What is computational publishing?

Simon Bowie

Community-led Open Publication Infrastructures for Monographs (COPIM)

Published on: Jul 07, 2022

URL: https://copim.pubpub.org/pub/computational-publishing

License: Creative Commons Attribution 4.0 International License (CC-BY 4.0)
Computational publishing is an emerging area of experimental book publishing that we are investigating as part of COPIM’s Work Package 6 on experimental book publishing and reuse. There’s a lot to unpack in the term ‘computational publishing’: computers are used for a range of processes in all types of publishing from large-scale commercial publishing to small-scale zine publishing. What is computational publishing and how is it distinguished from the regular use of computation in publishing?

As a broad definition, we use ‘computational publishing’ to refer to publishing a book which combines human-readable text and computational functionality. However this simple definition requires some refinement in order to distinguish the contemporary experimental publishing practice from historical precursors like the hyperlinked publishing that forms the basis of the World Wide Web and interpreted programming languages which combine text and machine-readable computer code.

Contemporary computational book publishing refers to publishing which incorporates new and more advanced computational elements into human-readable books. Andrew Odewahn (2017) outlines a few of the computational elements that can be incorporated into computational publications including but not limited to:

- rich dynamic media that can play in a browser;
- interactive data visualisations;
- executable code blocks;
- data repositories.

Computational book publications enable the reader to run code within the book itself. This can be used to demonstrate an example of computer programming or to dynamically adjust a data visualisation to alter parameters. Bret Victor’s interactive essay Explorable Explanations (Victor, 2011) provides a few examples of this in a website format combining HTML and JavaScript. The gif below illustrates how a reader can manipulate a computational book to dynamically change the data represented in an example encouraging what Victor refers to as “active reading”.

[Image of a computational book with interactive elements]

1. [Footnote:](#)
2. Explorable Examples

The following is a typical description of a digital filter, as you might find in a typical textbook.

Below is a simplified digital adaptation of the analog state variable filter.

This topology is particularly useful for embedded audio processing, because $F_c$ (cutoff frequency) and $Q$ (resonance) are controlled by independent coefficients, $k_f$ and $k_q$.

(With most filters, the coefficients are functions of both parameters, which precludes pre-calculated lookup tables.)

The coefficients and transfer function are:

$$k_f = 2 \sin \left(\frac{\pi F_c}{F_s}\right) \quad k_q = \frac{1}{Q}$$

$$H(z) = \frac{k_f^2}{1 - (2 - k_f(k_f + k_q))z^{-1} + (1 - k_fk_q)z^{-2}}$$

Some example frequency responses:

- $F_c = 417$ Hz, $Q = 0.8$
- $F_c = 121$ KHz, $Q = 3.03$

Gif of interaction in Victor’s Explorable Explanations essay showing a reader dynamically adjusting parameters on a technical diagram.

A good example of a computational book is Winnie Soon & Geoff Cox’s book *Aesthetic Programming: A Handbook of Software Studies* *(Soon & Cox, 2020, p. 17)* published by Open Humanities Press. As well as the ‘frozen’ hardcopy and PDF version of the book, the book is available as a *GitLab repository* and as a *static site* generated from the repository. This allows the reader to actively run the various examples of JavaScript programming in the book in their web browser. Soon & Cox write that “text is in code […] and code is in text”:
the book sets out to express how writing and coding are deeply entangled, and how neither should be privileged over the other: we learn from their relationality. Writing code and writing about code are forced together in ways that reflect broader cultural and technical shifts in data practices and open publishing initiatives, and, moreover, emphasize that writing a book is necessarily a work in progress. Like software, this is a book to be read, and acted upon, shared and rewritten.

(Soon & Cox, 2020, p. 17)

Computational books often use software publishing tools such as interpreted programming languages or version control tools like Git or Apache Subversion. As Winnie Soon (2022) discussed at an Experimental Publishing seminar organised by Coventry University’s Centre for Postdigital Cultures, this means that computational book publishing can involve practices like collaborative writing or versioning that make writing a book similar to writing a piece of software. Static site generator tools like Jekyll or Hugo are also important for computational book publishing as they allow for the automated conversion of a code repository to a static website. Online publishing platforms like PubPub also allow some degree of computational publishing via integration with sites like CodePen. The sample code below combines HTML, CSS, and JavaScript to render an interactive model of the Solar System.

Visit the web version of this article to view interactive content.

However dedicated tools specifically for computational publishing are becoming more available and may require less software development skills than the tools mentioned above. Jupyter Notebook is a free and open source web application for creating and sharing computational documents and is becoming a popular choice for authors creating documents with live interactive code. Below there is an interactive example of a Jupyter Notebook document running using JupyterLite to provide a self-contained Jupyter environment in the browser. This notebook combines text documentation with examples of Python code running to create data visualisations.
Odewahn (2017) refers to Jupyter Notebook as a “computation-native authoring [tool]” distinguishing it from the bespoke software development required with a code repository and a static site generator. Jupyter Notebook documents can then be published on the web using a range of publishing frontends such as Curvenote or Jupyter Book.

As part of COPIM’s work package looking at experimental book publishing, we are working with the X-Sketchbook project investigating computational book publishing in the area of architecture. The project is using Jupyter Notebook, Jupyter Book, and Curvenote to investigate how an existing publisher’s infrastructures and workflows can incorporate computational publishing practices. We’re excited to be working with the TIB Open Science Lab on this project and look forward to seeing what can be done with computational book publishing in the future.

Footnotes

1. Or, more accurately, within certain digital versions of the book. Print versions or ‘frozen’ digital versions like PDFs may have ‘de-interactivized’ representations of the interactive code elements or alternative elements that work within a more traditional print context. ↩

Citations

http://www.openhumanitiespress.org/books/titles/aesthetic-programming/
