AGISTIN T4.6 Usage examples of the tool

CITCEA-UPC

January 2025

Contents

1. Introduction
2. Example 0
3. Example 1
4. Example 2
5. Example 3

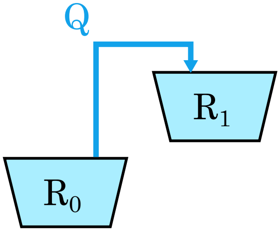
Introduction

- Built in Python3
- Using Pyomo library and extension Pyomo Network
- Object oriented approach

AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

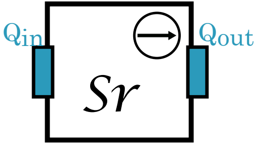
EXAMPLE 0

Two Reservoirs + one Source (imposes Q)



AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

Source



Param	Desc.	Units
$Q_{in}(t)$	Inlet flow	[m^3/s]
$Q_{out}(t)$	Outlet flow	[m^3/s]

Var	Desc.	Units

Port	Var
Q_{in}	$Q_{in}(t)$
Q_{out}	$Q_{out}(t)$

Constraints

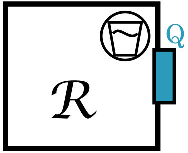
AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

Source

```python
import pyomo.environ as pyo
from pyomo.network import Arc, Port


def Source(b, t, data, init_data=None):


    # Parameters
    b.Qin = pyo.Param(t, initialize=[-k for k in data['Q']])
    b.Qout = pyo.Param(t, initialize=data['Q'])

    # Variables

    # Ports
    b.port_Qin = Port(initialize={'Q': (b.Qin, Port.Extensive)})
    b.port_Qout = Port(initialize={'Q': (b.Qout, Port.Extensive)})

    # Constraints
```

AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

Reservoir



Param	Desc.	Units
W_0	Initial volume	[m^3/s]
$\underline{W}$, $\overline{W}$	min, max volume	[m^3]

Var	Desc.	Units
$W(t)$	Volume	[m^3/s]
$Q(t)$	Flow	[m^3/s]

Port	Var
Q	$Q(t)$

Equations

$$W^{[k]} = \begin{cases} W^{[k-1]} + Q^{[k]} & \text{if } k > 0 \\ W_0 + Q^{[k]} & \text{otherwise} \end{cases}$$

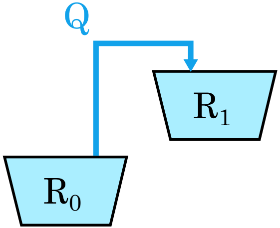
AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

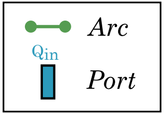
Reservoir

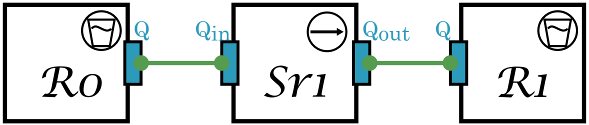
```
def Reservoir_Ex0(b, t, data, init_data):


    # Parameters
    b.W0 = pyo.Param(initialize=data['W0'])

    # Variables
    b.Q = pyo.Var(t, initialize=init_data['Q'], within=pyo.Reals)
    b.W = pyo.Var(t, initialize=init_data['W'], bounds=(data['Wmin'], data['Wmax']), within=pyo.NonNegativeReals
            )

    # Ports
    b.port_Q = Port(initialize={'Q': (b.Q, Port.Extensive)})

    # Constraints
    def Constraint_W(_b, _t):
        if _t>0:
            return _b.W[_t] == _b.W[_t-1] + (_b.Q[_t])
        else:
            return _b.W[_t] == _b.W0 + (_b.Q[_t])
    b.c_W = pyo.Constraint(t, rule = Constraint_W)
```

Introduction
Example 0
Example 1
Example 2
Example 3

Connect *Blocks*' *Ports* with *Arcs*







AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

Create the *Blocks*

```
# ===== Create the system =====

# Source 1
m.Source1 = pyo.Block()
data_s1 = {'Q':[1,1,1,1,1]}
Source(m.Source1, m.t, data_s1)

# Reservoir0
m.Reservoir0 = pyo.Block()
data_r0 = {'W0':10, 'Wmin':0, 'Wmax':20}
init_r0 = {'Q':[0,0,0,0,0], 'W':[5,5,5,5,5]}
Reservoir_Ex0(m.Reservoir0, m.t, data_r0, init_r0)

# Reservoir1
m.Reservoir1 = pyo.Block()
data_r1 = {'W0':5, 'Wmin':0, 'Wmax':20}
init_r1 = {'Q':[0,0,0,0,0], 'W':[5,5,5,5,5]}
Reservoir_Ex0(m.Reservoir1, m.t, data_r1, init_r1)
```

Create the *Arcs*

```
# Connections
m.s1r0 = Arc(ports=(m.Source1.port_Qin, m.Reservoir0.port_Q), directed=True)
m.s1r1 = Arc(ports=(m.Source1.port_Qout, m.Reservoir1.port_Q), directed=True)

pyo.TransformationFactory("network.expand_arcs").apply_to(m) # apply arcs to model
```

AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

Run the optimization

```python
#%% RUN THE OPTIMIZATION

# Objective function (Feasibility problem in this example)
def obj_fun(m):
    return 0
m.goal = pyo.Objective(rule=obj_fun, sense=pyo.minimize)

instance = m.create_instance()
solver = pyo.SolverFactory('ipopt')
solver.solve(instance, tee=False)
```

Results

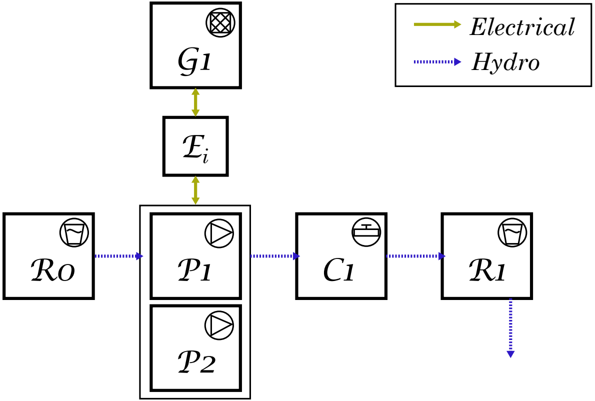
```python
instance.Reservoir1.W.pprint()
instance.Reservoir0.W.pprint()

# RESULTS:
    # W : Size=5, Index=t
    #     Key : Lower : Value : Upper : Fixed : Stale : Domain
    #       0 :     0 :   6.0 :    20 : False : False : NonNegativeReals
    #       1 :     0 :   7.0 :    20 : False : False : NonNegativeReals
    #       2 :     0 :   8.0 :    20 : False : False : NonNegativeReals
    #       3 :     0 :   9.0 :    20 : False : False : NonNegativeReals
    #       4 :     0 :  10.0 :    20 : False : False : NonNegativeReals
    # W : Size=5, Index=t
    #     Key : Lower : Value : Upper : Fixed : Stale : Domain
    #       0 :     0 :   9.0 :    20 : False : False : NonNegativeReals
    #       1 :     0 :   8.0 :    20 : False : False : NonNegativeReals
    #       2 :     0 :   7.0 :    20 : False : False : NonNegativeReals
    #       3 :     0 :   6.0 :    20 : False : False : NonNegativeReals
    #       4 :     0 :   5.0 :    20 : False : False : NonNegativeReals
```

AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

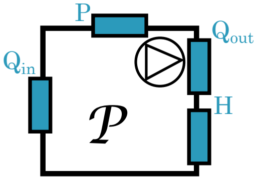
EXAMPLE 1

Two Reservoirs with Irrigation + two Pumps connected to Grid w/ time-dependent cost

Optimization: $\min \sum_{k=0}^{k=T} -P_{grid}(k) \cdot cost(k)\, \Delta k$ w/ $cost = [10, 5, 1, 5, 10]$



Pump



Var	Desc.	Units
$\mathbf{A}$, $\mathbf{B}$, η	Pump params	
$\mathbf{n_n}$, $\mathbf{Q_n}$	Nominal	
$Q_{in}(t)$, $Q_{out}(t)$	I/O flow	[m^3/s]
$H(t)$	Head	[m]
$n(t)$	Speed	[rad/s]
$P_h(t)$, $P_e(t)$	Power	[W]

Port	Var
Q_{in}, Q_{out}	$Q_{in}(t)$, $Q_{out}(t)$
H	$H(t)$
P	$P_e(t)$

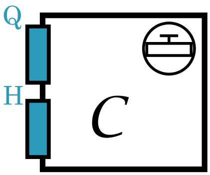
Equations (etc.)

$$Q_{in}^{[k]} = -Q_{out}^{[k]}$$

$$H^{[k]} = \frac{n^{2\,[k]}}{n_n^2} \cdot A + B \cdot Q_{out}^{2\;[k]}$$

AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

Pipe



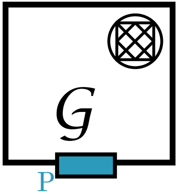
Var	Desc.	Units
$\mathbf{H_0}$	Height	[m]
$\mathbf{K}$	Losses cf.	
$\mathbf{Q_{max}}$	Max. flow	$[m^3/s]$
$Q(t)$	Flow	$[m^3/s]$
$H(t)$	Head	[m]

Port	Var
Q	$Q(t)$
H	$H(t)$

Equations (etc.)

$$H^{[k]} = H_0 + K \cdot Q^{2\,[k]}$$

Grid



Var	Desc.	Units
$\mathbf{P_{max}}$	Max power	[W]
$P(t)$	Power	[W]

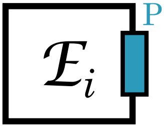
Port	Var
P	$P(t)$

Equations:

No constraints defined

AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

Hub[1]



Var	Desc.	Units
$P_{bal(t)}$	Power balance	[W]

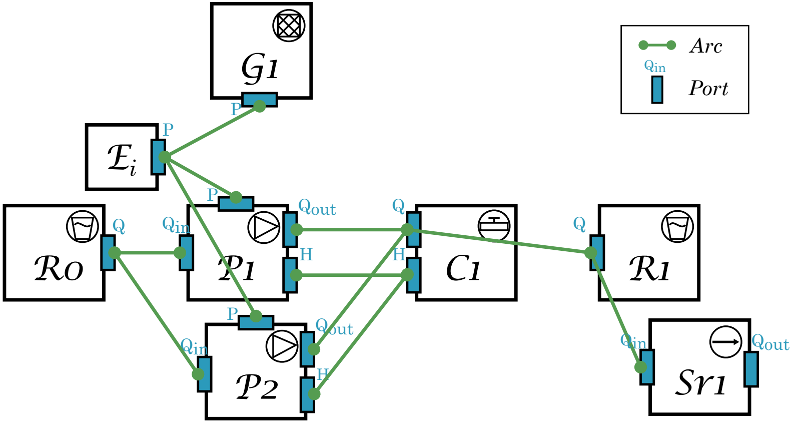
Port	Var
P	$P_{bal}(t)$

Equations:

$$P_{bal}^{[k]} = 0$$

[1] E for *Estació de Bombeig* (Pumping Station) in catalan. It is meant to be a *Node* block to define a power balance (i.e.: sum of powers equal to 0)

Connect *Blocks*' *Ports* with *Arcs*



Introduction
Example 0
Example 1
Example 2
Example 3

Create the *Blocks*

```
# ===== Create the system =====
m.Reservoir1 = pyo.Block()
m.Reservoir0 = pyo.Block()
m.Irrigation1 = pyo.Block()
m.Pump1 = pyo.Block()
m.Pump2 = pyo.Block()
m.Pipe1 = pyo.Block()
m.Grid = pyo.Block()
m.EB = pyo.Block()
```

```
data_irr = {'Q':[2,1,1,1,1]} # irrigation
Source(m.Irrigation1, m.t, data_irr, None)

data_res = {'W0':5, 'Wmin':0, 'Wmax':20} # reservoirs (both equal)
init_res = {'Q':[0,0,0,0,0], 'W':[5,5,5,5,5]}
Reservoir_Ex0(m.Reservoir1, m.t, data_res, init_res)
Reservoir_Ex0(m.Reservoir0, m.t, data_res, init_res)

data_c1 = {'H0':20, 'K':0.05, 'Qmax':50} # canal
init_c1 = {'Q':[0,0,0,0,0], 'H':[20,20,20,20,20]}
Pipe_Ex0(m.Pipe1, m.t, data_c1, init_c1)

data_p = {'A':50, 'B':0.1, 'nmax':1, 'eff':0.9, 'Qmax':4, 'Qnom':5, 'Pmax':9810*50*20} # pumps (both equal)
init_p = {'Q':[0,0,0,0,0], 'H':[20,20,20,20,20], 'Pe'
      :[9810*0.1*20,9810*0.1*20,9810*0.1*20,9810*0.1*20,9810*0.1*20]}
Pump(m.Pump1, m.t, data_p, init_p)
Pump(m.Pump2, m.t, data_p, init_p)

Grid(m.Grid, m.t, {'Pmax':100e3}, None) # grid
EB(m.EB, m.t, None, None) # node
```

CITCEA

Example 1

AGISTIN

CITCEA-UPC

Introduction
Example 0
Example 1
Example 2
Example 3

Create the *Arcs*

```
# Connections
m.p1r0 = Arc(ports=(m.Pump1.port_Qin, m.Reservoir0.port_Q), directed=True)
m.p1c1_Q = Arc(ports=(m.Pump1.port_Qout, m.Pipe1.port_Q), directed=True)
m.p1c1_H = Arc(ports=(m.Pump1.port_H, m.Pipe1.port_H), directed=True)
m.p2r0 = Arc(ports=(m.Pump2.port_Qin, m.Reservoir0.port_Q), directed=True)
m.p2c1_Q = Arc(ports=(m.Pump2.port_Qout, m.Pipe1.port_Q), directed=True)
m.p2c1_H = Arc(ports=(m.Pump2.port_H, m.Pipe1.port_H), directed=True)
m.c1r1 = Arc(ports=(m.Pipe1.port_Q, m.Reservoir1.port_Q), directed=True)
m.r1i1 = Arc(ports=(m.Irrigation1.port_Qin, m.Reservoir1.port_Q), directed=True)
m.EBp1 = Arc(ports=(m.Pump1.port_P, m.EB.port_P), directed=True)
m.EBp2 = Arc(ports=(m.Pump2.port_P, m.EB.port_P), directed=True)
m.EBgrid = Arc(ports=(m.Grid.port_P, m.EB.port_P), directed=True)

pyo.TransformationFactory("network.expand_arcs").apply_to(m) # apply arcs to model
```

Example 1

Run the optimization[2] $cost = [10, 5, 1, 5, 10]$

```
#%% RUN THE OPTIMIZATION

# Objective function
def obj_fun(m):
    return sum(-m.Grid.P[t]*m.cost[t] for t in l_t)
m.goal = pyo.Objective(rule=obj_fun, sense=pyo.minimize)

instance = m.create_instance()
solver = pyo.SolverFactory('ipopt')
results = solver.solve(instance, tee=True)
```

Results

```
    # P : Size=5, Index=t
    #     Key : Lower     : Value                 : Upper    : Fixed : Stale : Domain
    #       0 : -100000.0 :  9.31716873473775e-09 : 100000.0 : False : False :  Reals
    #       1 : -100000.0 :     -59037.07788285201 : 100000.0 : False : False :  Reals
    #       2 : -100000.0 :              -100000.0 : 100000.0 : False : False :  Reals
    #       3 : -100000.0 :    -59037.077882781225 : 100000.0 : False : False :  Reals
    #       4 : -100000.0 : 9.317168818435555e-09 : 100000.0 : False : False :  Reals
```

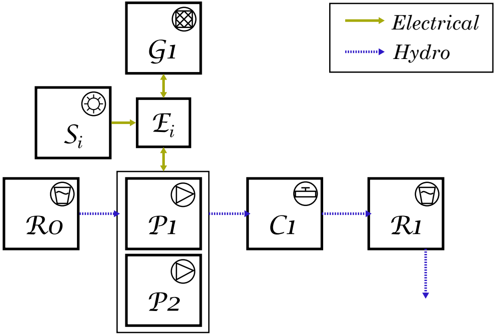
[2] Notice that $P < 0$ implies power injected and $P > 0$ power consumed by an element. E.g.: Negative grid power means power consumed from the grid

EXAMPLE 2

Add a solar PV plant to Example 1

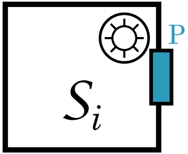
Optimization: $\min \sum_{k=0}^{k=T} \left[P_{buy,grid}(k) \cdot C_{buy}(k) - P_{sell,grid}(k) \cdot C_{sell}(k)\right] \Delta k + P_{inst,PV} \cdot C_{newPV}$

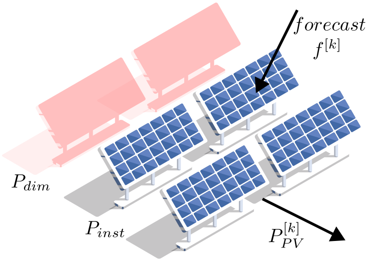
with: $C_{buy} = [10, 5, 1, 5, 10]$, $C_{sell} = \frac{C_{buy}}{2}$ and $C_{newPV} = 10$



Example 2

Solar PV plant





Var	Desc.	Units
$\mathbf{P_{inst}}$	Installed power	[W]
$\mathbf{P_{max}}$	Max available power	[W]
$\mathbf{f(t)}$	Forecast	[p.u.]
η	Efficiency	[p.u.]
$P(t)$	Power	[W]
P_{dim}	Dimensioned power[a]	[W]

Port	Var
P	$P(t)$

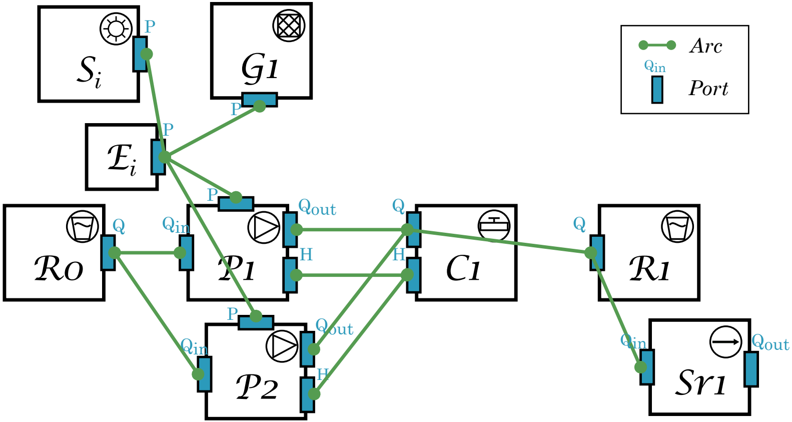
Equations

$$P_{max} \geq P_{inst} + P_{dim}$$

$$P^{[k]} \geq -(P_{inst} + P_{dim}) \cdot f^{[k]} \cdot \eta$$

[a] Refers to new power to install

Connect *Blocks*' *Ports* with *Arcs*



Create the *Blocks*

```
data_irr = {'Q':[2,1,1,1,1]} # irrigation
Source(m.Irrigation1, m.t, data_irr, {})

data_res = {'W0':5, 'Wmin':0, 'Wmax':20} # reservoirs (both equal)
init_res = {'Q':[0,0,0,0,0], 'W':[5,5,5,5,5]}
Reservoir_Ex0(m.Reservoir1, m.t, data_res, init_res)
Reservoir_Ex0(m.Reservoir0, m.t, data_res, init_res)

data_c1 = {'H0':20, 'K':0.05, 'Qmax':50} # canal
init_c1 = {'Q':[0,0,0,0,0], 'H':[20,20,20,20,20]}
Pipe_Ex0(m.Pipe1, m.t, data_c1, init_c1)

data_p = {'A':50, 'B':0.1, 'nmax':1, 'eff':0.9, 'Qmax':4, 'Qnom':5, 'Pmax':9810*50*20} # pumps (both equal)
init_p = {'Q':[0,0,0,0,0], 'H':[20,20,20,20,20], 'Pe'
      :[9810*0.1*20,9810*0.1*20,9810*0.1*20,9810*0.1*20,9810*0.1*20]}
Pump(m.Pump1, m.t, data_p, init_p)
Pump(m.Pump2, m.t, data_p, init_p)

data_pv = {'Pinst':50e3, 'Pmax':100e3, 'forecast':[0.0,0.2,0.8,1.0,0.1], 'eff':0.98} # PV
SolarPV(m.PV, m.t, data_pv)

Grid(m.Grid, m.t, {'Pmax':100e3}) # grid

EB(m.EB, m.t)
```

AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

Create the *Arcs*

```
# Connections
m.p1r0 = Arc(ports=(m.Pump1.port_Qin, m.Reservoir0.port_Q), directed=True)
m.p1c1_Q = Arc(ports=(m.Pump1.port_Qout, m.Pipe1.port_Q), directed=True)
m.p1c1_H = Arc(ports=(m.Pump1.port_H, m.Pipe1.port_H), directed=True)
m.p2r0 = Arc(ports=(m.Pump2.port_Qin, m.Reservoir0.port_Q), directed=True)
m.p2c1_Q = Arc(ports=(m.Pump2.port_Qout, m.Pipe1.port_Q), directed=True)
m.p2c1_H = Arc(ports=(m.Pump2.port_H, m.Pipe1.port_H), directed=True)
m.c1r1 = Arc(ports=(m.Pipe1.port_Q, m.Reservoir1.port_Q), directed=True)
m.r1i1 = Arc(ports=(m.Irrigation1.port_Qin, m.Reservoir1.port_Q), directed=True)
m.ebp1 = Arc(ports=(m.Pump1.port_P, m.EB.port_P), directed=True)
m.ebp2 = Arc(ports=(m.Pump2.port_P, m.EB.port_P), directed=True)
m.grideb = Arc(ports=(m.Grid.port_P, m.EB.port_P), directed=True)
m.pveb = Arc(ports=(m.PV.port_P, m.EB.port_P), directed=True)

pyo.TransformationFactory("network.expand_arcs").apply_to(m) # apply arcs to model
```

Run the optimization

```
#%% RUN THE OPTIMIZATION

# Objective function
def obj_fun(m):
    return sum((m.Grid.Pbuy[t]*m.cost[t] - m.Grid.Psell[t]*m.cost[t]/2) for t in l_t ) + m.PV.Pdim*cost_new_pv
m.goal = pyo.Objective(rule=obj_fun, sense=pyo.minimize)

instance = m.create_instance()
solver = pyo.SolverFactory('ipopt')
results = solver.solve(instance, tee=True)
```

AGISTIN

CITCEA-UPC

Introduction
Example 0
Example 1
Example 2
Example 3

Results

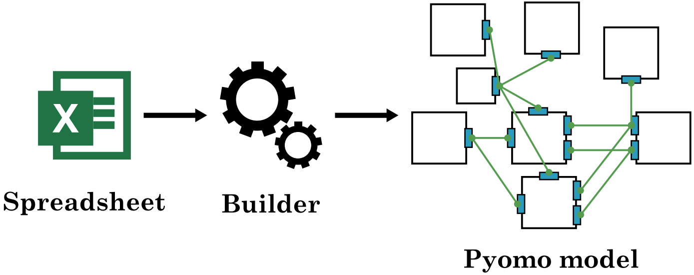
```
# RESULTS
    # W : Size=5, Index=t
    #     Key : Lower : Value              : Upper : Fixed : Stale : Domain
    #       0 :     0 :  3.000000000000006 :    20 : False : False : NonNegativeReals
    #       1 :     0 :  2.054209870661257 :    20 : False : False : NonNegativeReals
    #       2 :     0 : 1.7289984025070158 :    20 : False : False : NonNegativeReals
    #       3 :     0 : 0.9999999899999171 :    20 : False : False : NonNegativeReals
    #       4 :     0 :                0.0 :    20 : False : False : NonNegativeReals
    # W : Size=5, Index=t
    #     Key : Lower : Value              : Upper : Fixed : Stale : Domain
    #       0 :     0 :  4.999999999999994 :    20 : False : False : NonNegativeReals
    #       1 :     0 :  4.945790129338743 :    20 : False : False : NonNegativeReals
    #       2 :     0 :  4.271001597492984 :    20 : False : False : NonNegativeReals
    #       3 :     0 : 4.0000000100000825 :    20 : False : False : NonNegativeReals
    #       4 :     0 :  4.000000009999999 :    20 : False : False : NonNegativeReals
    # P : Size=5, Index=t
    #     Key : Lower     : Value                  : Upper    : Fixed : Stale : Domain
    #       0 : -100000.0 :  8.508149532847933e-09 : 100000.0 : False : False :  Reals
    #       1 : -100000.0 : -6.103573816990847e-09 : 100000.0 : False : False :  Reals
    #       2 : -100000.0 :              -100000.0 : 100000.0 : False : False :  Reals
    #       3 : -100000.0 : -5.986341529814071e-09 : 100000.0 : False : False :  Reals
    #       4 : -100000.0 :      5908.919313237837 : 100000.0 : False : False :  Reals
    # Pdim : Size=1, Index=None
    #     Key  : Lower : Value              : Upper   : Fixed : Stale : Domain
    #     None :     0 : 10295.094533227224 : 50000.0 : False : False : NonNegativeReals
```

Introduction
Example 0
Example 1
Example 2
Example 3

EXAMPLE 3

Autoloading of Example 2

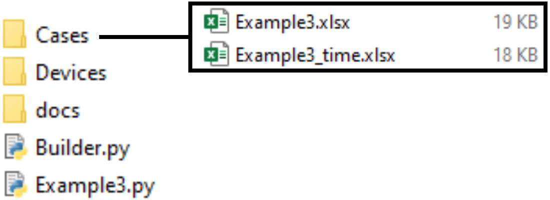
An automatic builder has been added to the project.
The *Builder* reads the data from excel files and generates the Pyomo model.



Example 3

Fill the spreadsheets with the devices' info

- *[Case].xlsx* - Definition and parameters
- *[Case]_time.xlsx* - Time dependent values



- *Example3.xlsx*

	A	B	C	D	E	F	G	H	I
1	Name	A	B	Pmax	Qmax	Qnom	n_n	eff	CONNECTION
2	Pump1	50	0.1	9.81E+06	20	5	1450	0.9	P,EB,P;Qout,Pipe1,Q;H,Pipe1,H;Qin,Reservoir0,Q;
3	Pump2	50	0.1	9.81E+06	20	5	1450	0.9	P,EB,P;Qout,Pipe1,Q;H,Pipe1,H;Qin,Reservoir0,Q;

Grid | Reservoir_Ex0 | Pipe_Ex0 | **Pump** | Source | EB | SolarPV | Reservoir | Hydr

- *Example3_time.xlsx*

	A	B	C	D	E	F	G	H
1	Pump1_Q	Pump1_H	Pump1_n	Pump1_Pe	Pump2_Q	Pump2_H	Pump2_n	Pump2_Pe
2	0	20	1450	9.81E+05	0	20	1450	9.81E+05
3	0	20	1450	9.81E+05	0	20	1450	9.81E+05
4	0	20	1450	9.81E+05	0	20	1450	9.81E+05
5	0	20	1450	9.81E+05	0	20	1450	9.81E+05
6	0	20	1450	9.81E+05	0	20	1450	9.81E+05

Grid | Reservoir_Ex0 | Pipe_Ex0 | **Pump** | Source | EB | SolarPV | Reservoir | Hydr

CONNECTION refers to the *Arcs*

AGISTIN
CITCEA-UPC
Introduction
Example 0
Example 1
Example 2
Example 3

Run the *Builder*

```
from Utilities import clear_clc

#clean console and variable pane
# clear_clc() #consider removing if you are not working with Spyder

data_filename = "Example3"

# generate system json file
data_parser(data_filename, dt=1) # dt = value of each timestep (if using SI this is seconds)

m = pyo.ConcreteModel()

# time
l_t = list(range(5)) #TODO this should be inferred from the number of rows in the excel time series,
#TODO it would be nice to have a consistency check ensuring that data has been correctly filled in all sheets.
```

Introduction
Example 0
Example 1
Example 2
Example 3

Run the optimization

```
#%% RUN THE OPTIMIZATION

# Objective function
def obj_fun(m):
    return sum((m.Grid.Pbuy[t]*m.cost[t] - m.Grid.Psell[t]*m.cost[t]/2) for t in l_t ) + m.PV.Pdim*cost_new_pv
m.goal = pyo.Objective(rule=obj_fun, sense=pyo.minimize)

instance = m.create_instance()
solver = pyo.SolverFactory('ipopt')
results = solver.solve(instance, tee=True)
```

Example 3

You should get the same results as in Example 2

```
instance.Reservoir1.W.pprint()
instance.Reservoir0.W.pprint()
instance.Grid.P.pprint()
instance.PV.Pdim.pprint()

# RESULTS
    # W : Size=5, Index=t
    #     Key : Lower : Value              : Upper : Fixed : Stale : Domain
    #       0 :     0 :  3.000000000000006 :    20 : False : False : NonNegativeReals
    #       1 :     0 :  2.054209870661257 :    20 : False : False : NonNegativeReals
    #       2 :     0 : 1.7289984025070158 :    20 : False : False : NonNegativeReals
    #       3 :     0 : 0.9999999899999171 :    20 : False : False : NonNegativeReals
    #       4 :     0 :                0.0 :    20 : False : False : NonNegativeReals
    # W : Size=5, Index=t
    #     Key : Lower : Value              : Upper : Fixed : Stale : Domain
    #       0 :     0 :  4.999999999999994 :    20 : False : False : NonNegativeReals
    #       1 :     0 :  4.945790129338743 :    20 : False : False : NonNegativeReals
    #       2 :     0 :  4.271001597492984 :    20 : False : False : NonNegativeReals
    #       3 :     0 : 4.0000000100000825 :    20 : False : False : NonNegativeReals
    #       4 :     0 :  4.000000009999999 :    20 : False : False : NonNegativeReals
    # P : Size=5, Index=t
    #     Key : Lower     : Value                  : Upper    : Fixed : Stale : Domain
    #       0 : -100000.0 :  8.508149532847933e-09 : 100000.0 : False : False :  Reals
    #       1 : -100000.0 : -6.103573816990847e-09 : 100000.0 : False : False :  Reals
    #       2 : -100000.0 :              -100000.0 : 100000.0 : False : False :  Reals
    #       3 : -100000.0 : -5.986341529814071e-09 : 100000.0 : False : False :  Reals
    #       4 : -100000.0 :      5908.919313237837 : 100000.0 : False : False :  Reals
    # Pdim : Size=1, Index=None
    #     Key  : Lower : Value              : Upper   : Fixed : Stale : Domain
    #     None :     0 : 10295.094533227224 : 50000.0 : False : False : NonNegativeReals
```

AGISTIN T4.6 Usage examples of the tool

CITCEA-UPC

January 2025