Drentkiewicz – Project 3 – Addition of a PV System

The created module provides functionality for adding photovoltaic (PV) systems to the existing circuit/power flow solver. Users can add a PV system using the “add_PV” method, which requires an identification (name), to which bus it is connected, its power rating in megawatts (MW), a relative irradiance (a value between 0 and 1 representing sunlight availability), and the system’s efficiency (a value between 0 and 1). For example, calling network.add_PV("PV1", "bus4", 20, 0.65, 0.95) adds a PV system named "PV1" connected to bus 4, with a 20 MW rating, 65% irradiance, and 95% efficiency.

```
network.add_PV( name: "PV1",  bus: "bus4", P: 20, RI: 0.65, eff: 0.95)
network.add_PV( name: "PV2",  bus: "bus5", P: 30, RI: 0.75, eff: 0.95)
```

Figure 1

The “add_PV” method (Figure3) creates an instance of the PV_System class (Figure 2) using the provided parameters and stores it in the network’s PVs dictionary. The PV_System class represents individual PV systems and calculates their power output based on the formula -P × RI × eff, where the negative sign indicates power injected into the system. Once created, the “add_PV” method calls the “PV_load” method, which uses the previously created “add_load” method to register the PV’s power generation as a negative load, delivering only real power at the corresponding bus in the system.

The implementation can be realized by uncommenting the two lines in Figure 1 in the Network script, which serves as the main script for running the power flow and fault computing algorithms.

```
2 usages  new *
class PV_System:

    new *
    def __init__(self, name, bus, P, RI, eff):

        self.name = name            #system reference
        self.bus = bus              #bus connection
        self.P = P                  #Power Rating MW
        self.RI = RI                #Relative Irradiance: 0-1
        self.eff = eff              #system efficiency: 0-1

        self.power_gen = self.get_power()

    1 usage  new *
    def get_power(self):

        return -self.P * self.RI * self.eff
```

Figure 2

```
2 usages  new *
def add_PV(self, name, bus, P, RI, eff):
    self.PVs[name] = PV_System(name, bus, P, RI, eff)
    self.PV_load(name, bus)

1 usage  new *
def PV_load(self, name, bus):
    self.add_load(name, bus, self.PVs[name].power_gen,  reactive_power: 0)
```

Figure 3

Adding two PV systems with the parameters shown in Figure 1 results in a real power injection of 12.35 MW into bus 4 and 21.375 MW into bus 5. Figure 4 shows the results of the solver displaying the magnitudes and angles of the voltages at each bus.

```
Run        Network
C:\Users\jjdre\AppData\Local\Programs\Python\Python312\python.exe C:\Users\jjdre\PycharmProjects\Group7_Project2\Network.py
Converged after 3 iterations

Final Voltage Magnitudes:
[1.         0.93866938 0.92289985 0.93218762 0.92959835 0.94204505
 1.        ]

Final Voltage Angles (degrees):
[ 0.         -3.01934373 -3.85098638 -3.09217368 -3.07511559 -2.24032481
  3.85872519]

Process finished with exit code 0
```

Figure 4

These results make intuitive sense. The voltage magnitudes have increased, and angles have decreased on the load buses due to less power/current needed to be provided by the slack bus generator. Note that the voltage angle increased on bus 7. This is due to the larger difference in power supplied between the two generators in this case.

Adding these power injections as a negative load in Power World, as shown in Figure 5, produced the results shown in Figure 6.

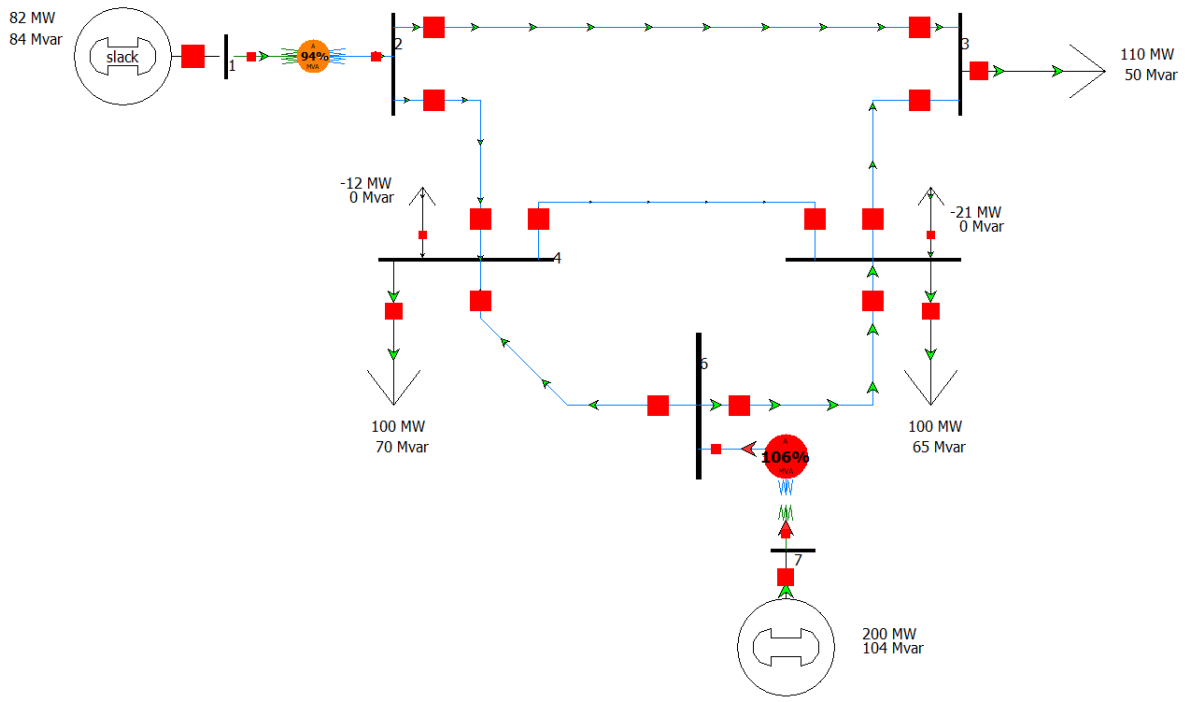

Figure 5

Buses

Records Geo Set Columns f(x)

Filter Advanced Bus Find... Remove Quick Filter

	Number	Name	Area Name	Nom kV	PU Volt	Volt (kV)	Radians	Angle (Deg)
1	1	1	1	20.00	1.00000	20.000	0.0000000	0.00
2	2	2	1	230.00	0.93865	215.889	-0.0527297	-3.02
3	3	3	1	230.00	0.92288	212.263	-0.0672457	-3.85
4	4	4	1	230.00	0.93217	214.399	-0.0540015	-3.09
5	5	5	1	230.00	0.92958	213.804	-0.0537055	-3.08
6	6	6	1	230.00	0.94203	216.667	-0.0391349	-2.24
7	7	7	1	18.00	1.00000	18.000	0.0673147	3.86

Figure 6

The python solver results agree with the Power World results within 0.48%.

Even though this PV system addition is very trivial and ignores the real-world implementation which more than likely includes battery storage and would not depend on a fixed solar irradiance, etc., it still demonstrates how the solver can be added to in other cases. Objects can be modelled based upon user inputs, added to the circuit at any bus and included in the computation appropriately.