OPTIMIZATION

Introduction to optimization

What is optimization?

First, define a measurable objective that depends on system characteristics, called variables. The optimization objective is to find the value of the variables that maximize or minimize the objective.

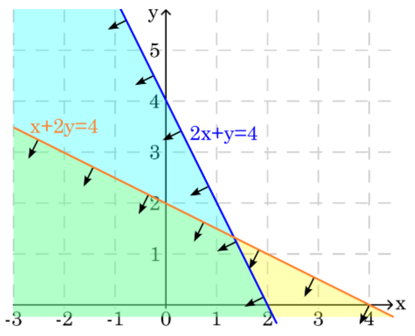
$$\begin{aligned} \min_{x \in R^n} \quad & f(x) && \text{Objective function} \\ s.t. \quad & g(x) \leq 0 && \text{Inequality constraints} \\ & h(x) = 0 && \text{Equality constraints} \end{aligned} \tag{1}$$

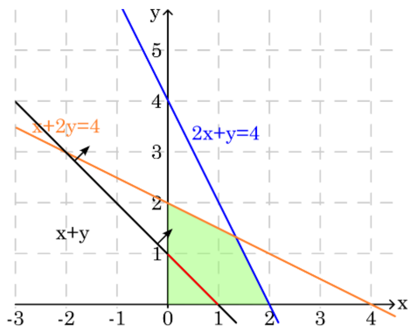
Modelling is the process of identification and definition of objective, variables and constraints. Important to balance precision and simplicity.

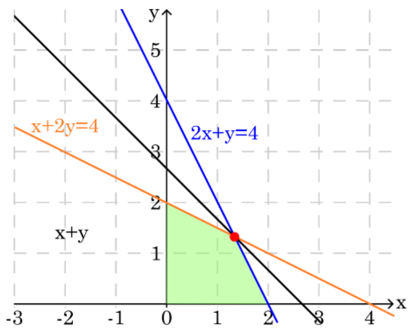
Later, a resolution algorithm is applied, its selection depends on the problem type and affects the solution.

Example:

$$\begin{aligned} \max_{x \in R^n} \quad & x + y \\ \text{subject to} \quad & x + 2y \leq 4 \\ & 2x + y \leq 4 \quad \rightarrow \quad x = 4/3,\ y = 4/3 \\ & x \geq 0 \\ & y \geq 0 \end{aligned} \tag{2}$$



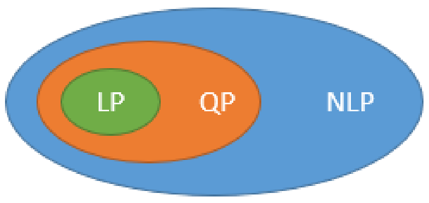




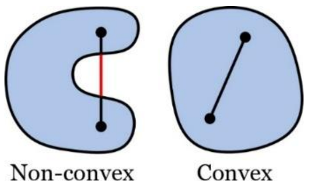
Types of optimization problems

Can be classified according to:

- Variables: Mixed-Integer Programming (MIP)
- Objective function and constraints:
  - linear programming (LP)
  - quadratic programming (QP)
  - non-linear programming (NLP)
- Model uncertainty: deterministic or stochastic
- Multi-objective, if they consider several objective functions:
  - function weighting
  - iterative hierarchical resolution

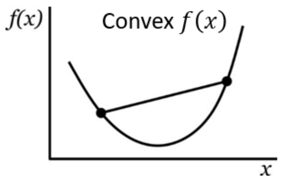


Convexity



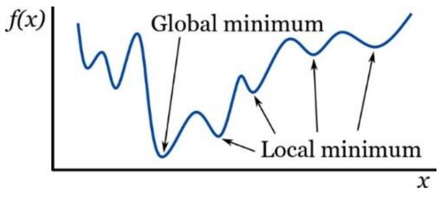
A set is convex if the line between any two points is contained within the set.

A function $f(x)$ is convex if its domain is convex and the line between any two points is above the function.



A function is concave if $-f(x)$ is convex.

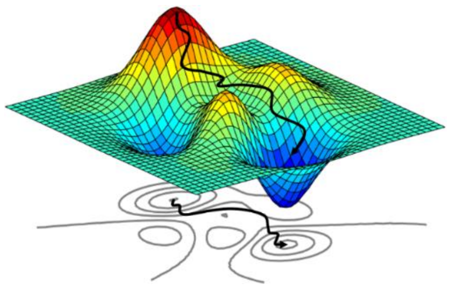
Solutions



There are local and global solutions.

In convex problems, the local solution is also global.

Non-linear problems tend to have local solutions different from the global one.



In general, optimization algorithms are iterative: they start at a point and move toward better values.

The difference between algorithms is the strategy they use to get from one point to another.

Many algorithms tend to find local solutions, since global solutions are difficult to find and identify.

Implementation

There are different softwares to model and solve optimizations, for example:

- **NEOS Server for Optimization** (http://neos-server.org/neos). Open access server to solve optimization problems, includes several optimizers.
- **CPLEX** (from ILOG, trial version at http://www.ibm.com).
- **GUROBI OPTIMIZER** (free academic version at http://www.gurobi.com).
- **GAMS** (modelling language that can link to several optimizers; it is used in NEOS Server).
- **AMPL** (platform where you can get access to several open source and commercial solvers; https://ampl.com/products/solvers/open-source-solvers/)

Implementation in Python

Pyomo

Pyomo is a Python-based open-source software package that supports a diverse set of optimization capabilities for formulating, solving, and analysing optimization models.

Its documentation can be found in https://pyomo.readthedocs.io/en/stable/

Pyomo generates the optimization model inside a single Python object. The first step is to create this variable.

```
import pyomo.environ as pyo
model = pyo.ConcreteModel()
```

Pyomo establishes 5 main modelling components: sets, parameters, variables, constraints, and objectives.

Set. Indicator that works as an index in the model to make it easily scalable. Here, a list with the values that the index can have has to be provided. For example, the time. So, the model is written just once in a general time step.

```
import pyomo.environ as pyo
model.t = pyo.Set(initialize=[0, 1, 2, 3, 4])
```

Parameter. Data used to define a model. It must be provided the value of the parameter, its domain (Reals, Integers, NonNegativeReals, Binary, etc.), and the applicable sets. The format to introduce a parameter with sets is through a dictionary whose keys are the set values.

```
model.Lprj = pyo.Param(initialize=25, within=pyo.NonNegativeReals)
model.PV_capex = pyo.Param(model.i_PV, initialize={i_PV: value}, within=pyo.NonNegativeReals)
model.PV_forecast = pyo.Param(model.i_PV, model.t, initialize={(i_PV, t): value}, within=pyo.NonNegativeReals)
```

Variable. Decision variable in a model. Here, it must be provided the domain (Reals, Integers, NonNegativeReals, Binary, etc.), and also the applicable sets. Optionally, lower and upper bounds can be defined, and also an initial value that will be considered for the first iteration of the solver. The format of variables with sets is a dictionary whose keys are the set values.

```
model.total_cost = pyo.Var(within=pyo.NonNegativeReals)
model.total_cost = pyo.Var(within=pyo.NonNegativeReals, bounds=(0,100000), initialize=0)
model.PV_C_capex = pyo.Var(model.i_PV, within=pyo.NonNegativeIntegers)
model.PV_P = pyo.Var(model.i_PV, model.t, within=pyo.NonNegativeReals)
```

Constraint. Expression that imposes restrictions on variable values.

```
def Constraint_total_cost(m):
    return m.total_cost == sum(m.PV_C_capex[i_PV] for i_PV in list_PV)
model.Constr_total_cost = pyo.Constraint(rule=Constraint_total_cost)
def Constraint_PV_cost(m, i_PV):
    return m.PV_C_capex[i_PV] == m.PV_capex[i_PV] * m.PV_G[i_PV]
model.Constr_PV_cost = pyo.Constraint(model.i_PV, rule=Constraint_PV_cost)
def Constraint_PV_generation(m, i_PV, t):
    return m.PV_P[i_PV, t] <= m.PV_G[i_PV] * m.PV_forecast[i_PV, t]
model.Constr_PV_generation = pyo.Constraint(model.i_PV, model.t, rule=Constraint_PV_generation)
```

Objective. Expression that is minimized or maximized in a model. For example, we aim to minimize the sum of all costs.

```
def func_obj(m):
    return m.total_cost
model.goal = pyo.Objective(rule=func_obj, sense=pyo.minimize)
```

After writing the model, it is needed to call an external solver. Pyomo supports several solvers both opensource (GLPK, IPOPT, CBC…) and commercial (BARON, Gurobi, CPLEX…). For linear problems I recommend you try highs (through AMPL, see next section) or Gurobi.

```
opt = pyo.SolverFactory('highs')  # gurobi
```

Finally, the solver is applied to our model:

```
results = opt.solve(model)
```

To access the values of a variable, parameter or objective funcion after running the optimization, it is used:

```
model.varName.get_values()
model.paramName.value
pyo.value(instance.goal)
```

As mentioned before, variables and parameters have a dictionary format. If the variable or parameter has a single value, then it is expressed as a dictionary with key `None`.

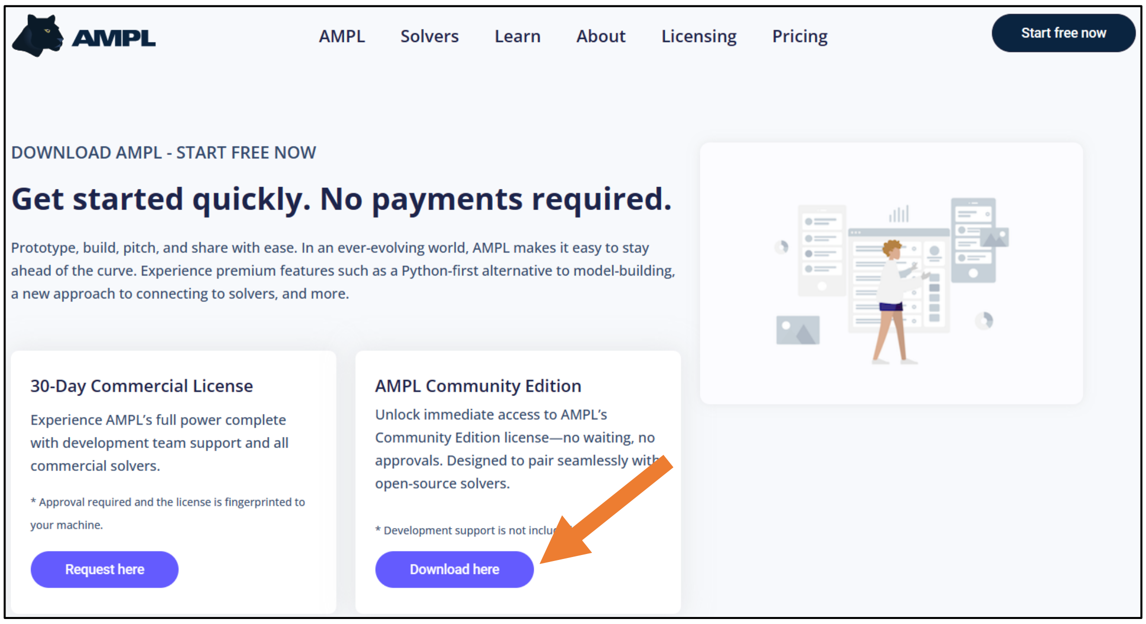
How to install solvers with AMPL

AMPL includes commercial and open-source solvers. By using this platform, it's easier to install different solvers.

Go to the following website:

https://ampl.com/start-free-now/

Click *Download here*



DOWNLOAD AMPL - START FREE NOW

Get started quickly. No payments required.

Prototype, build, pitch, and share with ease. In an ever-evolving world, AMPL makes it easy to stay ahead of the curve. Experience premium features such as a Python-first alternative to model-building, a new approach to connecting to solvers, and more.

30-Day Commercial License

Experience AMPL's full power complete with development team support and all commercial solvers.

* Approval required and the license is fingerprinted to your machine.

Request here

AMPL Community Edition

Unlock immediate access to AMPL's Community Edition license—no waiting, no approvals. Designed to pair seamlessly with open-source solvers.

* Development support is not inclu

Download here

Sign up with e-mail

Download AMPL installer

Open the downloaded executable and proceed with the installation.

Example

Energetic exercise: we have an electrical load supplied with PV, gas and nuclear power. Find the size of the PV, gas and nuclear power plants that minimizes the total cost. Assumptions:

- **Nuclear power is constant**
- **Gas power can change each hour**
- **PV can be curtailed**
- **24 h optimization**
- **Project lifetime of 40 years**

	CAPEX	**OPEX**	**Replacement**
Nuclear	**3600 €/kW**	**0.02 €/kWh**	**-**
Gas	**823 €/kW**	**0.15 €/kWh**	**-**
PV	**650 €/kW**	**0.13 €/kW**	**325 €/kW (20 years)**

Parameters:

- Electrical load ($P_{load-1}, P_{load-2}, \ldots$) in kW,
- Available PV generation in kW for 1 kWp installed ($P_{PVforecast-1}, P_{PVforecast-2}, \ldots$)

Variables:

- hourly nuclear power ($P_{nuc-1}, P_{nuc-2}, \ldots$),
- nuclear power plant (P_{pnuc}),
- hourly gas power ($P_{gas-1}, P_{gas-2}, \ldots$),
- gas power plant (P_{pgas}),
- hourly PV power ($P_{pv-1}, P_{pv-2}, \ldots$),
- PV power plant (P_{pv}).

All variables are defined positive, and in kW.

Constraints:

- Power balance: total generation = demand at each hour

$$P_{nuc-t} + P_{gas-t} + P_{pv-t} = P_{load-t} \quad \forall t \in \{1,2 \ldots 24\} \tag{3}$$

- Nuclear power plant sizing

$$P_{nuc-t} = P_{pnuc} \quad \forall t \in \{1,2 \ldots 24\} \tag{4}$$

- Gas power plant sizing

$$P_{gas-t} \leq P_{pgas} \quad \forall t \in \{1,2 \ldots 24\} \tag{5}$$

- PV power plant sizing

$$P_{pv-t} \leq P_{pv} \cdot P_{PVforecast-t} \quad \forall t \in \{1,2 \ldots 24\} \tag{6}$$

Objective function: minimize sum of costs in 40 years

$$CAPEX_{nuc} = P_{nuc} \cdot 3600 \tag{7}$$

$$OPEX_{nuc} = P_{nuc} \cdot 0.02 \cdot 8760 \tag{8}$$

$$CAPEX_{gas} = P_{pgas} \cdot 823 \tag{9}$$

$$OPEX_{gas} = 365 \cdot \sum_{t=1}^{24} P_{gas-t} \cdot 1 \cdot 0.15 \tag{10}$$

$$CAPEX_{pv} = P_{pv} \cdot 650 \tag{11}$$

$$OPEX_{pv} = P_{pv} \cdot 0.13 \tag{12}$$

$$Replacement_{pv} = P_{pv} \cdot 325 \cdot 1 \tag{13}$$

$$\begin{aligned} \min \quad & CAPEX_{nuc} + OPEX_{nuc} \cdot 40 + CAPEX_{gas} + OPEX_{gas} \cdot 40 + \\ & +CAPEX_{pv} + OPEX_{pv} \cdot 40 + Replacement_{pv} \end{aligned} \tag{14}$$

The Python code that solves this exercise can be found in *example.py*, the load [MWh] and PV forecast [kWh/kWp] are given in *data.xlsx*. The sizing results are printed, and the system operation is saved in *Results.xlsx*, where each row corresponds to one hour.

```
model = pyo.ConcreteModel()

model.t = pyo.Set(initialize=l_t)

model.Ppnuc = pyo.Var(within=pyo.NonNegativeReals)
model.Pnuc_t = pyo.Var(model.t, within=pyo.NonNegativeReals)
model.Ppgas = pyo.Var(within=pyo.NonNegativeReals)
model.Pgas_t = pyo.Var(model.t, within=pyo.NonNegativeReals)
model.Ppv = pyo.Var(within=pyo.NonNegativeReals)
model.Ppv_t = pyo.Var(model.t, within=pyo.NonNegativeReals)

def Constraint_balance(m, t):
    return m.Pnuc_t[t] + m.Pgas_t[t] + m.Ppv_t[t] == Pload[t]
model.Constr_balance = pyo.Constraint(model.t, rule=Constraint_balance)

def Constraint_nuclear_size(m, t):
    return m.Pnuc_t[t] == m.Ppnuc
model.Constr_nuclear_size = pyo.Constraint(model.t, rule=Constraint_nuclear_size)

def Constraint_gas_size(m, t):
    return m.Pgas_t[t] <= m.Ppgas
model.Constr_gas_size = pyo.Constraint(model.t, rule=Constraint_gas_size)

def Constraint_PV_size(m, t):
    return m.Ppv_t[t] <= m.Ppv * P_PVforecast[t]
model.Constr_PV_size = pyo.Constraint(model.t, rule=Constraint_PV_size)

def func_obj(m):
    years = 40
    CAPEX_nuc = m.Ppnuc * 3600
    OPEX_nuc = m.Ppnuc * 0.02 * 8760
    CAPEX_gas = m.Ppgas * 823
    OPEX_gas = sum(m.Pgas_t[t] * 1 for t in l_t) * 0.15  # for 1 day
    CAPEX_PV = m.Ppv * 650
    OPEX_PV = m.Ppv * 0.13
    replacement_PV = m.Ppv * 325 * 1  # PV should be replaced once
    return CAPEX_nuc + OPEX_nuc * years + CAPEX_gas + OPEX_gas * 365 * years +
CAPEX_PV + OPEX_PV * years + replacement_PV
model.goal = pyo.Objective(rule=func_obj, sense=pyo.minimize)

opt = pyo.SolverFactory('highs')
results = opt.solve(model)
```

Exercises

The following exercises are intended to help you learn Python, Pyomo and import/export data to Excel files.

These exercises focus on PV and energy exchange with the main grid.

1) Solve an easy case (example from introduction section):

$$\begin{aligned} \max_{x \in R^n} \quad & x + y \\ \text{subject to} \quad & x + 2y \leq 4 \\ & 2x + y \leq 4 \\ & x \geq 0 \\ & y \geq 0 \end{aligned} \quad \rightarrow \quad \begin{aligned} x &= 4/3 \\ y &= 4/3 \end{aligned} \tag{2}$$

2) We have two loads of 30 kWh and 10 kWh, respectively, which are connected to the electricity grid with a price of 0.05 €/kWh. Find how much energy is consumed from the grid and its price.

3) We have a domestic consumption of 3 kWh and the photovoltaics on the roof can produce up to 2 kWh in that hour. Assume a price of the photovoltaic energy generated of 0.01 €/kWh and a price of 0.05 €/kWh of the energy purchased from the grid. If we want to minimize the cost, how much energy does the photovoltaic generate? How much energy is purchased from the grid? What is the total cost?

4) We have a domestic consumption of 1 kWh and the photovoltaics on the roof can produce up to 2 kWh in that hour. Assume a price of the photovoltaic energy generated of 0.01 €/kWh and a price of 0.05 €/kWh of the energy purchased from the grid. Energy cannot be injected into the grid. If we want to minimize the cost, how much energy does the photovoltaic generate? How much energy is purchased from the grid? What is the total cost?
Note: photovoltaics can generate less energy than is available (it's called curtailment), and the energy exchange with the grid must be positive (defined as energy purchased)

5) We have a domestic consumption of 1 kWh and the photovoltaics on the roof can produce up to 2 kWh in that hour. Assuming a zero cost for the photovoltaics and a price of 0.05 €/kWh for the energy exchanged with the grid, energy can be injected into the grid. If we want to minimize the cost and maximize the benefits, how much energy does the photovoltaics generate? How much energy is purchased from the grid? How much energy is sold to the grid? What is the total cost or benefit?
*Note: you can assume as objective function: minimize exchange*0.05*
so that if the energy exchange with the network is positive it means that it is bought, and if the energy exchange is negative it means that it is sold.

6) We have a domestic consumption of 1 kWh and the photovoltaics on the roof can produce up to 2 kWh in that hour. Assume a price of the photovoltaic energy generated of 0.01 €/kWh, a price of 0.05 €/kWh of the energy purchased from the grid and a price of 0.005 €/kWh of the energy sold to the grid. If we want to minimize the cost (and maximize profits), how much energy does the photovoltaic generate? How much energy is purchased from the grid? How much energy is sold to the grid? What is the total cost or benefit?
Note: in this case, you must define two positive terms of energy exchange with the network, energy purchased and energy sold, each with its own price.

7) We have a domestic consumption of 1 kWh and the photovoltaics on the roof can produce up to 2 kWh in that hour. Assume a price of the photovoltaic energy generated of 0.01 €/kWh, a price of 0.05 €/kWh of the energy purchased from the grid and a price of 0.025 €/kWh of the energy sold to the grid. If we want to minimize the cost (and maximize profits), how much energy does the photovoltaic generate? How much energy is purchased from the grid? How much energy is sold to the grid? What is the total cost or benefit?

8) We have a domestic consumption and photovoltaics on the roof, the following table shows the values. Let's assume a cost of 0 on the photovoltaics, a price of 0.05 €/kWh of the energy purchased from the grid, and no energy can be injected into the grid. If we want to minimize the cost (and maximize profits), how much energy does the photovoltaic generate each hour? How much energy is purchased from the grid each hour? How much energy is sold to the grid each hour? What is the total cost or benefit?

	Consumption [kWh]	PV available [kWh]
7:00 h	1	0
8:00 h	2	1
9:00 h	2	2
10:00 h	1	3

   a. Solve using the following lines to define the parameters:
```
l_t = list(range(4))
consumption = {0: 1, 1: 2, 2: 2, 3: 1}
PV_max = {0: 0, 1: 0.5, 2: 1, 3: 1.5}
```
   b. Copy the table to excel and import the data, then solve the optimization.

9) We have a domestic consumption and photovoltaics on the roof, assuming the values in the following table and a cost of 0 on the photovoltaics. If we want to minimize the cost (and maximize the benefits), how much energy does the photovoltaics generate each hour? How much energy is purchased from the grid each hour? How much energy is sold to the grid each hour? What is the total cost or benefit?

	Consumption [kWh]	PV available [kWh]	Purchase price from the grid [€/kWh]	Sell price to the grid [€/kWh]
7:00 h	1	0	0,03	0,01
8:00 h	2	1	0,04	0,02
9:00 h	2	2	0,05	0,02
10:00 h	1	3	0,05	0,005

a. Solve using the following lines to define the parameters:

```
l_t = list(range(4))
consumption = {0: 1, 1: 2, 2: 2, 3: 1}
PV_max = {0: 0, 1: 0.5, 2: 1, 3: 1.5}
buy_price = {0: 0.03, 1: 0.04, 2: 0.05, 3: 0.05}
sell_price = {0: 0.01, 1: 0.02, 2: 0.02, 3: 0.005}
```

b. Copy the table to excel and import the data, then solve the optimization.

10) Repeat the energetic example of sizing PV, gas and nuclear power plants, but considering the whole year.
Note: changing only `l_t` length to be 8760 instead of 24 hours is enough.

11) Electrolyser operation exercise. We have a system with a hydrogen load and an electrolyser. The electrolyser needs water and electricity from the grid, and produces both hydrogen and oxygen. Find the operation of the system: electricity purchased, water consumed and oxygen produced at each hour.
Note: you can consider the electrolyser as in our paper, with an efficiency for each conversion. For the hydrogen load and electricity purchase prices, you can try different values as in previous exercises, always considering that the electrolyser size has to be higher or equal than the electrical maximum power.

a. Solve for a single hour
b. Solve for several hours
c. Solve for 1 year

12) Electrolyser sizing exercise. We have a system with a hydrogen load and an electrolyser. The electrolyser needs water and electricity from the grid, and produces both hydrogen and oxygen. Find the electrolyser size.
Note: do the same assumptions as in the previous exercise.