El uso de Python en la ingeniería química

Computer-**a**ided **Chem**ical **E**ngineering

24/11/2013

CAChemE
cacheme.org

Esquema

- Presentación
- ¿Por qué Python?
- Resolución de ecuaciones en derivadas parciales
- Resolución de problemas de optimización
- Diseño de reactores en la industria de procesos químicos

CAChemE
cacheme.org

Presentación

- Asociación formada por ingenieros químicos (profesionales, docentes y estudiantes) que pretende estimular las posibilidades de software en la ingeniería de procesos.
- Promueve las ventajas de las nuevas herramientas de software libre disponibles y fomenta su uso en la universidad e industria.
- Especialización en simulación y programación matemática (optimización).

CAChemE
cacheme.org

¿Quiénes somos?

CAChemE
cacheme.org

Actividades

- Formación
- Webinars
- Desarrollo
- ¡Podcast!

CAChemE
cacheme.org

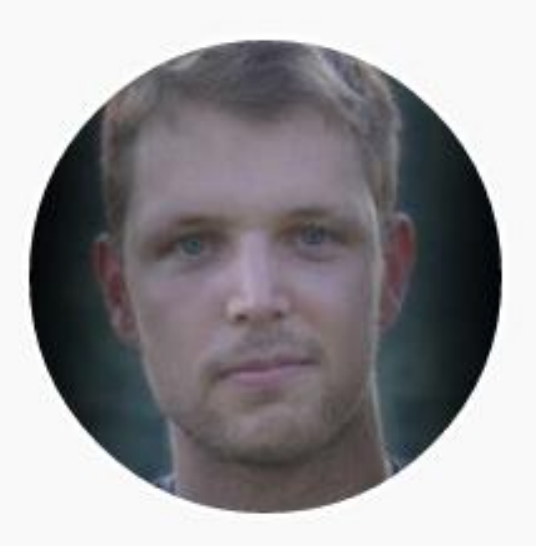
¿Por qué Python?

- Python es lenguaje de alto nivel conveniente para un desarrollo rápido de código
- Su filosofía de diseño enfatiza la simplicidad y legibilidad de código
- Posee núcleo de lenguaje relativamente pequeño con el apoyo de magníficas librerías (NumPy, SciPy, scikit-learn pandas, matplotlib etc.)
- Es lenguaje multiparadigma, en el que varios estilos de programación son compatibles (imperativo, orientado a objetos, funcional)
- Lenguaje de programación interpretado en lugar de compilado.
- Es multiplataforma (Windows, MacOS y Linux)
- Software libre

CAChemE
cacheme.org

"As a simulation engineer in 2013, you really are standing on the shoulders of giants"

Abhishek Chintagunta
CFD and Coffee (blog)



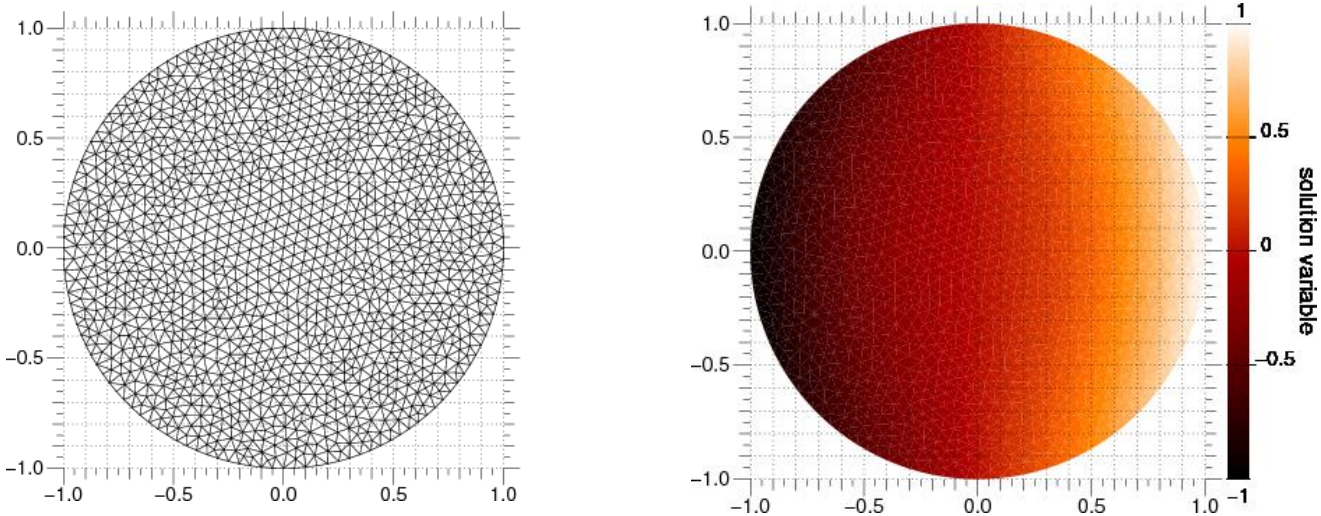
- Franz Navarro
- Ingeniero Químico
- Universidad de Alicante
- *francisco.navarro@cacheme.org*

1. Resolución de EDPs con Python

CAChemE
cacheme.org

1. Resolución de EDP

- Las ecuaciones en derivadas parciales (EDP) permiten modelar fenómenos físicos como la propagación del sonido o del calor, la electrostática, la electrodinámica, la dinámica de fluidos, etc.
- Existen numerosos paquetes de software para resolver EDP, usando una variedad de lenguajes y métodos numéricos.
  - Comerciales: COMSOL Multiphysics, ANSYS, Abaqus...
  - Free/Open source: OpenFOAM, FreeFem++, Elmer, FiPy…



Problema simple de difusión resuelto en Python (FiPy) con menos de 40 líneas de código

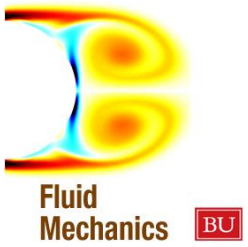
CFD with Python: 12 steps to Navier Stokes

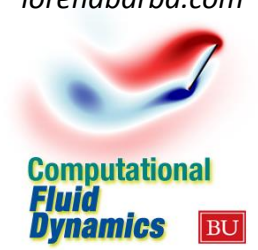
CAChemE cacheme.org

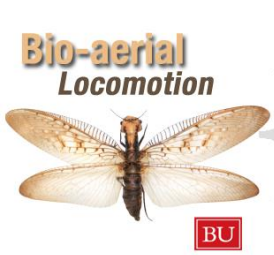
Lorena A. Barba

lorenabarba.com

- Modulo interactivo online de CFD con Python impartido por la profesora Lorena A. Barba (Boston University)
- Diseñado para principiantes en programación y en CFD
- Resolución numérica mediante el método de las diferencias finitas
- Videos explicativos de sus clases también online
- Material y código libre y gratuito







CFD with Python: 12 steps to Navier Stokes

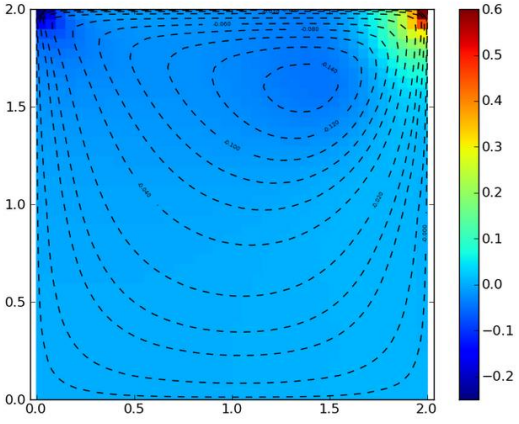
Lorena A. Barba

CAChemE cacheme.org

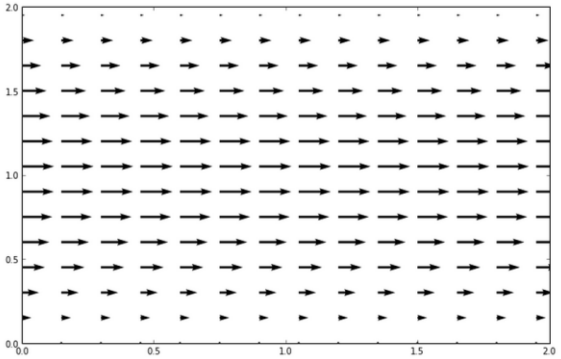
- Pasos 1-4 son en una dimensión:
  - Convección lineal
  - Convección no lineal
  - Difusión
  - Ecuación de Burgers
- Pasos 5-10 pasan a 2D:
  - Convección lineal
  - Convección no lineal
  - Difusión
  - Ecuación de Burgers
  - Ecuación de Laplace
  - Ecuación de Poisson

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{v}$$

(a)



(b)



- Pasos 11-12 resuelve la ecuación de Navier-Stokes en 2D:
  - Flujo en una cavidad (a)
  - Flujo en un canal (b)

+Introducción a Python, Numpy, SimPy, matplotlib y Numba

CFD with Python: 12 steps to Navier Stokes

Lorena A. Barba

- Paso 7: Difusión en 2D

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2} + \nu \frac{\partial^2 u}{\partial y^2}$$

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} = \nu \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} + \nu \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2}$$

Reorganizando la ecuación discretizada:

$$u_{i,j}^{n+1} = u_{i,j}^n + \frac{\nu \Delta t}{\Delta x^2}(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + \frac{\nu \Delta t}{\Delta y^2}(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n)$$

CFD with Python: 12 steps to Navier Stokes

Lorena A. Barba

CAChemE cacheme.org

- Paso 7: Difusión en 2D

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \nu \frac{u_{i+} \qquad \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{\Delta y^2}$$

Reorganizando la ecuación di

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{\nu \Delta t}{\Delta x^2}(u_{i} \qquad \frac{\Delta t}{\Delta y^2}(u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n})$$

KEEP CALM AND trust in Python

CFD with Python: 12 steps to Navier Stokes

Lorena A. Barba

- Paso 7: Difusión en 2D

- `numpy` es una biblioteca que proporciona un grupo de operaciones matriciales de forma similar a MATLAB
- `matplotlib` es una biblioteca de gráficas 2D que vamos a utilizar para representar los resultados

```
# Recuerda: Los comentarios en Python se indican con el signo de almohadilla

%pylab inline # Representa las gráficas en línea con el texto

import numpy as np # Importación de la librería NumPy, acceso a ella como np
import matplotlib.pyplot as plt #Importación de la librería gráfica
from mpl_toolkits.mplot3d import Axes3D ## Librería para proyecciones en 3D
from matplotlib import cm ##cm = "colormap" para cambiar la paleta de colores

### Declaración de variables
nx = 31 # número de puntos x en la malla
ny = 31 # número de puntos y en la malla
nt = 17 # número de intervalos de tiempo que se desea calcular
nu=.05
dx = 2.0/(nx-1)
dy = 2.0/(ny-1)
sigma = .25
dt = sigma*dx*dy/nu #dt es definido usando el valor sigma

x = np.linspace(0,2,nx)
y = np.linspace(0,2,ny)

u = np.ones((ny,nx)) ## crea un vector 1xn de unos
un = np.ones((ny,nx)) ##
```

Condición CFL para asegurar la convergencia, se trabaja con ella en el Paso 3.

$$\frac{u_x \cdot \Delta t}{\Delta x} + \frac{u_y \cdot \Delta t}{\Delta y} < C$$

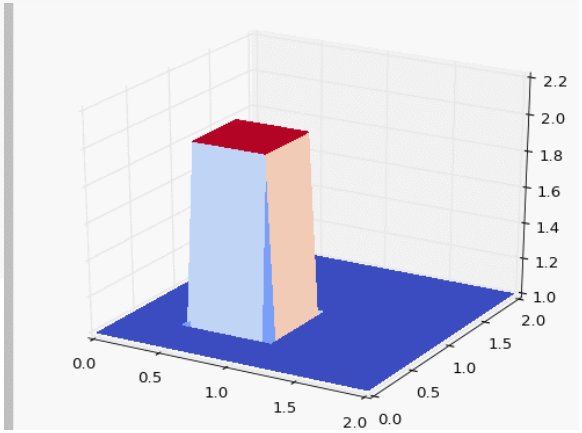
CFD with Python: 12 steps to Navier Stokes

Lorena A. Barba

CAChemE cacheme.org

- Paso 7: Difusión en 2D

```
### Asignación de condiciones inciales

## Se establece función de sobrero como C.I. : u(.5<=x<=1 && .5<=y<=1 ) es 2
u[.5/dy:1/dy+1,.5/dx:1/dx+1]=2

# Representación de datos
fig = plt.figure()
ax = fig.gca(projection='3d')
X,Y = np.meshgrid(x,y)
surf = ax.plot_surface(X,Y,u[:], rstride=1, cstride=1, cmap=cm.coolwarm,
        linewidth=0, antialiased=False)

plt.show()

ax.set_xlim(1,2)
ax.set_ylim(1,2)
ax.set_zlim(1,2.5)
```



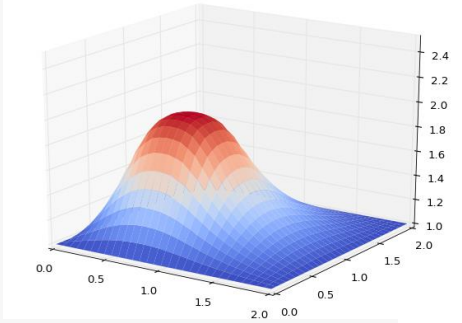
CFD with Python: 12 steps to Navier Stokes

Lorena A. Barba

CAChemE cacheme.org

$$u_{i,j}^{n+1} = u_{i,j}^n + \frac{\nu\Delta t}{\Delta x^2}\left(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n\right) + \frac{\nu\Delta t}{\Delta y^2}\left(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n\right)$$

```
### Función que ejecuta la solución hasta un tiempo nt
def diffuse(nt):
    u[.5/dy:1/dy+1,.5/dx:1/dx+1]=2

    for n in range(nt+1):
        un[:] = u[:]
        u[1:-1,1:-1]=un[1:-1,1:-1]+\
            nu*dt/dx**2*(un[2:,1:-1]-2*un[1:-1,1:-1]+un[0:-2,1:-1])+\
            nu*dt/dy**2*(un[1:-1,2:]-2*un[1:-1,1:-1]+un[1:-1,0:-2])

        u[0,:]=1
        u[-1,:]=1
        u[:,0]=1
        u[:,-1]=1


    fig = plt.figure()
    ax = fig.gca(projection='3d')
    surf = ax.plot_surface(X,Y,u[:], rstride=1, cstride=1, cmap=cm.coolwarm,
        linewidth=0, antialiased=True)
    ax.set_zlim(1,2.5)
    plt.show()
```



CAChemE
cacheme.org

FiPy

- FiPy resuelve EDPs mediante el método de los volúmenes finitos (FVM) con programación orientada a objetos y estando escrito en Python.
- Mayor grado de abstracción (integradas funciones de mallado de *Gmesh*)
- Framework maduro y bien documentado (permite Python 3.x)
- 100% gratuita y de dominio de público (open source)
- Utilizado en investigación y academia (especializado en ciencia de los materiales)
- Desarrollado en centros de investigación norteamiercanos (CTCMS y NIST).

ctcms.nist.gov/fipy/

FiPy

CAChemE cacheme.org

- Resolución de ecuaciones con la siguiente forma:

$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{\text{transitorio}} + \underbrace{\nabla\cdot(\vec{u}\phi)}_{\text{difusión}} = \underbrace{\left[\nabla\cdot(\Gamma_i\nabla)\right]^n\phi}_{\text{convección}} + \underbrace{S_\phi}_{\text{fuente (source)}}$$

Siendo ρ, $\vec{u}$, Γ_i los coeficientes de sus respectivos términos

- Permite acoplar fenómenos multifísicos de forma *sencilla*.

ctcms.nist.gov/fipy/

FiPy: Ejemplo

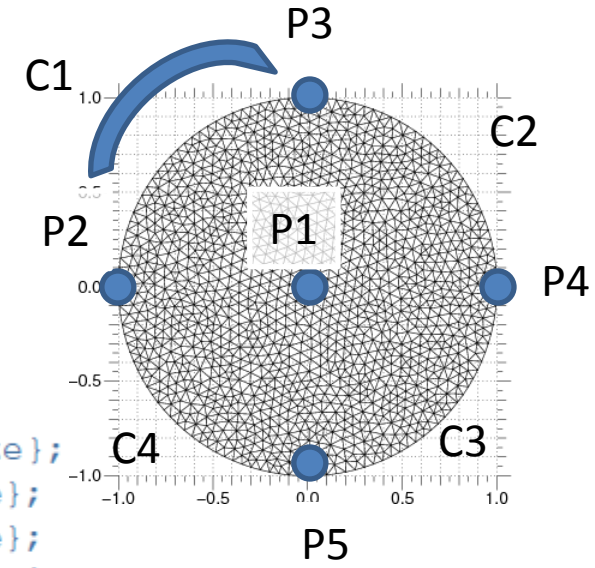
$$\frac{\partial(\rho\phi)}{\partial t} + \nabla\cdot(\vec{u}\phi) = [\nabla\cdot(\Gamma_i\nabla)]^n\,\phi + S_\phi$$

- Ejemplo de resolución de un problema de difusión en estado estacionario pero con una geometría circular y FiPy.

- Mallado

$$D\nabla^2\phi = 0$$

```
>>> cellSize = 0.05
>>> radius = 1.
>>> from fipy import *
>>> mesh = Gmsh2D('''
...                cellSize = %(cellSize)g;
...                radius = %(radius)g;
...                Point(1) = {0, 0, 0, cellSize};
...                Point(2) = {-radius, 0, 0, cellSize};
...                Point(3) = {0, radius, 0, cellSize};
...                Point(4) = {radius, 0, 0, cellSize};
...                Point(5) = {0, -radius, 0, cellSize};
...                Circle(6) = {2, 1, 3};
...                Circle(7) = {3, 1, 4};
...                Circle(8) = {4, 1, 5};
...                Circle(9) = {5, 1, 2};
...                Line Loop(10) = {6, 7, 8, 9};
...                Plane Surface(11) = {10};
...                ''' % locals())
```



CAChemE
cacheme.org

FiPy: Ejemplo

$$D\nabla^2\phi = 0$$

CAChemE cacheme.org

```
>>> phi = CellVariable(name = "solution variable",
...                    mesh = mesh,
...                    value = 0.)
```

Usando la malla construimos las *variables de las celdas*

```
>>> viewer = None
>>> if __name__ == '__main__':
...     try:
...         viewer = Viewer(vars=phi, datamin=-1, datamax=1.)
...         viewer.plotMesh()
...         raw_input("Irregular circular mesh. Press <return> to proceed...")
...     except:
...         print "Unable to create a viewer for an irregular mesh (try Gist2DViewer
```

Representar la malla

```
>>> D = 1.
```

Coeficiente de difusión

```
>>> eq = TransientTerm() == DiffusionTerm(coeff=D)
```

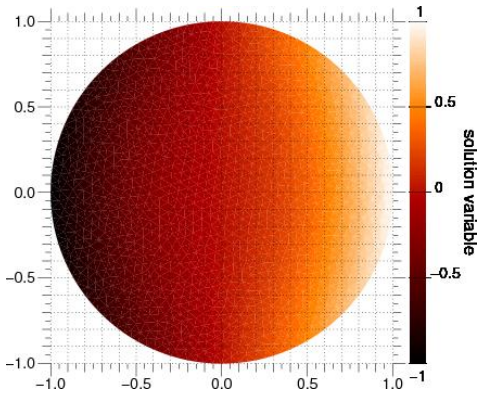
```
>>> X, Y = mesh.faceCenters
```

Condiciones de contorno

```
>>> phi.constrain(X, mesh.exteriorFaces)
```

```
>>> DiffusionTerm(coeff=D).solve(var=phi)
```

Resolución

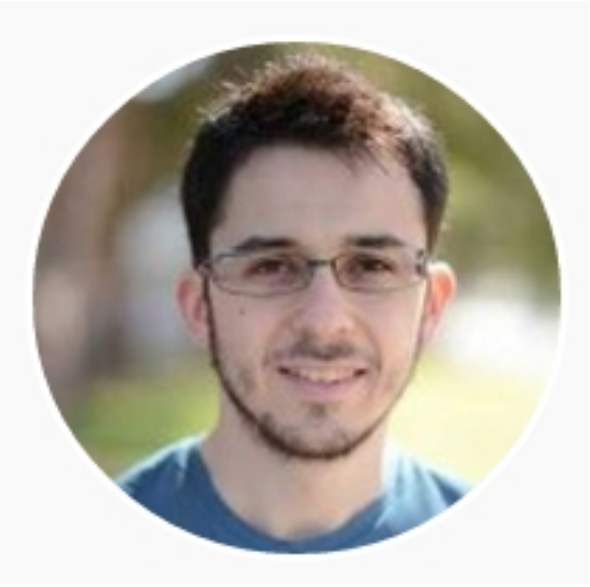



```
>>> if viewer is not None:
...     viewer.plot()
...     raw_input("Steady-state diffusion. Press <return> to proceed...")
```

Representación (ejecutado en un script)

CAChemE
cacheme.org

PyConES
2013



- Jorge Bernabé
- Ingeniero Químico
- Universidad de Alicante
  jorge.bernabe@cacheme.org

2. Programación matemática (optimización)

CAChemE
cacheme.org

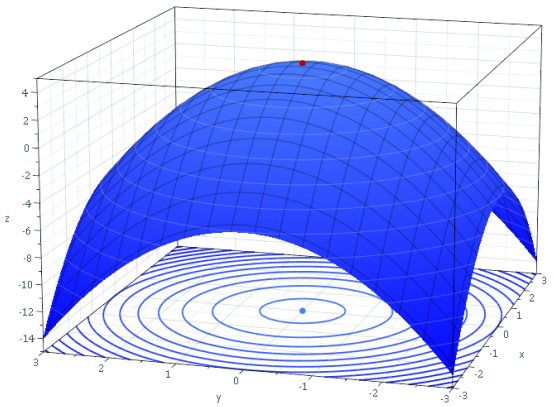
Programación matemática

Optimización: ¿por qué es importante?

- Mejorar calidad de un producto
- Aumentar beneficios
- Reducir riesgos ambientales
- Reducir costes de producción

Interés empresarial

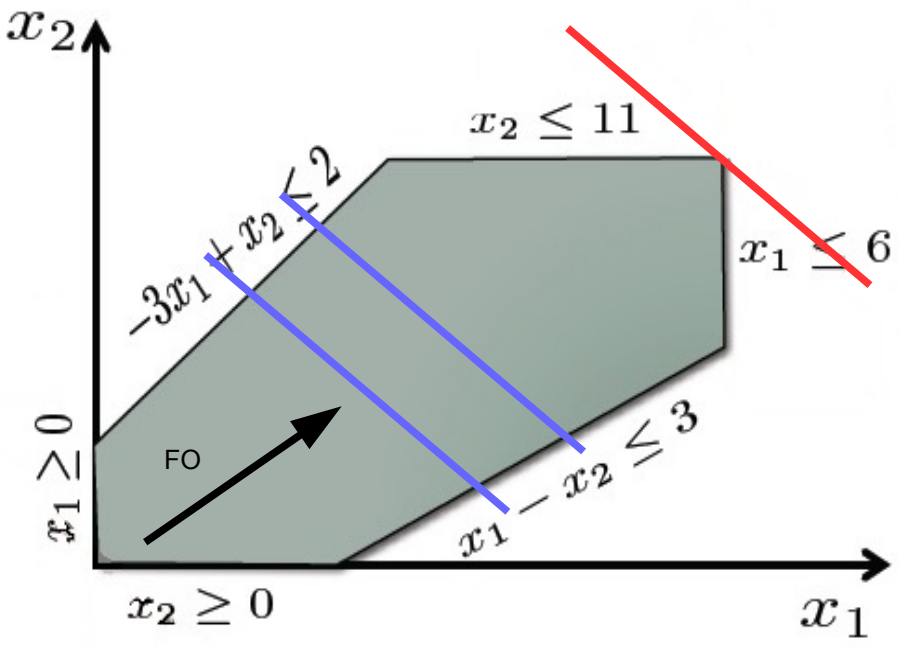
IQ: - selección de equipos y recursos
- gestión logística



CAChemE
cacheme.org

Optimización con Python en IQ

- Optimización matemática
  - min f(x)
  - s.a g(x) = 0
    h(x) ≤ 0
- LP, NLP, MILP, MINLP

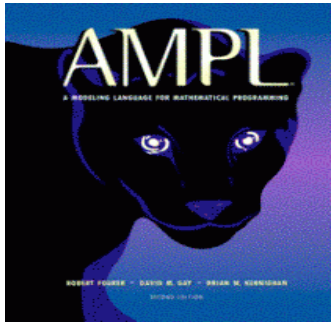


CAChemE
cacheme.org

AML (Algebraic modelling languages)

Software propietario:

- AMPL (www.ampl.com) – Lenguaje sencillo, pero complicado interactuar
  - GLPK – Alternativa libre a AMPL para LP y MILP



- GAMS (www.gams.com) – Se comunica con solvers incluso para resolver MINP

GAMS

- AIMMS (www.aiims.com) – Diseñado para resolver problemas de optimización a gran escala y programación de actividades.

AIMMS
THE MODELING SYSTEM

CAChemE
cacheme.org

Optimización en Python

- Free/Open source:
  - CVXOPT – M.Andersen, J.Dahl, L.Vandenberghe
    Notación matricial. Optimización convexa
  - PuLP – Trabaja con lenguaje Python. Muy buenos resultados para LP o MIP. No resuelve NLP
  - OpenOpt – Más de 30 solvers para solucionar el modelado, incluyendo NLP
  - Pyomo – Coopr. (Sandia National Laboratories, USA)
    Permite la formulación de modelos algebraicos en el lenguaje de programación en Python.

COmmon Optimization Python Repository

CAChemE
cacheme.org

¿Por qué Pyomo?

- Se comunica directamente los principales solvers de AMPL, GLPK, Gurobi, CPLEX, CBC y PICO.
- Programación en Python tipo AMPL/GAMS
- Open source (COIN-OR)
- Pyomo (Coopr) es un paquete gratuito con licencia BSD, maduro y bien documentado (en migración a Python 3.x)
- Fácil instalación “pip install coopr”
- Adaptabilidad a modelado de problemas de IQ

- open source for the operations research community -

coin-or.org

CAChemE
cacheme.org

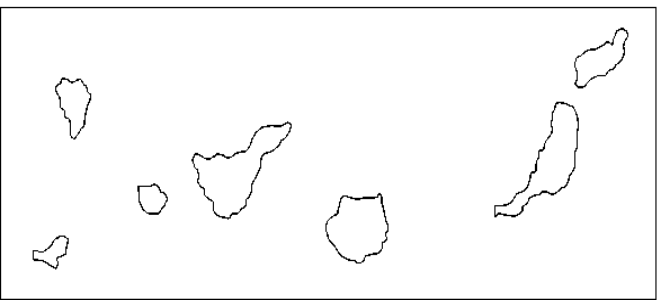
Problema I: Logística empresarial

- Problema clásico en optimización
- LP → Se puede resolver con cualquier herramienta
- Fundamental en el mundo empresarial
- Minimización de coste en transporte

PLANTAS Y MERCADOS DEL SEÑOR EMPRESARIO

·Plantas -

·Mercados -



CAChemE
cacheme.org

Problema I: Logística empresarial

- Plantas ⟶ producción máxima
- Mercados ⟶ demanda mínima

- Coste de transporte por unidad:

	Alicante	Castellón	Vitoria	Cádiz	Cáceres	PRODUCCIÓN
Madrid	4.17	4.10	3.48	6.37	2.96	250
Barcelona	5.16	2.69	5.35	11.14	9.10	200
Almería	2.95	5.42	9.10	3.80	6.47	300
La Coruña	10.26	10.19	6.56	10.65	6.85	300
DEMANDA	150	150	100	150	175	-

CAChemE
cacheme.org

Problema I: Logística empresarial

IP[y]: Notebook ProblTransporte (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

In [ ]:
```
from coopr.pyomo import *

model = AbstractModel()
```

- Sentencia para importar pyomo
- Sentencia para definir el modelo, abstracto o concreto

CAChemE
cacheme.org

Problema I: Logística empresarial

```
model.plantas = Set()
model.mercados = Set()

model.produccion = Param(model.plantas)
model.demanda = Param(model.mercados)
model.costes = Param(model.plantas, model.mercados)

model.unidades = Var(model.plantas, model.mercados, within = NonNegativeReals)
```

- Definición de sets – series de datos
- Definición de parámetros – valores de los sets
- Definición de variables

} Sus valores en archivo .dat

CAChemE
cacheme.org

Problema I: Logística empresarial

```
def CosteTotal(model):
    return sum(
        model.costes[n,i] * model.unidades[n,i]
        for n in model.plantas
        for i in model.mercados
    )

model.costefinal=Objective(rule=CosteTotal)
```

- Definición de ecuaciones: def Nombre_Ecuación (modelo y variables de las que depende)
- return: se escribe la ecuación
- Modelo.nombre: va a mostrar el valor de la ecuación al ejecutar el programa

CAChemE
cacheme.org

Problema I: Logística empresarial

```
def MinDemanda(model, mercado):
    return sum(model.unidades[i, mercado] for i in model.plantas) >= model.demanda[mercado]

model.DemandaConstraint = Constraint(model.mercados, rule=MinDemanda)

def MaxProduccion(model, planta):
    return sum(model.unidades[planta, j] for j in model.mercados) <= model.produccion[planta]

model.ProdConstraint = Constraint(model.plantas, rule=MaxProduccion)
```

- Definición de restricciones: Constraint (indica que la palabra mercado se refiere a model.mercados)

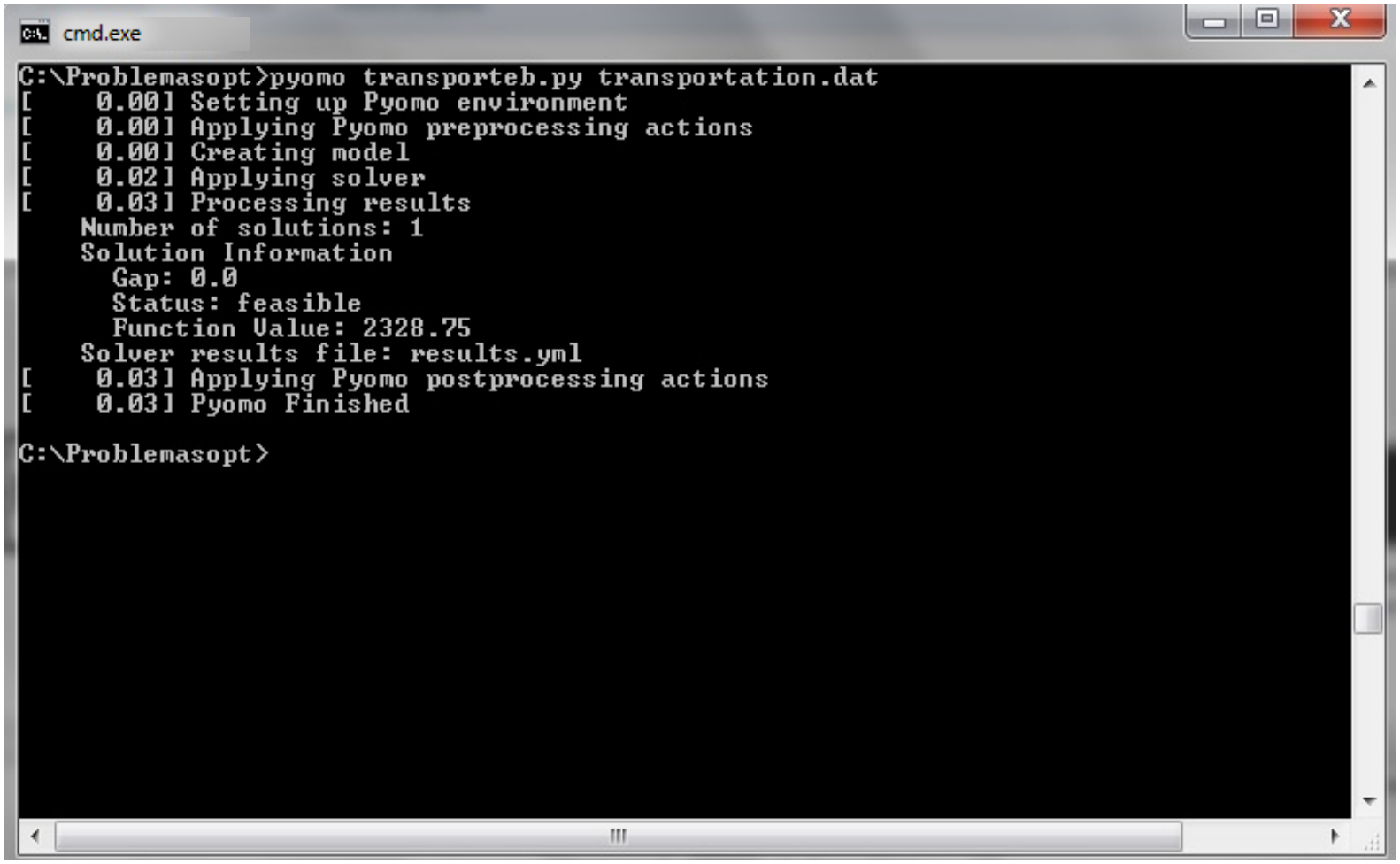
CAChemE
cacheme.org

Problema I: Logística empresarial

```
set plantas := Madrid Barcelona Almeria Coruna;
set mercados := Alicante Castellon Vitoria Cadiz Caceres;

param: produccion :=
Madrid    250
Barcelona 200
Almeria   300
Coruna    300;

param: demanda :=
Alicante  150
Castellon 150
Vitoria   100
Cadiz     150
Caceres   175;

param costes:
           Alicante Castellon Vitoria Cadiz  Caceres :=
Madrid     4.17     4.10      3.48    6.37   2.96
Barcelona  5.16     2.69      5.35    11.14  9.10
Almeria    2.95     5.42      9.10    3.80   6.47
Coruna     10.26    10.19     6.56    10.65  6.85  ;
```

CAChemE
cacheme.org

Problema I: Logística empresarial



pyomo archivo.py datos.dat

CAChemE
cacheme.org

Problema I: Logística empresarial

```
#   Solution Information
# ---------------------------------
Solution:
- number of solutions: 1
  number of solutions displayed: 1
- Gap: 0.0
  Status: feasible
  Objective:
    costefinal:
      Id: 0
      Value: 2328.75
  Variable:
    unidades[Barcelona,Vitoria]:
      Id: 0
      Value: 25
    unidades[Barcelona,Castellon]:
      Id: 8
      Value: 150
    unidades[Almeria,Cadiz]:
      Id: 10
      Value: 150
    unidades[Madrid,Vitoria]:
      Id: 11
      Value: 75
    unidades[Madrid,Caceres]:
      Id: 18
      Value: 175
    unidades[Almeria,Alicante]:
      Id: 19
      Value: 150
```

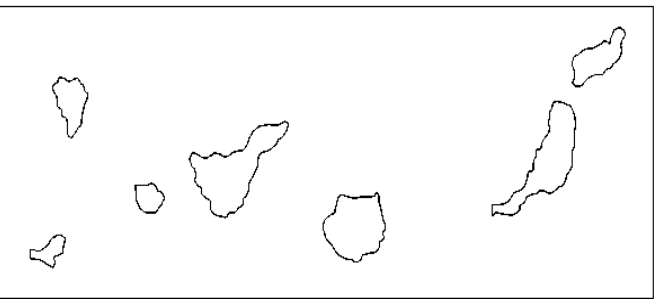
PLANTAS Y MERCADOS DEL SEÑOR EMPRESARIO

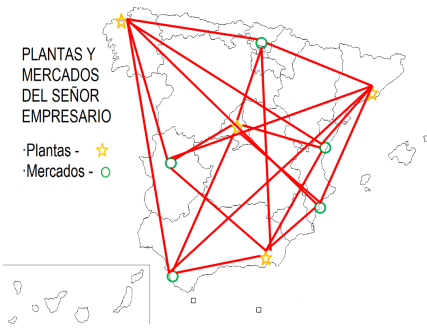
·Plantas -

·Mercados -

Solución óptima

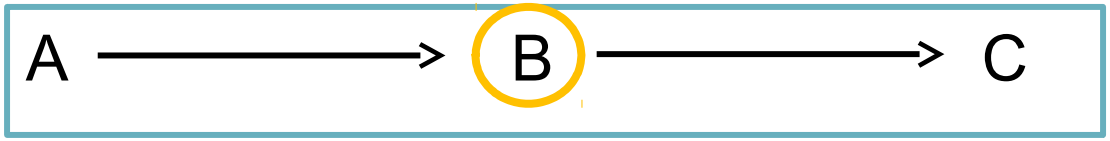
- Todas las posibilidades (superestructura)

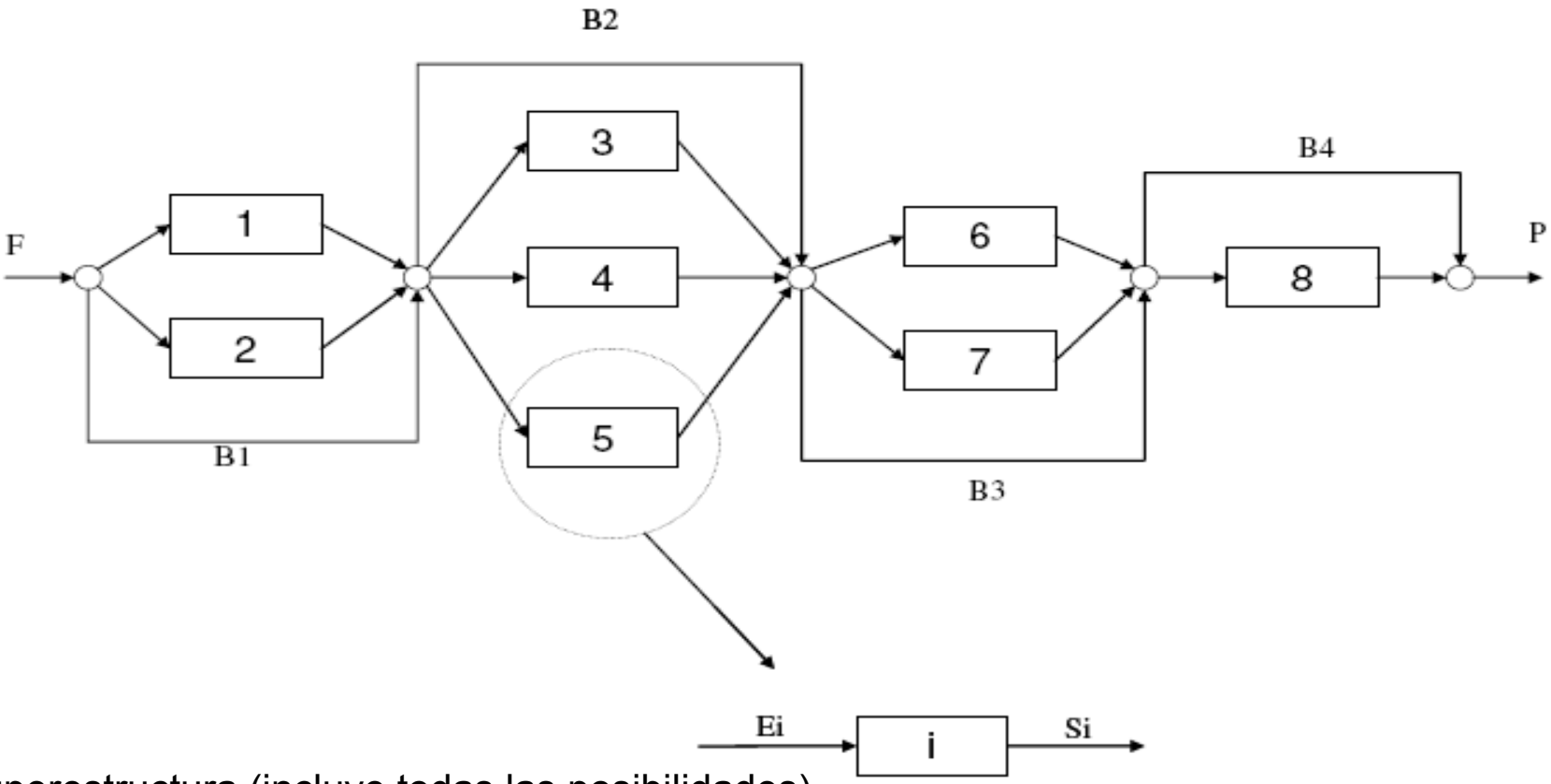




CAChemE
cacheme.org

Problema II: Selección de equipos





- Superestructura (incluye todas las posibilidades)

CAChemE
cacheme.org

Problema II: Selección de equipos

Aumento beneficio	Disminución beneficio
B	A
	C
	Coste uso reactor

- Objetivo: Seleccionar la óptima distribución de reactores y bypass que aumentan el beneficio

Problema II: Selección de equipos

$$A_S = A_E \, exp(-\tau)$$

$$B_S = \left(A_E \tau + B_E \right) exp(-\tau)$$

$$Coste = \begin{cases} Cf_1 + CV_1 A_E & Si \quad 0 \le A_E \le 7 \\ Cf_2 + CV_2 A_E & Si \quad 7 \le A_E \le 10 \end{cases}$$

	τ (h)	C_{f1} (um/h)	C_{f2} (um/h)	CV_1 (um/kmol)	CV_2 (um/kmol)
Reactor 1	0.1	10	5	1	8
Reactor 2	0.4	20	10	2	10
Reactor 3	0.4	50	25	1	9
Reactor 4	0.2	20	10	1	50
Reactor 5	0.7	60	30	2	70
Reactor 6	0.2	10	20	1	10
Reactor 7	0.9	50	25	3	15
Reactor 8	0.5	100	50	5	10

Problema II: Selección de equipos

- Toma de decisiones – MILP
- Resolución de la parte disyuntiva del problema mediante la reformulación de la envolvente convexa
- Disgregación de variables

$$\begin{bmatrix} \begin{bmatrix} W_{i,c1} \\ C_i = Cf_i + CV_i \cdot E_{i,A} \\ 0 \le E_{i,A} \le 7 \end{bmatrix} \underline{\vee} \begin{bmatrix} W_{i,c2} \\ C_i = Cf_i + CV_i \cdot E_{i,A} \\ 7 \le E_{i,A} \le 10 \end{bmatrix} \end{bmatrix}_{Y_{Ri}} \underline{\vee} \begin{bmatrix} \neg Y_{Ri} \\ C_i = 0 \\ S_{i,j} = 0 \end{bmatrix}$$

$$y_i = w_{i,c1} + w_{i,c2}$$

CAChemE
cacheme.org

Problema II: Selección de equipos

- Disyunciones para el bypass

$$\begin{bmatrix} by_b \\ BY_{b,j} \leq 10 \end{bmatrix} \veebar \begin{bmatrix} \neg\, by_b \\ BY_{b,j} = 0 \end{bmatrix}$$

- Balances de materia

$$F_j = E_{1,j} + E_{2,j} + BY_{1,j}$$

$$y_1 + y_2 + by_1 = 1$$

CAChemE
cacheme.org

Problema II: Selección de equipos

```
#DEFINICIÓN DEL MODELO
model = AbstractModel()

#DEFINICIÓN DE LOS SETS DEL PROBLEMA
model.reactores = Set()
model.bypass = Set()
model.elementos = Set()
model.zonascoste = Set()
```

```
model.tau = Param(model.reactores)
model.coste_inf = Param(model.zonascoste)
model.coste_sup = Param(model.zonascoste)
model.entrada = Param(model.elementos)

model.coste_fijo = Param(model.reactores, model.zonascoste)
model.coste_var = Param(model.reactores, model.zonascoste)
```

CAChemE
cacheme.org

Problema II: Selección de equipos

```
model.P = Var(model.elementos, within = NonNegativeReals)
model.E = Var(model.elementos, model.reactores, within = NonNegativeReals)
model.S = Var(model.elementos, model.reactores, within = NonNegativeReals)
model.coste = Var(model.reactores, within = NonNegativeReals)
model.Edis = Var(model.reactores, model.zonascoste, within = NonNegativeReals)
model.coste_dis = Var(model.reactores, model.zonascoste, within = NonNegativeReals)
model.byp_dis = Var(model.elementos, model.bypass, within = NonNegativeReals)



model.bina_reactor = Var(model.reactores, within = Binary)
model.bina_bypass = Var(model.bypass, within = Binary)
model.bina_coste = Var(model.reactores, model.zonascoste, within = Binary)
```

- within = NonNegativeReals – Valores reales no negativos
- within = Binary – Valores binarios

Problema II: Selección de equipos

```
#Función objetivo


def FuncionObjetivo(model):
    return 100*model.P['B'] - 10*model.P['A'] - 20*model.P['C']
      - sum(model.coste[o] for o in model.reactores)
model.fo = Objective(rule = FuncionObjetivo, sense = maximize)
```

- Por defecto, pyomo, minimiza la FO. Para maximizar, se escribe sense = maximize

CAChemE
cacheme.org

Problema II: Selección de equipos

```
#Balances de Materia en cada nodo

def BM1(model, i):
    return  sum(model.E[i,'r1'] + model.E[i,'r2'] + model.byp_dis[i,'by1']
                for i in model.elementos) == sum(model.entrada[i]
                                                 for i in model.elementos)
model.balmat1 = Constraint(model.elementos, rule = BM1)
```

```
#Disyunciones para elección de bypass o reactor

def Disy1(model):
    return model.bina_reactor['r1'] + model.bina_reactor['r2'] + model.bina_bypass['by1'] == 1
model.dis1 = Constraint(rule = Disy1)
```

```
#Ecuación para la disyunción interior de cada disgregación referida al coste

def Disy5(model, reactor):
    return sum(model.bina_coste[reactor,k] for k in model.zonascoste)  ==
    model.bina_reactor[reactor]
model.dis5 = Constraint(model.reactores, rule = Disy5)
```

CAChemE
cacheme.org

Problema II: Selección de equipos

```
#Ecuaciones para las disgregaciones

def Disgr1(model,b):
    return model.E['A',b] == sum(model.Edis[b,c] for c in model.zonascoste)
model.disgrega1 = Constraint(model.reactores, rule = Disgr1)

def Disgr2(model,b):
    return model.coste[b] == sum(model.coste_dis[b,c] for c in model.zonascoste)
model.disgrega2 = Constraint(model.reactores, rule = Disgr2)
```

```
#Ecuación para indicar el cálculo del coste
def EcuacionCoste(model, q, w):
    return model.coste_dis[q ,w] == model.coste_fijo[q,w] * model.bina_coste[q,w]
    + model.coste_var[q,w] * model.Edis[q,w]
model.cost = Constraint(model.reactores, model.zonascoste, rule = EcuacionCoste)
```

CAChemE
cacheme.org

Problema II: Selección de equipos

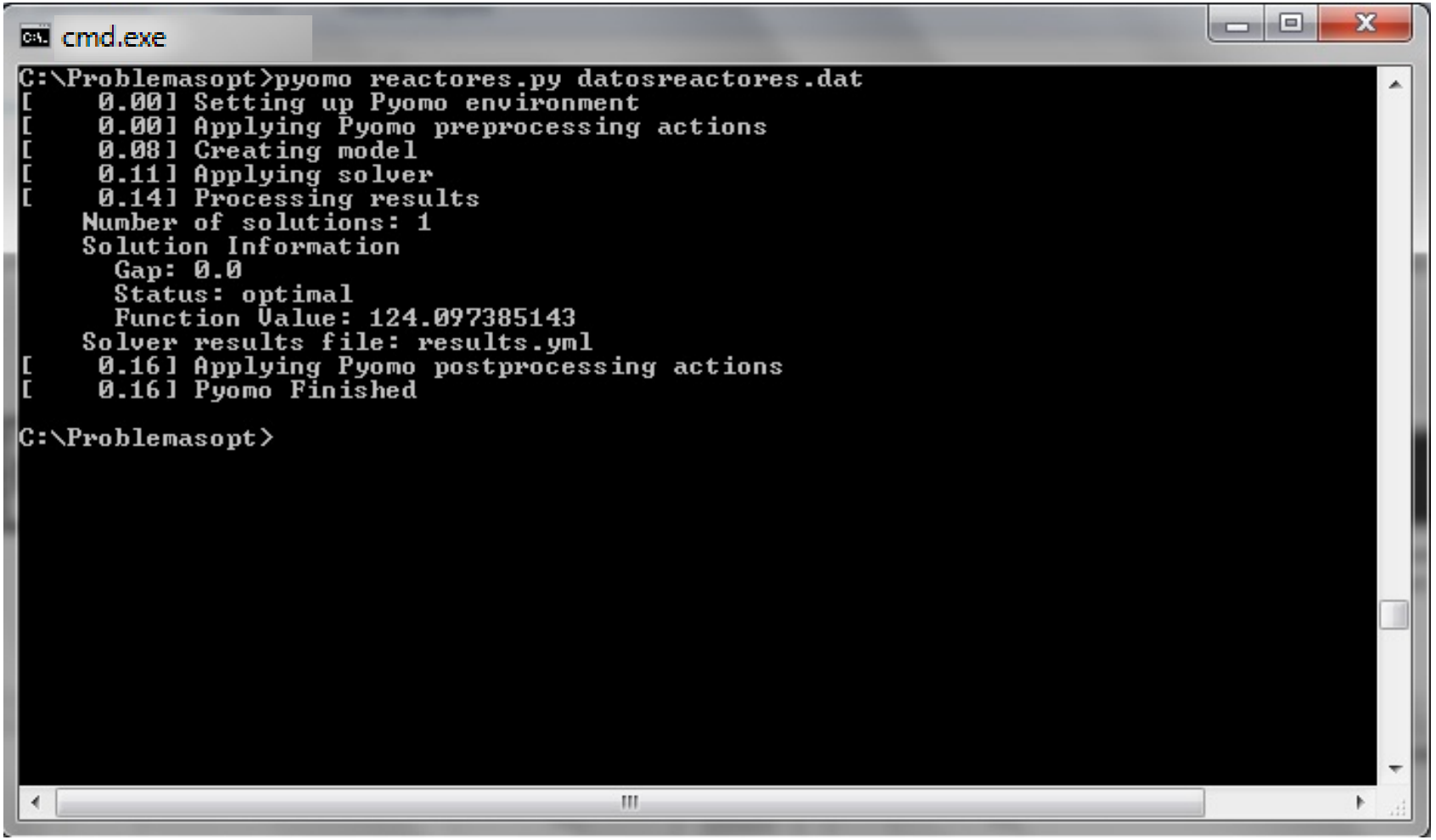
In [ ]:

```
set reactores := r1 r2 r3 r4 r5 r6 r7 r8 ;
set bypass := by1 by2 by3 by4 ;
set elementos := A B C ;
set zonascoste := c1 c2;


param: tau :=
r1 0.1
r2 0.4
r3 0.1
r4 0.2
r5 0.7
r6 0.2
r7 0.9
r8 0.5 ;

param: coste_inf :=
c1 0
c2 7 ;

param: coste_sup :=
c1 7
c2 10 ;

param: entrada :=
A 10
B 0
c 0 ;
```

```
param: coste_fijo :=
     c1    c2
r1   10    5
r2   20    10
r3   50    25
r4   20    10
r5   60    30
r6   10    20
r7   50    25
r8   100   50 ;

param: coste_var :=
     c1    c2
r1   1     8
r2   2     10
r3   1     9
r4   1     50
r5   2     70
r6   1     10
r7   3     15
r8   5     10 ;
```

CAChemE
cacheme.org

Problema II: Selección de equipos



CAChemE
cacheme.org

Problema II: Selección de equipos

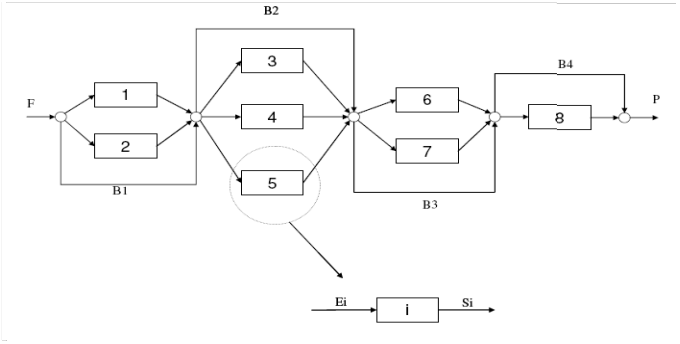
results.yml

```
# ------------------------------------
#   Solution Information
# ------------------------------------
Solution:
- number of solutions: 1
  number of solutions displayed: 1
- Gap: 0.0
  Status: optimal
  Objective:
    fo:
      Id: 0
      Value: 124.097385143
  Variable:
    P[A]:
      Id: 0
      Value: 4.49328964117
    P[C]:
      Id: 1
      Value: 1.91207864589
    P[B]:
      Id: 2
      Value: 3.59463171294
    coste[r4]:
      Id: 3
      Value: 26.7032004604
    coste[r6]:
      Id: 5
      Value: 15.4881163609
```

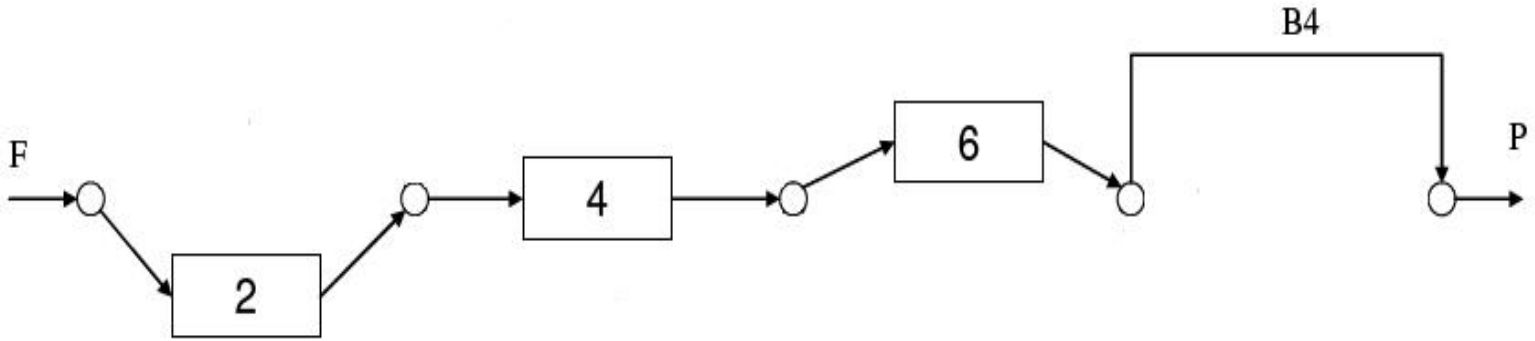
CAChemE
cacheme.org

Problema II: Selección de equipos

- Superestructura:



- Solución óptima:



CAChemE
cacheme.org

PyConES 2013

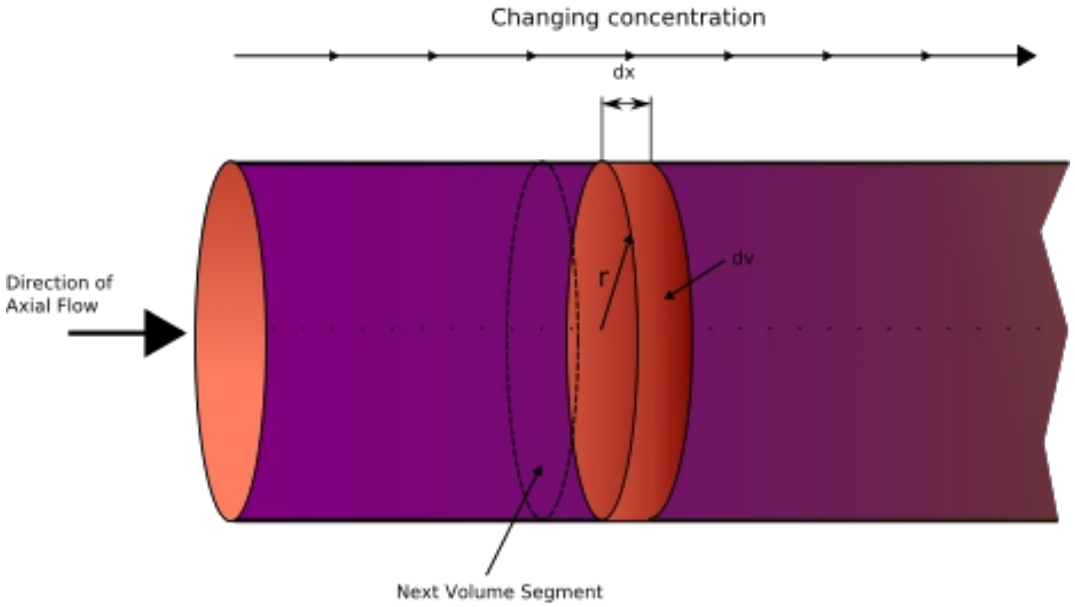
- Isaías Cuenca
- Ingeniero Químico
- Universidad de Alicante
  isais.cuenca@cacheme.org

3. Diseño de reactores en la industria de procesos químicos

CAChemE
cacheme.org

3.1. Reactor Flujo Pistón

- Craqueo térmico del etano.
- Tubos horizontales en el interior de un horno de llama.
- Quemadores a ambos lados de los tubos.
- Se considera la variación de presión a lo largo del reactor.
- Sistema ODE, 8 compuestos, T y P



CAChemE
cacheme.org

3.1. Reactor Flujo Pistón

Balance de Materia

Entrada - Salida + Generación = Acumulación

$$n_j - (n_j + dn_j) + r_j dV = 0 \rightarrow r_j = \frac{dn_j}{dV}$$

La velocidad de reacción:

$$r_i = k_i(T) \prod_j^S C_j^{\nu_{ij}}$$

Donde Cj:

$$C_j = \frac{n_j}{Qv}$$

Siendo nj:

$$n_j = n_{j0} + \sum_i^R \alpha_{ij} X_i$$

Y Qv se define como:

$$Qv = \frac{RT}{p} \sum_j^S n_j$$

Finalmente la variación de los moles de componente a lo largo del reactor es:

$$\frac{dn_j}{dz} = A \sum_i^R \alpha_{ij} r_i = A \sum_i \alpha_{ij} k_i(T) \prod_j^S \left(\frac{n_j}{Qv}\right)^{\nu_{ij}} = A \sum_i^R \alpha_{ij} k_i(T) \prod_j^S \frac{n_j^{\nu_{ij}}}{\left(\frac{RT}{p} \sum_j^S n_j\right)^{\nu_{ij}}} = f(n_j, T, p)$$

CAChemE
cacheme.org

3.1. Reactor Flujo Pistón

Balance de Energía

Entrada - Salida + Generación = Acumulación

$$dQ + \sum_j^S n_j h_j - \sum_j^S \left(n_j h_j + d(n_j h_j)\right) + 0 = 0$$

$$Cp_j \neq f(T), \Delta Cp \neq 0$$

$$\frac{dT}{dL} = \frac{\frac{dQ}{dL} - A\sum_i^R r_i \Delta H_i}{\sum_j^S n_j Cp_j} = f(n_j, T, p)$$

Balance de Energía Mecánica

$$\frac{dp}{dz} = \frac{\frac{2fQ_V}{D} + \frac{R}{\alpha P}\left(n_t \frac{dT}{dz} + T\sum_j^S \frac{dn_j}{dz}\right)}{\frac{RTn_t}{\alpha p^2} - \frac{A^2}{\rho Q_o}} \overset{n_j = n_{j0} + \sum_i^R \alpha_{ij} X_i}{=} \frac{\frac{2fQ_V}{D} + \frac{R}{\alpha P}\left(n_t \frac{dT}{dz} + T\sum_j^S \sum_i^R \alpha_{ij} \frac{dX_i}{dz}\right)}{\frac{RTn_t}{\alpha p^2} - \frac{A^2}{\rho Q_o}}$$

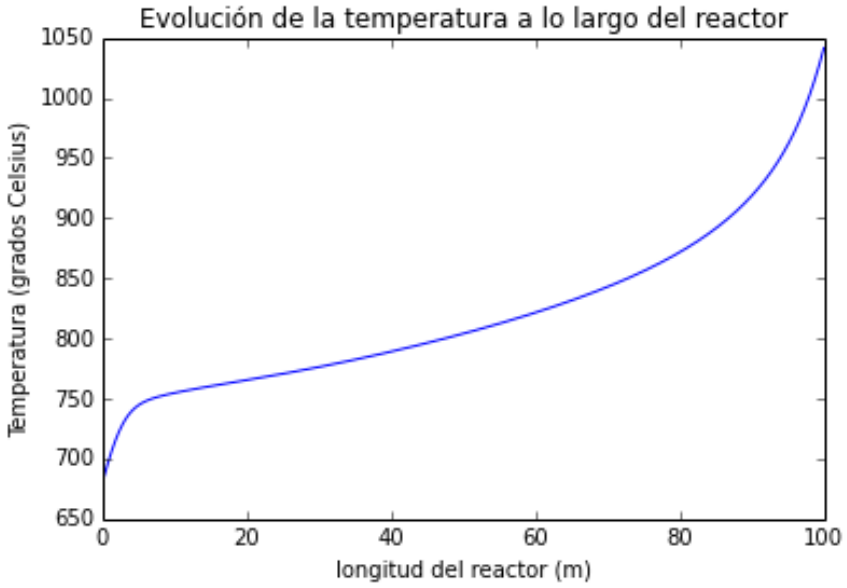
$$\mathrm{Re} = \frac{\rho v D}{\mu}$$

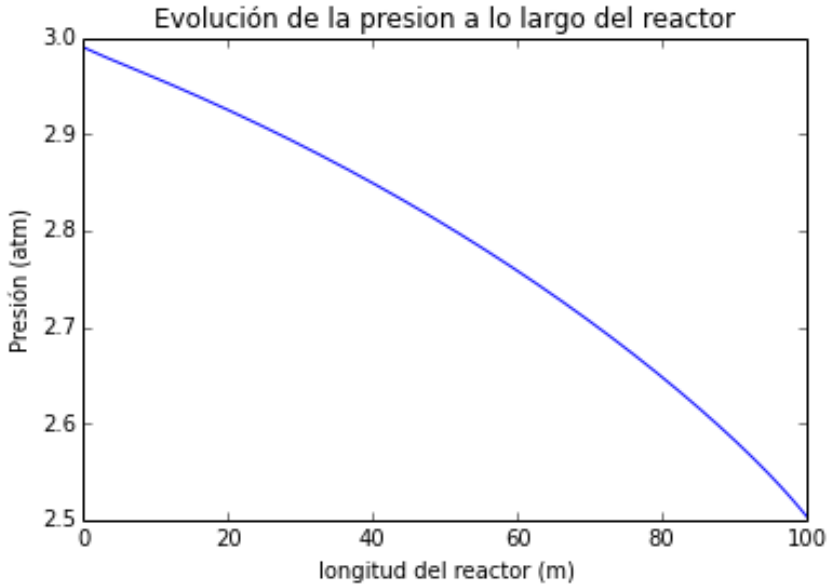
$$\rho = \frac{m(\mathrm{kg/s})}{Q_o} \rightarrow \rho Q_v = m = \sum_j^S n_j^{\mathrm{kmol/s}} PM_j^{\mathrm{kg/kmol}}$$

CAChemE
cacheme.org

3.1. Reactor Flujo Pistón

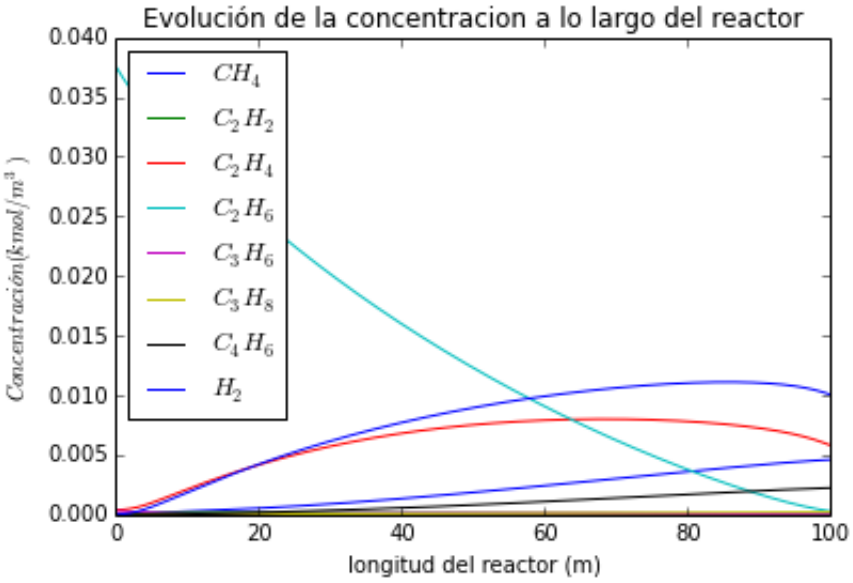
```
L = np.linspace (Lo, Lf, 1000)
# LLAMADA al solver de ecus. diff. ordinarias
#[L,y]=rk4(@prob6_f,[Lo Lf],yo,nint)
y=odeint(RFP_ode_system, yo, L)

X= y[:, 0:R]
T= y[:, R]
P= y[:, R + 1]
```

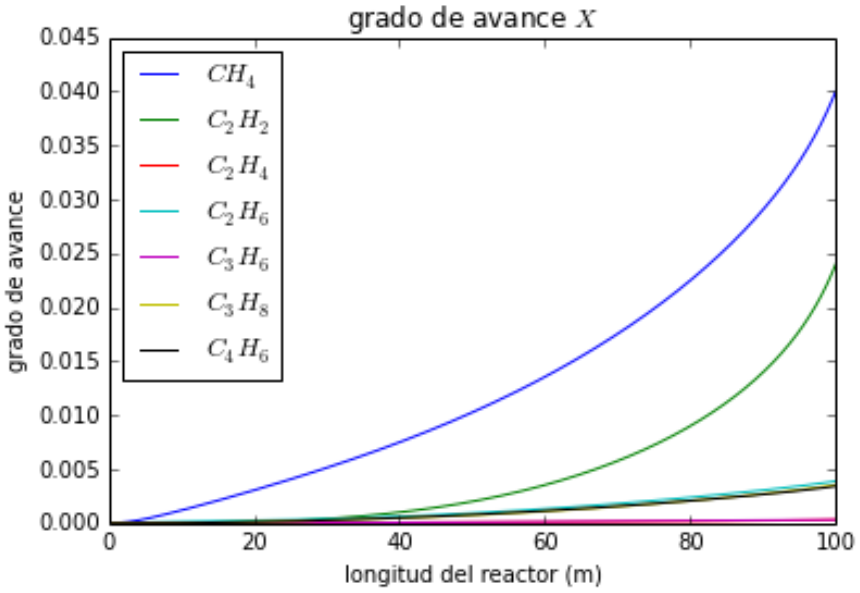




CAChemE
cacheme.org

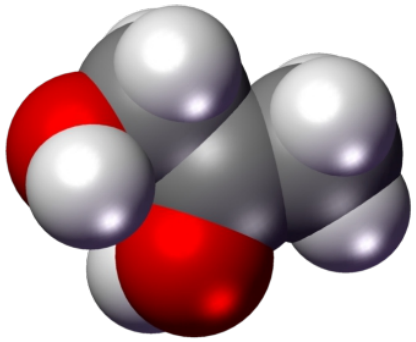
3.1. Reactor Flujo Pistón



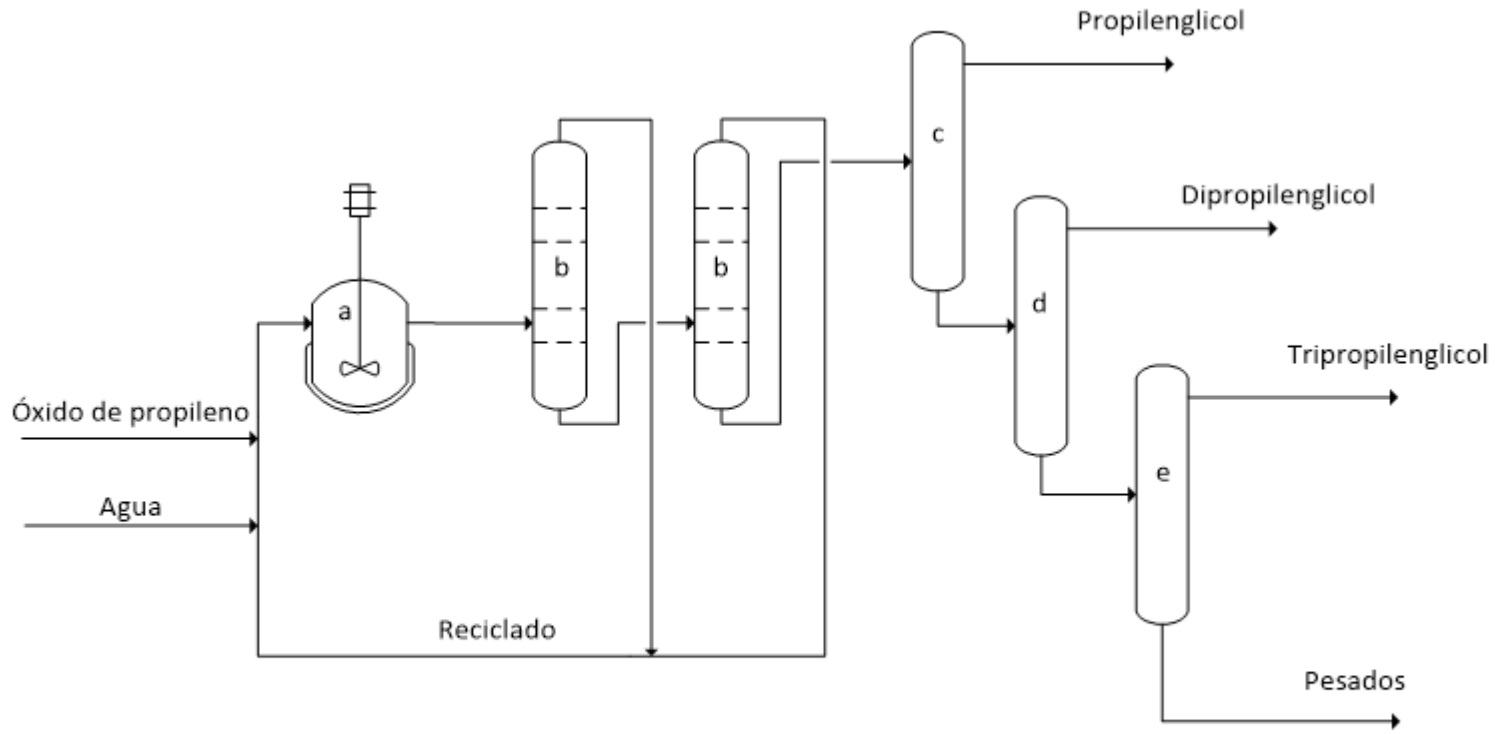


3. 2. Reactor Continuo Tanque Agitado

- Propilenglicol (IUPAC: propano-1,2-diol)
- Compuesto orgánico inodoro, incoloro e insípido
- Líquido aceitoso claro, higroscópico y miscible con agua, acetona, y cloroformo.
- Utilizado en una amplia gama de productos de consumo, incluidos los alimentos, piensos, cosméticos y productos farmacéuticos, así como las aplicaciones industriales.
- Producción mundial: 900 000 t /año *(Fuente: Ullmanns's)*
- Producido típicamente en dos calidades
- Calidad industrial
- Calidad USP/EP
-



3. 2. Reactor Continuo Tanque Agitado

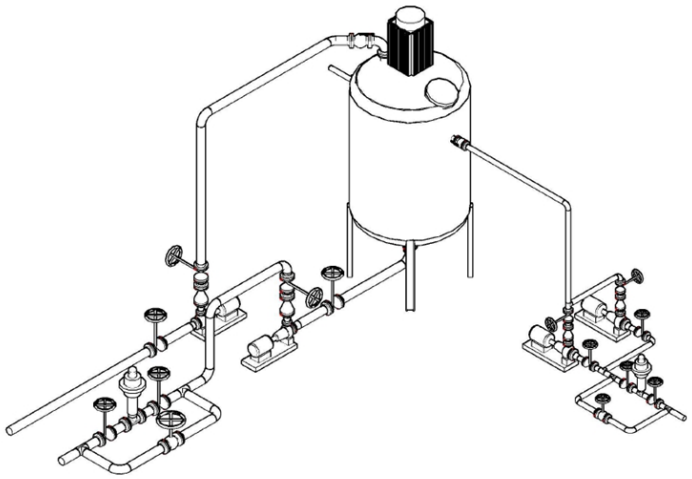


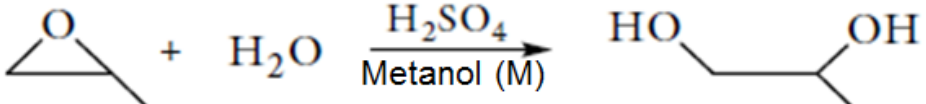
a) Reactor; b) evaporadores (normalmente 2-4 columnas); c, d, e) columnas de destilación de productos.

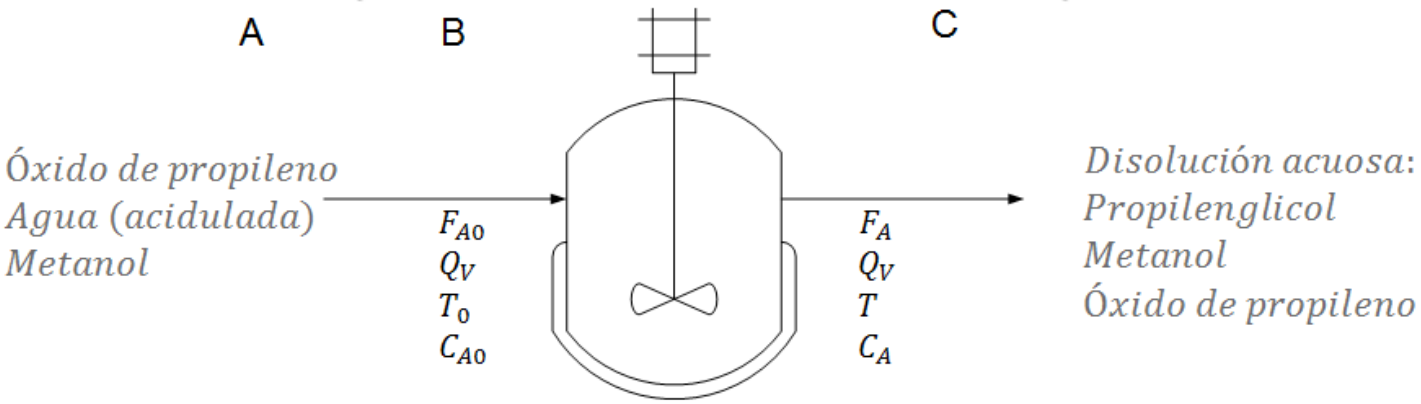
CAChemE
cacheme.org

3. 2. Reactor Continuo Tanque Agitado

Variable	Descripción	Valor
F_{A0}	Flujo molar de entrada de óxido de propileno (kmol/h)	36.3
F_{B0}	Flujo molar de entrada de agua y ácido sulfúrico (kmol/h)	453.6
$6F_{M0}$	Flujo molar de entrada de metanol (kmol/h)	45.4
Q_0	Caudal volumétrico de entrada (m^3/h)	12.5







3. 2. Reactor Continuo Tanque Agitado

Balance de Materia

ENTRADA – SALIDA + GENERACIÓN = 0

$$\frac{dC_A}{dt} = \frac{Q_o}{V_r}(C_{A0} - C_A) + r_A$$
$$\frac{dC_B}{dt} = \frac{Q_o}{V_r}(C_{B0} - C_B) + r_A$$
$$\frac{dC_C}{dt} = \frac{Q_o}{V_r}(-C_C) - r_A$$
$$\frac{dC_M}{dt} = \frac{Q_o}{V_r}(C_{M0} - C_M)$$

Balance de Energía

Estado estacionario

$$X_{BE} = -\frac{\sum \Theta_i \tilde{C}_{p_i} \cdot (T - T_0) + \dot{Q}_{ext}/F_{A0}}{\Delta H_r(T)}$$

Siendo el primer término:

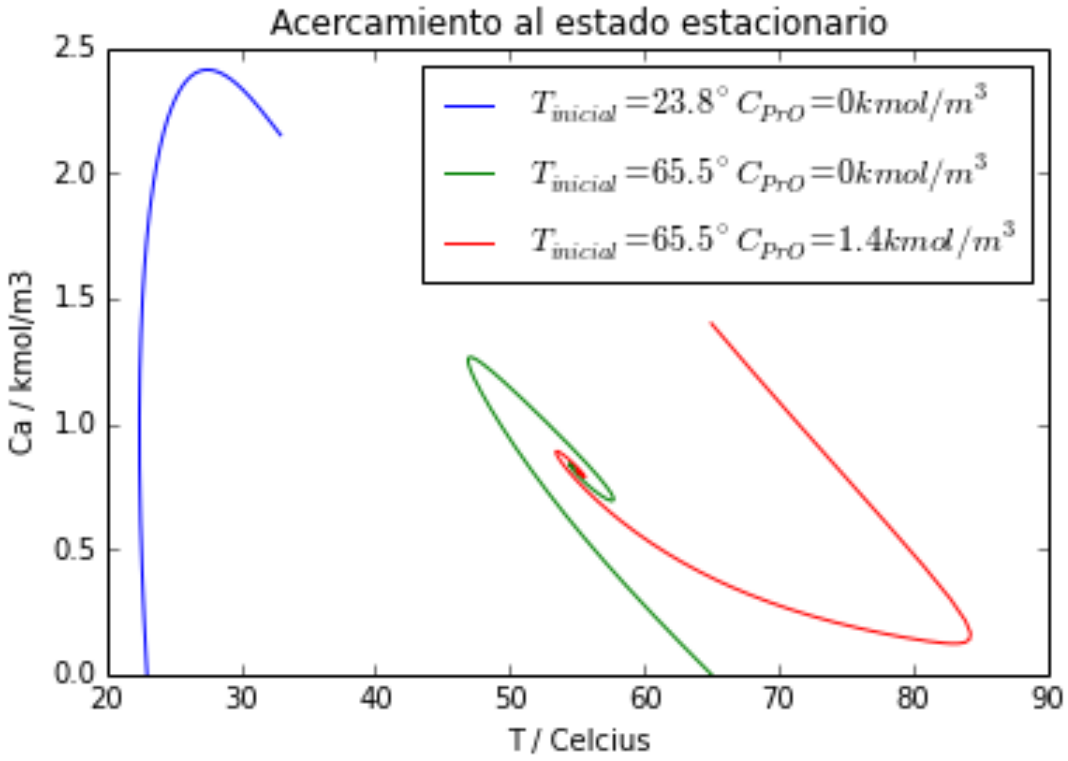
$$\sum \Theta_i \tilde{C}_{p_i} = C_{p\,A} + \frac{F_{B0}}{F_{A0}} C_{p\,B} + \frac{F_{C0}}{F_{A0}} C_{p\,C} + \frac{F_{M0}}{F_{A0}} C_{p\,M}$$

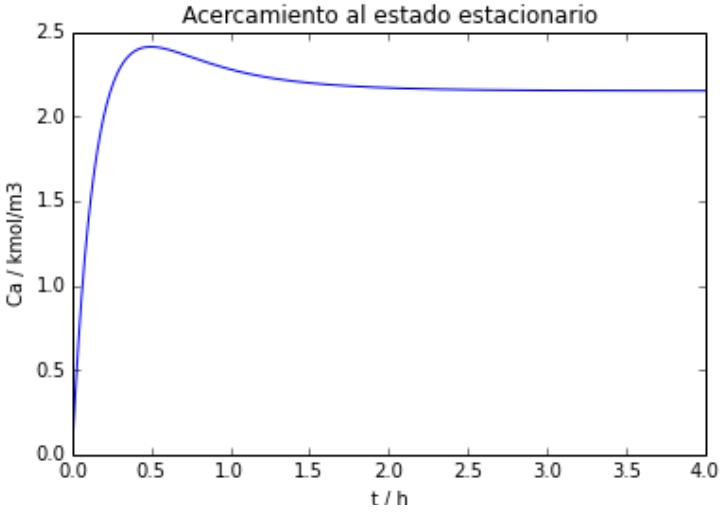
Arranque del reactor

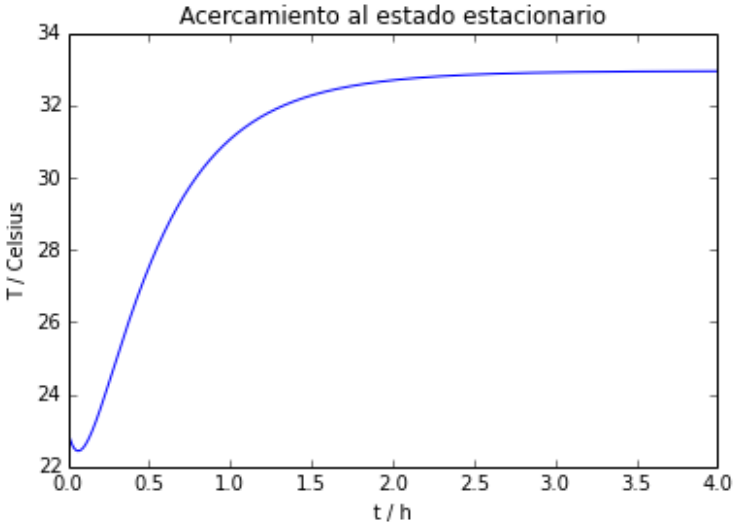
$$\frac{dT}{dt} = \frac{\dot{Q}_{ext} - F_{A0} \sum \Theta_i \tilde{C}_{p_i} \cdot (T - T_0) + V_r \Delta H_r r_A}{\sum C_i V C_{Pi}}$$

CAChemE
cacheme.org

3. 2. Reactor Continuo Tanque Agitado



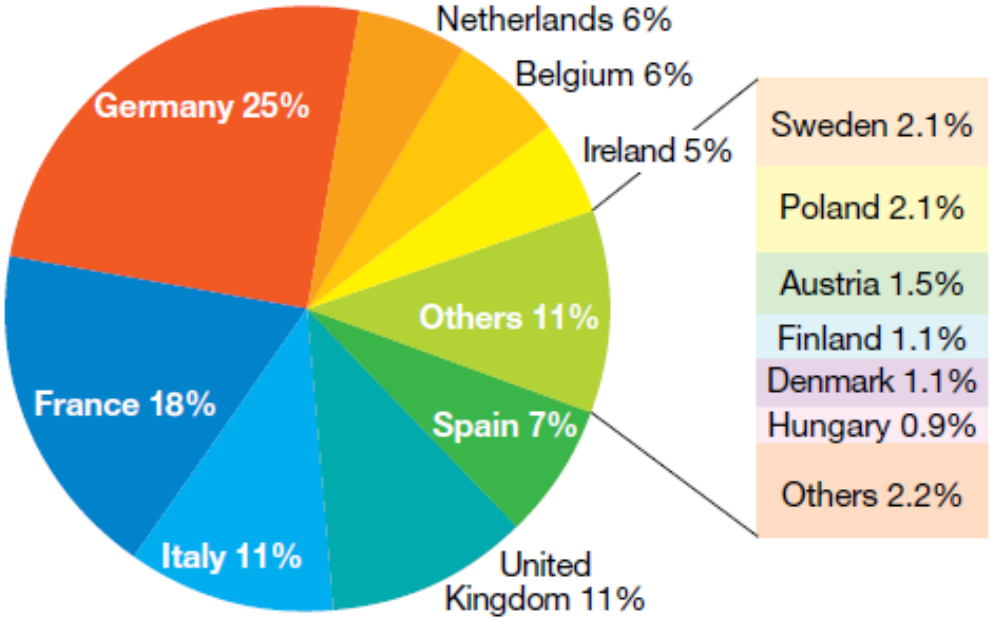




CAChemE
cacheme.org

La Industria Química en España

- 13% PIB Industrial
- 5º Europa y 8º Mundial
- 2º mayor exportación solo superado por la automoción
-



Fuente: American Institute of Chemical Engineers (AICHE) 2012

CAChemE
cacheme.org

La Industria Química en España

- 239.300 empleo
- Ocupación especializada y cualificada
- Una de ellas, técnico en química computacional

Table 1. Economic characteristics of Spain's chemical industry.					
Year	**2000**	**2005**	**2008**	**2009**	**2010**
Companies	N/A	3,649	3,571	3,408	3,311
Jobs	164,500	166,400	191,100	166,800	167,600
Sales*	35,771	44,035	52,585	47,714	53,169
Exports*	11,738	18,476	23,230	21,201	26,367
Imports*	18,147	26,421	32,203	26,837	32,713

* million euros
Source: Federación Española de Industrias Químicas (FEIQUE).

CAChemE
cacheme.org

Conclusiones

- Python junto a sus librerías permite la resolución de problemas típicos de ingeniería y todo ello en un mismo lenguaje-entorno.

- Python permite centrarse en el algoritmo y no en la sintaxis del lenguaje. Es la *navaja suiza* de los lenguajes permitiendo pasar a C en cualquier momento que se necesite.

- Es multiplataforma, libre y gratuito por lo que su adopción en universidades y empresa es sencilla

- Si se quiere una herramienta lo más compatible con MATLAB, Octave UPM es otra opción interesante pero mantendrá sus mismas capacidades (y limitaciones) técnicas.

- **Python es posiblemente la mejor opción como primer lenguaje de programación en el ámbito de programación científica.**

¡Muchas gracias!

CAChemE

www.cacheme.org


@CAChemEorg


CAChemEorg


CAChemE

info@cacheme.org


PyConES 2013