



# **Introducción a la programación matemática**

Isabela Fons

[isabela.fons@ua.es](mailto:isabela.fons@ua.es)

Grupo COnCEPT - Departamento de Ingeniería Química

UA

UNIVERSITAT D'ALACANT  
UNIVERSIDAD DE ALICANTE

Escola Politècnica Superior  
Escuela Politécnica Superior



# Instrucciones

## GoogleColab

Haz click en los siguientes links para acceder al contenido:

- [Python Básico](#)
- [Fundamentos Pyomo](#)
- [Optimización discreta](#)

## Local

```
git clone https://github.com/isfons/intro-pyomo.git
cd intro_pyomo
pip install -e
```

\* Necesaria instalación previa de Git, Python y solvers de optimización.





**¿Por qué Pyomo?**

# Python



Fácil de usar



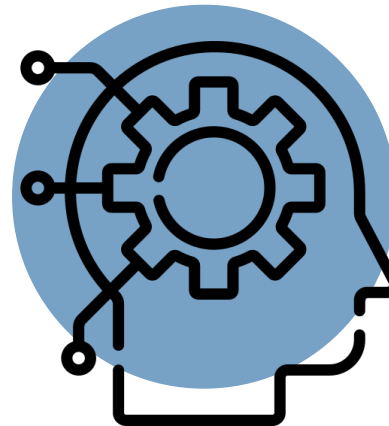
Eficiencia



Multiplataforma



Variedad de  
paquetes

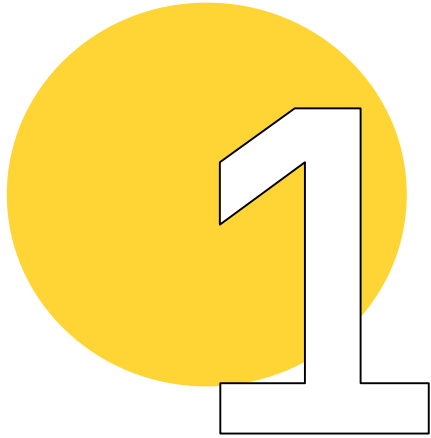


Uso en ML

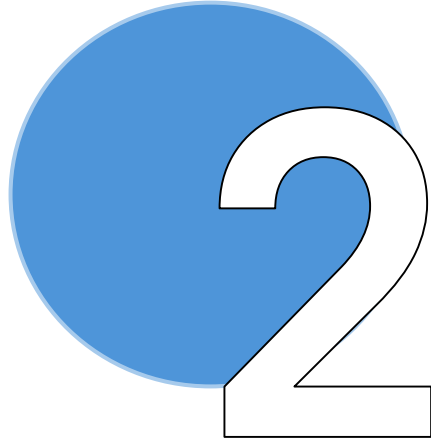


Soporte

# Pyomo



Gratuito y bien  
documentado



Fácil  
instalación



Compatible con  
solvers  
principales



Versatilidad y  
extensiones



**IDAES/idaes-pse**

The IDAES Process Systems Engineering  
Framework



**Pyomo/mpi-sppy**

MPI-based Stochastic Programming in Python





# Programación matemática

Conceptos básicos de optimización

# Motivación

¿Cómo organizar la producción para maximizar la utilización de los equipos?

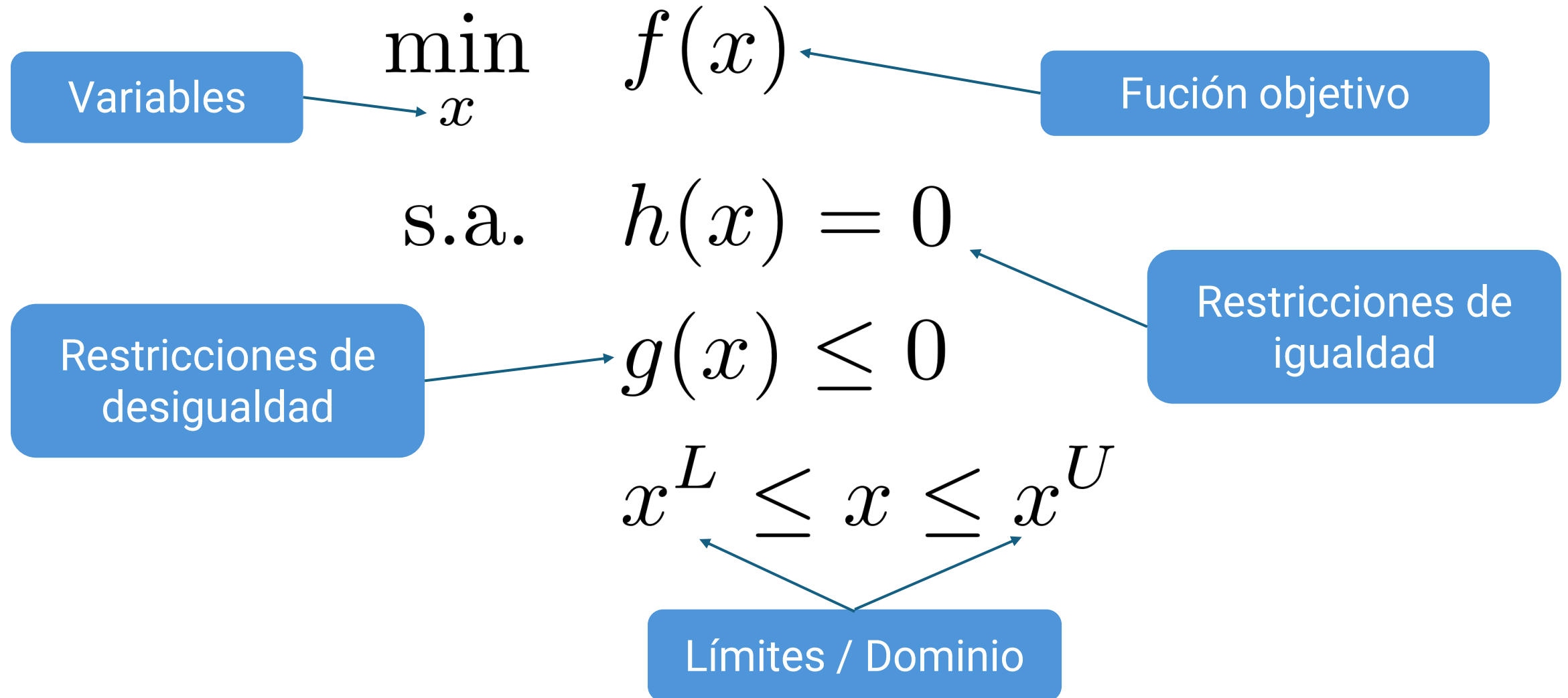
¿Cuál es la mezcla óptima cuando se trabaja con ingredientes naturales?

¿Condiciones de operación que maximizan el beneficio?

¿Cuál es la mejor ruta para entregar los pedidos?



# Elementos básicos



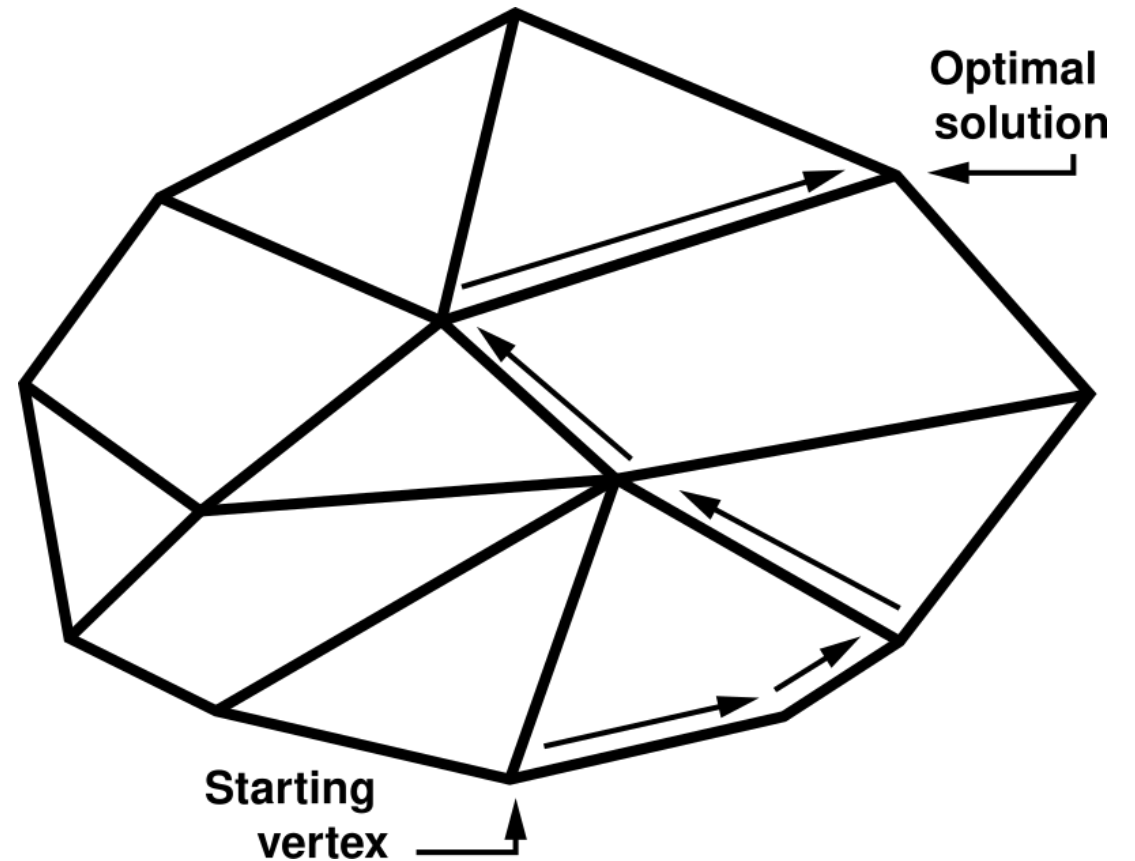
# Clasificación

## Programación lineal (LP)

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.}, \quad & Ax = b \\ & x \geq 0 \\ & x \in \mathbb{R}^n \end{aligned}$$

**Resolución:** Simplex

**Ejemplo:** Problema de transporte



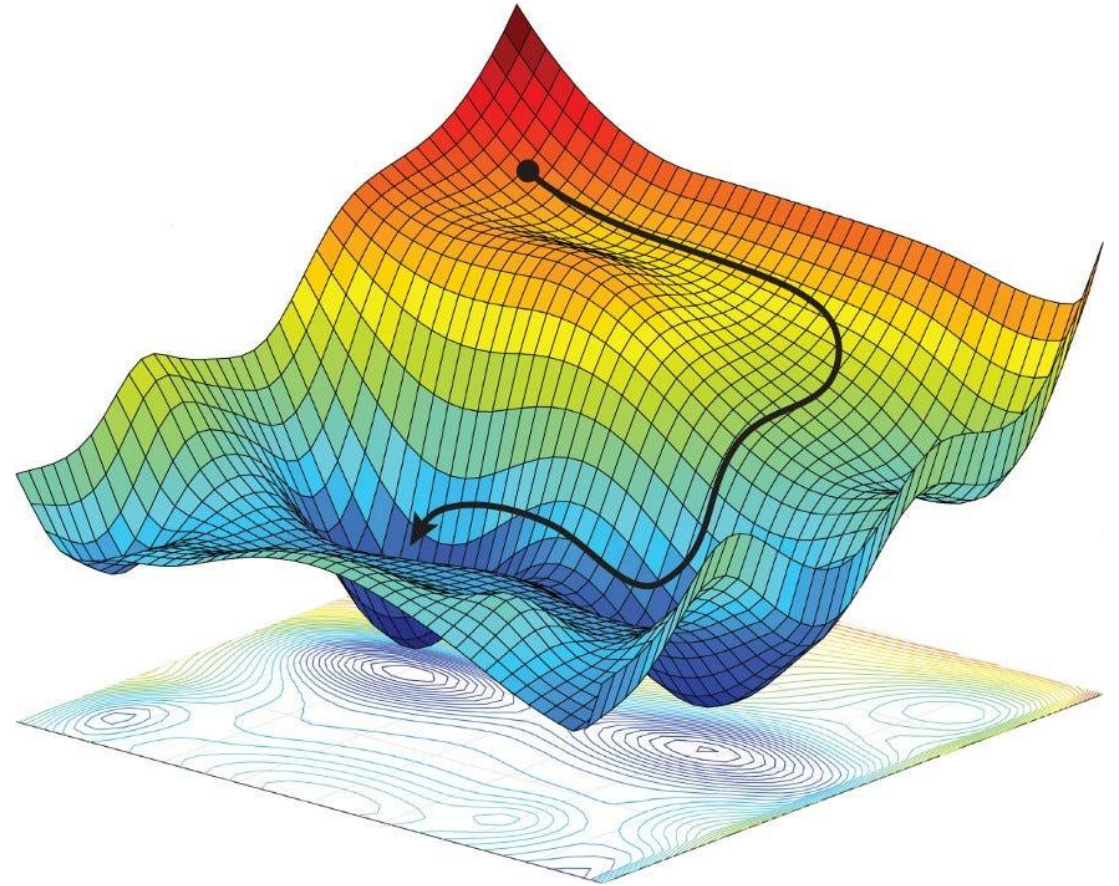
# Clasificación

Programación no lineal (NLP)

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.}, \quad & h(x) = 0 \\ & g(x) \leq 0 \\ & x^L \leq x \leq x^U \end{aligned}$$

**Resolución:** Condiciones KKT

**Ejemplo:** Diseño RCTA

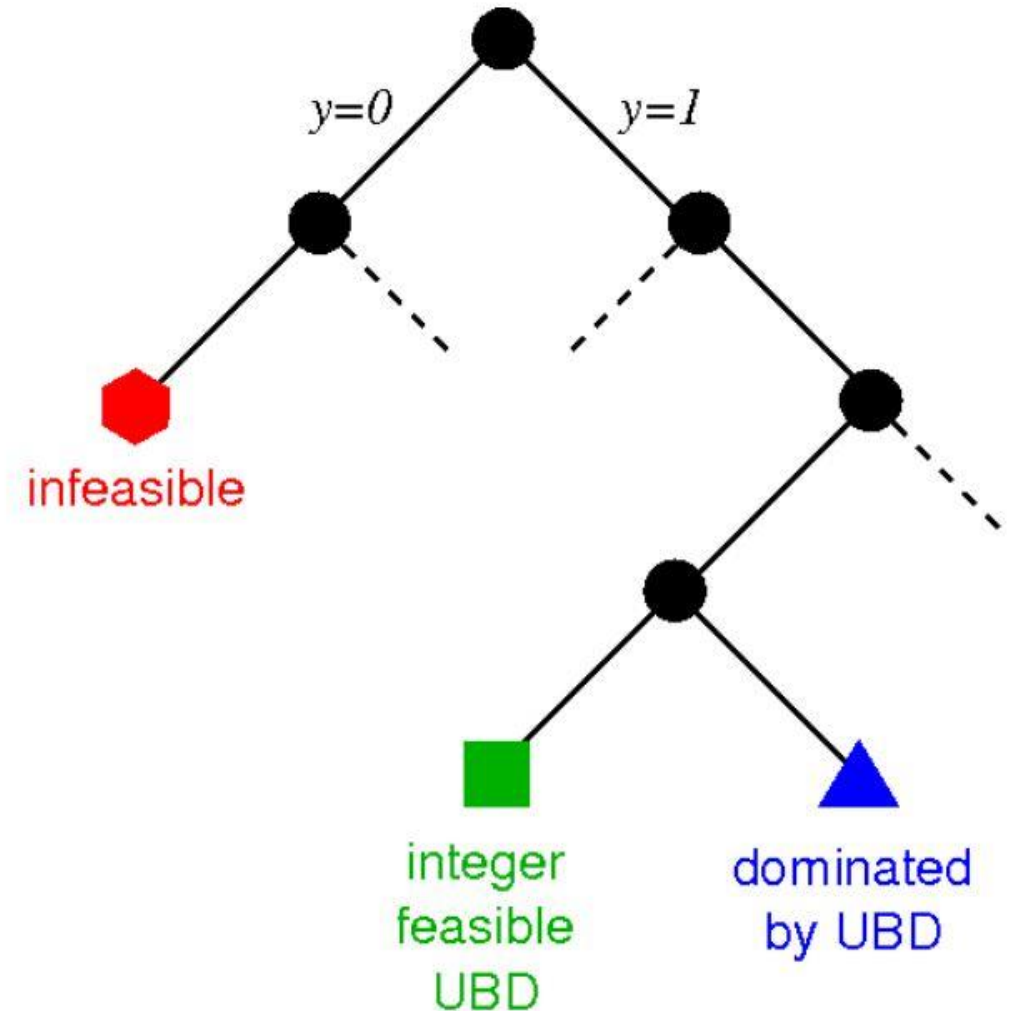


# Clasificación

## Programación lineal entera mixta (MILP)

$$\begin{aligned} \min \quad & c^T x + d^T y \\ \text{s.t.}, \quad & Ax + By \leq b \\ & x \geq 0, \quad x \in \mathbb{R}^n \\ & y \in \{0,1\}^m \end{aligned}$$

**Resolución:** Branch and Bound  
**Ejemplo:** Selección de equipos



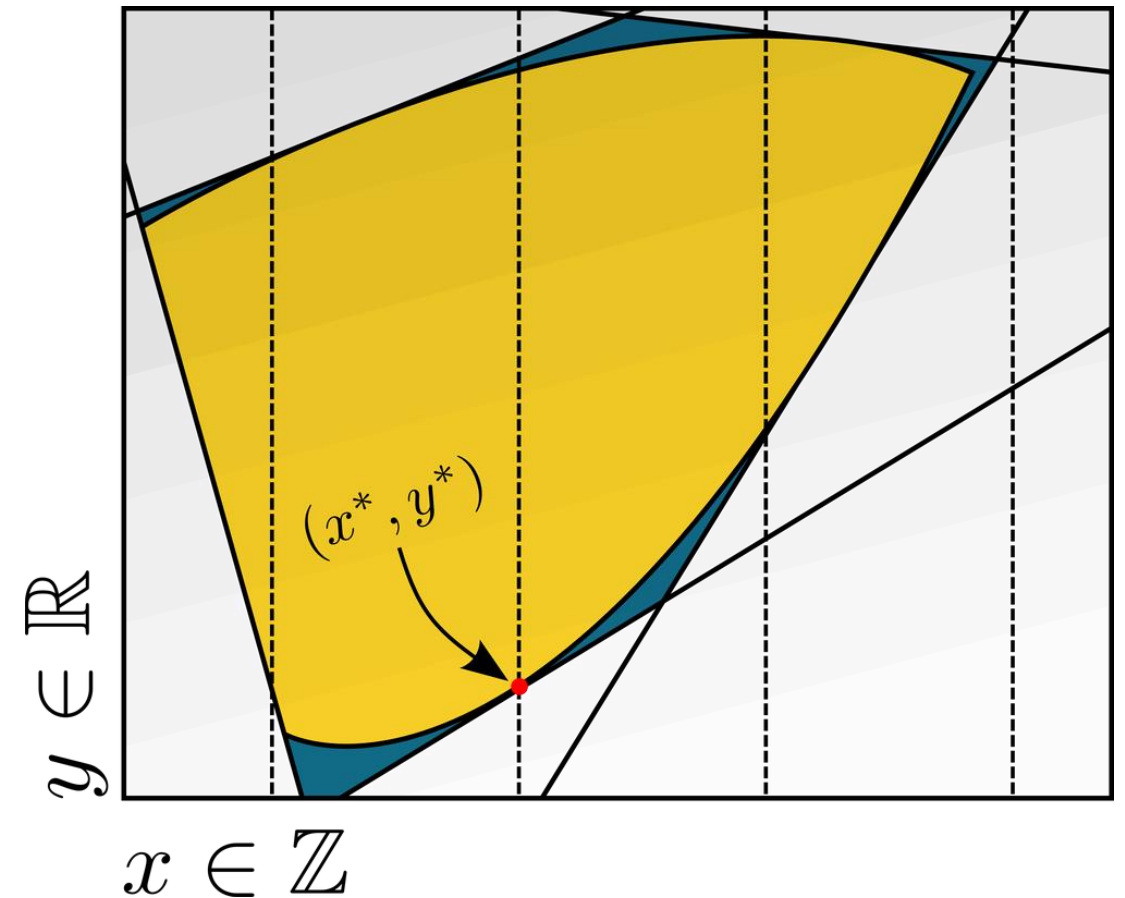
# Clasificación

Programación no lineal entera mixta (MINLP)

$$\begin{aligned} \min \quad & f(x) + d^T y \\ \text{s.a.}, \quad & h(x) + By = 0 \\ & g(x) + Dy \leq 0 \\ & x \in \mathbb{R}^n \\ & y \in \{0,1\}^m \end{aligned}$$

**Resolución:** Outer Approximation

**Ejemplo:** Síntesis diagramas flujo



# Clasificación

SOLVER	LP	MILP	NLP	MINLP
ALPHAECP				X
ANTIGONE			X	X
BARON	X	X	X	X
CONOPT4	X		X	
CPLEX	X	X		
DICOPT				X
GUROBI	X	X	X	X
GLPK	X	X		
IPOPT	X		X	
SCIP		X	X	X
CBC		X		X



# Fundamentos de Pyomo

Ejemplo 1: problema teórico

# Workflow

$$\begin{array}{ll}\min_x & x_1^2 + 2x_2^2 - x_3 \\ \text{s.t.} & x_1 + x_2 = 1 \\ & x_1 + 2x_2 - x_3 = 5 \\ & -10 \leq x_1, x_2, x_3 \leq 10\end{array}$$



# Workflow

Importar Pyomo y crear el modelo

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

```
from pyomo.environ import *  
model = ConcreteModel()
```

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

# Workflow

## Definir variables y parámetros

```
m.C = Set( initialize = ['A','B','W'] , doc = "Componentes" )
```

- `initialize`: iterable (list) con los elementos que forman el set
- `doc`: descripción (opcional)

```
m.X = Var(m.C, within=pyo.NonNegativeReals, doc = "Volumen m3")
```

- 1r argumento: Sets de los que depende la variable
- `within / domain`: dominio de la variable (NonNegativeReals, Reals, Integer...)

```
m.P = Param(m.C, initialize = {'A': 0.32, 'B': 0.25, 'C': 0.05},  
            mutable = True, doc = "Coste €/kg")
```

- 1r argumento: Sets de los que depende la variable
- `initialize`: *mapping* (python dict) con los valores correspondientes
- `mutable`: True si el valor del parámetro se puede modificar después de definirse

# Workflow

## Definir función objetivo y restricciones

```
m.obj = Objective( expr= sum(m.x[i]*m.P[i] for i in m.C),sense=maximize)
```

- `expr`: función que devuelve un único valor
- `sense` : minimize/maximize

```
m.con_demand = pyo.Constraint(expr = vol == sum(m.x[c] for c in m.C))
```

```
def num_warehouses_rule(m):  
    return sum(m.y[w] for w in m.W) <= m.P  
model.num_warehouses = pyo.Constraint(rule=num_warehouses_rule)
```

- 1r argumento : sets de los que depende la restricción
- `expr / rule`: función que devuelve `True/False` si se cumple la restricción o no

# Workflow

## Resolver

```
opt = pyo.SolverFactory('glpk')
results= opt.solve(m)

# Todo el modelo
m.pprint()

# Valor de la función objetivo
m.obj()

# Valor que toman las variables
m.x.pprint()

# Status del solver
print(results.solver.status)
print(results.solver.termination_condition)
```



$$\begin{array}{ll}\min_x & x_1^2 + 2x_2^2 - x_3 \\ \text{s.t.} & x_1 + x_2 = 1 \\ & x_1 + 2x_2 - x_3 = 5 \\ & -10 \leq x_1, x_2, x_3 \leq 10\end{array}$$

Ejemplo 1: problema teórico



$$\min \sum_{c \in C} x_c P_c$$

$$s.a., \sum_{c \in C} x_c = V$$

$$A^* = \frac{\sum_{c \in C} x_c A_c}{\sum_{c \in C} x_c}$$

Ejemplo 2: Mezcla lineal

# Tipos de modelos

## ConcreteModel

### 1 paso

Se especifican los datos al mismo tiempo que se construye el modelo (parámetros, sets, restricciones...)

**Sencillez**

**Similar a GAMS**

## AbstractModel

### 2 pasos

PASO 1: Crear un “patrón” abstracto

PASO 2: se cargan los datos conocidos para crear una instancia concreta

**Permite reutilizar modelos**

**Similar a AMPL**



$$\begin{aligned} \min \quad & \sum_{w \in W} \sum_{c \in C} d_{wc} x_{wc} \\ \text{s.t.}, \quad & \sum_{w \in W} x_{wc} = 1 \quad \forall c \in C \\ & x_{wc} \leq y_w \quad \forall w \in W, c \in C \\ & \sum_{w \in W} y_w = P \\ & 0 \leq x_{wc} \leq 1 \quad \forall w \in W, c \in C \\ & y_w \in \{0, 1\} \quad \forall w \in W \end{aligned}$$

Ejemplo 3: Ubicación de almacenes



# Programación disyuntiva

Reformulación  $M$  grande y envolvente convexa

# Programación disyuntiva generalizada (GDP)

$$\min \quad z = f(x)$$

$$s.a., \quad h(x) = 0$$

$$g(x) \leq 0$$

$$\bigvee_{i \in D_k} \left[ \begin{array}{c} Y_{k,j} \\ r_{k,j}(x) \leq 0 \end{array} \right] \quad k \in K$$

$$\bigvee_{i \in D_k} Y_{k,j} \quad k \in K$$

$$\Omega(Y) = True$$

$$x^L \leq x \leq x^U$$

$$x \in \mathbb{R}^n$$

$$Y \in \{True, False\} \quad k \in K, i \in D_k$$

**Función objetivo y restricciones globales**

**Disyunciones**

Restricciones  $r(x)$  solo se aplican si  $Y_{k,j} = True$

**Proposiciones lógicas**

# Programación disyuntiva generalizada (GDP)

## GDP

$$\begin{aligned} \min : & z = f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

$$\bigvee_{i \in D_k} \left[ \begin{array}{c} Y_{k,i} \\ r_{i,k}(x) \leq 0 \end{array} \right] \quad k \in K$$

$$\bigvee_{i \in D_k} Y_{k,i} \quad k \in K$$

$$\Omega(Y) = \text{True}$$

$$x^{lo} \leq x \leq x^{up}, \quad x \in \mathbb{R}^n$$

$$Y \in \{\text{True}, \text{False}\} \quad k \in K, i \in D_k$$

## Big M

$$\begin{aligned} \min : & z = f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

$$r_{ki}(x) \leq M^{ki} (1 - y_{ki}) \quad k \in K, i \in D_k$$

$$\sum_{i \in D_k} y_{ki} = 1 \quad k \in K$$

$$Hx \geq h$$

$$x^{lo} \leq x \leq x^{up}, \quad x \in \mathbb{R}^n$$

$$y_{ki} \in \{0, 1\} \quad k \in K, i \in D_k$$

## Envolvente convexa

$$\begin{aligned} \min : & z = f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

$$x = \sum_{i \in D_k} v^{ki} \quad k \in K$$

$$y_{ki} r_{ki} \left( v^{ki} / y_{ki} \right) \leq 0 \quad k \in K, i \in D_k$$

$$x^{lo} y_{ki} \leq v^{ki} \leq x^{up} y_{ki} \quad k \in K, i \in D_k$$

$$\sum_{i \in D_k} y_{ki} = 1 \quad k \in K$$

$$Hx \geq h$$

$$x \in \mathbb{R}^n$$

$$y_{ki} \in \{0, 1\} \quad k \in K, i \in D_k$$

# Pyomo GDP

## Disyunciones

$$\left[ \begin{array}{c} Y_1 \\ \exp(x_2) - 1 = x_1 \\ x_3 = x_4 = 0 \end{array} \right] \vee \left[ \begin{array}{c} Y_2 \\ \exp(2x_4) - 1 = x_3 \\ x_1 = x_2 = 0 \end{array} \right]$$

`m.term1 = pyo.Disjunct()` ← Crear un elemento `Disjunct()` para cada bloque

`m.term1.cons1 = pyo.Constraint(expr=exp(m.x[2]) - 1 == m.x[1])`  
`m.term1.cons2 = pyo.Constraint(expr= m.x[3] == 0)`  
`m.term1.cons3 = pyo.Constraint(expr= m.x[4] == 0)` ← Añadir restricciones

`m.term2 = pyo.Disjunct()`  
`m.term2.cons1 = pyo.Constraint(expr=exp(2*m.x[4]) - 1 == m.x[3])`  
`m.term2.cons2 = pyo.Constraint(expr= m.x[2] == 0)`  
`m.term2.cons3 = pyo.Constraint(expr= m.x[1] == 0)`

`m.logicOR = pyo.Disjunction(expr = [m.term1, m.term2])` ← Combinar cada término en forma de list

# Pyomo GDP

## Transformaciones

```
pyo.TransformationFactory(trf_name).apply_to(model)
opt = pyo.SolverFactory('cbc')
results = opt.solve(model)
```



Por defecto, Pyomo aplica una disyunción exclusiva  $\vee$

## Transformaciones disponibles

Reformulación	trf_name
Big M (BM)	gdp.bigm
Envolvente convexa (HR)	gdp.hull
Híbrido BM/HR	gdp.cuttingplane



$$\begin{aligned} \min \quad & \sum_{c \in C} x_c P_c \\ \text{s.t.}, \quad & \sum_{c \in C} x_c = 1 \\ & \sum_{c \in C} x_c \text{Vit}A_c \leq 0.4 \\ & \sum_{c \in C} x_c \text{Vit}B_c \geq 0.2 \\ & \left[ \begin{array}{c} Y \\ x_B = 0 \end{array} \right] \vee \left[ \begin{array}{c} \neg Y \\ x_A = 0 \end{array} \right] \\ & 0 \leq x_c \leq 1 \quad \forall c \in C \end{aligned}$$

Ejemplo 4: Formulación de mezclas



# Recursos

- ✓ Pyomo documentation  
<https://pyomo.readthedocs.io/en/stable/>
- ✓ ND Pyomo Cookbook  
<https://jckantor.github.io/ND-Pyomo-Cookbook/README.html>
- ✓ Materiales Cacheme  
<https://github.com/CASheme>
- ✓ Foros
  - StackOverflow (“pyomo” tag)
  - Pyomo Forum (pyomoforum@googlegroups.com)



# **Introducción a la programación matemática**

Isabela Fons

[isabela.fons@ua.es](mailto:isabela.fons@ua.es)

Grupo COnCEPT - Departamento de Ingeniería Química

UA

UNIVERSITAT D'ALACANT  
UNIVERSIDAD DE ALICANTE

Escola Politècnica Superior  
Escuela Politécnica Superior