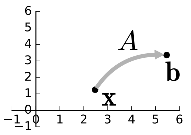
New Tools for Teaching Linear Algebra

Mark Crovella, Department of Computer Science

Students learning Linear Algebra need to develop three modes of thinking. The first is *algebraic* thinking – how to correctly manipulate symbols in a consistent logical framework. The second is *geometric* thinking: learning to extend familiar two- and three-dimensional concepts to higher dimensions in a rigorous way. The third is *computational* thinking: understanding the relationship between abstract algebraic machinery and the actual computations which arrive at the correct answer to a specific problem in an efficient way. These three modes of thinking are quite different, and often students are better at some modes than others. For example, here are the three views of matrix-vector multiplication:

$$A\mathbf{x} = \mathbf{b}$$



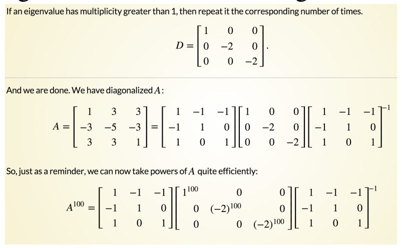
$$\texttt{b[i]} = \texttt{sum(A[i,:]*x)}$$

The ideal teaching environment demonstrates all three of these modes well, and importantly, interweaves these modes. Each mode provides a distinct, powerful way of thinking about a problem, and so using the full power of linear algebra requires being able to switch between these modes with fluidity.

In designing the teaching materials for CAS/CS132, which covers linear algebra from a Computer Science viewpoint, I wanted to provide a lecture experience that smoothly incorporates the three modes. Furthermore, this lecture experience needed to be reproducible by students as they study outside of the classroom, and modifiable by students so that they can explore and develop understanding experimentally. This led me to deliver lectures using a set of newly emerging software from the scientific computing community: *Jupyter notebook* and *RISE*. These are open-source tools developed around the programming language Python; I'll refer to them together as *Jupyter/RISE*. These tools were not developed for teaching and so a significant effort in developing this course involved adapting them for this purpose.

To understand the fundamental change when using Jupyter/RISE as an educational tool, one should set aside the idea of a static set of "slides" presented by the instructor. Instead, what is presented by the instructor is a stream of content that is *computed* in real time, interactively, during the class. That is, all aspects of the presentation are the output of running (Python) code, and that code can be modified for instructional purposes during the lecture. Crucially, students may recreate the lecture on their own at any time, and make their own exploratory modifications.

More specifically, Jupyter/RISE provides features that support all three modes: (a) typesetting software that creates beautifully-displayed mathematical notation, for algebraic work; (b) graphics and plotting tools that can create dynamically modifiable figures and animations, for geometric work; and (c) a running programming language interpreter (Python) that executes dynamically modifiable code and shows the output in the same visual context as the algebraic notation and the figures.



If an eigenvalue has multiplicity greater than 1, then repeat it the corresponding number of times.

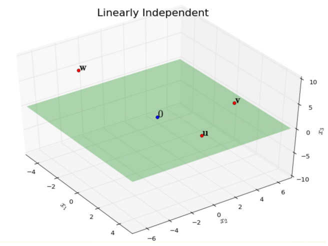
$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{bmatrix}.$$

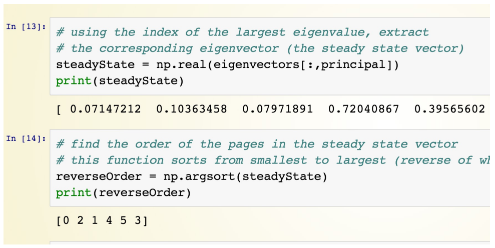
And we are done. We have diagonalized A:

$$A = \begin{bmatrix} 1 & 3 & 3 \\ -3 & -5 & -3 \\ 3 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}^{-1}$$

So, just as a reminder, we can now take powers of A quite efficiently:

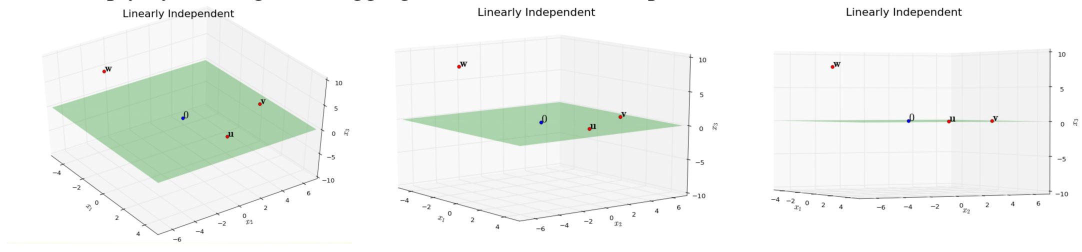
$$A^{100} = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1^{100} & 0 & 0 \\ 0 & (-2)^{100} & 0 \\ 0 & 0 & (-2)^{100} \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}^{-1}$$





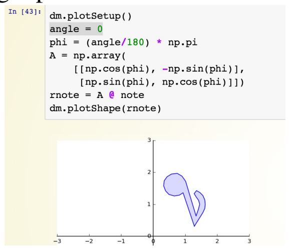
```
In [13]: # using the index of the largest eigenvalue, extract
         # the corresponding eigenvector (the steady state vector)
         steadyState = np.real(eigenvectors[:,principal])
         print(steadyState)

         [ 0.07147212  0.10363458  0.07971891  0.72040867  0.39565602

In [14]: # find the order of the pages in the steady state vector
         # this function sorts from smallest to largest (reverse of wh
         reverseOrder = np.argsort(steadyState)
         print(reverseOrder)

         [0 2 1 4 5 3]
```

To see the power of dynamically computed figures, consider the middle figure above. This is taken from a lesson explaining the concept of linear independence. The key to understanding is to grasp that three of the points lie in the green plane, while the fourth point lies outside the plane. To make this intuitive, the instructor simply rotates the figure in real time – simply by clicking and dragging – to demonstrate this point:



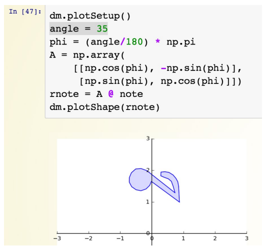
A lecture excerpt illustrating this in practice can be found [here].

Next is an example showing the power of dynamic interaction with both code and figures. To illustrate the concept of a rotation matrix, the instructor creates one in Python, and then uses it to rotate a graphical object. The instructor can modify the code (shown in gray) and re-execute the code block, showing the rotation effect in real time. Because this demonstration is visually inline with the previous algebraic discussion, the three modes of thinking are combined in a single place.



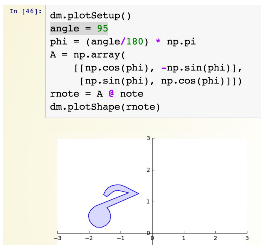
```
In [43]: dm.plotSetup()
         angle = 0
         phi = (angle/180) * np.pi
         A = np.array(
             [[np.cos(phi), -np.sin(phi)],
              [np.sin(phi), np.cos(phi)]])
         rnote = A @ note
         dm.plotShape(rnote)
```

$\longrightarrow$



```
In [47]: dm.plotSetup()
         angle = 35
         phi = (angle/180) * np.pi
         A = np.array(
             [[np.cos(phi), -np.sin(phi)],
              [np.sin(phi), np.cos(phi)]])
         rnote = A @ note
         dm.plotShape(rnote)
```

$\longrightarrow$



```
In [46]: dm.plotSetup()
         angle = 95
         phi = (angle/180) * np.pi
         A = np.array(
             [[np.cos(phi), -np.sin(phi)],
              [np.sin(phi), np.cos(phi)]])
         rnote = A @ note
         dm.plotShape(rnote)
```

A lecture excerpt illustrating this in practice can be found [here].

Finally, support for algebraic reasoning is greatly enhanced by moving away from the notion of "slides" – fixed-size rectangles of text presented sequentially. Rather, the environment is based on a *stream* of content, in which scrolling is the default interaction. This is crucial for following chains of logical reasoning, as when presenting a proof. The ability to scroll keeps relevant context always on screen and allows students to look back and forth to understand the flow of argument. Screen-clearing page breaks are only used when it is time to establish a new context.

The key parts of this toolset are all open-source, community-developed. The Python interpreter is provided by Jupyter notebook, while the presentation software that turns the notebook into slides is provided by RISE. These tools arose from a need in scientific computing to organize and document computationally-driven experiments. In particular, Jupyter is used as a laboratory notebook and is popular in the open science movement. However using these tools together in pedagogy is much less developed – I have not found any other examples of instructors using Jupyter notebook specifically combined with RISE in the classroom. Given the outstanding benefits, there is considerable potential for wider use of this toolset in education.

Note that while interactive lecture materials are good for active learning, students also want hardcopy materials for offline study. Hence I have developed and use tools that automatically convert every lecture into an equivalent typeset PDF document.

In teaching CS132 I make all materials – the interactive lecture notebooks, hardcopy materials, and all associated code, syllabi, etc, available to students via a `github` repository. `Github` is the standard site used by the technology industry for sharing code and other resources, in an open, social fashion. `Github` incorporates well-designed tools that encourage collaboration. A number of students have made use of `github's` online tools to automatically keep their personal copy of all materials up to date. The course's `github` repository is accessible [here].

Student reaction has been very positive. Furthermore, the entire set of tools and teaching methods was used during Summer 2015 by another instructor in our department (John Magee), also with positive feedback. The materials are also being incorporated in the course as taught by a new instructor at BU this fall.

Beyond the teaching materials, another drawback of the traditional classroom experience is the well-known 20 to 30 minute limitation on learning that takes place when students continuously take notes. This is a critical issue for Linear Algebra. A typical lecture sequence may take the following form: (a) introduce a concept (say, a vector space); (b) define it formally, and give examples; (c) prove important facts about the concept that will be needed for later lessons. When such a sequence is presented in a lecture-only format, I have found that there is a critical disconnect that happens between steps (b) and (c). In order for a student to transition from learning about a concept to actually using the concept, they must engage a mode of active, critical thinking that is not generally triggered just by taking notes.

For this reason, I introduced the use of student response devices and peer instruction into CS132. The key transition between steps (b) and (c) above is aided by a short question, answered via peer instruction. The question is designed to require students to engage with and internalize the concept just introduced. I give around 3 questions per lecture, which breaks lecturing up into segments of 30 minutes or less. There are consistently large improvements in student comprehension (as measured by responses before and after peer discussion) which has been a very positive factor in overall student success.