Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale, Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

Comparative and Statistical Study on Aspen Plus Interfaces Used for Stochastic Optimization

Josué J. Herrera Velázquez[a,c], Erik L. Piñón Hernández[a], Luis A. Vega Vega[a], Dana E. Carrillo Espinoza[a], J. Rafael Alcántara Avila[b], and Julián Cabrera Ruiz[a]*

[a] Universidad de Guanajuato, Departamento de Ingeniería Química, Guanajuato, 36050, Mexico
[b] Pontificia Universidad Católica del Perú, Departamento de Ingeniería, Lima, 15074, Peru
[c] Instituto Tecnológico Superior de Guanajuato, Departamento de Ingeniería en Industrias Alimentarias, Guanajuato, 36262, Mexico
* Corresponding Author: j.cabreraruiz@ugto.mx.

ABSTRACT

New research on complex intensified distillation schemes has popularized the use of several commercial process simulation software. The interfaces between process simulation and optimization-oriented software have allowed the use of rigorous and robust models. This type of optimization is mentioned in the literature as "*Black Box Optimization*", since successive evaluations exploits the information from the simulator without altering the model that represents the given process. Among process simulation software, Aspen Plus® has become popular due to their rigorous calculations, model customization, and results reliability. This work proposes a comparative study for Aspen Plus software and Microsoft Excel VBA®, Python® and MATLAB® interfaces. Five distillation schemes were analyzed: conventional column, reactive column, extractive column, column with side rectifier and a Petlyuk column. The optimization of the TAC (Total Annual Cost) was carried out by a modified Simulated Annealing Algorithm (m-SAA). The evaluation criteria are the time per iteration (TI) and TAC values. The results indicate that the best option to carry out the optimization was by using the VBA interface, however the one carried out with Python did not differ radically (12%).

Keywords: Aspen Plus, Stochastic Optimization, Process Optimization, Python, MATLAB, Visual Basic

INTRODUCTION

The optimization of complex processes is challenging because optimization problems tend to be nonconvex, discontinuous and nonlinear, and consequently, there is not certainty to the global optimum solution [1]. The absence of precise derivatives in their models and the convergence failures that can generate discontinuities in the optimization "surfaces" of the functions make the task of finding the global optimum very difficult.

The optimization of processes using stochastic algorithms has become very relevant because they can be implemented in a relatively simple way by making an interface with commercial process simulators that are responsible for solving mathematical problems by modeling the involved units in the process [2]. Commercial process simulators act as a powerful calculator for highly nonlinear mathematical models without losing model rigorously. This type of software is also known as black box modeling software [3] because the user typically does not come into direct contact with the model equations that describe the process it simulates.

Optimization algorithms began to be developed to directly use the interface between Aspen Plus-VBA using Excel as an IDE [4-5]. MATLAB developed its direct interface with Aspen Plus becoming one of the most popular options to solve chemical process optimization problems [6]. Most recently, the Aspen Plus–Python connection has been studied showing good results in its implementation [7] making use of stochastic optimization libraries developed by the scientific community [8]. The literature is extensive for all the cases and has been applied for single and multi-objective optimization.

There are studies comparing interface methodologies for Aspen Hysys® software [9], but not for Aspen Plus. Ponce-Rocha et al. [10] compare the computing time between iterations and the best solution between the Aspen Plus-MATLAB and Aspen Plus-Python interfaces for a case study of a bottom distillation train, finding that the interface with Python has the shortest time and the best solution compared to the one made with MATLAB.

METHODOLOGY

Optimization framework

In the black box optimization, the writing/reading results are from the contribution of 1) Process simulation software, 2) Middleware protocol, 3) Wrapper protocol and the 4) Platform with the optimization algorithm [11]. The middleware protocol allows the automation of the process simulator and the transfer of information in both directions, for Windows it is ActiveX automation. The Wrapper protocol works to interpret the information transferred by the previous protocol and make it useful for both parties. Accordingly, to Muñoz López et al. [11] interface can be divided into four other parts:

- **Simulation:** The simulation is carried out with the desired characteristics, for a set of variables where its convergence is ensured. The necessary techniques are applied to ensure that the simulation is not corrupted when beginning the optimization process. Aspen Plus V12 software was used.
- **Control interface:** ActiveX (COM) framework is used to build this interface. It allows you to have the API (Application Programming Interface) to manipulate applications that run on Windows.
- **Information interface:** Transfer the required data. The middleware is a set of ASCII type files that contain variable information (name, value, scale factor, specification, units, etc.) and is generated only when the simulation converges. This information makes the Aspen Plus–Optimizer routine possible. Middleware is considered the bottleneck of computational performance.
- **Optimization:** The optimization is carried out according to the optimization algorithm. For Python, Spyder IDE 5.4.1 was used. In the case of MATLAB, R2022a was used. For VBA, Excel Microsoft® 365 2022 version was used.

This overall process is shown in Figure 1. Table 1 presents some general commands to perform any activity that requires connection with any of the programming languages discussed. Step (1) refers to the line of code to open the Aspen Plus document.

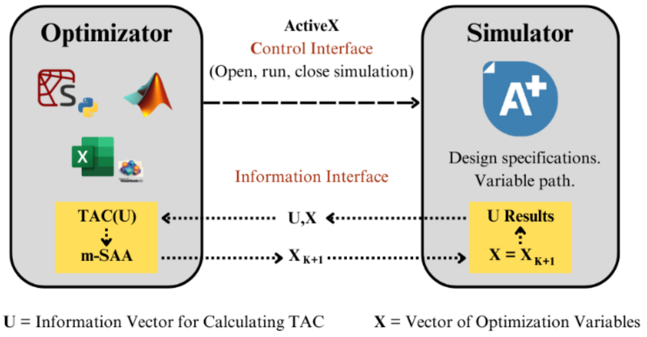

Figure 1. Overview of the interface implemented between Aspen Plus and MATLAB/Excel (VBA)/Spyder (Python).

It is possible to work with the *.bkp* or *.apw* extension file. In this work the *.apw* extension file was used. This form of programming is designed so that both, the code file and the simulation are at the same address, so that simply entering the name of the simulation is enough. In the case of MATLAB, it is necessary to define the version of Aspen Plus that will be used. In the case of VBA, the "On Error Resume Next" code is used, and this is to close the pop-up screens that are typical of the language each time a step is carried out. For Python, the "import os" library is used to identify the location address where the code file is saved, while the “import win32com.client” library is the one that allows the Aspen Plus software to operate.

Step (2) and (3) are commands for filling information in Aspen Plus and calling simulation results respectively (In the case of Python it is necessary to place double \ to avoid errors in communication). `y` is a number, and `X` is a variable that will be used for subsequent operations. Step (4) is the command to restart the simulation. This command is not recommended if working with complex simulations difficult to converge. In the cases of MATLAB and VBA the use of this command affects the time per iteration to almost double it. Step (5) is the command to run the simulation and step (6) is to close it.

It must be considered that in an optimization process steps (1) and (6) are only performed once per optimization, that is, the simulation is opened, the optimization process is executed for n iterations and once this finished, the simulation is closed. This is because opening and closing the simulation as an environment takes a long time and this could affect the demand for RAM memory resources of the computer equipment. In the case of Python, the iterative process of this type of optimization causes RAM to accumulate, so it is necessary to reset the connection every certain number of iterations to avoid exponential growth in the time per iteration. This work recommends resetting the connection every 125 iterations.

Table 1: Starting and execution of the Aspen Plus application from each platform and its main commands for filling and reading information.

Platform	Step	Code
MATLAB	(1)	`Aspen = actxserver('Apwn.Document.38.0');` `[stat,mess]=fileattrib;` `Aspen.invoke('InitFromArchive2',[mess.Name '\Simulation_Name.apw']);`
	(2)	`Aspen.Application.Tree.FindNode("Simulation\Variable\Path").Value = y;`
	(3)	`X = Aspen.Application.Tree.FindNode("Simulation\Variable\Path").Value;`
	(4)	`Aspen.Reinit;`
	(5)	`Aspen.Engine.Run2();`
	(6)	`Aspen.Close;`
Excel VBA	(1)	`Set Aspen = GetObject(ThisWorkbook.Path & "\Simulation_Name.apw")` `On Error Resume Next`
	(2)	`Aspen.Tree.FindNode("Simulation\Variable\Path").Value = y`
	(3)	`X = Aspen.Tree.FindNode("Simulation\Variable\Path").Value`
	(4)	`Aspen.Reinit`
	(5)	`Aspen.Engine.Run`
	(6)	`Aspen.Close`
Python	(1)	`import os` `import win32com.client as win32` `Aspen = win32.Dispatch('Apwn.Document')` `Aspen.InitFromArchive2(os.path.abspath('Simulation_Name.apw'))`
	(2)	`Aspen.Tree.FindNode("Simulation\\Variable\\Path").Value = y`
	(3)	`X = Aspen.Tree.FindNode("Simulation\\Variable\\Path").Value`
	(4)	`Aspen.Reinit()`
	(5)	`Aspen.Engine.Run2()`
	(6)	`Aspen.close()`

An error called "ExceptionDump" may occur, which appears when the call is fast enough to prevent the completion of the protocols described above and corrupt the optimization process. This error displays the legend:

```
"An unexpected error has occurred. An error log has been generated in: C:\file\location\aspenplus_ExceptionDump_date.dmp".
```

The authors recommend using a delay time between 0.05 and 0.1 seconds that gives the platforms time to finish the calling protocols. These are shown in Table 2. `delay` is a number that refers to the delay time that will be carried out between each iteration. The line of code must be written before line (5) mentioned in Table 1.

In the case of the Python platform, it is necessary to call "from time import sleep" to be able to execute the action. In VBA it is not necessary to place a delay as it is rare to see this error in your process.

Table 2: Recommended lines of code to avoid the connection error between the platform and Aspen Plus.

Platform	Code
MATLAB	`pause(delay);`
Python	`From time import sleep` `sleep(delay)`

The optimization was performed using the m-SAA algorithm proposed by Cabrera Ruiz et al. [2] which makes use of dynamic bounds and normalization of variables to improve the search for the optimum and find it faster. This algorithm was programmed in the same way for all platforms without the reinit function (line (4) in Table 1), the only difference between programming languages are their own random number functions. Additional parameters of the optimization algorithm are presented in Herrera Velázquez et al. [12]. The normalization of the variables is done as shown in Eq. 1.

$$VN_i = \left(B_{upper} - V_i\right)/\left(B_{upper} - B_{lower}\right) \quad (1)$$

where VN_i is the value of the normalized variable V_i, B_{upper} and B_{lower} are the upper and the lower bound of variable i respectively. The stopping criterion was defined the same way for all schemes and platforms: 6 reannealing's of 350 iterations each, giving a total of 2,100 iterations, ensuring a study of at least 200 iterations for each variable. The computer used for all optimizations has an Intel(R) Xeon(R) W-3235 CPU at 3.30 GHz with 48 GB of RAM with the Windows 10 operating system (which is the most stable version for all platforms).

Case Studies

The five case studies are shown in Figure 2. N_T

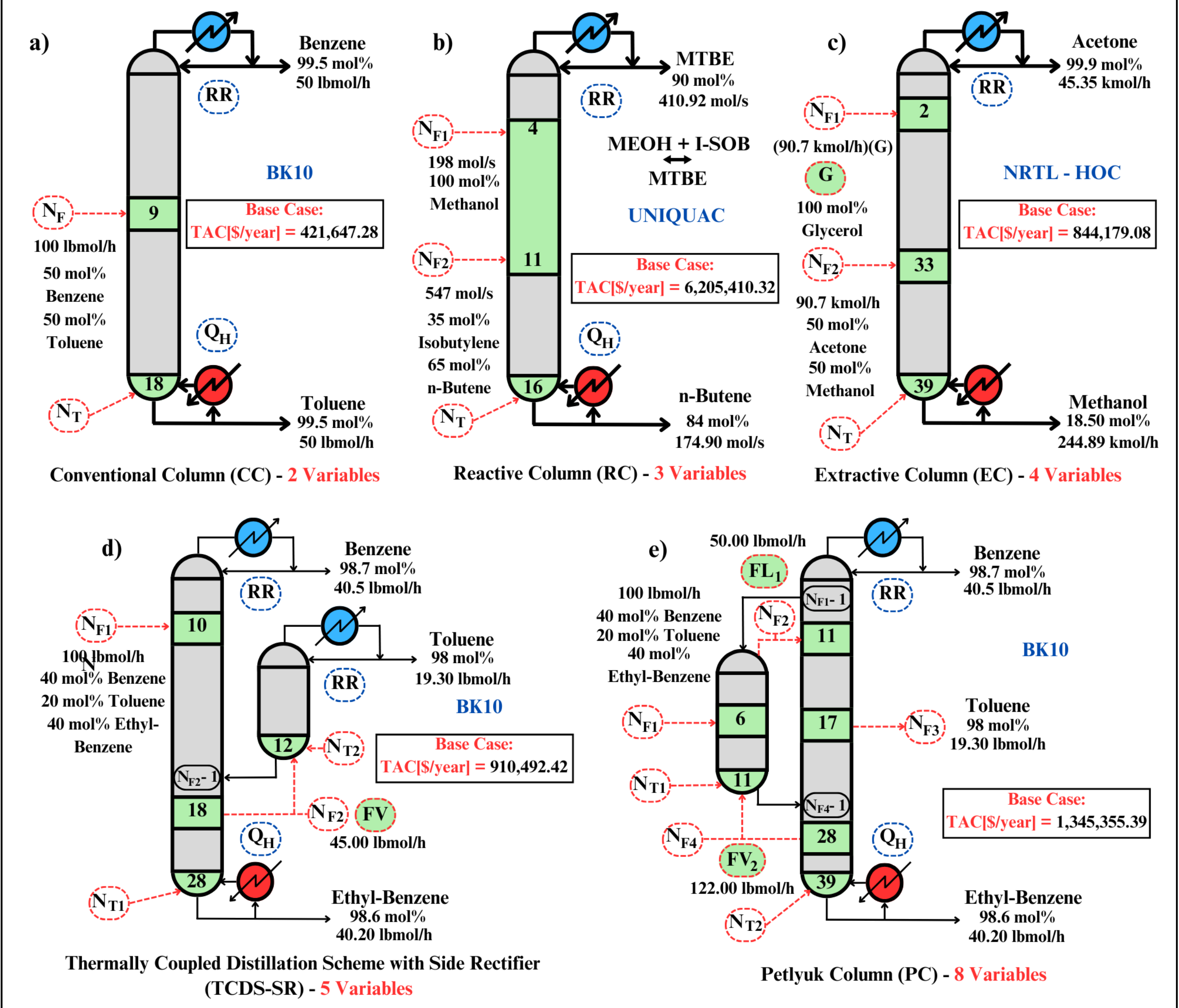

Figure 2: Representation of the proposed schemes to carry out the tests and their proposed optimization variables and their thermodynamic model used: a) Conventional Column, b) Reactive Column [13] and c) Extractive Column [14], d) Thermally Coupled Distillation Scheme with Side Rectifier and e) Petlyuk Column.

refers to the total number of stages, N_F is the feed stage, G is the ratio of entrainer to feed, FV is the vapor flow and FL is the liquid flow, both in kmol/h. For the CC, RC and PC schemes (Figure 2a, 2b and 2e) the purity and recovery constraints are satisfied by adjusting the reflux ratio (RR) and the reboiler load (Q_H) through internal design specifications in the simulator. For the PC scheme it is done on the largest column. In the case of the EC and $TCDS-SR$ schemes (Figure 2c and 2d), the constraints are adjusted with the RR on the respective domes, and the Q_H for the $TCDS-SR$ scheme. The optimization limits for each scheme are presented in Table 3.

Table 3: Optimization bounds [lower, upper].

	Variables
CC	$N_T = [10, 30]; N_F = [2,\ N_{T1} - 1]$
RC	$N_{T1} = [10, 30]; N_{F1} = [2,\ N_{T1} - 2];$ $N_{F2} = [N_{F1} + 1,\ N_{T1} - 1]$
EC	$N_{T1} = [35, 45]; N_{F1} = [2,\ N_{T1} - 2];$ $N_{F2} = [3,\ N_{T1} - 1]; G = [1, 2.5]$
$TCDS - SR$	$N_{T1} = [20, 40]; N_{T2} = [5, 15];\ N_{F1} = [2,\ N_{T1} - 2];$ $N_{F2} = [3,\ N_{T1} - 1]; FV = [40, 100]$
PC	$N_{T1} = [5, 15]; N_{T2} = [25, 45];$ $N_{F1} = [2,\ N_{T1} - 2]; N_{F2} = [3,\ N_{T1} - 1];$ $N_{F3} = [N_{F2} + 1,\ N_{T2} - 3]; N_{F4} = [N_{F3} + 2,\ N_{T2} - 1];$ $FL_1 = [20, 110]; FV_2 = [20, 150]$

Mathematical Treatment

The TAC is calculated as show in Eq. 2.

$$TAC\left[\frac{\$}{Year}\right] = C_{Vapor} + C_{Cooling} + \frac{\left(\frac{CEPCI_{2022}}{CEPCI_{2001}}\right)(C_{Shell} + C_{Plates} + C_{Reboiler} + C_{Condenser})}{Recovery\ time\ (Years)} \quad (2)$$

$$s.t. \quad Conv = 0$$

The TAC equation is calculated using the Guthrie method [15] where C is the cost of each module. 5 years of recovery time was used. $Conv$ is a convergence constraint, $Conv = 1$ when the simulation does not converge in Aspen Plus, in this case, the objective functions have a penalty value that is 3 times the value of each function evaluated for the base design, these values are given in Figure 3. $Conv = 0$ if the simulation converges and the functions are calculated with the described model. TI is taken from the time of the complete optimization process (without considering the (1) and (6) steps of Table 1, and in Python considering the time of the restarts every 125 iterations) divided by the total number of iterations.

The optimization was repeated 10 times for each case study. To find anomalous data, hypothesis testing (Grubbs Test) was used [16], which is denoted by Eq. 3 for this work. The test was applied to the TAC and TI. The test was done to achieve 90% confidence (i.e., 2.29).

$$\begin{aligned} H_O&: \left[\frac{|TAC_i - \overline{TAC}|}{S_{TAC}} \leq 2.29, \frac{|TI_i - \overline{TI}|}{S_{TI}} \leq 2.29\right] \\ H_A&: \left[\frac{|TAC_i - \overline{TAC}|}{S_{TAC}} > 2.29, \frac{|TI_i - \overline{TI}|}{S_{TI}} > 2.29\right] \end{aligned} \quad (3)$$

where TAC_i is the TAC of optimization i, $\overline{TAC}$ and S_{TAC} are the average and standard deviation. The same procedure was followed for TI. If H_O is true for both, the TAC and TI are accepted and if H_A is true these results are rejected.

RESULTS

A delay of 0.05 second is considered to avoid communication errors, this time is considered in the results shown. Table 4 summarizes the results of 10 independent optimizations for each of the platforms and for each of the proposed schemes, totaling 150 total optimizations (30 for each scheme). The results show the average value and standard deviation calculated after applying the Grubbs test. Figure 3 shows the frequency of appearance of each platform as the best solution in each of the evaluated aspects (shown in bold in Table 4).

Of 130 optimizations performed, 6 were eliminated via Grubbs test (4.61%), of which 4 are from the MATLAB platform (3 by TI and 1 by TAC), 1 from the Python platform and 1 from VBA (both by TAC).

Table 4: Summary of results for each scheme and each platform. (M = MATLAB, V = VBA, P = Python).

CASE		TAC [\$/$Year$]	TI [s]
CC	M	284,384 ± 1,181	0.8783 ± 0.0066
	V	290,131 ± 1,181	**0.7445 ± 0.0246**
	P	**283,154 ± 204**	0.8678 ± 0.0085
RC	M	4,425,792 ± 143,638	0.7678 ± 0.0246
	V	4,323,736 ± 72,576	**0.6545 ± 0.0455**
	P	**4,267,495 ± 48,562**	0.8433 ± 0.0438
EC	M	519,843 ± 14,398	0.9600 ± 0.0078
	V	**478,751 ± 7,185**	**0.8251 ± 0.0199**
	P	498,593 ± 28,551	0.9419 ± 0.0053
TCDS-SR	M	436,695 ± 10,651	1.9364 ± 0.1753
	V	470,608 ± 1,515	**1.2959 ± 0.0611**
	P	**430,604 ± 5,634**	1.6973 ± 0.0306
PC	M	605,647 ± 91,986	3.3907 ± 0.8229
	V	**604,607 ± 74,327**	2.3790 ± 0.0971
	P	615,599 ± 75,098	**2.2708 ± 0.1907**

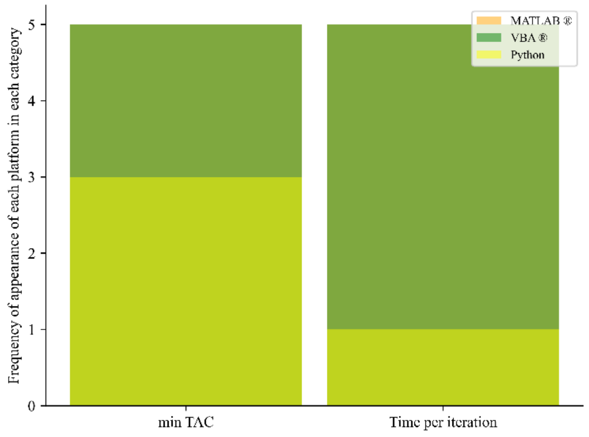

Figure 3. Frequency of appearance of platforms in each category.

CONCLUSIONS

The results indicate that Python is the platform that shows the best numerical results from the TAC, especially in simple and complex schemes. The platform that shows the best TI is VBA, although in complex schemes it shows a slight disadvantage compared to Python. Although the TAC results show a certain deviation between platforms, the TI vary a little (12%) between them. This study does not seek to define which platform is better or worse than another, it only seeks to find the areas of opportunity and improvement between them, as well as to define which is the best for the researcher who uses it. According to the results, VBA is the best option; however, MATLAB and

Python may be better options for numerical results (due to the way random numbers are generated on each platform). If a more customizable and open-source environment is needed, Python is the best option, and if more specialized mathematical tools are needed, MATLAB may be the best option.

REFERENCES

1. Javaloyes Antón J, Kronqvist J, Caballero JA. Simulation-based optimization of distillation processes using an extended cutting plane algorithm. *Comp Chem Eng* 159:107655 (2022) https://doi.org/https://doi.org/10.1016/j.compchemeng.2021.107655
2. Cabrera Ruiz J, Hasebe S, Alcántara Avila JR. Improvement of the Optimal Design Procedure Using Randomized Algorithm and Process Simulators. *MATEC Web Conf* 333:06004 (2021) https://doi.org/10.1051/matecconf/202133306004
3. Penteado AT, Esche E, Weigert J, Repke JU. A framework for stochastic and surrogate-assisted optimization using sequential modular process simulators. *Comp Aid Chem Eng* 48:1903-1908 (2020) https://doi.org/https://doi.org/10.1016/B978-0-12-823377-1.50318-9
4. Sharma S, Rangaiah GP, Cheah KS. Multi-objective optimization using MS Excel with an application to design of a falling-film evaporator system. *Food Bioprod Process* 90:123-134 (2012) https://doi.org/https://doi.org/10.1016/j.fbp.2011.02.005
5. Sánchez Ramírez E, Huerta Rosas B, Quiroz Ramírez JJ, Suárez Toriello VA, Contreras Zarazua G, Segovia Hernández JG. Optimization-based framework for modeling and kinetic parameter estimation. *Chem Eng Res Des* 184:647-660 (2022) https://doi.org/https://doi.org/10.1016/j.cherd.2022.08.040
6. Zheng Y, Wang Z, Pan H, Ling H. Initial design and multi-objeective optimization of four-product dividing wall column. *Sep Purif Technol* 309:122961 (2023) https://doi.org/https://doi.org/10.1016/j.seppur.2022.122961
7. Sánchez Gómez JA, Cabrera Ruiz J, Hernández S. Design and optimization of an intensified process to produce acrylic acid as added product value from glycerol generated in the biodiesel production. *Chem Eng Res Des* 184:543-553 (2022) https://doi.org/https://doi.org/10.1016/j.cherd.2022.08.040
8. Blank J, Deb K. Pymoo: Multi-objective optimization in Python. *IEEE Access* 8:89497-89509 (2020) https://doi.org/10.1109/ACCESS.2020.2990567
9. Santos Bartolome P, Van-Gerven. A comparative study on Aspen Hysys interconnection methodologies. *Comp Chem Eng* 162:107785 (2022) https://doi.org/10.1016/j.compchemeng.2022.107785
10. Ponce Rocha JD, Ramírez Márquez C, Morales Rodríguez R. Performance analysis of virtual platforms focused on multi-objective process optimization. *Comp Aid Chem Eng* 52:843-848 https://doi.org/10.1016/B978-0-443-15274-0.50135-9
11. Muñoz López CA, Telen D, Nimmegeers P, Cabianca L, Logist F, Van-Impe J. A process simulator interface for multiobjective optimization of chemical processes. *Comp Chem Eng* 109:119-137 (2017) https://doi.org/10.1016/j.compchemeng.2017.09.014
12. Herrera Velázquez JJ, Zavala Durán FM, Chávez Díaz LA, Cabrera Ruiz J, Alcántara Avila JR. Hybrid two-step optimization of internally heat-integrated distillation columns. *J Taiwan Inst Chem Eng* 130:103967 (2022) https://doi.org/10.1016/j.jtice.2021.06.061
13. Huang K, Nakaiwa M, Wang SJ, Tsutsumi A. Reactive distillation design with considerations of heats of reaction. *AIChE J* 52:2518-2534 (2006) https://doi.org/https://doi.org/10.1002/aic.10885
14. Sazonova AY, Raeva VM, Frolkova AK. Design of extractive distillation process with mixed entrainer. *Chem Papers* 70:594-601 (2015) https://doi.org/10.1515/chempap-2015-0247
15. Turton R, Bailie RC, Whiting WB, Shaeiwitz JA, Bhattacharyya D. Appendix A: Cost equations and curves for the CAPCOST program. In: Analysis, synthesis, and design of chemical processes. Ed: 4th. Prentice Hall (2012).
16. Grubbs, F. E. (1969). Procedures for Detecting Outlying Observations in Samples. Technometrics, 11(1), 1–21. https://doi.org/10.1080/00401706.1969.10490657