# STRING_ARMA

2.0

Generated by Doxygen 1.8.17

Chapter 1

# Class Index

## 1.1 Class List

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

# Chapter 2

# Namespace Documentation

## 2.1 stringarma Namespace Reference

### Classes

- class BypassDiode

  *Represents a common component of a photovoltaic generator, a bypass diode.*
- struct CellsGroup

  *Structure to gather global information of a group of cells that share, at least, the same short-circuit current.*
- struct Classcomp

  *Comparison function object for the multimap.*
- struct SameIshortcircuitAndVbreakdownGroup

  *Structure with info of groups of cells with the same Isc and Vbrx.*
- struct SameIshortcircuitGroup

  *Vector meant to contain info of groups of cells with the same short-circuit current (Isc).*
- class solar_string

  *Represents a string of solar cells group under the same bypass diode.*
- class SolarCell

  *Represents a PV cell, the most basic element of a solar generator.*
- class SolarPanel

  *Represents a solar panel of a PV generator.*
- class SolarSolver
- struct TotalsOfCellsGroup

  *Defines the total parameters of a group of PV cells in the same string that have the same short-circuit current.*

### Functions

- bool equalIsc (const SameIshortcircuitGroup &iVC1, const SameIshortcircuitGroup &iVC2)
- Col$<$ double $>$ loadInitialValues (solar_string $*$st, int n, int nS)
- bool operator$<$ (const TotalsOfCellsGroup &T1, const TotalsOfCellsGroup &T2)
- bool operator== (const TotalsOfCellsGroup &T1, const TotalsOfCellsGroup &T2)
- bool SameCurrent (TotalsOfCellsGroup first, TotalsOfCellsGroup second)

## Variables

- constexpr double BREAKDOWN_ALPHA_REF {0.002}

  *Breakdown alpha parameter.*
- constexpr double VOLTAGE_BREAKDOWN_REF {-15.0}

  *Breakdown voltage [V].*
- constexpr double CURRENT_REVERSE_SATURATION_REF {1.26E-9}

  *Reverse saturation current [A].*
- constexpr double CURRENT_PHOTOGENERATED_REF {3.798}

  *Photogenerated current of reference, in case it is not specified [A].*
- constexpr double CURRENT_SHORTCIRCUIT_REF {3.798}

  *Short-circuit current of reference, in case it is not specified [A].*
- constexpr double VOLTAGE_OPEN_CIRCUIT_REF {0.9}

  *Open circuit voltage of reference, in case it is not specified [A].*
- constexpr double BOLTZMANN_CONST {1.38e-23}

  *Boltzmann constant [J/ºK].*
- constexpr double TEMPERATURE_CELL_REF {25.0}

  *Temperature of the cell of reference, in case it is not specified [ºC].*
- constexpr double IRRADIANCE_REF {1000}

  *Irradiance of reference, in case it is not specified [W/m2].*
- constexpr double SOILING_FACTOR_REF {1}

  *Soiling Factor, 1 by default.*
- constexpr double IDEALITY_FACTOR_REF {1.5}

  *Ideality factor.*
- constexpr double RESISTANCE_SERIES_REF {0.00895}

  *Total resistance of the cell in series.*
- constexpr double RESISTANCE_SHUNT_REF {30.0}

  *Total shunt resistance of the cell.*
- constexpr double ELECTRONS_CHARGE {1.602e-19}

  *Charge of an electron [C].*
- constexpr double TEMPERATURE_COEFF_REF {0.0004}

  *Temperature coefficient. Depends on the material, but this one belongs to silicon [A/ºC].*
- constexpr double VOLTAGE_TEMPERATURE_COEFF_REF {-0.0023}

  *Voltage temperature coefficient. Depends on the material, but this one belongs to silicon [V/ºC].*
- constexpr double BREAKDOWN_EXPONENT_REF {3.0}

  *Breakdown exponent.*

## 2.1.1 Function Documentation

### 2.1.1.1 equalIsc()

```
bool stringarma::equalIsc (
            const SameIshortcircuitGroup & iVC1,
            const SameIshortcircuitGroup & iVC2 )
```

#### 2.1.1.2 loadInitialValues()

```
Col<double> stringarma::loadInitialValues (
            solar_string * st,
            int n,
            int nS )
```

#### 2.1.1.3 operator<()

```
bool stringarma::operator< (
            const TotalsOfCellsGroup & T1,
            const TotalsOfCellsGroup & T2 )
```

#### 2.1.1.4 operator==()

```
bool stringarma::operator== (
            const TotalsOfCellsGroup & T1,
            const TotalsOfCellsGroup & T2 )
```

#### 2.1.1.5 SameCurrent()

```
bool stringarma::SameCurrent (
            TotalsOfCellsGroup first,
            TotalsOfCellsGroup second )
```

### 2.1.2 Variable Documentation

#### 2.1.2.1 BOLTZMANN_CONST

```
constexpr double stringarma::BOLTZMANN_CONST {1.38e-23}  [constexpr]
```

Boltzmann constant [J/ºK].

#### 2.1.2.2 BREAKDOWN_ALPHA_REF

```
constexpr double stringarma::BREAKDOWN_ALPHA_REF {0.002}  [constexpr]
```

Breakdown alpha parameter.

#### 2.1.2.3 BREAKDOWN_EXPONENT_REF

```
constexpr double stringarma::BREAKDOWN_EXPONENT_REF {3.0}  [constexpr]
```

Breakdown exponent.

#### 2.1.2.4 CURRENT_PHOTOGENERATED_REF

```
constexpr double stringarma::CURRENT_PHOTOGENERATED_REF {3.798}  [constexpr]
```

Photogenerated current of reference, in case it is not specified [A].

#### 2.1.2.5 CURRENT_REVERSE_SATURATION_REF

```
constexpr double stringarma::CURRENT_REVERSE_SATURATION_REF {1.26E-9}  [constexpr]
```

Reverse saturation current [A].

#### 2.1.2.6 CURRENT_SHORTCIRCUIT_REF

```
constexpr double stringarma::CURRENT_SHORTCIRCUIT_REF {3.798}  [constexpr]
```

Short-circuit current of reference, in case it is not specified [A].

#### 2.1.2.7 ELECTRONS_CHARGE

```
constexpr double stringarma::ELECTRONS_CHARGE {1.602e-19}  [constexpr]
```

Charge of an electron [C].

#### 2.1.2.8 IDEALITY_FACTOR_REF

```
constexpr double stringarma::IDEALITY_FACTOR_REF {1.5}  [constexpr]
```

Ideality factor.

#### 2.1.2.9 IRRADIANCE_REF

```
constexpr double stringarma::IRRADIANCE_REF {1000}  [constexpr]
```

Irradiance of reference, in case it is not specified [W/m2].

#### 2.1.2.10 RESISTANCE_SERIES_REF

```
constexpr double stringarma::RESISTANCE_SERIES_REF {0.00895}  [constexpr]
```

Total resistance of the cell in series.

#### 2.1.2.11 RESISTANCE_SHUNT_REF

```
constexpr double stringarma::RESISTANCE_SHUNT_REF {30.0}  [constexpr]
```

Total shunt resistance of the cell.

#### 2.1.2.12 SOILING_FACTOR_REF

```
constexpr double stringarma::SOILING_FACTOR_REF {1}  [constexpr]
```

Soiling Factor, 1 by default.

#### 2.1.2.13 TEMPERATURE_CELL_REF

```
constexpr double stringarma::TEMPERATURE_CELL_REF {25.0}  [constexpr]
```

Temperature of the cell of reference, in case it is not specified [ºC].

#### 2.1.2.14 TEMPERATURE_COEFF_REF

```
constexpr double stringarma::TEMPERATURE_COEFF_REF {0.0004}  [constexpr]
```

Temperature coefficient. Depends on the material, but this one belongs to silicon [A/ºC].

#### 2.1.2.15 VOLTAGE_BREAKDOWN_REF

```
constexpr double stringarma::VOLTAGE_BREAKDOWN_REF {-15.0} [constexpr]
```

Breakdown voltage [V].

#### 2.1.2.16 VOLTAGE_OPEN_CIRCUIT_REF

```
constexpr double stringarma::VOLTAGE_OPEN_CIRCUIT_REF {0.9} [constexpr]
```

Open circuit voltage of reference, in case it is not specified [A].

#### 2.1.2.17 VOLTAGE_TEMPERATURE_COEFF_REF

```
constexpr double stringarma::VOLTAGE_TEMPERATURE_COEFF_REF {-0.0023} [constexpr]
```

Voltage temperature coefficient. Depends on the material, but this one belongs to silicon [V/ºC].

# Chapter 3

# Class Documentation

## 3.1 stringarma::BypassDiode Class Reference

Represents a common component of a photovoltaic generator, a bypass diode.

`#include <pv_diode.h>`

### Public Member Functions

- BypassDiode (void)

  *Constructor of the class bypass_diode.*
- void setTemperatureDiode (double)

  *Set a double value for the Temperature of the diode [ºC].*
- void setCurrentReverseSaturation (void)

  *Updates the value for the reverse saturation current [A] according to the current value of the temperature of the diode Tc.*
- void setCurrentDiode (double)

  *Updates the value for the current in the diode [A].*
- void setIdealityFactor (double)

  *Updates the value od the ideality factor of the bypass diode.*
- double getCurrentReverseSaturation (void)

  *Gets the diode's reverse saturation current [A].*
- double getTemperatureDiode (void)

  *Gets the diode's temperature [ºC].*
- double getCurrentDiode (void)

  *Gets the diode's current [A].*
- double getIdealityFactor (void)

  *Gets the ideality factor applied in the bypass diode.*
- double calcFunctionD (double)

  *Calculates the fd function described in the bypass_ch part of the mainPage.*
- double calcFuntionDiodeDerivativeRespectVoltage (double)

  *Calculates the partial derivative respect the voltage of the diode, Vd, of the fd function described in the math part of the mainPage.*

## Protected Attributes

- double current_reverse_saturation

    *Reverse saturation current [A].*
- double temperature_diode

    *Current temperature of the bypass diode [ºC].*
- double current_diode

    *Current through the diode [A].*
- double ideality_factor

    *Ideality factor.*

### 3.1.1 Detailed Description

Represents a common component of a photovoltaic generator, a bypass diode.

These diodes are placed in anti-parallel configuration with one or more cells. The cells grouped by the same bypass diode are considered strings. This class contains all the parameters that define a diode and to perform the calculations needed.

When the class is created, it contains the following reference values:

- **Irref** The reverse saturation current is 5e-6 A.
- **Tcref** The temperature of the cell equals 25.0 ºC.
- **m_ref** Ideality factor of 1.5.

Other mathematical constants used in the calculations are:

- **k** Boltzmann constant: 1.38e-23 J/ºK
- **q** Charge of an electron: 1.602e-19 C

**See also**

SolarCell

SolarString

**Note**

The theoretical concepts behind this class are explained in the bypass_ch section of the mainPage.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 BypassDiode()

```
stringarma::BypassDiode::BypassDiode (
            void )
```

Constructor of the class bypass_diode.

Uses all the reference values.

### 3.1.3 Member Function Documentation

#### 3.1.3.1 calcFunctionD()

```
double stringarma::BypassDiode::calcFunctionD (
            double Vdiode )
```

Calculates the fd function described in the bypass_ch part of the mainPage.

**Parameters**

| | |
|---|---|
| *Double* | value of the diode's voltage Vd [V]. |

**Returns**

A double type with the value of the funtion fd [A].

#### 3.1.3.2 calcFuntionDiodeDerivativeRespectVoltage()

```
double stringarma::BypassDiode::calcFuntionDiodeDerivativeRespectVoltage (
            double Vd )
```

Calculates the partial derivative respect the voltage of the diode, Vd, of the fd function described in the math part of the mainPage.

**Parameters**

| | |
|---|---|
| *Double* | value of the diode's voltage Vd [V]. |

**Returns**

A double type with the value of the partial derivative respect the voltage of the diode of the funtion fd.

**See also**

math

#### 3.1.3.3 getCurrentDiode()

```
double stringarma::BypassDiode::getCurrentDiode (
            void )
```

Gets the diode's current [A].

**Returns**

A double type with the value of the current [A].

#### 3.1.3.4 getCurrentReverseSaturation()

```
double stringarma::BypassDiode::getCurrentReverseSaturation (
            void )
```

Gets the diode's reverse saturation current [A].

**Returns**

A double type with the value of the reverse saturation current [A].

#### 3.1.3.5 getIdealityFactor()

```
double stringarma::BypassDiode::getIdealityFactor (
            void )
```

Gets the ideality factor applied in the bypass diode.

**Returns**

Ideality factor.

#### 3.1.3.6 getTemperatureDiode()

```
double stringarma::BypassDiode::getTemperatureDiode (
            void )
```

Gets the diode's temperature [ºC].

**Returns**

A double type with the value of the temperature [ºC].

#### 3.1.3.7 setCurrentDiode()

```
void stringarma::BypassDiode::setCurrentDiode (
            double _Id )
```

Updates the value for the current in the diode [A].

**Parameters**

| | |
|---|---|
| *Id* | Double value for the diode's current [A]. |

#### 3.1.3.8 setCurrentReverseSaturation()

```
void stringarma::BypassDiode::setCurrentReverseSaturation (
            void )
```

Updates the value for the reverse saturation current [A] according to the current value of the temperature of the diode Tc.

#### 3.1.3.9 setIdealityFactor()

```
void stringarma::BypassDiode::setIdealityFactor (
            double _n )
```

Updates the value od the ideality factor of the bypass diode.

**Parameters**

| | |
|---|---|
| *m* | Ideality factor. |

#### 3.1.3.10 setTemperatureDiode()

```
void stringarma::BypassDiode::setTemperatureDiode (
            double _Td )
```

Set a double value for the Temperature of the diode [ºC].

**Parameters**

| | |
|---|---|
| *Double* | value for the temperature of the diode [ºC]. |

### 3.1.4 Member Data Documentation

#### 3.1.4.1 current_diode

`double stringarma::BypassDiode::current_diode [protected]`

Current through the diode [A].

#### 3.1.4.2 current_reverse_saturation

`double stringarma::BypassDiode::current_reverse_saturation [protected]`

Reverse saturation current [A].

#### 3.1.4.3 ideality_factor

`double stringarma::BypassDiode::ideality_factor [protected]`

Ideality factor.

#### 3.1.4.4 temperature_diode

`double stringarma::BypassDiode::temperature_diode [protected]`

Current temperature of the bypass diode [ºC].

## 3.2 stringarma::CellsGroup Struct Reference

Structure to gather global information of a group of cells that share, at least, the same short-circuit current.

`#include <pv_solver.h>`

### Public Attributes

- double current_shortcircuit

    *Short-circuit current of all the cells in the group.*
- double sum_voltage_breakdown_in_group

    *Breakdown voltage of the group.*
- double sum_voltage_open_circuit_all_cells

    *Sum of all the open circuit voltage of the active cells.*
- double sum_voltage_open_circuit_non_active_cells

    *Sum of all the open circuit voltage of the non-active cells.*
- int group_size

    *Number of cells in this group.*
- double limit_voltage

    *Voltage between the terminals of the panel where either a change in the distribution of the tensions or currents in the panel will take place.*

### 3.2.1 Detailed Description

Structure to gather global information of a group of cells that share, at least, the same short-circuit current.

### 3.2.2 Member Data Documentation

#### 3.2.2.1 current_shortcircuit

```
double stringarma::CellsGroup::current_shortcircuit
```

Short-circuit current of all the cells in the group.

#### 3.2.2.2 group_size

```
int stringarma::CellsGroup::group_size
```

Number of cells in this group.

#### 3.2.2.3 limit_voltage

```
double stringarma::CellsGroup::limit_voltage
```

Voltage between the terminals of the panel where either a change in the distribution of the tensions or currents in the panel will take place.

#### 3.2.2.4 sum_voltage_breakdown_in_group

```
double stringarma::CellsGroup::sum_voltage_breakdown_in_group
```

Breakdown voltage of the group.

Calculated by adding all the breakdown voltages calculated in the group Vbrx of every cell.

**See also**

Definition of Vbrx in the Theoretical documentation.

#### 3.2.2.5 sum_voltage_open_circuit_all_cells

```
double stringarma::CellsGroup::sum_voltage_open_circuit_all_cells
```

Sum of all the open circuit voltage of the active cells.

Sum of all the open circuit voltage of the cells in the group that are actually driving its short-circuit current (active cells) [V].

#### 3.2.2.6 sum_voltage_open_circuit_non_active_cells

```
double stringarma::CellsGroup::sum_voltage_open_circuit_non_active_cells
```

Sum of all the open circuit voltage of the non-active cells.

Sum of all the open circuit voltage of the cells in the group that are not driving its short-circuit current (non-active cells), but an inferior current [V].

## 3.3 stringarma::Classcomp Struct Reference

Comparison function object for the multimap.

```
#include <pv_solver.h>
```

### Public Member Functions

- bool operator() (const std::pair< double, double > &k1, const std::pair< double, double > &k2)

### 3.3.1 Detailed Description

Comparison function object for the multimap.

Compares the Isc of both groups. In case they are equal, compares the SVbr of the groups.

**Returns**

True in case the Isc of the first element is lower than the first one. If they are equal, returns TRUE if the first element has higher SVbr.

### 3.3.2 Member Function Documentation

#### 3.3.2.1 operator()()

```
bool stringarma::Classcomp::operator() (
            const std::pair< double, double > & k1,
            const std::pair< double, double > & k2 )  [inline]
```

## 3.4 stringarma::SameIshortcircuitAndVbreakdownGroup Struct Reference

Structure with info of groups of cells with the same Isc and Vbrx.

`#include <pv_solver.h>`

### Public Attributes

- CellsGroup sum_same_i_shortcircuit_and_v_breakdown_group

  *Global information of the group of cells with the same Isc and Vbrx.*
- std::list< std::pair< int, CellsGroup > > detailed_same_i_shortcircuit_and_v_breakdown_group

  *Detailed information of the group of cells with the same Isc and Vbrx.*

### 3.4.1 Detailed Description

Structure with info of groups of cells with the same Isc and Vbrx.

Contains global information (inMapSum) and detailed information (inMapDetails) about groups of cells with the same short-circuit current and breakdown voltage.

**See also**

inVectorCell

### 3.4.2 Member Data Documentation

#### 3.4.2.1 detailed_same_i_shortcircuit_and_v_breakdown_group

```
std::list<std::pair<int,CellsGroup> > stringarma::SameIshortcircuitAndVbreakdownGroup::detailed↩
_same_i_shortcircuit_and_v_breakdown_group
```

Detailed information of the group of cells with the same Isc and Vbrx.

The cells in the group are split in smaller groups that share the same string. Every entry in the list contains a pair with an integer corresponding to the index of the string and a grupString structure with the grouped info of this group.

#### 3.4.2.2 sum_same_i_shortcircuit_and_v_breakdown_group

```
CellsGroup stringarma::SameIshortcircuitAndVbreakdownGroup::sum_same_i_shortcircuit_and_v_↩
breakdown_group
```

Global information of the group of cells with the same Isc and Vbrx.

Contains the short-circuit current of the group and the total sum of certain parameters. The grupString's Limit attribute stored in this structure refers to the internal limits.

These "internal" limits are the total voltage in the panel needed to get every bypass diode in conducting state. In case there's no diode, the limit will match the lower external limit. The internal limits represent a change in the distribution of the total voltage.

**See also**

grupString

## 3.5 stringarma::SameIshortcircuitGroup Struct Reference

Vector meant to contain info of groups of cells with the same short-circuit current (Isc).

```
#include <pv_solver.h>
```

### Public Attributes

- CellsGroup sum_same_i_shortcircuit_group

  *Global information of the group of cells with the same short-circuit current.*
- std::map< double, SameIshortcircuitAndVbreakdownGroup > detailed_same_i_shortcircuit_group

  *Detailed information of the group of cells with the same short-circuit current.*

### 3.5.1 Detailed Description

Vector meant to contain info of groups of cells with the same short-circuit current (Isc).

Contains global information (inVectorSum) and detailed information (inVectorDetail). Global information refers to the sum of certain parameters. Detailed information distinguish smaller groups that share the same Isc and Vbrx.

**See also**

inMapCell

### 3.5.2 Member Data Documentation

#### 3.5.2.1 detailed_same_i_shortcircuit_group

```
std::map<double,SameIshortcircuitAndVbreakdownGroup> stringarma::SameIshortcircuitGroup↩
::detailed_same_i_shortcircuit_group
```

Detailed information of the group of cells with the same short-circuit current.

The cells in the group are split in smaller groups that share the same breakdown voltage calculated in the group Vbrx. Every entry in the map is composed by key, which is a double corresponding to Vbrx, and a inMapCell structure with the information of this reduced group.

#### 3.5.2.2 sum_same_i_shortcircuit_group

```
CellsGroup stringarma::SameIshortcircuitGroup::sum_same_i_shortcircuit_group
```

Global information of the group of cells with the same short-circuit current.

Contains the short-circuit current of the group and the total sum of certain parameters. The grupString's Limit attribute stored in this structure refers to the external limits. These "external" limits are the total voltage in the panel needed to get every cell into breakdown. The external limits represent a change in the total current.

**See also**

grupString

## 3.6 stringarma::solar_string Class Reference

Represents a string of solar cells group under the same bypass diode.

```
#include <pv_string.h>
```

### Public Member Functions

- solar_string (void)

  *Constructor of the class solar_string.*
- ∼solar_string (void)

  *Destructor of the class solar_string.*
- int getWithDiode (void)

  *Indicates whether the string of PV cells has a by-pass diode or not.*
- double getMinimCurrentShortcircuit (void)

  *Gets the minimum short-circuit current in the string [A].*
- double getSumVoltageOpenCircuit (void)

  *Gets the sum of all the open circuit voltage of the cells in the string [V].*
- double getSumVoltageBreakdown (void)

  *Gets the sum of the breakdown voltage of all the cells in the string [V].*
- double getVoltageString (void)

  *Gets the voltage between the terminals of the string [V].*
- double getSumVoltageAllCells (void)

  *Gets the sum of the voltage between the terminals of every cell in the string [V].*
- double getVoltageDiode (void)

  *Gets the voltage between the terminals of the bypass diode [V].*
- void setSumVoltageOpenCircuit (void)

  *Updates the Svoc attribute with the current value of Voc of every cell.*
- void setSumVoltageBreakdown (void)

  *Updates the Svbreak attribute with the current value of Vbreak of every cell.*
- void setVoltageString (double)

  *Set a new value for the voltage between the terminals of the string.*
- void setSumVoltageAllCells (void)

  *Updates the value of the sum of the voltage between the terminals of every cell in the string (Svcell) with its current value.*
- void setSumVolageBreakdownInGroup (void)

  *Updates the value of the sum of the breakdown voltage (SVbrx) of every group of cells in CellsGr.*
- void setVoltageDiode (double)

  *Set a new value for the knee voltage of the bypass diode of the string.*
- void updateStringsData (std::pair< bool, std::vector< std::pair< double, double >>> &, SolarCell &)

  *Fills the array cells_array with cells like the one provided as parameter and update them.*
- void findInitialState (double &, double &)

  *Approximates the initial values for the iterative method.*

## Public Attributes

- SolarCell * cells_array
  *Array of solar_cell objects.*
- BypassDiode diode_bypass
  *bypass_diode object.*
- std::list< TotalsOfCellsGroup > groupsByCurrentShortcircuit
  *List that contains all the info about the different groups of cells in the string that share the same short-circuit current Isc.*
- int string_size
  *Number of cells contained in the string.*

## Friends

- class SolarSolver

### 3.6.1 Detailed Description

Represents a string of solar cells group under the same bypass diode.

However, the diode can be missing, broken or non-active. This class contains all the details of the components of the string. The cells in the string are divided in groups according to their short-circuit current Isc. These groups are distinguished:

- **Active cells groups**: Cells working under their own short-circuit current Isc.
- **Non-active cells groups**: Cells working under a different current from their Isc (a lower value).
- **Breakdown cells groups**: Cells working in the breakdown zone of the cell. Therefore working under a different current from their Isc (a higher value).

Given the total current and voltage between the terminals of the string, this class can find an initial estimation of the state of every component. The values used are the following:

- **Non-active cells**: Current is imposed by the rest of the panel or an active group in the string. Working voltage is its open circuit voltage.
- **Breakdown cells**: Current is imposed by the rest of the panel or an active group in the string. Working voltage is its breakdown voltage.
- **Active cells**: Current is its short-circuit current. Voltage is deducted from the total voltage in the string, the diode's voltage and voltage in the rest of the groups.

**See also**

SolarCell

BypassDiode

**Note**

The theoretical concepts behind this class are explained in the string_ch section of the mainPage.

## 3.6.2 Constructor & Destructor Documentation

#### 3.6.2.1 solar_string()

```
stringarma::solar_string::solar_string (
            void )
```

Constructor of the class solar_string.

Uses all the reference values for the attributes.

#### 3.6.2.2 ∼solar_string()

```
stringarma::solar_string::∼solar_string (
            void )
```

Destructor of the class solar_string.

## 3.6.3 Member Function Documentation

#### 3.6.3.1 findInitialState()

```
void stringarma::solar_string::findInitialState (
            double & Iin,
            double & Vin )
```

Approximates the initial values for the iterative method.

Given the total current and voltage between terminals in the string, assigns the electrical working point of every component in the string. This function first finds the state of the bypass diode and then the state and electrical conditions of every component. No value is returned, the changes are done in the solar_cell and bypass_diode objects contained by the string object.

**Parameters**

| | |
|---|---|
| *Iin* | Total current through the string [A]. |
| *Vin* | Total potential difference between terminals of the string [V]. |

#### 3.6.3.2 getMinimCurrentShortcircuit()

```
double stringarma::solar_string::getMinimCurrentShortcircuit (
            void )
```

Gets the minimum short-circuit current in the string [A].

**Returns**

Double data type with the value of the minimum short-circuit current [A].

#### 3.6.3.3 getSumVoltageAllCells()

```
double stringarma::solar_string::getSumVoltageAllCells (
            void )
```

Gets the sum of the voltage between the terminals of every cell in the string [V].

This value can be different than the obtained with getVstring.

**Returns**

Double data type with the value of the sum of the voltages in every cell [V].

#### 3.6.3.4 getSumVoltageBreakdown()

```
double stringarma::solar_string::getSumVoltageBreakdown (
            void )
```

Gets the sum of the breakdown voltage of all the cells in the string [V].

**Returns**

Double data type with the value of the sum of breakdown voltages [V].

#### 3.6.3.5 getSumVoltageOpenCircuit()

```
double stringarma::solar_string::getSumVoltageOpenCircuit (
            void )
```

Gets the sum of all the open circuit voltage of the cells in the string [V].

**Returns**

Double data type with the value of the sum of open circuit voltages [V].

#### 3.6.3.6 getVoltageDiode()

```
double stringarma::solar_string::getVoltageDiode (
            void )
```

Gets the voltage between the terminals of the bypass diode [V].

**Returns**

Double data type value of the voltage in the bypass diode [V].

#### 3.6.3.7 getVoltageString()

```
double stringarma::solar_string::getVoltageString (
            void )
```

Gets the voltage between the terminals of the string [V].

**Returns**

Double data type with the value of the voltage in the string [V].

#### 3.6.3.8 getWithDiode()

```
int stringarma::solar_string::getWithDiode (
            void )
```

Indicates whether the string of PV cells has a by-pass diode or not.

By default it is 1.

**Returns**

An integer data type. 1 indicates that there is a diode, 0 indicates that there is not.

#### 3.6.3.9 setSumVolageBreakdownInGroup()

```
void stringarma::solar_string::setSumVolageBreakdownInGroup (
            void )
```

Updates the value of the sum of the breakdown voltage (SVbrx) of every group of cells in CellsGr.

If there is no bypass diode in the string, it is equal to the sum of the breakdown voltage between the terminals of every cell. If there is a bypass diode, it is obtained as explained in the theorical documentation.

#### 3.6.3.10 setSumVoltageAllCells()

```
void stringarma::solar_string::setSumVoltageAllCells (
            void )
```

Updates the value of the sum of the voltage between the terminals of every cell in the string (Svcell) with its current value.

It does the sum again. In case any value of any cell has changed.

#### 3.6.3.11 setSumVoltageBreakdown()

```
void stringarma::solar_string::setSumVoltageBreakdown (
            void )
```

Updates the Svbreak attribute with the current value of Vbreak of every cell.

#### 3.6.3.12 setSumVoltageOpenCircuit()

```
void stringarma::solar_string::setSumVoltageOpenCircuit (
            void )
```

Updates the Svoc attribute with the current value of Voc of every cell.

#### 3.6.3.13 setVoltageDiode()

```
void stringarma::solar_string::setVoltageDiode (
            double _Vdiode )
```

Set a new value for the knee voltage of the bypass diode of the string.

**Parameters**

| | |
|---|---|
| *Vdiode* | New knee voltage of the bypass diode of the string [V]. |

#### 3.6.3.14 setVoltageString()

```
void stringarma::solar_string::setVoltageString (
            double _Vstring )
```

Set a new value for the voltage between the terminals of the string.

**Parameters**

| | |
|---|---|
| *Vstring* | New voltage between the terminals of the string [V]. |

### 3.6.3.15 updateStringsData()

```
void stringarma::solar_string::updateStringsData (
            std::pair< bool, std::vector< std::pair< double, double >>> & string_input,
            SolarCell & sc )
```

Fills the array cells_array with cells like the one provided as parameter and update them.

The cells are set with the proper values of Irradiance and Temperature. Their electrical parameters are updated according to these values. Then sorts and groups the cells according to their short-circuit current.

**Parameters**

| | |
|---|---|
| *sc* | solar_cell object that represents all the cells in the string. |
| *string_input* | The first value of the pair is the state of the bypass diode. Every element in the vector represents a cell, and the pair of doubles its values of irradiance and temperature. |

**See also**

solarCell_ch

## 3.6.4 Friends And Related Function Documentation

### 3.6.4.1 SolarSolver

```
friend class SolarSolver [friend]
```

## 3.6.5 Member Data Documentation

### 3.6.5.1 cells_array

```
SolarCell* stringarma::solar_string::cells_array
```

Array of solar_cell objects.

This is a representation of the PV cells contained in this string. The cells in this array must have the same manufacturing properties, but the electrical or physical working values may differ.

**See also**

SolarCell

#### 3.6.5.2 diode_bypass

```
BypassDiode stringarma::solar_string::diode_bypass
```

bypass_diode object.

Represents the bypass diode of the string.

#### 3.6.5.3 groupsByCurrentShortcircuit

```
std::list<TotalsOfCellsGroup> stringarma::solar_string::groupsByCurrentShortcircuit
```

List that contains all the info about the different groups of cells in the string that share the same short-circuit current Isc.

Every element in the list is a TableStr struct with the info of the group of cells.

**See also**

TotalsOfCellsGroup structure.

#### 3.6.5.4 string_size

```
int stringarma::solar_string::string_size
```

Number of cells contained in the string.

## 3.7 stringarma::SolarCell Class Reference

Represents a PV cell, the most basic element of a solar generator.

```
#include <pv_cell.h>
```

## Public Member Functions

- SolarCell (void)

  *Constructor of the class solar_cell.*
- SolarCell (const SolarCell &)

  *Constructor of the class solar_cell.*
- void setIndex (int)

  *Set an integer value for the index.*
- void setIrradiance (double)

  *Set a double value for the irradiance [W/m2].*
- void setTemperatureCell (double)

  *Set a double value for the temperature of the cell [ºC].*
- void setCurrentReverseSaturation (void)

  *Updates the value for the reverse saturation current [A] according to the current value of the temperature of the cell Tc.*
- void setCurrentShortcircuit (void)

  *Updates the value for the short-circuit current [A] according to the current values of the temperature of the cell Tc and the irradiance G.*
- void setCurrentPhotogenerated (void)

  *Updates the value for the photogenerated current [A] according to the current values of the temperature of the cell Tc and the irradiance G.*
- void setVoltageOpenCircuit (void)

  *Updates the value for the open circuit voltage [V] according to the current values of the temperature of the cell Tc and the irradiance G.*
- void setCurrentCell (double)

  *Set a double value for the current [A].*
- void setVoltageCell (double)

  *Set a double value for the voltage [V].*
- void setVoltageBreakdown (double)

  *Set a double value for the breakdown voltage [V].*
- void setBreakdownAlpha (double)

  *Set a double value for the alpha parameter.*
- void setSoilingFactor (double)

  *Sets the soiling factor.*
- void setIdealityFactor (double)

  *Sets the ideality factor.*
- void setResistanceSeries (double)

  *Sets the total resistance of the cell in series.*
- void setResistanceShunt (double)

  *Sets the total shunt resistance of the cell.*
- void setTemperatureCoeff (double)

  *Sets the temperature coefficient.*
- void setVoltageTemperatureCoeff (double)

  *Sets the voltage temperature coefficient.*
- void setBreakdownExponent (double)

  *Sets the breakdown exponent.*
- int getIndex (void)

  *Gets the cell's index.*
- double getIrradiance (void)

  *Gets the irradiance [W/m2].*
- double getTemperatureCell (void)

  *Gets the temperature of the cell [ºC].*

- double getCurrentReverseSaturation (void)

    *Gets the reverse saturation current [A].*
- double getCurrentShortcircuit (void)

    *Gets the short-circuit current [A].*
- double getCurrentPhotogenerated (void)

    *Gets the photogenerated current [A].*
- double getVoltageOpenCircuit (void)

    *Gets the open circuit voltage [V].*
- double getCurrentCell (void)

    *Gets the cell's current [A].*
- double getVoltageCell (void)

    *Gets the cell's voltage [V].*
- double getVoltageBreakdown (void)

    *Gets the breakdown voltage [V].*
- double getBreakdownAlpha (void)

    *Gets the alpha parameter.*
- double getSoilingFactor (void)

    *Gets the soiling factor.*
- double getIdealityFactor (void)

    *Gets the ideality factor.*
- double getResistanceSeries (void)

    *Gets the total resistance of the cell in series.*
- double getResistanceShunt (void)

    *Gets the total shunt resistance of the cell.*
- double getTemperatureCoeff (void)

    *Gets the temperature coefficient.*
- double getVoltageTemperatureCoeff (void)

    *Gets the voltage temperature coefficient.*
- double getBreakdownExponent (void)

    *Gets the breakdown exponent.*
- double calcFunctionC (void)

    *Calculates the fc function described in the math part of the mainPage.*
- double calcFunctionCellDerivativeRespectCurrent (void)

    *Calculates the partial derivative respect the current of the cell, Icell, of the fc function described in the math part of the mainPage.*
- double calcFunctionCellDerivativeRespectVoltage (void)

    *Calculates the partial derivative respect the voltage of the cell, Vcell, of the fc function described in the math part of the mainPage.*

## Protected Attributes

- int index

    *Index of the cell. Serves as an identifier (ID) of the cell once it is grouped inside a string.*
- double current_photogenerated

    *Photogenerated current [A].*
- double current_reverse_saturation

    *Reverse saturation current [A].*
- double current_shortcircuit

    *Shortcircuit current [A].*
- double voltage_open_circuit

*Open circuit voltage [V].*
- double temperature_cell

  *Temperature of the cell [ºC].*
- double irradiance

  *Irradiance [W/m2].*
- double current_cell

  *Current through the cell [A].*
- double voltage_cell

  *Voltage between the terminals of the cell [V].*
- double voltage_breakdown

  *Breakdown voltage [V].*
- double breakdown_alpha

  *Breakdown alpha.*
- double soiling_factor

  *Soiling factor.*
- double ideality_factor

  *Ideality factor.*
- double resistance_series

  *Total resistance of the cell in series.*
- double resistance_shunt

  *Total shunt resistance of the cell.*
- double temperature_coeff

  *Temperature coefficient.*
- double voltage_temperature_coeff

  *Voltage temperature coefficient.*
- double breakdown_exponent

  *Breakdown exponent.*

### 3.7.1 Detailed Description

Represents a PV cell, the most basic element of a solar generator.

This class contains all the parameters that define a single PV cell and to perform the calculations needed. Only the operational parameters of a PV cell are considered as attributes of this class. For intrinsic parameters of a PV cell, such as those that depend on the PV cell's material, they are implemented as constant and can not be edited. These values correspond to a silicon, multicrystalline PV cell.

The editable attributes of this class are:

- **index** Identifier of the cell.
- **Iph** Photogenerated current [A].
- **Io** Reverse saturation current [A].
- **Isc** Short-circuit current [A].
- **Voc** Open circuit voltage [V].
- **Tc** Temperature of the cell [ºC].
- **G** Irradiance [W/m2].
- **Icell** Current [A].

- **Vcell** Voltage [V].
- **Vbreak** Breakdown voltage [V].

When it is not specified in the constructor of the class some attributes are initialized with reference values. These reference values are:

- **Ioref** The reverse saturation current of reference is 1.26E-9 A.
- **Iphref** The photogenerated current of reference is 3.798 A.
- **Iscref** The short-circuit current of reference is 3.798 A.
- **Vocref** The open circuit voltage of reference is 0.9 V.
- **Tcref** The temperature of the cell of reference is 25.0 ºC.
- **Gref** The irradiance of reference is 1000 W/m2.

The mathematical models of this library use some constants or approximations. These parameters can be set by the user. The values of reference used related to this class are:

- $\alpha$ Breakdown alpha parameter: 0.002
- **Vbr** Breakdown voltage: -15.0 V
- **k** Boltzmann constant: 1.38e-23 J/ºK
- **SF** Soiling Factor: 1
- **Rs** Total resistance of the cell in series: 0.00895
- **Rsh** Total shunt resistance of the cell: 30.0
- **q** Charge of an electron: 1.602e-19 C
- **a** Temperature coefficient: 0.0004 A/ºC
- **B** Voltage temperature coefficient: -0.0023 V/ºC
- **m** Breakdown exponent: 3

**See also**

SolarString

**Note**

The theoretical concepts behind this class are explained in the solarCell_ch section of the mainPage.

**Warning**

This library contemplates the calculations of solar panels under mismatched conditions where irradiance (G) and temperature of the cell Tc are different across the facility. Scenarios where the cells that compose the panels have different intern parameters are NOT in the scope of this library and will not compute.

### 3.7.2 Constructor & Destructor Documentation

#### 3.7.2.1 SolarCell() [1/2]

```
stringarma::SolarCell::SolarCell (
              void )
```

Constructor of the class solar_cell.

Uses all the reference values for the attributes.

#### 3.7.2.2 SolarCell() [2/2]

```
stringarma::SolarCell::SolarCell (
              const SolarCell & cell )
```

Constructor of the class solar_cell.

Uses the same attributes as the solar_cell object introduced as a parameter.

**Parameters**

| | |
|---|---|
| *solar_cell* | object to copy the attributes from. |

### 3.7.3 Member Function Documentation

#### 3.7.3.1 calcFunctionC()

```
double stringarma::SolarCell::calcFunctionC (
              void )
```

Calculates the fc function described in the math part of the mainPage.

The current values of Vcell and Icell are used to calculate this function.

**Returns**

A double type with the value of the funtion fc.

**See also**

math

#### 3.7.3.2 calcFunctionCellDerivativeRespectCurrent()

```
double stringarma::SolarCell::calcFunctionCellDerivativeRespectCurrent (
            void )
```

Calculates the partial derivative respect the current of the cell, Icell, of the fc function described in the math part of the mainPage.

It is used to build the jacobian matrix explained in the math_newton_part2 section. The current values of Vcell and Icell are used to calculate this function.

**Returns**

A double type with the value of the partial derivative respect the current of the cell of the funtion fc.

**See also**

math

#### 3.7.3.3 calcFunctionCellDerivativeRespectVoltage()

```
double stringarma::SolarCell::calcFunctionCellDerivativeRespectVoltage (
            void )
```

Calculates the partial derivative respect the voltage of the cell, Vcell, of the fc function described in the math part of the mainPage.

It is used to build the jacobian matrix explained in the math_newton_part2 section. The current values of Vcell and Icell are used to calculate this function.

**Returns**

A double type with the value of the partial derivative respect the voltage of the cell of the funtion fc.

**See also**

math

#### 3.7.3.4 getBreakdownAlpha()

```
double stringarma::SolarCell::getBreakdownAlpha (
            void )
```

Gets the alpha parameter.

**Returns**

A double type with the value of alpha parameter.

#### 3.7.3.5 getBreakdownExponent()

```
double stringarma::SolarCell::getBreakdownExponent (
            void )
```

Gets the breakdown exponent.

**Returns**

A double type with the value of breakdown exponent.

#### 3.7.3.6 getCurrentCell()

```
double stringarma::SolarCell::getCurrentCell (
            void )
```

Gets the cell's current [A].

**Returns**

A double type with the value of the current [A].

#### 3.7.3.7 getCurrentPhotogenerated()

```
double stringarma::SolarCell::getCurrentPhotogenerated (
            void )
```

Gets the photogenerated current [A].

**Returns**

A double type with the value of the photogenerated current [A].

#### 3.7.3.8 getCurrentReverseSaturation()

```
double stringarma::SolarCell::getCurrentReverseSaturation (
            void )
```

Gets the reverse saturation current [A].

**Returns**

A double type with the value of the reverse saturation current [A].

### 3.7.3.9 getCurrentShortcircuit()

```
double stringarma::SolarCell::getCurrentShortcircuit (
            void )
```

Gets the short-circuit current [A].

**Returns**

A double type with the value of the short-circuit current [A].

### 3.7.3.10 getIdealityFactor()

```
double stringarma::SolarCell::getIdealityFactor (
            void )
```

Gets the ideality factor.

**Returns**

A double type with the value of ideality factor.

### 3.7.3.11 getIndex()

```
int stringarma::SolarCell::getIndex (
            void )
```

Gets the cell's index.

**Returns**

An integer type with the value of the index.

### 3.7.3.12 getIrradiance()

```
double stringarma::SolarCell::getIrradiance (
            void )
```

Gets the irradiance [W/m2].

**Returns**

A double type with the value of the Irradiance [W/m2].

#### 3.7.3.13 getResistanceSeries()

```
double stringarma::SolarCell::getResistanceSeries (
            void )
```

Gets the total resistance of the cell in series.

**Returns**

A double type with the value of total resistance of the cell in series.

#### 3.7.3.14 getResistanceShunt()

```
double stringarma::SolarCell::getResistanceShunt (
            void )
```

Gets the total shunt resistance of the cell.

**Returns**

A double type with the value of total shunt resistance of the cell.

#### 3.7.3.15 getSoilingFactor()

```
double stringarma::SolarCell::getSoilingFactor (
            void )
```

Gets the soiling factor.

**Returns**

A double type with the value of soiling factor.

#### 3.7.3.16 getTemperatureCell()

```
double stringarma::SolarCell::getTemperatureCell (
            void )
```

Gets the temperature of the cell [ºC].

**Returns**

A double type with the value of the temperature of the cell [ºC].

#### 3.7.3.17 getTemperatureCoeff()

```
double stringarma::SolarCell::getTemperatureCoeff (
            void  )
```

Gets the temperature coefficient.

**Returns**

A double type with the value of temperature coefficient.

#### 3.7.3.18 getVoltageBreakdown()

```
double stringarma::SolarCell::getVoltageBreakdown (
            void  )
```

Gets the breakdown voltage [V].

**Returns**

A double type with the value of the breakdown voltage [V].

#### 3.7.3.19 getVoltageCell()

```
double stringarma::SolarCell::getVoltageCell (
            void  )
```

Gets the cell's voltage [V].

**Returns**

A double type with the value of the voltage [V].

#### 3.7.3.20 getVoltageOpenCircuit()

```
double stringarma::SolarCell::getVoltageOpenCircuit (
            void  )
```

Gets the open circuit voltage [V].

**Returns**

A double type with the value of the open circuit voltage [V].

#### 3.7.3.21 getVoltageTemperatureCoeff()

```
double stringarma::SolarCell::getVoltageTemperatureCoeff (
            void )
```

Gets the voltage temperature coefficient.

**Returns**

A double type with the value of voltage temperature coefficient.

#### 3.7.3.22 setBreakdownAlpha()

```
void stringarma::SolarCell::setBreakdownAlpha (
            double _alpha )
```

Set a double value for the alpha parameter.

**Parameters**

| | |
|---|---|
| *Double* | value of the alpha parameter. |

#### 3.7.3.23 setBreakdownExponent()

```
void stringarma::SolarCell::setBreakdownExponent (
            double _breakdown_exponent )
```

Sets the breakdown exponent.

**Parameters**

| | |
|---|---|
| *m* | A double type with the value of breakdown exponent. |

#### 3.7.3.24 setCurrentCell()

```
void stringarma::SolarCell::setCurrentCell (
            double _Icell )
```

Set a double value for the current [A].

**Parameters**

| | |
|---|---|
| *Double* | value of the cell's current [A]. |

#### 3.7.3.25 setCurrentPhotogenerated()

```
void stringarma::SolarCell::setCurrentPhotogenerated (
            void )
```

Updates the value for the photogenerated current [A] according to the current values of the temperature of the cell Tc and the irradiance G.

#### 3.7.3.26 setCurrentReverseSaturation()

```
void stringarma::SolarCell::setCurrentReverseSaturation (
            void )
```

Updates the value for the reverse saturation current [A] according to the current value of the temperature of the cell Tc.

#### 3.7.3.27 setCurrentShortcircuit()

```
void stringarma::SolarCell::setCurrentShortcircuit (
            void )
```

Updates the value for the short-circuit current [A] according to the current values of the temperature of the cell Tc and the irradiance G.

#### 3.7.3.28 setIdealityFactor()

```
void stringarma::SolarCell::setIdealityFactor (
            double _n )
```

Sets the ideality factor.

**Parameters**

| | |
|---|---|
| *n* | A double type with the value of ideality factor. |

#### 3.7.3.29 setIndex()

```
void stringarma::SolarCell::setIndex (
            int _index )
```

Set an integer value for the index.

**Parameters**

| | |
|---|---|
| *Integer* | number of the index. |

#### 3.7.3.30 setIrradiance()

```
void stringarma::SolarCell::setIrradiance (
            double _G )
```

Set a double value for the irradiance [W/m2].

**Parameters**

| | |
|---|---|
| *Doble* | value of the new irradiance [W/m2]. |

#### 3.7.3.31 setResistanceSeries()

```
void stringarma::SolarCell::setResistanceSeries (
            double _Rs )
```

Sets the total resistance of the cell in series.

**Parameters**

| | |
|---|---|
| *Rs* | A double type with the value of total resistance of the cell in series. |

#### 3.7.3.32 setResistanceShunt()

```
void stringarma::SolarCell::setResistanceShunt (
            double _Rsh )
```

Sets the total shunt resistance of the cell.

**Parameters**

| | |
|---|---|
| *Rsh* | A double type with the value of total shunt resistance of the cell. |

#### 3.7.3.33 setSoilingFactor()

```
void stringarma::SolarCell::setSoilingFactor (
            double _SF )
```

Sets the soiling factor.

**Parameters**

| | |
|---|---|
| *SF* | A double type with the value of soiling factor. |

#### 3.7.3.34 setTemperatureCell()

```
void stringarma::SolarCell::setTemperatureCell (
            double _Tc )
```

Set a double value for the temperature of the cell [ºC].

**Parameters**

| | |
|---|---|
| *Doble* | value of the new temperature [ºC]. |

#### 3.7.3.35 setTemperatureCoeff()

```
void stringarma::SolarCell::setTemperatureCoeff (
            double _a )
```

Sets the temperature coefficient.

**Parameters**

| | |
|---|---|
| *a* | A double type with the value of temperature coefficient. |

#### 3.7.3.36 setVoltageBreakdown()

```
void stringarma::SolarCell::setVoltageBreakdown (
            double _Vbreak )
```

Set a double value for the breakdown voltage [V].

**Parameters**

| | |
|---|---|
| *Double* | value of the cell's breakdown voltage [V]. |

#### 3.7.3.37 setVoltageCell()

```
void stringarma::SolarCell::setVoltageCell (
            double _Vcell )
```

Set a double value for the voltage [V].

**Parameters**

| | |
|---|---|
| *Double* | value of the cell's voltage [V]. |

#### 3.7.3.38 setVoltageOpenCircuit()

```
void stringarma::SolarCell::setVoltageOpenCircuit (
            void )
```

Updates the value for the open circuit voltage [V] according to the current values of the temperature of the cell Tc and the irradiance G.

#### 3.7.3.39 setVoltageTemperatureCoeff()

```
void stringarma::SolarCell::setVoltageTemperatureCoeff (
            double _B )
```

Sets the voltage temperature coefficient.

**Parameters**

| | |
|---|---|
| *B* | A double type with the value of voltage temperature coefficient. |

## 3.7.4 Member Data Documentation

### 3.7.4.1 breakdown_alpha

`double stringarma::SolarCell::breakdown_alpha [protected]`

Breakdown alpha.

### 3.7.4.2 breakdown_exponent

`double stringarma::SolarCell::breakdown_exponent [protected]`

Breakdown exponent.

### 3.7.4.3 current_cell

`double stringarma::SolarCell::current_cell [protected]`

Current through the cell [A].

### 3.7.4.4 current_photogenerated

`double stringarma::SolarCell::current_photogenerated [protected]`

Photogenerated current [A].

### 3.7.4.5 current_reverse_saturation

`double stringarma::SolarCell::current_reverse_saturation [protected]`

Reverse saturation current [A].

#### 3.7.4.6 current_shortcircuit

```
double stringarma::SolarCell::current_shortcircuit  [protected]
```

Shortcircuit current [A].

#### 3.7.4.7 ideality_factor

```
double stringarma::SolarCell::ideality_factor  [protected]
```

Ideality factor.

#### 3.7.4.8 index

```
int stringarma::SolarCell::index  [protected]
```

Index of the cell. Serves as an identifier (ID) of the cell once it is grouped inside a string.

#### 3.7.4.9 irradiance

```
double stringarma::SolarCell::irradiance  [protected]
```

Irradiance [W/m2].

#### 3.7.4.10 resistance_series

```
double stringarma::SolarCell::resistance_series  [protected]
```

Total resistance of the cell in series.

#### 3.7.4.11 resistance_shunt

```
double stringarma::SolarCell::resistance_shunt  [protected]
```

Total shunt resistance of the cell.

#### 3.7.4.12 soiling_factor

```
double stringarma::SolarCell::soiling_factor  [protected]
```

Soiling factor.

#### 3.7.4.13 temperature_cell

```
double stringarma::SolarCell::temperature_cell  [protected]
```

Temperature of the cell [ºC].

#### 3.7.4.14 temperature_coeff

```
double stringarma::SolarCell::temperature_coeff  [protected]
```

Temperature coefficient.

#### 3.7.4.15 voltage_breakdown

```
double stringarma::SolarCell::voltage_breakdown  [protected]
```

Breakdown voltage [V].

#### 3.7.4.16 voltage_cell

```
double stringarma::SolarCell::voltage_cell  [protected]
```

Voltage between the terminals of the cell [V].

#### 3.7.4.17 voltage_open_circuit

```
double stringarma::SolarCell::voltage_open_circuit  [protected]
```

Open circuit voltage [V].

#### 3.7.4.18 voltage_temperature_coeff

```
double stringarma::SolarCell::voltage_temperature_coeff [protected]
```

Voltage temperature coefficient.

## 3.8 stringarma::SolarPanel Class Reference

Represents a solar panel of a PV generator.

```
#include <pv_panel.h>
```

### Public Member Functions

- SolarPanel (std::string)

    *Constructor of the class solar_panel.*
- SolarPanel ()
- void setCellVoltageBreakdown (double)

    *Set a double value for the breakdown voltage [V] of the cells in the panel.*
- void setCellBreakdownAlpha (double)

    *Set a double value for the alpha parameter of the cells in the panel.*
- void setCellSoilingFactor (double)

    *Sets the soiling factor of the cells in the panel.*
- void setCellIdealityFactor (double)

    *Sets the ideality factor of the cells in the panel.*
- void setCellResistanceSeries (double)

    *Sets the total resistance of the cell in series of the cells in the panel.*
- void setCellResistanceShunt (double)

    *Sets the total shunt resistance of the cells in the panel.*
- void setCellTemperatureCoeff (double)

    *Sets the temperature coefficient of the cells in the panel.*
- void setCellVoltageTemperatureCoeff (double)

    *Sets the voltage temperature coefficient of the cells in the panel.*
- void setCellBreakdownExponent (double)

    *Sets the breakdown exponent of the cells in the panel.*
- void setVoltageKneeDiode (double)

    *Sets the knee voltage of the bypass diodes in the panel [V].*
- double getCellVoltageBreakdown (void)

    *Gets the breakdown voltage [V] of the cells in the panel.*
- double getCellBreakdownAlpha (void)

    *Gets the alpha parameter of the cells in the panel.*
- double getCellSoilingFactor (void)

    *Gets the soiling factor of the cells in the panel.*
- double getCellIdealityFactor (void)

    *Gets the ideality factor of the cells in the panel.*
- double getCellResistanceSeries (void)

    *Gets the total resistance in series of the cells in the panel.*
- double getCellResistanceShunt (void)

    *Gets the total shunt resistance of the cells in the panel.*

- double getCellTemperatureCoeff (void)

    *Gets the temperature coefficient of the cells in the panel.*
- double getCellVoltageTemperatureCoeff (void)

    *Gets the voltage temperature coefficient of the cells in the panel.*
- double getCellBreakdownExponent (void)

    *Gets the breakdown exponent of the cells in the panel.*
- double getVoltageKneeDiode (void)

    *Gets the knee voltage of the bypass diodes in the panel [V].*
- int getPanelSize ()

    *Returns the number of strings in the panel.*
- std::vector< std::pair< bool, std::vector< std::pair< double, double > > > > readInput (std::string)

    *Reads the input file with the operational data described in the User's Guide.*

## Friends

- class SolarSolver

## 3.8.1 Detailed Description

Represents a solar panel of a PV generator.

This object contains all the information to create the PV panel. It reads an input file with all the information related to the panel structure and temperature and irradiance values for every cell. Although, it does not create SolarString objects. It only contains the information, since no calculations are done. It also contains the information related to the PV cell and bypass diode that will conform the panel. Only allows the creation of panels with cells with the same properties.

**See also**

SolarCell
BypassDiode
SolarString

## 3.8.2 Constructor & Destructor Documentation

### 3.8.2.1 SolarPanel() `[1/2]`

```
stringarma::SolarPanel::SolarPanel (
            std::string filepath )
```

Constructor of the class solar_panel.

**Parameters**

| | |
|---|---|
| *∗filepath* | Absolute path of the input file with the operational data. Its format should follow the one described in the User's Guide. |

**See also**

Input file format

#### 3.8.2.2 SolarPanel() [2/2]

```
stringarma::SolarPanel::SolarPanel ( )
```

### 3.8.3 Member Function Documentation

#### 3.8.3.1 getCellBreakdownAlpha()

```
double stringarma::SolarPanel::getCellBreakdownAlpha (
            void )
```

Gets the alpha parameter of the cells in the panel.

**Returns**

A double type with the value of alpha parameter.

#### 3.8.3.2 getCellBreakdownExponent()

```
double stringarma::SolarPanel::getCellBreakdownExponent (
            void )
```

Gets the breakdown exponent of the cells in the panel.

**Returns**

A double type with the value of breakdown exponent.

#### 3.8.3.3 getCellIdealityFactor()

```
double stringarma::SolarPanel::getCellIdealityFactor (
            void )
```

Gets the ideality factor of the cells in the panel.

**Returns**

A double type with the value of ideality factor.

#### 3.8.3.4 getCellResistanceSeries()

```
double stringarma::SolarPanel::getCellResistanceSeries (
            void  )
```

Gets the total resistance in series of the cells in the panel.

**Returns**

A double type with the value of total resistance of the cell in series.

#### 3.8.3.5 getCellResistanceShunt()

```
double stringarma::SolarPanel::getCellResistanceShunt (
            void  )
```

Gets the total shunt resistance of the cells in the panel.

**Returns**

A double type with the value of total shunt resistance of the cell.

#### 3.8.3.6 getCellSoilingFactor()

```
double stringarma::SolarPanel::getCellSoilingFactor (
            void  )
```

Gets the soiling factor of the cells in the panel.

**Returns**

A double type with the value of soiling factor.

#### 3.8.3.7 getCellTemperatureCoeff()

```
double stringarma::SolarPanel::getCellTemperatureCoeff (
            void  )
```

Gets the temperature coefficient of the cells in the panel.

**Returns**

A double type with the value of temperature coefficient.

#### 3.8.3.8 getCellVoltageBreakdown()

```
double stringarma::SolarPanel::getCellVoltageBreakdown (
            void )
```

Gets the breakdown voltage [V] of the cells in the panel.

**Returns**

A double type with the value of the breakdown voltage [V].

#### 3.8.3.9 getCellVoltageTemperatureCoeff()

```
double stringarma::SolarPanel::getCellVoltageTemperatureCoeff (
            void )
```

Gets the voltage temperature coefficient of the cells in the panel.

**Returns**

A double type with the value of voltage temperature coefficient.

#### 3.8.3.10 getPanelSize()

```
int stringarma::SolarPanel::getPanelSize ( )
```

Returns the number of strings in the panel.

**Returns**

Integer with the number of strings in the panel.

#### 3.8.3.11 getVoltageKneeDiode()

```
double stringarma::SolarPanel::getVoltageKneeDiode (
            void )
```

Gets the knee voltage of the bypass diodes in the panel [V].

**Returns**

The knee voltage of the bypass diodes in the panel [V].

#### 3.8.3.12 readInput()

```
vector< pair< bool, vector< pair< double, double > > > > stringarma::SolarPanel::readInput
(
            std::string filepath )
```

Reads the input file with the operational data described in the User's Guide.

Reads the information and generates the panel according to it.

**Parameters**

| | |
|---|---|
| *filepath* | String with the absolute path of the input file. |

**Returns**

A vector of pairs. Every element represents a string. Every pair contains a bool value, representing the state of the diode, and a vector of pairs of double values, the G and Tc correspondingly, representing every cell in the string.

#### 3.8.3.13 setCellBreakdownAlpha()

```
void stringarma::SolarPanel::setCellBreakdownAlpha (
            double _alpha )
```

Set a double value for the alpha parameter of the cells in the panel.

**Parameters**

| | |
|---|---|
| *Double* | value of the alpha parameter. |

#### 3.8.3.14 setCellBreakdownExponent()

```
void stringarma::SolarPanel::setCellBreakdownExponent (
            double _m )
```

Sets the breakdown exponent of the cells in the panel.

**Parameters**

| | |
|---|---|
| *m* | A double type with the value of breakdown exponent. |

#### 3.8.3.15 setCellIdealityFactor()

```
void stringarma::SolarPanel::setCellIdealityFactor (
            double _n )
```

Sets the ideality factor of the cells in the panel.

**Parameters**

| | |
|---|---|
| *n* | A double type with the value of ideality factor. |

#### 3.8.3.16 setCellResistanceSeries()

```
void stringarma::SolarPanel::setCellResistanceSeries (
            double _Rs )
```

Sets the total resistance of the cell in series of the cells in the panel.

**Parameters**

| | |
|---|---|
| *Rs* | A double type with the value of total resistance of the cell in series. |

#### 3.8.3.17 setCellResistanceShunt()

```
void stringarma::SolarPanel::setCellResistanceShunt (
            double _Rsh )
```

Sets the total shunt resistance of the cells in the panel.

**Parameters**

| | |
|---|---|
| *Rsh* | A double type with the value of total shunt resistance of the cell. |

#### 3.8.3.18 setCellSoilingFactor()

```
void stringarma::SolarPanel::setCellSoilingFactor (
            double _SF )
```

Sets the soiling factor of the cells in the panel.

**Parameters**

| | |
|---|---|
| *SF* | A double type with the value of soiling factor. |

#### 3.8.3.19 setCellTemperatureCoeff()

```
void stringarma::SolarPanel::setCellTemperatureCoeff (
            double _a )
```

Sets the temperature coefficient of the cells in the panel.

**Parameters**

| | |
|---|---|
| *a* | A double type with the value of temperature coefficient. |

#### 3.8.3.20 setCellVoltageBreakdown()

```
void stringarma::SolarPanel::setCellVoltageBreakdown (
            double _Vbreak )
```

Set a double value for the breakdown voltage [V] of the cells in the panel.

**Parameters**

| | |
|---|---|
| *Double* | value of the cell's breakdown voltage [V]. |

#### 3.8.3.21 setCellVoltageTemperatureCoeff()

```
void stringarma::SolarPanel::setCellVoltageTemperatureCoeff (
            double _B )
```

Sets the voltage temperature coefficient of the cells in the panel.

**Parameters**

| | |
|---|---|
| *B* | A double type with the value of voltage temperature coefficient. |

#### 3.8.3.22 setVoltageKneeDiode()

```
void stringarma::SolarPanel::setVoltageKneeDiode (
            double Vknee )
```

Sets the knee voltage of the bypass diodes in the panel [V].

**Parameters**

| | |
|---|---|
| *Vknee* | Knee voltage of the bypass diodes in the panel. |

### 3.8.4 Friends And Related Function Documentation

#### 3.8.4.1 SolarSolver

```
friend class SolarSolver [friend]
```

## 3.9 stringarma::SolarSolver Class Reference

```
#include <pv_solver.h>
```

### Public Member Functions

- SolarSolver (stringarma::SolarPanel &)

  *Constructor of the class SolarSolver.*
- void setMaxIterations (int)

  *Set a double value for the maximum number of iterations to solve the Newton-Raphson iterative method.*
- void setEpsilon (double)

  *Set a double value for the condition of convergence (epsilon).*
- int getMaxIterations (void)

  *Gets the maximum number of iterations to solve the Newton-Raphson iterative method.*
- double getEpsilon (void)

  *Gets the condition of convergence (epsilon).*
- void calcIVcharacteristic (std::string)

  *Calculates the I-V characteristic of the SolarPanel object introduced in the constructor of the SolarSolver object.*
- void calcIVcharacteristic (std::string, double, double, int)

  *Calculates the I-V characteristic of the SolarPanel object introduced in the constructor of the SolarSolver object.*
- void calcState (std::string, double)

  *Calculates the state the SolarPanel object introduced in the constructor of the SolarSolver object for a single value of voltage.*

### Protected Member Functions

- void retrieveDataFromStringArray (std::multimap< std::pair< double, double >, SameIshortcircuitAndVbreakdownGroup, Classcomp > &MultiMapElDets)

  *Fulfills the multimap structure with the data contained in the array of SolarString objects.*
- double findMaxVoltageLimit ()

  *Returns the "first upper limit" of the I-V characteristic.*
- void findVoltageLimitsForChangesInCurrent (const double LTO)

  *Fulfills the voltage_limit parameter of the sum_same_i_shortcircuit_group CellsGroup.*
- void findVoltageLimitsForChangesInVoltage ()

  *Fulfills the voltage_limit parameter of the sum_same_i_shortcircuit_and_v_breakdown_group CellsGroup.*
- void generatePanelVector ()

  *Fulfills the vector with the information contained in the array of strings that represents the panel.*
- int findWorkingZone (double Vin)

  *Given the external voltage limits (for changes in current), and numbering every zone in between them (from higher V zones to lower V zones), returns the operational zone that belongs to a certain voltage.*
- void calcUpperZones (int m, std::vector< double > &vVector)

  *Adds the voltage to the strings corresponding to the cells that belong to an upper working zone than the current one.*
- void calcLowerZones (int m, std::vector< double > &vVector)

  *Adds the voltage to the strings corresponding to the cells that belong to a lower working zone than the current one.*

- void calcMiddleZones (int m, double Vpan, std::vector< double > &vVector)

    *Adds the voltage to the strings corresponding to the cells that belong to the current working zone.*
- double findTotalCurrent (double Vpan)

    *Returns the total current that the panel will generate given a certain voltage.*
- void assignStringVoltages (double Vpan, std::vector< double > &VString)

    *Given a total voltage through the panel, assigns the corresponding voltage to every string.*
- double calcNewtonRaphson (solar_string ∗st, double Vp, int _dimX, int nS)

    *Calculates the state of a given PV panel (an array of SolarString objects) by using the Newton-Raphson iterative method.*

## Protected Attributes

- int number_strings

    *Number of strings in the panel.*
- solar_string ∗ string_array

    *Array of SolarString objects that compose the PV panel.*
- std::vector< SameIshortcircuitGroup > panel_vector

    *Main vector where all the info will be organized by short-circuit current, breakdown voltage and number of string.*
- int max_iterations

    *Maximum number of iterations to solve the Newton-Raphson iterative method.*
- double epsilon

    *Condition of convergence.*

## 3.9.1 Constructor & Destructor Documentation

### 3.9.1.1 SolarSolver()

```
stringarma::SolarSolver::SolarSolver (
            stringarma::SolarPanel & panel )
```

Constructor of the class SolarSolver.

It automatically generates the objects to represent the class, update their parameters and classify them into groups in order to calculate the initial estimation.

**Parameters**

| | |
|---|---|
| *The* | SolarPanel object with the information to be simulated already loaded. |

## 3.9.2 Member Function Documentation

#### 3.9.2.1 assignStringVoltages()

```
void stringarma::SolarSolver::assignStringVoltages (
            double Vpan,
            std::vector< double > & VString )  [protected]
```

Given a total voltage through the panel, assigns the corresponding voltage to every string.

**Parameters**

| | |
|---|---|
| *Vpan* | Total voltage in the panel [V]. |
| *VString* | Vector with the values of the voltage in every string. The index of the vector matches the index of the string in the panel. |

**Returns**

No value. But the VString vector is updated with the calculated values.

#### 3.9.2.2 calcIVcharacteristic() [1/2]

```
void stringarma::SolarSolver::calcIVcharacteristic (
            std::string output_path )
```

Calculates the I-V characteristic of the SolarPanel object introduced in the constructor of the SolarSolver object.

The resulting characteristic is stored in a file, specified as a parameter.

**Parameters**

| | |
|---|---|
| *output_path* | Full path of the file where to store the I-V characteristic. If the file exists it will be replaced. If it doesn't, it will be created. |

#### 3.9.2.3 calcIVcharacteristic() [2/2]

```
void stringarma::SolarSolver::calcIVcharacteristic (
            std::string output_path,
            double start_v,
            double end_v,
            int numb_points )
```

Calculates the I-V characteristic of the SolarPanel object introduced in the constructor of the SolarSolver object.

The resulting characteristic is stored in a file, specified as a parameter.

**Parameters**

| | |
|---|---|
| *output_path* | Full path of the file where to store the I-V characteristic. If the file exists it will be replaced. If it doesn't, it will be created. |
| *start_v* | First voltage value in the characteristic. |
| *end_v* | Last voltage value in the characteristic. |
| *numb_points* | Number of points in the characteristic. |

#### 3.9.2.4 calcLowerZones()

```
void stringarma::SolarSolver::calcLowerZones (
            int m,
            std::vector< double > & vVector )  [protected]
```

Adds the voltage to the strings corresponding to the cells that belong to a lower working zone than the current one.

The working zone 0 is understood as the highest one. This implies that the active group of cells (imposing their Isc) has a lower Isc than the studied cells. Then the cells are in non-active state.

**Parameters**

| | |
|---|---|
| *m* | The current working zone. |
| *vVector* | Vector containing the voltage of every string. The index of the vector matches the number of the string. |

**Returns**

No value. The vVector is updated.

**See also**

findWorkingZone()

#### 3.9.2.5 calcMiddleZones()

```
void stringarma::SolarSolver::calcMiddleZones (
            int m,
            double Vpan,
            std::vector< double > & vVector )  [protected]
```

Adds the voltage to the strings corresponding to the cells that belong to the current working zone.

This implies that the internal voltage limits (for changes in voltage) shall be taken into account. Since diodes may be or not conducting, in this zone cells can be in any state.

**Parameters**

| | |
|---|---|
| *m* | The current working zone. |
| *vVector* | Vector containing the voltage of every string. The index of the vector matches the number of the string. |

**Returns**

No value. The vVector is updated.

**See also**

findWorkingZone()

#### 3.9.2.6 calcNewtonRaphson()

```
double stringarma::SolarSolver::calcNewtonRaphson (
            solar_string * st,
            double Vp,
            int _dimX,
            int nS )  [protected]
```

Calculates the state of a given PV panel (an array of SolarString objects) by using the Newton-Raphson iterative method.

Certain parameters such as the convergence condition or the maximum number of iteration can be set through other member methods.

**Parameters**

| | |
|---|---|
| *st* | Array of SolarString objects to be solved. |
| *Vp* | Total voltage in the panel [V]. |
| *dimX* | Total number of variables. That is the total number of cells plus the number of strings plus one (the total current). |
| *nS* | Number of strings. |

**Returns**

The total current generated by the panel. The values of voltage and current through every component of the panel are updated in the corresponding object.

#### 3.9.2.7 calcState()

```
void stringarma::SolarSolver::calcState (
            std::string output_path,
            double Vpan )
```

Calculates the state the SolarPanel object introduced in the constructor of the SolarSolver object for a single value of voltage.

The resulting .csv file, specified as a parameter, contains the number of string, position in the string, irradiance, temperature, current and voltage of every cell. It also contains the current of every diode.

**Parameters**

| | |
|---|---|
| *output_path* | Full path of the file where to store the calculated state. If the file exists it will be replaced. If it doesn't, it will be created. |
| *Vpan* | Total voltage in the panel. |

#### 3.9.2.8 calcUpperZones()

```
void stringarma::SolarSolver::calcUpperZones (
            int m,
            std::vector< double > & vVector )  [protected]
```

Adds the voltage to the strings corresponding to the cells that belong to an upper working zone than the current one.

The working zone 0 is understood as the highest one. This implies that the active group of cells (imposing their Isc) has a higher Isc than the studied cells. This is possible if either the cells are in breakdown or the bypass diode of the string is conducting.

**Parameters**

| | |
|---|---|
| *m* | The current working zone. |
| *vVector* | Vector containing the voltage of every string. The index of the vector matches the number of the string. |

**Returns**

No value. The vVector is updated.

**See also**

findWorkingZone()

#### 3.9.2.9 findMaxVoltageLimit()

```
double stringarma::SolarSolver::findMaxVoltageLimit ( )  [protected]
```

Returns the "first upper limit" of the I-V characteristic.

That is the sum of Voc of every cell.

#### 3.9.2.10 findTotalCurrent()

```
double stringarma::SolarSolver::findTotalCurrent (
            double Vpan )  [protected]
```

Returns the total current that the panel will generate given a certain voltage.

**Parameters**

| | |
|---|---|
| *Vpan* | Total voltage in the panel [V]. |

#### 3.9.2.11 findVoltageLimitsForChangesInCurrent()

```
void stringarma::SolarSolver::findVoltageLimitsForChangesInCurrent (
            const double LTO )  [protected]
```

Fulfills the voltage_limit parameter of the sum_same_i_shortcircuit_group CellsGroup.

These "external" limits are the total voltage in the panel needed to get every cell into breakdown or to get the bypass diode to conducting state, whatever happens first. Notice that it is possible (and panels are designed so) that the bypass diode will enter conduction state before the breakdown.

The external limits represent a change in the total current.

**Parameters**

| | |
|---|---|
| *LTO* | Maximum voltage limit of the panel (sum of all the open circuit voltages). |

#### 3.9.2.12 findVoltageLimitsForChangesInVoltage()

```
void stringarma::SolarSolver::findVoltageLimitsForChangesInVoltage ( )  [protected]
```

Fulfills the voltage_limit parameter of the sum_same_i_shortcircuit_and_v_breakdown_group CellsGroup.

These "internal" limits are the total voltage in the panel needed to get every bypass diode in conducting state. In case there's no diode, the limit will match the lower external limit. These limits are relative to the inferior voltage limit for changes in current.

The internal limits represent a change in the distribution of the total voltage.

#### 3.9.2.13 findWorkingZone()

```
int stringarma::SolarSolver::findWorkingZone (
            double Vin )  [protected]
```

Given the external voltage limits (for changes in current), and numbering every zone in between them (from higher V zones to lower V zones), returns the operational zone that belongs to a certain voltage.

LT2 ----zone 2-----LT1-----zone 1-----LT0----zone 0 where LT2<LT1<LT0

**Parameters**

| | |
|---|---|
| *Vin* | Total voltage in the panel [V]. |

**Returns**

An integer of the working zone.

#### 3.9.2.14 generatePanelVector()

```
void stringarma::SolarSolver::generatePanelVector ( )  [protected]
```

Fulfills the vector with the information contained in the array of strings that represents the panel.

It also organize all this info in the vector. By Isc, then by Isc and Vbrx, and by Isc, Vbrx and Index of string. In addition, calculates totals of every group and the limits of the I-V characteristic.

#### 3.9.2.15 getEpsilon()

```
double stringarma::SolarSolver::getEpsilon (
             void  )
```

Gets the condition of convergence (epsilon).

**Returns**

A double type with the condition of convergence (epsilon).

#### 3.9.2.16 getMaxIterations()

```
int stringarma::SolarSolver::getMaxIterations (
             void  )
```

Gets the maximum number of iterations to solve the Newton-Raphson iterative method.

**Returns**

A double type with the value of the maximum number of iterations.

#### 3.9.2.17 retrieveDataFromStringArray()

```
void stringarma::SolarSolver::retrieveDataFromStringArray (
             std::multimap< std::pair< double, double >, SameIshortcircuitAndVbreakdownGroup,
Classcomp > & MultiMapElDets )  [protected]
```

Fulfills the multimap structure with the data contained in the array of SolarString objects.

The groups of cells of different strings working under the same breakdown current (Isc) and voltage (Vbr) are grouped inside the multimap.

**Parameters**

| | |
|---|---|
| *&MultiMapElDets* | Type multimap. The key is composed by two double values: Iscx and Vbrx in this order. The SameIshortcircuitAndVbreakdownGroup struct contains the corresponding sum_same_i_shortcircuit_and_v_breakdown_group (CellsGroup struct) with the total sums of the parameters of the groups of groups, and the detailed_same_i_shortcircuit_and_v_breakdown_group list of pairs with the integer pointing the number of the string and the CellsGroup struct with the specifics of this group. |
| *∗&panel* | Array of SolarString objects. |

#### 3.9.2.18 setEpsilon()

```
void stringarma::SolarSolver::setEpsilon (
            double _epsilon )
```

Set a double value for the condition of convergence (epsilon).

**Parameters**

| | |
|---|---|
| *Double* | value for the condition of convergence. |

#### 3.9.2.19 setMaxIterations()

```
void stringarma::SolarSolver::setMaxIterations (
            int maxIt )
```

Set a double value for the maximum number of iterations to solve the Newton-Raphson iterative method.

**Parameters**

| | |
|---|---|
| *Double* | value for the maximum number of iterations. |

### 3.9.3 Member Data Documentation

#### 3.9.3.1 epsilon

```
double stringarma::SolarSolver::epsilon  [protected]
```

Condition of convergence.

#### 3.9.3.2 max_iterations

`int stringarma::SolarSolver::max_iterations [protected]`

Maximum number of iterations to solve the Newton-Raphson iterative method.

#### 3.9.3.3 number_strings

`int stringarma::SolarSolver::number_strings [protected]`

Number of strings in the panel.

#### 3.9.3.4 panel_vector

`std::vector<SameIshortcircuitGroup> stringarma::SolarSolver::panel_vector [protected]`

Main vector where all the info will be organized by short-circuit current, breakdown voltage and number of string.

#### 3.9.3.5 string_array

`solar_string* stringarma::SolarSolver::string_array [protected]`

Array of SolarString objects that compose the PV panel.

## 3.10 stringarma::TotalsOfCellsGroup Struct Reference

Defines the total parameters of a group of PV cells in the same string that have the same short-circuit current.

`#include <pv_string.h>`

### Public Attributes

- std::vector< int > index

  *Vector with the physical position index of every cell in the group.*
- double current_shortcircuit

  *Short-circuit current of every cell in this group.*
- double sum_voltage_breakdown

  *Sum of all breakdown voltages of the PV cells in this group.*
- double sum_voltage_open_circuit

  *Sum of all open circuit voltages of the PV cells in this group.*
- double sum_voltage_breakdown_in_group

  *Sum of all breakdown voltages of the PV cells calculated on the group.*

### 3.10.1 Detailed Description

Defines the total parameters of a group of PV cells in the same string that have the same short-circuit current.

### 3.10.2 Member Data Documentation

#### 3.10.2.1 current_shortcircuit

```
double stringarma::TotalsOfCellsGroup::current_shortcircuit
```

Short-circuit current of every cell in this group.

#### 3.10.2.2 index

```
std::vector<int> stringarma::TotalsOfCellsGroup::index
```

Vector with the physical position index of every cell in the group.

The size of this vector is equal to the number of cells included in this group.

#### 3.10.2.3 sum_voltage_breakdown

```
double stringarma::TotalsOfCellsGroup::sum_voltage_breakdown
```

Sum of all breakdown voltages of the PV cells in this group.

#### 3.10.2.4 sum_voltage_breakdown_in_group

```
double stringarma::TotalsOfCellsGroup::sum_voltage_breakdown_in_group
```

Sum of all breakdown voltages of the PV cells calculated on the group.

**See also**

Solar string's theoretical documentation

#### 3.10.2.5 sum_voltage_open_circuit

```
double stringarma::TotalsOfCellsGroup::sum_voltage_open_circuit
```

Sum of all open circuit voltages of the PV cells in this group.

# Index