ACTA SCIENTIAE

ISSN: 2178-7727
DOI: 10.64856/acta.scientiae.8460

Instrumental orchestration in the teaching of recurrent sequences: A computational approach for basic education

Jailson França dos Santos[a]
Francisco Régis Vieira Alves[b]
Ana Paula Aires[c]

[a] Universidade Federal do Oeste da Bahia, Barreiras, Bahia, Brasil.
[b] Instituto Federal de Educação, Ciência e Tecnologia do Estado do Ceará, Fortaleza, Ceará, Brasil.
[c] Universidade de Trás-os-Montes e Alto Douro, Vila Real, Portugal; CIDTFF – Centro de Investigação Didática e Tecnologia na Formação de Formadores, Universidade de Aveiro, Aveiro, Portugal.

ABSTRACT

Background: The integration of mathematics and computer science has become increasingly established in basic education, expanding the possibilities for teaching mathematical content through programming languages. However, there is still a lack of didactic proposals that connect recurrent sequences from a computational perspective grounded in mathematics education frameworks. **Objectives:** To investigate how to organise a didactic proposal that integrates recurrent sequences and programming in basic education, grounded in the theory of didactical situations (TDS) and the notion of instrumental orchestration. **Design:** This is a theoretical study grounded in instrumental theory, with particular emphasis on the organisation of teaching and the teacher's role in mediating mathematical knowledge through programming. **Setting and participants:** As this is a theoretical investigation, the study is based on an analysis of academic literature in mathematics education, TDS, and instrumental theory, including studies on the use of programming in teaching mathematics. **Data collection and analysis:** Data were collected through a conceptual review and analysis of theoretical frameworks, using an analytical and interpretative approach. **Results:** The study proposes an instrumental orchestration that integrates algebraic, computational, and visual representations for the study of recurrent sequences. **Conclusions:** The proposal may contribute to the development of algebraic and algorithmic thinking, provide theoretical support for designing didactic tasks, and expand pedagogical possibilities in basic education.

Keywords: Programming language; Recurrent sequences; Instrumental orchestration; Didactical situation.

Corresponding author: Jailson França dos Santos. Email: jailson.santos@ufob.edu.br

Orquestração instrumental no ensino de sequências recorrentes: Uma abordagem computacional para o ensino básico

RESUMO

Contexto: A integração entre matemática e computação tem se consolidado na educação básica, ampliando as possibilidades de ensino de conteúdos matemáticos por meio de linguagens de programação. Entretanto, ainda são escassas as propostas didáticas que articulem sequências recorrentes sob uma perspectiva computacional fundamentada em referenciais da educação matemática. **Objetivos**: Investigar como organizar uma proposta didática que integre sequências recorrentes e programação no ensino básico, a partir de uma abordagem teórica ancorada na teoria das situações didáticas (TSD) e na noção de orquestração instrumental. **Design**: Trata-se de um estudo de natureza teórica, fundamentado na teoria instrumental, com ênfase na organização do ensino e no papel do professor na mediação do conhecimento matemático por meio da programação. **Ambiente e participantes**: Por se tratar de investigação teórica, o trabalho baseia-se na análise de produções acadêmicas da educação matemática, da TSD e da teoria Ii, contemplando estudos que abordam programação aplicada ao ensino de matemática. **Coleta e análise de dados**: A coleta de dados ocorreu por meio de revisão e análise conceitual de referenciais teóricos, conduzida de forma analítica e interpretativa. **Resultados**: O estudo sistematiza uma proposta de orquestração instrumental que integra registros algébricos, computacionais e visuais no tratamento de sequências recorrentes. **Conclusões**: Conclui-se que a proposta pode contribuir para o desenvolvimento do pensamento algébrico e algorítmico, oferecendo subsídios teóricos para a elaboração de tarefas didáticas e ampliando as possibilidades pedagógicas no ensino básico.

Palavras-chave: Linguagem de programação; Sequências recorrentes; Orquestração instrumental; Situação didática.

INTRODUCTION

The integration of mathematics and computing has been consolidated in basic education (Goldenberg et al., 2021; Humble, 2022; Yusuf & Noor, 2024), driven by advances in digital technologies and the need to develop algebraic, logical, and algorithmic thinking skills (Kieran, 2017). In this scenario, the insertion of the programming language in mathematics teaching (Santos et al., 2026) has expanded the possibilities of exploring concepts approached in an abstract way, favouring meanings through experimentation, visualisation, and problem solving. Among these concepts, the study of recursive sequences (Vieira Alves, 2022; Alves et al., 2024), which articulates algebraic relationships, growth patterns, and iterative processes, constitutes a broad field for bringing mathematical and computational languages closer together.

Parallel to the educational interest, scientific research on recursive sequences is growing, extending beyond classic cases such as the Fibonacci and Lucas sequences. Research has investigated extended families of recursiveness, such as Tribonacci, Pentanacci, Padovan, Jacobsthal, and Narayana, as well as their generalisations, with an emphasis on algebraic properties, asymptotic growth, and generating functions (Djordjević & Milovanović, 2014; Koshy, 2019). In this context, studies such as Soykan's (2025), Dişkaya et al.'s (2023), Hori and Kida's (2025), Berend and Kumar's (2025), and Alecci et al.'s (2025) consolidate this field by expanding classes of recursiveness and deepening the analysis of their structures and interrelationships.

This movement reveals the interest in understanding recursive structures as mathematical objects and reinforces the importance of exploring sequences through computational approaches. It is possible to observe in the literature that different studies with computational approaches show significant contributions to mathematical learning: Binotto et al. (2023) point out that programming activities can favour mathematical creativity, reflection, and student engagement; Rais and Xuezhi (2024) identified positive effects on student motivation and performance; Borg and Fahlgren (2025) point out that these benefits depend on didactic mediation. Thus, the need for proposals that integrate programming, mathematical content, and teaching performance is evident.

Despite the growing interest in programming in mathematics teaching (Munthe & Naalsund, 2024; Brandsaeter & Berge, 2025), many initiatives still focus on instrumental aspects that are not always linked to theoretical frameworks of mathematics education. In the literature consulted, few studies address the teaching of recursive sequences from a computational approach that centrally considers the teacher's role in the organisation of instruments, tasks, and interactions in the classroom. This situation suggests a possible gap, especially when considering the integration of algebraic, computational, and visual records in the teaching of this content.

Research has highlighted that effective pedagogical practices using digital resources require an intentional organisation of tasks, tools, and interactions in the classroom, supported by theoretical frameworks that guide teaching practice and account for diverse perspectives on learning (Clark-Wilson et al., 2014). The research question of this article emerges in the scenario above: *How to organise a didactic proposal that integrates the teaching of recurrent sequences with a computational approach in the context of basic education?* This question guides the development of this research and

delimits its focus on the field of mathematics education, with an emphasis on pedagogical mediation and programming as a digital teaching instrument.

In this context, this work presents a proposal for instrumental orchestration for teaching recurrent sequences, supported by a computational approach, in which the mediation of mathematical content is understood in light of the theory of didactic situations (TDS) (Brousseau, 1986). Thus, we seek to show that programming can serve not only as a technical resource but also as a didactic instrument that reorganises interactions among students, mathematical concepts, and representations.

Methodologically, the study is characterised as theoretical research, anchored in instrumental theory (Rabardel, 1995) and in the notion of instrumental orchestration, proposed by Trouche (2003, 2004, 2005) and expanded by Drijvers et al. (2010). The theoretical corpus was constituted by selecting academic publications on recurrent sequences, programming in mathematics teaching, and references in mathematics education, including classic works and recent studies published in journals in the field. The analysis was carried out through analytical and interpretative reading, focusing on identifying concepts, theoretical relationships, and didactic elements to support the preparation of the proposal. In this context, the teacher's role in the organisation of environments, computational instruments, and tasks stands out, as does the structuring of didactic interactions mediated by programming.

As its main contribution, the article proposes a theoretical systematisation that links mathematical and computational language in the teaching of recurrent sequences, thereby supporting the development of didactic activities aligned with a theoretically grounded computational approach. By discussing programming as an instrument mediated and orchestrated by the teacher, this research broadens the debate on the pedagogical use of digital technologies in basic education.

The results presented contribute to the development of algebraic and algorithmic thinking, as well as to teacher education, by highlighting opportunities to establish connections between mathematics and computing. In addition, the aim is to expand the methodological repertoire available for teaching recurrent sequences, favouring pedagogical practices that value multiple registers of representation and the exploration of patterns.

AN OVERVIEW OF RECURRENT SEQUENCES

Recurrent sequences are a class of mathematical objects defined by the functional dependence between consecutive terms, usually expressed through

linear or nonlinear recurrence relations, with varying initial orders and conditions. Although traditionally explored in basic education through the Fibonacci sequence, these structures have been the subject of increasing interest in the mathematical literature, which has begun to investigate extended families of recurrent sequences, their generalisations and asymptotic, combinatorial, and algebraic properties (Koshy, 2019).

In recent decades, there has been a movement to amplify recurrent sequences, with particular attention to sequences defined by higher-order recurrences, generalised recurrences, and parametric variations (Kızılateş et al., 2025). Recent work has analysed, for example, sequences such as Lucas, Jacobsthal, Narayana, Padovan, Perrin, Tribonacci, Tetranacci, and other extensions (Tastan et al., 2021; Alves et al., 2024), exploring properties such as asymptotic growth, relationships between distinct sequences, generating functions, and connections with more general algebraic structures. This panorama reveals that recurrent sequences are no longer treated merely as mathematical curiosities or isolated examples but occupy a significant place in mathematical research.

In this context, the production of Yüksel Soykan stands out[1], as it systematises and proposes several new recurrent sequences, analysing their structural properties and growth behaviours. In his works, the author presents families of sequences defined by second- and third-order linear recurrences, often associated with classical sequences, but with adjustable parameters that generate new mathematical behaviours. This approach shows how small variations in the recurrence rule or initial conditions can produce sequences with different dynamics, thereby expanding the exploratory potential of these objects.

Table 1 presents a synthetic overview of some recurrent sequences discussed in the literature, including classic sequences and recent extensions, highlighting their recursive definitions and initial conditions.

[1] https://scholar.google.com/citations?user=7yBSCNAAAAAJ&hl=en

Table 1

Recurrent sequences and their recurrence relationships.

Sequence	Recurrence relation	Initial conditions
Fibonacci	$F_n = F_{n-1} + F_{n-2}$	$F_0 = 0, F_1 = 1$
Tribonacci	$T_n = T_{n-1} + T_{n-2} + T_{n-3}$	$T_0 = 1, T_2 = 1, T_3 = 2$
Lucas	$L_n = L_{n-1} + L_{n-2}$	$L_0 = 2, L_1 = 1$
Jacobsthal	$J_n = J_{n-1} + 2J_{n-2}$	$J_0 = 0, J_1 = 1$
Narayana	$N_n = N_{n-1} + N_{n-3}$	$N_0 = N_1 = N_2 = 1$
Mersenne	$M_n = 2M_{n-1} + 1$	$M_0 = 0$
Padovan	$P_n = P_{n-2} + P_{n-3}$	$P_0 = P_1 = P_2 = 1$
Perrin	$R_n = R_{n-2} + R_{n-3}$	$P_0 = 3, P_1 = 0, P_2 = 2$
Tribonacci	$T_n = T_{n-1} + T_{n-2} + T_{n-3}$	$T_0 = 0, T_1 = 0, T_2 = 1$
Pell	$P_n = 2P_{n-1} + P_{n-2}$	$P_0 = 0, P_1 = 1$
Pell-Lucas	$PL_n = 2PL_{n-1} + PL_{n-2}$	$PL_0 = 2, PL_1 = 2$
Horadam[2]	$H_n = pH_{n-1} + qH_{n-2}$	$H_0 = a, H_1 = b$
Leonardo	$Le_n = Le_{n-1} + Le_{n-2} + 1$	$Le_0 = Le_1 = 1$
Oresme	$O_n = O_{n-1} - 1/4O_{n-2}$	$O_0 = 0, O_1 = 1/2$
Phone numbers	$T_n = T_{n-1} + (n-1)T_{n-2}$	$T_0 = T_1 = 1$
Ernst	$E_n = 2E_{n-1} + E_{n-2} - 2E_{n-3}$	$E_0 = 0, E_1 = 1, E_2 = 2$
Guglielmo	$G_n = 3G_{n-1} - 3G_{n-2} + G_{n-3}$	$G_0 = 0, G_1 = 1, G_2 = 3$
Blaise	$B_n = B_{n-2} + 2B_{n-3} + 1$	$B_0 = 0, B_1 = B_2 = 1$
Pierre	$P_n = P_{n-1} + P_{n-2} + P_{n-3} + 1$	$P_0 = 0, P_1 = 1, P_2 = 2$
Richard	$R_n = R_{n-2} + R_{n-3} + 1$	$R_0 = 0, R_1 = R_2 = 1$

The diversity of structures presented in Table 1 shows that recurrent sequences share a common logic: dependence on previous terms but differ in the order of recurrence and initial conditions. From a mathematical standpoint, this variety allows for the investigation of asymptotic patterns, regularities, and properties. From a didactic standpoint, such differences offer opportunities for conceptual exploration, especially when mediated by computational tools.

[2] The values of a, b, p and q are adjustable.

The computational approach, in this scenario, plays a central role, since the algorithmic implementation of recurrent sequences allows to quickly explore different recurrence rules, compare behaviours and visualise growth patterns that would be less evident only through symbolic analysis. Programming makes it possible, for example, to change parameters, modify initial conditions, and generate graphical representations that make structural differences between sequences visible, favouring the understanding of concepts such as exponential growth, recursive dependence, and pattern stability.

However, despite the mathematical content and exploratory potential of these sequences, we observe that in the context of basic education, the exploration of recurrent sequences remains restricted to a few classic cases, often approached procedurally and disconnected from a broader reflection on their common structures. This finding reinforces the need for didactic proposals that, without losing sight of the curriculum and the cognitive limitations at this level of education, enable students to understand recurrent sequences as a class of mathematical objects rather than merely as isolated examples.

Therefore, the study does not intend to hierarchise or assign intrinsic formative value to certain sequences to the detriment of others. On the contrary, a didactic and methodological approach is explicitly assumed, in which some classic and generalised recurrent sequences, such as Fibonacci, Lucas, Jacobsthal, Narayana, and Mersenne, are mobilised as representative examples of different types of recurrence rules and initial conditions. In this sense, the focus now becomes less on the sequences themselves and more on the possibility of using recurrences as a general mathematical object, from which it becomes possible to investigate, compare, and make visible, via algebraic, computational, and visual records, as well as the effects of recursive choices on the behaviour of the sequences.

Thus, the panorama presented in this section justifies the choice of the sequences explored in the following sections and the adoption of a computational approach mediated by instrumental orchestration, in which programming is mobilised as a didactic instrument to make the mathematical structures underlying the recurrent sequences visible, manipulable, and generalisable.

MATHEMATICS AND PROGRAMMING: CONNECTIONS VIA RECURRENT SEQUENCES

The approach proposed in this article assumes that students have introductory programming knowledge, such as understanding variables,

arithmetic operations, and repetition commands, which are common elements in initial computing courses and necessary for a computational language to be mobilised as a didactic instrument in mathematics teaching. These conditions make it possible to explore recurrent sequences as a point of convergence between mathematics and programming, since their algebraic structure, based on dependency relationships between successive terms, naturally translates into iterative processes.

Mathematics and computing share a set of competencies that naturally bring them closer together, including logical rigour, procedural precision, and the structuring of algorithmic processes (Ferragina & Luccio, 2024). Both mathematical thinking and computational thinking operate on principles that involve analysis, abstraction, algebraic reasoning, and the ability to break down problems into organised steps. Mathematical logic, when translated into computational operations, becomes algorithmic reasoning (Santos & Menezes, 2025). In the same way, the development of algorithms in programming reflects the same foundations that guide the construction of proofs, the elaboration of recurrences, and the study of numerical patterns. Thus, mathematics and computing are interdependent fields that reinforce each other.

Given the diversity of recurrent sequences presented in the previous section, this section focuses on how such structures can be computationally operationalised, regardless of the specific sequence. From this perspective, mathematical recurrences occupy a prominent place, as they clearly materialise the interaction between algebraic reasoning and algorithmic thinking. The functional dependence between successive terms, a fundamental characteristic of recurrences, captures the very kind of structured logic that underpins both mathematics and computing. When such recurrent patterns are translated into programming, the mathematical structure that defines the sequence is directly reflected in recursive or iterative computational processes, allowing students to recognise the correspondence between the algebraic formulation and its algorithmic execution.

Thus, the very rule that generates the classic Fibonacci sequence:

$$F_n = F_{n-1} + F_{n-2} \text{ for } \quad F_0 = 0 \text{ and } F_1 = 1,$$

For example, it can naturally become an algorithm implemented using repetition constructs, such as a *for* loop in Python. In this way, mathematical recurrence is no longer just a symbolic expression but becomes an executable procedure, in which each term is constructed step by step, showing the student the direct correspondence between mathematical logic and algorithmic logic.

Figure 1 schematically shows the transposition of symbolic mathematical reasoning to its computational representation.

Figure 1

Correspondence between the mathematical representation and the computational algorithm.

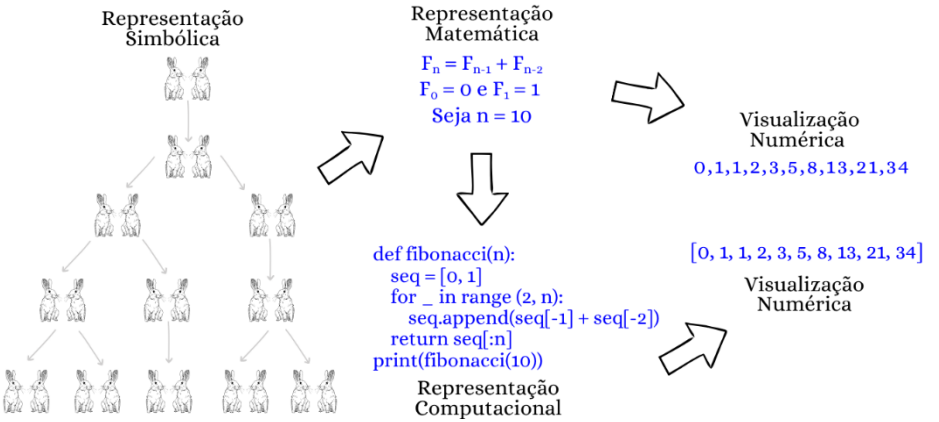


Figure 1 shows the conceptual path that connects different ways of representing the Fibonacci sequence. It begins with the symbolic representation, traditionally associated with the rabbit problem, and moves on to the algebraic formalisation of recurrence relations. From this well-defined structure, it becomes possible to establish its corresponding computational representation, the execution of which depends directly on the previously formulated logical rigour. Thus, we note that both representations — mathematical and computational — converge to the same numerical values, demonstrating the consistency and complementarity between the different records.

The transition from mathematical to computational language, as illustrated in Figure 1, constitutes a conceptual transformation framework that requires students to understand the algebraic and operational logic of algorithms. In the case of sequences, this transposition becomes evident: the recurrence relations that define each sequence are converted, in the computational environment, into iterative structures that repeatedly apply the mathematical principle. Table 2 shows different sequences and allows us to observe the path of understanding that is established when the loop *for* goes through each step of the calculation.

Table 2

Different sequences written in the Python programming language.

a) $\boldsymbol{N_{n+1} = N_n + N_{n-2}}$

$N_0 = 1$, $N_1 = 1$ *and* $N_2 = 1$

```
1 def narayana(n):
2  seq = [1, 1, 1]
3  for _ in range(3,n):
4   seq.append(seq[-1]+seq[-3])
5  return seq[:n]
```

b) $\boldsymbol{L_{n+1} = L_n + F_{n-1}}$

$L_0 = 2$ *and* $L_1 = 1$

```
1 def lucas(n):
2  seq = [2, 1]
3  for _ in range(2, n):
4   seq.append(seq[-1] + seq[-2])
5  return seq[:n]
```

c) $\boldsymbol{J_{n+1} = J_n + 2J_{n-1}}$

$J_0 = 0$, *and* $J_1 = 1$

```
1 def jacobsthal(n):
2  seq = [0, 1]
3  for _ in range(2, n):
4   seq.append(seq[-1] + 2 * seq[-2])
5  return seq[:n]
```

d) $\boldsymbol{M_{n+1} = 3M_{n+1} - 2M_n}$

$M_0 = 0$, *and* $M_1 = 1$

```
1 def mersenne(n):
2  seq = [0, 1]
3  for _ in range(2, n):
4    seq.append(3*seq[-1] - 2*seq[-2])
5  return seq[:n]
```

In all the functions presented, the mathematical structure of the sequences is directly translated into the *for* loop (line 4), where each new term is generated iteratively. The initial conditions corresponding to the base values of each recurrence are defined before the loop (line 2), since they need to be established only once for the sequence to develop correctly. Thus, all generated terms are stored in the *seq* list, which serves as a structured repository for the sequence, ensuring both the retrieval of previous values and the orderly construction of subsequent ones.

This articulation among symbolic, algebraic, and geometric registers aligns with the theory of registers of semiotic representation (TRSR) (Duval, 1995), according to which mathematical learning becomes more consistent when the student can coordinate and convert information between registers. In this sense, a programming language can function as a complementary representation record by allowing algebraic expressions that define the sequences to be converted into visual representations.

This difference in behaviour, resulting from the recurrent orders, initial conditions, and growth rates associated with the recurrent sequences, can be observed in Figure 2b. When the sequences are plotted, the visualisation shows that they all exhibit exponential growth at different rates, an aspect that would be less apparent from a numerical analysis of the terms alone. The curve

associated with the Mersenne and Jacobsthal sequences shows a progression close to base 2, while the Lucas sequence has a more moderate slope, with an approximate growth ratio of 1.618; and the Narayana sequence exhibits slower growth, due to its higher-order recurrence. Thus, the coordinated use of different representations — algebraic, computational and graphical — broadens the understanding of the structural patterns involved in recurrent sequences.

Figure 2

(a) Code for generating the Lucas sequence graph. (b) Graphical comparison of the asymptotic growth of the sequences.

a) **b)**

```
1 import matplotlib.pyplot as plt
2
3 N = 10
4 x = list(range(1, N + 1))
5
6 seq = lucas(N)
7 color = "royalblue"
8 formula = (r"Asymptotic growth:" "\n"
9 r"$L_n \sim \varphi^n,\ "
10 r"\varphi=\frac{1+\sqrt{5}}{2}$")
11
12 plt.plot(x, seq, "-o",ms=10,
13   lw=2, alpha=0.7,mfc=color,
14   mec=color, color=color,
15   label="Lucas"
16   "\n[2,1,3,4,7,11,18,29,47,76]")
17 plt.grid(True, linestyle="--")
18
19 for xi, yi in zip(x, seq):
20     plt.text(xi - 0.15, yi, f"{yi}")
21
22 plt.annotate(formula,
23     xy=(9.5, 60),xytext=(0.4, 0.6),
24     textcoords="axes fraction",
25     bbox=dict(boxstyle="round,pad=0.4",
26     fc="white", ec=color, lw=2),
27     arrowprops=dict(arrowstyle="->",
28     lw=2,color=color))
29
30 plt.xlabel("Iteration (n)")
31 plt.ylabel("Sequence value")
32 plt.legend()
33 plt.show()
```

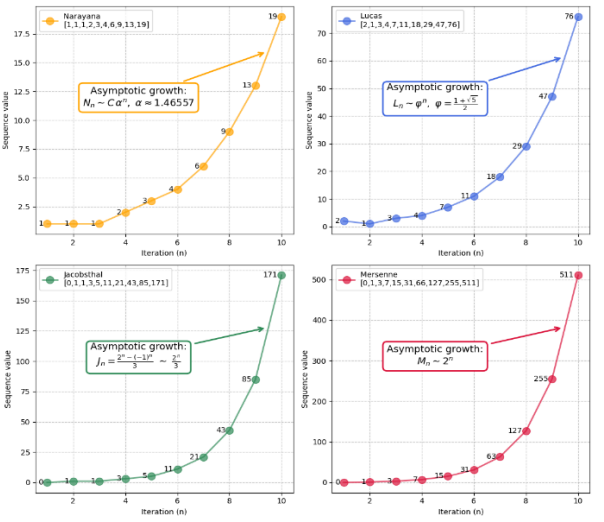


Figure 2a) shows the section of code responsible for generating the Lucas sequence graph. The script is relatively simple to interpret: lines 12 to 16 define the information regarding *asymptotic growth*, using the formula stated in lines 8 to 10. Next, the numerical values for each point on the graph are entered in lines 19 and 20. Finally, lines 22 to 28 implement the writing routine with an arrow, which graphically highlights the asymptotic behaviour and helps the student identify the sequence's growth trend.

In summary, this section highlights the importance of integrating mathematical and computational language into students' education and shows how recurrent sequences can enable the simultaneous development of algebraic, logical, and algorithmic thinking. By exploring symbolic, algebraic, computational, and visual representations, we demonstrated that programming not only materialises the logic of recurrences but also broadens understanding of numerical patterns and supports learning through multiple representations.

INSTRUMENTAL ORCHESTRATION IN THE TEACHING OF RECURRENT SEQUENCES

What has been shown so far demonstrates that recurrent sequences are an important point of connection between mathematical language and computational language, allowing students to move between algebraic, algorithmic, and visual registers. However, for this integration to be didactically effective in basic education, it is necessary to consider not only the technological tools involved but, above all, how these tools are organised and mediated by the teacher in the classroom. It is in this context that the notion of instrumental orchestration is situated, understood as a central element for the organisation of interactions among students, programming, and mathematical objects.

Based on Rabardel's instrumental theory (1995), instrumental orchestration is a didactic framework aimed at the organisation and management of teaching mediated by technological artefacts. This notion was systematised by Trouche (2004) in discussing ways to intentionally configure both the mathematical content and the conditions for its exploration in the classroom, considering how tasks are carried out and the teacher's role in regulating activities. Subsequently, this approach was further developed by Drijvers et al. (2010), expanding the understanding of instrumental orchestration beyond the organisation of instruments to include aspects of didactic performance and interactions within technological environments.

In this sense, instrumental orchestration refers to the set of didactic decisions teachers make regarding didactic configurations, the selection and sequencing of tasks, and the conduct of classroom interactions (Drijvers & Trouche, 2008). From this perspective, the programming language is not conceived solely as a technical tool but as a teaching instrument whose educational potential depends directly on how it is used, guided, and regulated by the teacher throughout the teaching process.

Therefore, in teaching recurring sequences, programming serves as a mediating tool that operationalises algebraic relationships through repetition structures, as discussed. In this case, instrumental orchestration involves intentionally organising situations that lead students to establish explicit correspondences between the mathematical definition of the sequence and its algorithmic implementation. By orchestrating these situations, the teacher guides students towards the elements of recurrence, thereby facilitating an understanding of iterative processes.

Structuring the teaching situation

A possible instrumental orchestration configuration can be structured through a didactic sequence that integrates the study of recurrent sequences and programming in basic education. From this perspective, the proposal is to create a didactic situation based on Brousseau's (1986) TDS and aligned with the principles of instrumental orchestration.

The proposed task consists of:

Problem: *Develop a computational algorithm that asks the user for a positive integer value "n" and, from that value, generates and prints the first "n" terms of the Fibonacci sequence.*

This problem was chosen because it simultaneously mobilises mathematical knowledge (recurrence, initial conditions, and sequence growth) and computational knowledge (variables, repetition structures and flow control), enabling the teacher to orchestrate different modes of representation throughout the didactic situation.

Within the scope of the TDS, the didactic situation corresponds to the set of conditions the teacher organises for the student to interact with a mathematical problem. From the perspective of instrumental orchestration, it is up to the teacher to define the role of the programming language, the arrangement of resources (code editor, projector, boards, graphics) and the forms of interaction between students and digital artefacts. This situation is structured in the stages of action, formulation, validation, and institutionalisation, which must be orchestrated by the teacher.

Action

In the action phase, students make initial contact with the mathematical problem proposed by the teacher, interacting with it according to the conditions organised in the teaching environment. Within the scope of the TDS, this phase is characterised by a situation in which the student is led to act on a problem,

mobilising prior knowledge and personal strategies, before mathematical knowledge has been formalised. It is, therefore, a time when the focus is on students exploring and taking the initiative, interacting with an environment carefully structured by the teacher (Mangiante-Orsola et al., 2018).

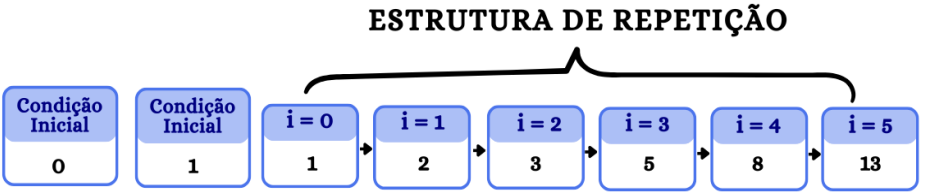
From this perspective, the teacher can ask students to investigate how the recursive definition of the Fibonacci sequence can be interpreted as an iterative process in programming. It is up to the teacher to define the available tools, the teaching time, and the type of interaction expected (individual or group). From the perspective of instrumental orchestration, these decisions can constitute an intentional organisation of interactions, in which the teacher anticipates the role that certain computational artefacts may play in the development of the activity, creating conditions for these artefacts to become instruments of mathematical learning (Gueudet & Trouche, 2011).

In this first phase, the teacher can explain the computational problem, highlighting essential elements such as the definition of the Fibonacci sequence, the need to prompt the user for a value n, and the iterative generation of the terms in the sequence. From the perspective of instrumental orchestration, the teacher can anticipate that the structure of repetition (loop) will be the key instrument in the activity and guide students to think of mathematical recurrence as a repetitive process controlled by an index. It is not yet a matter of teaching the code directly, but of creating a medium that elicits algorithmic action, favouring the mobilisation of prior knowledge of basic programming and mathematical concepts related to recurrence.

In addition, the teacher can orchestrate the activity by encouraging students to draw a flowchart representing the operation of a repetition structure, explaining the role of the index i as the controller of the iterative process. This type of intermediate representation acts as a mediator between the algebraic and computational records, favouring processes of instrumental genesis. By proposing this task, the teacher can anticipate recurrent difficulties, such as confusion between the initial conditions of the sequence and the start of the iteration itself, by organising a teaching method that favours understanding of the logic underlying the algorithm (Trouche, 2003). An example of this organisation is shown in Figure 3, which illustrates the correspondence between the mathematical structure of the recurrence and an iterative process controlled by a repetition loop.

Figure 3

Fibonacci Sequence.



Furthermore, the teacher can guide the action situation by encouraging students to build a step-by-step explanation of how the repetition structure works, describing how the loop index *for* advances and how, with each iteration, a new term in the Fibonacci sequence is calculated from the previous ones. This orchestrated explanation is aimed not only at the correct execution of the algorithm, but also at understanding the mathematical process that occurs within the loop. To this end, it is essential that the teacher set aside specific teaching time for individual reflection and collective discussion, regulating interactions to encourage the clarification of reasoning and the comparison of strategies, a central aspect of action situations (Sensevy, 2012).

Table 3 presents an example of a descriptive summary that students can construct and organise under the teacher's guidance, helping them understand the iterative dynamics of the sequence. This summary aims to clarify the actions carried out and the regularities observed, which is characteristic of the action phase in TDS.

Table 3

Repetition structure for the Fibonacci sequence considering n = 5.

Index (*i*)	Calculation performed	Explanation
0	Sum = 1+0 = 1	*i* starts at 0, adds the first two terms (0 and 1) and goes back to the loop.
1	Sum = 1+1 = 2	*i* increments to 1, adds the two previous terms (1 and 1), and returns to the loop.
2	Sum = 2+1 = 3	*i* increments to 2, adds the two previous terms (2 and 1) and returns to the loop.
3	Sum = 3+2 = 5	*i* increments to 3, adds the two previous terms (3 and 2) and returns to the loop.
4	Sum = 5+3 = 8	*i* increments to 4, adds the two previous terms (5 and 3) and returns to the loop.
5	Sum = 8+5 = 13	*i* goes to 5, adds the two previous terms (8 and 5) and returns to the loop. i increments to 6 but ends at the condition n < 6.

Formulation

In the formulation phase, the teacher can guide students to explain and organise the strategies used in resolving the task, promoting a transition from exploratory action to a discursive, structured activity. Within the framework of TDS, this stage corresponds to the point at which students are led to formulate, through different registers, the procedures and relationships that guided their actions, making them communicable and open to collective analysis (Almouloud, 2010). This is, therefore, a key moment in the educational situation, in which knowledge begins to emerge explicitly, albeit not in an institutionalised way.

Thus, within the framework of instrumental orchestration, it is up to the teacher to organise situations that favour this clarification, regulate interactions, and guide the choice of the most appropriate registers for the progressive formalisation of reasoning. In this context, students can be encouraged to express their strategies by writing pseudocode, inserting explanatory comments in the code or providing organised verbal descriptions. These forms of expression allow the articulation of different registers of representation, algebraic, computational, and verbal, which are aspects of the different registers of semiotic representation (Duval, 1995).

In this way, the teacher can serve as a mediator between the recursive formula of the Fibonacci sequence and the iterative structure used in the algorithm, facilitating the conversion from algebraic to computational notation. This conversion requires the teacher to intervene by highlighting regularities, explaining analogies, and anticipating common conceptual difficulties, such as confusion about auxiliary variables, the order in which terms in the sequence are updated, or the role of initial conditions in the iterative process. This mediation work is characteristic of instrumental orchestrations, in which the teacher guides the use of artefacts to facilitate the construction of mathematical meanings (Trouche, 2004).

Thus, one path the teacher can orchestrate at this stage is the guided construction of pseudocode. This feature serves as an intermediary between mathematical language and implementation in a programming language, reinforcing the idea that, as in mathematics, programming requires logical rigour, clarity in defining steps, and coherence in organising procedures. The pseudocode presented in Table 4 exemplifies this intermediate function by explaining, in a structured way, the sequence of actions necessary to generate the recurrence terms: initially, reading the value n; then, defining the initial

conditions of the sequence; and, finally, implementing the repetition structure responsible for the iterative calculation of the terms.

Table 4

Pseudocode for the Fibonacci sequence.

```
1. Start
2.   Read n
3.   Set a ← 0
4.   Set b ← 1
5.   For i from 0 to n - 1 do
6.   Show
7.   sum ← a + b
8.      a ← b
9.   b ← sum
10.  End For
11. End
```

Based on this pseudocode, it is essential for the teacher to explore the meaning of each instruction with the students. This must occur from both mathematical and computational perspectives. This is the right time for students to understand that, in the mathematical definition of the Fibonacci sequence, the initial conditions $F_0 = 0$ and $F_1 = 1$ are fundamental values, as the recurrence can only be applied from them (lines 3 and 4).

Thus, the teacher can highlight that:

- Variable *a* represents the term: F_{n-2};
- Variable *b* represents the term:F_{n-1}.

Thus, before looping begins, the algorithm assumes the initial conditions established by the mathematical definition of the recursive formula

$$F_n = F_{n-1} + F_{n-2},$$

represented in line 7. At that point, already inside the loop, a stores F_{n-2}, b stores F_{n-1} . The instruction on line 7 calculates F_n by adding these two terms and assigns the result to the variable sum. After calculating the new term, the algorithm needs to move forward in the sequence, which, in mathematical terms, corresponds to moving from F_n to F_{n+1}. This is done by following the instructions in lines 8 and 9. These two lines perform a shift in recurrence, where:

- $a \leftarrow b$: the value that was F_{n-1} now takes the place of F_{n-2};

- $b \leftarrow sum$: the newly calculated value F_n becomes F_{n-1}.

In this way, the teacher can show that although the Fibonacci sequence is mathematically defined by a recursive relationship, its implementation in pseudocode transforms this recursion into an iterative process controlled by a loop. The loop on line 5, "for *i* from 0 to n − 1 do", does not represent the terms of the sequence; it controls how many times the recurrence is applied, directly linked to the value of n. This conceptual distinction, carefully orchestrated by the teacher, is essential to avoid misinterpretations and to consolidate understanding of the connection between mathematical language and computational language.

Validation

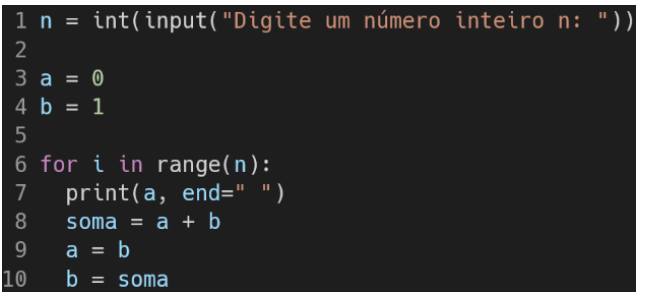
The validation phase is when the computational algorithms are developed and then tested, analysed, and compared with the expected results of the Fibonacci sequence. At this stage, students can run their code with different values of n, observe the outputs, and check whether the terms generated correspond to the sequence. Within the framework of TDS, this stage is characterised by the need to justify, argue, and verify solutions, establishing explicit relationships between the actions taken and mathematical knowledge (Brousseau, 2002).

Based on instrumental orchestration, validation does not occur spontaneously: it is intentionally organised by the teacher, who regulates interactions, defines verification criteria, and structures the environment to favour critical analysis of the results. It is up to the teacher to anticipate possible conceptual and technical errors, such as incorrect initialisation of initial conditions, inappropriate use of the loop index, incorrect updating of auxiliary variables, or even confusion between iterative and recursive processes. By anticipating these difficulties, the teacher can orchestrate situations in which mistakes emerge as topics for group discussion, turning them into opportunities for learning.

When students rely on the pseudocode in Table 4, the final algorithm implemented in Python tends to have a structure similar to that shown in Figure 4. In this context, the teacher can explore with the students the correspondence between the parts of the code and the elements of the mathematical recurrence, asking, for example, whether the values generated for n = 2, 5, 7, or 10 are in accordance with the expected sequence. This type of intervention reinforces validation as an argumentative process in which the results need to be explained, not just observed.

Figure 4

Python computational code for the Fibonacci sequence.



```
n = int(input("Digite um número inteiro n: "))

a = 0
b = 1

for i in range(n):
  print(a, end=" ")
  soma = a + b
  a = b
  b = soma
```

At this stage, the *milieu* consists of the source code itself, the numerical outputs generated by the programme, the tests carried out with different values of *n*, and the collective interactions fostered in the classroom. Producing this feedback can allow students to compare results, identify inconsistencies and adjust their implementation strategies. The teacher, in turn, can serve as a mediator, guiding the interpretation of the results, encouraging students to justify their algorithmic choices and promoting the connection between the computational and mathematical definitions of the sequence.

At the end of the validation phase, the teacher can expand the educational challenge by testing students' ability to generalise the algorithm they have developed. This expansion can be predicted by modifying the recurrence rule, asking students to adapt the code (Figure 4) to generate other sequences (Table 1). This proposal makes it possible to verify whether students have understood the structural logic of recurrent sequences, i.e., the dependence between terms, the role of initial conditions, and the organisation of the loop, rather than just the specific implementation of the Fibonacci sequence.

By modifying the mathematical definition and adjusting the corresponding algorithm, we believe that students can be led into a new validation cycle, in which they need to test the consistency of their solutions, identify new difficulties, and justify their procedures again. In this way, validation takes on an expanded and dynamic character, highlighting the flexibility of the computational tool and its didactic potential for teaching various recurring models.

Institutionalisation

Finally, in the institutionalisation phase, the teacher must explicitly assume the role of mediator of mathematical knowledge, revisiting the work

produced in the previous stages and attributing meaning to the knowledge mobilised throughout the teaching situation. According to Almouloud (2010), the student carries out the action, formulation, and validation phases, while the teacher:

> […] oversees the institutionalisation phase. He must determine the form and content of the knowledge he wishes to grant official status, taking into account the effects of didactic transposition (Almouloud, 2010, p. 42).

Therefore, at this point, the central concepts can be systematised, including the formal definition of a recurrent sequence, the role of initial conditions in determining the terms of the sequence, and the interpretation of the loop as a computational representation of the recurrence relation. This formalisation helps consolidate the articulation of algebraic language and algorithmic language.

In the process of institutionalisation, it is essential for teachers to monitor and analyse students' knowings actually mobilised, identifying possible conceptual gaps, ambiguities in the interpretation of algorithms or difficulties in understanding the correspondence between mathematical formulae and their computational implementation. To this end, the teacher can organise opportunities for students to share their work, hold group discussions or present the solutions they have developed, creating conditions for students to explain their thinking processes. These interactions allow the teacher to regulate mathematical discourse, highlighting essential elements and correcting partial or mistaken interpretations.

Furthermore, institutionalisation constitutes the moment to clarify the generality of the knowledge constructed. The teacher can emphasise that the recurrent sequences studied are not limited to the examples addressed but belong to a broader class of mathematical objects characterised by rules governing the dependence among successive terms. From a computational point of view, this generalisation means that the algorithmic structure developed can be adapted to model different sequences by modifying only the initial conditions and the recurrence rule.

At the end of the institutionalisation process, the teacher can observe that when intentionally and theoretically integrated into mathematics teaching, the programming language ceases to be merely a technical artefact and becomes an institutionalised mathematical tool (Rabardel, 1995). In this sense, institutionalisation can consolidate not only the content covered but also the

methodology adopted, highlighting the central role of teacher orchestration in connecting mathematics and computing in primary education.

FINAL CONSIDERATIONS

This article aimed to analyse how the connection between mathematics and programming, mediated by the study of recurrent sequences, can foster conceptual understanding of this mathematical object in basic education, with consideration of instrumental orchestration. The guiding question of the research was how programming can be organised and mobilised as a teaching tool to promote the connection between algebraic and computational registers associated with recursive sequences. The analyses indicate that this connection is theoretically and didactically consistent, since programming, when orchestrated by the teacher, serves to explain, operationalise, and generalise the mathematical structure of recurrence relations.

When discussing the connections between mathematics and programming through recurrent sequences, it became clear that algorithmic language is not merely a technical resource but a semiotic register that complements the algebraic register. The computational implementation of the Fibonacci sequence allows students to observe the dynamics of the recurrence step by step, making visible aspects that often remain implicit in the symbolic expression of the formula. This approach favours understanding the role of initial conditions, the rule of formation, and the dependence between consecutive terms, strengthening the construction of mathematical meanings.

Thus, by adopting instrumental orchestration as a theoretical framework, we discussed the role of the teacher in organising the teaching environment, anticipating difficulties, and guiding possible interactions of students, the computational artefact, and mathematical knowledge. Within the scope of this discussion, programming is understood as an instrument in the process of instrumental genesis, whose appropriation depends on the teacher's didactic choices and the tasks proposed. This perspective reinforces the idea that the use of digital technologies alone does not guarantee learning, and that well-founded pedagogical mediation is necessary.

Also in this regard, the structure of the teaching situation was presented in terms of the phases of action, formulation, validation, and institutionalisation. Analysed in theoretical terms, this organisation outlines a possible didactic path that guides students from initial exploration of the problem to the formalisation of the mathematical and computational concepts involved. The proposition of pseudocode, the testing of algorithms with different values, and the

institutionalisation of knowings can foster the articulation of mathematical and computational thinking, as well as support the generalisation of other recurrent sequences, broadening the conceptual scope of the proposal.

Despite the discussions presented, it is important to recognise that the nature of this study is theoretical, not involving a direct application of the proposal in a school context or collection of empirical data from students. Thus, the results discussed pertain to didactic and analytical potential rather than to empirical evidence of learning. As prospects, we highlight the need to develop empirical studies that implement the proposed teaching sequence in the classroom, enabling analysis of students' work, the difficulties they face, the strategies they use, and the impacts on mathematical learning.

We can therefore conclude that articulating mathematics and programming, guided by instrumental orchestration, can be a promising path for teaching recurrent sequences, contributing to a more integrated and meaningful approach aligned with mathematics education mediated by digital technologies.

ACKNOWLEDGMENTS

The first author would like to thank the support received for carrying out the research conducted during the postdoctoral fellowship, linked to the Postgraduate Programme in Education of the Northeast Education Network (RENOEN/UFAL), under the supervision of Prof. Dr. Francisco Régis Vieira Alves, CNPq Research Productivity Fellow (2020-2026). He also expresses his gratitude for the opportunity to join an international partnership with Prof. Dr. Ana Paula Aires to conduct research in the field of mathematics education linked to programming.

AUTHORSHIP CONTRIBUTION STATEMENT

The first author, JFS, was responsible for designing the study and drafting the first version of the article. The authors FRVA and APA contributed to the proposal of the methodological design and to the elaboration of the theoretical framework. All authors participated in the overall design of the study, the discussion of the results, the critical review of the manuscript and approved the final version submitted to the journal.

DECLARATION OF DATA AVAILABILITY

Data sharing is not applicable to this article as it is a publicly available bibliography search.

REFERENCES

Alecci, G., Barbero, S., & Murru, N. (2025). Some notes on the algebraic structure of linear recurrent sequences. *Ricerche di Matematica, 74*, 431–447. https://doi.org/10.1007/s11587-023-00826-5

Almouloud, S. A. (2010). *Fundamentos da didática da matemática*. Editora UFPR.

Alves, F. R. V., Catarino, P. M. M. C., Vieira, R. P. M., & Spreafico, E. V. P. (2024). Combinatorial approach on the recurrence sequences: An evolutionary historical discussion about numerical sequences and the notion of the board. *International Electronic Journal of Mathematics Education*, *19*(2). https://doi.org/10.29333/iejme/14387

Berend, D., & Kumar, R. (2025). Kepler sets of linear recurrence sequences. *Acta Mathematica Hungarica, 175*(1), 54–95. https://doi.org/10.1007/s10474-025-01506-6

Binotto, R. R., Maltempi, M. V., & Silva, R. D. (2023). *Potencialidades da programação em Python para o desenvolvimento do pensamento criativo em matemática. Zetetike, 31(00), e023015*. DOI: https://doi.org/10.20396/zet.v31i00.8672180

Borg, A., & Fahlgren, M. (2025). Students' development of instrumented action schemes for numerically determining limits using programming. *Digital Experiences in Mathematics Education*, 1–33. https://doi.org/10.1007/s40751-025-00177-w

Brandsaeter, A., & Berge, R. L. (2025). Promoting mathematical competence development through programming activities. *Educational Studies in Mathematics, 119*(2), 225–247. https://doi.org/10.1007/s10649-024-10380-y

Brousseau, G. (1986). Fondements et méthodes de la didactique des mathématiques. *Recherches en Didactique des Mathématiques, 7*(2), 33–115.

Brousseau, G. (2002). *Theory of didactical situations in mathematics: Didactique des mathématiques, 1970–1990*. Springer.

Clark-Wilson, A., Robutti, O., & Sinclair, N. (Eds.). (2014). *The mathematics teacher in the digital era: An international perspective on technology focused professional development*. Springer. https://doi.org/10.1007/978-94-007-4638-1

Dişkaya, O., Menken, H., & Catarino, P. M. M. C. (2023). On the hyperbolic Leonardo and hyperbolic Francois quaternions. *Journal of New Theory*, (42), 74–85. https://doi.org/10.53570/jnt.1199465

Djordjević, G. B., & Milovanović, G. V. (2014). *Special classes of polynomials*. University of Nis.

Drijvers, P., Doorman, M., Boon, P., Reed, H., & Gravemeijer, K. (2010). The teacher and the tool: Instrumental orchestrations in the technology-rich mathematics classroom. *Educational Studies in Mathematics, 75*(2), 213–234. https://doi.org/10.1007/s10649-010-9254-5

Drijvers, P., & Trouche, L. (2008). From artifacts to instruments. In *Research on technology and the teaching and learning of mathematics* (Vol. 2, pp. 363–391). Information Age Publishing.

Duval, R. (1995). *Sémiosis et pensée humaine: registres sémiotiques et apprentissages intellectuels*. Peter Lang.

Ferragina, P., & Luccio, F. (2024). *Computational thinking*. Springer Nature.

Goldenberg, E. P., Carter, C. J., Mark, J., Reed, K., Spencer, D., & Coleman, K. (2021). Programming as language and manipulative for second-grade mathematics. *Digital Experiences in Mathematics Education, 7*(1), 48–65. https://doi.org/10.1007/s40751-020-00083-3

Gueudet, G., & Trouche, L. (2011). Teachers' work with resources: Documentational geneses and professional geneses. In *From text to "lived" resources: Mathematics curriculum materials and teacher development* (pp. 23–41). Springer.

Hori, H., & Kida, M. (2025). Linear recurrent sequences providing decomposition law in number fields. *The Ramanujan Journal, 66*(3), 58. https://doi.org/10.48550/arXiv.2402.16357

Humble, N. (2022). Teacher observations of programming affordances for K-12 mathematics and technology. *Education and Information Technologies, 27*(4), 4887–4904. https://doi.org/10.1007/s10639-021-10811-w

Kieran, C. (Ed.). (2017). *Teaching and learning algebraic thinking with 5- to 12-year-olds*. Springer. https://doi.org/10.1007/978-3-319-68351-5

Kızılateş, C., Polatlı, E., Terzioğlu, N., & Du, W. S. (2025). On higher-order generalized Fibonacci hybrid numbers with q-integer components. *Symmetry, 17*(4), 584. https://doi.org/10.3390/sym17040584

Koshy, T. (2019). *Fibonacci and Lucas numbers with applications* (Vol. 2). John Wiley & Sons.

Mangiante-Orsola, C., Perrin-Glorian, M.-J., & Strømskag, H. (2018). Theory of didactical situations as a tool to understand and develop mathematics teaching practices. *Annales de Didactique et de Sciences Cognitives*, Special Issue, 145–174. https://doi.org/10.4000/adsc.334

Munthe, M., & Naalsund, M. (2024). Designing mathematical programming problems. *Digital Experiences in Mathematics Education*, *10*(2), 260–286. https://doi.org/10.1007/s40751-024-00143-y

Rabardel, P. (1995). *Les hommes et les technologies: une approche cognitive des instruments contemporains*. Armand Colin.

Rais, D., & Xuezhi, Z. (2024). Elevating student engagement and academic performance: A quantitative analysis of Python programming integration in the “Merdeka Belajar” curriculum. *Journal on Mathematics Education*, *15*(2), 495–516.

Santos, J. F., & de Menezes, I. F. (2025). Between facilities and learning: connecting computational thinking and mathematical skills through Python programming. *Revista Internacional de Pesquisa em*

Educação Matemática, *15*(3), 1–19. https://doi.org/10.37001/ripem.v15i3.4528

Santos, J. F. dos, Santos, M. B. Q. de C. P. dos, & Alves, F. R. V. (2026). Programação Python e Ensino de Matemática: Uma proposta interdisciplinar com Engenharia Didática. *PARADIGMA*, *47*(1), e2026004. https://doi.org/10.37618/PARADIGMA.1011-2251.2026.e2026004.id1761

Sensevy, G. (2012). About the joint action theory in didactics. *Zeitschrift für Erziehungswissenschaft, 15*(3), 503–516. https://doi.org/10.1007/s11618-012-0305-9

Soykan, Y. (2025). Generalized Jacobsthal-Narayana numbers and generalized co-Jacobsthal-Narayana numbers. *Asian Journal of Advanced Research and Reports, 19*(5), 9–55. https://doi.org/10.9734/ajarr/2025/v19i5998

Taştan, M., Özkan, E., & Shannon, A. (2021). The generalized k-Fibonacci polynomials and generalized k-Lucas polynomials. *Notes on Number Theory and Discrete Mathematics, 27*(2). https://doi.org/10.7546/nntdm.2021.27.2.148–158

Trouche, L. (2003). From artifact to instrument: Mathematics teaching mediated by symbolic calculators. *Interacting with Computers, 15*(6), 783–800. https://doi.org/10.1016/j.intcom.2003.09.004

Trouche, L. (2004). Managing the complexity of human/machine interactions in computerized learning environments. *International Journal of Computers for Mathematical Learning, 9*(3), 281–307.

Trouche, L. (2005). An instrumental approach to mathematics learning in symbolic calculator environments. In *The didactical challenge of symbolic calculators* (pp. 137–162). Springer. https://doi.org/10.1007/0-387-23435-7_7

Vieira Alves, F. R. (2022). Didactic Engineering (DE) and Professional Didactics (PD): A proposal for historical research in Brazil on

recurring number sequences. *The Mathematics Enthusiast*, *19*(1), 239-274. https://doi.org/10.54870/1551-3440.1551

Yusuf, A., & Noor, N. M. (2024). Modeling students' algorithmic thinking growth trajectories in different programming environments. *Smart Learning Environments, 11*(1), 38. https://doi.org/10.1186/s40561-024-00324-7