CHECKLIST SDOM TRAINING

Table of Contents

1. BEFORE START

1.1. DOWNLOAD AND EXTRACT THE FOLDER WITH THE TRAINING MATERIAL

- Do not put the folder in a folder that synchronizes with the cloud (One drive, Google Drive, share point box, etc)

1.2. INTRODUCE uv AND VIRTUAL ENVIROMENT

- Explore folder .venv

2. Show SDOM inputs csv files

- OPEN PDF AND SHOW IN PARALLEL WITH CSVs
- All input files should be in the directory specified in `data_dir = os.path.join(current_folder, "sample_data", "br_test_daily_b")`
- Don't change column names!
- Keep the root name of the files!

3. Run Code and show help, explore objects

3.1. Load Data

```
data = load_data( data_dir )
```

3.1.1. DOCSTRINGS WITH DOCUMENTATION

- Type in the debug console `help(sdom.load_data)` and then see the terminal with docstrings.

3.1.2. SHOW DICTIONARY (OUTPUT)

- `data.keys()`

```
data.keys()
dict_keys(['formulations', 'solar_plants', 'wind_plants', 'load_data',
'nuclear_data', 'large_hydro_data', 'other_renewables_data', 'cf_solar',
'cf_wind', 'cap_solar', 'cap_wind', 'storage_data', 'STORAGE_SET_J_TECHS',
'STORAGE_SET_B_TECHS', 'thermal_data', 'scalars', 'large_hydro_max',
'large_hydro_min', 'cap_imports', 'price_imports', 'cap_exports', 'price_exports',
'filtered_cap_solar_dict', 'complete_solar_data', 'filtered_cap_wind_dict',
'complete_wind_data'])
```

- Next development efforts of SDOM include integration with NLR infrasys

3.1.3. EMULATE ERRORS LOADING DATA

- Change file name `lahy_hourly_2025.csv` to `lahy_hourl.csv`

```
data = load_data( data_dir )
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "c:\Users\smachado\repositories\pySDOM\sdom_epe_trainning\.venv\Lib\site-
packages\sdom\io_manager.py", line 103, in load_data
    input_file_path = check_file_exists(input_data_dir,
INPUT_CSV_NAMES["large_hydro_data"], "large hydro data")

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^
  File "c:\Users\smachado\repositories\pySDOM\sdom_epe_trainning\.venv\Lib\site-
packages\sdom\common\utilities.py", line 35, in check_file_exists
    raise FileNotFoundError(f"Expected {file_description} file not found:
{filepath}{file_name}")
FileNotFoundError: Expected large hydro data file not found:
C:\Users\smachado\repositories\pySDOM\sdom_epe_trainning\sample_data\br_test_daily
_blahy_hourly.csv
```

- Error that can happen if change a `col_name` only will manifest while initializing the model:

```
Traceback (most recent call last):
  File "c:\Users\smachado\repositories\pySDOM\sdom_epe_trainning\.venv\Lib\site-
packages\pandas\core\indexes\base.py", line 3812, in get_loc
    return self._engine.get_loc(casted_key)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "pandas/_libs/index.pyx", line 167, in
pandas._libs.index.IndexEngine.get_loc
```

```
  File "pandas/_libs/index.pyx", line 196, in
pandas._libs.index.IndexEngine.get_loc
  File "pandas/_libs/hashtable_class_helper.pxi", line 7088, in
pandas._libs.hashtable.PyObjectHashTable.get_item
  File "pandas/_libs/hashtable_class_helper.pxi", line 7096, in
pandas._libs.hashtable.PyObjectHashTable.get_item
KeyError: 'LargeHydro'
...
```

3.2 INITIALIZE MODEL

```
model = initialize_model(
    data,
    n_hours = n_steps,
    with_resilience_constraints = with_resilience_constraints,
    model_name='SDOM_training'
    )
```

3.2.1 EXPLORE PYOMO MODEL

- Look for param, scalars, obj, blocks...etc

SDOM leverages on pyomo blocks to separate in different blocks the variables, parameters, expressions, constraints, etc of diferent model components. In this way, in that pyomo instance, SDOM creates the following blocks:

- Open PDF at the end
- Core optimization Blocks (Blocks that include variables, sets and parameters)
  - thermal
  - pv
  - wind
  - hydro
  - storage
- Blocks containing only parameters (Do not include any decision variables)
  - demand
  - nuclear
  - other_renewables
- Optional blocks (These are created depending on the configuration provided by the user)
  - imports
  - exports
  - resiliency
- USE `pprint()` to explore a parameter

```
model.demand.pprint()
```

```
...
            738 : 14506.74
            739 : 13414.05
            740 :  14012.9
            741 : 14299.85
            742 : 14413.19
            743 : 15337.52
            744 : 15347.54

    1 Declarations: ts_parameter
```

- Indexing an specific value

```
model.demand.ts_parameter[1]
13702.34
```

- USE `pprint()` to explore a constraint

```
model.thermal.capacity_generation_constraint.pprint()

...

    (744, '235c') :  -Inf : thermal.generation[744,235c] -
thermal.plant_installed_capacity[235c] :   0.0 :   True
    (744, '241c') :  -Inf : thermal.generation[744,241c] -
thermal.plant_installed_capacity[241c] :   0.0 :   True
     (744, '83c') :  -Inf :   thermal.generation[744,83c] -
thermal.plant_installed_capacity[83c] :   0.0 :   True
     (744, '98c') :  -Inf :   thermal.generation[744,98c] -
thermal.plant_installed_capacity[98c] :   0.0 :   True
```

3.3 SOLVER CONFIGURATION DICT

- Use same Solver names and configuration parameters detailed in Pyomo documentation

```
solver_dict = get_default_solver_config_dict(
    solver_name="highs",
    executable_path=""
    )
```

3.4. RUN SOLVER

```
best_result = run_solver(model, solver_dict)
```

3.5. EXPLORE CSV OUTPUTS

In the path specified by "output_dir", sdom will writhe the following output csv files:

File name	Description
OutputGeneration_CASENAME.csv	Hourly generation results aggregated by technology, curtailment, imports/exports and Load.
OutputStorage_CASENAME.csv	Hourly storage operation results (charging/discharging and SOC).
OutputSummary_CASENAME.csv	Summary of key simulation results and statistics.
OutputThermalGeneration_CASENAME.csv	Hourly results for thermal generation plants.

3.6. RUN PLOTS

```
venv_python = os.path.join(current_folder, ".venv", "Scripts", "python.exe")
script_path = os.path.join("training_material", "training_sesion_1",
"sdom_plots.py")
subprocess.run([venv_python, script_path], check=True)
```

- See plots
- Open script

4. RUN SENSITIVE CASES

4.1. DECREASE STORAGE COSTS

- Change in the input file `StorageData_2025.csv` the costs for BESS of 4h:

```
,Li-Ion-2h,Li-Ion-4h,Li-Ion-6h,Li-Ion-8h
P_Capex,931,0.1551,2171,2791
E_Capex,0,0,0,0
Eff,0.85,0.85,0.85,0.85
Min_Duration,2,4,6,8
Max_Duration,2,4,6,8
Max_P,10000,10000,10000,20000
MaxCycles,5000,5000,5000,5000
Coupled,1,1,1,1
FOM,23,0.39,54,70
VOM,0,0,0,0
```

```
Lifetime,16,16,16,16
CostRatio,0.5,0.5,0.5,0.5
```

- It will install maximum capacity
- To use PLOT SCRIPTS DON`T FORGET TO CHANGE `case_name` value accordingly to take the correct data

4.2. ADD CUSTOM CONSTRAINT

- Remember that, in general, the matematical model of each device/technology in SDOM is isolated in Pyomo blocks
- Let's suppose you want to add a constraint related to storage.
- If you add this code, you'll be adding a `Constraint` in the `storage` block which enforces the model to install at least 500 MW of each storage technology

```
#ADD A SIMPLE CUSTOM CONSTRAINT
from pyomo.core import Var, Constraint, Expression
block = model.storage
block.min_Pinstalled_constraint = Constraint( block.j, rule = lambda m,j: m.Pdis[j] >= 500 )  # Set a minimum installed power capacity of 500 MW for the storage technology 'Li-Ion-4h'
```