2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023

The Harbor of Engineering
Education for 130 Years

ASEE

Paper ID #38671

Numerical Problem Solving across the Curriculum with Python and MATLAB Using Interactive Coding Templates: A Workshop for Chemical Engineering Faculty

Austin N. Johns, The State University of New York, Buffalo

Austin N. Johns is an active-duty captain and developmental engineer in the United States Air Force. In 2017, he earned a B.S. in Chemical Engineering from Oklahoma State University. In 2023, he earned a M.S. in Chemical Engineering from the University at Buffalo, The State University of New York. His graduate research focused on developing computational educational resources for use in the chemical engineering curriculum. The views expressed in this paper are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

Dr. Robert P. Hesketh, Rowan University

Robert Hesketh is a Professor of Chemical Engineering at Rowan University. He received his B.S. in 1982 from the University of Illinois and his Ph.D. from the University of Delaware in 1987. After his Ph.D. he conducted research at the University of Cam

Prof. Matthew D. Stuber, University of Connecticut

Dr. Matt Stuber is an Assistant Professor with the Dept. of Chemical & Biomolecular Engineering and the Institute for Advanced Systems Engineering at the University of Connecticut. He received his PhD from the Massachusetts Institute of Technology (MIT) and his BS from the University of Minnesota – Twin cities, both in chemical engineering. In his post-doctoral work, he cofounded a water-tech start-up company focusing on developing flexible high-efficiency solar-driven desalination technologies for diverse applications where membrane technologies prove inadequate. At UConn, his core research focus is on optimization theory, methods, and software for modeling and simulation, robust simulation and design, and controls and operations. His application interests lie in addressing challenging and timely applications from a spectrum of industries including food, energy, water and natural resources, chemicals, finance, and healthcare. The systems-level thinking combined with quantitative rigor enables the development of novel solutions to emerging and intractable problems across these diverse areas.

Dr. Ashlee N. Ford Versypt, The State University of New York, Buffalo

Dr. Ashlee N. Ford Versypt is an Associate Professor in the Department of Chemical and Biological Engineering at the University at Buffalo (UB), The State University of New York. She is also an Affiliated Faculty in the Department of Engineering Education and the Institute for Artificial Intelligence and Data Science. She received her B.S. from the University of Oklahoma and her M.S. and Ph.D. from the University of Illinois. She did a postdoc at the Massachusetts Institute of Technology before starting her academic career at Oklahoma State University (OSU), where she was an assistant professor 2014-2020 and then a tenured associate professor until January 2021 before moving to UB. Dr. Ford Versypt leads the Systems Biomedicine and Pharmaceutics Laboratory. She was the 2020-2021 Chair for the ASEE Chemical Engineering Division (CHED). Dr. Ford Versypt has been recognized with the NSF CAREER Award, ASEE CHED Ray W. Fahien Award and Joseph J. Martin Award, and AIChE CAST Division David Himmelblau Award for Innovations in Computer-Based Chemical Engineering Education. She is an Academic Trustee of Computer Aids for Chemical Engineering Corporation.

Numerical Problem Solving across the Curriculum with Python and MATLAB Using Interactive Coding Templates: A Workshop for Chemical Engineering Faculty

Abstract

With the fourth industrial revolution well underway, the proportion of occupations requiring "high" or "medium" digital skills has never been greater. Among those most in demand are engineers skilled in computing and advanced problem solving to support the ongoing digitalization, networking, and automation. A numerical analysis course in the core undergraduate engineering curriculum is a natural place for students to learn numerical methods for advanced problem solving across engineering applications. The use of computing across the entire chemical engineering curriculum also offers opportunities to hone students' abilities as computational thinkers and effective problem solvers to meet the current and future needs of an increasingly complex and digital industry and society. While the current chemical engineering curriculum includes computational training, there is a need to efficiently increase the exposure of students to computing within mathematical problem-solving contexts and develop their proficiency in computer programming, all while balancing demands to reduce credit hours. Some chemical engineering faculty interested in enhancing the computational nature of their courses face a barrier to doing so due to unfamiliarity with some modern computational educational resources that may not have been covered in their training or may not be used in their research areas. The authors developed a workshop to teach chemical engineering faculty to use and develop interactive coding templates (MATLAB Live Scripts and Jupyter Notebooks) and to equip faculty to incorporate these techniques across the undergraduate curriculum. The workshop was presented at the 2022 ASEE/AIChE Summer School for Engineering Faculty. The purpose of this paper is to disseminate the workshop resources, providing educators with a suite of interactive templates focused on chemical engineering-related case studies and with training to create and adapt their own related materials. The paper details the interactive coding templates provided during the workshop along with the relevant pedagogical background and some lessons learned for future related workshops. Educators who did not attend the workshop are also a target audience of this paper as it provides tips and access to the relevant materials for implementing computational thinking through interactive coding templates into their classroom practices.

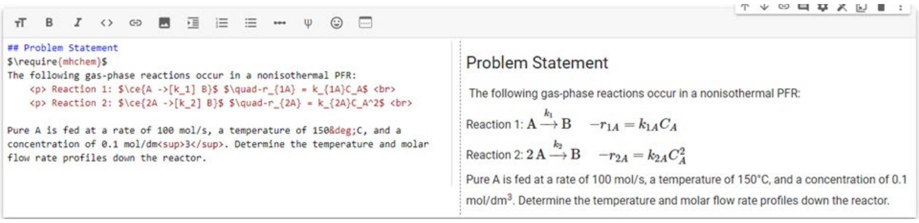
Introduction

We developed a workshop for the 2022 ASEE/AIChE Summer School for Engineering Faculty to address the need for training chemical engineering faculty in modern computational techniques, with the goal of equipping faculty to incorporate these techniques into the undergraduate chemical engineering curriculum. This paper is about both the workshop and the resources we created [1] and curated (Table 1). We have made these resources available for others to reproduce the workshop and for educators to learn from the materials and the expanded descriptions of them in this paper even if they did not have the opportunity to attend the workshop. The workshop was titled: "Numerical Problem Solving across the Curriculum with Python and MATLAB Using Interactive Coding Templates".

A recent survey of chemical engineers in industry and academia indicated that MATLAB and Python are the most commonly taught programming languages in chemical engineering curriculums and the most commonly used programming languages in industry [2]. The report on the survey results indicated a gap in students' ability to use computer software and programming to solve engineering problems [2]. Additionally, these comments suggested that one of the primary reasons that faculty forgo the use of computer aids is a lack of training and time to learn how to use these tools [2]. The growing shift towards open source software over proprietary licensed software is also driving many chemical engineering departments and educators to consider shifting historical use of MATLAB in education to emerging high-level programming languages such as Python that are open source [2, 3]. These findings and software use trends provided the impetus for creating this workshop.

We provided workshop materials for MATLAB and Python, partly due to the survey results [2], our own familiarity with the software for engineering problem solving, and informal discussions with other engineering educators. MATLAB and Python both support creation of interactive documents (via MATLAB Live Scripts and Jupyter Notebooks, respectively) that combine code and explanatory text, formatted equations, images, and code outputs into single files. We refer to the MATLAB Live Scripts and Jupyter Notebooks that we created for the workshop as "interactive coding templates."

Our rationale for using interactive coding templates over traditional scripts (i.e., MATLAB .m and Python .py files) is based on their demonstrated pedagogical value in teaching science and engineering students in an engaging and organized manner that promotes learning [4-21]. A key element of the interactive coding templates is the use of literate programming. Literate programming was coined by Donald Knuth, an American computer scientist and developer of the TeX document formatting language, as a programming methodology that elevates the documentation of programs to the level of literature [22]. A literate program explains in human-readable language what it is asking a computer to do rather than simply instructing the computer to do it, which ideally results in a program with documentation superior to one using traditional commenting [22]. This is typically accomplished by interweaving blocks of formatted text and blocks of code that execute the functions of the program [23]. MATLAB Live Scripts and Jupyter Notebooks both follow this paradigm and allow users to create code blocks using MATLAB and Python, respectively, and to incorporate text, equations, and visuals using formatting menu options or a markup language (a human-readable set of computer markings to define parts of a document or formatting of the text, e.g., see Figure 1). Our interactive coding templates were designed to merge elements of traditional teaching aids (e.g., a slide-based lecture, document-based assignment, or textbook content) with codes that can be run by users to interact with the educational materials and visualize the results, all in the same file. MATLAB Live Scripts run MATLAB code through MathWorks software. Jupyter Notebooks support running Python code and several other programming languages (including recently MATLAB [24]) through open source software. Because of literate programming, the interactive coding templates facilitate use of computers for problem solving or exploration of parameter impacts on models without extensive programming skills [25] on the part of the students or educators who reuse the materials we curated for the workshop. The templates can be adapted for purposes where programming is an explicit learning objective, and the text, visual elements, and/or codes in the templates can be edited and customized by other educators or students.



```
## Problem Statement
$\require{mhchem}$
The following gas-phase reactions occur in a nonisothermal PFR:
    <p> Reaction 1: $\ce{A ->[k_1] B}$ $\quad-r_{1A} = k_{1A}C_A$ <br>
    <p> Reaction 2: $\ce{2A ->[k_2] B}$ $\quad-r_{2A} = k_{2A}C_A^2$ <br>

Pure A is fed at a rate of 100 mol/s, a temperature of 150&deg;C, and a
concentration of 0.1 mol/dm<sup>3</sup>. Determine the temperature and molar
flow rate profiles down the reactor.
```

Problem Statement

The following gas-phase reactions occur in a nonisothermal PFR:

Reaction 1: $\mathrm{A} \xrightarrow{k_1} \mathrm{B} \quad -r_{1A} = k_{1A}C_A$

Reaction 2: $2\,\mathrm{A} \xrightarrow{k_2} \mathrm{B} \quad -r_{2A} = k_{2A}C_A^2$

Pure A is fed at a rate of 100 mol/s, a temperature of 150°C, and a concentration of 0.1 mol/dm^3. Determine the temperature and molar flow rate profiles down the reactor.

Figure 1: Jupyter Notebook text formatting example (in Google Colab environment).
Left: formatting menu options appear across the top, and markup language and LaTeX equation editing syntax are combined with plain text. Right: the output formatted text.

Computational thinking is the pedagogical basis of our workshop. Computational thinking refers to the cognitive processes involved in using computers to solve problems, including but not limited to problem reformulation, recursion/algorithmic thinking, problem decomposition, abstraction, pattern recognition, data visualization and analysis, selection of tools, systematic testing/debugging, automation, and computational modeling [26-29]. These cognitive processes are useful for learning programming, and they can also be used for improving problem solving more generally with or without a computer. Computational thinking is being widely emphasized across PreK-12 and higher education to prepare all students for the digital skills demands of future jobs across all disciplines and society more broadly [29-34]. Our emphasis in this paper is on providing materials to foster computational thinking in the chemical engineering undergraduate curriculum; others have developed computational thinking assessments [35-39] that may also be of interest to readers who adopt our materials in their classrooms. Three "computational practices" apply computational thinking for problem solving [34]: (1) automating procedures and processes, (2) using models to understand systems, and (3) collecting, analyzing, and communicating data. These are all relevant practices for chemical engineering undergraduates to learn and apply. Our interactive coding templates exemplify the first two of these computational practices. Other data science educational tools provide nice lessons related to the third [40-42].

Workshop materials

We developed nine interactive coding templates for the workshop. Seven templates are available as both MATLAB Live Scripts and Jupyter Notebooks to accommodate the software preference of the workshop participants; the other two are available as MATLAB Live Scripts only. No prior experience with MATLAB, Python, or Jupyter Notebooks is required to open and run the templates. Customization of the code blocks in the templates requires some familiarity with MATLAB or Python coding. The interactive coding templates include examples from material and energy balances, fluid mechanics, heat transfer, separations, thermodynamics, and reaction engineering. These examples showcase several educational use cases including lecture notes with in-class activities, pre-class readings with embedded activities, worked case studies, and homework problems. These examples also feature a diverse array of computational techniques including solving algebraic systems of equations, ordinary differential equations, and parameter estimation. We provided tutorials on how to create interactive coding templates; the tutorials are MATLAB Live Scripts and Jupyter Notebooks, further modeling the flexibility of these instructional tools. We also created a YouTube video introducing literate programming and highlighting the key features of MATLAB Live Scripts and Jupyter Notebooks [43].

The following workshop materials are available in a dedicated GitHub repository [1]:

0. Pre-Workshop Set-Up

   Instructions for downloading .mlx files and .ipynb files from Google Drive and for opening them in MATLAB, MATLAB Online, or Google Colab. The instructions were deemed necessary as workshop participants were expected to bring their own devices to the workshop rather than using preconfigured devices. We provided this document to workshop participants a week prior to the Chemical Engineering Summer School to prepare them to access the workshop's Google Drive folder, how to acquire a MATLAB license, how to open and edit MATLAB Live Scripts using the local MATLAB client and MATLAB Online, and how to open and edit Jupyter Notebooks using Google Colab. This consideration also drove us to provide web-based tools to open and edit MATLAB Live Scripts and Jupyter Notebooks to minimize the workshop's system requirements. The instructions include screen captures of the webpages and applications that may need to be updated in the future if any changes are made to the interfaces for Google Colab or MATLAB Online or to update the source where the files are shared with participants if outside a Google Drive.

1. Workshop Files Table of Contents

2. Workshop Objectives & Overview

3. Numerical Problem Solving Workshop

   Workshop instructional slide deck available in pdf and pptx formats.

4. Additional Resources

   Spreadsheet of related computational resources with hyperlinks, which are elaborated in Table 1 at the end of this manuscript. This serves as our literature review of collections of codes relevant to the workshop that may be useful for educators to extend beyond the interactive coding templates we provided.

5. Interactive Coding Template

   These are the main files intended to be shared for the workshop and are detailed further below.

6. Shareable Handout Numerical Problem Solving across the Curriculum with Python and MATLAB Using Interactive Coding Templates

   This pdf document is intended to serve as a handout with all workshop materials directly embedded in/attached to this file, making it shareable as one file to avoid potentially broken links in the future.

Interactive coding templates

This section includes a brief description of each of the interactive coding templates that we developed. This section is organized by the type of numerical method used, except the starting tutorials on how to create MATLAB Live Scripts and Jupyter Notebooks. Some numerical methods have more than one template. In addition to the numerical techniques used for each template, the chemical engineering applications are described, and information as to how the template is designed to be used by students is provided. Nine interactive coding templates and one tutorial are MATLAB Live Scripts. Six of these MATLAB Live Scripts and the tutorial are duplicated as Jupyter Notebooks to allow workshop participants to choose their preferred programming language. Additional solution files are available for four of the MATLAB Live Scripts and two of the Jupyter Notebooks.

- MATLAB Live Script and Jupyter Notebook Tutorials

  M0_HowToCreate.mlx & J0_HowToCreate.ipynb
  The files cover how to create MATLAB Live Scripts and use Google Colab to create Jupyter Notebooks, respectively.

- Linear Equations

  Template 1: M1_MassBalance.mlx (separate solution file available)
  The case study in this MATLAB Live Script is adapted from [44] and is designed to be used by students as a homework problem. The case study involves solving a system of linear mass balances on a reaction and separation system with recycle by formatting the mass balances as a linear algebraic system in matrix-vector form and subsequently using the Gauss elimination algorithm to solve the linear system.

- Nonlinear Equations

  Template 1: M2_NonlinearSystems.mlx (separate solution file available)
  The worked examples and case study in this MATLAB Live Script are adapted from [44]. The template contains three worked numerical methods examples and a chemical engineering case study that requires students to input a set of initial guesses to solve the problem. The case study involves solving a system of nonlinear mass balances on a reaction and separation system with recycle. Newton's method, Picard's method, and Newton-Raphson's method of solving systems of nonlinear equations are the numerical techniques used in the worked examples and case study. The template is designed to be used as an interactive textbook.

  Template 2: M3_PipeNetwork.mlx & J3_PipeNetwork.ipynb
  The case study in this MATLAB Live Script and Jupyter Notebook is adapted from Problem 8.11 from [45]. The case study models the behavior of water flowing through a pipe system. The template is designed to be used as in-class activity or case study for part

"a" and as a homework problem for part "b" and part "c". The case study involves solving a system of nonlinear equations using a built-in nonlinear equation solver.

- Ordinary Differential Equations (ODEs): Initial Value Problems

  Template 1: M4_NonisothermalPFR.mlx & J4_NonisothermalPFR.ipynb (separate solution files available)
  The case study in this MATLAB Live Script and Jupyter Notebook is adapted from Example 12-5 from [46]. The case study models the behavior of parallel reactions in a nonisothermal plug flow reactor (PFR). The template is designed to be used by students as a worked example or case study for part "a" and as a homework problem or in-class exercise for part "b" and part "c". The case study involves solving a system of first-order ODEs using a given set of initial conditions and a built-in ODE solver.

  Template 2: M5_ParEstKinetics.mlx & J5_ParEstKinetics.ipynb (separate solution files available)
  The case study and homework problem in this MATLAB Live Script and Jupyter Notebook include equations, illustrations, and data from [47]. The template is designed to be used by students as a worked example or case study for the first problem and then as a homework problem for the second problem. The case study and homework problem involve fitting reaction rate constants to data for the kinetics of a fluidized catalytic cracker. The solution requires parameter estimation using built-in curve fitting functions applied to a dynamic model involving multiple ODEs.

  Template 3: M6_TankDrainage.mlx & J6_TankDrainage.ipynb
  The case study in this MATLAB Live Script and Jupyter Notebook is adapted from Example 10.2-1 from [48]. The case study models the mass balance of a water tank using a system of ODEs. The template is designed to be used a case study or interactive textbook and involves solving a system of ODEs using a built-in ODE solver. The template ends with an extension of the case study that can be used as homework problem or in-class example.

- ODEs: Boundary Value Problems

  Template 1: M7_StefanTubeDiffn.mlx & J7_StefanTubeDiffn.ipynb
  The case study in this MATLAB Live Script and Jupyter Notebook is adapted from Problem 10.1 from [45]. The case study models diffusion in a Stefan tube as a second order ODE with split boundary conditions. Particular attention is given to how a student might generate and iterate on a set of initial guesses using the secant method or automated code when determining the split boundary conditions. The template is designed to be used as a worked example or case study and ends with an extension of the case study that can be used as homework problem or in-class example.

  Template 2: M8_LaminarPipe.mlx & J8_LaminarPipe.ipynb
  The case study in this MATLAB Live Script and Jupyter Notebook is adapted from Problem 8.1 from [45]. The case study models laminar flow in a horizontal pipe as a second order ODE with split boundary conditions. Particular attention is given to how a

student might generate and iterate on a set of initial guesses when determining the split boundary conditions. The template is designed to be used as a worked example or case study and ends with an extension of the case study that can be used as homework problem or in-class example.

- Partial Differential Equations

  Template 1: M9_HeatTransfer.mlx (separate solution file available)
  The problem worked in this MATLAB Live Script is adapted from [44]. The template is designed to be used as a worked example, interactive textbook, or in-class activity and asks students to modify the value of specific variables to explore the behavior of the system. The template models heat transfer as a partial differential equation initial boundary value problem. The method of lines and a built-in ODE solver are used to determine the solution to the problem.

Workshop activities and experiences

Learning objectives

By the end of this workshop, participants will be able to:

- Create interactive coding templates (MATLAB Live Scripts or Jupyter Notebooks using Python) for teaching chemical engineering concepts and problem solving
- Select, run, and interact with a MATLAB Live Script or Jupyter Notebook template applied to a chemical engineering topic of their choice

Mini-lecture

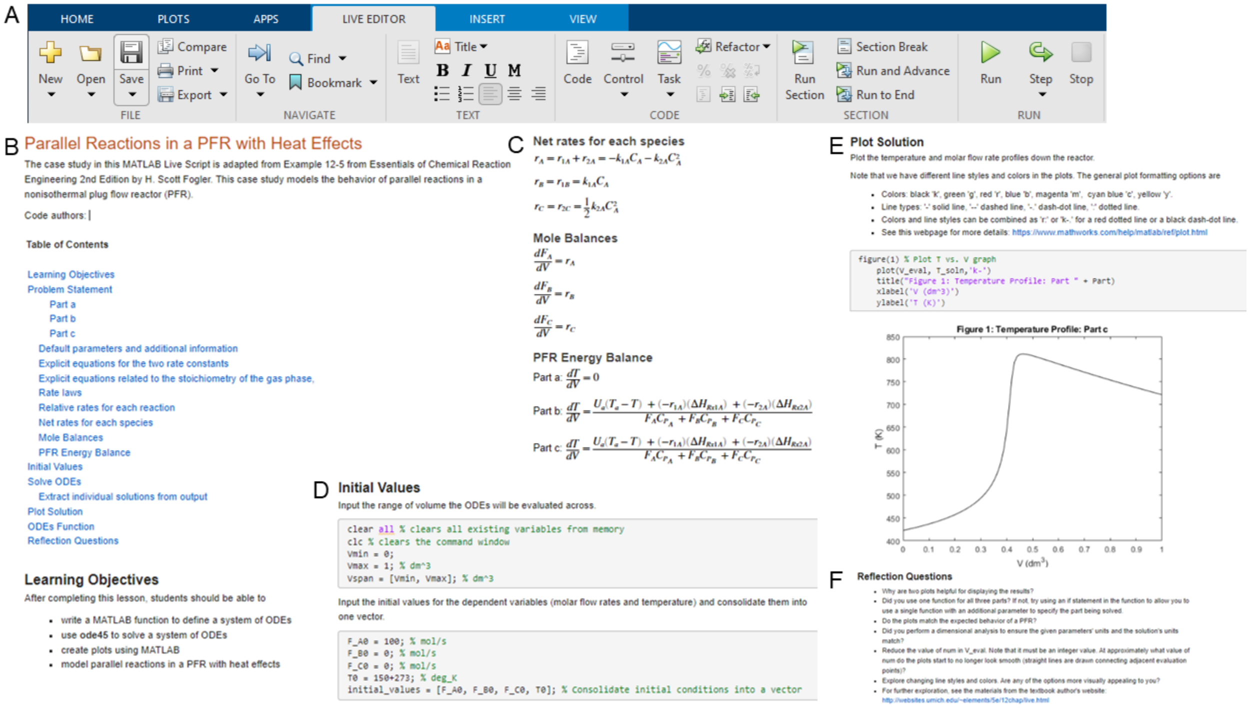
We began the workshop by introducing ourselves and relevant expertise and the workshop learning objectives (above). Next, one of the interactive coding templates was introduced, and we showed how it may be used in a chemical engineering classroom. Figure 2 illustrates some of the salient features for using MATLAB Live Scripts for educational purposes using one of our examples; similarly, Figure 3 shows a Jupyter Notebook example. Some of the interactive coding templates are designed to serve as class lecture notes or textbook alternatives with embedded activities. Figure 4 highlights a few examples of the types of visual aids that have been included for these purposes. Then, the pedagogical background of computational thinking and literate programming were introduced to participants with an emphasis on MATLAB Live Scripts and Jupyter Notebooks. The literate programming materials were the same used for our YouTube video [43], and the video is available for all outside the workshop. After the mini-lecture, we used active learning exercises (described below) to guide the participants towards achieving the learning objectives.

Active learning exercise 1

The activity focused on working through the provided tutorials on how to create interactive coding templates. Our use of the tutorials in the workshop was designed to model an educational use case where students read short segments of background text and follow along with embedded images and starting code blocks to complete the interactive activities. We used the following activity prompt:

- Choose either MATLAB Live Script or Jupyter Notebook (Google Colab)
- Open M0_HowToCreate.mlx or J0_HowToCreate.ipynb file on How to create a MATLAB Live Script/How to create a Jupyter Notebook
- Read through the file and complete the interactive activities
- We will circulate to answer your questions

The hands-on active learning exercise 1 was critical for participants to get experience with the tools for opening, running, editing, and creating MATLAB Live Scripts or Jupyter Notebooks. We spent a substantial portion of the workshop assisting participants with questions. Most participants had read the instructions in the Pre-Workshop Set-Up document in advance, which minimized the time needed to trouble shoot accessing the files and opening the software. An unexpected minor issue arose with some users of Google Colab not being allowed to use their university credentials to login due to some university-specific license agreements. This issue was bypassed by either use of a personal Google account or switching to the MATLAB software. The biggest technical challenge for participants was in editing code blocks or writing new functions if they were less familiar with the language.



Parallel Reactions in a PFR with Heat Effects

The case study in this MATLAB Live Script is adapted from Example 12-5 from Essentials of Chemical Reaction Engineering 2nd Edition by H. Scott Fogler. This case study models the behavior of parallel reactions in a nonisothermal plug flow reactor (PFR).

Code authors: |

Table of Contents

Learning Objectives
Problem Statement
Part a
Part b
Part c
Default parameters and additional information
Explicit equations for the two rate constants
Explicit equations related to the stoichiometry of the gas phase,
Rate laws
Relative rates for each reaction
Net rates for each species
Mole Balances
PFR Energy Balance
Initial Values
Solve ODEs
Extract individual solutions from output
Plot Solution
ODEs Function
Reflection Questions

Learning Objectives

After completing this lesson, students should be able to

- write a MATLAB function to define a system of ODEs
- use ode45 to solve a system of ODEs
- create plots using MATLAB
- model parallel reactions in a PFR with heat effects

Net rates for each species

$$r_A = r_{1A} + r_{2A} = -k_{1A}C_A - k_{2A}C_A^2$$

$$r_B = r_{1B} = k_{1A}C_A$$

$$r_C = r_{2C} = \frac{1}{2}k_{2A}C_A^2$$

Mole Balances

$$\frac{dF_A}{dV} = r_A$$

$$\frac{dF_B}{dV} = r_B$$

$$\frac{dF_C}{dV} = r_C$$

PFR Energy Balance

Part a: $\frac{dT}{dV} = 0$

Part b: $\frac{dT}{dV} = \frac{U_a(T_a - T) + (-r_{1A})(\Delta H_{Rx1A}) + (-r_{2A})(\Delta H_{Rx2A})}{F_A C_{P_A} + F_B C_{P_B} + F_C C_{P_C}}$

Part c: $\frac{dT}{dV} = \frac{U_a(T_a - T) + (-r_{1A})(\Delta H_{Rx1A}) + (-r_{2A})(\Delta H_{Rx2A})}{F_A C_{P_A} + F_B C_{P_B} + F_C C_{P_C}}$

Initial Values

Input the range of volume the ODEs will be evaluated across.

```
clear all % clears all existing variables from memory
clc % clears the command window
Vmin = 0;
Vmax = 1; % dm^3
Vspan = [Vmin, Vmax]; % dm^3
```

Input the initial values for the dependent variables (molar flow rates and temperature) and consolidate them into one vector.

```
F_A0 = 100; % mol/s
F_B0 = 0; % mol/s
F_C0 = 0; % mol/s
T0 = 150+273; % deg_K
initial_values = [F_A0, F_B0, F_C0, T0]; % Consolidate initial conditions into a vector
```

Plot Solution

Plot the temperature and molar flow rate profiles down the reactor.

Note that we have different line styles and colors in the plots. The general plot formatting options are

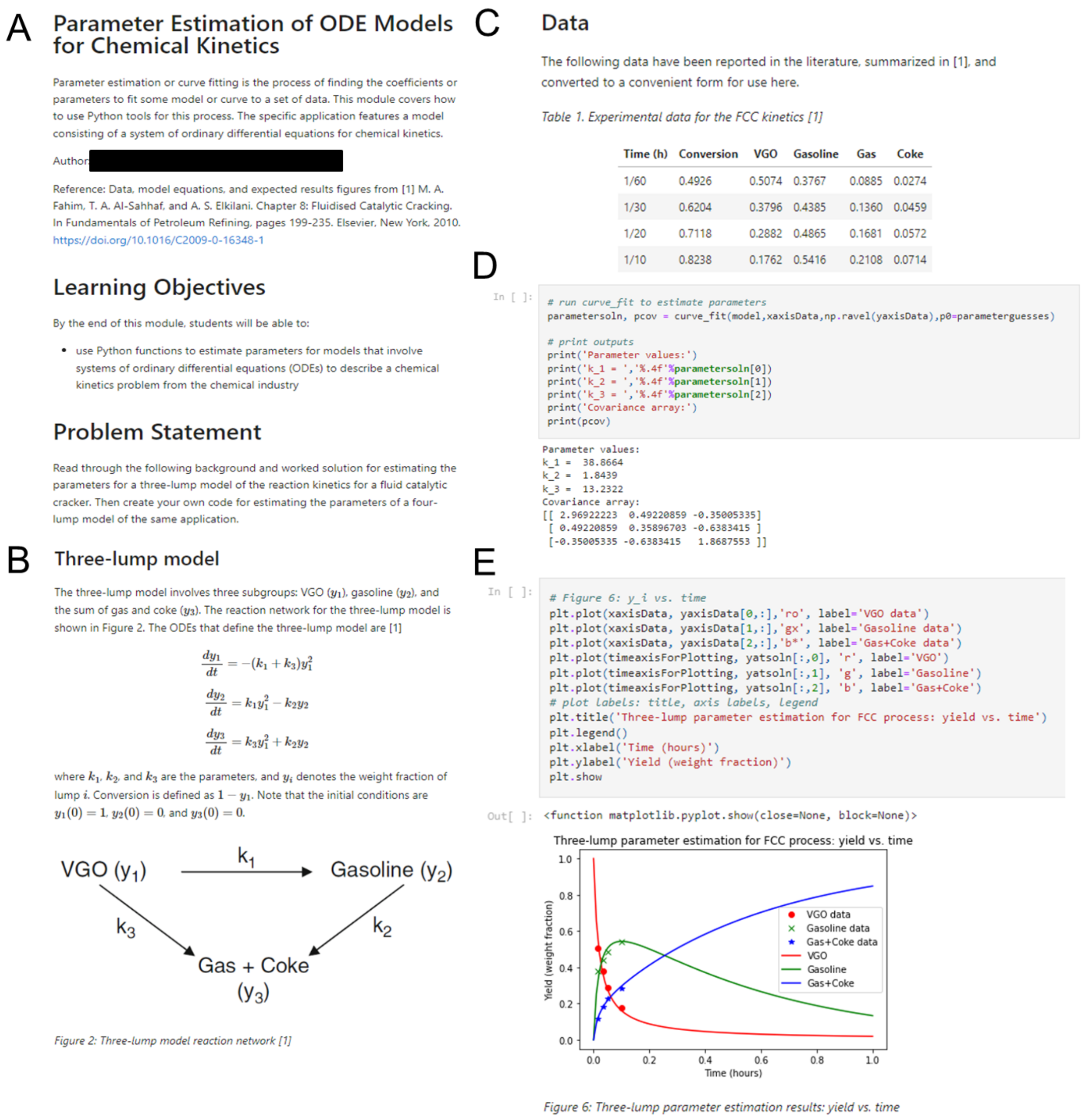
- Colors: black 'k', green 'g', red 'r', blue 'b', magenta 'm', cyan blue 'c', yellow 'y'.
- Line types: '-' solid line, '--' dashed line, '-.' dash-dot line, ':' dotted line.
- Colors and line styles can be combined as 'r:' or 'k-.' for a red dotted line or a black dash-dot line.
- See this webpage for more details: https://www.mathworks.com/help/matlab/ref/plot.html

```
figure(1) % Plot T vs. V graph
    plot(V_eval, T_soln,'k-')
    title("Figure 1: Temperature Profile: Part " + Part)
    xlabel('V (dm^3)')
    ylabel('T (K)')
```

Reflection Questions

- Why are two plots helpful for displaying the results?
- Did you use one function for all three parts? If not, try using an if statement in the function to allow you to use a single function with an additional parameter to specify the part being solved.
- Do the plots match the expected behavior of a PFR?
- Did you perform a dimensional analysis to ensure the given parameters' units and the solution's units match?
- Reduce the value of num in V_eval. Note that it must be an integer value. At approximately what value of num do the plots start to no longer look smooth (straight lines are drawn connecting adjacent evaluation points)?
- Explore changing line styles and colors. Are any of the options more visually appealing to you?
- For further exploration, see the materials from the textbook author's website: http://websites.umich.edu/~elements/5e/12chap/live.html

Figure 2: Example MATLAB Live Script with screenshots from M4_NonisothermalPFR_solution.mlx. A) MATLAB Live Editor menu, B) formatted text blocks with title, text, code authors (blinded), Table of Contents, and bulleted list of learning objectives, C) equations formatted with LaTeX directly in MATLAB, D) instructional text interwoven with code blocks written in MATLAB syntax, E) formatted text, code block, and plot output rendered inline, F) formatted text block with sample reflection questions at the end of the template.



Parameter Estimation of ODE Models for Chemical Kinetics

Parameter estimation or curve fitting is the process of finding the coefficients or parameters to fit some model or curve to a set of data. This module covers how to use Python tools for this process. The specific application features a model consisting of a system of ordinary differential equations for chemical kinetics.

Author [redacted]

Reference: Data, model equations, and expected results figures from [1] M. A. Fahim, T. A. Al-Sahhaf, and A. S. Elkilani. Chapter 8: Fluidised Catalytic Cracking. In Fundamentals of Petroleum Refining, pages 199-235. Elsevier, New York, 2010. https://doi.org/10.1016/C2009-0-16348-1

Learning Objectives

By the end of this module, students will be able to:

- use Python functions to estimate parameters for models that involve systems of ordinary differential equations (ODEs) to describe a chemical kinetics problem from the chemical industry

Problem Statement

Read through the following background and worked solution for estimating the parameters for a three-lump model of the reaction kinetics for a fluid catalytic cracker. Then create your own code for estimating the parameters of a four-lump model of the same application.

Three-lump model

The three-lump model involves three subgroups: VGO (y_1), gasoline (y_2), and the sum of gas and coke (y_3). The reaction network for the three-lump model is shown in Figure 2. The ODEs that define the three-lump model are [1]

$$\frac{dy_1}{dt} = -(k_1 + k_3)y_1^2$$

$$\frac{dy_2}{dt} = k_1 y_1^2 - k_2 y_2$$

$$\frac{dy_3}{dt} = k_3 y_1^2 + k_2 y_2$$

where k_1, k_2, and k_3 are the parameters, and y_i denotes the weight fraction of lump i. Conversion is defined as $1 - y_1$. Note that the initial conditions are $y_1(0) = 1$, $y_2(0) = 0$, and $y_3(0) = 0$.

Figure 2: Three-lump model reaction network [1]

Data

The following data have been reported in the literature, summarized in [1], and converted to a convenient form for use here.

Table 1. Experimental data for the FCC kinetics [1]

Time (h)	Conversion	VGO	Gasoline	Gas	Coke
1/60	0.4926	0.5074	0.3767	0.0885	0.0274
1/30	0.6204	0.3796	0.4385	0.1360	0.0459
1/20	0.7118	0.2882	0.4865	0.1681	0.0572
1/10	0.8238	0.1762	0.5416	0.2108	0.0714

In [ ]:

```
# run curve_fit to estimate parameters
parametersoln, pcov = curve_fit(model,xaxisData,np.ravel(yaxisData),p0=parameterguesses)

# print outputs
print('Parameter values:')
print('k_1 = ','%.4f'%parametersoln[0])
print('k_2 = ','%.4f'%parametersoln[1])
print('k_3 = ','%.4f'%parametersoln[2])
print('Covariance array:')
print(pcov)
```

```
Parameter values:
k_1 =  38.8664
k_2 =  1.8439
k_3 =  13.2322
Covariance array:
[[ 2.96922223  0.49220859 -0.35005335]
 [ 0.49220859  0.35896703 -0.6383415 ]
 [-0.35005335 -0.6383415   1.8687553 ]]
```

In [ ]:

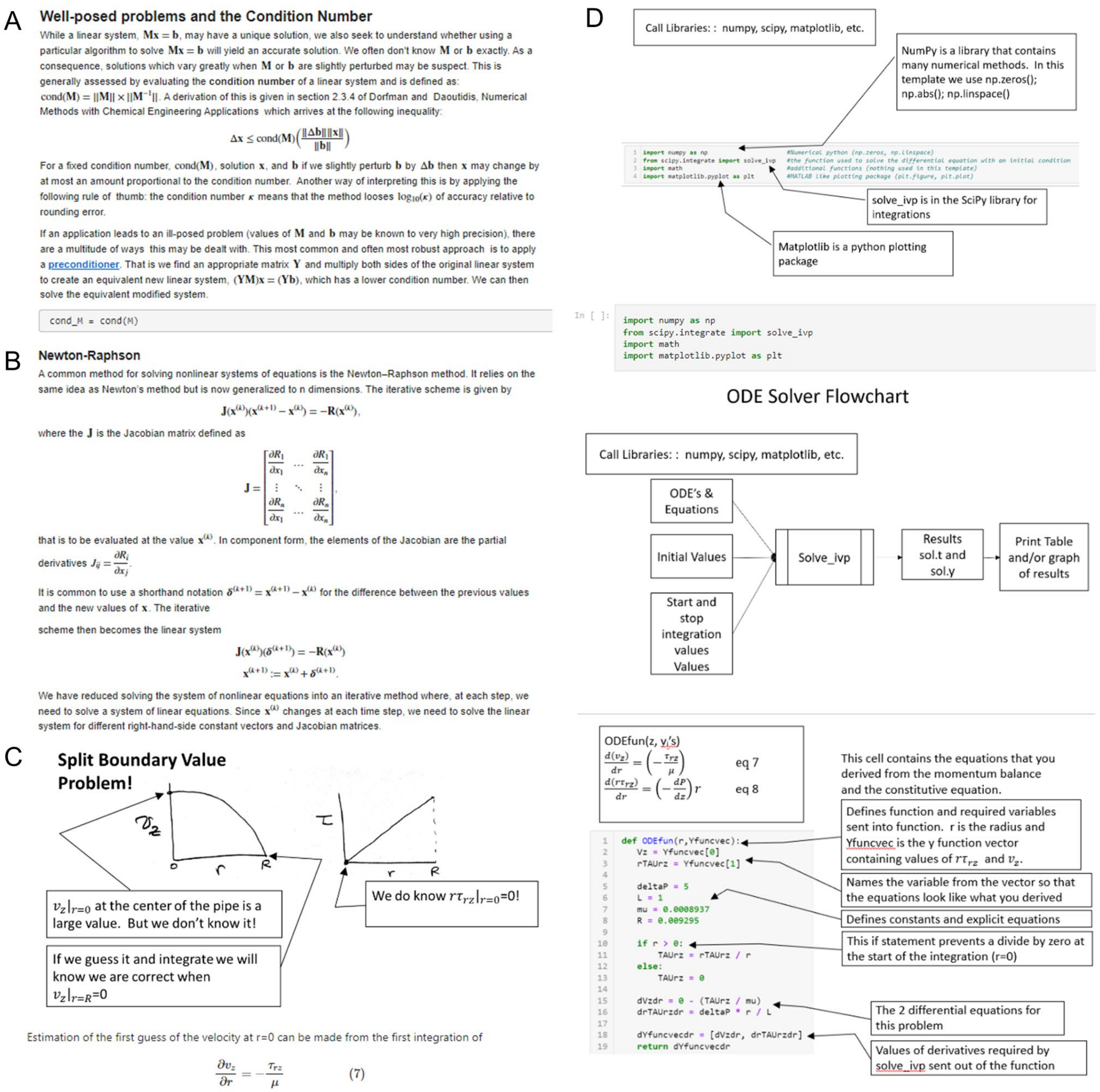
```
# Figure 6: y_i vs. time
plt.plot(xaxisData, yaxisData[0,:],'ro', label='VGO data')
plt.plot(xaxisData, yaxisData[1,:],'gx', label='Gasoline data')
plt.plot(xaxisData, yaxisData[2,:],'b*', label='Gas+Coke data')
plt.plot(timeaxisForPlotting, yatsoln[:,0], 'r', label='VGO')
plt.plot(timeaxisForPlotting, yatsoln[:,1], 'g', label='Gasoline')
plt.plot(timeaxisForPlotting, yatsoln[:,2], 'b', label='Gas+Coke')
# plot labels: title, axis labels, legend
plt.title('Three-lump parameter estimation for FCC process: yield vs. time')
plt.legend()
plt.xlabel('Time (hours)')
plt.ylabel('Yield (weight fraction)')
plt.show
```

Out[ ]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

Figure 6: Three-lump parameter estimation results: yield vs. time

Figure 3: Example Jupyter Notebook with screenshots from J5_ParEstKinetics.ipynb. A) formatted text blocks with title, text, code authors (blinded), and bulleted list of learning objectives (note Table of Contents is visible as a side panel if the file is opened in Google Colab), B) formatted text including equations written in markup language in Jupyter Notebook and an image embedded, C) table of data written in markup language, D) code block written in Python syntax and text output displayed inline, E) code block written in Python syntax and plot output rendered inline.



Well-posed problems and the Condition Number

While a linear system, $\mathbf{Mx} = \mathbf{b}$, may have a unique solution, we also seek to understand whether using a particular algorithm to solve $\mathbf{Mx} = \mathbf{b}$ will yield an accurate solution. We often don't know $\mathbf{M}$ or $\mathbf{b}$ exactly. As a consequence, solutions which vary greatly when $\mathbf{M}$ or $\mathbf{b}$ are slightly perturbed may be suspect. This is generally assessed by evaluating the **condition number** of a linear system and is defined as: $\mathrm{cond}(\mathbf{M}) = ||\mathbf{M}|| \times ||\mathbf{M}^{-1}||$. A derivation of this is given in section 2.3.4 of Dorfman and Daoutidis, Numerical Methods with Chemical Engineering Applications which arrives at the following inequality:

$$\Delta\mathbf{x} \le \mathrm{cond}(\mathbf{M})\left(\frac{\|\Delta\mathbf{b}\|\|\mathbf{x}\|}{\|\mathbf{b}\|}\right)$$

For a fixed condition number, $\mathrm{cond}(\mathbf{M})$, solution $\mathbf{x}$, and $\mathbf{b}$ if we slightly perturb $\mathbf{b}$ by $\Delta\mathbf{b}$ then $\mathbf{x}$ may change by at most an amount proportional to the condition number. Another way of interpreting this is by applying the following rule of thumb: the condition number κ means that the method looses $\log_{10}(\kappa)$ of accuracy relative to rounding error.

If an application leads to an ill-posed problem (values of $\mathbf{M}$ and $\mathbf{b}$ may be known to very high precision), there are a multitude of ways this may be dealt with. This most common and often most robust approach is to apply a preconditioner. That is we find an appropriate matrix $\mathbf{Y}$ and multiply both sides of the original linear system to create an equivalent new linear system, $(\mathbf{YM})\mathbf{x} = (\mathbf{Yb})$, which has a lower condition number. We can then solve the equivalent modified system.

```
cond_M = cond(M)
```

Newton-Raphson

A common method for solving nonlinear systems of equations is the Newton–Raphson method. It relies on the same idea as Newton's method but is now generalized to n dimensions. The iterative scheme is given by

$$\mathbf{J}(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = -\mathbf{R}(\mathbf{x}^{(k)}),$$

where the $\mathbf{J}$ is the Jacobian matrix defined as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial R_1}{\partial x_1} & \cdots & \frac{\partial R_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial R_n}{\partial x_1} & \cdots & \frac{\partial R_n}{\partial x_n} \end{bmatrix},$$

that is to be evaluated at the value $\mathbf{x}^{(k)}$. In component form, the elements of the Jacobian are the partial derivatives $J_{ij} = \frac{\partial R_i}{\partial x_j}$.

It is common to use a shorthand notation $\delta^{(k+1)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ for the difference between the previous values and the new values of $\mathbf{x}$. The iterative

scheme then becomes the linear system

$$\mathbf{J}(\mathbf{x}^{(k)})(\delta^{(k+1)}) = -\mathbf{R}(\mathbf{x}^{(k)})$$
$$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + \delta^{(k+1)}.$$

We have reduced solving the system of nonlinear equations into an iterative method where, at each step, we need to solve a system of linear equations. Since $\mathbf{x}^{(k)}$ changes at each time step, we need to solve the linear system for different right-hand-side constant vectors and Jacobian matrices.

Split Boundary Value Problem!

$v_z|_{r=0}$ at the center of the pipe is a large value. But we don't know it!

If we guess it and integrate we will know we are correct when $v_z|_{r=R}$=0

We do know $r\tau_{rz}|_{r=0}$=0!

Estimation of the first guess of the velocity at r=0 can be made from the first integration of

$$\frac{\partial v_z}{\partial r} = -\frac{\tau_{rz}}{\mu} \qquad (7)$$

Call Libraries: : numpy, scipy, matplotlib, etc.

```
import numpy as np                             #Numerical python (np.zeros, np.linspace)
from scipy.integrate import solve_ivp          #the function used to solve the differential equation with an initial condition
import math                                    #additional functions (nothing used in this template)
import matplotlib.pyplot as plt                #MATLAB like plotting package (plt.figure, plt.plot)
```

NumPy is a library that contains many numerical methods. In this template we use np.zeros(); np.abs(); np.linspace()

solve_ivp is in the SciPy library for integrations

Matplotlib is a python plotting package

In [ ]:

```
import numpy as np
from scipy.integrate import solve_ivp
import math
import matplotlib.pyplot as plt
```

ODE Solver Flowchart

Call Libraries: : numpy, scipy, matplotlib, etc.
ODE's & Equations
Initial Values
Start and stop integration values Values
Solve_ivp
Results sol.t and sol.y
Print Table and/or graph of results

ODEfun(z, y_i's)

$$\frac{d(v_z)}{dr} = \left(-\frac{\tau_{rz}}{\mu}\right) \qquad \text{eq 7}$$
$$\frac{d(r\tau_{rz})}{dr} = \left(-\frac{dP}{dz}\right)r \qquad \text{eq 8}$$

This cell contains the equations that you derived from the momentum balance and the constitutive equation.

```
def ODEfun(r,Yfuncvec):
    Vz = Yfuncvec[0]
    rTAUrz = Yfuncvec[1]

    deltaP = 5
    L = 1
    mu = 0.0008937
    R = 0.009295

    if r > 0:
        TAUrz = rTAUrz / r
    else:
        TAUrz = 0

    dVzdr = 0 - (TAUrz / mu)
    drTAUrzdr = deltaP * r / L

    dYfuncvecdr = [dVzdr, drTAUrzdr]
    return dYfuncvecdr
```

Defines function and required variables sent into function. r is the radius and Yfuncvec is the y function vector containing values of $r\tau_{rz}$ and v_z.

Names the variable from the vector so that the equations look like what you derived

Defines constants and explicit equations

This if statement prevents a divide by zero at the start of the integration (r=0)

The 2 differential equations for this problem

Values of derivatives required by solve_ivp sent out of the function

Figure 4: Examples of teaching aids embedded in the interactive coding templates with screenshots from A) M1_MassBalance.mlx, B) M2_NonlinearSystems.mlx, and C & D) J8_LaminarPipe.ipynb. A) Formatted text block with equations, hyperlink, and symbols, B) formatted text block with matrices, derivatives, and more, C) handwritten sketch and typed notes embedded as a image followed by text and an equation, D) coding tutorials with typed notes overlayed on code snippets embedded as images interspersed with executable code blocks.

Active learning exercise 2

For the second exercise, the workshop participants created their own interactive coding templates. Workshop participants formed teams of 2-3 based on mutual interest in the same programming language to provide some support in generating ideas and debugging. We used the following activity prompt:

- Choose either MATLAB Live Script or Jupyter Notebook (Google Colab)
- Find a partner who is interested in using the same language (MATLAB or Python)
- Work together to build a notebook for students to solve an equation of your choice. The notebook must include code blocks for numerical computation (solving) and the following text blocks (in any order)
    - learning objectives
    - problem statement
    - mathematical equation(s)
    - an image

The hands-on active learning exercise 2 challenged participants to apply the tools that they learned in exercise 1 for a lesson on a topic of their choice. Participants were asked to include code blocks, text blocks, learning objectives, a problem statement, mathematical equations, and an image because these were the most frequently used types of elements in the educational interactive coding templates developed by the authors (see Figures 2-4). The participants were very engaged in this activity. Some opened notes or problem sets from classes they teach and directly adapted that existing content into their MATLAB Live Script or Jupyter Notebook. Others found text and visual materials from the internet to include in their lessons. Due to how active participants were in this exercise, we did not proceed to final exercise. It is included below for reference as it was also shared with the participants.

Active learning exercise 3

We left a third exercise for exploration after the workshop. The activity prompt was:

- We have prepared a set of interactive coding templates that correspond to mathematical topics and chemical engineering topics across the curriculum.
- Pick a file from our set of templates. Play with the interface and reflect on how to adapt for your purposes: lecture/in-class activity, homework, group projects, other.

Conclusions and reflections for the future

The workshop and materials described in this paper were developed to provide faculty with resources for incorporating modern computational tools and computational thinking into their classrooms through lecture materials, interactive textbook-like content, case studies, in-class activities, homework, and course projects. We focused on instructing and equipping chemical engineering faculty rather than students to scale our efforts to reach the target student audience for the overall goal of improving their use of computational thinking and practices across chemical engineering curriculum. The workshop materials focused on a suite of interactive coding templates educators can either adapt to their needs or use as inspiration for their curricula. This paper serves to further disseminate the workshop materials and broaden the impact of these materials on the chemical engineering curriculum.

Using the interactive coding templates required no previous coding experience or installed software. The templates were designed to introduce students to relevant coding topics in a just-in-time fashion, i.e., teach only the programming aspects that are needed for a particular lesson when it is encountered. However, creating or editing the code blocks in the templates required some knowledge of the underlying software language. While the goal was to be beginner friendly, the expert amnesia of the presenters prevented us from seeing the workshop from a completely novice perspective. During and after the workshop, some participants mentioned that they were originally interested in the workshop because they were looking for guided instruction on how to create and use Python. While this was not the focus of the workshop, these comments illustrated that there was interest among chemical engineering faculty to receive additional computational training. Self-paced resources targeted to scientists and engineers on how to code using MATLAB and Python have been developed [40, 41, 49-56]. However, busy faculty may be interested in guided real-time workshops or coding bootcamps to efficiently train them to the point that they are then familiar enough with Python (or MATLAB) to further expand their skillsets independently. Additional efforts in disseminating existing Python resources could also help address the perceived need for Python instruction among chemical engineering faculty. For those interested in offering workshops similar to ours in the future, we recommend pairing this workshop with preliminary training in fundamentals of MATLAB and/or Python programming to prepare participants for editing the code blocks in the interactive coding templates.

For readers of this paper who are interested in deploying MATLAB Live Scripts or Jupyter Notebooks in your classes, we recommend the following:

- Gain familiarity with basics of MATLAB and/or Python programming. This could be through prior experience, a workshop such as those offered through AIChE or Software Carpentry, or through the self-paced resources for Python [40, 41, 49-55] and MATLAB [49-52, 56] (annotated in Table 1).
- Access our workshop materials online [1], particularly our library of interactive coding templates.
- Work through Active learning exercise 1 (described above) to learn how to create a MATLAB Live Script or Jupyter Notebook.
- Work through Active learning exercise 2 (described above) to make a simple lesson plan in a MATLAB Live Script or Jupyter Notebook.
- Skim through our library of interactive coding templates [1] and the curated materials [42, 44, 45, 57-77] annotated in Table 1 to find existing codes that you can adapt for various educational purposes across many chemical engineering applications and courses.
- Contact the authors if assistance is needed or to discuss reuse of any of these materials.

Table 1: Curated additional resources for chemical engineering educators. Computational materials relevant to the chemical engineering undergraduate curriculum that include MATLAB and/or Python among the available software types.

Resource Description	**Software Types Available**				**Source**
	MATLAB Live Scripts (.mlx)	**Jupyter Notebooks (.ipynb)**	**Static Scripts (.m / .py)**	**Other**	
Workshop materials					
Repository contains all the workshop materials described in this paper.	X	X			[1]
Materials for learning Python and/or MATLAB					
Webpage for Software Carpentry “Programming with Python” workshop focused on Python basics for handling data.			X		[40]
Webpage for Software Carpentry “Plotting and Programming with Python” workshop that introduces Python basics and plotting functions.			X		[41]
Textbook for modeling with MATLAB and Python.			X		[49]
Companion website for the code described in textbook for modeling with MATLAB and Python [49].			X		[50]
Course website on programming for engineers covering MATLAB and Python among other software.		X	X	X	[51]
Course website on applied numerical computing [11] covering Git for version control, LaTeX for typesetting, and MATLAB and Python for high-level programming and scientific computing.	X	X	X	X	[52]
Repository contains notes (see pdf at the link) on using Python in scientific and engineering calculations to demonstrate the utility of Python as a computational platform in engineering education.			X		[54]
AIChE Academy course website on Python for chemical engineers, focused on individuals who would benefit from receiving continuing education credits for such training.		X	X		[53]
Repository contains files used for an AIChE Python workshop for chemical engineers.			X		[55]
Website for catalogue of MATLAB self-paced online courses available through MathWorks.			X		[56]

Resource Description	Software Types Available				Source
	MATLAB Live Scripts (.mlx)	Jupyter Notebooks (.ipynb)	Static Scripts (.m / .py)	Other	
Computer Aids for Chemical Engineering (CACHE) chemical engineering teaching resources categorized by subject including material and energy balances, fluid mechanics, heat transfer, thermodynamics, and kinetics. CACHE teaching resources include recommended textbooks, interactive simulations, and software.	X	X	X	X	[57]
Course website on data-driven engineering covering data science topics in Python.		X	X		[42]
Textbook for numerical methods includes full codes for all example problems from chemical engineering applications.			X		[44]
Textbook for problem solving across chemical engineering includes full codes for all example problems from chemical engineering applications.			X	X	[45]
Companion website for the code for all living example problems described in textbook for reaction engineering [46].	X		X	X	[58]
Repository contains supplemental MATLAB Live Scripts and Jupyter Notebooks (running Julia, not Python) for use in courses centered around teaching and applying numerical methods with a chemical engineering context.	X	X		X	[59]
Course website for introduction to chemical engineering analysis covering basic chemical engineering calculations using Python. These calculations include stoichiometry, reactor performance, separations, and energy analysis.		X			[60]
Course website for practical numerical methods with Python, massive open online course (MOOC) covering Jupyter Notebook modules for relevant chemical engineering problems including finite-difference solutions of PDEs, convection problems, diffusion problems, and elliptic problems. Problems are worked step-by-step with an explanation for each step.		X	X		[61]
Course website on a second principles of chemical processes course covering energy balances for chemically non-reacting and reacting systems and how to use property tables and diagrams.	X	X			[62]

Resource Description	Software Types Available				Source
	MATLAB Live Scripts (.mlx)	Jupyter Notebooks (.ipynb)	Static Scripts (.m / .py)	Other	
Course website on separations covering modes of diffusion of mass and chemical composition, mass transfer analysis, molecular diffusion, and convective mass transfer and introducing equilibrium-staged mass transfer operations such as absorption/stripping and extraction/leaching operations.	X	X			[63]
Course website on a second semester fluids course for chemical engineers.	X	X			[64]
Course website on chemical reaction engineering covering homogeneous and heterogeneous reaction kinetics, idealized reactor models for batch and flow systems, corrections for non-ideal residence times, and heat and mass transfer effects. It also introduces homogeneous and heterogeneous catalytic processes and industrial catalytic reactors.		X			[65]
Repository of simulations for chemical and process engineering courses. These simulations allow the user to change equation parameters using sliders and buttons to obtain a better understanding of the system being modeled.		X			[66]
Repository for a lesson on solving systems of linear equations for material balances.	X				[67]
Repository for ht open source software and includes modules for various heat transfer functions.			X		[68]
Repository for Thermo open source software that facilitates the retrieval of constants of chemicals, the calculation of temperature and pressure dependent chemical properties (both thermodynamic and transport), and the calculation of the same for chemical mixtures (including phase equilibria) using various models.			X		[69]
Repository for Fluids open source software that includes modules for piping, fittings, pumps, tanks, compressible flow, open-channel flow, atmospheric properties, particle size distributions, two phase flow, friction factors, control valves, orifice plates and other flow meters, ejectors, relief valves, and more.			X		[70]

Resource Description	Software Types Available				Source
	MATLAB Live Scripts (.mlx)	Jupyter Notebooks (.ipynb)	Static Scripts (.m / .py)	Other	
Repository for the Chemics package, a collection of Python functions for performing calculations in the field of chemical engineering.			X		[71]
Website for Cantera, an open source suite of tools for problems involving chemical kinetics, thermodynamics, and transport processes that can be used from Python and MATLAB.		X	X	X	[72]
Jupyter community-curated collection of notable notebooks and includes sections on engineering education, mathematics, physics, chemistry, and biology. Links are all rendered using nbviewer.		X			[73]
Website for catalogue of available MathWorks courseware. Relevant subjects include intro to engineering, chemistry, and controls.			X		[74]
Website for MathWorks Grader contains prebuilt problem sets for system dynamics and control, statistics, numerical methods, and more.			X		[75]
Website for the MATLAB file exchange includes over 300 MathWorks files, nearly 45,000 Community files, and nearly 90 files tagged "Chemical Engineering."	X		X		[76]
Website for the MATLAB Live Script Gallery includes a selection of MATLAB Live Script interactive examples.	X				[77]

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. 2133411 (ANFV) and No. 1932723 (MDS) and resources from Computer Aids in Chemical Engineering (CACHE) Corporation (grants to ANFV and MDS and travel support for ANFV and RPH) and the University at Buffalo. We also acknowledge the support from the Chemical Engineering Summer School. We would like to thank Dr. David Kofke (master's project committee member for ANJ) and members of ANFV's lab for their thorough feedback on drafts of this manuscript. MDS would like to acknowledge Matthew E. Wilhelm and Chenyu Wang for designing the original MATLAB resources during his CACHE grant, and ANFV acknowledges Duncan H. Mullins for contributions to MATLAB and Python resources during her CACHE grant.

References

[1] A. N. Ford Versypt, R. Hesketh, A. Johns, and M. Stuber. "ChESS2022." https://github.com/ashleefv/ChESS2022 (accessed Dec. 23, 2022).

[2] R. P. Hesketh, M. Grover, and D. L. Silverstein, "CACHE/ASEE Survey on Computing in Chemical Engineering," in *ASEE Annual Conference*, Virtual, 2020. [Online]. Available: https://peer.asee.org/34249.

[3] J. Hedengren and B. Nicholson, "[Preprint] Open-Source Modeling Platforms." [Online]. Available: https://apm.byu.edu/prism/uploads/Members/Hedengren2023.pdf

[4] L. A. Barba *et al.*, *Teaching and Learning with Jupyter*, Creative Commons, 2019. [Online]. Available: https://jupyter4edu.github.io/jupyter-edu-book.

[5] L. A. Barba, "Engineers Code: Reusable Open Learning Modules for Engineering Computations," *Computing in Science & Engineering,* vol. 22, no. 4, pp. 26-35, 2020, doi: 10.1109/MCSE.2020.2976002.

[6] J. A. Lyon, A. Jaiswal, and A. J. Magana, "The Use of MATLAB Live as a Technology-enabled Learning Environment for Computational Modeling Activities within a Capstone Engineering Course," in *ASEE Annual Conference*, Virtual, 2020. [Online]. Available: https://peer.asee.org/35380.

[7] L. Ni and K. Hekman, "Improving Student Learning Experience with MATLAB Grader and Live Scripts," presented at the ASEE Annual Conference, Minneapolis, MN, 2022. [Online]. Available: https://peer.asee.org/40610.

[8] D. J. Antunes. "Using MATLAB Live Scripts to Teach Optimal Control and Dynamic Programming Online." https://www.mathworks.com/company/newsletters/articles/using-matlab-live-scripts-to-teach-optimal-control-and-dynamic-programming-online.html (accessed Feb. 18, 2023).

[9] N. Nevaranta, P. Jaatinen, K. Gräsbeck, and O. Pyrhönen, "Interactive Learning Material for Control Engineering Education Using Matlab Live Scripts," in *IEEE 17th International Conference on Industrial Informatics (INDIN)*, Helsinki, Finland, 2019, pp. 1150-1154, doi: 10.1109/INDIN41052.2019.8972282.

[10] M. Borowczak and A. C. Burrows, "Interactive Web Notebooks Using the Cloud to Enable CS in K-16+ Classrooms and PDs," in *ASEE Annual Conference*, Columbus, OH, 2017. [Online]. Available: https://peer.asee.org/28571.

[11] A. N. Ford Versypt, "An Interdisciplinary Elective Course to Build Computational Skills for Mathematical Modeling in Science and Engineering," in *ASEE Annual Meeting*, Tampa, FL, 2019. [Online]. Available: https://peer.asee.org/32072.

[12] B. Weber, "Work in Progress: Using Jupyter Notebooks to Climb Bloom's Taxonomy in Thermodynamics," in *ASEE Annual Conference*, Virtual, 2020. [Online]. Available: https://peer.asee.org/35700.

[13] M. Müller and S. Rosenzweig, "PCP Notebooks: A Preparation Course for Python with a Focus on Signal Processing," *Journal of Open Source Education,* vol. 5, no. 57, p. 148, 2022, doi: 10.21105/jose.00148.

[14] M. Duda *et al.*, "Teaching Python for Data Science: Collaborative development of a modular & interactive curriculum," *Journal of Open Source Education,* vol. 4, no. 46, p. 138, 2021, doi: 10.21105/jose.00138.

[15] J. Wagemann, S. H. Szeto, S. Mantovani, and F. Fierli, "LTPy - Learning tool for Python on Atmospheric Composition," *Journal of Open Source Education,* vol. 6, no. 60, p. 172, 2023, doi: 10.21105/jose.00172.

[16] V. Danchev, "Reproducible Data Science with Python: An Open Learning Resource," *Journal of Open Source Education,* vol. 5, no. 56, p. 156, 2022, doi: 10.21105/jose.00156.

[17] L. A. Barba and G. F. Forsyth, "CFD Python: the 12 steps to Navier-Stokes equations," *Journal of Open Source Education,* vol. 1, no. 9, p. 21, 2018, doi: 10.21105/jose.00021.

[18] L. A. Barba and O. Mesnard, "Aero Python: classical aerodynamics of potential flow using Python," *Journal of Open Source Education,* vol. 2, no. 15, p. 45, 2019, doi: 10.21105/jose.00045.

[19] M. Silva *et al.*, "Innovating and modernizing a Linear Algebra class through teaching computational skills," in *ASEE Annual Conference*, Minneapolis, MN, 2022. [Online]. Available: https://peer.asee.org/40766.

[20] K. Suthar *et al.*, "Real Data and Application-based Interactive Modules for Data Science Education in Engineering," presented at the ASEE Annual Conference, Virtual, 2021. [Online]. Available: https://peer.asee.org/37640.

[21] A. Dowling, "Toward Integrating Python Throughout the Chemical Engineering Curriculum: Using Google Colaboratory in the Classroom," in *Future of Cyber Assisted Chemical Engineering Education*, Breckenridge, CO, 2019. [Online]. Available: https://psecommunity.org/LAPSE:2019.0640.

[22] D. E. Knuth, "Literate Programming," *The Computer Journal,* vol. 27, no. 2, pp. 97-111, 1984, doi: 10.1093/comjnl/27.2.97.

[23] B. Childs, "Literate Programming, A Practioner's View," *TUGboat,* vol. 13, no. 3, pp. 261-269, 1992.

[24] M. Croucher. "Official MathWorks MATLAB kernel for Jupyter released." https://blogs.mathworks.com/matlab/2023/01/30/official-mathworks-matlab-kernel-for-jupyter-released (accessed Feb. 1, 2023).

[25] T. Zimmerman, "Computational Modeling in Introductory Physics Courses and Across the Curriculum," presented at the ASEE Annual Conference, Virtual, 2020. [Online]. Available: https://peer.asee.org/34319.

[26] J. M. Wing, "Computational thinking," *Communications of the ACM,* vol. 49, no. 3, pp. 33-35, 2006, doi: 10.1145/1118178.1118215.

[27] J. M. Wing, "Computational thinking and thinking about computing," *Philosophical Transactions of the Royal Society A,* vol. 366, no. 1881, pp. 3717-25, 2008, doi: 10.1098/rsta.2008.0118.

[28] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review,* vol. 22, pp. 142-158, 2017, doi: 10.1016/j.edurev.2017.09.003.

[29] K. Mills, M. Coenraad, P. Ruiz, Q. Burke, and J. Weisgrau, "Computational Thinking for an Inclusive World: A Resource for Educators to Learn and Lead." [Online]. Available: http://hdl.handle.net/20.500.12265/138

[30] F. B. Flórez, R. Casallas, M. Hernández, A. Reyes, S. Restrepo, and G. Danies, "Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming," *Review of Educational Research,* vol. 87, no. 4, pp. 834-860, 2017, doi: 10.3102/0034654317710096.

[31] D. Barr, J. Harrison, and L. Conery, "Computational Thinking: A Digital Age Skill for Everyone," *Learning & Leading with Technology,* vol. 38, no. 6, pp. 20-23, 2011. [Online]. Available: https://eric.ed.gov/?id=EJ918910.

[32] S. Grover and R. Pea, "Computational Thinking in K–12: A Review of the State of the Field," *Educational Researcher,* vol. 42, no. 1, pp. 38-43, 2013, doi: 10.3102/0013189X12463051.

[33] M. Israel, J. N. Pearson, T. Tapia, Q. M. Wherfel, and G. Reese, "Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis," *Computers & Education,* vol. 82, pp. 263-279, 2015, doi: 10.1016/j.compedu.2014.11.022.

[34] Digital Promise, "Powerful learning with computational thinking: Our why, what, and how of computational thinking." [Online]. Available: http://hdl.handle.net/20.500.12265/115

[35] C. Lu, R. Macdonald, B. Odell, V. Kokhan, C. Demmans Epp, and M. Cutumisu, "A scoping review of computational thinking assessments in higher education," *Journal of Computing in Higher Education,* vol. 34, pp. 416-461, 2022, doi: 10.1007/s12528-021-09305-y.

[36] J. Guggemos, S. Seufert, and M. Román-González, "Computational Thinking Assessment – Towards More Vivid Interpretations," *Technology, Knowledge and Learning,* vol. In Press, 2022, doi: 10.1007/s10758-021-09587-2.

[37] T. Durham Brooks, R. Burks, E. Doyle, M. Meysenburg, and T. Frey, "Digital imaging and vision analysis in science project improves the self-efficacy and skill of undergraduate students in computational work," *PLoS One,* vol. 16, no. 5, p. e0241946, 2021, doi: 10.1371/journal.pone.0241946.

[38] M. Román-González, J. Moreno-León, and G. Robles, "Combining assessment tools for a comprehensive evaluation of computational thinking interventions," *Computational Thinking Education*, S. C. Kong and H. Abelson, Eds.: Springer, 2019, pp. 79-98. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-6528-7_6

[39] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in *Annual Meeting of the American Educational Research Association*, Vancouver, BC, Canada, 2012, pp. 1-25. [Online]. Available: https://www.media.mit.edu/publications/new-frameworks-for-studying-and-assessing-the-development-of-computational-thinking/.

[40] Software Carpentry. "Programming with Python." https://swcarpentry.github.io/python-novice-inflammation/ (accessed Feb. 19, 2023).

[41] Software Carpentry. "Plotting and Programming in Python." http://swcarpentry.github.io/python-novice-gapminder/ (accessed Feb. 19, 2023).

[42] J. D. Hedengren. "Data-Driven Engineering." https://apmonitor.com/dde (accessed Feb. 19, 2023).

[43] A. N. Johns and A. N. Ford Versypt. "[YouTube Video] Literate Programming Using MATLAB Live Scripts and Jupyter Notebooks." https://youtu.be/u5YkzFl6FbE (accessed Jul. 24, 2022).

[44] K. Dorfman and P. Daoutidis, *Numerical Methods with Chemical Engineering Applications*, 1st ed. New York: Cambridge University Press, 2017.

[45] M. Cutlip and M. Shacham, *Problem Solving in Chemical and Biochemical Engineering with POLYMATH, Excel, and MATLAB*, 2nd ed. Hoboken, NJ: Prentice Hall, 2007.

[46] H. S. Fogler, *Essentials of Chemical Reaction Engineering*, 2nd ed. New York: Pearson Education, 2018.

[47] M. A. Fahim, T. A. Al-Sahhaf, and A. S. Elkilani, "Fluidised Catalytic Cracking," in *Fundamentals of Petroleum Refining*, 1st ed. New York: Elsevier, 2010, ch. 8, pp. 199-235.

[48] R. M. Felder, R. W. Rousseau, and L. G. Bullard, *Elementary Principles of Chemical Processes*, 4th ed. Hoboken, NJ: John Wiley & Sons, Inc., 2016.

[49] S. I. Gordon and B. Guilfoos, *Introduction to Modeling and Simulation with MATLAB and Python* 1st ed. Boca Raton, FL: CRC Press, 2020.

[50] S. I. Gordon and B. Guilfoos. "Introduction to Modeling and Simulation with MATLAB and Python Companion Site." http://www.intromodeling.com/ (accessed Dec. 30, 2022).

[51] J. D. Hedengren. "Programming for Engineers." https://apmonitor.com/che263/ (accessed Dec. 30, 2022).

[52] A. N. Ford Versypt and D. H. Mullins. "ApplNumComp: Applied Numerical Computing Course." https://github.com/ashleefv/ApplNumComp (accessed Dec. 30, 2022).

[53] J. Hedengren. "Introduction to Python for Chemical Engineers." https://www.aiche.org/academy/courses/ela270/introduction-python-chemical-engineers#course-tab-who-should-attend (accessed Dec. 30, 2022).

[54] J. Kitchin. "pycse - Python Computations in Science and Engineering." https://github.com/jkitchin/pycse (accessed Dec. 30, 2022).

[55] P. Adamson. "Python for Chemical Engineers: Getting Started." https://github.com/padamson/python_cheme (accessed Feb. 19, 2023).

[56] MathWorks. "Self-Paced Online Courses." https://matlabacademy.mathworks.com/ (accessed Dec. 30, 2022).

[57] CACHE Corporation. "Teaching Resources." https://cache.org/teaching-resources-center (accessed Dec. 30, 2022).

[58] H. S. Fogler. "Living Example Problems." http://websites.umich.edu/~elements/6e/live/index.html (accessed Dec. 30, 2022).

[59] M. Wilhelm, C. Wang, and M. Stuber. "Chemical Engineering - Analysis Notebooks." https://github.com/PSORLab/Chemical_Engineering_Analysis_Notebooks (accessed Dec. 30, 2022).

[60] J. Kantor. "CBE20255 Introduction to Chemical Engineering Analysis." https://github.com/jckantor/CBE20255 (accessed Dec. 30, 2022).

[61] L. A. Barba, I. Hawke, and B. Knaepen. "Practical Numerical Methods with Python." https://github.com/numerical-mooc/numerical-mooc (accessed Dec. 30, 2022).

[62] R. Hesketh. "Principles of Chemical Processes II." https://github.com/heskethrp/PrinciplesChemProcesses (accessed Dec. 30, 2022).

[63] R. Hesketh. "Separations I." https://github.com/heskethrp/Separations-I (accessed Dec. 30, 2022).

[64] R. Hesketh. "Process Fluid Transport." https://github.com/heskethrp/ProcessFluidTransport (accessed Dec. 30, 2022).

[65] R. Hesketh. "Chemical Reaction Engineering." https://github.com/heskethrp/CRE (accessed Dec. 30, 2022).

[66] CAChemE, "Chemical and Process Engineering Interactive Simulations," 2021. [Online]. Available: https://github.com/CAChemE/learn#chemical-and-process-engineering-interactive-simulations.

[67] A. Johns and A. N. Ford Versypt. "MEBLinearSystems." https://github.com/ashleefv/MEBLinearSystems (accessed Feb. 19, 2023).

[68] C. Bell. "ht: Heat transfer component of Chemical Engineering Design Library (ChEDL)." https://github.com/CalebBell/ht (accessed Dec. 30, 2022).

[69] C. Bell. "Thermo: Chemical properties component of Chemical Engineering Design Library (ChEDL)." https://github.com/CalebBell/thermo (accessed Dec. 30, 2022).

[70] C. Bell. "fluids: Fluid dynamics component of Chemical Engineering Design Library (ChEDL)." https://github.com/CalebBell/fluids (accessed Dec. 30, 2022).

[71] G. Wiggins. "Chemics." https://github.com/wigging/chemics (accessed Dec. 30, 2022).

[72] Cantera Developers. "Cantera." https://cantera.org/ (accessed Dec. 30, 2022).

[73] Project Jupyter. "Jupyter Wiki." https://github.com/jupyter/jupyter/wiki (accessed Dec. 30, 2022).

[74] MathWorks. "Courseware." https://www.mathworks.com/academia/courseware.html (accessed Dec. 30, 2022).

[75] MathWorks. "Grader." https://www.mathworks.com/products/matlab-grader.html (accessed Dec. 30, 2022).

[76] MathWorks. "File Exchange." https://www.mathworks.com/matlabcentral/fileexchange/ (accessed Dec. 30, 2022).

[77] MathWorks. "Live Script Gallery." https://www.mathworks.com/products/matlab/live-script-gallery.html (accessed Dec. 30, 2022).