Not Just Another Linear Algebra Book

Martina M. Jardim, Marcello L. R. de Campos, and Amit Bhaya

Abstract— **This paper describes the use of a suite of software tools that can be used as part of an integrated learning environment, including text, interactivity and projects for the teaching of undergraduate linear algebra.**

Keywords— **Linear algebra, interactive learning, web-based, cloud-based, Julia, Python, JupyterLite.**

I. Introduction

Knowledge of linear algebra is essential to all engineering fields and, in the particular case of signal processing, this has been cogently argued in Strang [5]. Despite the profusion of excellent linear algebra textbooks, most lack the interactivity demanded by contemporary audiences. Students used to *viewing* information online, rather than *reading* it in a book, may find the subject less interesting than their instructors expect. On the other hand, sites and videos devoted to teaching linear algebra often lack the cadence and instructional design needed to support a regular course. Using acessible computing resources, it is possible to create live exemples that can be followed step by step and altered by the reader, and so, will reduce the gap between studying and learning.

This article describes the use of a suite of software tools that can be used as part of such an integrated learning environment, including text, interactivity and projects for the teaching of undergraduate linear algebra. The tools chosen for the project were the ones found to provide the lowest latency, best portability, and cloud-based options, allowing students to participate without requiring personal computing equipment.

A. Brief review of existing interactive approaches

There have been some notable efforts to make the subject of linear algebra attractive to GenZ students.

Pavel Grinfeld's lemma site [1] is a linear algebra learning site, with a web-based interface that includes smart textboxes to enter equations from the keyboard that are nicely typeset, short video lessons, followed by worksheets with hints and automated grading, leading onto the next lesson. The interactivity is limited to dialog boxes in which the student enters answers, comments or questions, but not allowing the student to change problem parameters through sliders, for example.

Dan Margalit and Joseph Rabinoff authored an interactive linear algebra book, covering the basics of undergraduate linear algebra [4] with all code available via GitHub (https://github.com/QBobWatson/ila). Interactivity is limited to manipulation of the figures, some of which incorporate sliders, as well as examples in which students can enter parameter values and observe the resulting changes to plots.

Martina M. Jardim is with the Department of Electronics Engineering, Federal University of Rio de Janeiro, Rio de Janeiro-RJ, e-mail: martinamarques@poli.ufrj.br; Marcello L. R. de Campos and Amit Bhaya are with the Electrical Engineering Program, Federal University of Rio de Janeiro, Rio de Janeiro-RJ, e-mail: campos@smt.ufrj.br and amit@nacad.ufrj.br. This work was partially supported by CNPq contracts 305948/2019-9 [MLRdC] and 313335/2022-2 [AB], and FAPERJ contract E-26/204.347/2022.

David Lay's well known textbook [3] is available in an interactive electronic format. Using Wolfram CDF Player, a free Mathematica player available from Wolfram (www.wolfram.com/player), "*students can interact with figures and experiment with matrices by looking at numerous examples. ...The resources in the interactive version of the text give students the opportunity to interact with mathematical objects and ideas ...*" [quote from the preface]

Finally, the book authored by David I. Ketcheson, Randall J. LeVeque, and Mauricio J. del Razo [2] "*is available in electronic form as a collection of Jupyter notebooks that contain executable computer code and interactive figures and animations, allowing readers to grasp how the concepts presented are affected by important parameters and to experiment by varying those parameters themselves*" This book also served as one of the inspirations for the tools and approach adopted in the present paper.

With regard to the existing interactive approaches, the contribution of this paper is to provide a suite of software tools, all open access and web-/cloud-based that allow a greater degree of student interactivity, without requiring personal access to computing power or proprietary software. In contrast, the approach in [3] uses code written in Mathematica, which is proprietary, although it runs on Wolfram's CDF player which is freely downloadable. However, although the student can interactively change problem parameters, she does not have access to the code itself. The tools used in [4], as mentioned above, have limited interactivity, but are written in open source code, made available on Github. This means that a user who is familiar with coding can, in principle, increase the level of interactivity, although this will usually not be the case with most of the targets users that we have in mind. The approach in [2] provides the highest level of interactivity, but is implemented on a platform (Jupyter+Binder) that relies on external infrastructure and has large launch times and latency, which makes it impractical and often inaccessible.

This paper proposes a solution that has low launch times (using JupyterLite), a high degree of interactivity through the application of different modules and libraries (Bokeh, Matplotlib, ipywidgets) and allows the user to approach the code and the mathematical theory in the same platform. Therefore, it is possible to construct an online environment in which the student can actively interact with different algebraic structures and algorithms, through data input, plots and code visualization.

II. The Tools

This paper reports on preliminary results in a project whose goal is to create an interactive and online Linear Algebra book, which will allow the student to learn both theoretical and practical computational details of the subject.

In order to construct a suitable online environment, three different aspects need to be considered: ability to switch between markdown text and code; a programming language in which mathematical concepts can be expressed naturally; availability of tools that simplify visualization and plotting tasks. Jupyter notebooks, chosen in our project, provide one such well known environment, in which code and markdown cells can be mixed freely.

Initially, following [2], an attempt was made to use Binder. Binder is an online tool that accesses Jupyter Notebooks in public GitHub repositories and generates a URL to an executable environment, where the notebooks can be run and temporarily altered by the user. Despite some initial successes, several accessibility limitations (e.g., through a URL or creation of a new URL) which occur because Binder uses servers owned by different entities, led to the substitution of Binder by JupyterLite.

JupyterLite is a JupyterLab distribution that can be executed in a browser. The URL for JupyterLite is created through GitHub pages, so that it is lightweight (as the name suggests) with low launch times, primarily due to the fact that it is independent of infrastructure, depending solely on GitHub.

In regards to the programming language, two different options were considered: Julia and Python. Julia is a recent language for scientific computing, and offers an intuitive syntax for algorithms involving vectors and matrices. It is native to the Jupyter Notebook, and also offers Pluto for an interactive Julia environment. However, in our experiments using Pluto and Binder, very large launch times were observed, detrimental to the learning process. For our purposes, the older, richer Python ecosystem offered the best plotting and interactivity modules, which could be interpreted and rapidly executed by JupyterLite and Binder.

III. Examples

The toy examples described below were chosen to help us explore and assess the advantages and limitations of the tools.

In the activity shown in Fig. 1, the student observes two sets of linearly independent vectors that can be used as bases for the same vector space, and is asked to find different linear combinations resulting in the same vector for each of the bases. A Float Text input was used for improved input possibilities. The activity shown below uses the Matplotlib library, which requires usage with ipywidgets in order to create interactivity. In our project, the Python library Bokeh will also be used. It creates plots that can be dragged or zoomed by the user using the mouse cursor.

Fig 2 is an example of a Python code that generates a matrix of any dimension interactively. The student inserts the dimensions of the matrix through ipywidgets' FloatText widget. Subsequently, ipywidgets is used again to create an input structure that is visually similar to the matrix notation.

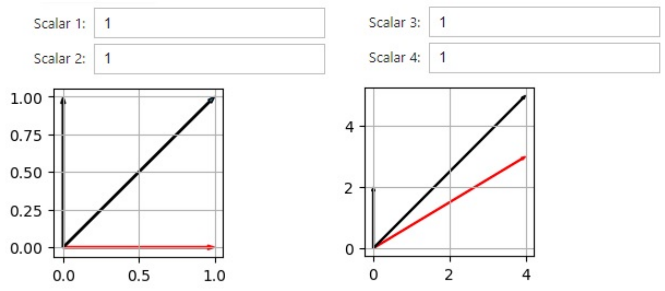
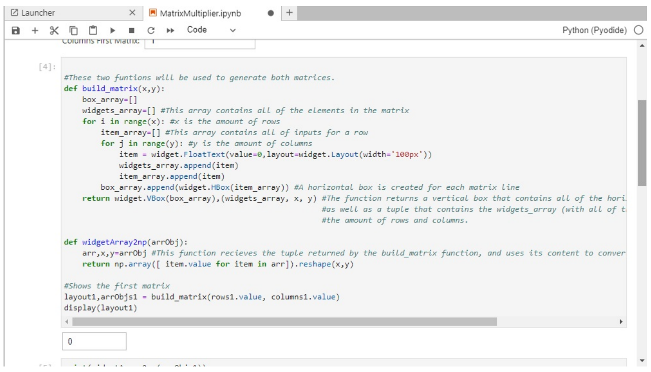


Fig. 1: Interactive Plots using MatPlotLib in JupyterLite.



```
#These two funtions will be used to generate both matrices.
def build_matrix(x,y):
    box_array=[]
    widgets_array=[] #This array contains all of the elements in the matrix
    for i in range(x): #x is the amount of rows
        item_array=[] #This array contains all of inputs for a row
        for j in range(y): #y is the amount of columns
            item = widget.FloatText(value=0,layout=widget.Layout(width='100px'))
            widgets_array.append(item)
            item_array.append(item)
        box_array.append(widget.HBox(item_array)) #A horizontal box is created for each matrix line
    return widget.VBox(box_array),(widgets_array, x, y) #The function returns a vertical box that contains all of the hori
                                                        #as well as a tuple that contains the widgets_array (with all of t
                                                        #the amount of rows and columns.


def widgetArray2np(arrObj):
    arr,x,y=arrObj #This function recieves the tuple returned by the build_matrix function, and uses its content to conver
    return np.array([ item.value for item in arr]).reshape(x,y)

#Shows the first matrix
layout1,arrObjs1 = build_matrix(rows1.value, columns1.value)
display(layout1)
```

Fig. 2: Use of ipywidgets to generate matrix.

IV. Conclusions

The project seeks to create an online interactive environment for teaching Linear Algebra, with free access for all. After extensive experimentation, we chose JupyterLite as a rendering tool for JupyterNotebooks written in Python. This combination was found to be the most effective since it allows interactive inputs and plots, has high-level programming syntax and uses an adequate rendering tool that publishes the codes and text correctly. The use of this set of tools achieves the objective of creating an interactive and democratic environment for learning and teaching Linear Algebra. Current versions of all interactive material developed in the project are available in the GitHub pages website: https://martinamj.github.io/JupyterLiteTest/lab/index.html
The official repository for the project is https://github.com/MartinaMJ/Online-Linear-Algebra.

References

[1] Pavel Grinfeld, "The Lem.ma website," *https://www.lem.ma/library*, 2023.

[2] David I. Ketcheson and Randall J. LeVeque and Mauricio J. del Razo, *Riemann Problems and Jupyter Solutions: Theory and Approximate Solvers for Hyperbolic PDEs*, SIAM, Phialdelphia, 2020.

[3] David Lay and Steven R. Lay and Judi J. McDonald, *Linear Algebra and its Applications*. Pearson Education, Fifth ed., 2016.

[4] Dan Margalit and Jospeh Rabinoff, "Interactive Linear Algebra," *https://textbooks.math.gatech.edu/ila/index2.html*, 2019.

[5] Gilbert Strang, *Signal processing for everyone*, In: Burkard, R.E., et al. Computational Mathematics Driven by Industrial Problems. Lecture Notes in Mathematics, vol 1739, pp.366-412, Springer, Berlin, Heidelberg.