Smart Energy Market Study Project

First Name:----------

Last Name:----------

ESME IN3 2024-2025

Windows installation

1. Firstly you need to download and install the latest version of HD5 from https://www.hdfgroup.org/download-hdf5/
2. Website ask you to create an account to be able to download the package. Proceed with a free account creation and download the application.
3. Add HDF5 to the system PATH:
   Open Environment Variables:
   Right-click on "This PC" > Properties > Advanced system settings.
   Click on Environment Variables.
   Under System Variables, find Path, click Edit, and add the directory path where HDF5 is installed (e.g., C:\Program Files\HDF_Group\HDF5\bin).
4. Install h5py using pip:
5. Open a command prompt (in search bar type cmd then press enter you will have a new cmd prompt appeared.
6. Run following command "`pip install h5py`"
7. If this fails, specify the HDF5 library and header locations explicitly:
8. `pip install --global-option=build_ext --global-option="-I<path_to_hdf5_include>" --global-option="-L<path_to_hdf5_lib>" h5py`
9. Replace `<path_to_hdf5_include>` and `<path_to_hdf5_lib>` with the appropriate paths from your HDF5 installation.
10. Once HDF5 and h5py are installed, install PyPSA
11. Run following command "`pip install pypsa`"

MacOs

1. Install Homebrew (if not already installed):

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

2. Install HDF5:

brew install hdf5

3. Proceed to install Python libraries (e.g., h5py or pypsa):

pip install h5py

pip install pypsa

Testing the Python Setup

After installing HDF5, test your Python setup by importing h5py:

import h5py

print(h5py.version.info)

Assignment using pyPSA

From the following link you would be able to run different scenario for simple electricity market examples. Inspire from them and model :

https://pypsa.readthedocs.io/en/latest/examples/simple-electricity-market-examples.html

for each case you need to put your pyPSA code and description about the output. It should contain maximum 10 lines of description per assignment.

1. Single bidding zone with fixed load, 2 periods
    a. Interpret the results you obtained regarding the cleared price
2. Four bidding zones connected by transmission, one period
    a. Interpret the results you obtained regarding the cleared price

---

Single bidding zone with fixed load, 2 periods

```
import numpy as np

import pypsa
marginal_costs = {"Wind": 0, "Hydro": 0, "Coal": 30, "Gas": 60, "Oil": 80}

# power plant capacities (nominal powers in MW) in each country (not necessarily realistic)
power_plant_p_nom = {
   "South Africa": {"Coal": 35000, "Wind": 3000, "Gas": 8000, "Oil": 2000},
   "Mozambique": {
      "Hydro": 1200,
   },
   "Swaziland": {
      "Hydro": 600,
   },
}

# transmission capacities in MW (not necessarily realistic)
transmission = {
   "South Africa": {"Mozambique": 500, "Swaziland": 250},
   "Mozambique": {"Swaziland": 100},
}

# country electrical loads in MW (not necessarily realistic)
loads = {"South Africa": 42000, "Mozambique": 650, "Swaziland": 250}
```

country = "South Africa" if we need 2 countries we have to add one more country and include it in the you have to being in other countries.

```
network = pypsa.Network()

# snapshots labelled by [0,1](2 periods)
network.set_snapshots(range(2))

network.add("Bus", country)


for tech in power_plant_p_nom[country]:
   network.add(
      "Generator",
      f"{country} {tech}",
      bus=country,
      p_nom=power_plant_p_nom[country][tech],
      marginal_cost=marginal_costs[tech],
      p_max_pu=([0.6, 0.4] if tech == "Wind" else 1), Power maximum 0,6 and 0.4 are variables of wind
we took from mostly south Africa with
   )

# load which varies over the snapshots
network.add(
   "Load",
   f"{country} load",
   bus=country,
   p_set=loads[country] + np.array([1000, 3000]),
)
network.optimize()
print(network.loads_t.p)
print(network.generators_t.p)
print(network.buses_t.marginal_price)
```

Description:

```
[2 rows x 4 columns]
Bus       South Africa
snapshot
0                 60.0
1                 80.0
```

```
Presolve : Reductions: rows 0(-18); columns 0(-8); elements 0(-24) - Reduced to empty
Solving the original LP from the solution after postsolve
Model name          : linopy-problem-76lkq0og
Model status        : Optimal
```

Network optimal message. In the pypsa function network.optimize() where network is assigned to network = pypsa.Network() function it takes in different constraints and optimizes the network using objective function where supply should be equal to demand and input cost should be minimal. In the above example at the wind factor I took 0.6, 0.4 I have both highest price and lowest price. If I increase the wind factor 0.6, 0.5 I could get 60 and 60 price at 2 periods or if decrease the load to balance it. As it is the only factor changing. I could change the loads to but that is a bit more complicated as demand should be equal to supply and the sources are already set. I did tried to change and it is going into negative prices with errors.

Four bidding zones connected by transmission, one period

Code:

```
import numpy as np
import pypsa

# Marginal costs for generation technologies
marginal_costs = {"Wind": 0, "Hydro": 0, "Coal": 30, "Gas": 60, "Oil": 80}

# Power plant capacities (nominal powers in MW)
power_plant_p_nom = {
   "South Africa": {"Coal": 35000, "Wind": 3000, "Gas": 8000, "Oil": 2000},
   "Mozambique": {"Hydro": 1200},
   "Swaziland": {"Hydro": 600},
   "Namibia": {"Wind": 1500, "Oil": 1000},
}

# Transmission capacities (MW)
```

```
transmission = {
   "South Africa": {"Mozambique": 500, "Swaziland": 250, "Namibia": 300},
   "Mozambique": {"Swaziland": 100},
   "Namibia": {"Swaziland": 150},
}

# Electrical loads (MW)
loads = {"South Africa": 42000, "Mozambique": 650, "Swaziland": 250, "Namibia": 2000}

# Create the network
network = pypsa.Network()

countries = ["Swaziland", "Mozambique", "South Africa","Namibia" ]

for country in countries:
   network.add("Bus", country)

   for tech in power_plant_p_nom[country]:
      network.add(
         "Generator",
         f"{country} {tech}",
         bus=country,
         p_nom=power_plant_p_nom[country][tech],
         marginal_cost=marginal_costs[tech],
      )

   network.add("Load", f"{country} load", bus=country, p_set=loads[country])

   # add transmission as controllable Link
   if country not in transmission:
      continue

   for other_country in countries:
      if other_country not in transmission[country]:
         continue

      # NB: Link is by default unidirectional, so have to set p_min_pu = -1
      # to allow bidirectional (i.e. also negative) flow
      network.add(
         "Link",
         f"{country} - {other_country} link",
         bus0=country,
         bus1=other_country,
         bus2=other_country,
```

```
        p_nom=transmission[country][other_country],
        p_min_pu=-1,
    )
network.optimize()
print(network.loads_t.p)
print(network.generators_t.p)
print(network.buses_t.marginal_price)
```

Description:

```
Presolve : Reductions: rows 0(-18); columns 0(-8); elements 0(-24) - Reduced to empty
Solving the original LP from the solution after postsolve
Model name          : linopy-problem-76lkq0og
Model status        : Optimal
```

```
[1 rows x 8 columns]
Bus       Swaziland  Mozambique  South Africa  Namibia
snapshot
now            30.0        30.0          60.0     60.0
```

I this new one I have added Namibia as the new country and I have defined constraints and adjusted the load to match the supply. I have the optimal price of the countries it I slow for new country Namibia because the load is 2000units but the production is 2500 of oil and wind so it is always surplus in production.