Numerical Fluid Dynamics I-text Project Edit

Overview Edit

We are designing a new generation, highly interactive textbook (i-text) for use in MIT undergraduate and graduate courses in fluid mechanics (found in courses 1, 2, 8, 10, 12, 16, 18, 20, 22) and for learning numerical methods used for such problems, as well as providing a resource for students doing research. The i-text will go well beyond what is currently available, allowing:

- students to interact with virtual fluid experiments
- the textbook to evolve over time based on student input and feedback
- more appropriate interaction from touch enabled tablet and laptop devices as well as from desktop systems
- leading practitioners in the field to include state-of-the-art, high-performance-computing based research tools in teaching with relative ease
- users to examine, alter, and run codes

It will also provide a field test of an approach that could enhance OCW and the recent MITx initiatives and demonstrate a scalable approach to online interaction-intensive education.

Description Edit

While many excellent but conventional fluid mechanics texts exist, ones melding numerics and fluids are few. Ours will be unique in bringing MIT's forte – the *mens et manus* approach to teaching – into virtual fluids and numerical methods labs. The primary teaching goal is numerical methods for fluid problems, and the bulk of the "textbook-y" material will be aimed at that; however, many of the examples have a much broader use in teaching fluid dynamics in general. Most of us who teach fluid physics are constrained both in presenting full examples and in problem sets by the mathematical abilities of our students and the limitations of analytical methods. For example, we can teach about convection in the linear limit of high viscosity and conductivity, but without graduate-level math cannot really address the behavior when that limit no longer applies and the flow becomes nonlinear. But with a simulator, students can easily increase the forcing and see when the flow becomes unsteady and then turbulent. They can analyze the output to see how the heat flux depends on the forcing and other parameters. In short, they can really dig into the material and learn it.

Just as analytical work rapidly becomes difficult, computational approaches face a host of difficult issues such as numerical diffusion and dispersion, preservation of maximum and minimum values, formation of shocks, moving boundaries (e.g. surface waves), and resolution of the wide range of scales of motion typical in turbulent motions. Understanding the advantages and disadvantages of a variety of methods again benefits greatly from a hands-on approach, with the possibility for altering the code and observing what happens directly. This year, in the advising seminar 12.A32, we were looking at a discretized 1-dimensional heat diffusion equation which has a numerical instability if the time step is too large; the criterion appeared as a constraint during the setup, but of course they wanted to see what happened if it was violated. The "Oh Wow" response was gratifying proof that they understood immediately why the limitation on Δt was indeed a strict requirement; further experimentation showed that the rule was "less than" not "less than or equal to," something more difficult to see from the math. Having written many models ourselves, we realize the need to experiment with test cases to ensure the pieces work correctly and the necessity for looking carefully at both the theory and the actual behavior – the kind of experience an i-text can give.

To accomplish our goal, we need to establish the infrastructure for an open-source i-text and generate initial text and examples, bearing in mind that it is intended to grow and be extended. To this end, the text will be "wiki-fied," and the process of adding interactive codes and experiments likewise will be open (but with editorial oversight). Thus, we envision an on-going process of adding material as students, but also faculty here and elsewhere, add to the effort. The i-text will be provided on virtual machines, giving experiment developers a known environment, while making them readily available to students, using only a java-enabled browser or remote desktop client. The structure and platform we develop can serve as a basis for many other scientific i-texts.

Initially, we will target a mix of courses – 2.005 (Thermal-Fluids Engineering I), 12.010 (Computational Methods of Scientific Programming), 2.086 (Numerical Computation for Mechanical Engineers), 2.29 (Numerical Fluid Mechanics)and 12.800 (Geophysical Fluid Dynamics I)[†] to build some virtual fluid experiments or coding p-sets, to test them, and to get feedback from students.

Wiki approach Edit

We propose to take a user-editable ("wiki") approach to the text itself. This provides a means for the content to expand as faculty and students add or revise material. The text would be freely accessible, and registered users (students and TA's in a class as well as ones using it for help in research and graduate students) could generate new variants available to the whole class, modifying any of the content, text or interactive material. During a class, the various versions should all be accessible, prioritized by student ranking; incorporating the readers' viewpoints on how to make material clearer to the target audience will improve the i-text. We would exert editorial control, probably at the end of a semester and taking into account the student input, to decide what material and what versions are included going forward. This task could, in time, devolve to a larger group of primary authors.

Interactive Edit

Interactive textbooks are, of course, popular now. Sometimes the term simply means being able to examine three-D figures in detail, to rotate them and zoom in and out. In other cases, people have taken a more inclusive view: Wolfram, for example, has introduced a "Computable document format" allowing users to work with Mathematica within a book, changing parameters and even equations. We are aiming at something similar, but open source, not restricted to a single, proprietary, language and system. For us, interactive means:

- Figures which can be manipulated – zoomed, panned, rotated, changed to log axes, replotted, etc.
- Programs which can be run with user-selected parameters and configuration, so that students can explore, for example, the relationship between flow speed and numerical time and space discretization. Such programs also can be used in science/ engineering classes to explore how the physics changes as the flow speed increases and to investigate real problems such as turbulent heat transport. The codes and visualization routines would be included for studying but also for modification to fit the user's problem.
- Methods for user entry or modification of programs – from subroutines to full codes – which will then be compiled (if necessary) and executed. In addition to the obvious application to problem sets, working with a code (e.g. to switch time-stepping algorithms) can ensure that one really understands the scheme.

Platform Edit

At MIT, we can count on Athena to provide a consistent platform all students can use, although they are more and more relying on their own computers. We plan to build virtual machines which contain the text, the software, and the required system components. These could be run under VirtualBox or VMWare on the students' computers or provided on servers and accessed via desktop-sharing. For our initial effort, we will use some of our own machines as hosts for the VM's. (Non MIT users could use virtual machines on the Amazon cloud.) This approach gives content-providers a consistent environment. Many pieces would be downloaded from the main i-text server into the VM as needed. This development leverages off our NSF-funded "Cloud-computing Infrastructure Technology for Education" project, which has built a prototype of the kind of platform we think is needed. It has allowed us to make models ranging from simple Octave/ Matlab codes to the MITgcm (ocean-atmosphere general circulation model and Navier-Stokes code) accessible and usable. But for an i-text, we may need additional functionality (e.g. for working on-line with codes) and a different "look-and-feel."

[†] The latter two have occasional senior undergraduates; we hope to use these to engage some of the graduate students as contributors and critics. We have used preliminary versions of the platform with students in 12.804 (Large-scale Flow Dynamics Laboratory) for some time, with enthusiastic response.

Educational objectives Edit

Several MIT courses treat numerical methods for fluids, while fluid mechanics itself can be found in a wide variety of undergraduate courses throughout the Institute. Because exploration of the topic beyond the simplest, often unrealistic, cases requires either sophisticated applied mathematics or numerical methods, our i-text can be a primary text or a valuable resource for all of these. Like the I-Campus Fluid Mechanics modules (development of which seems to have stopped 8-10 years ago), our book would provide simulations of many different kinds of fluid motion, but with the crucial difference that the reader can alter the parameters freely or even the code itself. Rather than canned movies, we would favor hands-on, real-time calculations and display.

The i-text would also provide a resource for students whose research (UROP or senior thesis) involves fluid flows and who want to use numerical methods. They would have both codes which could be adapted to their problem and explanation of the principles behind them.

In our own courses, this project will be of growing importance as we use the examples and feed new ones into the i-text. sLikewise, we and our colleagues can point students to this i-text for aid in their research. We will be talking with other faculty members to entrain them into using and contributing to the effort.

Finally, we note that this approach seems to fit extremely well with the MITx initiative. The platform and i-text structure can be adapted to many fields in which computation and experimentation with computer models is important (which, these days, is a broad umbrella indeed). By putting servers and clients in the cloud, MITx can offload the maintenance and usage fees while maintaining control over the content. Using the cloud gives world-wide access with large numbers of users able to invoke their own instances of our server/ client images, to read the text, and to examine and use the models. Certainly, development of a community of learners will provide resources for MIT students as well as improving the i-text and examples.

We have prepared a Movie giving some idea how an interactive version might look compared to a standard text. If desired, we can post a few pages with live examples.

Usage [remote desktop version] Edit

Login Edit

You may already have done this, but the link is http://synoptic2.mit.edu/fg/numfl.html ; enter your username and the password. After some a short time, you should get a display of the workspace numer, IP address; it then waits for the machine to start, and gives instructions for access. If you get connection information without a machine name or IP address, the startup has failed. Otherwise follow the relevant instructions, and you should get window onto the remote estop showing a browser.

Note that these machines are transient; **all work you desire to keep should be copied back to some machine which you have an account on and which is running ssh**. The links on the opening page facilitate this by making a file `mitcite.tgz` containing any new material since you started the machine.

Alternatively, I may have given you a direct VNC/ Remote desktop port on `cite01.mit.edu`; those are either futuregrid or our own VM's. Again, you do want to save work from these, but they tend to be up for longer periods.

Navigation Edit

Click to the desired text as usual. The head page will have links to individual chapters. Do not use the left/right buttons. Navigate with the "breadcrumb" trail at the top of the pages.

When you click on the title page, it will check for updates; if there are such, you may be faced with update screens as described below.

When you click on a chapter it will show the PDF for reading. There will be four windows to deal with: the browser, the PDF, a window with an `octave>` prompt, and (when a plot is made) a graphics window.

Fluxbox window manager Edit

The window manager is set to rasie the window under the mouse and give it the focus. You can change this and other behavior using the right mouse button when the cursor is on the background and altering the configuration options. The bar along the bottom lists the windows; clicking on the name will raise it or iconify it. You can move windows by dragging the top bar with the left button doen. You can get a window menu by right clicking in the top bar. Resizing can be down with the box icon or bay dragging the lower left or lower right corner of the border with the left button.

The window manager is set to iconify the current window (F12) or push it back behind others (F4). You can raise the PDF (F1) or browser (F2).

A right-click on the background brings up a menu allowing you to open a terminal. Do not use the exit option here! The working directory may not be the home directory, so you should probably start with `cd`. A middle-click allows you to see what's iconified and raise it. The bottom bar also has buttons for individual windows.

Octave window Edit

You can type commands to the Octave interpreter here to view variables, do calculations, etc. Control-C is a bit flakey, but can be used in emergencies.

PDF window Edit

Mostly, you'll just uses the page up and page down keys, but you can look at the "?" help for more information. When you are done with the PDF, you can hit "q" to quit. For some reason, the browser thinks it is not done loading the PDF; you can bring the browser to the front and hit the escape "Esc" key or the stop loading button to tell it loading is ccmplete.

Interaction Edit

insert note here

The following list gives the kinds of interaction currently being used with some notes on points which might not be obvious.

- Movies If `mplayer` is being used, you can pause and resume with the space bar or quit with `q`.
- 2D plots You can find see the coordinates of the mouse at the bottom. The middle botton writes these onto the plot. If you drag a rectangle out with the right button down, you can zoom. The buttons at the top allow you to undo the zoom.
- 3D plots Dragging with the left button rotates the figure. Dragging with the middle button will stretch it in z.
- Animations Like movies, but produced real-time from a program. You can't do much with the animation (perhaps cancel it with a Control-C in the octave window if you can get to it). But you can edit the program which produces it.
- Short-form parameter variations This brings up two windows, one with a command-line and the graphics window showing the result of the command. You can edit the command line and hit return to see the changes. Suggestion: try `f=0.1` or `f=10` as examples. This plot shows the height and into-the-page velocity. Hit the "Quit" button when you are done – this is important; otherwise the window will remain active even if it is hidden by another one.
- Long-form In this case, the browser appears with a new tab holding a more extensive input regiion. You can try an initial height of `0.3*x.*gs(x,y)` or velocity using Octave/Matlab syntax. This is set to run longer, so you may wish to use the "Stop" button. You can close the tab when done. Hit "F1" to bring the PDF back to the foreground.
- Programs You can also work directly with codes; you can edit the form and oberve the results by hitting the "Run" button. Again, close the tab and hit "F1" when finished.
- Direct: the Octave window accepts commands directly. The up, down, left, right arrows allow you to recall and edit commands. It is almost like running Octave from the command line, but is missing the history aspect. Ask an example, type '`who` or `x=0:10`.

Wiki Edit

Changes to the text can be made using the wiki feature of the i-text. You can see a small "Edit" link after section and subsection headers. Clicking on that link brings up the TEX or LaTEX file with that section or subsection. Actions include:

- Editing: you can modify the text, inserting, deleteing, highlighting and cutting/ pasting, etc.
- Preview: when you have it in the form you want, the preview link will remake the PDF and display it. If there is an error, yiu will see a page dislaying the log; the links at the top to edit and msgs allow you to go back and forth between the text and the messages; once you've corrected the error, you can try previewing it again.
- History: This gives access to all the previous versions of this file in the version-controlled repository (and perhaps the last down-loaded copy in the file with the backup suffix ~). The list appears in the "msgs" window. Clicking on a version retrieves it from the repository and then, if it is different from the current version, displays the merge window.
- Save: This pushes the file to the repository and makes it available to everyone else. To avoid simultaneous editing conflicts, this will first pull from the repository so you can reconcile your changes with someone else has made since the time you last synced with the repository. Then your version is pushed to the repository and committed to version control, and a second pull is done to update the timestamp.
- Pull: the logic is a bit complicated here:
  - you have a timestamp in the top directory of the book, indicating when you last pulled from the repository
  - first all files in the repository that are newser than the timestamp are broaght back and saved as *filename*~.

- if the current version of any of these files is not newer than the timestamp, the file is replaced with the one from the repository.
- if the current version is newer than the timestamp, the merge window is brought up to allow you to reconcile the versions
- the timestamp is updated

There may still be logic errors here.

Merge window Edit

This shows two versions of the file, with lines from the current version in blue, from the older version in red, and common lines in black. Clicking on one of the colored lines will switch it. If you have a blue section followed by a red section, clicking will swap the colors. You can also edit, using Emacs key bindings. The "help" will eventually display these. The "Edit" will at some point allow you to change all the marked sections. When you are done, you can save the file, which overwrites the text using the black and blue text only and then puts it into the html edit window. It does not remake the PDF; you need to hit preview to do that. Or you can quit, leaving the file unchanged.

The merge window is also called up at various other times – upon startup of the book if your copy has newer material than the repository or when saving if something has changed while you were editing.