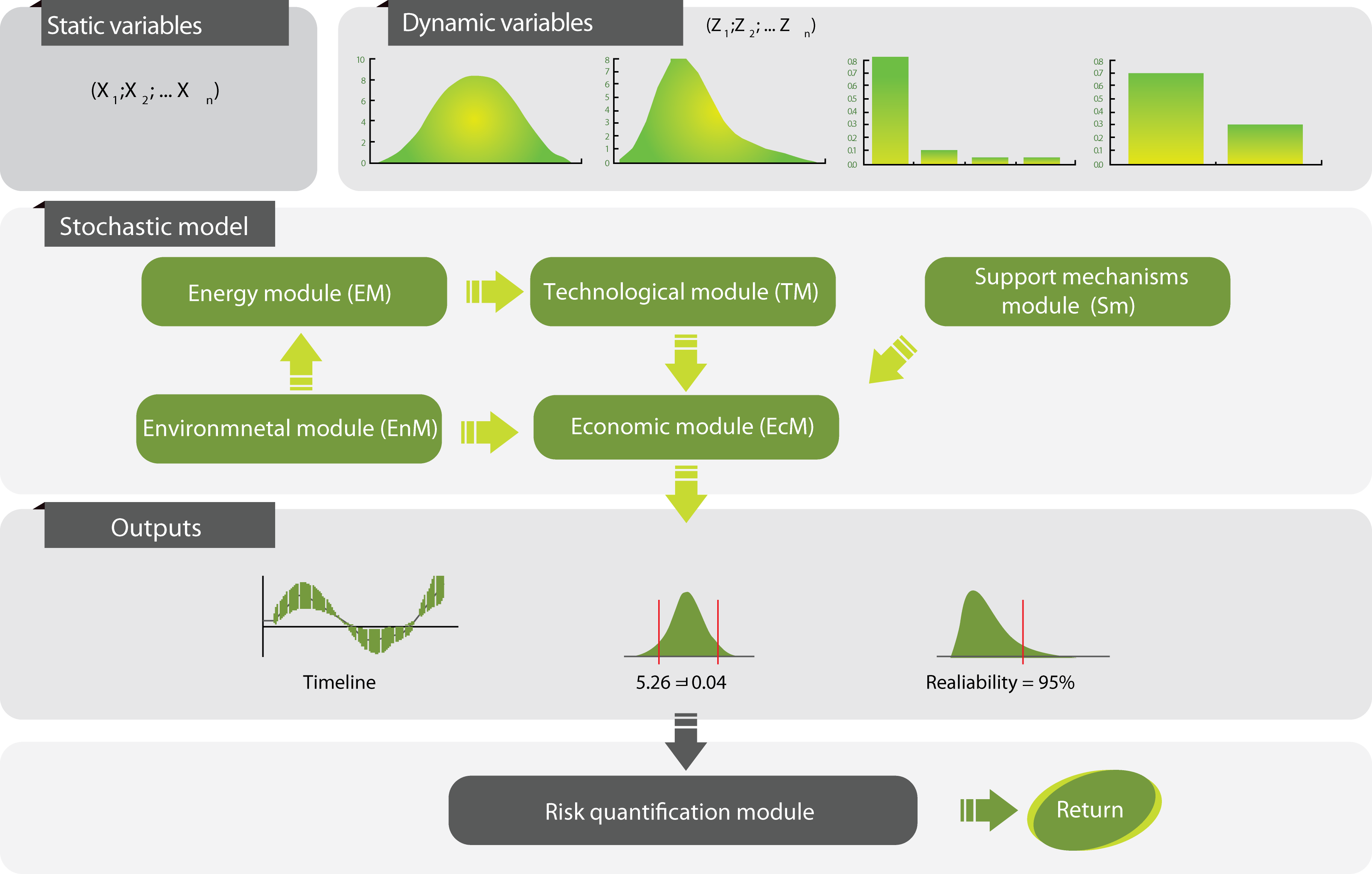
Specification of modules

**The model simulates the financial results & cash flow of an investment in a solar power plant. The aim is to make the model stochastic using Monte Carlo approach. The variability of sun irradiation, technical reliability and variability of economic parameters will be taken into account.**

****

**Table of contents**

1 General functioning of the model 3

2 Energy module (EM) 3

2.1 Module description 3

2.2 Key inputs 3

2.3 Outputs 3

2.4 Definition of functions 3

3 Technology module (TM) 4

3.1 Module description 4

3.2 Key inputs 4

3.3 Outputs 4

3.4 Definition of functions 4

4 Economic module (EcM) 5

4.1 Module description 5

4.2 Key inputs 5

4.3 Outputs 5

4.4 Definition of functions 5

5 Environmental module (EcM) – for now does nothing 7

5.1 Module description 7

5.2 Key inputs 7

5.3 Outputs 7

5.4 Definition of functions 7

6 Support mechanism module (SM) 7

6.1 Module description 7

6.2 Key inputs 7

6.3 Outputs 8

6.4 Definition of functions 8

7 Risk quantification module (RM) 9

7.1 Module description 9

7.2 Key inputs 9

7.3 Outputs 9

7.4 Definition of functions 9

# General functioning of the model

The program works in this sequence:

1. Inputs are read from a text files – for each module a separate text file
2. The following loop is run X cycles (X is set in inputs):
   1. Based on inputs some values are randomly generated by running function initModule for each module
   2. the expected cash flow is generated and written to a database by running EcM.generateISandBS
3. Results are displayed:
   1. RM. outputIRRHistogram
   2. RM. outputIRRXYGraph

# Energy module (EM)

## Module description

Takes care of:

* prediction of daily availability of primary energy (transforming average monthly values into daily data for a sequence of 365x30 days (for 30 years))

## Key inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Input | type | Source | Comment |
| 1 | monthly solar insolation | matrix 12 x 2– float | user inputs | averages for each month in the year |
| 2 | monthly average daily max T | user inputs | averages for each month in the year |
| 3 |  |  |  |  |

## Outputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Output | type | To be used in | Comment |
| 1 | 30 years of daily solar insolation | matrix (30\*365) x 2- float | TM |  |
| 2 |  |  |  |  |

## Definition of functions

### initModule

1. reads input data
2. runs generatePrimaryEnergyAvaialbility

### generatePrimaryEnergyAvaialbility

Parameters: start date

based on monthly daily averages creates daily data

for each day of month: .insulation = average monthly insulation for respective month \* (random factor according to normal distribution)

**Example input data obtained from http://re.jrc.ec.europa.eu/pvgis/apps4/pvest.php#**

**Average Daily Solar Irradiance**

**PVGIS Estimates of average daily profiles**

Results for: December

|  |
| --- |
| Solar radiation database used: PVGIS-CMSAF   Inclination of plane: 36 deg. Orientation (azimuth) of plane: 0 deg. |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Time** | ***G*** | ***Gd*** | ***Gc*** | ***DNI*** | ***DNIc*** | ***Td*** | | 06:07 | 0 | 0 | 0 | 0 | 0 | 3.0 | | 06:22 | 0 | 0 | 0 | 0 | 0 | 3.2 | | 06:37 | 0 | 0 | 0 | 0 | 0 | 3.4 | | 06:52 | 0 | 0 | 0 | 0 | 0 | 3.6 | | 07:07 | 0 | 0 | 0 | 0 | 0 | 3.7 | | 07:22 | 0 | 0 | 0 | 0 | 0 | 3.9 | | 07:37 | 0 | 0 | 0 | 0 | 0 | 4.1 | | 07:52 | 0 | 0 | 0 | 0 | 0 | 4.3 | | 08:07 | 20 | 19 | 15 | 0 | 0 | 4.5 | | 08:22 | 81 | 36 | 157 | 107 | 290 | 4.7 | | 08:37 | 114 | 49 | 228 | 139 | 376 | 4.9 | | 08:52 | 148 | 62 | 301 | 166 | 450 | 5.0 | | 09:07 | 176 | 70 | 364 | 189 | 512 | 5.2 | | 09:22 | 203 | 78 | 424 | 207 | 562 | 5.4 | | 09:37 | 228 | 84 | 480 | 223 | 604 | 5.6 | | 09:52 | 251 | 90 | 531 | 236 | 638 | 5.8 | | 10:07 | 271 | 95 | 578 | 246 | 667 | 6.0 | | 10:22 | 290 | 100 | 620 | 255 | 691 | 6.1 | | 10:37 | 306 | 103 | 657 | 262 | 711 | 6.3 | | 10:52 | 320 | 107 | 689 | 268 | 727 | 6.5 | | 11:07 | 331 | 109 | 716 | 273 | 740 | 6.6 | | 11:22 | 341 | 111 | 738 | 277 | 750 | 6.8 | | 11:37 | 348 | 113 | 754 | 279 | 757 | 6.9 | | 11:52 | 352 | 114 | 765 | 281 | 762 | 7.1 | | 12:07 | 355 | 114 | 771 | 282 | 764 | 7.2 | | 12:22 | 355 | 114 | 771 | 282 | 764 | 7.3 | | 12:37 | 352 | 114 | 765 | 281 | 762 | 7.4 | | 12:52 | 348 | 113 | 754 | 279 | 757 | 7.5 | | 13:07 | 341 | 111 | 738 | 277 | 750 | 7.6 | | 13:22 | 331 | 109 | 716 | 273 | 740 | 7.7 | | 13:37 | 320 | 107 | 689 | 268 | 727 | 7.7 | | 13:52 | 306 | 103 | 657 | 262 | 711 | 7.8 | | 14:07 | 290 | 100 | 620 | 255 | 691 | 7.8 | | 14:22 | 271 | 95 | 578 | 246 | 667 | 7.9 | | 14:37 | 251 | 90 | 531 | 236 | 638 | 7.9 | | 14:52 | 228 | 84 | 480 | 223 | 604 | 7.9 | | 15:07 | 203 | 78 | 424 | 207 | 562 | 7.9 | | 15:22 | 176 | 70 | 364 | 189 | 512 | 7.8 | | 15:37 | 148 | 62 | 301 | 166 | 450 | 7.8 | | 15:52 | 114 | 49 | 228 | 139 | 376 | 7.7 | | 16:07 | 81 | 36 | 157 | 107 | 290 | 7.6 | | 16:22 | 20 | 19 | 15 | 0 | 0 | 7.5 | | 16:37 | 0 | 0 | 0 | 0 | 0 | 7.4 | | 16:52 | 0 | 0 | 0 | 0 | 0 | 7.2 | | 17:07 | 0 | 0 | 0 | 0 | 0 | 7.0 | | 17:22 | 0 | 0 | 0 | 0 | 0 | 6.9 | | 17:37 | 0 | 0 | 0 | 0 | 0 | 6.6 | | 17:52 | 0 | 0 | 0 | 0 | 0 | 6.4 | | 18:07 | 0 | 0 | 0 | 0 | 0 | 6.2 | | 18:22 | 0 | 0 | 0 | 0 | 0 | 5.9 | | 18:37 | 0 | 0 | 0 | 0 | 0 | 5.6 | | 18:52 | 0 | 0 | 0 | 0 | 0 | 5.2 | | 19:07 | 0 | 0 | 0 | 0 | 0 | 4.9 | | 19:22 | 0 | 0 | 0 | 0 | 0 | 4.5 | | 19:37 | 0 | 0 | 0 | 0 | 0 | 4.1 | | 19:52 | 0 | 0 | 0 | 0 | 0 | 3.6 | | | The time shown is local solar time. To find GMT time, add -0.92 hours | | *G*: Global irradiance on a fixed plane (W/m2) | | *Gd*: Diffuse irradiance on a fixed plane (W/m2) | | *Gc*: Global clear-sky irradiance on a fixed plane (W/m2) | | *DNI*: Direct normal irradiance (W/m2) | | *DNIc*: Clear-sky direct normal irradiance (W/m2) | | *Td*: Average daytime temperature profile (°C) | |
|  |
|  |

### outputPrimaryEnergy

Parameters: start\_date; end\_date; resolution

Makes a graph (x-axis displays time; y-axis= displays primary energy). The minimum interval on the x-axis is set by resolution (integer number of days). Values on the y-axis are sum of the energy in the time interval.

# Technology module (TM)

## Module description

Takes care of:

1. conversion of primary energy into electricity
2. reliability of operation

## Key inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Input | type | Source | Comment |
| 1 | 30 years of daily solar insolation | matrix - float | EM |  |
| 2 |  |  |  |  |

## Outputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Output | type | To be used in | Comment |
| 1 | electricity in kWh for each day | matrix for energy with added one row (365\*30)x2 float |  |  |
| 2 |  |  |  |  |

## Definition of functions

### initModule

1. reads inputs
2. runs assembleSystem
3. Runs generateElectiricityProduction

### assembleSystem

generates objects for each solarmodule

number of solar modules = total installed power / solar module power

generates one inverter per X solar modules

groups inverter and respective modules in groups that function together

### generateElectiricityProduction

based on insolation generates electricity production values for each day

produced electricity = insolation \* energyConversionFactor \* isSystemOperational

### isSystemOperational

based on probability (e.g. 99%)that is defined in input data returns 0 or 1

isSystemOperational = isNetworkAvailable \* isSystemUnderMaintenance

### isNetworkAvailable

Availability of system network – user input as % of availability e.g. 99,9%

### isSystemUnderMaintenance

check for maintenace of components (objects Equipment)

### getElectricityProduction

Parameters: dateStart, dateEnd

gives/reads the sum of the electricity produced in the specified period

getElectiricityProduction = sum of electricity in kWh for each day for the selected period

### outputElectricityProduction

Parameters: start\_date; end\_date; resolution

Makes a graph (x-axis displays time; y-axis= displays electricity produced). The minimum interval on the x-axis is set by resolution (integer number of days). Values on the y-axis are sum of the electricity produced in the time interval.

### Object: Equipment

Is the principal class for all equipment. The equipment has following characteristics:

1. state – working, failure, maintenance
2. crucialForSystem – boolean – if crucial then if state is not working then the whole system does not work
3. crucialForGroup – boolean – if crucial then if state is not working then the respective group to which belongs does not work
4. powerEfficiency
5. invesmentCost
6. reliability - % as of probability that it is working

#### Object: equipmentSolarModule

#### Object: equipmentInverter

# Economic module (EcM)

## Module description

Calculates a series of monthly cash flows

## Key inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Input | type | Source | Comment |
| 1 | electricity market price | float | user input |  |
| 2 | expected rate of growth of electricity prices | float | user input |  |
| 3 | values of investment | float | user input |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Outputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Output | type | To be used in | Comment |
| 1 | IS&BS for 30x12 months | matrix | RM |  |
| 2 | IS&BS cash flow for 30 years | matrix | RM |  |

**See: IS & BS. xlsx**

## Definition of functions

### initModule

1. reads input data

### getRevenue

parameters: dateStart; dateEnd

E:= getElectiricityProduction;

sum for all days in the period: E \* price + subsidies (E; date)

### getCosts

parameters: dateStart; dateEnd

sum of costs for all days in the period = DevelopmentCost + OperationalCosts

### getDevelopmentCosts

parameters: dateStart; dateEnd

if the production of electricity has already started then 0

else

if the construction has not yet started then use developmentCostDuringPermitProcurement

else use developmentCostDuringConstruction

developmentCostDuringPermitProcurement & developmentCostDuringConstruction are inputs as EUR/month.

### getOperationalCosts

parameters: dateStart; dateEnd

if the production of electricity has already started then = getInsuranceCosts+getAdministrativeCosts

else 0

### getInsuranceCosts

parameters: dateStart; dateEnd

insuranceCosts= insuranceFeeEquipment \* investmentEquipment

insuranceFeeEquipment is user input

investmentEquipment is userInput

### getAdministrativeCosts

parameters: dateStart; dateEnd

administrativeCosts are user input in EUR/month

### calculateTaxes

EBT in the year \* taxrate – accumulated losses are taken into account

enetered only in december

### getDebtPayment

parameters: dateStart; dateEnd,.....

calculate the payment of the debt principal based on constant annuity repayment - http://en.wikipedia.org/wiki/Fixed\_rate\_mortgage

### calculateInterests

((debt in previous period + debt in current period) /2) \* interest rate \* num of days/365

### calculateFCF

calculates the free cash flow based on IS nad BS

= net earnings + amortisation – investments in long term assests

### generateISandBS

parameters: dateStart; dateEnd

1. generate monthly IS (Income Statements (IS) and BS (Balance Sheet))
2. fill in revenues by running getRevenues for each month
3. fill in costs by running getCosts for each month
4. fill in start positions for assets
5. fill in start position of equity and debt (equity = 30% of investment; longterm debt = 70% of investment)
6. fill in loan by running getDebtRepayment for each period and decrease the debt for the specified amount
7. fill in depreciation by running getDepreciation for each month
8. fill in interest by running calculateInterests
9. calculate EBIT, EBT (earnings before taxes)
10. calculate taxes by running calculateTaxes
11. calculate net income
12. update the balance sheet (with net income, depreciation etc.)
13. if funds in a period are missing – increase short term loans; if excess cash, decrease short term loans

### calculateReturn

calculates IRR and writes it to the database

inputs are the monthly free cash flows (FCF) for the whole project

# Environmental module (EcM) – for now does nothing

## Module description

1. Sets limits to production due to environment protection factors (e.g. for hydro – sets biological minimum)
2. Calculates the obligation at the end of the projetc (e.g. cost of disposal)

## Key inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Input | type | Source | Comment |
| 1 | cost of disposal per kW installed power |  |  |  |
| 2 |  |  |  |  |

## Outputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Output | type | To be used in | Comment |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

## Definition of functions

### initModule

### setDelayConstructionStartForPermitProcurement

sets random value of delay

upper and lower limit are set as user inputs in months

### delayConstructionStartForPermitProcurement

parameter: date

gives true if construction should not be started yet

false if construction should be started

# Support mechanism module (SM)

## Module description

Takes care of:

1. defining the amount of subsidy for investment
2. defining the amount of subsidy for production
3. definition of tax relief/subsidy

## Key inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Input | type | Source | Comment |
| 1 | country | text | user input | used to define the subsidy system |
| 2 | amounts of investment | float | user input |  |
| 3 | amount of subsidy per kWh | float | user input |  |
| 4 |  |  |  |  |

## Outputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Output | type | To be used in | Comment |
| 1 | amount of subsidy for investment | float | economic module |  |
| 2 | amount of subsidy for energy | float |  | for now juts equals to the input – later on should be retrieved from a database based on country |

## Definition of functions

### initModule

sets values for:

1. dateStartOfSubsidy (set to day 1 of the project)
2. dateEndOfSubsidy (set to 12 years after start of the project)
3. susbsidyPerKWh (reading from inputs)

### subsidyInvestment

Input parameters:

1. investment type (land, equipment)
2. amount of investment in EUR
3. nominal installed power in kW

Output: amount of subsidy = 0

### subsidyProduction

Input parameters: energy, date

Output: amount of subsidy = if date < dateEndOfSubsidy then energy \* subsidy/kWh else 0.

### subsidyTax

Input parameters:

Output: amount of tax decrease = 0

# Risk quantification module (RM)

## Module description

Takes care of:

* calculates the variability of return
* calculates VaR of internal rate of return
* calculates the average rate of return

## Key inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Input | type | Source | Comment |
| 1 | multiple series of cash flows for 30x12 months + returns |  | EM |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Outputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Output | type | To be used in | Comment |
| 1 | variability of return | histogram - table |  |  |
| 2. | average rate of return |  |  |  |

## Definition of functions

### initModule

### outputIRRHistogram

outputs histogram of IRR values into a jpg or other format

### outputIRRXYchart

outputs XY chart of Y = IRR values for each cycle and X = consecutive number of the cycle