Visions and Realizations of a Computational eTextBook[*]

Rubin H Landau
Department of Physics
Oregon State University
Corvallis, OR 97331
rubin@science.oregonstate.edu

Manuel J Páez
Department of Physics
University of Antioquia
Medellin, Colombia
mpaezenator@gmail.com

Cristian C Bordeianu
Faculty of Physics
University of Bucharest
Bucharest, Romania
cbord1@gmail.com

ABSTRACT

A series of implementations of a multisensory, interactive eTextBook in Computational Physics with multiple executable elements is described. Various ways to include text, computational laboratories, demonstrations and video–based lecture modules are described. Advances and setbacks in the realizations of our original vision, as well as modifications instigated by changing Web technologies and mobile devices are described. The happy ending of forthcoming realizations in the form of commercial paper and electronic versions is presented.

Categories and Subject Descriptors

D.3.2 [**Programming Languages**]: Language Classifications—*Python*; D.4.6 [**Operating Systems**]: Security and Protection; H.4 [**Information Systems Applications**]: Miscellaneous; I.1 [**Computing Methodologies**]: Symbolic and Algebraic Manipulations; J.2 [**Computer Applications**]: Physical Science and Engineering—*Physics*; J.72 [**Computer Applications**]: Computers in Other Systems—*Publishing*; K.3.1 [**Computers and Education**]: Computer Uses in Education—*Computer-assisted Instruction*; K.3.1 [**Computers and Education**]: Computer Uses in Education—*Distance Learning*

General Terms

Theory, Education

Keywords

eBooks, Computational Physics, Computational Science, Executable Paper, Multimedia, Sonification

[*]Paper and "flat" digital versions of the text become available in summer 2015 from Wiley-VCH. The interactive HTML5 eTextBook is presently in production. Alas, the pdf draft we gave away for several years is no longer available. Work supported in part by several National Science Foundation Grants.

1. HOW THIS ALL STARTED

After two decades of basic research in computational, subatomic, few-body physics, in 1986 one of us (RHL) organized a group of Oregon State University faculty with the aim of creating a program of study in the still-developing field of "computational science". After several years of discussions and negotiations it became clear that it was too early for the university decision makers to see the importance that computation would play in science, and that our "bottom-up" approach involving multiple departments, multiple colleges and a collection of existing courses was leading to an impossible lengthy program of study. And thus was born Landau's idea that he would instead work at introducing a new course in Computational Physics [1], that in time blossomed into a full degree program [2]. That course and its offsprings were eventually approved by a series of committees and got going when IBM gave us a single RT Unix workstation on which all of the students could write and run their programs simultaneously. (As the internet started developing, that same workstation was simultaneously used as the department's router and bridge.)

By the early 1990's a series of lecture notes were being written with the aim of eventually producing a computational physics textbook. Although there a few computational physics textbooks in existence then, they were not appropriate for our students and for our viewpoint. In particular, they did *not* emphasize the computational science point of view that the applied math (algorithm) and the computer science (error, memory management and such) were just as important as the physics. And so it seemed to make sense for us to develop a Computational Physics textbook with a computational science point of view and that surveyed a larger number of topics in order to introduce a broad range of computational science tools.

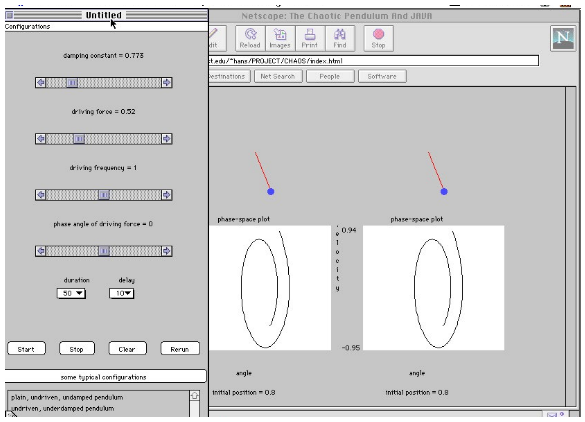
Manuel Páez joined the project in 1995 with his creation of the chapters and codes on partial differential equations, and then the collections of applets. He has since contributed in multiple ways, most recently with discussion of GPU programming. Cristian Bordeianu joined the project in 2002 with his creation of the Java versions of the codes as well as the chapter on wavelets. Since then he too has contributed in multiple ways. The paper textbook has now gone through four editions, and in Table 1 we list the chapters in the forthcoming edition [3].

Our realization of the great potential of an electronic text began in November 1994 during an early meeting of the Undergraduate Computational Engineering and Sciences program [4]. There, individuals interested in computational

Table 1: *Computational Physics* Book Chapters

1.	Introduction	15.	Continuous Nonlinear Dynamics
2.	Computing Software Basics	16.	Fractals & Statistical Growth Models
3.	Errors & Uncertainties in Computations	17.	Thermodynamic Simulations, Feynman Path Integrals
4.	Monte Carlo: Randomness, Walks & Decays	18.	Molecular Dynamics Simulations
5.	Differentiation & Integration	19.	PDE Review, Electrostatics via Finite Differences
6.	Matrix Computing	20.	Heat Flow via Time Stepping
7.	Trial-and-Error Searching & Data Fitting	21.	Wave Equations I: Strings & Membranes
8.	Differential Equations; Nonlinear Oscillations	22.	Wave Equations II: Quantum Packets & E-M
9.	ODE Apps; Eigenvalues, Scattering, Projectiles	23.	Electrostatics via Finite Elements
10.	High-Performance Hardware, Parallel Computers	24.	Shock Waves and Solitons
11.	Applied HPC: Optimization, Tuning, GPU Coding	25.	Fluid Dynamics
12.	Fourier Analysis; Signals & Filters	26.	Integral Equations of Quantum Mechanics
13.	Wavelet & Principal Components Analyses	A.	Codes, Applets & Animations
14.	Nonlinear Population Dynamics	B.	Video Lecture Modules

science education shared ideas, learned about HTML and placed versions of their developments on the UCES computer, which resided on this new thing called the World Wide Web. It was clear from the start that HTML had problems with mathematics that LaTeX did not, and we will discuss that later. It was also clear to us from the start that there was a great potential in using hypermedia to enhance education. Specifically, enhanced engagement and interactivity provided by animations, sonification of data, simulations, and computer visualizations seemed to be ideal elements for education. And thus began our project to both develop computational physics educational materials and to simultaneously include the multimedia and interactive aspects of the Web into the developed educational materials.


Untitled
Configurations
Netscape: The Chaotic Pendulum And JAVA
Reload
Images
Print
Find
Stop
t.edu/~hans/PROJECT/CHAOS/index.html
Destinations
Net Search
People
Software
phase-space plot
phase-space plot
velocity
0.94
-0.95
angle
angle
Start
Stop
Clear
Rerun
some typical configurations
plain, undriven, undamped pendulum
undriven, underdamped pendulum

damping constant = 0.773
driving force = 0.52
driving frequency = 1
phase angle of driving force = 0
duration 50
delay 10
initial position = 0.8
initial position = 0.8

Figure 1: A screenshot of an applet written by Hans Kowallik that shows animations (top right) of two nearly-identical pendula, their corresponding phase-space diagrams (bottom right) and the control panel (left) used to adjust the pendula's parameters. Here they are starting off looking much the same, but in time they do diverge in behavior.

2. THE VISION

Learning is often hard and especially so for those who may not learn in the most common way. Diagrams, equations, multi-dimensional visualizations and sophisticated graphs are often not accessible for learners with sensory impairments or learning disabilities, and high-tech teaching seems may only to be making it worse [5]. Regardless of how clear we believe we are at explaining difficult concepts, it is important to realize that different learners learn in different ways, and that a reader's ability to form their own mental models of abstract concepts, such as those in physics, is essential for learning, and is improved by using multiple senses to "view" a subject [6, 7]. Accordingly, we want learners to be able to use multiple senses to interact with the materials they study and thereby get more parts of the reader's brain to get activated in the "viewings" [8]. Thus making materials more accessible for those with learning disabilities also makes them more accessible for all learners. And yet, as often seems to be the case in education, although we may believe things to be true, there seems to be little, if any, objective proof that teaching to each student's "learning style" actually increases the effectiveness of education [9].

As faculty members at research universities we felt that part of our jobs was to push explore and push the boundaries as to what future textbooks might be able to do. Yet as researchers we knew that failures were more common than successes. But also that writing and running simulations often led to a greater understanding than just reading, and that there seems to be no better way to learn the computing part of a computational science than by sitting down at a computer in a trial–and–error mode [10]. And so a digital textbook seemed particularly appropriate for a computational science course, and we started exploring ways in which a textbook might contain various types of interactive and executable elements that a reader can use without leaving the text.

For example, the lab parts of our computational courses involved programs. Accordingly, our Computational Physics text is full of sample programs, and we wanted a reader to be able to look at a program's listing in the text and be able to execute the code right there and then. The reader would then be able to create their own versions of the figures in the text or explore the effects of making changes in the program. This type of interaction engagement with the text would be a boom for learning. For example, running a simulation and a visualization of how two identical realistic pendula with minuscule differences in initial conditions eventually leads to quite different behaviors (Figure 1), provides an understanding of the "butterfly effect" and of abstract phase-space diagrams (bottom part of Figure 1) that words alone cannot provide.

But why stop with flat 2-D figures and animations within a text? Why not include 3-D surface and volume realizations as well? Likewise, if animations can be included, why not include actual videos of demonstrations, lectures and physical events like soliton tsunamis? In fact, if we include lectures and labs as part of the "book" then the book becomes more like an encapsulated entire course from which a reader can read, watch, interact *etc* in whatever mix the reader (or their instructor) prefers.

While we usually associate "reading" a text with sight, the words and equations in a digital text can be converted to speech, which clearly would be of value to readers with sight impairment. Yet even beyond that, there might be value in "visualizing" data with sound, a process known as *sonification*[11]. For example, converting the numerical results of a simulation of spontaneous decay (Figure 2) into sound produces the familiar sound pattern of a Geiger counter [12]. To us this was a convincing demonstration of how the sense of hearing convinces the reader to truly believe that the decay simulation is a realistic description of nature. Likewise, being able to hear a linear, nonlinear and chaotic oscillations (Figure 3) might deepen the understanding of what to many students might be abstract concepts regarding nonlinearity and Fourier decompositions.

Scientific texts tend to have many diagrams, some simple line drawing and, increasingly, multicolored 3–D visualizations. These diagrams are great learning aides for some readers, but can be very difficult to visually–disabled readers or for learners who have trouble "seeing" the point of complicated diagrams. While simple line drawings can be printed out on braille-like devices, that will not work for complicated figures.

One solution to this problem might be the creation of figures in a format such as Scalar Vector Graphics (SVG) in which the figures can have various visual and textual layers, and in which these layers can be peeled away progressively to provide simpler diagrams or explanations of the figures. If we have a digital book that permits hypertext and multimedia, there is no reason why the figures have to be static.

Game consoles are big business now and their force feedback joysticks permit both user input to the game as well as a haptic response from the game. It would be great to have a similar device, often called a "haptic mouse", that would permit a reader to feel a figure as well as the output of simulations. For example, learning might well be enhanced by being able to feel your way around a force field, feeling the depths of holes or peaks, or to move and feel your way around a macromolecule feeling for missing pieces. Of course these types of figures would have to contain move than "flat images", but that does not seem much different from the interactive 3-D images that users can rotate and resize.

One of the ways in which science has been able to advance through the years has been by creating mathematical descriptions of natural systems, and then manipulating the mathematics. Many of us find that manipulating the equations is much easier than trying to figure out the consequences of a theory using just words and concepts. In addition, once you learn how to view them, the equations of mathematics are beautifully compact and meaningful objects that can be manipulated to provide new understanding. Consequently we believed that a reader should be able to see beautifully displayed equations and to work with them, for example by using symbolic manipulation programs such

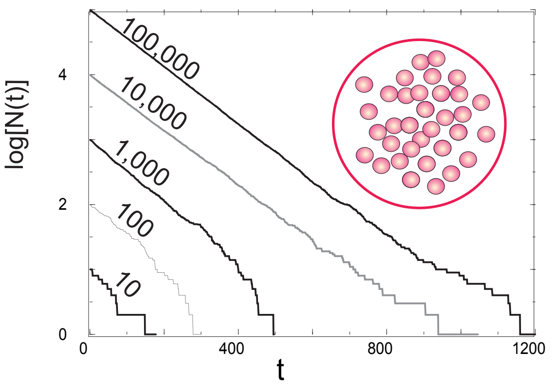


Figure 2: The output of a spontaneous decay simulation for five different numbers of initial nuclei. Note how the decays look exponential when there are a large number of nuclei still existing, but becomes stochastic when there is approximate 100 or fewer nuclei. (In some sense, the bottom four curves are just translated portions of the top–most curve.)

as Sage, Maple or Mathematica. However, right from the start it was clear that there was a problem with including mathematics in HTML Web documents. Early Web editors and converters displayed equation as bit maps, which contain no information regarding the equation's meaning and lost clarity if the page was scaled up in size. If the contents of equations were retained in digital documents they could be exported to manipulation programs or understood by *reader* programs for learners with disabilities. For example, the ASTER program [13] can read LaTeX source and generate a spoken equation with variations in pitch indicating sub- and superscripts. In addition, if one can extract a meaningful representation of an equation from a document, then it would be possible to search the document or the Web for other uses of this same equation.

One solution to using math in digital documents is to write the document in a "notebook" format appropriate to a specific problem–solving environment such as Mathematica, Maple or Sage [14]. This of course restricts the computer language that may be used, may require the reader to run such an environment on their local computer, and suffers from frequent obsolescence as upgrades are made to the environment. A better solution seemed to be to write the equations in *MathML*, a macro package for the *Extensible Markup Language*, XML. Being a markup language, like LaTeX the content of the equation is not lost, and being a Web language, it is possible to display the equation properly in Web documents and even to search on the equation.

3. REALIZATIONS

First Realization Our first experiments with eBooks were in the late 90's and were computational Physics and Unix tutorials that attempted to illuminate both physics and computation (examples at [15]). We tried everything we could think of: sound files, animations (first gifs and then movies), CGI-bin scripts, haptic feedback devices and even a Unix terminal to our computer so users could run Unix from their local browsers (this was before security was such a big issue).

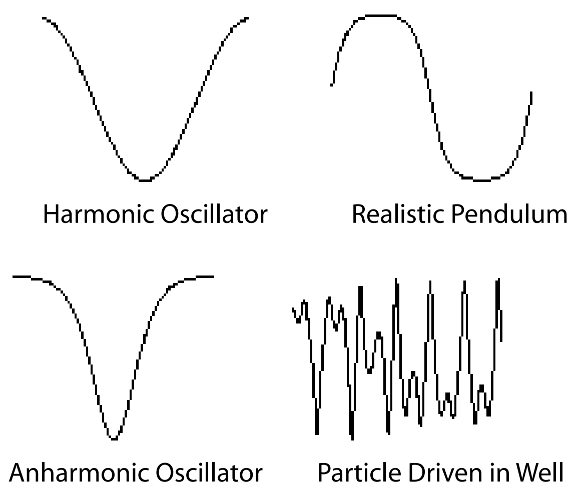


Figure 3: ***From upper left to lower right:*** **The linear oscillation of a harmonic oscillator, the nonlinear oscillations of a realistic pendulum and an anharmonic oscillator, and the chaotic oscillations of a particle in an anharmonic well driven by an external force. The eBook contains different sounds associated with each of these oscillations.**

For example, Figure 2 shows the results from a simulation of spontaneous decay of nuclei. Each of these graphs was converted into a sound file (which are still on the Web at [12]) and, sure enough, when played the results of the simulation do sound like a Geiger counter.

As part of developing understanding of nonlinear dynamics and chaos, which were just then entering the physics curriculum, and of learning how to solve ordinary differential equations, extensive discussions of oscillations were prepared with the oscillations converted into visualizations and sound files. For example, in Figure 3 we show the graphical representations of the output of four simulations of linear and nonlinear oscillations, and in the Web document [11] we have each oscillation coupled to a corresponding sound files. Although the linear oscillation sounds rather simple and boring, the nonlinear oscillations have overtones and sound more interesting, while the chaotic oscillation sounds like noise. In this way we are using an additional sense to understand some abstract ideas.

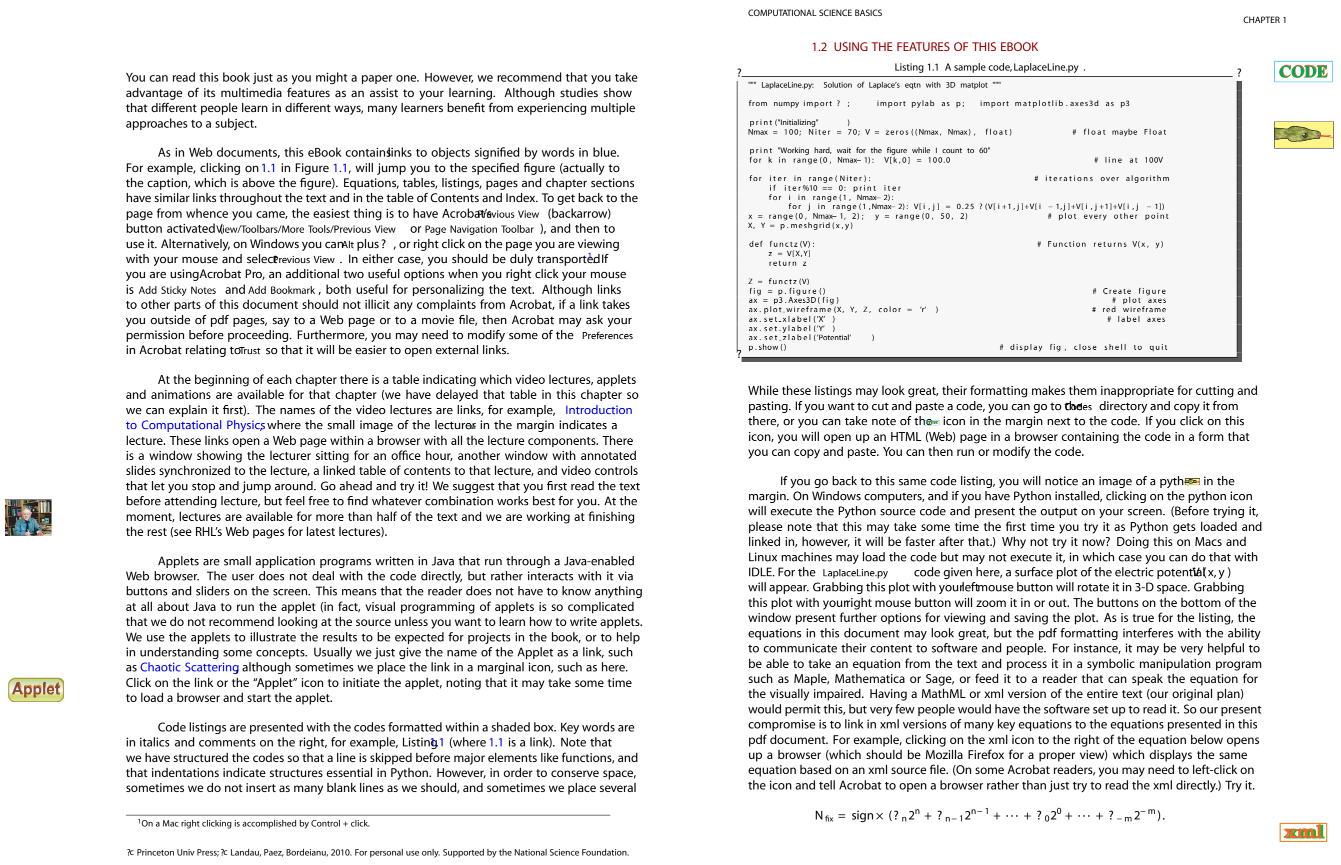
Since computational science is very much about developing and running programs to do science and engineering, we have developed many sample codes (first in Fortran, then in C, then in Java and most recently in Python). Yet running codes within an electronic document has its challenges (more on that soon). Accordingly, soon after the 1995 introduction of Java applets that ran within a browser on a user's computer, our group developed collections of applets based on our codes. The emphasis was on user engagement and interactivity, important elements for education. While this is more secure for us than having users run code on our machines, the programming of graphics and interactive features all designed to operate within a browser was highly labor intensive, and there is a security risk for the user. An sample applet has already been shown in Figure 1 where the user controls the parameters of the simulation and can chose several means to view the system.

Second Realization Our second effort to produce an eTextBook came in 2007, right after Princeton University Press published (in paper) our text *A Survey of Computational Physics* [16]. Although that book used Java for it example programs, by the time the book was published we had already moved to the Python ecosystem [17] as our preferred environment for teaching computational science, and so we wanted to develop a Python version of our text as well. After our editor explained how the economics of book publishing would not permit another version of our text until after it was proven to be a universal best seller, we decided that this might be the perfect time to create a Python version as an eTextBook. With the business model for electronic publishing even less clear then than it is now, the publisher agreed to our posting a free online version our book on Merlot and Compadre [18]. Yet as two of us moved closer to retirement, we still wanted to see a commercial eTextBook that would represent an acceptance and an institutionalization of our vision.

Even though the most senior of the coauthors had his heart set on a MathML/XML document, his coauthors were successful at convincing him that the eBook format should use Adobe's portable document format, pdf. First, the pdf format was essentially universal, whereas a MathML format would be accessible to only a few. Secondly, our text was already written in LaTeX and there existed the *hyperref* LaTeX macro package [19] that automatically created internal hyperlinks for all equations, figures, book sections *etc* as well as internal and external hyperlinks to multimedia. The hyperref package also permitted the execution of code within the created pdf final document, which in the process could produce interactive 3–D visualizations of a type available only at supercomputer centers when we started this project. And so we created pdf versions of our Computational Physics text with links to sound files, animations, applets, video lecture modules, executable codes, a dynamic table of contents and external sites. As a partial realization of our vision for executable equations, many of the text's equations were linked to corresponding *MathML* versions which were ported over to and executed within symbolic manipulation programs such as *Maple* or *Mathematica.*

A two-page spread from the pdf eBook is shown in Figure 4, where you can note the icons in the margins that are actually links: a professor's head to a video lecture module, an "Applet" tag to a Java-based applet, a "CODE" tag that links to a text version of the sample code that could be pasted into a file for execution (copying the pdf version of the code may not preserve Python's essential indentations), a snake head that will execute a Python code, and an "xml" tag that brings up a MathML version of the equation, which can be imported into a symbol manipulation program. Although the text did not have an automatically-linked dictionary, it did link many specialized words to both a Glossary and to a sound files that read out the word's meaning without the reader having to take his or her eye off the text.

If all of this sounds too good to be true, well it was. Shortly after we posted free copies of our text on Compadre, Merlot, CSERD and the OSU science server, Adobe decided that executable code within a pdf document is a security risk and modified their pdf readers so as to block the code's execution. The reader could still copy the code from



You can read this book just as you might a paper one. However, we recommend that you take advantage of its multimedia features as an assist to your learning. Although studies show that different people learn in different ways, many learners benefit from experiencing multiple approaches to a subject.

As in Web documents, this eBook contains links to objects signified by words in blue. For example, clicking on 1.1 in Figure 1.1, will jump you to the specified figure (actually to the caption, which is above the figure). Equations, tables, listings, pages and chapter sections have similar links throughout the text and in the table of Contents and Index. To get back to the page from whence you came, the easiest thing is to have Acrobat's Previous View (backarrow) button activated (View/Toolbars/More Tools/Previous View or Page Navigation Toolbar), and then to use it. Alternatively, on Windows you can Alt plus ? , or right click on the page you are viewing with your mouse and select Previous View . In either case, you should be duly transported.[1] If you are using Acrobat Pro, an additional two useful options when you right click your mouse is Add Sticky Notes and Add Bookmark , both useful for personalizing the text. Although links to other parts of this document should not illicit any complaints from Acrobat, if a link takes you outside of pdf pages, say to a Web page or to a movie file, then Acrobat may ask your permission before proceeding. Furthermore, you may need to modify some of the Preferences in Acrobat relating to Trust so that it will be easier to open external links.

At the beginning of each chapter there is a table indicating which video lectures, applets and animations are available for that chapter (we have delayed that table in this chapter so we can explain it first). The names of the video lectures are links, for example, Introduction to Computational Physics, where the small image of the lecturer in the margin indicates a lecture. These links open a Web page within a browser with all the lecture components. There is a window showing the lecturer sitting for an office hour, another window with annotated slides synchronized to the lecture, a linked table of contents to that lecture, and video controls that let you stop and jump around. Go ahead and try it! We suggest that you first read the text before attending lecture, but feel free to find whatever combination works best for you. At the moment, lectures are available for more than half of the text and we are working at finishing the rest (see RHL's Web pages for latest lectures).

Applets are small application programs written in Java that run through a Java-enabled Web browser. The user does not deal with the code directly, but rather interacts with it via buttons and sliders on the screen. This means that the reader does not have to know anything at all about Java to run the applet (in fact, visual programming of applets is so complicated that we do not recommend looking at the source unless you want to learn how to write applets. We use the applets to illustrate the results to be expected for projects in the book, or to help in understanding some concepts. Usually we just give the name of the Applet as a link, such as Chaotic Scattering, although sometimes we place the link in a marginal icon, such as here. Click on the link or the "Applet" icon to initiate the applet, noting that it may take some time to load a browser and start the applet.

Applet

Code listings are presented with the codes formatted within a shaded box. Key words are in italics and comments on the right, for example, Listing 1.1 (where 1.1 is a link). Note that we have structured the codes so that a line is skipped before major elements like functions, and that indentations indicate structures essential in Python. However, in order to conserve space, sometimes we do not insert as many blank lines as we should, and sometimes we place several

[1] On a Mac right clicking is accomplished by Control + click.

© Princeton Univ Press; © Landau, Paez, Bordeianu, 2010. For personal use only. Supported by the National Science Foundation.

COMPUTATIONAL SCIENCE BASICS

CHAPTER 1

1.2 USING THE FEATURES OF THIS EBOOK

Listing 1.1 A sample code, LaplaceLine.py .

CODE

```
""" LaplaceLine.py:  Solution of Laplace's eqtn with 3D matplot """

from numpy import ? ;     import pylab as p;    import matplotlib.axes3d as p3

 print ("Initializing"        )
Nmax = 100; Niter = 70; V = zeros((Nmax, Nmax),  float)            # float maybe Float

 print "Working hard, wait for the figure while I count to 60"
 for k in range(0, Nmax-1):   V[k,0] = 100.0                             # line at 100V

 for iter in range(Niter):                                  # iterations over algorithm
     if iter%10 == 0: print iter
     for i in range(1, Nmax-2):
         for j in range(1,Nmax-2): V[i,j] = 0.25 ?(V[i+1,j]+V[i -1,j]+V[i,j+1]+V[i,j -1])
x = range(0, Nmax-1, 2);   y = range(0, 50, 2)                     # plot every other point
X, Y = p.meshgrid(x,y)

 def functz(V):                                              # Function returns V(x, y)
     z = V[X,Y]
     return z

Z = functz(V)
fig = p.figure()                                                         # Create figure
ax = p3.Axes3D(fig)                                                         # plot axes
ax.plot_wireframe(X, Y, Z, color = 'r'  )                              # red wireframe
ax.set_xlabel('X'  )                                                       # label axes
ax.set_ylabel('Y'  )
ax.set_zlabel('Potential'     )
p.show()                                               # display fig, close shell to quit
```

While these listings may look great, their formatting makes them inappropriate for cutting and pasting. If you want to cut and paste a code, you can go to the Codes directory and copy it from there, or you can take note of the CODE icon in the margin next to the code. If you click on this icon, you will open up an HTML (Web) page in a browser containing the code in a form that you can copy and paste. You can then run or modify the code.

If you go back to this same code listing, you will notice an image of a python in the margin. On Windows computers, and if you have Python installed, clicking on the python icon will execute the Python source code and present the output on your screen. (Before trying it, please note that this may take some time the first time you try it as Python gets loaded and linked in, however, it will be faster after that.) Why not try it now? Doing this on Macs and Linux machines may load the code but may not execute it, in which case you can do that with IDLE. For the LaplaceLine.py code given here, a surface plot of the electric potential $V(x,y)$ will appear. Grabbing this plot with your *left* mouse button will rotate it in 3-D space. Grabbing this plot with your *right* mouse button will zoom it in or out. The buttons on the bottom of the window present further options for viewing and saving the plot. As is true for the listing, the equations in this document may look great, but the pdf formatting interferes with the ability to communicate their content to software and people. For instance, it may be very helpful to be able to take an equation from the text and process it in a symbolic manipulation program such as Maple, Mathematica or Sage, or feed it to a reader that can speak the equation for the visually impaired. Having a MathML or xml version of the entire text (our original plan) would permit this, but very few people would have the software set up to read it. So our present compromise is to link in xml versions of many key equations to the equations presented in this pdf document. For example, clicking on the xml icon to the right of the equation below opens up a browser (which should be Mozilla Firefox for a proper view) which displays the same equation based on an xml source file. (On some Acrobat readers, you may need to left-click on the icon and tell Acrobat to open a browser rather than just try to read the xml directly.) Try it.

$$N_{fix} = \text{sign} \times (?_n 2^n + ?_{n-1} 2^{n-1} + \cdots + ?_0 2^0 + \cdots + ?_{-m} 2^{-m}).$$

xml

Figure 4: A two-page spread from the pdf version of the eTextBook with marginal icons indicating hyperlinks to lectures, codes, code execution, applets and MathML equations.

the text and run it separately, but this took them away from the text. Indeed, the problem of code execution is not our's alone; for example, Elsevier Publishers would also like to see more flexibility with electronic documents and so sponsor an annual Executable Paper Challenge [20]. At present the concept seems best fulfilled with Mathematica and Maple players that contain their own execution kernel [14]. In addition, while the use of applets has served us well for approximately 15 years, the extra security requirements on present-day browsers keep making existing applets so difficult to run or so obsolete that it seems impractical to include them in an eBook.

The second and present realizations of our eTextBook contain some 60 **video-based lecture modules** that cover most every topic in the printed book and that took some five years to complete production. (In addition to problems within the text, there are also quizzes on the lectures to encourage their viewing before attempting the problems.) These modules can be used as a replacement for lectures so that instructors can interact with students in the computer lab, as a way to stimulate new courses, or for online education. As seen in Figure 5, each module opens a Web page containing a video picture–in–picture of a professor discussing and demonstrating the material in his office, coordinated dynamic slides (sometimes with red scribbling on them), a dynamic table of contents, and links to codes and applets. Having the lectures produced in a studio setting with controlled sound, lighting, movement and video monitors is a great improvement in quality compared to that of "live" lectures where the subject moves and often faces away from the viewer and the sound-level varies too much.

After viewing a number of online lectures including some from the Great Lectures Collection [21], we concluded that a scripted "lecture", while sounding polished with its few speaking errors, is rarely as engaging as a professor speaking informally from slides with all the imperfections that occur when a knowledgeable person thinks and speaks about a subject they obviously care about. There is, however, a price to be paid for many high-quality videos, namely, they take up a lot of space, in our case over 14 GB. Although this presented a problem when we imagined including the lectures on a DVD, the development of cloud computing has provided a solution.

Third Realization Technology does not stand still, and while we waited for Princeton Press to decide if they could publish an eTextBook, eBook readers and mobile computing devices became very popular. And so we started to try out different formats for eReaders, tablets and smart phones. First we tried converting our book to the native Kindle format ePub. Although the text converted well, we could not get the formatting of the equations and tables to be acceptable.

We next explored alternative pdf versions of the book with different margin and page sizes. These could be made to look good, but there were issues with hyperlinks and the use of Flash within the lecture modules on both Android and Applet devices (recently Android devices appear able to run Flash within some browsers). And of course we could get neither the Python codes with their ecosystem libraries nor the Java-based applets to execute. Finally, in a multistep and cumbersome process we produced an Applet iBook. As read on an iPad, this realization had some interactivity and hyperlinks, but was overly restrictive in what was permitted and was still far from what we would view as acceptable for a computational text book. Clearly, the PC was needed for full interactivity, while mobile devices were useful for reading, but not executing.

3.1 Present Realizations

After spending six years trying unsuccessfully to get Princeton University Press to distribute what might have been the most complete eTextBook available at that time, in late 2013 we reached an agreement with John Wiley & Sons, an international scientific, technical, medical, and scholarly publishing house, to publish a Python-based Third Edition of our *Computational Physics* text in paper, in a "flat" online edition as well as in an interactive, multimedia eTextBook version [3] that will appear in the Wiley Online Library [22]. The paper and flat versions are scheduled to be published in the summer of 2015, while the multimedia version is being prepared simultaneously with the writing of this paper.

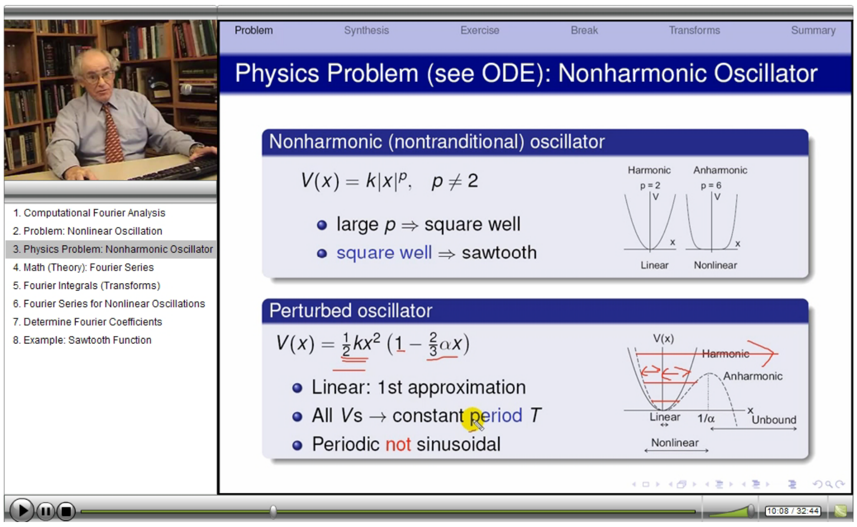
This latest version of our eTextBook starts with our LaTeX source file and then uses LaTeXml and LaTeXmlPost to create XML and to xhtml/HTML5 versions of the text [23]. This software is free and has been developed by the US National Institute of Science and Technology as part of their development of the *Digital Library of Mathematical Functions.* Although going from one markup language to another should be straightforward, it takes some effort to get the software to work properly, and especially so because LaTeXml does not understand all of the LaTeX style (.sty) files that are used in creating a book with a particular publisher's look and style. As one might expect, there is much hand work needed to convert the included multimedia to modern HTML5 standards: animations as mp4's, figures as SVG's, applets to animations, sound files as mp3 (MPEG-1 and MPEG-2 Audio Layer III), *etc.* Furthermore, we still need to determine whether the equations will be in Presentation MathML, or, given the current state of various browsers, whether we will use the "polyfill" library MathJax [24], which ends up rendering the equations with LaTeX.

4. EVALUATION AND ASSESSMENT

Evaluating cyber-enabled learning is a grand challenge problem, with multiple variables involved as well as a society in which the role of computers and communications is seeing a historically rapid change. For example, we are just now working with students who view electronic engagement and interactions as completely normal, who read news online more than from paper, and who buy, and presumably read, more electronic books from Amazon than paper ones.

Our eTextBook evolved from our paper textbooks, which in turn evolved from over a decade's worth of class notes for our Computational Physics courses. A number of professors have taught the courses, and the texts have been used throughout the world at some 50 schools that we know of. The courses and lecture modules have had formative and summative assessment via student evaluations, pre- and post-course student inter-views and surveys conducted by an external evaluator. The texts have had pre- and post-publication reviews, market surveys conducted by several publishers, feedback from faculty and students using the materials at other schools, and multiple discussions with collaborators.

The instructors for the blended course both had the im-



1. Computational Fourier Analysis
2. Problem: Nonlinear Oscillation
3. Physics Problem: Nonharmonic Oscillator
4. Math (Theory): Fourier Series
5. Fourier Integrals (Transforms)
6. Fourier Series for Nonlinear Oscillations
7. Determine Fourier Coefficients
8. Example: Sawtooth Function

Physics Problem (see ODE): Nonharmonic Oscillator

Nonharmonic (nontranditional) oscillator

$V(x) = k|x|^p, \quad p \neq 2$

- large $p \Rightarrow$ square well
- square well $\Rightarrow$ sawtooth

Perturbed oscillator

$V(x) = \frac{1}{2}kx^2\left(1 - \frac{2}{3}\alpha x\right)$

- Linear: 1st approximation
- All Vs $\rightarrow$ constant period T
- Periodic not sinusoidal

Figure 5: A screenshot of a video lecture module. On the left under the talking prof is a dynamic table of contents, and on the right is a slide getting scribbled on.

pression that the students were better prepared for lab than when we had live lectures, although this may be due in part to our requirement that the students view the lectures before the lab. The instructors also concluded that the students' projects were at least of the same quality as before, with student questions and discussion of higher quality. In recent times Landau has taught workshops at various schools and at the SuperComputing conferences using and distributing the eTextBook. A number of schools have used the developing forms of the eText and have provided feedback.

The detailed external assessment surveyed how much use each of the different features of the book received and how helpful was each. Here are some of the results: 1) The students used the free eTextBook about 80% of the time and the paper one about 20%. All students used the eText some of the time. 2) About 40% of the students did not read the instruction on how to use the text. 3) Hyperlinks to figures, equations and codes were used about 50% of the time, with technical problems rare. 4) There was a rather even distribution in all ranking categories as to the usefulness of the applets. 5) None of the students bothered printing pages from the eTextBook, although code listings were printed some 10% of the time. 6) 100% of the students thought the written pages of the text were essential or fairly essential. 7) 90% of the students thought the lecture modules were essential or fairly essential. 10) The code listings were considered essential. 11) Running and seeing the results of simulations were considered essential or fairly essential by 90% of the students. 12) The eTextbook was rated 4 stars out of 5.

5. SUMMARY AND CONCLUSIONS

We have traced the history that produced our vision of an eTextBook, as well as the history of the various practical realizations of that vision and how those realizations changed as the technology advanced in unexpected ways. Part of our motivation has been to support the advancement of computational science by ensuring that there are commercially–supported textbooks. This may encourage others to introduce computational science courses, or to improve existing courses. While it seems like a PC is at present the best way to interact with an eTextBook including its multimedia and executable content, mobile devices would still be fine for reading the text. However, the latest commercial version of our text is in HTML5, and we are waiting to see how that version works on the increasingly popular mobile device

6. REFERENCES

[1] Scientific Computing and Computational Physics Courses, `"/COURSES/`. Here and below we use " to abbreviate `http://physics.oregonstate.edu/ rubin`.

[2] R.H. Landau, Computational Physics for Undergraduates, the CPUG Degree Program at Oregon State University. *IEEE Computing in Sci & Engr, 6, 68-75*, (2004); `"/CPUG/`.

[3] R. H. Landau, M. J. Paez, & C.C. Bordeianu, `"/Books/CPbook/; http://www.wiley-vch.de/publish/en/books/ISBN3-527-41315-4/`.

[4] The Department of Energy Undergraduate Computational Engineering and Science (UCES) program, `http://www.krellinst.org/collaborations/project-archives`.

[5] C. Fabis, As High-Tech Teaching Catches On, Students With Disabilities Can Be Left Behind, Chronicle of Higher Education, 25 February 2015.

[6] J. M. Carroll, and J. R. Olson. Mental Models in Human—Computer Interaction. M. Helander, Ed., *Handbook of Human Computer Interaction*, Amsterdam, North Holland, 1988;
C. Dede, M. Salzman, R.B. Loftin and D. Sprague. Multisensory Immersion as a Modeling Environnment for Learning Complex Scientific Concepts, *Computer Modeling and Simulation in Science Education*, eds. N. Roberts, W. Feurzeig, W. and B. Hunter, New York, Springer–Verlag, 1999;
M.J. Farah, K.M. Hammond, D.N. Levine and R. Calvanio. Visual and spatial mental imagery, *Cognitive Psychology*, 20, 439-462, 1988.

[7] National Research Council. *How People Learn.* M. S. Donovan, J. D. Bransford and J. W. Pellegrino Eds., National Academy Press, Washington DC, 1991.

[8] M.C. Salzman, C. Dede, R.B. Loftin and J. Chen. A Model for Understanding How Virtual Reality Aids Complex Conceptual Learning, *Presence*, 8(3) 293-316, 1999.

[9] A. North, Are 'Learning Styles' a Symptom of Education's Ills?, New York Times, 25 February 2015.

[10] P. Davis, How Undergraduates Learn Computer Skills, *T.H.E Journal*, 26, 69, 1999, `http://ywww.thejournal.com/articles/14120`; O. Hazzan and J.E. Tomayko. Reflection and Abstraction in Learning Software Engineering's Human Aspects, *IEEE Computer*, 39-46, 2005; National Science Foundation. Shaping the Future: New Expectations for Undergraduate Education in Science, Mathematics, Engineering, and Technolgy. `http://ywww.nsf.gov/publications/pub_summ.jsp?ods_key=nsf96139`, 1996; J.Larkin. The role of problem representation in physics, 75-98, *Mental Models*, D. Gentner and A. Stevens Eds. Lawrence Earlbaum Associates, Hillsdale, NJ, 1993.

[11] Visualizing Physics With Sound, Direct Transformation, `"/nacphy/ComPhys/SOUND/`.

[12] Spontaneous Decay Simulation, `"/nacphy/ComPhys/SOUND/geiger.html`.

[13] T.V. Raman. *AsTeR: Audio System for Technical Readings.* `http://yeasi.cc/itd/volume1/number4/article2.html`.

[14] See, for example, Mathematica's Computabale Document Format, `http://www.wolfram.com/cdf/`; The Maple Player, `http://http://www.maplesoft.com/products/maple/mapleplayer/`.

[15] Northwest Alliance for Computational Science, Landau's Nacphy Research Group, `http://physics.oregonstate.edu/ rubin/nacphy/`.

[16] R. H. Landau, M. J. Paez, & C.C. Bordeianu, *A Survey of Computational Physics, Introductory Computational Science*, Princeton Univ Press, Princeton, NJ, 2008.

[17] *The Python Ecosystem*, March/April 2011 and May/June 2007 issues of COMPUTING IN SCIENCE & ENGINEERING, `http://www.computer.org/portal/web/cise/home`.

[18] Merlot, Multimedia Educational Resource for Learning and Online Teaching, `http://ywww.merlot.org/merlot/viewMaterial.htm?id=604391`; Compadre, Physical Science Resource Center, `http://ywww.compadre.org/psrc/items/detail.cfm?ID=11578`.

[19] The LaTeX `hyperref` package, `http://yen.wikibooks.org/wiki/LaTeX/Hyperlinks`, 2012.

[20] *The Executable Paper Grand Challenge*, `http://http://www.executablepapers.com/`.

[21] The Teaching Company, *The Great Courses.* `http://ywww.teach12.com`.

[22] Wiley Online Library, `http://onlinelibrary.wiley.com/`.

[23] LaTeXML A LaTeX to XML/HTML/MathML Converter, `http://dlmf.nist.gov/LaTeXML/`.

[24] MathJax, A JavaScript display engine for mathematics that works in all browsers, `http://www.mathjax.org/`.