Journal of the Foundation Year Network, Volume 8 (2026), pp. 37-58

Enhancing Digital Skills as Part of an Integrated Curriculum in a STEM Foundation Year

LEWIS A. BAKER, CAROL SPENCELY, ALIFAH RAHMAN and RICHARD HARRISON
University of Surrey

This study provides an overview of a structured approach to digital skill development in a transitional university programme and offers insights into student perceptions of these learning activities, which can inform future curriculum design and support mechanisms. Specifically, we review the deliberate design of a digital skills curriculum for a Science, Technology, Engineering, and Mathematics Foundation Year programme, aimed at equipping students from diverse educational backgrounds with essential computational and academic competencies. The curriculum integrates foundational skills in Microsoft Excel, programming with MATLAB, and web development using HTML/JavaScript. To evaluate the curriculum's effectiveness and the student experience, a mixed-methods approach was employed, gathering survey data from 147 students across two academic years. The findings reveal that while students found learning new programming concepts, particularly in MATLAB, to be the most challenging aspect of the module, they also identified it as one of the most valuable skills they acquired. Students self-reported improvement across a range of digital skills, including programming, data analysis, and website development.

Background

Enhancing digital skills within a STEM Foundation Year presents a unique set of challenges. Students arrive at university with a pre-existing set of digital skills. Indeed, digital skills development is a compulsory element of the UK National Curriculum (DfE, 2014). However, students' opportunities to develop digital skills and computational thinking at the secondary stage vary greatly; see for instance, Overland (2025). Whilst digital skills development is incorporated across subject areas in UK secondary schools, computer science, focusing solely on a range of digital skills, is an optional subject at GCSE and, as of 2017, was available to around 75% of GCSE students (Kemp et al., 2024). The current situation has improved (to around 82% as of 2023), but at the time of our Foundation Year programme's introduction (2018), this was highly relevant. It is still the case that the proportion of students choosing to study GCSE and A Level computer science, when offered, is lower than in other science subjects such as biology and physics (DfE, 2023). This is exacerbated by shortages of teachers within all these subjects, but notably in computer science (TeachFirst, 2025). Yet computer science as a university degree subject continues to be an attractive and competitive proposition for

students, since in many cases an A Level in computer science is not a prerequisite. Irrespective of any *a priori* formal computer science education, it is difficult to quantify students' preparedness for studying computer science at university, both in terms of their more general digital skills capabilities and, equally importantly, in their capacity to 'think computationally' (Wing, 2008). Although there are many definitions (Shute et al., 2017; Cansu and Cansu, 2019; Kandemir et al., 2021), computational thinking in its broadest sense relates to translating a problem into a computational implementation. This presents a scenario whereby student cohorts commencing computer science-related degree programmes have an exceptionally diverse range of experiences and capabilities in digital skills and their capacity to solve problems computationally (Tarling et al., 2023). For instance, some of the students may already be proficient in one or more programming languages, whereas others may not have had any experience with programming. In foundation year programmes, this range of experience can be amplified due to the broad spectrum of students typically enrolled on our programmes (Dampier et al., 2019). Further specific challenges can be presented in a modular course where designing an appropriate introductory computer science module with consideration of students' previous digital skills development is required. Two challenges in our Foundation Year are firstly, much lower entry tariffs such as CCD at A Level compared to direct entry undergraduate programmes, which are typically AAB for many computing-related degree programmes. This generally results in the need to provide more extensive guidance and individual support to encourage students to develop the necessary skills. Secondly, our introductory computing module needs to prepare students progressing onto over twenty different STEM degree programmes, including Computer Science. All these programmes will contain elements of computing and, by implication, digital skills development, but their content is varied and the emphasis and the software tools used are specific to the programme of study. For example, almost all undergraduates in the Faculty of Engineering & Physical Sciences will need to learn to write code, but the language could be Python, MATLAB, C, or Java, depending on the degree course. Therefore, the design of computing modules for the Foundation Year programme tends to be far less clearly defined compared to, say, designing mathematics modules, owing to the degree of variability in the initial digital skills capabilities of the incoming students and likewise variability in the curricula of the destination undergraduate programmes.

A deliberate curriculum (Felder and Brent, 2016) is a good design principle, but in practice, we must evaluate it across several dimensions. One dimension that is explicit in yearly programme reviews is academic attainment, where we review coursework performance and iterate assignments and guidance for students where appropriate. A second domain is the student experience, capturing the voices and opinions of the students who complete the assignments and understanding how the module can

continue to develop both skills and the learning environment. To capture all these dimensions, beyond the detailed curriculum design offered here, we have also distributed a survey across two academic years (2023-24 and 2024-25) to collect the student voice on their experience in engaging with the curriculum. As indicated in Figure 1, these digital skills are developed within an integrated curriculum. The module in which these skills are a focus, ENG0018 - Computer Laboratory, also encompasses a range of academic skill development opportunities within the 'Conference Project', described in detail elsewhere (Spencely et al., 2023).

Towards a design for a bridging computing module – ENG0018

Designing a suitable introductory computing module proved to be very challenging (Bosse et al., 2019). We began by reviewing the digital skills being developed by students in the first year of their undergraduate programmes. Although the content of these programmes was clearly defined and therefore a reliable point to map our learning outcomes onto, the diversity in content and context made the common ground across the programmes difficult to specify and the priorities unclear. Discussions with undergraduate teaching colleagues concerning the foundation computing module content echoed the differences in the published undergraduate module specifications and there were disagreements about what the foundation students should be learning to have the best preparation for the various undergraduate

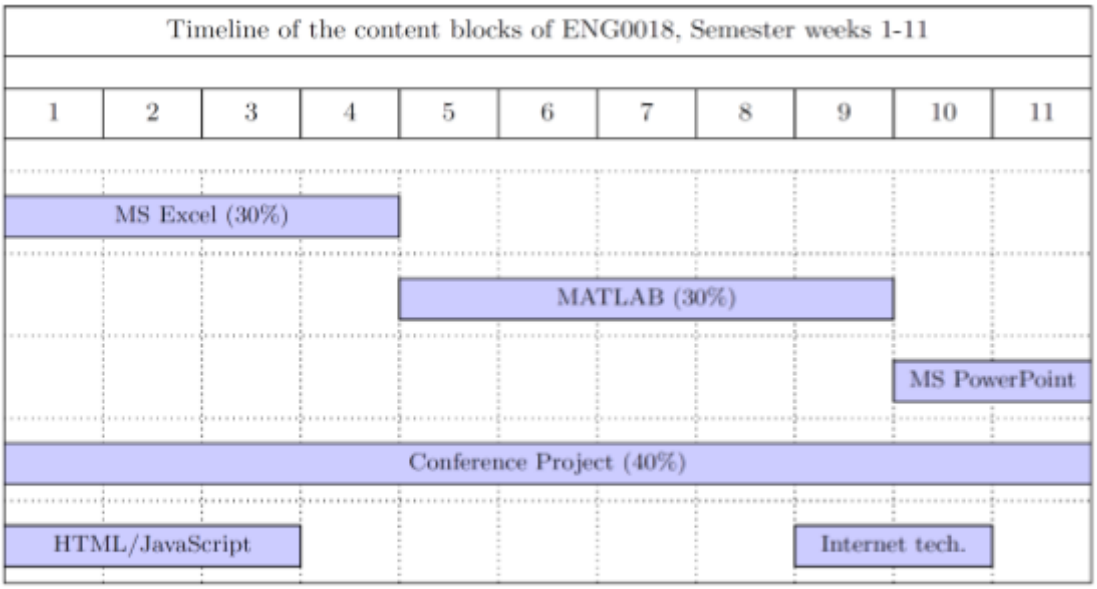

Figure 1. The timeline of the content blocks of this module spans the first semester (11 teaching weeks). The Microsoft (MS) PowerPoint, HTML/JavaScript content blocks are accessed within the Conference Project.

programme routes. Proposals such as focusing on learning three different programming languages had to be ruled out as this would not provide the necessary

breadth of foundational digital skills in a one-semester (11 teaching weeks) module. The module design problem was further compounded by the need for cross-curricular mapping; our module needed to be relevant to the foundation Mathematics, Physics, and Laboratory modules. Furthermore, from the perspective of the student experience, we wanted to introduce novelty to promote interest and engagement whilst simultaneously providing an accessible and challenging module to cater for the diverse range of students' previous experiences.

Our review of the undergraduate modules and our discussions with teaching colleagues revealed that most undergraduates would be using MATLAB in their first year and would also need to process data and prepare reports. As the module would be practically focused, the initial assessment strategy was 100% coursework, integrating authentic assessments wherever possible (Wootton, 2021). We designed the first iteration of the module in six blocks, as summarised in Figure 1. The Cconference Project has already been discussed in an earlier publication (Spencely et al., 2023), and since the PowerPoint presentations were a component of the Conference Project, this paper focuses on the design and development of the Microsoft (MS) Excel, MATLAB and HTML/JavaScript components.

Designing the Excel component

Starting with the Excel component, we identified three different design aims: (i) preparation for the MATLAB component, (ii) cross-curricular mapping, and (iii) novelty of learning resources.

(i) *Preparation for the MATLAB component*

A fundamental assumption, which subsequently proved to be correct, was that many incoming students would not have advanced digital skills, such as being able to program. Neither would they have had much opportunity to develop computational thinking. It was therefore expected that the introductory MATLAB component would be one of the most challenging elements of the Foundation Year academic curriculum. The Excel component would expose the students to some of the constructs that would be encountered when programming for the first time, for example, recognising that an Excel sheet is a visual representation of an indexed two-dimensional array. We therefore decided to deliver the Excel component before MATLAB as a less challenging start to the module. Students would gain familiarity with carrying out tasks with cell referencing using the indices and therefore develop a basic understanding of how to carry out operations within an array before meeting the same structures in MATLAB. Students would also gain experience in using different built-in mathematical functions and constructing simple algorithms to carry out data processing tasks. The algorithm

design includes branching using the built-in conditional statements in Excel, a key programming concept that would again arise in the MATLAB component. Collectively, as well as developing quite specific digital skills in working with Excel, these tasks would also serve to initiate the development of students' computational thinking.

(ii) *Cross-curricular mapping*

All students would be expected to produce laboratory reports or other technical reports on the Foundation Year programme and in their subsequent undergraduate programmes. This would involve the management, processing, analysis and visualisation of data. A substantial part of the Excel component and the associated coursework assessment was dedicated to processing and visualising data. This Excel coursework task was set for the students as the first coursework assessment in the academic year. It was sequenced to furnish the students with the necessary skills before completing their first engineering/science laboratory coursework assessment, in which they would have to produce graphs of experimental data. Students learned how to process multiple sets of data, including log data for the same experiment, identifying and replacing missing or erroneous values, as shown in Figure 2. In addition to calculating summary statistics, the students developed skills in preparing professional visualisations suitable for a formal report. These included histograms and plots of experimental data with error bars and customised data markers. Students were able to make use of these skills in both semesters of the Foundation Year programme.

As well as embedding these data processing and statistical skills within the component, we also planned to integrate some elements of mathematical skills development to align with the mathematics module. Early semester content in the mathematics module included basic algebra, polynomials, curve sketching and numerical methods. Tasks were set in the Excel laboratory worksheets and coursework to reinforce students' practice on a variety of techniques associated with using these mathematical concepts. This included using formulae in Excel to construct polynomial functions and their graphs. Students also created a quadratic equation solver, using cell-referenced formulae to implement the quadratic formula with logical tests to evaluate the discriminant. Finding roots of equations was then extended to polynomials with an implementation of the Newton-Raphson numerical method. The trapezium rule was also implemented to find a numerical approximation of the area under a curve. These concepts were brought together in the coursework, where a typical task involved the students graphing a polynomial function, finding the roots using Newton's method and determining the area enclosed between the curve and the horizontal axis, on the sub-domains defined by the roots determined previously.

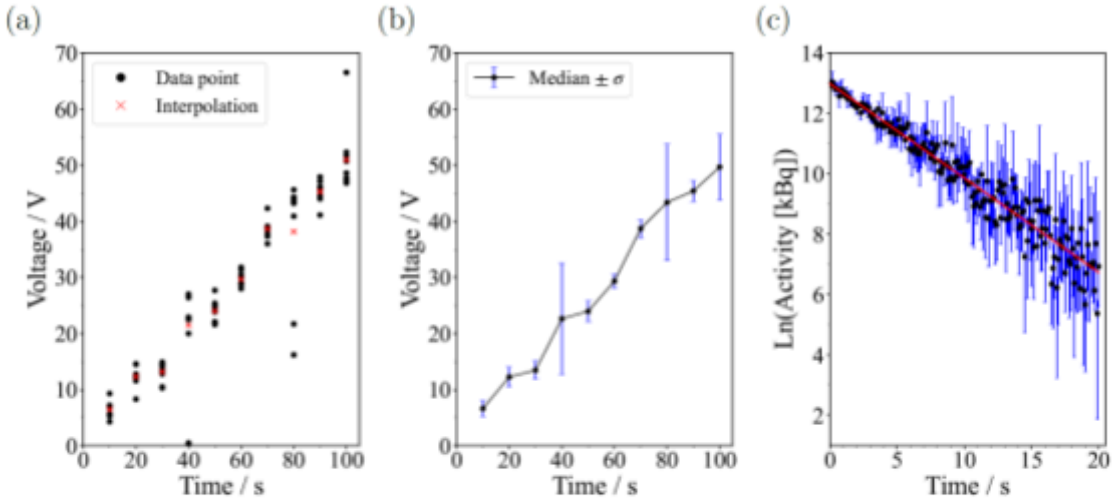

Figure 2. *The EXCEL coursework assignments.* (a) Students are guided through cleaning up a data set through interpolation of missing data points (red x's) for repeats of experimental trials (black circles). (b) Students are then guided through representing combined data sets through suitable statistical calculations, for example, here, the median is taken, and a symmetrical distribution of variation across trials is represented as the standard deviation in error bars. (c) The coursework task then involved students cleaning a large data set of radioactive decay measurements, representing the trials by calculating the median, and calculating an asymmetrical error based on upper and lower bounds compared to this median. A line of best fit is finally calculated to answer questions about the physical insights into this experimental data.

(iii) *Novelty of learning resources*

All our resources are contained within a module area on our virtual learning environment (VLE). In the 'traditional' sense, instruments such as laboratory worksheets are typically passive electronic documents. It is commonplace to have one laboratory worksheet for each laboratory session. Following a verbal introduction at the start of the laboratory session, students are left to work through the worksheet at their own pace, supported by the tutor and demonstrators (Roberts et al., 1995). Of necessity, the worksheets contain many sequenced steps to break the task down, with detailed written descriptions for each step, accompanied by graphics such as flowcharts and screenshots of the Excel sheet with specific actions highlighted. This is an established approach, and it would be suitable for most students most of the time. However, we wanted to provide the students with a more interactive learning experience with a cycle comprising: studying an example, doing a task based on the example, assessing the learning and receiving feedback. To facilitate this, an e-book was developed to parallel the PDF worksheets. The e-book enabled a one step per page horizontal navigation compared to the continuous vertical navigation of the PDF. Most of the steps had embedded video content demonstrating the procedure to be

carried out in the Excel worksheet. This presented a clear visualisation of the cursor movements, selections and data entry, supported by narration explaining what was being done. After watching a short video clip, the students would carry out a similar task for themselves. They would then answer embedded multiple-choice questions concerning the task and receive instant feedback on their answer. This approach was applied to each of the steps as the task developed. Students were presented with a choice of using either the passive PDF worksheet or the interactive e-book. In our experience over the past six years, most students (around 90%) have chosen to study with the e-book option.

Designing the MATLAB component

For most students, the MATLAB component would be their first experience of programming. To write any computer program, it is necessary to have an in-depth understanding of the problem to be solved and how to break the problem down into a sequence of tasks and a processing flow, independently of the programming language used. Once implemented as code, the program needs to be tested to verify that it is working correctly and meets the design criteria. The approach to the design of the MATLAB component required a consideration of: (i) developing familiarity with MATLAB, (ii) introduction to fundamental programming constructs, and (iii) cross-curricular considerations.

(i) *Developing familiarity with MATLAB*

Practical tasks in the familiarisation stage were designed to support an 'adjustment phase', providing students with multiple opportunities to digest and respond to some of the new ways of working. As well as developing more advanced digital skills, they would learn the importance of paying attention to detail and fault-finding (Robins, et al., 2003). To gain familiarity in working with MATLAB and to develop the new skills required, students were first introduced to the command line in the MATLAB environment. Here, through detailed laboratory worksheets and live demonstrations, they learned how to make variable assignments and carry out calculations symbolically. Starting with simple arithmetic, we progressed to more complex calculation examples, with 'built-in' errors to generate error messages that students learned to decipher then trace the error in the statement. They subsequently learned how to edit their statements, apply appropriate corrections and verify that the output was correct. Fault-finding is a fundamental skill in programming. Initially, the temptation for most students when programming for the first time is to write several lines of code and then expect the code to run. Naturally, there is great scope for the introduction of errors by novice programmers. To try to instil good habits of verification, we encouraged an approach of entering one line of code at a time, then executing to check that it works and produces the expected output. This was particularly useful when working with a

file path to read or write a file. Many students struggled with the level of precision required and the syntax in the file path definition. To help the students to develop more general skills in fault finding and verification, many of the early examples contained numerical answers and visualisations, such as line charts from a data set. Since the students' task would be to replicate the graph, including specific formatting of axes and text, it was immediately apparent visually if there was an error. This visualisation approach was continued throughout the MATLAB component and the coursework task.

(ii) *Introduction to fundamental programming constructs*

From our analysis of the year 1 undergraduate curricula, we established that students would commonly need to work with arrays as well as use loops and conditional statements, so, given the short timeframe available to us, we decided to focus on these elements of programming. Following on from the familiarisation stage, we moved to working with MATLAB files to create some simple programs involving these three constructs. To assist with developing the problem-solving process and fault-finding, we continued with the visualisation theme. An example programming task was to create a binary image containing a pattern of black and white regions. Students could immediately see whether their loops or conditional statements were operating as expected on the correct region of the two-dimensional array used to create the specified pattern in the image. For instance, constructing a diagonal white line against a black background required setting up a loop to place 0s in the leading diagonal of a square, two-dimensional array from indices (1, 1) to (100, 100). Introducing more complex patterns gradually increased the level of the problem-solving skills required and the sophistication of the program design, such as the use of multiple conditional statements within loops. For the final stage of their programming development, we moved from two-dimensional to three-dimensional arrays. Again, the focus was on the visualisation of the output. Using a stack of three two-dimensional arrays, the contents could be rendered as a colour image, enabling easy identification of any errors in each of the layers.

Figure 3. *The MATLAB coursework assignments.* (a) Students were asked to generate the following binary colour image, building the image row-by-row within a single 'for loop' using conditional statements. (b) Students were then required to produce the following similarly red-green-blue image, once again row-by-row using a single 'for loop' and suitable conditional statements. This also required applying a colour gradient to different parts of the image. (c) Finally, students were to simulate a simple random walk of playing several games of European roulette. By researching the probabilities of a specific win (red or black), they can simulate 200 games and report the net winnings, along with additional stopping criteria.

(iii) *Cross-curricular considerations*

One of the key features of MATLAB is the ease with which matrix operations can be carried out. Matrices are not covered within our mathematics curriculum until the second semester. Since the computing module is in semester 1, it was necessary to embed an accessible introduction to some basic matrix concepts and operations within our teaching of the MATLAB component. This began with definitions such as row and column vectors, square matrices, the identity matrix, zero matrices and so on. We then introduced arithmetical operations on matrices and the properties of those operations. Some applications were considered, including how matrices (arrays) can be used explicitly to construct binary and colour images (Figure 3(a) and 3(b)), and implicitly in data management within a program, such as storing spatial location history in a random walk model (Figure 3(c)). In the image construction tasks, which introduced the students to arrays, conditional statements, and loops, the students would initially experiment with trial and error to generate the correct pattern shapes from their arrays. Almost all the bounds for the conditional statements and array indexing in arithmetical sequences could be established by forming and solving single or simultaneous linear equations. The students had already covered linear equations as well as sequences and series in their mathematics module, and when presented with guidance on how to apply these concepts in their programming to reliably create the required values at specific locations in their arrays, many moved away from trial and error to this more procedural approach. Some other fundamental mathematical concepts which were common to the MATLAB component and the mathematics modules were gradients and, more generally, rates of change. Although the students were not explicitly using calculus in their programming tasks, they were applying important calculus concepts and learning that these concepts can be applied to more

abstract problems, such as setting up colour gradients in an image and finding the mean intensity value of a colour over an image region.

Module Evolution – HTML and JavaScript

The core structure and content of the Excel and MATLAB components of the module have remained largely unchanged since its introduction in 2018. There have been some minor adjustments to the balance of these components, such as reducing the Excel content by around 25% and shortening the duration of the component by one week. Correspondingly, an additional week was added to the duration of the MATLAB component. This was accompanied by new content on one-dimensional random walk models, to further consolidate the students' programming and problem-solving skills development, with tentative applications to further skill development, such as Educational Dialogue (Baker, 2025). It also introduced the students to the practice of modifying existing code to suit a new but related scenario. By this stage, the students have developed the capability to grasp what a relatively simple but unfamiliar piece of code is doing and identify where and how it should be modified to apply it to a related problem. In a specific assessed case, this involved modifying the one-dimensional random walk model to simulate a game of European Roulette. Perhaps the most substantial changes have taken place in the Conference Project (Spencely et al., 2023). Originally, the student output for the assessed component was a Word document submission for their conference paper and PowerPoint slides for their presentation. In consultation with a student focus group, we realised that we could embed further digital skills development opportunities, as well as create a new and interesting challenge for our students by moving towards a web-based project. The academic learning outcomes for the Conference Project itself remained unchanged. However, the associated digital skills development scope was greatly expanded as the students learned to use HTML and JavaScript to create their own webpages. Similar to MATLAB, the workshops included worksheets and demonstrations to scaffold student learning in these often-new concepts (Roberts et al., 1995). Furthermore, the students were introduced to a professional development environment (GitHub) in which they constructed their live webpages. The practical laboratory-based sessions were complemented by a series of new lectures on 'Internet Technology' covering the historical development of the internet and its technical infrastructure. This is assessed by a short and relatively low-weighted (5%) online test at the end of the module. The practical HTML and JavaScript output is assessed separately within the Conference Project coursework assessment structure (Spencely et al., 2023).

Methods

Now that the deliberate nature of the digital skills curriculum has been reviewed, we turn to the methodology used to evaluate the student experience. All students in the 2023-24 and 2024-25 FEPS Foundation Year programmes were invited to complete a survey. The survey was written using JISC Online Surveys and distributed by email and by QR codes within lectures, with an attached participant information sheet for this research. The survey consisted of ranking scales (1 = easy to 10 = very hard) for the tasks students were required to complete during the module, Likert scales for student agreement on their self-reported perception of skill development, and several open questions to invite reflection on challenges or suggest improvements to the module. A total of 147 students completed the survey, with a response rate of 41%. This included 62 of 154 students in the 2023-24 academic year (40% response rate) and 85 of 201 students in the 2024-25 academic year (42% response rate). In each academic year, approximately 80% of the respondents were studying the Engineering and Physical Science programmes and approximately 20% of the respondents were studying the Mathematics and Computer Science programmes. Before data collection, the researchers completed an ethical self-assessment, indicating that a further review by a University committee was not required.

Results and discussion

Except where stated otherwise, statistical data are aggregated across cohorts.

Student-perceived difficulty of assignments

To understand how students perceived the difficulty of the individual assessments, they were asked: ‘On a scale of 1 (EASY) to 10 (VERY HARD), how difficult did you find completing the following coursework tasks: [Tasks]’, where [Tasks] were the key assessment tasks described in the ENG0018 module (see Figure 4). Students were then asked, ‘In your words, what made your top three “Hardest” tasks challenging to complete?’ so that themes could be generated describing the reasons for the perceived difficulties. A total of 83 free-text responses were received in response to this follow-up question.

Quantitative perception of difficulty

The overall perceived difficulties were considered by learning objective as shown in Figure 4.

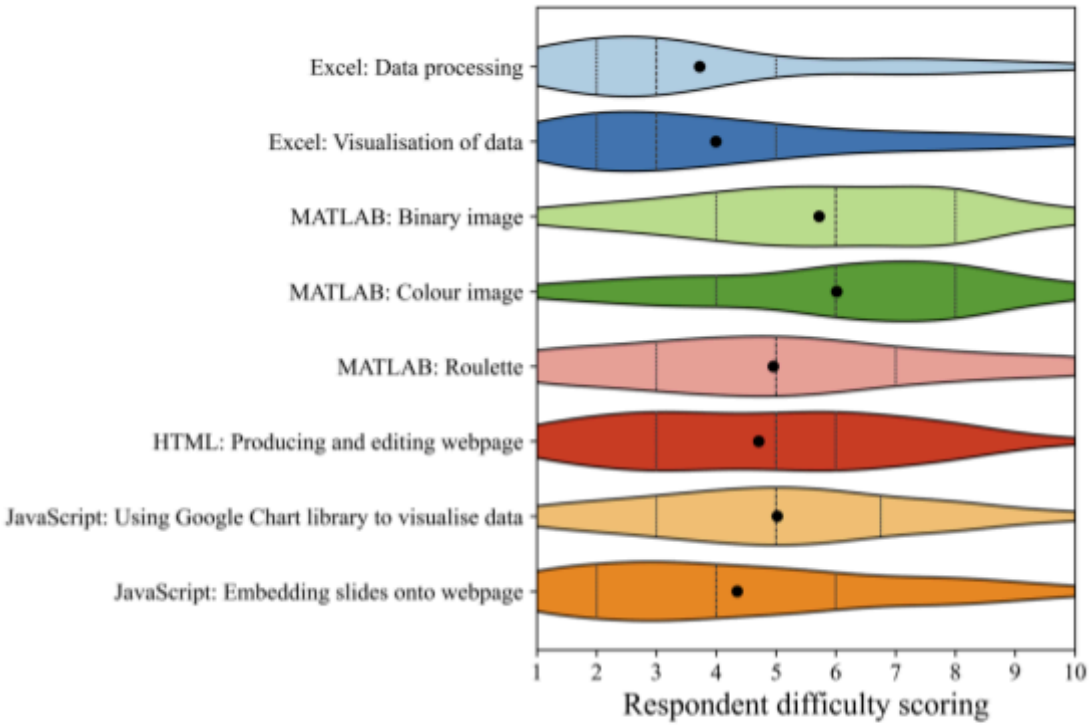

Figure 4. Students were asked, '*On a scale of 1 (EASY) to 10 (VERY HARD), how difficult did you find completing the following coursework tasks?*'. Histograms of the self-reported difficulty ratings for each task are given, with the mean response rating indicated by a black circle, and lower, median, and upper quartiles are indicated by dashed vertical lines.

The Excel assignments

Students indicated that out of all the assessments, the Excel assignments were perceived as the least challenging, giving an aggregate difficulty rating of approximately 4 across both tasks.

The MATLAB assignments

Students reported that the MATLAB image generation tasks, Figures 3(a) and 3(b), were the most difficult, with a difficulty score of around 6, followed by the third MATLAB task, the roulette simulation (Figure 3(c)), which scored around 5.

The HTML and JavaScript assignments

Students on aggregate reported the webpage components of the assignments with a difficulty rating of approximately 5 across all components, noting that embedding slides onto the webpage was perceived as easier than using Google Chart Library or editing the webpage into its final form.

Qualitative thematic analysis of perceived difficulty

Theme 1: The challenge of having to learning something new

Perhaps unsurprisingly, the most frequent sentiment expressed across the free-text responses, occurring in 37% of responses, was that it is hard to learn new things, with many of these responses directly referring to 'coding', 'MATLAB', or 'computing'. For example the comments: 'Didn't do coding ever so struggled'; 'MATLAB was hard due to my unfamiliarity with the code beforehand'; 'I've never done any form of coding before, which meant that I have zero background on how to code and had to look up the meanings for many things'; 'All of the computing. Didn't know any of the topics prior to joining the foundation year', highlight this sentiment. Where a coding method was explicitly mentioned, MATLAB was the most frequently referenced, which is in line with the higher perceived difficulty of these tasks, notably the generation of the images (Figure 3(a) and 3(b)). Example responses are: 'The MATLAB images must be created in a single run, without doing differ[ent] shapes at different times', or 'MATLAB coursework Task 2'. The next most commonly mentioned within this theme were aspects of HTML and JavaScript assignments, particularly the requirement to embed their PowerPoint slides onto their webpages, which was mentioned in around 8% of responses. This is interesting given that the aggregate perceived difficulty of this task was notably lower than most of the other tasks completed.

As mentioned, our Foundation Year does not require any computing prerequisites, and resources are written with this in mind. Students frequently acknowledged that the things they found hardest were the things in which they had the least prior knowledge. However, this should be capitalised on, supporting students to build the resilience they will need for their further studies (Martin and Marsh, 2006; Cassidy, 2015).

Theme 2: *Didn't like the guided resources*

The next most common theme, with 28% of responses, was that students did not like the guided resources. These resources are step-by-step worksheets, which explore different aspects of the software used. The intention is to guide students at their own pace through the exercises, as well as offering a persistent resource they can refer to for coursework assessments. Additionally, two one-hour timetabled sessions with groups of thirty students are run by academics and PhD demonstrators as students navigate these worksheets. Indicative responses suggest, however, that the potential pedagogical benefits of this design were not realised, or at least not consciously realised, by some students: 'MATLAB as it just wasn't explained, and expected to teach it to ourselves'; 'MATLAB needed more practice than the worksheets', and

> The MATLAB coursework was the one I found most challenging, the colour image I found difficult because it took a lot of trial and error to finally get close to getting it right, the binary image was difficult because I had forgotten how to code it, so I had to keep referring to what was learnt in lesson.

This last statement reinforces that many students perceive something as more difficult when it is unfamiliar, because the marks for these assessments were generally very high. It is also worth noting that, according to academic teachers in these computing workshops, attendance was generally quite poor, with around 30% of those registered regularly turning up. This may exacerbate the feeling of not understanding how to engage with the resources. This is not exclusive to the MATLAB worksheet; some responses indicated a similar dissatisfaction with the Excel and HTML/JavaScript worksheets. For example, '…HTML worksheets often had steps that weren't explained', and perhaps the best summary: '…it was quite time consuming and hard to do as we had to figure out it ourselves.' In some ways this is true; students used the guided resources to explore the tasks, but in other ways, does not acknowledge the individual opportunities for help which were embedded into the teaching structure of this module, notably the workshops. This is an important area for the teaching staff to review; students are voicing an opinion that the resources are not aligning well with their expectations or preferences for study, but it is not immediately obvious where this mismatch arises. The evidence is clear, a proportion of students indicate dissatisfaction, and many students regularly miss timetabled sessions, but the reason needs to be better understood. Indeed, the perspective of the teaching staff who run these workshops is that queries and difficulties were usually quickly resolved within a session, and that persistent absentees were missing an important peer learning and collaboration opportunity (Tarling et al., 2023). Researchers investigating computational thinking skills in an engineering context suggest that '…educators need to support students better to analyse their designs, identify relevant issues and resolve them' (Zhu et al., 2025). They highlight that instructing students in strategies for troubleshooting requires further research and consideration (Zhu et al., 2025). Therefore, further scaffolding of the troubleshooting process in timetabled sessions could encourage attendance, allowing students to build skills and experience in 'figuring things out'.

Theme 3: *Time management*

The final theme generated across 16% of the responses was students reflecting that their time management caused them difficulties across the assignments. For example, not planning their time ahead of key deadlines, 'Not dividing my workload', 'Not being able to manage my time as I should have,' and 'Leaving things till the last week'. And of

course, some noted their low attendance because of the 'early morning lectures and computing sessions.'

Student perception of skill development

To understand how students reflect on the skills they have developed through this module, they were asked to rate their improvement against a list of digital skills (Table 1) and academic skills (Table 2): 'By completing the ENG0018 module, I have improved my ... [Digital skills].' Students were then asked, 'In your words, what were the most useful things you learned during the ENG0018 module?' so that themes could be generated around the areas of skill development identified by students. A total of 141 free-text responses were received in response to this follow-up question.

Table 1. Students were asked how much they agreed with the statement, 'By completing the ENG0018 module, I have improved my… [Digital skills]'. There were 147 responses. SA = strongly agree, A = agree, N = neutral, D = disagree and SD = strongly disagree.

	% Agreement				
Digital skills	**SA**	**A**	**N**	**D**	**SD**
Knowledge about what the internet is and how it works	14	48	25	9	4
Knowledge about Machine Learning and AI	10	47	29	12	3
Awareness of the importance of computing skills to professional development and employability	22	48	17	9	3
Problem solving: Being able to break a problem down into steps and finding a way of working through each step	22	46	25	5	1
Translation: Being able to convert given instructions into a process that can be carried out computationally	20	53	21	4	1
Implementation: Being able to implement your process as computer code (e.g., using MATLAB)	26	49	18	5	3
Fault finding: When your code doesn't work as expected, being able to narrow down and identify what is causing the problem	22	47	22	7	2
Verification: Checking that your task (e.g., code/HTML) meets the written design specifications	18	61	17	3	1
Validation: Checking that output from your code (e.g., a numerical result) is accurate	19	55	21	3	1
Awareness of the importance of visualisation for identifying logical errors	22	53	22	1	2

Table 2. Students were asked how much they agree with the statement, '*By completing the ENG0018 module, I have improved my... [Academic skills]*'. There were 147 total responses.

	% Agreeability				
Academic skills	**SA**	**A**	**N**	**D**	**SD**
Using SurreySearch (library) to find sources of information	42	35	15	5	2
Understanding about using credible sources of information	31	47	18	3	1
Developing an understanding of a topic through research and reading of multiple sources of information	29	52	17	1	1
Preparing an academic written article suitable for a specific audience	21	60	16	3	1
Seeking and using feedback	18	48	29	5	1
Academic referencing (e.g., Harvard style) – principles and application	33	39	19	7	1
Writing an abstract	24	52	19	4	1
Using Turnitin similarity checker as a tool for helping with writing	34	41	20	3	3
Writing an introduction and conclusion	17	47	29	7	1
Presentation skills – preparation and delivery	15	48	28	5	3

Quantitative perception of skill development

It is perhaps unsurprising that there is student agreement on aggregate that the twenty skills listed have all been developed to some extent, given that these are the purpose of the module. Still, it is encouraging that students agree that these skills have been developed. The skills that displayed the greatest variation (although still with overall agreement) were 'Knowledge about what the internet is and how it works' and 'Knowledge about Machine Learning and AI.' This might be explained by the fact that these knowledge-based skills were centred on single, one-off lectures for each, as opposed to the other skills listed, which were developed over several teaching sessions and activities.

Qualitative thematic analysis of skill development

We considered thematically the qualitative comments related to development of the digital skills listed in Table 1.

Theme 1: Learning to program

By far, the most frequent response referred to the theme of learning to program, mentioned in 45% of the responses. Despite the frequent occurrence of this theme, students often responded briefly, stating 'MATLAB', 'HTML' or simply 'coding', but several respondents provided additional context for why they mentioned these terms, for example, 'I was given a strong understanding of coding, coming from not knowing

anything about coding', and 'I think learning about the real world utilisation of programs such as MATLAB allowed me to gain a better insight into why the work we are doing is important.' Given the prevalence of the 'learning something new' theme for self-reported challenges, it is likely that many students had little to no coding experience and therefore identified that the acquisition of this skill (regardless of how far they developed it) represented a clear learning gain for them. In comparison, a student who had some experience, which developed further, may be less likely to identify a significant learning gain. Similarly, in 2% of responses, students referred to 'AI' but did not elaborate further, perhaps suggesting acquired knowledge of the subject, rather than any specific skill development.

Theme 2: Designing and managing a website

A clear reference to developing skills associated with designing and managing a website was identified in 7% of responses. Whilst there will be some overlap between this theme and learning to program, explicit references to websites differentiate it from programming. For example, 'How to design and manage a website', 'How to make a website', and 'Making a webpage.' These responses, therefore, map onto a tangible product of their programming, as opposed to the sense of skill acquisition that many alluded to in the 'learning to program' theme.

Reflections and recommendations

Reflecting on the students' perceptions of their digital and academic skills development in this foundation year module, there are a few areas that could be explored further by the teaching team to inform future curriculum design, and which may be beneficial insights for those who lead modules at other institutions.

Attendance

There is a well-documented link between attendance, engagement, and attainment across the higher education sector (Summers, et al., 2021), with much discussion about more recent declines in student attendance and the potential contributing factors to this (Otte, 2024). Whilst these sector-wide issues should be acknowledged as potentially contributing to the poor attendance at the workshops for the module studied in this paper, it is also important to question and critique the design of the sessions and resources. No matter how well-designed a module may be, if students are not fully engaging with the learning and skills development opportunities within the sessions and resources, then the success of the approach needs to be reviewed. The recent introduction of university-wide student attendance monitoring and attendance and engagement procedures has seemingly had minimal effects on attendance or on the engagement of students with their learning. Ideally, the voice of those students

who did not attend timetabled sessions or who faced challenges with submitting assignments on time should be sought. This presents challenges in engaging with the 'non-engaged' but could be addressed through personal tutor meetings or through peer-led mentoring schemes already in place within the Foundation Year; students may be more willing to share insights into the reasons behind non-attendance with their peers than with academic staff members.

Student expectations

One theme that emerged from this study was that learning new things is challenging. Checking alignment between academic and student expectations is addressed in general sessions in the Foundation Year, but there may be a benefit in reinforcing these messages within this module, for example, what to do if you follow the instructions on the worksheet but your code does not do what you expected.

Troubleshooting process

Linked to the previous point of student expectations is guiding students through how to troubleshoot their own work. The development of computational thinking skills may be problematic for some students who have navigated their previous educational experience through memorising information rather than developing skills in learning and thinking (Baker, 2022), and acknowledging that, regardless of technological differences between generations of learners, students are not digital natives (Kirschner and Bruyckere, 2017). Therefore, students may benefit from a more explicit framework for troubleshooting and problem-solving in computer coding, alongside the clearly laid out expectation that this process is valued more than the result of producing a piece of code that 'works.' Indeed, could mark schemes be produced that reward students navigating this troubleshooting process over producing a correct answer?

Learning Resources

Comments from students about the worksheets and learning resources may be addressed through exploring the three previous points but are also worth considering separately. The visual appeal, clarity, indexing and accessibility of resources should be reviewed. E-book versions of worksheets for MATLAB could be developed (as was done for the Excel component), allowing students to choose the format that appeals to them. Students should be involved in the review and development of these resources.

Concluding remarks

This paper reports on a deliberately designed, integrated curriculum that fosters both digital and academic skill development within a STEM foundation year. Through a series of authentic assessments, students are exposed to the practical applications of

several competencies. As might be expected, students found the most challenging components of the module to be those that demanded new ways of thinking and problem-solving. The programming tasks in MATLAB were frequently reported as the most difficult. The student-reported data demonstrate that the areas of high challenge correspond directly to the areas of greatest perceived skills development. Ultimately, this work provides an example of how an integrated curriculum, built upon a foundation of authentic assessments, can effectively embed essential digital skills. The challenges students face, and report on, can be seen as an indicator that they are actively engaging in the demanding process of skill acquisition, and as a result, emerging more confident and capable learners. Furthermore, exposure to professional tools which computer scientists (among others) will use increases the authenticity of the assessments.

Perhaps the student experience is best expressed in the words of one respondent:

> This course reminded me of my love hate relationship with my keyboard. Sometimes lovingly tapping away at the keys, sometimes wishing it was alive so it could feel the pain when I punch it :)

Acknowledgements

We thank all the students from the 2023-24 and 2024-25 academic years who shared their thoughts on their digital and academic skill development.

References

Baker, L.A. (2022) The utility of distributed practice in curriculum design and effective learning strategies, *Impact: Journal of the Chartered College of Teaching*, 1(14), pp. 17–20.

Baker, L.A. (2025) Integrating educational dialogue into a classic random walk activity, *EdArXiv*, pp. 1–8. https://doi.org/10.35542/osf.io/a84u7_v1.

Bosse, Y., Redmiles, D. and Gerosa, M.A. (2019) Pedagogical content for professors of introductory programming courses, in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 429–435.

Cansu, F.K. and Cansu, S.K. (2019) An overview of computational thinking, *International Journal of Computer Science Education in Schools*, 3(1), pp. 17–30.

Cassidy, S. (2015) Resilience building in students: The role of academic self-efficacy, *Frontiers in psychology*, 6, p. 1781.

Dampier, G., Baker, L.A., Spencely, C., Edwards, N.J., White, E. and Taylor, A.M. (2019) Avoiding the Deficit Model and Defining Student Success: Perspectives from a New Foundation Year Context, *Journal of the Foundation Year Network*, 2, pp. 41–52.

DfE, (2014) National Curriculum https://www.gov.uk/government/collections/national-curriculum (accessed 9th march 2026).

DfE (2023) *Provisional entries for GCSE and A level summer 2023 exam series*. London: Department for Education.

Felder, R.M. and Brent, R. (2016) *Teaching and Learning STEM: A Practical Guide*. 1st edn. San Francisco, CA: John Wiley and Sons, Inc.

Kandemir, C.M., Kalelioglu, F. and Gülbahar, Y. (2021) Pedagogy of teaching introductory text-based programming in terms of computational thinking concepts and practices, *Computer Applications in Engineering Education*, 29(1), pp. 29–45.

Kemp, P. E. J., Wong, B., Hamer, J., and Copsey-Blake, M. (2024) Computing provision report: English school data 2011-2023, *King's College London* [Preprint]. https://scari.sites.er.kcl.ac.uk/cpre/chapters/access.html#gcse (Accessed 30 Nov. 2025).

Kirschner, P.A. and Bruyckere, P.D. (2017) The myths of the digital native and the multitasker, *Teaching and Teacher Education*, 67, pp. 135–142.

Martin, A.J. and Marsh, H.W. (2006) Academic resilience and its psychological and educational correlates: A construct validity approach, *Psychology in the Schools*, 43(3), pp. 267–281.

Otte, J. (2024) I see little point: UK university students on why attendance has plummeted. https://www.theguardian.com/education/article/2024/may/28/i-see-little-point-uk-university-students-on-why-attendance-has-plummeted (Accessed 12th Jan. 2026).

Overland, E.E. (2025) *The impact of the Computing National Curriculum in English secondary education. A comparative case study*. Manchester Metropolitan University.

Roberts, E., Lilly, J. and Rollins, B. (1995) Using undergraduates as teaching assistants in introductory programming courses: An update on the Stanford experience, in *Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education*, pp. 48–52.

Robins, A., Rountree, J. and Rountree, N. (2003) Learning and teaching programming: A review and discussion, *Computer science education*, 13(2), pp. 137–172.

Shute, V.J., Sun, C. and Asbell-Clarke, J. (2017) Demystifying computational thinking, *Educational research review*, 22, pp. 142–158.

Spencely, C., Baker, L.A. and Harrison, R. (2023) The Life and Times of 'The Conference Project' for Engineers, Physical Scientists and Mathematicians, *Journal of the Foundation Year Network*, 6, pp. 29–38.

Summers, R.J., Higson, H.E. and Moores, E. (2021) Measures of engagement in the first three weeks of higher education predict subsequent activity and attainment in first year undergraduate students: a UK case study, *Assessment & Evaluation in Higher Education*, 46(5), pp. 821–836.

Tarling, G., Melro, A., Kleine Staarman, J. and Fujita, T. (2023) Making coding meaningful: university students' perceptions of bootcamp pedagogies, *Pedagogies: An International Journal*, 18(4), pp. 578–595.

TeachFirst (2025) Teacher shortages deny nearly a third of poorer pupils' access to computer science A-level. https://www.teachfirst.org.uk/press-release/missing-teachers (Accessed 9th March 2026).

Wing, J.M. (2008) Computational thinking and thinking about computing, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), pp. 3717–3725.

Wootton, A.J. (2021) Authentic assessment: A foundation year case study, *Journal of the Foundation Year Network*, 4, pp. 75–85.

Zhu, G., Kow, J.F., Fan, X., Yeter, I.H., Chit, L.S. and Ong, Y.S. (2025) Exploring Undergraduate Students' Computational Thinking Skills Across Engineering Design Processes, *Computer Applications in Engineering Education*, 33(3), p. e70035.

About the Authors

Dr Lewis A. Baker completed his MPhys in 2013 along with an MSc and PhD in Mathematical Biology and Biophysical Chemistry at the University of Warwick in 2017. He then qualified as a secondary school teacher before combining these experiences as a Teaching Fellow in the Faculty of Engineering and Physical Sciences inaugural Foundation Year at the University of Surrey in 2019, and Senior Lecturer in 2022. He has been recognised as a Senior Fellow of the Higher Education Academy (SFHEA) and a Chartered Science Teacher (CSciTeach).
ORCID: 0000-0002-0097-4011

Dr Carol Spencely is currently a Senior Lecturer in Learning Development for the Faculty of Engineering and Physical Sciences Foundation Years at the University of Surrey. Carol originally trained as a clinical immunologist and completed a BSc (Hons) in Physiology and Pharmacology, and a MSc in Pathological Sciences. Following a PhD in Immunology at Liverpool and research and teaching positions at Imperial College London, Carol moved into professional services and helped to set up the Postdoc Development Centre at Imperial. Carol joined the University of Surrey's Doctoral College in 2012 before taking up her role on the Foundation Year in 2018.
ORCID: 0000-0002-0773-3632
c.spencely@surrey.ac.uk

Dr Alifah Rahman is a lecturer at the University of Surrey with nearly twenty years of experience in physics and computing education. She earned her PhD in Physics from the University of Warwick in 2019. She leads all computing modules for the Foundation Year, designing inclusive, skills-focused curricula that integrate programming, digital tools, and e-learning to foster engagement, computational thinking, and applied problem-solving.

Dr Richard Harrison is an Associate Professor and the Senior Programme Lead on the suite of Foundation Year Programmes in the Faculty of Engineering and Physical Sciences at the University of Surrey. He also chairs the Institutional Mathematics Education Group and is a Fellow of the Institute of Mathematics & its Applications. He has extensive industrial and academic experience as a Mathematician and Computer Scientist both in the UK and overseas.