Demo Course

Author

2024–25

Contents

Chapter 1

Conversion from LaTeX to HTML

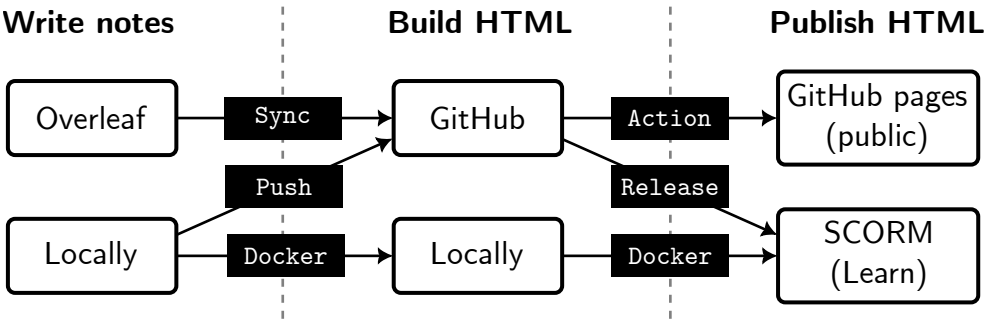
1.1 Choosing your preferred workflow

There are several tools available to perform conversion of LaTeX to HTML. Here, we focus on BookML, which uses LaTeXML in the background; note that this is the same technology used by arXiv to produce HTML versions of research papers from the `.tex` source.

There are three steps to converting LaTeX notes to HTML using BookML:

1. write notes in LaTeX;
2. use BookML to convert LaTeX code to HTML;
3. share the HTML version with students.

Your precise workflow will depend on how you wish to share the HTML version with students and whether you prefer to work in Overleaf, with GitHub or on your local machine. Some possible workflows are summarised in the figure below.



We give details of two of these workflows below. You may wish to deviate from these documented workflows if you feel comfortable doing so.

1.2 Overleaf → GitHub → GitHub Pages

If you are happy to write your LaTeX notes in Overleaf, to have your source code stored (privately) on GitHub and to have your HTML notes available on a public website then the **Overleaf → GitHub → GitHub Pages** workflow will likely be most convenient. After the initial setup, there is no need to interract directly with GitHub.

Step 1: Make a copy of the course notes template from GitHub

Step 2: Create an Overleaf project linked your GitHub repository

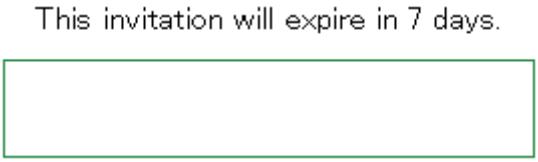
Step 3: Write your notes in Overleaf and sync with GitHub

1.2.1 Step 1: Make a copy of the course notes template from GitHub

Create a GitHub account and join the UoE School of Mathematics organization

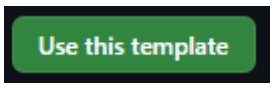
If you already have a GitHub account then please email the Learning Technology Team lt@maths.ed.ac.uk giving your GitHub username and ask to be added to the UoE School of Mathematics GitHuB organization.

If you do not have a GitHub account then please email the Learning Technology Team lt@maths.ed.ac.uk and ask for an invitation to the UoE School of Mathematics GitHuB organization. You will receive an email from GitHub with a link to create an account and join the SoM organization. Please note that the email may not render properly in your mail client – the link may appear as an empty box but this is the invitation link!



Make a new GitHub repository for your notes using the SoM template

Sign in to your GitHub account and visit the template repository. Click the "Use this template" button at the top right of the page.



The "Create a new repository" page has some options – the defaults are fine. **Please ensure you the box to "Include all branches" is not ticked.** By default, your notes will be stored in a private repository on GitHub in the UoE School of Mathematics organization. You are free to change these settings if you wish.

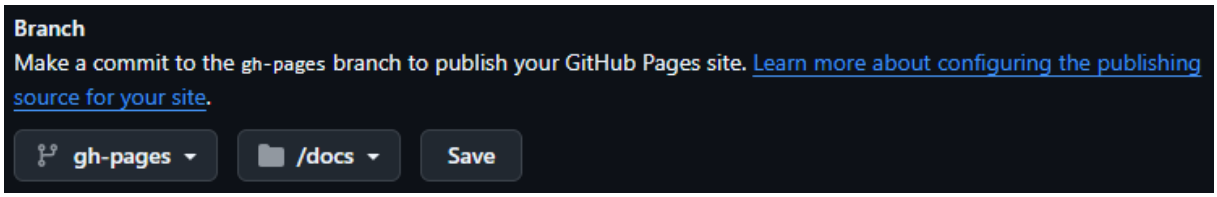
Please wait at least 3 minutes before moving on. GitHub needs some time to initialise the project.

Set up the automatic creation of a website for your HTML notes

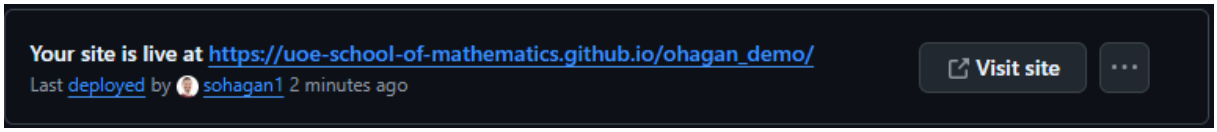
You can set up a GitHub Action to automatically generate the HTML version of your notes from the GitHub repository, and keep this up to date. This means you only need to add a link to your notes on Learn once and this will always point to the most up-to-date version of your notes.

To set up a publicly-accessible website with your HTML notes in GitHub, first visit your new repository in GitHub. In the top menu click **Settings** then **Pages** in the left menu.

Under **Build and deployment**, look for the **Branch** option. From the **Select branch** option, choose `gh-pages` and **Select folder** `/docs`.



Wait 1 minute then refresh the page. You should now see a link to the publicly-available site hosting your HTML notes. This includes the URL that you should use to share your notes with students.



1.2.2 Step 2: Create an Overleaf project linked your GitHub repository

Link your Overleaf account to your GitHub account

Visit your Overleaf account settings. Under **Project Synchronisation**, click **Link** next to **GitHub Sync**. Follow the prompts to link Overleaf and GitHub.

Make an Overleaf project from your GitHub repository

At this stage you should have a GitHub repository created from the SoM template.

In Overleaf, create a new project ensuring you choose **Import from GitHub**. Then choose **Import to Overleaf** for the appropriate repository.

Once this is complete, you should see a copy of this documentation in your own Overleaf project.

1.2.3 Step 3: Write your notes in Overleaf and sync with GitHub

At this stage you should have an Overleaf project for your notes, linked to your GitHub repository, and have an action set up to automatically build a public website with the

HTML version of your notes.

Now you can work on your notes just like any other Overleaf project.

Whenever you want to publish changes to the HTML version, you need to push the changes to your GitHub repository. To do this, in Overleaf, click **Menu** (top left) and then choose **GitHub** under the **Sync** menu.

Click **Push Overleaf changes to GitHub**, enter a description of the changes you have made (optional) and then click **Commit**.

Behind the scenes, GitHub will now rebuild the HTML version of your notes and update the public website. You don't need to interact directly with GitHub again.

1.3 Locally → Locally → SCORM

1.3.1 Using Docker

To run BookML, we can use a Docker container — a lightweight, portable unit that packages an application and all its dependencies, ensuring that it runs consistently across different environments. This removes the need to download software and maintain dependencies on your computer.

Importantly for us: there is a Docker image available for BookML, which is ready-to-use and contains everything we need.

To use it, we can use Docker Desktop, which is available for Windows and Mac, or Docker Engine, which is available for Linux. An open-source alternative (compatible with Docker) is Podman, which is available for Windows, Mac, and Linux. In any case, the key steps are:

1. Setup (just once): install Docker or Podman, and download the Docker image.
2. Convert: run a single command in the folder with your `.tex` files.

Both Docker and Podman allow us to use either the command line or a graphical interface. If you don't know which one to choose, just pick one and try it!

Notes:

- If you have a managed Windows machine, Docker Desktop is available from the software centre. This is a slightly complicated process, see Appendix **??** for instructions. It may be simpler to request 'Make Me Admin' and install from the internet.TODO If you wish to use Podman, you will need to request 'Make Me Admin'. Guidance on 'Make Me Admin'.
- The Learning Technology team will be able to assist you with installing Docker or Podman — email `lt@maths.ed.ac.uk` to ask for help.

Setup (first-time only)

Option 1 - to set up Docker:

1. - **Windows/MacOS/Linux:** Download Docker Desktop from **this** link or the Software Centre, and install as instructed by the interface. Launch Docker Desktop. There are versions of the download file of Docker Desktop which cause an error. Ensure you are downloading the version that is listed in the documentation.
   - **Alternatively for Linux only:** Download and install Docker Engine, following the instructions for your distribution.
2. Launch a terminal (use `cmd.exe` on Windows), and run the command:
   `docker pull ghcr.io/vlmantova/bookml:latest`
   This will download the Docker image to your machine.

Option 2 - to set up Podman: Note that this requires "Make Me Admin" on Windows managed machines.

1. Download Podman Desktop from here and install as instructed by the interface.
2. Podman desktop will open, and you will be asked to setup. Click the 'Setup' button and follow through the instructions that follow. When prompted, select 'Yes', and then 'Install'.
3. Once fully installed, we can download the image. To do this, open the 'Images' tab (the 4th option on the left-hand side of the screen) and click `Pull` on the top right of the screen, then enter `ghcr.io/vlmantova/bookml:latest` and click "Pull image".

Conversion

Before you start, for better results, add `\usepackage{bookml/bookml}` to the preamble of your `.tex` file(s) (after the `\ documentclass{...}` command). This is not necessary, if using the template provided by the School of Mathematics. This package may cause an error in overleaf, unless you upload the BookML folder to your project.

Option (a) - using the command line:

1. Start a terminal in the directory containing your `.tex` file(s). (Use `cmd.exe` in Windows.)
2. Run the command:
   `{software} run -t -v .:/source ghcr.io/vlmantova/bookml:latest`
   (replace `{software}` with `docker` or `podman`).

Option (b) - using the graphical interface:

1. In Docker/Podman Desktop, click on the 'Images' tab, and you will see the image you have just downloaded. Click the 'Play' button (Figure 1.1).
2. Under "Volumes", specify the "Host path" to be the path to your folder, and the "Container path" to be `/source`. (In Docker Desktop, this is under "Optional settings" – see Figure 1.2.)
3. Click 'Run' (Docker) or 'Start Container' (Podman) to produce the PDF and HTML outputs.

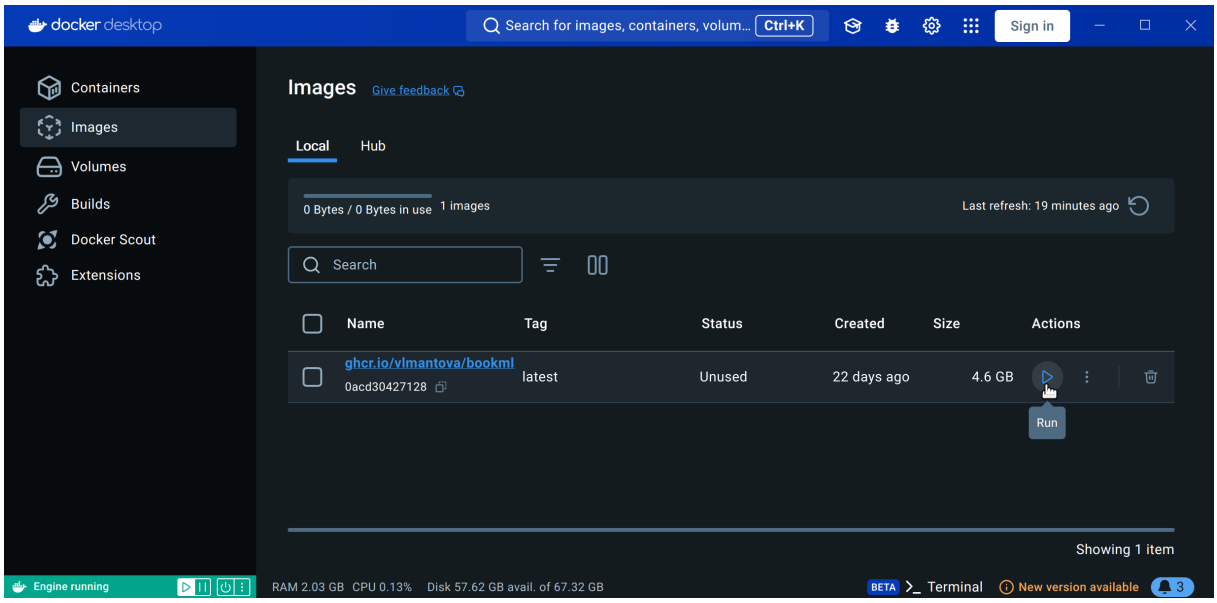


Figure 1.1: Step 1: click the 'play' button to run the command.

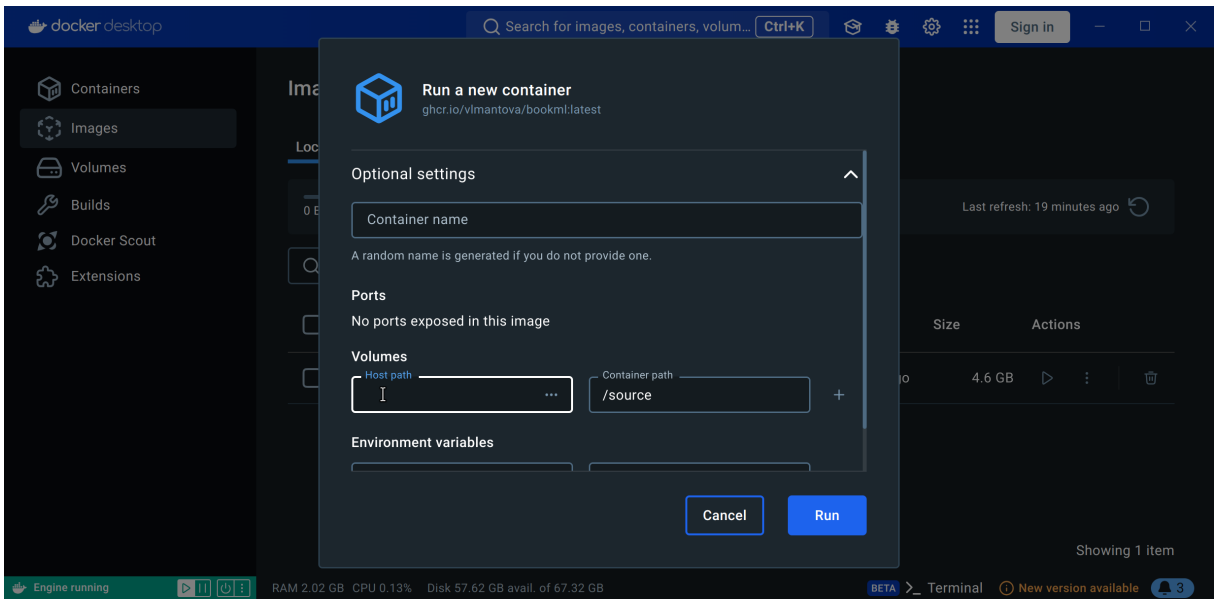


Figure 1.2: Step 2: give Docker the path to your folder.

1.3.2 Not using Docker

If you prefer to install the necessary packages and dependencies to your machine, you can follow the instructions in the BookML documentation.

Running BookML will produce PDF files (with `pdflatex`), as well as `.zip` files containing the HTML versions of your `.tex` files, which can be uploaded directly to Learn. The output (both PDF and HTML) will be found in the same folder as your `.tex` files.

To convert files locally (on your own machine), there are two options: installing the required packages and dependencies yourself, or using a Docker container. This image is 4.6GB so be prepared for a large download. Whichever option you use, conversion is likely to take a few hours if you have large files ($\approx$ 200 pages).

Upload SCORM

The easiest way to upload the HTML materials to Learn is using the `SCORM.myfolder.zip` which is created automatically.

- On your Learn page, create a new content item.
- Choose the "SCORM package" option.
- Upload the file `SCORM.myfolder.zip`.
- Once it's uploaded, untick the "Mark SCORM" checkbox, and click "Save".

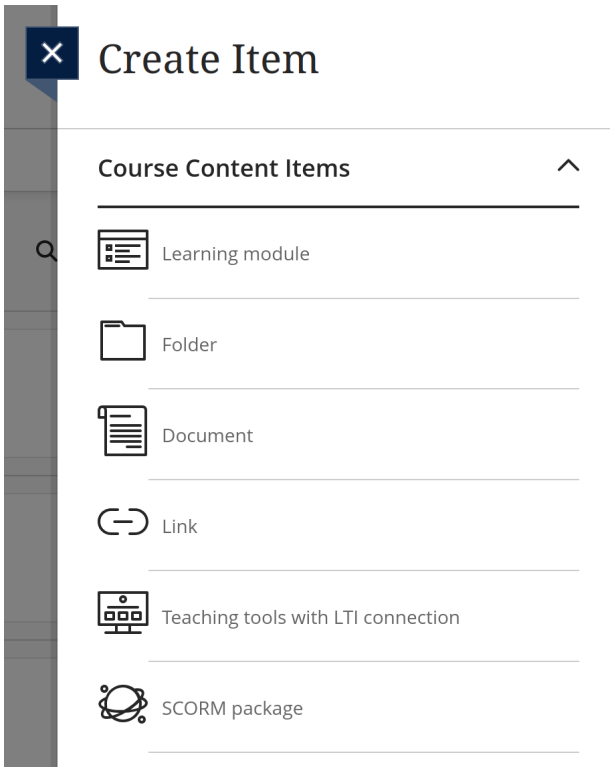


Figure 1.3: The "SCORM package" option on Learn Ultra.

1.3.3 Learning Technology Mailbox

If you would rather, you can also email `lt@maths.ed.ac.uk` with a .zip of the folder containing all your course materials (everything needed for your .tex files to compile normally). The Learning Technology team will perform the conversion, and email you back with a .zip file containing the HTML version of your course materials; optionally, they can also upload it to your Learn page. If they spot anything which didn't convert well, they will let you know.

1.4 FAQs

Q: Error: `Warning: no .tex files with \documentclass found in this directory`

A: The software cannot reach your files, this may be due to running the software in the wrong place, permissions, or a remote directory. Double check your file path, and try moving the files to a local directory and try again.

Q: The conversion process is taking a very long time.

A: The conversion process can take a while, especially for larger documents — but if it's 30 minutes or more, then please flag this. Note that, if you are using a local installation (not Docker), there are some known issues related to this:

- There is a known issue with the package `expl3` when used with TeXlive 2022 and older which can cause the conversion to take longer.
- There is a known issue with the "postprocessing XSLT" stage taking several minutes (instead of seconds) on Ubuntu 22.04.

Q: Will my .tex files be converted if I comment out `\documentclass`?

A: Yes, and it may cause errors. For files which you don't want to be converted standalone (e.g. files which are `\input`ted into another file), delete the `\documentclass` command entirely or move them into a folder.

Q: Why aren't my files converting at all, there is not an error, BookML simply states there are no files to convert.

A: This is likely due to the fact that the software cannot reach your files. This can be because your files are in a folder, or that your files have forbidden file characters or a space.

Chapter 2

Template document: usage and examples

The School of Mathematics provides a default template, which is designed for writing new course materials – it aims to improve accessibility and convert nicely to HTML. It should also be relatively easy to convert existing materials to this template. The template provides a preamble in `preamble/SoM.tex`. You can add your own preamble in `preamble/custom.tex`.

The examples shown in this section are intended to demonstrate the use of the packages and environments included by default in the template, and to give an overview of how different elements are rendered in both the PDF and the HTML versions of a document. Workarounds and LaTeX snippets are also given for certain packages or commands which are not compatible with BookML; these may also be useful to those not using the template.

2.1 File structure

All content should go either in `main.tex` (single file), or in separate `.tex` files located in the same folder as `main.tex`. If using separate sub-files, they should be included into `main.tex` using `\input{}`, as is done in this document. Sub-files should not have `\begin{document}` or `\end{document}` commands.

The `main.tex` template uses the `book` document class; therefore, the levels of sectioning available are `\part`, `\chapter`, `\section`, ... If you prefer to use `article` as the document class, then you should use the code in `main-article.tex~`.

You can use `preamble/custom.tex` to complement the template preamble with any additional packages, macros, environments, and so on.

2.2 Including or excluding content from either version

The `bookml` LaTeX package includes the command `\iflatexml` to include or exclude particular code from either the PDF or the HTML version of your notes.

- Use `\iflatexml` to delimit anything (content, packages, macros, ...) that should only be used in the HTML version.
- Use `\iflatexml\else` to delimit anything (content, packages, macros, ...) that should only be used in the PDF notes.

You can check the following example in the source file for this document:

This will only show up in the PDF version.

2.2.1 Theorem styles

`preamble/SoM.tex` defines a range of theorem-like styles, and sets up sequential numbering indexed on `chapter`. Here is an example of a `theorem` and a `proof` environment.

Theorem 2.1 and its proof will be in the exam:

Theorem 2.1. Let $\boldsymbol{e}_k = \mathbf{R}^k \boldsymbol{e}_0$, for $k \in \mathbb{N}$ and some $\mathbf{R} \in \mathbb{R}^{n \times n}$. If $\|\mathbf{R}\|_p < 1$, then $\|\boldsymbol{e}_k\|_p \to 0$ as $k \to \infty$.

Proof. We have

$$\begin{aligned} \|\boldsymbol{e}_k\|_p &= \left\|\mathbf{R}^k \boldsymbol{e}_0\right\|_p \\ &\leqslant \left\|\mathbf{R}^k\right\|_p \|\boldsymbol{e}_0\|_p \\ &\leqslant \|\mathbf{R}\|_p^k \|\boldsymbol{e}_0\|_p \end{aligned}$$

and so $\|\boldsymbol{e}_k\|_p \to 0$ as $k \to \infty$ if $\|\mathbf{R}\|_p < 1$. □

Note that the `align` environment is used in the proof, and also works as expected in the HTML version.

There is also a non-numbered "Example" environment, which has a left-side bar to differentiate it from the rest of the notes. TODO

Example 2.2. Let

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 4 & 3 & 3 \\ 8 & 7 & 9 \end{bmatrix}.$$

We want to put $\mathbf{A}$ into row echelon form. We create zeros below the diagonal in the

first column by subtracting multiples of the first row from the other rows:

$$\mathbf{L}_1\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 & 1 \\ 4 & 3 & 3 \\ 8 & 7 & 9 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 3 & 5 \end{bmatrix}.$$

Repeating this for the second column

$$\mathbf{L}_2\left(\mathbf{L}_1\mathbf{A}\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 3 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix} = \mathbf{U}.$$

The above procedure factorises $\boldsymbol{A}$ as $\mathbf{A} = \left(\mathbf{L}_2\mathbf{L}_1\right)^{-1}\mathbf{U}$.

And some extra text after the example to show the return to the regular layout.

If you wish to use a different theorem style, you can do so by editing `preamble/SoM.tex`. To reflect these changes in the HTML, you can edit the CSS in `bmluser/SoM.css`. `.ltx_theorem_theorem` is the class for theorems, and `.ltx_theorem_proof` and similar. to edit specifically wish to edit the way it looks in sepia mode or night mode. Add css to `.color-theme-1 .ltx_theorem_theorem` and `.color-theme-1 .ltx_theorem_theorem` respectively.

2.2.2 Figures

Raster images

Figure 2.1 is a PNG image included with `\includegraphics`. The `\img` macro is set in `preamble/SoM.tex`, which you can change there if your graphics folder has a different name.

Use the `\includegraphics[alt={alternative text}]{\img/image.png}` to provide **informative alternative text** for all your images — this is an **important accessibility feature**. Here is useful advice on writing good alternative text.

To avoid images being inverted in night mode in HTML, add `\bmlPlusClass{bml_no_invert}` after the figure. This can also be used for tikz images.

`tikz` images

Figure 2.2 is a `tikz` diagram included with `\input`.

`pgfplots`

PGF plots are also supported, as shown in Figure 2.3.

Figure 2.1: A PNG image included with `\includegraphics`.

cell1 dummy text dummy text dummy text	cell2	cell3
cell1 dummy text dummy text dummy text	cell5	cell6
cell7	cell8	cell9

Table 2.1: Example table produced with the `tabular` environment.

Subfigures

The `subfigure` environment from the `subcaption` package can be used for both the PDF and HTML versions, as seen in Figure 2.4.

2.2.3 Tables and floats

Tables

Tables can be typeset using the `tabular` environment (see Table 2.1). Note that the `tabularx` package is also compatible.

Wrapping text around figures

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis

Figure 2.5: Text wrapped around an image using `wrapfigure`.

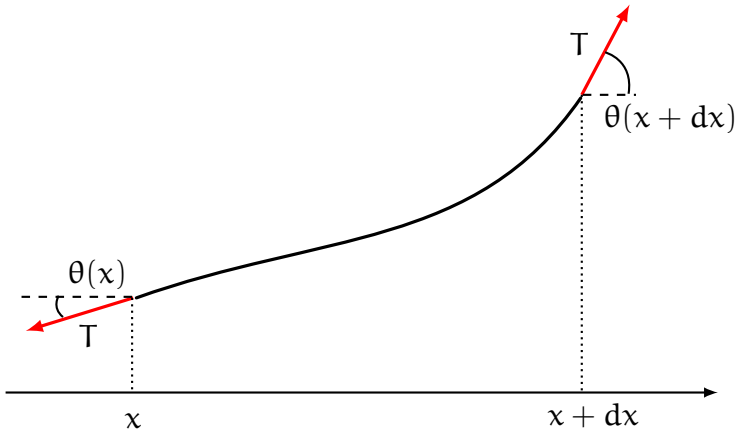


Figure 2.2: Small string element under tension.

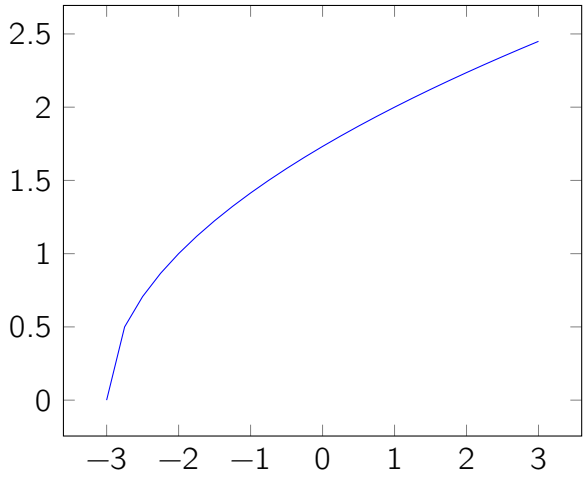
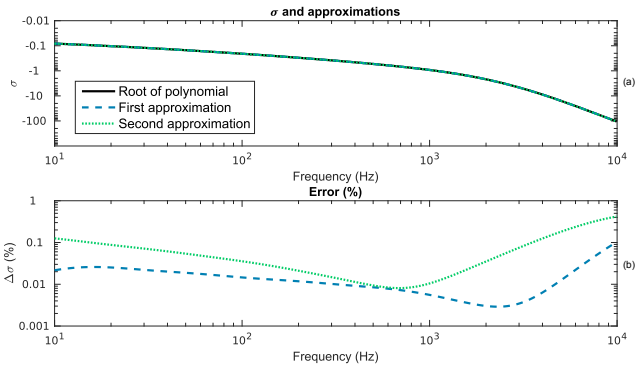


Figure 2.3: Plot created using the `pgfplots` package.

egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

(a) A JPEG image as a subfigure.



(b) An EPS file as a subfigure.

Figure 2.4: This is the main caption for the two figures. Figure 2.4b is a `.eps` file.

2.2.4 Referencing from a bibliography

The package `natbib` is compatible with BookML. Note that `biblatex` is not yet compatible.

This is an example reference [Str04], and another [PFCC16].

Appendix A

Context: why are we doing this?

The University is legally required to provide learning materials in accessible formats[1]. Although guidance is provided through Information Services[2] for some document formats (e.g., MS Office), workflows deviating from these are not currently supported.

Particularly in subjects which require the extensive use of mathematical notation or other non-standard characters, the LaTeX typesetting language is widely used to generate PDF lecture notes, worksheets, and slides. However, these documents do not currently fulfil accessibility requirements (see Section A.1). While there are various approaches to producing accessible documents from LaTeX source files, none of these are widely adopted across the University, or elsewhere. Barriers include technological expertise and means, workload, inconsistent quality of output, and lack of institutional and technical support.

The PTAS project "A systematic approach to enable production of accessible course materials" aimed to close this gap by identifying and overcoming such barriers in a systematic manner. Working in collaboration with a diverse community of educators and students, we have developed a set of guidance documents, technical instructions, and LaTeX templates to facilitate the creation of course materials in accessible, flexible output formats. We have particularly focused on HTML output, which is not only supported by a wide variety of devices, but can also be displayed using user-defined preferences, such as fonts, text sizes, colours, and wrapping for different screen sizes; it is also compatible with many screen readers.

A.1 Why are PDFs not "accessible"?

In our context, **accessibility** relates to course content being accessible (as in readable and usable) to everyone, in particular people with disabilities or health conditions. Written content can present accessibility barriers to different people. For example,

- those with a visual impairments may use assistive technology (e.g. screen readers) to access course materials;

[1] Accessible and Inclusive Learning - University of Edinburgh
[2] Creating accessible materials - University of Edinburgh

- partially sighted or colourblind people, or those with difficulties related to processing text (for example people with dyslexia or ADHD), may need to customise fonts, font sizes, spacing, and colours to improve accessibility.

None of the above is practical to do with PDF files. In particular:

- PDF does not allow fonts, sizes or colours to be changed easily;
- LaTeX-produced PDF documents are not "tagged"[3] because LaTeX does not (yet) support this feature[4]. A tagged PDF provides metadata on the document structure, which makes it possible for assistive technologies to navigate between sections — and therefore makes screen readers usable.
- Crucially, screen readers cannot read mathematical expressions inside a PDF file, because the PDF file does not contain any semantic information about the mathematical expressions — it only contains a graphical rendering of the expressions.

A.2 A solution: HTML with MathML

HTML documents are accessible in some ways that PDF documents are not.

- The user can easily change font size, font family, and colours[5]. It is also possible to print from HTML to paper while conserving your font settings – so you can print in large font, for example (which, again, is not straightforward with a PDF).
- The document structure and navigation relies on HTML tags, which are accessible by screen readers.
- Providing alternative text (or alt-text) for figures is easier. This is different from figure captions — it describes the key features of a figure under the assumption that the person reading the alt-text cannot see the image.
- Mathematical expressions are displayed using MathML[6], which is compatible with a range of assistive technologies (e.g. screen readers and Braille displays)[7].

MathML renders all mathematical expressions in a document, whether inline ($x = 1$) or in display mode:

$$f(x) = \int_0^\infty \left(\frac{e^{-t^2}}{\sqrt{\pi}}\right)\left(\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)!!}\right)\,dt \tag{A.1}$$

[3] The accessibility checker *Ally*, which is the tool embedded in Learn Ultra, determines the accessibility score of a PDF file largely depending on whether or not it is tagged.

[4] There is ongoing work to improve this at the LaTeX Project, but this is still in early stages; see the paper Enhancing LaTeX to automatically produce tagged and accessible PDF for a summary of progress as of March 2024.

[5] This can be done with custom style sheets (for instance via browser extensions such as Stylus), or Javascript (for example, try the toolbar at the top of the HTML version of this document).

[6] MathML homepage

[7] Brief explanation and demonstration of MathML, including a video showing use with a screen reader.

You may be familiar with MathJax[8], used for instance on STACK. MathJax is a Javascript engine used to display MathML in the browser. This is a good primer on what they both do. In particular, MathJax offers a contextual menu on right-click with further accessibility features, including the possibility to copy the LaTeX source for an expression (try it with the expression above, in the HTML version of this document).

It is possible to generate HTML with MathML *automatically* from `.tex` source files — so we don't have to worry about doing any of this manually. A key part of the PTAS project was to review different workflows for doing so, in order to provide recommendations and appropriate support for robust, user-friendly production of accessible course materials.

A.3 Insights from staff interviews

During the Spring semester of 2023–24, we conducted interviews with teaching staff from the Schools of Mathematics, Physics and Astronomy, Engineering, and Philosophy, Psychology, and Language Sciences to better understand their current workflows in producing course materials. We specifically chose staff members who work extensively with LaTeX.

Several common themes emerged from these discussions, which helped inform our approach:

Teaching staff have limited time. Any proposed adjustments must be integrable into existing workflows with absolutely minimal increase in workload. Solutions should prioritize simplicity, efficiency, and robustness, without the need to extensively consult documentation or complex instructions. Technical support should be provided to resolve problems.

Openness to adjustments in workflow. While the staff we interviewed expressed willingness to adjust their workflows to enhance accessibility of their materials, this may well be biased by us interviewing staff who put themselves forward to discuss this in the first place. This underscores the importance of making all recommendations user-friendly and low-friction to ensure widespread adoption.

Challenges of shared LaTeX documents. Documents in LaTeX authored by multiple contributors or handed down over time often exhibit inefficiencies such as redundant or inefficient use of packages. Additionally, they may contain large blocks of commented-out code from previous versions or authors, which can clutter the document and complicate maintenance. These remnants, while intended for reference or future use, often become outdated and may not serve future iterations of the document effectively. Such practices can hinder readability, increase file size, and make it challenging to navigate and update course materials efficiently.

[8] `https://www.mathjax.org/`

There are more similarities than differences. While a multitude of resources are employed in creating course materials, the majority of LaTeX documents typically share similar foundational structures. Variations typically arise from specialized packages tailored to specific subjects, use of different engines to compile PDFs, the number of authors involved, and other contextual factors. This implies that supporting solutions which accommodate different packages, custom macros, and figures/images will enable us to develop a solution applicable to a wide range of LaTeX documents.

A.4 Flexibility is the key

A driving principle we choose to adopt in this project is **flexibility**, both on the side of the users (usually students) and producers (usually lecturers) of course materials.

- Writers of course materials should be able (and appropriately supported) to produce accessible versions with as little change to their preferred workflow as possible. In any automatic conversion process, technical issues and incompatibilities are inevitable — but technical support, consultation, testing, and iterative refinement will drastically help to reduce this.
- Users should be able (and appropriately supported) to customise the display of course materials to a format that is most accessible to them. There are certainly general guidelines we can follow to produce accessible documents — but prescribing a specific format (e.g. prescribing a specific font family or colour palette) is not as good as giving people options.

Importantly: accessibility is about ensuring that **everyone** can engage fully. For instance, for many students, PDF course materials will always be their preferred option: they are useful for printing, annotating and note-taking on tablets, and can be easier to use offline. We are very much **not asking lecturers to get rid of their PDF notes** unless they want to — the idea is to also provide alternative options.

Appendix B

Upload as ZIP

You can upload a `.zip` file directly to the Content Collection on your Learn page, as a "zip package". PLease note this method will not work if you upload the `.zip` file to your general content collection, rather than on the direct on the specific learn page.

1. On your main Learn page, click the + button to add content, then select 'Content Collection' at the bottom.
2. Select 'Browse content collection'.
3. At the top, click 'Upload', then 'Upload zip package'.
4. Select 'Browse local files' and select the `.zip` file you have just created, or drag and drop the file into the box.
5. Click 'Submit'.
6. Click on the folder you have just uploaded, check the box next to `index.html` and click 'Submit'.
7. Click on the `...` on the right to change the name from `index.html` to something more helpful for students (e.g. "Course notes"), and click 'Save'.

Important: with this method, you then need to add permission to the folder for students to view the whole site.

1. From the Learn home page, select "Tools", then "Content Collection".
2. Click on the folder corresponding to your course.
3. Hover on the folder containing your HTML course materials, click the arrow to get the contextual menu, and select "Permissions".
4. Click on "Select Specific Users by Place", then "Course".
5. Tick the name of your course, then tick "All course users" below.
6. Tick the "Read" option, leave the others unticked, and also leave "Overwrite" unticked.

7. Click “Submit”.

Bibliography

[PFCC16] Augustin Parret-Fréaud, Benjamin Cotté, and Antoine Chaigne, Time-domain damping models in structural acoustics using digital filtering, Mechanical Systems and Signal Processing **68** (2016), 587–607.

[Str04] John C. Strikwerda, Finite difference schemes and partial differential equations, second edition, Society for Industrial and Applied Mathematics, 2004.