FEASIBILITY OF INTERACTIVE ETEXTBOOKS WITH COMPUTATIONALLY INTENSE CONTENT

Jacques C. Richard, Logan N. Collins, Kristi J. Shryock, John D. Whitcomb
Aerospace Engineering Department
Texas A & M University

John Edward Angarita
Civil Engineering Department
Columbia University

Abstract

We evaluate the technical feasibility of creating pedagogically valuable, highly interactive content in eTextbooks for the purpose of education in computationally intense fields. This research was motivated by the observation that emerging eTextbook technologies could help enhance the education of engineering students. Engineers often want to experiment and to be able to quickly see meaningful results. They want to receive immediate feedback or response for their inputs. They want interactive learning tools. Engineers want trial-and-error with a realistic system, with which they can interact, even if it is a virtual one. The most interactivity in many eTextbooks is clicking links, resizing and rotating images, or pausing/playing audio/video. Currently, emerging technologies associated with eTextbooks, and eBooks in general, are approaching a developmental level where it is possible to provide realistic virtual systems embedded in an eTextbook environment that could help build students' physical intuition. Since students may wish to interact with simulations in real-time, one of our feasibility tests involved the real-time rendering and simulation of different example cases of fluid flows within a sample eTextbook chapter. The simulation comes with controls the student can use to manipulate key flow parameters to see the response of the flow field to student inputs. Ultimately, the goal is to enable the creation of universal learning tools where the student can selectively use the interactive modules needed to complete said student's education and physical intuition.

Introduction

Modern Technology-savvy Engineering Students and eTextbooks

Consider the modern technology-savvy engineering student going through an eTextbook and interacting with an embedded simulation of concepts of interest. The student's technological background most likely includes experience with video games but how much of a ”gamer” is uncertain. However, student technological background can vary. The modern student has likely used the internet - searching, blogging, instant sharing, instant messaging, social media, etc. Different students may expect different levels of interaction from an eTextbook with embedded simulations but first, what is the technological readiness level to produce such an eTextbook?

In this paper, we examine the feasibility of highly interactive eTextbooks with computationally intense content to enhance student learning and understanding, and help build engineering intuition. The level of interaction with simulation embedded within the eTextbook should enhance pedagogy. An eTextbook with embedded simulations should help the professor trying to get the engineering concept across and help the student make the engineering connection.

While the degree of technological background of the modern engineering student can vary, the impact of technology in the education environment cannot be underestimated. Studies have shown that games have considerable impact on training pilots [1-5] and surgeons [6], building language-learning skills [7], etc. These studies suggest that the technological capabilities of the modern engineering student should be incorporated into the learning environment of said student. In fact, the technological background of the student should be more and more integrated into the modern engineering curriculum and pedagogy.

Motivation of Emerging eBook Technology

It may appear to be self-evident what the purpose of e[lectronic]Books (eBooks) and electronic publication (EPUB and often written as simply ePub) are: to be able to peruse through a book on electronic devices like tablets, laptops, desktops, etc. One can zoom, scroll, rotate, etc. One can have embedded audio and video. Better still is having links to web resources. Even better is the ability to search for key items of interests. Modern engineering textbooks emphasize the strategic use of colors, sketches, figures, plots, chapter outlines and summaries, realistic situations, etc., to improve pedagogy [8-11]. Furthermore, the tools for producing eBooks should not distract the professor from pedagogy [12-14. The eBook readers should not be distracted either.

The eTextbook concept should not interfere with emerging plans for transforming undergraduate engineering education [15-20]. An eTextbook with embedded simulations may complement changes envisioned throughout undergraduate engineering curricula but assessing that is beyond the scope of this study.

We cannot take for granted that the engineering eTextbook should offer more than the abilities of eBooks mentioned earlier. Indeed, why should the plot just change size or orientation? Why not change the parameters and see the plot change? Why just watch physically realistic situations? Why not interact with them? Being able to interact provides a just-in-time (JIT) teaching moment that should not be missed [21-25].

The advent of the smart-phone and tablet includes many applications or ”apps” available for them. These are tools with which the modern technology-savvy engineering student may be familiar. In fact, given the existence of many software packages for engineering analyses that have migrated from desktops to mobile devices such as tablets and smart-phones, there may also be simulations that can be embedded within an eTextbook to enable the student to interact with plots, sketches, physically realistic situations, etc. Engineers already have a wealth of simulation tools at their disposal. The question then is can they be embedded in an eTextbook in a manner that enhances pedagogy?

The key here is to embed the simulations in the eTextbooks as opposed to remote simulations over the internet or cloud [26,27], virtual laboratories [28-31], or remote control of real experiments using the internet [32,33,] etc. This could allow some independence from the internet just like existing textbooks do not have to depend on the internet. This has its advantages and disadvantages, but the students would have access to the eTextbook just like access to print textbooks, the most important difference, that we wish to exploit, is that eTextbooks can have a lot more interactivity, like simulations. Print textbooks are large and sometimes difficult to attain while eTextbooks can be downloaded. Print textbooks do not require anything else while eTextbooks require a computer or similar systems. Print textbooks can be easier to manage while eTextbooks require specific types of maintenance and support (such as battery life, easily broken, etc.) It is simple to take notes in the print version, but it can be a bit more difficult to write notes on the eTextbook although that is also a desirable capability that we wish to expand for the students. In this paper, we focus on the feasibility and technological readiness level for bringing an eTextbook with embedded simulations to fruition.

Approach

Engineering Field Needs

Engineers often want to experiment and to be able to receive immediate feedback or response per their inputs. They want interactive analysis tools. Engineers want to perform trial-and-error experiments with a realistic system, with which they can interact, even if it is a simulation of a real system.

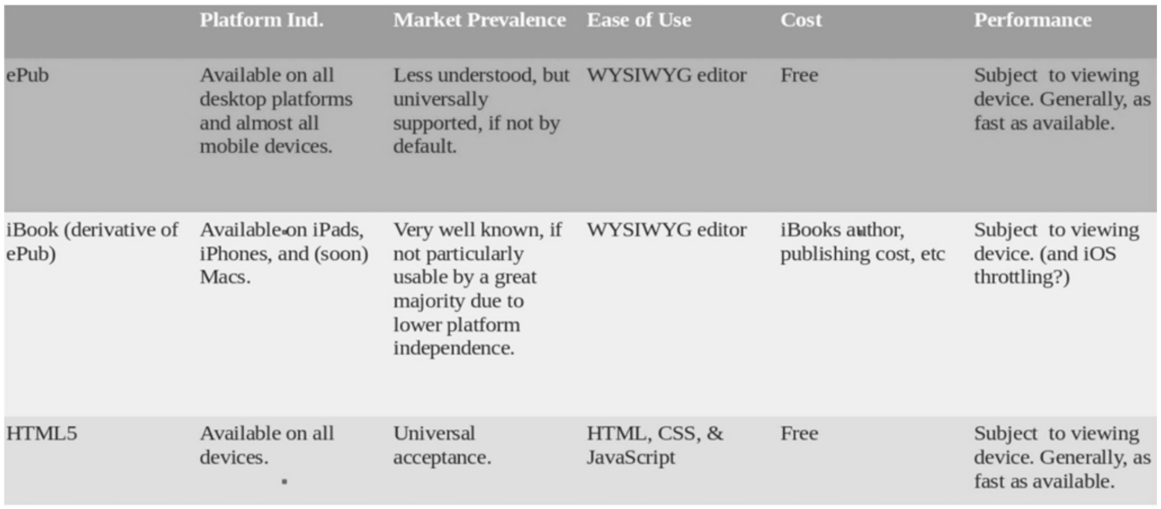
Many current engineering analysis tools are, as intended, best for analyses and not education. None of the analysis tools easily allow incorporation of simulations for education with an eTextbook. The popular MATLAB is heavily used in engineering and is integrated with some textbooks but as a separate software, e.g., MATLAB source code is provided to use with MATLAB [34–37]. Mathematica offers a Computable Document Format (CDF), akin to the Portable Document Format (PDF) used for computer representation of static textbooks [38,39]. That has a considerable learning curve even for those already familiar with Mathematica. It requires a CDF reader, a feature that can help protect proprietary simulation code. It is not as polished as a typical textbook and still needs some refinement. Maple and Maple TA are also promising but have similar advantages and disadvantages [40–43]. Maple TA, in particular, provides a very promising system for assessing the actual learning of the student[43]. Scientific Workplace has been very good in integrating use of LATEX [44], which is already familiar to many scientists and engineers, and some of Maple, in a desktop publishing environment [45]. However, the level of interaction is not there despite some use of a Maple compute engine. Furthermore, it is optimized for only one operating system which is limiting.

Since the ultimate goal is the creation of a learning environment valuable to students in a computationally intense field, it is necessary to consider what students want. Our sample chapter was for an Aerospace textbook so the needs of Aerospace engineering students were primarily considered.

Current eBook Technologies

eBook Formats

The important factors considered for the eTextbook format were Platform Independence, Market Prevalence, Ease of Use, Ease of Creation, Cost, and, perhaps most importantly, Performance. Very quickly it was narrowed down to four formats, of which the resulting winner was the ePub 3 format (see Figure 1).



	Platform Ind.	Market Prevalence	Ease of Use	Cost	Performance
ePub	Available on all desktop platforms and almost all mobile devices.	Less understood, but universally supported, if not by default.	WYSIWYG editor	Free	Subject to viewing device. Generally, as fast as available.
iBook (derivative of ePub)	Available on iPads, iPhones, and (soon) Macs.	Very well known, if not particularly usable by a great majority due to lower platform independence.	WYSIWYG editor	iBooks author, publishing cost, etc	Subject to viewing device. (and iOS throttling?)
HTML5	Available on all devices.	Universal acceptance.	HTML, CSS, & JavaScript	Free	Subject to viewing device. Generally, as fast as available.

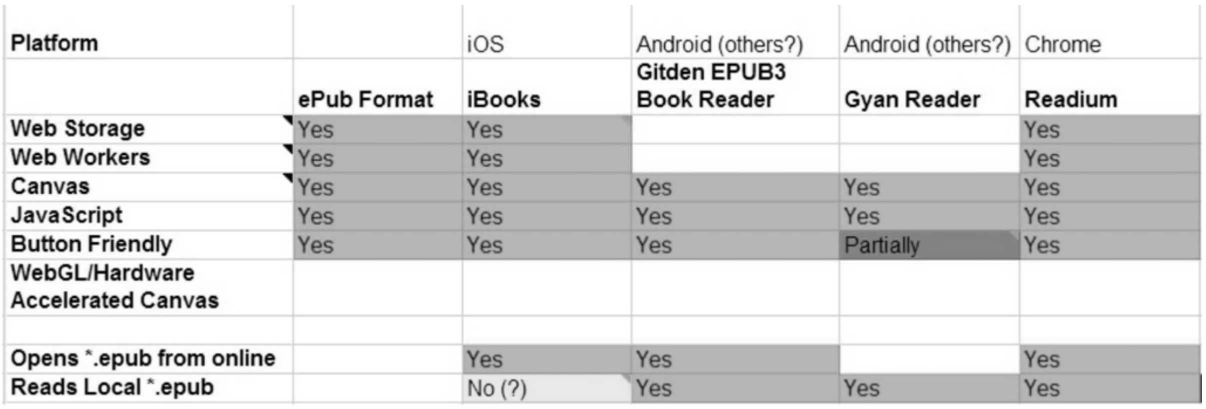
Figure 1: eBook Format Comparison.

The four formats that stood out were: ePub 3, iBooks, Kindle Format 8 (KF8), and a custom hyper-text markup language (HTML) implementation [46–51]. An initial example began with iBooks. However, the only way to create content was through iBooks Author which had extremely limited options for content with the level of interactivity desired. For example, while you could provide an HTML widget, it could not be directly embedded in the page. Instead, it was treated as a picture-link on the margin. At this point, we were aware that iBooks is a derivative of the ePub 3 specification. As such, we eliminated iBooks in favor of just adhering to the ePub 3 specification.

The Kindle Format 8 (KF8) was quickly eliminated due to several technical restrictions. It was limited to a small subset of Kindle devices, as well as having a file size cap that would make it virtually impossible to embed all of the media and content that a self-contained interactive eBook would require. Further, JavaScript[1] (JS) [52] and Canvas[2] [53], the two components required for anything beyond the simplest Cascading Style Sheets (CSS)[3][54] interaction, particularly simulations, were not available.

eReader Support

Figure 2 shows the technologies supported by the eReaders that students would likely be using. While the ePub 3 specification supports a huge feature set, only a small number of clients supported even a fraction of its features at the time of this work[4]. A large majority of available readers are still using ePub 2 specifications which support relatively little more than text, images, and a small amount of CSS [54]. A large number of readers were tested for support of the basic features we needed. Many readers were discarded at this step, while a few went on to further testing. Another small fraction were not discarded, per se, but could not be tested as they did not allow side-loading[5] of ePubs through a non-official source (i.e., Google Play Books). Initially, the reader thought to be most compliant was iBooks, however we discovered the open source reference implementation of the Readium Software Development Kit (SDK) [55,56]. There was not enough time for full testing, but it appears to be as ePub 3 compliant as iBooks, if not more so despite being aesthetically rough around the edges.



Platform		iOS	Android (others?)	Android (others?)	Chrome
	ePub Format	iBooks	Gitden EPUB3 Book Reader	Gyan Reader	Readium
Web Storage	Yes	Yes			Yes
Web Workers	Yes	Yes			Yes
Canvas	Yes	Yes	Yes	Yes	Yes
JavaScript	Yes	Yes	Yes	Yes	Yes
Button Friendly	Yes	Yes	Yes	Partially	Yes
WebGL/Hardware Accelerated Canvas					
Opens *.epub from online		Yes	Yes		Yes
Reads Local *.epub		No (?)	Yes	Yes	Yes

Figure 2: Many readers were still using ePub 2 specifications at the time of the writing of this paper.

[1]JavaScript, not Java [52]. This is a common mistake.
[2]An HTML5 element that allows raster-based, scriptable (through JavaScript) graphics.
[3]Cascading Style Sheets - a way to style structured web content.
[4]as in a client process (e.g., a computer running a browser following a specification) initiates a connection to a remote server where the software resides. However, the server may follow a different specification. Some setups have the client and server on the same computer.
[5]loading electronic files, data or app via what used to be via a card on the side of a computer but is more and more done wirelessly between any types of devices.

Once we decided to move forward with the ePub 3 format, we found that not all of the readers tested passed all the tests for feature support. The minimum features required for *real* interactivity (JavaScript, Canvas, and Document Object Model (DOM) Manipulation[6] [57]) were either absent or incomplete in the majority of the readers tested [51].

The features that iBooks provided that many others did not were a mostly complete JavaScript environment with DOM manipulation, Web Storage[7] [58], and Web Workers[8] [51]. Only slightly less important than JavaScript, the Canvas tag was not supported in a large majority of readers. These two feature sets are the minimum required to make effective interactive eTextbooks. Without Canvas, certain simulations and visualizations would simply not have the performance necessary to run[9].

An additional oddity that had to be checked was whether the readers supported input tags well. While iBooks supported them, it did not bring up a virtual keyboard for typing into text input fields. Buttons, sliders, radio buttons, and check boxes all function well enough, but not text input fields. This is one of the few things we were not able to overcome. Even attempting to manually get focus in a text field and type via a JS touch keyboard failed.

This was not an issue with the Readium Chrome Web App. As an almost fully featured reference implementation across all desktop platforms, it bodes well for the future of ePub 3 readers. While ePub3 support is currently poor, with only two mostly effective readers, it should only get better with time. It will get better even quicker if content begins to be made for ePub 3.

Developing an interactive eTextbook is actually no more difficult than developing individual interactive web pages designed with the mobile browser in mind. This is not to say that web development is particularly easy, but it is further along. As such, efficient ePub creation is nothing more than an extension of that process (see Figure 3).

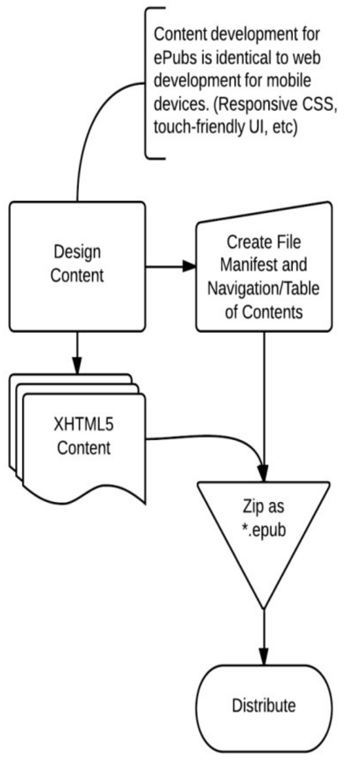

Content development for ePubs is identical to web development for mobile devices. (Responsive CSS, touch-friendly UI, etc)
Design Content
Create File Manifest and Navigation/Table of Contents
XHTML5 Content
Zip as *.epub
Distribute

Figure 3: ePub Flow.

Developing a Sample eTextbook Chapter

Needs of Interactive, Computationally Intense Content

At the time of the writing of this paper, the central processing units (CPU) of most tablets are adequate for two-dimensional (2D) 64 × 64 computational fluid dynamics (CFD) simulations, as desired. The flow simulator or virtual wind tunnel embedded in the sample eTextbook chapter displays from known

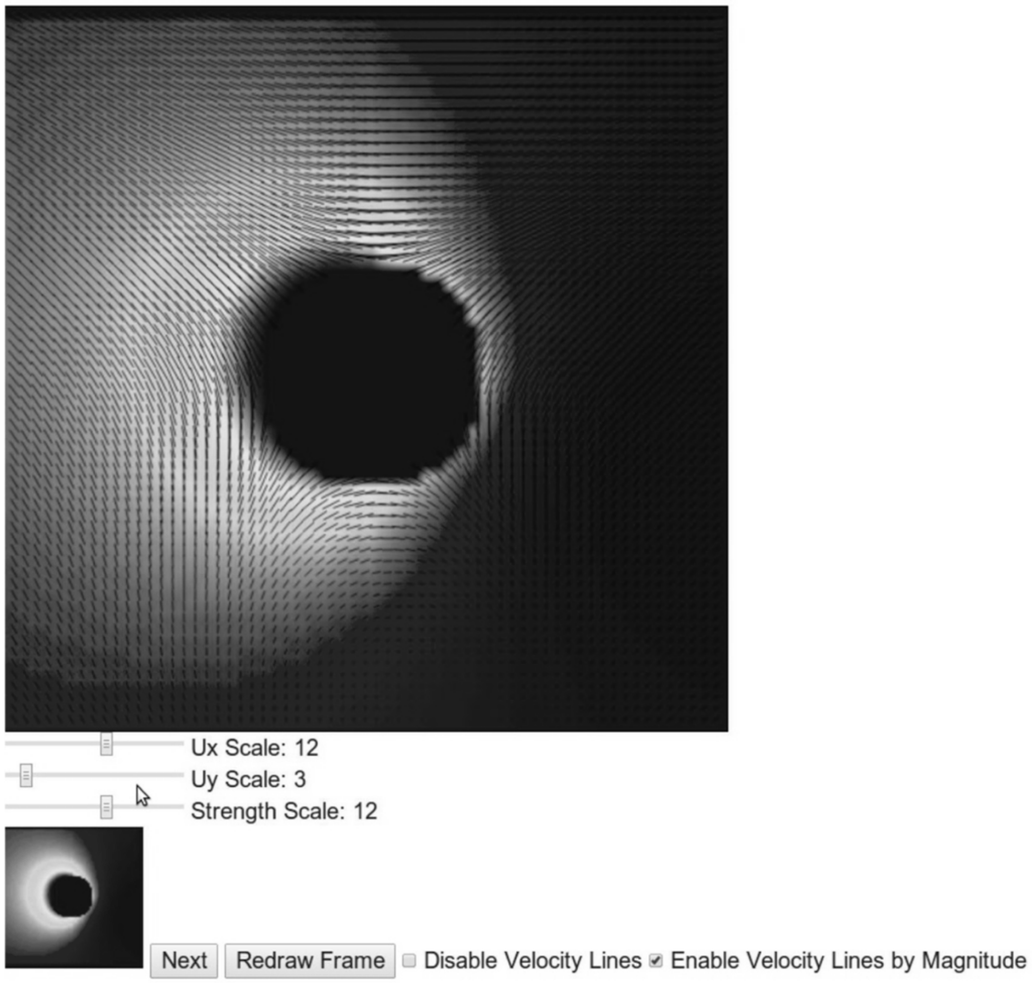
[6]Manipulation of elements of a web page programmatically using JavaScript.
[7]Web Storage is a new, simple JavaScript API.
[8]JavaScripts that perform computationally expensive task in the background without interrupting the user interface.
[9]This is due to the fact that manipulating the DOM tree incurs significant overhead in addition to calculating and displaying the graphics. On the other hand Canvas is essentially a blank window of pixels without any DOM overhead.

results of classical textbook closed-form solutions and is simulated by a 2D CFD solver. The CFD solver uses the Lattice-Boltzmann method (LBM) [59–65]. The LBM solver was originally implemented in C++ then ported to JavaScript for integration into an eTextbook. The simulation then ran like any other embedded JavaScript code (see Figure 3). The simulation embedded in the eTextbook is shown in Figure 4.

Server vs. Client Computation

As will be demonstrated through our performance benchmarks, it is viable to port existing code to JavaScript in order to achieve high-performing computations on the client machine. There are two viable techniques to port to JavaScript: a handwritten port or a compiled-to-JavaScript port.

Since the majority of modern computationally intense code will be written in C/C++, a handwritten port is quite simple. With a rudimentary understanding of JavaScript and C/C++, anyone can port the code, and quickly, due to the syntax similarities. This is our suggested method of embedding existing code bases due to the small file-size footprint and resulting fast code. Also, this enables the JavaScript engines to take advantage of their years of work on optimizing human written code during the execution cycle. This optimization is not always available with compiled-to-JS techniques.


Ux Scale: 12
Uy Scale: 3
Strength Scale: 12

Disable Velocity Lines ✓ Enable Velocity Lines by Magnitude

Figure 4: Flow simulator or “virtual wind tunnel” embedded in eTextbook chapter, showing velocity vectors and colored pressure contours for cross-flow over a circular cylinder with vorticity.

If, however, the existing codebase is written in a language too dissimilar to JavaScript to be easily ported (say, Fortran) and there is no desire to rewrite the code in a new paradigm, the code can be compiled to JavaScript using a relatively new tool called Emscripten [66–68]. Emscripten works by compiling Low Level Virtual Machine (LLVM) bytecode to JavaScript [69–71], normally into a subset of JavaScript called asm.js.[10]

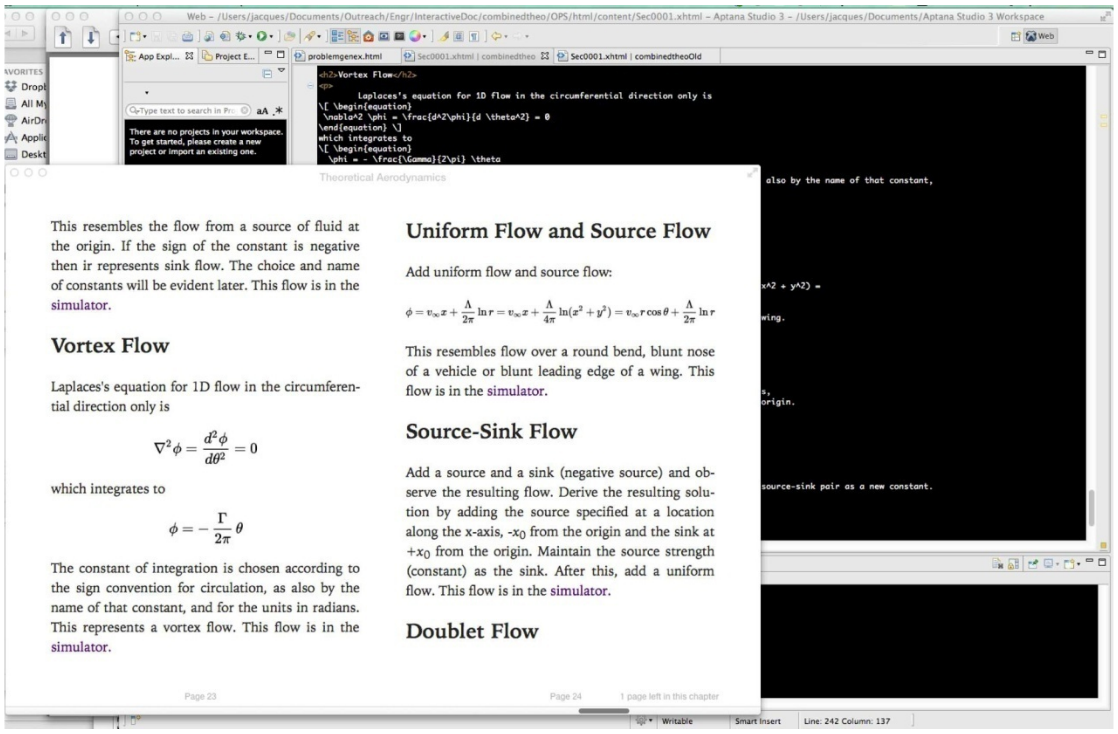
Now, the unfortunate consequences of compiling LLVM bytecode to JavaScript is that it results in extremely bloated, inefficient, and completely incomprehensible JavaScript code. It still runs fast, almost as fast as the handwritten code with no optimizations by the front-end compiler (usually the gcc C/C++ Compiler), but it takes up several times as much disk space. Further, the API exposed to the JavaScript code in the ePubs can be awkward depending on how easily the code is compiled to JavaScript. In short, there are many caveats for the price of not reimplementing the original algorithm.

In an educational setting, we suggest reimplementation of the algorithms in JavaScript. This way they are ensured to be quick and can be made available to curious students who wish to delve deeper into the subject that way.

Results

Sample Chapter

To demonstrate the efficacy of ePubs for the more usual content of a textbook, a sample chapter was made. Figure 5 shows the creation of a sample chapter. The word ”simulator” in the sample chapter is a live link to the interactive ”virtual wind tunnel” in Figure 4, placed after each case discussed, so that a student may go right to it as a just-in-time teaching moment. In the sample chapter, the main things used are CSS accordions that the students can open for more details on a subject, as desired, the embedding of


Web – /Users/jacques/Documents/Outreach/Engr/InteractiveDoc/combinedtheo/OPS/html/content/Sec0001.xhtml – Aptana Studio 3 – /Users/jacques/Documents/Aptana Studio 3 Workspace

There are no projects in your workspace. To get started, please create a new project or import an existing one.

```
<h2>Vortex Flow</h2>
<p>
        Laplaces's equation for 1D flow in the circumferential direction only is
\[ \begin{equation}
 \nabla^2 \phi = \frac{d^2\phi}{d \theta^2} = 0
\end{equation} \]
which integrates to
\[ \begin{equation}
  \phi = - \frac{\Gamma}{2\pi} \theta
```

Theoretical Aerodynamics

This resembles the flow from a source of fluid at the origin. If the sign of the constant is negative then ir represents sink flow. The choice and name of constants will be evident later. This flow is in the simulator.

Vortex Flow

Laplaces's equation for 1D flow in the circumferential direction only is

$$\nabla^2 \phi = \frac{d^2\phi}{d\theta^2} = 0$$

which integrates to

$$\phi = -\frac{\Gamma}{2\pi}\theta$$

The constant of integration is chosen according to the sign convention for circulation, as also by the name of that constant, and for the units in radians. This represents a vortex flow. This flow is in the simulator.

Page 23

Uniform Flow and Source Flow

Add uniform flow and source flow:

$$\phi = v_\infty x + \frac{\Lambda}{2\pi}\ln r = v_\infty x + \frac{\Lambda}{4\pi}\ln(x^2+y^2) = v_\infty r\cos\theta + \frac{\Lambda}{2\pi}\ln r$$

This resembles flow over a round bend, blunt nose of a vehicle or blunt leading edge of a wing. This flow is in the simulator.

Source-Sink Flow

Add a source and a sink (negative source) and observe the resulting flow. Derive the resulting solution by adding the source specified at a location along the x-axis, $-x_0$ from the origin and the sink at $+x_0$ from the origin. Maintain the source strength (constant) as the sink. After this, add a uniform flow. This flow is in the simulator.

Doublet Flow

Page 24 1 page left in this chapter

Figure 5: Part of the sample chapter demonstrating HTML editing of document with LATEX and MathJax used for equations and hyperlinks to the embedded flow simulator (key word).

[10]asm.js is a subset of JavaScript that allows for JavaScript engines to execute it quicker than normal.

instructional HTML5 video elements, and the use of LATEX mathtype through MathJax [72] put together via an ePub editor [73].

Finally, there is the displaying of math in the chapter. This is obviously an important feature. Originally we attempted to use MathML (which ePub 3 is supposed to support), but had great difficulty getting this to work. Having used LATEX and MathJax in a similar project [72], we tried it with great results. We used the Scalable Vector Graphics (SVG) [74] option in MathJax exclusively which allowed us to trim MathJax down by deleting some unused resources that come bundled with it.

Performance

The primary concern in creating interactive eBooks for computationally intensive applications is obviously performance. A test case computed is that of an inviscid, uniform, incompressible cross-flow over a circular cylinder with circulation. This case contains many calls to math functions and checks for cylinder boundaries to ensure that flow inside the cylinder is not shown. This case is called 3000 times on a laptop using Chrome's Developer's Console.

Port Version	Test Case	ms/call
Hand Written	2273.543s	0.757847
Emscripten Comp.	6098.333s	2.032777

Porting unoptimized C++ code into JavaScript yielded a module that feeds input to a (*canvas*) element to visualize the pressure and velocity fields of various fluid flow cases. Hand written code is roughly 2.7x faster than the default compiled code produced by Emscripten. This particular code could not be compiled with the optimization flags turned on due to certain limitations of Emscripten in dealing with pointers used in the C++ code. It is likely possible that if the original code was structured differently we would see near handwritten speeds when using optimization flags.

With the observed performance, the student could easily interact with 2D cases simulated in real-time at high resolution, particularly as hardware accelerated hcanvasi elements become more common across platforms. Even a 64^3 case would only take 48ms per call to complete (assuming 64x time for each 64^3 call). This may not be fast enough to be viewed in real-time, but given a small buffer period it could still be extremely pedagogically valuable.

In addition to the power available through JavaScript in the main thread, ePub 3 supports the Web Worker JS API which allows for threading of JavaScript applications [51]. In this way, it is possible to attain faster computation and increased rendering speeds through parallelization of computation.

Discussion

Open Questions

Simulation – Intuition Connection

Are we making better or more physically intuitive engineering students? Are we getting to an adequate level of engineering intuition faster or is it really slower? These types of questions will need to be answered by education professionals and by observing actual results of students using this technology.

How much content interaction

It is unclear how much content should be interactive. Many elements could be made interactive, but would they needlessly complicate the book? Further, how can you present all of the content to the student while not overwhelming them? i.e., how can we show them what they minimally need to learn and give them the freedom to choose when and how to interact with more?

How should they be placed

How can we optimally place content to reinforce the lessons from the texts? Should there be no more than one *featured* module for any given block of text and simply provide inter-eBook links to other optional modules?

How to handle non-compliance

The best thing an author can do to ensure readers are able to properly display content is to use native JavaScript where possible and native JavaScript polyfills[11] elsewhere. Authors should also take the opportunity to report these instances of non-compliance to the developers of the eReader to fix similar future non-compliances.

Adhering to an ePub specification should address non-compliance but that is not guaranteed [46–48]. Items like Canvas and advanced JS APIs are not necessarily required in the specifications. They may simply be incidental as the developer was trying to support the Web Engine that is used to display ePubs.

Some amount of JavaScript coding time may have to be invested on the part of the eTextbook author to ensure that the content is delivered in the manner that they feel properly goes with said content, or just plainly gets the concept understood. Authors would certainly prefer to spend most of their time on content. The amount of time is subjective as it depends on the author, the content, the level of interactivity, etc.

Optimizing Simulations

Due to the availability of Web Workers, calculations can be parallelized. For non-trivial calculations this is highly encouraged despite the additional complexity. Of course, optimizations should take into account whether they are on a mobile device or a desktop PC. A mobile device will not be able to sustain as many parallel threads as a desktop PC. In any case, it is also best practice for non-trivial computations to take place in a Web Worker so that the thread running the user interface (UI) is not blocked by the computation.

Conclusion

Interactive eBook technology is perfectly viable in computationally intense applications and offers great possibilities to ease education and cognitive barriers to learning in these fields. The available technologies exist that can be combined to produce an eTextbook with interactive embedded simulations of engineering concepts. Combining these technologies is not always easy. There are hurdles, variations in eReaders, with or without their platforms, versions, hardware/software systems or sub-systems, etc, but they may be overcome. Further, the ease of development will only increase as ePub 3 readers become more prevalent.

Future Work

Assessing Effectiveness

While it would seem that these techniques could only help students, testing with an educational professional is needed to hone in on areas that actually benefit from these interactive techniques. We acknowledge the needs to include student user feedback, learning outcomes, etc., but this paper primarily was investigating what was feasible. An eTextbook needs to be tested in a classroom and compared to a classroom that uses conventional textbooks to quantify eTextbook effectiveness. Other evaluations and assessments would also need to be conducted. We need to ensure that 1) we do not distract the student and negatively impact their education and 2) do not sacrifice a thorough understanding of the subject for a minor increase in physical intuition. We also want to eventually assess if such an eTextbook is complementing other efforts for transforming undergraduate engineering education.

[11]using native JavaScript to mimic missing native JS API's.

Suggestions for Technological Developments/Developers

Interactive eBooks on a large scale can be looked at as a special class of web app. It should be possible to design a general purpose framework for designing interactive eBooks. This could greatly enhance the quality of code and speed of development during creation. The real key is for future ePub developments to avoid an engineering eTextbook author having to become an ePub expert over the engineering content expert.

Acknowledgments

This work was conducted under the auspices of the National Science Foundation (NSF) Research Experiences for Undergraduates (REU) grant EEC-0552951.

References

1. R Andy McKinley, Lindsey K McIntire, and Margaret A Funke. Operator selection for unmanned aerial systems: comparing video game players and pilots. *Aviation, space, and environmental medicine*, 82(6):635–642, 2011.

2. Tricia Mautone, V Alan Spiker, and M Ron Karp. Using serious game technology to improve aircrew training. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*, volume 2008. NTSA, 2008.

3. Johnny E Triplett. The effects of commercial video game playing: a comparison of skills and abilities for the Predator UAV. Technical report, 2008.

4. Winston Bennett Jr, Brian T Schreiber, and Dee H Andrews. Developing competency-based methods for near-real-time air combat problem solving assessment. *Computers in Human Behavior*, 18(6):773–782, 2002.

5. Daniel Gopher, Maya Weil, and Tal Bareket. The transfer of skill from a computer game trainer to actual flight. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 36, pages 1285–1290. SAGE Publications, 1992.

6. Nancy J Hogle, Warren D Widmann, Aku O Ude, Mark A Hardy, and Dennis L Fowler. Does training novices to criteria and does rapid acquisition of skills on laparoscopic simulators have predictive validity or are we just playing video games? *Journal of surgical education*, 65(6):431–435, 2008.

7. Yolanda Rankin, Rachel Gold, and Bruce Gooch. 3d role-playing games as language learning tools. In *Proceedings of EuroGraphics*, volume 25, pages 211–225, 2006.

8. John D. Anderson. *Fundamentals of Aerodynamics (Series in Aeronautical and Aerospace Engineering)*. McGraw Hill, 2010.

9. Michael J Moran, Howard N Shapiro, Daisie D Boettner, and Margaret Bailey. *Fundamentals of engineering thermodynamics*. Wiley. com, 2010.

10. Ferdinand Pierre Beer. *Vector mechanics for engineers: statics and dynamics*. Tata McGraw-Hill Education, 2009.

11. RC Hibbeler. *Engineering Mechanics-Statics And Dynamics, 11/E*. Pearson Education India, 2009.

12. Maria Langer. *iBooks Author: Publishing Your First Ebook*. Flying M Productions, 2012.

13. Nellie McKesson and AdamWitwer. *Publishing with iBooks Author*. O'Reilly, 2012.

14. Karl Fb Payne, ALEXANDER Mc Goodson, Arpan Tahim, Heather J Wharrad, and Kathleen Fan. Using the ibook in medical education and healthcare settings-the ibook as a reusable learning object; a report of the author's experience using ibooks author software. *Journal of visual communication in medicine*, 35(4):162–169, 2012.

15. Louis J Everett, PK Imbrie, and Jim Morgan. Integrated curricula: Purpose and design. *Journal of Engineering Education*, 89(2):167–175, 2000.

16. L Katehi, Katherine Banks, H Diefes-Dux, D Follman, John Gaunt, Kamyar Haghighi, PK Imbrie, L Jamieson, R Montgomery,W Oakes, et al. A new framework for academic reform in engineering education. In *American Society for Engineering Education Conference, Salt Lake City, UT*, 2004.

17. Peter D Washabaugh, Leslie A Olsen, and JM Kadish. An experiential introduction to aerospace engineering. In *45th Aerospace Sciences Meeting*, pages 8–11, 2007.

18. Sven G Bil´en, Luis P Bernal, Brian E Gilchrist, and Alec D Gallimore. The student space-systems fabrication laboratory: Enhancing engineering education through student-run, real-world projects. In *ASEE-NCS 1999 Spring Conference, Pennsylvania State University Erie-Behrend, Erie, PA*, pages 68–72, 1999.

19. Xu Bing and Sun Haiquan. Construction and practice of t-cdio course system [j]. *Research in Higher Education of Engineering*, 2:008, 2009.

20. Karen Swan, Philip Vahey, Mark van't Hooft, Annette Kratcoski, Ken Rafanan, Tina Stanford, Louise Yarnall, and Dale Cook. Problem-based learning across the curriculum: Exploring the efficacy of a cross-curricular application of preparation for future learning. *Interdisciplinary Journal of Problem-based Learning*, 7(1):8, 2013.

21. Gregor M Novak. Just-in-time teaching. *New Directions for Teaching and Learning*, 2011(128):63–73, 2011.

22. Gregor M Novak, Evelyn T Patterson, Andrew D Gavrin,Wolfgang Christian, and Kyle Forinash. Just in time teaching. *American Journal of Physics*, 67:937, 1999.

23. GREGOR M Novak, EVELYN T Patterson, A Gavrin, and RC Enger. Just-in-time teaching: Active learner pedagogy with www. In *IASTED International Conference on Computers and Advanced Technology in Education*, pages 27–30, 1998.

24. Kathleen A Marrs and Gregor Novak. Just-in-time teaching in biology: creating an active learner classroom using the internet. *Cell Biology Education*, 3(1):49–61, 2004.

25. Adam P Fagen, Catherine H Crouch, and Eric Mazur. Peer instruction: Results from a range of classrooms. *The Physics Teacher*, 40:206, 2002.

26. Rocael Hern´andez Rizzardini, Byron H Linares Roman, Alexander Mikroyannidis, and Hans-Christian Schmitz. Cloud services within a role-enabled personal learning environment, in a. mikroyannidis, r. hernandez, h.-c. schmitz (eds.), workshop on cloud education environments. 2012.

27. Sylvana Kroop, Marcel Berthold, Alexander Nussbaumer, and Dietrich Albert. Supporting self-regulated learning in personalized learning environments. In *Proceedings of 1st International Workshop on Cloud Education Environments, Antigua, Guatemala*, pages 47–52, 2012.

28. Firoz Alam, Hao Tang, and Jiyuan Tu. The development of an integrated experimental and computational teaching and learning tool for thermal fluid science. *World Trans. on Engng. and Technology Educ*, 3(2): 249–252, 2004.

29. Shubhendu S Mukherjee, Steven K Reinhardt, Babak Falsafi, Mike Litzkow, Mark D Hill, David A Wood, Steven Huss-Lederman, and James R Larus. Wisconsin wind tunnel ii: a fast, portable parallel architecture simulator. *Concurrency, IEEE*, 8(4):12–20, 2000.

30. Gary Steven Strumolo and Viswanathan Babu. Method and system for providing a virtual wind tunnel, July 11 2000. US Patent 6,088,521.

31. Steve Bryson and Creon Levit. The virtual wind tunnel: An environment for the exploration of three-dimensional unsteady flows. In *Proceedings of the 2nd conference*

on Visualization'91, pages 17–24. IEEE Computer Society Press, 1991.

32. Euan Lindsay, Phil Long, and PK Imbrie. Workshop-remote laboratories: Approaches for the future. In *Frontiers In Education Conference-Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007. FIE'07. 37th Annual*, pages W1C–1. IEEE, 2007.

33. Denis Gillet and Georgios Fakas. emersion: A new paradigm for web-based training in engineering education. In *International Conference on Engineering Education*, pages 6–10, 2001.

34. Alexander Behrens, Linus Atorf, Robert Schwann, Bernd Neumann, Rainer Schnitzler, Johannes Balle, Thomas Herold, Aulis Telle, Tobias G Noll, Kay Hameyer, et al. Matlab meets lego mindstorms - a freshman introduction course into practical engineering. *Education, IEEE Transactions on*, 53(2):306–317, 2010.

35. David M Smith. *Engineering computation with MATLAB*. Pearson/AddisonWesley, 2008.

36. Dava J Newman. *Interactive aerospace engineering and design*. McGraw-Hill, 2002.

37. Len Colgan. Matlab in first-year engineering mathematics. *International Journal of Mathematical Education in Science and Technology*, 31(1):15–25, 2000.

38. Andreas Holzinger. Computer-aided mathematics instruction with mathematica 3.0. *Mathematica in Educa*, 1997.

39. F Olness. Integrating mathematica in the undergraduate science curriculum: Teaching computer literacy with mathematica. In *World Wide Mathematica Conf*, 1998.

40. Chvatalova Zuzana and Hrebicek Jiri. Education of economics with maple. In *Proceedings of the 11th WSEAS international conference on Applied informatics and communications, and Proceedings of the 4th WSEAS International conference on Biomedical electronics and biomedical informatics, and Proceedings of the international conference on Computational engineering in systems applications*, pages 435–440.World Scientific and Engineering Academy and Society (WSEAS), 2011.

41. Olga Caprotti. Webalt! deliver mathematics everywhere. In *Society for Information Technology & Teacher Education International Conference*, volume 2006, pages 2164–2168, 2006.

42. David Fisher. Review of maple ta. *MSOR Connections*, 4(4), 2004.

43. Andr´e Heck. Assessment with maple ta: creation of test items. *AMSTEL Institute, UvA, available online from Adept Scientific via: http://www. adeptscience. co. uk/products/mathsim/mapleta/MapleTA whitepaper. Pdf [Accessed 19 June 2008]*, 2004.

44. Leslie Lamport. *LATEX Document*, volume 14. pub-AW, 1994.

45. Malcolm Anderson, Lyn Bloom, Ute Mueller, and PENDER Pedler. Enhancing the teaching of engineering differential equations with scientific notebook. *International Journal of Engineering Education*, 16(1):73–79, 2000.

46. Matt Garrish. *What is EPUB 3?* O'Reilly Media, Inc., 2011.

47. Bill Kasdorf. Epub 3: Not your father's epub. *Information Standards Quarterly*, 23(2):4–11, 2011.

48. International Digital Publishing Forum. Epub 3 specification. URL http://idpf.org/epub/30.

49. Amazon. Kindle format 8. URL http://www.amazon.com/gp/feature.html?docId=1000729511.

50. Irene E McDermott. Ebooks and libraries. *Information Today*, 19(2):7–55, 2011.

51. Web Hypertext Application Technology Working Group. Html specification. URL http://www.whatwg.org/specs/web-apps/current-work/.

52. David Flanagan. *JavaScript: the definitive guide*. O'reilly, 2011.

53. W Winfield W Salisbury. Canvas, June 10 1952. US Patent 2,599,944.

54. Joel Sklar. *Cascading Style Sheets*. Course Technology Press, 2001.

55. Readium. Readium sdk. URL http://readium.org/projects/readium-sdk.

56. Marıa Blanca Ib´a˜nez and Carlos Delgado Kloos. Dynamic customization of etextbooks. *Alexander Mikroyannidis*, page 18.

57. World Wide Web Consortium et al. Document object model (dom), 2001.

58. Ian Hickson. Web storage. *Draft, W3C, October 4th 2011 http://dev. w3. org/html5/ webstorage/accessed on*, 4 (11), 2011.

59. J.C. Richard, B.M. Riley, and S.S. Girimaji. Magnetohydrodynamic turbulence decay under the influence of uniform or random magnetic fields. *Journal of Fluids Engineering*, 133:081205, 2011.

60. Benjamin M. Riley, Sharath S. Girimaji, and Jacques C. Richard. Magnetic field effects on axis-switching and instabilities in rectangular plasma jets. *Flow, Turbulence and Combustion*, 82(3), 2009. URL http://www.springerlink.com/content/j2kn8p11308671l0/fulltext.pdf.

61. Benjamin M. Riley, Jacques C. Richard, and Sharath S. Girimaji. Assessment of magnetohydrodynamic lattice Boltzmann schemes in turbulence and rectangular jets. *International Journal of Modern Physics C (IJMPC) Computational Physics and Physical Computation*, 18(8):1211 – 1220, August 2008. doi: 10.1142/S0129183108012881.

62. X. He and L.-S. Luo. Lattice Boltzmann model for the incompressible Navier-Stokes equation. *J. Stat. Phys.*, 88:927, 1997.

63. X. He and L.-S. Luo. A priori derivation of the lattice Boltzmann equation. *Phys. Rev. E.*, 55:R6333, 1997.

64. X. He and L.-S. Luo. Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation a priori derivation of the lattice Boltzmann equation. *Phys. Rev. E.*, 56:6811–6817, 1997.

65. H. Chen, S. Chen, and W. H. Mathaeus. Recovery of the navier-stokes equations using a lattice gas Boltzmann method. *Phys. Rev. A.*, 45:R5339, 1991.

66. Alon Zakai. Emscripten: an llvm-to-javascript compiler. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pages 301–312. ACM, 2011.

67. Jeff Terrace, Stephen R Beard, and Naga Praveen Kumar Katta. Javascript in javascript (js. js): Sandboxing third-party scripts. *USENIX WebApps*, 2012.

68. Christoph Schinko, Martin Strobl, Torsten Ullrich, and Dieter W Fellner. Scripting technology for generative modeling. *International Journal On Advances in Software*, 4(3 and 4):308–326, 2012.

69. Chris Lattner. Llvm and clang: Next generation compiler technology. In *The BSD Conference*, pages 1–2, 2008.

70. Chris Lattner and Vikram Adve. The llvm compiler framework and infrastructure tutorial. In *Languages and Compilers for High Performance Computing*, pages 15–16. Springer, 2005.

71. Chris Lattner and Vikram Adve. Llvm: A compilation framework for lifelong program analysis & transformation. In *Code Generation and Optimization, 2004. CGO 2004. International Symposium on*, pages 75–86. IEEE, 2004.

72. Davide Cervone. Mathjax: A platform for mathematics on the web. *Notices of the AMS*, 59(2):312–316, 2012.

73. Ben Martin. The aptana ide for ajax development. *Linux Journal*, 2007(157):15, 2007.

74. World Wide Web Consortium. Scalable vector graphics (svg) 1.1 specification. *W3C Candidate Recommendation*, 2, 2000.

Biographical Information

Dr. Jacques Richard got his Ph.D. at Rensselaer Polytechnic Institute, 1989 & a B. S. at Boston University, 1984. He was at NASA Glenn, 1989-1995, taught at Northwestern for Fall 1995, worked at Argonne National Lab, 1996-1997, Chicago State, 1997-2002. He is a Sr. Lecturer & Research Associate in Aerospace Engineering @ Texas A&M since 1/03. His research is focused on computational plasma modeling using spectral and lattice Boltzmann methods for studying plasma turbulence and plasma jets. His research has also included fluid physics and electric propulsion using Lattice-Boltzmann methods, spectral element methods, Weighted Essentially Non-Oscillatory (WENO), etc. Past research includes modeling single and multi-species plasma flows through ion thruster optics and the discharge cathode assembly; computer simulations of blood flow interacting with blood vessels; modeling ocean-air interaction; reacting flow systems; modeling jet engine turbomachinery going unstable at NASA for 6 years (received NASA Performance Cash awards). He is involved in many outreach activities: e.g., tutoring, mentoring, directing related grants (for example, a grant for an NSF REU site). He is active in professional societies (American Physical Society (APS), American Institute for Aeronautics and Astronautics (AIAA), etc.), ASEE, ASME. He has authored or co-authored about 25 technical articles (19 of which are refereed publications). He teaches courses ranging from first-year introductory engineering design, fluid mechanics, to space plasma propulsion.

Logan N. Collins is a graduating senior pursuing a B.S. of Applied Mathematics at Texas A&M, class of 2015. His primary research interests lie in Algebra, with a focus in Number Theory as applied to Cryptography.

Dr. Kristi J. Shryock is Assistant Department Head for Undergraduate Programs in the Department of Aerospace Engineering at Texas A&M University. She is also an Instructional Associate Professor in the Department. She received her Ph.D. in Interdisciplinary Engineering with a research focus on engineering education. She works to improve the undergraduate engineering experience through evaluating preparation in mathematics and physics, incorporating experiential activities in the classroom, and introducing multidisciplinary design.

John D. Whitcomb began his career at NASA Langley Research Center in 1974, where he stayed until moving to Texas A&M University in 1989. While employed at NASA, he completed his Masters and Ph.D. degrees at Stanford University and Virginia Polytechnic Institute and State University, respectively. His research has primarily focused on predicting the performance of composite materials subjected to static and fatigue loads. He has also explored non-mechanical behaviors, such as moisture diffusion, oxygen diffusion and subsequent oxidation, and permeability to cryogenic fuels. The overall goal is to predict the response of potential composite structures without the need for extensive experimental effort. He is also keenly interested in developing techniques for leveraging the power of symbolic and numerical computation to enhance learning opportunities for students.

Mr. John Edward Angarita, is a first year graduate student pursuing a PhD of Aerospace Engineering at Virginia Polytechnic Institute and State University. He graduated in the spring of 2014 from Columbia University with a B.S. in Engineering Mechanics. His research focuses on Space Situational Awareness with a focus on satellite formation flying.