Systems & Control Transactions

Original Research Article
Peer Reviewed Conference Proceeding
ESCAPE 36 - European Symposium on Computer Aided Process Engineering
Sheffield, UK. 21-24 June 2026

PSE PRESS

Understanding Student's Preferences for Computational Tools in Chemical Engineering Assessment

Sakiru Badmos

Department of Chemical Engineering, University College London, Torrington Place, London WC1E 7JE, United Kingdom
Corresponding Author: s.badmos@ucl.ac.uk

ABSTRACT

Computational tools are widely used in solving engineering problems and are now embedded within chemical engineering education. At the UCL Department of Chemical Engineering, students are taught gPROMS ModelBuilder in modules requiring coding; however, many choose alternative tools such as MATLAB, Python, or Polymath for coursework and capstone design project reactor design. This study investigates the reasons behind these preferences using a survey of fourth-year students who had completed their third-year design project. The results show that perceived ease of use, availability of external resources, and ease of debugging could strongly influence tool selection. The findings highlight the importance of accessibility, community support, and perceived relevance in shaping sustained student engagement with computational tools.

Keywords: Technology Adoption, Computational tools, Matlab, Python, Polymaths, Engineering Education, gProms

INTRODUCTION

Computational tools have been part of chemical engineering education for several decades, supporting the solution of algebraic equations, differential equations, and optimisation problems in core areas such as reaction engineering, separation processes, and process control [1-2]. What has changed in recent years is not their presence in the curriculum, but the *type of tools used, their accessibility, and how students engage with them*. The emergence of open-source programming environments, large online user communities, and data-driven methods has expanded the range of tools available to students and influenced how they approach computational problem-solving [3].

A range of platforms is currently used in undergraduate chemical engineering programmes. Tools such as POLYMATH remain effective for solving numerical problems, particularly in reaction engineering, due to its simplicity, presenting minimal challenges when picked up by students without prior knowledge of programming [4]. MATLAB is widely used in academia and industry for numerical methods, process modelling, and control because of its powerful numerical solvers, extensive libraries and visualisation capabilities [5-7]. More recently, Python has gained significant traction as an open-source alternative, supported by extensive scientific libraries and a large community of users, making it attractive for both teaching and independent learning [3]. The integration of Jupyter notebooks has made it attractive for teaching in undergraduate engineering programs [8-12].

Alongside these tools, commercial process modelling environments such as gPROMS ModelBuilder are increasingly used to represent industrial workflows, particularly for dynamic modelling, optimisation, and parameter estimation [13].

At University College London (UCL), gPROMS is taught within the undergraduate programme to expose students to equation-oriented modelling approaches used in industry. However, unlike widely used scripting tools, gPROMS requires a licence and has a relatively smaller user community and fewer openly available learning resources. These differences may influence how students engage with the tool, particularly when they have a choice in assessment. Despite being taught gPROMS, many students choose alternative tools such as MATLAB, Python, or POLYMATH for their coursework and design project reactor designs. This raises an important question: *why do students move away from tools that are formally taught, even when support is provided?* While prior studies have explored the pedagogical value of individual tools [5–12], less attention has been given to students'

tool preferences in assessments, where time constraints, confidence, and access to support play a significant role.

This study investigates the factors influencing students' choices of computational tools in chemical engineering assessment, using gPROMS ModelBuilder as a case study. Drawing on principles from the Technology Acceptance Model [14], this work examines how perceptions of ease of use, usefulness, and access to support shape students' decisions. The study addresses three research questions: (i) what factors influence students' decisions to use alternative tools when gPROMS is taught, (ii) how students perceive the teaching and support provided for gPROMS, and (iii) what changes could improve engagement with computational tools in the curriculum.

METHODOLOGY

A Mentimeter survey was presented to the fourth-year students (n = 40) of the 2025/2026 academic year, who had completed their design projects in the previous year. This cohort was chosen because they had direct experience applying a computational tool in their coursework and the third-year design project. The survey consists of 9 questions of different types including multiple choice and an open-ended question using a five-point Likert scale. Ethical approval was obtained from UCL's Research Ethics Office to process and publish the results of this survey. The survey questions are as follows:

- Q1 – Which primary tool did you use for your Year-3 design project reactor design?
- Q2 – Which primary tool did you use for other coursework e.g CREII?
- Q3 - How confident did you feel using your primary tool for design project and coursework?
- Q4 - Why did you choose that tool?
- Q5 – What is the most important factor to consider when choosing a tool for your work
- Q6 - How adequate was the gPROMS teaching you received before the design project?
- Q7 - How important is tool popularity/community in your decision?
- Q8 - How important is tool popularity/community in your decision?
- Q9 - What would make gPROMS more usable/attractive for students?

Free texts from students' responses to Q9

- T1: 'Better UI'
- T2: 'Make the interface more user friendly and intuitive'
- T3: 'Better interface (why model and process? One would be simpler)'
- T4: 'More tutorials with solution'
- T5: 'I prefer to learn code from online sources like stack overflow examples and github and YouTube videos. gPROMS doesn't have this in the same way as Python and even MATLAB'.
- T6: 'Popularity – it is so unnecessarily hard to find help for this'
- T7: 'Clearer availability of support/more support sessions'

RESULTS AND DISCUSSION

Chemical engineering education has computational tools as integral part of the undergraduate curriculum. These tools are used for solving problems involving numerical methods, ordinary differential equations (ODEs), optimisation, and, in some cases, dynamic simulation. Traditionally, software such as MATLAB, Excel, and Visual Basic has been used for this purpose, with more recent additions including Python, gPROMS, and GAMS. At UCL Chemical Engineering Department, students are taught GAMS and gPROMS within core chemical engineering modules, while MATLAB is introduced through the Mathematical Modelling and Analysis (MMA) module delivered by the faculty as part of the Integrated Engineering Programme (IEP). These tools are embedded in the curriculum to satisfy accreditation requirements set by the Institution of Chemical Engineers (IChemE) [15]. The skills developed using these tools are applied across several modules throughout the undergraduate programme. Further details on how the tools are taught can be found in Kamel et al. [16].

Despite students being introduced to gPROMS from their first year, they appear to show low enthusiasm towards the tool both in teaching and learning and in applying it for their coursework. When given the opportunity to select a preferred software, many students will choose alternative tools, even though gPROMS is taught on the programme. This pattern is observed in both the second-year Chemical Reaction Engineering module (CREII) and the third-year capstone design project, where students are required to implement reactor models computationally. Although gPROMS is supported through teaching sessions, questions and answers (Q&A) and forums on the learning platform (Moodle), students are permitted to use other tools if they feel more comfortable doing so.

From the survey results, as shown in Figure 1a, 50% of the respondents (20 students) designed a reactor as part of their design project, and among these, 90% used MATLAB or Polymath, while only 10% used gPROMS. This occurred despite the availability of dedicated

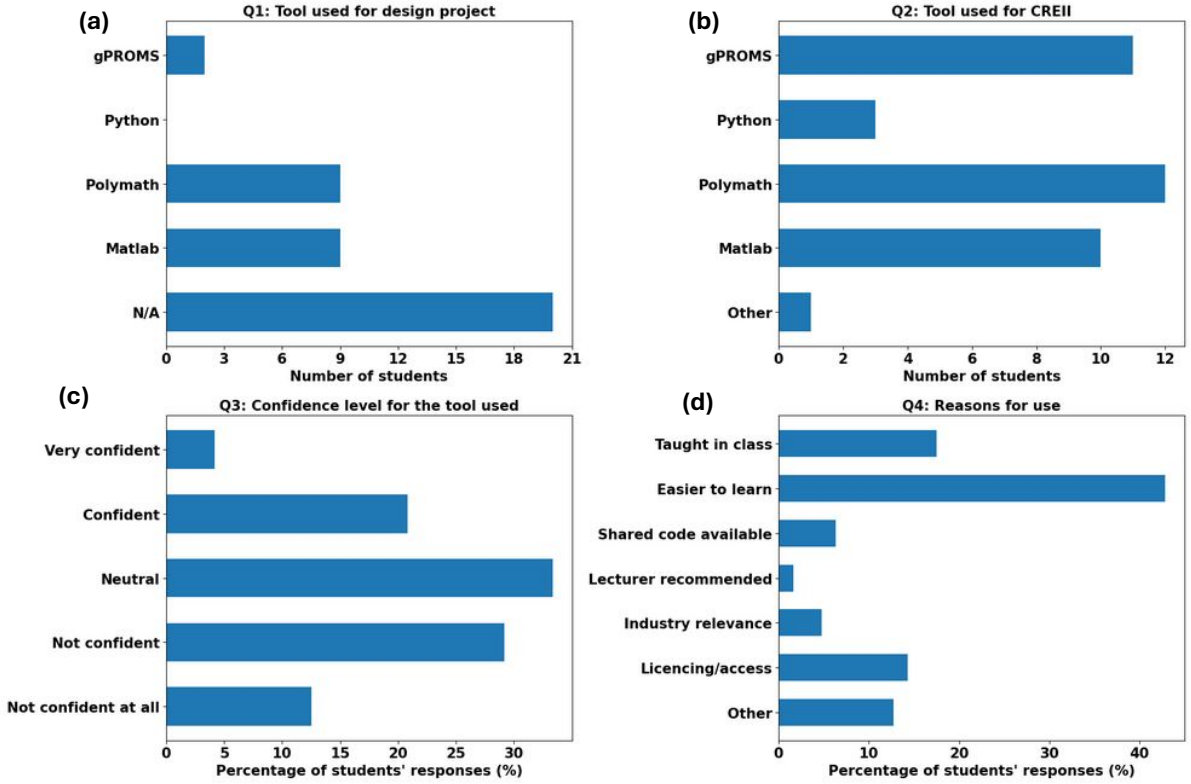

Figure 1. Students' responses to Q1 – Q4. N/A (n=20) in panel 1a indicates respondents who designed a separator during the detailed design of their third-year design project and does not require coding.

computational support for gPROMS on the design project. Similarly, in CREII, where all students must implement a reactor model in a software, approximately 70% selected a tool other than gPROMS (Figure 1b). These results indicate a clear reluctance to use gPROMS and suggest that factors beyond simple availability of support influence students' choices.

Confidence in using computational tools also appears to be an important issue. Figure 1c shows that although students were free to choose their preferred software, around 42% reported that they were not confident in using the selected tool, and a further 33% were uncertain about their confidence level. This suggests that students may be selecting tools based on short-term use rather than genuine competence. It also points to a possible need to review current teaching approaches or to provide additional learning resources to better support students in developing confidence with the tools they are taught.

The survey results indicate that students' tool selection is shaped by a combination of practical and perceived benefits. As shown in Figure 1d, ease of learning was the most influential factor, with approximately 42% of respondents identifying this as the main reason for their choice. About 30% selected their tool because it was taught in class, which likely corresponds to those who used MATLAB or gPROMS, as these are the two scripting tools formally introduced during the programme. Licensing and access were also relevant, with 14% of students citing this as a determining factor. Smaller proportions identified the availability of shared code, industry relevance, lecturer recommendation, and other factors as influencing their decision.

When asked what factor should be prioritised when selecting a tool for assessment, students again emphasised ease of learning and external support. As shown in Figure 2a, 27% selected ease of learning as the most important criterion, while 23% selected the availability of resources and community help. Ease of debugging and industry employability were each selected by 18% of respondents, and supervisor preference by 14%. These responses suggest that students tend to prioritise immediate academic needs over longer-term considerations such as industrial relevance. This may reflect a focus on completing assessments efficiently rather than on developing skills perceived as valuable for future employment. It may also indicate limited awareness of the tools commonly used in professional practice, or a tendency to defer such concerns until later stages of their studies.

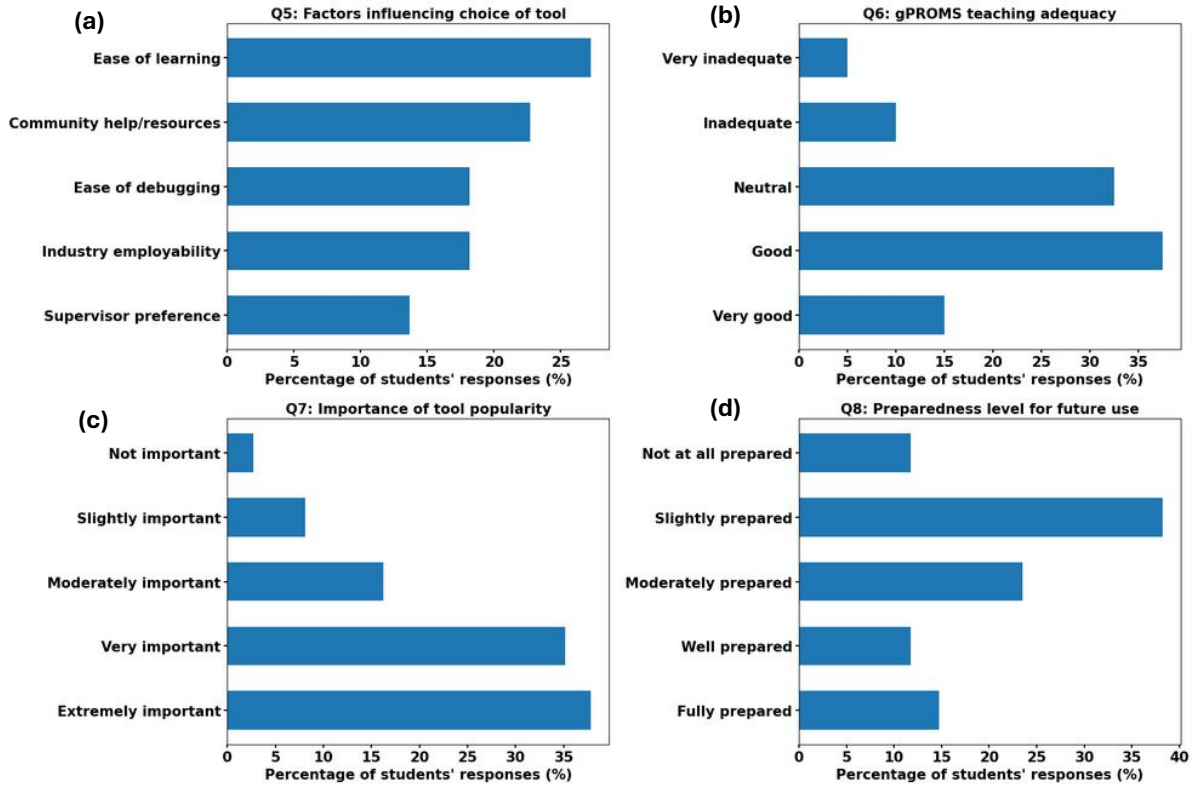

Figure 2. Students' responses to Q5 – Q8 on factors affecting choice of tools (2a), adequacy of gPROMS teaching (2b), importance of tool popularity in their decision making (2c) and their level of preparedness for using the tool in future roles (2d)

The importance placed on external resources and community support highlights the role of informal learning environments in students' decision-making. Large user communities and readily available online examples make tools such as Python and MATLAB particularly attractive, especially when students encounter errors or difficulties. Availability of template codes in Fogler's Reaction engineering textbook makes Polymath a good choice. These features directly affect their ability to complete coursework within limited timeframes and were therefore prioritised. A substantial number of students also valued ease of debugging and the potential for transferability of skills for industrial applications. At the same time, some students preferred to follow lecturer recommendations, possibly viewing these as aligned with assessment expectations.

Figures 1a and 1b show that most respondents used tools other than gPROMS for both the design project and CREII assessment, with only 10% selecting gPROMS for reactor design in the capstone project. To explore whether this preference resulted from inadequate teaching instructions, students were asked in Q6 to evaluate the adequacy of gPROMS teaching and support. As shown in Figure 2b, 52% considered the teaching and learning provision was sufficient to support their use of the software, while 32% were unsure and only 16% felt that support was inadequate. This suggests that dissatisfaction with teaching provision alone does not fully explain the low uptake of gPROMS. Instead, the findings indicate that students' choices are more strongly influenced by perceived ease of use and perceived usefulness, consistent with the Technology Acceptance Model (TAM). Even where formal teaching and support are available, students may still avoid a tool if it is perceived as difficult to use, slow to debug, or poorly supported by external resources. This implies that improving uptake may require more than simply increasing teaching hours or technical support through forums and Q&A sessions. Factors such as user interface design as reflected in free-text responses from respondents (T1–T3 in Q9 responses), access to examples and tutorials (T5), size of the user community (T6), and licensing constraints also play an important role in students' decisions.

Tool popularity further reinforces these trends. Students were asked how much popularity influenced their choice of tool, and as shown in Figure 2c, 38% reported it as extremely important, 35% as very important, and 16% as moderately important. Only 8% considered it slightly important and 3% not important. This indicates that perceived popularity, which is closely linked to the

availability of learning resources and external help with debugging, significantly affects students' selection of computational tools.

Students were also asked how well the teaching and learning of computational tools has prepared them for future industrial roles. As shown in Figure 2d, majority of respondents felt adequately prepared, while only 12% reported feeling unprepared. This reinforces the overall effectiveness of the current teaching approach to computational tools within the UCL chemical engineering programme. However, it also suggests that there may be value in reviewing the specific tools taught, particularly with regard to accessibility and long-term sustainability, and in considering a gradual transition towards more open-source alternatives.

Overall, the results suggest that students' preferences for computational tools in assessment are shaped by a complex interaction of pedagogical, technical, and social factors. While gPROMS is formally embedded in the curriculum, its lower adoption compared with MATLAB, Python, and Polymath appears to be driven primarily by perceptions of difficulty, limited external support, and usability rather than by deficiencies in teaching provision. Although the sample size does not allow for definitive conclusions, this study provides a preliminary basis for further investigation. It highlights the need for educators to consider not only what tools are taught, but also students' experience, particularly in relation to workload, error handling, and access to help beyond the classroom.

REFLECTION

My experience in teaching computational tools suggests that students often adopt a grade-driven approach to assessments and learning. As educators, we should encourage students to recognise the value of learning that extends beyond immediate assessment outcomes. This can be better addressed through the lens of constructive alignment [17], where intended learning outcomes, teaching activities, and assessment tasks are deliberately aligned. If assessments primarily reward correct outputs rather than the modelling process, tool selection, and problem-solving approach, students are more likely to prioritise efficiency over critical thinking. Aligning assessment with the intended development of computational thinking, modelling skills, and tool fluency may encourage students to engage more meaningfully with the tools taught, rather than selecting alternatives based solely on convenience or familiarity.

CONCLUSION

This study examined the factors influencing students' preferences for computational tools in chemical engineering assessments, using gPROMS ModelBuilder as a case study. The results show that students' choices are driven primarily by perceived ease of learning, availability of external support, and ease of debugging, rather than by formal exposure to a tool within the curriculum. Although gPROMS is taught and supported, it is used by only a small proportion of students when alternative tools are allowed. This suggests that technical capability alone is insufficient to ensure adoption. Instead, usability, community support, and perceived relevance play a significant role. These findings highlight the need for educators to consider not only what tools are taught, but how they align with students' learning priorities and future employability. Rather than focusing solely on introducing advanced tools, there is a need to rethink how these tools are integrated into the curriculum. This could include providing earlier exposure to modelling environments, embedding gPROMS more consistently across multiple modules, and designing scaffolded learning activities that gradually build confidence before assessments. In addition, integrating more guided examples, improving access to reusable code templates, and creating opportunities for peer learning may help reduce the initial barrier to engagement. Consideration could also be given to combining proprietary tools with open-source alternatives to support flexibility and independent learning. Future work will explore how these targeted curriculum adjustments influence students' confidence, engagement, and sustained use of computational tools.

ACKNOWLEDGEMENTS

The author acknowledges financial support from UCL Chemical Engineering department.

REFERENCES

1. Shacham M, Cutlip MB. Selecting the appropriate numerical software for a chemical engineering course. Computers & Chemical Engineering 23:S645-S648 (1999). https://doi.org/10.1016/s0098-1354(99)80158-6
2. Edgar, T.F., Himmelblau, D.M., & Lasdon, L.S. (2006). *Optimization of Chemical Processes* (2nd ed.). McGraw-Hill.
3. Udugama, I.A., Heintz, S., & Gernaey, K.V. (2023). *Digitalisation in chemical engineering education: Status, challenges and future perspectives*. Education for Chemical Engineers, 42, 1–16. https://doi.org/10.1016/j.ece.2022.10.003
4. Brenner A, Shacham M, Cutlip MB. Applications of mathematical software packages for modelling and simulations in environmental engineering education. Environmental Modelling & Software 20:1307-1313 (2005).

https://doi.org/10.1016/j.envsoft.2004.09.007
5. Lee, K., et al., 2014. Survey: MATLAB & MathCAD education in biochemical engineering. Chem. Eng. Educ. 48 (1), 59.
6. Richelle A, Bogaerts P. Systematic methodology for bioprocess model identification based on generalized kinetic functions. Biochemical Engineering Journal 100:41-49 (2015). https://doi.org/10.1016/j.bej.2015.04.003
7. Talbert, R., 2012. Learning MATLAB in the Inverted Classroom. Paper presented at 2012 ASEE Annual Conference & Exposition, San Antonio, Texas. 10.18260/1-2—21640.
8. Tang, C., 2021. Computer-aided linear algebra course on jupyter-python notebook for engineering undergraduates. J. Phys.: Conf. Ser. 1815, 012004–012009. https://doi. org/10.1088/1742-6596/1815/1/012004.
9. Inguva P, Bhute VJ, Cheng TNH, Walker PJ. Introducing students to research codes: a short course on solving partial differential equations in python. Education for Chemical Engineers 36:1-11 (2021). https://doi.org/10.1016/j.ece.2021.01.011
10. Lorandi Medina AP, Ortigoza Capetillo GM, Saba GH, Perez MAH, Garcia Ramirez PJ. A simple way to bring python to the classrooms. 2020 IEEE International Conference on Engineering Veracruz (ICEV) :1-6 (2020). https://doi.org/10.1109/icev50249.2020.9289692
11. Hunter JD. Matplotlib: a 2D graphics environment. Comput. Sci. Eng. 9:90-95 (2007). https://doi.org/10.1109/mcse.2007.55
12. Lee Y, Cho J. The influence of python programming education for raising computational thinking. IJUNESST 10:59-72 (2017). https://doi.org/10.14257/ijunesst.2017.10.8.06
13. Siemens Process Systems Engineering, 2022. gPROMS, https://www.psenterprise.com/products/gproms/process, 1997-2022.
14. Davis FD. Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly 13:319-340 (1989). https://doi.org/10.2307/249008
15. Institution of Chemical Engineers (2021). 'Accreditation of chemical engineering programmes – A guide for education providers and assessors.' Available at https://www.icheme.org/media/26729/university-accreditation-guide.pdf [last accessed: 20th March 2026]
16. Kamel D, Tsatse A, Badmos S. Teaching computational tools in chemical engineering curriculum in preparation for the capstone design project. Systems and Control Transactions 4:2197-2202 (2025). https://doi.org/10.69997/sct.126494
17. Biggs J. Enhancing teaching through constructive alignment. High Educ 32:347-364 (1996). https://doi.org/10.1007/bf00138871