

STRING\_ARMA

2.0

Generated by Doxygen 1.8.17



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 Namespace Documentation</b>	<b>3</b>
2.1 stringarma Namespace Reference	3
2.1.1 Function Documentation	4
2.1.1.1 equalIsc()	4
2.1.1.2 loadInitialValues()	5
2.1.1.3 operator<()	5
2.1.1.4 operator==( )	5
2.1.1.5 SameCurrent()	5
2.1.2 Variable Documentation	5
2.1.2.1 BOLTZMANN_CONST	5
2.1.2.2 BREAKDOWN_ALPHA_REF	5
2.1.2.3 BREAKDOWN_EXPONENT_REF	6
2.1.2.4 CURRENT_PHOTOGENERATED_REF	6
2.1.2.5 CURRENT_REVERSE_SATURATION_REF	6
2.1.2.6 CURRENT_SHORTCIRCUIT_REF	6
2.1.2.7 ELECTRONS_CHARGE	6
2.1.2.8 IDEALITY_FACTOR_REF	6
2.1.2.9 IRRADIANCE_REF	7
2.1.2.10 RESISTANCE_SERIES_REF	7
2.1.2.11 RESISTANCE_SHUNT_REF	7
2.1.2.12 SOILING_FACTOR_REF	7
2.1.2.13 TEMPERATURE_CELL_REF	7
2.1.2.14 TEMPERATURE_COEFF_REF	7
2.1.2.15 VOLTAGE_BREAKDOWN_REF	8
2.1.2.16 VOLTAGE_OPEN_CIRCUIT_REF	8
2.1.2.17 VOLTAGE_TEMPERATURE_COEFF_REF	8
<b>3 Class Documentation</b>	<b>9</b>
3.1 stringarma::BypassDiode Class Reference	9
3.1.1 Detailed Description	10
3.1.2 Constructor & Destructor Documentation	10
3.1.2.1 BypassDiode()	11
3.1.3 Member Function Documentation	11
3.1.3.1 calcFunctionD()	11
3.1.3.2 calcFuntionDiodeDerivativeRespectVoltage()	11
3.1.3.3 getCurrentDiode()	12
3.1.3.4 getCurrentReverseSaturation()	12
3.1.3.5 getIdealityFactor()	12
3.1.3.6 getTemperatureDiode()	12
3.1.3.7 setCurrentDiode()	12

3.1.3.8 setCurrentReverseSaturation()	13
3.1.3.9 setIdealityFactor()	13
3.1.3.10 setTemperatureDiode()	13
3.1.4 Member Data Documentation	13
3.1.4.1 current_diode	14
3.1.4.2 current_reverse_saturation	14
3.1.4.3 ideality_factor	14
3.1.4.4 temperature_diode	14
3.2 stringarma::CellsGroup Struct Reference	14
3.2.1 Detailed Description	15
3.2.2 Member Data Documentation	15
3.2.2.1 current_shortcircuit	15
3.2.2.2 group_size	15
3.2.2.3 limit_voltage	15
3.2.2.4 sum_voltage_breakdown_in_group	15
3.2.2.5 sum_voltage_open_circuit_all_cells	16
3.2.2.6 sum_voltage_open_circuit_non_active_cells	16
3.3 stringarma::Classcomp Struct Reference	16
3.3.1 Detailed Description	16
3.3.2 Member Function Documentation	16
3.3.2.1 operator()	16
3.4 stringarma::SameIshortcircuitAndVbreakdownGroup Struct Reference	17
3.4.1 Detailed Description	17
3.4.2 Member Data Documentation	17
3.4.2.1 detailed_same_i_shortcircuit_and_v_breakdown_group	17
3.4.2.2 sum_same_i_shortcircuit_and_v_breakdown_group	17
3.5 stringarma::SameIshortcircuitGroup Struct Reference	18
3.5.1 Detailed Description	18
3.5.2 Member Data Documentation	18
3.5.2.1 detailed_same_i_shortcircuit_group	18
3.5.2.2 sum_same_i_shortcircuit_group	18
3.6 stringarma::solar_string Class Reference	19
3.6.1 Detailed Description	20
3.6.2 Constructor & Destructor Documentation	21
3.6.2.1 solar_string()	21
3.6.2.2 ~solar_string()	21
3.6.3 Member Function Documentation	21
3.6.3.1 findInitialState()	21
3.6.3.2 getMinimCurrentShortcircuit()	21
3.6.3.3 getSumVoltageAllCells()	22
3.6.3.4 getSumVoltageBreakdown()	22
3.6.3.5 getSumVoltageOpenCircuit()	22

3.6.3.6	<a href="#">getVoltageDiode()</a>	23
3.6.3.7	<a href="#">getVoltageString()</a>	23
3.6.3.8	<a href="#">getWithDiode()</a>	23
3.6.3.9	<a href="#">setSumVoltageBreakdownInGroup()</a>	23
3.6.3.10	<a href="#">setSumVoltageAllCells()</a>	24
3.6.3.11	<a href="#">setSumVoltageBreakdown()</a>	24
3.6.3.12	<a href="#">setSumVoltageOpenCircuit()</a>	24
3.6.3.13	<a href="#">setVoltageDiode()</a>	24
3.6.3.14	<a href="#">setVoltageString()</a>	24
3.6.3.15	<a href="#">updateStringsData()</a>	25
3.6.4	<a href="#">Friends And Related Function Documentation</a>	25
3.6.4.1	<a href="#">SolarSolver</a>	25
3.6.5	<a href="#">Member Data Documentation</a>	25
3.6.5.1	<a href="#">cells_array</a>	25
3.6.5.2	<a href="#">diode_bypass</a>	26
3.6.5.3	<a href="#">groupsByCurrentShortcircuit</a>	26
3.6.5.4	<a href="#">string_size</a>	26
3.7	<a href="#">stringarma::SolarCell Class Reference</a>	26
3.7.1	<a href="#">Detailed Description</a>	29
3.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	31
3.7.2.1	<a href="#">SolarCell() [1/2]</a>	31
3.7.2.2	<a href="#">SolarCell() [2/2]</a>	31
3.7.3	<a href="#">Member Function Documentation</a>	31
3.7.3.1	<a href="#">calcFunctionC()</a>	31
3.7.3.2	<a href="#">calcFunctionCellDerivativeRespectCurrent()</a>	32
3.7.3.3	<a href="#">calcFunctionCellDerivativeRespectVoltage()</a>	32
3.7.3.4	<a href="#">getBreakdownAlpha()</a>	32
3.7.3.5	<a href="#">getBreakdownExponent()</a>	33
3.7.3.6	<a href="#">getCurrentCell()</a>	33
3.7.3.7	<a href="#">getCurrentPhotogenerated()</a>	33
3.7.3.8	<a href="#">getCurrentReverseSaturation()</a>	33
3.7.3.9	<a href="#">getCurrentShortcircuit()</a>	34
3.7.3.10	<a href="#">getIdealityFactor()</a>	34
3.7.3.11	<a href="#">getIndex()</a>	34
3.7.3.12	<a href="#">getIrradiance()</a>	34
3.7.3.13	<a href="#">getResistanceSeries()</a>	35
3.7.3.14	<a href="#">getResistanceShunt()</a>	35
3.7.3.15	<a href="#">getSoilingFactor()</a>	35
3.7.3.16	<a href="#">getTemperatureCell()</a>	35
3.7.3.17	<a href="#">getTemperatureCoeff()</a>	36
3.7.3.18	<a href="#">getVoltageBreakdown()</a>	36
3.7.3.19	<a href="#">getVoltageCell()</a>	36

3.7.3.20	<a href="#">getVoltageOpenCircuit()</a>	36
3.7.3.21	<a href="#">getVoltageTemperatureCoeff()</a>	37
3.7.3.22	<a href="#">setBreakdownAlpha()</a>	37
3.7.3.23	<a href="#">setBreakdownExponent()</a>	37
3.7.3.24	<a href="#">setCurrentCell()</a>	37
3.7.3.25	<a href="#">setCurrentPhotogenerated()</a>	38
3.7.3.26	<a href="#">setCurrentReverseSaturation()</a>	38
3.7.3.27	<a href="#">setCurrentShortcircuit()</a>	38
3.7.3.28	<a href="#">setIdealityFactor()</a>	38
3.7.3.29	<a href="#">setIndex()</a>	39
3.7.3.30	<a href="#">setIrradiance()</a>	39
3.7.3.31	<a href="#">setResistanceSeries()</a>	39
3.7.3.32	<a href="#">setResistanceShunt()</a>	39
3.7.3.33	<a href="#">setSoilingFactor()</a>	40
3.7.3.34	<a href="#">setTemperatureCell()</a>	40
3.7.3.35	<a href="#">setTemperatureCoeff()</a>	40
3.7.3.36	<a href="#">setVoltageBreakdown()</a>	41
3.7.3.37	<a href="#">setVoltageCell()</a>	41
3.7.3.38	<a href="#">setVoltageOpenCircuit()</a>	41
3.7.3.39	<a href="#">setVoltageTemperatureCoeff()</a>	41
3.7.4	<a href="#">Member Data Documentation</a>	42
3.7.4.1	<a href="#">breakdown_alpha</a>	42
3.7.4.2	<a href="#">breakdown_exponent</a>	42
3.7.4.3	<a href="#">current_cell</a>	42
3.7.4.4	<a href="#">current_photogenerated</a>	42
3.7.4.5	<a href="#">current_reverse_saturation</a>	42
3.7.4.6	<a href="#">current_shortcircuit</a>	43
3.7.4.7	<a href="#">ideality_factor</a>	43
3.7.4.8	<a href="#">index</a>	43
3.7.4.9	<a href="#">irradiance</a>	43
3.7.4.10	<a href="#">resistance_series</a>	43
3.7.4.11	<a href="#">resistance_shunt</a>	43
3.7.4.12	<a href="#">soiling_factor</a>	44
3.7.4.13	<a href="#">temperature_cell</a>	44
3.7.4.14	<a href="#">temperature_coeff</a>	44
3.7.4.15	<a href="#">voltage_breakdown</a>	44
3.7.4.16	<a href="#">voltage_cell</a>	44
3.7.4.17	<a href="#">voltage_open_circuit</a>	44
3.7.4.18	<a href="#">voltage_temperature_coeff</a>	45
3.8	<a href="#">stringarma::SolarPanel Class Reference</a>	45
3.8.1	<a href="#">Detailed Description</a>	46
3.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	46

3.8.2.1 SolarPanel() [1/2]	46
3.8.2.2 SolarPanel() [2/2]	47
3.8.3 Member Function Documentation	47
3.8.3.1 getCellBreakdownAlpha()	47
3.8.3.2 getCellBreakdownExponent()	47
3.8.3.3 getCellIdealityFactor()	47
3.8.3.4 getCellResistanceSeries()	48
3.8.3.5 getCellResistanceShunt()	48
3.8.3.6 getCellSoilingFactor()	48
3.8.3.7 getCellTemperatureCoeff()	48
3.8.3.8 getCellVoltageBreakdown()	49
3.8.3.9 getCellVoltageTemperatureCoeff()	49
3.8.3.10 getPanelSize()	49
3.8.3.11 getVoltageKneeDiode()	49
3.8.3.12 readInput()	49
3.8.3.13 setCellBreakdownAlpha()	50
3.8.3.14 setCellBreakdownExponent()	50
3.8.3.15 setCellIdealityFactor()	50
3.8.3.16 setCellResistanceSeries()	51
3.8.3.17 setCellResistanceShunt()	51
3.8.3.18 setCellSoilingFactor()	51
3.8.3.19 setCellTemperatureCoeff()	51
3.8.3.20 setCellVoltageBreakdown()	52
3.8.3.21 setCellVoltageTemperatureCoeff()	52
3.8.3.22 setVoltageKneeDiode()	52
3.8.4 Friends And Related Function Documentation	52
3.8.4.1 SolarSolver	53
3.9 stringarma::SolarSolver Class Reference	53
3.9.1 Constructor & Destructor Documentation	54
3.9.1.1 SolarSolver()	54
3.9.2 Member Function Documentation	54
3.9.2.1 assignStringVoltages()	55
3.9.2.2 calcIVcharacteristic() [1/2]	55
3.9.2.3 calcIVcharacteristic() [2/2]	55
3.9.2.4 calcLowerZones()	56
3.9.2.5 calcMiddleZones()	56
3.9.2.6 calcNewtonRaphson()	57
3.9.2.7 calcState()	57
3.9.2.8 calcUpperZones()	58
3.9.2.9 findMaxVoltageLimit()	58
3.9.2.10 findTotalCurrent()	59
3.9.2.11 findVoltageLimitsForChangesInCurrent()	60

3.9.2.12 findVoltageLimitsForChangesInVoltage()	60
3.9.2.13 findWorkingZone()	60
3.9.2.14 generatePanelVector()	61
3.9.2.15 getEpsilon()	61
3.9.2.16 getMaxIterations()	61
3.9.2.17 retrieveDataFromStringArray()	61
3.9.2.18 setEpsilon()	62
3.9.2.19 setMaxIterations()	62
3.9.3 Member Data Documentation	62
3.9.3.1 epsilon	62
3.9.3.2 max_iterations	63
3.9.3.3 number_strings	63
3.9.3.4 panel_vector	63
3.9.3.5 string_array	63
3.10 stringarma::TotalsOfCellsGroup Struct Reference	63
3.10.1 Detailed Description	64
3.10.2 Member Data Documentation	64
3.10.2.1 current_shortcircuit	64
3.10.2.2 index	64
3.10.2.3 sum_voltage_breakdown	64
3.10.2.4 sum_voltage_breakdown_in_group	64
3.10.2.5 sum_voltage_open_circuit	64
<b>Index</b>	<b>65</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">stringarma::BypassDiode</a>	Represents a common component of a photovoltaic generator, a bypass diode . . . . .	9
<a href="#">stringarma::CellsGroup</a>	Structure to gather global information of a group of cells that share, at least, the same short-circuit current . . . . .	14
<a href="#">stringarma::Classcomp</a>	Comparison function object for the multimap . . . . .	16
<a href="#">stringarma::SameIshortcircuitAndVbreakdownGroup</a>	Structure with info of groups of cells with the same Isc and Vbrx . . . . .	17
<a href="#">stringarma::SameIshortcircuitGroup</a>	Vector meant to contain info of groups of cells with the same short-circuit current (Isc) . . . . .	18
<a href="#">stringarma::solar_string</a>	Represents a string of solar cells group under the same bypass diode . . . . .	19
<a href="#">stringarma::SolarCell</a>	Represents a PV cell, the most basic element of a solar generator . . . . .	26
<a href="#">stringarma::SolarPanel</a>	Represents a solar panel of a PV generator . . . . .	45
<a href="#">stringarma::SolarSolver</a>	. . . . .	53
<a href="#">stringarma::TotalsOfCellsGroup</a>	Defines the total parameters of a group of PV cells in the same string that have the same short-circuit current . . . . .	63



## 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/°K].*
- constexpr double [TEMPERATURE\\_CELL\\_REF](#) {25.0}  
*Temperature of the cell of reference, in case it is not specified [°C].*
- 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/°C].*
- constexpr double [VOLTAGE\\_TEMPERATURE\\_COEFF\\_REF](#) {-0.0023}  
*Voltage temperature coefficient. Depends on the material, but this one belongs to silicon [V/°C].*
- 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/K].

### 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]
```

Idealty factor.

### 2.1.2.9 IRRADIANCE\_REF

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

Irradiance of reference, in case it is not specified [W/m<sup>2</sup>].

### 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 [°C].

### 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/°C].

#### 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/°C].



## 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 [ $^{\circ}\text{C}$ ].*
- void [setCurrentReverseSaturation](#) (void)  
*Updates the value for the reverse saturation current [ $\text{A}$ ] according to the current value of the temperature of the diode  $T_c$ .*
- void [setCurrentDiode](#) (double)  
*Updates the value for the current in the diode [ $\text{A}$ ].*
- void [setIdealityFactor](#) (double)  
*Updates the value of the ideality factor of the bypass diode.*
- double [getCurrentReverseSaturation](#) (void)  
*Gets the diode's reverse saturation current [ $\text{A}$ ].*
- double [getTemperatureDiode](#) (void)  
*Gets the diode's temperature [ $^{\circ}\text{C}$ ].*
- double [getCurrentDiode](#) (void)  
*Gets the diode's current [ $\text{A}$ ].*
- 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,  $V_d$ , 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 [°C].*
- 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 °C.
- **m\_ref** Ideality factor of 1.5.

Other mathematical constants used in the calculations are:

- **k** Boltzmann constant: 1.38e-23 J/°K
- **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

<i>Double</i>	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

<i>Double</i>	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 [°C].

Returns

A double type with the value of the temperature [°C].

#### 3.1.3.7 `setCurrentDiode()`

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

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

## Parameters

<i>Id</i>	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 of the ideality factor of the bypass diode.

## Parameters

<i>m</i>	Ideality factor.
----------	------------------

**3.1.3.10 setTemperatureDiode()**

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

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

## Parameters

<i>Double</i>	value for the temperature of the diode [°C].
---------------	--

**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 [°C].

## 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](#).*
- [getWithDiode](#) (void)  
*Indicates whether the string of PV cells has a by-pass diode or not.*
- [getMinimCurrentShortcircuit](#) (void)  
*Gets the minimum short-circuit current in the string [A].*
- [getSumVoltageOpenCircuit](#) (void)  
*Gets the sum of all the open circuit voltage of the cells in the string [V].*
- [getSumVoltageBreakdown](#) (void)  
*Gets the sum of the breakdown voltage of all the cells in the string [V].*
- [getVoltageString](#) (void)  
*Gets the voltage between the terminals of the string [V].*
- [getSumVoltageAllCells](#) (void)  
*Gets the sum of the voltage between the terminals of every cell in the string [V].*
- [getVoltageDiode](#) (void)  
*Gets the voltage between the terminals of the bypass diode [V].*
- [setSumVoltageOpenCircuit](#) (void)  
*Updates the Svoc attribute with the current value of Voc of every cell.*
- [setSumVoltageBreakdown](#) (void)  
*Updates the Svbreak attribute with the current value of Vbreak of every cell.*
- [setVoltageString](#) (double)  
*Set a new value for the voltage between the terminals of the 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.*
- [setSumVoltageBreakdownInGroup](#) (void)  
*Updates the value of the sum of the breakdown voltage (SVbrx) of every group of cells in CellsGr.*
- [setVoltageDiode](#) (double)  
*Set a new value for the knee voltage of the bypass diode of the string.*
- [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.*
- [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

<i>Iin</i>	Total current through the string [A].
<i>Vin</i>	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 theoretical 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**

<i>Vdiode</i>	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

<i>Vstring</i>	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

<i>sc</i>	<code>solar_cell</code> object that represents all the cells in the string.
<i>string_input</i>	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 I<sub>sc</sub>.

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 [°C].*
- 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 [°C].*

- 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,  $I_{cell}$ , 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,  $V_{cell}$ , 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 [°C].*
- 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 [°C].
- **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 °C.
- **Gref** The irradiance of reference is 1000 W/m<sup>2</sup>.

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/°K
- **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/°C
- **B** Voltage temperature coefficient: -0.0023 V/°C
- **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  $T_c$  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

<i>solar_cell</i>	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, lcell, 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 lcell 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 lcell 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 [°C].

##### Returns

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

### 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

<i>Double</i>	value of the alpha parameter.
---------------	-------------------------------

### 3.7.3.23 setBreakdownExponent()

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

Sets the breakdown exponent.

#### Parameters

<i>m</i>	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

<i>Double</i>	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  $T_c$  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  $T_c$ .

**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  $T_c$  and the irradiance  $G$ .

**3.7.3.28 setIdealityFactor()**

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

Sets the ideality factor.

## Parameters

<i>n</i>	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

<i>Integer</i>	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

<i>Doble</i>	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

<i>Rs</i>	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

<i>Rsh</i>	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

<i>SF</i>	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 [°C].

## Parameters

<i>Doble</i>	value of the new temperature [°C].
--------------	------------------------------------

**3.7.3.35 setTemperatureCoeff()**

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

Sets the temperature coefficient.

## Parameters

<i>a</i>	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

<i>Double</i>	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

<i>Double</i>	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  $T_c$  and the irradiance  $G$ .

### 3.7.3.39 setVoltageTemperatureCoeff()

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

Sets the voltage temperature coefficient.

#### Parameters

<i>B</i>	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/m<sup>2</sup>].

#### 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 [°C].

#### 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

<i>*filepath</i>	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

<i>filepath</i>	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

<i>Double</i>	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

<i>m</i>	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

<i>n</i>	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

<i>Rs</i>	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

<i>Rsh</i>	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

<i>SF</i>	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

<i>a</i>	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

<i>Double</i>	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

<i>B</i>	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

<i>Vknee</i>	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](#) > &MultiMapEIDets)  
*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< [SamelshortcircuitGroup](#) > [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	<a href="#">SolarPanel</a> 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

<i>Vpan</i>	Total voltage in the panel [V].
<i>VString</i>	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

<i>output_path</i>	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

<i>output_path</i>	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.
<i>start_v</i>	First voltage value in the characteristic.
<i>end_v</i>	Last voltage value in the characteristic.
<i>numb_points</i>	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  $I_{sc}$ ) has a lower  $I_{sc}$  than the studied cells. Then the cells are in non-active state.

## Parameters

<i>m</i>	The current working zone.
<i>vVector</i>	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

<i>m</i>	The current working zone.
<i>vVector</i>	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

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

<i>output_path</i>	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.
<i>Vpan</i>	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  $I_{sc}$ ) has a higher  $I_{sc}$  than the studied cells. This is possible if either the cells are in breakdown or the bypass diode of the string is conducting.

#### Parameters

<i>m</i>	The current working zone.
<i>vVector</i>	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  $V_{oc}$  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

<i>V<sub>pan</sub></i>	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

<i>LTO</i>	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

<i>Vin</i>	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

<i>&amp;MultiMapElDets</i>	Type multimap. The key is composed by two double values: Iscx and Vbrx in this order. The <a href="#">SameIshortcircuitAndVbreakdownGroup</a> struct contains the corresponding sum_same_i_shortcircuit_and_v_breakdown_group ( <a href="#">CellsGroup</a> 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 <a href="#">CellsGroup</a> struct with the specifics of this group.
<i>*&amp;panel</i>	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

<i>Double</i>	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

<i>Double</i>	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

~solar\_string  
stringarma::solar\_string, 21

assignStringVoltages  
stringarma::SolarSolver, 54

BOLTZMANN\_CONST  
stringarma, 5

breakdown\_alpha  
stringarma::SolarCell, 42

BREAKDOWN\_ALPHA\_REF  
stringarma, 5

breakdown\_exponent  
stringarma::SolarCell, 42

BREAKDOWN\_EXPONENT\_REF  
stringarma, 5

BypassDiode  
stringarma::BypassDiode, 10

calcFunctionC  
stringarma::SolarCell, 31

calcFunctionCellDerivativeRespectCurrent  
stringarma::SolarCell, 31

calcFunctionCellDerivativeRespectVoltage  
stringarma::SolarCell, 32

calcFunctionD  
stringarma::BypassDiode, 11

calcFuntionDiodeDerivativeRespectVoltage  
stringarma::BypassDiode, 11

calcIVcharacteristic  
stringarma::SolarSolver, 55

calcLowerZones  
stringarma::SolarSolver, 56

calcMiddleZones  
stringarma::SolarSolver, 56

calcNewtonRaphson  
stringarma::SolarSolver, 57

calcState  
stringarma::SolarSolver, 57

calcUpperZones  
stringarma::SolarSolver, 58

cells\_array  
stringarma::solar\_string, 25

current\_cell  
stringarma::SolarCell, 42

current\_diode  
stringarma::BypassDiode, 13

current\_photogenerated  
stringarma::SolarCell, 42

CURRENT\_PHOTOGENERATED\_REF

stringarma, 6

current\_reverse\_saturation  
stringarma::BypassDiode, 14  
stringarma::SolarCell, 42

CURRENT\_REVERSE\_SATURATION\_REF  
stringarma, 6

current\_shortcircuit  
stringarma::CellsGroup, 15  
stringarma::SolarCell, 42  
stringarma::TotalsOfCellsGroup, 64

CURRENT\_SHORTCIRCUIT\_REF  
stringarma, 6

detailed\_same\_i\_shortcircuit\_and\_v\_breakdown\_group  
stringarma::SameIshortcircuitAndVbreakdownGroup,  
17

detailed\_same\_i\_shortcircuit\_group  
stringarma::SameIshortcircuitGroup, 18

diode\_bypass  
stringarma::solar\_string, 25

ELECTRONS\_CHARGE  
stringarma, 6

epsilon  
stringarma::SolarSolver, 62

equallsc  
stringarma, 4

findInitialState  
stringarma::solar\_string, 21

findMaxVoltageLimit  
stringarma::SolarSolver, 58

findTotalCurrent  
stringarma::SolarSolver, 58

findVoltageLimitsForChangesInCurrent  
stringarma::SolarSolver, 60

findVoltageLimitsForChangesInVoltage  
stringarma::SolarSolver, 60

findWorkingZone  
stringarma::SolarSolver, 60

generatePanelVector  
stringarma::SolarSolver, 61

getBreakdownAlpha  
stringarma::SolarCell, 32

getBreakdownExponent  
stringarma::SolarCell, 32

getCellBreakdownAlpha  
stringarma::SolarPanel, 47

getCellBreakdownExponent

- stringarma::SolarPanel, 47
- getCellIdealityFactor
  - stringarma::SolarPanel, 47
- getCellResistanceSeries
  - stringarma::SolarPanel, 47
- getCellResistanceShunt
  - stringarma::SolarPanel, 48
- getCellSoilingFactor
  - stringarma::SolarPanel, 48
- getCellTemperatureCoeff
  - stringarma::SolarPanel, 48
- getCellVoltageBreakdown
  - stringarma::SolarPanel, 48
- getCellVoltageTemperatureCoeff
  - stringarma::SolarPanel, 49
- getCurrentCell
  - stringarma::SolarCell, 33
- getCurrentDiode
  - stringarma::BypassDiode, 12
- getCurrentPhotogenerated
  - stringarma::SolarCell, 33
- getCurrentReverseSaturation
  - stringarma::BypassDiode, 12
  - stringarma::SolarCell, 33
- getCurrentShortcircuit
  - stringarma::SolarCell, 33
- getEpsilon
  - stringarma::SolarSolver, 61
- getIdealityFactor
  - stringarma::BypassDiode, 12
  - stringarma::SolarCell, 34
- getIndex
  - stringarma::SolarCell, 34
- getIrradiance
  - stringarma::SolarCell, 34
- getMaxIterations
  - stringarma::SolarSolver, 61
- getMinimCurrentShortcircuit
  - stringarma::solar\_string, 21
- getPanelSize
  - stringarma::SolarPanel, 49
- getResistanceSeries
  - stringarma::SolarCell, 34
- getResistanceShunt
  - stringarma::SolarCell, 35
- getSoilingFactor
  - stringarma::SolarCell, 35
- getSumVoltageAllCells
  - stringarma::solar\_string, 22
- getSumVoltageBreakdown
  - stringarma::solar\_string, 22
- getSumVoltageOpenCircuit
  - stringarma::solar\_string, 22
- getTemperatureCell
  - stringarma::SolarCell, 35
- getTemperatureCoeff
  - stringarma::SolarCell, 35
- getTemperatureDiode
  - stringarma::BypassDiode, 12
- getVoltageBreakdown
  - stringarma::SolarCell, 36
- getVoltageCell
  - stringarma::SolarCell, 36
- getVoltageDiode
  - stringarma::solar\_string, 22
- getVoltageKneeDiode
  - stringarma::SolarPanel, 49
- getVoltageOpenCircuit
  - stringarma::SolarCell, 36
- getVoltageString
  - stringarma::solar\_string, 23
- getVoltageTemperatureCoeff
  - stringarma::SolarCell, 36
- getWithDiode
  - stringarma::solar\_string, 23
- group\_size
  - stringarma::CellsGroup, 15
- groupsByCurrentShortcircuit
  - stringarma::solar\_string, 26
- ideality\_factor
  - stringarma::BypassDiode, 14
  - stringarma::SolarCell, 43
- IDEALITY\_FACTOR\_REF
  - stringarma, 6
- index
  - stringarma::SolarCell, 43
  - stringarma::TotalsOfCellsGroup, 64
- irradiance
  - stringarma::SolarCell, 43
- IRRADIANCE\_REF
  - stringarma, 6
- limit\_voltage
  - stringarma::CellsGroup, 15
- loadInitialValues
  - stringarma, 4
- max\_iterations
  - stringarma::SolarSolver, 62
- number\_strings
  - stringarma::SolarSolver, 63
- operator<
  - stringarma, 5
- operator()
  - stringarma::Classcomp, 16
- operator==
  - stringarma, 5
- panel\_vector
  - stringarma::SolarSolver, 63
- readInput
  - stringarma::SolarPanel, 49
- resistance\_series
  - stringarma::SolarCell, 43

RESISTANCE\_SERIES\_REF  
     stringarma, 7  
 resistance\_shunt  
     stringarma::SolarCell, 43  
 RESISTANCE\_SHUNT\_REF  
     stringarma, 7  
 retrieveDataFromStringArray  
     stringarma::SolarSolver, 61  
  
 SameCurrent  
     stringarma, 5  
 setBreakdownAlpha  
     stringarma::SolarCell, 37  
 setBreakdownExponent  
     stringarma::SolarCell, 37  
 setCellBreakdownAlpha  
     stringarma::SolarPanel, 50  
 setCellBreakdownExponent  
     stringarma::SolarPanel, 50  
 setCellIdealityFactor  
     stringarma::SolarPanel, 50  
 setCellResistanceSeries  
     stringarma::SolarPanel, 51  
 setCellResistanceShunt  
     stringarma::SolarPanel, 51  
 setCellSoilingFactor  
     stringarma::SolarPanel, 51  
 setCellTemperatureCoeff  
     stringarma::SolarPanel, 51  
 setCellVoltageBreakdown  
     stringarma::SolarPanel, 52  
 setCellVoltageTemperatureCoeff  
     stringarma::SolarPanel, 52  
 setCurrentCell  
     stringarma::SolarCell, 37  
 setCurrentDiode  
     stringarma::BypassDiode, 12  
 setCurrentPhotogenerated  
     stringarma::SolarCell, 38  
 setCurrentReverseSaturation  
     stringarma::BypassDiode, 13  
     stringarma::SolarCell, 38  
 setCurrentShortcircuit  
     stringarma::SolarCell, 38  
 setEpsilon  
     stringarma::SolarSolver, 62  
 setIdealityFactor  
     stringarma::BypassDiode, 13  
     stringarma::SolarCell, 38  
 setIndex  
     stringarma::SolarCell, 38  
 setIrradiance  
     stringarma::SolarCell, 39  
 setMaxIterations  
     stringarma::SolarSolver, 62  
 setResistanceSeries  
     stringarma::SolarCell, 39  
 setResistanceShunt  
     stringarma::SolarCell, 39  
  
 setSoilingFactor  
     stringarma::SolarCell, 40  
 setSumVolageBreakdownInGroup  
     stringarma::solar\_string, 23  
 setSumVoltageAllCells  
     stringarma::solar\_string, 23  
 setSumVoltageBreakdown  
     stringarma::solar\_string, 24  
 setSumVoltageOpenCircuit  
     stringarma::solar\_string, 24  
 setTemperatureCell  
     stringarma::SolarCell, 40  
 setTemperatureCoeff  
     stringarma::SolarCell, 40  
 setTemperatureDiode  
     stringarma::BypassDiode, 13  
 setVoltageBreakdown  
     stringarma::SolarCell, 40  
 setVoltageCell  
     stringarma::SolarCell, 41  
 setVoltageDiode  
     stringarma::solar\_string, 24  
 setVoltageKneeDiode  
     stringarma::SolarPanel, 52  
 setVoltageOpenCircuit  
     stringarma::SolarCell, 41  
 setVoltageString  
     stringarma::solar\_string, 24  
 setVoltageTemperatureCoeff  
     stringarma::SolarCell, 41  
 soiling\_factor  
     stringarma::SolarCell, 43  
 SOILING\_FACTOR\_REF  
     stringarma, 7  
 solar\_string  
     stringarma::solar\_string, 21  
 SolarCell  
     stringarma::SolarCell, 31  
 SolarPanel  
     stringarma::SolarPanel, 46, 47  
 SolarSolver  
     stringarma::solar\_string, 25  
     stringarma::SolarPanel, 52  
     stringarma::SolarSolver, 54  
 string\_array  
     stringarma::SolarSolver, 63  
 string\_size  
     stringarma::solar\_string, 26  
 stringarma, 3  
     BOLTZMANN\_CONST, 5  
     BREAKDOWN\_ALPHA\_REF, 5  
     BREAKDOWN\_EXPONENT\_REF, 5  
     CURRENT\_PHOTOGENERATED\_REF, 6  
     CURRENT\_REVERSE\_SATURATION\_REF, 6  
     CURRENT\_SHORTCIRCUIT\_REF, 6  
     ELECTRONS\_CHARGE, 6  
     equallsc, 4  
     IDEALITY\_FACTOR\_REF, 6

- IRRADIANCE\_REF, 6
- loadInitialValues, 4
- operator<, 5
- operator==, 5
- RESISTANCE\_SERIES\_REF, 7
- RESISTANCE\_SHUNT\_REF, 7
- SameCurrent, 5
- SOILING\_FACTOR\_REF, 7
- TEMPERATURE\_CELL\_REF, 7
- TEMPERATURE\_COEFF\_REF, 7
- VOLTAGE\_BREAKDOWN\_REF, 7
- VOLTAGE\_OPEN\_CIRCUIT\_REF, 8
- VOLTAGE\_TEMPERATURE\_COEFF\_REF, 8
- stringarma::BypassDiode, 9
  - BypassDiode, 10
  - calcFunctionD, 11
  - calcFuntionDiodeDerivativeRespectVoltage, 11
  - current\_diode, 13
  - current\_reverse\_saturation, 14
  - getCurrentDiode, 12
  - getCurrentReverseSaturation, 12
  - getIdealityFactor, 12
  - getTemperatureDiode, 12
  - ideality\_factor, 14
  - setCurrentDiode, 12
  - setCurrentReverseSaturation, 13
  - setIdealityFactor, 13
  - setTemperatureDiode, 13
  - temperature\_diode, 14
- stringarma::CellsGroup, 14
  - current\_shortcircuit, 15
  - group\_size, 15
  - limit\_voltage, 15
  - sum\_voltage\_breakdown\_in\_group, 15
  - sum\_voltage\_open\_circuit\_all\_cells, 15
  - sum\_voltage\_open\_circuit\_non\_active\_cells, 16
- stringarma::Classcomp, 16
  - operator(), 16
- stringarma::SameIshortcircuitAndVbreakdownGroup, 17
  - detailed\_same\_i\_shortcircuit\_and\_v\_breakdown\_group, 17
  - sum\_same\_i\_shortcircuit\_and\_v\_breakdown\_group, 17
- stringarma::SameIshortcircuitGroup, 18
  - detailed\_same\_i\_shortcircuit\_group, 18
  - sum\_same\_i\_shortcircuit\_group, 18
- stringarma::solar\_string, 19
  - ~solar\_string, 21
  - cells\_array, 25
  - diode\_bypass, 25
  - findInitialState, 21
  - getMinimCurrentShortcircuit, 21
  - getSumVoltageAllCells, 22
  - getSumVoltageBreakdown, 22
  - getSumVoltageOpenCircuit, 22
  - getVoltageDiode, 22
  - getVoltageString, 23
  - getWithDiode, 23
  - groupsByCurrentShortcircuit, 26
  - setSumVolageBreakdownInGroup, 23
  - setSumVoltageAllCells, 23
  - setSumVoltageBreakdown, 24
  - setSumVoltageOpenCircuit, 24
  - setVoltageDiode, 24
  - setVoltageString, 24
  - solar\_string, 21
  - SolarSolver, 25
  - string\_size, 26
  - updateStringsData, 25
- stringarma::SolarCell, 26
  - breakdown\_alpha, 42
  - breakdown\_exponent, 42
  - calcFunctionC, 31
  - calcFunctionCellDerivativeRespectCurrent, 31
  - calcFunctionCellDerivativeRespectVoltage, 32
  - current\_cell, 42
  - current\_photogenerated, 42
  - current\_reverse\_saturation, 42
  - current\_shortcircuit, 42
  - getBreakdownAlpha, 32
  - getBreakdownExponent, 32
  - getCurrentCell, 33
  - getCurrentPhotogenerated, 33
  - getCurrentReverseSaturation, 33
  - getCurrentShortcircuit, 33
  - getIdealityFactor, 34
  - getIndex, 34
  - getIrradiance, 34
  - getResistanceSeries, 34
  - getResistanceShunt, 35
  - getSoilingFactor, 35
  - getTemperatureCell, 35
  - getTemperatureCoeff, 35
  - getVoltageBreakdown, 36
  - getVoltageCell, 36
  - getVoltageOpenCircuit, 36
  - getVoltageTemperatureCoeff, 36
  - ideality\_factor, 43
  - index, 43
  - irradiance, 43
  - resistance\_series, 43
  - resistance\_shunt, 43
  - setBreakdownAlpha, 37
  - setBreakdownExponent, 37
  - setCurrentCell, 37
  - setCurrentPhotogenerated, 38
  - setCurrentReverseSaturation, 38
  - setCurrentShortcircuit, 38
  - setIdealityFactor, 38
  - setIndex, 38
  - setIrradiance, 39
  - setResistanceSeries, 39
  - setResistanceShunt, 39
  - setSoilingFactor, 40
  - setTemperatureCell, 40

- setTemperatureCoeff, [40](#)
- setVoltageBreakdown, [40](#)
- setVoltageCell, [41](#)
- setVoltageOpenCircuit, [41](#)
- setVoltageTemperatureCoeff, [41](#)
- soiling\_factor, [43](#)
- SolarCell, [31](#)
- temperature\_cell, [44](#)
- temperature\_coeff, [44](#)
- voltage\_breakdown, [44](#)
- voltage\_cell, [44](#)
- voltage\_open\_circuit, [44](#)
- voltage\_temperature\_coeff, [44](#)
- stringarma::SolarPanel, [45](#)
  - getCellBreakdownAlpha, [47](#)
  - getCellBreakdownExponent, [47](#)
  - getCellIdealityFactor, [47](#)
  - getCellResistanceSeries, [47](#)
  - getCellResistanceShunt, [48](#)
  - getCellSoilingFactor, [48](#)
  - getCellTemperatureCoeff, [48](#)
  - getCellVoltageBreakdown, [48](#)
  - getCellVoltageTemperatureCoeff, [49](#)
  - getPanelSize, [49](#)
  - getVoltageKneeDiode, [49](#)
  - readInput, [49](#)
  - setCellBreakdownAlpha, [50](#)
  - setCellBreakdownExponent, [50](#)
  - setCellIdealityFactor, [50](#)
  - setCellResistanceSeries, [51](#)
  - setCellResistanceShunt, [51](#)
  - setCellSoilingFactor, [51](#)
  - setCellTemperatureCoeff, [51](#)
  - setCellVoltageBreakdown, [52](#)
  - setCellVoltageTemperatureCoeff, [52](#)
  - setVoltageKneeDiode, [52](#)
  - SolarPanel, [46, 47](#)
  - SolarSolver, [52](#)
- stringarma::SolarSolver, [53](#)
  - assignStringVoltages, [54](#)
  - calcIVcharacteristic, [55](#)
  - calcLowerZones, [56](#)
  - calcMiddleZones, [56](#)
  - calcNewtonRaphson, [57](#)
  - calcState, [57](#)
  - calcUpperZones, [58](#)
  - epsilon, [62](#)
  - findMaxVoltageLimit, [58](#)
  - findTotalCurrent, [58](#)
  - findVoltageLimitsForChangesInCurrent, [60](#)
  - findVoltageLimitsForChangesInVoltage, [60](#)
  - findWorkingZone, [60](#)
  - generatePanelVector, [61](#)
  - getEpsilon, [61](#)
  - getMaxIterations, [61](#)
  - max\_iterations, [62](#)
  - number\_strings, [63](#)
  - panel\_vector, [63](#)
  - retrieveDataFromStringArray, [61](#)
  - setEpsilon, [62](#)
  - setMaxIterations, [62](#)
  - SolarSolver, [54](#)
  - string\_array, [63](#)
- stringarma::TotalsOfCellsGroup, [63](#)
  - current\_shortcircuit, [64](#)
  - index, [64](#)
  - sum\_voltage\_breakdown, [64](#)
  - sum\_voltage\_breakdown\_in\_group, [64](#)
  - sum\_voltage\_open\_circuit, [64](#)
- sum\_same\_i\_shortcircuit\_and\_v\_breakdown\_group
  - stringarma::SameIshortcircuitAndVbreakdownGroup, [17](#)
- sum\_same\_i\_shortcircuit\_group
  - stringarma::SameIshortcircuitGroup, [18](#)
- sum\_voltage\_breakdown
  - stringarma::TotalsOfCellsGroup, [64](#)
- sum\_voltage\_breakdown\_in\_group
  - stringarma::CellsGroup, [15](#)
  - stringarma::TotalsOfCellsGroup, [64](#)
- sum\_voltage\_open\_circuit
  - stringarma::TotalsOfCellsGroup, [64](#)
- sum\_voltage\_open\_circuit\_all\_cells
  - stringarma::CellsGroup, [15](#)
- sum\_voltage\_open\_circuit\_non\_active\_cells
  - stringarma::CellsGroup, [16](#)
- temperature\_cell
  - stringarma::SolarCell, [44](#)
- TEMPERATURE\_CELL\_REF
  - stringarma, [7](#)
- temperature\_coeff
  - stringarma::SolarCell, [44](#)
- TEMPERATURE\_COEFF\_REF
  - stringarma, [7](#)
- temperature\_diode
  - stringarma::BypassDiode, [14](#)
- updateStringsData
  - stringarma::solar\_string, [25](#)
- voltage\_breakdown
  - stringarma::SolarCell, [44](#)
- VOLTAGE\_BREAKDOWN\_REF
  - stringarma, [7](#)
- voltage\_cell
  - stringarma::SolarCell, [44](#)
- voltage\_open\_circuit
  - stringarma::SolarCell, [44](#)
- VOLTAGE\_OPEN\_CIRCUIT\_REF
  - stringarma, [8](#)
- voltage\_temperature\_coeff
  - stringarma::SolarCell, [44](#)
- VOLTAGE\_TEMPERATURE\_COEFF\_REF
  - stringarma, [8](#)