SN Computer Science (2021) 2:75
https://doi.org/10.1007/s42979-021-00461-7

REVIEW ARTICLE

Teaching Mathematics to Computer Scientists: Reflections and a Case Study

Ferrante Neri[1]

Received: 30 September 2020 / Accepted: 8 January 2021

Abstract

Mathematics, despite being the foundation of computer science, is nowadays often considered a totally separate subject. The fact that many jobs in computer science do not explicitly require any specific mathematical knowledge posed questions about the importance of mathematics within computer science undergraduate curricula. In many educational systems, a prior high school knowledge of mathematics is often not a mandatory requirement to be enrolled into a degree of computer science. On the other hand, several studies report that mathematics is important to computer scientists since it provides essential analytical and critical skills and since many professional and research tasks in computer science require an in-depth understanding of mathematical concepts. From this assumption, this article proposes an analysis of the cohort of computer science' students, with a specific reference to British Universities, and identifies some challenges that lecturers of mathematical subjects normally face. On the basis of this analysis this article proposes two teaching techniques to promote effective learning. The proposed techniques aim at addressing the diversity of cohorts in terms of mathematical background and skepticism from part of the cohort of students to consider mathematics as an essential element of their education. Numerical results indicate the validity and effectiveness of the proposed teaching techniques.

Keywords Education · Mathematics for computer science · British educational system · Research informed teaching

Introduction

In the 1830s, Charles Babbage developed the idea of an automatic calculator and in the 1840s Ada Lovelace conceptualised computer programming. These scientific contributions are allegedly the first visionary foundations of computer science [24]. However, the beginning of modern computer science is usually dated about one century later, when Alan Turing and Alonzo Church introduced the concepts of *algorithm* and *model of computation*, see [12, 13, 49]. An important stepping stone from theoretical model to hardware implementation is that the *computer architecture* formalised by John von Neumann in the 1940s [32].

These pioneers of computer science have something in common: they were all mathematicians. Hence, the research published at the time was presented and perceived as part of mathematics. Thus, we may observe that computer science originated as a branch of mathematics that over the second half of the twentieth century became a discipline separate and independent from it.

On the other hand, when we analyse computer science today, it appears like a broad and complex subject composed of heterogeneous parts and whose specialists possess diverse and heterogeneous skills. For example, among the plethora of its sub-fields, computer science (and its taught curricula) includes subjects very close to mathematics like theory of computation and algorithmics [46], programming subjects whose focus is in the computer implementation and hardware exploitation [36], subjects that focus on the human user, their psychology and aesthetic preferences to build efficient front-end interfaces [9].

By analysing the job market in computer science, many of the jobs most in demand, like *Applications developer*, *Game designer/developer*, *Information systems manager*, *IT consultant* do not require any specific mathematical training. Hence, by echoing the (rhetorical) question posed by Anthony Ralston in [38]:

Do We Need ANY Mathematics in computer science Curricula?

✉ Ferrante Neri
ferrante.neri@nottingham.ac.uk

[1] COL Laboratory, School of Computer Science, University of Nottingham, Nottingham NG8 1BB, UK

Published online: 05 February 2021

The answer to this question is not straightforward and is controversial, see [37, 39]. Ralston acknowledges the importance of mathematics in computer science degrees and points out that it is important "to insure that mathematics does play a proper role in CS/SE programs and, in particular, to do so by breaking the stranglehold of calculus on first and second year college mathematics". By paraphrasing this statement, mathematics should harmonically sit within a computer science degree taking into account the learners, the job market, and the nature of the subject.

The role of mathematics within computer science education has been recently discussed by Lincoln Sedlacek in [45] where it is stated that mathematics is an essential subject of computer science education and the following four reasons are given

- Mathematics teaches understanding and communication through an abstract language. This general argument, also mentioned in [38], means that mathematics "rewires the brain" of the learner and enables a general broader understanding, see in the context of school education [3]. The abstract nature of programming and other areas of computer science would greatly benefit from this skill.
- Mathematics teaches how to work with algorithms. Algorithms are a fundamental part of computer science and appear explicitly or implicitly in most computer related tasks. The skill of conceptualising algorithms as mathematical entity helps to better understand and solve these tasks, [3, 19, 26].
- Mathematics teaches computer scientists how to analyse their work. The analytical skills provided by the study and understanding of mathematics enable students to strengthen their critical skills. These skills are useful to programmers, designers, and developers to assess their own work and that made by others to identify mistakes and areas for improvement, see [15, 47].
- A lot of computer science still involves mathematics. Many computer-related tasks require knowledge and understanding of mathematics. For example, the programming of 3D graphics and animation in games requires the implementation of mathematical equations [17, 27]. There is a degree of presence of mathematics in various computer science tasks such as cybersecurity [5, 40], artificial intelligence [18, 43] and data science [11].

While assuming, on the basis of considerations above, that some degree of mathematics provision is crucially important in computer science education, the present paper offers reflections about how mathematics can be effectively and efficiently taught to computer science undergraduates. In other words, this paper addresses the following research question:

How to successfully teach mathematics to computer science undergraduates?

This research question makes an implicit assumption: there is a specific way to efficiently teach mathematics in a computer science degree (which would differ from the way mathematics is taught to mathematics students). More generally, this article puts the learners at the centre of the attention of the lecturer who adapts their teaching on the basis of the inclinations (what they easily understand) needs (what can be useful in their professional life) of the cohort. This is in line with the study reported in [10] where some tangible tools are proposed to enhance the understanding of mathematics among engineering students.

To address this question, this paper proposes an analysis of the features of a computer science undergraduate cohort and two teaching techniques that, on the basis of the experience of the author, promote a large-scale engagement, understanding of mathematics, and improved exam results.

To further clarify the main purpose and significance of this study, mathematics, albeit very impactful on the careers of computer scientists, is often overlooked in computer science' curricula and its importance in teaching practice often not enough recognised.

In the literature, numerous studies are devoted to the teaching of mathematics with several journals focussed solely on mathematics' education. The link between mathematics and computer science/engineering has also been intensively studied. However, the most popular approaches revolve around the use of computer technologies to enhance the learning of mathematics, see, e.g. [21, 34, 44]. Furthermore, several books of mathematics refer to a computer science audience, e.g. [22, 50], thus implicitly proposing examples of teaching practice. The present paper proposes the first study, to the knowledge of the author, that conceptualises some educational techniques specific to the teaching of mathematics to computer science' cohorts.

The remainder of this paper is organised in the following way. The next section provides some observations about cohorts of undergraduate students of computer science and their attitude towards modules of mathematics. The subsequent section outlines the developed teaching techniques.

Computer Science Cohorts

As a premise of this work, the observations reported in this section are the result of a decade of teaching experience of mathematics in Schools of Computer Science across two British institutions, De Montfort University and the university of Nottingham. During this time, the author published a textbook entitled "Linear Algebra for Computational Sciences and Engineering" [28] which then has

been substantially re-written in a second edition by taking into account the feedback of multiple cohorts of students, see [29].

With respect to the learning of mathematics, the following challenges associated with the (often large) cohorts of students have been noted:

- Since in many universities **there are no specific mathematical pre-requisites, the cohorts can be very diverse in terms of mathematics' background**. Some students may have encountered advanced mathematical studies in high school (A levels in further maths), some others may have studied basic mathematics in high school and others may have not studied mathematics at school in the two years immediately preceding university education. Furthermore, international students may have a strong mathematical background and have not necessarily met the same content as local students in their high schools, see [7, 25].

  Thus, the preparation of a lecture of mathematics that is suitable for the entire cohort is a challenging task. The lecture is likely to be either excessively demanding for some students or not stimulating enough for others. The search for the correct balance can easily lead to ineffective learning since it would not target large portions of the cohorts.
- In continuity with the Ralston's observations [38], part of the cohort is likely to not fully appreciate the importance of mathematics within their curriculum. To the experience of the author, **many computer science students, especially in the early undergraduate years, do not see the benefits of mathematics to their future career**. Mathematics is sometimes perceived as an abstract subject that has no relation at all with the work of a professional computer scientist.

  Another challenge for the lecturer is to motivate the entire cohort and overcome the initial resistance of many students to learn mathematics. This attitude may also link to individual psychological issues such as maths anxiety, see, e.g. [23, 48] in case of students who have not studied any mathematics in the two years preceding the university studies.

On the other hand, these challenges can be mitigated by an important feature of the cohort: since there are normally prerequisites in computer science discipline, the entire cohort is guaranteed to have a minimum understanding of programming, Information Technology, and computing disciplines. In the opinion of the author, this feature can be exploited by the lecturer of mathematics designing a module that is interesting and engaging for all the students and contains new learning material and approaches for the entire cohort of students.

Teaching Mathematics to Computer Scientist: Two Proposed Techniques

This section describes at the conceptual level and by means of a concrete example two proposed teaching techniques used to address the two challenges outlined in Sect. "Computer Science Cohorts".

Addressing the Diversity in Mathematical Background

The research question above is broken into two question to address the challenges outlined in Sect. "Computer Science Cohorts". With reference to the first challenge and with the purpose of proposing a technique addressing it, let us formulate the first research sub-question.

> *How to design a lecture (entire module) that is interesting for a cohort with a diverse mathematical background and promote the learning for all the students regardless of their starting point?*

The first underpinning principle embraced by the author in his teaching and in his textbook [29] (as explicitly declared on the back cover), is that no compromises should be made on the content nor on the mathematical rigour of the lectures. To enable that computer science students benefit in their career from modules of mathematics, it is fundamental that the four points outlined by Lincoln Sedlacek [45] are covered. This means that a number of mathematical topics relevant to computer science are presented and assessed. Furthermore, rigorous mathematical reasoning must be used throughout the mathematical modules and be part of the assessment. This is done to allow students to develop analytical and critical skills that will then be transferred to their professional life.

On the other hand, in the opinion of the author, the way mathematics is taught to students of computer science should take into great consideration the composition and features of the audience/cohort. To address the diversity in mathematical background, the author proposes to introduce and explain each mathematical topic in three different ways and from different perspectives. More specifically, each topic is presented

- *By formal mathematics*. This presentation immediately targets that part of the cohort with prior mathematical studies and is available to the other students after they achieved an intuitive understanding of the concept.
- *By abstraction of an example*. This presentation allows an initial understanding to the students without solid prior mathematical bases. These students have an opportunity to quickly achieve some degree of understanding of the

explained mathematical concept and remain engaged throughout the lecture. Then, following an initial understanding of the subject, these students can revise the formal presentation of the concept and understand it more in depth and at a more general level. In the meantime, students with a solid mathematical background have the opportunity to check and consolidate their understanding of the formal presentation by seeing this second presentation as its numerical example.

– *By "algorithmification" of formal mathematics*. Mathematical concepts and proofs can be interpreted and presented as procedures/algorithms that achieve a numerical result or a logical goal. Since the entire cohort is already familiar with programming, and procedural description of instructions, the author exploits the common background of the cohort to offer an alternative (and original) view of the subject that is easily accessible to everybody. It must be remarked, that this algorithmification, albeit a powerful teaching tool, always allows a procedural understanding of mathematics, i.e. what needs to be done to achieve a goal, but not always allows an in-depth understanding of the concept for which a revision of the formal presentation may be necessary. On the other hand, the algorithmification of mathematics enables the development of a common language, understandable by all students and offers a further support to better learn and understand rigorous mathematics.

Figure 1 displays in a schematic way the proposed teaching technique and displays the three ways the mathematical concept is explained, categorising the learners on the basis of their mathematical background. Two learning phases are included, a first approach where the students are introduced to the topic and revision where the students study the topic again after having familiarised with the multiple explanations. As shown, in the first phase, students with a mathematical background are expected to prefer a formal approach whereas students without a mathematical background are likely to prefer an intuitive explanation. During the revision, the background becomes less relevant since the students had the opportunity to study the concept and reflect about it. In revision phase, students are expected to choose the approach they prefer on the basis of their personal inclinations and are expected to refer to both formal and intuitive approach to study the concept from complementary perspectives. The explanation by algorithmification is expected to be easily accessible for the entire cohort and be a further form of support to enable another level of understanding of the subject. The proposed approach is in agreement with the inclusive education theories [2] and in particular with the *cognitivism-based inclusive education practices* and *constructivism-based inclusive education practices*. The former focuses on the mental information processing of the learners, see [1] while the latter makes use of real-life experiences as learning tools, see [14].

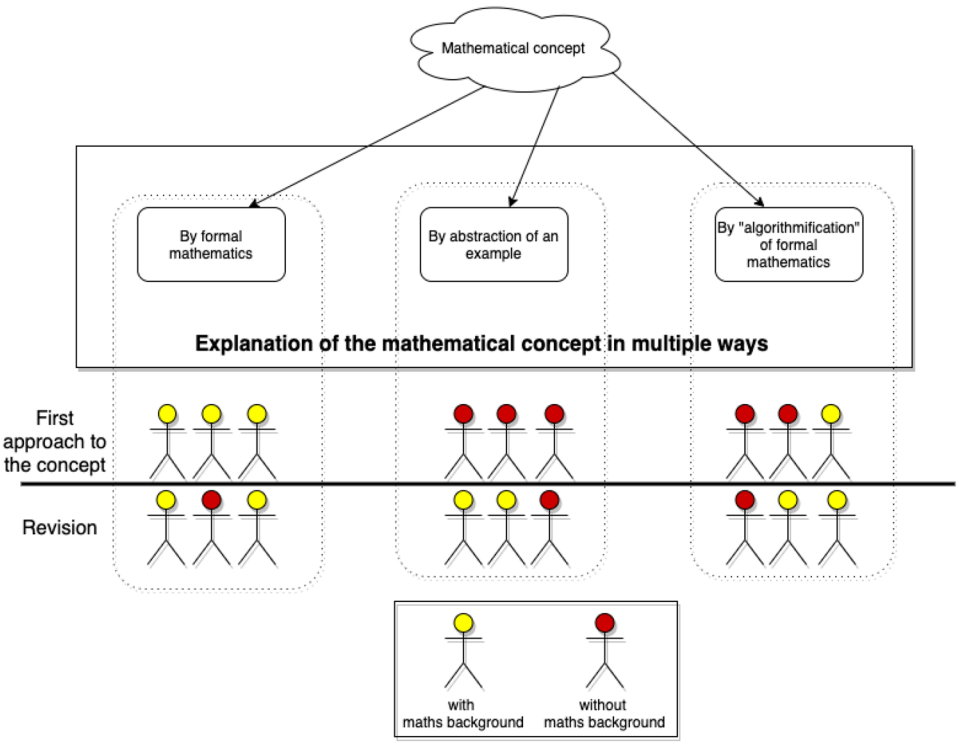

Fig. 1 Scheme of the teaching technique to address the diversity in mathematical background

To better demonstrate the proposed teaching technique, in the following example a mathematical concept is explained in the three different ways outlined above.

Example: LU Factorisation Explained to a Computer Science Cohort

Let us consider a popular topic in mathematics which is fundamental in the career of a computer scientist, that is the solution a large system of linear questions. Let us assume that the problem has been presented as

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = b_2 \\ \ldots \\ a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = b_n \end{cases}$$

that is a matrix equation of the type $\mathbf{Ax} = \mathbf{b}$. In the following, the solution of this problem by a direct method called **LU** factorisation is presented, see [29]. At first, a general premise is made and then the concept is explained by means of the three different ways explained above.

Premise. The LU factorization is a direct method that transforms a matrix **A** into a matrix product **LU** where **L** is a lower triangular matrix having the diagonal elements all equal to 1 and **U** is an upper triangular matrix. Thus, if we aim at solving a system of linear equations $\mathbf{Ax} = \mathbf{b}$, we obtain

$$\mathbf{Ax} = \mathbf{b} \Rightarrow$$
$$\Rightarrow \mathbf{LUx} = \mathbf{b}.$$

If we pose $\mathbf{Ux} = \mathbf{y}$, we solve at first the triangular system $\mathbf{Ly} = \mathbf{b}$ and then extract **x** from the triangular system $\mathbf{Ux} = \mathbf{y}$. Thus, instead of solving a computationally complex system of linear equations **LU** factorisation transforms **A** into the product **LU** and then poses two extremely straightforward systems (triangular systems are immediate to solve by substitution).

Explanation by formal mathematics.

Theorem 1 *Let* $\mathbf{A} \in \mathbb{R}_{n,n}$ *be a non-singular matrix. Let us indicate with* $\mathbf{A_k}$ *the submatrix having order k composed of the first k rows and k columns of* **A**. *If* $\det \mathbf{A_k} \neq 0$ *for* $k = 1, 2, \ldots, n$ *then* $\exists!$ *lower triangular matrix* **L** *having all the diagonal elements equal to* 1 *and* $\exists!$ *upper triangular matrix* **U** *such that* $\mathbf{A} = \mathbf{LU}$.

Let us now derive the general transformation formulas. Let **A** be

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n,1} & a_{n,2} & \ldots & a_{n,n} \end{pmatrix}$$

while **L** and **U** are, respectively,

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ l_{2,1} & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ l_{n,1} & l_{n,2} & \ldots & 1 \end{pmatrix}$$

$$\mathbf{U} = \begin{pmatrix} u_{1,1} & u_{1,2} & \ldots & u_{1,n} \\ 0 & u_{2,2} & \ldots & u_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & u_{n,n} \end{pmatrix}.$$

If we impose $\mathbf{A} = \mathbf{LU}$, we obtain

$$a_{i,j} = \sum_{k=1}^{n} l_{i,k}u_{k,j} = \sum_{k=1}^{\min(i,j)} l_{i,k}u_{k,j}$$

for $i, j = 1, 2, \ldots, n$.

In the case $i \leq j$, i.e. in the case of the triangular upper part of the matrix, we have

$$a_{i,j} = \sum_{k=1}^{i} l_{i,k}u_{k,j} = \sum_{k=1}^{i-1} l_{i,k}u_{k,j} + l_{i,i}u_{i,j} = \sum_{k=1}^{i-1} l_{i,k}u_{k,j} + u_{i,j}.$$

This equation is equivalent to

$$u_{i,j} = a_{i,j} - \sum_{k=1}^{i-1} l_{i,k}u_{k,j}$$

that is the formula to determine the elements of **U**.

Let us consider the case $j < i$, i.e. the lower triangular part of the matrix

$$a_{i,j} = \sum_{k=1}^{j} l_{i,k}u_{k,j} = \sum_{k=1}^{j-1} l_{i,k}u_{k,j} + l_{i,j}u_{j,j}.$$

This equation is equivalent to

$$l_{i,j} = \frac{1}{u_{j,j}}\left(a_{i,j} - \sum_{k=1}^{j-1} l_{i,k}u_{k,j}\right)$$

that is the formula to determine the elements of **L**.

Explanation by abstraction of an example. If we consider the following system of linear equations

$$\begin{cases} x + 3y + 6z = 17 \\ 2x + 8y + 16z = 42 \\ 5x + 21y + 45z = 91 \end{cases}$$

and the corresponding incomplete matrix $\mathbf{A}$

$$\mathbf{A} = \begin{pmatrix} 1 & 3 & 6 \\ 2 & 8 & 16 \\ 5 & 21 & 45 \end{pmatrix},$$

we can impose the factorization $\mathbf{A} = \mathbf{LU}$. This means

$$\mathbf{A} = \begin{pmatrix} 1 & 3 & 6 \\ 2 & 8 & 16 \\ 5 & 21 & 45 \end{pmatrix} = \begin{pmatrix} l_{1,1} & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} \end{pmatrix} \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{pmatrix}.$$

If we perform the multiplication of the two matrices we obtain the following system of 9 equations in 12 variables.

$$\begin{cases} l_{1,1}u_{1,1} = 1 \\ l_{1,1}u_{1,2} = 3 \\ l_{1,1}u_{1,3} = 6 \\ l_{2,1}u_{1,1} = 2 \\ l_{2,1}u_{1,2} + l_{2,2}u_{2,2} = 8 \\ l_{2,1}u_{1,3} + l_{2,2}u_{2,3} = 16 \\ l_{3,1}u_{1,1} = 5 \\ l_{3,1}u_{1,2} + l_{3,2}u_{2,2} = 21 \\ l_{3,1}u_{1,3} + l_{3,2}u_{2,3} + l_{3,3}u_{3,3} = 45. \end{cases}$$

Since this system has infinite solutions we can impose some extra equations. Let us impose that $l_{1,1} = l_{2,2} = l_{3,3} = 1$. By substitution, we find that

$$\begin{cases} u_{1,1} = 1 \\ u_{1,2} = 3 \\ u_{1,3} = 6 \\ l_{2,1} = 2 \\ u_{2,2} = 2 \\ u_{2,3} = 4 \\ l_{3,1} = 5 \\ l_{3,2} = 3 \\ u_{3,3} = 3. \end{cases}$$

The $\mathbf{A} = \mathbf{LU}$ factorization is then

$$\begin{pmatrix} 1 & 3 & 6 \\ 2 & 8 & 16 \\ 5 & 21 & 45 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 5 & 3 & 1 \end{pmatrix} \begin{pmatrix} 1 & 3 & 6 \\ 0 & 2 & 4 \\ 0 & 0 & 3 \end{pmatrix}.$$

Explanation by "algorithmification" of formal mathematics. The **LU** factorisation can be expressed by the equation

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ l_{2,1} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ l_{n,1} & l_{n,2} & \dots & 1 \end{pmatrix} \begin{pmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,n} \\ 0 & u_{2,2} & \dots & u_{2,n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & u_{n,n} \end{pmatrix},$$

where $\forall i, j$, $a_{i,j}$ are known while $l_{i,j}$ and $u_{i,j}$ must be found. We may consider the matrices $\mathbf{L}$ and $\mathbf{U}$ as data structures that can be viewed as vectors of row vectors $\mathbf{l_i}$ and column vector $\mathbf{u^j}$, respectively

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} = \begin{pmatrix} \mathbf{l_1} \\ \mathbf{l_2} \\ \dots \\ \mathbf{l_n} \end{pmatrix} \left(\mathbf{u^1}, \mathbf{u^2}, \dots, \mathbf{u^n} \right).$$

Let us indicate with $\mathbf{l_i u^j}$ the scalar product of the vector $\mathbf{l_i}$ by $\mathbf{u^j}$ that is $a_{i,j}$:

$$a_{i,j} = \mathbf{l_i u^j} = l_{i,1}u_{1,j} + l_{2,1}u_{2,j} + \dots + l_{n,1}u_{n,j}.$$

If the equations are performed in a certain order, from each scalar product an element $l_{i,j}$ or $u_{i,j}$ can be calculated. Then we may think about an empty data structure $\mathbf{B}$ that will store the representation of the result of the **LU** factorisation. The algorithm initialises the first row of the matrix $\mathbf{B}$ as the first row of $\mathbf{A}$. The following rows of the matrix $\mathbf{B}$ are filled by solving the equations $a_{i,j} = \mathbf{l_i u^j}$ with the data previously calculated and allocated in $\mathbf{B}$. More specifically, each of these equations is a simple linear equation with only one unknown. The value of this unknown is allocated in $b_{i,j}$. At the end of this procedure the matrix $\mathbf{B}$ contains the data of the factorisation:

$$\mathbf{B} = \begin{pmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,n} \\ l_{2,1} & u_{2,2} & \dots & u_{2,n} \\ \dots & \dots & \dots & \dots \\ l_{n,1} & l_{n,2} & \dots & l_{n,n} \end{pmatrix}.$$

Algorithm 1 displays the pseudocode of the **LU** factorisation.

Algorithm 1 Algorithms of the **LU** factorisation

INPUT matrix **A**
Copy the first row of **A** into **B**, i.e. $\mathbf{b_1} = \mathbf{a_1}$
for $i = 2 : n$ **do**
  for $j = 1 : n$ **do**
    if $j < i$ **then**
      from $a_{i,j} = \mathbf{l_i u^j}$ substitute the available values from **B** and calculates $b_{i,j}$ i.e. $l_{i,j}$
    else
      from $a_{i,j} = \mathbf{l_i u^j}$ substitute the available values from **B** and calculates $b_{i,j}$ i.e. $u_{i,j}$
    end if
  end for
end for
OUTPUT matrix **B**

Addressing the Resisting Attitude to Mathematics

With reference to the second challenge, let us formulate the corresponding research sub-question.

> How to keep the full computer science cohort engaged and interested in learning mathematics?

On the basis of trials and errors and observations of the behaviour in the classroom as well as the results at the exam, the author argues that a good strategy is to explicitly highlight the impact of mathematics on the career of a computer scientist. When a mathematical topic is introduced, some context about the practical use of mathematics in computer science should be provided. Two types of contextualisation have been identified.

- **Report the links between mathematics and computer science professions**. As mentioned above, computer science jobs can be of various type. Students are likely to have heard of some types of profession and may even have the ambition of undertaking one of them (or one among some of them). The author observed that references to the links between mathematical theory and computer science professions greatly help to keep the audience engaged and willing to learn.
- **Share personal experience of mathematics in research/profession**. As a computer scientist who actively (and enthusiastically) uses mathematics in his research and profession, the author can share his personal experience. This approach may genuinely interest and enthuse part of the student cohort who may decide to continue their studies in a final year project (thesis) and can be considered part of Research Informed Teaching (RIT). With reference to the theory reported in [6] that classifies different types of RIT, the proposed approach is a combination of *research-led*, *research-oriented* and *research-tutored* learning. The first refers to the illustration of research concept during the teaching, the second refers to the research methodologies, the third refers to critical discussions about research. Furthermore, even when the students do not share the same scientific interest of the lecturer, they may appreciate and participate the passion for the subject that naturally the lecturer would share when talking about their research experience and achievements, see [35, 41]. One of the purposes of sharing the personal professional experience is to be inspirational and promote, among students, reflections about their own skills, passions, and ambitions, see [42].

The proposed approach is in line with the relevance aspect of the Attention Relevance Confidence Satisfaction (ARCS) instructional design model designed by Keller, see [20]. In this model Relevance refers to the usefulness of the information to motivate the learners. Following this principle, the author suggests that the integration of examples related to the prospective careers to the students supports the student to remain motivated and catalyses effective learning sessions.

The following example shows how one of the most abstract and difficult concept of undergraduate mathematics, eigenvalues and eigenvectors, can be linked to computer science profession and research.

Example: The Importance of Eigenvectors in Computer Science Profession

Before entering into the details, let us informally introduce the context of the topic. When a multivariate linear mapping is considered, its eigenvector is a special direction along which the function behaves like a multiplier of a scalar by a vector [29]. A function of n variables has n eigenvectors. These eigenvectors can be seen a new reference system, a new set of variables that can replace the original one. In this new reference system, the original function (and thus the mathematical model approximating the reality), is very easy

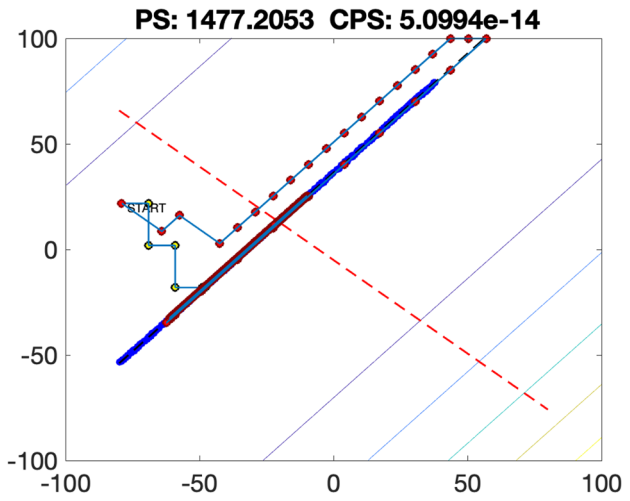

Fig. 2 Functioning and performance of standard Pattern Search (PS) and its enhanced version that exploits the mathematical knowledge about eigenvectors (CPS)

to handle since its variables are independent on each other. This transformation is called *diagonalisation*.

Link between eigenvectors and a computer science profession. One popular profession in computer science is the data scientist. When a large number of data are handled, it is fundamental to extract the most useful piece of information so that the data set can be interpreted correctly. Data can be viewed as multivariate distributions (distributions of vectors) characterised by a mean vector and a covariance matrix. A covariance matrix can be interpreted as a linear mapping and its diagonalisation allows the detection of the direction that best fits the data. This method, commonly known as Principal Component Analysis (PCA) [16], enables the detection of the most represented variables in the dataset that are the most important ones.

Link between eigenvectors and personal experience. Eigenvectors can play a very important role also in the specific research field of the author that is optimisation. When the optimum of a multivariate function is searched, a set of candidate solutions can be interpreted as a multivariate distribution, see [4, 8]. If only a distribution of points whose objective function value is below a threshold (in a minimisation problem) are saved in the data set, then this distribution describes the geometry of the optimisation problem, see [31]. Like for the case of the PCA, the diagonalisation of the associated covariance matrix, that is the detection of its eigenvectors provides the optimisation algorithm with a set of preferential search directions to perform the search for the the optimum, see [30]. However, unlike the case of the PCA, the most important direction (variable) is the least represented one as it would correspond to the direction with maximum directional gradient.

Table 1 Student performance of four cohorts of students over a 5-year span

Exam board	2015	2016	2017	2018	2019
Year 1 module					
Group size	22	36	38	21	–
Pass rate (%)	56	82	87	100	–
First rate (%)	16	39	42	43	–
Average mark $\pm\sigma$	42±28	58±24	62±21	70±18	–
Year 2 module					
Group size	–	16	29	30	21
Pass rate (%)	–	82	76	83	86
First rate (%)	–	31	28	36	38
Average mark $\pm\sigma$	–	59±16	56±21	65±18	66±19

To provide a graphical representation of the research idea, let us consider a problem in two variables and let us assume we generated a set of points whose objective function value is below a certain threshold. Figure 2 shows this distribution as blue points with a simple geometry, that is a line. The dashed lines indicate the directions of the eigenvectors. Then Fig. 2 displays the trajectory of a classical algorithm named Pattern Search (PS) using the standard set of variables (line with yellow markers) and the eigenvectors of the covariance matrix of the distribution. The latter algorithm, namely Covariance Pattern Search (CPS, line with red markers) is identical to PS except it used a different set of variables (it works in a different reference system). We may observe that the version that exploits the mathematics of eigenvectors achieves a result that is seventeen orders of magnitude better than its vanilla version.

Case Study

The outlined teaching techniques have been tested in the classrooms over the years 2014–2019 in the School of Computer Science and Informatics at De Montfort University. More specifically the author designed and taught two modules of 30 Credits each (one fourth of the year credits) to undergraduate students in Years 1 and 2, respectively. Data have been collected for four cohorts of students corresponding to

- Year 1 - Academic Years from 2014/2015 to 2017/2018
- Year 2 - Academic Years from 2015/2016 to 2018/2019

The assessment of each module, in each year, was composed of two classroom tests and one exam. Each piece of assessment had the same structure: 50% of the marks we assigned to numerical exercises while 50% were theoretical questions (which imposed the use of formal mathematics).

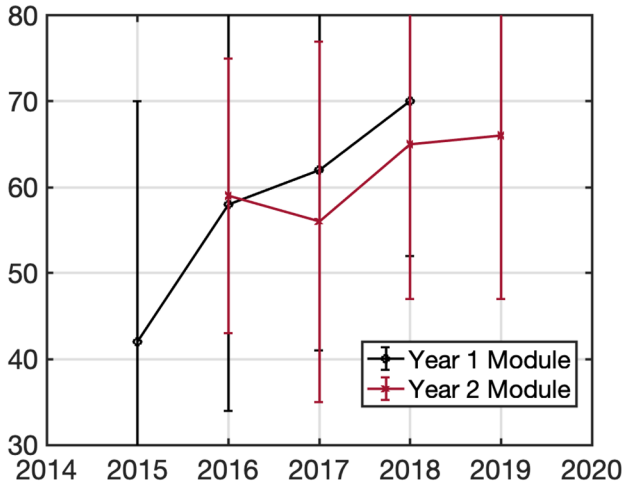

Fig. 3 Trend of the average mark of the students over the years

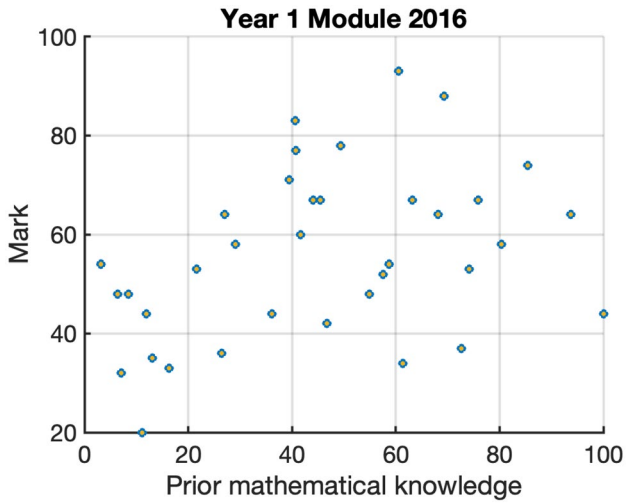

Fig. 4 Scatter plot of the marks against the prior mathematical background for the group of 2016 and Year 1 Module. Although students with prior mathematical knowledge appear to achieve better marks, also students with a limited prior knowledge in mathematics may achieve high marks

The minimum average mark to pass each module is 40/100. While the structure of the assessment remained unaltered throughout the 2014–2019 period, the student feedback coming from explicit comments and through their performance affected the way the modules were delivered by progressively adopting the teaching techniques above. This iterative process affected the classroom teaching as well as the study material, see [29], which eventually presented, for almost each topic, an explanation in multiple ways as in Fig. 1 and its applicability in the context of the computer science job market.

For each cohort, the performance of a group of students has been monitored. Each group has been selected to represent the diversity in terms of mathematical prior competence, since some of the students had a limited mathematical background (in the British education systems no A levels in Mathematics or less than C in A levels), some students had a moderate mathematical background (passed A levels in Mathematics with at least C), some students had an advanced mathematical background (passed A levels in Further Mathematics with at least C). For each group and each module (1) pass rate (percentage of students achieving at least 40/100); (2) first rate (percentage of students achieving at least 70/100); (3) average mark of the group $\pm$ standard deviation σ have been recorded at each June exam board. Table 1 displays these data.

Figure 3 displays the trend over the years of the student average with the respective error bars.

The results on the cohorts show that year after year the students achieve better results. Although many factor may have contributed to this outcome, the consistent updates in the material and teaching style indicate the effectiveness of the proposed teaching techniques.

To provide further (qualitative) evidence of the effectiveness of the proposed teaching techniques, some of the comments given by the students in the questionnaire of the module are given in the following. In these comments, the students refer to the approach of explaining topics from different perspectives and to the book [29] as a study manual.

"Theory part always was clearly shown in practice part of module, we had support of book to complete our knowledge, and answer any queries." (Year 1 student 2018)

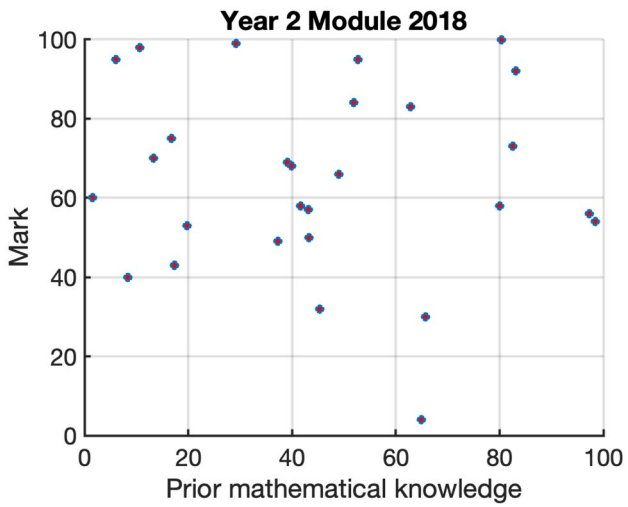

Fig. 5 Scatter plot of the marks against the prior mathematical background for the group of 2018 and Year 2 Module. Most of the students appear to perform well regardless of their prior background

Table 2 Pearson's correlation coefficients r to analyse the correlation between students' performance and prior background

Exam board	2015	2016	2017	2018	2019
Year 1 module					
Pearson's correlation coefficient r	0.4082	0.3849	0.3951	0.3879	–
Year 2 module					
Pearson's correlation coefficient r	–	0.0837	0.0684	-0.0599	0.0589

"Really enjoyed this module and the teaching of the module is clear and well explained and examples are good to help with the learning of the theory" (Year 2 student 2018)

The results of the Year 1 Module are especially interesting since students performed way better from 2016 onward. From the academic year 2015/2016, many topics have been re-written and the algorithmified explanation has been added to the formal mathematical description and the abstraction by example. Furthermore, from the academic year 2015/2016, a lot of attention had been paid to link the topics of the module to realistic scenarios of the job market. It must be remarked that the composition of mathematical background of each group was broadly constant. While in the Academic Year 2014/2015, various students without a high-school mathematical background failed the module (at the first sit), in the following years the lack of prior mathematical knowledge no longer appeared to be a clear disadvantage when the proposed teaching techniques were applied. Figure 4 shows that prior mathematical knowledge may be an advantage to achieve a better performance. However, this is not always true: students with no prior mathematics successfully passed the module and students with a modest mathematical background achieved marks greater than 70. This tendency mitigates over the academic years. In the group of 2018, the performance of students in the Year 1 Module did not appear to be related to their high-school education in mathematics. This fact seems to demonstrate the effectiveness of the proposed teaching.

The results of the Year 2 Module also show an overall improvement of pass rate and average mark, thus indicating that the suggested teaching techniques may have a successful impact on the cohorts. However, since these students previously progressed to from Year 1 to Year 2, the dependency between their pre-university education and results in the module do not appear to be correlated. To exemplify this fact, Fig. 5 shows the scatter plot of the marks of the Year 2 Module in year 2018 against the prior mathematical background. It can be observed that very high marks have been achieved by students in Year 2 regardless of their mathematical knowledge achieved in high school.

To study the correlation between marks in the module and prior mathematical knowledge, a score has been assigned to quantify the mathematical background of each student: 0 has been assigned in case of no A levels, 20, 40, 60, 80, 100 have been assigned students achieving E, D, C, B, A in A levels in Mathematics and Further mathematics. The the score has been normalised to the highest value and expressed within the range [0, 100].

The correlation displayed in Figs. 4 and 5 has been quantitatively studied by calculating the Pearson's coefficient r [33]. Table 2 lists the Pearson's coefficients for the cohorts studied in this paper. We may observe that Pearson's coefficients are thirty to forty times higher for the Year 1 data than the coefficients for Year 2 data. Thus, the quantitative analysis confirms that whilst there is a correlation between prior mathematical knowledge and performance in Year 1, the correlation is negligible in Year 2 (r is close to zero).

Conclusion

This paper investigates the teaching of mathematics in schools of computer science with specific reference to British Universities. After embracing the assumption that teaching mathematics is beneficial to computer science students and to the professional career of computer scientists, this article provides some suggestions to engage the students and teach effectively.

Two specific challenges have been identified and discussed: (1) computer science students typically have a diverse mathematical background; (2) often mathematics is not perceived as a subject that relates directly to computer science jobs. Two teaching techniques are proposed on the basis of the experience of teaching and writing a textbook in these specific circumstances. The first technique, in line with the literature proposes the explanation of each mathematical topic in different ways including the explanation of mathematics as an algorithm (algorithmification of mathematics). The second technique blends research informed teaching with a provision of references how mathematics impacts the every day job of a computer scientists. Specific classroom tested examples are provided to enrich and clarify the proposed techniques. In any case, it is advocated that no compromises are made on the taught content or on the mathematical rigour of the teaching.

A case study based on multiple years teaching indicates that the proposed techniques can be effective to enhance the performance of the students. Furthermore, some observations based on the correlation between prior mathematical knowledge and performance show that in Year 1 prior

mathematical education may have a bias on the student performance which appears to be mitigated by the proposed techniques (students without prior mathematics can perform equally well as their colleagues with prior advanced mathematical studies). The performance of students in Year 2 does not appear to be correlated anymore with their high school history.

This study implicates that teaching of mathematics should be targeted to the specific cohorts/degrees where the teaching occurs. Since mathematics plays a fundamental role in various programmes of applied sciences and engineering, the proposed study indicates that an adaptation of context-related teaching techniques can enhance the engagement of the students and the efficacy of the learning experience. Hence, the proposed teaching techniques can be interpreted as a template expandable to a broader context such as physics and engineering degrees.

Funding This study was funded by the institutions indicated in the list of affiliations.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Al-Shammari Z. Using evidence-based cognitive teaching strategies with effect size in inclusion classrooms in kuwait. Saudi J Spec Educ. 2019;10
2. Al-Shammari Z, Faulkner PE, Forlin C. Theories-based inclusive education practices. Educ Q Rev. 2019;2:408–14.
3. Benton L, Hoyles C, Kalas I, Noss R. Bridging primary programming and mathematics: some findings of design research in england. Digit Exp Math Educ. 2017;3:115–38.
4. Blum C, Chiong R, Clerc M, Jong KD, Michalewicz Z, Neri F, Weise T. Evolutionary optimization. In: R. Chiong, T. Weise, Z. Michalewicz (eds.) Variants of Evolutionary Algorithms for Real-World Applications. Berlin, Heidelberg
5. Boavida F, Praitano A, Lioudakis GV. Topical issue on privacy, data protection, and digital identity. SN Comput. Sci. 2020;1(5):250.
6. Burgum S, Stoakes G. What does research informed teaching look like? HEA booklet 2016; URL https://www.heacademy.ac.uk/blog/what-does-research-informed-teaching-look
7. Byram M. Internationalisation in higher education—an internationalist perspective. On the Horizon. 2018;26(2):148–56.
8. Caraffini F, Neri F, Cheng J, Zhang G, Picinali L, Iacca G, Mininno E. Super-fit multicriteria adaptive differential evolution. In: 2013 IEEE congress on evolutionary computation, 2013; 1678–1685
9. Card SK, Newell A, Moran TP. The Psychology of Human-Computer Interaction. USA: L. Erlbaum Associates Inc.; 1983.
10. Ceragioli F, Spreafico M. Tangible tools in mathematics for engineering students: experimental activity at politecnico di torino. Digit Exp Math Educ. 2020;6:244–56.
11. Ceri S, Pinoli P. Data science for genomic data management: challenges, resources, experiences. SN Comput Sci. 2020;1(1):5:1–7.
12. Church A. On carnap's analysis of statements of assertion and belief. Analysis. 1950;10(5):97–9.
13. Church A. Introduction to Mathematical Logic. Annals of Mathematics Studies. Princeton University Press 1996; URL https://books.google.co.uk/books?id=JDLQOMKbdScC
14. Ertmer PA, Newby TJ. Behaviorism, cognitivism, constructivism: comparing critical features from an instructional design perspective. Performance Improvement Quarterly. 2013;26:43–71.
15. Firdaus F, Kailani I, Bakar M, Bakry B. Developing critical thinking skills of students in mathematics learning. J Educ Learn (EduLearn). 2015;9:226.
16. F.R.S., K.P.: Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**(11), 559–572 (1901)
17. Ghorashi A, Ghorashi M. Theoretical and computational analysis of the falling ladder problem. SN Comput Sci. 2020;1(1):20:1–11.
18. Islam MM, Haque MR, Iqbal H, Hasan MM, Hasan M, Kabir MN. Breast cancer prediction: A comparative study using machine learning techniques. SN Comput Sci. 2020;1(5):290.
19. Jonsson B, Norqvist M, Liljekvist Y, Lithner J. Learning mathematics through algorithmic and creative reasoning. J Math Behav. 2014;36:20–32.
20. Keller JM. Motivational design of instruction. In: C.M. Reigeluth (ed.) Instructional-design theories and models: an overview of their current status, pp. 386–434. Lawrence Erlbaum Associates 1983.
21. Kumar A, Kumaresan S. Use of mathematical software for teaching and learning mathematics. In: Proceedings of the 11th international congress on mathematical education, 2008;373–388
22. Lehman E, Thomson Leighton F, Meyer AR. Mathematics for Computer Science Hardcover. Cambridge: MIT Press; 2017.
23. Luttenberger S, Wimmer S, Paechter M. Spotlight on math anxiety. Psychol Res Behav Manag. 2018;11:311–22.
24. Menabrea L, Babbage C, Lovelace, A., L, A. Sketch of the Analytical Engine invented by Charles Babbage with notes by the translator. Extracted from the 'Scientific Memoirs. R. & J. E. Taylor 1843;
25. Mittelmeier J, Rienties B, Tempelaar D, Whitelock D. Overcoming cross-cultural group work tensions: mixed student perspectives on the role of social relationships. High Educ. 2018;75:149–66.
26. Morley A. Teaching and learning algorithms. Learning Math. 1981;2(2):50–1.
27. Musse SR, Thalmann D. Hierarchical model for real time simulation of virtual human crowds. IEEE Trans Vis Comput Graph. 2001;7(2):152–64.
28. Neri F. Linear algebra computational sciences and engineering. Berlin: Springer; 2016.
29. Neri F. Linear algebra for computational sciences and engineering. 2nd ed. Berlin: Springer; 2019.

30. Neri F, Rostami S. A local search for numerical optimisation based on covariance matrix diagonalisation. In: P.A. Castillo, J.L.J. Laredo, F.F. de Vega (eds.) Applications of evolutionary computation—23rd European conference, EvoApplications 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings, *Lecture Notes in Computer Science*, vol. 12104, pp. 3–19. Springer 2020.
31. Neri F, Zhou Y. Covariance local search for memetic frameworks: A fitness landscape analysis approach. In: 2020 IEEE congress on evolutionary computation (CEC), 2020;1–8
32. von Neumann J. First draft of a report on the edvac. Tech. rep. 1945.
33. NIST/SEMATECH: e-Handbook of Statistical Methods 2003; http://www.itl.nist.gov/div898/handbook/
34. Ochkov VF, Bogomolova EP. Teaching mathematics with mathematical software. J Humanistic Math. 2015;5:265–85.
35. Palmer BL. Teacher passion as a teaching tool. In: Electronic Theses and Dissertations, 2017;3269
36. Press W, Teukolsky S, Vetterling W, Flannery B. Numerical recipes in C. 2nd ed. Cambridge: Cambridge University Press; 1992.
37. Ralston A. The first course in computer science needs a mathematics corequisite. Commun ACM. 1984;27(10):1002–5.
38. Ralston A. Do we need ANY mathematics in computer science curricula? ACM SIGCSE Bull. 2005;37(2):6–9.
39. Ralston A, Shaw M. Curriculum '78—is computer science really that unmathematical? Commun. ACM. 1980;23(2):67–70.
40. Reddy NCS, Madhuravani B, Sneha DP. An approach for efficient and secure data encryption scheme for spatial data. SN Compu Sci. 2020;1(3):117.
41. Revell A, Wainwright E. What makes lectures 'unmissable'? insights into teaching excellence and active learning. J Geography Higher Educ. 2009;33(2):209–23.
42. Robinson K, Aronica L. Finding Your Element: How to Discover Your Talents and Passions and Transform Your Life. penguin books 2014.
43. Rostami S, Neri F, Epitropakis MG. Progressive preference articulation for decision making in multi-objective optimisation problems. Integr Comput Aided Eng. 2017;24(4):315–35.
44. Rønning F. Influence of computer-aided assessment on ways of working with mathematics. Teaching Math Appl. 2017;36(2):94–107.
45. Sedlacek L. Math education: The roots of computer science. https://www.edutopia.org/blog/math-education-roots-computer-science-lincoln-sedlacek 2016.
46. Sipser M. Introduction to the Theory of Computation. 1st ed. : International Thomson Publishing; 1996.
47. Su H, Ricci FA, Mnatsakanian M. Mathematical teaching strategies: pathways to critical thinking and metacognition. J Res Educ Sci (IJRES). 2016;1:190–200.
48. Tobias S, Weissbrod C. Anxiety and mathematics: an update. Harvard Educ Rev. 1980;50(1):63–70.
49. Turing A. On computable numbers, with an application to the entscheidungsproblem. Proc Lond Math Soc. 1936;42(1):230–65. https://doi.org/10.2307/2268810.
50. Vince J. Foundation mathematics for computer science: a visual approach. Berlin: Springer; 2015.