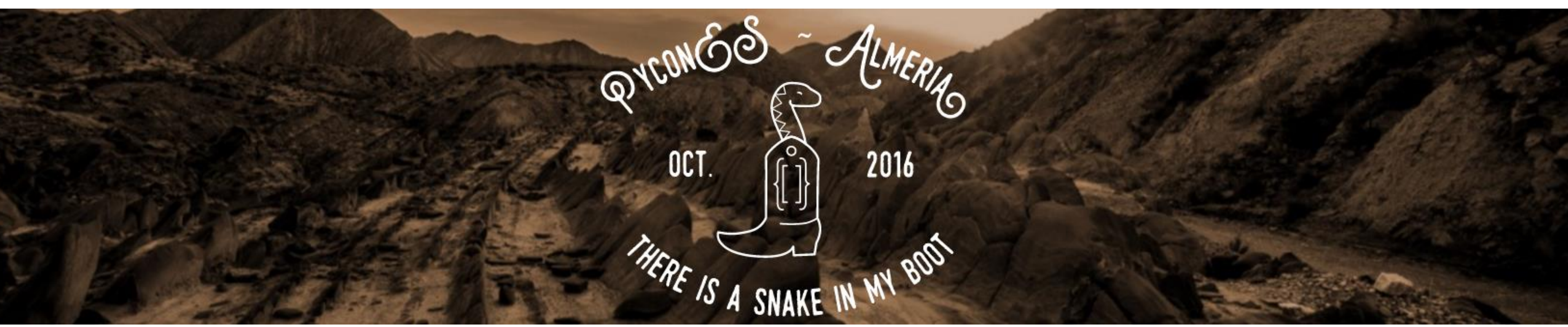
CAChemE

[>>COnCEPT]
Computer Optimization
in Chemical Engineering
Processes and Technologies

Simulation-Based Optimization using the Particle Swarm Optimization Algorithm

(Aprendiendo magia negra con Python, optimización estocástica y simuladores)

Juan Javaloyes & Francisco Navarro

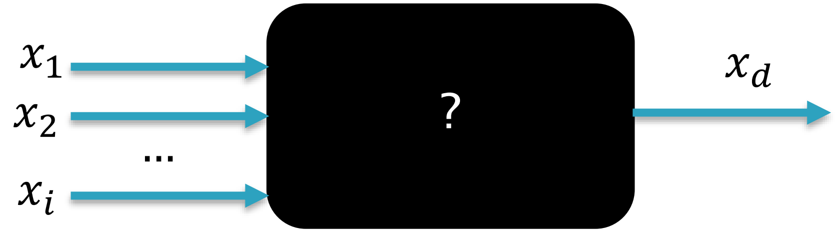


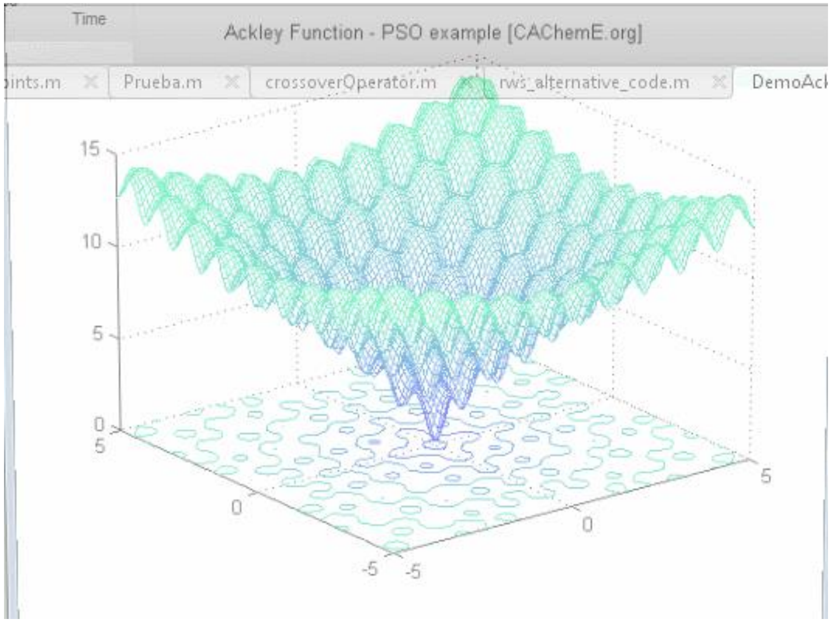
https://github.com/CAChemE/stochastic-optimization

October 2016

Particle Swarm Optimization Algorithm

- Introduction
- Mathematical Programming (optimization)
- Particle Swarm Optimization Algorithm
  - Standard PSO Algorithm
  - Implementation
- Case Studies
  - Conventional Distillation Column Optimization
  - Divided Wall Column Optimization
  - Vapor Recompression Cycle Optimization





Optimization (background and motivation)

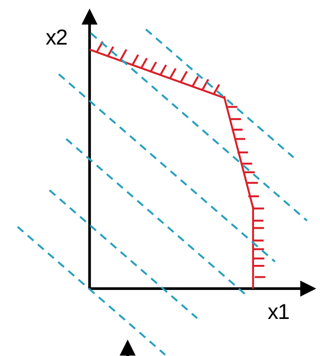
Optimization Problems

$$\min \quad f_0(x)$$ ← Objective Function

$$s.t \quad f_i(x) \le b_i \qquad i = 1, \ldots, m.$$ ← Constraints

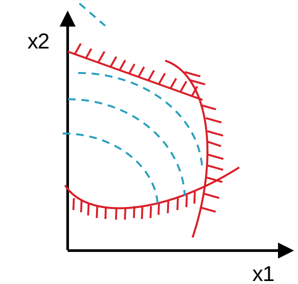
$$x \in R^n$$

$$f_0(x): R^n \rightarrow R, \; f_i(x): R^n \rightarrow R$$



$$f_i(\alpha x_1 + \beta x_2) = \alpha f_i(x_1) + \beta f(x_2) \quad x_1, x_2 \in R^n \quad \alpha, \beta \in \mathrm{R}$$

← Linear Program



$$f_i(\alpha x_1 + \beta x_2) \ne \alpha f_i(x_1) + \beta f(x_2) \quad x_1, x_2 \in R^n \quad \alpha, \beta \in \mathrm{R}$$

← Nonlinear Program

$$f_i(\alpha x_1 + \beta x_2) \le \alpha f_i(x_1) + \beta f(x_2) \quad x_1, x_2 \in R^n$$

← Convex Optimization Problem

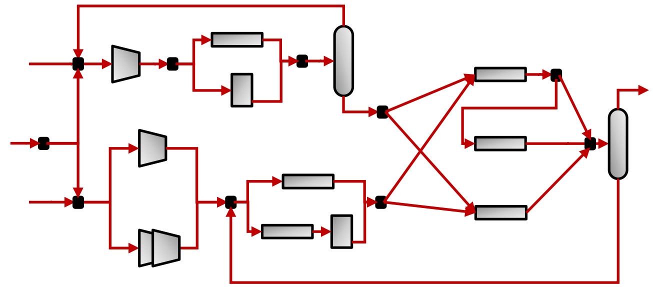
$$\alpha, \beta \in R \text{ with } \alpha + \beta = 1, \quad \alpha, \beta \ge 0$$

Mathematical Programming in Chem. Eng.

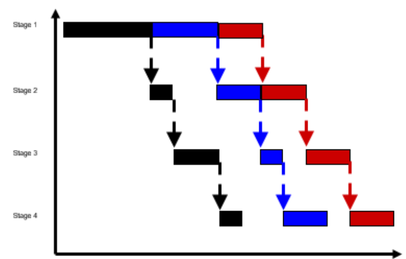
Application of Mathematical Programming in Chemical Engineering

Process Design

Process Synthesis/Integration

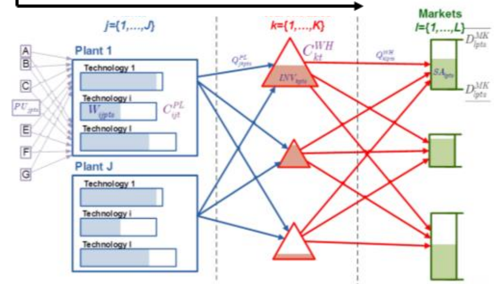


Production Planning

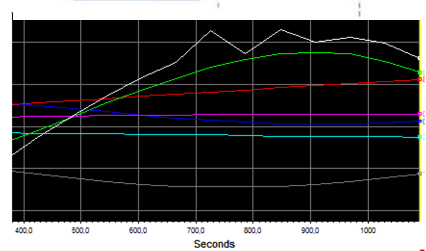


Process Scheduling

Supply Chain Management



Process Control



Parameter-tuning

Slide adapted from
Recent Developments in the Application of Mathematical Programming to Process Integration. Grossmann (2013)

Mathematical Programming in Chem. Eng.

Mixed-integer nonlinear programming

[Most general and most common optimization problem in Process System Engineering (PSE)]

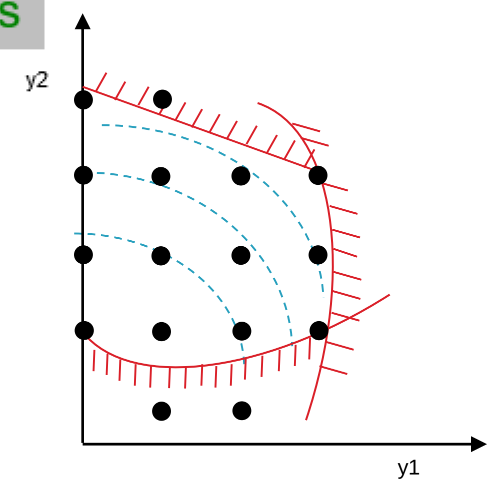
$$\min \quad f(x,y)$$
← Objective Function

$$s.t \quad h_i(x,y)=0 \qquad i=1,\ldots,m.$$
← Process equations

$$g_j(x,y)\le 0 \qquad j=1,\ldots,q.$$
← Process specifications

$$x\in R^n, \quad y\in\{0,1\}^q$$

$$f(x,y):R^n\to R,\ h_i(x,y):R^n\to R,\ g_j(x,y):R^n\to R$$



Mathematical Programming in Chem. Eng.

Solvers

$$
\begin{aligned}
\min \quad & f(x,y) \\
s.t \quad & h_i(x,y) = 0 \qquad i = 1,\ldots,m. \\
& g_j(x,y) \le 0 \qquad j = 1,\ldots,q. \\
& x \in R^n, \quad y \in \{0,1\}^q
\end{aligned}
$$

Branch and Bound
Ravindran & Gupta 1985;
Leyffer & Fletcher 2001
Branch & Cut: *Stuubs & Mehrota 1999*

Generalized Benders Decomposition
Geofrion, 1972

Outer Approximation
Duran & Grossmann 1986;
Yuan et al 1988;
Fletcher & Leyffer 1994

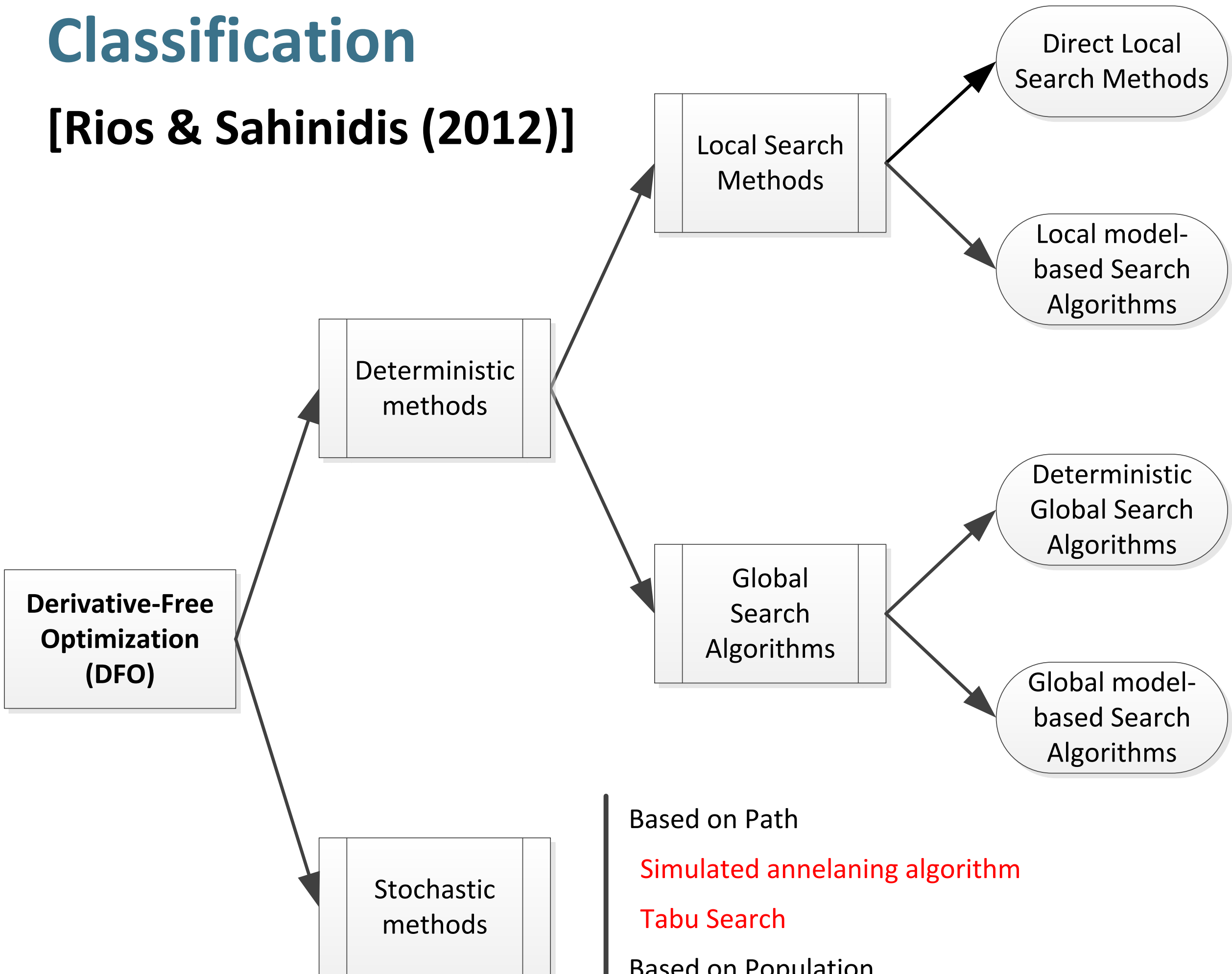
LP/NLP Based Branch and Bound
Quesada & Grossmann 1992

Extended Cutting Plane
Westerlund & Petersen 1995

Derivative Free Optimization Algorithms

Classification

[Rios & Sahinidis (2012)]



Nelder-Mead simplex algorithm
Generalized Patern Search (GPS) method
Generating Set Search (GSS) method
...

Implictit filtering

Trust-region methods

...

Lipschizian-based partitionint techniques

Multilevel coordinate search (MCS) Algorithm

...

Response surface methods (RSMs)
Surrogate managment framework (SMF)
Optimization by branch-and-fit
...

Based on Path
- Simulated annelaning algorithm
- Tabu Search

Based on Population
- Genetic algorithms
- **Particle swarm algorithm**
- ...

Derivative Free Optimization Algorithms

"… if you can obtain clean derivatives (even if it requires considerable effort) and the functions defining your problem are smooth and free of noise you should not use derivative-free methods."

Introduction to Derivative-Free Optimization
Andrew R. Conn, 2009

- Simulated Annealing, Genetic Algorithms etc are usually for the ignorant or the desperate or both.

Andrew R. Conn – IBM T. J. Watson Research Center
MINLP Workshop 2014, Pittsburgh (link)

Particle Swarm Optimization Algorithm

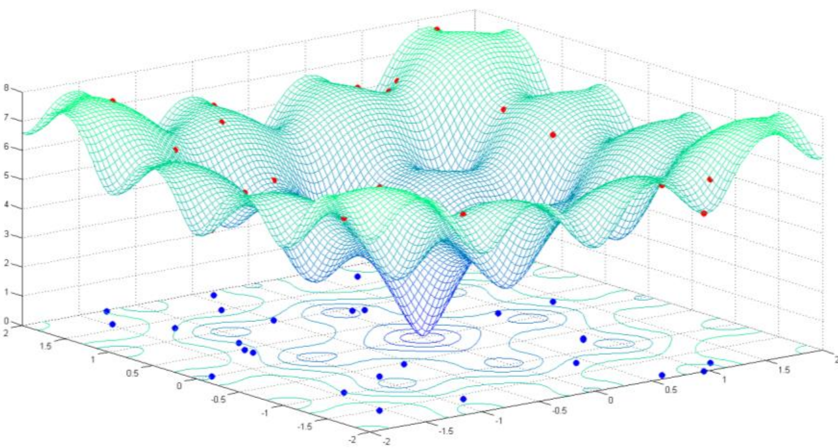
Particle Swarm Optimization Algorithm

PSO Overview

The original particle swarm optimization algorithm

(Kennedy & Eberhart, 1995)

PSO is a robust stochastic method for solving global optimization problems, inspired by the flocking and schooling patterns of birds and fish



Each particle is represented by

- Its current position $\left(x^{i}, y^{i}\right)$
- Its current velocity v^{i}
- Its personal best position p^{i}

PSO parameters

Initialize Swarm
(random position and velocity vectors)

Evaluate objective function for all the particles
$f\left(x_k^i, y_k^i\right)$

Update particle best p_k^i
Uptade particle global best p_k^{global}

Update velocity for all particles
$$v_{k+1}^i = \omega_k v_k^i + c_1 r_1 \left(p_k^i - x_k^i\right) + c_2 r_2 \left(p_k^{global} - x_k^i\right)$$

Update continuous variables
$$x_{k+1}^i = x_k^i + v_{k+1}^i$$

Update discrete variables
$$y_{k+1}^i = \begin{cases} 1 \ \ if \ \ r_{3,l} \le sig\left(v_{l,k}^i\right) \\ 0 \ \ if \ \ r_{3,l} > sig\left(v_{l,k}^i\right) \end{cases} \qquad sig(x) = \frac{1}{1+e^{(-x)}}$$

Stopping criteria satisfied?

No: $k = k+1$

Yes

Ouput Results
(p_k^{global} optimal solution)

CAChemE

Particle Swarm Optimization Algorithm

PSO Overview

The original particle swarm optimization algorithm

(Kennedy & Eberhart, 1995)

PSO is a robust stochastic method for solving global optimization problems

PSO parameters

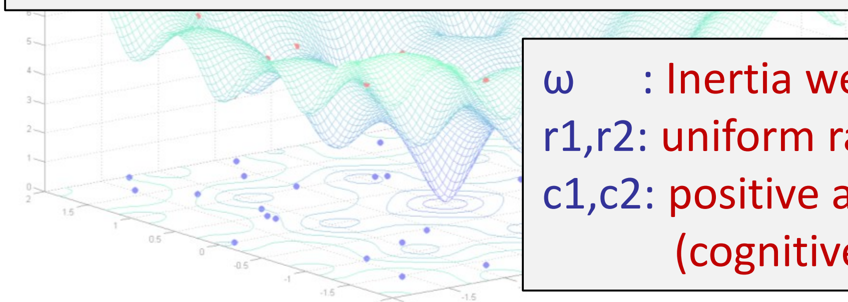
Initialize Swarm
(random position and velocity vectors)

Evaluate objective function for all the particles
$f\left(x_k^i, y_k^i\right)$

Update velocity for all particles

$$v_{k+1}^i = \omega_k v_k^i + c_1 r_1 \left(p_k^i - x_k^i \right) + c_2 r_2 \left(p_k^{global} - x_k^i \right)$$

Update continuous variables
$x_{k+1}^i = x_k^i + v_{k+1}^i$



ω : Inertia weight
r1,r2: uniform random vectors [0,1]
c1,c2: positive acceleration coefficients (cognitive and social)

...te discrete variables
$\le sig\left(v_{l,k}^i\right)$
$> sig\left(v_{l,k}^i\right)$
$sig(x) = \frac{1}{1+e^{(-x)}}$

No

Stopping criteria satisfied?

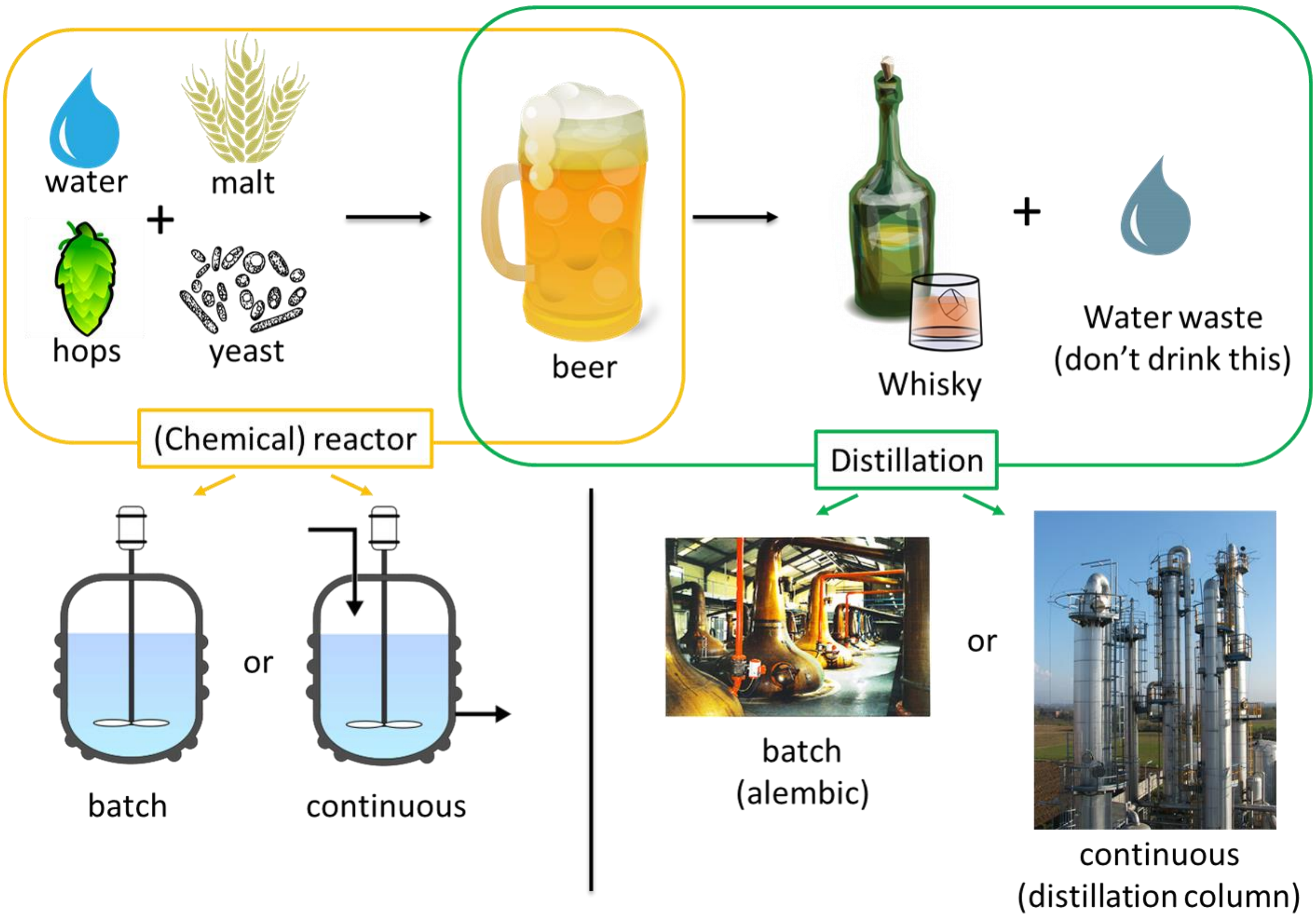
Yes

Ouput Results
(p_k^{global} optimal solution)

Each particle is represented by

- Its current position $\left(x^i, y^i\right)$
- Its current velocity v^i
- Its personal best position p^i

Chemical engineering in a nutshell:



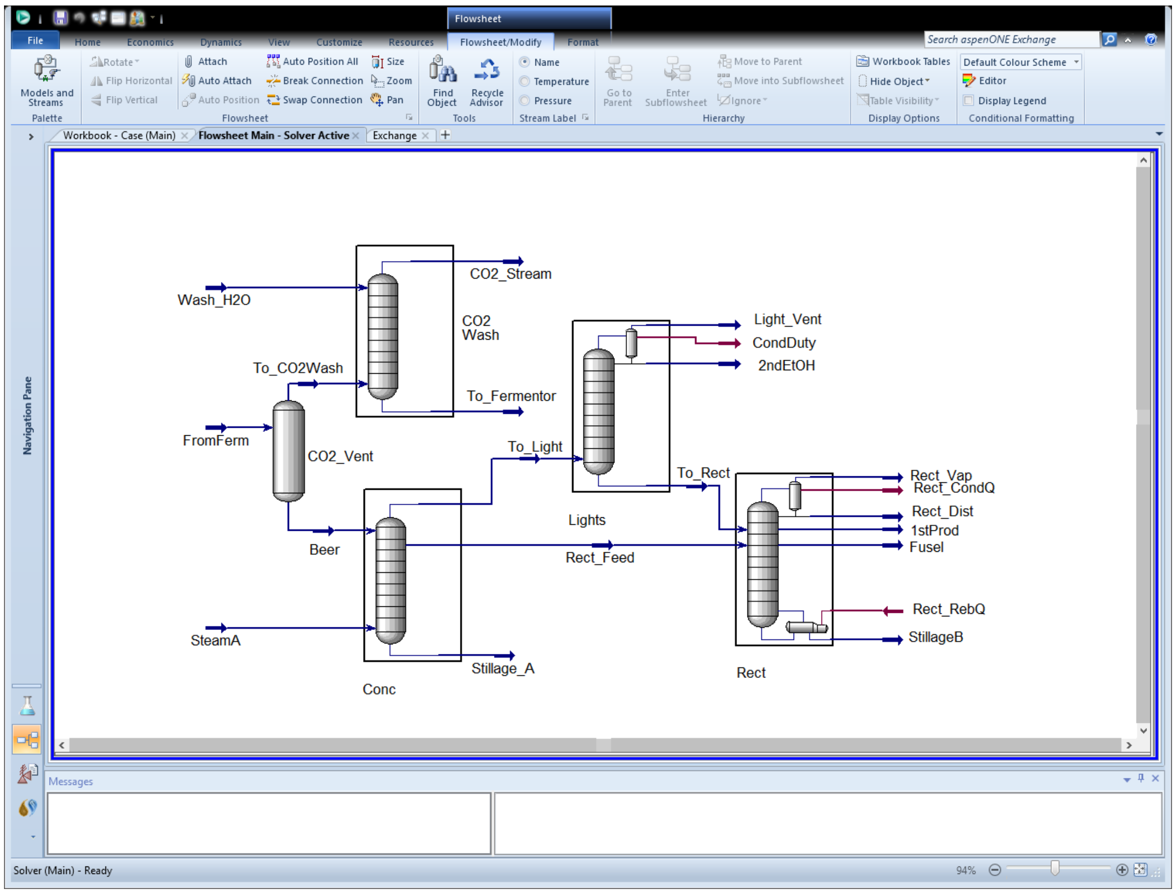
Case Studies

Process Flowsheeting Programs

Flowsheeting

Flowsheeting is a systemic description of material and energy streams in a process plant by means of computer simulation with the scope of designing a new plant or improving the performance of an existing plant. Flowsheeting can be used as aid to implement a plantwide control strategy, as well as to manage the plant operation.

Alexander C. Dimian (2003)



Process Flowsheeting Programs

Main challenges in Simulation-Based Optimization

[Sequential-Modular approach]

[# 1] Derivatives are not directly available for many SM flowsheeting programs.

[# 2] Although derivatives can be calculated by numerical differentiation. They can be very expensive to compute and most of the process units introduce numerical noise.

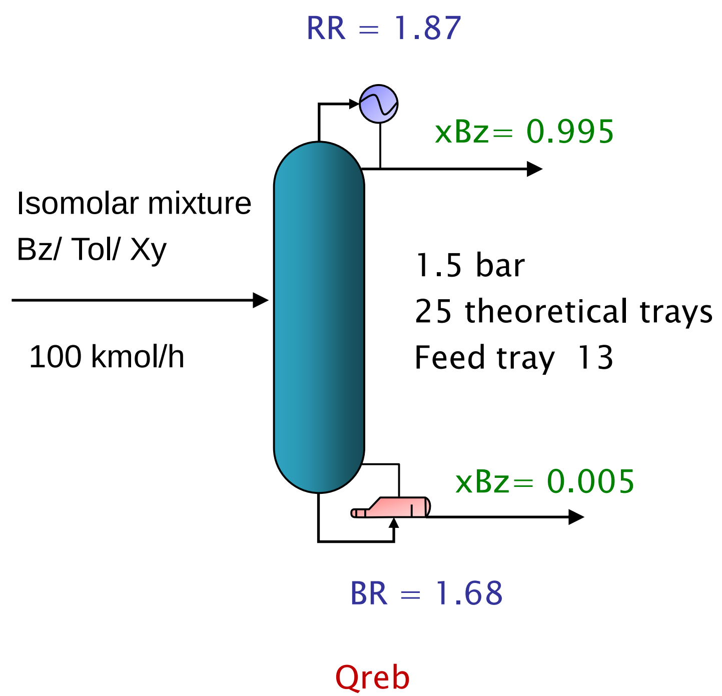
[# 3] As simulations become more complex, the robustness (in terms of convergence) decreases, and the simulations become prone to convergence failures.

Main challenges in Simulation-Based Optimization

[Sequential-Modular approach]

Consider the following numerical experiment

1. Converge the Distillation Column using fix values of RR and BR and read heat load in reboiler
2. Randomly select new values for RR and BR and converge the column again.
3. Repeat step 1. The heat load should be the same that in step 1, but…

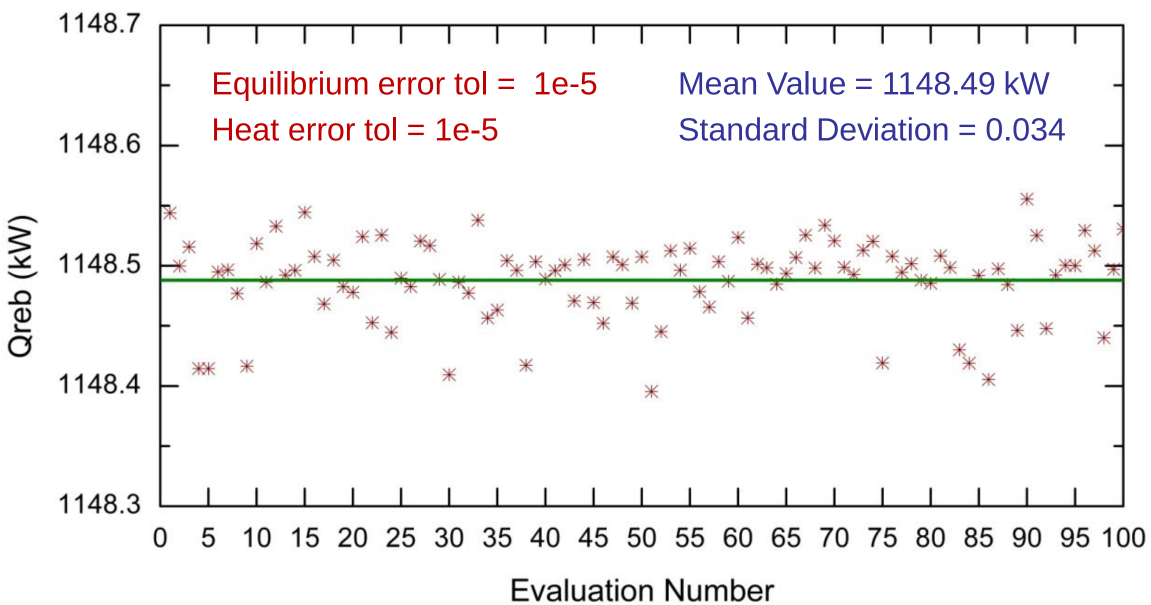


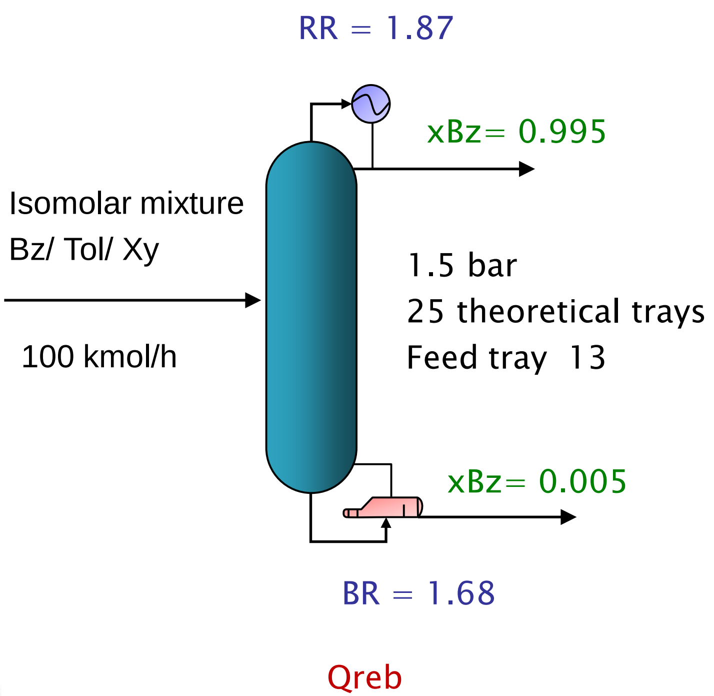
Process Flowsheeting Programs

Main challenges in Simulation-Based Optimization

[Sequential-Modular approach]

Consider the following numerical experiment



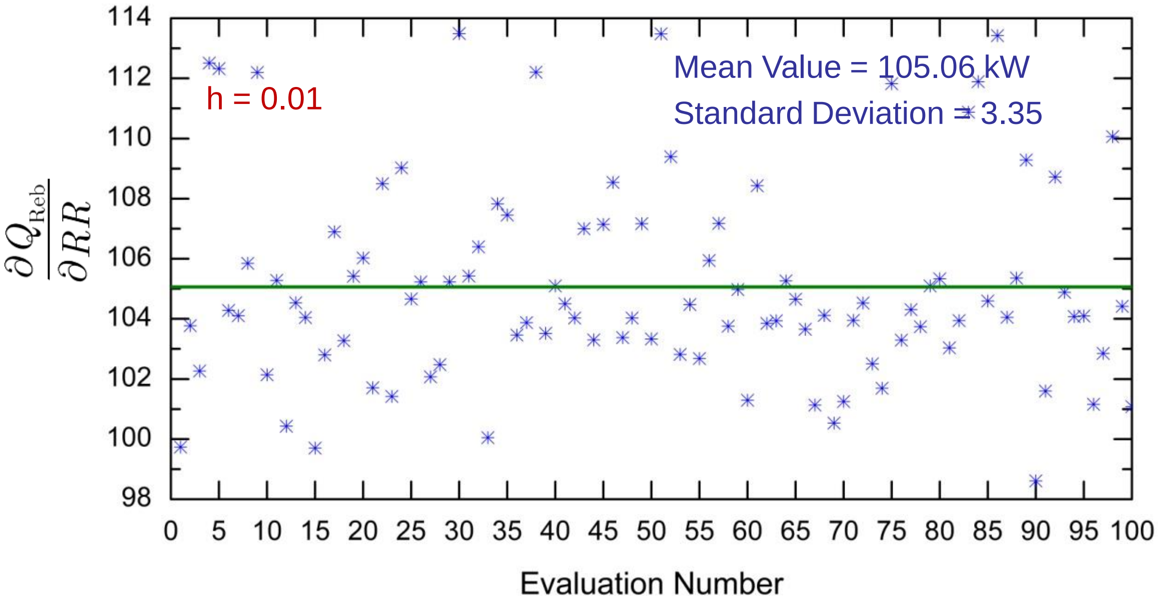


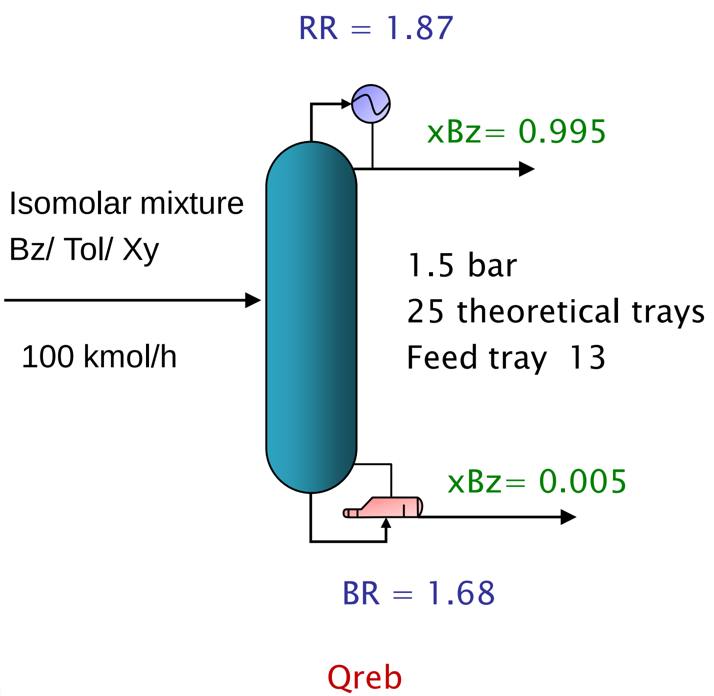
Process Flowsheeting Programs

Main challenges in Simulation-Based Optimization

[Sequential-Modular approach]

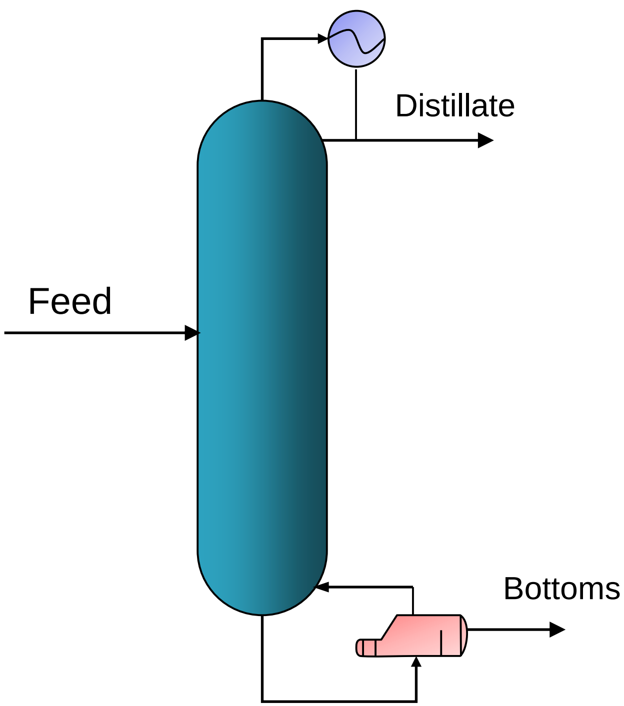
Consider the following numerical experiment





Economic Optimization of Distillation Columns

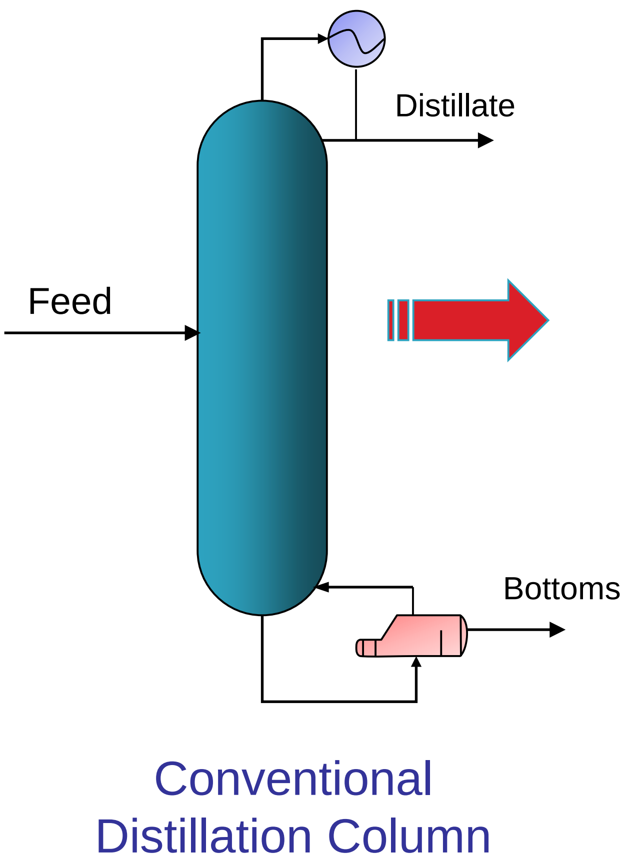
Conventional Distillation Column

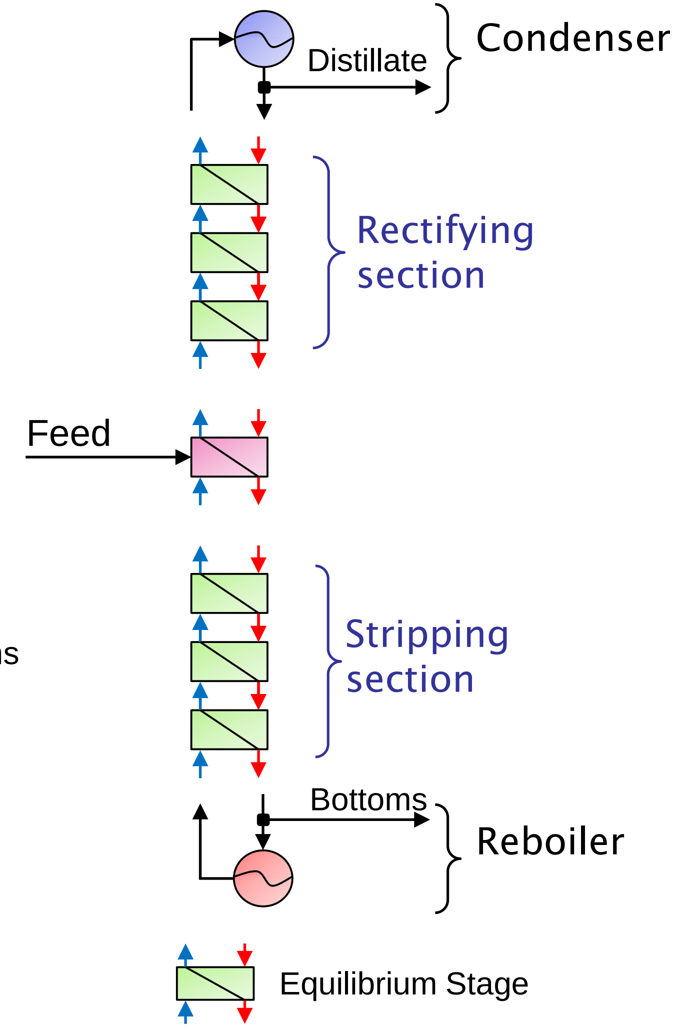


Conventional Distillation Column

Economic Optimization of Distillation Columns

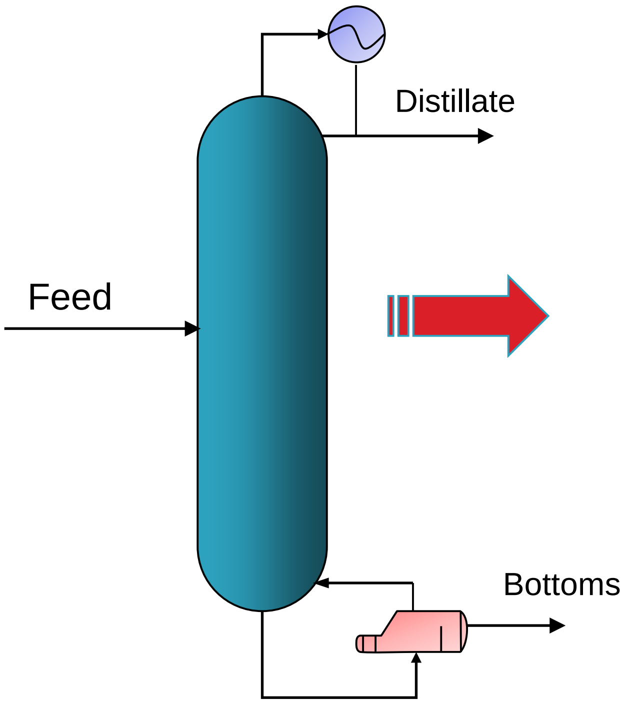
Conventional Distillation Column





Economic Optimization of Distillation Columns

Conventional Distillation Column



Conventional Distillation Column

$$\min TAC = C_{op}\left(Qreb, Qcond\right) + FC_{cap}\left(NT_{Col}, D_{Col}, A_{Reb}, A_{Cond}\right)$$

$s.t.$ Design Constraints

Fixed trays

$MESH_{\ Equations}$

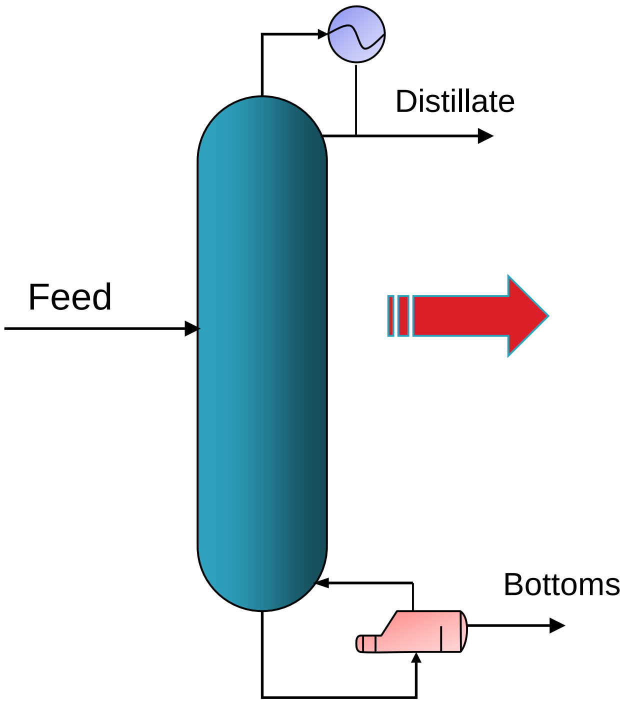
Conditional trays

$MSH_{\ Equations}$

$$\begin{bmatrix} Y_n \\ VLE \end{bmatrix} \underline{\vee} \begin{bmatrix} \neg Y_n \\ Not\,VLE\ \text{(bypass)} \end{bmatrix}$$

Economic Optimization of Distillation Columns

Conventional Distillation Column



Conventional Distillation Column

Objective Function

$$\min \quad TAC = f\left(NT, DC, AC\right) + CS\ QR + CW\ QC$$

Overall constraints

$$\left.\begin{array}{l} F_i = D_i + B_i \\ D_i \geq \xi_i F_i \\ y_n^i \geq \tau_i \end{array}\right\} i \in C$$

← Column mass balance

← Component recovery

← Component purity

$$NT = \sum_{n \in TC} STG_n$$

← Number of stages

$$DC \geq g\left(T_n^V, P_n, VAP_n\right) \qquad n \in TC$$

← Column diamete

$$AR = QR \big/ U^R \left(T^S - T_1^L\right)$$

← Reboiler area

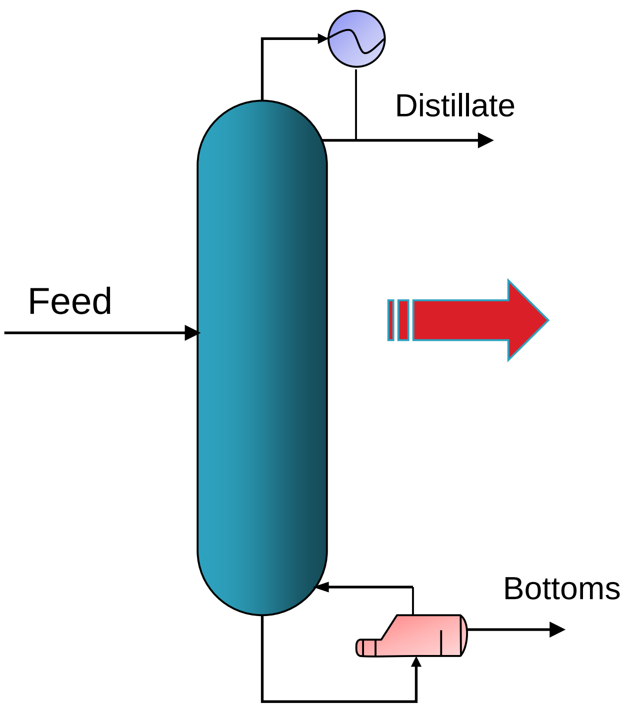
$$AC = QC \big/ U^C \left(T_N^V - T^{cw}\right)$$

← Condenser area

Yeomans & Grossmann (2000)

Economic Optimization of Distillation Columns

Conventional Distillation Column



Conventional Distillation Column

Feed Tray ← Fixed tray

$$\left.\begin{array}{l}
F_n^i + L_{n+1}^i + V_{n-1}^i - L_n^i - V_n^i = 0 \\
F_n^i = FT\ x_F^i \\
L_n^i = LIQ_n\ x_n^i \\
V_n^i = VAP_n\ y_n^i \\
\sum_{i \in C} \begin{pmatrix} F_n^i\, hF_i\, h + L_{n+1}^i\ hL_{n+1}^i + V_{n-1}^i\, hV_{n-1}^i \\ -L_n^i\ hL_n^i - V_n^i\, hV_n^i \end{pmatrix} = 0 \\
hL_n^i = f\left(T_n^L\right) \\
hV_n^i = f\left(T_n^V\right) \\
hF_i = f\left(T_n^L\right) \\
\sum_{i \in C} x_n^i = 1 \qquad \sum_{i \in C} y_n^i = 1 \\
f_{i,n}^L = f\left(T_n^L, P_n, x_n\right) \\
f_{i,n}^V = f\left(T_n^V, P_n, y_n\right) \\
f_{i,n}^L = f_{i,n}^V \\
T_n^L = T_n^V \\
STG_n = 1
\end{array}\right\} n \in NFT$$

← Component MB

← Energy Balance

← Summation of mole fractions

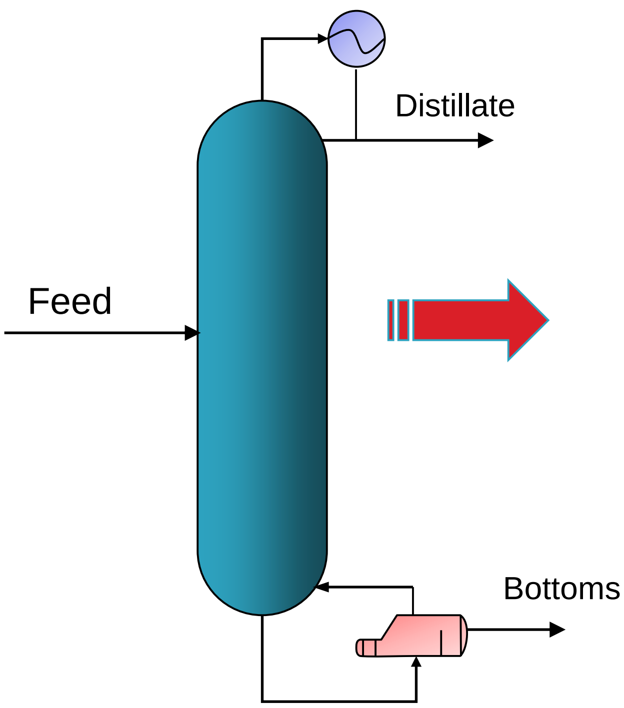
← Equilibrium equations

← Liq-Vap equilibrium condition

Yeomans & Grossmann (2000)

Economic Optimization of Distillation Columns

Conventional Distillation Column



Conventional Distillation Column

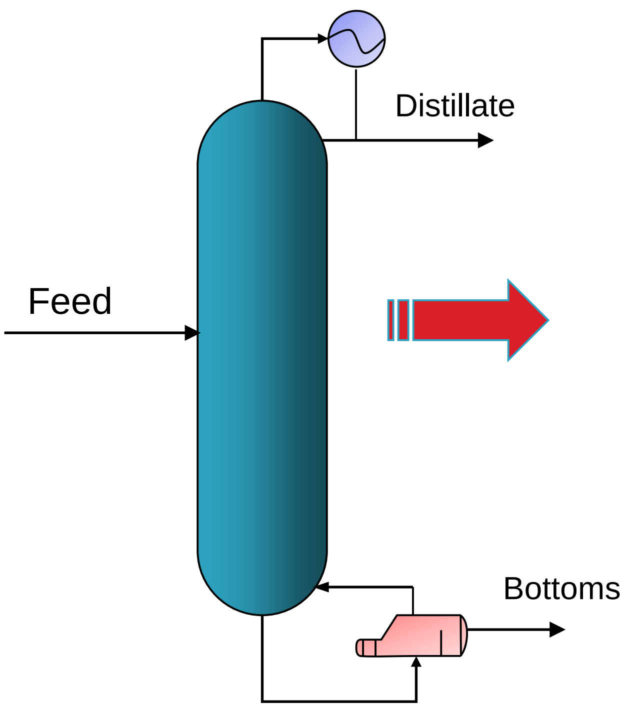
$* \; Condenser$

$$
\left.
\begin{aligned}
& V^i_n - L^i_{n+1} - D_i = 0 \\
& D_i = DIS\; y^i_n \\
& D_i = R\; L^i_n \\
& L^i_n = LIQ_n\; x^i_n \\
& V^i_n = VAP_n\; y^i_n \\
& \sum_{i \in C} \left(V^i_n\, hV^i_n - L^i_{n+1}\, hL^i_{n+1} - D_i\, hD_i \right) = QC \\
& hL^i_n = f\left(T^L_n\right) \\
& hV^i_n = f\left(T^V_n\right) \\
& hD_i = f\left(T^L_n\right) \\
& \sum_{i \in C} x^i_n = 1 \qquad \sum_{i \in C} y^i_n = 1 \\
& f^L_{i,n} = f\left(T^L_n, P_n, x_n\right) \\
& f^V_{i,n} = f\left(T^V_n, P_n, y_n\right) \\
& f^L_{i,n} = f^V_{i,n} \\
& T^L_n = T^V_n \\
& STG_n = 1
\end{aligned}
\right\} n \in NCT
$$

Yeomans & Grossmann (2000)

Economic Optimization of Distillation Columns

Conventional Distillation Column



Conventional Distillation Column

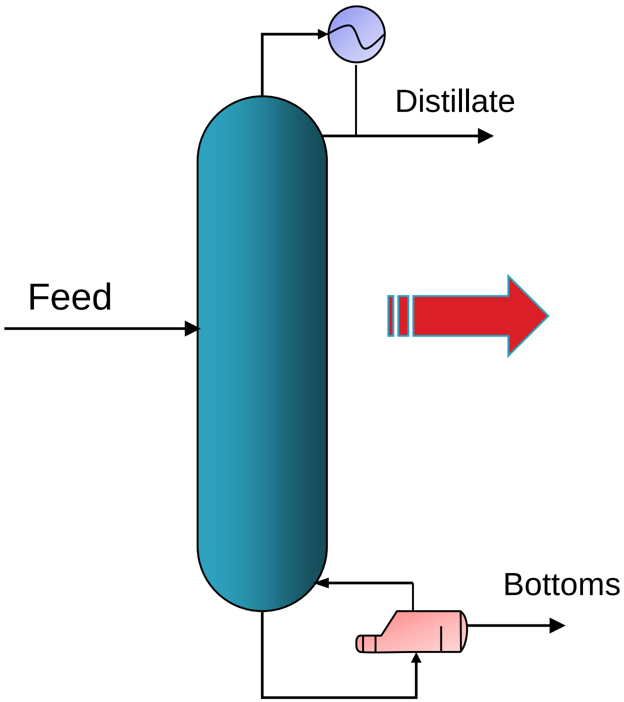
$$* \ \mathrm{Re}boiler$$

$$\left.\begin{aligned}
& L^i_{n+1} - B_i - V^i_n = 0 \\
& B_i = BOT\, x^i_n \\
& L^i_n = LIQ_n\, x^i_n \\
& V^i_n = VAP_n\, y^i_n \\
& \sum_{i\in C}\left(L^i_{n+1}\, hL^i_{n+1} - V^i_n\, hV^i_n - B_i\, hB_i\right) = QR \\
& hL^i_n = f\left(T^L_n\right) \\
& hV^i_n = f\left(T^V_n\right) \\
& hB_i = f\left(T^L_n\right) \\
& \sum_{i\in C} x^i_n = 1 \qquad \sum_{i\in C} y^i_n = 1 \\
& f^L_{i,n} = f\left(T^L_n, P_n, x_n\right) \\
& f^V_{i,n} = f\left(T^V_n, P_n, y_n\right) \\
& f^L_{i,n} = f^V_{i,n} \\
& T^L_n = T^V_n \\
& STG_n = 1
\end{aligned}\right\} n \in NRT$$

Yeomans & Grossmann (2000)

Economic Optimization of Distillation Columns

Conventional Distillation Column



Conventional Distillation Column

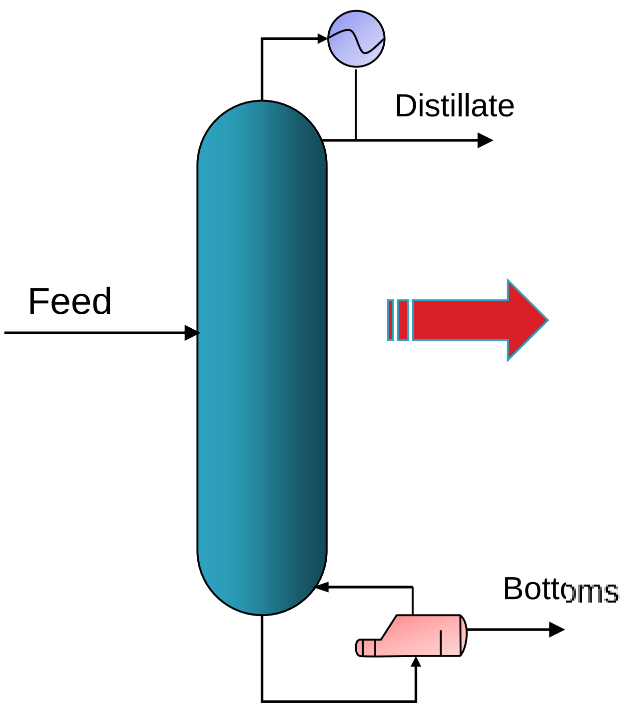
MESH equations for the intermediate trays excluding the equations related to the VLS

$$
\left.
\begin{aligned}
& L^i_{n+1} + V^i_{n-1} - L^i_n - V^i_n = 0 \\
& LIQ_n = \sum_{i \in C} L^i_n \\
& VAP_n = \sum_{i \in C} V^i_n \\
& \sum_{i \in C} \begin{pmatrix} L^i_{n+1} hL^i_{n+1} + V^i_{n-1} hV^i_{n-1} \\ -L^i_n hL^i_n - V^i_n hV^i_n \end{pmatrix} = 0 \\
& hL^i_n = f\left(T^L_n\right) \\
& hV^i_n = f\left(T^V_n\right) \\
& \sum_{i \in C} x^i_n = 1 \qquad \sum_{i \in C} y^i_n = 1
\end{aligned}
\right\} n \in TM
$$

Yeomans & Grossmann (2000)

Economic Optimization of Distillation Columns

Conventional Distillation Column



Conventional Distillation Column

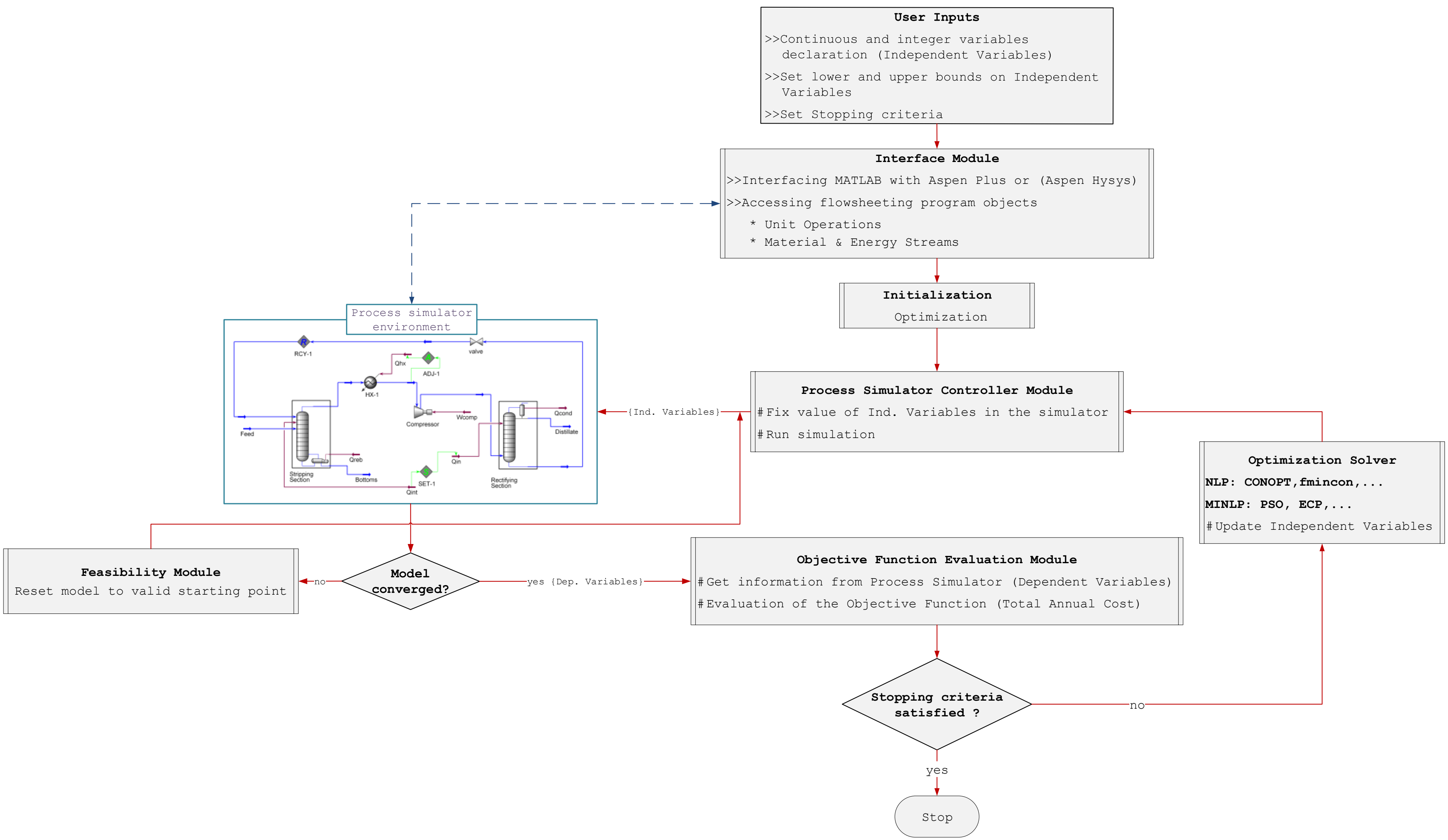
Constraints associated with the discret choice of enforcing VLE in a tray or not

$$\begin{bmatrix} Z_n \\ f^L_{i,n} = f\left(T^L_n, P_n, x_n\right) \\ f^V_{i,n} = f\left(T^V_n, P_n, y_n\right) \\ f^L_{i,n} = f^V_{i,n} \\ T^L_n = T^V_n \\ STG_n = 1 \\ L^i_n = LIQ_n x_i \\ V^i_n = VAP_n y^i_n \end{bmatrix} \underline{\vee} \begin{bmatrix} \neg Z_n \\ x^i_n = x^i_{n+1} \\ y^i_n = y^i_{n+1} \\ L^i_n = L^i_{n+1} \\ V^i_n = V^i_{n+1} \\ T^L_n = T^L_{n+1} \\ T^V_n = T^V_{n+1} \\ f^L_{i,n} = 0 \\ f^V_{i,n} = 0 \\ STG_n = 0 \end{bmatrix} \qquad n \in TM$$

Yeomans & Grossmann (2000)

Economic Optimization of Distillation Columns

Algorithm Flowchart



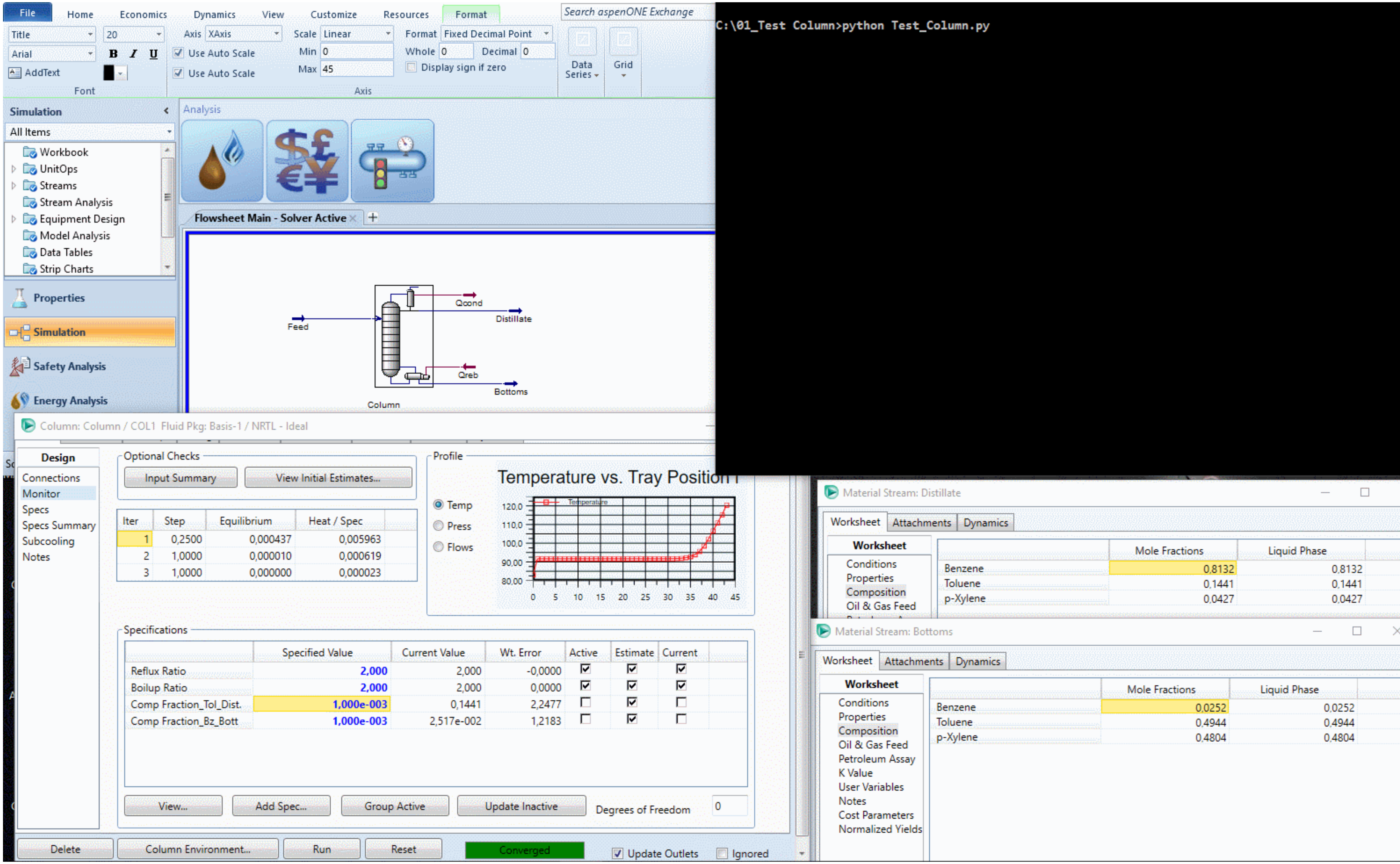
Economic Optimization of Distillation Columns

*/hyinterface.py

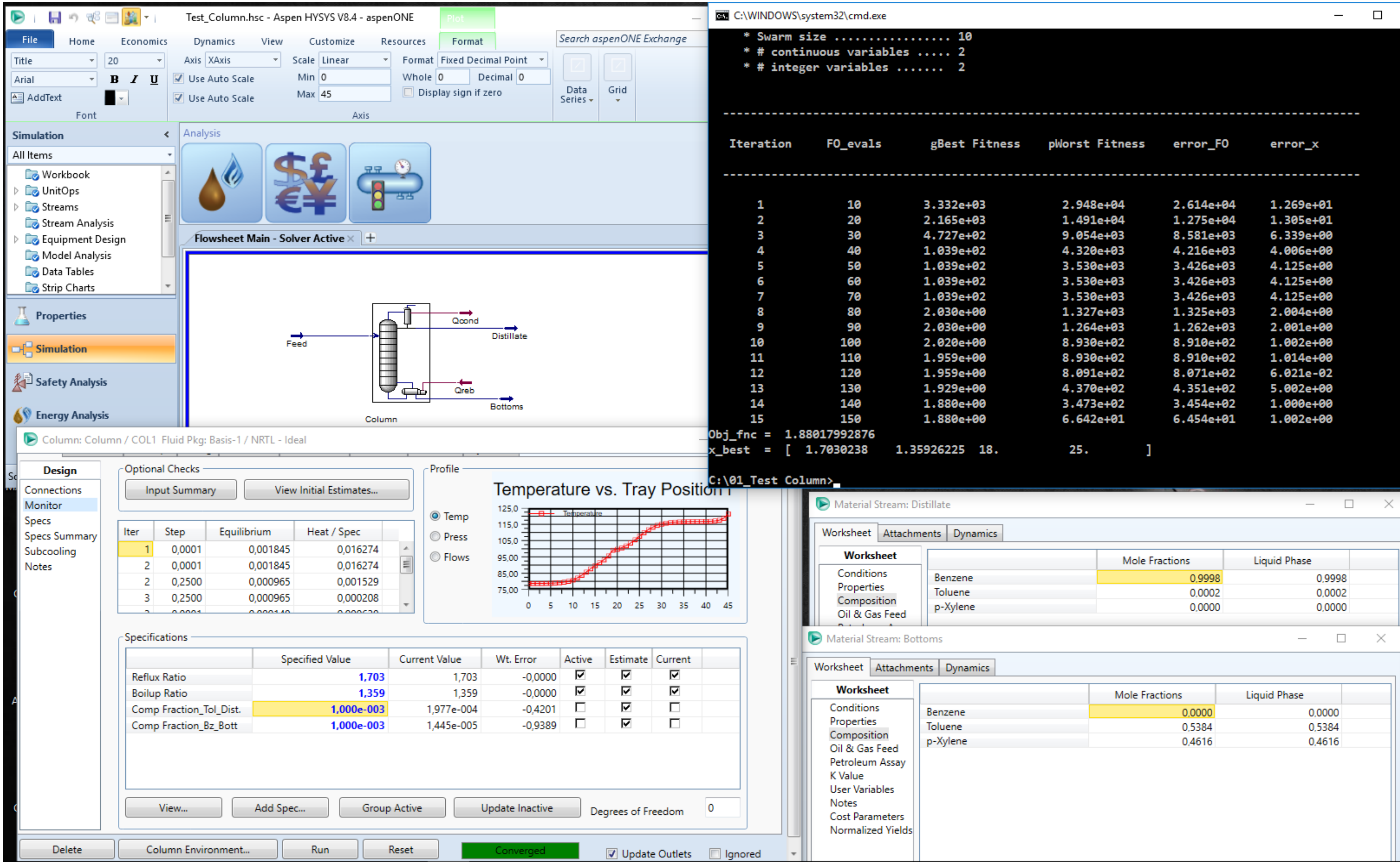
```
# -*- coding: utf-8 -*-

import os
import win32com.client as win32

"""
```

```
def hy_Dist_Col_Object(Problem, *varargin):

    hy_filename            = Problem.hy_filename
    hy_best_model_filename = Problem.hy_best_model_filename
    hy_visible             = Problem.hy_visible

# 01  Full path to Aspen Hysys File & Best Solution Hysys File

    hyFilePath = os.path.abspath(hy_filename)
    hy_beswt_solution_FilePath = os.path.abspath(hy_best_model_filename)

# 02 Initialize  Aspen Hysys application
    print(' # Connecting to the Aspen Hysys App ... ')
    HyApp = win32.Dispatch('HYSYS.Application')

# 03 Open Aspen Hysys File
    HyCase = HyApp.SimulationCases.Open(hyFilePath)

```

Economic Optimization of Distillation Columns

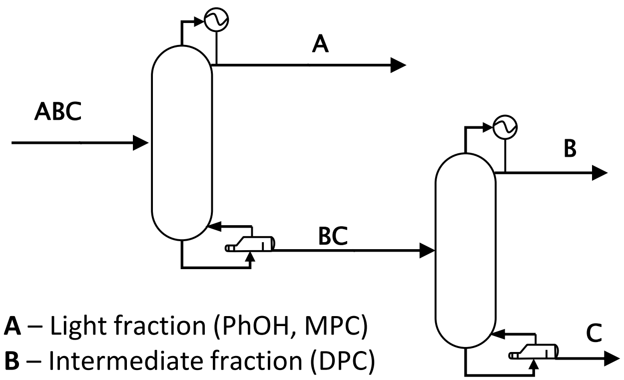


Economic Optimization of Distillation Columns



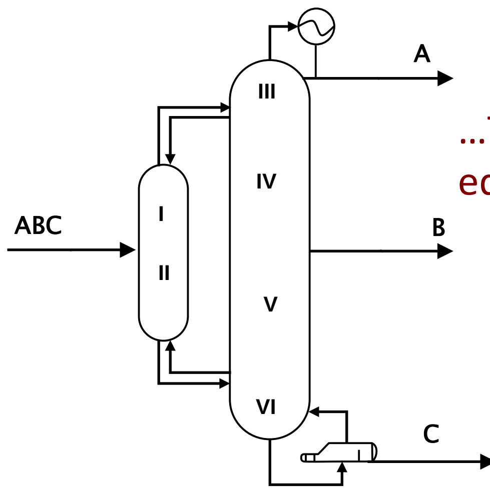
Economic Optimization of Divided Wall Columns

Divided Wall Column (DWC)



A – Light fraction (PhOH, MPC)
B – Intermediate fraction (DPC)
C – Heavy fraction (DPC, Bz, Cat)

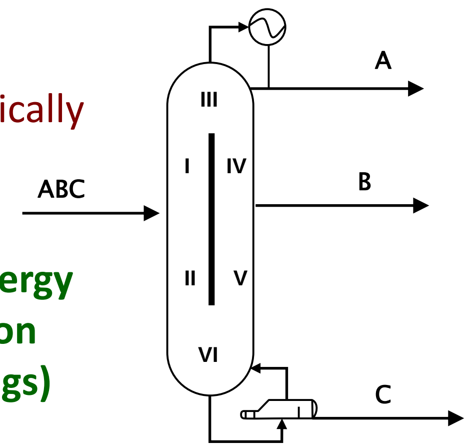
Direct separation sequence of three components (or fractions)



Petlyuk configuration

...Thermodynamically equivalent to...

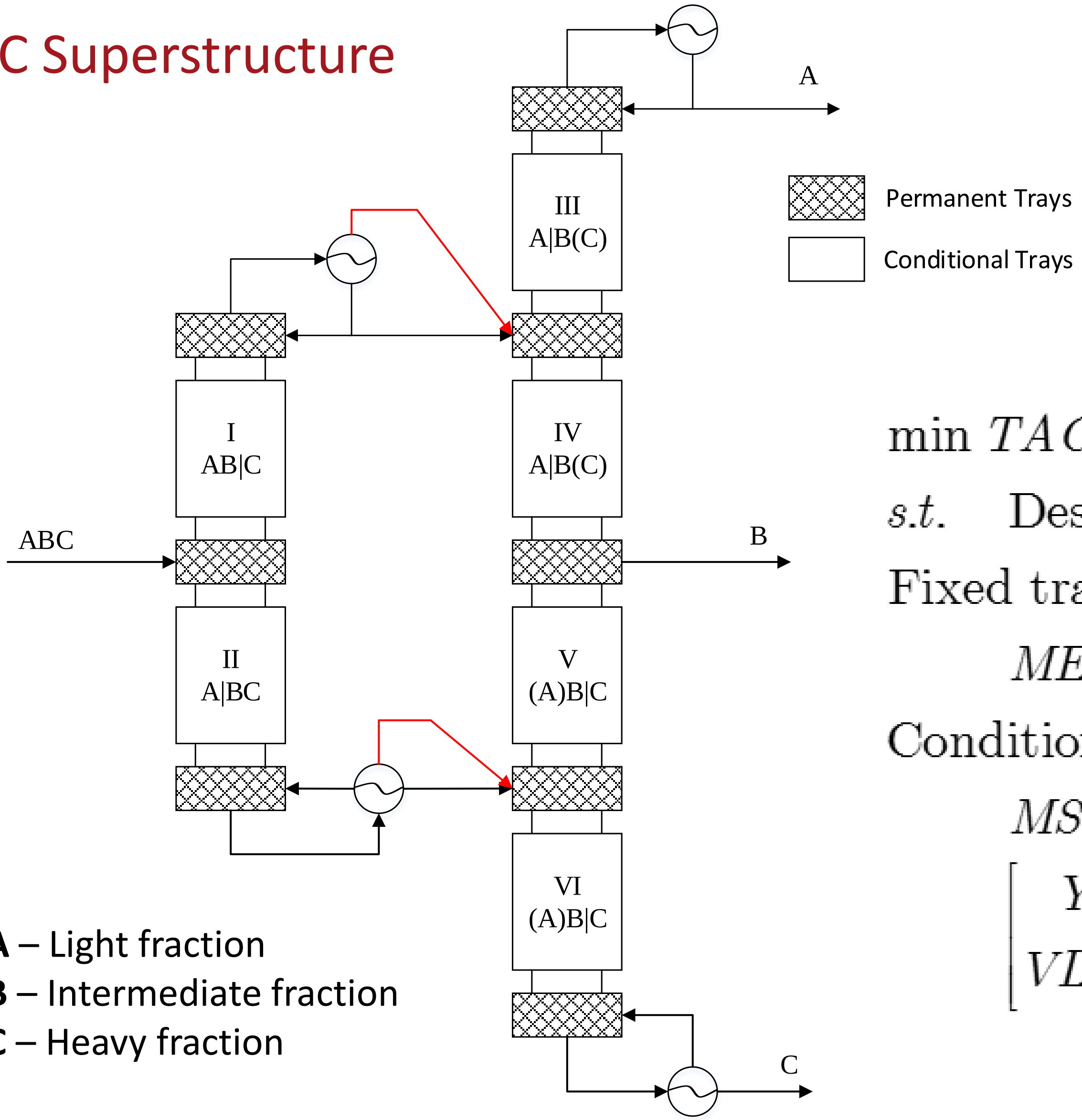
Minimum energy consumption (~20% savings)



Divided Wall Column (DWC)

Economic Optimization of Distillation Columns

DWC Superstructure



$$\min\ TAC = C_{op} + FC_{cap}$$

$$s.t.\quad \text{Design Constraints}$$

$$\text{Fixed trays}$$

$$MESH_{\ Equations}$$

$$\text{Conditional trays}$$

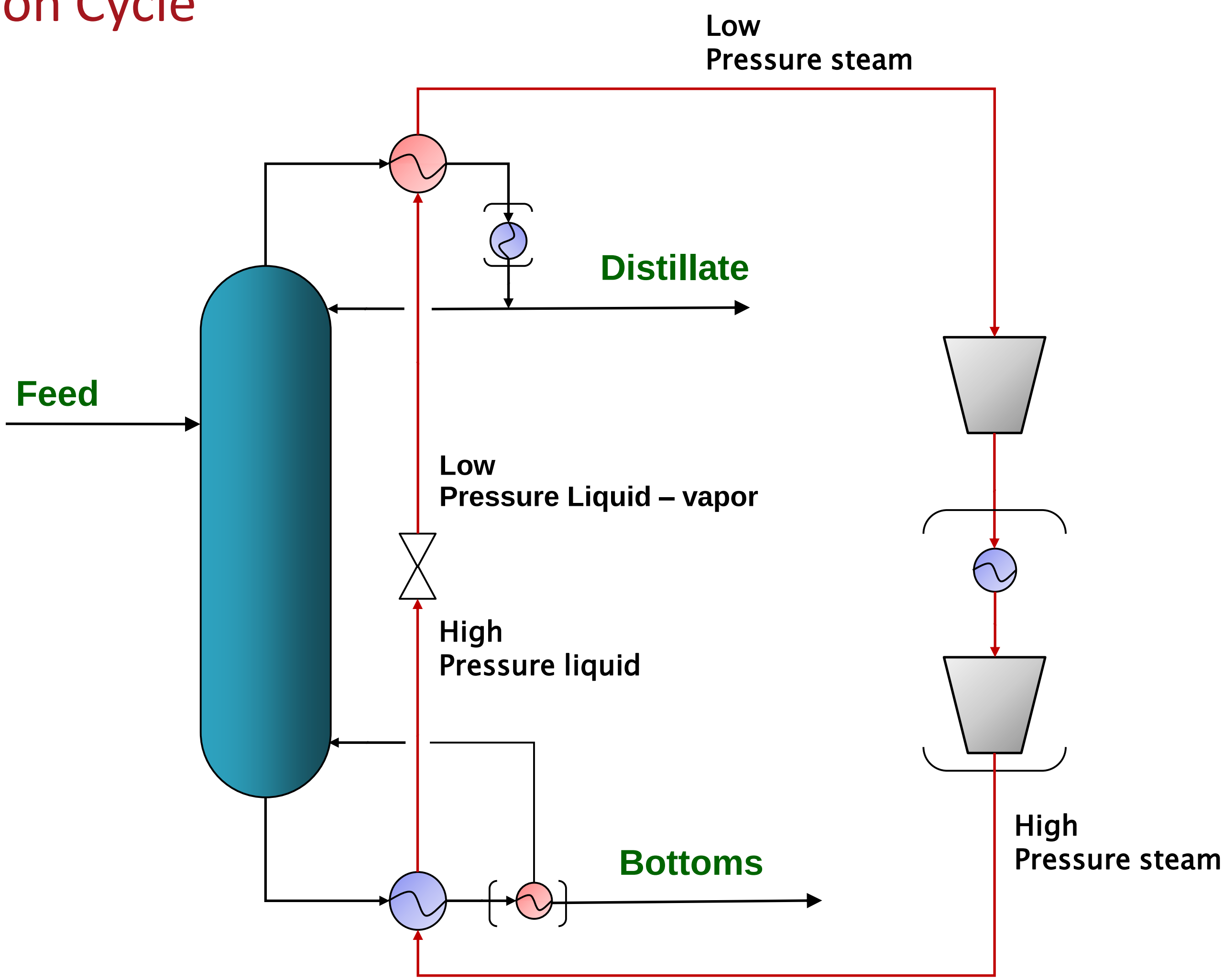
$$MSH_{\ Equations}$$

$$\begin{bmatrix} Y_n \\ VLE \end{bmatrix} \underline{\vee} \begin{bmatrix} \neg Y_n \\ Not VLE\ (\text{bypass}) \end{bmatrix}$$

A – Light fraction
B – Intermediate fraction
C – Heavy fraction

Vapor Compression Cycle

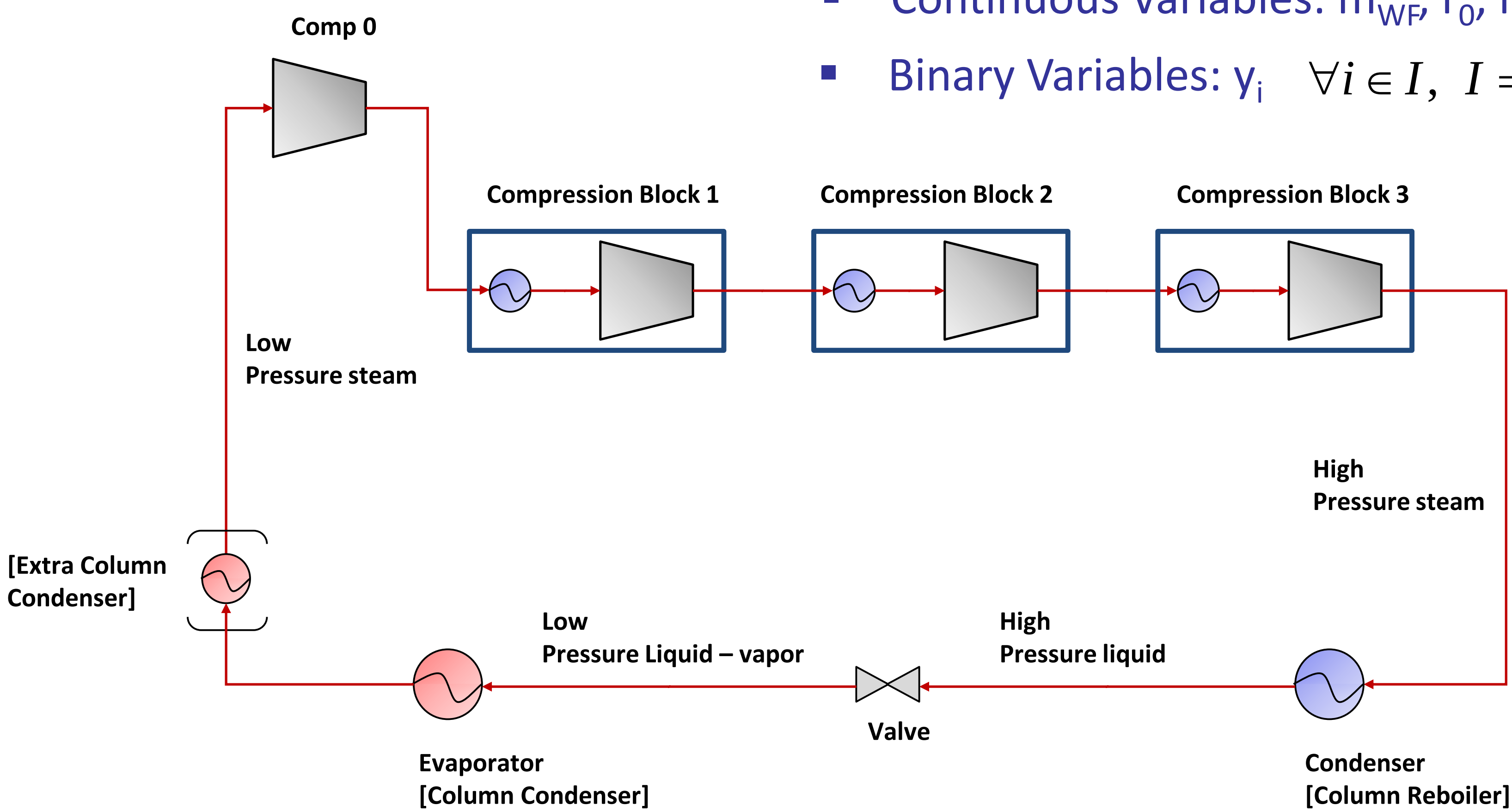
Conventional Distillation Column Equipped with a Closed Vapor Compression Cycle



Vapor Compression Cycle

Vapor Compression Cycle Superstructure

- Continuous Variables: m_{WF}, r_0, r_i, λ_i
- Binary Variables: y_i $\quad \forall i \in I, \; I = \{1,2,3\}$



CAChemE

Vapor Compression Cycle

$$\min_{m_{WF}, r_0, r_i, \lambda_i, Y_i} \quad TAC_{Comp\ 0} + \sum_{i}^{I} TAC_{CompBlock\ i}$$

$$s.t. \quad \left[W_{Comp\ 0}, W_{Comp\ i}, Q_{Cooler\ i}, Q_{VRC\ Condenser}, Q_{VRC\ Evaporator} \right] = f_{VRC}\left(m_{WF}, r_0, r_i, \lambda_i\right)$$

$$Q_{\text{Reb}} := g_{1\ VRC}\left(m_{WF}, r_0, r_i, \lambda_i\right) \geq \underline{Q_{Reb}}$$

$$\text{T}_{\text{Reb}} := g_{2\ VRC}\left(m_{WF}, r_0, r_i, \lambda_i\right) \geq \underline{T_{Reb}}$$

$$\text{T}_{CompBlock\ I} := g_{3\ VRC}\left(m_{WF}, r_0, r_i, \lambda_i\right) \leq \overline{T_{max}}$$

$$\begin{bmatrix} Yi \\ r^{lo} \leq r_i \leq r^{up} \\ \lambda^{lo} \leq \lambda_i \leq \lambda^{up} \end{bmatrix} \underline{\vee} \begin{bmatrix} \neg Yi \\ r_i = 1 \\ \lambda_i = 0 \end{bmatrix} \qquad \forall i \in I$$

$$Y_i \Rightarrow Y_{i-1} \qquad \forall i \in I > 1$$

$$Y_i \in \{True, False\}$$

Generalized Disjunctive Programming (GDP) Formulation

- Continuous Variables: m_{WF}, r_0, r_i, λ_i
- Binary Variables: y_i $\quad \forall i \in I, \ I = \{1,2,3\}$

Conclusions

- **PSO Advantages/disadvantages over GA (as stochastic algorithms)**
  - **Simpler** algorithm and concept
  - **Easily programmable and tuneable**[1]
  - **Faster in convergence**[2]: less number of function of evaluations thus computationally more efficient (inertial term and particle's memory benefits)
  - Better for continuous variables (or parameters), whereas GA excels in combinatorial problems

- **Take-home message**
  - PSO can provide 'good enough' (optimal) solutions for a minimum effort and knowledge of your problem*

(*)Unfortunately, it won't scale for problems where the computational cost of the objective function is too high

[1] A locally convergent rotationally invariant particle swarm optimization algorithm (Bonyadi et al. 2014)

[2] A Comparison of Particle Swarm Optimization and the Genetic Algorithm (R. Hassan et al. 2005)

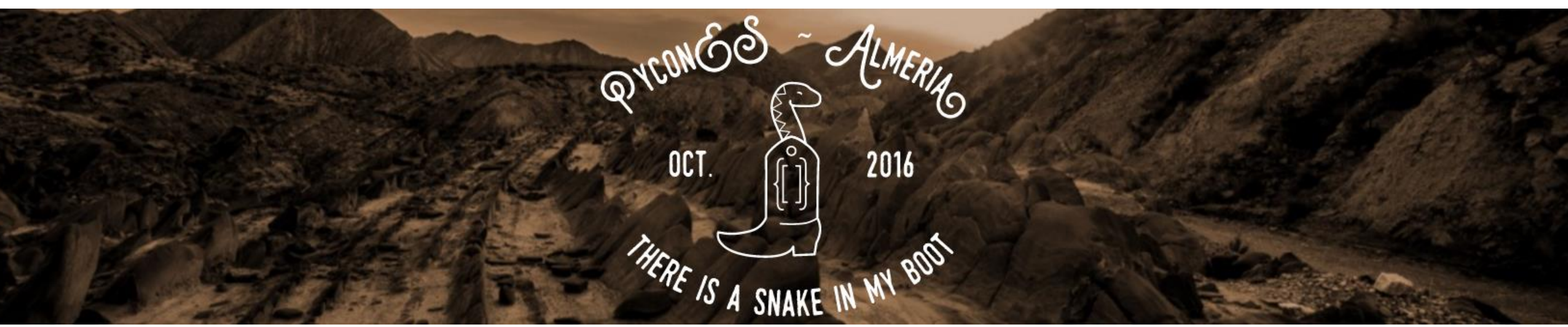
CAChemE

[>>COnCEPT]
Computer Optimization in Chemical Engineering Processes and Technologies

Simulation-Based Optimization using the Particle Swarm Optimization Algorithm

(Aprendiendo magia negra con Python, optimización estocástica y simuladores)

Juan Javaloyes & Francisco Navarro



https://github.com/CAChemE/stochastic-optimization

October 2016