The Mathematician Educator
2024, Vol 5, No. 1, 3-15

I am Afraid of Technology: Some Tips to Overcome the Fear of Coding

TAY Eng Guan
National Institute of Education
Nanyang Technological University, Singapore

Fear of technology to some is understandable and to others is absurd. The way forward is to acknowledge the unease and to learn from others how to effectively use the technology. Coding is one such aspect of technology. Computational thinking remains theoretical until actualized as a computer programme. In this article, we will ease readers into coding by providing a number of relevant secondary and junior college mathematics examples. We will work on simple VBA coding in the familiar Microsoft Excel environment.

Keywords: Fourth Industrial Revolution; coding; computational thinking; secondary mathematics

The Fourth Industrial Revolution

The Fourth Industrial Revolution is a term introduced by a group of German scientists (Schwab, 2015) to describe a particular kind of technological advancement in the twentieth century. It distinguishes itself from previous numbered[1] "Industrial Revolutions" by: the use of large-scale Machine-to-Machine (M2M) and Internet of Things (IoT) deployments to provide increased connected automation (contrasted with the localized use of computers in the Third Industrial Revolution); the use of smart machines that can make decisions without human intervention (Artificial Intelligence (AI) and Machine Learning contrasted with the deterministic programming of the Third Industrial Revolution); and the use of cyber-physical systems to collect vast amounts of data for analysis and improvement of work (contrasted with pre-programmed functions in the computers of the Third Industrial Revolution).

Implications for education

The rise of machines that depend on Artificial Intelligence does not mean that humans do not think any more. Indeed, the machines still need to be 'taught' since AI and Machine Learning programmes are heavily dependent on statistical methods and sophisticated programming. The cyber-physical systems used to collect useable data for work improvement often need to be customized to the workplace. This implies that many digitally literate workers are needed to programme the sensors and machines for the many factories, production and logistic centres. Customisation means that the programmes need to be continually updated and adapted in-situ unlike computer products of the Third Industrial Revolution, such as Microsoft Office, which are developed centrally and used as given at the workplace.

[1] First Industrial Revolution (1760-1820) powered by steam; Second Industrial Revolution (1871-1914) powered by electricity, railroads and telegraphs; Third Industrial Revolution (late 20th century) powered by computers; Fourth Industrial Revolution (21st century) powered by data.

It is a fact that communicating between end-users and software developers is often difficult. They must find a common language to communicate in. Developing enough shared vocabulary to communicate can often take a while. The current common workplace situation is that computer programmers and system analysts (the vendors) need to spend time understanding the industry that require their expertise. Looking and educating ahead, communication will be much better if the end-users (with good domain knowledge) have learnt how to code while in school or the university.

Responses from educators

Nations that were early in recognizing the advent and implications of the Fourth Industrial Revolution responded by trying to infuse coding into their school curriculums. In September 2014, schools in England and Wales replaced the subject of Information and Communications Technology (ICT) with Computing, which includes the teaching of coding in the classroom (Cuthbertson & Murakoshi, 2015). The European Union (EU) and Canada implemented coding programmes as enrichment in schools. Singapore's approach is quite similar to the EU and Canada, with the Minister of Education explaining it as follows:

> "We recognise the importance of coding in our young people, especially in this digital age. But programming languages can change and become outdated over time. So it is important to focus on the underlying skills such as computational thinking, problem solving, logical reasoning and data handling. These underlying skills are developed through the learning of subjects, particularly mathematics and science. At the same time, we have worked with IMDA to expose all students to basic coding and technology through the 'Code For Fun' programme." (Ong, 2020)

The Singapore Prime Minister, a savvy coder himself, offered the important perspective that 'coding is creating technology' when he said:

> "We need the right organisations, the right skills, the right mindsets to be a Smart Nation. We have to start with our education system. We are equipping students with up-to-date knowledge and skills to use the technology. But *schools must also teach students how to create the technology of the future; teach them to code* [emphasis added], to prototype and build things, to fail fast and learn quickly, to use the latest gadgets, the latest tools and be up with the latest technology." (Lee, 2014)

However, Cuthbertson and Murakoshi (2015) reported that most teachers were not happy teaching computing — only 40% of the 1,009 teachers polled felt confident in teaching coding. They also reported that this low percentage reflected criticism from some teachers about inadequate teacher preparation. My personal sensing from interactions with Singapore teachers is the same – that most are apprehensive about including coding in their mathematics lessons.

In this article, I would like to insert my personal experiences and expertise as empathetic solutions for mathematics teachers to respond to the urgent and important challenge for the new economy. Fifteen years ago, I was in a car with three other graph theorists – none of us had a mobile phone. Fear of technology to some is understandable and to others is absurd. The way forward is to acknowledge the unease and to learn from others how to effectively use technology to make one's life better. Indeed, I started using a mobile phone regularly only when email could be accessed as it made my work much easier. Coding is one such aspect of technology. As a mathematics researcher, I found it useful for exploring conjectures. As a

mathematics educator, I found it useful to develop customized activities for my students, for example, Monty Hall and lottery simulations (Tay, 2004).

The Singapore Ministry of Education stated that "it is important to focus on the underlying skills such as computational thinking" (Ong, 2020). According to Stephens and Kadijevich (2020), Computational Thinking (CT) consists of the four components, decomposition, abstraction, algorithmisation, and automation. Further, they stated that Algorithmic Thinking (AT) consists of decomposition, abstraction, and algorithmisation. AT is one form of mathematical reasoning, required whenever one has to comprehend, test, improve, or design an algorithm. Mathematicians have done this long before the electronic computer. Putting the two descriptions together, we have that CT is thus "AT with a computer". Kadijevich et al. (2023) argued that one rationale for including CT in compulsory education is to enable students to solve problems as an information-processing agent to meet the increasing demands of the Fourth Industrial Revolution. They cite the description of CT by the Royal Society (Royal Society, 2012) "as enabling persons to recognise aspects of computations in various problem situations, and to deal with those aspects, by applying tools and techniques from computer science" (p. 377). It seems that CT as described above requires experience with coding for otherwise, it would be difficult to accurately recognise aspects of computation (a CEO of a bank may have no idea how many extra servers his bank may need when they implement a new online product), and it would be incomplete to compare algorithms, such as the relative speeds of finding a Highest Common Factor using prime factorization versus the Euclidean algorithm.

In the rest of this article, I will ease teachers into coding by giving a number of relevant secondary and junior college mathematics examples. We will work on simple VBA coding in the familiar Microsoft Excel environment.

Pedagogical Approach: Full code before step by step – Complex before simple

The pedagogical approach we propose can be said to be "full code before simple steps, complex before simple" in contrast to the opposite "usual" approach of learning the syntax step-by-step. This contrast is aptly pictured humorously in a cartoon from the book Arithmetic for Parents: A Book for Grownups about Children's Mathematics by Ron Aharoni (2015) where a mystic sitting on a bed of nails invites a novice to sit on a plank with only one nail.

Students should be given the whole code, such as the VBA code for a program, and learn to explore and work on it using the PRIMM approach (Sentance et al., 2019). PRIMM stands for Predict, Run, Investigate, Modify, and Make. The whole code for a programme is given and the student is to *predict* what some parts would do. The student then *runs* the programme and *investigates* to see if the prediction is true. The student can then *modify* certain parts of the code to build on the investigation and to modify the output. The end product is for the student to be able to *make* a new product based on the code that is now fairly understood.

Coding within the Excel spreadsheet

I recommend the use of Microsoft Excel in the classroom because it is an almost ubiquitous programme in computers. Students and teachers can easily access it without having to download additional software. As a preliminary, I explain in this section some key features in Excel.

A formula in Excel always starts with an equal sign (=), which can be followed by numbers, mathematics operators (+ or - sign for addition or subtraction; * or / sign for multiplication or division), and built-in Excel functions.

Here are a few examples:

○ =2*A1+A2-5	Multiplies 2 by the value in A1, adds the value in A2 and then subtracts 5.
○ =SUM(A1:B10)	Uses the SUM function to return sum of the values in A1 through A10 and B1 through B10.
○ =COUNTIF(range, criteria)	Counts the number of cells within a range that meet the given condition
○ =IF(logical test, value if true, value if false)	Returns one value if a condition you specify evaluates to TRUE and another value if it evaluates to FALSE.

The last function is a very important function for learning coding in Excel for Mathematics. For example, if you want to compare the values of expressions in two different cells A1 and B1, you may write in C1 the following: =IF(A1>B1,“Yes!”,“Try again.”). This is illustrated for four rows in Figure 1 below.

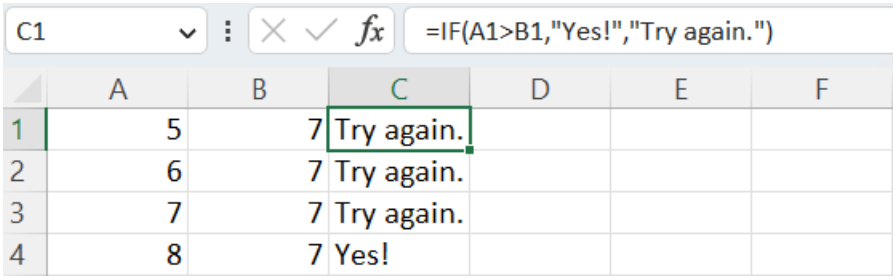


Figure 1. Examples of the IF logical test

Another important feature of Excel is akin to looping. Suppose we want to perform the following pseudocode:

```
For i = 1 to 5
        Cells(i, 2).Value = 3*i + 2
Next
```

We begin by obtaining a column of consecutive integers from 1 to 5 as follows. Key in 1 in A1 and 2 in A2. Highlight A1 and A2. Move the cursor, which is normally a white cross, onto the bottom right of A2, where it will change into a black cross. Click and hold the black cross and drag down to A5. Now key in the formula “=3*A1 + 2” into B1 and press Enter. Highlight B1. Move the cursor onto the bottom right of B1, where it will change into a black cross. Click and hold the black cross and drag down to B5. See Figure 2 for the end result. Note that the formula “=3*A5 + 2” appears in the Function Window but the Cell B5 shows the value 17.

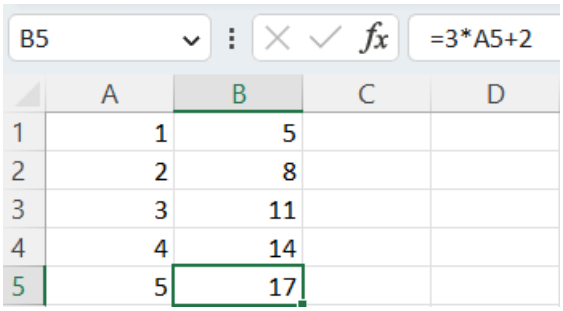


Figure 2. Example of a simple "loop"

Students should be taught to see coding in Excel as a tool similar to calculators. There should be no assessment on the use of coding in Excel just as there is no assessment for the use of calculators. Students are to use coding to explore and solve mathematics problems. To this end, no more than half a lesson is to be used to introduce the above features to students. The key thing is to get them to quickly use the programme to work on some activities relevant to the learning of mathematics.

Some Activities for the Classroom

In this section, I describe some activities that can be used to learn mathematics and improve fluency in coding as a by-product.

Pattern Match

The objective of this activity is for students to key in the correct general formula for a number pattern.

A 10-number pattern is given in Column A. (See Figure 3.) The students are to construct consecutive numbers from 1 to 10 in Column B. Then they are required to key into Column C a general formula for the number pattern in Column A. Feedback has been provided by the teacher in Column D using the "=IF" formula. Students can try as often as they like until they get the formula correct.

Figure 3.

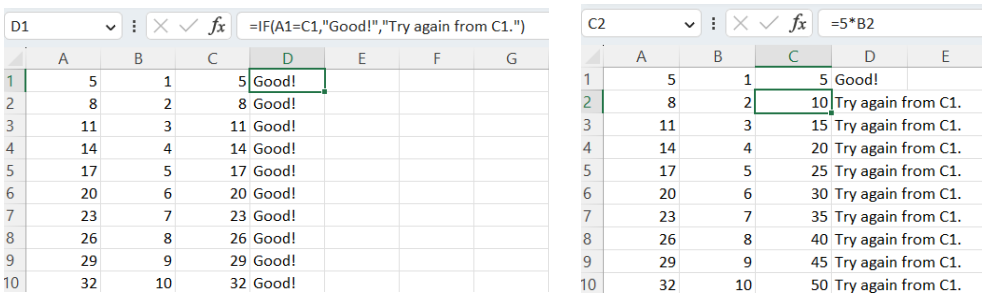


Figure 3. Pattern Match

Highest Common Factor

The highest common factor (HCF) of numbers x and y is defined to be the largest positive integer d such that d is both a factor of x and of y. In secondary school, the usual method of calculating the HCF is by obtaining the prime factorisations of the two numbers and extracting

the lower power of each common prime. Let students calculate the HCF for increasingly larger pairs. When students realise the calculation becomes increasingly tedious, explain that a more efficient method exists, i.e., the Euclidean algorithm explained as follows.

Euclidean Algorithm

Given $x > y$, find HCF(x, y).
Set $x_1 = x$ and $y_1 = y$.
For $n \geq 1$, while $r_n > 0$,

$$x_n = y_n q_n + r_n$$
$$x_{n+1} = y_n$$
$$y_{n+1} = r_n$$

Output HCF(x, y) = r_{n-1}.

Here is a numerical example: Find HCF(718740, 495000).

$$\begin{aligned}
718740 &= 1 \times 495000 + 223740 \\
495000 &= 2 \times 223740 + 47520 \\
223740 &= 4 \times 47520 + 33660 \\
47520 &= 1 \times 33660 + 13860 \\
33660 &= 2 \times 13860 + 5940 \\
13860 &= 2 \times 5940 + 1980 \\
5940 &= 3 \times 1980 + 0.
\end{aligned}$$

Thus, HCF(718740, 495000) = 1980.

From the first equation, we observe that HCF(718740, 495000) divides 718740 and 495000 and so must also divide the first remainder 223740. Thus, in the second equation, since HCF(718740, 495000) divides 495000 and 223740, it must divide the second remainder 47520. Continuing on to the other equations, we see that HCF(718740, 495000) divides all the remainders, including the final non-zero remainder 1980. On the other hand, starting from the last equation, we see that the final non-zero remainder 1980 divides 5940. Thus, using the second last equation, 1980 also divides 13860. Working backwards, 1980 also divides 495000 and 718740. Hence, since the final non-zero remainder 1980 divides HCF(718740, 495000), and HCF(718740, 495000) divides the final non-zero remainder 1980, we have HCF(718740, 495000) = 1980. We use a generalised approach to show that HCF(x, y) = final non-zero r_n.

We now see how to get the students to code this programme into Excel. After explaining the Euclidean algorithm and giving a numerical example, we ask the students to state the variables of interest. They should be x_n, y_n and r_n. Thus, we write x, y and r in A1, B1 and C1. The initial values x_1 and y_1 are keyed into A2 and B2. The question for students now is how to obtain r_1 to be keyed into C2. There is a function in Excel which gives the remainder when x is divided by y: = MOD(x, y). (See Figure 4.)

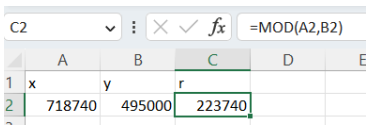
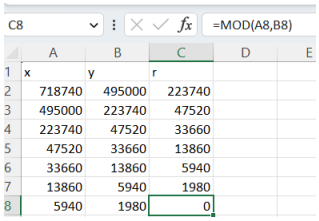


Figure 4. Example of MOD function

This solution is for teachers. Since $x_2 = y_1$ and $y_2 = r_1$, we key "=B2" into A3 and "=C2" into B3. Highlight C2, move the cursor onto the bottom right of C2, where it will change into a black cross. Click and hold the black cross and drag down to C3. This makes the formula for C2 adjust itself for C3. We now highlight A2, B2, C2, move the cursor onto the bottom right of C2, where it will change into a black cross. Click and hold the black cross and drag down until we see 0 in Column C. The HCF will be the last non-zero entry in Column C. (See Figure 5.)



	A	B	C	D	E
1	x	y	r		
2	718740	495000	223740		
3	495000	223740	47520		
4	223740	47520	33660		
5	47520	33660	13860		
6	33660	13860	5940		
7	13860	5940	1980		
8	5940	1980	0		

Figure 5. Example of Euclidean Algorithm

Power Series

The power series, including the Maclaurin Series, expansion of a function is taught in junior college. Some famous expansions are:

$$\frac{1}{1+x} = 1 - x + x^2 \ldots + (-1)^n x^n + \cdots, \text{ for } -1 < x < 1.$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} + \cdots + (-1)^{n-1}\frac{x^n}{n} + \cdots, -1 < x < 1.$$

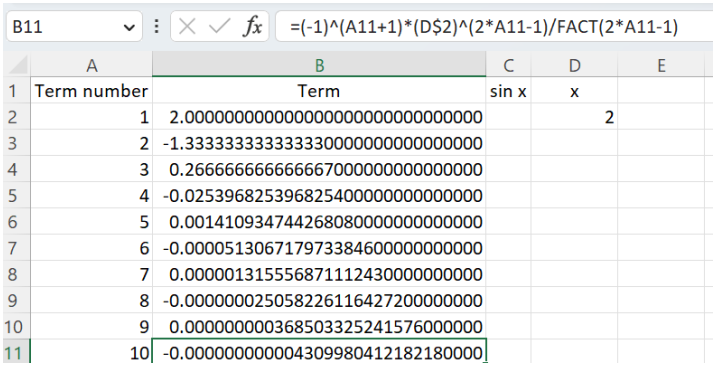
$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots, \text{ for all } x.$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + (-1)^{n+1}\frac{x^{2n-1}}{(2n-1)!} + \cdots, \text{ for all } x.$$

Students seldom ask and most students never understand why power series expansions are developed. The teacher may ask the students if they know how a calculator can compute the value of sin 2 (in radians). Cheekily, she may ask if there is a little elf inside the calculator who when the right buttons are punched would rise from sleep, draw a right-angled triangle with the required angle 2, measure the opposite and the hypotenuse, and finally deliver the ratio as the answer! The students are now ready to understand that the initial definition of sine as the ratio of the opposite over the hypotenuse is not computable. Hence, a computable form of the sine function, which includes the power series expansion, is needed to calculate sine values.

Here is how we can make the series expansions come alive. After explaining the purpose of power series, we ask the students to state the variables of interest for expanding sin *x*. They are the term number, the terms of sin *x*, and the approximate value of sin *x* computed as the sum of the terms. Thus, we write term number, term and sin *x* in A1, B1 and C1. The text *x* and the value of *x* is keyed into D1 and D2. The question for students now is how to obtain the term of sin *x* to be keyed into C2. (There is a function in Excel for *x*!: = FACT(x).)

This solution is for teachers. Obtain a list of consecutive numbers in Column A in the usual way. Since the general term is $(-1)^{n+1}\frac{x^{2n-1}}{(2n-1)!}$, we key "=(-1)^(A2+1)*(D$2)^(2*A2-1)/FACT(2*A2-1)" into B2. Note that we put a $ sign between D and 2 to ensure that when we drag the formula later, the formula will continue to refer to D2 and not move on to D3, D4, etc. Maximise the number of decimal places (up to 30) for the cells. Now drag from B2 down Column B to obtain the values of each term. (See Figure 6.)

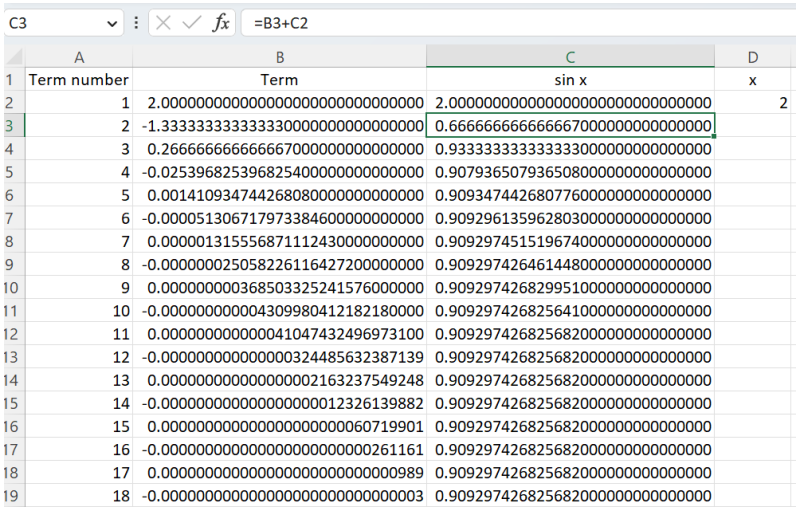


B11 =(-1)^(A11+1)*(D$2)^(2*A11-1)/FACT(2*A11-1)

	A	B	C	D	E
1	Term number	Term	sin x	x	
2	1	2.000000000000000000000000000000		2	
3	2	-1.333333333333330000000000000000			
4	3	0.266666666666667000000000000000			
5	4	-0.025396825396825400000000000000			
6	5	0.001410934744268080000000000000			
7	6	-0.000051306717973384600000000000			
8	7	0.000001315556871112430000000000			
9	8	-0.000000025058226116427200000000			
10	9	0.000000000368503325241576000000			
11	10	-0.000000000004309980412182180000			

Figure 6. Computing each term of the power series expansion for sin x

To obtain the approximation to sin x, we key "=B2" into B3 to obtain the first approximation to sin x as the first term. Then we obtain the second approximation by adding the second term to the first approximation. We thus key "=B3+C2" into C3. Finally, we drag from C3 down Column C to get better and better approximations to sin x. (See Figure 7.)



C3 =B3+C2

	A	B	C	D
1	Term number	Term	sin x	x
2	1	2.000000000000000000000000000000	2.000000000000000000000000000000	2
3	2	-1.333333333333330000000000000000	0.666666666666667000000000000000	
4	3	0.266666666666667000000000000000	0.933333333333333000000000000000	
5	4	-0.025396825396825400000000000000	0.907936507936508000000000000000	
6	5	0.001410934744268080000000000000	0.909347442680776000000000000000	
7	6	-0.000051306717973384600000000000	0.909296135962803000000000000000	
8	7	0.000001315556871112430000000000	0.909297451519674000000000000000	
9	8	-0.000000025058226116427200000000	0.909297426461448000000000000000	
10	9	0.000000000368503325241576000000	0.909297426829951000000000000000	
11	10	-0.000000000004309980412182180000	0.909297426825641000000000000000	
12	11	0.000000000000041047432496973100	0.909297426825682000000000000000	
13	12	-0.000000000000000324485632387139	0.909297426825682000000000000000	
14	13	0.000000000000000002163237549248	0.909297426825682000000000000000	
15	14	-0.000000000000000000012326139882	0.909297426825682000000000000000	
16	15	0.000000000000000000000060719901	0.909297426825682000000000000000	
17	16	-0.000000000000000000000000261161	0.909297426825682000000000000000	
18	17	0.000000000000000000000000000989	0.909297426825682000000000000000	
19	18	-0.000000000000000000000000000003	0.909297426825682000000000000000	

Figure 7. Computing the power series expansion for sin x

VBA coding in the Excel programme

As teachers and students develop and learn to enjoy using the Excel spreadsheet, they may want to venture into 'real' coding. Excel provides this avenue since it incorporates the Visual Basic for Applications (VBA) programming language. (I am told that it will soon incorporate Python as well.)

VBA can be accessed by clicking the Developer tab. (If you cannot see this tab in your programme, click in this order to make it visible: File > Options > Customize ribbon > Main Tabs > *select* Developer.

Following the PRIMM principle, the syntax of VBA can be learnt by recording a macro as follows:

- Developer > Record macro
- *Do anything you want*
- Click square button (STOP) at bottom left corner

- Give a name to your macro in the popup box
- Developer > Visual Basic: to see your code

It is very important in coding to know the syntax for conditionals and loops. The following is the syntax for VBA:

Conditional:	If < > Then
	…
	End If

Loop:	For < > … Next
Conditional and Loop:	While < > … Wend

Here is a code for HCF which you may use as a template.

```
Sub HCF()
Dim x As Integer
Dim y As Integer
Dim k As Integer

x = InputBox("What is the larger number?")
y = InputBox("What is the smaller number?")
Cells(1, 1).Value = x
Cells(1, 2).Value = y
Cells(1, 3).Value = x Mod y

For i = 2 To y
   Cells(i, 1).Value = Cells(i - 1, 2).Value
   Cells(i, 2).Value = Cells(i - 1, 3).Value
   Cells(i, 3).Value = Cells(i, 1).Value Mod Cells(i, 2).Value
   If Cells(i, 3) = 0 Then
         k = i
         Exit For
   End If
Next

Cells(1, 13).Value = "HCF(" & Cells(1, 1).Value & ", " & Cells(1, 2).Value & ") is "
& Cells(k - 1, 3).Value & "."

End Sub
```

The following are screenshots of some programmes that I have coded in Excel VBA. They include a Sudoku solver in Figure 8, the Floyd Warshall algorithm for finding the shortest distances in Figure 9, and a fractal in Figure 10 (see Tay, 2021). (I have also provided the VBA code for the fractal in the Appendix.)

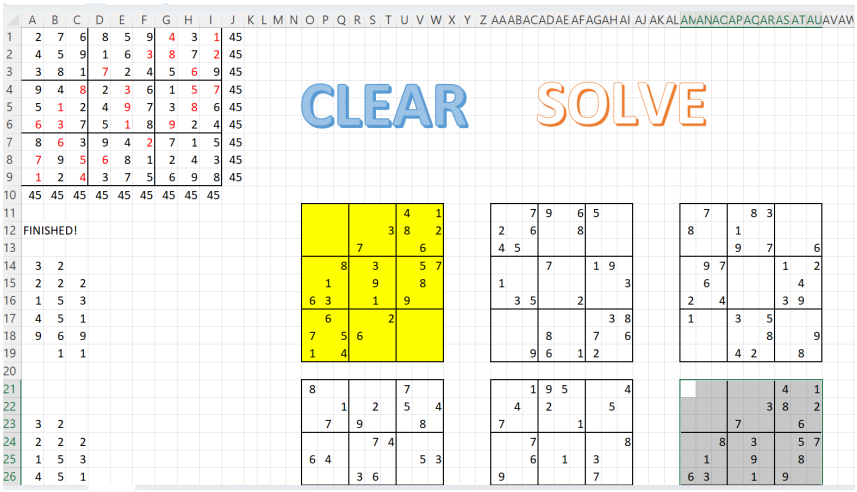


Figure 8. Screenshot of a Sudoku solver

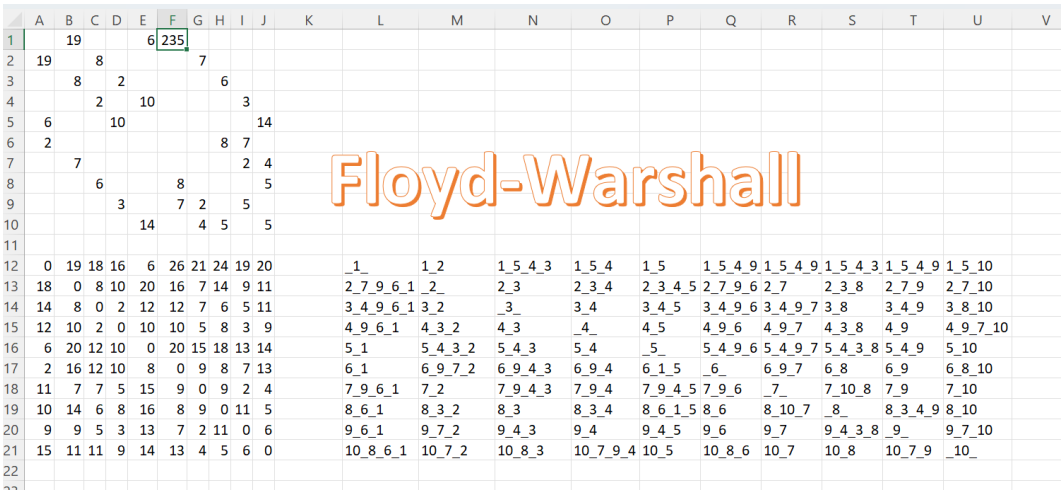


Figure 9. Screenshot of the Floyd-Warshall algorithm

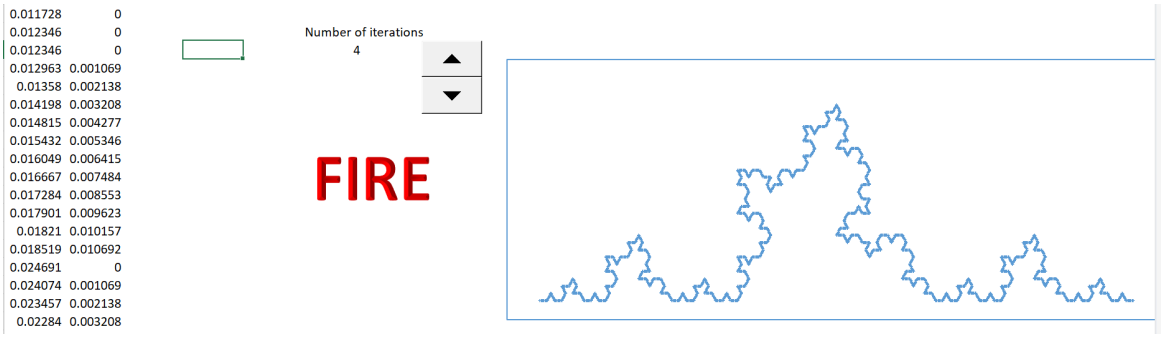


Figure 10. Screenshot of fractal

Conclusion

The Fourth Industrial Revolution provides a strong reason for the teaching and learning of coding in the schools. Computational thinking is the result of algorithmic thinking and actual coding experience. The article gives some examples of how coding can be infused into the

teaching of school mathematics. In the *Pattern Match* activity, students learn how to guess a formula and obtain quick feedback on whether it is correct. This is an important approach in coding where mistakes and bugs are to be expected and overcome. The *Euclidean Algorithm* activity teaches the students computational complexity. The usual school approach to find HCF(x, y), $x > y$, using prime factorization takes much longer (of the order of $\sqrt{x}$) than the Euclidean Algorithm (of the order of $\log(y)$). The speed of an algorithm is an important aspect of Computational Thinking because codes need to be as fast as possible, and a slow programme can take up time and maybe even burn up the computer! The *Power Series* activity shows students the 'computability' in computing. It also allows them the opportunity to code mathematical formulas that they have learnt to produce numerical solutions. It is important for students to know that they can code formulas to create working programmes — in this case, a sine calculator. In the future, they may need to code more complex statistical formulas to create AI machines.

I strongly recommend the Excel programme because it is readily available, and it is also often used in the workplace. Thus, the coding skills can be easily picked up again if necessary. Also, the spreadsheet allows some form of pseudo-coding which may be the easiest way to get into Computational Thinking. All that has been said was said to overcome the fear of coding, which stems from the fear of installing and using a new computer programme (overcome by using the familiar Excel), the fear of learning a language (overcome by providing full codes as examples), and the fear of failure over small bugs (by learning from someone else's success and from seeing one's code actually work).

References

Aharoni, R. (2015) *Arithmetic for Parents: A Book for Grownups about Children's Mathematics*. World Scientific.

Cuthbertson, A., & Murakoshi, S. (2015, February 27). Coding in the classroom: How has the UK curriculum overhaul fared six months on? *International Business Times* https://www.ibtimes.co.uk/coding-classroom-how-has-uk-curriculum-overhaul-fared-six-months-1489714

Lee, H. L. (2014) Speech at Smart Nation launch 24 November 2014. Retrieved 9 May 2024: https://www.smartnation.gov.sg/media-hub/speeches/smart-nation-launch/

Kadijevich, D. M., Stephens, M., & Rafiepour, A. (2023). Emergence of Computational/Algorithmic Thinking and its impact on the mathematics curriculum. In Shimizu, Y. & Vithal, R. (Eds.), *Mathematics curriculum reform around the world: The 24th ICMI study* (pp. 375–388). Springer.

Ong, Y. K. (2020). Coding as part of the mainstream curriculum. Retrieved 9 May 2024: https://www.moe.gov.sg/news/parliamentary-replies/20200904-coding-as-part-of-the-mainstream-curriculum

Royal Society (2012). Shut down or restart? The way forward for computing in UK schools. *The Royal Society.* https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

Schwab, K. (2015, December 12). The Fourth Industrial Revolution. *Foreign Affairs.* https://www.foreignaffairs.com/world/fourth-industrial-revolution

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, 29(2–3),136–176. https://doi.org/10.1080/08993408.2019.1608781

Stephens, M., & Kadijevich, D. M. (2020). Computational/Algorithmic Thinking. In Lerman, S. (Ed.), *Encyclopedia of Mathematics Education* (pp. 117–123). Springer.

Tay, E. G. (2004). The psychology of a 'near-miss' in the 4-Digit lottery: A spreadsheet simulation. In *9th Asian Technology Conference in Mathematics ATCM2004: Technology in Mathematics: Engaging learners, empowering teachers, enabling researchers* (pp. 387–392). ATCM Inc.

Tay, E. G. (2021). Tangrams: On attention and error. *The Mathematician Educator*, 2(2), 67–75. https://ame.org.sg/2021/12/29/tme2021-vol-2-no-2-pp-67-75/

Author:
TAY, Eng Guan (engguan.tay@nie.edu.sg),
National Institute of Education, Nanyang Technological University, Singapore,
1 Nanyang Walk, Singapore 637616

Appendix: VBA code for Fire Fractal

```
Sub Fire()

Dim Seq As Variant
Dim Trans As Variant

ReDim Seq(1 To 12, 1 To 2) As Double

Seq(1, 1) = 0 : Seq(2, 1) = 0.1 : Seq(3, 1) = 0.2 : Seq(4, 1) = 0.3 : Seq(5, 1) = 0.4
Seq(6, 1) = 0.5 : Seq(7, 1) = 0.6 : Seq(8, 1) = 0.7 : Seq(9, 1) = 0.8 : Seq(10, 1) = 0.9
Seq(11, 1) = 0.95 : Seq(12, 1) = 1 : Seq(1, 2) = 0 : Seq(2, 2) = 0 : Seq(3, 2) = 0
Seq(4, 2) = 0 : Seq(5, 2) = 0 : Seq(6, 2) = 0 : Seq(7, 2) = 0 : Seq(8, 2) = 0
Seq(9, 2) = 0 : Seq(10, 2) = 0 : Seq(11, 2) = 0 : Seq(12, 2) = 0

n = 12
For k = 1 To Cells(13, 6).Value
   ReDim Trans(1 To 4 * n, 1 To 2) As Double
   For i = 1 To n
      Trans(i, 1) = Seq(i, 1) / 3
      Trans(i + n, 1) = (Seq(i, 1) - Sqr(3) * Seq(i, 2) + 2) / 6
      Trans(i + 2 * n, 1) = (-Seq(i, 1) - Sqr(3) * Seq(i, 2) + 4) / 6
      Trans(i + 3 * n, 1) = (Seq(i, 1) + 2) / 3
      Trans(i, 2) = Seq(i, 2) / 3
      Trans(i + n, 2) = (Sqr(3) * Seq(i, 1) + Seq(i, 2)) / 6
      Trans(i + 2 * n, 2) = (Sqr(3) * Seq(i, 1) - Seq(i, 2)) / 6
      Trans(i + 3 * n, 2) = Seq(i, 2) / 3
   Next
   n = 4 * n
   ReDim Seq(1 To n, 1 To 2) As Double
   For j = 1 To n
      Seq(j, 1) = Trans(j, 1)
      Seq(j, 2) = Trans(j, 2)
   Next
Next

For i = 1 To n
   Cells(i, 1).Value = Seq(i, 1)
   Cells(i, 2).Value = Seq(i, 2)
Next

End Sub
```