Reference Manual for etaOpt
Contents

1 container — Class *(Notebook: etaOpt.m) (Context: etaOpt)* .......................................................... 4
2 etaOpt — Class *(Notebook: etaOpt.m) (Context: etaOpt)* .............................................................. 11
3 mechStack — Class *(Notebook: etaOpt.m) (Context: etaOpt)* ......................................................... 22
4 monoStack — Class *(Notebook: etaOpt.m) (Context: etaOpt)* ......................................................... 26
5 oneDiodeJunction — Class *(Notebook: etaOpt.m) (Context: etaOpt)* .......................................... 32
6 oneDiodeJunctionI0FirstTerm — Class *(Notebook: etaOpt.m) (Context: etaOpt)* ......................... 36
7 oneDiodeJunctionSemiEmpirical — Class *(Notebook: etaOpt.m) (Context: etaOpt)* .................... 40
8 singleJunction — Class *(Notebook: etaOpt.m) (Context: etaOpt)* .............................................. 44
9 spectrum — Class *(Notebook: etaOpt.m) (Context: etaOpt)* ....................................................... 56

10 Public Functions .......................................................................................................................... 70
10.1 connectDirs() — *(Notebook: Hilfsfunktionen.m)* ............................................................... 70
10.2 dateTime() — *(Notebook: Hilfsfunktionen.m)* ................................................................. 71
10.3 dbgPrint() — *(Notebook: Hilfsfunktionen.m)* ................................................................. 71
10.4 dateTime() — *(Notebook: Hilfsfunktionen.m)* ................................................................. 72
10.5 export2Origin() — *(Notebook: Hilfsfunktionen.m)* ........................................................ 72
10.6 getFilename() — *(Notebook: Hilfsfunktionen.m)* ............................................................ 74
10.7 getFilenameWithoutExtension() — *(Notebook: Hilfsfunktionen.m)* .................................... 74
10.8 makeUsage() — *(Notebook: Hilfsfunktionen.m)* .............................................................. 75
10.9 interpolatingFunctionQ() — *(Notebook: Hilfsfunktionen.m)* ........................................... 75
10.10 num() — *(Notebook: Hilfsfunktionen.m)* ........................................................................ 76
10.11 outFkt() — *(Notebook: Hilfsfunktionen.m)* ....................................................................... 76
10.12 outStr() — *(Notebook: Hilfsfunktionen.m)* ....................................................................... 76
10.13 readListWithComment() — *(Notebook: Hilfsfunktionen.m)* ............................................. 77
10.14 showStatus() — *(Notebook: Hilfsfunktionen.m)* .............................................................. 78

11 Functions sorted by Notebooks .................................................................................................. 79
11.1 Package ../..//Hilfsfunktionen.m ..................................................................................... 79

Class Graph ..................................................................................................................................... 84
Liste aller Contexte:

- etaOpt
- Hilfsfunktionen

Enjoy!
class container : public Object

Class (Notebook: etaOpt.m) (Context: etaOpt')

Inheritance

```
Object
  ↓
  1
container
    ↓
    4
monoStack
    ↓
    3
mechStack
```

Public Members

1.4 Public Method      appendTo () ......................................................... 5
1.5 Public Method      evaluateForEachItem () ........................................... 6
1.6 Public Method      giveCurrent () ..................................................... 6
1.7 Public Method      giveFirst () ........................................................ 6
1.8 Public Method      giveItem () .......................................................... 6
1.9 Public Method      giveLast () ......................................................... 7
1.10 Public Method      giveList () ........................................................ 7
1.11 Public Method      giveName () ....................................................... 7
1.12 Public Method      giveNext () ........................................................ 7
1.13 Public Method      giveNumberOf () ............................................... 8
new[] creates an Object container, which is used to navigate through a list of items. Following options are available:

- **name->**.... each container can have a name, which can be retrieved with giveName[].
- **item->**{..... a list of items, which can be anything, i.e. an object, a number, another container...

---

### Public Methods

- **givePosition ()**
- **givePrevious ()**
- **isFirst ()**
- **isLast ()**
- **moveTo ()**
- **moveToFirst ()**
- **moveToLast ()**
- **setList ()**
- **setName ()**

### Protected Members

- **curPos**
- **myList**
- **myName**

**appendTo [item]** appends `item` to the itemlist.
1.5  Public Method  `evaluateForEachItem` ()

evaluateForEachItem[method] this method makes only sense, if container only contains container and Objects. In this case evaluateForEachItem goes down in the hierarchy of container, and if it finds an object which is not a container, evaluates the method, and returns the returnvalue if any. I.e. evaluateForEachItem[giveName[]] returns `{{cell1,cell2},{cell3}}`

1.6  Public Method  `giveCurrent` ()

giveCurrent[] returns the current item in the list. If there is no list or no item to return, it returns Null.

1.7  Public Method  `giveFirst` ()

giveFirst[] returns the first item in the list. If there is no list or no item to return, it returns Null.

1.8  Public Method  `giveItem` ()

giveItem[position] returns the item at given position in the list. I.e. giveItem[3] returns the 3th item in the list. If there is no list or no item to return, it returns Null.
1.9

Public Method giveLast ()

giveLast[] returns the last item in the list. If there is no list or no item to return, it returns Null.

1.10

Public Method giveList ()

giveList[] returns the whole list of items.

1.11

Public Method giveName ()

giveName[] returns the name of the container as string.

1.12

Public Method giveNext ()

giveNext[] returns the next item in list. If there is no list or no item to return, it returns Null.
1.13 

Public Method  `giveNumberOf()`

giveNumberOf[] returns the number of items in the list.

1.14 

Public Method  `givePosition()`

givePosition[] returns the current position in the item list. I.e. `giveItem[givePosition]` is the same as `giveCurrent[]`.

1.15 

Public Method  `givePrevious()`

givePrevious[] returns the previous item in list. If there is no list or no item to return, it returns Null.

1.16 

Public Method  `isFirst()`
isFirst[] returns the True if the current item is the first item. If there is no list or no item to return, it returns False.
1.17 Public Method isLast ()

isLast[] returns the True if the current item is the last item. If there is no list or no item to return, it returns False.

1.18 Public Method moveTo ()

moveToLast[position] moves to a given position in the list but returns nothing. I.e. moveTo[3];givePosition[] returns 3.

1.19 Public Method moveToFirst ()

moveToFirst[] moves to the first item of the list but returns nothing. I.e. moveToFirst[];isFirst[] returns True.

1.20 Public Method moveToLast ()

moveToLast[] moves to the last item of the list but returns nothing. I.e. moveToLast[];isLast[] returns True.
setList[list] sets the whole list. I.e. setList[{item1,item2}].

setName[name] sets the name of the container. I.e. setName["my list"].

1.1

Instancevariable curPos

1.2

Instancevariable myList

1.3

Instancevariable myName
class etaOpt : public Object

Class (Notebook: etaOpt.m) (Context: etaOpt')

Inheritance

Object

|   | 2 | etaOpt |

Public Members

2.8 Public Method calcAll () .................................................. 13
2.9 Public Method calcEtaMax () .................................................. 13
2.10 Public Method calcMpp () .................................................. 14
2.11 Public Method export () .................................................. 14
2.12 Public Method generateFilename () ..................................... 14
2.13 Public Method giveBandgapList () .................................... 15
2.14 Public Method giveBandgapRangeList () ................................ 15
2.15 Public Method giveCell () ................................................. 15
2.16 Public Method giveData () .................................................. 16
2.17 Public Method giveEta () .................................................. 16
2.18 Public Method giveLambdaBegin () .................................... 16
2.19 Public Method giveMatrix () ............................................. 16
2.20 Public Method giveMaxValue () ......................................... 17
2.21 Public Method giveMpp () .................................................. 17
### Public Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>giveName()</code></td>
<td></td>
<td>17</td>
</tr>
<tr>
<td><code>giveSpectrum()</code></td>
<td></td>
<td>18</td>
</tr>
<tr>
<td><code>giveSubset()</code></td>
<td></td>
<td>18</td>
</tr>
<tr>
<td><code>giveSubset2Matrix()</code></td>
<td></td>
<td>18</td>
</tr>
<tr>
<td><code>giveTitle()</code></td>
<td></td>
<td>18</td>
</tr>
<tr>
<td><code>loadData()</code></td>
<td></td>
<td>19</td>
</tr>
<tr>
<td><code>plot()</code></td>
<td></td>
<td>19</td>
</tr>
<tr>
<td><code>saveData()</code></td>
<td></td>
<td>19</td>
</tr>
<tr>
<td><code>setBandgapRangeList()</code></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td><code>setCell()</code></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td><code>setLambdaBegin()</code></td>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>

### Protected Members

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>myCell</code></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td><code>myData</code></td>
<td></td>
<td>21</td>
</tr>
<tr>
<td><code>myEta</code></td>
<td></td>
<td>21</td>
</tr>
<tr>
<td><code>myLambdaBegin</code></td>
<td></td>
<td>21</td>
</tr>
<tr>
<td><code>myName</code></td>
<td></td>
<td>21</td>
</tr>
<tr>
<td><code>myRangeList</code></td>
<td></td>
<td>21</td>
</tr>
<tr>
<td><code>mySpectrum</code></td>
<td></td>
<td>21</td>
</tr>
</tbody>
</table>

---

new[] creates an object of type etaOpt. Following options are available (showing default values):
- `name` - sets the name of the simulation. This is used by save and export messages for comments.
- `item.mechStack.new[]` - tells etaOpt which mechStack to use for calculations.
- `spectrum->spectrum.new[]` - tells etaOpt which spectrum to use.
- `lambdaBegin->0` - tells etaOpt where to begin the spectrum of the top cell.
- `bandgapRange->[]` - tells etaOpt for which bandgaps the calculations should be performed. For each junction the list must contain a list with minimum value, maximum value, and step size in eV. I.e. To calculate for tandem the top cell 1.1 - 2 eV every 0.05 eV and bottom cell the range 0.5-1 eV every 0.1 eV following must be typed: `{{1.1,2,0.05},{0.5,1,0.1}}`
2.8

Public Method **calcAll ()**

calcAll[] starts the calculation through the whole range. For calculating the efficiency spectrum, giveTotalPower[] is used. The result can be read with giveData[], graphically displayed with plot[], saved with saveAll[] or exported as plain text for use in other programs like Origin with export[]

2.9

Public Method **calcEtaMax ()**

calcEtaMax[] tries to find the maximum efficiency for a specified range.

Options (given values are defaults):
- bandgapRangeList->[]... This option must be specified, and gives the initial set of bandgaps.
- finestStep->0.01....... if all bandgapsteps are less or equal to this the calculation stops.
- stepsPerInterval->3... how many points should be calculated in each interval. Must be bigger than 3.
- debug->False.......... If set to true some debug information will be printed on the screen during calc.

I.e. for a tandem cell: calcEtaMax[bandgapRangeList->[1,2],[0.4,1.4]], finestStep->0.01, stepsPerInterval->4].

In a first step eta is calculated for all bandgaps defined by step (1.25,1.5,1.75,2) for the top cell (0.4,0.65,0.9,1.15,1.4) for the bottom cell. Than the maximum is calculated. The new range is set to maxvalue +/- interval size. So if 1.4 eV was found in a first step for the top cell the new interval becomes 1.4 +/- 0.25. This is divided in 4 steps.....This iteration continues until the steps size is less or equal to finestStep.
2.10 Public Method `calcMpp()`

`calcMpp[bandgapList]` starts the calculation of the power at maximum powerpoint for a given bandgapList. I.e. for a tandem `calcMpp({2,1})` calculates the power for top cell set to 2 eV bottom cell set to 1 eV.

For calculating the efficiency spectrum, `giveTotalPower[]` is used.

The results can be retrieved as a List `{eta,...}` with `giveMpp[]`.

2.11 Public Method `export()`

`export[]` saves a subset of values as plain text to a file for use in other programs. Options:

- `subset->{}`: a list specifying which subset of data should be retrieved. x and y are placeholders other bandgaps must be set to a fix value. if only x is specified than a line of data will be returned, see also `giveSubset`.
- `value->'eta'`: a string specifying the value which should be retrieved. at the moment only `'eta'` is available. if this option is not used, `value->'eta'` is used.
- `file->`: the filename where to save the file. If this Option is not given a filename is generated via `generateFilename` and saved in the current directory use `Directory[]` to get the current directory.
- `comment->{}`: additional comments for `export2Origin` can be applied.

I.e. `export[subset->[x,y,0.7]]` saves

2.12 Public Method `generateFilename()`

`generateFilename[]` returns a filename based on the most important simulation parameters like cellstructure, spectrum, temperature, junction model, currentmodelling as string. I.e. `(EE)(E)_100xAM1.5d_300K_1D_CM.dat`. Available options:

- `extension->'dat'`: appends the given extension instead of the default `'dat'`, i.e. `extension->'txt'` add-> `''.` appends the given string to the filename without extension. i.e. `generateFilename[add->'muell']->'(E)_am15d_1000x1000W_300K_1D_CM_muell.dat'`
2.13 Public Method giveBandgapList()

giveBandgapList[] returns a table of all bandgap combination for which the calculations should be performed. The range is set by the bandgapRAnge option in new[] or the setBandgapList[] method. giveBandgapPatternString is used to determine the cell structure. For a tandem cell the list looks like this:

```
{{1.8,0.8},{1.9,0.8},{2,0.8},{1.8,0.9}...}
```

2.14 Public Method giveBandgapRangeList()

giveBandgapRangeList[] returns the bandgap range list for which the calculation should be done. I.e. a 'RangeList' in the form of `{2,3,0.05}{1,2,0.1}` means that the bandgap of the first cell should be varied from 2 to 3 eV with stepheights of 0.05 eV the bandgap of the second cell from 1 - 2 eV with a step height of 0.1eV.

2.15 Public Method giveCell()

giveCell[] returns the cell as object.
2.16

Public Method  
giveData ()

giveData[] returns the raw data in the form {{E1,E2,},{eta}},. 

2.17

Public Method  
giveEta ()

giveEta[] returns the efficiency calculated by calcMpp in %.

2.18

Public Method  
giveLambdaBegin ()

giveLambdaBegin[] returns beginning of the spectrum of the top cell in nm.

2.19

Public Method  
giveMatrix ()

giveMatrix[] gives the calculated value in form of a Matrix. I.e. value>‘eta’,subset->{1,2,x,y} gives a matrix of efficiency, where cell1 = 1eV,cell2=2eV and cell3 and cell4 is varried. Options: subset->{}... a list specifying which subset of data should be retrieved. x and y are placeholders other bandgaps must be set to a fix value. if only x is specified than a line of data will be returned. see also giveSubset
value → "eta". A string specifying the value which should be retrieved. At the moment only "eta" is available. If this option is not used, value → "eta" is used. I.e. giveMatrix[subset → {x,y,0.7}] returns a subset of data of a triple cell where the third bandgap = 0.7 eV.

### 2.20

**Public Method** giveMaxValue ()

giveValueMax[value] returns the complete data set for which value becomes maximal. I.e. giveValueMax["eta"] returns the maximum for a given set like this: 

\[
\{\{1.6,1.\},\{46.81\}\}
\]

### 2.21

**Public Method** giveMpp ()

giveMpp[] returns the results of the calculation initiated with calcMpp[].

### 2.22

**Public Method** giveName ()

giveName[] returns the name of the calculation as string.
2.23 Public Method  \texttt{giveSpectrum} \hspace{1em} ()

\texttt{giveSpectrum[]} returns the selected spectrum as object.

2.24 Public Method  \texttt{giveSubset} \hspace{1em} ()

give\texttt{Subset[]} returns a subset of values. Options:
\texttt{subset}→\{\} ... a list specifying which subset of data should be retrieved. \texttt{x} and \texttt{y} are placeholders other bandgaps must be set to a fix value. if only \texttt{x} is specified then a line of data will be returned. see also give\texttt{Subset} 
\texttt{value}→'\texttt{eta}' ... a string specifying the value which should be retrieved. at the moment only '\texttt{eta}' is available. if this option is not used, value->'\texttt{eta}' is used. I.e. give\texttt{Subset[subset}→\{x,y,0.7\} returns a subset of data of a tripple cell where the third bandgap = 0.7 eV.

2.25 Public Method  \texttt{giveSubset2Matrix} \hspace{1em} ()

give\texttt{Subset2Matrix[subset]} converts a subset \{\{E11,E21,E31\},value1\} to a matrix\{\{value1,value2\},\{value10\}\}

2.26 Public Method  \texttt{giveTitle} \hspace{1em} ()

give\texttt{Title[]} returns
2.27

Public Method loadData ()

loadData[] loads back a simulation previously saved with saveData; uses Get[]Options:
file->´´´´. must be specified. loads this file.

2.28

Public Method plot ()

plot[] a contour or a line plot of the calculated data Options:
subset→{ }, see givesubset[]
value→´´eta´´. see givesubset[]
shaded→True. only valid for contour plots, if True the area between to lines is filled, else only lines with values area drawn.
color→True. only valid for contour plots, if True color is used else a greyscale will be used.

2.29

Public Method saveData ()

saveData[] saves the whole data with ´´Save´´, so that it can be loaded back with loadData[]. Options:
file->´´generateFilename[]´´. saves to the file given by this option. If this is option is not specified, saveData uses generateFilename[] to automatically set a filename.
setBandgapRangeList[RangeList] sets the range, for which the calculation should be done. I.e. a `RangeList` in the form of `{2,3,0.05}{1,2,0.1}` means that the bandgap of the first cell should be varied from 2 to 3 eV with step heights of 0.05 eV the bandgap of the second cell from 1 - 2 eV with a step height of 0.1 eV.

setCell[mechStackCell] sets the complete stack of cell to the object of type mechStackCell.

setLambdaBegin[] sets the begining of the spectrum of the top cell in nm.
Instance variable \texttt{myData}

Instance variable \texttt{myEta}

Instance variable \texttt{myLambdaBegin}

Instance variable \texttt{myName}

Instance variable \texttt{myRangeList}

Instance variable \texttt{mySpectrum}
class mechStack : public container

Class (Notebook: etaOpt.m) (Context: etaOpt')

Inheritance

Object

1

container

3

mechStack

Public Members

3.5 Public Method calcMpp () ......................................................... 23
3.6 Public Method giveBandgapPatternString () ........................................ 23
3.7 Public Method givePmpp () ......................................................... 23
3.8 Public Method setBandgap () ..................................................... 24
3.9 Public Method setSpectrumBegin () .............................................. 24
3.10 Public Method setTemperature () ............................................... 24

Protected Members

3.1 Instancevariable curPos .......................................................... 24
3.2 Instancevariable myList ......................................................... 25
3.3 Instancevariable myName .......................................................... 25
new[] creates an object of type mechStack which contains monolithic stacks. As this class is derived from the class container every option of container can be used in addition.

Example: to create a mechanically stacked tandem of two onediode junctions:
```cpp
t mechStack.new[name->``mechanically stacked tandem´´,item->{
    monoStack.new[name->``top monoStack´´,item->{ oneDiodeJunction[name->``top cell´´.bandgap->2] }],
    monoStack.new[name->``bottom monoStack´´,item->{ oneDiodeJunction[name->``bottom cell´´.bandgap->1] }]
}
```

3.5

Public Method  

calcMpp ()
calcMpp[] calculates the maximum power point of all monoStacks at sets Pmpp which can be retrieved with givePmpp[].

3.6

Public Method  

giveBandgapPatternString ()
giveBandgapPatternString[] returns a string containing the structure of the cell. I.e. a monolithical tandem with a single junction under it has a structure of \{\{E1,E2\},\{E3\}\}.

3.7

Public Method  

givePmpp ()
givePmpp[] returns the power at maximum power point previously calculated with calcMpp[] in mW/cm$^2$. 
3.8 Public Method **setBandgap ()**

setBandgap[list] sets the bandgaps of all junctions to the values in list. I.e. setBandgap[{3,2,1}] sets the bandgap of the first cell to 3 eV the second 2 eV...

3.9 Public Method **setSpectrumBegin ()**

setSpectrumBegin[] sets the spectrum and the starting point for calculating Isc of the junctions. Following options can be used (showing default values):
spectrum->..., no default value, an object of type spectrum
lambdaBegin->.. starting point of the spectrum in nm beginning at this wavelength the current integration should be done.

3.10 Public Method **setTemperature ()**

setTemperature[T] sets the Temperature of each junction in monoStack to T in K.

3.1 Instancevariable **curPos**
3.2 Instance variable myList

3.3 Instance variable myName

3.4 Instance variable myPmpp
class `monoStack` : public `container`

Class (Notebook: `etaOpt.m`) (Context: `etaOpt`)

Inheritance

```
Object
   \hline
   \textbf{1} container
   \hline
   \textbf{4} monoStack
```

Public Members

4.9 Public Method `calcIsc()` ................................................................. 27
4.10 Public Method `calcMpp()` ................................................................. 27
4.11 Public Method `giveCurrentMatching()` ................................................ 28
4.12 Public Method `giveImp()` ................................................................. 28
4.13 Public Method `giveIsc()` ....................................................................... 28
4.14 Public Method `givePmp()` ................................................................. 29
4.15 Public Method `giveVmmp()` ............................................................... 29
4.16 Public Method `setBandgap()` .............................................................. 29
4.17 Public Method `setCurrentMatching()` ................................................... 29
4.18 Public Method `setSpectrumBegin()` ...................................................... 30
4.19 Public Method `setTemperature()` ........................................................ 30

This page has been automatically generated with DOC++

DOC++ is © 1995 by Roland Wunderling

Malte Zöckler
Protected Members

4.1 Instancevariable curPos .......................................................... 30
4.2 Instancevariable currentMatching ............................................. 30
4.3 Instancevariable myImpp ......................................................... 31
4.4 Instancevariable myIsc ......................................................... 31
4.5 Instancevariable myList ......................................................... 31
4.6 Instancevariable myName ....................................................... 31
4.7 Instancevariable myPmpp ....................................................... 31
4.8 Instancevariable myVmpp ....................................................... 31

new[] creates an object of type monoStack. Following options are available (default values are shown):
currentMatching->True... if set to true an algorithm which implements semi-transparent top cells is used to achieve current matching
As this class is derived from the class container every option of container can be used in addition.
Example: to create a monolithic tandem of two onediode junctions:
monoStack.new[name->´monolithic tandem´,item->{
oneDiodeJunction[name->´top cell´,bandgap->2],
oneDiodeJunction[name->´bottom cell´,bandgap->1]
}]

4.9

Public Method calcIsc ()

calcIsc[] calculates the shortcircuit current of the stack. If currentMatching is True an algorithm which implements semi-transparent top cells is used to achieve current matching. Otherwise the shortcircuit current of all junction is set to their minimum.

4.10

Public Method calcMpp ()
calcMpp[] calculates the current at maximum power point. Before this method can be used Isc of the junctions must be calculated. To do this use calcIsc[].

I0 is calculated in this method calling calcI0 of the single junctions. Mpp is found by building a function \( P=I^* \text{giveV}(I) \) where \( \text{giveV} \) is a method of the single junction and then setting the first derivative=0.

The values for \( I_{mpp}, P_{mpp} \) and \( V_{mpp} \) are set and can be retrieved by the appropriate give method - i.e. giveImpp[]...

### 4.11

**Public Method**

```cpp
giveCurrentMatching ()
```

giveCurrentMatching[] returns True if currentmatching is turned on for this stack otherwise False.

### 4.12

**Public Method**

```cpp
giveImpp ()
```

giveImpp[] returns the current at maximum power point in mA/cm\(^2\) calculated with calcMpp[]. giveImpp[] returns 0 if calcMpp[] was not called in advance.

### 4.13

**Public Method**

```cpp
giveIsc ()
```

giveIsc[] returns the shortcircuit current in mA/cm\(^2\) calculated with calcIsc[]. giveIsc[] returns 0 if calcIsc[] was not called in advance.
4.14  
Public Method  
givePmpp ()

givePmpp[] returns the power at maximum power point in mW/cm$^2$ calculated with calcMpp[]. givePmpp[] returns 0 if calcMpp[] was not called in advance.

4.15  
Public Method  
giveVmpp ()

giveVmpp[] returns the voltage at maximum power point in mV calculated with calcMpp[]. giveVmpp[] returns 0 if calcMpp[] was not called in advance.

4.16  
Public Method  
setBandgap ()

setBandgap[bandgap] sets the bandgap of junction in eV. I.e. setBandgap[{3,2,1}] sets the bandgap of the first cell to 3 eV the second 2 eV...

4.17  
Public Method  
setCurrentMatching ()

setCurrentMatching[True/False] if parameter is True (False) turns currentmatching on (off).
setSpectrumBegin[spectrum, lambdaBegin] Perform two things:
1. let all junctions know, which spectrum to use for calculating Isc.
2. let all junctions know, where to start integration of the spectrum. Thereby the start point of the topmost junction is set to lambdaBegin. The starting point of each following junction is set to the bandgap of his previous neighbour.

setTemperature[temp] sets the temperature of each junction in the stack to temp in K.

Instancevariable curPos

Instancevariable currentMatching
4.3 Instancevariable myImp

4.4 Instancevariable myIsc

4.5 Instancevariable myList

4.6 Instancevariable myName

4.7 Instancevariable myPmpp

4.8 Instancevariable myVmpp
class oneDiodeJunction : public singleJunction

Inheritance

Object

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>singleJunction</td>
</tr>
<tr>
<td>5</td>
<td>oneDiodeJunction</td>
</tr>
<tr>
<td>7</td>
<td>oneDiodeJunctionSemiEmpirical</td>
</tr>
<tr>
<td>6</td>
<td>oneDiodeJunctionI0FirstTerm</td>
</tr>
</tbody>
</table>

Protected Members

<table>
<thead>
<tr>
<th></th>
<th>Instancevariable</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>myBandgap</td>
<td>.........................</td>
</tr>
<tr>
<td>5.2</td>
<td>myEqeMax</td>
<td>.........................</td>
</tr>
<tr>
<td>5.3</td>
<td>myI0</td>
<td>.........................</td>
</tr>
<tr>
<td>5.4</td>
<td>myIsc</td>
<td>.........................</td>
</tr>
<tr>
<td>5.5</td>
<td>myLambdaBegin</td>
<td>.........................</td>
</tr>
<tr>
<td>5.6</td>
<td>myName</td>
<td>.........................</td>
</tr>
<tr>
<td>5.7</td>
<td>myno</td>
<td>.........................</td>
</tr>
<tr>
<td>5.8</td>
<td>mynu</td>
<td>.........................</td>
</tr>
</tbody>
</table>
### 5.1 Instancevariable myBandgap

### 5.2 Instancevariable myEqeMax

### 5.3 Instancevariable myI0

### 5.4 Instancevariable myIsc

`new[]` creates a single Junction object based on the one diode model. The dark current $I_0$ is calculated with the following formula:

$$
\frac{2 q P_i}{h^2 c^2} \left[ k T (q E_g)^2 - 2 (k T)^2 (q E_g) + 2 (k T)^3 \right] \exp\left[-\frac{q E_g}{k T}\right] 0.1
$$

All options of the abstract parent class `singleJunction` can be used for setup. Only the methods `calcI0`, `calcIsc`, `calcVoc` and `giveV` are overwritten.
5.5
Instance variable myLambdaBegin

5.6
Instance variable myName

5.7
Instance variable myno

5.8
Instance variable mynu

5.9
Instance variable mySpectrum

5.10
Instance variable myTemperature
5.11

Instance variable \texttt{myVoc}
class oneDiodeJunctionI0FirstTerm : public oneDiodeJunction
new[] creates an single Junction object based on the one diode model. The dark current Io is calculated with the following formula:

\[
\frac{2 \cdot q \cdot \pi \cdot (\text{myno}^2 + \text{mynu}^2)}{(h \cdot 3 \cdot c \cdot 2) \cdot (k \cdot T \cdot (q \cdot E_g)^2) \cdot \exp\left(-\frac{q \cdot E_g}{k \cdot T}\right)} \cdot 0.1;
\]

All options of the abstract parentclass `oneDiodeJunction` can be used for setup. only the methods calcIo, calcIsc, calcVoc and giveV are overwritten.

### 6.1

**Instancevariable** myBandgap

### 6.2

**Instancevariable** myEqeMax

### 6.3

**Instancevariable** myI0

### 6.4

**Instancevariable** myIsc
6.5 Instancevariable myLambdaBegin

6.6 Instancevariable myName

6.7 Instancevariable myno

6.8 Instancevariable mynu

6.9 Instancevariable mySpectrum

6.10 Instancevariable myTemperature
6.11

Instance variable myVoc
class oneDiodeJunctionSemiEmpirical : public oneDiodeJunction

Class (Notebook: etaOpt.m) (Context: etaOpt')

Inheritance

Object

singleJunction

oneDiodeJunction

oneDiodeJunctionSemiEmpirical

Public Members

7.13 Public Method giveI0Func () ................................................................. 41
7.14 Public Method setI0Func () ................................................................. 41

Protected Members

7.1 Instancevariable myBandgap ................................................................. 42
7.2 Instancevariable myEqeMax ................................................................. 42
7.3 Instancevariable myI0 ................................................................. 42
7.4 Instancevariable myI0Func ................................................................. 42
7.5 Instancevariable myIsc ................................................................. 42
new[] creates an single Junction object based on the one diode model. The dark current Io is calculated with an empirical formula which can be set with setI0Func.

### 7.13 Public Method giveI0Func ()

giveI0Func[] returns the pure function to calculated I0 with. The pure function is called with 3 slots. 1. slot: bandgap [eV] 2. slot: temperature [K] 3. slot: concentration [suns]. The function returns the current in mA/cm^2. E.g.: func=giveI0Func[];func[1.3,300,100]=

### 7.14 Public Method setI0Func ()

setI0Func[purefunction] sets the function to calculated I0 with. pureFunction must be a pure function with up to 3 slots. 1. slot: bandgap [eV] 2. slot: temperature [K] 3. slot: concentration [suns]. the function must return the current in mA/cm^2 E.g.: setI0Func[(#3 Exp[-#1/(k #3)])&]

This page has been automatically generated with DOC++

DOC++ is ©1995 by Roland Wunderling

Malte Zöckler
7.1 Instance variable myBandgap

7.2 Instance variable myEqeMax

7.3 Instance variable myI0

7.4 Instance variable myI0Func

7.5 Instance variable myIsc

7.6 Instance variable myLambdaBegin
Instancevariable myName

Instancevariable myno

Instancevariable mynu

Instancevariable mySpectrum

Instancevariable myTemperature

Instancevariable myVoc
class singleJunction : public Object

Inheritance

Object

⇒

dsingleJunction

⇒

d oneDiodeJunction

Public Members

8.12 Public Method calcAll () ......................................................... 46
8.13 Public Method calcI0 () ......................................................... 46
8.14 Public Method calcIsc () ......................................................... 46
8.15 Public Method calcVoc () ......................................................... 47
8.16 Public Method giveBandgap () ................................................ 47
8.17 Public Method giveEg () ......................................................... 47
8.18 Public Method giveEqeMax () .................................................. 47
8.19 Public Method giveI0 () ......................................................... 48
8.20 Public Method giveIsc () ......................................................... 48
8.21 Public Method giveLambdaBegin () .......................................... 48
8.22 Public Method giveLambdaG () ................................................ 48
8.23 Public Method giveModelAcronym () ...................................... 49

Class (Notebook: etaOpt.m) (Context: etaOpt')
8.24 Public Method  giveName () ................................................................. 49
8.25 Public Method  giveno () ................................................................. 49
8.26 Public Method  givenu () ................................................................. 49
8.27 Public Method  giveSpectrum () ....................................................... 50
8.28 Public Method  giveTemperature () .................................................. 50
8.29 Public Method  giveV () .................................................................. 50
8.30 Public Method  setBandgap () ........................................................... 50
8.31 Public Method  setEqeMax () ............................................................. 51
8.32 Public Method  setI0 () .................................................................. 51
8.33 Public Method  setIsc () .................................................................. 51
8.34 Public Method  setLambdaG () ............................................................ 51
8.35 Public Method  setName () ................................................................. 52
8.36 Public Method  setno () .................................................................. 52
8.37 Public Method  setnu () .................................................................. 52
8.38 Public Method  setSpectrumBegin () ................................................. 53
8.39 Public Method  setTemperature () ..................................................... 53

Protected Members
8.1 Instancevariable  myBandgap ............................................................... 53
8.2 Instancevariable  myEqeMax ............................................................... 53
8.3 Instancevariable  myI0 ................................................................ 53
8.4 Instancevariable  myIsc ................................................................ 54
8.5 Instancevariable  myLambdaBegin .................................................... 54
8.6 Instancevariable  myName ................................................................. 54
8.7 Instancevariable  myno ...................................................................... 54
8.8 Instancevariable  mynu ...................................................................... 54
8.9 Instancevariable  mySpectrum ........................................................... 54
new[] Creates an abstract class of a singleJunction. Following options can be used:
bandgap->0...... sets the bandgap of the junction in eV
name->''''''''... sets the name of the junction.
temperature->300. sets the temperature of the junction in K
eqeMax->1...... sets the external quantum efficiency of a junction is a squarefunction with \( \text{EQE} = \text{EqeMax} \) for \( E \geq E_g \) and \( \text{EQE} = 0 \) for \( E < E_g \).
no->1........... sets the value of refractive index of the material over the junction. For more information see setno[].
nu->0........... sets the value of refractive index of the material under the junction. For more information see setnu[].

### 8.12

Public Method **calcAll** ()

calcAll[] calculates Isc, I0, Voc of the cell in this order.

### 8.13

Public Method **calcI0** ()

calcIsc[] calculates I0. This method must be defined in child classes.

### 8.14

Public Method **calcIsc** ()

calcIsc[] calculates Isc for a given spectrum. This method must be defined in child classes.
Public Method `calcVoc()`

calcVoc[] calculates Voc. This method must be defined in child classes.

Public Method `giveBandgap()`
giveBandgap[] returns the Bandgap of the Junction in eV (same as giveEg[]).

Public Method `giveEg()`
giveEg[] returns the Bandgap of the Junction in eV (same as giveBandgap[]).

Public Method `giveEqeMax()`
giveEqeMax[] returns the maximum of the eqe. The external quantum efficiency of a junction is a square function with \( \text{EQE} = \text{EqeMax} \) for \( E > E_g \) and \( \text{EQE} = 0 \) for \( E < E_g \).
8.19  
Public Method  \texttt{giveI0()} 

giveI0[] returns \(I_0\) in mA/cm\(^2\). Must be calculated with \texttt{calcI0[]}.

8.20  
Public Method  \texttt{giveIsc()} 

giveIsc[] returns \(I_{sc}\) in mA/cm\(^2\). Must be calculated with \texttt{calcIsc[]} or set with \texttt{setIsc[]}.

8.21  
Public Method  \texttt{giveLambdaBegin()} 

giveLambdaBegin[] returns the beginning of the spectra in nm. LambdaBegin is used to calculate \(I_{sc}\).

8.22  
Public Method  \texttt{giveLambdaG()} 

giveLambdaG[] returns the Bandgap of the junction in nm (see \texttt{giveBandgap[]}).
Public Method  **giveModelAcronym** ()

**giveModelAcronym**[] returns a acronym for the JunctionModel eg. SJ for SIngleJunction, 1D for oneDiodeModel 1D1T for oneDiodeJunctionI0FirstTerm.

Public Method  **giveName** ()

giveName[] returns the name of the junction as String.

Public Method  **giveno** ()

giveno[] returns the refractive index of the material situate over the junction. This factor is used for calculating I0 see setno[].

Public Method  **givenu** ()

givenu[] returns the refractive index of the material situate under the junction. This factor is used for calculating I0 see setnu[].
Public Method  
giveSpectrum ()

giveSpectrum[] returns the spectrum from which Isc can be calculated as object of type spectrum.

Public Method  
giveTemperature ()

giveTemperature[] returns the temperature of the junction in K.

Public Method  
giveV ()

giveV[I] returns the voltage mV for a given current in mA/cm².

Public Method  
setBandgap ()

setBandgap[bandgap] sets the bandgap of junction in eV.
Public Method  **setEqeMax** ()

setEqeMax[EqeMax] assumes the EQE of a junction to be a squarefunction with EQE = EqeMax for E>=Eg and EQE = 0 for E<Eg.

Public Method  **setI0** ()

setI0[I0] sets I0 of the junction in mA/cm^2.

Public Method  **setIsc** ()

setIsc[Isc] sets Isc of the junction in mA/cm^2. This is needed to achieve currentmatching for semi-transparent cells.

Public Method  **setLambdaG** ()

setLambdaG[bandgap] sets the bandgap of junction in nm.
8.35

Public Method **setName ()**

setName[name] sets the name of junction. Name is a string.

8.36

Public Method **setno ()**

setno[value] sets the refractive index of the material situate over the junction
A value of 0 means that no radiation take place on this side
A value of 1 means that all photons with in a cone of sin(theta)<1/n will emerge of the cell - where theta is the angle between the ray and the surface normal
A value of n (where n is the refractive index of the junction itself and typically around 36) means that all photons will emerge of the cell
Default is 1
In the calculation of I0 no and nu are used as a factor (no^2+nu^2).

8.37

Public Method **setnu ()**

setnu[value] sets the refractive index of the material situate under the junction
A value of 0 means that no radiation take place on this side
A value of 1 means that all photons with in a cone of sin(theta)<1/n will emerge of the cell - where theta is the angle between the ray and the surface normal
A value of n (where n is the refractive index of the junction itself and typically around 36) means that all photons will emerge of the cell
Default is 0
In the calculation of I0 no and nu are used as a factor (no^2+nu^2)
8.38 Public Method  `setSpectrumBegin()`

`setSpectrumBegin[spectrum, lambdaBegin]` sets the spectrum, and the beginning of the spectrum. This must be done before calculating `Isc`. `lambdaBegin` is needed because not the whole spectrum is transferred to `Isc`. `spectrum` is an object of type `spectrum`. `lambdaBegin` is in nm.

8.39 Public Method  `setTemperature()`

`setTemperature[temp]` sets the temperature of junction in K.

8.1 Instancevariable `myBandgap`

8.2 Instancevariable `myEqeMax`

8.3 Instancevariable `myI0`
8.4 Instancevariable myIsc

8.5 Instancevariable myLambdaBegin

8.6 Instancevariable myName

8.7 Instancevariable myno

8.8 Instancevariable mynu

8.9 Instancevariable mySpectrum
8.10 Instancevariable myTemperature

8.11 Instancevariable myVoc
class spectrum : public Object

Class (Notebook: etaOpt.m) (Context: etaOpt')

Inheritance

Object

↓

spectrum

Public Members

9.18 Public Method generateBlackBody () ................................................................. 59
9.19 Public Method giveConcentration () ................................................................. 59
9.20 Public Method giveFileName () ........................................................................ 59
9.21 Public Method giveGenFunc () ......................................................................... 60
9.22 Public Method giveGenFuncType () ................................................................... 60
9.23 Public Method giveList () ................................................................................ 60
9.24 Public Method giveName () .............................................................................. 61
9.25 Public Method giveNameList () ....................................................................... 61
9.26 Public Method giveNumberOfPhotons () ............................................................ 61
9.27 Public Method givePower () ............................................................................. 61
9.28 Public Method givePowerOfOneSun () ............................................................... 62
9.29 Public Method givePowerOfOrigin () ................................................................. 62
9.30 Public Method giveSpectrumPath () ................................................................. 62
9.31 Public Method giveTemperature () ................................................................. 63
Public Method
giveTotalPower () .............................................................................. 63
Public Method
giveTotalPowerOfOrigin () ............................................................................. 63
Public Method
giveValue () .......................................................................................... 63
Public Method
giveValueList () .......................................................................................... 64
Public Method
giveValueMax () .......................................................................................... 64
Public Method
giveValueMin () .......................................................................................... 64
Public Method
giveWavelengthList () ................................................................................... 64
Public Method
giveWavelengthMax () ................................................................................... 65
Public Method
giveWavelengthMin () ................................................................................... 65
Public Method
setConcentration () ........................................................................................... 65
Public Method
setGenFunc () .................................................................................................. 65
Public Method
setGenFuncType () ............................................................................................. 66
Public Method
setPowerOfOneSun () .......................................................................................... 66
Public Method
setSpectrum () ...................................................................................................... 66
Public Method
setSpectrumPath () ............................................................................................. 66

Protected Members
Instancevariable
myC ...................................................................................................................... 67
Instancevariable
myFileName ......................................................................................................... 67
Instancevariable
myFkt .................................................................................................................. 67
Instancevariable
myGenFunc .......................................................................................................... 67
Instancevariable
myGenFuncType .................................................................................................. 67
Instancevariable
myLambdaBegin .................................................................................................... 68
Instancevariable
myLambdaEnd ....................................................................................................... 68
Instancevariable
myList .................................................................................................................. 68
Instancevariable
myName ................................................................................................................ 68
Instancevariable
myP0 .................................................................................................................... 68
sp = spectrumnew[] generates a new spectrum object. Special care has been taken for scaling the spectrum. Following parameters are essential:

- \( P_{0,\text{org}} \): total power \([0, \infty]\) of the origin spectrum as loaded from the file or generated for blackbodies (given)
- \( P_{\text{ur,org}} \): integrated spectrum \([\lambda_b, \lambda_e]\) of the origin spectrum as loaded from the file or generated for blackbodies (given)
- \( C \): concentration of the spectrum in suns (to be set)
- \( P_1 \): total power \([0, \infty]\) for the spectrum for one-sun concentration \((C=1)\) (to be set)
- \( P_0 \): total power \([0, \infty]\) of the scaled spectrum
- \( P_{\text{ur}} \): integrated power \([\lambda_b, \lambda_e]\) of the scaled spectrum

Calculation is performed as follows:

\[
P_0 = C \frac{P_1}{P_{0,\text{org}}} \frac{P_{\text{ur,org}}}{P_{\text{ur}}}
\]

All properties of the spectrum can be modified during creation by Options. In the following options with their default values are given:

- spectrumPath>: whichFile["pvsim", "Math/QuellDaten/Spectra"] The default path, from where the spectra files should be loaded
- concentration>: the concentration factor.
- genFuncType>: sets the type for calculating the number of photons see setGenFuncType for more information.
- powerOfOneSun>: power of One Sun in \(\text{W/m}^2\)
- name>: the name for the spectral file including subdirs - this option can only be used if no blackbody spectrum with `{blackbody=True}` is generated. If you want to use your own spectra use `{spectrumPath}` to switch to your directory and look at `pvsim/Math/QuellDaten/Spectra/AM1_5g/iec1000.dat` for an example of data file.
- blackBody>: if set to True a blackbody spectrum is generated and the options for `{generateBlackBody}` are available in addition. (see generateBlackBody[])

A spectrum.new[] with no options generates a blackbody with \(T=6000\) K.
generateBlackBody[] generates a blackbody spectrum with a certain emisivity in a range from lambdaBegin to lambdaEnd with numberOfLambda values in the list. The total spectral power (P0org) is calculated via the T^4 Stefan Boltzmann law. The OneSunPower is set to this value automatically. The parameters are set with options. In the following options with there default values are given:

- temperature->2000..... the temperature of the black body in K
- epsilon->1............ the emissivity of the body (1 for blackbody is default)
- lambdaBegin->1........ begining of the spectral range in nm. 0 < lambdaBegin < lambdaEnd!
- lambdaEnd->10^9....... end of the spectral range in nm
- numberOfLambdas->2.... number of datapoints for the list. All calculations are performed with the analytic expression. Thus this option does not influence any integrated results like power generatedPhotons....
- solidAngle->4 Pi...... sets the solidangle in which the spectrum is emitted. This scales the intensity of the spectrum, and has only effect on the PowerOfOriginSpectrum as the spectrum is once more scaled to fit the OneSunPower
- dLambda->............. optional to ‘`numberOfLambdas’` dLambda can be used, which give the distance between two spectral datapoints in nm. All calculations are performed with the analytic expression. Thus this option does not influence any integrated results like power generatedPhotons....

giveConcentration[] returns the concentration factor.

giveFileName[] returns the name of the loaded file. if a blackbody was generated ‘’’ is returned
giveGenFunc[] returns the function used in giveNumberOfPhotons to calculate the number of photons \texttt{Integrate[GenFunc]}. To use the function do something like \texttt{f=sp.giveGenFunc[]; f[400]}

\subsection{9.22 Public Method \texttt{giveGenFuncType ()}}

giveGenFuncType[type] returns the integration method in giveNumberOfPhotons to calculate the number of photons. type can be one of the following:
1: \texttt{Integrate[\lambda \text{ Interpolation}[[\lambda,E],\text{Order->0}]=\sum \lambda_m E_m \lambda_m and \lambda_m are meanvalues of the interval recommended by ASTM}
2: \texttt{Integrate[\lambda \text{ Interpolation}[[\lambda,E],\text{Order->1}]] possible method}
3: \texttt{Integrate[\text{Interpolation}[[\lambda,E \lambda],\text{Order->1}]] quick and dirty but false}

\subsection{9.23 Public Method \texttt{giveList ()}}

giveList[] returns a list containing wavelength in nm and the spectral density in \{nm,W/(m^2 \mu m)\}, eg. \{{400,1.2223}\},{450,1.432},...}
9.24 Public Method  giveName ()

giveName[] returns the name of the selected spectrum with information on concentration and P0 as string, eg. `am15d.100x1000W`.

9.25 Public Method  giveNameList ()

giveNameList[] gives a list of names of all available spectra, eg. `{AM1.5g/standard.dat, AM0/standard.dat}`.

9.26 Public Method  giveNumberOfPhotons ()

giveNumberOfPhotons[] gives the number of photons of the spectrum. If no options are used, the whole spectrum is used. With the following options one can limit the range for which the number of photons should be calculated:
lambdaBegin -> `Begin of spectra` .. gives the wavelength in nm where to start calculations - default is spectrum begin.
lambdaEnd -> `End of spectra` ...... gives the wavelength in nm where to end calculations - default is spectrum end.

9.27 Public Method  givePower ()

givePower[] returns the integrated power of the spectrum for the used range determined by lambdaBegin and lambdaEnd in W/m² taking into account the
PowerOfOneSun and the concentration factor. Options:
lambdaBegin->”Begin of spectra” gives the wavelength in nm where to start calculations - default is spectrum begin.
lambdaEnd->”End of spectra” gives the wavelength in nm where to end calculations - default is spectrum end.

9.28
Public Method  givePowerOfOneSun ()
givePowerOfOneSun[] returns the total power $[0,\infty]$ for 1 sun concentration of the spectrum in W/m$^2$

9.29
Public Method  givePowerOfOrigin ()
givePowerOfOrigin[] returns the integrated power for spectrum as loaded from the file or generated for blackbody W/m$^2$.

9.30
Public Method  giveSpectrumPath ()
giveSpectrumPath[] give the absolute path to the spectra data files, which can be loaded via setSpectrum. See also setSpectrumPath, giveNameList, giveName.
9.31 Public Method  

giveTemperature ()

giveTemperature[] returns the temperature of the blackbody spectrum in K.

9.32 Public Method  

giveTotalPower ()

giveTotalPower[] returns the total power \([0, \infty]\) of the spectrum in \(W/m^2\) taking into account the PowerOfOneSun and the concentration factor.

9.33 Public Method  

giveTotalPowerOfOrigin ()

giveTotalPowerOfOrigin[] returns the total power \([0, \infty]\) of the origin spectrum as loaded from the file or generated for blackbody in \(W/m^2\).

9.34 Public Method  

giveValue ()

giveValue[\lambda] returns for a given \(\lambda\) [in nm] the spectral density in \(W/(m^2 \mu m)\).
giveValueList[] returns a list containing the spectral density in W/(m²µm), eg. {1.2223,1.432,...}

giveValueMax[] returns the maximum of the spectral density in W/(m²µm), eg. 1.432. For a loaded spectrum the list is used. For blackbody WiensKonstant is used I(λ_max), λ_max=2.897756 10^6/T [nm]

giveValueMin[] returns the minimum of the spectral density in W/(m²µm), eg. 1.2223. For a loaded spectrum the list is used for blackbody returns 0.

giveWavelengthList[] returns a list containing the wavelengths in nm, eg. {450,470...}
Public Method  
giveWavelengthMax ()

giveWavelengthMax[] returns the maximum of the wavelength range in nm, eg. 1200

Public Method  
giveWavelengthMin ()

giveWavelengthMin[] returns the minimum of the wavelength range in nm, eg. 450

Public Method  
setConcentration ()

setConcentration[Concentration] Scales the power density of the spectrum according to the OneSunPower. Default = 1.

Public Method  
setGenFunc ()

setGenFunc[pureFunction] sets the function used in giveNumberOfPhotons to calculated the number of photons Integrate[GenFunc]. purFunction must be a pure function with one slot. E.g. setGenFunc[(100 + #)&]
setGenFuncType[type] sets the Integration Method in giveNumberOfPhotons to calculate the number of photons. type must be one of the following: 1: Integrate[\(\lambda\) Interpolation[{\(\lambda, E\)}, Order->0]}]=\(\sum E_m \lambda_m\). \(E_m\) and \(\lambda_m\) are meanvalues of the interval recommended by ASTM 2: Integrate[\(\lambda\) Interpolation[{\(\lambda, E\)}, Order->1]}] possible method 3: Integrate[Interpolation[{\(\lambda, E, \lambda\)}, Order->1]] quick and dirty but false

setPowerOfOneSun[PowerInWatt] scales the spectrum, so that the integrated powerdensity = PowerInWatt. Default is 1000 W/m\(^2\).

setSpectrum[spectrumName] sets the spectrum. Default is ’´AM1.5g/iec1000.dat´´. Use ’´giveNameList´´ to see a list of all available spectra, or use ’´setSpectrum-Path[]´´ to change to the appropriate directory of spectra.

setSpectrum[absDir] sets the directory to which the spectrumName will be appended. absDir must be absolute and can be in DOS or Unix convention but must be avalaible for the MathKernel.
9.1 Instancevariable myC

9.2 Instancevariable myFileName

9.3 Instancevariable myFkt

9.4 Instancevariable myGenFunc

9.5 Instancevariable myGenFuncType
9.6 Instancevariable myLambdaBegin

9.7 Instancevariable myLambdaEnd

9.8 Instancevariable myList

9.9 Instancevariable myName

9.10 Instancevariable myP0

9.11 Instancevariable myP0Org
9.12  Instancevariable myP1

9.13  Instancevariable myPur

9.14  Instancevariable myPurOrg

9.15  Instancevariable mySolidAngle

9.16  Instancevariable mySpectrumPath

9.17  Instancevariable myT
### Public Functions

#### Names

<table>
<thead>
<tr>
<th>Number</th>
<th>Function</th>
<th>Location</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>connectDirs()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>70</td>
</tr>
<tr>
<td>10.2</td>
<td>dateAndTime()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>71</td>
</tr>
<tr>
<td>10.3</td>
<td>dateTime()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>71</td>
</tr>
<tr>
<td>10.4</td>
<td>dbgPrint()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>72</td>
</tr>
<tr>
<td>10.5</td>
<td>export2Origin()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>72</td>
</tr>
<tr>
<td>10.6</td>
<td>getFilename()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>74</td>
</tr>
<tr>
<td>10.7</td>
<td>getFilenameWithoutExtension()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>74</td>
</tr>
<tr>
<td>10.8</td>
<td>getPath()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>75</td>
</tr>
<tr>
<td>10.9</td>
<td>interpolatingFunctionQ()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>75</td>
</tr>
<tr>
<td>10.10</td>
<td>makeUsage()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>76</td>
</tr>
<tr>
<td>10.11</td>
<td>num()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>76</td>
</tr>
<tr>
<td>10.12</td>
<td>outFkt()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>76</td>
</tr>
<tr>
<td>10.13</td>
<td>outStr()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>77</td>
</tr>
<tr>
<td>10.14</td>
<td>readListWithComment()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>77</td>
</tr>
<tr>
<td>10.15</td>
<td>showStatus()</td>
<td>(Notebook: Hilfsfunktionen.m)</td>
<td>78</td>
</tr>
</tbody>
</table>

#### 10.1 connectDirs()

(Notebook: Hilfsfunktionen.m)

### 10.2 dateAndTime() (Notebook: Hilfsfunktionen.m)

dateAndTime gives a String containing the current date and time including the name of the day and seconds.

### 10.3 dateTime() (Notebook: Hilfsfunktionen.m)

dateTime gives a String containing the current date and time.
10.4

dbgPrint()

Package: ../../Hilfsfunktionen.m || Context: Hilfsfunktionen'

dbgPrint[id,msg] Prints the message msg if id is in Global'debugList.
id should be a string, which can be remembered easily i.e. the class name.
if id is not given the string will be ´´misc´´
debugList is a list of strings. You can use the wildcard * to search for patterns.
i.e. debugList={´´*´´} prints all dbgPrint messages.
if debugList is not defined dbgPrint only takes 0.1 ms to evaluate.

10.5

export2Origin()

Package: ../../Hilfsfunktionen.m || Context: Hilfsfunktionen'

export2Origin[opts] writes out a file in a origin compatible format which can be read into origin using the package ´´ImportSpecial´´. This means that the file begins with a comment section in the form of ´´# variable = value´´ then the data follows.
Options must be specified in the form export2Origin[{´´option1´´->value1,´´option2´´->value2,...}):

´´filename´´-> the filename where to wrote the parfile. This option must be specified.
´´data´´-> contains the data in the form {{x1,y1,z1,...},{x2,y2,z2,...}},...
´´dataType´´-> specifies how the data given by the option ´´data´´ should be wrote.
The value will be added as comment # dataType = ...
dataType can be one of the following:
- "string" the data will be written without any modification as given by data
- any other type the 2d-list of data will be written in matrix form
  \[
  \begin{pmatrix}
  v_1, v_2 \\
  v_3, v_4
  \end{pmatrix}
  \]
  
  - `dataExpLength` \(\rightarrow 2\) if dataType is not "string" this specifies the length of
  the exponent see outStr
  
  - `dataMantLength` \(\rightarrow 8\) if dataType is not "string" this specifies the length of
  the mantis see outStr
  
  - `wrapColumn` \(\rightarrow \)Infinity if this option is specified the comment is wrapped at the specified column.

```
\`comment\` \rightarrow \) if the value is a string, this string will be written
without any
modifications if the value is a list, then it must be in the form
\[
\begin{pmatrix}
\`var1\`, val1 \\
\`var2\`, val2
\end{pmatrix}
\]
Each entry will be formatted in the following way: \# var1 = val1. line wrapping is controlled by
the option `wrapColumn`
```

Following options can be set in origin:

Worksheet options
- wksname = name of worksheet
- wkslabel = label of worksheet
- datatype = type of the data _list_ / matrix
- col1name = name of worksheet column 1
- col1label = label of worksheet column 1

Plot options:
- plotaxistype = type of the plot axis (xy) _linlin_, _loglin_ _linlog_ _loglog_
- plotlinetype = type of plotline _linesymb_, _scatter_ _line_
- plotname = name of the plot window
- plotlabel = label of the plot window
- plotlegend = shall I generate a legend? _yes_, no
- xtitle = title inserted as label right over the plot bold and big size
- xmin = Min. value of the x-axis
- xmax = Max. value of the x-axis
- ytitle = title of the y-axis
- ymin = Min. value of the y-axis
- ymax = Max. value of the y-axis
- ztitle = title of the z-axis
- zmin = Min. value of the z-axis
- zmax = Max. value of the z-axis

For contour-plots in addition to plot options:
- ztitle = title of the z-axis
- zmin = Min. value of the z-axis
- zmax = Max. value of the z-axis
10 Public Functions

zcount = number of colours to be used

10.6

getFilename()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m || Context: Hilfsfunktionen'

getFilename[str] returns the filename of string

10.7

getFilenameWithoutExtension()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m || Context: Hilfsfunktionen'

getFilename[str] returns the filename of string without extension
### 10.8 getPath()

**Description:**

getPath[str] returns the path of string with slash ( or backslash) at the end.

**Package:** `../../Hilfsfunktionen.m`

**Context:** Hilfsfunktionen'

### 10.9 interpolatingFunctionQ()

**Description:**

interpolatingFunctionQ[interpolationsFkt] gibt True zurück wenn die Übergene Funktion eine Interpolationsfunktion ist sonst False

z.B.: interpolatingFunctionQ[f]

**Package:** `../../Hilfsfunktionen.m`

**Context:** Hilfsfunktionen'

### 10.10 makeUsage()

**Description:**

This page has been automatically generated with DOC++

DOC++ is ©1995 by

Malte Zockler

This page has been automatically generated with DOC++
Package: ../../Hilfsfunktionen.m || Context: Hilfsfunktionen'

makeUsage[functionName_String, contextPath_String, usageText_String]; Defines the usage for the given function Name in two contexts:
a) In the current context, so that the function is public.
b) In the Context <Usages'contextPath'functionName>, so that the different usages can be defined for the same methodName. The contextPath usually has the form <className>'.

10.11

num()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m || Context: Hilfsfunktionen'

Gibt den Zahlenwert einer physikalischen Konstante aus dem Package Miscellaneous'PhysicalConstants'. Beispiel: num[ElectronCharge].

10.12

outFkt()

(Notebook: Hilfsfunktionen.m)

Package: ../../Hilfsfunktionen.m || Context: Hilfsfunktionen'

outFkt[value, mantissalength, exponentlength] formatierte Ausgabe mit FixedNumberForm, defaults: mantissalength=8, exponentlength=2
### 10.13 outStr()

Package: ../../Hilfsfunktionen.m  Context: Hilfsfunktionen'

`outStr[value, mantissalength, exponentlength]` formatierte Ausgabe mit `FixedNumberForm`, Rückgabe als String, defaults: `mantissalength=8`, `exponentlength=2`

Wird als `exponentLength 0` angegeben so wird die Angabe des Exponenten weggelassen z.b. `outStr[11.23654, 4, 0]->+11.24`

`mantissalength` steht dann für die Gesamtlänge der Ziffern.

### 10.14 readListWithComment()

Package: ../../Hilfsfunktionen.m  Context: Hilfsfunktionen'

`readListWithComment[file_String, opts]` reads in a file similar to `ReadList` but ignoring any line beginning with `#`.

Options:
- `giveComment` — Default is `False`. Kleiner Tip: mit `//TableForm` sieht man den Kommentar schön formatiert.

### 10.15 showStatus()

(Notebook: Hilfsfunktionen.m)
Package: ../../Hilfsfunctionen.m || Context: Hilfsfunctionen'

showStatus[string] writes String to the StatusArea of a Notebook.
11 Functions sorted by Notebooks

Names

11.1 Package ../..//Hilfsfunktionen.m

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectDirs()</td>
<td>80</td>
</tr>
<tr>
<td>dateAndTime()</td>
<td>80</td>
</tr>
<tr>
<td>dateTime()</td>
<td>80</td>
</tr>
<tr>
<td>dbgPrint()</td>
<td>80</td>
</tr>
<tr>
<td>export2Origin()</td>
<td>81</td>
</tr>
<tr>
<td>getFilename()</td>
<td>81</td>
</tr>
<tr>
<td>getFilenameWithoutExtension()</td>
<td>81</td>
</tr>
<tr>
<td>getPath()</td>
<td>81</td>
</tr>
<tr>
<td>interpolatingFunctionQ()</td>
<td>82</td>
</tr>
<tr>
<td>makeUsage()</td>
<td>82</td>
</tr>
<tr>
<td>num()</td>
<td>82</td>
</tr>
<tr>
<td>outFkt()</td>
<td>82</td>
</tr>
<tr>
<td>outStr()</td>
<td>83</td>
</tr>
<tr>
<td>readListWithComment()</td>
<td>83</td>
</tr>
</tbody>
</table>

This page has been automatically generated with DOC++

© 1995 by Roland Wunderling

Malte Zöckler
11.1.15 \hfill \Ref{showStatus()}

@Man connectDirs()

11.1.2 \hfill \Ref{dateAndTime()}

@Man dateAndTime()

11.1.3 \hfill \Ref{dateTime()}

@Man dateTime()

11.1.4 \hfill \Ref{dbgPrint()}

@Man dbgPrint()
11.1.5
\hfill{}\Ref{export2Origin()}

@Man export2Origin()

11.1.6
\hfill{}\Ref{getFilename()}

@Man getFilename()

11.1.7
\hfill{}\Ref{getFilenameWithoutExtension()}

@Man getFilenameWithoutExtension()

11.1.8
\hfill{}\Ref{getPath()}

@Man getPath()
11.1.9 \hfill \Ref{interpolatingFunctionQ()}

@Man interpolatingFunctionQ()

11.1.10 \hfill \Ref{makeUsage()}

@Man makeUsage()

11.1.11 \hfill \Ref{num()}

@Man num()

11.1.12 \hfill \Ref{outFkt()}

@Man outFkt()
11.1.13
\hfill{}\Ref{outStr()}

@Man outStr()

11.1.14
\hfill{}\Ref{readListWithComment()}

@Man readListWithComment()

11.1.15
\hfill{}\Ref{showStatus()}

@Man showStatus()
Class Graph

1. container

4. monoStack

3. mechStack

2. etaOpt

8. singleJunction

5. oneDiodeJunction

7. oneDiodeJunctionSemiEmpirical

6. oneDiodeJunctionI0FirstTerm
spectrum