Published in final edited form as:
Comput Appl Eng Educ. 2023 November ; 31(6): 1662–1677. doi:10.1002/cae.22660.

Using PharmaPy with Jupyter Notebook to teach digital design in pharmaceutical manufacturing

Daniel J. Laky[1,2], **Daniel Casas-Orozco**[1], **Mesfin Abdi**[3], **Xin Feng**[3], **Erin Wood**[3], **Gintaras V. Reklaitis**[1], **Zoltan K. Nagy**[1]

[1]Davidson School of Chemical Engineering, Purdue University, West Lafayette, Indiana, USA

[2]Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, Wisconsin, USA

[3]Office of Pharmaceutical Quality, Center for Drug Evaluation and Research, Food & Drug Administration, Silver Spring, Maryland, USA

Abstract

The use of digital tools in pharmaceutical manufacturing has gained traction over the past two decades. Whether supporting regulatory filings or attempting to modernize manufacturing processes to adopt new and quickly evolving Industry 4.0 standards, engineers entering the workforce must exhibit proficiency in modeling, simulation, optimization, data processing, and other digital analysis techniques. In this work, a course that addresses digital tools in pharmaceutical manufacturing for chemical engineers was adjusted to utilize a new tool, PharmaPy, instead of traditional chemical engineering simulation tools. Jupyter Notebook was utilized as an instructional and interactive environment to teach students to use PharmaPy, a new, open-source pharmaceutical manufacturing process simulator. Students were then surveyed to see if PharmaPy was able to meet the learning objectives of the course. During the semester, PharmaPy's model library was used to simulate both individual unit operations as well as multiunit pharmaceutical processes. Through the initial survey results, students indicated that: (i) through Jupyter Notebook, learning Python and PharmaPy was approachable from varied coding experience backgrounds and (ii) PharmaPy strengthened their understanding of pharmaceutical manufacturing through active pharmaceutical ingredient process design and development.

Keywords

digital design; engineering education; Jupyter Notebook; pharmaceutical manufacturing; Python; smart manufacturing

Correspondence Zoltan K. Nagy, Davidson School of Chemical Engineering, Purdue University, West Lafayette, IN 47906, USA. znagy@purdue.edu.

SUPPORTING INFORMATION
Additional supporting information can be found online in the Supporting Information section at the end of this article.

1 | INTRODUCTION

In recent years, the Industry 4.0 wave has accelerated the development of digital tools, data science and data analysis, and the use of the Internet of things to connect sensing and automation networks in the context of manufacturing [3]. Modern manufacturing facilities must develop smart, digital systems that can communicate with each other in real-time to effectively take advantage of the quickly evolving Industry 4.0 concepts. As industrial manufacturing facilities adopt Industry 4.0 and related paradigms, engineers entering the workforce will need to exhibit proficiency in both traditional manufacturing disciplines as well as competency in communicating with and utilizing digital tools, which presents newfound and persistent challenges for educators [14, 31]. One recent review of the scope of Industry 4.0 [16], particularly with a focus on the new responsibilities of educators and students themselves [23], provides great motivation for traditional engineering disciplines, such as chemical or mechanical engineering, to include dedicated data-centric and digitally focused material into both elective and core courses to modernize curriculum.

Education on the digital aspects of Industry 4.0, specifically those related to data science and machine learning, have been analyzed several times in recent literature. A tool that has consistently been found to be useful in digital education applications is the Jupyter Notebook, which provides a flexible and practical canvas to present graphics, code, mathematics, and instructional text in the same place [24]. For this reason, computational notebooks in digital education have been widely adopted throughout the data science and machine learning community. Examples of Jupyter Notebook are evident in Germany [4, 13, 17], Pakistan [20], and Portugal [6] for data science and engineering education with and without the use of graphical user interfaces. The book "Jupyter for Data Science" [28] provides interactive Jupyter Notebook examples alongside readings and technical material. Other areas of the world are following suit by confirming the need to implement digital manufacturing training programs along-side standard curriculum: for instance, in Azerbaijan [1] and the Netherlands [9]. Interestingly, [9] point out that the traditional call for modernization in curriculum primarily addresses engineering disciplines. However, in their work, they identify that the massive Dutch agriculture, food sciences, and plant sciences industries also require data science and other advanced manufacturing concepts to move the industry forward. They note that digitally focused courses and modernization to make these strides are lacking in life sciences disciplines as well. Although many components of Industry 4.0 are connected to data science or machine learning through data analysis, in this work, our focus is on: (i) the digital design of manufacturing facilities, (ii) the development of digital twins, and (iii) the use of other software tools during process design and monitoring.

There are some recent examples of chemical engineering departments across the United States and Canada that have been incorporating digital tools. Jupyter Notebook (both Python and Julia programming languages) and other computational notebooks were integrated into lectures, exercises, and projects to teach digital analysis techniques for optimization, applied statistics, and reactor design and control, at both undergraduate and graduate level [5]. The development of computational modules for courses in the areas of process control, process optimization, and experimental methods by utilizing a combination of commercial and open-source software tools were also discussed [31]. In Brazil, some examples include the development of digital tools for designing ideal chemical reactors [25], optimization of traditional manufacturing problems with mixed-integer programming formulations [26], and the deployment of other tools for optimization and design of chemical engineering systems [15, 29]. Throughout these developments, varying levels of coding experience are required for the students enrolled in the corresponding course. The focus in these efforts is on the development of teaching software or digital tools where the user is not required to have prior coding experience [25], to have little coding experience [26], or to have high levels of coding competency [15, 27].

Existing chemical engineering software has also seen recent innovations in the form of self-generating problems and codes. A tool that generates chemical engineering homework and test problems with varying inputs and outputs within a Jupyter Notebook framework was also reported in the literature [10], in which the option is given to completely hide the code (i.e., use only a graphical user interface to the tool and forgo looking at the source code), enabling students to explore the digitalization aspect of the educational tool in addition to completing the homework assignments. The modeling and optimization tool MOSAIC [11] has also been proposed to have teaching utility as the software automatically generates code in a chosen modeling format based on a user-friendly, object-oriented design [12], which showed encouraging educational results.

One important field closely related to chemical engineering is the manufacturing of pharmaceuticals. However, the tools that represent the traditional chemical industry and are a standard component in the education of chemical engineers (i.e., AspenPlus) require tailored approaches to adequately model a pharmaceutical manufacturing process. In small molecule oral drug processing, there is a diverse set of unit operations required to process raw materials into medicines that involve complex phase interactions including liquid-liquid, gas-liquid, and solid–liquid systems. Also, even with modernization of control and process intensification, pharmaceuticals are typically manufactured using batch processes. When batch processes are used in conjunction with continuous unit operations, or are simulated over extended campaigns, discontinuities associated with the start and stops of batch steps cause numerical difficulties that require custom-written simulation tools, or numerical simulation expertise. For these reasons, a digital tool for pharmaceutical manufacturing that can simulate batch, continuous, and hybrid (i.e., a process which contains both batch and continuous unit operations) operating modes addressed a major gap in the software space. Also, as digitalization continues, it is imperative to equip and educate engineers pursuing a career in pharmaceutical manufacturing with powerful numerical tools that are easy to use.

For these reasons, there has been a concerted effort at the University in West Lafayette, IN USA toward the development of a pharmaceutical process simulator, PharmaPy [7], to facilitate rapid in-silico design of manufacturing alternatives across batch, continuous, and hybrid operating modes. PharmaPy is written in Python and is an open-source tool encouraging wide-spread availability, transparency with respect to the internal unit operation model library, as well as community development of the tool. So far, the value in addressing research problems has been shown through a series of studies involving flowsheet simulation-optimization [7], economic comparison of the effect of manufacturing scale and operating mode on total cost [8], design space analysis [21], and derivative-based optimization strategies [22] using PharmaPy. However, the value of including PharmaPy in a teaching and education setting has not yet been demonstrated. Fortunately, given that PharmaPy is written in Python, it can readily be combined with the interactive teaching environment through Jupyter Notebook to provide the basis for teaching pharmaceutical manufacturing to both undergraduate and graduate students by exploring design, parameter estimation, and optimization of pharmaceutical processes. The accessibility of Python as a programming language and the format of the Jupyter Notebook, which may contain code, mathematics, figures, and interactive graphics in a coherent manner, provide an excellent framework for tutorials and self-led training modules [4-6, 10, 13, 17, 20, 24, 26, 28]. In this work, initial feedback on the tools used and learning format indicate that PharmaPy simulations within a Jupyter Notebook present a promising, open-source framework to teach digital design in pharmaceutical manufacturing.

The rest of this article is organized into six sections. In Section 2, we present a brief description of the course, and introduce the tools used (i.e., Jupyter Notebook and PharmaPy). Then, in Section 3, we present the course layout, some learning objectives of ChE 55300, and general survey topics and questions used to gather initial student feedback. In Section 4, we describe each project offered by the course in more detail. Section 5 details an example of the crystallization project described in Section 4.3 with some insight on student solutions. Then, Section 6 summarizes and analyzes results from the survey of the course. Finally, Section 7 concludes the work with discussion and future ideas for improving the tool, improving the educational experience, and expanding the influence of digital tools within the chemical engineering and manufacturing curriculum.

2 | COURSE AND TOOL OVERVIEW

In the School of Chemical Engineering at Purdue, there are technical elective course offerings that cover development of pharmaceutical products and processes, important operations related to pharmaceutical manufacturing such as reaction and crystallization, and digital analysis techniques that aid in adopting good manufacturing practices. ChE 55300 is one such course teaching *Pharmaceutical API Process Development and Design*.

The development and design of processes to produce pharmaceutical products involves three important tasks:

1. Translation of the recipe for the drug substance (or active pharmaceutical ingredient [API]) from a laboratory development recipe to a process that is, usable for manufacture.
2. Selection, preliminary design, and scale-up of equipment used to carry out the steps of the recipe for a given API.
3. Selection, preliminary design, and scale-up of equipment used to make the drug product (e.g., tablets or capsules), which is the vehicle for delivery of the API to the patient.

Typically, tasks 2 and 3 are considered as the drug substance and drug product manufacturing process, respectively. With this in mind, ChE 55300 primarily focuses on the drug substance side, considering the implementation of digital tools to perform model calibration (parameter estimation), design, optimization, and other digital analyses to complete tasks 1 and 2 successfully.

The course provides background information on regulatory aspects of pharmaceutical manufacturing, and introduction to the various unit operations relevant to the pharmaceutical manufacturing industry. Also, the course presents the benefits and drawbacks of batch, semibatch, and continuous operating modes for both drug substance and drug product processes. However, in this work, the focus will be on the digital design of the upstream drug substance unit operations (e.g., synthesis, separations, crystallization, filtration, drying) both on an individual basis and as a complete process flowsheet. There is work underway to address the operations relevant to drug product manufacture but is outside the scope of this paper.

PharmaPy was used to address digital process design and simulation throughout the course. Given the scope of the course, there is significant overlap between the principles used in the previously mentioned research and the currently available unit operations within PharmaPy (e.g., reactors, crystallizers, vaporization units, etc.) and the course objectives, as shown in Table 1. PharmaPy is written as an object-oriented tool, using an intuitive representation of unit operations and other necessary components, such as material phases or process kinetics, when constructing digital process models. When represented visually, the abstraction of object-oriented programming overlaps with traditional engineering schematics and process flowsheet diagrams commonly taught and used throughout undergraduate and graduate engineering curriculum. This promotes an intellectual connection between a physical unit operation, a schematic or drawing of that unit operation, and ultimately the digital representation of that unit operation through objects. PharmaPy is also open source, which allows students to look directly at model equation implementations to understand how the process model translates first principles equations to a digitally or numerically realizable simulation.

Object-oriented software is inherently modular, which also promotes key concepts in manufacturing. PharmaPy is no exception, giving the user capability in connecting both continuous and discontinuous unit operations together to form a complete process, spanning multiple processing steps. During the course, several projects were completed by the students to analyze unit operations both individually and as part of a complete manufacturing process. Early course projects covered the design of individual unit operations, including semibatch reaction, batch filtration, batch cooling crystallization, and continuous cooling crystallization. Then, the learnings from running standalone units were assembled into a final project, where students were asked to choose the best economical multiunit manufacturing route to produce a set amount of API, contrasting a fully continuous manufacturing process with an alternative manufacturing process that was not fully continuous (e.g., fully batch or hybrid batch-continuous). The final project allowed students to analyze and understand the overall impact of adjusting the operating mode of specified unit operations within the full process. This exercise demonstrates the modular nature of the architecture with respect to manufacturing, the flexibility of modeling tools that are capable of interchanging unit operations with minimal effort, and the impact of operating mode on product cost and efficacy.

Jupyter Notebooks represent an optimal environment for teaching PharmaPy, or any process design software tool. Within Jupyter Notebooks, one can render figures of a process schematic, mathematics of the process model equations, instructional text, and code using different cells in the same notebook, as shown in Figure 1. As stated previously, the direct comparison of the process schematic to the code side-by-side provides an important connection between technical understanding of the process and digital understanding of the process. The instructional text and mathematical expressions allow in-depth, self-guided tutorials and interactive sessions to be created and taught throughout any course, which employs Jupyter Notebooks. In this case, a Jupyter Notebook tutorial for each project was sent to the students before the class when the project was assigned, where one of those tutorial notebooks was completed during class as a follow-along exercise. Also, later in the course for the final project, the modular architecture of PharmaPy, and of pharmaceutical manufacturing as a whole, can be easily broken down between different cells of a notebook. This allows students to define each unit operation as an object in a separate block of code before combining the units together into a fully defined flowsheet. Overall, using Jupyter Notebooks provided a lens into the structure of PharmaPy and facilitated understanding of the modular architecture of many engineering and manufacturing problems.

To address the diversity of coding expertise throughout the student population, two dedicated lectures were given at the beginning of the course covering introduction to Python, the use of Anaconda for package control and package installation, and the general syntax of PharmaPy using both an integrated development environment (IDE) and Jupyter Notebooks. An example of the semibatch reactor tutorial notebook is shown in Figures 1 and 2. Here, Figure 1 demonstrates the inclusion of figures and mathematical equations in markdown format to give background information on the engineering problem and visually connect the model schematic, model equations, and the digital representation of that unit in the object-oriented PharmaPy syntax. Figure 2 continues the tutorial with partially filled coding blocks, where the students are led by an instructor to complete missing coding snippets during an interactive lecture.

In addition to single-unit operations, such as the semibatch reaction case study presented in Figures 1 and 2, students used PharmaPy to simulate a 3-unit pharmaceutical process containing chemical synthesis, crystallization, and filtration. In PharmaPy, single units are defined as objects. A flowsheet is also an object but can be more accurately described as a collection of those unit operations and connections between them. The flowsheet can then be simulated as a process with dynamic information transferring from unit to unit using a sequential-modular simulation scheme [18]. The modular architecture of PharmaPy is conceptually supported well by the format of Jupyter Notebooks. In a Jupyter Notebook, code can be divided into cells where a specific function is defined, or task is completed. During the final project, students were supplied with a Jupyter Notebook that defined a continuous synthesis-crystallization process ending with batch filtration. A portion of the notebook is shown below in Figure 3.

In Figure 3, the definition of the filtration unit is shown in an individual cell. The students are guided on the new concept of defining the units of a flowsheet. At this point, they must aggregate the units into the flowsheet object “flst” so the units can be accessed by the simulation executive when the flowsheet is to be simulated. The full Jupyter Notebook is available in Supporting Information, where the definitions of the reactor, crystallization, and dynamic holding tank units are shown. Once all units have been defined, the user must connect each unit to the respective downstream unit using a “Connection” object. As shown in Figure 3, there is an individual code cell to define unit connections for the flowsheet object.

Finally, arguments necessary to run the process must be passed to the simulator. These arguments are typically the runtime of the unit operation, and for volume-discretized units, or a number of discretization segments for the model. Additional arguments can also be passed to units, for instance options to the SUNDIALS integrator [2, 19], may be passed to improve numerical stability or behavior of the model. A time grid is passed to the dynamic holder in this case to discard the startup material in the first 3 h of operation. Finally, the pressure used in filter operation is supplied. Once all the arguments necessary to run each unit in the processing sequence are defined, they are compiled into a dictionary and passed to the simulation executive of the flowsheet to run the process. Selected results from the continuous simulation are shown with plots and printed values in the full Jupyter Notebook as shown in the supplemental material.

Since PharmaPy is written in Python, students were encouraged to use other Python packages to supplement their analyses. For instance, using Pandas for data management, SciPy’s optimization suite for derivative-free and derivative-based optimization, or scikit-learn for regression or data-driven regression or machine learning modeling. NumPy and SciPy are used throughout PharmaPy and were briefly introduced during the Python tutorial lectures at the beginning of the course. Anaconda provides package management capabilities through virtual environments, and the use of these environments was also taught and encouraged throughout completion of the projects.

3 | MATERIALS AND METHODS

During the fall semester of 2021, a cohort of 16 students who enrolled in ChE 55300, were taught using solely PharmaPy for simulation for the first time instead of utilizing combination of MATLAB, DynoChem, and Microsoft Excel. This course was offered as a cross-over undergraduate-graduate course, with five undergraduate students, seven professional master’s students, and four PhD students. Most of the students were chemical engineers, with one student coming from a biology and chemistry background, one student from materials engineering, and one indicating dual-focus on chemical engineering and computer science. Thirteen of the 16 students enrolled completed the survey.

The major goal of this study was to see if PharmaPy would be an adequate replacement for the previously used tools in ChE 55300. To this end, students were surveyed on the quality of PharmaPy, whether PharmaPy enhanced the learning experience or not, if Python was approachable as a coding language, and if they planned to use PharmaPy or Python in the future for digital analysis. Three majors learning objectives were also analyzed: (i) if the students were able to utilize new, digital tools in Python with or without previous experience, (ii) if the students understood pharmaceutical manufacturing in the context of digital simulation (in this case with PharmaPy), and (iii) if the students were able to adequately report and disseminate findings using completely digital tools. The survey questions focused on objectives (i) and (ii) as well as requesting feedback on PharmaPy to improve the software and improve the student experience within ChE 55300.

To gain insight into whether the software tool was adequate to teach pharmaceutical manufacturing, a survey was organized. Initial questions asked how approachable Jupyter Notebook and Python were during the course. It was also polled to determine where students went to get assistance with Python or PharmaPy (i.e., Google, the T.A.s of the course, etc.). Each of the projects described in Section 4 had survey questions that were specific to each project. Then, two free response questions were asked: (i) Do you have any suggestions to improve PharmaPy, or issues with PharmaPy that you had difficulty with, and (ii) Do you have any feedback or suggestions on PharmaPy's inclusion in the core curriculum? A final question was also added to ask whether they would be likely to use PharmaPy and Python for pharmaceutical modeling and general modeling and data analysis in the future, respectively.

4 | PROJECT DESCRIPTIONS

During ChE 55300, five projects, only four of them including unit operation simulation, were given to the students. Each simulation-oriented project was adapted from previous iterations of the course to be taught completely within PharmaPy. For the first four projects (each worth 12.5% of the final grade), students worked alone, wrote a report on their findings, and 25% of the students were asked to present their work during each project. For the final project (worth 25% of the final grade), students were allowed to choose a partner to write a final report and give a team presentation. For each project, the students were supplied with a tutorial Jupyter Notebook to guide them in how to use PharmaPy for the specific task. Also, the students were given one lecture on PharmaPy before each project, as mentioned in the previous section. The four projects where PharmaPy was used are described in more detail below.

4.1 | Semibatch reactor design

Chemical synthesis is commonly used in pharmaceutical manufacturing to produce an API. In this project, students were tasked with designing an optimal fed-batch reactor for a simple reaction system to create a desired product, species C. The reaction system also produced an undesired product, D. The students were given a few reactor variables to alter, and were asked to compare various feed profiles of the key reactant, species A. Given these parameters, students were asked to optimize the fed-batch reactor system using any strategy of their choosing but were given ideas of parameter sweep approaches (enumerative), and callback-based optimization approaches (systematic). Students were also introduced to multiobjective optimization during this project.

4.2 | Filtration parameter estimation and scale-up

In the filtration project, students were given experimental filtration data to determine specific cake resistance and the dependence of cake resistance on pressure drop. Then, with the newfound parameters, students were tasked with scaling up the filter from lab-scale to pilot-scale through digital analysis. Students were asked to analyze the effect of scale on the filter, as well as the effect of running simultaneous smaller batches to filter instead of filtering the whole volume in one go.

Students were given the option to perform parameter estimation wherever they saw fit, but capabilities using PharmaPy for parameter estimation were in place. Even with these capabilities, most students used Microsoft Excel to perform the parameter estimation and simulated the filter with PharmaPy.

4.3 | Crystallizer design comparison

In pharmaceutical manufacturing, separation of the API to an acceptable purity is an important processing step. In this project, students were introduced to the comparison of batch and continuous manufacturing modes for separating an API from a reaction mixture via crystallization. The comparison centered around the trade-offs between yield, mean crystal size, and other crystal size distribution attributes. Students were tasked with optimizing both the batch and continuous systems over the same set of questions and ultimately recommend which system, batch or continuous, should be used for crystallization. More information on student responses and the exact details of the crystallization project are included in Section 5.

4.4 | Flowsheet design comparison

The final project combined elements from each of the previous projects. Groups of two were given a fully continuous flowsheet to optimize, which included synthesis, crystallization, and filtration. The systems were identical to those used in previous projects, but were now not individually represented, but instead linked into a connected process. To further analyze the batch versus continuous trade-off, groups were assigned 1 of 4 flowsheets that were either fully batch, or hybrid processes. The groups were asked to minimize production cost while on a per-time basis. Then, groups were asked to reformulate the optimization of these flowsheets into constrained optimization problems. Here, the students were to minimize production costs while adhering to product guidelines, often referred to as critical quality attributes (CQAs) or critical operating constraints. As a final challenge, the students were given a small block of code that generated samples of the uncertainty space of the reaction kinetic parameters. The students were asked a series of questions to analyze the robustness of their optimal solutions.

5 | SAMPLE PROJECT RESPONSES

In this section, a selection of student solution methodologies and expected responses to the project described in section 4.3, where an API was to be purified via crystallization is presented. This project covered both batch and continuous crystallization systems, and the crystallization step(s) involved nucleation and growth kinetics. Students were asked to optimize a batch cooling crystallization process, as well as a continuous mixed suspension-mixed product removal (MSMPR) crystallization process with respect to yield (f_{yield}), crystal size (d_{avg}), or some combination of these two objectives, as shown below in the below equation:

$$\min_{T(t),\, \tau} \; w_1(d_{\text{avg}}) + w_2(f_{\text{yield}}), \tag{1}$$

where w_1 and w_2 are objective function weights for mean size, d_{avg}, and yield, f_{yield}, respectively. The decision variables are $T(t)$, which is a temperature profile in the batch case and constant temperature in the continuous case, and τ, which is the batch time for the batch case and residence time for the continuous case. The only requirement the students had in terms of software was to utilize PharmaPy for crystallization simulations while completing the project. The methodology for optimization was left completely to the student. Throughout the course, grid-based optimization and parameter sweep approaches were presented to students, and the use of common packages with black-box optimization capabilities, for instance, SciPy's optimize package [30], were also suggested to the students.

With this in mind, most students designed a small number of crystallizer configurations, or organized a parameter sweep to choose the best design configuration over a discrete set of operating conditions. These methods were presented as potential derivative-free solution methodologies during the course. However, some did venture outside these methodologies and decided to use SciPy's optimization suite, for instance using a differential evolutionary global optimization algorithm to design a cooling profile. In the batch case, the temperature profile and batch time were the operating conditions for the optimal design problem. Examples of standard cooling profiles, simulated as piecewise linear cooling trajectories, are shown below in Figure 4. Using these profiles, many students found the optimal cooling profile for crystal size was similar to the exponential cooling profile.

The students also optimized conditions for the continuous MSMPR crystallizer. In this case, students used both a parameter sweep approach and a SciPy optimization method to find the optimal operating temperature and residence time of the unit. All students were asked to plot the average crystal size over time to clarify how the startup period in the MSMPR to reach some constant mean crystal size could impact production. An example of that result is shown below in Figure 5. As seen, the average crystal size does not reach a steady value until approximately 1500 s into operation.

However, no students identified that the crystal size distribution may still be changing even though the average crystal size, a statistic calculated by the ratio between the first and zeroth moments of the crystal size distribution, has reached a relatively steady value. As shown in Figure 6, even after 4000 s, the distribution still has some remaining bimodality from the initial nucleation event, which is flushed out of the system after sufficient operating time (after 7000 s).

Overall, students had mixed opinions on which operating mode was better. Depending on what scenarios were chosen for parameter sweep or grid-based optimization, the solutions varied slightly with respect to optimal size. Some students chose the method, which produced the largest crystals, whereas a few students did an analysis on the production rates of both methods, discovering that even with the startup period, the continuous MSMPR crystallizer would very quickly outproduce the batch system with the fixed unit sizes for this problem. Therefore, the slightly smaller crystal sizes seen with continuous crystallization could be accepted because the productivity of the continuous crystallizer greatly outweighed that of the batch case. This is just a small, but representative, insight into the variation of student responses over many projects and concepts tested throughout the course. It should be noted that individual unit operation optimization can result in suboptimal solutions for multiunit flowsheets. This fact was explained during the course lectures and was explored during the final project.

6 | INITIAL STUDENT FEEDBACK

Of the cohort of 16 students, 13 students responded to the survey. These 13 students were predominantly chemical engineers entering the course with varying levels of coding expertise. Previous coding experience was surveyed, with 5 of 13 respondents indicating they had the minimum coding experience required to complete their previous core coursework, and the remaining eight indicating they had some intermediate coding experience. As seen in Figure 7, with respect to Python approachability, the distribution of responses ranged from somewhat difficult to very easy to use, with the most common response being that Python was somewhat easy to use. Also shown in the figure, previous coding experience indicated a higher likelihood that Python was approachable for this small group of students. To mitigate difficulty with Python and improve students' agency over the programming language, the instructors made a deliberate effort throughout the semester to encourage self-sufficiency in handling syntax errors during coding. From the 13 responses, 11 found help online through Google and StackExchange, and of those 11, 10 listed Google or StackExchange as the first source of help. Of course, there were some issues that required instructor assistance, which was also reported in 9 of the 13 responses. However, only one student indicated the instructors as the first source of assistance.

Regarding student feedback on PharmaPy and Jupyter Notebooks in general, Table 2 summarizes the responses from students on whether PharmaPy was helpful for understanding course material, and more broadly API process development. Also, all students agreed or strongly agreed that Jupyter Notebooks played a large role in their success throughout the course.

For PharmaPy, 10 of the 13 students agreed that the software helped in understanding pharmaceutical API process development, which would be considered a major goal for the course. Very similar results are seen for PharmaPy adding value beyond the core technical lecture materials, with 12 of 13 students agreeing. However, here one student did disagree, indicating that there is still room for improvement. A total of nine responses to written student feedback were given to the first free response question (Do you have any suggestions to improve PharmaPy, or issues with PharmaPy that you had difficulty with?). Of those nine, select responses from three students are listed below:

> Mostly just have better notation and comments for what certain lines of code do. Other than that, there were not any major issues I noticed that were from using PharmaPy itself.
>
> Definitely create documentation for everything, as multiple times I found myself going through the source code just to understand syntax.
>
> I think having very well-documented information about function calls would be helpful. For example, units can be hard to determine if the parameter is somewhat generic.

The major theme throughout the written feedback was the constructive criticism that documentation was lacking, leading to difficulty understanding exactly what the process models were doing, or what units of measure were required for kinetic constants as inputs. At the time PharmaPy was used in the Fall 2021 semester, detailed documentation was not yet available. These student responses encouraged the developers to dedicate more resources to create more comprehensive documentation of the software for the initial public release.

As shown in Figure 8, more than 80% of the students agreed that each project benefitted their understanding of pharmaceutical manufacturing, another major course goal. The semibatch reactor design appeared to be less accepted than the other projects, which may be attributed to it being the first project using PharmaPy and Python. However, all of the projects were well-received, and students felt that they learned about digital modeling and pharmaceutical manufacturing by completing these projects.

The second free response question (In terms of ChE55300, do you have any feedback or suggestions on PharmaPy's inclusion in the core curriculum?) saw only six responses. Of those 6 responses, three are listed below.

> I think it suited the course very well and I thought the projects were very intellectually stimulating.
>
> I thought PharmaPy was a decent tool to understand pharma manufacturing and optimization.
>
> Would be nice to consider a pharmapy intensive mini course or lectures (videos, in person) to help those with little to none coding experience.

Given these responses and others about Python proficiency, the mention of Python coding in the syllabus and some prerecorded mini lectures on using PharmaPy and Python in general are two aspects that will be added to improve ChE 55300 for future course offerings.

As shown in Table 3, most students responded that they would be likely or very likely to use PharmaPy and Python in the future. However, five students were neutral or unlikely to use PharmaPy in the future. As mentioned previously, the development of documentation and fine-tuning of tutorial materials likely would increase the number of students who would indicate to use PharmaPy in the future, and a study over multiple semesters of

CHE55300 student responses could identify if this initial semester was more difficult than future semesters for that reason.

7 | DISCUSSION

PharmaPy had been shown to be a powerful tool in research settings in the past [7, 8, 21, 22], but now has been successful in the classroom as well. PharmaPy was originally written to provide an open-source modeling library that addressed both batch and continuous pharmaceutical modeling simultaneously. Subsequently, the dissemination of these materials to the public provided a unique opportunity to modernize a course in pharmaceutical modeling at Purdue that successfully encouraged many students to consider using software tools in future work. Jupyter Notebooks were already known to succeed in the engineering education community [4-6, 10, 13, 17, 20, 24, 26, 28], and this education application with PharmaPy follows that trend. Also, the idea of embedding software tools within each other in a Pythonic way was taught throughout the course. This represents a unique opportunity for students to weave together multiple tools for future analyses, which shows the utility of such a flexible software framework in an engineering setting. One important note is that the tools that come with a graphical user interface, such as the one discussed in [27], provide a future direction for PharmaPy that could lower the threshold for some users to pursue pharmaceutical modeling. As [27] reports, 96% of the students thought REAJA was user-friendly, which identifies that through future course offerings, there is still room for improvement with Python/PharmaPy's 62% user-friendliness. However, the balance between coding autonomy (i.e., the ability to design and execute complex analysis frameworks without workarounds) and user-friendliness cannot be ignored. This trade-off may provide interesting future research directions for software tools used for teaching modeling to users with varying coding experience, especially in engineering settings.

8 | CONCLUSIONS

This was the first time PharmaPy was used in the classroom. Most students in ChE 55300 found that using PharmaPy with Jupyter Notebook was approachable. Even further, most students found that Jupyter Notebook was a key part of success in their understanding of the software and of digital modeling. As shown in previous educational studies, the responses from this survey indicate that computational notebooks represent an excellent teaching tool for students with varying coding experience. These notebooks provide a mechanism for both student-led learning and instructor-led interactive lectures to combine learning coding and digital tools with technical manufacturing curriculum.

Also, exploring and adopting new digital tools and programming languages for digital analysis is an important skill considering the quick pace at which Industry 4.0 is evolving. Although searching for help with coding error messages, accessing material outside of lecture to aid in Python development, or digging through source code to understand a software tool are obvious to those experienced with coding, these do-it-yourself problem-solving skills can be intimidating for students when learning new programming languages; especially when learning their first or second programming language. As described in the results, instructors bear some responsibility to foster this skill in students. Instructors should encourage students to identify resources that can help their digital proficiency by themselves, while also encouraging direct interaction for problems that cannot be resolved through these do-it-yourself routes.

With this preliminary study, the survey results indicate that PharmaPy and Python are a promising alternative to traditional modeling and computation tools used within chemical engineering to teach digital design within pharmaceutical manufacturing. Students were able to utilize PharmaPy with little or no experience with Python. They also indicated that PharmaPy helped them understand and explore the digital space for pharmaceutical manufacturing. Finally, when compared with the previous course offering without PharmaPy (Spring semester 2020), there was no clear trend of student performance change. One possible inference is that PharmaPy is not overly difficult for students to use and is a reasonable modern replacement to the traditional tools used in the course, but further studies with other cohorts of students should be analyzed for a stronger conclusion. Even with this success, there is still room for improvement. There are plans to have in-depth documentation on public release to demystify user errors when developing digital pharmaceutical processes within PharmaPy, which would address thematic feedback from students. Also, utilizing PharmaPy over more offerings of the course would be helpful to solidify this initial feedback that PharmaPy is a promising tool for not only research, but education, in pharmaceutical manufacturing processes.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

ACKNOWLEDGMENTS

This project was supported by the United States Food and Drug Administration through grant U01FD006738. The views expressed by the authors do not necessarily reflect official policies of the Department of Health and Human Services; nor does any mention of trade names, commercial practices, or organization imply endorsement by the United States Government.

Funding information

United States Food & Drug Administration, Grant/Award Number: U01FD006738

AUTHOR BIOGRAPHIES

Daniel J. Laky obtained his BEng in Chemical and Biomolecular Engineering from Vanderbilt University in 2016, and his PhD in Chemical engineering from Purdue University in 2022. He currently holds a position as a postdoctoral researcher at the University of Wisconsin-Madison. His research interests include the nexus of chemical engineering, applied mathematics, and computer science through applications in pharmaceutical modeling, software design, computer science education, and data science.

Daniel Casas-Orozco received his PhD in Chemical Engineering in 2018 from Universidad de Antioquia (Colombia), where he developed frameworks to analyze batch processing of upgraded essential oils. In 2019, he joined Purdue University as a postdoctoral researcher, where he used his abilities in applied mathematics and scientific computing, dynamic simulation, first principles modeling, and software and systems engineering in the design and optimization of pharmaceutical processes. During his appointment at Purdue University, he was the main developer of PharmaPy, which encapsulates all his expertise and that of his team members in the analysis of pharmaceutical manufacturing via digital frameworks.

Mesfin Abdi is a senior pharmaceutical reviewer in FDA. He holds a PhD in Chemical Engineering from Dresden University of Technology (TUD). His research interest includes implementation of innovative and advanced manufacturing technologies in pharmaceuticals.

Dr. Xin Feng holds a PhD degree in Pharmaceutical Sciences, and he serves as a Staff Fellow in the US FDA, Office Pharmaceutical Quality. His research focuses on formulation and manufacturing process development of complex drug products.

Erin Wood obtained her PhD in Mechanical Engineering from the University of Vermont and is currently at the FDA within CDER's Office of Pharmaceutical Quality serving as a senior research scientist. Her research interests include advanced manufacturing, nanotechnology, and complex drug formulations.

Gintaras V. Reklaitis is Burton and Kathryn Gedge Distinguished Professor of Chemical Engineering in the Davidson School of Chemical Engineering at Purdue University in West Lafayette, IN, USA. He received his BS from Illinois Institute of Technology (1965), and his MS (1969) and PhD (1969) from Stanford University. His current research interests include applications of process systems methodology to improve pharmaceutical product design, development, manufacture and administration, especially in applications which have not been introduced to process systems methodology.

Zoltan K. Nagy is a Professor of Chemical Engineering in the School of Chemical Engineering at Purdue University, US. He obtained his BS (1994), MSc (1995), and PhD (2001) degrees from the "Babes-Bolyai" University of Cluj, Romania. His research interests include process control, process intensification and digital design of smart manufacturing systems with applications in pharmaceutical, food, fine chemical, and energetic industries.

DATA AVAILABILITY STATEMENT

The results of the questionnaires and Jupyter notebooks used or analyzed during the current study are available from the corresponding author upon reasonable request.

REFERENCES

1. Ahadov A, Asgarov ES, and El-Thalji I, A summary of adapting industry 4.0 vision into engineering education in Azerbaijan, IOP Conf. Ser.: Mater. Sci. Eng. 700. IOP Publishing Ltd, Bristol, England, 2019.
2. Andersson C, Führer C, and Åkesson J, Assimulo: a unified framework for ode solvers, Math. Comput. Simul. 116 (2015), 26–43.
3. Arden NS, Fisher AC, Tyner K, Yu LX, Lee SL, and Kopcha M, Industry 4.0 for pharmaceutical manufacturing: Preparing for the smart factories of the future,Int. J. Pharm. 602 (2021).
4. Biehler R, Fleischer Y, Budde L, Frischemeier D, Gerstenberger D, Podworny S, and Schulte C, Data science education in secondary schools: Teaching and learning decision trees with codap and Jupyter Notebooks as an example of integrating machine learning into statistics education. (Proc. Roundtable Conf. Int. Assoc. Statis. Educ., Voorborg, The Netherlands, 2020.
5. Boukouvala F, Dowling A, Verrett J, Ulissi Z, and Zavala V, Computational notebooks in chemical engineering curricula, Chem. Eng. Educ. 54 (2020), 143–150.
6. Cardoso A, Leitão J, and Teixeira C, Using the Jupyter Notebook as a tool to support the teaching and learning processes in engineering courses, (Auer M. and Tsiatsos T, eds.), The Challenges of the Digital Transformation in Education. ICL 2018, Advances in Intelligent Systems Computer, 917. Springer, Cham, Switzerland, 2019, pp. 227–236.

7. Casas-Orozco D, Laky DJ, Laird CD, Reklaitis GV, Wang V, Abdi M, Feng X, Wood E, and Nagy ZK, Pharmapy: an object-oriented tool for the development of hybrid pharmaceutical processes, Comp. Chem. Eng. 153 (2021). 10.1016/j.compchemeng.2021.107408

8. Casas-Orozco D, Laky DJ, Wang V, Abdi M, Feng X, Wood E, Reklaitis GV, and Nagy ZK, Techno-economic analysis of dynamic, end-to-end optimal pharmaceutical campaign manufacturing using PharmaPy, AIChE J. 1 (2023), e18142. 10.1002/aic.18142

9. Catal C. and Tekinerdogan B, Aligning education for the life sciences domain to support digitalization and industry 4.0, Proc. Comput. Sci. 158. (2019), 99–106.

10. Domínguez JC, Alonso MV, González EJ, Guijarro MI, Miranda R, Oliet M, Rigual V, Toledo JM, Villar-Chavero MM, and Yustos P, Teaching chemical engineering using Jupyter Notebook: problem generators and lecturing tools, Educ. Chem. Eng. 37 (2021), 1–10.

11. Esche E, Hoffmann C, Illner M, Müller D, Fillinger S, Tolksdorf G, Bonart H, Wozny G, and Repke JU, Mosaic enabling large-scale equation-based flow sheet optimization, Chemie-Ingenieur-Technik 89 (2017), 620–635.

12. Esche E, Tolksdorf G, Fillinger S, Bonart H, Wozny G, and Repke JU (2017b). Support of education in process simulation and optimization via language independent modelling and versatile code generation. (Symp. Comp. Aided Process Eng. ESCAPE 27), October 1st–5th, Barcelona, Spain. pp. 2929–2934.

13. Frischemeier D, Biehler R, Podworny S, and Budde L, A first introduction to data science education in secondary schools: teaching and learning about data exploration with codap using survey data, Teach. Statis. 43 (2021), S182–S189.

14. Giffi C, McNelly J, Dollar B, Carrick G, Drew M, and Gangula B, The skills gap in U.S. manufacturing 2015 and beyond, Deloitte (2015).

15. Henrique JP, de Sousa R, Secchi AR, Ravagnani MASS, and Costa CBB, Optimization of chemical engineering problems with EMSO software, Comp. Appl. Eng. Educ. 26 (2018), 141–161.

16. Hernandez-de-Menendez M, Escobar Díaz CA, and Morales-Menendez R, Engineering education for smart 4.0 technology: a review, Int. J. Interact. Design Manufact. 14 (2020), 789–803.

17. Herta C, Voigt B, Baumann P, Strohmenger K, Jansen C, Fischer O, Zhang G, and Hufnagel P. (2019). Deep teaching: Materials for teaching machine and deep learning. (5th Int. Conf. Higher Educ. Adv. HEAd '19) Universitat Politecnica de Valencia. pp. 1153–1161.

18. Hillestad M. and Hertzberg T, Dynamic simulation of chemical engineering systems by the sequential modular approach, Comput. Chem. Eng. 10 (1986), 377–388.

19. Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, and Woodward CS, Sundials: suite of nonlinear and differential/algebraic equation solvers, ACM Trans. Math. Softw. 31 (2005), no. 3, 363–396.

20. Hussain Z. and Khan MS, Introducing Python programming for engineering scholars, IJCSNS Int. J. Comp. Sci. Netw. Security 18 (2018), 26–33.

21. Laky D, Casas-Orozco D, Laird CD, Reklaitis GV, and Nagy ZK, Simulation-optimization framework for the digital design of pharmaceutical processes using pyomo and pharmacy, Industrial and Engineering Chemistry Research, 2022.

22. Laky D, Casas-Orozco D, Rossi F, Mackey JS, Reklaitis GV, and Nagy ZK, Determination of probabilistic design spaces in the hybrid manufacture of an active pharmaceutical ingredient using the python-based framework pharmacy. (Proc. 14th Int'l Symp of PSE. June 19th–23rd, Kyoto, Japan. Computer Aided Chemical Engineering), Elsevier, 2022.th

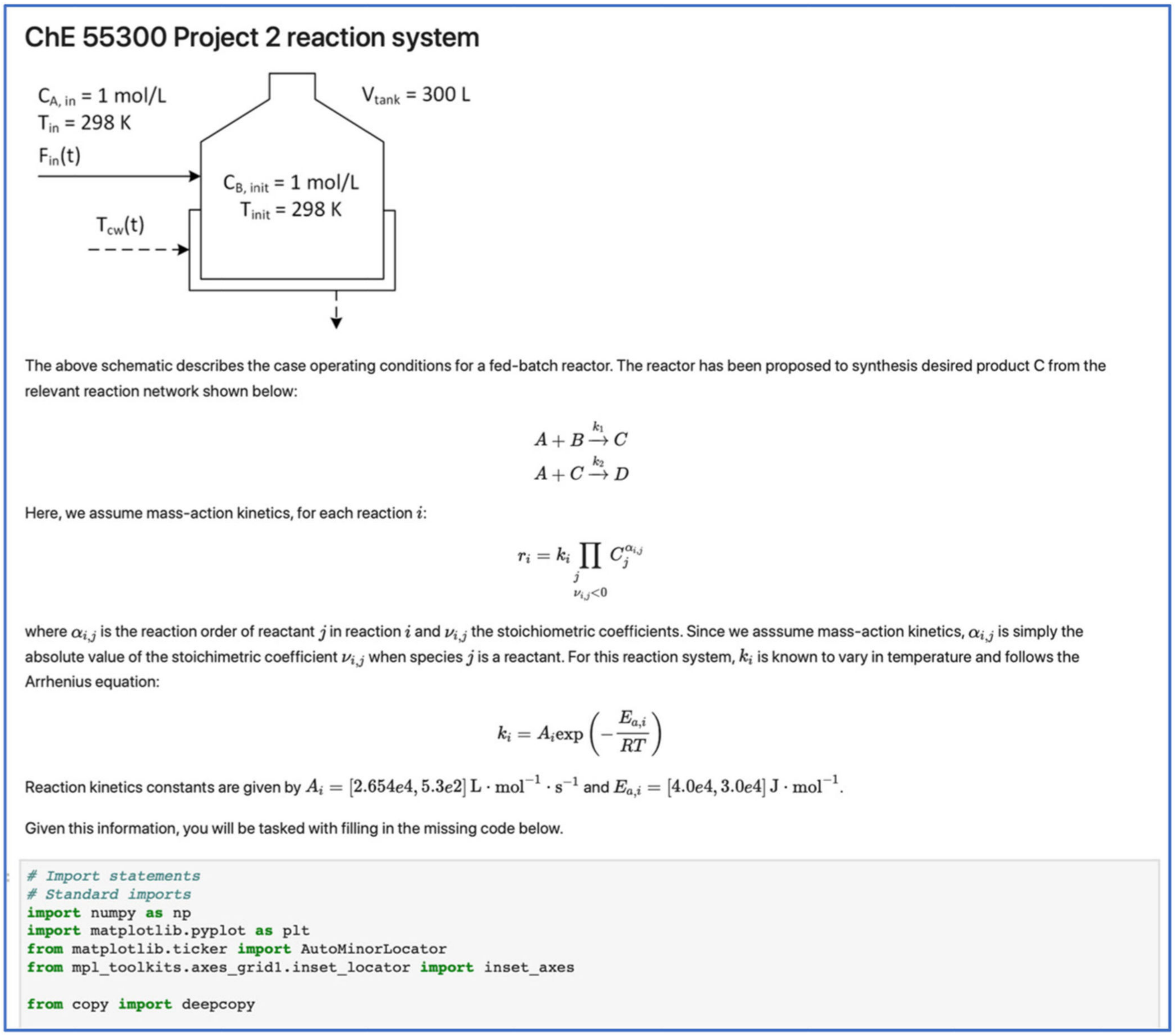
23. Mogoş R-I, Bodea C-N, Dascălu M-I, Safonkina O, Lazarou E, Trifan E-L, and Nemoianu IV, Technology enhanced learning for industry 4.0 engineering education, Rev. Roum. Sci. Techn. Électrotechn. Énerg 63 (2018), no. 4, 429–435.

24. Perez F. and Granger BE, Project Jupyter: Computational narratives as the engine of collaborative data science, 2015.

25. Sawaki RV, Tannous K, and Filho JBF, Development of an educational tool aimed at designing ideal chemical reactors, Comp. Appl. Eng. Educ. 28 (2020), 459–476.

26. Suárez A, Alvarez-Feijoo MA, Fernández González R, and Arce E, Teaching optimization of manufacturing problems via code components of a Jupyter notebook, Comp. Appl. Eng. Educ. 26 (2018), 1102–1110.

27. Teles dos Santos M, Vianna AS Jr. and Le Roux GAC, Programming skills in the industry 4.0: are chemical engineering students able to face new problems? Educ. Chem. Eng. 22 (2018), 69–76.
28. Toomey D, Jupyter for data science, Packt Publishing, Burmingham, UK, 2017.
29. Turcu CO and Elena C, Industrial internet of things as a challenge for higher education, Int. J. Adv. Comp. Sci. Appl. 9 (2018), no. 11, 55–60.
30. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ,Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, Vijaykumar A, Bardelli AP, Rothberg A, Hilboll A, Kloeckner A, Scopatz A, Lee A, Rokem A, Woods CN, Fulton C, Masson C, Häggström C, Fitzgerald C, Nicholson DA, Hagen DR, Pasechnik DV, Olivetti E, Martin E, Wieser E, Silva F, Lenders F, Wilhelm F, Young G, Price GA, Ingold GL, Allen GE, Lee GR, Audren H, Probst I, Dietrich JP, Silterra J, Webber JT, Slavič J, Nothman J, Buchner J, Kulick J, Schönberger JL, de Miranda Cardoso JV, Reimer J, Harrington J, Rodríguez JLC, Nunez-Iglesias J, Kuczynski J, Tritz K, Thoma M, Newville M, Kümmerer M, Bolingbroke M, Tartre M, Pak M, Smith NJ, Nowaczyk N, Shebanov N, Pavlyk O, Brodtkorb PA, Lee P, McGibbon RT, Feldbauer R, Lewis S, Tygier S, Sievert S, Vigna S, Peterson S, More S, Pudlik T, Oshima T, Pingel TJ, Robitaille TP, Spura T, Jones TR, Cera T, Leslie T, Zito T, Krauss T, Upadhyay U, Halchenko YO, and Vázquez-Baeza Y, Scipy 1.0: fundamental algorithms for scientific computing in python, Nat. Methods 17 (2020), 261–272. [PubMed: 32015543]
31. Yenkie KM, Enhanced undergraduate learning through integration of theory and computational tools, Chem. Eng. Educ. 54 (2020), 129–136.



ChE 55300 Project 2 reaction system

The above schematic describes the case operating conditions for a fed-batch reactor. The reactor has been proposed to synthesis desired product C from the relevant reaction network shown below:

$$A + B \xrightarrow{k_1} C$$
$$A + C \xrightarrow{k_2} D$$

Here, we assume mass-action kinetics, for each reaction i:

$$r_i = k_i \prod_{\substack{j \\ \nu_{i,j}<0}} C_j^{\alpha_{i,j}}$$

where $\alpha_{i,j}$ is the reaction order of reactant j in reaction i and $\nu_{i,j}$ the stoichiometric coefficients. Since we asssume mass-action kinetics, $\alpha_{i,j}$ is simply the absolute value of the stoichimetric coefficient $\nu_{i,j}$ when species j is a reactant. For this reaction system, k_i is known to vary in temperature and follows the Arrhenius equation:

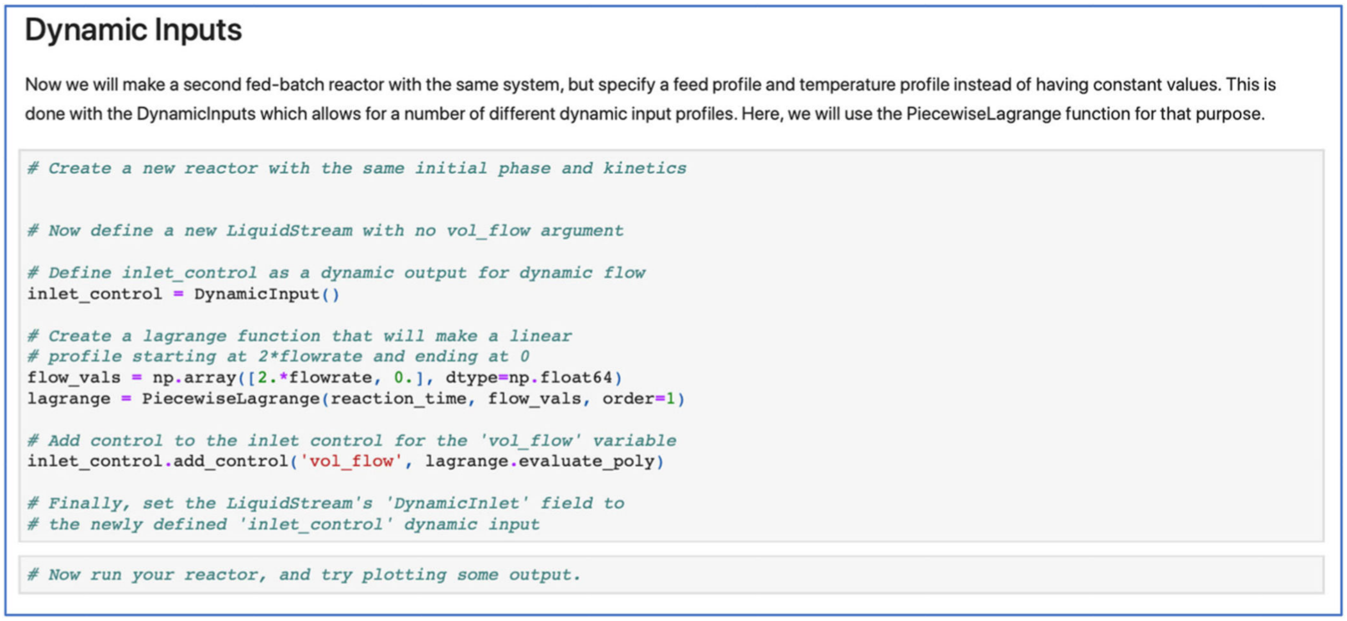
$$k_i = A_i \exp\left(-\frac{E_{a,i}}{RT}\right)$$

Reaction kinetics constants are given by $A_i = [2.654e4, 5.3e2]\ \mathrm{L \cdot mol^{-1} \cdot s^{-1}}$ and $E_{a,i} = [4.0e4, 3.0e4]\ \mathrm{J \cdot mol^{-1}}$.

Given this information, you will be tasked with filling in the missing code below.

```
# Import statements
# Standard imports
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
from mpl_toolkits.axes_grid1.inset_locator import inset_axes

from copy import deepcopy
```

FIGURE 1.
Introductory material for the Jupyter Notebook tutorial on semibatch reactor design. Schematic, reaction schemes, mathematics, textual instructions, and code all rendered in one place. Jupyter Notebook mt'l (1)—Schematic, Chemistry, Math Equations, Code.



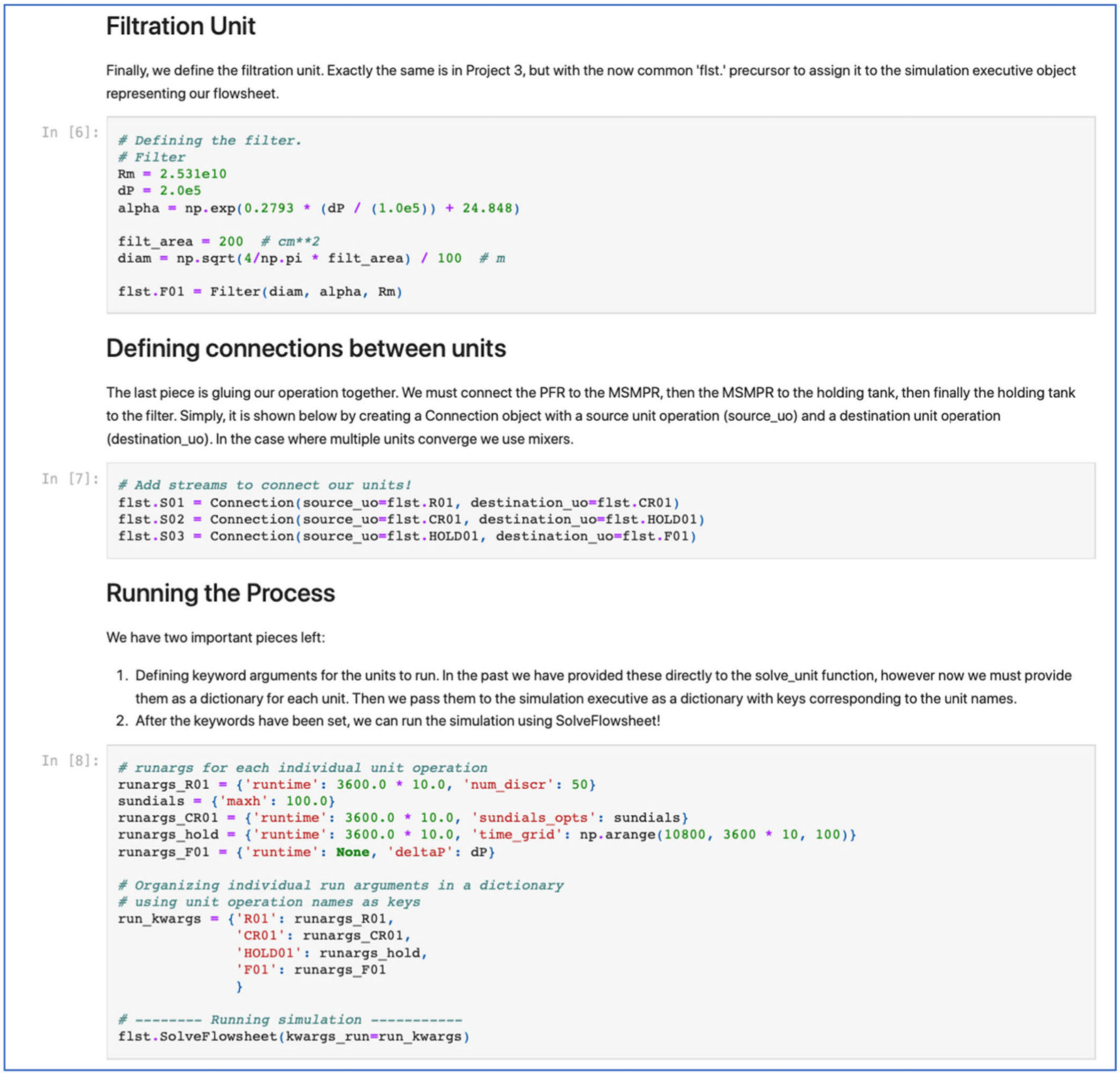
Dynamic Inputs

Now we will make a second fed-batch reactor with the same system, but specify a feed profile and temperature profile instead of having constant values. This is done with the DynamicInputs which allows for a number of different dynamic input profiles. Here, we will use the PiecewiseLagrange function for that purpose.

```
# Create a new reactor with the same initial phase and kinetics


# Now define a new LiquidStream with no vol_flow argument

# Define inlet_control as a dynamic output for dynamic flow
inlet_control = DynamicInput()

# Create a lagrange function that will make a linear
# profile starting at 2*flowrate and ending at 0
flow_vals = np.array([2.*flowrate, 0.], dtype=np.float64)
lagrange = PiecewiseLagrange(reaction_time, flow_vals, order=1)

# Add control to the inlet control for the 'vol_flow' variable
inlet_control.add_control('vol_flow', lagrange.evaluate_poly)

# Finally, set the LiquidStream's 'DynamicInlet' field to
# the newly defined 'inlet_control' dynamic input
```

```
# Now run your reactor, and try plotting some output.
```

FIGURE 2.
Continuation of the tutorial shown in Figure 1 with a focus on coding blocks. This material was taken from the interactive lecture on semibatch reactor design, with an instructor guiding the students on how to fill in the missing coding pieces, for example, shown here is the definition of a dynamic inlet feed for the semibatch reactor unit. Jupyter Notebook mt'l (2)—Fill-in-the-blank coding example.



Filtration Unit

Finally, we define the filtration unit. Exactly the same is in Project 3, but with the now common 'flst.' precursor to assign it to the simulation executive object representing our flowsheet.

In [6]:
```
# Defining the filter.
# Filter
Rm = 2.531e10
dP = 2.0e5
alpha = np.exp(0.2793 * (dP / (1.0e5)) + 24.848)

filt_area = 200  # cm**2
diam = np.sqrt(4/np.pi * filt_area) / 100  # m

flst.F01 = Filter(diam, alpha, Rm)
```

Defining connections between units

The last piece is gluing our operation together. We must connect the PFR to the MSMPR, then the MSMPR to the holding tank, then finally the holding tank to the filter. Simply, it is shown below by creating a Connection object with a source unit operation (source_uo) and a destination unit operation (destination_uo). In the case where multiple units converge we use mixers.

In [7]:
```
# Add streams to connect our units!
flst.S01 = Connection(source_uo=flst.R01, destination_uo=flst.CR01)
flst.S02 = Connection(source_uo=flst.CR01, destination_uo=flst.HOLD01)
flst.S03 = Connection(source_uo=flst.HOLD01, destination_uo=flst.F01)
```

Running the Process

We have two important pieces left:

1. Defining keyword arguments for the units to run. In the past we have provided these directly to the solve_unit function, however now we must provide them as a dictionary for each unit. Then we pass them to the simulation executive as a dictionary with keys corresponding to the unit names.
2. After the keywords have been set, we can run the simulation using SolveFlowsheet!

In [8]:
```
# runargs for each individual unit operation
runargs_R01 = {'runtime': 3600.0 * 10.0, 'num_discr': 50}
sundials = {'maxh': 100.0}
runargs_CR01 = {'runtime': 3600.0 * 10.0, 'sundials_opts': sundials}
runargs_hold = {'runtime': 3600.0 * 10.0, 'time_grid': np.arange(10800, 3600 * 10, 100)}
runargs_F01 = {'runtime': None, 'deltaP': dP}

# Organizing individual run arguments in a dictionary
# using unit operation names as keys
run_kwargs = {'R01': runargs_R01,
              'CR01': runargs_CR01,
              'HOLD01': runargs_hold,
              'F01': runargs_F01
              }

# -------- Running simulation -----------
flst.SolveFlowsheet(kwargs_run=run_kwargs)
```

FIGURE 3.
Portion of a Jupyter Notebook tutorial for a continuous synthesis-crystallization process with batch filtration. Code for individual unit definition is kept in individual coding cells, promoting modular architecture. Jupyter Notebook mt'l (3)—More in-depth code, unit-operation definition.

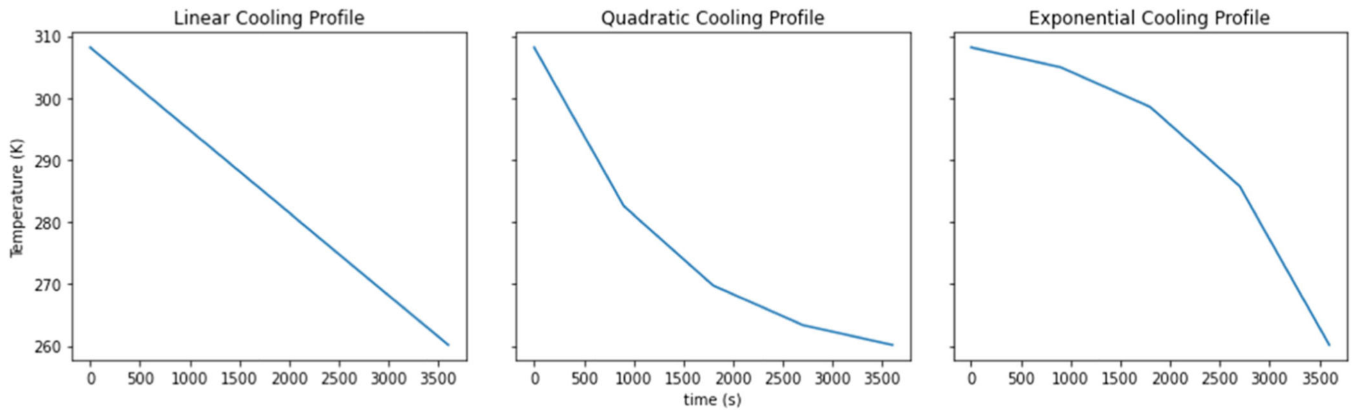


FIGURE 4.

Various piecewise linear cooling profiles for crystallization operation. Cooling profiles to consider for Project 4 optimization.

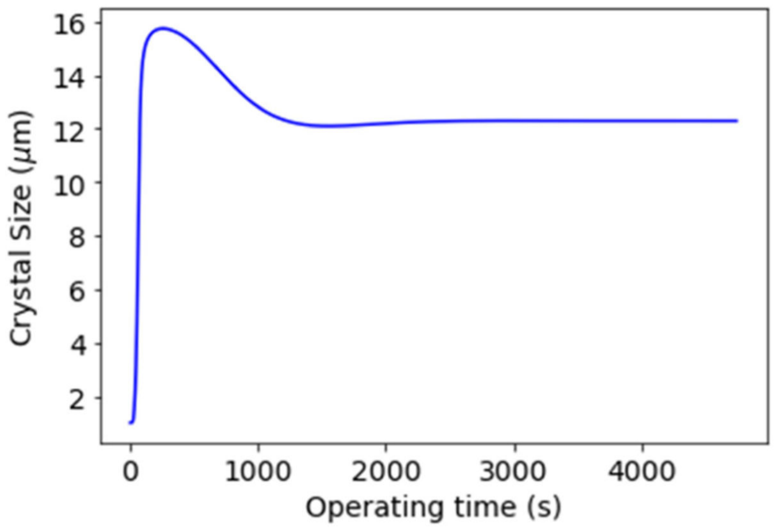


FIGURE 5.
Average crystal size during operation of the mixed suspension-mixed product removal (MSMPR) crystallizer. Crystallizer dynamic mean size profile.

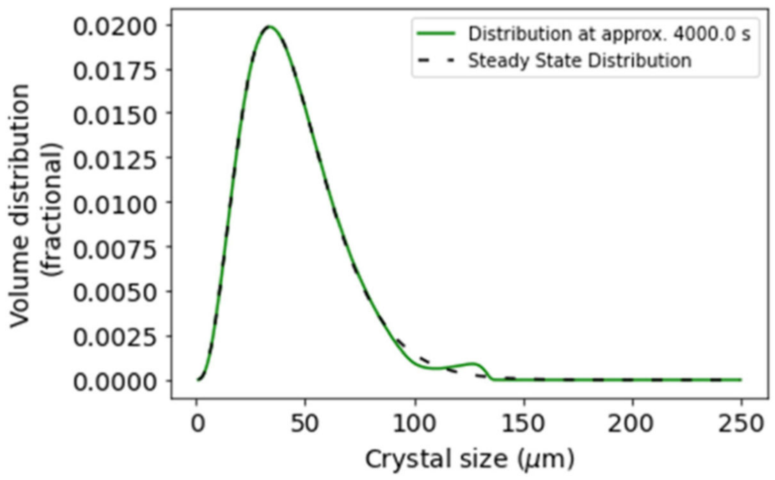
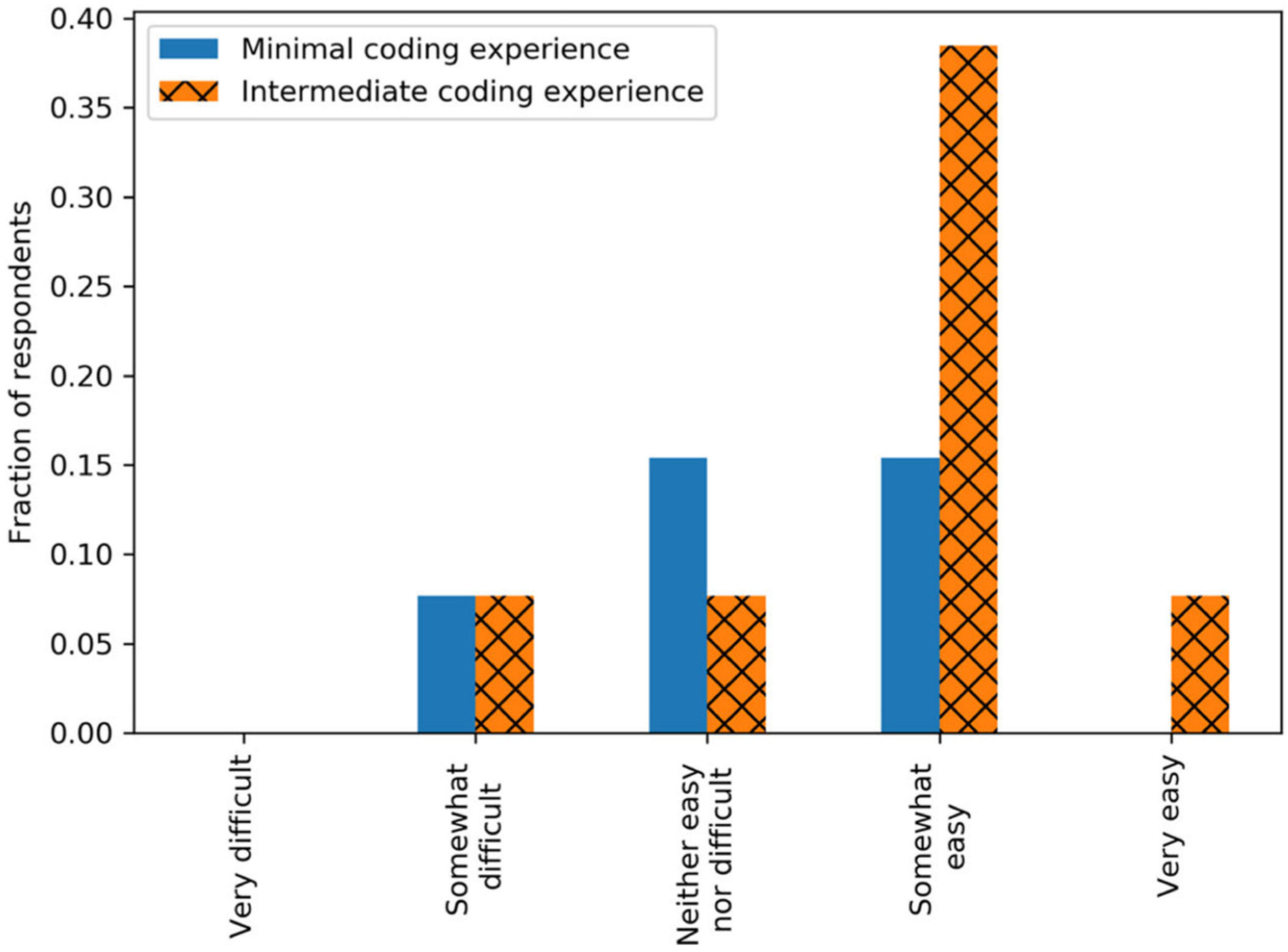


FIGURE 6.
Crystal size distribution in the mixed suspension-mixed product removal (MSMPR) after 4000 s, and after it has reached a steady state with respect to crystal size distribution. Crystal size distribution at steady state for the optimal solution.



FIGU RE 7.

Students' responses on the approachability of Python with previous coding experience indicated. Student feedback (1)—previous coding experience.

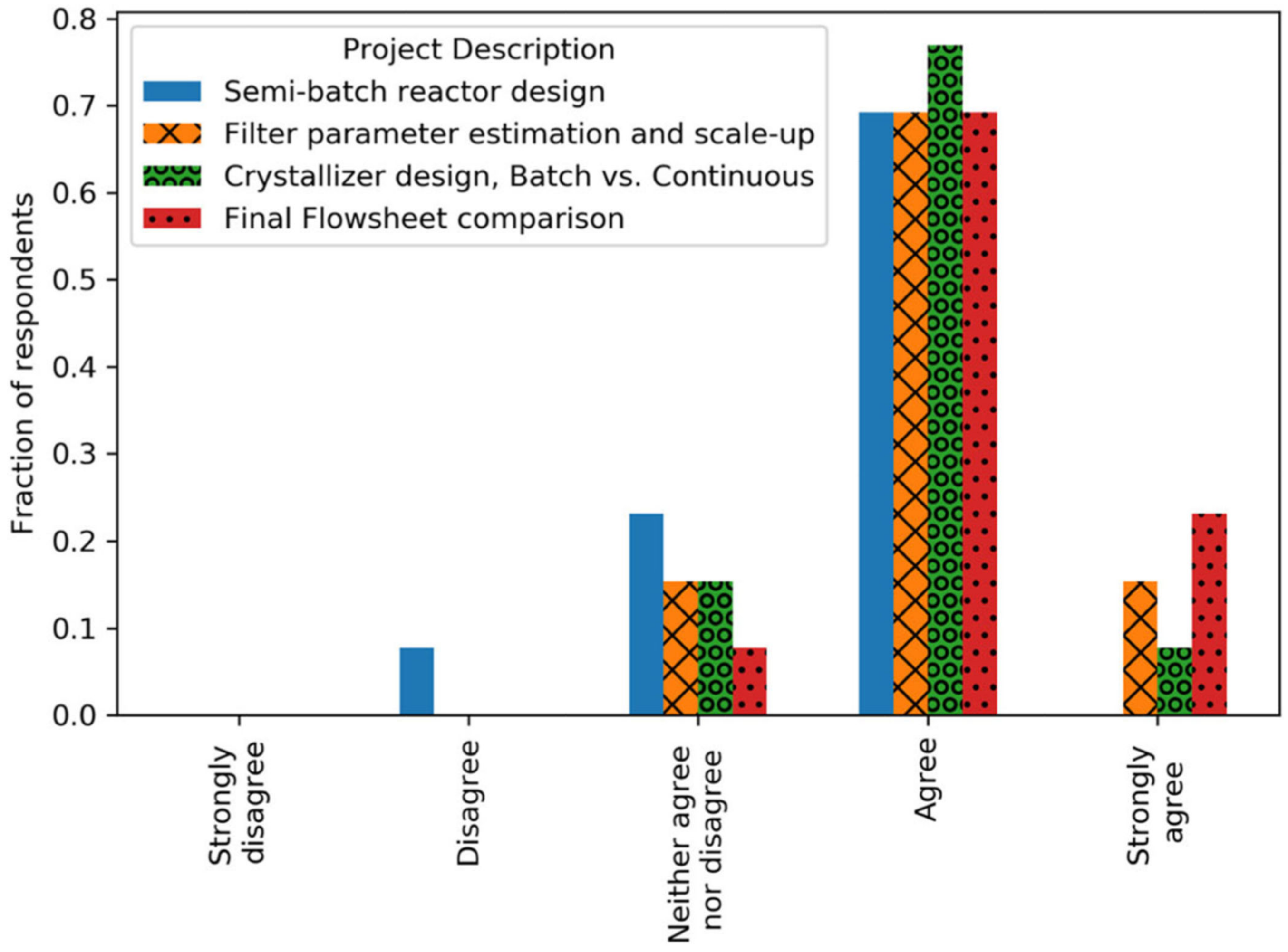


FIGURE 8.

Students' responses on whether the specified project helped them better understand pharmaceutical manufacturing. From left-to-right: (i) semibatch reactor design filtration parameter estimation and scale-up crystallizer design (iv) final reactor-crystallizer-filter flowsheet comparison. Student feedback (2)—interest with each project involving PharmaPy.

TABLE 1

Overlap between (i) ChE55300 course content and objectives and (ii) research using PharmaPy.

ChE55300 course content	Research using PharmaPy
Synthesis unit operations	Casas-Orozco et al. [7, 8], Laky et al. [21, 22]
Crystallization unit operations	Casas-Orozco et al. [7, 8], Laky et al. [21]
Filtration unit operations	Casas-Orozco et al. [7, 8], Laky et al. [21]
API process design and optimization	Casas-Orozco et al. [7, 8], Laky et al. [21, 22]
Scale-up and equipment selection	Casas-Orozco et al. [8], Laky et al. [21]

TABLE 2

Students' responses for whether PharmaPy and Jupyter were useful during the course and added value.

Question	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Jupyter Notebook tutorial materials were pivotal to your success during the course	0	0	0	9	4
PharmaPy helped in understanding pharmaceutical API process development	0	0	3	10	0
PharmaPy added value beyond the core curriculum	0	1	0	11	1

TABLE 3

Students' responses for how likely they would be to use PharmaPy and Python in the future.

In the future, how likely would you be to:	Very unlikely	Unlikely	Neutral	Likely	Very likely
Use PharmaPy as a tool for pharmaceutical modeling	0	2	3	7	1
Use Python as a language for modeling and data analysis	0	1	1	8	3