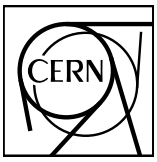
EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH



Summer Student Report
August 17, 2023

Validating and testing a biasing procedure using the O^2 simulations for ALICE

Sarah Andersson

Summer Student 2023

Supervisors: Mike Sas and Sandro Wenzel

Abstract

This project revolves around the current O^2 simulations of the ALICE detector performance for LHC run 3 and 4. The goal is to bias the simulations to efficiently increase the statistical sample of certain interesting events. Specifically looking into extracting events containing the photon conversions from neutral pions ($\pi^0 \to \gamma, \gamma \to e^+e^-, e^+e^-$) and events containing photon conversions from any source. Several validation tests of the biasing implementation have been done. These tests have so far yielded some unexplained results, that encourage further investigation into the biasing procedure. Additionally a run time bench marking analysis of the biasing versus no biasing has been computed. It was found that the biasing improves the run time by a factor of ~ 9 when wanting to compute ~ 300 events containing at least one $\gamma \to e^+e^-$ conversion.

Contents

1 Introduction

The current O^2[1] simulation framework of the ALICE detector uses Pythia8[2] as back end, then propagates the particles through the detector using Geant4[3] workers, which can be multithreaded on multiple cores and finally the resulting data is merged into one file. The main goal of this study is to validate the implementation of biasing in these simulations and to investigate the time gain of the biasing implementation compared to the standard full simulations.

The motivation for this study, is that certain physical processes occurs rare enough, that obtaining a sufficient statistical sample of events containing these processes by solely running the full simulation would be impossible on any reasonable time scale.

What we do to accomodate this, is implementing a biasing procedure in the simulations. This is done by splitting the simulation procedure up in multiple steps, scanning trial events for the trigger condition, in this case either the full conversion ($\pi^0 \to \gamma, \gamma \to e^+e^-, e^+e^-$) or the partial conversion $\gamma \to e^+e^-$. If the trigger condition is fulfilled we save the data in a file and then we propagate the rest of the primaries[1] through the simulation and then write down the all of the data in a final merged file.

The way this biasing is implemented is via configure scripts telling the simulation which particles to transport, and then which particles to trigger on. So in reality we set up one simulation service, running proton proton collisions at 14 TeV and propagating the particles through the PIPE ITS TPC and TOF of ALICE. Then we check these trial events for the trigger condition and we use a second service to propagate the rest of the primary particles from the main collision through the detector, so we "finish" the triggered event one could say.

The simulation and biasing takes several arguments, where the most relevant ones (for this analysis) are the amount of trial events, triggered events and geant workers. The amount of trial events N_{trial} are the amount of events we simulate in each batch. Then one batch is simulated, and then all the events in that batch is checked for the trigger condition, if the event is triggered it is pushed on to the stack and the simulation is completed for the remaining primaries. The triggered events $N_{triggered}$ is the total amount of events fulfilling the trigger condition that we want the biased simulation to return. So we loop over the batches searching for triggered events as long as $N_{good} < N_{triggered}$, and then $N_{good} += 1$ everytime we find an event fulfilling the trigger condition in the batch. The amount of geant workers defines the multi threading, the more geant workers, the more cores you use for running the simulations, so this definitely affects the run time.

This biasing procedure has been tested in attempt of validating the implementation and making sure we actually transport and trigger the events as intended. This has done by analysing the triggered events before propagating the rest of the primaries, to see if we actually see the conversions as intended. So we check the amount of π^0, γ, e^+e^- and link those by their respective mother and daughter particles[2] to filter out the actual conversions.

2 Simulation Structure

I will just briefly run over the structure of the simulation framework and the biasing implementation to clarify what we actually compare, when we do the analysis of the run time gain and the

[1] Particles produced in the collision, and not in the surrounding material

[2] via their track IDs

validation of the biasing procedure.

2.1 The full O^2 simulation

As mentioned, the full simulation is split in three parts. The particle collision generator, the propagation of the generated particles through the detector and at last the merger of data from each thread.

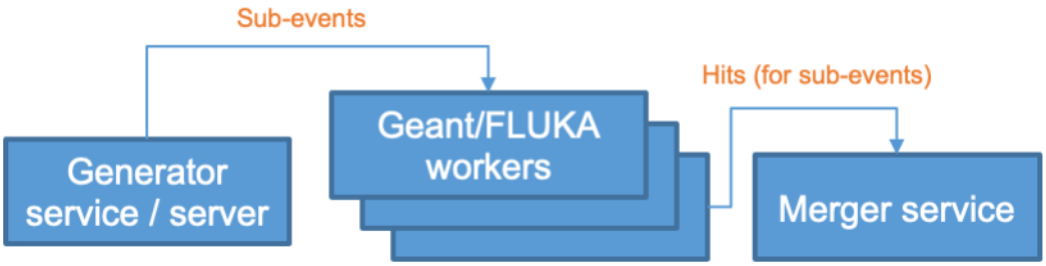


Figure 1: The structure of the full O^2 simulation framework[4].

For this analysis the generator used is Pythia8, where we run minimum bias proton-proton collisions at $\sqrt{s} = 14\,\text{TeV}$. The geant workers determine the amount of threads used for running the simulation and each thread contains the ALICE detector elements: PIPE ITS TPC and TOF. For this analysis three workers have generally been used. Additionally there is the merger service that merges the data from the respective hits and tracks in each of the geant threads and creates a final kinematics file: o2sim_Kine.root, and it is those kinematics we use for analysis of the full simulation and comparison with the biased simulation.

2.2 The biased simulation

The biased simulation implements some additional steps, and filter out the ”interesting” events at an early stage of the simulation.

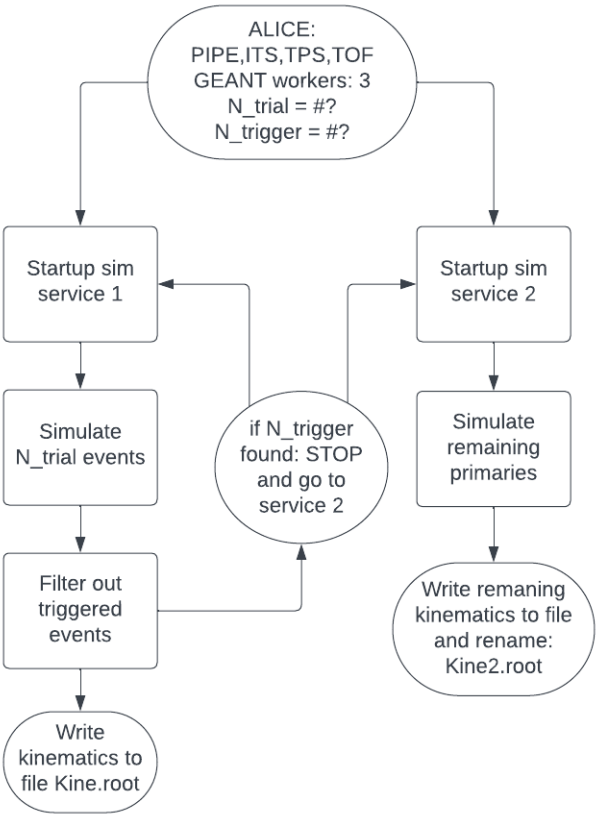


Figure 2: The structure of the biasing procedure.

The biasing, as is, takes several arguments, where the ones I have been adjusting is seen on Fig. 2. The biasing example sets up two simulation services with the same kinematic and geometry specs. In reality, the first service could be used again, instead of starting sim service 2, if it wasn't for the reconfiguration of engines/stacks currently being limited. As earlier explained, using sim service 1 we simulate a batch of events and then we check all the primaries of the events in that batch for the trigger condition, and if it is fulfilled we save the data. Then we fully simulate the remaining part of the triggered event using service 2. It is then the Kine.root file we use for validating the biasing procedure. This file is scanned to see if it contains the conversion and then the spacial and kinematic properties of the conversion and conversion particles is investigated.

3 Run time and benchmarking analysis

This analysis aims at determining how much time is gained running the biasing example, compared to the full simulation, when you want to extract a certain amount of triggered events. For this analysis we use the partial conversion $\gamma \to e^+e^-$ as trigger condition. We are running pp collisions at $\sqrt{s} = 14\,\text{TeV}$ and use three geant workers. Using the biasing example to obtain sufficient statistics, I computed $x_{ratio} = \langle \frac{N_{trigger}}{N_{trial}} \rangle$, which was then used to estimate the amount of full events one would need to get a certain number of triggered events and the time it would take to get a certain amount of triggered "good" events running the full simulation.

$$t_{full} = t_{mean} \frac{N_{good}}{x_{ratio}} \tag{1}$$

Here t_{mean} is the average time it takes to run one event, N_{good} is the amount of events containing

the conversion, meaning the amount we want to trigger on, and x_{ratio} is explained earlier.

Finding t_{mean} and x_{ratio} yielded what we see in Fig. 3

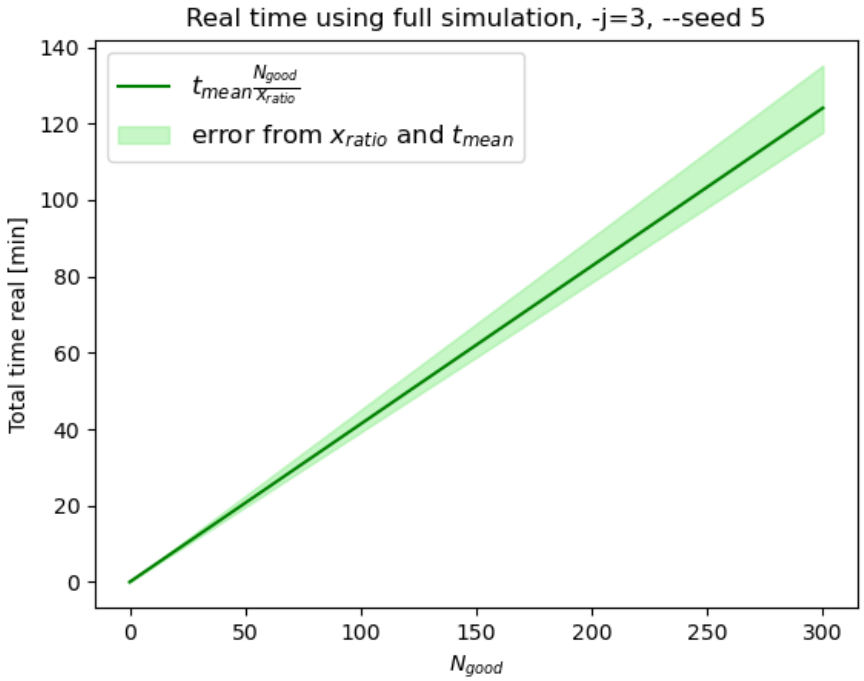


Figure 3: The time it would take to get N_{good} events containing the conversion running the full simulation without biasing.

The error on x_{ratio} is determined by sampling $\frac{N_{trigger}}{N_{trial}}$, which turned out unsurprisingly to be normally distributed, so x_{ratio} is the mean and the error is simply $\frac{\sigma}{N}$. Concerning the error on t_{mean} I have simply applied a conservative 5% uncertainty. The ”real” on the y-axis simply means it is the real time that passed running the simulations, -j3 is the amount of threads and –seed is the fixed random seed.

To compare, it was investigated how long it would take to get same amount of N_{good} using the biasing procedure. Here I simply timed the biased simulations for $N_{trigger} = N_{good} = [50, 100, 150, 200, 250, 300]$, and it yielded

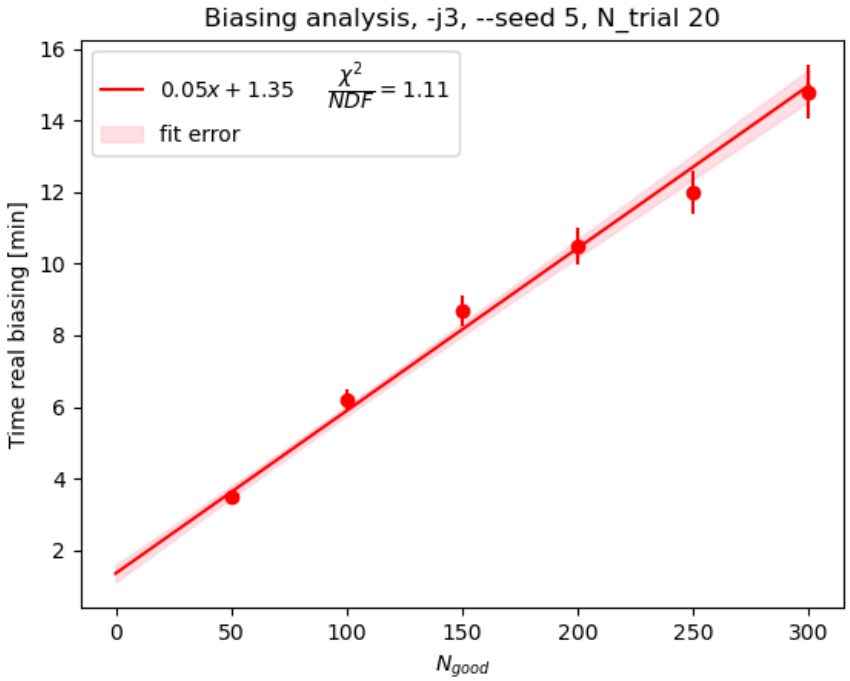


Figure 4: The time it would take to get N_{good} events using the biasing procedure.

The error bars on the points are once again a conservative 5% uncertainty on the run time. The linear dependence of the run time on the wanted amount of good events was fitted and the $\frac{\chi^2}{NDF}$ seemed reasonable. From this fit we get the initialization time, the time it takes the biasing procedure to start up, as $t_{ini} = 1.35 \pm 0.06$ min.

We can now compare the full simulation time to the biased simulations time. We do that by comparing the time it would take to generate N_{good} using the full simulation and using the biased simulations. What I got was the following:

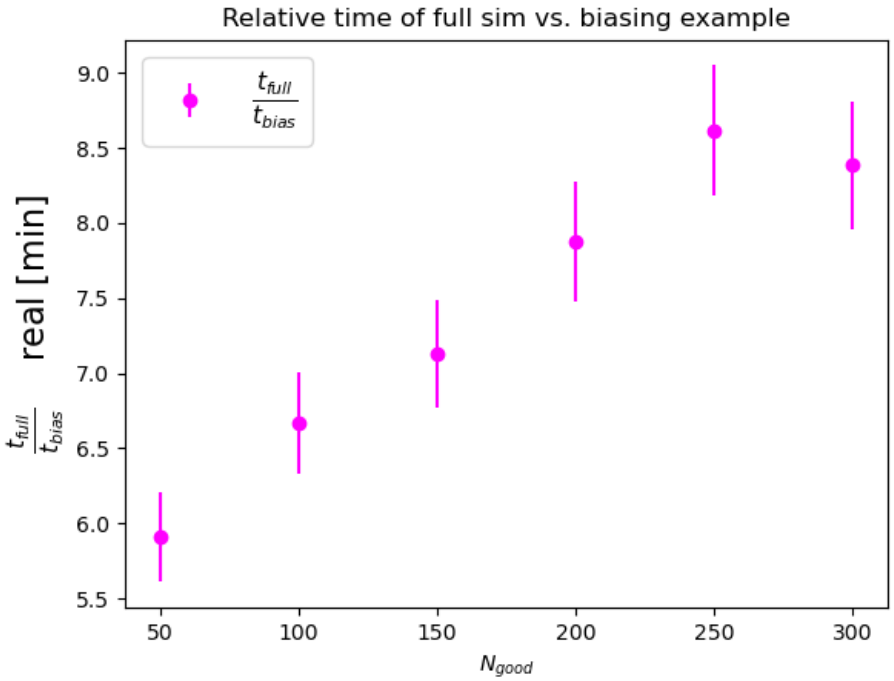


Figure 5: The time it would take to get N_{good} using the full simulations relative to using the biased simulations.

Here we divide the full time with the biased time and what we get is that it is 8.5 times faster to use the biasing procedure, when you want to generate 250-300 good events. We see a steep rise in this relation as a function of the amount of good events until $N_{good} = 250$ and then it might flatten, so either the behaviour changes around 250 or the point at $N_{good} = 300$ might be an outlier. But we do not have sufficient information to conclude what happens there. What can be concluded with certainty, is that the biasing example definitely runs faster.

4 Testing and Validating the biasing procedure

In this section I run through a certain amount of tests I made of the biasing procedure as is. The goal is to make sure we filter the right events, save the correct information and continue the simulation of the right events.

4.1 Amount of photon conversions

Firstly I checked the amount of various particles from each specie we got in the filtered file Kine.root, to see if these magnitudes made sense. Looping over each filtered events and extracting the amount of particles of each specie yielded the following histograms:

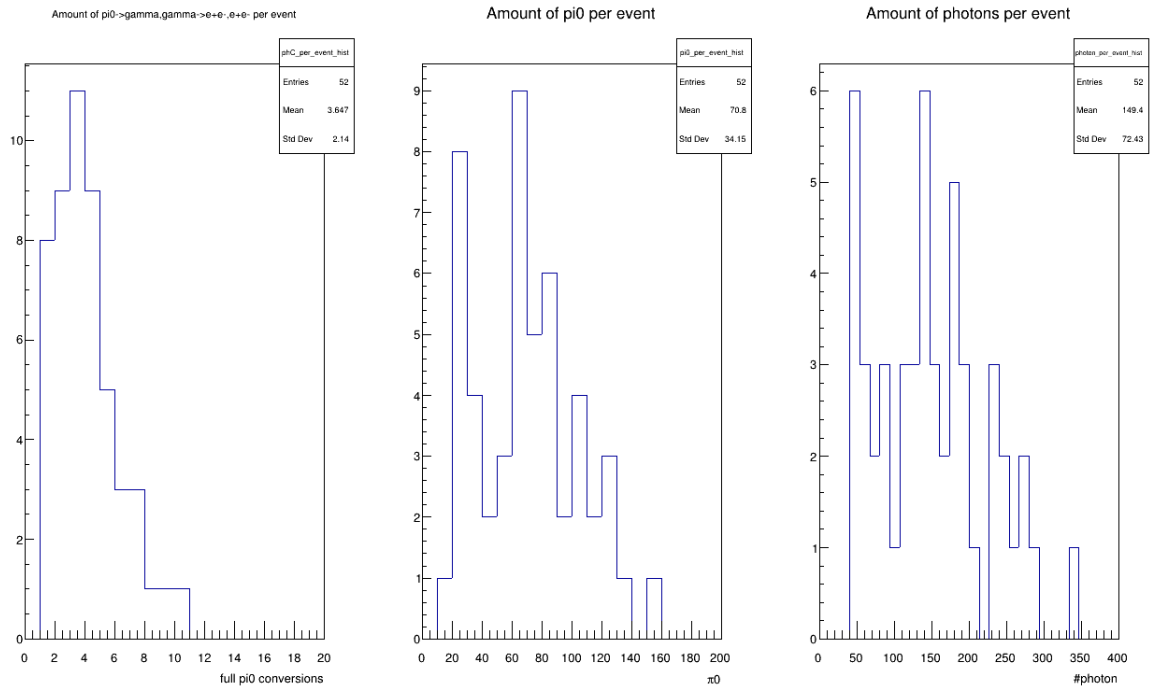


Figure 6: Average amount of particles from each specie in one triggered event before merging.

We determine the average amount of full photon conversions, meaning the amount of $(\pi^0 \to \gamma, \gamma \to e^+e^-, e^+e^-)$ processes, the average amount of π^0 particles and the average amount of γ in each triggered event/file. What is found is that we get 3.5 full photon conversions per. event on average, which is quite unlikely. What is also found is that we get 70 and 150 netrual pions and photons respectively, which compared to standard minimum bias full simulations, are also quite a large amount of particles. We actually get almost a factor of 2 more π^0 and γ in these biased files, than we do running the standard simulation. The back end Pythia8 setting will influence this, but a factor of 2 is too extensive to be "normal" setting fluctuations.

The p_{T} distribution of the photons was also briefly investigated, and this peaked at low p_{T} which is to be expected, when we get this extensive amount of photons. But it is still not a sufficient explanation for the biasing example returning more particles of each specie than a normal full minimum-bias simulation run.

4.2 Spacial behaviour of single photon conversion

Here I have singled out an event with only one full photon conversion and then I have used the spacial starting coordinates (z, η, ϕ) to map the conversion as a 3D plot. Then I have weighted the entries in the figure to distinguish the particle species.

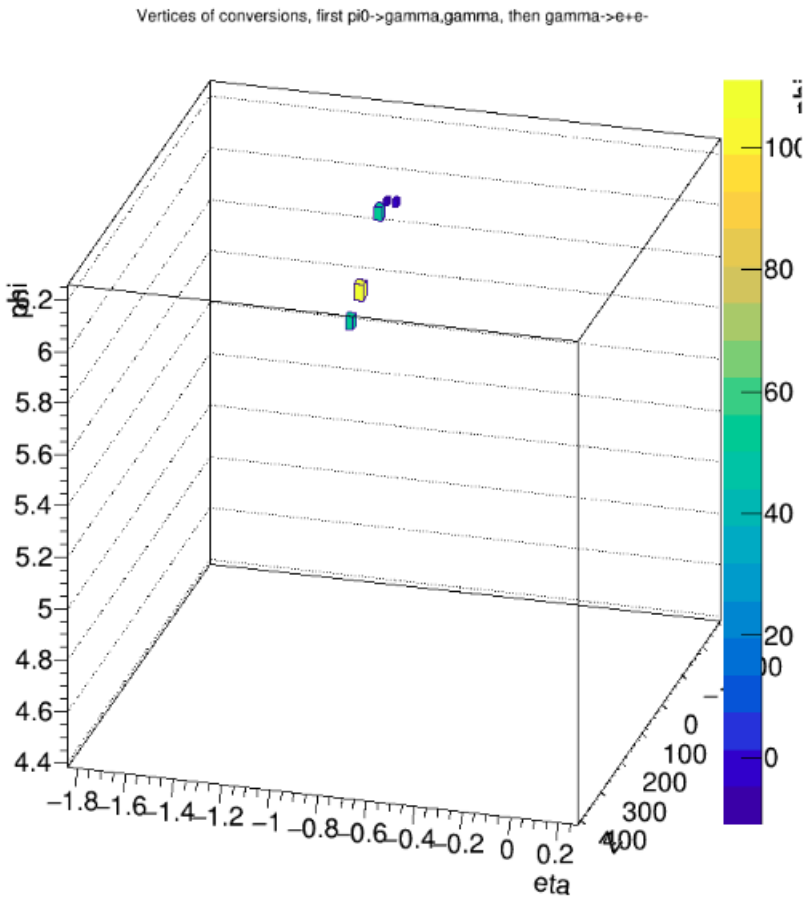


Figure 7: Spacial starting coordinates of particles in the single full conversion. Yellow = pi^0, light blue = γ, dark blue = e^+, e^-

For clarification you have the plotted coordinates of γ and e^+, e^- here:

γ_1 coordinates:
η -0.892214
ϕ 6.14305
z -0.0123035

e+e- coordinates originating from γ_1:
η for e+ and e- from γ_1 -0.900097 -0.876136
ϕ for e+ and e- from γ_1 6.13856 6.15212
z for e+ and e- from γ_1 -46.7379 -46.7379

γ_2 coordinates:
η -1.01643
ϕ 5.75132
z -0.0123035

e+e- coordinates originating from γ_2:
η for e+ and e- from γ_2 -1.01535 -1.01713
ϕ for e+ and e- from γ_2 5.75143 5.75125

z for e+ and e- from γ_2 -65.4841 -65.4841

And this makes fine sense, since the two γ's have the same z-coordinate, as they should. Additionally, the corresponding e^+, e^- pairs have the same (η, ϕ) as their γ mother particle, which they also should. We also see that the pair creation coordinates for the matching e^+, e^- are so close that they are not visually distinguishable. Conclusively, this is the spacial behaviour we would expect from a full photon conversion.

5 Conclusion

The overall conclusion is that the biasing procedure, as is, saves times in the process of computing certain "good" events, where "good" is defined by the trigger condition. The linear fit yielded an initialization time of $t_{ini} = 1.35 \pm 0.06$ min. We still have some rather large uncertainties on the computation of the full simulation time in Fig. 3 due to the limited statistics on x_{ratio}. This should be improved. Additionally, the conservative 5% uncertainty on the time, is just a rough estimate as of now. Concerning the validation of the trigger condition, we get unusually large amounts of particle species in the first filtered data file Kine.root. Comparing this to a full minimum-bias simulation is worrying, and it should be further investigated, why we get these large particle samples, and also why we get this unlikely large amount of conversions per event. At last, the spacial analysis of the single full conversion from the biasing example seem to behave as expected.

References

[1] O2: https://aliceo2group.github.io/docs/

[2] Pythia8: https://pythia.org/

[3] Geant4: https://geant4.web.cern.ch/

[4] Running Run3 simulations, Sandro Wenzel, 26.4.2023, ALICE Analysis tutorial, https://indico.cern.ch/event/1267433/contributions/5359482/attachments/2635575/4560367/ALICE-Run3-MC-HowTo.pdf