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

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 →

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

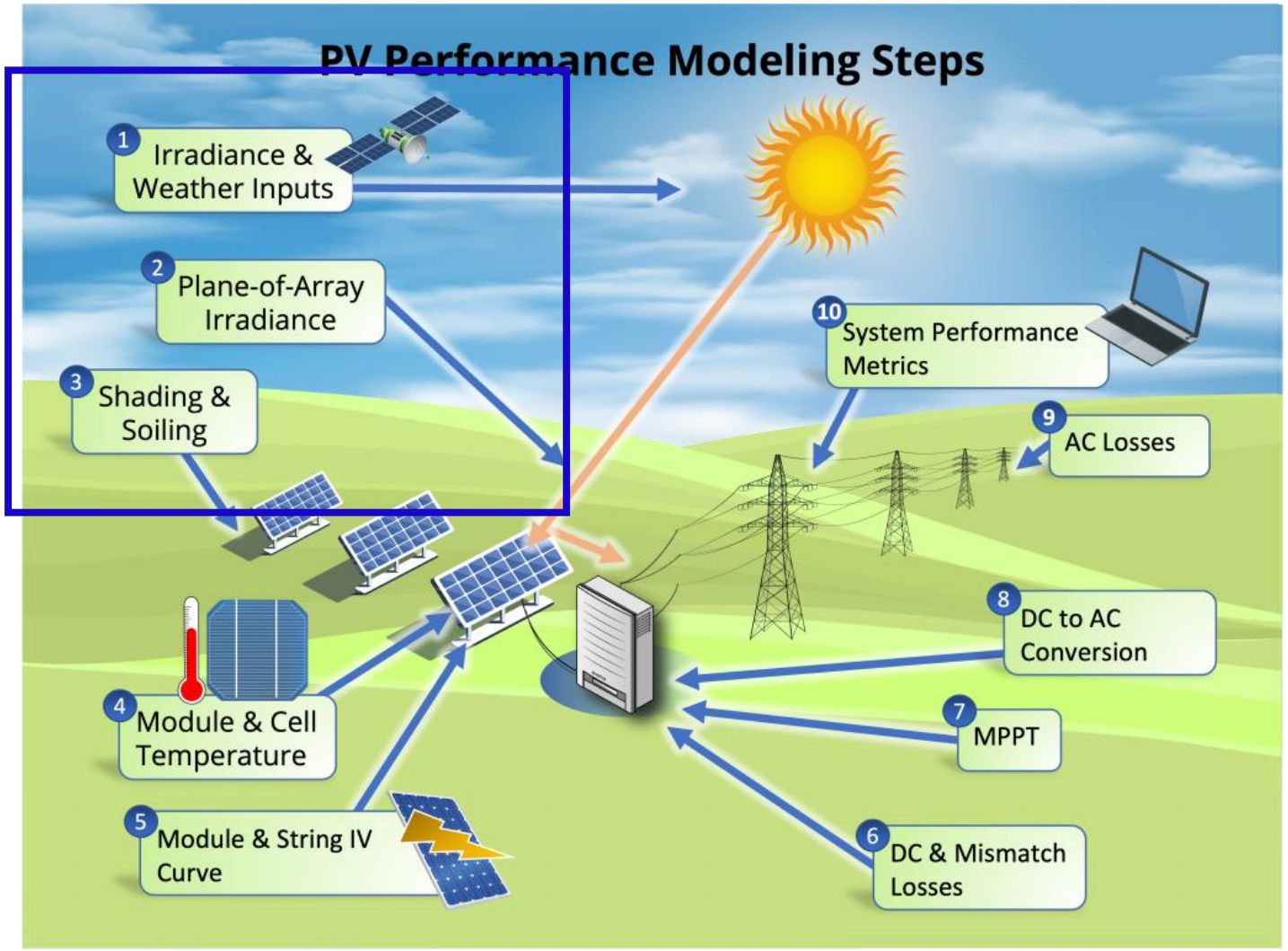


*image from: PVSC48 python tutorial

Modeling steps

27/11/2023

Today

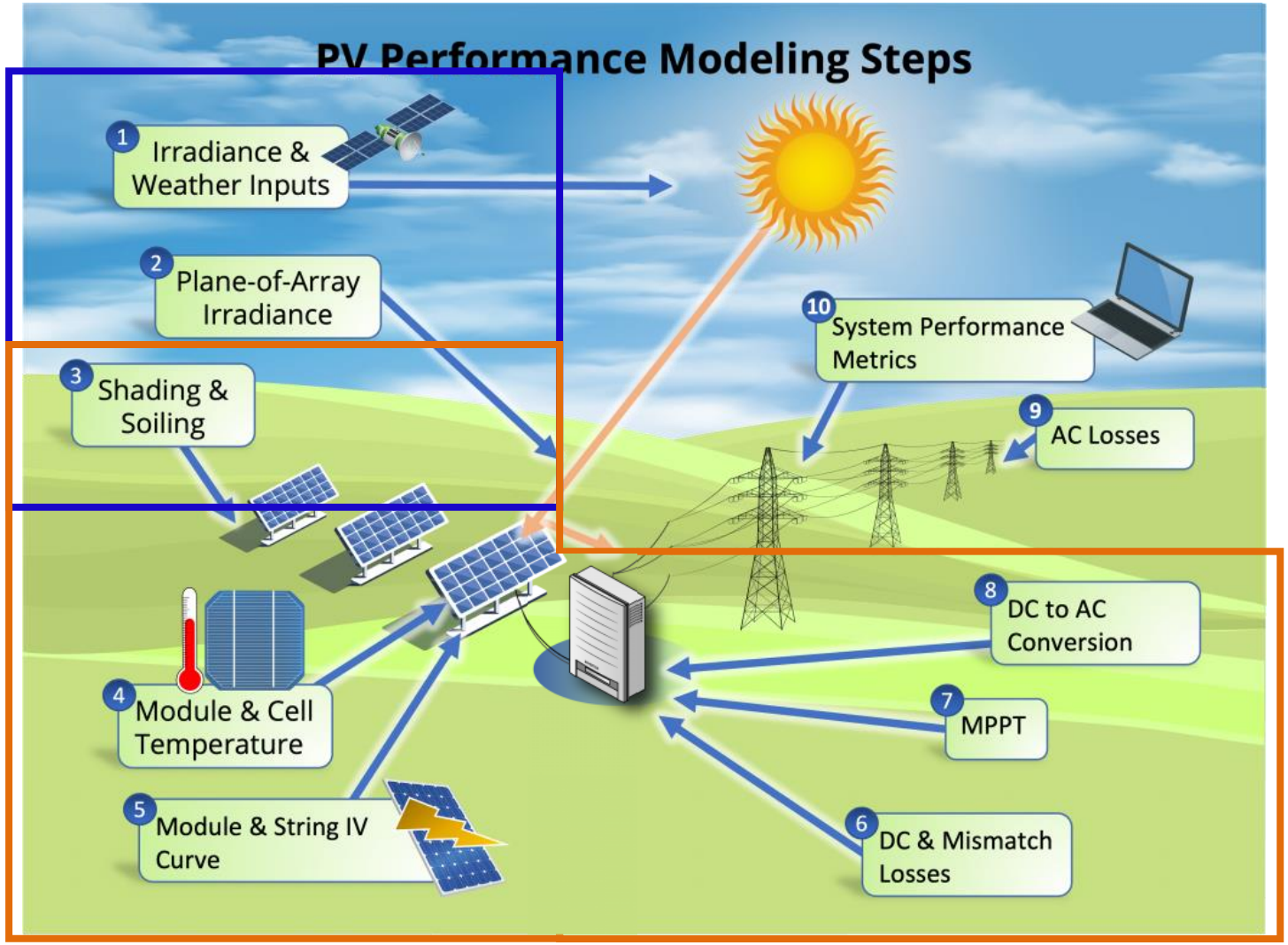


*image from: PVSC48 python tutorial

UNIVERSITÉ SAVOIE MONT BLANC

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

Modeling steps

Notebook recap 27/11/2023

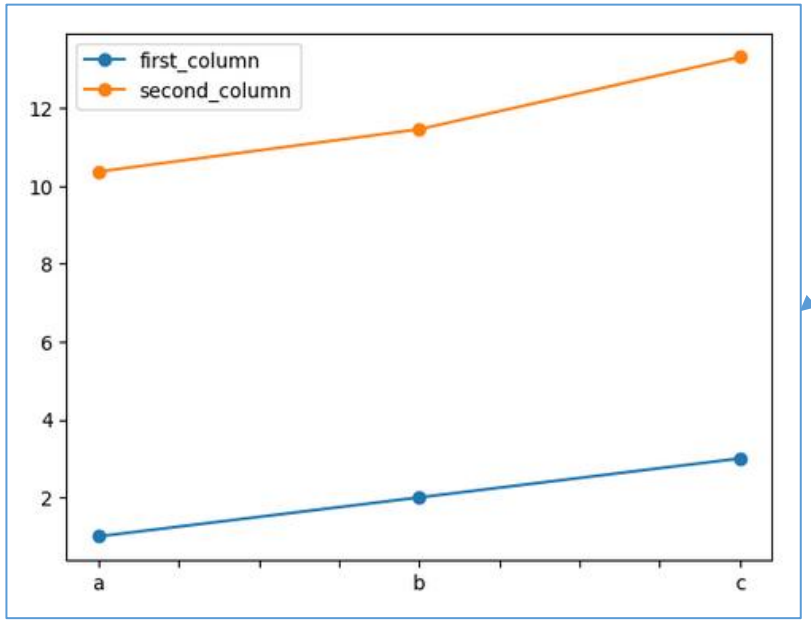
Python commands 1/2

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

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("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(12)) # Show the first 12 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

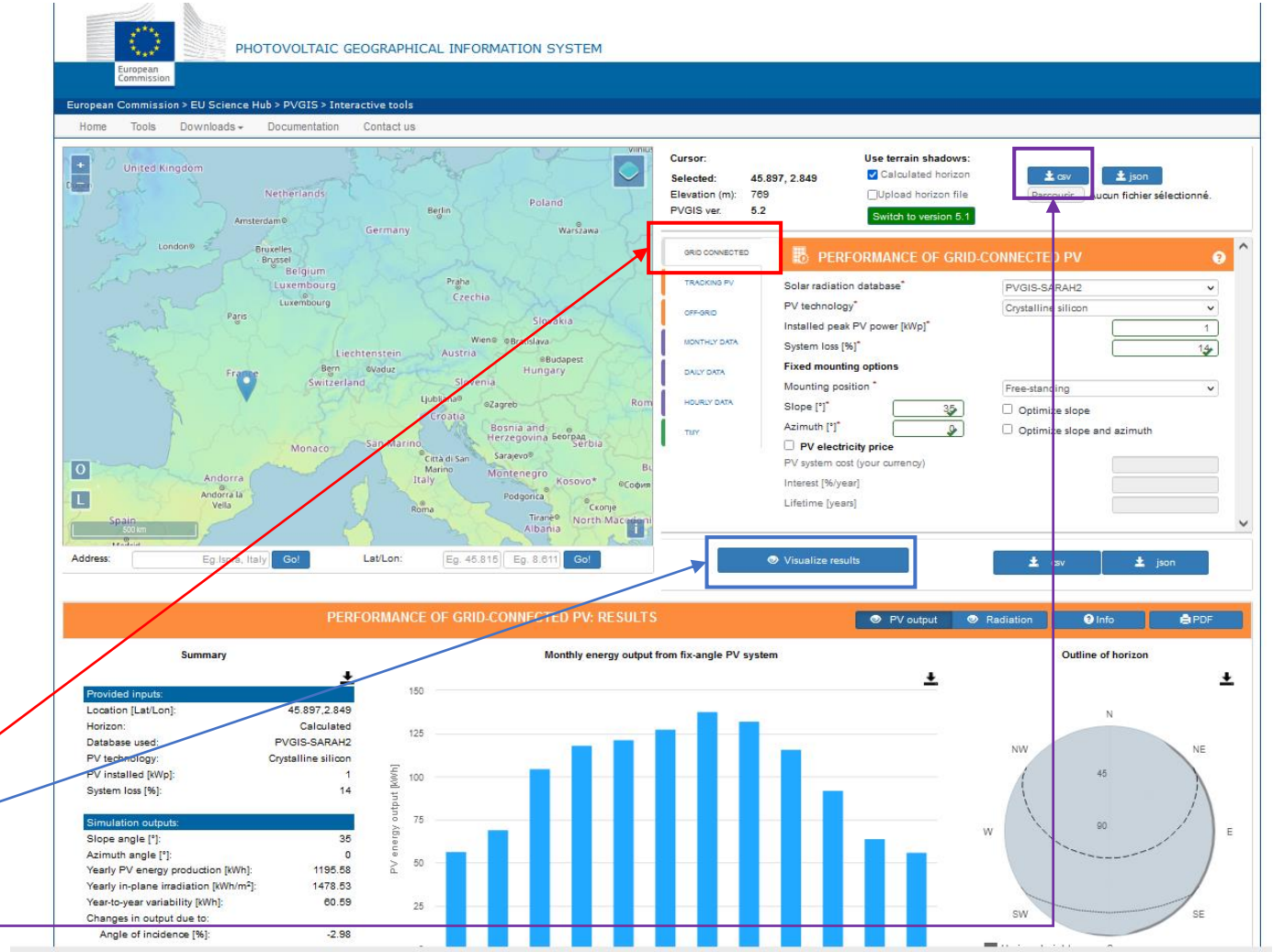
3. Shading / Terrain horizon mask

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