

Kilian Lieret, Alejandro Mora, Moritz Bauer, Michel Bertemes, Sviatoslav Bilokin, Giulia Casarosa, Racha Cheaib, Samuel Cunliffe, Angelo Di Canto, Giacomo De Pietro, Nathalie Eberlein, Michael Eliachevitch, Marcela Garcia Hernandez, Philip Grace, Daniel Greenwald, Oskar Hartbrich, Michel Hernandez Villanueva, Chia-Ling Hsu, Ilya Komarov, Thomas Kuhr, Yaroslav Kulii, Frank Meier, Marco Milesi, Gian Luca Pinna Angioni, Markus Tobias Prim, Martin Ritter, Pascal Schmolz, Umberto Tamponi, Francesco Tenchini, Hannah Wakeling, Xing-Yu Zhou **Contact:** <u>kilian.lieret@posteo.de</u>, <u>software-doc@belle2.org</u>

Abstract

The physics output of modern experimental HEP collaborations hinges not only on the quality of its software but also on the ability of the collaborators to make the best possible use of it. With the COVID-19 pandemic making in-person training impossible, the training paradigm at Belle II was shifted towards one of guided self-study.

To that end, the study material was rebuilt from scratch as a series of modular and hands-on lessons tightly integrated with the software documentation on Sphinx. Each lesson contains multiple exercises that are supplemented with hints and complete solutions. Rather than duplicating information, students are systematically taught to work with the API documentation to find the important sections for themselves. Unit tests ensure that all examples work with different software versions, and feedback buttons make it easy to submit comments for improvements.

On this poster, we detail our experiences with the new setup and training model.

Challenges

The training model at Belle II faces the following challenges:

- Suitability for both self-study and in-person events
- Versioning: Analyst-facing API of the Belle II software is still evolving \Rightarrow different versions of lessons for different versions of software
- **Testability:** Code snippets in the lessons should be tested to ensure that they are working
- Maintainability and sustainability: Lessons should be consistent, stable, and easy to update.
- **Interactivity:** Exercises improve the learning experience and keep students engaged

Some trade-offs between these points are necessary. For example, self-study-ready material requires a level of verbosity that may be overwhelming when presented in traditional in-person events.

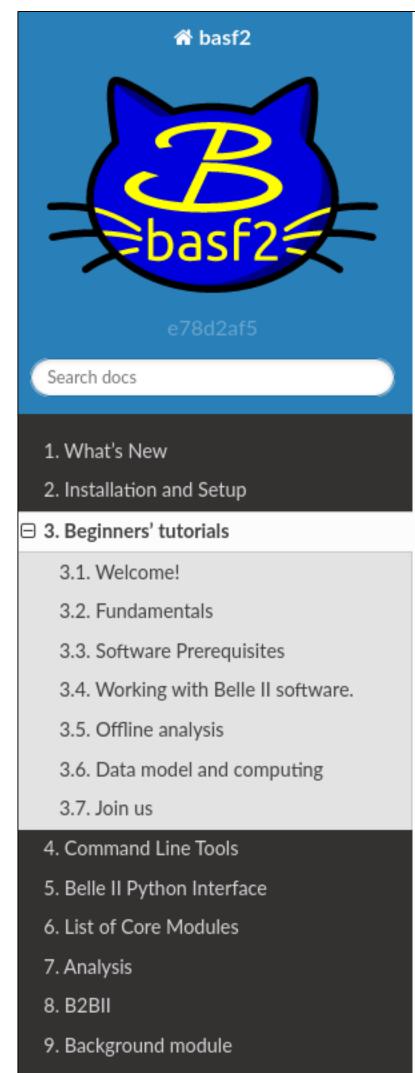
General principles

- Lessons are available as web pages built with <u>Sphinx</u>¹
- Verbose text. Didactical style inspired by work of <u>software carpentries² / HSF training³ / LHCb StarterKit⁴</u>
- Source code is hosted in git together with the main **software** (basf2, now <u>open source⁵!</u>) \Rightarrow Natural correspondence of software versions and

lesson versions \Rightarrow Good for unit testing

- **Do not replicate existing documentation!** Either link to it or make it an exercise for students to find the information.
 - Helps with maintainability
 - Teaches students to work with "normal" documentation
- **Refer to outside training** for some software prerequisites (e.g., python). Currently using material of sw carpentries.
- **Events:** Guided self-study with Q & A sessions, mentoring sessions, and self-study time in between. Chat channels for asynchronous help.

A new Software Training Model at Belle II



Calibration

67

View page source We want YOU to contribute to this book! Tip Just as there are many versions of the Belle 2 software, there are many versions of this documentation to match it. After all, if a new feature is added in our software, we also want to have the documentation for it. The current version of this documentation is shown on the top left of this page, just below the logo. You can also change your version by clicking on other versions. If you are a new to all of this, we recommend you to select the recommended release version (release-xx-xx-xx (recommended) in the above list). You can also take a sneak peek at the most recent version of the documentation by selecting the development version. However not all of the code examples might work for you yet. The earliest release version which contains this online book is release-05-01-12. • Warning If you change the version to an earlier version than the current one, some pages (also this page!) might not exist. **45 figures** 🕸 Overview Teaching: 10-20 min Exercises: 10-20 mir Prerequisites

🕷 » 3. Beginners' tutorials 3. Beginners' tutorials This online textbook aims to help new Belle II members to get started with the software by following through a series of hands-on lessons. Figure 1. Home of the lessons at training.belle2.org overview

244 code snippets = ["roeE()", "roeM()", "roeP()", "roeMbc()", "roeDeltae()"] 68 b vars += roe kinematics + roe multiplicities

57x keeping an The previous lesson The ROE of a selection is build with buildRestOfEven For many analyses ROE is used as middleware to get tag vertex fit, continuum suppre • Usage of ROE without a mask is not recommended.

212 exercises, 159 hints, 191 solutions ⑦ Task

Add PID and track variables for all charged final state particles and the invariant mass of the intermediate resonances to the ntuple. Also add the standard variables from before for all particles in the decay chain, the kinematics both in the lab and the CMS frame.

♀ Hint: Where to look ▼ Wint: Partial solution for final state particles

- ♀ Hint: CMS variables ▼
- Hint: Partial solution for the CMS variables Solution

Quick feedback form below every lesson

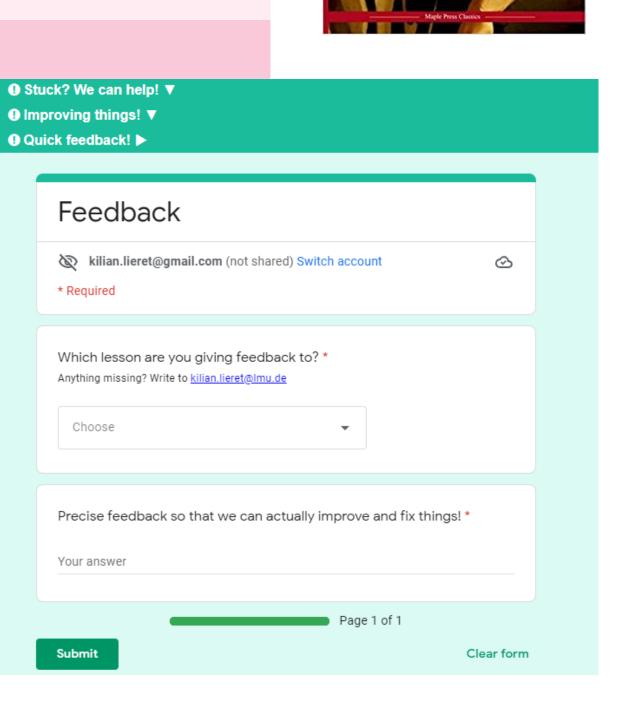
Figure 2. Features and statistics

How happ 33 responses		ou with 1	the lesso	ons provi	ded?	20	20					How happy 24 responses 10.0
20											16 (48.5%)	7.5
15												5.0
10								5 (15.2%)				2.5
5	0 (0%)	1 (3%)	1 (3%)	1 (3%)	0 (0%)	2 (6.1%)	1 (3%)		3 (9.1%)	3 (9.1%)		0.0
0 ·	0	1	2	3	4	5	6	7	8	9	10	0.0 -

Figure 3. Customer satisfaction: "How happy were you with the lessons?"

I combined several particle nto 🗴 . How do I selec everything that is not "in" 🗴

from this Rest of Event / what

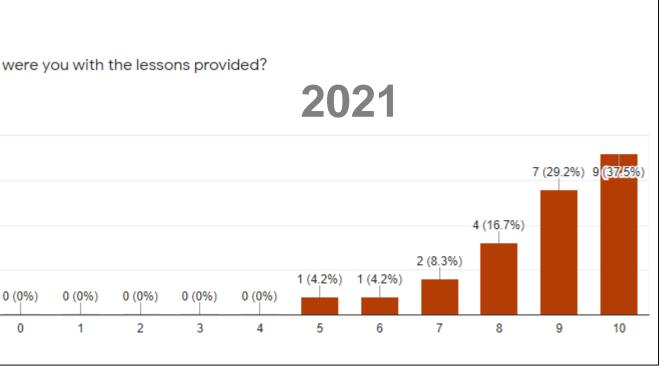


2.5x Hamlet

(excluding most code)

HAMLET

POG



Technical setup

- branches

Code inclusion:

- issues

Experiences

- (rst), and sphinx directives.

- unit testing
- ¹ https://www.sphinx-doc.org/
- ² https://software-carpentry.org/
- ³ https://hepsoftwarefoundation.org/training





Continuous roll out: Developed on the main branch of the basf2 git repository (view package on github⁶) Recommended training versions live on release

• E.g., training.belle2.org automatically points at lessons of the latest full release

• For substantial improvements/hotfixes: Needs cherrypicking into minor or fix-releases

Generally, this means that changes propagate relatively slowly (but you can always point a student to the latest development build)

 Put code in separate files and use code inclusion • These files are then executed as unit tests

• Allows to include complete "solutions" at the end • Allows using code-inspection/formatting tools for a uniform experience and fewer bugs

• For step-by-step build-up of larger code examples: Use partial code-inclusion

• Initial version: line-number based \Rightarrow very cumbersome with code changes as line numbers need to be updated Now: include small marker comments in code and use start-after/end-before directives of Sphinx

 \Rightarrow Using explicit markers consistently solves ambiguity

 \Rightarrow makes it obvious where code is included

Additional admonition types for Sphinx: e.g., exercise blocks, foldable hints and solutions (Fig. 2) • Quick **feedback form** below every lesson for bug

reports, however still rarely used (Fig. 2)

Technical complexity: Contributions require basic knowledge of git, PR workflows, reStructuredText

• Automatic Sphinx builds in PRs for preview & checks, but take ~20 min. Local builds are faster but require initial compilation of (part of) main software + 1-3 min per Sphinx build.

Contributing to documentation teaches the entire software development workflow

 Additional step-by-step tutorials and dedicated hackathons could help beginners to contribute, growing number of unique contributors

Very complete onboarding experience, even for newcomers who join "off-season." Lessons cover everything from basic physics knowledge and software prerequisites to submitting grid jobs.

Very **positive feedback** (Fig. 3), e.g., "Very solid work regarding the textbook! Congrats! Everything was very clear which significantly minimized the need for guidance (...)Software Prerequisites was the highlight of the workshop as it summarized all the necessary tools that no one really spends time on explaining thoroughly to newcomers."

Very few issues with broken code snippets due to

⁴ https://lhcb.github.io/starterkit-lessons/

⁵ https://github.com/belle2/basf2