Documentation</b>          | <b>11</b> |
| 6.1      | constants Namespace Reference . . . . . | 11        |
| 6.1.1    | Detailed Description . . . . .          | 11        |
| 6.2      | numerics Namespace Reference . . . . .  | 11        |
| 6.2.1    | Detailed Description . . . . .          | 12        |
| 6.2.2    | Function Documentation . . . . .        | 12        |
| 6.2.2.1  | sort . . . . .                          | 12        |
| 6.2.2.2  | sort_pair . . . . .                     | 12        |
| 6.2.2.3  | trapz . . . . .                         | 13        |
| 6.2.2.4  | trapz . . . . .                         | 13        |
| 6.2.2.5  | deriv . . . . .                         | 13        |
| 6.2.2.6  | interp1 . . . . .                       | 13        |
| 6.2.2.7  | interp1 . . . . .                       | 14        |
| 6.2.2.8  | error_L2 . . . . .                      | 14        |

|          |  |           |
|----------|--|-----------|
| <b>7</b> | <b>Class Documentation</b>             | <b>15</b> |
| 7.1      | Bim1D Class Reference                  | 15        |
| 7.1.1    | Detailed Description                   | 16        |
| 7.1.2    | Constructor & Destructor Documentation | 16        |
| 7.1.2.1  | Bim1D                                  | 16        |
| 7.1.3    | Member Function Documentation          | 16        |
| 7.1.3.1  | log_mean                               | 16        |
| 7.1.3.2  | bernoulli                              | 16        |
| 7.1.3.3  | assembleAdvDiff                        | 16        |
| 7.1.3.4  | assembleStiff                          | 17        |
| 7.1.3.5  | assembleMass                           | 17        |
| 7.2      | Charge Class Reference                 | 17        |
| 7.2.1    | Detailed Description                   | 18        |
| 7.2.2    | Constructor & Destructor Documentation | 18        |
| 7.2.2.1  | Charge                                 | 18        |
| 7.2.3    | Member Function Documentation          | 18        |
| 7.2.3.1  | charge                                 | 18        |
| 7.2.3.2  | dcharge                                | 18        |
| 7.3      | CsvParser Class Reference              | 19        |
| 7.3.1    | Detailed Description                   | 20        |
| 7.3.2    | Constructor & Destructor Documentation | 20        |
| 7.3.2.1  | CsvParser                              | 20        |
| 7.3.3    | Member Function Documentation          | 20        |
| 7.3.3.1  | importRow                              | 20        |
| 7.3.3.2  | importRows                             | 20        |
| 7.3.3.3  | importFirstRows                        | 21        |
| 7.3.3.4  | importCol                              | 21        |
| 7.3.3.5  | importCols                             | 21        |
| 7.3.3.6  | importFirstCols                        | 21        |
| 7.3.3.7  | importCell                             | 22        |
| 7.3.3.8  | importAll                              | 23        |
| 7.4      | DosModel Class Reference               | 23        |
| 7.4.1    | Detailed Description                   | 24        |
| 7.4.2    | Constructor & Destructor Documentation | 24        |
| 7.4.2.1  | DosModel                               | 24        |
| 7.4.3    | Member Function Documentation          | 24        |
| 7.4.3.1  | simulate                               | 24        |
| 7.4.3.2  | post_process                           | 24        |
| 7.4.3.3  | gnuplot_commands                       | 25        |
| 7.4.3.4  | save_plot                              | 25        |

|          |  |    |
|----------|--|----|
| 7.5      | GaussHermiteRule Class Reference       | 25 |
| 7.5.1    | Detailed Description                   | 26 |
| 7.5.2    | Constructor & Destructor Documentation | 26 |
| 7.5.2.1  | GaussHermiteRule                       | 26 |
| 7.5.3    | Member Function Documentation          | 26 |
| 7.5.3.1  | apply                                  | 26 |
| 7.6      | GaussianCharge Class Reference         | 26 |
| 7.6.1    | Detailed Description                   | 27 |
| 7.6.2    | Constructor & Destructor Documentation | 27 |
| 7.6.2.1  | GaussianCharge                         | 27 |
| 7.6.3    | Member Function Documentation          | 27 |
| 7.6.3.1  | charge                                 | 27 |
| 7.6.3.2  | dcharge                                | 28 |
| 7.6.3.3  | n_approx                               | 28 |
| 7.6.3.4  | dn_approx                              | 28 |
| 7.7      | NonLinearPoisson1D Class Reference     | 29 |
| 7.7.1    | Detailed Description                   | 29 |
| 7.7.2    | Constructor & Destructor Documentation | 30 |
| 7.7.2.1  | NonLinearPoisson1D                     | 30 |
| 7.7.3    | Member Function Documentation          | 30 |
| 7.7.3.1  | apply                                  | 30 |
| 7.7.3.2  | computeJac                             | 30 |
| 7.8      | ParamList Class Reference              | 30 |
| 7.8.1    | Detailed Description                   | 32 |
| 7.8.2    | Constructor & Destructor Documentation | 32 |
| 7.8.2.1  | ParamList                              | 32 |
| 7.9      | PdeSolver1D Class Reference            | 32 |
| 7.9.1    | Detailed Description                   | 33 |
| 7.9.2    | Constructor & Destructor Documentation | 34 |
| 7.9.2.1  | PdeSolver1D                            | 34 |
| 7.9.3    | Member Function Documentation          | 35 |
| 7.9.3.1  | assembleAdvDiff                        | 35 |
| 7.9.3.2  | assembleStiff                          | 35 |
| 7.9.3.3  | assembleMass                           | 35 |
| 7.10     | QuadratureRule Class Reference         | 35 |
| 7.10.1   | Detailed Description                   | 36 |
| 7.10.2   | Constructor & Destructor Documentation | 36 |
| 7.10.2.1 | QuadratureRule                         | 36 |

|              |  |           |
|--------------|--|-----------|
| 8.1          | <a href="#">/home/Data/Dropbox/Progetto-PACS/C++/Source/src/charge.h File Reference</a>            | 39        |
| 8.1.1        | Detailed Description   | 39        |
| 8.2          | <a href="#">/home/Data/Dropbox/Progetto-PACS/C++/Source/src/csvParser.h File Reference</a>         | 39        |
| 8.2.1        | Detailed Description   | 40        |
| 8.3          | <a href="#">/home/Data/Dropbox/Progetto-PACS/C++/Source/src/dosModel.h File Reference</a>          | 40        |
| 8.3.1        | Detailed Description   | 40        |
| 8.4          | <a href="#">/home/Data/Dropbox/Progetto-PACS/C++/Source/src/numerics.h File Reference</a>          | 41        |
| 8.4.1        | Detailed Description   | 41        |
| 8.5          | <a href="#">/home/Data/Dropbox/Progetto-PACS/C++/Source/src/paramList.h File Reference</a>         | 41        |
| 8.5.1        | Detailed Description   | 42        |
| 8.6          | <a href="#">/home/Data/Dropbox/Progetto-PACS/C++/Source/src/physicalConstants.h File Reference</a> | 42        |
| 8.6.1        | Detailed Description   | 42        |
| 8.7          | <a href="#">/home/Data/Dropbox/Progetto-PACS/C++/Source/src/quadratureRule.h File Reference</a>    | 43        |
| 8.7.1        | Detailed Description   | 43        |
| 8.8          | <a href="#">/home/Data/Dropbox/Progetto-PACS/C++/Source/src/solvers.h File Reference</a>           | 43        |
| 8.8.1        | Detailed Description   | 44        |
| 8.9          | <a href="#">/home/Data/Dropbox/Progetto-PACS/C++/Source/src/typedefs.h File Reference</a>          | 44        |
| 8.9.1        | Detailed Description   | 45        |
| 8.9.2        | Typedef Documentation  | 45        |
| 8.9.2.1      | VectorX  | 45        |
| 8.9.2.2      | VectorXpair  | 45        |
| <b>Index</b> |  | <b>46</b> |

# Chapter 1

## Index

### 1.1 Introduction

This program allows to extract the Density of States, assessed by mean capacitance-voltage measurements, in an organic semiconductor device. Simulated values are fitted to experimental data. Source files and headers are written in C++11 language. The software is intended to be used on a UNIX operating system.

### 1.2 Dependancies:

The program requires the following libraries to be installed on your system:

- **Eigen**, to handle with matrices, vectors and linear algebra;
- **GetPot**, to parse command-line and configuration files;
- **Gnuplot**, a graphical utility to generate plots; its interface to C++ also requires:
- **Boost**, a C++ library;
- **OpenMP**, for parallel computing (recommended but not compulsory).

### 1.3 Compile

In order to compile a test executable, simply execute one of these commands in a terminal pointing the root directory:

```
$ make
```

or, if you want the compiler to produce debugging informations:

```
$ make debug
```

You can specify the name of the test to be compiled (without extension, for example: *simulate\_dos*) by passing the variable **NAME** through command-line:

```
$ make NAME=test_filename
```

The compiler will generate the *test\_filename* executable under the *bin/* directory.

Repeat these instructions for each test you want to compile.

## 1.4 Set up the configuration file

### Warning

each file described in this section must be placed under the *config/* directory.

Before you can an executable, you have to set up the configuration file, called *config.pot*. Within it, you can find a list of parameters, each of which is commented out to explain what modifying it will entail.

Particularly, you can set the variables *input\_params* and *input\_experim*, i.e. the filenames where to find input fitting parameters and experimental data respectively. It's recommended (but not compulsory) to put these files in the same directory as the configuration file (otherwise you can specify a relative or absolute path to them).

You can create multiple configuration files, each with different parameter values: the one you aim to use can be specified in the command-line before running.

## 1.5 Run!

Executables are placed under the *bin/* directory.

To run by using the default configuration filename (*config.pot*) simply execute in the root:

```
$ bin/test_filename
```

or, if your terminal is already pointing the *bin/* directory:

```
$ ./test_filename
```

To specify a different configuration file previously saved in the *config/* directory:

```
$ bin/test_filename -f configuration_filename
```

or:

```
$ bin/test_filename --file configuration_filename
```

Once complete, you can find the results of the simulation(s) in the output directory specified in the configuration file (default: *output*) under *bin/*.



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

|  |    |
|--|----|
| <a href="#">constants</a>                          |    |
| Numerical constants . . . . .                      | 11 |
| <a href="#">numerics</a>                           |    |
| Namespace for generic numeric algorithms . . . . . | 11 |



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

|                              |    |
|------------------------------|----|
| Charge . . . . .             | 17 |
| GaussianCharge . . . . .     | 26 |
| CsvParser . . . . .          | 19 |
| DosModel . . . . .           | 23 |
| NonLinearPoisson1D . . . . . | 29 |
| ParamList . . . . .          | 30 |
| PdeSolver1D . . . . .        | 32 |
| Bim1D . . . . .              | 15 |
| QuadratureRule . . . . .     | 35 |
| GaussHermiteRule . . . . .   | 25 |



## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

|                                    |   |    |
|------------------------------------|---|----|
| <a href="#">Bim1D</a>              | Class derived from <a href="#">PdeSolver1D</a> , providing a finite volume Box Integration Method (BIM) solver                  | 15 |
| <a href="#">Charge</a>             | Abstract class providing methods to calculate total electric charge (the rhs in the Poisson equation) . . . . .                 | 17 |
| <a href="#">CsvParser</a>          | Class providing methods to read content from a .csv file and store it in matrices or vectors . . .                              | 19 |
| <a href="#">DosModel</a>           | Class providing methods to process a simulation to extract the Density of States starting from a parameter list . . . . .       | 23 |
| <a href="#">GaussHermiteRule</a>   | Class derived from <a href="#">QuadratureRule</a> providing the Gauss-Hermite quadrature rule . . . . .                         | 25 |
| <a href="#">GaussianCharge</a>     | Class derived from <a href="#">Charge</a> , under the hypothesis that Density of States is a combination of gaussians . . . . . | 26 |
| <a href="#">NonLinearPoisson1D</a> | Provide a solver for a non-linear Poisson equation . . . . .  | 29 |
| <a href="#">ParamList</a>          | Class providing methods to handle a list of parameters . . . . .  | 30 |
| <a href="#">PdeSolver1D</a>        | Abstract class providing methods to assemble matrices to solve one-dimensional PDEs . . . .                                     | 32 |
| <a href="#">QuadratureRule</a>     | Abstract class providing a quadrature rule . . . . .  | 35 |



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all documented files with brief descriptions:

|  |    |
|--|----|
| /home/Data/Dropbox/Progetto-PACS/C++/Source/src/ <a href="#">charge.h</a>            |    |
| Classes for computing total electric charge . . . . .                                | 39 |
| /home/Data/Dropbox/Progetto-PACS/C++/Source/src/ <a href="#">csvParser.h</a>         |    |
| Tools to store content from a .csv file in matrices or vectors . . . . .             | 39 |
| /home/Data/Dropbox/Progetto-PACS/C++/Source/src/ <a href="#">dosModel.h</a>          |    |
| Mathematical model for Density of States extraction . . . . .                        | 40 |
| /home/Data/Dropbox/Progetto-PACS/C++/Source/src/ <a href="#">numerics.h</a>          |    |
| Generic numeric algorithms . . . . .   | 41 |
| /home/Data/Dropbox/Progetto-PACS/C++/Source/src/ <a href="#">paramList.h</a>         |    |
| List of simulation parameters . . . . .  | 41 |
| /home/Data/Dropbox/Progetto-PACS/C++/Source/src/ <a href="#">physicalConstants.h</a> |    |
| Physical constants . . . . .   | 42 |
| /home/Data/Dropbox/Progetto-PACS/C++/Source/src/ <a href="#">quadratureRule.h</a>    |    |
| Quadrature rules . . . . .   | 43 |
| /home/Data/Dropbox/Progetto-PACS/C++/Source/src/ <a href="#">solvers.h</a>           |    |
| Generic solvers for PDEs . . . . .   | 43 |
| /home/Data/Dropbox/Progetto-PACS/C++/Source/src/ <a href="#">typedefs.h</a>          |    |
| Typedefs and utility functions . . . . .   | 44 |





## Chapter 6

# Namespace Documentation

### 6.1 constants Namespace Reference

Numerical constants.

#### Variables

- const double **Q** = 1.602176530000000e-19  
*Electron charge [C].*
- const double **Q2** = **Q** \* **Q**  
*Electron charge squared [C<sup>2</sup>].*
- const double **K\_B** = 1.380650500000000e-23  
*Boltzmann's constant [J · K<sup>-1</sup>].*
- const double **EPS0** = 8.854187817e-12  
*Vacuum electrical permittivity [C · V<sup>-1</sup> · m<sup>-1</sup>].*
- const double **T** = 300  
*Reference temperature [K].*
- const double **V\_TH** = **K\_B** \* **T** / **Q**  
*Threshold voltage [V].*
- const unsigned **PARAMS\_NO** = 22  
*Number of parameters required in input file.*
- const double **PI** = M\_PI  
 *$\pi$ .*
- const double **SQRT\_PI** = std::sqrt(**PI**)  
 *$\sqrt{\pi}$ .*
- const double **PI\_M4** = 0.7511255444649425  
 *$\pi^{-\frac{1}{4}}$ .*
- const double **SQRT\_2** = std::sqrt(2)  
 *$\sqrt{2}$ .*

#### 6.1.1 Detailed Description

Numerical constants.

### 6.2 numerics Namespace Reference

Namespace for generic numeric algorithms.

## Functions

- `template<typename ScalarType >  
VectorX< ScalarType > sort (const VectorX< ScalarType > &vector)`  
*Function to sort Eigen vectors.*
- `template<typename ScalarType >  
VectorXpair< ScalarType > sort_pair (const VectorX< ScalarType > &vector)`  
*Function to sort Eigen vectors, keeping track of indexes.*
- `double trapz (const VectorXd &x, const VectorXd &y)`  
*Function to compute approximate integral of y with spacing increment specified by x, using trapezoidal rule.*
- `double trapz (const VectorXd &y)`  
*Compute the approximate integral of y with unit spacing, using trapezoidal rule.*
- `VectorXd deriv (const VectorXd &, const VectorXd &)`  
*Compute the numeric derivative:  $\frac{dy}{dx}$ .*
- `double interp1 (const VectorXd &, const VectorXd &, const double &)`  
*Linear 1D interpolation. Interpolate y, defined at points x, at the point xNew.*
- `VectorXd interp1 (const VectorXd &, const VectorXd &, const VectorXd &)`  
*Linear 1D interpolation. Interpolate y, defined at points x, at the points xNew.*
- `double error_L2 (const VectorXd &, const VectorXd &, const VectorXd &, const double &)`  
*Compute the  $L^2$ -norm error between simulated and interpolated values, using trapz.*

### 6.2.1 Detailed Description

Namespace for generic numeric algorithms.

### 6.2.2 Function Documentation

#### 6.2.2.1 `VectorX< ScalarType > sort ( const VectorX< ScalarType > & vector )`

Function to sort Eigen vectors.

##### Template Parameters

|                   |                    |
|-------------------|--------------------|
| <i>ScalarType</i> | : the scalar type. |
|-------------------|--------------------|

##### Parameters

|           |               |                            |
|-----------|---------------|----------------------------|
| <i>in</i> | <i>vector</i> | : the vector to be sorted. |
|-----------|---------------|----------------------------|

##### Returns

the sorted vector.

#### 6.2.2.2 `VectorXpair< ScalarType > sort_pair ( const VectorX< ScalarType > & vector )`

Function to sort Eigen vectors, keeping track of indexes.

##### Template Parameters

|                   |                    |
|-------------------|--------------------|
| <i>ScalarType</i> | : the scalar type. |
|-------------------|--------------------|

## Parameters

|           |               |                            |
|-----------|---------------|----------------------------|
| <i>in</i> | <i>vector</i> | : the vector to be sorted. |
|-----------|---------------|----------------------------|

## Returns

an Eigen vector of pairs: (sorted value, corresponding index in the unsorted vector).

## 6.2.2.3 double trapz ( const VectorXd &amp; x, const VectorXd &amp; y )

Function to compute approximate integral of *y* with spacing increment specified by *x*, using trapezoidal rule.

## Parameters

|           |          |                                      |
|-----------|----------|--------------------------------------|
| <i>in</i> | <i>x</i> | : the vector of the discrete domain; |
| <i>in</i> | <i>y</i> | : the vector of values to integrate. |

## Returns

the approximate integral value.

## 6.2.2.4 double trapz ( const VectorXd &amp; y )

Compute the approximate integral of *y* with unit spacing, using trapezoidal rule.

## Parameters

|           |          |                                      |
|-----------|----------|--------------------------------------|
| <i>in</i> | <i>y</i> | : the vector of values to integrate. |
|-----------|----------|--------------------------------------|

## Returns

the approximate integral value.

## 6.2.2.5 VectorXd deriv ( const VectorXd &amp; y, const VectorXd &amp; x )

Compute the numeric derivative:  $\frac{dy}{dx}$ .

## Parameters

|           |          |  |
|-----------|----------|--|
| <i>in</i> | <i>y</i> | : the vector of values to differentiate; |
| <i>in</i> | <i>x</i> | : the vector of the discrete domain.     |

## Returns

a vector of the same length as *y* containing the approximate derivative.

## 6.2.2.6 double interp1 ( const VectorXd &amp; x, const VectorXd &amp; y, const double &amp; xNew )

Linear 1D interpolation. Interpolate *y*, defined at points *x*, at the point *xNew*.

## Parameters

|    |             |  |
|----|-------------|--|
| in | <i>y</i>    | : the vector of values to interpolate; |
| in | <i>x</i>    | : the vector of the discrete domain;   |
| in | <i>xNew</i> | : the point to interpolate at.         |

**Returns**

a scalar containing the interpolated value.

**6.2.2.7 VectorXd interp1 ( const VectorXd & x, const VectorXd & y, const VectorXd & xNew )**

Linear 1D interpolation. Interpolate *y*, defined at points *x*, at the points *xNew*.

**Parameters**

|    |             |   |
|----|-------------|---|
| in | <i>y</i>    | : the vector of values to interpolate;    |
| in | <i>x</i>    | : the vector of the discrete domain;      |
| in | <i>xNew</i> | : the vector of points to interpolate at. |

**Returns**

a vector of the same length as *xNew* containing the interpolated values.

**6.2.2.8 double error\_L2 ( const VectorXd & interp, const VectorXd & simulated, const VectorXd & V, const double & V\_shift )**

Compute the  $L^2$ -norm error between simulated and interpolated values, using *trapz*.

**Parameters**

|    |                  |   |
|----|------------------|---|
| in | <i>interp</i>    | : the interpolated values;              |
| in | <i>simulated</i> | : the simulated values;                 |
| in | <i>V</i>         | : the vector of the electric potential; |
| in | <i>V_shift</i>   | : shift to the electric potential.      |

**Returns**

the value of the  $L^2$ -norm error.

## Chapter 7

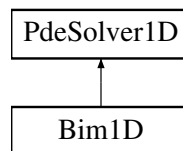
# Class Documentation

### 7.1 Bim1D Class Reference

Class derived from [PdeSolver1D](#), providing a finite volume Box Integration Method (BIM) solver.

```
#include <solvers.h>
```

Inheritance diagram for Bim1D:



#### Public Member Functions

- [Bim1D](#) ()=delete  
*Default constructor (deleted since it is required to specify the mesh).*
- [Bim1D](#) (VectorXd &)  
*Constructor.*
- virtual [~Bim1D](#) ()=default  
*Destructor (defaulted).*
- virtual void [assembleAdvDiff](#) (const VectorXd &, const VectorXd &, const VectorXd &, const VectorXd &) override  
*Assemble the matrix for an advection-diffusion term.*
- virtual void [assembleStiff](#) (const VectorXd &, const VectorXd &) override  
*Assemble the matrix for a diffusion term.*
- virtual void [assembleMass](#) (const VectorXd &, const VectorXd &) override  
*Assemble the matrix for a reaction term.*

#### Static Public Member Functions

- static VectorXd [log\\_mean](#) (const VectorXd &, const VectorXd &)  
*Compute the element-wise logarithmic mean of two vectors.*
- static std::pair< VectorXd, VectorXd > [bernoulli](#) (const VectorXd &)  
*Compute the values of the Bernoulli function.*

## Additional Inherited Members

### 7.1.1 Detailed Description

Class derived from [PdeSolver1D](#), providing a finite volume Box Integration Method (BIM) solver.

Matrices are held in a sparse format.

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 Bim1D ( VectorXd & mesh )

Constructor.

Parameters

|    |             |                         |
|----|-------------|-------------------------|
| in | <i>mesh</i> | : the mesh coordinates. |
|----|-------------|-------------------------|

### 7.1.3 Member Function Documentation

#### 7.1.3.1 VectorXd log\_mean ( const VectorXd & x1, const VectorXd & x2 ) [static]

Compute the element-wise logarithmic mean of two vectors.

$$M_{\log}(x_1, x_2) = \frac{x_2 - x_1}{\log x_2 - \log x_1} = \frac{x_2 - x_1}{\log \left( \frac{x_2}{x_1} \right)}.$$

Parameters

|    |           |                      |
|----|-----------|----------------------|
| in | <i>x1</i> | : the first vector;  |
| in | <i>x2</i> | : the second vector. |

Returns

the vector of the logarithmic means.

#### 7.1.3.2 std::pair< VectorXd, VectorXd > bernoulli ( const VectorXd & x ) [static]

Compute the values of the Bernoulli function.

$$\mathfrak{B}(x) = \frac{x}{e^x - 1}.$$

Parameters

|    |          |  |
|----|----------|--|
| in | <i>x</i> | : the vector of the values to compute the Bernoulli function at. |
|----|----------|--|

Returns

the pair  $(\mathfrak{B}(x), \mathfrak{B}(-x))$ .

#### 7.1.3.3 void assembleAdvDiff ( const VectorXd & alpha, const VectorXd & gamma, const VectorXd & eta, const VectorXd & beta ) [override], [virtual]

Assemble the matrix for an advection-diffusion term.

Build the Scharfetter-Gummel stabilized stiffness matrix for:  $-\nabla \cdot (\alpha \cdot \gamma(\eta \nabla u - \beta u)) = f$ .

## Parameters

|    |              |   |
|----|--------------|---|
| in | <i>alpha</i> | : $\alpha$ , an element-wise constant function; |
| in | <i>gamma</i> | : $\gamma$ , an element-wise linear function;   |
| in | <i>eta</i>   | : $\eta$ , an element-wise linear function;     |
| in | <i>beta</i>  | : $\beta$ , an element-wise constant function.  |

Implements [PdeSolver1D](#).

**7.1.3.4** `void assembleStiff ( const VectorXd & eps, const VectorXd & kappa )` [override],[virtual]

Assemble the matrix for a diffusion term.

Build the standard finite element stiffness matrix for the diffusion problem:  $-\nabla \cdot (\varepsilon \cdot \kappa \nabla u) = f$ .

## Parameters

|    |              |  |
|----|--------------|--|
| in | <i>eps</i>   | : $\varepsilon$ , an element-wise constant function; |
| in | <i>kappa</i> | : $\kappa$ , an element-wise linear function.        |

Implements [PdeSolver1D](#).

**7.1.3.5** `void assembleMass ( const VectorXd & delta, const VectorXd & zeta )` [override],[virtual]

Assemble the matrix for a reaction term.

Build the lumped finite element mass matrix for the reaction problem:  $\delta \cdot \zeta u = f$ .

## Parameters

|    |              |   |
|----|--------------|---|
| in | <i>delta</i> | : $\delta$ , an element-wise constant function; |
| in | <i>zeta</i>  | : $\zeta$ , an element-wise linear function.    |

Implements [PdeSolver1D](#).

The documentation for this class was generated from the following files:

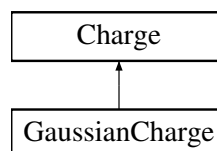
- /home/Data/Dropbox/Progetto-PACS/C++/Source/src/[solvers.h](#)
- /home/Data/Dropbox/Progetto-PACS/C++/Source/src/[solvers.cpp](#)

## 7.2 Charge Class Reference

Abstract class providing methods to calculate total electric charge (the rhs in the Poisson equation).

```
#include <charge.h>
```

Inheritance diagram for Charge:



### Public Member Functions

- [Charge](#) ()=delete

*Default constructor (deleted since it is required to specify a [ParamList](#) and a [QuadratureRule](#)).*

- [Charge](#) (const [ParamList](#) &, const [QuadratureRule](#) &)  
*Constructor.*
- virtual [~Charge](#) ()=default  
*Destructor (defaulted).*
- virtual VectorXd [charge](#) (const VectorXd &phi)=0  
*Compute the total charge.*
- virtual VectorXd [dcharge](#) (const VectorXd &phi)=0  
*Compute the derivative of the total charge with respect to the electric potential.*

## Protected Attributes

- const [ParamList](#) & [params\\_](#)  
*Parameter list handler.*
- const [QuadratureRule](#) & [rule\\_](#)  
*Quadrature rule handler.*

## 7.2.1 Detailed Description

Abstract class providing methods to calculate total electric charge (the rhs in the Poisson equation).

## 7.2.2 Constructor & Destructor Documentation

### 7.2.2.1 [Charge](#) ( const [ParamList](#) & *params*, const [QuadratureRule](#) & *rule* )

Constructor.

Parameters

|    |               |                                      |
|----|---------------|--------------------------------------|
| in | <i>params</i> | : the list of simulation parameters; |
| in | <i>rule</i>   | : a quadrature rule.                 |

## 7.2.3 Member Function Documentation

### 7.2.3.1 virtual VectorXd [charge](#) ( const VectorXd & *phi* ) [pure virtual]

Compute the total charge.

Parameters

|    |            |                                      |
|----|------------|--------------------------------------|
| in | <i>phi</i> | : the electric potential $\varphi$ . |
|----|------------|--------------------------------------|

Returns

the total charge  $q$  [C].

Implemented in [GaussianCharge](#).

### 7.2.3.2 virtual VectorXd [dcharge](#) ( const VectorXd & *phi* ) [pure virtual]

Compute the derivative of the total charge with respect to the electric potential.



## Parameters

|                 |                  |                                      |
|-----------------|------------------|--------------------------------------|
| <code>in</code> | <code>phi</code> | : the electric potential $\varphi$ . |
|-----------------|------------------|--------------------------------------|

## Returns

the derivative:  $\frac{dq}{d\varphi} [C \cdot V^{-1}]$ .

Implemented in [GaussianCharge](#).

The documentation for this class was generated from the following files:

- `/home/Data/Dropbox/Progetto-PACS/C++/Source/src/charge.h`
- `/home/Data/Dropbox/Progetto-PACS/C++/Source/src/charge.c++`

## 7.3 CsvParser Class Reference

Class providing methods to read content from a .csv file and store it in matrices or vectors.

```
#include <csvParser.h>
```

### Public Member Functions

- [CsvParser](#) ()=delete  
*Default constructor (deleted since it is required to specify at least a filename).*
- [CsvParser](#) (const std::string &, const bool &=true)  
*Constructor: load the input file and check its compatibility with the code.*
- virtual [~CsvParser](#) ()  
*Destructor: close the input file.*
- RowVectorXd [importRow](#) (const unsigned &)  
*Method to import a row from the input file.*
- MatrixXd [importRows](#) (const std::initializer\_list< unsigned > &)  
*Method to import multiple rows from the input file.*
- MatrixXd [importFirstRows](#) (const unsigned &)  
*Method to import the first nRows rows from the input file.*
- VectorXd [importCol](#) (const unsigned &)  
*Method to import a column from the input file.*
- MatrixXd [importCols](#) (const std::initializer\_list< unsigned > &)  
*Method to import multiple columns from the input file.*
- MatrixXd [importFirstCols](#) (const unsigned &)  
*Method to import the first nCols columns from the input file.*
- double [importCell](#) (const unsigned &, const unsigned &)  
*Method to import a single cell from the input file.*
- MatrixXd [importAll](#) ()  
*Method to import the whole input file.*

### Getter methods.

- const unsigned & [nRows](#) () const
- const unsigned & [nCols](#) () const

## Private Member Functions

- void `reset ()`  
*Reset all the flags for `input_` and go back to the beginning of file (possibly by ignoring headers).*

## Private Attributes

- bool `hasHeaders_`  
*bool to determine if first row contains header information or not.*
- unsigned `nRows_`  
*No. of rows in the input file.*
- unsigned `nCols_`  
*No. of columns in the input file.*
- `std::ifstream` `input_`  
*Input stream to `input_filename`.*
- `std::string` `line_`  
*Auxiliary variable to store currently processed line.*
- char `separator_`  
*The separator character detected.*

### 7.3.1 Detailed Description

Class providing methods to read content from a .csv file and store it in matrices or vectors.

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 `CsvParser ( const std::string & input_filename, const bool & hasHeaders = true )`

Constructor: load the input file and check its compatibility with the code.

##### Parameters

|    |                             |  |
|----|-----------------------------|--|
| in | <code>input_filename</code> | : the input filename;  |
| in | <code>hasHeaders</code>     | : bool to determine if first row contains header information or not; if <b>true</b> , first row is always ignored. |

### 7.3.3 Member Function Documentation

#### 7.3.3.1 `RowVectorXd importRow ( const unsigned & index )`

Method to import a row from the input file.

##### Parameters

|    |                    |                  |
|----|--------------------|------------------|
| in | <code>index</code> | : the row index. |
|----|--------------------|------------------|

##### Returns

a row vector containing the content read.

#### 7.3.3.2 `MatrixXd importRows ( const std::initializer_list< unsigned > & indexes )`

Method to import multiple rows from the input file.

**Parameters**

|           |                |  |
|-----------|----------------|--|
| <i>in</i> | <i>indexes</i> | : initializer list containing the row indexes (e.g. something like {1, 3, 4}). |
|-----------|----------------|--|

**Returns**

a matrix containing the content read (row by row).

**7.3.3.3 MatrixXd importFirstRows ( const unsigned & *nRows* )**

Method to import the first *nRows* rows from the input file.

**Parameters**

|           |              |                                 |
|-----------|--------------|---------------------------------|
| <i>in</i> | <i>nRows</i> | : the number of rows to import. |
|-----------|--------------|---------------------------------|

**Returns**

a matrix containing the content read (row by row).

**7.3.3.4 VectorXd importCol ( const unsigned & *index* )**

Method to import a column from the input file.

**Parameters**

|           |              |                     |
|-----------|--------------|---------------------|
| <i>in</i> | <i>index</i> | : the column index. |
|-----------|--------------|---------------------|

**Returns**

a column vector containing the content read.

**7.3.3.5 MatrixXd importCols ( const std::initializer\_list< unsigned > & *indexes* )**

Method to import multiple columns from the input file.

**Parameters**

|           |                |   |
|-----------|----------------|---|
| <i>in</i> | <i>indexes</i> | : initializer list containing the column indexes (e.g. something like {1, 3, 4}). |
|-----------|----------------|---|

**Returns**

a matrix containing the content read (column by column).

**7.3.3.6 MatrixXd importFirstCols ( const unsigned & *nCols* )**

Method to import the first *nCols* columns from the input file.

**Parameters**

|           |              |                                    |
|-----------|--------------|------------------------------------|
| <i>in</i> | <i>nCols</i> | : the number of columns to import. |
|-----------|--------------|------------------------------------|

**Returns**

a matrix containing the content read (column by column).

7.3.3.7 `double importCell ( const unsigned & rowIndex, const unsigned & colIndex )`

Method to import a single cell from the input file.

## Parameters

|    |                 |                          |
|----|-----------------|--------------------------|
| in | <i>rowIndex</i> | : the cell row index.    |
| in | <i>colIndex</i> | : the cell column index. |

## Returns

a scalar containing the value read.

## 7.3.3.8 MatrixXd importAll ( )

Method to import the whole input file.

## Returns

a matrix containing the content read (cell by cell).

The documentation for this class was generated from the following files:

- /home/Data/Dropbox/Progetto-PACS/C++/Source/src/[csvParser.h](#)
- /home/Data/Dropbox/Progetto-PACS/C++/Source/src/[csvParser.cpp](#)

## 7.4 DosModel Class Reference

Class providing methods to process a simulation to extract the Density of States starting from a parameter list.

```
#include <dosModel.h>
```

## Public Member Functions

- [DosModel](#) ()  
*Default constructor.*
- [DosModel](#) (const [ParamList](#) &)  
*Explicit conversion constructor.*
- virtual [~DosModel](#) ()=default  
*Destructor (defaulted).*
- const [ParamList](#) & [params](#) () const  
*Getter method.*
- void [simulate](#) (const GetPot &, const std::string &, const std::string &, const std::string &, const std::string &) const  
*Perform the simulation.*
- void [post\\_process](#) (const GetPot &, const std::string &, std::ostream &, std::ostream &, const VectorXd &, const VectorXd &, const VectorXd &, const VectorXd &) const  
*Perform post-processing.*
- void [gnuplot\\_commands](#) (const std::string &, std::ostream &) const  
*Defines commands to generate Gnuplot output files.*
- void [save\\_plot](#) (const std::string &, const std::string &, const std::string &, const std::string &) const  
*Save the Gnuplot output files.*

## Private Attributes

- bool `initialized_`  
bool to determine if `DosModel` `param_` has been properly initialized.
- `ParamList` `params_`  
The parameter list.

### 7.4.1 Detailed Description

Class providing methods to process a simulation to extract the Density of States starting from a parameter list.

### 7.4.2 Constructor & Destructor Documentation

#### 7.4.2.1 `DosModel ( const ParamList & params ) [explicit]`

Explicit conversion constructor.

##### Parameters

|    |                     |                     |
|----|---------------------|---------------------|
| in | <code>params</code> | : a parameter list. |
|----|---------------------|---------------------|

### 7.4.3 Member Function Documentation

#### 7.4.3.1 `void simulate ( const GetPot & config, const std::string & input_experim, const std::string & output_directory, const std::string & output_plot_subdir, const std::string & output_filename ) const`

Perform the simulation.

##### Parameters

|    |                                 |   |
|----|---------------------------------|---|
| in | <code>config</code>             | : the GetPot configuration object;            |
| in | <code>input_experim</code>      | : the file containing experimental data;      |
| in | <code>output_directory</code>   | : directory where to store output files;      |
| in | <code>output_plot_subdir</code> | : sub-directory where to store Gnuplot files; |
| in | <code>output_filename</code>    | : prefix for the output filename.             |

#### 7.4.3.2 `void post_process ( const GetPot & config, const std::string & input_experim, std::ostream & output_fitting, std::ostream & output_CV, const VectorXd & x_semic, const VectorXd & dens, const VectorXd & V_simulated, const VectorXd & C_simulated ) const`

Perform post-processing.

##### Parameters

|     |                             |   |
|-----|-----------------------------|---|
| in  | <code>config</code>         | : the GetPot configuration object;                              |
| in  | <code>input_experim</code>  | : the file containing experimental data;                        |
| out | <code>output_fitting</code> | : output file containing infos about fitting experimental data; |
| out | <code>output_CV</code>      | : output file containing infos about capacitance-voltage data;  |
| in  | <code>x_semic</code>        | : the mesh corresponding to the semiconductor domain;           |
| in  | <code>dens</code>           | : charge density;   |

|    |                    |                                 |
|----|--------------------|---------------------------------|
| in | <i>V_simulated</i> | : simulated voltage values;     |
| in | <i>C_simulated</i> | : simulated capacitance values. |

#### 7.4.3.3 void gnuplot\_commands ( const std::string & output\_CV\_filename, std::ostream & os ) const

Defines commands to generate Gnuplot output files.

##### Parameters

|     |                           |                       |
|-----|---------------------------|-----------------------|
| in  | <i>output_CV_filename</i> | : output CV filename; |
| out | <i>os</i>                 | : output stream.      |

#### 7.4.3.4 void save\_plot ( const std::string & output\_directory, const std::string & output\_plot\_subdir, const std::string & output\_CV\_filename, const std::string & output\_filename ) const

Save the Gnuplot output files.

##### Parameters

|    |                           |   |
|----|---------------------------|---|
| in | <i>output_directory</i>   | : directory where to store output files;      |
| in | <i>output_plot_subdir</i> | : sub-directory where to store Gnuplot files; |
| in | <i>output_CV_filename</i> | : output CV filename;                         |
| in | <i>output_filename</i>    | : prefix for the output filename.             |

The documentation for this class was generated from the following files:

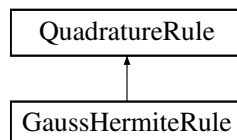
- /home/Data/Dropbox/Progetto-PACS/C++/Source/src/dosModel.h
- /home/Data/Dropbox/Progetto-PACS/C++/Source/src/dosModel.c++

## 7.5 GaussHermiteRule Class Reference

Class derived from [QuadratureRule](#) providing the Gauss-Hermite quadrature rule.

```
#include <quadratureRule.h>
```

Inheritance diagram for GaussHermiteRule:



### Public Member Functions

- [GaussHermiteRule](#) ()=delete  
*Default constructor (deleted since it is required to specify the no. of nodes).*
- [GaussHermiteRule](#) (const unsigned &)  
*Constructor.*
- virtual [~GaussHermiteRule](#) ()=default  
*Destructor (defaulted).*

- virtual void [apply](#) () override  
*Apply the quadrature rule in order to compute the nodes and weights.*
- void [apply](#) (const GetPot &)  
*Apply the quadrature rule reading parameters from a configuration file.*
- void [apply\\_iterative\\_algorithm](#) (const unsigned &=1000, const double &=1.0e-14)  
*Compute nodes and weights using an adapted version of the algorithm presented in: William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. Numerical Recipes: The Art of Scientific Computing (3rd edition). Cambridge University Press, New York, NY, USA.*
- void [apply\\_using\\_eigendecomposition](#) ()  
*Compute nodes and weights using an eigendecomposition-based algorithm.*

## Additional Inherited Members

### 7.5.1 Detailed Description

Class derived from [QuadratureRule](#) providing the Gauss-Hermite quadrature rule.

Compute nodes and weights for the  $nNodes\_$  -points approximation of

$$\int_{-\infty}^{+\infty} w(x)f(x) \, dx$$

where  $w(x) = e^{-x^2}$ .

### 7.5.2 Constructor & Destructor Documentation

#### 7.5.2.1 GaussHermiteRule ( const unsigned & $nNodes$ )

Constructor.

Parameters

|                 |                                  |  |
|-----------------|----------------------------------|--|
| <code>in</code> | <code><math>nNodes</math></code> | : the no. of nodes to be used for the quadrature rule. |
|-----------------|----------------------------------|--|

### 7.5.3 Member Function Documentation

#### 7.5.3.1 void [apply](#) ( const GetPot & *config* )

Apply the quadrature rule reading parameters from a configuration file.

Parameters

|                 |                            |                                    |
|-----------------|----------------------------|------------------------------------|
| <code>in</code> | <code><i>config</i></code> | : the GetPot configuration object. |
|-----------------|----------------------------|------------------------------------|

The documentation for this class was generated from the following files:

- `/home/Data/Dropbox/Progetto-PACS/C++/Source/src/quadratureRule.h`
- `/home/Data/Dropbox/Progetto-PACS/C++/Source/src/quadratureRule.c++`

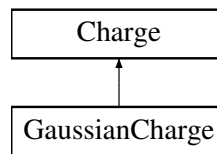
## 7.6 GaussianCharge Class Reference

Class derived from [Charge](#), under the hypothesis that Density of States is a combination of gaussians.

```
#include <charge.h>
```



Inheritance diagram for GaussianCharge:



## Public Member Functions

- [GaussianCharge](#) ()=delete  
*Default constructor (deleted since it is required to specify a [ParamList](#) and a [QuadratureRule](#)).*
- [GaussianCharge](#) (const [ParamList](#) &, const [QuadratureRule](#) &)  
*Constructor.*
- virtual [~GaussianCharge](#) ()=default  
*Destructor (defaulted).*
- virtual VectorXd [charge](#) (const VectorXd &) override  
*Compute the total charge.*
- virtual VectorXd [dcharge](#) (const VectorXd &) override  
*Compute the derivative of the total charge with respect to the electric potential.*

## Private Member Functions

- double [n\\_approx](#) (const double &, const double &, const double &) const  
*Compute electrons density (per unit volume).*
- double [dn\\_approx](#) (const double &, const double &, const double &) const  
*Compute the approximate derivative of electrons density (per unit volume) with respect to the electric potential.*

## Additional Inherited Members

### 7.6.1 Detailed Description

Class derived from [Charge](#), under the hypothesis that Density of States is a combination of gaussians.

Provide methods to compute total electric charge and its derivative under the hypothesis that Density of States is a linear combination of multiple gaussians, whose parameters are read from a [ParamList](#) object.

### 7.6.2 Constructor & Destructor Documentation

#### 7.6.2.1 [GaussianCharge](#) ( const [ParamList](#) & *params*, const [QuadratureRule](#) & *rule* )

Constructor.

Parameters

|    |               |                                      |
|----|---------------|--------------------------------------|
| in | <i>params</i> | : the list of simulation parameters; |
| in | <i>rule</i>   | : a quadrature rule.                 |

### 7.6.3 Member Function Documentation

#### 7.6.3.1 VectorXd [charge](#) ( const VectorXd & *phi* ) `[override]`, `[virtual]`

Compute the total charge.

## Parameters

|    |            |                                      |
|----|------------|--------------------------------------|
| in | <i>phi</i> | : the electric potential $\varphi$ . |
|----|------------|--------------------------------------|

## Returns

the total charge  $q$  [C].

Implements [Charge](#).

7.6.3.2 `VectorXd dcharge ( const VectorXd & phi ) [override],[virtual]`

Compute the derivative of the total charge with respect to the electric potential.

## Parameters

|    |            |                                      |
|----|------------|--------------------------------------|
| in | <i>phi</i> | : the electric potential $\varphi$ . |
|----|------------|--------------------------------------|

## Returns

the derivative:  $\frac{dq}{d\varphi}$  [C · V<sup>-1</sup>].

Implements [Charge](#).

7.6.3.3 `double n_approx ( const double & phi, const double & N0, const double & sigma ) const [private]`

Compute electrons density (per unit volume).

## Parameters

|    |              |  |
|----|--------------|--|
| in | <i>phi</i>   | : the electric potential $\varphi$ ;         |
| in | <i>N0</i>    | : the gaussian mean $N_0$ ;                  |
| in | <i>sigma</i> | : the gaussian standard deviation $\sigma$ . |

## Returns

the electrons density  $n(\varphi)$  [m<sup>-3</sup>].

7.6.3.4 `double dn_approx ( const double & phi, const double & N0, const double & sigma ) const [private]`

Compute the approximate derivative of electrons density (per unit volume) with respect to the electric potential.

## Parameters

|    |              |  |
|----|--------------|--|
| in | <i>phi</i>   | : the electric potential $\varphi$ ;         |
| in | <i>N0</i>    | : the gaussian mean $N_0$ ;                  |
| in | <i>sigma</i> | : the gaussian standard deviation $\sigma$ . |

## Returns

the derivative:  $\frac{dn}{d\varphi}$  [m<sup>-3</sup> · V<sup>-1</sup>].

The documentation for this class was generated from the following files:

- /home/Data/Dropbox/Progetto-PACS/C++/Source/src/[charge.h](#)
- /home/Data/Dropbox/Progetto-PACS/C++/Source/src/charge.c++

## 7.7 NonLinearPoisson1D Class Reference

Provide a solver for a non-linear Poisson equation.

```
#include <solvers.h>
```

### Public Member Functions

- [NonLinearPoisson1D](#) ()=delete  
*Default constructor (deleted since it is required to specify the solver to be used).*
- [NonLinearPoisson1D](#) (const [PdeSolver1D](#) &, const unsigned &=100, const double &=1.0e-6)  
*Constructor.*
- virtual [~NonLinearPoisson1D](#) ()=default  
*Destructor (defaulted).*
- void [apply](#) (const VectorXd &, const VectorXd &, [Charge](#) &)  
*Apply a Newton method to the equation and then discretize it using the solver specified.*

### Getter methods.

- const VectorXd & [phi](#) () const
- const VectorXd & [norm](#) () const
- const double & [qTot](#) () const
- const double & [cTot](#) () const

### Private Member Functions

- [SparseXd computeJac](#) (const VectorXd &) const  
*Compute the Jacobi matrix.*

### Private Attributes

- const [PdeSolver1D](#) & [solver\\_](#)  
*Solver handler.*
- unsigned [maxIterationsNo\\_](#)  
*Maximum no. of iterations.*
- double [tolerance\\_](#)  
*Tolerance.*
- VectorXd [phi\\_](#)  
*The electric potential.*
- VectorXd [norm\\_](#)  
*Vector holding  $L^\infty$ -norm errors for each iteration.*
- double [qTot\\_](#)  
*Total charge.*
- double [cTot\\_](#)  
*Total capacitance.*

#### 7.7.1 Detailed Description

Provide a solver for a non-linear Poisson equation.

A Newton method is applied in order to solve:

$$-\frac{d}{dz} \left( \epsilon(z) \cdot \frac{d\varphi}{dz}(z) \right) = -q \cdot \frac{N_0}{\sqrt{\pi}} \int_{-\infty}^{+\infty} \exp(-\alpha^2) \left( 1 + \exp \left( \frac{\sqrt{2}\sigma\alpha - q\varphi(z)}{K_B \cdot T} \right) \right)^{-1} d\alpha .$$

## 7.7.2 Constructor & Destructor Documentation

### 7.7.2.1 NonLinearPoisson1D ( const PdeSolver1D & solver, const unsigned & maxIterationsNo = 100, const double & tolerance = 1.0e-6 )

Constructor.

Parameters

|    |                        |                                      |
|----|------------------------|--------------------------------------|
| in | <i>solver</i>          | : the solver to be used;             |
| in | <i>maxIterationsNo</i> | : maximum no. of iterations desired; |
| in | <i>tolerance</i>       | : tolerance desired.                 |

## 7.7.3 Member Function Documentation

### 7.7.3.1 void apply ( const VectorXd & mesh, const VectorXd & init\_guess, Charge & charge\_fun )

Apply a Newton method to the equation and then discretize it using the solver specified.

Parameters

|    |                   |  |
|----|-------------------|--|
| in | <i>mesh</i>       | : the mesh;  |
| in | <i>init_guess</i> | : initial guess for the Newton algorithm;  |
| in | <i>charge_fun</i> | : an object of class <a href="#">Charge</a> specifying how to compute total electric charge. |

### 7.7.3.2 SparseXd computeJac ( const VectorXd & x ) const [private]

Compute the Jacobi matrix.

Parameters

|    |          |                                   |
|----|----------|-----------------------------------|
| in | <i>x</i> | : the vector where to start from. |
|----|----------|-----------------------------------|

Returns

the Jacobi matrix in a sparse format.

The documentation for this class was generated from the following files:

- /home/Data/Dropbox/Progetto-PACS/C++/Source/src/[solvers.h](#)
- /home/Data/Dropbox/Progetto-PACS/C++/Source/src/[solvers.c++](#)

## 7.8 ParamList Class Reference

Class providing methods to handle a list of parameters.

```
#include <paramList.h>
```

### Public Member Functions

- [ParamList](#) ()=default  
*Default constructor (defaulted).*
- [ParamList](#) (const RowVectorXd &)  
*Explicit conversion constructor.*
- virtual [~ParamList](#) ()=default

*Destructor (defaulted).*

#### Getter methods.

- const unsigned & **simulationNo** () const
- const double & **t\_semic** () const
- const double & **t\_ins** () const
- const double & **eps\_semic** () const
- const double & **eps\_ins** () const
- const double & **Wf** () const
- const double & **Ea** () const
- const double & **N0** () const
- const double & **sigma** () const
- const double & **N0\_2** () const
- const double & **sigma\_2** () const
- const double & **shift\_2** () const
- const double & **N0\_3** () const
- const double & **sigma\_3** () const
- const double & **shift\_3** () const
- const double & **N0\_4** () const
- const double & **sigma\_4** () const
- const double & **shift\_4** () const
- const unsigned & **nNodes** () const
- const unsigned & **nSteps** () const
- const double & **V\_min** () const
- const double & **V\_max** () const

#### Private Attributes

- unsigned [simulationNo\\_](#)  
*Index of the simulation.*
- double [t\\_semic\\_](#)  
*Semiconductor layer thickness [m].*
- double [t\\_ins\\_](#)  
*Insulator layer thickness [m].*
- double [eps\\_semic\\_](#)  
*Semiconductor layer relative electrical permittivity [ ].*
- double [eps\\_ins\\_](#)  
*Insulator layer relative electrical permittivity [ ].*
- double [Wf\\_](#)  
*Work-function [V].*
- double [Ea\\_](#)  
*Electron affinity [V].*
- double [N0\\_](#)  
*1st gaussian mean [ $m^{-3}$ ].*
- double [sigma\\_](#)  
*1st gaussian standard deviation (normalized by  $K_B \cdot T$ ) [ ].*
- double [N0\\_2\\_](#)  
*2nd gaussian mean.*
- double [sigma\\_2\\_](#)  
*2nd gaussian standard deviation.*
- double [shift\\_2\\_](#)  
*2nd gaussian shift with respect to the 1st gaussian electric potential.*
- double [N0\\_3\\_](#)  
*3rd gaussian mean.*

- double [sigma\\_3\\_](#)  
*3rd gaussian standard deviation.*
- double [shift\\_3\\_](#)  
*3rd gaussian shift with respect to the 1st gaussian electric potential.*
- double [N0\\_4\\_](#)  
*4th gaussian mean.*
- double [sigma\\_4\\_](#)  
*4th gaussian standard deviation.*
- double [shift\\_4\\_](#)  
*4th gaussian shift with respect to the 1st gaussian electric potential.*
- unsigned [nNodes\\_](#)  
*No. of nodes that form the mesh.*
- unsigned [nSteps\\_](#)  
*No. of steps to simulate.*
- double [V\\_min\\_](#)  
*Minimum voltage [V].*
- double [V\\_max\\_](#)  
*Maximum voltage [V].*

## Friends

- class **GaussianCharge**
- class **DosModel**

### 7.8.1 Detailed Description

Class providing methods to handle a list of parameters.

It can include up to 4 gaussians, later combined to compute total charge.

### 7.8.2 Constructor & Destructor Documentation

#### 7.8.2.1 `ParamList ( const RowVectorXd & list ) [explicit]`

Explicit conversion constructor.

##### Parameters

|                 |                   |  |
|-----------------|-------------------|--|
| <code>in</code> | <code>list</code> | : a row vector containing a parameters list (for example got by a <a href="#">CsvParser</a> object). Parameters should be sorted in the same order as specified above. |
|-----------------|-------------------|--|

The documentation for this class was generated from the following files:

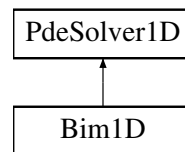
- `/home/Data/Dropbox/Progetto-PACS/C++/Source/src/paramList.h`
- `/home/Data/Dropbox/Progetto-PACS/C++/Source/src/paramList.c++`

## 7.9 PdeSolver1D Class Reference

Abstract class providing methods to assemble matrices to solve one-dimensional PDEs.

```
#include <solvers.h>
```

Inheritance diagram for PdeSolver1D:



## Public Member Functions

- [PdeSolver1D](#) ()=delete  
*Default constructor (deleted since it is required to specify the mesh).*
- [PdeSolver1D](#) (VectorXd &)  
*Constructor.*
- virtual [~PdeSolver1D](#) ()=default  
*Destructor (defaulted).*
- virtual void [assembleAdvDiff](#) (const VectorXd &alpha, const VectorXd &gamma, const VectorXd &eta, const VectorXd &beta)=0  
*Assemble the matrix for an advection-diffusion term.*
- virtual void [assembleStiff](#) (const VectorXd &eps, const VectorXd &kappa)=0  
*Assemble the matrix for a diffusion term.*
- virtual void [assembleMass](#) (const VectorXd &delta, const VectorXd &zeta)=0  
*Assemble the matrix for a reaction term.*

## Getter methods.

- const [SparseXd](#) & **AdvDiff** () const
- const [SparseXd](#) & **Stiff** () const
- const [SparseXd](#) & **Mass** () const

## Protected Attributes

- VectorXd [mesh\\_](#)  
*The mesh.*
- unsigned [nNodes\\_](#)  
*No. of nodes that form the mesh.*
- [SparseXd](#) [AdvDiff\\_](#)  
*Matrix for an advection-diffusion term.*
- [SparseXd](#) [Stiff\\_](#)  
*Stiffness matrix.*
- [SparseXd](#) [Mass\\_](#)  
*Mass matrix.*

## Friends

- class **NonLinearPoisson1D**

### 7.9.1 Detailed Description

Abstract class providing methods to assemble matrices to solve one-dimensional PDEs.

Matrices are held in a sparse format.

## 7.9.2 Constructor & Destructor Documentation

### 7.9.2.1 PdeSolver1D ( *VectorXd* & *mesh* )

Constructor.



## Parameters

|                 |                   |             |
|-----------------|-------------------|-------------|
| <code>in</code> | <code>mesh</code> | : the mesh. |
|-----------------|-------------------|-------------|

## 7.9.3 Member Function Documentation

**7.9.3.1** `virtual void assembleAdvDiff ( const VectorXd & alpha, const VectorXd & gamma, const VectorXd & eta, const VectorXd & beta ) [pure virtual]`

Assemble the matrix for an advection-diffusion term.

Build the matrix for the advection-diffusion problem:  $-\nabla \cdot (\alpha \cdot \gamma (\eta \nabla u - \beta u)) = f$ .

## Parameters

|                 |                    |   |
|-----------------|--------------------|---|
| <code>in</code> | <code>alpha</code> | : $\alpha$ , an element-wise constant function; |
| <code>in</code> | <code>gamma</code> | : $\gamma$ , an element-wise linear function;   |
| <code>in</code> | <code>eta</code>   | : $\eta$ , an element-wise linear function;     |
| <code>in</code> | <code>beta</code>  | : $\beta$ , an element-wise constant function.  |

Implemented in [Bim1D](#).

**7.9.3.2** `virtual void assembleStiff ( const VectorXd & eps, const VectorXd & kappa ) [pure virtual]`

Assemble the matrix for a diffusion term.

Build the matrix for the diffusion problem:  $-\nabla \cdot (\varepsilon \cdot \kappa \nabla u) = f$ .

## Parameters

|                 |                    |  |
|-----------------|--------------------|--|
| <code>in</code> | <code>eps</code>   | : $\varepsilon$ , an element-wise constant function; |
| <code>in</code> | <code>kappa</code> | : $\kappa$ , an element-wise linear function.        |

Implemented in [Bim1D](#).

**7.9.3.3** `virtual void assembleMass ( const VectorXd & delta, const VectorXd & zeta ) [pure virtual]`

Assemble the matrix for a reaction term.

Build the mass matrix for the reaction problem:  $\delta \cdot \zeta u = f$ .

## Parameters

|                 |                    |   |
|-----------------|--------------------|---|
| <code>in</code> | <code>delta</code> | : $\delta$ , an element-wise constant function; |
| <code>in</code> | <code>zeta</code>  | : $\zeta$ , an element-wise linear function.    |

Implemented in [Bim1D](#).

The documentation for this class was generated from the following files:

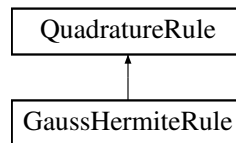
- `/home/Data/Dropbox/Progetto-PACS/C++/Source/src/solvers.h`
- `/home/Data/Dropbox/Progetto-PACS/C++/Source/src/solvers.cpp`

## 7.10 QuadratureRule Class Reference

Abstract class providing a quadrature rule.

```
#include <quadratureRule.h>
```

Inheritance diagram for QuadratureRule:



## Public Member Functions

- [QuadratureRule](#) ()=delete  
*Default constructor (deleted since it is required to specify the no. of nodes).*
- [QuadratureRule](#) (const unsigned &)  
*Constructor.*
- virtual [~QuadratureRule](#) ()=default  
*Destructor (defaulted).*
- virtual void [apply](#) ()=0  
*Apply the quadrature rule in order to compute the nodes and weights.*

## Getter methods.

- const unsigned & **nNodes** () const
- const VectorXd & **nodes** () const
- const VectorXd & **weights** () const

## Protected Attributes

- unsigned [nNodes\\_](#)  
*The no. of nodes to be used for the quadrature rule.*
- VectorXd [nodes\\_](#)  
*Vector containing the computed nodes coordinates.*
- VectorXd [weights\\_](#)  
*Vector containing the computed weights.*

## Friends

- class **GaussianCharge**

### 7.10.1 Detailed Description

Abstract class providing a quadrature rule.

Approximate the integral:

$$\int_a^b f(x) \, dx$$

with the finite sum:

$$\sum_{i=1}^{nNodes} w_i \cdot f(x_i)$$

where  $\{x_i\}_{i=1}^{nNodes}$  and  $\{w_i\}_{i=1}^{nNodes}$  are called respectively nodes and weights.

### 7.10.2 Constructor & Destructor Documentation

#### 7.10.2.1 [QuadratureRule](#) ( const unsigned & *nNodes* )

Constructor.

## Parameters

|                 |                     |  |
|-----------------|---------------------|--|
| <code>in</code> | <code>nNodes</code> | : the no. of nodes to be used for the quadrature rule. |
|-----------------|---------------------|--|

The documentation for this class was generated from the following files:

- [/home/Data/Dropbox/Progetto-PACS/C++/Source/src/quadratureRule.h](#)
- [/home/Data/Dropbox/Progetto-PACS/C++/Source/src/quadratureRule.c++](#)



## Chapter 8

# File Documentation

### 8.1 /home/Data/Dropbox/Progetto-PACS/C++/Source/src/charge.h File Reference

Classes for computing total electric charge.

```
#include "paramList.h"
#include "quadratureRule.h"
#include "typedefs.h"
```

#### Classes

- class [Charge](#)  
*Abstract class providing methods to calculate total electric charge (the rhs in the Poisson equation).*
- class [GaussianCharge](#)  
*Class derived from [Charge](#), under the hypothesis that Density of States is a combination of gaussians.*

#### 8.1.1 Detailed Description

Classes for computing total electric charge.

##### Author

Pasquale Claudio Africa [pasquale.africa@gmail.com](mailto:pasquale.africa@gmail.com)

##### Date

2014

### 8.2 /home/Data/Dropbox/Progetto-PACS/C++/Source/src/csvParser.h File Reference

Tools to store content from a .csv file in matrices or vectors.

```
#include "typedefs.h"
#include <string>
#include <fstream>
#include <sstream>
#include <utility>
```

## Classes

- class [CsvParser](#)

*Class providing methods to read content from a .csv file and store it in matrices or vectors.*

### 8.2.1 Detailed Description

Tools to store content from a .csv file in matrices or vectors.

#### Author

Pasquale Claudio Africa [pasquale.africa@gmail.com](mailto:pasquale.africa@gmail.com)

#### Date

2014

## 8.3 /home/Data/Dropbox/Progetto-PACS/C++/Source/src/dosModel.h File Reference

Mathematical model for Density of States extraction.

```
#include "charge.h"
#include "csvParser.h"
#include "numerics.h"
#include "paramList.h"
#include "quadratureRule.h"
#include "solvers.h"
#include "typedefs.h"
#include "../include/gnuplot-iostream.h"
#include <chrono>
#include <limits>
```

## Classes

- class [DosModel](#)

*Class providing methods to process a simulation to extract the Density of States starting from a parameter list.*

### 8.3.1 Detailed Description

Mathematical model for Density of States extraction.

#### Author

Pasquale Claudio Africa [pasquale.africa@gmail.com](mailto:pasquale.africa@gmail.com)

#### Date

2014

## 8.4 /home/Data/Dropbox/Progetto-PACS/C++/Source/src/numerics.h File Reference

Generic numeric algorithms.

```
#include "typedefs.h"
#include <limits>
```

### Namespaces

- [numerics](#)

*Namespace for generic numeric algorithms.*

### Functions

- `template<typename ScalarType >`  
[VectorX](#)< ScalarType > [sort](#) (const [VectorX](#)< ScalarType > &vector)  
*Function to sort Eigen vectors.*
- `template<typename ScalarType >`  
[VectorXpair](#)< ScalarType > [sort\\_pair](#) (const [VectorX](#)< ScalarType > &vector)  
*Function to sort Eigen vectors, keeping track of indexes.*
- `double` [trapz](#) (const [VectorXd](#) &x, const [VectorXd](#) &y)  
*Function to compute approximate integral of y with spacing increment specified by x, using trapezoidal rule.*
- `double` [trapz](#) (const [VectorXd](#) &y)  
*Compute the approximate integral of y with unit spacing, using trapezoidal rule.*
- [VectorXd](#) [deriv](#) (const [VectorXd](#) &, const [VectorXd](#) &)  
*Compute the numeric derivative:  $\frac{dy}{dx}$ .*
- `double` [interp1](#) (const [VectorXd](#) &, const [VectorXd](#) &, const double &)  
*Linear 1D interpolation. Interpolate y, defined at points x, at the point xNew.*
- [VectorXd](#) [interp1](#) (const [VectorXd](#) &, const [VectorXd](#) &, const [VectorXd](#) &)  
*Linear 1D interpolation. Interpolate y, defined at points x, at the points xNew.*
- `double` [error\\_L2](#) (const [VectorXd](#) &, const [VectorXd](#) &, const [VectorXd](#) &, const double &)  
*Compute the  $L^2$ -norm error between simulated and interpolated values, using trapz.*

#### 8.4.1 Detailed Description

Generic numeric algorithms.

##### Author

Pasquale Claudio Africa [pasquale.africa@gmail.com](mailto:pasquale.africa@gmail.com)

##### Date

2014

## 8.5 /home/Data/Dropbox/Progetto-PACS/C++/Source/src/paramList.h File Reference

List of simulation parameters.

```
#include "typedefs.h"
```

## Classes

- class [ParamList](#)

*Class providing methods to handle a list of parameters.*

### 8.5.1 Detailed Description

List of simulation parameters.

#### Author

Pasquale Claudio Africa [pasquale.africa@gmail.com](mailto:pasquale.africa@gmail.com)

#### Date

2014

## 8.6 /home/Data/Dropbox/Progetto-PACS/C++/Source/src/physicalConstants.h File Reference

Physical constants.

```
#include "typedefs.h"
```

## Namespaces

- [constants](#)

*Numerical constants.*

## Variables

- const double [Q](#) = 1.602176530000000e-19  
*Electron charge [C].*
- const double [Q2](#) = Q \* Q  
*Electron charge squared [ $C^2$ ].*
- const double [K\\_B](#) = 1.380650500000000e-23  
*Boltzmann's constant [ $J \cdot K^{-1}$ ].*
- const double [EPS0](#) = 8.854187817e-12  
*Vacuum electrical permittivity [ $C \cdot V^{-1} \cdot m^{-1}$ ].*
- const double [T](#) = 300  
*Reference temperature [K].*
- const double [V\\_TH](#) = K\_B \* T / Q  
*Threshold voltage [V].*

### 8.6.1 Detailed Description

Physical constants.



**Author**

Pasquale Claudio Africa [pasquale.africa@gmail.com](mailto:pasquale.africa@gmail.com)

**Date**

2014

## 8.7 /home/Data/Dropbox/Progetto-PACS/C++/Source/src/quadratureRule.h File Reference

Quadrature rules.

```
#include "typedefs.h"
```

**Classes**

- class [QuadratureRule](#)  
*Abstract class providing a quadrature rule.*
- class [GaussHermiteRule](#)  
*Class derived from [QuadratureRule](#) providing the Gauss-Hermite quadrature rule.*

### 8.7.1 Detailed Description

Quadrature rules.

**Author**

Pasquale Claudio Africa [pasquale.africa@gmail.com](mailto:pasquale.africa@gmail.com)

**Date**

2014

## 8.8 /home/Data/Dropbox/Progetto-PACS/C++/Source/src/solvers.h File Reference

Generic solvers for PDEs.

```
#include "charge.h"  
#include "typedefs.h"  
#include <utility>  
#include <limits>
```

**Classes**

- class [PdeSolver1D](#)  
*Abstract class providing methods to assemble matrices to solve one-dimensional PDEs.*
- class [Bim1D](#)  
*Class derived from [PdeSolver1D](#), providing a finite volume Box Integration Method (BIM) solver.*
- class [NonLinearPoisson1D](#)  
*Provide a solver for a non-linear Poisson equation.*

### 8.8.1 Detailed Description

Generic solvers for PDEs.

#### Author

Pasquale Claudio Africa [pasquale.africa@gmail.com](mailto:pasquale.africa@gmail.com)

#### Date

2014

## 8.9 /home/Data/Dropbox/Progetto-PACS/C++/Source/src/typedefs.h File Reference

Typedefs and utility functions.

```
#include "physicalConstants.h"
#include <iostream>
#include <fstream>
#include <Eigen/Dense>
#include <Eigen/Sparse>
#include <GetPot>
```

### Namespaces

- [constants](#)  
*Numerical constants.*

### Typedefs

- `typedef SparseMatrix< double > SparseXd`  
*Typedef for sparse dynamic-sized matrices.*
- `template<typename ScalarType >`  
`using VectorX = Matrix< ScalarType, Dynamic, 1 >`  
*Template alias for Eigen vectors.*
- `template<typename T >`  
`using VectorXpair = VectorX< std::pair< T, unsigned > >`  
*Template alias for an Eigen vector of pairs: (ScalarType, unsigned int).*

### Functions

- `void print\_block (const char *, std::ostream &=std::cout)`  
*Auxiliary function to print a string inside a block.*
- `void print\_done (std::ostream &=std::cout)`  
*Auxiliary function to print a "DONE!" string.*

### Variables

- `const unsigned PARAMS\_NO = 22`  
*Number of parameters required in input file.*
- `const double PI = M_PI`

- $\pi$ .  
const double `SQRT_PI` = std::sqrt(PI)
- $\sqrt{\pi}$ .  
const double `PI_M4` = 0.7511255444649425
- $\pi^{-\frac{1}{4}}$ .  
const double `SQRT_2` = std::sqrt(2)
- $\sqrt{2}$ .

### 8.9.1 Detailed Description

Typedefs and utility functions.

#### Author

Pasquale Claudio Africa [pasquale.africa@gmail.com](mailto:pasquale.africa@gmail.com)

#### Date

2014

### 8.9.2 Typedef Documentation

#### 8.9.2.1 using `VectorX` = `Matrix`<`ScalarType`, `Dynamic`, 1>

Template alias for Eigen vectors.

##### Template Parameters

|                   |                    |
|-------------------|--------------------|
| <i>ScalarType</i> | : the scalar type. |
|-------------------|--------------------|

#### 8.9.2.2 using `VectorXpair` = `VectorX`<std::pair<`T`, unsigned> >

Template alias for an Eigen vector of pairs: (*ScalarType*, unsigned int).

##### Template Parameters

|                   |                    |
|-------------------|--------------------|
| <i>ScalarType</i> | : the scalar type. |
|-------------------|--------------------|

# Index

/home/Data/Dropbox/Progetto-PACS/C++/Source/src/chargecomputeJac  
h, 39 NonLinearPoisson1D, 30  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/csv- constants, 11  
Parser.h, 39 CsvParser, 19  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/dos- CsvParser, 20  
Model.h, 40 CsvParser, 20  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/numerics.importAll, 23  
h, 41 importCell, 21  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/param- importCol, 21  
List.h, 41 importCols, 21  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/physical- importFirstCols, 21  
Constants.h, 42 importFirstRows, 21  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/quadrature.importRow, 20  
Rule.h, 43 importRows, 20  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/solvers.-  
h, 43 dcharge  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/typedefs.-Charge, 18  
h, 44 GaussianCharge, 28  
deriv  
numerics, 13  
dn\_approx  
GaussianCharge, 28  
DosModel, 23  
DosModel, 24  
DosModel, 24  
gnuplot\_commands, 25  
post\_process, 24  
save\_plot, 25  
simulate, 24  
error\_L2  
numerics, 14  
GaussHermiteRule, 25  
apply, 26  
GaussHermiteRule, 26  
GaussHermiteRule, 26  
GaussianCharge, 26  
charge, 27  
dcharge, 28  
dn\_approx, 28  
GaussianCharge, 27  
GaussianCharge, 27  
n\_approx, 28  
gnuplot\_commands  
DosModel, 25  
importAll  
CsvParser, 23  
importCell

/home/Data/Dropbox/Progetto-PACS/C++/Source/src/chargecomputeJac  
h, 39 NonLinearPoisson1D, 30  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/csv- constants, 11  
Parser.h, 39 CsvParser, 19  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/dos- CsvParser, 20  
Model.h, 40 CsvParser, 20  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/numerics.importAll, 23  
h, 41 importCell, 21  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/param- importCol, 21  
List.h, 41 importCols, 21  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/physical- importFirstCols, 21  
Constants.h, 42 importFirstRows, 21  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/quadrature.importRow, 20  
Rule.h, 43 importRows, 20  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/solvers.-  
h, 43 dcharge  
/home/Data/Dropbox/Progetto-PACS/C++/Source/src/typedefs.-Charge, 18  
h, 44 GaussianCharge, 28  
deriv  
numerics, 13  
dn\_approx  
GaussianCharge, 28  
DosModel, 23  
DosModel, 24  
DosModel, 24  
gnuplot\_commands, 25  
post\_process, 24  
save\_plot, 25  
simulate, 24  
error\_L2  
numerics, 14  
GaussHermiteRule, 25  
apply, 26  
GaussHermiteRule, 26  
GaussHermiteRule, 26  
GaussianCharge, 26  
charge, 27  
dcharge, 28  
dn\_approx, 28  
GaussianCharge, 27  
GaussianCharge, 27  
n\_approx, 28  
gnuplot\_commands  
DosModel, 25  
importAll  
CsvParser, 23  
importCell

apply  
GaussHermiteRule, 26  
NonLinearPoisson1D, 30  
assembleAdvDiff  
Bim1D, 16  
PdeSolver1D, 35  
assembleMass  
Bim1D, 17  
PdeSolver1D, 35  
assembleStiff  
Bim1D, 17  
PdeSolver1D, 35  
bernoulli  
Bim1D, 16  
Bim1D, 15  
assembleAdvDiff, 16  
assembleMass, 17  
assembleStiff, 17  
bernoulli, 16  
Bim1D, 16  
Bim1D, 16  
log\_mean, 16  
Charge, 17  
Charge, 18  
charge, 18  
dcharge, 18  
charge  
Charge, 18  
GaussianCharge, 27

- CsvParser, 21
- importCol
  - CsvParser, 21
- importCols
  - CsvParser, 21
- importFirstCols
  - CsvParser, 21
- importFirstRows
  - CsvParser, 21
- importRow
  - CsvParser, 20
- importRows
  - CsvParser, 20
- interp1
  - numerics, 13, 14
- log\_mean
  - Bim1D, 16
- n\_approx
  - GaussianCharge, 28
- NonLinearPoisson1D, 29
  - apply, 30
  - computeJac, 30
  - NonLinearPoisson1D, 30
  - NonLinearPoisson1D, 30
- numerics, 11
  - deriv, 13
  - error\_L2, 14
  - interp1, 13, 14
  - sort, 12
  - sort\_pair, 12
  - trapz, 13
- ParamList, 30
  - ParamList, 32
  - ParamList, 32
- PdeSolver1D, 32
  - assembleAdvDiff, 35
  - assembleMass, 35
  - assembleStiff, 35
  - PdeSolver1D, 34
  - PdeSolver1D, 34
- post\_process
  - DosModel, 24
- QuadratureRule, 35
  - QuadratureRule, 36
  - QuadratureRule, 36
- save\_plot
  - DosModel, 25
- simulate
  - DosModel, 24
- sort
  - numerics, 12
- sort\_pair
  - numerics, 12
- trapz
  - numerics, 13
- typedefs.h
  - VectorX, 45
  - VectorXpair, 45
- VectorX
  - typedefs.h, 45
- VectorXpair
  - typedefs.h, 45