Operational research for urban solar development

“PV failure detection based on operational time series”

05/12/2023 - Morning

Alexandre Mathieu

Agenda

Curriculum

PV performance model steps

UNIVERSITÉ SAVOIE MONT BLANC

Curriculum
Plan

Today →

Day	Time	Duration	Content
Monday 27/11/2023	9h45-11h15 12h30-14h	1h30 + 1h30	50% Lecture / 50 % Hands-on
Tuesday 05/12/2023	8h-9h30 9h45-11h15	1h30 + 1h30	50% Lecture / 50 % Hands-on
Thursday 07/12/2023	8h-11h 12h45-15h45	6h	25% Lecture / 75 % Project
Monday 11/11/2023	8h-11h 12h30-15h30	6h	10% Lecture / 90 % Project
Friday 22/12/2023	8h-9h30	1h30	100 % Project

UNIVERSITÉ SAVOIE MONT BLANC

Curriculum
Plan

Day	Time	Duration	Content
Monday 27/11/2023	9h45-11h15 12h30-14h	1h30 + 1h30	50% Lecture / 50 % Hands-on
Tuesday 05/12/2023	8h-9h30 9h45-11h15	1h30 + 1h30	50% Lecture / 50 % Hands-on
Thursday 07/12/2023	8h-11h 12h45-15h45	6h	25% Lecture / 75 % Project
Monday 11/11/2023	8h-11h 12h30-15h30	6h	10% Lecture / 90 % Project
Friday 22/12/2023	8h-9h30	1h30	100 % Project

Today →

For next time.
Make groups of 2 for the project.

Agenda

Curriculum

PV performance model steps

Modeling steps

27/11/2023

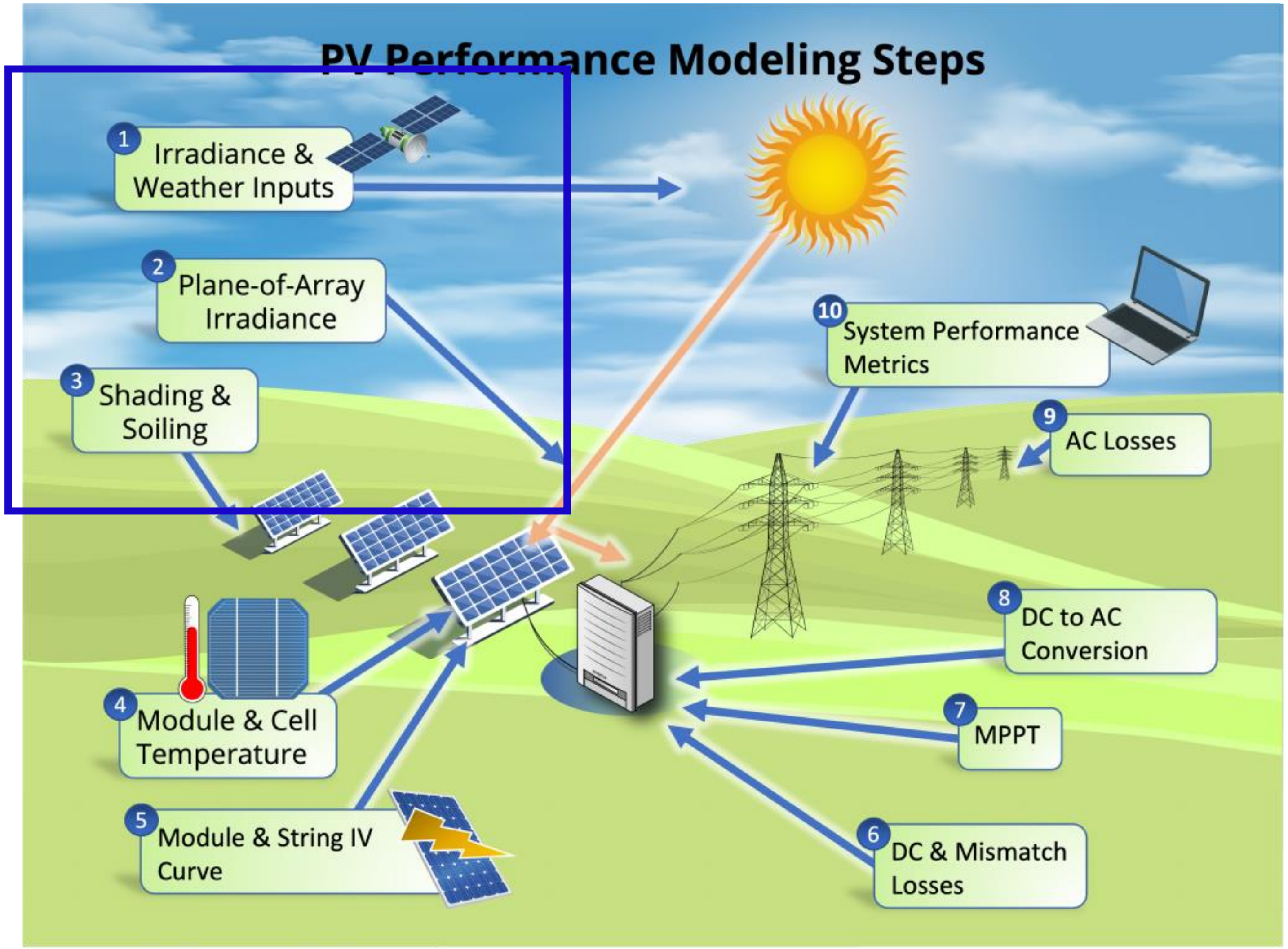


*image from: PVSC48 python tutorial

Modeling steps

27/11/2023

Today

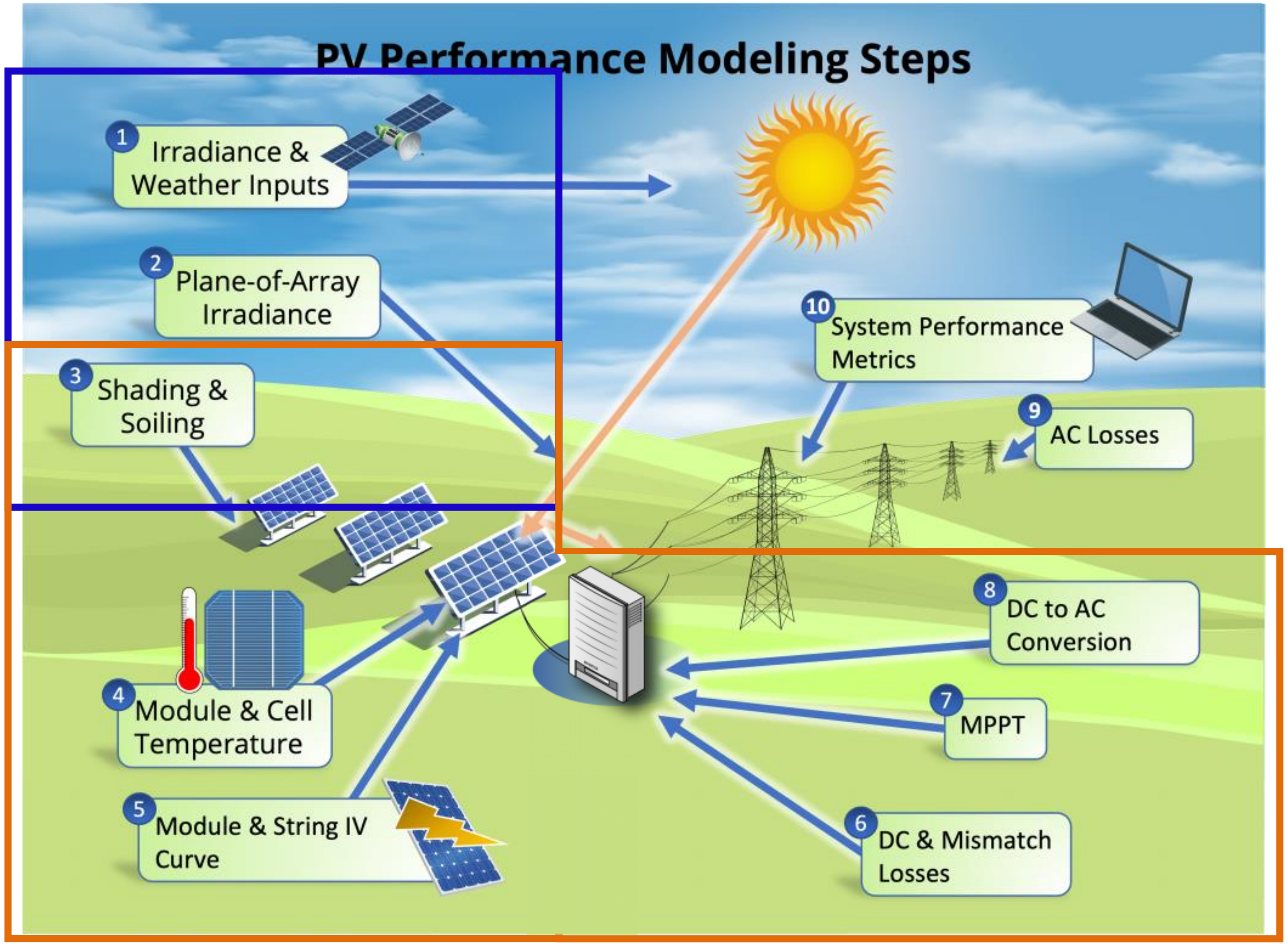


*image from: PVSC48 python tutorial

Modeling steps

Notebook recap 27/11/2023

The notebook is now corrected and can be read online:
https://github.com/AlexandreHugoMathieu/pvfault_detection_solar_academy/blob/master/notebooks/python_intro_poa.ipynb

UNIVERSITÉ SAVOIE MONT BLANC

Modeling steps

Notebook recap 27/11/2023

Python commands 1/2

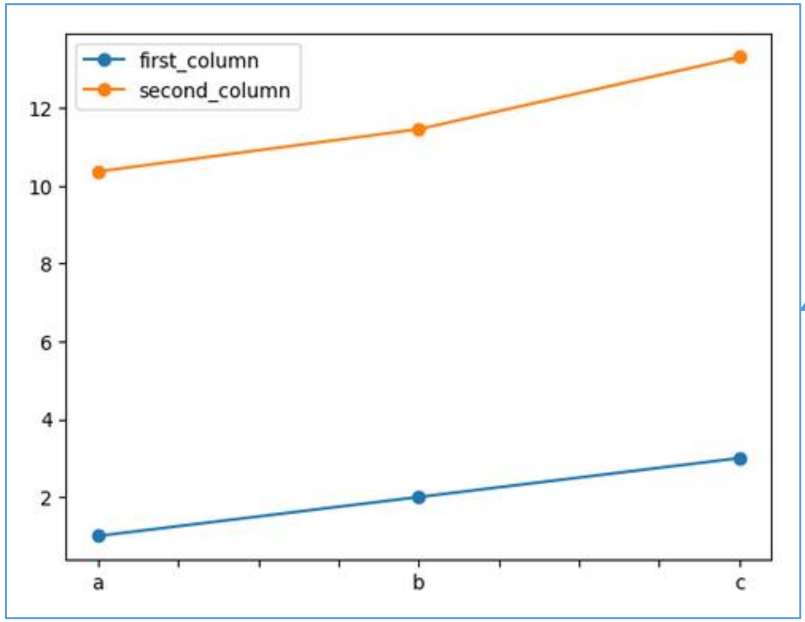
ts

index	values
a	1
b	2
c	3

df

index		
a	1	2
b	2	4
c	3	6

```
import numpy as np # import to your python instance the package "numpy" and rename it "np" (helpful for math calculations)
import pandas as pd # import to your python instance the package "pandas" (helpful for data structure and calculations)

ts = pd.Series([1, 2,3], index=['a','b','c']) # Initiate a pandas serie into variable "ts"
ts2 = ts + ts/2 + np.cos(ts) + np.pi # Make calculate with "ts" and store it into "ts2"
print(ts2) # print serie ts

ts.plot(marker="o") # Make a plot of ts with "o" (circle) marker

df = pd.DataFrame() # Initiate an empty dataframe into variable "df"
df["first_column"] = ts # Store "ts" serie in a column labeled "first_column"
df["second_column"] = ts2 * 2 # Store "ts2" serie in another column labeled "second_column"

df.plot(marker="o") # Make a plot of df with "o" (circle) marker

df.loc["a", :] # Select the entire row with "a" as index
df.loc["a", "first_column"] # Select the value with "a" as index and "first_column" as column
```



Pvlib ref
*William F. Holmgren, Clifford W. Hansen, and Mark A. Mikofski. "pvlib python: a python package for modeling solar energy systems." Journal of Open Source Software, 3(29), 884, (2018). https://doi.org/10.21105/joss.00884

UNIVERSITÉ SAVOIE MONT BLANC

Modeling steps

Notebook recap 27/11/2023

Python commands 2/2

	poa_global	poa_direct	poa_diffuse	poa_sky_diffuse	poa_ground_diffuse
2022-01-01 00:00:00+01:00	NaN	NaN	NaN	NaN	NaN
2022-01-01 01:00:00+01:00	0.000000	0.000000	0.000000	0.000000	0.000000
2022-01-01 02:00:00+01:00	0.000000	0.000000	0.000000	0.000000	0.000000
2022-01-01 03:00:00+01:00	0.000000	0.000000	0.000000	0.000000	0.000000
2022-01-01 04:00:00+01:00	0.000000	0.000000	0.000000	0.000000	0.000000
2022-01-01 05:00:00+01:00	0.000000	0.000000	0.000000	0.000000	0.000000
2022-01-01 06:00:00+01:00	0.000000	0.000000	0.000000	0.000000	0.000000
2022-01-01 07:00:00+01:00	0.000000	0.000000	0.000000	0.000000	0.000000
2022-01-01 08:00:00+01:00	0.593235	0.000000	0.593235	0.589085	0.004150
2022-01-01 09:00:00+01:00	71.788066	46.706296	25.081770	24.724777	0.356993
2022-01-01 10:00:00+01:00	210.546485	150.470436	60.076049	59.167899	0.908150
2022-01-01 11:00:00+01:00	376.976885	313.961064	63.015821	61.519000	1.496821

```
####### Calculate POA, the lazy way #######

from pvlib.irradiance import get_total_irradiance # import the function "get_total_irradiance from pvlib"
# On  another note, pvlib* is a very useful package for PV modeling with plenty of convenient functions, do not hesitate to look it up on the web

beta = 20 # tilt [°]
azimuth = 180 # azimuth [°]
rho = 0.2 # albedo
                                                                                  values
solar_position = pd.read_csv("solarpos_data.csv") # Import the data file "solarpos_data.csv" which contains the sun path (azimuth and elevation)  with datetime index
weather_data = pd.read_csv(DATA_PATH / "sat_data.csv", index_col=0) # Import the data file "sat_data.csv" which irradiance (dni, ghi, dhi) with datetime index

data = get_total_irradiance(beta, azimuth, solar_position["zenith"], solar_position["azimuth"], weather_data["dni"], weather_data["ghi"], weather_data["dhi"], albedo=rho) # Directly apply the isotropic models

print(data.head(24)) # Show the first 24 lines of the DataFrame
```

Pvlib ref
*William F. Holmgren, Clifford W. Hansen, and Mark A. Mikofski. "pvlib python: a python package for modeling solar energy systems." Journal of Open Source Software, 3(29), 884, (2018). https://doi.org/10.21105/joss.00884

Modeling steps

3. Shading / Terrain horizon mask

PVGIS: Website/Online Tool to estimate power production:

- Enables to extract the horizon mask with a Digital Surface Model (DSM).

Go to: https://re.jrc.ec.europa.eu/pvg_tools/en/

Time for some hands-on exercises, Again !

Modeling steps

3. Shading / Terrain horizon mask

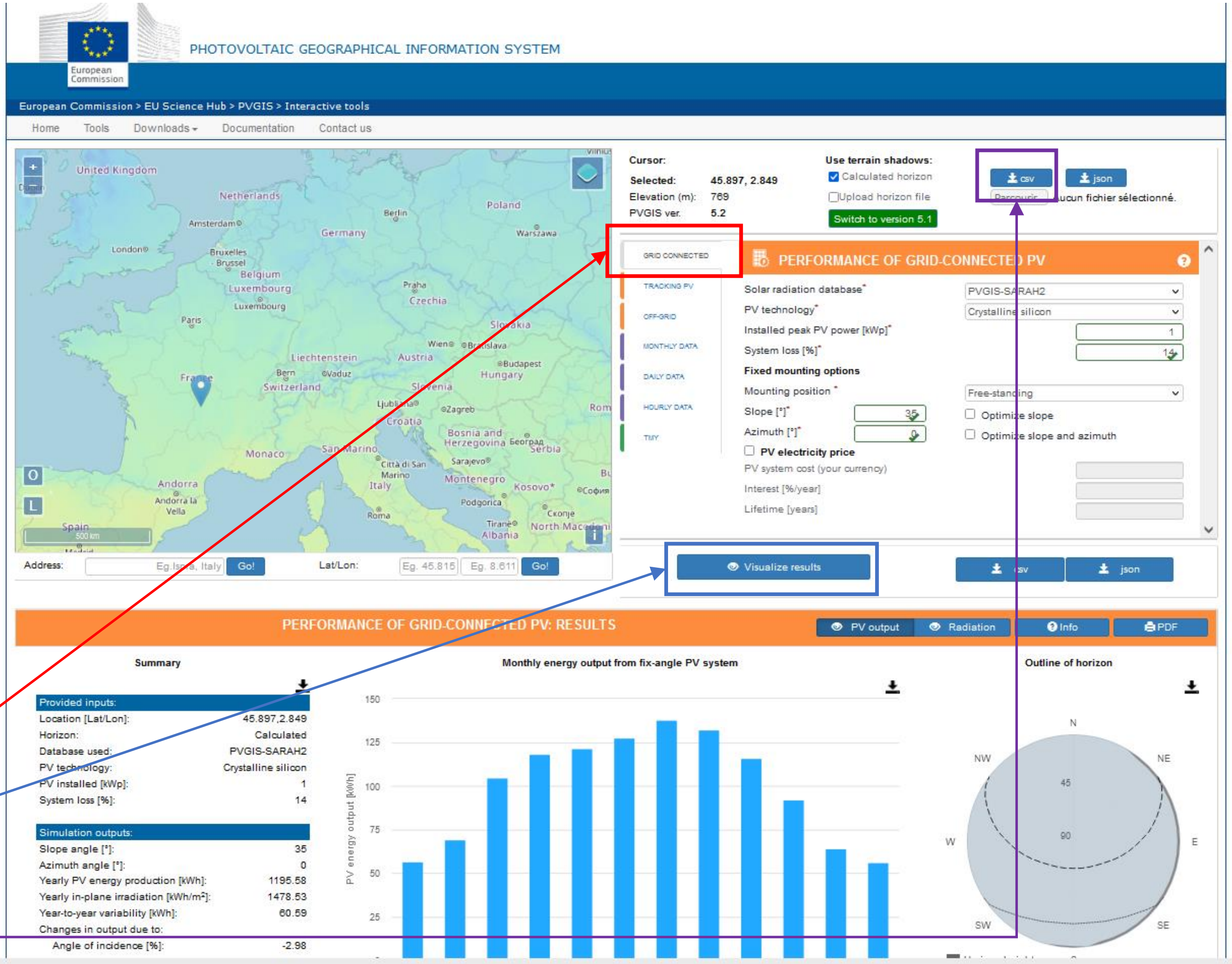
PVGIS: Website/Online Tool to estimate power production: https://re.jrc.ec.europa.eu/pvg_tools/en/

- Enables to extract the horizon mask with a Digital Surface Model (DSM).

Instructions:

1. Generate a simulation on PVGIS
    a. Click on the map on Grenoble and select the « Grid connected tab »
    b. Vizualize
    c. Extract the horizon file in csv format

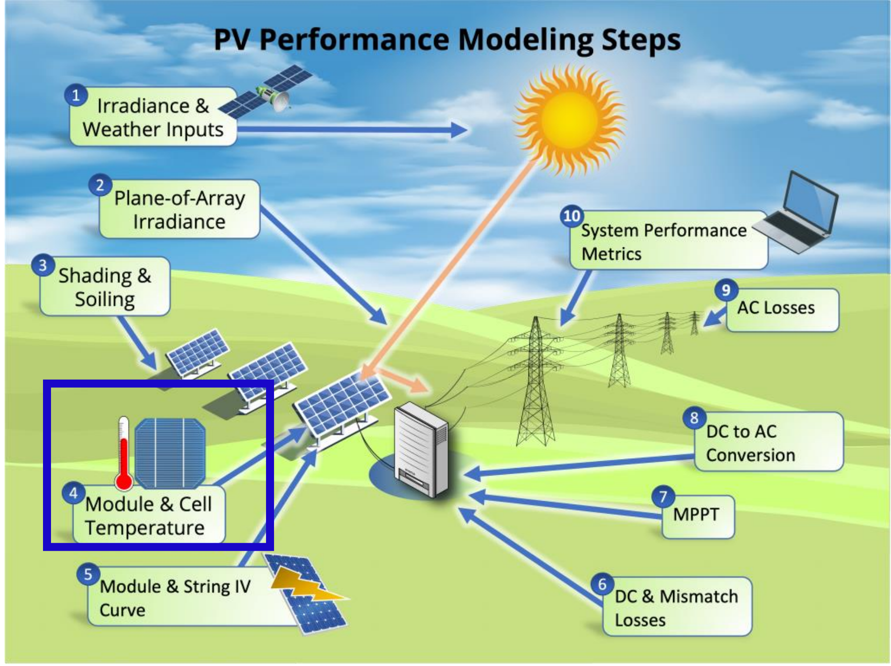
2. Follow the instructions on the jupyter notebook and calculate the modified POA on one year.
https://github.com/AlexandreHugoMathieu/pvfault_detection_solar_academy/blob/master/notebooks/python_intro2_horizon_mask.ipynb



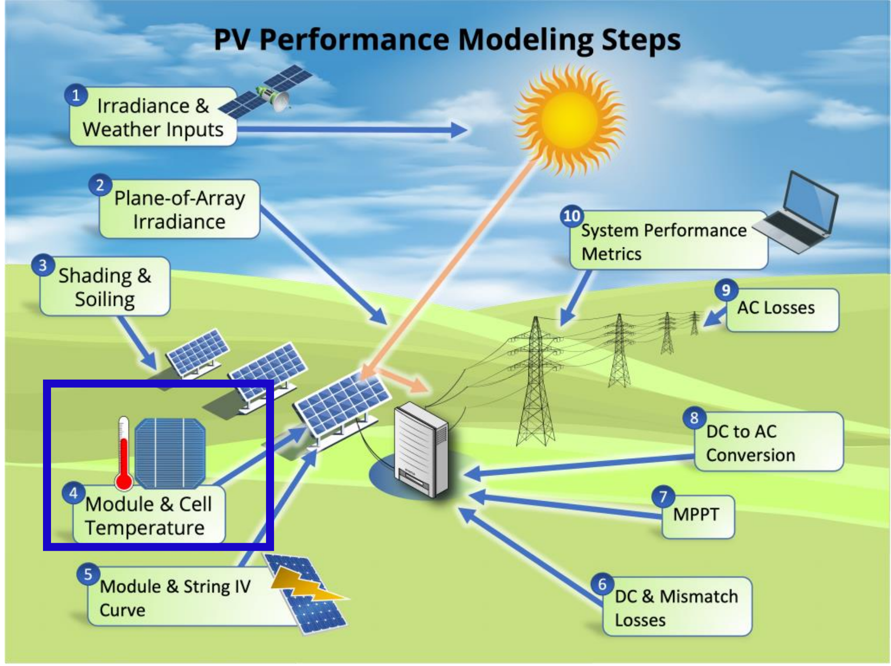
Modeling steps

4. Module and Cell temperature

The hotter a module is, the less efficient it is !



Modeling steps



4. Cell temperature

<u>Ross model:</u>
Model to estimate the cell temperature $\boldsymbol{T_c}$ [°C] as function of ambient temperature and irradiance $\boldsymbol{G_{POA}}$ [W/m²].

$$T_c = T_a + G_{POA} \cdot k_{Ross}$$

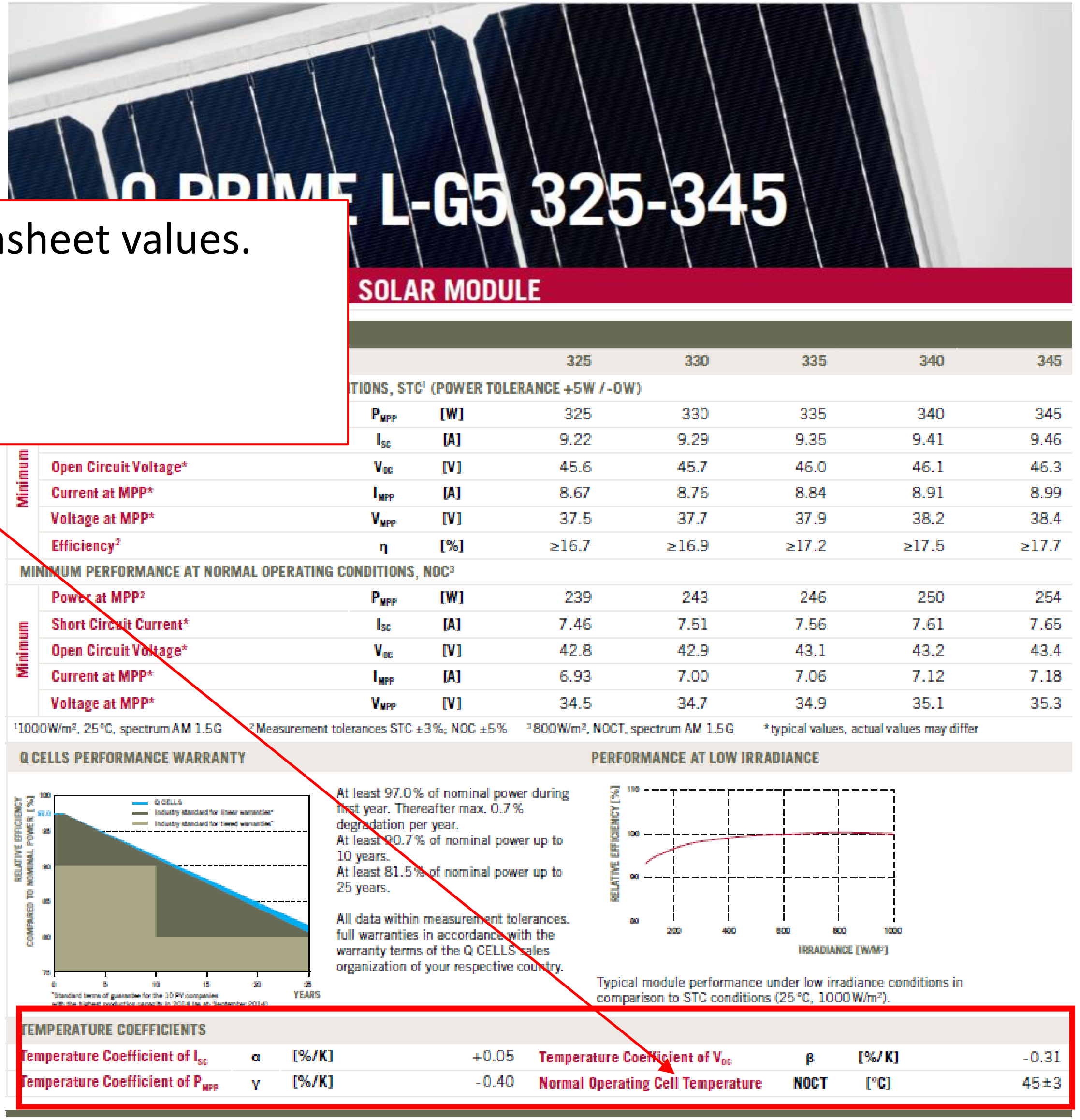
k_{Ross}, typically in the range 0.02-0.05 K/m²/W.

Ross, R. G. Jr., (1981). "Design Techniques for Flat-Plate Photovoltaic Arrays"

UNIVERSITÉ SAVOIE MONT BLANC

Modeling steps

k_{Ross} can be fitted from datasheet values.
NOCT conditions:
G_{POA} = 800 W/m²
T_a = 20°C



				325	330	335	340	345
...TIONS, STC[1] (POWER TOLERANCE +5W / -0W)								
		P_{MPP}	[W]	325	330	335	340	345
		I_{SC}	[A]	9.22	9.29	9.35	9.41	9.46
Minimum	Open Circuit Voltage*	V_{OC}	[V]	45.6	45.7	46.0	46.1	46.3
	Current at MPP*	I_{MPP}	[A]	8.67	8.76	8.84	8.91	8.99
	Voltage at MPP*	V_{MPP}	[V]	37.5	37.7	37.9	38.2	38.4
	Efficiency[2]	η	[%]	≥16.7	≥16.9	≥17.2	≥17.5	≥17.7
MINIMUM PERFORMANCE AT NORMAL OPERATING CONDITIONS, NOC[3]								
Minimum	Power at MPP[2]	P_{MPP}	[W]	239	243	246	250	254
	Short Circuit Current*	I_{SC}	[A]	7.46	7.51	7.56	7.61	7.65
	Open Circuit Voltage*	V_{OC}	[V]	42.8	42.9	43.1	43.2	43.4
	Current at MPP*	I_{MPP}	[A]	6.93	7.00	7.06	7.12	7.18
	Voltage at MPP*	V_{MPP}	[V]	34.5	34.7	34.9	35.1	35.3

[1]1000 W/m², 25°C, spectrum AM 1.5G [2]Measurement tolerances STC ±3%; NOC ±5% [3]800 W/m², NOCT, spectrum AM 1.5G *typical values, actual values may differ

Q CELLS PERFORMANCE WARRANTY

At least 97.0% of nominal power during first year. Thereafter max. 0.7% degradation per year.
At least 90.7% of nominal power up to 10 years.
At least 81.5% of nominal power up to 25 years.

All data within measurement tolerances. full warranties in accordance with the warranty terms of the Q CELLS sales organization of your respective country.

PERFORMANCE AT LOW IRRADIANCE

Typical module performance under low irradiance conditions in comparison to STC conditions (25°C, 1000 W/m²).

TEMPERATURE COEFFICIENTS

Temperature Coefficient of I_{SC}	α	[%/K]	+0.05	Temperature Coefficient of V_{OC}	β	[%/K]	-0.31
Temperature Coefficient of P_{MPP}	γ	[%/K]	-0.40	Normal Operating Cell Temperature	NOCT	[°C]	45±3

4. Cell temperature

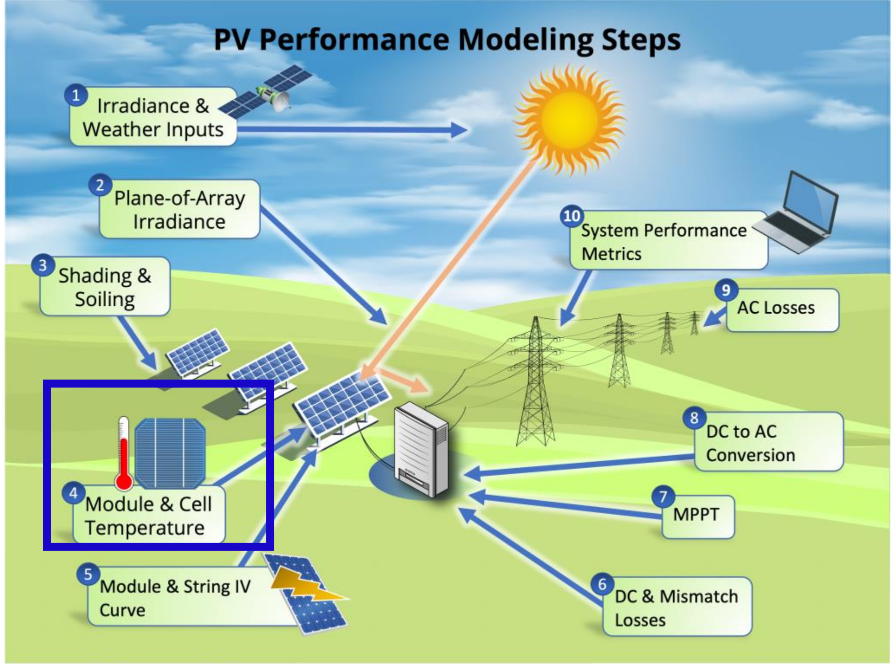
Ross model:
Model to estimate the cell temperature $\boldsymbol{T_c}$ [°C] as function of ambient temperature and irradiance $\boldsymbol{G_{POA}}$ [W/m²].

$$T_c = T_a + G_{POA} \cdot k_{Ross}$$

k_{Ross}, typically in the range 0.02-0.05 K/m²/W.

Ross, R. G. Jr., (1981). “Design Techniques for Flat-Plate Photovoltaic Arrays”

Modeling steps



4. Cell temperature

Faiman model:

Model to estimate the cell temperature $\boldsymbol{T_c}$ [°C] as function of ambient temperature and irradiance $\boldsymbol{G_{POA}}$ [W/m²] AND wind WS $[\frac{m}{s}]$.

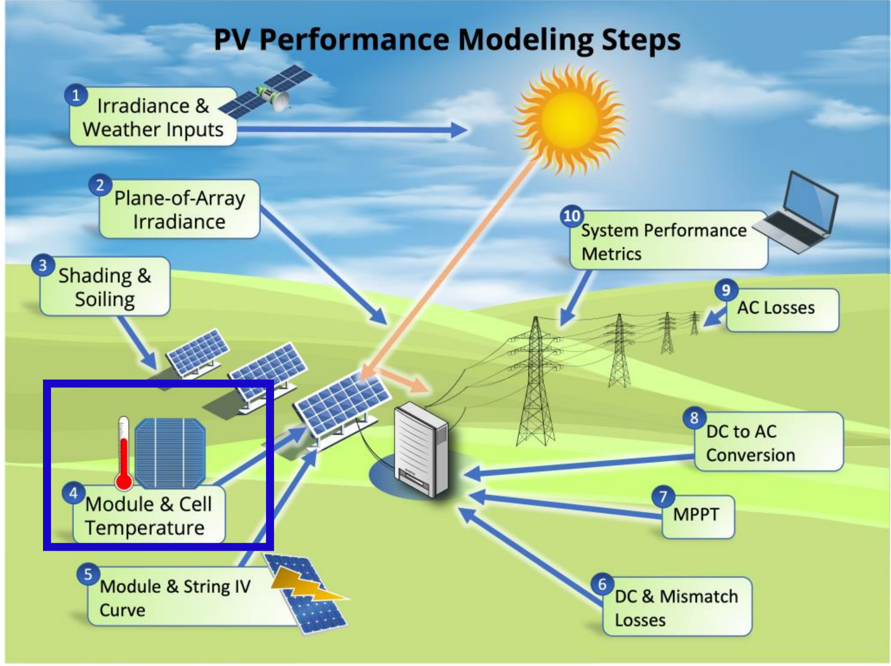
$$T_m = T_a + \frac{G_{POA}}{U_0 + U_1 \cdot WS}$$

U_0 is the constant heat transfer component $[\frac{W}{Km^2}]$

U_1 is the convective heat transfer component$[\frac{W}{Km^2(\frac{m}{s})}]$

Faiman, D. (2008). Assessing the outdoor operating temperature of photovoltaic modules.

Modeling steps



4. Cell temperature

<u>Faiman model:</u>
Model to estimate the cell temperature $\boldsymbol{T_c}$ [°C] as function of ambient temperature and irradiance $\boldsymbol{G_{POA}}$ [W/m²] AND wind WS $[\frac{m}{s}]$.

$$T_m = T_a + \frac{G_{POA}}{U_0 + U_1 \cdot WS}$$

In some cases, $T_c \simeq T_m$ can be assumed
Between T_c and T_m, only few degrees of difference

U_0 is the constant heat transfer component $[\frac{W}{Km^2}]$

U_1 is the convective heat transfer component$[\frac{W}{Km^2(\frac{m}{s})}]$

Faiman, D. (2008). Assessing the outdoor operating temperature of photovoltaic modules.

Modeling steps

4. Cell temperature

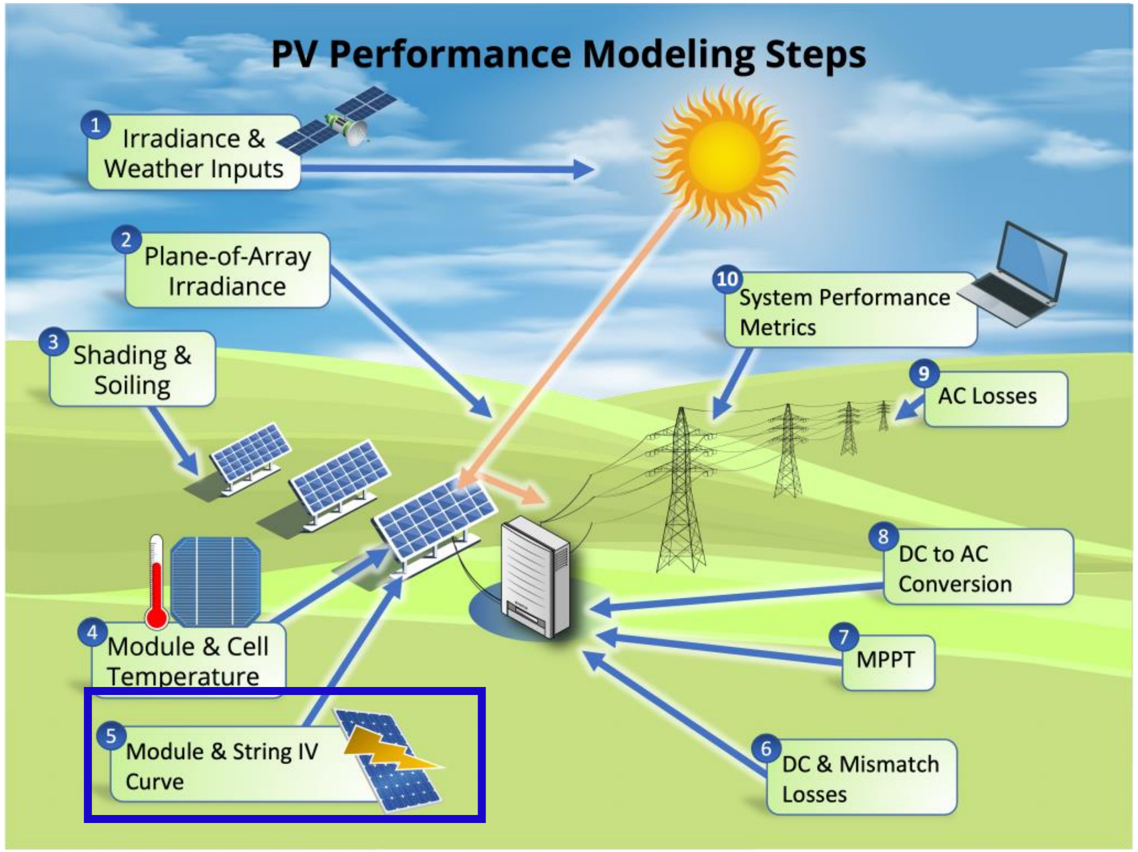
Time for some
hands-on exercises !

Use the following notebook:
https://github.com/AlexandreHugoMathieu/pvfault_detection_solar_academy/blob/master/notebooks/dc_power_estimation.ipynb

Follow the python tutorial and estimate the cell temperature for one year.

Modeling steps

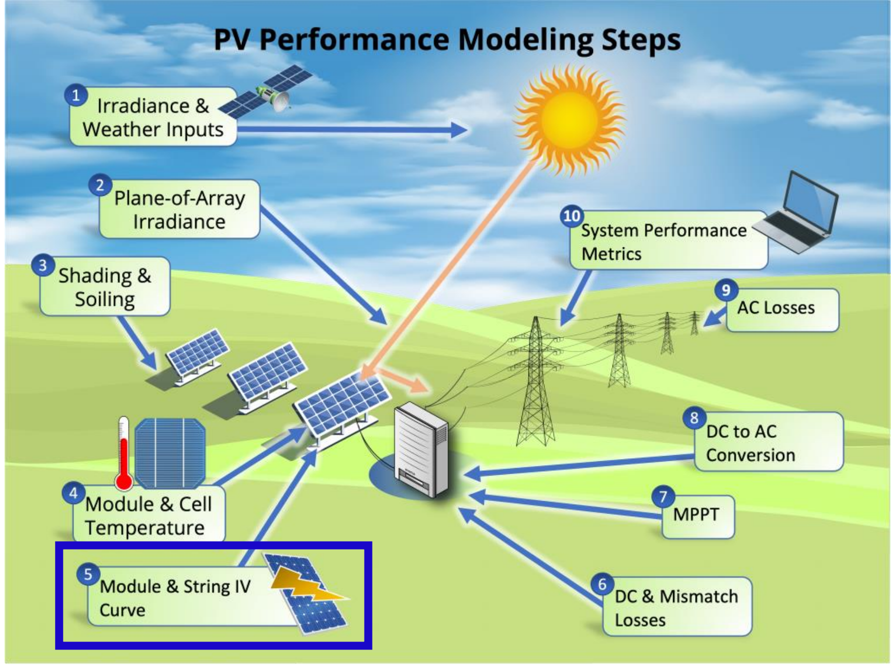
5. Module and String IV Curve

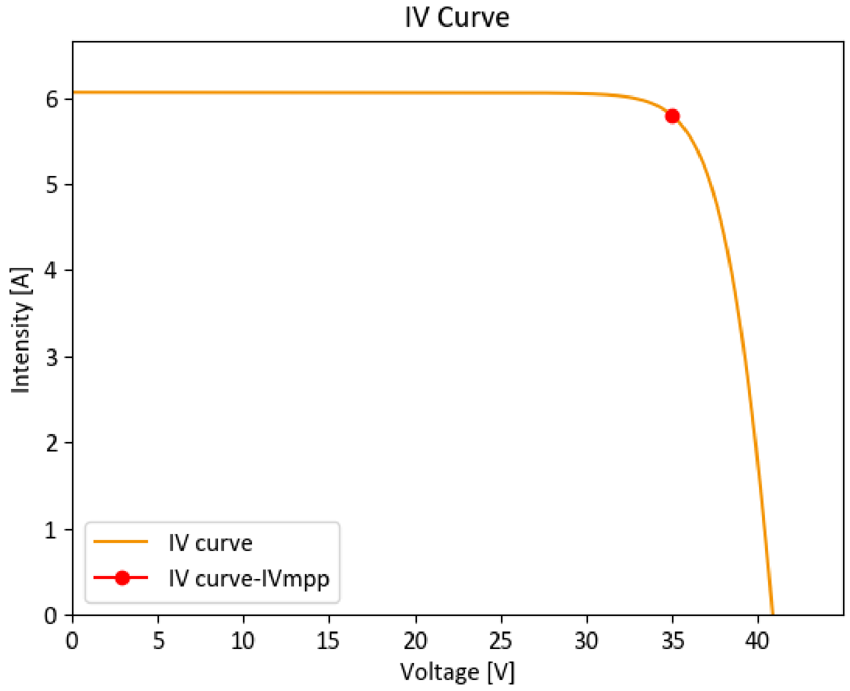


Modeling steps

5. Module and String IV Curve

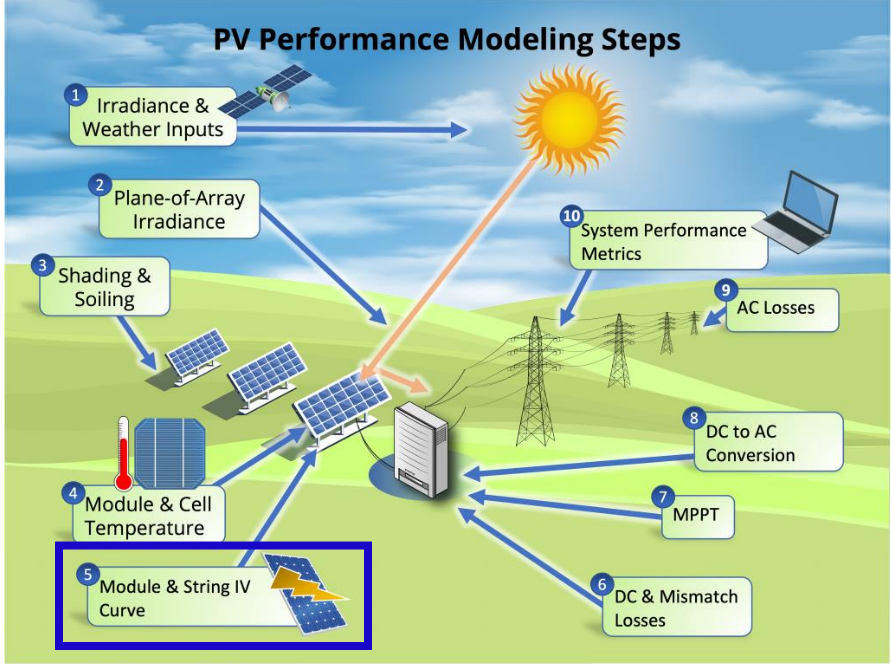
For a fixed irradiance and module temperature, the PV module has its I, current which depends on V, voltage and it can take many operating points.





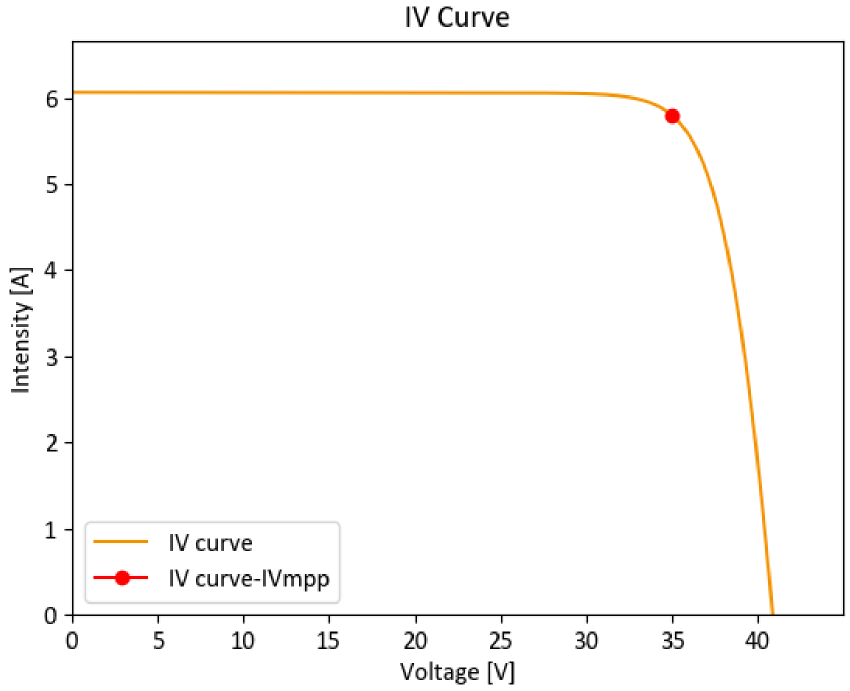
UNIVERSITÉ SAVOIE MONT BLANC

Modeling steps

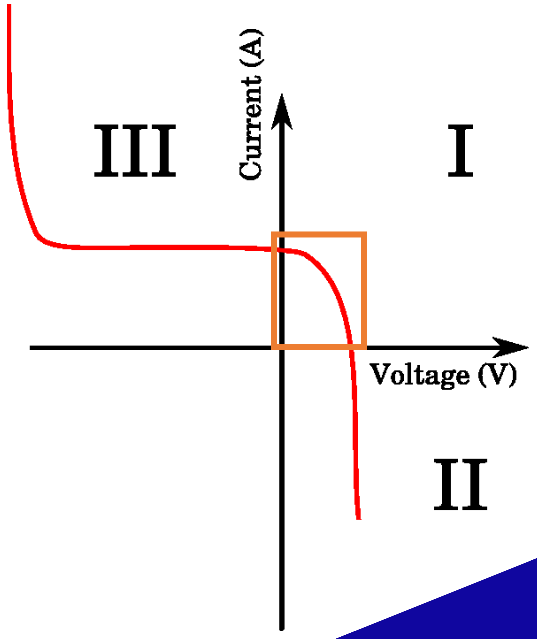


5. Module and String IV Curve

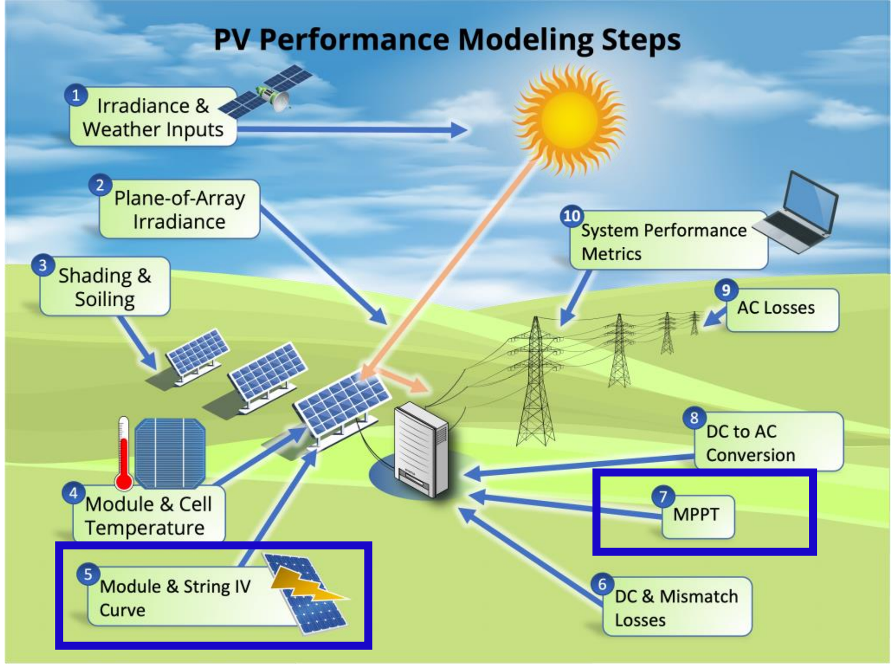
For a fixed irradiance and module temperature, the PV module has its I, current which depends on V, voltage and it can take many operating points.



In reality, the IV characteristics go out of the 1st quadrant and the module can potentially consume power.



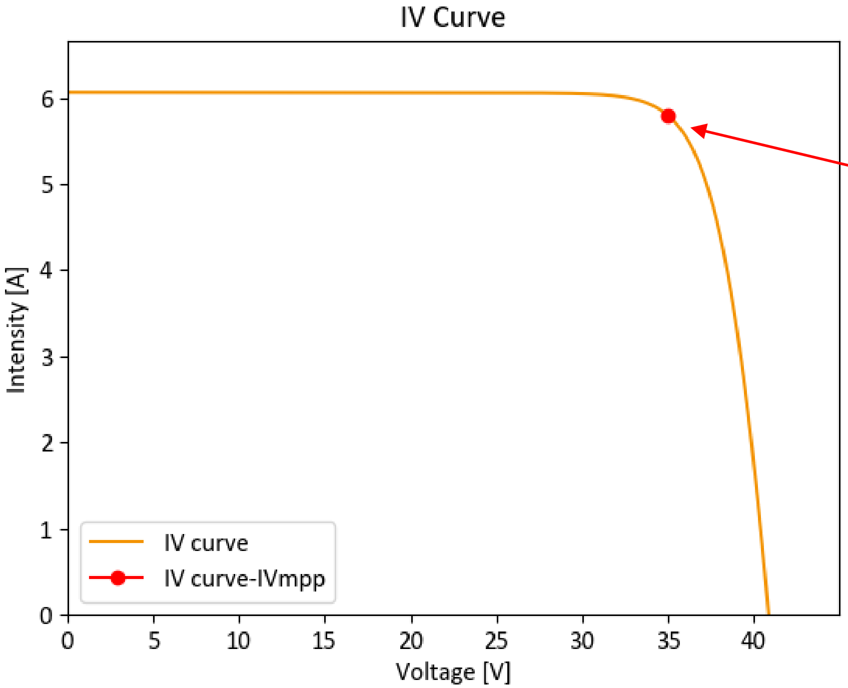
UNIVERSITÉ SAVOIE MONT BLANC



Modeling steps

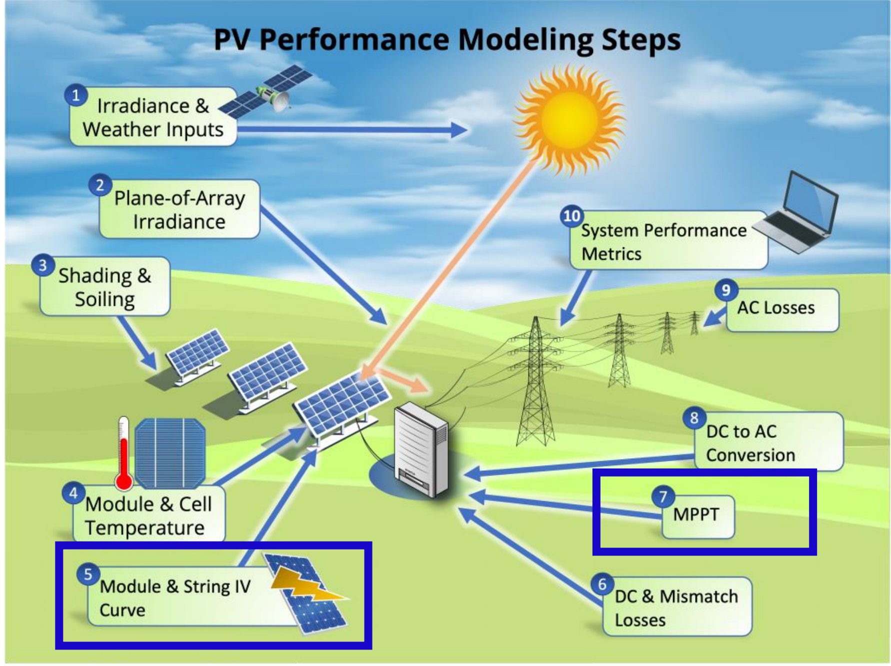
5. Module and String IV Curve

For a fixed irradiance and module temperature, the PV module has its I, current which depends on V, voltage and it can take many operating points



Then, the inverter is constantly searching for the operating point which maximizes the power MPP: Maximum Power Point.

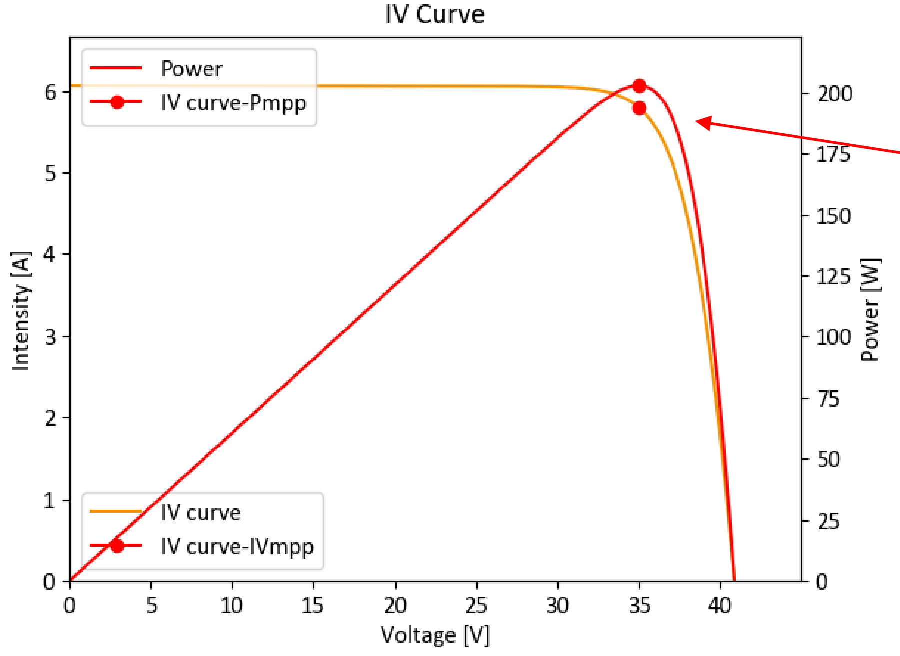
UNIVERSITÉ SAVOIE MONT BLANC



Modeling steps

5. Module and String IV Curve

For a fixed irradiance and module temperature, the PV module has its I, current which depends on V, voltage and it can take many operating points.



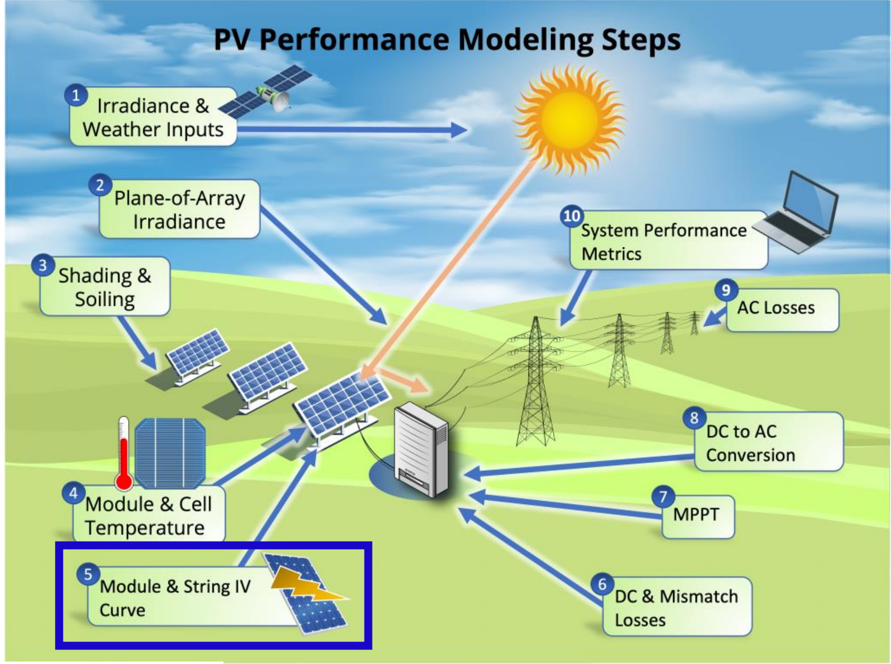
Then, the inverter is constantly searching for the operating. point which maximizes the power MPP: Maximum Power Point.

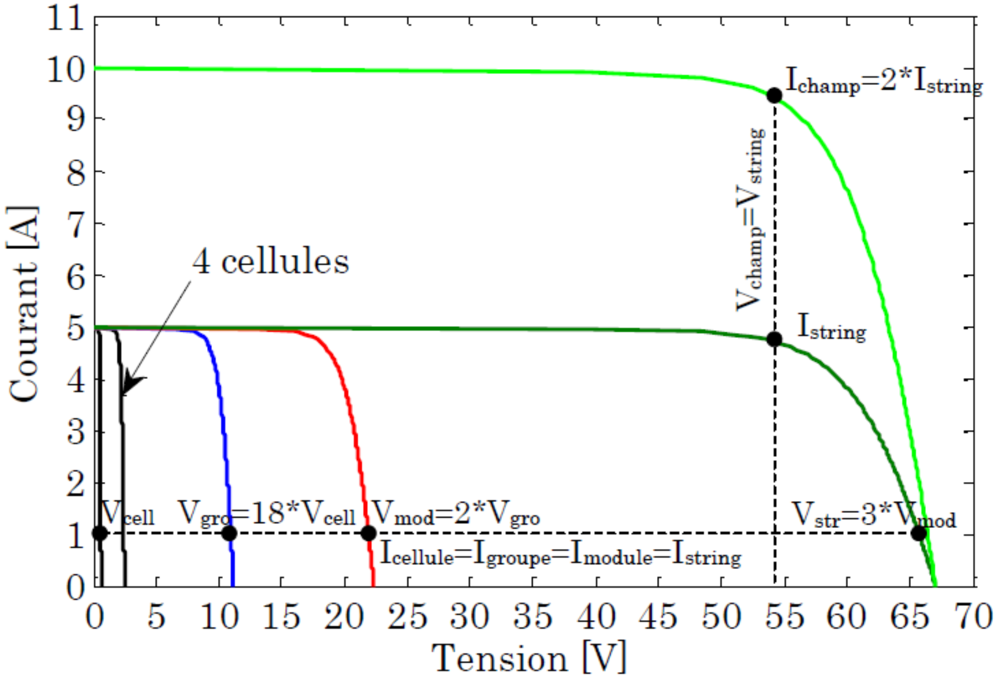
Especially, it changes the voltage with the MPP-Tracker (MPPT) to maximize power.

Modeling steps

5. Module and String IV Curve

By the way… the IV curves can be summed up when the modules are connected in series or parallel! The inverter, then, maximizes the power of the PV array IV curve.





Bun, L.. “Détection et localisation de défauts pour un système PV.” (2011).

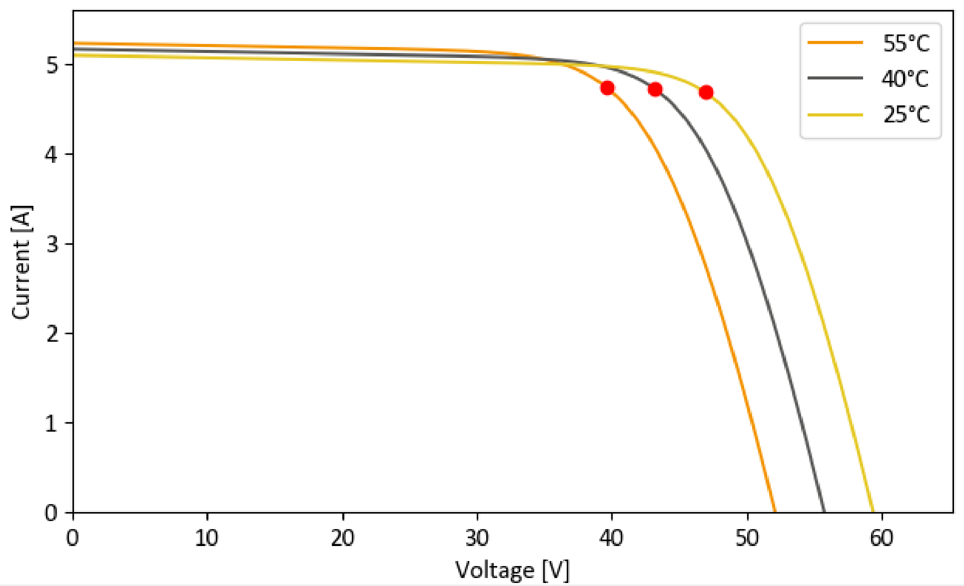
Modeling steps

5. Module and String IV Curve

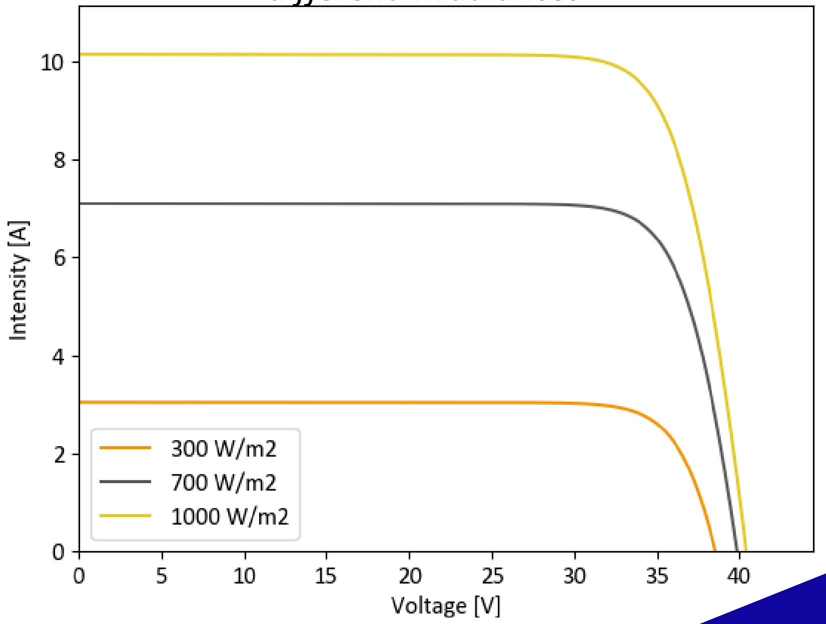
The IV curves' dependencies:

- Higher cell temperatures mostly decrease the voltage
- Higher irradiance level mostly increase the current

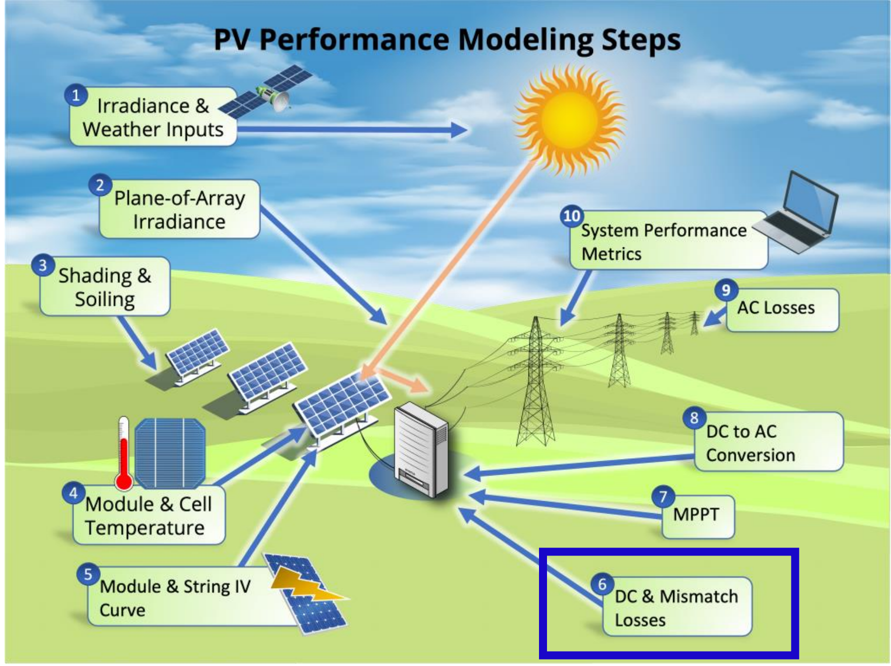
Example of I-V curves as function of different module temperature



Example of I-V curves as function of different irradiances



Modeling steps



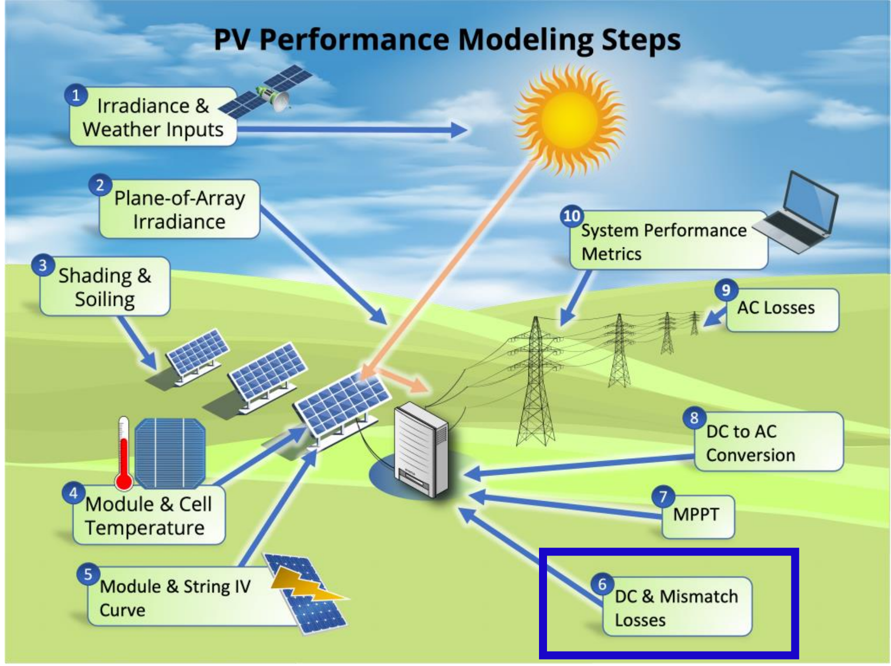
6. DC & Mismatch Losses

Not the focus of this class. However, keep in mind that:

- **DC wiring losses** are around 0.5%-2%.
- **Mismatch losses** refer to the fact that PV modules have different IV curves and this can entail significant losses.

UNIVERSITÉ SAVOIE MONT BLANC

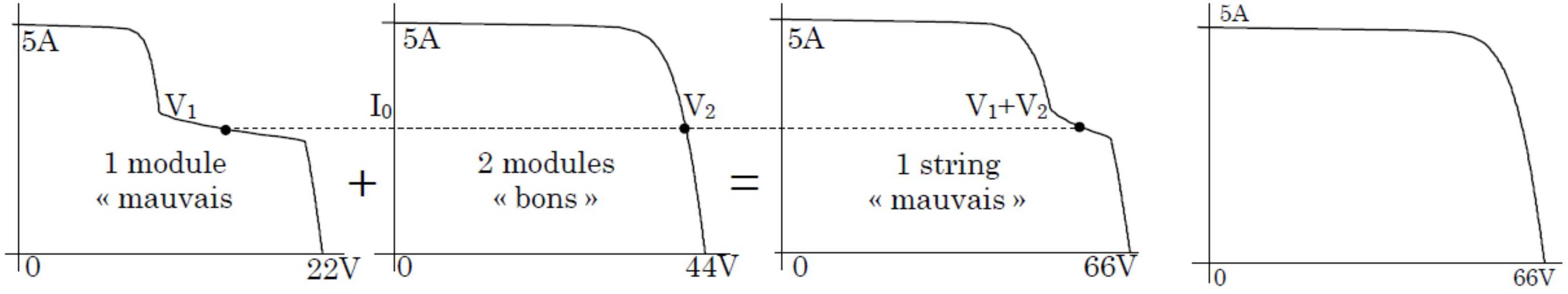
Modeling steps



6. DC & Mismatch Losses

Not the focus of this class. However, keep in mind that:

- **DC wiring losses** are around between 0.5% and 2%.
- **Mismatch losses** refers to the fact that PV modules have different IV curves and this can entail significant losses.
  For instance, if one of them has a very degraded IV curve (shading or other), it can significantly degrade the IV curve at the array level.

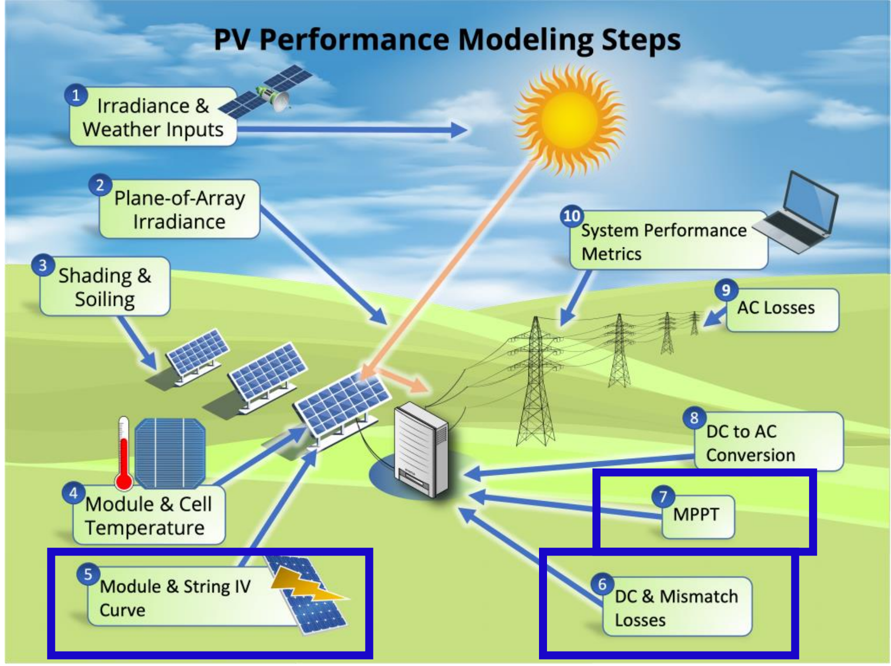


(a) string « mauvais » (b) string « bon »

Bun, L.. "Détection et localisation de défauts pour un système PV." (2011).

UNIVERSITÉ SAVOIE MONT BLANC

Modeling steps



5./6./7. Power model

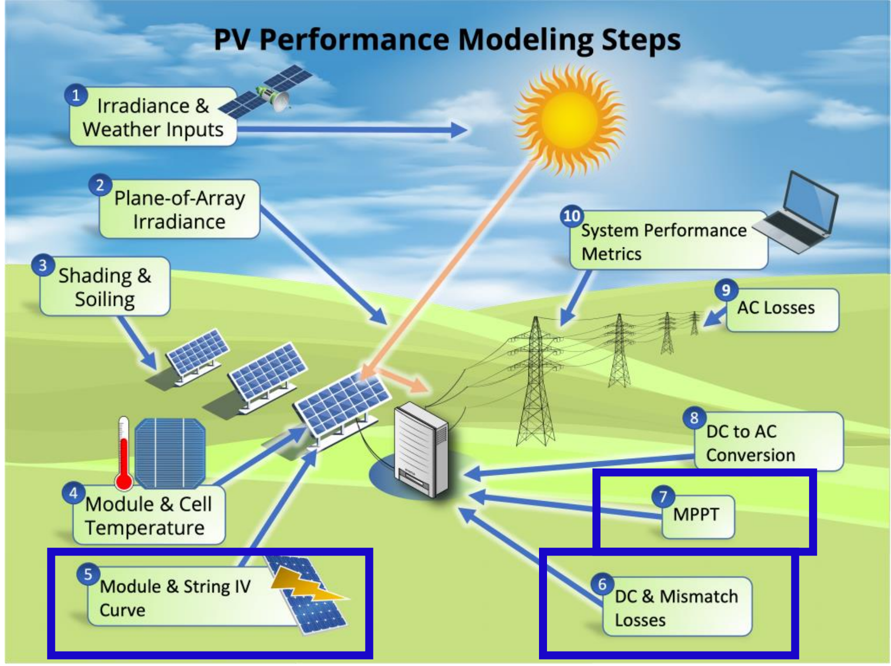
<u>Constant efficiency model:</u>

$$P_{dc} = \eta \cdot G_{POA} \cdot A$$

With:

- P_{dc}, DC power in [W]
- η efficiency around 20% (from datasheet)
- G_{POA} the irradiance in the plane of array [W/m²]
- A, the PV installation area [m2]

Modeling steps



5./6./7. Power model

<u>Constant efficiency model:</u>

$$P_{dc} = \eta \cdot G_{POA} \cdot A$$

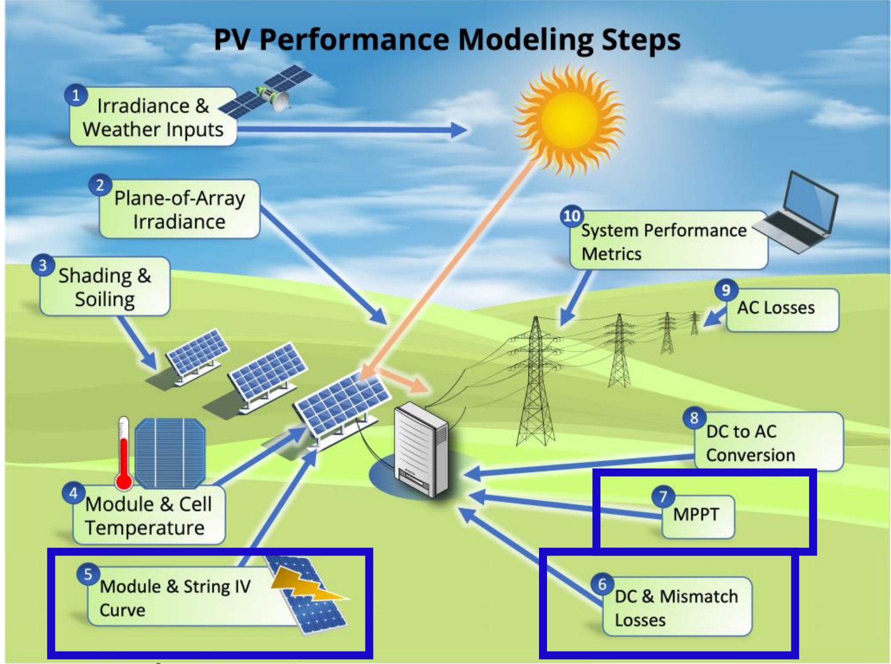
With:

- P_{dc}, DC power in [W]
- η efficiency around 20% (from datasheet)
- G_{POA} the irradiance in the plane of array [W/m²]
- A, the PV installation area [m2]

Not really precise for instantaneous values

UNIVERSITÉ SAVOIE MONT BLANC

Modeling steps



5./6./7. Power model

The PVWatts power model enables to take into account the effect of the cell temperature

$$P_{dc} = P_{dc0} \cdot \frac{G_{POA}}{1000\ W/m^2} \cdot \left(1 + \gamma_{pdc} \cdot (T_{cell} - 25°C)\right)$$

With:

- P_{dc0} Nominal DC power
- G_{POA} the irradiance in the plane of array [W/m²]
- γ_{pdc}, the temperature coefficient (negative, usually between -0.2 – -0.5 % W/m²/°C)
- T_{cell}, the cell temperature [°C]

PVWatts model: A. P. Dobos, “PVWatts Version 5 Manual” (2014).

UNIVERSITÉ SAVOIE MONT BLANC

Modeling steps

5./6./7. Power model

The Huld model (used in PVGIS) enables to take into account the module temperature and non-lineary with irradiance.

$$P_{dc} = \eta_{Huld}(G, T_m) \cdot G_{POA} \cdot A$$

$$\eta_{Huld}(G) = \eta_0 \cdot (1 + k_1 \cdot \ln(G') + k_2 \cdot \ln(G')^2 + k_3 \cdot T_m{}' + k_4 \cdot T_m{}' \cdot \ln(G') + k_5 \cdot T_m{}' \cdot \ln(G')^2 + k_6 \cdot T_m{}'$$

With:

- η_0, the reference efficiency (around 20%) [%]
- G_{POA} the irradiance in the plane of array [W/m²]
- A, the PV installation area [m2]
- $G' = \frac{G_{POA}}{1000\ W/m^2}$ the normalized irradiance
- $T_m' = T_m - 25°C$, the module temperature delta [°C]
- $k_1 \ldots k_6$, the model coefficients

Huld model: Thomas Huld et al., A power-rating model for crystalline silicon PV modules, 2011,

UNIVERSITÉ SAVOIE MONT BLANC

Modeling steps

5./6./7. Power model

The Huld model (used in PVGIS) enables to take into account the module temperature and non-lineary with irradiance.

$$P_{dc} = \eta_{Huld}(G, T_m) \cdot G_{POA} \cdot A$$

$$\eta_{Huld}(G) = \eta_0 \cdot (1 + k_1 \cdot \ln(G') + k_2 \cdot \ln(G')^2 + k_3 \cdot T_m{}' + k_4 \cdot T_m{}' \cdot \ln(G') + k_5 \cdot T_m{}' \cdot \ln(G')^2 + k_6 \cdot T_m{}'$$

With:

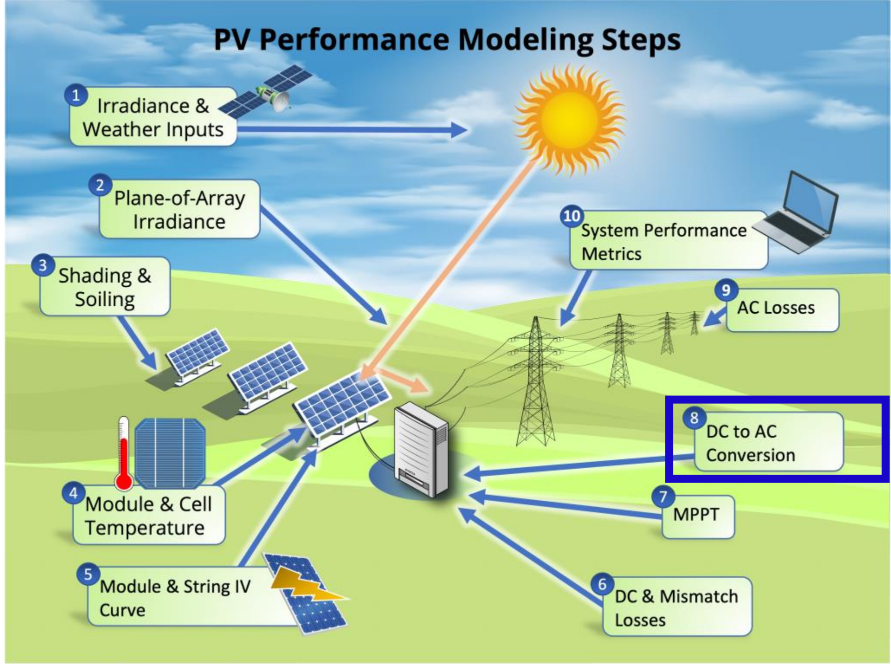
- η_0, the reference efficiency (around 20%) [%]
- G_{POA} the irradiance in the plane of array [W/m²]
- A, the PV installation area [m2]
- $G' = \frac{G_{POA}}{1000\ W/m^2}$ the normalized irradiance
- $T_m' = T_m - 25°C$, the module temperature delta [°C]
- $k_1 \ldots k_6$, the model coefficients

Reference values from PVGIS

Coefficient	c-Si	CIS	CdTe
k_1	-0.017237	-0.005554	-0.046689
k_2	-0.040465	-0.038724	-0.072844
k_3	-0.004702	-0.003723	-0.002262
k_4	0.000149	-0.000905	0.000276
k_5	0.000170	-0.001256	0.000159
k_6	0.000005	0.000001	-0.000006

Huld model: Thomas Huld et al., A power-rating model for crystalline silicon PV modules, 2011,

UNIVERSITÉ SAVOIE MONT BLANC



Modeling steps

8. Inverter model

The PVWatts inverter model enables to calculate a generic AC/DC efficiency

$$\eta = \frac{\eta_{nom}}{\eta_{ref}} \cdot \left(-0.0162 \cdot \frac{P_{dc}}{P_{dc0}} - \frac{0.0059}{\frac{P_{dc}}{P_{dc0}}} + 0.9858 \right)$$

With:

- η_{nom} $Nominal\ inverter\ efficiency$ The nominal inverter efficiency $[-]$
- η_{ref} The reference inverter efficiency $[-]$
- P_{dc} The DC power $\left[\frac{W}{m2}\right]$
- P_{dc0} The DC input power limit $[W/m2]$

PVWatts model: A. P. Dobos, “PVWatts Version 5 Manual” (2014).

UNIVERSITÉ SAVOIE MONT BLANC

Modeling steps

8. Inverter model

The Sandia inverter model enables to include the voltage and be more precise

$$P_{AC} = \left[\frac{P_{AC0}}{A - B} - C \cdot (A - B)\right] \cdot (P_{dc} - B) + C \cdot [P_{dc} - B]^2$$

Where:

- $A = P_{dc0} \cdot [1 + C1 \cdot (V_{dc} - V_{dc0})]$
- $B = P_{s0} \cdot [1 + C2 \cdot (V_{dc} - V_{dc0})]$
- $A = C_0 \cdot [1 + C3 \cdot (V_{dc} - V_{dc0})]$

Parameters:

- V_{dc}: DC input voltage (V). This is typically assumed to be the array's maximum power voltage.
- V_{dc0} : DC voltage level (V) at which the AC power rating is achieved at reference operating conditions.
- P_{AC} : AC output power (W)
- P_{AC0} : Maximum AC power rating for inverter at reference conditions (W). Assumed to be an upper limit.
- P_{dc0} : DC power level (W) at which the AC power rating is achieved at reference operating conditions.
- P_{s0} : DC power required to start the inversion process (W)
- C_0,C_1, C_2, C_3: Empirical coefficients

D. King, S. Gonzalez, G. Galbraith, W. Boyson, "Performance Model for Grid-Connected Photovoltaic Inverters", SAND2007-5036, Sandia National Laboratories

Modeling steps

5./6./7. Power model

Time for some
hands-on exercises !

Use the following notebook:
https://github.com/AlexandreHugoMathieu/pvfault_detection_solar_academy/blob/master/notebooks/dc_power_estimation.ipynb

Follow the python tutorial and estimate the DC power for one year.

Resources

- Modeling guide PVPMC: https://pvpmc.sandia.gov/modeling-guide/
- Python / Pvlib tutorial: https://pvsc-python-tutorials.github.io/PVSC48-Python-Tutorial/
- To go further:
    - The Use of Advanced Algorithms in PV Failure Monitoring: https://iea-pvps.org/wp-content/uploads/2021/10/Final-Report-IEA-PVPS-T13-19_2021_PV-Failure-Monitoring.pdf

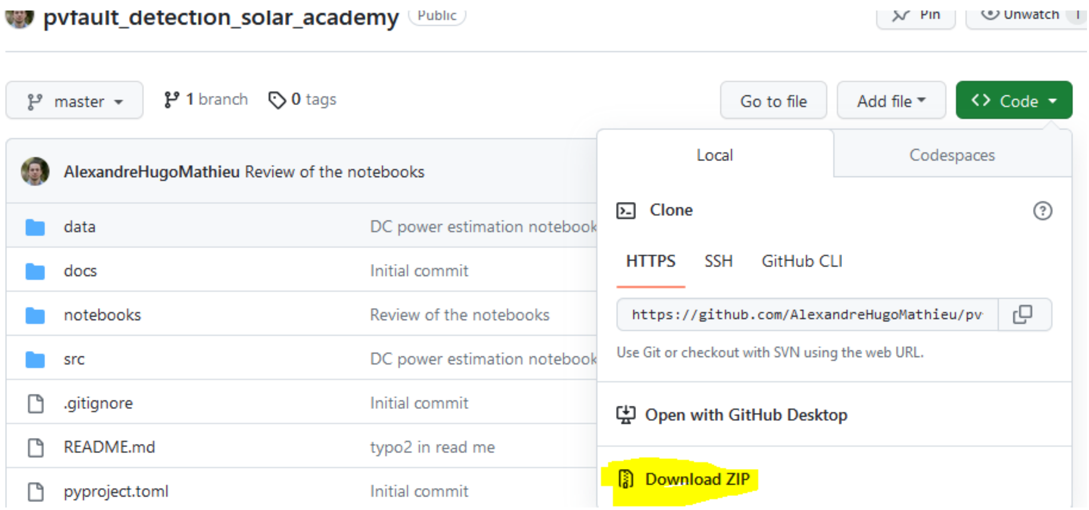
UNIVERSITÉ SAVOIE MONT BLANC

Appendix

How to install Python and import the course repository to use the notebooks on your local PC.

1. Install python: www.python.org/downloads/, download and install the 3.9.13 “release” (Add python to your Path)

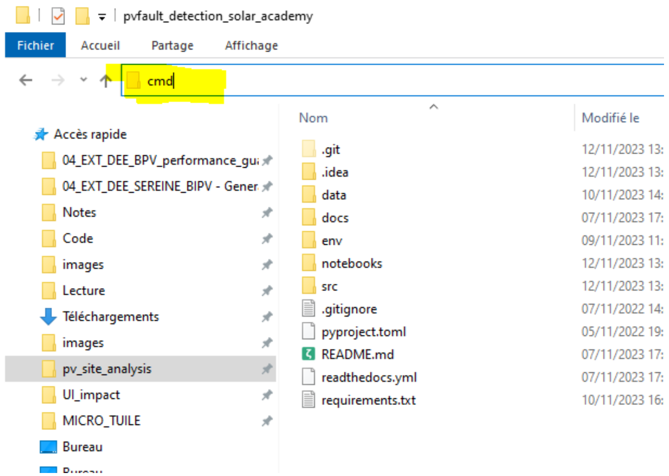
2. Go to https://github.com/AlexandreHugoMathieu/pvfault_detection_solar_academy, click on the green “Code” button and then download the folder as the zip



UNIVERSITÉ SAVOIE MONT BLANC

Appendix

How to install Python and import the course repository to use the notebooks on your local PC.



1. Install python: www.python.org/downloads/, download and install the 3.9.13 “release” (Add python to your Path)
2. Go to https://github.com/AlexandreHugoMathieu/pvfault_detection_solar_academy, click on the green “Code” button and then download the folder as the zip
3. Unzip it and put it in adequate location in your PC.
4. Let’s create a virtual environment where you will find all the functions for this course:
    1. Go in the folder and open the command line from that same folder by writing “cmd” in the path bar (with Windows)

Appendix

How to install Python and import the course repository to use the notebooks on your local PC.

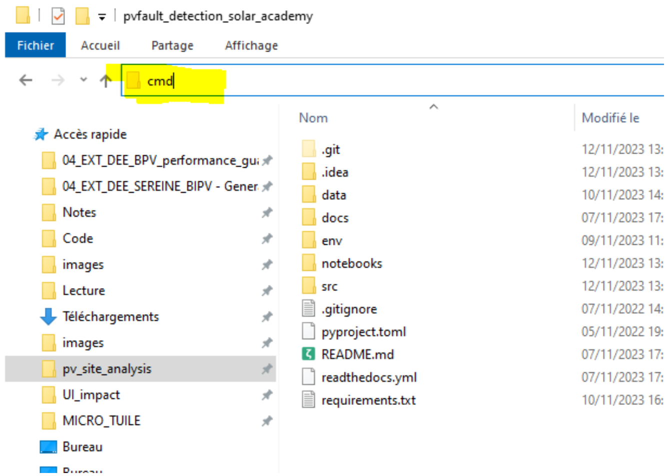
1. Install python: www.python.org/downloads/, download and install the 3.9.13 “release” (Add python to your Path)
2. Go to https://github.com/AlexandreHugoMathieu/pvfault_detection_solar_academy, click on the green “Code” button and then download the folder as the zip
3. Unzip it and put it in adequate location in your PC.
4. Let’s create a virtual environment where you will find all the functions for this course:
    1. Go in the folder and open the command line from that same folder by writing “cmd” in the path bar (with Windows)
    2. In the command bar: execute the following line to create the “solar_env” environnement that you will use in your notebooks
        1. “pip install virtualenv”
        2. “python –m virtualenv solar_env”
        3. “call solar_env\Scripts\activate” (you should have a ‘solar_env’ on the left of the command at this point)
        4. “pip install –r requirements.txt” (load all the libraries, take a little time, be patient)
        5. “python -m ipykernel install --name=solarkernel” (create a kernel for the notebooks)

It’s ready !

Appendix

How to start a notebook

1. Go in the folder and open the command line from that same folder by writing “cmd” in the path bar (with Windows)



2. In the command bar, exexute:
   1. “call solar_env\Scripts\activate” (go in the virtual env)
   2. “jupyter notebook” (open the notebooks browser)

3. Browse to the notebooks folder, choose one and pick the solarkernel when asked.