Improving LbMCSubmit

Project report for CERN summer student programme 2022

Simon Thor

Supervisors: Chris Burr and Adam Morris

August 26, 2022

Abstract

I present my work during the CERN Summer Student Programme 2022. I have contributed to LbMCSubmit, a program which simplifies the process of submitting simulation production requests at LHCb. My work can be divided into two main parts: adding support for non-proton-proton (pp) collisions in LbMCSubmit, and adding an interface between LbMCSubmit and `Professor`, a program used for parameter tuning in event generators. By adding support for non-pp collisions in LbMCSubmit and contributing to other improvements, it is now possible to use LbMCSubmit for more use cases, thereby making it ready for wide-spread use within LHCb. By developing programs that integrates `Professor` with LbMCSubmit, it is now easier to perform tuning, as much of the complexity of `Professor` abstracted away, while the simulation speed is improved as the MC simulations run on the grid.

1 Simulations at LHCb

Simulations of particle collisions, which particles are produced, how these particles decay, interact with the detector, which selection cuts are applied to only select the most interesting events and more is done on the grid to speed up the time-consuming simulations. By having all simulations in a centralized manner also has additional benefits. Namely, all settings used are stored and everyone uses approved software without any highly customized code, making the results reproducible. Furthermore, the resources are shared equally among all users.

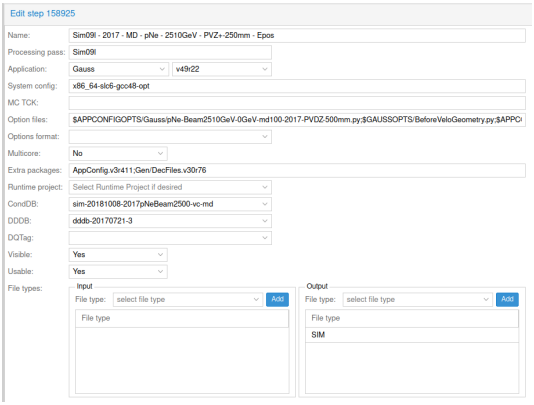

Figure 1: Example of what the web interface for simulation production requests looks like.

Currently, to specify the settings for this simulation, e.g., which software versions to use, which type of collision and interaction that should be simulated, is done using a web interface. This web interface is shown in Figure 1.

However, there are several problems with the current web interface. Firstly, When one wants to do many similar simulation production requests (e.g., one for each year that the LHCb detector was running, with different magnet polarities), the settings for every simulation production request must be done one by one, despite the fact that many of the settings are the same for these production requests. Secondly, when a new software version is released, all model production requests, which are used as templates for making actual production requests, must be updated manually by the maintainers. Not only does this updating process take several hours of work but it also means that models which are not used as frequently, e.g., non-proton-proton collision simulations, are not prioritized and thus not updated as frequently. Finally, since all of these simulation production requests are done by hand, it could and has lead to erroneous simulation production requests, which result in wasted computing resources.

2 LbMCSubmit

The solution to the above-mentioned problems is LbMCSubmit. When using LbMCSubmit, the user only needs to specify a short YAML file, which can then be passed to LbMCSubmit which will output a longer YAML file with all the settings, options, and software versions specified for each step in the simulation process. An example of an input YAML file to LbMCSubmit is shown below.

```yaml
sim-version: 09
name: My Analysis
inform:
  - auser
WG: Charm
samples:
  - event-types:
      - 27165175
      - 30000000
    data-types:
      - 2016
      - 2017
      - 2018
    num-events: 2_500_000
```

This will generate an output YAML file with settings all possible combinations of the three event types (i.e., what type of interaction that should take place), two data types (i.e., which year of the detector and beam that should be simulated) and two magnet polarities for the LHCb detector. This results in $3 \cdot 2 \cdot 2 = 12$ different simulation production requests which otherwise would need to be specified one by one. Additionally, it is still easy to customize the simulation production request, as one can simply edit the contents of the output YAML file from LbMCSubmit, should that be necessary.

While many features existed in LbMCSubmit before I began contributing to it, it was still in its development phase and was therefore not yet commisioned for wide-spread use. One missing feature was that the the latest software version for packages[1] was not used as the default. Another missing feature was that LbMCSubmit only supported pp collisions. There have however been many other types of collisions done at LHCb: proton-lead (pPb), lead-lead (PbPb), proton-Helium (pHe), proton-Neon (pNe), proton-Argon (pAr) and lead-Neon (PbNe). The four last collision types are SMOG collisions, which is when a noble gas (so far He, Ne, or Ar) is injected into the beam pipe and the accelerated particles collide with the gas[1]. The goal of the first part of my project was therefore to add these features.

3 Improvements to LbMCSubmit

Support for non-pp collisions new collision types was implemented by adding looking at previous production requests and the model requests. Based on this information, I identified which settings[2] were used for each part of the simulation stage. I then updated the code of LbMCSubmit so that it would output the correct options files for each collision type. Additionally, some collision types (PbPb and pPb) must also be supported for different data types (e.g., pPb collisions have been done at LHCb both in 2013 and 2016), which in turn also result in different settings that had to be taken into account for when writing the code. When some settings were inconsistent between different production requests, this was discussed with experts to resolve which settings should be used as the default. This also resulted in some discoveries of erroneous production requests that had been done.

The newly added support for different collision types resulted in a more complex code base, which in turn required some refactoring to make the code easier to maintain. While the intention of LbMCSubmit is to submit simulations to the grid, it can also be used for simulations locally. The test simulations I performed were done locally on `lxplus`, as only one event was simulated for each newly implemented collision type and data type. These simulations were performed to ensure that the output YAML file from LbMCSubmit would result in simulations that run without crashing. After this step, unit tests and documentation were added for how to use the new features and the code base was merged into the main branch.

[1] Specifically `AppConfig`, `ParticleGun` and `Gen/DecFiles`

[2] The settings for a simulation program, e.g., Gauss, is specified using Python files which are called options files.

Simulations of these collision types uses `Epos`, a different Monte Carlo event generator than `Pythia8`, which is the one that is usually used at LHCb for pp collisions. Adding support for non-pp collisions therefore also meant adding support for another MC event generator in LbMCSubmit.

Besides the non-pp collision support, I also changed the code so that LbMCSubmit automatically uses the latest version of certain software `AppConfig`, `Gen/DecFiles` and `ParticleGun` that is suitable for a certain simulation software version. This removes the need for hours of manual work to update model production requests every time a new version of these software is released.

Additionally, I also improved the documentation for how to contribute to LbMCSubmit, specifically how to run certain tests locally. This will make it easier for future contributors to get started.

Finally, I used LbMCSubmit to submit simulation production requests to the grid for other projects that would need these simulations. Besides the simulations being useful to the projects that needed them, this was also a good way of stress testing LbMCSubmit and the production request submission system. By using LbMCSubmit in production, bugs in the simulation production request system could be discovered, which could then be fixed (see e.g., this merge request).

My main contributions to the code base of LbMCSubmit can be found in the merge requests listed below (in reverse-chronological order):

- Add documentation about bats tests
- PbPb support
- SMOG
- Add support for 2016 pPb/Pbp collisions
- Add support for pPb and Pbp collisions in 2013
- Fix typos in documentation

4 Professor

`Professor` is a program used for tuning parameters in event generators such as `Pythia8` so that simulations are more accurate and similar to experimental measurements [2]. Tuning is the process where the one or more parameters in the event generator are varied. For each value that is tested for these parameters, a number of collisions are simulated using the event generator and the output data is analyzed, typically using `Rivet`, which outputs histograms [3]. These histograms are then compared to experimental measurements which have performed the same analysis. Based on these results, the parameter value(s) in the event generator that would result in the best match with experimental measurements can be determined and used to have a more accurate event generator. The typical workflow for `Professor` is shown in Figure 2. Additional intermediate steps can also be introduced into the `Professor` workflow.

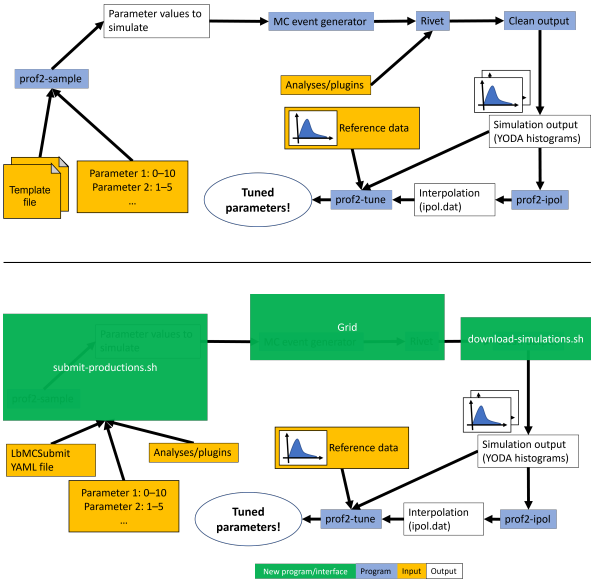

Figure 2: The standard workflow for `Professor` (top image) and the new workflow using the simpler interface (bottom image). A yellow block represents an input that the user has to provide, e.g., a file. Blue blocks correspond to programs or commands that are used within the workflow. White blocks represent output files from a program, which can then be passed to another program. The green blocks are sections that I have worked on to hide most of the complex and error-prone workflow from the user, to make it easier to tune parameters.

When tuning parameters, many events (typ-

ically $10^5 - 10^6$) must to be simulated for each point in the parameter space that is investigated. Running the simulations locally would be slow or even unfeasible. To solve this problem, during the second part of my project, I developed a program for running MC event generator simulations on the grid. Since LbMCSubmit simplifies the simulation production request process, it can be used under the hood for this program.

5 Improving the Professor workflow at LHCb

The program I developed works as follows (see also Figure 2 for a visual description): the user passes a YAML file that can be parsed by LbMCSubmit, the parameters that should be tuned and the region in parameter space which should be sampled, as well as the the Rivet analyses that should be applied to the simulation output from `Pythia8` to `submit-productions.sh`. This program will run a `Professor` command to randomly select points in the parameter region of interest. LbMCSubmit is then used to specify which settings should be used for the simulations. These two outputs will then be used to generate a set of files for each point in the parameter region. These files contain instructions and code for running the simulations, which will be submitted to the grid and thus start the simulations. Finally, instructions for what to do once the simulations have finished will be displayed.

On the grid, each simulation corresponding to a point in the parameter space is an independent job and they can therefore all run in parallel. The `Pythia8` output in the simulation will be passed to `Rivet`, which will apply the analyses that were specified by the user. The output from `Rivet` will be a set of histograms.

Once the simulations have finished running on the grid, the second program I developed, `download-simulations.sh`, can be used. This will download and clean the Rivet output histograms from the grid. Once this script has finished running, the downloaded data is ready to be used for tuning. Instructions for how to run the `Professor` commands to tune will be displayed. When running rest of the `Professor` commands, it will perform an interpolation between the randomly selected points in parameter space and compare the histograms generated from the simulations with reference histograms that are provided by the user. The parameter values which gives the best agreement between the simulation histograms and the reference histograms is then given as the output.

The source code and documentation for the programs I developed are publicly available on Gitlab.

There are two main benefits of the programs I developed: increased speed and improved ease of use. Since simulations now can run on the grid, simulations of different points in the parameter space can be simulated in parallel, thus improving the simulation time significantly. Additionally, since most of the complexity of how to run simulations, `Rivet` and `Professor` is hidden from the user, it is easier to perform tuning. The code I have developed will therefore make it easier to tune more parameters at LHCb to higher precision, which will improve the accuracy of the MC event generators used.

Besides developing these programs for tuning, there were several other benefits of my work. Firstly, While using `Professor`, I detected bugs in its source code. Fixes for these bugs were then sent to the maintainer of `Professor`. Secondly, since most of the documentation for `Professor` only existed for an old version of the program, understanding the workflow and how to use it is not trivial. Since I had gone through the process of learning the workflow and understanding how to install and work with `Professor`, I could also guide a PhD student at CERN on how to set up and work with it.

6 Parameter tuning

I performed a tune of six Pythia8 parameters:

- `StringFlav:mesonSvector`
- `StringFlav:mesonUDvector`
- `StringFlav:probQQ1toQQ0`
- `StringFlav:probQQtoQ`
- `StringFlav:probSQtoQQ`
- `StringFlav:probStoUD`

These parameters are all related to flavour composition and more information about them can be found in the Pythia8 documentation. These parameters were chosen for tuning as they had previously been tuned for Sim09 (the currently used simulation software stack version) and are therefore likely relevant to tune for Sim10, the upcoming release of the simulation software stack. Additionally, since it has previously been tuned, the output values of the new tune can be compared with the old tune for a sanity check.

The Rivet analyses chosen for this tune were LHCB_2015_I1396331 and the Rivet plugins LHCB_2018_I1670013 and LHCB_2015_I1391511, which are part of LbRivetPlugins [4]. 1000 sample points in the six-dimensional parameter space were randomly chosen and 100 000 events were simulated for each point on the grid. Three different tunes were performed: using the and an order 3 polynomial as interpolation (referred to as default), using an order 2 polynomial and ignoring the LHCB_2018_I1670013 analysis (order2), and an order 3 polynomial, also ignoring the LHCB_2018_I1670013 analysis (order3). The first tune is called "default" as the default settings for `prof2-ipol` and `prof2-tune` were used. The LHCB_2018_I1670013 analysis was ignored for two tunes as it still seems to be in development and the output histograms can therefore not be trusted.

In total, 16 histograms were used for tuning from LHCB_2015_I1391511 and LHCB_2015_I1396331, two of which are shown in Figure 3. As one can see, some histograms where tuned well to match the reference (bottom plot), whereas others did not (top plot). There are several potential reasons for this: the parameters are correlated to each other and can result in nearly non-continuous regions of goodness of fit in the parameter space, which makes optimization difficult. Another explanation could be that not enough sample points in the parameter space were simulated, resulting in a poor interpolation of the histograms. Finally, the statistical uncertainty can be reduced for a more accurate tune. Preferably 1 000 000 events could have been simulated for each parameter point.

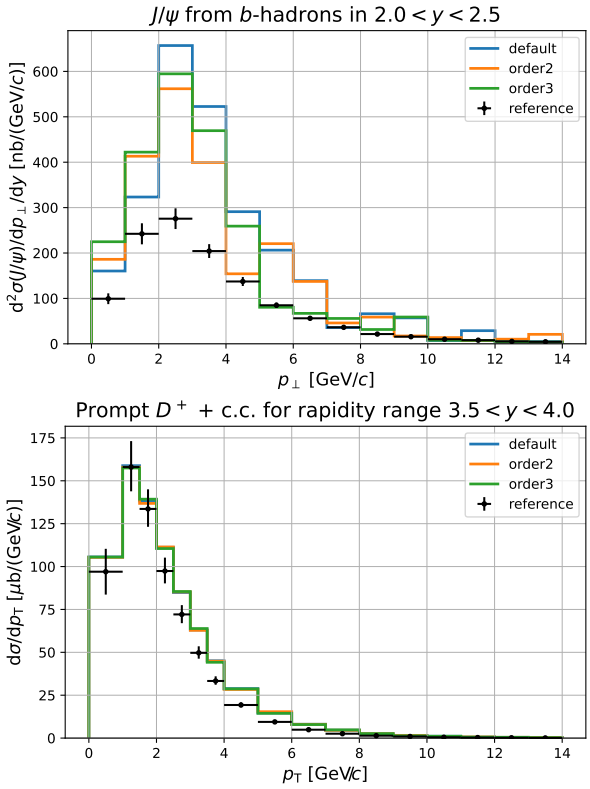

Figure 3: Interpolated histograms that correspond to the tuned parameters, compared to reference histograms. Here, only two histograms are shown instead of the 16 that are given from the two mian analyses that were used: the top one corresponding to LHCB_2015_I1391511/d02-x01-y01 and the bottom plot being LHCB_2015_I1396331/d02-x01-y04.

The tuned parameter values are shown in Table 1.

Parameter	def	o2	o3	Sim09
mesonSvector	1.55	0.57	1.12	0.74
mesonUDvector	1.64	1.56	1.89	0.60
probQQ1toQQ0	0.54	0.42	0.53	0.05
probQQtoQ	0.41	0.69	0.20	0.16
probSQtoQQ	0.58	0.26	0.31	0.40
probStoUD	0.42	0.23	0.40	0.35

Table 1: The values of the tuned parameters for the three different tunes that were performed, compared to the parameter values currently used at LHCb in Sim09. "def" means "default", "o2" means "order2" and "o3" means ´´order3".

Besides demonstrating that a tune can be

performed with the programs I have developed, this helped stress-test and identify bugs in the programs which would not have been discovered without performing a real tune.

7 Summary

During the summer student programme at CERN, I have worked on LbMCSubmit, a program which makes it easier to generate and submit simulation production requests at LHCb. My contributions can be divided into two parts: improving LbMCSubmit and creating a more user-friendly interface or `Professor`.

For the first part of my project, I have added support for all collision types that have been done at LHCb so far to LbMCSubmit. This new feature to LbMCSubmit means that it covers more use cases for simulations and makes it easier to simulate non-pp collisions, which otherwise had lacking support, as they were not prioritised as much as pp collisions. I also added other improvements to LbMCSubmit, such as automatically using the latest software, improving its documentation and stress-testing it.

For the second part of my project, I have created programs that can be used to submit simulation production requests for `Pythia8` and analyzing the simulation output using Rivet, all done on the grid. I also developed a program for downloading and cleaning the output of the simulations and `Rivet`. These two programs I developed makes it possible to easily run simulations for tuning on the grid, which significantly reduces the simulation time. Additionally, it makes it easier to use `Professor`, as it hides most of the complexity of the tuning workflow.

Finally, this project helped me better understand the various software used to simulate and analyze collision data at LHCb, as well as come in contact with many helpful developers and researchers at LHCb.

References

[1] CERN (Meyrin) LHCb Collaboration. LHCb SMOG Upgrade. Technical report, CERN, Geneva, May 2019. URL https://cds.cern.ch/record/2673690.

[2] Andy Buckley, Hendrik Hoeth, Heiko Lacker, Holger Schulz, and Jan Eike von Seggern. Systematic event generator tuning for the LHC. *The European Physical Journal C*, 65(1-2):331–357, nov 2009. doi: 10.1140/epjc/s10052-009-1196-7. URL https://doi.org/10.1140%2Fepjc%2Fs10052-009-1196-7.

[3] Christian Bierlich, Andy Buckley, Jonathan Butterworth, Christian Holm Christensen, Louie Corpe, David Grellscheid, Jan Fiete Grosse-Oetringhaus, Christian Gutschow, Przemyslaw Karczmarczyk, Jochen Klein, Leif Lönnblad, Christopher Samuel Pollard, Peter Richardson, Holger Schulz, and Frank Siegert. Robust independent validation of experiment and theory: Rivet version 3. *SciPost Physics*, 8(2), feb 2020. doi: 10.21468/scipostphys.8.2.026. URL https://doi.org/10.21468%2Fscipostphys.8.2.026.

[4] Alex Grecu. Lbrivetplugins. https://gitlab.cern.ch/lhcb/LbRivetPlugins, 2022.