Vol. 20, No. 3(2021) Sim2338
Revista Mexicana de Ingeniería Química

Simulación y Control

Development of an open-source tool for equation-oriented process simulation in Python computational language

Desarrollo de una herramienta de código abierto para la simulación de procesos orientada a ecuaciones en lenguaje computacional Python

H.F.S. de Freitas[1,3]*, P.H. Soares[1,2], J.E. Olivo[1], C.M.G. Andrade[1]

[1] *State University of Maringá, Maringá, Colombo Av. 5790, 87020-900, Brazil.*
[2] *Federal Technological University of Paraná - Guarapuava Campus, Guarapuava, Vereador Jacir de França St., Industrial, 85053-510, Brazil.*
[3] *Federal Institute of Education, Science and Technology of Rio Grande do Norte - Nova Cruz Campus, Nova Cruz, José Rodrigues de Aquino Filho Av., 640, Alto de Santa Luzia, 59215-000, Brazil.*

Received: January 20, 2021; Accepted: August 3, 2021

Abstract

Nowadays, the process modeling and simulation exhibit notable importance, allowing the experimentation between different process designs and control configurations, as well as quality assurance and process optimization studies. In this sense, the equation-oriented approach, in which all the equations describing each sub-process are compiled into a single equation set, stands out as an advantageous approach. The current work focuses on the discussion of a developed tool, SLOTH, that is open-source and developed in Python computational language. The tool was applied in three different cases of study of industrial relevance, and the results obtained shown that the tool was able to solve the problems, obtaining values coherent with those presented in the literature, although some tasks required prohibitive computational times. Those aspects will be optimized in future versions of the tool, which is built collaboratively through the open-source nature of the project.

Keywords: modeling, simulation, equation-oriented simulation, Python, open-source.

Resumen

En la actualidad, el modelado y simulación de procesos tiene una importancia notable permitiendo la experimentación entre diferentes diseños de procesos y configuraciones de control así como estudios de aseguramiento de la calidad y optimización de procesos. En este sentido, el enfoque orientado a ecuaciones, en el que todas las ecuaciones que describen cada subproceso se compilan en un solo conjunto de ecuaciones, se destaca como un enfoque ventajoso. El trabajo actual se centra en la discusión de una herramienta desarrollada, SLOTH, que es de código abierto y desarrollada en lenguaje computacional Python. La herramienta se aplicó en tres casos diferentes de estudio de relevancia industrial, y los resultados obtenidos mostraron que la herramienta fue capaz de resolver los problemas, obteniendo valores coherentes con los presentados en la literatura, aunque algunas tareas requirieron tiempos computacionales prohibitivos.Estos aspectos se optimizarán en futuras versiones de la herramienta, que se construye de forma colaborativa a través de la naturaleza de código abierto del proyecto.

Palabras clave: modelado, simulación, simulación orientada a ecuaciones, Python, código abierto.

* *Corresponding author. E-mail*: `hanniel.freitas@ifrn.edu.br`

https://doi.org/10.24275/rmiq/Sim2338
ISSN:1665-2738, *issn-e*: 2395-8472

1 Introduction

In the task of modern of industrial processes design, several aspects need to be considered, which comprehend economical, energetic and environmental requirements. Those demands are translated in the necessity of process energy integration, life-cycle analysis, alternative production routes assessment, among other studies. Thus, the utilization of integral solutions to the process analysis such as modeling and simulation are vital, due to the economical restrictions imposed to real-file tests, as well as safety assurance and quick response time. The application of those alternatives represent valuable resources, especially for modern systems, characterized by the large-scale integration between different subprocesses, energy integration and demand for process intensification, among other challenging characteristics (Dowling and Biegler, 2015; Tian *et al*., 2018; Wu *et al*., 2016).

In general terms, the modeling of a process consists in the development of a representation its behavior – frequently in a quantitative manner – accordingly to phenomena intrinsic to its operation, under an appropriated set of considerations and simplifications that suffice for the level of detail desired (e.g.: negligible air resistance, homogeneous material properties, and so on). Under a technical-scientific aspect, the most useful models are those expressed through mathematical expressions. Among those, the dynamics models (which account for model variables changes over time) are notably valuable for providing the information about the system under consideration (Azar & Vaidyanathan, 2015; Ingham *et al*., 2008). Those mathematical models can be applied in different scenarios: research and development, including the kinetic parameter determination through laboratory or pilot scale, as well as design, optimization, control and scale-up studies; industrial design, in which can be included the study of equipment arrangement and sizing and the dynamical process performance assessment, material and energetic integration, as well as simulation of process start-up, stop and emergency scenarios; industrial plant operation, including operator training and in the systematic process control failures detection, and exploratory safety analysis, among others (Luyben, 1989; Ogunnaike & Ray, 1994). Other complex studies, such as process design with phase transition systems are also made manageable through an economical and safety point of view, through the utilization of process modeling and simulation (Hernández-Díaz *et al*., 2020; Inei-Shizukawa *et al*., 2020; Sánchez-Vargas & Valdés- Parada, 2021; Tsay *et al*., 2017).

In this sense, the simulation of process models can be classified in two categories: the sequential approach or simultaneous (SM), for which each equipment composing the process (or sub-processes) is sequentially solved; and the simultaneous (or equation-oriented, EO) approach, in which all the models describing the sub-processes are compiled in one large array of equations, and then this equation set is solved. Despite the superior numerical effort needed for the resolution of problems using the EO approach when compared to SM, its numerical stability and versatility represent vital aspects to be considered (Dowling & Biegler, 2015), especially when the process under study rely on the mixture between differential and algebraic systems, the so-called differential algebraic equation (DAE) problems. Despite the choice of the system solving approach, the procedural steps for process modeling can be identified in broad terms as: problem definition, in terms of the objectives of the study; assessment of the comprehension regarding the intrinsic phenomena of the system; mathematical formulation of the problem, and its ulterior resolution through simulation; validation for the developed model, in terms of known results and revaluation of the systems considerations if necessary (Ingham *et al*., 2008).

In the present work, the implementation of an open-source tool for equation-oriented simulation process called SLOTH developed using the Python computational language, will be presented. The tool was developed (and it still under development) aiming to be very clear in terms of its code, using state-of-the-art libraries of the Python computational language for the resolution of the problems. In this sense, the SLOTH software has an internal dimensional coherence mechanism, a system for symbolic process of the model(s) that describe one case of study which are declared as classes, that could derive from other models, exploiting the object-oriented nature of the computational language and a numerical mechanism for resolution of the equation system formed, as well as an interface for optimization studies and graph plotting. Although other tools were developed with analogous intent (Nikolić, 2016; Soares & Secchi, 2003; Westerberg *et al*., 1994), SLOTH aims to represent a collaboratively structured alternative for modeling, optimization and simulation studies using the equation-oriented approach. The main purpose of the software is to provide an open-source code for EO process simulation and optimization problems using robust computational libraries for its necessary internal procedures, such as symbolic manipulation, differential equation system integration and optimization studies. The case studies selected for evaluation of the tool performance are typical chemical engineering problems, namely the optimization of a fed-batch fermentation, parameter estimation for a polymerization kinetics and a vapor-liquid equilibrium calculation for a non-ideal binary mixture. It should also be mentioned that the Python computational language was used in the development of the tool, due to its high-level syntax and rich libraries ecosystem for numerical computation purposes, as mentioned in several works (Abel *et al.*, 2016; Capocchi *et al.*, 2011; Larsen *et al.*, 2017; Nikolić, 2016). The tool is available in the Github repository (`https://github.com/hfsf/sloth`).

1.1 Concepts employed in the tool development

1.1.1 General principles

In the present subsection of the work, the conceptual principles that guided the development of the tool will be presented. In this intent, some cardinal aspects for the software development were accounted, in accordance with those mentioned by other authors in related works (Nikolić, 2016; Soares & Secchi, 2003): the classification of the objects used in the process modeling through the software among different categories according to their function: variables (elements that represent unknowns for which the problem need to be solved), parameters (elements which value are specified in the models simulation, or are used as free degrees of freedom for optimization studies) and constants (fixed value elements), which are used in the declaration of the equations that will compose the models; implementation of a dimensional coherence check mechanism in the declaration of the equations, thus the result of the operations between the equation elements were allowed for dimensionally equal ones (e.g.: addition, subtraction) or the resultant dimension were calculated (e.g.: multiplication, division, potentiation, and so on); implementation of a conversion mechanism of equation elements into symbolic ones, and operationalization of the equations through a symbolic computation routine; implementation of a detection mechanism for the mathematical type of system equations that compose the problem (e.g.: linear, non-linear, differential, and so on), and utilization of the corresponding solution mechanism for the symbolic system.

The open-source software represent important alternatives to the commercial tools in terms of the acquisition and license costs, which is frequently prohibitive for some academic institutions or small size private entities. In this sense, besides the aforementioned economical advantages, the open-source softwares allow for rapid development, testing and implementation of necessary features and eventual bug fixes. It is also worth to mention the important aspect of flexibility of the tool for problems that go beyond the application-oriented focused software development common to commercial closed-source applications. Furthermore, the utilization of open-source tools corroborates with research reproducibility, which is cardinal to the proper development of scientific method. Lastly, it can also foster the collaboration in joint research projects between academic institutions and industrial partners (Bilke *et al.*, 2019; Keilegavlen *et al.*, 2021; Pfenninger *et al.*, 2017).

The software employed in the present work was conceived through the utilization of object oriented program (OOP) paradigm, due to the intended versatility and reutilization capabilities intended for the tool. The implementation employs the concept of inheritance and polymorphism (Hiremath & Tavade, 2016) for the `Variable`, `Parameter` and `Constant` objects class, that derivate from a broader type of object class, called `Quantity`. Those objects can either have a specified numerical value (absent for undetermined parameters) and a dimensional definition through an `Unit` object. Similarly, the `UnitOp` objects are derived from the object class `Model` with the addition of mass conservation and mixture calculations obtained from their input and output stream properties, which receive information from other `UnitOp` objects. In the supplementary material (available in the on-line version of the paper) the aforementioned object classification in the SLOTH tool are summarized, with additional categories.

In broader terms, the SLOTH tool aims to constitute a higher level of computational routines, constituting an application programming interface (API) for model development, optimization, parameter estimation and control. Notable tools such as EMSO (Soares & Secchi, 2003), DAETOOLS (Nikolić, 2016), IDAES (Gunter *et al.*, 2018) and ASCEND (Westerberg *et al.*, 1994) stands out as direct inspiration for our software, although SLOTH was developed with the aim to be collaboratively built and available for reutilization of its code, constituting an educational resource for equation-oriented process modeling and simulation studies regarding the aforementioned capabilities of the tool. In this sense, SLOTH tool was developed using the Python computational language, due to its advantages such as high-level characteristics, broad ecosystem of libraries for diverse purposes (e.g.: symbolic computation, differential equation systems solution, parallel computation, graph plotting, among others) and easy integration with routines developed in other computational languages (Gowers *et al*., 2019; Hazan *et al*., 2018; Larsen *et al*., 2017).

1.1.2 Definition of quantitative objects and equation insertion

The `Variable`, `Parameters` and `Constants` objects represent the different types of mathematical entities that can be used in the equation declaration in the SLOTH tool, through a `Equation` object. Those classification types are presented in the Table 1. Those types of objects differ themselves by their utilization in the model declaration process. The `Variable` object represents one unknown of the equations system, and then contributing for the global number of degrees of freedom of the problem. Similarly, the `Parameter` object could contribute to the degrees of freedom if left unspecified, which is used for parameter determination studies through the `Optimization` objects. If its value is specified, this object is treated as a fixed value for all the numerical purposes, likewise the `Constant` objects, which need to be numerically specified during its declaration. It is worth to emphasize that those three classes constitute a group referred as *quantitative objects* in the tool scope, being derived from the `Quantity` primitive class, as discussed earlier. Thus, the objects defined from this class exhibit the characteristic properties of a numerical value (whether specified or not), and a dimensional value, defined through an `Unit` object, as the dimensional coherence is cardinal for the equation composition, as will be explained below. It is worth to mention that in the operation between two different types of quantitative objects (`Variable` and `Parameter`) in the equations, symbolic computation is employed to represent the operation between the corresponding symbols of those objects. If one of the objects has its value specified (e.g.: specified `Parameter` or a `Constant` object), its numerical value is used for the symbolic representation. In the Figure 1, the schematic representation of the symbolic computation in the SLOTH tool is presented, which synthesize the aforementioned description of the operations between SLOTH primitive types for readers convenience.

The quantitative objects can be directly used in the equation formulation, through their conversion routine in `EquationNode` objects, the proper equational terms. In turn, the `Equation` objects store the equations that compose the models declares in the tool, and are defined through the operation between the `EquationNode` objects. As aforementioned, the operation between adequate dimensional terms is a cardinal aspect for the equation formarion, as it is a requirement for the dimensional coherence of the resultant equational terms. As presented schematically in the Figure 1, the addition and subtraction of two quantitative object is only performed if those objects have equivalent dimensions, otherwise an error message is returned for the user. For the multiplication, division, potentiation and rooting, this dimensional check is not necessary, but the resultant quantitative object dimension is calculated accordingly. Lastly, for the transcendental operations (exponentiation, logarithm, and so on), the operand must be dimensionless, otherwise an error message is returned.

1.1.3 Model definition and composition of a problem

The `Model` objects represent the process models in the context of the SLOTH tool. For its composition, quantitative objects of the model are declared, as well as the equations that describe mathematically the process under study, forming a well-posed system. As previously described, the number of degrees of freedom of a model must be equal to zero for simulation studies, whereas a positive number is required for optimization studies, which will represent ultimately the parameters to be determined by the internal routines of the tool. The connection between two `Model` objects, for instance the case in which one output stream of a process is the input for another, is performed through a `Connection` object, which is generated by a homonym class, which is derived from `Equation` class. In the context of the tool, for the connection of the models, a special equation is created, coupling the variables from both models. In Figure 2, the connection between two models is schematically represented, obtaining a final set of equations to be solved, grouped into a single `EquationBlock` object.

Table 1. Categories of objects used in the SLOTH tool, describing the conceptual name of objects and their corresponding nomenclature within the software, and its utilization.

Object type	**Internal name**	**Utilization**
Variable	`Variable`	Unknowns that will have their value determined through the resolution of the equation system
Parameter	`Parameter`	Can have their value specified or determined through optimization studies, or represent one degree of freedom of the problem
Constant	`Constant`	Objects which value is fixed and specified in the declaration of the model
Quantity	`Quantity`	Primitive object for `Variable`, `Parameter` and `Quantity`, which can have its value specified and has a dimension expressed through a `Unit` object
Unit	`Unit`	Object which contains the dimensional information about a `Variable`, `Parameter` or `Constant`
Equation node	`EquationNode`	Object that integrates an equation, obtained through a `Variable`, `Parameter` or `Constant`
Equation	`Equation`	Expressions using variables, parameter and constants that constitute the models
Equation block	`EquationBlock`	Array of equations used to describe a model
Model	`Model`	Objects which represent the mathematical models used in the description of the process under study
Unit operation	`UnitOp`	Object derived from `Model`, which contains mechanisms for mass conservation and mixture calculations through input and output streams
Problem	`Problem`	Set of models used to describe the case of study
Optimization	`Optimization`	Object which holds an optimization study to be performed, using a predefined optimization problem
Optimization Problem	`OptimizationProblem`	Object which contains an optimization problem which objective function is related to a case of study

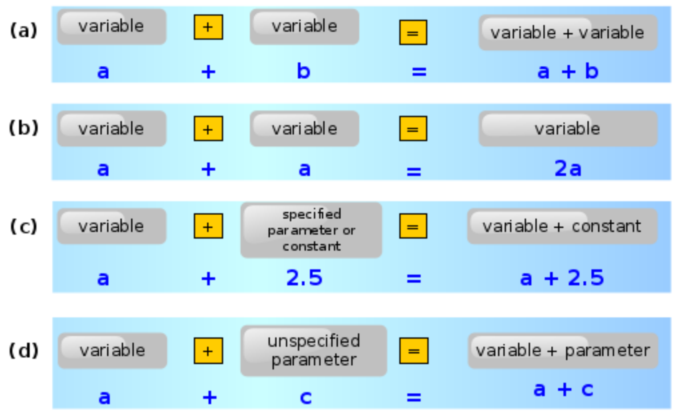

Figure 1. Schematic representation of the arithmetical operation process in the tool (addition) from two different `Variable` objects (a), two equivalent `Variable` objects (b), a `Variable` and a specified `Parameter` or a `Constant` object (c) and between a `Variable` and an unspecified `Parameter`, with their respective symbolic result.

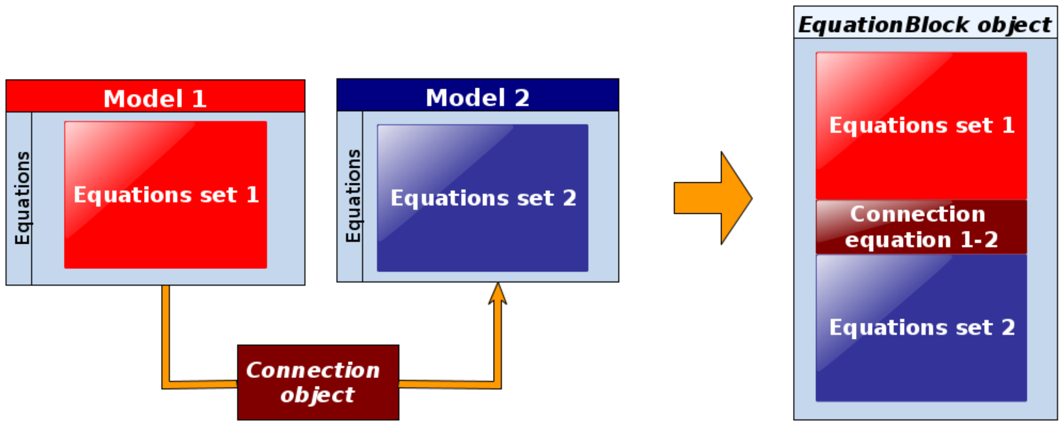

Figure 2. Schematic representation of the connection process between two `Model` objects, through a `Connection` object, obtaining a resultant `EquationBlock` object.

In the supplementary material of the present work, a code snippet for the creation of a generic model (Code 1) is presented, in which parameters and constants are declared, as well as the equations that describe it mathematically.

1.1.4 Simulation of a problem and optimization studies

In the context of the SLOTH tool, the simulation occurs through the `Simulation` objects, which in turn are defined from a case of study of the user, represented by `Problem` objects. During the simulation procedure, in the definition of the `Simulation` object, the initial conditions necessary for the resolution of the equation system are declared, as well as additional configurations (e.g.: solver type, tolerances, among others). The mathematical type of equation system in the `EquationBlock`, obtained after the consolidation of the case of study, is automatically detected among the types of equational systems covered by the SLOTH tool: linear equations, non-linear equations, ordinary differential equations and differential algebraic equations (DAE). Upon the detection of the equation system, the proper internal routines are used for the resolution of the problem. After the resolution, depending on the nature of the solved problem, the stationary set values for the variables (steady-state problems) or a dynamic profile for those (dynamic problems) is returned. Those calculation routines utilizes the resource of the powerful symbolic calculation library `sympy` (Meurer *et al.*, 2017), together with the libraries assimulo (Andersson *et al.*, 2015) and scipy (Jones *et al.*, 2001) for the resolution of the differential and differential-algebraic problems, and the library pyneqsys (Dahlgren, 2018) for the linear and nonlinear equations.

The optimization process in the SLOTH tool is performed through an `Optimization` object. During its declaration, an objective function must be provided, defined through an `OptimizationProblem` object, where the user should define the number of parameters of the objective function, their constraints and additional configurations. The user should define which algorithm will be used to solve the optimization problem. In this intent, the routines provided by the `pagmo` (Biscani et al., 2019) library are used, through its interface for Python computational language. It is worth to emphasize that this library is endorsed by ESA (European Spacial Agency), consisting in state-of-the-art tool for optimization studies. This library provide support for gradient numerical calculation through finite differences utilization, what could subsidize the utilization of gradient-based optimization routines, although the initial focus has being meta-heuristic routines, due to its inherent characteristics of flexibility. As mentioned in the Section *1.1*.3, the problem must have a positive number of degrees of freedom for the optimization process, as its mechanism operates through the iterative process of specifying a numerical value for the parameters, evaluating the response of the objective function (`OptimizationProblem` object) and storing the result, searching for the minimization of its value. The user can provide an external optimization routine for the optimization process in the SLOTH tool, using the same mechanisms implemented in the `pagmo` library.

In the supplementary material of the present work, a code snippet for the creation of a `Simulation` object from a model declared in SLOTH tool (Code 2) is presented and the subsequent simulation of the problem, specifically a Lotka-Voltera population dynamic model (Bomze, 1995) simplification. Likewise, in the Code 3, a code snippet for an optimization study of the referred model is presented, which make direct use of the Code 2, although it is omitted.

1.2 Architecture of the tool

In order to provide to the reader a broader view of the software implementation, the computational architecture of the SLOTH tool is described conceptually in the supplementary material of the present work. As aforementioned, the main functional structure of SLOTH was developed through the Python computational language.

2 Cases of study and methodology

In the present section, cases of study for the application of the SLOTH tool are presented. In order to demonstrate the capabilities of the tool, benchmarking problems were used, through the utilization of their corresponding mathematical models.

The employed stochastic meta-heuristic routines were employed due to the advantages over deterministic methods for nonlinear problems, such as the bioprocess presented in the case study I of the current work, for which generally good solutions are normally obtained in relatively modest computational times. Those methods are also referred by their flexibility when compared to their deterministic counterparts, as the analytical properties of the problem are not directly explored (Rocha *et al*, 2014; Ochoa, 2016).

2.1 Case I: Optimization of substrate and inducer feed in fed-batch cultivation

The present case of study discusses the cultivation of recombinant bacteria (*E. Coli*) for the production of induced foreign protein (β-galactosidase), considering the addition of glucose substrate and inducer (Isopropylthiogalactoside, IPTG) (Lee & Ramirez, 1994). The corresponding mathematical dynamical model is described in the Equations (1)-(7) (Rocha *et al.*, 2014).

$$\frac{dV}{dt} = u_1 + u_2 \tag{1}$$

$$\frac{dX}{dt} = \mu X - \frac{(u_1+u_2)}{V}X \tag{2}$$

$$\frac{dS}{dt} = \frac{u_1}{V}C_{nf} - \frac{(u_1+u_2)}{V}S - \frac{\mu}{Y}X \tag{3}$$

$$\frac{dP}{dt} = \Lambda_{fp}X - \frac{(u_1+u_2)}{V}P \tag{4}$$

$$\frac{dI}{dt} = \frac{u_2}{VC_{if}} - \frac{(u_1+u_2)}{V}I \tag{5}$$

$$\frac{dK}{dt} = -k_1 K \tag{6}$$

$$\frac{dR}{dt} = k_2(1-R) \tag{7}$$

The constitutive relations of the model are presented in the Equations (8)-(11) (Rocha *et al.*, 2014).

$$\Lambda_R = \frac{0.22}{0.22+I} \tag{8}$$

$$\mu = \frac{0.407S}{0.108 + S + 6.7495\times10^{-5}S^2}(K + R\Lambda_R) \tag{9}$$

$$\Lambda_{fp} = \frac{0.095S}{0.0108 + S + 6.7495\times10^{-5}S^2}\frac{0.0005+I}{0.022+I} \tag{10}$$

$$k_1 = k_2 = \frac{0.09I}{0.034+I} \tag{11}$$

The initial conditions for the simulation of the aforementioned model are presented in the Table 2, as well as the its parameters values.

The present optimization problem consists in the maximization of the profitability of the fed-batch process, expressed mathematically though the Equation (12) (Rocha *et al.*, 2014). By definition, the SLOTH tool treat optimization problems as minimizations, so the referred objective function is expressed in the negative form when compared to the literature definition (Rocha *et al.*, 2014).

$$J_f = -P(t_f)V(t_f) + Q\int_0^{t_f} u_2(t)dt \tag{12}$$

Table 2. Initial conditions and parameter values for the model used in the present case of study (Rocha *et al.*, 2014).

Term	**Unit**	**Value**	**Meaning**
V	L	1	Reactor volume
X	gL^{-1}	0.1	Celular biomass concentration
S	gL^{-1}	40	Glucose substrate concentration
P	gL^{-1}	0	Foreign protein (product) concentration
I	gL^{-1}	0	Inducer concentration
K	gL^{-1}	1	Concentration of inducer shock factor on cell growth rate
R	gL^{-1}	0	Concentration of inducer recovery factor on cell growth rate
C_{if}	gL^{-1}	4	Inducer feed-stream concentration
C_{nf}	gL^{-1}	100	Nutrient feed-stream concentration
Y	h^{-1}	0.51	Growth rate yield coefficient
k_1	h^{-1}	Calculated	Model calculated parameter
k_2	h^{-1}	Calculated	Model calculated parameter

In Equation (12), the term Q represent the ratio of the cost of the inducer to the value of the protein product, and t_f is the fed-batch duration, which are defined respectively as 5 and $15h$. Thus, the final product (J_f) is calculated through the profitability of the quantity of foreign protein produced, deduced by the cost of inducer feed-stream employed. For this open-loop optimization problem, the variables u_1 and u_2 constitute the manipulated variables, constrained to the interval [0; 1.0]. In order to provide a parametric representation of the substrate and inducer feed rates (u_1 and u_2), a cosinoidal expression was employed, that has the advantages of exhibiting a smooth flow rate profile (Ochoa, 2016). Thus, the functions for dynamic determination of the manipulated variables are presented in Equations (13)-(14)

$$u_1(t) = a_1 + cos(b_1 + c_1 t) \tag{13}$$

$$u_2(t) = a_2 + cos(b_2 + c_2 t) \tag{14}$$

In Equations (13)-(14) the terms a_1, b_1, c_1, a_2, b_2 and c_2 represent the parameters to be optimized, in order to minimize the output of the objective function of the problem (J_f), calculated through Equation (12). In the work of Rocha *et al.* (2014), the authors have employed a sequence of linear profiles for the optimization, with 4 segments for each manipulated variable (thus, 8 decision variables). In the present work, the cosinoidal parametrization was preferred due to the lower number of decision variables and the resultant smooth feeding profile.

2.2 Case II: Parameter estimation of polymerization kinetics

The present case of study presents the estimation of reaction parameters of ethylene polymerization, carried out with nickel complexes (Schwaab *et al.*, 2008). It is assumed that the reaction rates and the active species exhibit a dynamic behavior, which can be described mathematically through the Equation (15).

$$R_p(t) = \sum_{i=0}^{m}\left[\left(\sum_{n=i}^{m} Kp_n A_i^n\right) exp(-k_i t)\right] \tag{15}$$

In the Equation (15), the terms t, R_p, Kp_n, n, i, k_i, m represent respectively the reaction time (min), the rate of ethylene consumption along time ($mol\,min^{-1}$), the polymerization rate constant for the *n-th* active specie ($mol\,min^{-1}$), the *n-th* active specie, the *i-th* active specie, the rate constant for transformation of *i-th* active species into *(i+1)-th* (min^{-1}) and the maximum number of active species in the reaction medium of catalyst. The coefficients A_i^n are defined recursively as presented in the Equations (16)-(18)

$$A_0^0 = 1, \tag{16}$$

$$A_i^n = A_i^{n-1}\frac{k_{n-1}}{k_n - k_i},\ i = 1\ldots n-1, n > 0, \tag{17}$$

$$A_n^n = -\sum_{i=0}^{n-1} A_i^n \tag{18}$$

In the present case, the occurrence of three active species ($m = 3$) was assumed, due to the most accurate model results when compared to the experimental data (Schwaab *et al.*, 2008). Thus, under this consideration, the obtained model for the reaction rates are presented in Equation (19).

$$\begin{aligned} R_p(t,\tilde{p}) = & \left(Kp_1A_0^1 + Kp_2A_0^2 + Kp_3A_0^3\right)exp(-k_0t) \\ & + \left(Kp_1A_1^1 + Kp_2A_1^2 + Kp_3A_1^3\right)exp(-k_1t) \\ & + \left(Kp_2A_2^2 + Kp_3A_2^3\right)exp(-k_2t) \\ & + Kp_3A_3^3 \end{aligned} \tag{19}$$

In Equation (19), the term $\tilde{p}$ represent the parameters to be estimated, namely Kp_1, Kp_3, Kp_3 and k_0, k_1, k_2. Thus, there are six parameters to be determined. In the aforementioned equation, the terms k_3 and Kp_0 are considered zero, as the last specie is stable, and the initial catalyst species is not active for polymerization, respectively, as discussed in Schwaab *et al.* (2008). The expressions for calculation of the recursively defined coefficients (e.g.: $A_0^0, A_1^0, \ldots, A_3^3$) as functions of the rate constant of transformation (e.g.: $k_0, k_1, \ldots, k_3$) were omitted, however they could be promptly calculated through the Equations (16)-(18).

The parameter estimation consists in the minimization of an objective function that describes the deviation of the model results from the experimental data presented in the literature (Dias *et al.*, 2006), obtained graphically from Schwaab *et al.* (2008) using the DataThief software (Tummers, 2006). The objective function used is presented in the Equation 20, which correspond to the well-known square-loss equation (Tulsyan & Barton, 2016), or the sum of squared errors between the experimental and calculated data.

$$J_f = \sum_{i=1}^{N} (y - \hat{y})^2 \tag{20}$$

In Equation (20), the terms J_f, N, y, $\hat{y}$ represent respectively the objective function value, the number of experimental data-points, the experimental value and the output calculated by the model. The search space for each parameter to be estimated is presented in the Table 3, determined accordingly to the literature (Schwaab *et al.*, 2008).

Table 3. Search region (constraints) for parameter estimation for the model (Schwaab *et al.*, 2008).

Parameter	**Lower bound**	**Higher bound**
Kp_1	0	100
Kp_2	0	1
Kp_3	0	10
k_0	0	100
k_1	0	50
k_2	0	1

2.3 *Case III: Vapor-liquid equilibrium of a non-ideal binary mixture*

The present case of study presents the calculation of a vapor-liquid equilibrium for a non-ideal binary mixture using the modified Raoult's law and the Wilson equation. Those calculations are employed in the process design and analysis in the industrial context, as example of distillation columns (Ferro *et al.*, 2015; Kister *et al.*, 1992; Medina-Leaños *et al.*, 2020; Mendoza & Riascos, 2020). The thermodynamic model that describes the referred problem is presented in the Equations (21)-(30) (Shacham & Brauner, 2017), constituting one set of non-linear equations to be solved.

$$P_1^{sat} = \exp[17 - 3600/(T - 54)] \tag{21}$$

$$P_2^{sat} = \exp[16.5 - 3850/(T - 47)] \tag{22}$$

$$y_1 = \frac{x_1\gamma_1P_1^{sat}}{P} \tag{23}$$

$$y_2 = \frac{x_2\gamma_2P_2^{sat}}{P} \tag{24}$$

$$y_1 + y_2 = 1 \tag{25}$$

$$\gamma_1 = \frac{exp(x_2W)}{x_1 + x_2\Lambda_{12}} \tag{26}$$

$$\gamma_2 = \frac{exp(-x_1W)}{x_1\Lambda_{21} + x_2} \tag{27}$$

$$W = \frac{\Lambda_{12}}{x_1 + x_2\Lambda_{12}} - \frac{\Lambda_{21}}{x_1\Lambda_{21} + x_2} \tag{28}$$

$$\Lambda_{12} = \frac{V_2}{V_1}\exp\left(\frac{-a_{12}}{RT}\right) \tag{29}$$

Table 4. Parameter values for the model used in the present case of study (Shacham & Brauner, 2017).

Term	Unit	Value
V_1	$cm^3\,mol^{-1}$	77
V_2	$cm^3\,mol^{-1}$	18
P	kPa	100
x_1	–	0.85
x_2	–	0.85
a_{12}	$cal\,mol^{-1}$	440
a_{21}	$cal\,mol^{-1}$	1250
R	$cal\,K^{-1}\,mol^{-1}$	1.987

$$\Lambda_{21} = \frac{V_1}{V_2}\exp\left(\frac{-a_{21}}{RT}\right) \tag{30}$$

In the Equations (21)-(30), the terms P_1^{sat} and P_2^{sat}, T, x_1 and x_2, P, γ_1 and γ_2, V_1 and V_2 represent respectively the saturation pressure values (kPa), temperature (K), liquid mole fraction, pressure (kPa), the activity coefficients and the molar volume ($cm^3\,mol^{-1}$). The terms a_{12}, a_{21}, W, Λ_{12} and Λ_{21} represent the parameters of the Wilson equation, which are presented in the Table 4, along with other relevant parameters for the equation set.

3 Results and discussions

3.1 *Results and discussion for case I*

The solution of the optimization problem presented in the Equations (1)-(7) were performed through the utilization of the SLOTH tool. Due to the high non-linearity observed in the refereed problem, common to biotechnological systems, meta-heuristic algorithms were employed in the optimization, as aforementioned. In this sense, two different routines were selected: self-adaptive differential evolution (SADE) and particle swarm optimization (PSO) with constriction coefficient velocity update, available through the interface with the `pagmo` library (Biscani *et al.*, 2019). The configuration employed for both algorithms corresponds to 30 candidate solutions (or 30 particles, for PSO), and the stop criterion is defined as 500 iterations. Additionally, the PSO required the configuration of the social component (2.05), cognitive component (2.05), maximum particle velocity (0.5) and inertia value (0.7298), defined empirically. The configurations are presented in summarized form in the supplementary material (available in the on-line version of the paper). It is worth to mention that due to its self-adaptive nature, the SADE algorithm requires only the parameters of number of candidate solutions and number of generations to its operation.

Table 5. Results obtained for the estimation of the parameters for the model referent to the case I, for PSO and SADE routines.

Parameter	SADE	PSO
a_1	−0.833	−0.819
b_1	−4.768	−9.996
c_1	−4.814	5.015
a_2	−0.827	−0.832
b_2	−0.589	8.247
c_2	−8.384	8.368
$\lvert J_f \rvert$ (Equation 12)	5.922	5.873

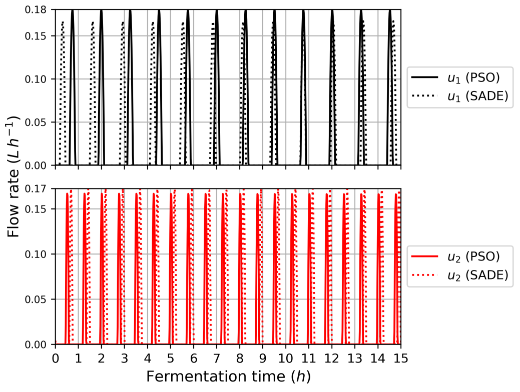

Fig. 3. Dynamic profiles for the manipulated variables for the problem presented in the case I (u_1 and u_2), for the parameter set obtained by PSO and SADE algorithms.

In Table 5, the results obtained for the estimation of the parameters for the model referent to the case I are presented, namely in the Equations (13)-(14), for both optimization routines employed in this intent, along with the final result for the objective function (J_f), calculated through the Equation (12). In Table 5, the value of the productivity index is presented in the positive form (inverse of calculated through Equation (12).

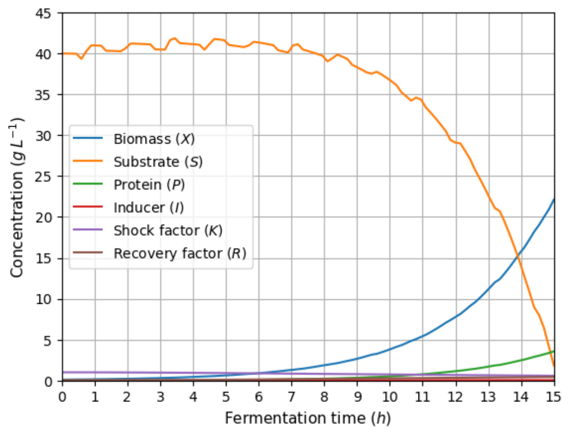

Fig. 4. Dynamic profile of the concentration of the components of the fermentation medium, for the solution obtained through the utilization of PSO algorithm.

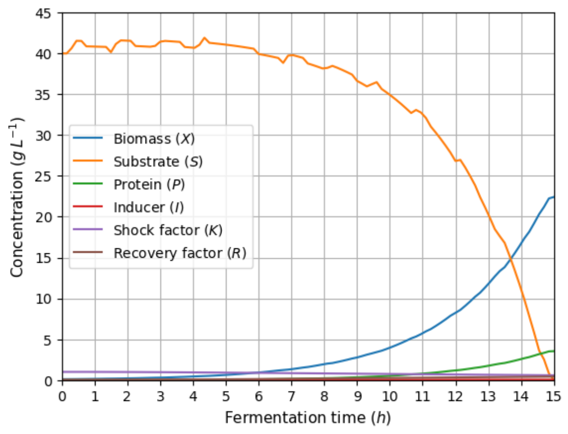

Fig. 5. Dynamic profile of the concentration of the components of the fermentation medium, for the solution obtained through the utilization of SADE algorithm.

In Figure 3, the dynamic profile for the manipulated variables for the problem presented in the case I, namely u_1 and u_2, are presented. The dynamic profile of the inducer and substrate flow rate exhibit a series of repetitive pulses with an approximate amplitude of 15 minutes. In Figures 4-5, the dynamic profiles obtained for each optimization routine are presented (PSO and SADE, respectively), in terms of the concentration of pertinent metabolites in the concentration medium. The almost identical graphical representation of the dynamic profile of the components of the fermentation medium endorse the assumption that both optimization algorithms were equivalent in terms of the objective function minimization for this specific problem, with some advantage for the results obtained with SADE algorithm (higher productivity index, J_f, as presented in the Table 5). Those results are far superior from those presented in the literature: 0.8165 (Rocha et al., 2014), and 0.7899 (Lee & Ramirez, 1994), which can be explained by the pulsative profile prodived by the cosinoidal parametrization of the feed rate, contributing to lower consumption of inducer, thus implying in a higher productivity. It is worth to emphasize that the exploration of superior optimization routines for the problem presented in this case of study is beyond the scope of the present work, which is more directly related towards the discussion of relevant aspects of EO simulation tools development, focusing in the applicability of SLOTH tool. The utilization of different optimization algorithms can be performed by the integration of an external solver, which can be easily integrated in the SLOTH tool, which can also include monolithic solvers developed in other languages (FORTRAN, C++, R, among others) through modern alternatives for interface between Python and other computational languages, which are beyond the scope of this work.

3.2 *Results and discussion for case II*

The solution of the parameter estimation problem presented in the Equations (15)-(20) were performed using the SLOTH tool, using the same meta-heuristic algorithms employed in the Section *3.1*. In the Table 6, the results obtained for the estimation of the model parameters for the case II are presented, for both optimization routines employed (SADE and PSO), along with the final result for the objective function (Equation (20)). The results presented in the Table 6 indicate that the PSO algorithm was able to find a better solution for the parameter estimation problem referent to the case II, with a final value of the objective function (Equation 20) significantly smaller than the found by SADE algorithm. It is reasonable to suppose that due to the self-adaptive nature of this last routine, which tests several strategies for recombination of the candidate solutions, it would be necessary to empirically determine the better combination of strategies for the minimization of the objective function. In the Figure 6, the model results are compared graphically to the experimental data for both optimization routines. Both results are far superior from those presented in the literature (Schwaab *et al.*, 2008), which include two minima

Table 6. Results obtained for the estimation of the parameters for the model referent to the case II, for PSO and SADE routines.

Parameter	**SADE**	**PSO**	**Minimum A (Schwaab *et al.*, 2008)**	**Minimum B (Schwaab *et al.*, 2008)**
Kp_1	99.989	78.153	55.98	249.6
Kp_2	1.000	0.584	0.000	0.000
Kp_3	9.999	8.757	7.587	7.587
k_0	48.222	3.239	10.31	2.313
k_1	49.999	4.858	2.313	10.30
k_2	0.868	0.208	0.232	0.232
J_f (Equation 20)	1530.314	42.267	15127.372	275.638

Table 7. Results obtained using the SLOTH tool for the problem presented in the case III.

Variable	**Result (this study)**	**Result (Shacham & Brauner, 2017)**	**Unit**
Λ_{12}	0.1228	0.123	-
Λ_{21}	0.6879	0.688	-
γ_1	1.022	1.022	-
γ_2	2.675	2.675	-
P_1^{sat}	99.077	99.077	kPa
P_2^{sat}	34.706	34.706	kPa
T	344.23	344.227	K
W	-0.7948	-0.795	-
y_1	0.8607	0.861	-
y_2	0.1393	0.139	-

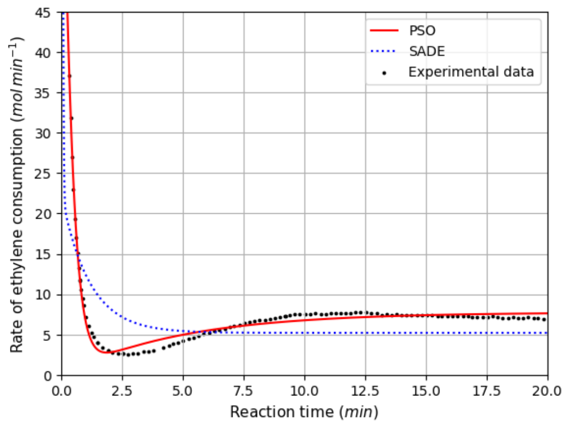

Fig. 6. Comparison between the model results and the experimental data, for the parameter set obtained by PSO and SADE algorithms.

points in the objective function (Equation 20). The authors discuss the challenging aspect of the underlying minimization problem, as the point of minima is located in a very narrow region of the parameter space, endorsing the capabilities of current tool in the parameter estimation task. This result is explained by the utilization of the algorithms provided by the robust library `pagmo` as discussed earlier. It is worth to mention that due to the stochastic nature of the PSO, SADE and other metaheuristic routines, the performance is specific to each particular problem, and the user need to test the individual performances of the algorithms.

3.3 *Results and discussion for case III*

The solution of the non-linear equation system presented in the Equations (21)-(30), were performed through the internal routines of the SLOTH tool. The results obtained are presented in the Table 7. It is worth to emphasize that those results are equivalent to those presented in the literature (Shacham & Brauner, 2017), endorsing the software as a reliable tool for related problem solving.

Table 8. Results obtained using the SLOTH tool for residuals of the equations of the problem presented in the case III.

Equation	**Residual value ξ (this study)**	**Residual value ξ (Shacham & Brauner, 2017)**
21	7.1×10^{-14}	4.16×10^{-17}
22	-7.10×10^{-14}	-1.11×10^{-16}
23	0.0	1.39×10^{-16}
24	-2.77×10^{-17}	-4.95×10^{-16}
25	5.55×10^{-17}	0.0
26	0.0	3.55×10^{-15}
27	4.44×10^{-16}	2.78×10^{-17}
28	0.0	-3.33×10^{-16}
29	1.66×10^{-16}	0.0
30	1.44×10^{-15}	1.71×10^{-12}

In Table 8, the residuals obtained through the SLOTH tool for each of the Equations (21)-(30) are presented, and compared to those presented in the work of Shacham & Brauner, (2017),. The residual form of the equation is obtained through the algebraic manipulation of the aforementioned expressions to the form described in the Equation (31) below.

$$L - R = \xi \tag{31}$$

In Equation (31), the L, R and ξ terms mean respectively the left-hand side of the equation, the right-hand side and the residual.

The obtained values are very similar to those presented in the literature, and minor differences could be attributed to rounding errors due to floating point precision numbers employed in the calculations. It is evident that the inclusion of a high precision calculation mode in the SLOTH tool could be beneficial for applications requiring high accuracy levels, which can be implemented through utilization of arbitrary precision mathematical support, e.g.`mpmath` library functions (Meurer *et al.*, 2017), which can however increase the computational demand for the calculations.

Conclusions

In this work, a study concerning the utilization of an in-house developed tool for equation-oriented simulation and optimization of processes, called SLOTH, was conducted. In this sense, three different cases of study were performed: maximization of the production of heterologous protein in a fed-batch fermentation process (case I); parameter estimation of a mathematical model for ethylene polymerization carried out with nickel complexes, using experimental data (case II); calculation of a vapor-liquid equilibrium for a non-ideal binary mixture (case III). The SLOTH tool has exhibited good capabilities in the resolution of the studied problems, being obtained numerical results coherent to those presented in the literature for the aforementioned study cases. The results obtained for case I represent superior productivity indexes to those presented in the literature, due to the utilization of a cosinoidal feed profile parametrization and meta-heuristic optimization routines employed; for the case II, good agreement with experimental data was obtained, also superior to the results presented in the literature for these specific case; for the case III, the obtained residual values for the equilibrium equations are compatible to those presented in the literature, with some divergences attributed to rounding errors and the floating-point precision employed in the calculations. Additionally, mathematical resources such as surrogate modelling could also be implemented for reducing the computational burden in function evaluation, especially for optimization problems that require several evaluations of an objective function. The inclusion of convenience routines for inclusion of external solvers in the tool is also a relevant aspect for further SLOTH software development.

However, it is worth to mention that the optimization process has exhibited extensive computational times, which is justified by the interpreted nature of the main computational language employed in the development of the software, Python. This fact suggested the importance to identify computational bottlenecks and migrate some of the calculation intensive tasks to be performed by a compiled language (e.g.: C++, FORTRAN), as well as the utilization of parallel computation. This aspect need to be considered for those interested in development of EO simulation tools. Also, it is worth to mention that the implementation of high precision floating-point calculations when necessary may benefit specific user cases.

Despite the aforementioned fact concerning the lack of speed of the SLOTH tool for some tasks, it has proven to be suited for the purpose of its development, the equation-oriented simulation of processes and correlated problems. The intent of the software is not to compete with well-established tools such as EMSO, GAMS, DAETOOLS, IDAES, MODELICA, among others. The main goal of the project is to provide an open-source code in a higher-level language such as Python for tackling equation-oriented process simulation, which could be continuously enhanced as the source-code is open to contributions, as well as to provide an auxiliary structure for those interested in develop their own tools.

References

Abel, J. H., Drawert, B., Hellander, A., & Petzold, L. R. (2016). Gillespy: A python package for stochastic model building and simulation. *IEEE Life Sciences Letters 2(3)*, 35-38. dx.doi.org/10.1109/LLS.2017.2652448

Andersson, C., Führer, C., & Åkesson, J. (2015). Assimulo: A unified framework for ODE solvers. *Mathematics and Computers in Simulation 116(0)*, 26-43. https://doi.org/10.1016/j.matcom.2015.04.007

Azar, A. T., & Vaidyanathan, S. (2015). *Chaos Modeling and Control Systems Design* (Vol. 581). Springer.

Bilke, L., Flemisch, B., Kalbacher, T., Kolditz, O., Helmig, R., & Nagel, T. (2019). Development of open-source porous media simulators: Principles and experiences. *Transport in Porous Media 130(1)*, 337-361. https://doi.org/10.1007/s11242-019-01310-1

Biscani, F., Izzo, D., Jakob, W., Märtens, M., Mereta, A., Kaldemeyer, C., Lyskov, S., Corlay, S., Pritchard, B., Manani, K., Mabille, J., Miąsko, T., Huebl, A., jakirkham, hulucc, polygon, Fu, Z., Badger, T. G., Nimier-David, M., . . . Mambrini, A. (2019). Esa/pagmo2: Pagmo 2.10. https://doi.org/10.5281/zenodo.2529931

Bomze, I. M. (1995). Lotka-Volterra equation and replicator dynamics: New issues in classification. *Biological Cybernetics 72(5)*, 447-453. https://doi.org/10.1007/BF00201420

Capocchi, L., Santucci, J. F., Poggi, B., & Nicolai, C. (2011). Devsimpy: A collaborative python software for modeling and simulation of devs systems. *2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 170-175. https://doi.org/10.1109/WETICE.2011.31

Dahlgren, B. (2018). Pyneqsys: Solve symbolically defined systems of non-linear equations numerically. *Journal of Open Source Software 3(21)*, 531. https://doi.org/10.21105/joss.00531

Dias, M. L., da Silva, L. P., Crossetti, G. L., Galland, G. B., Filgueiras, C. A., & Ziglio, C. M. (2006). Propylene polymerization with nickel-diimine complexes containing pseudohalides. *Journal of Polymer Science Part A: Polymer Chemistry 44(1)*, 458-466. https://doi.org/10.1002/pola.21013

Dowling, A. W., & Biegler, L. T. (2015). A framework for efficient large scale equationoriented flowsheet optimization. *Computers & Chemical Engineering 72*, 3-20. https://doi.org/10.1016/j.compchemeng.2014.05.013

Ferro, V., De Riva, J., Sanchez, D., Ruiz, E., & Palomar, J. (2015). Conceptual design of unit operations to separate aromatic hydrocarbons from naphtha using ionic liquids. Cosmo-based process simulations with multi-component "real" mixture feed. *Chemical Engineering Research and Design 94*, 632-647. https://doi.org/10.1016/j.cherd.2014.10.001

Gowers, R. J., Linke, M., Barnoud, J., Reddy, T. J. E., Melo, M. N., Seyler, S. L., Domanski, J., Dotson, D. L., Buchoux, S., Kenney, I. M., *et al.* (2019). Mdanalysis: A python package for the rapid analysis of molecular dynamics simulations (tech. rep.). Los Alamos National Lab.(LANL), Los Alamos, NM (United States). https://doi.org/10.25080/Majora-629e541a-00e

Gunter, D. K., Agarwal, D. A., Beattie, K. S., Boverhof, J. R., Cholia, S., Cheah, Y.-W., Elgammal, H., Sahinidis, N. V., Miller, D., Siirola, J., *et al.* (2018). Institute for the design of advanced energy systems process systems engineering framework (idaes pse framework) (tech. rep.). Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States).

Hazan, H., Saunders, D. J., Khan, H., Sanghavi, D. T., Siegelmann, H. T., & Kozma, R. (2018). Bindsnet: A machine learning-oriented spiking neural networks library in python. *Frontiers in Neuroinformatics 12*, 89. https://doi.org/10.3389/fninf.2018.00089

Hernández-Díaz, W., Hernández-Campos, F., Vargas-Galarza, Z., Rodríguez-Jimenes, G., & García-Alvarado, M. (2020). Coffee grain rotary drying optimization. *Revista Mexicana de Ingeniería Química 12(2)*, 315-325. http://www.rmiq.org/ojs311/index.php/rmiq/article/view/1475

Hiremath, S., & Tavade, C. (2016). Review paper on inheritance and issues in object oriented languages. *International Journal of Advancement in Engineering Technology, Management & Applied Sciences*, 2349-3224. https://doi.org/10.2139/ssrn.3451083

Inei-Shizukawa, G., Velasco-Bedrán, H., Gutiérrez-López, G., & Hernández-Sánchez, H. (2020). Statistical approach to optimization of ethanol fermentation by saccharomyces cerevisiae in the presence of valfor® 100 zeolite naa. *Revista Mexicana de Ingeniería Química 8(3)*, 265-270. http://www.rmiq.org/ojs311/index.php/rmiq/article/view/1755

Ingham, J., Dunn, I. J., Heinzle, E., Prenosil, J. E., & Snape, J. B. (2008). *Chemical Engineering Dynamics: An Introduction to Modelling and Computer Simulation* (Vol. 3). John Wiley & Sons.

Jones, E., Oliphant, T., Peterson, P., *et al.* (2001). SciPy: Open source scientific tools for Python. http://www.scipy.org/

Keilegavlen, E., Berge, R., Fumagalli, A., Starnoni, M., Stefansson, I., Varela, J., & Berre, I. (2021). Porepy: An open-source software for simulation of multiphysics processes in fractured porous media. *Computational Geosciences 25(1)*, 243-265. https://doi.org/0.1007/s10596-020-10002-5

Kister, H. Z., Haas, J. R., Hart, D. R., & Gill, D. R. (1992). *Distillation Design* (Vol. 1). McGraw-Hill New York.

Larsen, A. H., Mortensen, J. J., Blomqvist, J., Castelli, I. E., Christensen, R., Duak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., *et al.* (2017). The atomic simulation environment-a python library for working with atoms. *Journal of Physics: Condensed Matter 29(27)*, 273002. https://doi.org/10.1088/1361-648X/aa680e

Lee, J., & Ramirez, W. F. (1994). Optimal fed-batch control of induced foreign protein production by recombinant bacteria. *AIChE Journal 40(5)*, 899-907. https://doi.org/10.1002/aic.690400516

Luyben, W. L. (1989). *Process Modeling, Simulation and Control for Chemical Engineers*. McGraw-Hill Higher Education.

Medina-Leaños, R., Segovia-Hernandez, J., & Felix-Flores, M. (2020). Dynamic behavior of thermally coupled reactive distillation sequences for different operating conditions. *Revista Mexicana de Ingeniería Química 10(1)*, 147-160. http://www.rmiq.org/ojs311/index.php/rmiq/article/view/1640

Mendoza, D., & Riascos, C. (2020). Methodology for design, analysis and thermodynamic optimization of distillation columns with internal heat exchangers. *Revista Mexicana de Ingeniería Química 14(2)*, 523-542. http://www.rmiq.org/ojs311/index.php/rmiq/article/view/1285

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., . . . Scopatz, A. (2017). Sympy: Symbolic computing in python. *PeerJ Computer Science 3*, e103. https://doi.org/10.7717/peerj-cs.103

Johansson, F. *et al*. (2013). Mpmath: A Python library for arbitrary-precision floating-point arithmetic (version 0.18) [http://mpmath.org/].

Nikolić, D. D. (2016). Dae tools: Equation-based object-oriented modelling, simulation and optimisation software. *PeerJ Computer Science 2*, e54. https://doi.org/10.7717/peerjcs.54

Ochoa, S. (2016). A new approach for finding smooth optimal feeding profiles in fed-batch fermentations. *Biochemical Engineering Journal 105*, 177-188. https://doi.org/10.1016/j.bej.2015.09.004

Ogunnaike, B. A., & Ray, W. H. (1994). *Process Dynamics, Modeling, and Control* (Vol. 1). Oxford University Press New York.

Pfenninger, S., DeCarolis, J., Hirth, L., Quoilin, S., & Staffell, I. (2017). The importance of open data and software: Is energy research lagging behind? *Energy Policy 101*, 211-215. https://doi.org/10.1016/j.enpol.2016.11.046

Rocha, M., Mendes, R., Rocha, O., Rocha, I., & Ferreira, E. C. (2014). Optimization of fed-batch fermentation processes with bioinspired algorithms. *Expert Systems with Applications 41(5)*, 2186-2195. https://doi.org/10.1016/j.eswa.2013.09.017

Sánchez-Vargas, J., & Valdés-Parada, F. (2021). Multiscale modeling of a membrane bioreactor for the treatment of oil and grease rendering wastewaters. *Revista Mexicana de Ingeniería Química 20(2)*, 911-940. https://doi.org/10.24275/rmiq/Fen2368

Schwaab, M., Biscaia Jr, E. C., Monteiro, J. L., & Pinto, J. C. (2008). Nonlinear parameter estimation through particle swarm optimization. *Chemical Engineering Science 63(6)*, 1542-1552. https://doi.org/10.1016/j.ces.2007.11.024

Shacham, M., & Brauner, N. (2017). Solving a system of nonlinear algebraic equations you only get error messages-what to do next? *Chemical Engineering Education 51(2)*, 75-82.

Soares, R. d. P., & Secchi, A. (2003). Emso: A new environment for modelling, simulation and optimisation. *Computer Aided Chemical Engineering* (pp. 947-952). Elsevier. https://doi.org/10.1016/S1570-7946(03)80239-0

Tian, Y., Demirel, S. E., Hasan, M. F., & Pistikopoulos, E. N. (2018). An overview of process systems engineering approaches for process intensification: State of the art. *Chemical Engineering and Processing- Process Intensification 133*, 160-210. https://doi.org/10.1016/j.cep.2018.07.014

Tsay, C., Pattison, R. C., & Baldea, M. (2017). Equation-oriented simulation and optimization of process flowsheets incorporating detailed spiral-wound multistream heat exchanger models. *AIChE Journal 63(9)*, 3778-3789.

Tulsyan, A., & Barton, P. I. (2016). Perks: Software for parameter estimation in reaction kinetic systems. *Computer Aided Chemical Engineering* (pp. 25-30). Elsevier. https://doi.org/10.1016/B978-0-444-63428-3.50009-6

Tummers, B. (2006). Datathief iii. https://datathief.

Westerberg, A.W., Abbott, K., Allan, B., *et al*. (1994). Plans for ascend iv: Our next generation equational-based modeling environment. Carnegie Mellon University, Engineering Design Research Center.

Wu, W., Henao, C. A., & Maravelias, C. T. (2016). A superstructure representation, generation, and modeling framework for chemical process synthesis. *AIChE Journal 62(9)*, 3199-3214. https://doi.org/10.1002/aic.15300