Introduction to literate programming

Riccardo Maria Gesuè
Fellowship Of Clean Code

GSSI

April 2023

Table of Contents

What is literate programming

Literate programming is a programming paradigm in which a computer program is given as an explanation of how it works in natural language.

- Embedded snippets of code.
- Traditional source code can be generated and compiled.
- Depending on the implementation, snippets can be executed and their output saved and/or printed.

Literate programming was first introduced in 1984 by Donald Knuth[5]. The implementation he developed was called WEB[6].

A first example

Literate programming is the combination of documentation and source code.

```
#+title: Exam notes
#+DESCRIPTION: Notes for the statistics exam a.a. 2022-2023
#+AUTHOR: Riccardo Maria Gesuè
#+EMAIL: gesue.riccardo@gssi.it
#+FILETAGS: :exam:statistics:
#+PROPERTY: header-args :noweb yes :tangle no :results silent
#+HTML_MATHJAX: align: left indent: 5em tagside: left

⊛ Introduction...
⊛ Assignment...
⊛ Article notes
**⋄ Article: Csedreki et al..
****○ Introduction...
****○ The neutron detector array...
****○ Background contributions...
****○ Efficiency measurement
******▷ The ⁵¹V(p,n)⁵¹Cr reaction
```

Measurements performed at Atomki.
The $^{51}V(p,n)^{51}Cr$ reaction ($Q = -1534.8\,keV$) was used to produce mono-energetic neutrons in the energy range below $1\,MeV$. This reaction is characterized by slow variations of the neutron intensity and energy with angle. Moreover the target preparation and utilization is well-known.
However its application is limited by the opening of additional neutron channels above $E_{p,lab} = 2330\,keV$, which corresponds to the first excited state of ^{51}Cr at $E_x = 749\,keV$.
Above this energy, neutrons of different energies might be mixed in the emitted neutron spectrum of the reaction. This contribution is negligible up to $E_{p,lab} = 2600\,keV$.
Therefore measurements were performed at $E_{p,lab} = 1700, 2000, 2300, 2600\,keV$ assuming mononergetic n_0 neutrons emitted at 130, 420, 710 and 990 keV energy, respectively.
The determination of the total number of emitted neutrons was based on the activation technique.
The number of reactions that take place during the ... t_i can be obtained as:

$$N_R = \frac{N_\gamma}{B\eta_{320}}\frac{e^{\lambda t_w}}{1-e^{-\lambda t_c}}\frac{\lambda t_i}{1-e^{-\lambda t_i}}$$

```
\[
N_{R} = \frac{N_{\gamma}}{B\eta_{320}}\frac{e^{\lambda t_{w}}}{1-e^{-\lambda t_{c}}} \frac{\lambda t_{i}}{1-e^{-\lambda t_{i}}}
\]
```

where N_γ is the number of detected 320 keV γ rays, t_c is the counting time, t_w is the waiting time elapsed between the end of the irradiation and the start of the counting, and λ is the decay constant.

```
**⋄ Compute cross section with constant s-factor
```

The total reaction cross-section can be expressed in terms of the astrophysical s-factor as

$$\sigma(E) = \frac{S(E)e^{-2\pi\eta}}{E} \tag{1}$$

Where $\eta = \frac{Z_1 Z_2 e^2}{4\pi\epsilon_0\hbar v}$ is the Sommerfeld parameter.
With energy in the center of mass system in units of keV and the reduced mass μ given in atomic mass units:

$$2\pi\eta = 31.29 \times Z_1 Z_2\sqrt{\frac{\mu}{E}}.$$

```
****○ Code
******▷ Penetrability
```

Define a function.

```python
#+name:F_penetrability
#+begin_src python
def CalculateTwopieta(z1:int, z2:int, m1:float, m2:float, E:float)->float:
    k: float = 31.29
    mu: float = m1*m2/(m1+m2)
    twopieta: float = k * z1 * z2 * np.sqrt(mu / E)
    return twopieta
def CalculatePenetrability(z1:int, z2:int, m1:float, m2:float, E:float)->float:
    twopieta: float = CalculateTwopieta(z1, z2, m1, m2, E)
    penetrability: float = np.exp(-twopieta)
    return penetrability
#+end_src
```

In addition the moderator was surrounded with 25.3 mm (vertical config) and 50.8 mm (horizontal config) thick layer of 5% borated polyethylene to further reduce the environmental neutron background.
In both setups, a cooling loop running deionised water at 5°C is integrated in the target holders for beam power dissipation (~100 W).

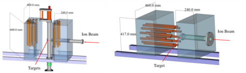

Figure 1: Vertical (left panel) and horizontal (right panel) setup of the LUNA neutron detector array. Dimensions of the moderators, target positions and beam direction are also indicated (see text for more details).

The vertical setup was used to measure the reaction cross section in the energy range $E_{\alpha,lab} = 360 - 400\,keV$, while the horizontal setup was used for the low energy measurements $E_{\alpha,lab} = 300 - 360\,keV$.

A first example

Typically end up with documentation file(s) in some readable format and source file(s) in your language(s) of choice.

Table of Contents

5.3. Compute cross section with constant s-factor

The total reaction cross-section can be expressed in terms of the astrophysical s-factor as

$$\sigma(E) = \frac{S(E)e^{-2\pi\eta}}{E}$$

Where $\eta = \frac{Z_1 Z_2 e^2}{4\pi\epsilon_0 \hbar v}$ is the Sommerfeld parameter. With energy in the center of mass system in units of keV and the reduced mass μ given in atomic mass units:

$$2\pi\eta = 31.29 \times Z_1 Z_2 \sqrt{\frac{\mu}{E}}$$

.

5.3.1. Code

1. Penetrability

   Define a function.

```python
def CalculateTwopieta(z1:int, z2:int, m1:float, m2:float, E:float)->float:
    k: float = 31.29
    mu: float = m1*m2/(m1+m2)
    twopieta: float = k * z1 * z2 * np.sqrt(mu / E)
    return twopieta
def CalculatePenetrability(z1:int, z2:int, m1:float, m2:float, E:float)->float:
    twopieta: float = CalculateTwopieta(z1, z2, m1, m2, E)
    penetrability: float = np.exp(-twopieta)
    return penetrability
```

   Call the function to calculate the penetrability with the given parameters.

```python
penetrability: float = CalculatePenetrability(zp, zt, mp, mt, energy)
```

In the horizontal setup, a single, water cooled target was mounted at the end of the horizontal tube. This setup allows for a higher neutron detection efficiency and better cooling capacity, and was used for the low-energy measurements where detection efficiency was of particular importance. In addition the moderator was surrounded with 25.3 mm (vertical config) and 50.8 mm (horizontal config) thick layer of 5% borated polyethylene to further reduce the environmental neutron background. In both setups, a cooling loop running deionised water at 5°C is integrated in the target holders for beam power dissipation (~100 W).

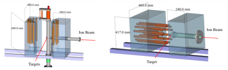

Figure 1: Vertical (left panel) and horizontal (right panel) setup of the LUNA neutron detector array. Dimensions of the moderators, target positions and beam direction are also indicated (see text for more details).

The vertical setup was used to measure the reaction cross section in the energy range $E_{\alpha,lab} = 360 - 400\,keV$, while the horizontal setup was used for the low energy measurements $E_{\alpha,lab} = 300 - 360\,keV$.

Table of Contents

Advantages of literate programming

- Focus on description of the approach in human-readable form.
- Follow human logic instead of computer logic.
- Understandable code.
- Open and reproducible research[8].
- Very easy to go back and revise an old (>2h) program.

Disadvantages of literate programming

- Can make coding slower.
- Can be difficult.
- Can be complicated to write bigger programs (arguable).
- Not many tools.

My use case

- Analysis in multiple steps.
- Not extremely complicated computationally.
- Physics/maths behind it requires attention.
- Started coding as undergrad, became more skilled/knowledgeable (did I?) during time.
- Show results in the most comprehensive and comprehensible way.
- Poor memory.

Table of Contents

Tools

A literate programming tool must be able to execute code blocks and/or export (tangle) them to an executable file. It should make it easy to write both code and plain text: it must work as both an IDE and a writing tool.

Jupyter [7] It is the most widespread tool.

- Browser based.
- Developed for Python.
- needs extensions to work with other languages(as far as I know only supports C++, ROOT, R).

Babel [1] Functionality of Emacs[3] org mode[4].

- This is what I personally use.

Others Check here for a complete list: `https://literateprogramming.org`

GNU Emacs

GNU Emacs is an extensible, customizable text editor.

- You can learn more about Emacs functionalities here: `https://www.gnu.org/software/emacs/tour/index.html`
- It has notoriously a steep learning curve:
  - Keyboard oriented.
  - Many different functionalities.
  - Loads of extensions.
- **But** it is less difficult than you would think:
  - Several different “distributions” with beginner friendly defaults, i.e. DOOM Emacs[2], Spacemacs[10].
  - Great documentation.
  - Command prompt and intuitive command names.
  - *Mouse* support.

Emacs is the IDE part of our tool.

Org mode

From the website

A GNU Emacs major mode for keeping notes, authoring documents, computational notebooks, literate programming, maintaining to-do lists, planning projects, and more — in a fast and effective plain text system.

Org is a highly flexible structured plain text file format, composed of a few simple, yet versatile, structures — constructed to be both simple enough for the novice and powerful enough for the expert.

Org is the “plain text”/text editor part of our tool.

Babel

- Babel is Org's ability to execute and tangle source code within Org documents.
- It was developed as a tool for literate programming and reproducible research[9].
- Babel supports a growing number of programming languages; more than ten dozen languages currently have some Babel support.
- The core Babel functions (viewing, export, tangling, etc.) are language agnostic and will work even for languages that are not explicitly supported.
- Explicit language-specific support is required only for evaluation of code blocks in a language.
- Babel is designed to be easily extended to support new languages.

Babel is the "code" part of our tool.

Typical document structure

```
#+title: Exam notes
#+DESCRIPTION: Notes for the statistics exam a.a. 2022-2023
#+AUTHOR: Riccardo Maria Gesuè
#+EMAIL: gesue.riccardo@gssi.it
#+FILETAGS: :exam:statistics:
#+PROPERTY: header-args :noweb yes :tangle no :results silent
#+HTML_MATHJAX: align: left indent: 5em tagside: left
```

Metadata

```
⊙ Introduction...
⊙ Assignment...
⊙ Article notes
**⋄ Article: Csedreki et al..
****○ Introduction...
****○ The neutron detector array...
****○ Background contributions...
```

Sections

```
*⋄ Compute cross section with constant s-factor
```

The total reaction cross-section can be expressed in terms of the astrophysical s-factor as

$$\sigma(E) = \frac{S(E)e^{-2\pi\eta}}{E} \quad (1)$$

Where $\eta = \frac{Z_1 Z_2 e^2}{4\pi\epsilon_0 \hbar v}$ is the Sommerfeld parameter.

With energy in the center of mass system in units of keV and the reduced mass μ given in atomic mass units:

$$2\pi\eta = 31.29 \times Z_1 Z_2 \sqrt{\frac{\mu}{E}}.$$

Text with LaTeX formatting

```
****○ Code
******▷ Penetrability
```

Define a function.

```
#+name:F_penetrability
#+begin_src python
def CalculateTwopieta(z1:int, z2:int, m1:float, m2:float, E:float)->float:
    k: float = 31.29
    mu: float = m1*m2/(m1+m2)
    twopieta: float = k * z1 * z2 * np.sqrt(mu / E)
    return twopieta
def CalculatePenetrability(z1:int, z2:int, m1:float, m2:float, E:float)->float:
    twopieta: float = CalculateTwopieta(z1, z2, m1, m2, E)
    penetrability: float = np.exp(-twopieta)
    return penetrability
#+end_src
```

Code snippets

```
**⋄ Run the code :noexport:
  #+name:execute
  #+begin_src python :results output replace :noweb yes :tangle test.py
  <<imports>>
  <<imports_sensitivity>>
  <<pretty_plots>>
  <<parameters>>
  <<F_penetrability>>
  <<F_sigma>>
  <<penetrability>>
  <<sfactor_convert>>
  <<energy_convert>>
  <<sigma>>
  <<F_stoppingpower>>
  <<F_stoppingpower_sane>>
  <<F_yield_correct>>
  <<yield_correct>>
  <<F_charge>>
  <<charge>>
  <<F_nevents>>
  <<nevents>>
  #------Sensitivity------#
  <<F_poisson>>
  <<F_gauss>>
  <<F_bkg_counts>>
  <<F_bsyst>>
  <<F_bkg_gen>>
  <<F_eff>>
  <<F_ns>>
  <<F_eff_correct>>
  <<F_eff_dice>>
  <<F_likelihood>>
  <<F_negloglikelihood>>
  <<F_minloglikelihood>>
  <<F_bkglikelihood>>
  <<Lparameters>>
  # <<gen_data>>
  # <<likelihood>>
  # <<maxlikelihood>>
  <<onetoy>>
  <<ntoys>>
  <<ntoys_plot>>
  plot: bool = False
  if plot:
      <<plot_data>>
  #+end_src
```

Tangling and running the code

Table of Contents

Useful links I

- *Babel*. https://orgmode.org/worg/org-contrib/babel/index.html.
- *DOOM emacs*. https://github.com/doomemacs/doomemacs.
- *Emacs*. https://emacs.org.
- *Emacs Org mode*. https://orgmode.org/.
- D. E. Knuth. “Literate Programming”. In: *The Computer Journal* 27.2 (Jan. 1984), pp. 97–111. ISSN: 0010-4620. DOI: 10.1093/comjnl/27.2.97. eprint: https://academic.oup.com/comjnl/article-pdf/27/2/97/981657/270097.pdf. URL: https://doi.org/10.1093/comjnl/27.2.97.
- Donald E. Knuth. *The WEB package*. https://www.ctan.org/pkg/web.
- *Project jupyter*. https://jupyter.org.
- *Reproducible research*. http://reproducibleresearch.net.

Useful links II

- Eric Schulte et al. “A Multi-Language Computing Environment for Literate Programming and Reproducible Research”. In: *Journal of Statistical Software* 46.3 (2012), pp. 1–24. DOI: 10.18637/jss.v046.i03. URL: https://www.jstatsoft.org/index.php/jss/article/view/v046i03.
- *Spacemacs*. https://www.spacemacs.org/.

Table of Contents

Questions