MathJSLab: Prova de conceito de computação científica no navegador e software como artefato acadêmico

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

4 de junho de 2026

Resumo

O avanço das tecnologias Web nas últimas décadas ampliou significativamente as capacidades computacionais dos navegadores, permitindo que estas ferramentas, originalmente voltadas à navegação hipertextual, executassem aplicações de elevada complexidade, incluindo ambientes de computação matemática e científica. Este trabalho analisa o MathJSLab como uma prova de conceito de computação científica executada integralmente no navegador e como exemplo de software concebido enquanto artefato acadêmico publicável. A pesquisa possui caráter qualitativo e exploratório, articulando fundamentos tecnológicos da computação Web contemporânea com uma discussão epistemológica acerca do software como produção científica. São discutidos aspectos relacionados à evolução das engines JavaScript, aos mecanismos de compilação Just-In-Time (JIT), à utilização do ANTLR como ferramenta de construção de linguagens e ao suporte ao MathML para representação matemática em navegadores. Paralelamente, o trabalho aborda o papel epistemológico do software científico educativo, livre e publicável, considerando aspectos relacionados à citabilidade, reprodutibilidade, preservação digital e ciência aberta. Conclui-se que o MathJSLab evidencia não apenas a viabilidade técnica da computação científica em ambiente Web, mas também transformações nas formas de mediação, circulação e produção do conhecimento científico mediado por software livre e interativo.

Palavras-chave: Computação científica Web. Software científico. Software educativo. Artefato acadêmico. Ciência aberta.

1 INTRODUÇÃO

A expansão das tecnologias digitais modificou profundamente os processos de produção, circulação e legitimação do conhecimento científico, elevando o software de uma posição estritamente instrumental para a de elemento central na

modelagem, simulação e nos processos de mediação da atividade científica de uma forma geral. Essa transformação possui implicações epistemológicas, uma vez que o código executável passa a participar ativamente da construção e disseminação do saber acadêmico.

Historicamente, a computação científica esteve restrita a ambientes locais especializados, como MATLAB e Octave (Moler 2004; Moler 2018; Moler e Little 2020; GNU Octave Project 2026), que consolidaram o código interpretado como linguagem legítima de modelagem, mas mantiveram a dependência de instalações específicas em sistemas operacionais hospedeiros. Paralelamente, a evolução das tecnologias Web, impulsionada por engines JavaScript modernas (Google V8 Project 2026), mecanismos de compilação JIT (Mozilla Developer Network 2025a), ferramentas de construção de linguagens (geradores de analisadores léxicos e sintáticos) como o ANTLR[1] (ANTLR 2026; Parr 2013) e padrões de representação como o MathML (World Wide Web Consortium (W3C) 2023; Mozilla Developer Network 2025b), permitiu que o navegador transcendesse o simples acesso à informação e constituísse um ambiente de experimentação e modelagem científica interativa. De uma maneira geral, aplicações isomórficas e multiplataforma podem ser feitas em JavaScript, que já possui um rico ecossistema de bibliotecas, em vários registros e bancos de repositórios (Abramyan 2020).

Essa transformação técnica produz também mudanças relevantes nas formas contemporâneas de mediação nos processos de produção do conhecimento científico. O navegador deixa de atuar apenas como interface de acesso à informação e passa a integrar práticas de experimentação, modelagem e interação científica de forma ampla. Nesse contexto, ferramentas executáveis diretamente no navegador podem assumir simultaneamente funções de laboratório computacional, ambiente educacional interativo e objeto de circulação acadêmica.

Além da dimensão tecnológica, observa-se também uma mudança na forma como o software é compreendido no meio acadêmico. Tradicionalmente tratado como instrumento auxiliar de pesquisa, o software científico passa gradualmente a ser reconhecido como produção intelectual autônoma (Katz et al. 2021), passível de publicação, identificação persistente, citação e preservação digital. Nesse cenário, softwares educativos e de código aberto assumem papel relevante por favorecerem a reprodutibilidade, a transparência e o compartilhamento do conhecimento científico (Flach e Kon 2021). A adoção de identificadores persistentes, como DOI e ISBN, amplia ainda mais sua inserção no ecossistema científico, consolidando esses programas como artefatos reconhecidos e preserváveis no âmbito da produção científica (Katz et al. 2021).

Diante desse cenário, o presente trabalho parte do seguinte problema de pesquisa: em que medida um software educativo de computação matemática

[1] ANTLR (sigla de ANother Tool for Language Recognition) foi criado por Terence Parr no final dos anos 1980 (Casperson 2016), num trocadilho com Yacc ("Yet Another Compiler-Compiler"). Em vez de utilizar ferramentas separadas para análise léxica e sintática, o ANTLR unificou estas definições em um só arquivo de gramática (utilizando parsing LL(*)), gerando automaticamente árvores sintáticas e tornando-se reconhecidamente "mais poderoso" que as abordagens clássicas como Lex/Yacc (Ortin et al. 2022).

executado integralmente no navegador pode constituir simultaneamente uma prova de conceito tecnológica e um artefato acadêmico publicável? Para responder a essa questão, toma-se como objeto de estudo o MathJSLab (Lindau 2026), software livre licenciado sob a licença MIT, desenvolvido para execução de um subconjunto da linguagem MATLAB/Octave diretamente no navegador, sem partes executáveis no servidor.

O objetivo geral consiste em analisar o MathJSLab como prova de conceito técnica e como modelo de software enquanto artefato acadêmico publicável. Especificamente, busca-se apresentar os fundamentos tecnológicos que viabilizam aplicações complexas em navegadores modernos e discutir os aspectos epistemológicos relacionados à publicação e à citabilidade de software científico educativo.

A relevância deste estudo fundamenta-se na articulação entre dimensões tecnológicas, educacionais e epistemológicas. Do ponto de vista tecnológico, a ampliação das capacidades computacionais dos navegadores Web cria novas possibilidades para o desenvolvimento de aplicações científicas acessíveis e multiplataforma. Na dimensão educacional, ferramentas interativas executadas diretamente no navegador ampliam as possibilidades de experimentação computacional em ambientes digitais de aprendizagem. Já sob a perspectiva epistemológica, o reconhecimento do software como produção científica legítima fortalece práticas relacionadas à ciência aberta, à reprodutibilidade e à preservação do conhecimento mediado por este tipo de artefato. Ao investigar o MathJSLab, este trabalho busca contribuir para as discussões sobre democratização do acesso a ferramentas científicas, valorização da ciência aberta e fortalecimento da reprodutibilidade computacional.

2 DESENVOLVIMENTO

Esta seção discute os fundamentos tecnológicos, educacionais e epistemológicos relacionados à computação científica em ambiente Web, articulando-os à análise do MathJSLab como prova de conceito e como artefato acadêmico publicável. Parte-se do entendimento de que as transformações recentes observadas nos navegadores Web não representam apenas uma evolução técnica das plataformas computacionais contemporâneas, mas também uma mudança nas formas de mediação, circulação e produção do conhecimento científico por meio do software.

Inicialmente, são apresentados os antecedentes históricos da computação científica e a consolidação de ambientes como MATLAB e Octave, responsáveis por legitimar o código executável como instrumento de modelagem, experimentação e produção científica. Em seguida, analisam-se as transformações tecnológicas que ampliaram as capacidades computacionais dos navegadores modernos, incluindo mecanismos de compilação *Just-In-Time* (JIT) nos *engines* JavaScript e tecnologias voltadas à representação matemática em ambiente Web, como o MathML.

Posteriormente, discute-se o software científico como artefato acadêmico e objeto de mediação epistemológica, abordando aspectos relacionados à reprodutibilidade, ciência aberta, preservação digital e citabilidade em trabalhos aca-

dêmicos. Nesse contexto, o caráter educativo e livre do MathJSLab assume papel relevante em razão de seu potencial de utilização em ambientes de ensino, pesquisa e experimentação computacional.

Por fim, o MathJSLab é analisado como prova de conceito de computação científica executada integralmente no navegador Web, buscando evidenciar sua viabilidade técnica e suas implicações epistemológicas e educacionais. A partir dessa análise, discute-se de que maneira softwares científicos executáveis, interativos e academicamente publicáveis podem contribuir para novas formas de produção, compartilhamento e preservação do conhecimento científico em ambientes digitais.

2.1 COMPUTAÇÃO CIENTÍFICA E A CONSOLIDAÇÃO DO NAVEGADOR COMO AMBIENTE COMPUTACIONAL

A computação científica contemporânea fundamenta-se na evolução da computação numérica de alto desempenho, consolidada a partir da linguagem Fortran nas décadas de 1950 e 1960 (BACKUS, 1998). Mais do que um instrumento técnico voltado à engenharia, o Fortran inaugurou um paradigma onde o código executável passou a ocupar posição central na prática científica, deixando de atuar apenas como mecanismo operacional para tornar-se meio de experimentação, simulação e produção de conhecimento.

Esse desenvolvimento foi impulsionado pela criação de bibliotecas especializadas de álgebra linear, como EISPACK e LINPACK nos anos 1970 (DONGARRA; LUSZCZEK; PETITET, 2003), posteriormente consolidadas nas bibliotecas BLAS e LAPACK. Tais iniciativas não apenas padronizaram algoritmos fundamentais, mas também consolidaram uma infraestrutura computacional para a pesquisa científica. Matrizes, vetores e rotinas matemáticas deixaram de ser abstrações exclusivamente teóricas para se tornarem entidades explícitas e operacionais na prática científica mediada por software.

Nesse contexto, o MATLAB, concebido originalmente por Cleve Moler como uma interface de alto nível para as bibliotecas LINPACK/EISPACK (MOLER, 2004, 2018; MOLER; LITTLE, 2020), consolidou-se como um dos principais ambientes de modelagem matemática e computação científica. Ao legitimar o uso de linguagens interpretadas e scripts (.m), o MATLAB aproximou modelagem matemática e execução computacional em um mesmo espaço de investigação. Complementarmente, o surgimento do GNU Octave como alternativa de software livre ampliou o acesso a esse paradigma computacional (GNU OCTAVE PROJECT, 2026), reforçando a concepção de ambientes científicos como plataformas abertas de produção e compartilhamento do conhecimento.

A trajetória que conecta Fortran a ambientes como MATLAB e Octave evidencia, portanto, a consolidação de uma cultura científica baseada em linguagens executáveis e manipulação algébrica interativa. Essa continuidade histórica permite compreender o navegador Web não como uma ruptura em relação à computação científica tradicional, mas como uma nova etapa evolutiva vol-

tada à ampliação da acessibilidade, distribuição e circulação do conhecimento científico mediado por software.

Originalmente concebidos para navegação hipertextual e exibição de documentos, os navegadores Web evoluíram progressivamente de plataformas documentais passivas para ambientes computacionais ativos e de elevada complexidade. Essa transformação foi impulsionada pela crescente demanda por interfaces dinâmicas e aplicações interativas, exigindo mecanismos capazes de transcender a simples renderização de páginas e a execução de scripts elementares.

O principal fator dessa transformação foi a evolução das engines JavaScript (MOZILLA DEVELOPER NETWORK, 2025a), especialmente da V8 Engine, responsável pela introdução de otimizações dinâmicas, gerenciamento avançado de memória e mecanismos modernos de execução. Essas inovações permitiram que o navegador assumisse funções anteriormente restritas a softwares instalados localmente, viabilizando a execução de aplicações significativamente mais robustas no ambiente Web.

A viabilidade computacional contemporânea do navegador está diretamente associada aos mecanismos de compilação Just-In-Time (JIT) (MOZILLA DEVELOPER NETWORK, 2025b), que combinam a flexibilidade da interpretação com a eficiência da compilação dinâmica durante a execução. Ao analisar padrões de uso em tempo real e converter trechos críticos em código de máquina otimizado, motores modernos permitem que linguagens como JavaScript atinjam níveis de desempenho anteriormente restritos a ambientes computacionais especializados.

Sob uma perspectiva epistemológica, o JIT desloca parte significativa da complexidade computacional para os próprios mecanismos automáticos de execução. Isso permite que pesquisadores operem em níveis mais elevados de abstração e utilizem arquiteturas modulares sem comprometer a viabilidade da aplicação, uma vez que o engine assume internamente processos de análise, especialização de tipos e otimização adaptativa.

Nesse contexto, ferramentas de construção de linguagens como o ANTLR deixam de representar apenas recursos teóricos e passam a constituir alternativas técnicas concretas para o desenvolvimento de linguagens científicas executáveis diretamente no navegador. A integração entre compilação JIT, análise sintática e processamento interpretado transforma o ambiente Web em uma infraestrutura capaz de suportar sistemas de modelagem matemática de elevada complexidade.

Paralelamente, a consolidação do MathML (Mathematical Markup Language), padrão mantido pelo W3C, permitiu superar limitações históricas da representação matemática em ambientes digitais. Historicamente, a notação matemática em sistemas computacionais esteve frequentemente limitada a representações lineares pouco adequadas à preservação da estrutura lógica das expressões. O MathML possibilita uma representação matemática estruturada e semântica diretamente no navegador, aproximando representação simbólica e execução computacional.

Sob perspectiva epistemológica, essa convergência transforma o navegador

em um espaço de formalização simbólica e mediação científica. Ao integrar execução algorítmica, representação matemática e interação humana em uma mesma infraestrutura distribuída, o ambiente Web passa a suportar formas mais profundas de experimentação, visualização e operacionalização do conhecimento científico.

Além disso, o MathML (W3C, 2023; MOZILLA DEVELOPER NETWORK, 2025c) amplia significativamente a acessibilidade e a interoperabilidade dos conteúdos científicos, permitindo integração com tecnologias assistivas e favorecendo a preservação de representações matemáticas em ambientes digitais distribuídos.

Assim, a computação científica Web, materializada em projetos como o MathJSLab, deixa de representar uma adaptação simplificada de ambientes tradicionais para consolidar-se como uma possibilidade efetiva de computação científica distribuída e multiplataforma. Ao eliminar dependências de instalação local e barreiras associadas a licenciamento proprietário, o navegador passa a atuar não apenas como infraestrutura técnica, mas também como um vetor de democratização, circulação e experimentação do conhecimento científico mediado por software.

2.2 SOFTWARE CIENTÍFICO, CIÊNCIA ABERTA E LEGITIMAÇÃO ACADÊMICA

A consolidação do software como elemento central da prática científica contemporânea produziu transformações que transcendem sua função instrumental, elevando-o à condição de parte integrante da própria infraestrutura epistemológica da ciência (KATZ et al., 2021). Mais do que automatizar cálculos ou operacionalizar rotinas técnicas, scripts executáveis e modelos algorítmicos passaram a estruturar formas de representação, experimentação, validação e circulação do conhecimento científico.

Nesse contexto, o software deixa progressivamente de ocupar posição periférica na pesquisa para assumir o papel de produção intelectual autônoma e artefato acadêmico legítimo. A evolução de ambientes como MATLAB e Octave evidencia essa transformação ao demonstrar que o código executável se tornou indissociável da própria descrição metodológica da atividade científica.

> "Software de Pesquisa" (Research Software) inclui arquivos de código-fonte, algoritmos, scripts, fluxos de trabalho computacionais e executáveis que foram criados durante o processo de pesquisa ou para fins de pesquisa. Componentes de software (por exemplo, sistemas operacionais, bibliotecas, dependências, pacotes, scripts etc.) utilizados para pesquisa, mas que não foram criados durante ou com uma clara intenção de pesquisa, devem ser considerados software em pesquisa e não Software de Pesquisa. Essa diferenciação pode variar entre as disciplinas. (GRUENPETER et al., 2021, tradução nossa).

A definição de Software de Pesquisa é, às vezes, controversa, mas o entendimento da RDA (Research Data Alliance) é que deve atender aos princípios

FAIR (CHUE HONG et al., 2022):

1. Localizável (Findable) – O software e seus metadados associados são fáceis de encontrar tanto para humanos quanto para máquinas.

2. Acessível (Accessible) – O software e seus metadados podem ser recuperados por meio de protocolos padronizados.

3. Interoperável (Interoperable) – O software interopera com outros softwares através da troca de dados e/ou metadados, e/ou por meio da interação via interfaces de programação de aplicativos (APIs), descritas por meio de padrões.

4. Reusável (Reusable) – O software é ao mesmo tempo utilizável (pode ser executado) e reutilizável (pode ser compreendido, modificado, aprimorado ou incorporado a outros softwares).

Sob a perspectiva da Teoria Ator-Rede (TAR) de Bruno Latour (2012), a produção científica emerge de redes heterogêneas compostas por humanos e não humanos. Nessa perspectiva, o software não atua como intermediário passivo, mas como participante ativo nos processos de mediação e produção do conhecimento. Como afirma Latour (2012, p. 72), "uma ação é assumida por diferentes tipos de forças conectadas", indicando que objetos técnicos também exercem agência nas redes sociotécnicas da ciência contemporânea.

Essa dependência crescente da infraestrutura computacional amplia a importância da reprodutibilidade científica mediada por software. O código-fonte passa a funcionar como registro metodológico rigoroso (KATZ et al., 2021), preservando fluxos de execução, parâmetros experimentais e mecanismos operacionais que descrições exclusivamente textuais frequentemente não conseguem representar integralmente.

No caso do MathJSLab, essa mediação adquire também dimensão educacional. Sob a ótica da Teoria Histórico-Cultural (THC) (VYGOTSKY, 2007; LEONTIEV, 2022), o software pode ser compreendido como instrumento cultural e simbólico capaz de reorganizar funções cognitivas e práticas de aprendizagem. Ao integrar modelagem matemática, execução computacional e representação simbólica diretamente no navegador, o sistema constitui um ambiente educacional interativo voltado à experimentação científica e ao pensamento computacional.

Essa integração entre computação científica e educação aproxima-se igualmente das concepções de Pensamento Computacional formuladas por Jeannette Wing (WING, 2006) e das perspectivas construcionistas de Seymour Papert (PAPERT; SOLOMON, 1972), nas quais abstração, decomposição e modelagem computacional passam a constituir habilidades fundamentais da formação científica contemporânea. No contexto brasileiro, essa perspectiva é reforçada pelo Anexo à BNCC – Computação (BRASIL, 2022), que incorpora o raciocínio algorítmico como diretriz para a Educação Básica.

Ao eliminar barreiras de instalação e permitir execução interativa diretamente no navegador, o MathJSLab amplia o acesso a ferramentas computacionais complexas, transformando a Web em um espaço de mediação pedagógica e construção ativa do conhecimento científico.

Paralelamente, o reconhecimento institucional do software científico exige sua integração aos ecossistemas formais de comunicação científica. Plataformas como o Zenodo, mantido pelo CERN/OpenAIRE (2026), permitem a atribuição de identificadores persistentes (DOI), assegurando preservação, versionamento e citabilidade formal do código-fonte.

No caso do MathJSLab, a atribuição de DOI amplia sua inserção no ecossistema acadêmico ao transformar o software em uma entidade citável, rastreável e academicamente preservável. A utilização de ISBN reforça adicionalmente sua natureza de objeto educacional digital e produção intelectual catalogável.

Sob essa perspectiva, mecanismos de identificação persistente deixam de representar apenas recursos administrativos ou bibliográficos e possuem relevância epistemológica. O software científico contemporâneo circula não apenas como ferramenta operacional, mas também como registro metodológico permanente e objeto institucionalmente reconhecido.

Essa transformação está profundamente associada às práticas contemporâneas de Ciência Aberta e Software Livre. A Ciência Aberta demanda transparência metodológica, auditabilidade e compartilhamento público das infraestruturas computacionais utilizadas na pesquisa científica. Nesse contexto, o software livre deixa de representar apenas uma opção técnica ou jurídica e passa a constituir um elemento central da reprodutibilidade científica contemporânea.

No MathJSLab, a adoção da licença MIT consolida uma postura epistemológica baseada na circulação aberta do conhecimento. O código-fonte é tratado como um bem comum passível de uso, modificação, auditoria e redistribuição pública, favorecendo transparência metodológica, colaboração distribuída e reutilização científica.

Essa abertura é operacionalizada por meio de plataformas como GitHub, npm, Zenodo e sistemas contemporâneos de integração contínua, que permitem desenvolvimento distribuído, versionamento público e preservação institucionalizada do software. A integração entre esses ecossistemas demonstra que a produção científica contemporânea não circula apenas em artigos e livros, mas também em modelos executáveis, repositórios distribuídos e infraestruturas computacionais abertas.

Em última análise, o MathJSLab materializa a convergência entre computação científica Web, ciência aberta, software livre e reconhecimento institucional do software como artefato acadêmico legítimo, preservável e citável.

2.3 O MATHJSLAB COMO PROVA DE CONCEITO DE COMPUTAÇÃO CIENTÍFICA WEB

O MathJSLab constitui uma prova de conceito da viabilidade contemporânea da computação científica executada integralmente em ambiente Web. O sistema foi desenvolvido inicialmente em JavaScript e posteriormente reestruturado em

TypeScript, tendo como proposta original a criação de uma calculadora capaz de interpretar expressões matemáticas com variáveis definidas pelo usuário. As primeiras versões baseavam-se predominantemente em substituição textual de caracteres e execução direta de expressões, sem mecanismos formais de interpretação sintática ou representação matemática estruturada.

Com a evolução do projeto, o sistema passou a incorporar mecanismos formais de análise léxica e sintática por meio do ANTLR, permitindo a interpretação de um subconjunto da linguagem MATLAB/Octave diretamente no navegador. Paralelamente, o MathJSLab passou a utilizar o padrão MathML do W3C para representação matemática estruturada, possibilitando que comandos digitados em linguagem computacional sejam automaticamente exibidos em notação matemática usual. Essa característica possui relevância educacional e epistemológica, ao aproximar a codificação computacional das formas tradicionais de representação simbólica utilizadas na matemática e na física.

A interface principal do sistema foi concebida precisamente para reforçar essa mediação entre linguagem computacional e representação matemática convencional. Conforme ilustrado na Figura 1, o ambiente de execução do MathJSLab apresenta simultaneamente o código digitado pelo usuário e sua tradução para notação matemática usual, permitindo que a própria interação com o prompt funcione como exercício contínuo de interpretação simbólica e tradução entre diferentes formas de representação matemática. Já a Figura 2 exemplifica a resolução sequencial de um problema algébrico no ambiente do sistema, evidenciando o encadeamento entre comandos computacionais, processamento simbólico e apresentação matemática estruturada dos resultados.

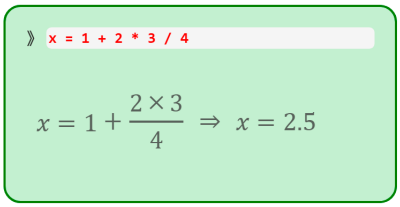

Figura 1: Prompt de comando do MathJSLab, mostrando a tradução e solução do código, exibindo em linguagem matemática usual.

Sob a perspectiva arquitetural, o MathJSLab organiza-se em torno de um núcleo computacional independente, publicado como pacote reutilizável no registro npm . Esse núcleo concentra os mecanismos de interpretação da linguagem, análise sintática, representação matemática e execução computacional, podendo ser incorporado a diferentes aplicações desenvolvidas em JavaScript e TypeScript. A aplicação web interativa atua, assim, como camada de interface e mediação do sistema, consumindo o núcleo matemático como componente modular reutilizável. Essa separação arquitetural favorece a manutenção, extensibilidade e reutilização do interpretador em diferentes contextos computacionais e educacionais.

Além do núcleo matemático propriamente dito, a aplicação Web incorpora um conjunto de tecnologias contemporâneas associadas ao desenvolvimento de aplicações Web distribuídas. A interface utiliza templates HTML e folhas de estilo estruturadas em SCSS (Sassy Cascading Style Sheets), permitindo modularização e organização avançada dos componentes visuais. O sistema de geração das páginas utiliza templates Nunjucks, favorecendo a reutilização estrutural e geração dinâmica da interface. Já a documentação integrada do sistema — incluindo o mecanismo de ajuda (help) de comandos — é construída a partir de arquivos Markdown processados automaticamente durante o fluxo de execução da aplicação .

Nesse contexto, a própria documentação do sistema passa a integrar o ecossistema computacional do MathJSLab. Expressões matemáticas presentes na documentação são renderizadas utilizando o próprio mecanismo simbólico desenvolvido para o sistema, assegurando uniformidade entre execução computacional e representação matemática exibida ao usuário. Paralelamente, diagramas e representações gráficas são incorporados por meio do Mermaid , permitindo geração declarativa de fluxogramas, diagramas estruturais e representações visuais diretamente em ambiente Web.

Diferentemente de ambientes científicos tradicionais dependentes de instalação local e infraestrutura específica, o MathJSLab opera integralmente sobre tecnologias padronizadas da Web contemporânea, explorando os mecanismos modernos das engines JavaScript dos navegadores atuais. A execução ocorre localmente no navegador do usuário, sem dependência contínua de processamento remoto, evidenciando uma transformação relevante do navegador em ambiente de modelagem, execução e mediação científica distribuída.

A viabilidade contemporânea desse modelo computacional está diretamente associada à evolução dos mecanismos de compilação Just-In-Time (JIT) incorporados às engines JavaScript modernas, como o V8 (GOOGLE V8 PROJECT, 2026; MOZILLA DEVELOPER NETWORK, 2025a; 2025b). Diferentemente das primeiras gerações de interpretadores Web, nas quais pequenas escolhas sintáticas podiam produzir diferenças expressivas de desempenho, os compiladores JIT atuais realizam otimizações dinâmicas durante a própria execução do programa, analisando padrões de uso, especializando trechos frequentemente executados e produzindo versões otimizadas do código em tempo real. Em consequência, muitas decisões microestruturais de otimização deixam de constituir preocupação central do programador, permitindo que o desenvolvimento de sistemas complexos privilegie legibilidade, modularidade e clareza algorítmica em vez de ajustes manuais contínuos de desempenho.

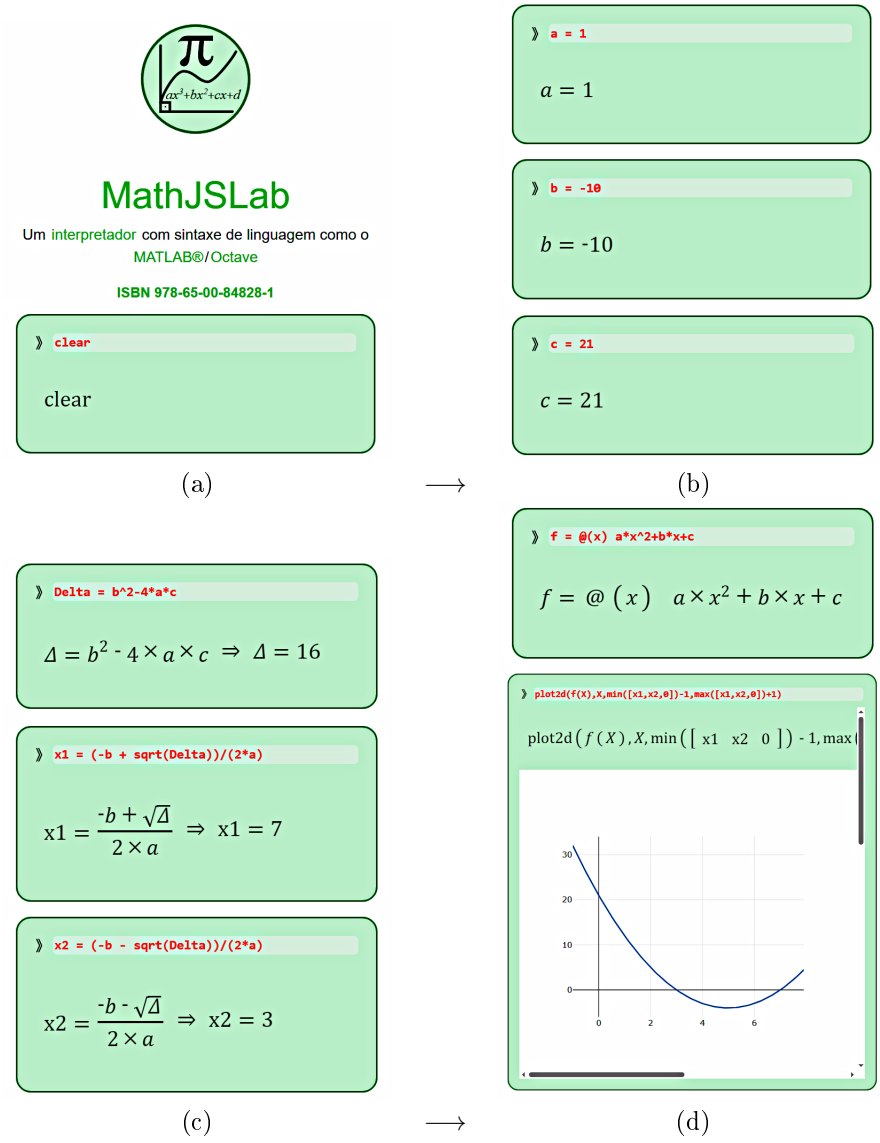

Figura 2: Telas do MathJSLab mostrando a resolução da equação do 2º grau: (a) começa apagando todas as variáveis; (b) definição dos coeficientes da equação; (c) solução pela fórmula de Bhaskara; (d) definição da função e construção do gráfico.

Em contextos anteriores da computação interpretada, diferenças aparentemente simples na escrita de um algoritmo frequentemente exigiam avaliação empírica detalhada de eficiência. Estruturas logicamente equivalentes poderiam apresentar tempos de execução significativamente distintos, tornando comum a

necessidade de reescrever rotinas inteiras apenas para contornar limitações do interpretador. Em JavaScript contemporâneo, entretanto, mecanismos JIT reduzem substancialmente essa dependência de micro-otimizações manuais. Por exemplo, operações iterativas semanticamente equivalentes como:

```
for (let i = 0; i < n; i++) {
        soma += vetor[i];
}
```

e:

```
for (const valor of vetor) {
        soma += valor;
}
```

tendem a convergir para desempenhos semelhantes após os ciclos de otimização realizados pela engine. Semelhantemente, expressões matemáticas escritas de forma mais declarativa e semanticamente legível podem atingir desempenho comparável a implementações manualmente especializadas, pois o compilador identifica padrões recorrentes e reorganiza dinamicamente a execução interna do código. Essa característica altera significativamente o próprio processo de desenvolvimento de software científico, permitindo uma programação relativamente mais "descuidada" do ponto de vista microestrutural, sem comprometer necessariamente a eficiência final da execução.

Sob perspectiva histórica, essa transformação possui relevância particular para a computação científica em ambiente Web. A combinação entre compilação dinâmica, otimização adaptativa e crescente poder computacional dos navegadores contemporâneos tornou viável a execução local de aplicações anteriormente restritas a ambientes científicos especializados. Assim, o navegador deixa de atuar apenas como interface de acesso remoto e passa a constituir efetivamente um ambiente distribuído de modelagem, experimentação e computação científica executável.

Além disso, o MathJSLab é implementado como uma Progressive Web Application (PWA) . Esse modelo arquitetural permite que a aplicação seja instalada diretamente pelo navegador, utilize mecanismos locais de armazenamento em cache e continue funcionando mesmo em cenários de conectividade limitada ou inexistente. Sob perspectiva educacional e científica, essa característica amplia significativamente a acessibilidade do sistema, reduzindo dependências de instalação tradicional e aproximando aplicações Web do comportamento historicamente associado a softwares científicos desktop. Em contextos educacionais, isso favorece a utilização do sistema em laboratórios escolares, dispositivos móveis e ambientes com infraestrutura computacional heterogênea.

A arquitetura distribuída do sistema também se articula com práticas contemporâneas de ciência aberta e desenvolvimento cooperativo. O núcleo computacional é publicado como pacote reutilizável no npm, os repositórios são mantidos publicamente no GitHub da organização MathJSLab e a distribuição da aplicação utiliza plataformas de hospedagem, integração e entrega contínua,

como Netlify e CircleCI . O projeto também adota a licença MIT, favorecendo reutilização, modificação e circulação aberta do software.

A formalização institucional do MathJSLab por meio de DOI e ISBN reforça sua inserção nos mecanismos contemporâneos de preservação e citabilidade científica. Dessa forma, o sistema evidencia que o software científico contemporâneo não circula apenas como ferramenta computacional, mas também como artefato acadêmico executável, distribuído e institucionalmente reconhecido. Assim, o MathJSLab demonstra que a computação científica Web contemporânea não constitui apenas uma adaptação simplificada de ambientes tradicionais, mas uma transformação mais ampla das formas de produção, mediação, circulação e preservação do conhecimento científico em ambientes digitais distribuídos.

2.4 IMPLICAÇÕES EPISTEMOLÓGICAS E EDUCACIONAIS

As transformações discutidas ao longo deste trabalho evidenciam que a computação científica contemporânea ultrapassa uma mudança meramente técnica das plataformas computacionais. A consolidação do navegador Web como ambiente capaz de executar aplicações matemáticas complexas altera também as formas de mediação, circulação e produção do conhecimento científico, aproximando infraestrutura computacional, experimentação acadêmica e interação educacional em um mesmo espaço distribuído.

Nesse contexto, o MathJSLab exemplifica uma convergência entre computação científica, mediação educacional e produção intelectual baseada em software. Ao integrar modelagem matemática, interpretação de linguagem, representação simbólica e execução interativa diretamente no navegador, o sistema evidencia que tecnologias Web contemporâneas podem atuar simultaneamente como ambiente computacional, instrumento pedagógico e infraestrutura científica distribuída.

Sob perspectiva educacional, essa mediação aproxima-se das discussões relacionadas ao Pensamento Computacional (AHO, 2011) e às teorias de aprendizagem mediadas por tecnologias digitais. Ao automatizar operações técnicas de baixo nível e favorecer práticas de modelagem, abstração e experimentação algorítmica, o software amplia possibilidades de interação ativa com objetos matemáticos e computacionais. A execução diretamente no navegador reduz barreiras técnicas de instalação e acesso, favorecendo ambientes de aprendizagem mais acessíveis, interativos e compatíveis com propostas contemporâneas de educação científica e computacional.

Paralelamente, a inserção do MathJSLab em ecossistemas de Software Livre e Ciência Aberta evidencia mudanças significativas na própria materialidade da produção científica contemporânea. O conhecimento científico deixa progressivamente de circular apenas por meio de textos estáticos e passa a incorporar modelos executáveis, repositórios distribuídos, infraestruturas computacionais abertas e artefatos digitais preserváveis. Nesse cenário, identificadores persistentes, como DOI e ISBN, não representam apenas mecanismos administrativos

de catalogação, mas instrumentos de legitimação institucional do software como produção intelectual acadêmica.

A integração entre desenvolvimento distribuído, preservação digital, versionamento público e citabilidade formal demonstra que o software científico contemporâneo pode assumir simultaneamente funções operacionais, metodológicas e epistemológicas. O MathJSLab materializa essa transformação ao atuar não apenas como ferramenta computacional executável, mas também como artefato acadêmico inserido em redes de circulação, validação e preservação do conhecimento científico.

Assim, o presente estudo reforça a compreensão de que a computação científica em ambiente Web não constitui apenas uma evolução tecnológica das plataformas digitais contemporâneas, mas também uma transformação mais ampla das formas pelas quais o conhecimento científico é produzido, compartilhado, operacionalizado e legitimado em contextos mediados por software.

3 CONCLUSÃO

O presente trabalho demonstrou que o MathJSLab constitui uma prova de conceito consistente da viabilidade da computação científica em ambiente Web e, simultaneamente, um modelo de software compreendido como artefato acadêmico institucionalmente reconhecido. A análise evidenciou que a evolução dos navegadores Web — impulsionada por engines JavaScript de alto desempenho, mecanismos de compilação Just-In-Time (JIT) e padrões como o MathML — ampliou significativamente as capacidades computacionais do navegador, permitindo que ele deixasse de atuar apenas como interface de acesso à informação para assumir funções de execução, modelagem e mediação científica distribuída.

Sob perspectiva epistemológica, concluiu-se que o software deixou de ocupar uma posição estritamente instrumental para integrar diretamente a infraestrutura contemporânea de produção do conhecimento científico. Ao operacionalizar estruturas conceituais, modelos matemáticos e procedimentos experimentais, o código executável torna-se indissociável das práticas de modelagem, simulação e validação científica. Essa transformação é reforçada por infraestruturas contemporâneas de Ciência Aberta, como o Zenodo (CERN/OpenAIRE, 2026), que, ao atribuírem DOI ao código-fonte, inserem o software nos mecanismos formais de preservação, citabilidade e circulação da comunicação científica.

No campo educacional, o MathJSLab materializa princípios associados ao Pensamento Computacional e às diretrizes da BNCC Computação (BRASIL, 2022), oferecendo um ambiente interativo de modelagem matemática e experimentação algorítmica executado diretamente no navegador. Ao reduzir barreiras técnicas relacionadas à instalação e configuração de ambientes especializados, o sistema amplia possibilidades de acesso a ferramentas computacionais voltadas ao ensino e à aprendizagem científica. A atribuição de ISBN reforça adicionalmente sua natureza de objeto educacional digital e produção intelectual catalogável.

Em síntese, o trabalho evidencia que a materialidade da produção científica

contemporânea vem sendo progressivamente reconfigurada por infraestruturas computacionais distribuídas, modelos executáveis e ecossistemas digitais abertos. Nesse contexto, o MathJSLab exemplifica como a computação científica em ambiente Web pode articular acessibilidade, reprodutibilidade, circulação aberta do conhecimento e preservação institucional do software científico. Mais do que uma adaptação técnica de ambientes computacionais tradicionais, o sistema evidencia transformações mais amplas nas formas de produção, operacionalização e legitimação do conhecimento científico mediado por software.

Referências

Armenuhi Abramyan. *JavaScript for Science*. Slides presented at the Inverted CERN School of Computing (iCSC) 2020, 16–18 March 2020. Inverted Computing School (iCSC). 2020. URL: https://indico.cern.ch/event/853710/contributions/3708132/attachments/1985053/3307323/Armina_Abramyan_JS_for_Science.pdf (acesso em 18/05/2026).

ANTLR. *ANTLR*. Acesso em: 16 maio 2026. 2026. URL: https://www.antlr.org/.

Matthew Casperson. *Q&A with Terence Parr on ANTLR*. 11 de jan. de 2016. URL: https://dzone.com/articles/qa-with-terence-parr-on-antlr (acesso em 18/05/2026).

Christina von Flach e Fabio Kon. "Software livre: pré-requisito para a ciência aberta". Em: *Computação Brasil* 46 (dez. de 2021), pp. 12–15. ISSN: 2965-9728. DOI: 10.5753/compbr.2021.46.4412.

GNU Octave Project. *GNU Octave*. Acesso em: 16 maio 2026. 2026. URL: https://octave.org/.

Google V8 Project. *V8 JavaScript Engine*. Open-source JavaScript and WebAssembly engine. 2026. URL: https://v8.dev/ (acesso em 20/05/2026).

Daniel S. Katz et al. "Recognizing the Value of Software: A Software Citation Guide". Em: *F1000Research* 9 (jan. de 2021), p. 1257. ISSN: 2046-1402. DOI: 10.12688/f1000research.26932.2. URL: https://doi.org/10.12688/f1000research.26932.2.

Sérgio Lindau. *MathJSLab*. Versão 1.9.2. Concept DOI: 10.5281/zenodo.8396265. 2026. DOI: 10.5281/zenodo.8396263. URL: https://mathjslab.com (acesso em 31/01/2026).

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

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

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

Mozilla Developer Network. *Just-In-Time Compilation (JIT)*. MDN Web Docs glossary entry. 2025. URL: https://developer.mozilla.org/en-US/docs/Glossary/Just_In_Time_Compilation (acesso em 19/05/2026).

Mozilla Developer Network. *MathML*. Acesso em: 16 maio 2026. 2025. URL: https://developer.mozilla.org/en-US/docs/Web/MathML.

Francisco Ortin et al. "An Empirical Evaluation of Lex/Yacc and ANTLR Parser Generation Tools". Em: *PLoS ONE* 17.3 (3 de mar. de 2022). Ed. por Sergio Consoli, e0264326. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0264326. URL: https://doi.org/10.1371/journal.pone.0264326.

Terence Parr. *The Definitive ANTLR 4 Reference*. Book version: P 2.0. The pragmatic programmers. Dallas, Texas: The Pragmatic Bookshelf, 19 de fev. de 2013. 307 pp. ISBN: 978-1934356999. URL: https://github.com/wojiaxiaoyueyue/Antlr4.7-Calculator/blob/master/The%20Definitive%20ANTLR%204%20Reference,%202nd%20Edition.pdf (acesso em 17/05/2026).

World Wide Web Consortium (W3C). *W3C Math Home*. Acesso em: 16 maio 2026. 2023. URL: https://www.w3.org/Math/.