A methodological proposal that integrates symbolic problem-solving in physics with the MathJSLab software

Sergio Lindau
sergiolindau@gmail.com
ORCID: 0009-0006-9115-0291

July 10, 2026

Abstract

This work proposes that the learning of Physics and Mathematics in High School should be guided by the symbolic resolution of problems, delegating numerical execution to computational tools. To make this proposal viable, we present MathJSLab, a free software that simulates a subset of the MATLAB language. The proposed methodology is supported by an epistemological, ontological, and semiotic foundation, based on Actor-Network Theory and Cultural-Historical Activity Theory. These frameworks structure the basis for the pillars of Computational Thinking in order to overcome its criticisms, culminating in a methodological approach exemplified and articulated with a didactic resource. The study concludes by providing a well-founded strategy to overcome traditional approaches centered on manual calculations, effectively integrating computational mathematics into the daily life of High School education.

Keywords: Computational Thinking. Symbolic Problem Solving. Physics Teaching. MATLAB.

1 Introduction

Symbolic (or literal) problem-solving consists of listing all the equations relevant to the solution and performing algebraic manipulations with symbolic variables before substituting numerical values, converting units, and carrying out the arithmetic calculations to obtain the specific numerical solution.

In physics education literature, this method is described as a way to foster meaningful problem-solving, as opposed to a mechanical approach. It also means that, by solving problems in this manner, students construct algebraic and symbolic representations of real physical situations, thereby recreating the modeling and deduction processes that characterize scientific activity. In general, problem-solving can be understood as a strategy that mobilizes prior knowledge

and develops logical reasoning (Karam and Pietrocola, 2009; Oliveira, Araujo, and Veit, 2017). Another motivation for adopting a methodology based on literal problem-solving stems from the recurring difficulty we observe in students when applying their established mathematical skills to the field of physics. Research indicates that these failures arise not from a deficiency in abstract mathematics, but from an inability to contextualize (Badmus and Jita, 2024; Ince, 2018). In physics, equations describe real systems and symbols carry specific physical meanings, differing from the purely abstract relations of mathematics. Research dating back over 30 years (Pérez et al., 1992) had already identified a tendency to reduce problem-solving, in general, to a mechanical and superficial exercise in calculation. Therefore, the central pedagogical challenge in physics education lies in translating and applying mathematical knowledge to models of physical phenomena. Furthermore, it is recognized that problem-solving is an essential 21st-century competency and a high-level cognitive skill (Ince, 2018).

Nowadays, the term *Symbolic Problem-Solving* has become somewhat unwieldy, and many of its original formulations have been integrated into the scope of *Computational Thinking* (CT). CT is defined as a cognitive method for solving complex problems in a way that allows both humans and machines to execute the solutions effectively (Wing, 2006). While CT can be defined in various other ways, a critical distinction is always made: whereas computing is a process defined in terms of an underlying model (language, algorithms, data structures, the machine, etc.), CT is the mental process of formulating the problem and deriving the solutions that such a process will execute. In other words, it is not merely about programming, but about employing analytical capabilities to develop mathematical abstractions and structure problems so that algorithm design and problem-solving techniques can be applied effectively (Aho, 2011).

We propose that students, while solving problems symbolically, engage in another common practice of modern scientific activity: delegating the numerical computation to a computational mathematics tool. Thus, the teaching methodology we propose is epistemologically and ontologically aligned with contemporary scientific practice: it transforms teaching activity into action, and an action analogous to the agencies of modern Science and Engineering.

This separation between symbolic and deductive work (performed by hand) and numerical and executable work (performed via code) allows the pedagogical focus to shift toward the development of deeper competencies: the ability to model, represent, deduce, abstract, and translate the physical world into symbolic languages.

To make our methodological proposal fully viable — and to ensure its long-term sustainability for use by teachers in daily classroom practice and as a foundation for educational research — we developed the MathJSLab software (Lindau, 2026). This software simulates a subset of the MATLAB language and runs within a web browser environment.

Additionally, we are developing a theoretical synthesis grounded in Actor-Network Theory (ANT) (Latour, 2012) and Cultural-Historical Activity Theory (CHAT) (Cenci and Damiani, 2018; Leontiev, 2022; Vygotsky, 2007); the convergence of these theories is justified by the centrality of the concept of me-

diation in both frameworks. While Computational Thinking (CT) offers an instrumental and systematized framework for teaching, adopting these broader perspectives provides the epistemological, ontological, and semiotic grounding necessary to establish clear pedagogical guidelines and justify the work's comprehensive proposal. From this integrated perspective, we aim to address recurring criticisms of CT and problem-solving-focused instruction by consolidating a theoretical framework that supports a physics teaching methodology based on literal problem-solving — one in which the execution of manual calculations is delegated to computational mathematics tools.

The synthesis of ANT and CHAT characterizes MATLAB as an actant possessing multiple epistemological and ontological agencies within the sociotechnical network of mathematics. As an active mediator, software has reconfigured human activity and collective cognition, expanding the scope of tractable problems and stabilizing new scientific practices. This foundation allows for overcoming conceptual criticisms of Computational Thinking while guiding pedagogical approaches to physics education.

This paper presents MathJSLab as a teaching resource for the symbolic resolution of problems. The research is motivated by a shift in pedagogical emphasis: moving away from traditional manual numerical calculation, it proposes developing skills in coding numerical solutions based on symbolic ones. By delegating the arithmetic component to a computational tool, this approach aligns mathematics with the reality of contemporary research — which assigns the numerical execution of models to computational tools — thereby prioritizing the intelligibility of physical phenomena.

2 Methods

2.1 Research Methodology

This work is grounded in two axiomatic principles of constructivism—rooted in genetic epistemology and historical-cultural development theory—to structure a didactic methodology within the framework of constructionism.

The first axiom establishes that learning should foster the recapitulation of phylogenesis within the individual's ontogenesis. Based on this premise, didactic processes are designed to allow the learner to reproduce—analogously and through didactic transpositions—the historical trajectory of scientific knowledge production by the human collective. Aligned with the ideas of Piaget (2010), Bruner (2000) and Vygotsky (2007), this guideline informs a pedagogy based on the imitation, reconstruction, and recapitulation of practices and concepts in their historical and logical genesis.

The second axiom is grounded in Vygotsky's thesis regarding the internalization of cultural mediations. Cognitive development is understood as a process mediated by sign systems and tools (such as language, writing, and mathematics) which, upon being incorporated and internally reconstructed, transform elementary psychological functions into higher psychological functions. Such

mediators are not merely communicative channels but instruments that reorganize thought and expand intellectual capacity through social interaction and instruction within the Zone of Proximal Development (ZPD). Successful appropriation of these mediations and the resulting autonomous development are manifested in the student's ability to articulate and navigate between different equivalent representation systems (e.g., verbal, symbolic, and graphic).

Based on these axiomatic principles, we propose a theoretical-methodological articulation between Actor-Network Theory (ANT) and Cultural-Historical Activity Theory (CHAT). The objective is to provide a robust conceptual framework for planning and analyzing educational practices in the teaching of Physics, Mathematics, and Computing, focusing specifically on *Symbolic Problem-Solving*. This approach aims to integrate symbolic modeling and the use of computational tools as essential actants and mediators for the intelligibility of contemporary scientific phenomena.

CHAT posits that a person's learning and cognitive development are fundamentally shaped by social interactions and the surrounding culture, just as the human collective is shaped by mediations within social interaction and cultural processes. ANT emphasizes the agency of non-human actors in establishing stable sociotechnical networks. An amalgamation of these two theories to address issues in the epistemology of science, education, or cognitive theory appears not only feasible but highly appropriate. The convergence between ANT and CHAT centers on the premise that human action and interaction are always mediated by intermediate elements. However, the theories operate at distinct scales and with different objectives: mediation (CHAT) relates to the development of higher psychological functions (internalization), whereas translation (ANT) deals with the stabilization of sociotechnical networks and collectives (external to subjectivity).

Computational Thinking (CT) has established itself as a core 21st-century competency, extending beyond computer science to become broadly relevant for problem-solving across multiple domains. Its historical roots date back to the 1950s, although many of its fundamental ideas are significantly older. The term was formally used by Seymour Papert in 1980 (Papert, 1993) and again in 1996. These concepts build upon work developed by the author in collaboration with Cynthia Solomon in the 1971 article "Twenty things to do with a computer" (Papert and Solomon, 1972). Prior to the term's popularization, related notions were already circulating under different names (Tedre and Denning, 2016). The concept gained global prominence in the field of education in 2006, following work by Jeannette Wing (Wing, 2006), who defined CT as a universal basic skill and argued that it should be integrated into the education of all children, much like reading, writing, and arithmetic.

Despite its widespread acceptance, computational thinking (CT) faces criticism regarding its conceptual imprecision and the difficulty of distinguishing it from other forms of logical, scientific, or engineering reasoning (Jones, 2016; Tedre and Denning, 2016). Critics also point to a tendency to force computational solutions into inappropriate contexts or unrelated fields (Tedre and Denning, 2016). Furthermore, it is argued that an excessive emphasis on technical

and algorithmic problem-solving can obscure the social, ethical, and environmental implications inherent in the technologies being created (Easterbrook, 2014; Tedre and Denning, 2016). Added to this is the debate regarding the best strategy for curricular integration: while some advocate for cross-curricular integration within subjects such as Mathematics and Science (Barr and Stephenson, 2011; Grover and Pea, 2013), others argue that computational thinking should be taught as a distinct, standalone subject (Wolfram, 2010). Researchers also highlight the limitations of generalizing educational models and research primarily developed in the United States and Europe, questioning their effectiveness and suitability when transferred to other cultural contexts (Saqr et al., 2021; Tedre and Denning, 2016).

Computational Thinking (CT) is commonly expressed through four fundamental pillars which, within this integrated framework, cease to be isolated techniques and are instead understood as higher psychological functions resulting from the appropriation of historical artifacts:

1. *Decomposition*: equivalent to *depunctualization*[1] (the opening of "black boxes") to trace actants (ANT) and to the internalization of the collective division of labor (CHAT).

2. *Pattern Recognition*: identification of immutable inscriptions (ANT) and sedimented cultural signs that allow for the anticipation of network behaviors.

3. *Abstraction and Generalization*: translation process to create a "mandatory point of passage" (TAR) and movement towards the formal concept via ZPD (CHAT).

4. *Algorithms*: active mediators that transform meanings (ANT) and intellectual tools that reorganize the learner's thinking (CHAT).

This rationale addresses key criticisms of CT. Conceptual vagueness is overcome by defining CT as the construction of mediation networks geared toward execution by agents. Computational chauvinism is countered by Generalized Symmetry, which mandates that social and ethical implications be treated as relevant components of the network. Finally, the critique regarding the generalization of Western models is addressed by CHAT, which emphasizes that all development is historically and culturally situated.

Since even a sketch of this theoretical integration between ANT and CHAT provides a solid foundation for the pillars of CT — thereby overcoming its main

[1] From the perspective of ANT, *punctualization* refers to the process by which a complex, heterogeneous network of elements (human and non-human — such as technologies, people, institutions, and documents) is "gathered" or "translated" to act in unison, as if it were a single agent or a simple entity — or, for analytical purposes, a single point or "actor-network." When a network functions well and is punctualized, its internal components become invisible. A computer, for instance, is a punctualized network (chips, wires, electricity, software, users) that we treat simply as "the computer," effectively becoming a "black box." Punctualization is never permanent; it must be constantly maintained, otherwise the network unravels (*depunctualization*) and the black box "opens up."

criticisms — we can proceed to describe mathematics and MATLAB from the perspective of this theoretical amalgam, without claiming to exhaust the subject. We do so with the aim of articulating and better illustrating how the didactic activities we propose — involving literal problem-solving using MathJSLab — can be conceived and implemented.

A contemporary understanding of mathematics and its computational tools requires a perspective that transcends the dichotomy between the thinking subject and the object of thought. Viewed through the lenses of Actor-Network Theory (ANT) and Cultural-Historical Activity Theory (CHAT), mathematics and algorithmic code reveal themselves to be inseparable from the systems of representation and the technical and cultural means of computation that underpin them. These are not merely supports or auxiliary tools, but active non-human actors — co-constituents of a vast, heterogeneous sociotechnical network. Mathematics, a fundamental pillar of civilizational development, operates today as an invisible infrastructure supporting everything from complex financial operations to cutting-edge scientific production — the result of a centuries-long process of historical weaving.

Numerous studies from an educational perspective characterize mathematics in this way, even without explicitly stating the theoretical frameworks we propose:

> Thus, mathematics is rooted in physical concepts; for instance, it is embodied in science as a mediating element rather than serving as a mere attachment or addendum. Consequently, while mathematics — as a formal and technical tool — can be understood in other contexts, mathematics as a structuring skill of physics cannot be understood or applied independently (Pietrocola, 2002 apud Oliveira Júnior et al., 2021, p. 2, author's translation).

From the perspective of Actor-Network Theory (ANT), mathematics is a complex network in which agency is distributed among humans and non-humans — scribes, papyri and clay tablets, numerals, engineers, abacuses, astrolabes, builders, notations, scientists, axioms, algorithms, instruments, equations, theorems, merchants, navigators, astronomers, accountants, weights and measures, standardization committees, economists, curricula, teaching and research institutions, teachers, textbooks, students, symbolic conventions, programming languages, gear-based calculators, and silicon-chip processors, among a multitude of other agents aggregated into a unified collective populated by subjects and objects. This network is not limited to material objects; many of its most powerful actants are symbolic, such as axioms, theorems, and symbolic conventions. Mathematical practice only materializes through effective ways of recording and manipulating relations and patterns by consensus. Consequently, applications of mathematics throughout history have consistently been well-recorded; furthermore, as it began to be applied in the realm of science, it became even more thoroughly documented — in an absolutely formal manner (Badmus and Jita, 2024).

Such records, however, are not neutral. They constitute inscriptions (material or symbolic) that reconfigure the problems open to analysis, stabilize practices, and reorganize human cognition itself; for some of these records are cultural mediators — as established by CHAT — that become internalized, giving rise to higher psychological functions. In other words, these inscriptions become ontological and epistemological agents.

From the perspective of CHAT, these inscriptions act as psychological tools and signs that mediate human activity. While written text conveys meaning through a one-dimensional, phonetic channel, mathematical representations operate within multidimensional structures possessing limitless instrumental and operational aspects. The move toward increasing abstraction is not an exercise in distancing oneself from reality, but rather a strategy to aggregate powerful actants and secure a consensual, absolute grasp of the real world. The stabilization of this network depends intrinsically on formal education institutions, where curricula and pedagogies shape collective modes of thought, cumulatively consolidating "know-how" across generations.

The MATLAB (Matrix Laboratory) software exemplifies the materialization of this instrumental and operational nature of algorithmic code, emerging as an epistemological actant. It is marketed as a platform optimized for solving scientific and engineering problems, and its matrix-based language is described as the "most natural way" to express computational mathematics. (MathWorks, 2025). However, beyond its technical definition as a multi-paradigm interpreter, MATLAB must be understood as an ontological and epistemological agent. It intervenes in the constitution of the real by defining what can be calculated and mediated, acting as a central node that translates algorithmic knowledge into concrete action.

The origin of MATLAB does not lie in a sudden flash of genius but is the result of a dense series of prior inscriptions and heterogeneous associations. Its roots trace back to the work of J. H. Wilkinson and other researchers in the 1960s and 1970s, whose algorithms for eigenvalues and linear equations were translated into a textbook (Wilkinson and Reinsch, 1971) and into Fortran libraries such as EISPACK and LINPACK[2] at *Argonne National Laboratory* (Haigh, 2008; Moler, 2004, 2018; Moler and Little, 2020). In this sense, MATLAB emerged from an aggregation of human actors who produced equations and papers; these were subsequently condensed into manuals and libraries that — upon being translated into executable code — became protagonists within the network in their own right. The birth of the first MATLAB, coded by Cleve Moler in 1976 at the University of New Mexico (Haigh, 2008), was an act of translation. Moler — a developer of the LINPACK library who was teaching Linear Algebra and Numerical Analysis at the University of New Mexico — sought to create a way for his students to utilize the capabilities of the EIS-

[2]EISPACK and LINPACK are Fortran libraries for linear algebra developed at *Argonne National Laboratory*: EISPACK focuses on the computation of eigenvalues and eigenvectors, while LINPACK solves linear equations and least-squares problems. Both were succeeded by LAPACK. Available at:`https://www.netlib.org/eispack/`, `https://www.netlib.org/linpack/` e `https://www.netlib.org/lapack/`.

PACK and LINPACK libraries without needing to write code or execute the full program-building cycle: editing, compiling, loading, and running. The original MATLAB was, therefore, a mediator that simplified the interaction between the student (subject) and computational linear algebra (object/activity). Since then, it has acted to mediate, translate, and stabilize inscriptions within the sociotechnical network of mathematics.

Throughout the 1980s, Moler's collaboration with John Little and Steve Bangert — aimed at reprogramming the software in C and launching it commercially via MathWorks in 1984 — marked a process of punctualization. The software evolved from a free academic tool into a stable "black box" — a globally recognized standard engineering tool. Its logo, a plot representing the eigenvalue problem of an L-shaped membrane, serves as a signifier referencing its memorable origins and the network of algorithms by Wilkinson and Forsythe that preceded it.

The integration of mathematics and MATLAB, viewed through the frameworks of Actor-Network Theory (ANT) and the Theory of Actor-Human-Computer (CHAT), reveals that intelligence and problem-solving capabilities reside not merely in the human brain but in the performance of the sociotechnical network. MATLAB's applications span fields such as "active automotive safety systems, interplanetary spacecraft, health monitoring devices, smart grids and Long-Term Evolution (LTE) cellular networks, machine learning, signal processing, image processing, computer vision, communications, computational finance, control design, and robotics" (MathWorks, 2025). Consequently, MATLAB cannot be viewed merely as an instrument; it is an actant that fosters new forms of agency. It reorganizes human activity by enabling the intuitive application of linear algebra operations to multidimensional matrices, acting as an effective agent that executes instructions and materializes computational thinking.

In this context, mathematics and software serve as mediators that shape our perception of the world. The stability of these actants is ensured by their continuous integration into curricula and professional practice, where humans and non-humans interact in a state of generalized symmetry. Synthesizing these theories allows us to recognize that, by using MATLAB, we align ourselves with centuries of mathematical history, stabilized code libraries, and educational institutions. We thus form a collective in which knowing, doing, and know-how are continuously distributed and performed to shape and comprehend reality, increasingly establishing a symmetry between the human and the algorithm.

2.2 The MathJSLab

MathJSLab was initially developed in JavaScript[3] and later evolved into TypeScript[4]. The initial development goal was to create a calculator capable of defining variables and evaluating mathematical expressions using those variables, allowing it to be embedded in a web page. Early versions relied solely on character substitution within a string (the input mathematical expression) followed by the execution of the `eval`[5] function on the resulting string. There was no mechanism to parse the input expression string, nor was there proper mathematical notation for displaying the input and the calculated result.

Currently, MathJSLab uses ANTLR[6] to generate the lexical and syntactic analyzer for its interpreter and employs MathML[7] to render expressions in standard mathematical notation. The initial concept was to create educational software that runs in a web browser and helps users practice coding mathematical expressions in a programming language. Consequently, solving physics problems symbolically emerged as the ideal application for the software, allowing for the development of coding skills within the context of physics education. The name "MathJSLab" alludes to the concept of a mathematical laboratory built in JavaScript.

The software was released with a strategy ensuring full availability, support for collaborative development, frequent maintenance, and the assignment of an ISBN and DOI to every released version. This allows it to be properly cited in any research work that uses it as a research tool or as a subject of study in the field of education. Furthermore, it is released under a permissive free software license — the MIT License[8] — and runs on any internet-connected device, while also supporting offline use[9]. The software was developed using modern web technologies[10] to ensure ease of maintenance and future extension. The entire application, including help documentation for each command, is available in three languages: English, Portuguese, and Spanish. All documentation was

[3] JavaScript implements the ECMAScript standard (the name of the language standardized by ECMA-262), defining its syntax and basic types, and adding environment APIs (DOM in the browser, file system in Node.js); Specification: `https://262.ecma-international.org/16.0/index.html`.

[4] TypeScript is a strongly typed superset of the JavaScript language that compiles to JavaScript. Specification: `https://www.typescriptlang.org/`.

[5] The `eval` function executes a string containing JavaScript commands, following the concept of *bootstrap*: a function that executes commands of the language itself.

[6] ANTLR is a parser generator that, based on a grammar specification, produces parsers for reading, processing, or translating text or bits. Available at: `https://www.antlr.org/`.

[7] MathML is a markup language for describing mathematical notation, capturing its structure and content, with the aim of enabling the use of mathematics on the World Wide Web. Specification: `https://www.w3.org/TR/mathml4/`.

[8] The MIT License is an extremely permissive and popular open-source software license that originated at the Massachusetts Institute of Technology (MIT). It allows anyone to use, copy, modify, distribute, sublicense, and sell the software with few restrictions, requiring only that the original copyright notice be retained. Available at: `https://opensource.org/license/mit`

[9] It is a Progressive Web App.

[10] HTML5, SCSS and Nunjucks templates, Markdown documentation, and Web Components.

written in Markdown format[11].

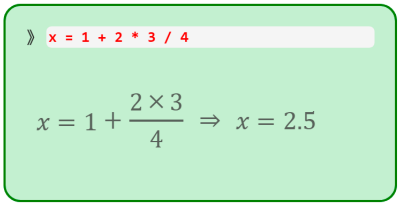

Figure 1: MathJSLab command prompt, showing the translation and solution of the code, displayed in standard mathematical notation.

The MathJSLab prompt, shown in Figure 1, differs from MATLAB's approach by being designed around the second axiom we adopted — regarding the translation of equivalent representations. The prompt accepts a command in code format and immediately displays the expression below it using standard mathematical notation, similar to the way one writes with paper and pencil. Thus, simply observing the result of each prompt serves as an exercise, training the user to translate code into standard mathematical notation.

To demonstrate the complete solution to a problem within the application environment, we present a sequence of prompts that solve a quadratic equation using Bhaskara's formula (Figure 2). Each prompt entered as code is interpreted and computed, with both the input and the result displayed in standard mathematical notation.

[11] Markdown is a lightweight markup language for formatting plain text using intuitive symbols and converting it into HTML or other formats. It was designed to be readable as plain text and easy to write, and is widely used in documentation, websites, and project READMEs. See `https://www.markdownguide.org/`.

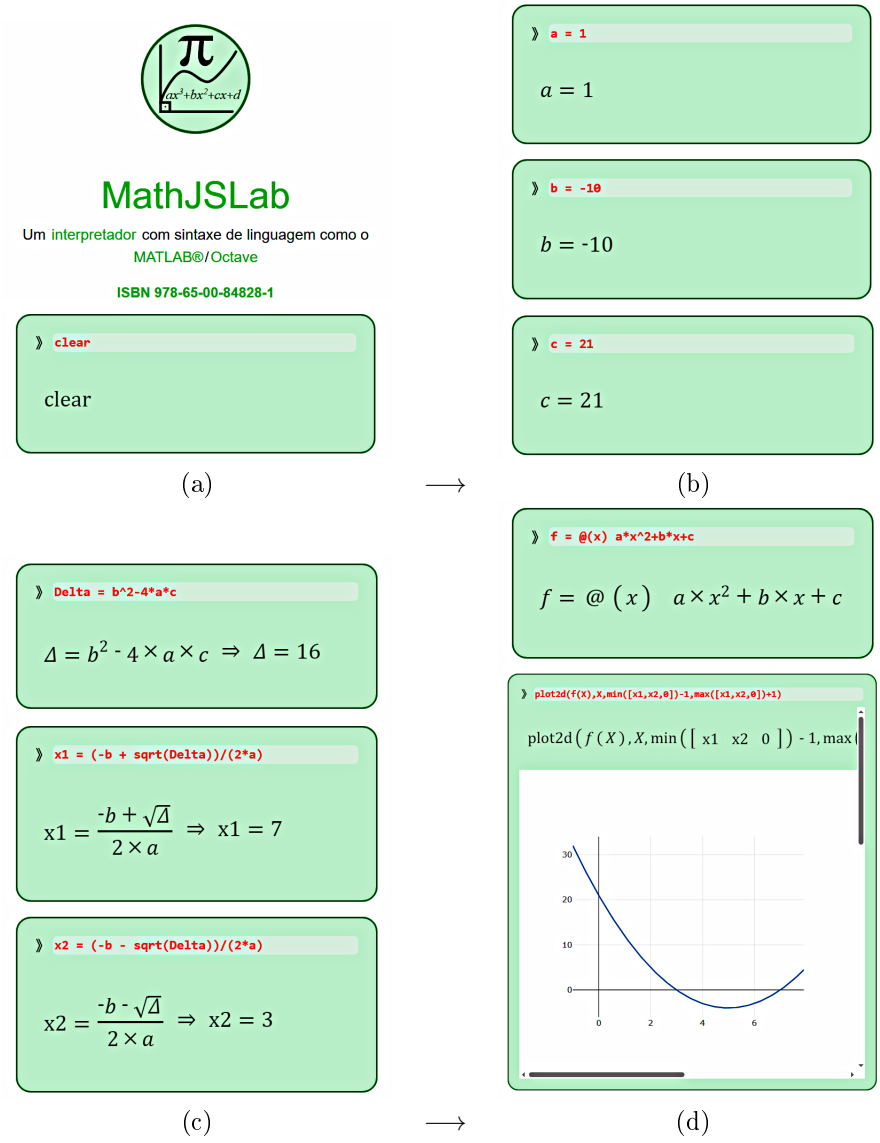

Figure 2: MathJSLab screens showing the solution of a quadratic equation: (a) starting by clearing all variables; (b) defining the equation's coefficients; (c) solving using the quadratic formula; (d) defining the function and plotting the graph.

3 Results and Discussion

3.1 Examples of classroom application at the high school level: symbolic problem-solving

Para demonstrar atividades didáticas de resolução literal de problemas no ensino de Física usando o MathJSLab, mostramos dois exemplos.

O primeiro exemplo é um problema do ENEM, onde é necessário solucionar o problema num tempo minimamente viável e justifica-se por mostrar toda a potencialidade dos métodos e recursos aqui apresentados.

O segundo exemplo é um problema análogo ao resolvido em experimentos com plano inclinado. Este exemplo justifica-se pela relevância do problema como actante e mediador cultural, e por exemplificar também as transposições didáticas sem limites que podemos fazer quando os objetos são pensados como actantes e mediadores socioculturais.

3.1.1 An ENEM test problem

O Exame Nacional do Ensino Médio (ENEM), com grande número de questões em cada caderno (90), tem tempo bastante limitado por questão para executar sua solução. Questões que envolvem a manipulação de duas ou mais equações exigem a resolução literal para que o tempo de solução da questão seja minimamente viável. Um bom exemplo de questão assim é a do ENEM de 2015, que está na Figura 3, que trata de um carro movido por energia solar, calculando o tempo que leva para atingir determinada velocidade.

A solução do problema consiste em equacionar a potência efetiva entregue pelo painel solar em termos dos dados fornecidos e considerar que essa energia é transformada em energia cinética, calculando o tempo total através da definição de potência como a taxa de variação da energia em relação ao tempo. Em resumo: a solução literal é o melhor caminho.

O código para a solução da questão no MathJSLab está na Figura 4. É uma sequência de definições e cálculo de expressões baseada na resolução literal do problema.

Um carro solar é um veículo que utiliza apenas a energia solar para a sua locomoção. Tipicamente, o carro contém um painel fotovoltaico que converte a energia do Sol em energia elétrica que, por sua vez, alimenta um motor elétrico. A imagem mostra o carro solar Tokai Challenger, desenvolvido na Universidade de Tokai, no Japão, e que venceu o World Solar Challenge de 2009, uma corrida internacional de carros solares, tendo atingido uma velocidade média acima de 100 km/h.

Disponível em: www.physics.hku.hk. Acesso em: 3 jun. 2015.

Considere uma região plana onde a insolação (energia solar por unidade de tempo e de área que chega à superfície da Terra) seja de 1 000 W/m^2, que o carro solar possua massa de 200 kg e seja construído de forma que o painel fotovoltaico em seu topo tenha uma área de 9,0 m^2 e rendimento de 30%.

Desprezando as forças de resistência do ar, o tempo que esse carro solar levaria, a partir do repouso, para atingir a velocidade de 108 km/h é um valor mais próximo de

A 1,0 s.
B 4,0 s.
C 10 s.
D 33 s.
E 300 s.

Figure 3: Questão de física do ENEM 2015: desempenho de um carro solar.

expression	**comment**
`D_P = 1000`	`% densidade de potência da radiação solar`
`m = 200`	`% massa do carro`
`A = 9`	`% área do painel solar`
`eta = 0.30`	`% rendimento do painel solar`
`Pe = D_P * A * eta`	`% potência efetiva do painel`
`km = 1000`	`% quilômetro em metros`
`h = 60*60`	`% hora em segundos`
`m_s = km/h`	`% fator de conversão km/h em m/s`
`v = 108 * m_s`	`% velocidade em m/s`
`Ec = m * v^2 / 2`	`% energia cinética`
`Delta_t = Ec / Pe`	`% tempo para adquirir a energia cinética`

Figure 4: Código da solução da questão do ENEM 2015: desempenho de um carro solar.

3.1.2 Free fall

An analysis of Galileo's inclined plane through the lens of Actor-Network Theory (ANT) reveals that this artifact transcends the status of a passive support, acting instead as a central actant within heterogeneous networks of scientific production. By slowing down motion and translating free fall into measurable sequences, the device reconfigures the physical phenomenon, enabling the stabilization of statements grounded in mathematical regularity. This material translation not only facilitates observation but also enacts new ontologies of motion by articulating dimensions of time, space, and number. Indeed, it was through Galileo, the inclined plane, and the laws of motion that mathematics became stably and irreversibly incorporated into the description of physical phenomena, spreading throughout and permeating the entire sociotechnical network of knowledge — as Pietrocola (2002) describes concisely and with the necessary emphasis:

> It was with the advent of modern science in the 17th century — with Galileo and others — that natural phenomena began to be systematically expressed through mathematical relations, a practice that came to serve as a criterion for scientific validity. This practice represents a legacy of the Pythagorean tradition (author's translation).

Within the framework of CHAT, the inclined plane is understood as a mediating tool that reorganizes the cognitive structure of investigative activity by mediating the relationship between the subject and accelerated motion. It links practical action to complex sign systems, situating the individual within historically established cultural practices. Its continued presence in educational laboratories reaffirms its agency as a historical actant, connecting students to digital sensors and pedagogical traditions.

As a material operator of intelligibility, the artifact remains fundamental to the constitution of physics and its ongoing reinscription within contemporary educational contexts.

Adapting inclined plane experiments to our proposed methodology would involve presenting free-fall problems to be solved using MathJSLab. This would be preceded by a presentation on deriving free-fall velocity—based on the conservation of energy—and unit conversion, as illustrated in Figure 5; in other words, a symbolic solution to the free-fall problem.

For this derivation, we have access to concepts — specifically energy and the principle of its conservation — that were unavailable to Galileo. The code (9 lines) for calculating free-fall velocity in km/h is shown in the Figure 6.

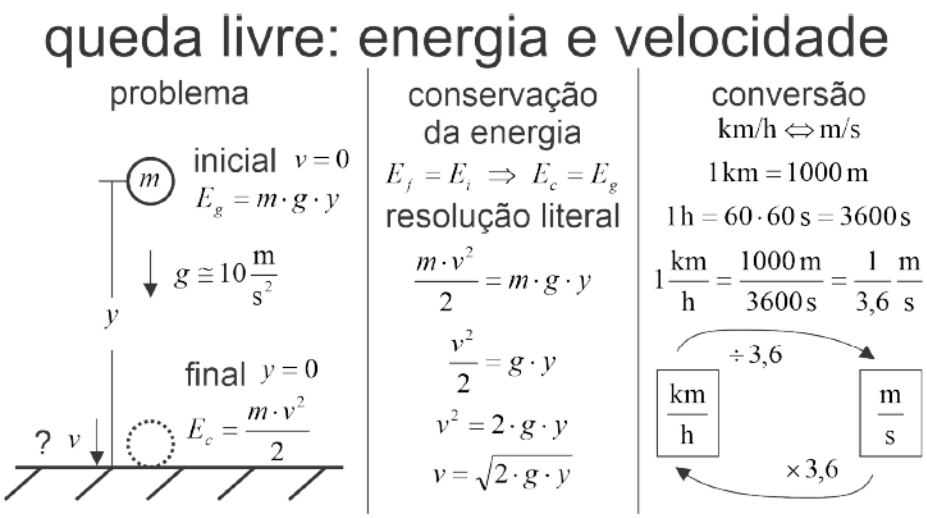

Figure 5: A scheme for deducing the velocity of free fall, based on the conservation of energy, for implementation in code applications.

expressão	**comentário**
`y = 3000`	`% definition of free-fall height in meters`
`g = 10`	`% definition of gravitational acceleration in m/s^2`
`v = sqrt(2*g*y)`	`% calculation of free-fall velocity in m/s`
`km = 1000`	`% kilometer to meters`
`h = 60*60`	`% hour to seconds`
`m_s = km/h`	`% conversion factor from km/h to m/s`
`km_h = 1/m_s`	`% conversion factor from m/s to km/h`
`v_m_s = v`	`% free-fall velocity in m/s`
`v_km_h = v * km_h`	`% free-fall velocity in km/h`

Figure 6: Code for calculating free-fall velocity.

This example runs counter to the first axiom we adopted — which relies on the principle of conservation of energy — thereby contradicting the problem's historical and logical genesis. We justify this approach because our goal is to practice numerical solution coding skills, while acknowledging that a derivation made in this manner "breaks" the problem's historical genesis by invoking the principle of conservation of energy.

It should be clarified that the two axiomatic principles we adopted serve to support and justify the theoretical synthesis we have outlined, but they should not be regarded as rigid rules of this methodology. The analytical solution, the use of a numerical computing tool like MathJSLab, and the second axiom — linked to the aim of practicing coding skills — ensure that this example remains aligned with the proposed methodology.

4 Conclusion

In this work, we aim to initiate a theoretical convergence between ANT (Actor-Network Theory) and CHAT (Cultural-Historical Activity Theory), with the goal of outlining a physics teaching methodology based on Literal Problem Solving. Although we only sketch this theoretical convergence without reaching a definitive synthesis, we identify consistent axiomatic guidelines derived from genetic epistemology and developmental psychology. These guidelines are indeed evident and definitive, as well as perfectly aligned with the methodology we propose.

We propose an innovative theoretical foundation for Computational Thinking, transcending purely technical perspectives by integrating it into a sociotechnical synthesis of ANT and CHAT. This approach redefines the discipline's four classic pillars not merely as cognitive skills, but as processes involving the internalization of cultural mediations and the construction of sociotechnical networks linking humans and non-humans.

We conclude by providing a didactic tool to implement our proposed methodology: a mature software application called MathJSLab. It is freely available, supports collaborative development, and is accessible via the web.

Future developments will enable the software to generate notes containing equations, graphs, figures, diagrams, flowcharts, and more — all easily edited in Markdown format — allowing students and teachers to effortlessly create high-quality notes. Work is already underway to adapt the software for release on online app stores.

References

Aho, Alfred V. (Jan. 2011). "Computation and Computational Thinking". In: *Ubiquity* 2011.January. ISSN: 1530-2180. DOI: 10.1145/1922681.1922682.

Badmus, O. T. and L. C. Jita (Mar. 2024). "Physics Difficulty and Problem-Solving: Exploring the Role of Mathematics and Mathematical Symbols". In: *Interdisciplinary Journal of Education Research* 6, pp. 1–14. ISSN: 2710-2114. DOI: 10.38140/ijer-2024.vol6.08.

Barr, Valerie and Chris Stephenson (2011). "Bringing Computational Thinking to K–12: What Is Involved and What Is the Role of the Computer Science Education Community?" In: *ACM Inroads* 2.1, pp. 48–54. ISSN: 2153-2184. DOI: 10.1145/1929887.1929905.

Bruner, J. S. (2000). *Sobre a Teoria da Instrução*. 1st ed. São Paulo: Phorte. 174 pp. ISBN: 9788599860052.

Cenci, A. and M. F. Damiani (Dec. 2018). "Desenvolvimento da Teoria Histórico-Cultural da Atividade em Três Gerações: Vygotsky, Leontiev e Engeström". In: *Roteiro* 43.3, pp. 919–948. ISSN: 0104-4311. DOI: 10.18593/r.v43i3.16594.

Easterbrook, Steve (2014). "From Computational Thinking to Systems Thinking: A Conceptual Toolkit for Sustainability Computing". In: *Proceedings of the 2014 Conference ICT for Sustainability*. Vol. 2. ict4s-14. Atlantis Press. ISBN: 9789462520226. DOI: 10.2991/ict4s-14.2014.28.

Grover, Shuchi and Roy Pea (Jan. 2013). "Computational Thinking in K–12: A Review of the State of the Field". In: *Educational Researcher* 42.1, pp. 38–43. ISSN: 1935-102X. DOI: 10.3102/0013189X12463051.

Haigh, T. (Jan. 2008). "Cleve Moler: Mathematical Software Pioneer and Creator of MATLAB". In: *IEEE Annals of the History of Computing* 30.1, pp. 87–91. ISSN: 1934-1547. DOI: 10.1109/MAHC.2008.2.

Ince, E. (May 2018). "An Overview of Problem Solving Studies in Physics Education". In: *Journal of Education and Learning* 7.4, pp. 191–200. ISSN: 1927-5250. DOI: 10.5539/jel.v7n4p191.

Jones, Elizabeth (2016). *The Trouble with Computational Thinking*. URL: https://www.scienceall.com/com/file/filedown?_ci=82348&_ck=8f36b4cab44611ee91d7f220ef63aea9 (visited on 09/22/2025).

Karam, R. A. S. and M. Pietrocola (2009). "Habilidades Técnicas Versus Habilidades Estruturantes: Resolução de Problemas e o Papel da Matemática como Estruturante do Pensamento Físico". In: *Alexandria: Revista de Educação em Ciência e Tecnologia* 2.2, pp. 181–205. URL: https://periodicos.ufsc.br/index.php/alexandria/article/view/37960 (visited on 08/13/2025).

Latour, Bruno (2012). *Reassembling the Social: An Introduction to Actor-Network-Theory. An introduction to Actor-Network-Theory*. Reprint. Clarendon lectures in management studies. Includes bibliographical references. - Orig. publ.: 2005. Oxford [u.a.]: Oxford University Press. 301 pp. ISBN: 9780199256051.

Leontiev, A. N. (2022). *Atividade, Consciência e Personalidade*. Trans. by Priscila Marques. Bauru: Mireveja Editora. 256 pp. ISBN: 9786586638165.

Lindau, Sérgio (2026). *MathJSLab*. Version 1.9.2. Concept DOI: 10.5281/zenodo.8396265. DOI: 10.5281/zenodo.8396263. URL: https://mathjslab.com (visited on 01/31/2026).

MathWorks (2025). *MATLAB Product Description*. URL: https://www.mathworks.com/help/matlab/learn_matlab/product-description.html (visited on 05/22/2025).

Moler, Cleve (2004). *The Origins of MATLAB*. URL: https://www.mathworks.com/company/technical-articles/the-origins-of-matlab.html (visited on 05/22/2025).

— (2018). *A Brief History of MATLAB*. URL: https://www.mathworks.com/company/technical-articles/a-brief-history-of-matlab.html (visited on 05/22/2025).

Moler, Cleve and Jack Little (June 2020). "A History of MATLAB". In: *Proceedings of the ACM on Programming Languages* 4.HOPL, pp. 1–67. ISSN: 2475-1421. DOI: 10.1145/3386331.

Oliveira, V. F. de, I. S. Araujo, and E. A. Veit (Mar. 2017). "Resolução de Problemas Abertos no Ensino de Física: Uma Revisão da Literatura". In: *Revista Brasileira de Ensino de Física* 39.3, e3402. ISSN: 1806-1117. DOI: 10.1590/1806-9126-RBEF-2016-0269.

Oliveira Júnior, V. F. de et al. (June 2021). "A Matemática como Habilidade Estruturante no Contexto Pedagógico, na Experimentação e na Modelagem em Física". In: *REMAT: Revista Eletrônica da Matemática* 7.1, e2011. ISSN: 2447-2689. DOI: 10.35819/remat2021v7i1id4825.

Papert, Seymour (1993). *Mindstorms: Children, Computers, and Powerful Ideas*. 2nd ed. New York: Basic Books. 252 pp. ISBN: 9780465046744.

Papert, Seymour and Cynthia Solomon (1972). "Twenty Things to Do with a Computer". In: *Educational Technology Magazine*. ISSN: 0013-1962. URL: https://www.stager.org/articles/twentythings.pdf (visited on 01/12/2026).

Pérez, D. G. et al. (1992). "Questionando a Didática de Resolução de Problemas: Elaboração de um Modelo Alternativo". In: *Caderno Brasileiro de Ensino de Física* 9.1, pp. 7–19. URL: https://periodicos.ufsc.br/index.php/fisica/article/view/7501 (visited on 06/14/2025).

Piaget, Jean (2010). *A Formação do Símbolo na Criança: Imitação, Jogo e Sonho, Imagem e Representação*. 4th ed. Rio de Janeiro: LTC. 376 pp. ISBN: 9788521617617.

Pietrocola, M. (2002). "A Matemática como Estruturante do Conhecimento Físico". In: *Caderno Brasileiro de Ensino de Física* 19.1, pp. 93–114. URL: https://periodicos.ufsc.br/index.php/fisica/article/view/9297 (visited on 06/14/2025).

Saqr, Mohammed et al. (Mar. 2021). "People, Ideas, Milestones: A Scientometric Study of Computational Thinking". In: *ACM Transactions on Computing Education* 21.3, 20:1–20:17. ISSN: 1946-6226. DOI: 10.1145/3445984.

Tedre, Matti and Peter Denning (Nov. 2016). "The Long Quest for Computational Thinking". In: *Proceedings of the 16th Koli Calling Conference on Computing Education Research*. Koli Calling 2016. ACM, pp. 120–129. DOI: 10.1145/2999541.2999542.

Vygotsky, L. S. (2007). *A Formação Social da Mente: O Desenvolvimento dos Processos Psicológicos Superiores*. 7th ed. São Paulo: Martins Fontes. 224 pp. ISBN: 9788533622647.

Wilkinson, J. H. and C. Reinsch (1971). *Handbook for Automatic Computation. Volume II: Linear Algebra*. Ed. by Christian Reinsch et al. 1st ed. Vol. 186. Grundlehren der Mathematischen Wissenschaften. Reprinted 2014. Berlin and Heidelberg: Springer-Verlag. 439 pp. ISBN: 9783642869426. DOI: 10.1007/978-3-642-86940-2.

Wing, Jeannette M. (Mar. 2006). "Computational Thinking". In: *Communications of the ACM* 49.3, pp. 33–35. ISSN: 1557-7317. DOI: 10.1145/1118178.1118215.

Wolfram, Conrad (2010). *Computational Thinking Is the Code to Success*. URL: https://www.tes.com/magazine/archived/computational-thinking-code-success (visited on 02/14/2026).