Abschlussbericht

Felix Benning

2025-07-17

Table of contents

Preface **1**

1 Overview **2**
1.1 The Goood . . . 2
1.1.1 H5P integration demo . . . 3
1.2 The Bad . . . 3
1.3 The Ugly . . . 4

2 Behind the Scenes **5**
2.1 Interactivity . . . 5
2.2 Equations . . . 5
2.2.1 Aligned-Overset . . . 6
2.3 Global TeX Macros . . . 6
2.4 Pedagogical Context and Future Use . . . 7

Appendices **8**

A User guide **8**
A.1 H5P Integration . . . 9
A.2 Global LaTeX Macros . . . 10
A.3 Mathematical equations . . . 11
A.3.1 Equations in LaTeX . . . 11
A.3.2 Equations in Quarto . . . 12
A.3.3 Aligned-Overset . . . 15
A.3.4 Theorem environments . . . 16
A.4 Publishing the book as a website . . . 17

B Useful extensions **18**

References **19**

Preface

This project aimed to create a reusable and well-documented Quarto template tailored to the needs of mathematicians, particularly for educational content. The result is available at https://www.FelixBenning.github.io/math-book/, which provides both the template and its extensive documentation.

This final report was generated using this template (reusing some of the documentation) and is also available as a webpage at https://www.FelixBenning.github.io/HDZ_Projectbericht/.

Chapter 1

Overview

This is an overview of the template for a Quarto math book or lecture script. Quarto[1] is a framework to render Markdown documents into `html` and via LaTeX into `pdf`. The reason you may want to target `html` is immediately illustrated if you read the web version of this documentation. For example, try to hover over the footnote on "Quarto", or over the following equation reference Equation 1.1.

This interactivity is something that is impossible with `pdf`. And since LaTeX can only really target `pdf` this is a problem. However LaTeX is really good at equations and other academic features that make it difficult to replace. Quarto provides these features.

1.1 The Goood

Quarto has

1. bibliographic reference management that can generate citations e.g. Knuth [1] based on the cite keys in bibtex `.bibtex` and biblatex `.bib` files. Try to hover over the citation!
2. It has machinery to produce theorems
3. It has machinery to cross-reference figures, tables, sections, theorems, etc.
4. And most importantly for maths, it is integrated with the mathjax JavaScript library that can render LaTeX equations on websites, like this one for example:

$$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C} \tag{1.1}$$

[1] Quarto is a language agnostic the successor to R Markdown, both of which are built on the universal document converter Pandoc. R Markdown allowed for R code snippets to be included in a markdown document which were executed before display. Quarto also allows this for Python and JavaScript.

1.1.1 H5P integration demo

In the `html` version a H5P Demo is displayed here.

1.2 The Bad

- **Global LaTeX Macros** do not work well out of the box. However this problem is fixable (see Section A.2 and this GitHub Discussion)
- **The Reference System**
  - It is impossible, or at least very difficult, to **reference individual lines** of aligned equations (see Section A.3.2)
  - It is not possible to use **shared counters** for theorems and lemmas as in LaTeX (see Section A.3.4).

    This makes it more difficult to find a theorem manually, because the numbers on the lemmas do not tell you whether to look before or after. Digitally however this is less relevant since you do not have to manually search. And it is most likely possible to fix this for `LaTeX` with templates, so this could be "fixed" for the print version.
- **Inconsistent Behavior** Quarto has idiosyncrasies it does not consider bugs: The behavior of

  ```
  ::: {.proof name="Proof name"}
  Proof content
  :::
  ```

  is different for `LaTeX` and `html`. In `LaTeX` it is typical that the name replaces everything. So instead of `Proof.` it would say `Proof name.`. Quarto does the same to be "consistent". However in `html` the name is only used in brackets after the name, i.e. the end result is `Proof (Proof name)`. This makes it difficult to use this feature for multi-target documents.
- **Sync to Code** SyncTeX is an amazing tool that stores crossreference data in a file. The LaTeX Workshop extension for VSCode uses this file to allow users to `Cmd` + Click on text in the pdf to be directed to the `LaTeX` code that produced this text. Similarly it is possible to jump from code to the corresponding place in the `pdf`. Synctex is similarly integrated with other editors. This is something that does not really exist for Quarto. To my knowledge this does not even exist for `html` preview in VSCode so this may be very difficult to implement. However there exists a partial WYSIWYG[2] editor for quarto as part of the VSCode extension (right click: "Edit in Visual Mode"). The downside of this mode is that

[2] What you see is what you get

1. it does not preserve existing code
2. it does not render Global Macros (Section A.2) correctly
3. it is incompatible with the Vim extension of VSCode that I personally like.

1.3 The Ugly

Since Quarto can target `html` it may eventually be possible to make proofs collapsible although at the moment this is not possible out of the box. Callouty Theorems is an extension to make this happen. But the way it does it is ugly:

1. It wraps theorems and proofs into callout blocks that are visually too prominent, especially for proofs.
2. Since it is only a wrapper, the hover-overs only reference the inner part
3. and the theorem has two titles, the normal title and the title of the wrapper

Chapter 2

Behind the Scenes

To keep the technical details light, I will only give an overview of what I done and provide links to the technical details.

2.1 Interactivity

The reason one may want to provide mathematical documents as a website is of course interactivity. Basic form such as `hover-overs` for references already work out of the box.

- **H5P Integration**: With help from the maintainer with (the execution order of Javascript and relative links) I integrated the h5p-standalone project into this template. The inclusion of `h5p` elements is documented in Section A.1.
- **Collapsible Proofs** are currently not available out of the box. But it should be relatively easy to create an extension to enable them in the future. There is an extension called Callouty Theorems but the solution is not ideal (Details in Section 1.3)

2.2 Equations

An equation

$$\begin{aligned}(a+b)^2 &= (a+b)(a+b) \\ &= a^2 + 2ab + b^2\end{aligned} \tag{2.1}$$

may want to be referenced like so, (Equation 2.1). While Quarto does clearly provide a mechanism to do so, it does not provide a mechanism to reference individual lines of an equation. This is a feature that `LaTeX` does provide and other users have already reported this feature as missing in this discussion on GitHub. This is not often a problem, since you mostly want to reference the

entire equation. But it always hurts, when a feature is taken away from you, so people are naturally looking for workarounds. However the workaround posted in the discussion linked above does not work for referencing equation across chapters and breaks all the Quarto features (like hover over) for equations. For this reason I suggest a different workaround (Workaround 2 in Section A.3.2). It works across chapters using the Quarto citation mechanism but causes more work in the few places where it is actually necessary to reference individual lines.

2.2.1 Aligned-Overset

The `LaTeX` package aligned-overset is something I use often. It is not available in `MathJax`, which renders mathematical equations in `html`. I therefore opened a feature request on the MathJax repository on whether it may implement this package. The maintainer suggested a partial solution which is now included in the template (see Section A.3.3).

2.3 Global TeX Macros

One of the main technical challenges was to enable global LaTeX macros across a Quarto project. This functionality is crucial for large mathematical documents, particularly textbooks or collections of exercises. You can find a motivation for the use of Macros and the way to enable them in Section A.2.

The issue that needs to be overcome boils down to the following:

- The `LaTeX`/`pdf` output is one large document with multiple chapters. Macros should therefore only be included once in the entire document. `LaTeX` in fact throws an error if a Macro is included twice to warn the user of accidental redefinitions.
- The `html` output produces a webpage for each chapter. Macros must therefore be included in *every* chapter.

While Quarto does provide mechanisms to include something in every file, the Macros also need to be processed. And for technical reasons it would be preferable if they were processed during the conversion from Quarto to `html` and not later by the `JavaScript` library. This design constraint made this very difficult. A full writeup was provided as a GitHub Discussion to be publicly available. The solution I used for this template is described as an "Alternative" there, because it requires some technical trickery that would be annoying if you had to do it yourself. But as part of a template this seems better, as it allows the macros to be documented with comments.

To package the solution nicely, I wanted to hide the technical stuff in the template. But this unfortunately does not work as anticipated, as discussed in this issue.

The end result of a few days of trying is a solution with some ugly quirks: You

need to include a special line at the top of every chapter file to get Macros to work and there is a `_macro_processing.qmd` file in your root directory that must not be touched.

2.4 Pedagogical Context and Future Use

The primary audience of this template are mathematicians that want to create interactive content. Svenja Kaiser, a PhD candidate in mathematics education, expressed interest in using the template to translate proprietary content from an InnoMA project into open formats. One option she considered was H5P which nicely integrates with this template. However a solution to enable equations in H5P is still needed.

While there are still some rough edges that may need polishing, I can see Quarto eventually satisfying all the requirements needed to create great interactive mathematical content.

Appendix A

User guide

1. Install Quarto, LaTeX and R[1]; and choose an editor (e.g. VSCode)
2. To use this template type the following into a terminal

   ```
   quarto use template FelixBenning/math-book
   ```

   > **Add to existing project**
   >
   > You can also use `quarto add FelixBenning/math-book` to add this template to an existing quarto project. However to use the global macros feature (Section A.2) you need to manually copy the file `_macro_processing.yml` into your root directory.

3. Update the configuration in the file `_quarto.yml` marked with a `TODO` (e.g. `title`, `author`, `site-url`, etc.)

For the standard features refer to Quarto's documentation we will focus on the features specific or at least important to this template.

> **Tip**
>
> In the web version of this documentation, try to click on the GitHub symbol next to the title in the top left. If you read it on a smaller device you may first need to open the left sidebar. This will lead you to the GitHub repository of this template. Next to it is a button to download the `pdf` version.
> If your device is wide enough for the Table of Contents to be displayed on

[1] required for the global macro feature Section A.2

the right, you can click on "View source" to be directed straight to the `.qmd` file that produced this chapter. This way you can see how certain features are used by example.
There is also an "Edit this page" button. If someone finds a spelling error for example this allows them to suggest an edit. Note that if they do not have access to the GitHub repository they cannot change it directly.

A.1 H5P Integration

This template includes the `tunapanda/h5p-standalone` project and thereby supports the addition of H5P files without the need for a separate H5P server. H5P files (with the `.h5p` file extension may be added as follows):

1. Change the `.h5p` file extension to a `.zip` extension
2. unzip the zip file and place the resulting folder (referred to as `my-h5p-folder` in the following) into the folder `assets/h5p-content/` of the Quarto project.
3. In Quarto markdown file where you want to add the `h5p` content add the following

   ```
   ::: {#my-h5p-folder .h5p}
   :::
   ```

   or alternatively

   ```
   <div id="my-h5p-folder" class="h5p"></div>
   ```

Possible issue: Missing libraries

Historically, every H5P file with the extension `.h5p` used to include the H5P libraries necessary to make the content work. More recently, these libraries are not always included (see this GitHub issue). In this case the `my-h5p-folder` does not include folders of the form `H5P.AdvancedText`, `H5P.Audio`, `H5P.Blanks`, etc. But without these libraries the content does not work.
To make it work, simply copy these libraries from an older `h5p` file into this folder (e.g. from the `multiple-choice` example of this template)

A.2 Global LaTeX Macros

A very important feature for LaTeX users is the possibility to define custom macros. For example the following custom macros

```
\newcommand{\real}{\mathbb{R}}
\newcommand{\complex}{\mathbb{C}}
```

can be used to produce $\mathbb{R} \subset \mathbb{C}$ from `\real \subset \complex`.

> **What is the benefit of macros?**
>
> The notation for the natural numbers $\mathbb{N} = \{1, 2, 3, \dots\}$ requires "Blackboard letters" for example. Those can be achieved using the command `\mathbb` – specifically `\mathbb{N}`. A custom macro may be defined as follows in LaTeX
>
> ```
> \newcommand{\nat}{\mathbb{N}}
> ```
>
> Observe that such custom macros may not only be used to abbreviate commands, e.g. `\nat` instead of `\mathbb{N}` to produce $\mathbb{N}$, but it can also be used to create semantic placeholders for symbols that can be changed later. For example one may initially choose the letter η to denote a step size before realizing that this letter is needed for something else. If a `\stepsize` macro is used it is trivial to replace
>
> ```
> \newcommand{\stepsize}{\eta}
> ```
>
> by a different letter, say h.
>
> ```
> \newcommand{\stepsize}{h}
> ```
>
> An update to the command immediately updates every occurrence of `\stepsize`.

These macro definitions can be placed into a global `macros.tex` file and annotated with LaTeX comments. This file will be automatically included in the LaTeX target and therefore the `pdf`.

Unfortunately the macros cannot be automatically included in the `html` target. To make sure macros also work for the `html` target it is necessary to add the following line to the start of any `.qmd` chapter file.

```
{{< include _macro_processing.qmd >}}
```

> **!** Important
>
> This templates includes the `_macro_processing.qmd` file in the root directory. If this template is added to an existing project, this file must be manually added to the root of the project.

Details about this design can be found in in this GitHub discussion. The Quarto developers may add a better mechanism in the future.

A.3 Mathematical equations

To understand how Quarto deals with mathematical equations it is necessary to understand how equations are handled in `LaTeX` and `html` (i.e. the JavaScript libraries `mathjax` or `KaTeX`). Readers that understand equations in Quarto and their limitations may want to skip to Section A.3.3 where the `aligned-overset` feature is discussed that is special to this template.

`LaTeX` is built on `TeX` which simply toggles between 'normal mode' and 'math mode' using dollar signs e.g. `$e^{i\pi} + 1 = 0$` renders as: $e^{i\pi} + 1 = 0$. For display equations `TeX` allows for the use of double dollar signs, e.g.

```
$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$
```

which is rendered like this

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

JavaScript libraries like mathjax or katex only implement math mode and generally scan for this dollar syntax in an `html` page to process it. However, at least mathjax can also be configured to work with the more advanced equation environments provided by LaTeX.

A.3.1 Equations in LaTeX

`LaTeX` built more sophisticated display equation environments on top of this simple toggle and it is therefore not recommended to use these primitive `TeX` macros for display equations in LaTeX. Instead of `$$ math $$` LaTeX authors are encouraged to use the syntax

```
\begin{environment}
    math
\end{environment}
```

Here the word `environment` is replaced by one of the following:

1. `equation` results in a numbered display equation, `equation*` in an un-numbered display equation.

   > **Note**
   >
   > `\[...\]` is an alias for the `equation*` environment.

   In contrast to the primitive `TeX` syntax (`$$ ... $$`), `LaTeX` equations adjusts the spacing of the equation depending on the length of the sentence before and after the equation. If the line before the equation only contains a single word, the equation is packed tighter since there is already a lot of white space to the right of this word.

2. `align` is a numbered display equation environment that allows the user to set alignment points with `&` letters. For example

   ```
   \begin{align}
       (a + b)^2
       &= (a + b)(a + b) \\
       &= a^2 + ab + ba + b^2 \\
       &= a^2 + 2ab + b^2
   \end{align}
   ```

   which is roughly[2] rendered as

   $$\begin{aligned}(a+b)^2 &= (a+b)(a+b) &(1)\\ &= a^2 + ab + ba + b^2 &(2)\\ &= a^2 + 2ab + b^2 &(3)\end{aligned}$$

   The `align*` environment is again the un-numbered pendant which results in

   $$\begin{aligned}(a+b)^2 &= (a+b)(a+b)\\ &= a^2 + ab + ba + b^2\\ &= a^2 + 2ab + b^2\end{aligned}$$

3. `alignat` (or `alignat*`) works similar to `align` (or `align*`), but allows for multiple alignment columns.

A.3.2 Equations in Quarto

Quarto essentially only allows the dollar sign syntax. However, the display equations using double dollar signs (`$$ ... $$`) are converted to LaTeX

[2] due to the issues with equation numbering in quarto explained in Section A.3.2 every line was manually tagged using `\tag` here instead of automatically numbered.

`equation` environments when targeting `pdf` via `LaTeX`. More specifically, un-annotated equations are converted to un-numbered `equation*` environments while a 'tagged equation'

```
$$
    f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}
$$ {#eq-differentiation}
```

is converted into a numbered equation

```
\begin{equation}
    \label{eq-differentiation}
    f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}
\end{equation}
```

rendered as

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{A.1}$$

The key `@eq-differentiation` can be used to produce a link (Equation A.1).

Unfortunately Quarto does not support aligned equations using the `align` environment. However it is possible to use the `aligned` environment inside of math mode; similarly the `alignedat` environment can be used in math mode in place of `alignat`.

To produce an aligned equation one can therefore use the syntax

```
$$
\begin{aligned}
    (a + b)^2
    &= (a + b)(a + b) \\
    &= a^2 + ab + ba + b^2 \\
    &= a^2 + 2ab + b^2
\end{aligned}
$$ {#eq-aligned-demo}
```

This renders as

$$\begin{aligned} (a + b)^2 &= (a + b)(a + b) \\ &= a^2 + ab + ba + b^2 \\ &= a^2 + 2ab + b^2 \end{aligned} \tag{A.2}$$

Observe that the 'Quarto-way' to tag equations only allows for one label per equation which is different from line-by-line numbering that LaTeX provides with the `align` environment. This is an unsolved issue at the moment.

There are two possible **workarounds**. The first workaround I strongly recommend against, under the assumption that:

- You want to reference equations from different chapters.
- You want to keep the equation preview on hover in `html`.

I recommend the second workaround if you are willing to duplicate the equation for the few cases you actually need to reference particular lines in aligned equations. This assumes that:

- Aligned equations explain how to get from the first element to the last. Afterwards the steps in the middle can generally be forgotten. So in the vast majority of cases it is unnecessary to label individual lines.

Workaround 1: Ignore Quarto

Recall that `mathjax` *can* deal with LaTeX equation environments? Instead of the dollar syntax you can simply place naked `\begin{...} \end{...}` tags into your `.qmd` files. This will be correctly rendered by `mathjax` in `html` and will correctly be converted into `LaTeX`. But there are caveats.
The first caveat can be fixed: `mathjax` does not treat `equation` differently from `equation*` by default and omits equation numbers altogether by default. This setting can be changed and `Mathjax` is already configured to behave like LaTeX in this template (this is not the default of Quarto). But since the `mathjax` equation numbering is independent of Quartos, the numbering is going to be incorrect if you use both. For this reason some authors recommend never using the Quarto syntax and only relying on `mathjax` for equation numbering. To do so simply use the `\label{eq-name}` and `\ref{eq-name}` tags you may know from `LaTeX`. However,

1. `mathjax` cannot see anything beyond the current `html` page. In particular it cannot see the equations in other chapters, which makes it impossible to reference equations from other chapters with this approach.
2. The hover-over (e.g. Equation A.2) Quarto provides will be broken
3. `mathjax` does not know the chapter it is in, so it cannot easily use the format `(1.3)` where `1` refers to the chapter and `3` means it is the third equation of this chapter. The `tagformat` of mathjax can however be changed and this template implements a heuristic to find the correct chapter number (assumes there is only one top level heading per `.qmd` file).

Workaround 2: Case-by-case treatment

Typically, you do not need to reference individual lines in an equation. The following workaround works well if you only need this in a handful of places and only reference these individual equations within the same chapter. It works by duplicating the equation and providing a version for

the `html` target separately for a version for the `pdf` (i.e. `LaTeX` target). It looks like this:

$$a = b \tag{A.3}$$
$$= c \tag{A.4}$$

< Text before you want to reference the equation.> Equation reference: Equation A.3
This is achieved using the following code that introduces a case-by-case treatment of the output

```
::: {.content-visible when-format="html"}
$$
    \begin{align}
        a &= b \tag{a}\\
        &=c \tag{b}
    \end{align}
$$ {#eq-test}
:::
::: {.content-visible when-format="pdf"}
    \begin{align}
        a &= b \label{eq-a}\\
        &=c \label{eq-b}
    \end{align}
:::

`<` Text before you want to reference the equation. `>`
Equation reference:
[@eq-test (a)]{.content-visible when-format="html"}
[Equation \ref{eq-a}]{.content-visible when-format="pdf"}
```

Note that `\begin{align}` is invalid syntax inside the math environment for `LaTeX` but it is accepted by `mathjax`. This allows double tags for the `html` target. In `LaTeX` you cannot have these duplicate tags, but you can use the fact that Quarto converts equation references into the `LaTeX` reference system and `LaTeX` will then take all labels into consideration and the numbering simply works. Check both outputs!

A.3.3 Aligned-Overset

A `LaTeX` package I love is `aligned-overset`. To understand what it does let me first explain the command `\overset` with the following demo

$$x \overset{\text{due to b}}{=} y.$$

This demo is produced by the code

```
$$
    x \overset{\text{due to b}}{=} y.
$$
```

The command `\overset{over}{under}` essentially places the first input over the second input. It is therefore a wonderful tool for explaining equation steps. In an aligned environment however this typically breaks alignment because the entire 'stack' is left-aligned if the alignment symbol `&` is placed before. And the explanation is usually longer than the `=`, like here

$$\begin{aligned} x_0 &= x_1 \\ &\overset{\text{due to b}}{=} x_2 \\ &= x_3. \end{aligned}$$

`aligned-overset` allows the placement of the alignment symbol `&` between the `{over}` and `{under}` of `\overset`, i.e.

```
\overset{over}&{under}
```

and aligns the stack only on the `under` part. With this feature it is possible to align the equations above like so

$$\begin{aligned} x_0 &= x_1 \\ &\overset{\text{due to b}}{=} x_2 \\ &= x_3 \end{aligned}$$

By default `mathjax` does not implement the `aligned-overset` package and therefore does not allow this placement of the alignment symbol `&`. However, a `mathjax` maintainer (Davide P. Cervone) showed me how to configure `mathjax` to allow this placement (see this GitHub issue). This is why the equation above works. However his solution only correctly adjusts the white-space to the right and may result in overlaps on the left, e.g.

$$\begin{aligned} x^2 &\overset{\text{due to b}}{=} y \\ &= z. \end{aligned}$$

Nevertheless for all equations, except for the first, this is a good solution and works out of the box with this template.

A.3.4 Theorem environments

Theorem A.1 (Demo Theorem)**.** *Quarto provides certain theorem environments out of the box (see documentation)*

Proof name. Usage

```
::: {#thm-demo name="Demo Theorem"}
content
:::

::: {.proof name="proof name"}
content
:::
```

□

Theorem A.2 (Links work)**.** *See Theorem A.1*

Lemma A.1. *Different types of theorems however do not use the same numbering. There is no such thing as theorem groups as in LaTeX amsmath.*

A.4 Publishing the book as a website

Quarto documents many ways to publish a Quarto book as a website, the way I recommend uses Github pages and the command `quarto publish`. This is a direct link to this part of the documentation.

Appendix B

Useful extensions

You may find the following quarto extensions useful

1. **Callouty Theorems** wraps Theorems and Proofs into a callout block of your choosing (in particular proofs can be made collapsible)
2. **Latex Environment** wraps quarto divs into a LaTeX environment of your choosing.
3. **Quarto TikZ** A filter that renders PGF/TikZ diagrams in HTML as SVG.
4. **honeypot** Add hidden instructions to HTML homework assignments to help detect cheating by unauthorized LLM usage.
5. **Font awesome Extension** This extension provides support including free icons (e.g. 👍) provided by Font Awesome.

References

[1] Donald E. Knuth. “Literate Programming”. In: *Comput. J.* 27.2 (May 1984), pp. 97–111. ISSN: 0010-4620. DOI: 10.1093/comjnl/27.2.97. URL: https://doi.org/10.1093/comjnl/27.2.97.