Masters Project
May 22, 2022

# Opening the Door for the Use of Herwig in LHCb: Report 2

Zachary Harper
10241694
Project work completed in collaboration with
Oliver Hay
10299033

*University of Manchester*

**Abstract**

This project has been a successful attempt to integrate the Herwig7 generator into the Gauss Simulation Application used by LHCb to produce simulated LHC data. This second half of the project has completed the goal of the last report to integrate Herwig7 into Gauss, and has extended this to produce successful tunes of the program that should provide a basis to bring the operation of Herwig7 in line with the goals of the LHCb group.

## 1 Introduction

This report follows on from the work completed in between September and December 2021, and the subsequent report [1] on this. To summarise the important points from this previous work, this project aims to integrate the Herwig7 Monte Carlo Event Generator (MCEG) into the Gauss System Framework used by LHCb for LHC event simulation. This allows for parameters used in these simulations to be changed, and then subsequently compared to real world data in order to best estimate the true values of these parameters. Event Simulations are also capable of providing more data, quicker, and tuning their simulations to fit specific, desired, criteria. The work from the last semester concluded the Herwig7 MCEG can be initiated using external code, allowing for it to be operated by a third party program [1]. Through a few adaptations, the Herwig7 MCEG has been adapted for Gauss, and this Simulation framework can be built including Herwig7 [1]. The aims of the second half of this project have been to complete the final steps of this integration, allowing for Gauss to fully operate Herwig7, and then move onto tuning the parameters used by Herwig7.

## 2 Refining the Code

### 2.1 The End of Gauss Integration

The project work this semester began with finalising the interface between the Herwig7 MCEG and Gauss. As mentioned in the previous report, an issue involving the '.run' files used by Herwig made this interfacing difficult [1]. These '.run' type files included a name that was unique for each specific instance of the Herwig7 generator created [1]. By moving this naming line from the '.run' files to the Herwig7 Production file, used by Gauss to initialise the generator, the instance of the Herwig7 generator created by Gauss could be reliably named and therefore reliably called.

However, this was not the only step left to complete. Another issue was the set of Herwig7 files used by Gauss. As Gauss is a program most commonly used in central productions [1,2], it needs to be able to access Herwig7 files that are accessible by the users of these central productions. This means using files installed on the CVMFS files system which is accessible to all of the CERN userbase [1]. Thankfully there is an installation of Herwig7 already on this file system. By passing Gauss the location of this installation via an environment variable, this code can be passed into ThePEG to set up the Herwig7 generator.

Further Environment variables also need to be set, specifying the location of the Herwig Library to be used by Gauss Job files, and to locate parton density function (PDF) sets for use by Herwig.

Additionally, the inclusion of these libraries in the CMakeLists file used by the Gauss Herwig Library was also required in order to successfully build a Gauss instance.

In order to aid in the completion of this project a GitLab repository was created. This greatly aided in the time spent working on the code base, as new branches for various modifications could be established. By creating a git instance of the Herwig library, updates to the code could be quickly pulled onto the lxplus system, and varying branches switched between. All that was required after this is a make purge/make command to rebuild Gauss using the new Herwig Library.

## 2.2 Implementing Herwig7's Caching

As discussed in the previous report [1], the structure of Herwig breaks the internal processes down in a few subroutines before the final output is produced. Some of these can take significantly more time than others, and this section was an attempt to speed up that process.

The subroutine, or function, of primary interest here is the initialisation. This function is responsible for building an instance of the Herwig7 generator that will then be used to generate events. One step of this process is calculating a sizeable series of variables for use by the event generator. These variables and their values depend on the settings Herwig is operating on, and so it often makes sense for these to be calculated on run. However, it is likely that the most common use of the Herwig7 event generator, once it is integrated with Gauss, will be using the default settings for this integration. Therefore, it makes sense to attempt to reduce the initialisation time for the generator by removing this calculation step.

Herwig7 has inbuilt functions to allow a previously used set of variables and their values to be loaded into the program and event generator. These variables are stored in a file named "DEFAULT.run", and initialising Herwig with this file should allow for the event generator to be initialised more quickly, as it does not have to calculate values to populate this file with.

From this, significant time cut-downs were expected, and in order to establish the quantity of this reduction, the processing time for event generation using the Herwig7 MCEG with Gauss were recorded. These were then graphed over the operation of several different orders of magnitudes of events generated, and compared to similar data for Herwig7 operation without using caching functions, and then Pythia8 as a frame of reference. The results of this can be seen in section 4.1

## 2.3 Issues with the Collision Angle

An additional major code alteration made during this half of the project was the implementation of the "Boost For Epos" function. This piece of code is currently implemented in Gauss [3] for utility purposed, was was also implemented into the Herwig7 production file after it was deemed necessary.

The "Boost For Epos" function is used to alter the frame of reference for HepMC format particles. This was required due to the fact the Herwig and Gauss have their default operations active in difference frames of reference [3, 4]. Therefore, in order for the data to be successfully passed from Herwig in Gauss' Generation phase to Gauss' Simulation phase, these particles must be boosted between the two frames of reference.

This is the purpose of "Boost for Epos". Minor re-writes of sections of the function had to be made in order for the function to correctly interface with the rest of the Herwig7 production file. Fundamentally, the function calculates the boost to the centre of mass frame, then loops over the particles and sets their new energy and momenta (hence

Epos). This function is not an ideal implementation of this operation, as it completely overwrites the original values. However, given the time constraints of the project and the prototypical nature of it, this implementation is adequate for now.

When this function was implemented, it was also believed it was necessary for improved analysis of Herwig7 events. This was being completed using Rivet, software with a more in-depth discussion in section 3.2.1 later in this report. The data being passed to Rivet would be more in line with the usual Gauss data if it was in Gauss' usual reference frame. However, after a review of Rivet after the implementation of this function, it was determined that Rivet is capable of performing these boosts on its own, and therefore this function is only required for passing data to Gauss' simulation phase.

# 3 Refining the Physics

## 3.1 The Physics of the Herwig7 Generator

The second half of the project work undertaken this semester was looking to tune the Herwig7 MCEG to fit better with the LHCb group. A demonstration of this has been successfully achieved, which can be used as a basis to move further forward.

The tuning of an MCEG is an important process which alters the model used by the MCEG in order to bring its output data closer in line to real-world data [5–7]. This is done by altering the values of variables known as 'parameters' [7], and comparing these output data against real world samples that the user of the MCEG is interested in [7].

This tuning is important for the Herwig7 Generator, as it is being integrated into the Gauss System used by LHCb. This means that the Herwig instance used by Gauss should be producing data that is in line with the physics LHCb studies.

This project has achieved some preliminary tunes of the Herwig7 MCEG, Unfortunately, due to time constraints the Herwig7 output has not been compared to real-world data, however a comparison to the Pythia8 generator, which is used by Gauss and here is tuned to LHCb [3], has been made. The information acquired by these tunings should provide a useful starting point to bring Herwig7 in line with the aims of LHCb.

Three parameters have been selected for altering in these preliminary tunes. These are the Inverse Hadron Radius $\mu^2$ and two contributors to $p_\perp^{min}$, $p_{\perp,0}^{min}$ and $b$. The relationship between the two is

$$p_\perp^{min} = p_{\perp,0}^{min}(\frac{\sqrt{s}}{E_0})^b \tag{1}$$

[8].

where $E_0$ is the reference energy scale, and $\sqrt{s}$ is the centre-of-mass energy of the hadron-hadron collision [8,9]. This $p_\perp^{min}$ is the "Transition scale between soft and semi-hard interactions" [8]. The quantity of Multi-Parton Interations (MPI) depends on the value of this variable, as it is essentially the cutoff of transverse momentum ($p_\perp$) required for a multi-parton interaction.

The Inverse Hadron Radius is a parameter that defines the likelyhood of interaction between two particles [5,9], much like a cross-section, although it is inversely proportional.

The parameters used for the tuning demonstration have been selected using a CMS article [9] on tuning the Pythai6, 8 and Herwig++ Event generators. As discussed in the previous report [1], Herwig7 and Herwig++ are functionally identical. While there is little overlap between CMS and LHCb due to the difference in pseudorapidity (angle relative to incoming particle beam) these detectors cover [10], the tuning of similar parameters between Herwig++/Herwig7 and Pythia8 is very useful, as Pythia8 is already tuned for LHCb and provides a good comparison for Herwig7 tuning. Additionally, as these are preliminary tunes, an understanding of how these parameters affect the Herwig7 generator is the main aim.

## 3.2 Tuning Herwig7

Tuning an MCEG is quite an involved process, requiring a large range of datasets with a range of parameter values. Optimising one parameter can be completed with relative ease, simply by minimising a $\chi_2$ . However, every additional parameter for fitting increases the dimensionality of this solution, requiring more datasets and more processing time and/or power.

In order to complete this process, the use of many different pieces of software was required.

### 3.2.1 Rivet and YODA

Rivet (Robust Independent Validation of Experiment and Theory) is CERN-affiliated particle collider data analysis software [11]. It is used to analyse data from MCEGs. It is a general framework for analysis, with specific analyses being completed via 'plugins' [11]. These plugins are known as "Rivet Analyses" [11]. They instruct Rivet on operations to perform to produce a desired output [11]. Common features in these analyses are cuts, and the types of histograms to produce binnings for [11, 12]. In this project, Rivet has been used to produce detailed graphs for analysis.

YODA (Yet more Objects for Data Analysis) is a small set of classes for data analysis histogramming [13]. In this project is it primarily used for storing output data from Herwig for later use by Rivet, but a lot of the software used here includes the module.

Gauss runs on 'job' files, as discussed in the previous report [1]. These job files can be written to run a Rivet Analysis on the data collected from the job. These Analyses output YODA files containing data for creating histograms, processed as according to the Rivet Analysis [11]. These YODA files can then be passed back into the Rivet program, and will produce the final histograms [11].

For this project the Rivet Analysis MC_FSPARTICLES [12] was used. This is a very basic Rivet Analysis, with no specialised cuts or histrograms, but it is sufficiant for the initial purposes of the project. This has produced the histograms seen in section 4.2

### 3.2.2 Professor

The Professor program is a framework built on the Minuit Minimisation software [7, 14] specifically for the purpose of tuning event generators. It performs two important functions, laying out parameter samples to be collected, and then processing this data to find minima between the parameters [7].

The first function is reasonably simple, Professor can be provided with a text file consisting of parameter names and their appropriate ranges. When the appropriate command is executed, Professor will create a sample space with a dimensionality equal to the number of specified parameters [15], and then return a set of points within this space. This set of points should be used by the MCEG being tuned to collect a set of data generated at each point.

Once this data has been collected, it can then be passed back into Professor, along with a set of comparison data, representing the ideal tune for the MCEG [7]. Usually this comparison data is real data reflecting the physics the MCEG is being tuned to produce. Professor will create a parameterised response function for the parameter space data [7]. This function is essentially a self-determined fit function, based on the data set Professor has been given [7]. This fit function ideally represents the physics that the MCEG is attempting to simulate. Professor then minimises the $\chi^2$ of this function against the parameter space and comparison data using a goodness of fit function [7]. Professor then finds the minimum of this goodness of fit function [7]. This minimum then corresponds to the set of parameter values that best matches the ideal physics of the MCEG according to Professor's computations. Whether this is true or not is then left up to the user of the program and further analysis.

The Professor program has other functions beyond these two most fundamental ones however. It is capable of changing the weighting and therefore importance of numerous physics variables in this fit [7]. This allows the tuning of the MCEG to be directed towards certain specific physics, which is an important quality of MCEGs, as discussed in the previous report [1]. The Professor program also evaluates the error on its outcome with a covariance matrix, and is capable of calculating the correlation between all of the parameters used in the fits [7, 15]. The Professor program can also be passed a template file alongside the specifications for parameters, and will fill out this template with different parameter value sets in the same way it selects points in the parameter space [15]. This was made use of here, to generate various '.in' files, a type of options file used by Herwig, as discussed in the previous report. This '.in' file can be used to set the values of specified parameters, and therefore this function of professor can be used to generate the complete set of '.in' files required to run Herwig with across the sample points in the specified parameter space.

#### 3.2.3 LHCb Grid Processing

The Professor program requires a lot of data in order to complete a useful tune. Not only does this mean running jobs of a large magnitude of events, different Gauss jobs need to be run in order to collect data at different parameter sets. The default number of sample points in the parameter space from professor is 100 [15]. Collecting 100 separate datasets from Gauss by running the program manually would take a prohibitively large amount of time, at least for the scope of this project.

This led to the use of the LHCb computer Grid for processing large amounts of Gauss jobs. The interface for the grid is Ganga, a computational task-management tool based on Python [16]. Ganga operates similarly to a python shell, allowing for files to be loaded, and then sent to Dirac, the software that manages the grid resources [17]. The Grid is an international network of LHCb affiliated computers, organised via Dirac to ease the generation of large about of MCEG data [17].

For this project, a few short python files were composed to submit the range of jobs required for Professor to Gauss. After Professor was used to create the range of '.in' files for use in Herwig, these could be cycled through by the job submission file in order to submit 100 jobs, each with a difference set of parameter values. These jobs collected the Herwig7 default of 10,000 events, but to save computation time, the Grid allows for these to be split up into subjobs. Running 100 subjobs at 100 events each allows for the Dirac system to take advantage of the large amount of computers accessible to it, and run these jobs in parallel to significantly cut down on computation time. Each of these subjobs generates a YODA file containing the histogram data from the RIVET analyses, which can then be saved from the Dirac system to Local files, and merged. This data is what is processed by Professor to tune Herwig.

## 4 Results

### 4.1 Caching Implementation Results

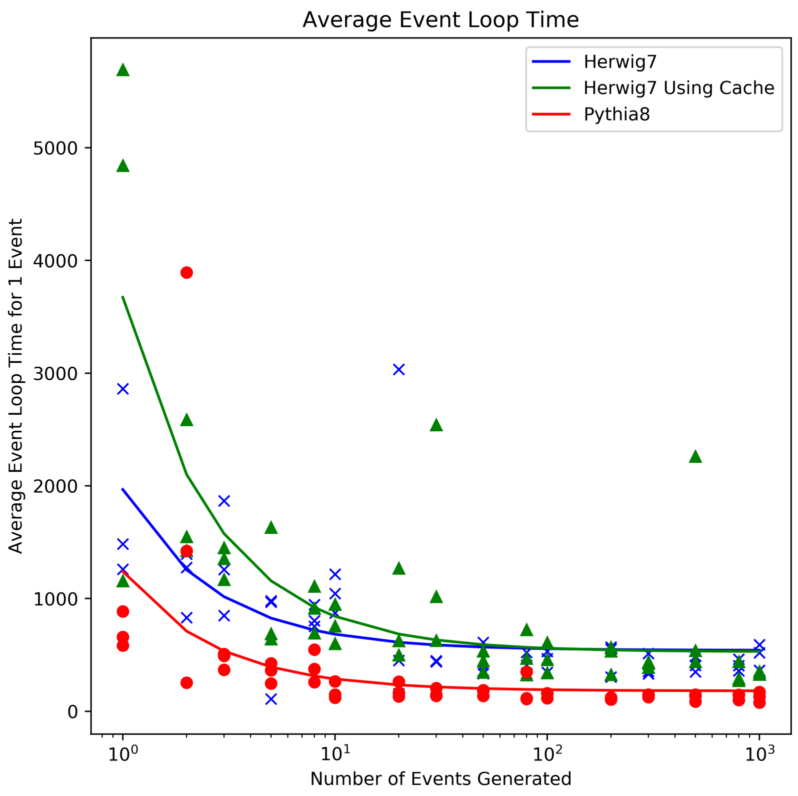

Figure 1: This graph displays the average amount of time spent generating an event, depending on the total amount of events generated by the job. The fits are $a/x + b$, with -errors calculated from covariance matrices at 10-20across the difference generator fits. The most important characteristic is the large increase in time at lower amounts of events. Note the logarithmic x-axis.

The Event loop of Herwig7 covers all parts of the process from the generator being initialised by Gauss through to the events being generated and stored. The average time here is important for jobs generating more than one event, as this means the average can be raised by a specific event taking a long time to complete.

The first immediately noticeable feature of figure 1 is that the fit line for the version of Herwig7 loading in variables from a file is actually above the line for the base Herwig7 generator, which calculates these variables on every initialisation. This is a surprising result, and seems to indicate that it takes longer to load in and read the file than it does to simple calculate the variables. However, these fits appear to converge at higher numbers of events generated, perhaps prompting more investigation at higher quantities of events.

The second important feature is the high variance across the graph. The cached variance of Herwig7 appears to have the highest variance, but even the well-performing Pythia8 still has quite a high variance. There are several factors that contribute to this variance. The first is the nature of MCEGs. Sometimes events will occur that will simply be more complex to simulate than others. This is likely the driving factor behind the variance at the lower end of the graph, which also seems to be the highest, as there is no chance for the event time to average out between long and short computing time events. Then there is the nature of the Grid used to collect the sample. Different computers across the Grid have difference processing powers. Whilst this should be able to be standardised with clock speed, this does not always seem to be the case. Finally, the sample size will also contribute to this. Each event generator has 3 events per quantity of events generated. A higher number should reduce this variance. Additionally, in a plot using more data, with more focus on producing reliable information than demonstrating behaviour, some of the more extreme points can be cut for anomalous behaviour.

Thirdly, Pythia8 is clearly faster and less varied than the other two versions of Herwig7. This should come as no surprise, as Pythia8 has seen significantly more modern use, and is already adapted Gauss and LHCb. Subsequently, it has much better optimisation than Herwig7.

In figure 2 the Average generation time refers to the amount of time during the event loop that is spend purely on generating the event, not including generator initialisation or any other processes. The results from this graph appear rather similar to those of figure 1, as to be expected due them being the whole of and the main contributing factor to the event loop time. However there are a few significant and important differences.

Firstly, the fits are much poorer, with much higher percentage errors. This is somewhat to be expected - a different graph may well have a different curve that best fits it. However, in the interests of comparison to figure 1 another inverse fit has been chosen. And as can be clearly seen by comparing the two plots, there is a significant difference in the coefficient $a$. It is an order of magnitude smaller in figure 2 than in figure 1. This makes it clear that the initialisation must be a large factor for lower values in 1. Due to the averaging process, the contribution of the initialisation to the amount of time it takes to run the MCEG starts to become insignificant, leading to the convergence in the times of the two versions of Herwig7.This, along with the reasonably constant separation of the two versions of Herwig7 in 2 implies that the initialisation causes the majority of the difference between the versions in 1.

Significantly, the cached version of Herwig7 still takes slightly longer on average per generation phase than the un-cached version, however this appears to mostly be due to its higher variance. However, as figure 2 should not take the initialisation into account, which is where the main difference between the two Herwig7 versions occurs, this behaviour is intriguing and may require more investigation.

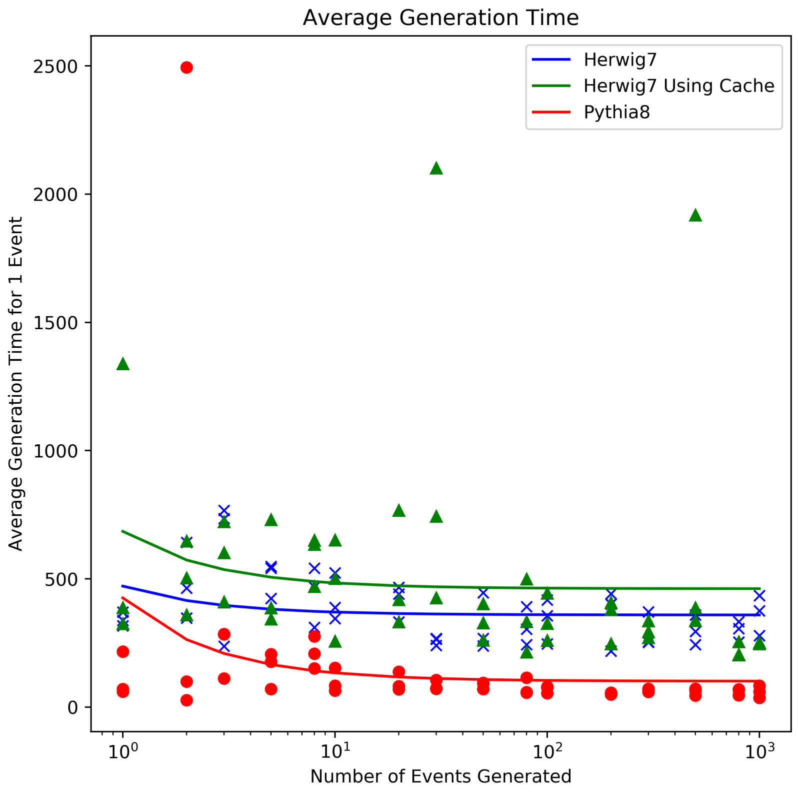

Figure 2: This graph displays the average amount of time spent purely on the generation section of an event generation, depending on total amount of events. They are still fit with $a/x + b$, with -errors calculated from covariance matrices at 50-100. This is very high, mostly due to the obviously high variance. The fit has been primarily chosen to ease comparison to figure 1. Importantly, the rise at lower event numbers is much less significant. Note again the logarithmic x-axis.

Once again, Pythia8 is the consistently lowest on processing time, as to be expected.

The final graph, figure 3, on the Total Time to complete a full event generation job is the simplest of the three. A clear and low error linear fit confirms a linear increase in time taken, as would be expected. This means there a no hidden factors in Herwig7 causing issues that might result in non-linear growth with event quantity for generation. The sub 1 gradients also indicate that running jobs with higher numbers of events

The fit lines for the two versions of Herwig are interesting, although due to the only minor differences in gradient, interpretations should be further investigated. These fits agree that the non-caching version of Herwig7 is more effective than the cached version at low numbers of events, and that variance is highest here, as expected. However, at high numbers of events, it appears that the cached version begins to pull ahead on event loop speed. To confirm this, more data at higher numbers of events generated should be taken, ideally along with more repeats to confirm the results at lower event numbers.

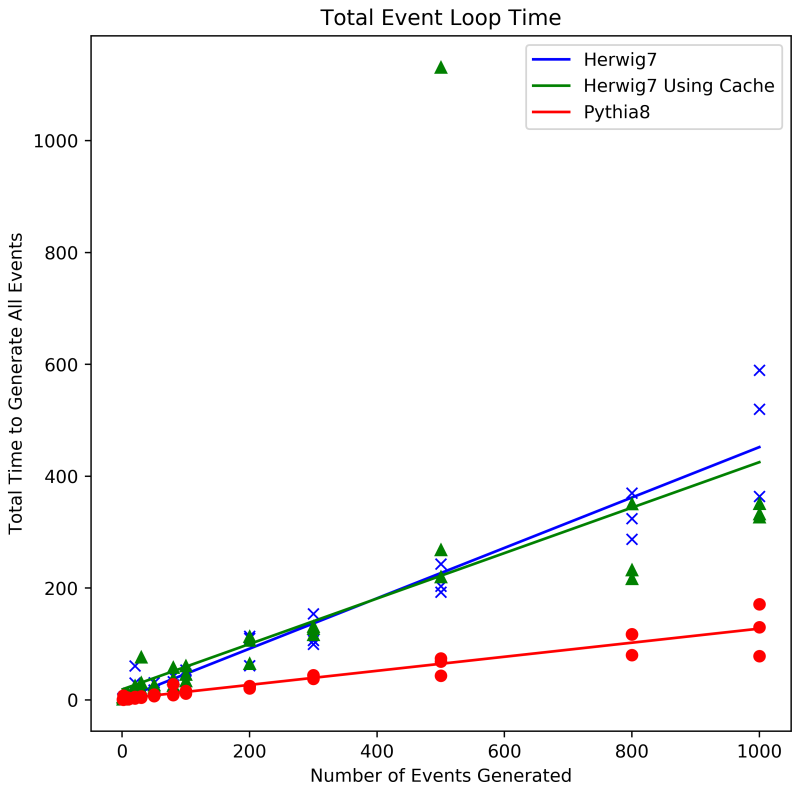

Figure 3: Final Timing Graph showing a clear linear relationship between the total number of events generated and total time taken. The linear $mx + c$ fits have -errors on the order of 1.

## 4.2 Tuning Results

### 4.2.1 First Tune

The first tune was largely for demonstrative purposes, and to provide a baseline for subsequent tuning.

It successfully showed the ability of Professor to tune Herwig7, with the tune's completion proving the methods used to achieve it were useful to follow for subsequent tunes.

Probably the most interesting feature of the tune was the Inverse Hadron Radius failing to reach a minimum, as shown in figure 4. This means the true minimum value is likely below the lowest value sampled in the parameter space. As the limits for the inverse hadron radius were taken from a CMS paper [9], this is a significant result, implying that LHCb data may require a significantly different value.

An inspection of the correlation matrices shows a 23 correlation between $p_{\perp,0}^{min}$ and the power $b$, somewhat high and to be expected since these parameters are part of equation 1. The Inverse Hadron Radius has a correlation of $-0.11$ to $p_{\perp,0}^{min}$ and $-0.02$ to the power $b$. In this tune it is not clear whether this is because the parameters are truly unrelated, or because the Inverse Hadron Radius did not reach a true minimum.

A goodness of fit of 1983 was obtained, which appears a very high result.

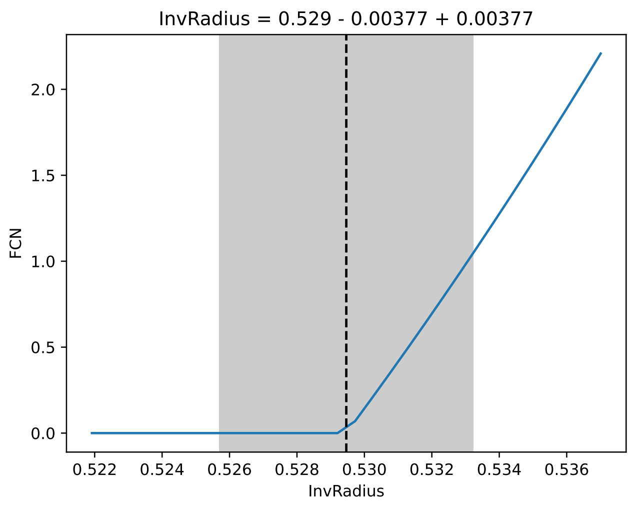

Figure 4: The Minimisation of the FCN parameter for the Inverse Radius. Dashed line shows determined minimum. The discrete change in gradient here follows from the discrete set of parameter sample points. No real minimum has been determined here, the abrupt turning point is simply from the parameter hitting the bottom of the parameter range.

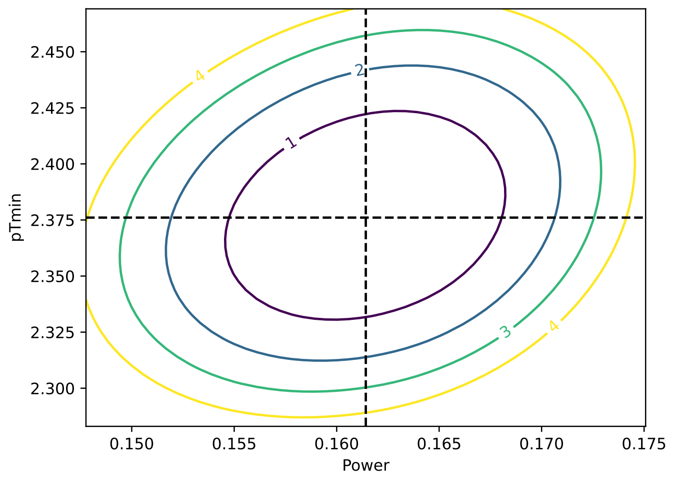

Figure 5: The more successful of the tuned parameters. The crossing point of the dashed line shows the minima of the FCN $\chi^2$ fit to both parameters. The coloured bands show the varying $\sigma$ confidence levels of the $\chi^2$ fit. Additionally, the diagonal slant from bottom left to top right indicates a positive correlation between the two parameters. $p_{\perp,0}^{min}$ is fit to 2.3760, and the power $b$ is fit to 0.1614

To aid in the brevity of this report, the vast majority of Rivet Variable Analysis will be discussed with regards to the final and most successful tune, but these were completed and reviewed for each tune, providing information that was considered for subsequent tunes.

### 4.2.2 Second Tune

The second tune was completed with the lower bound of the the inverse hadron radius to 0. Additionally the sample space was increased to 300 points, and the Power $b$ lower bound was also decreased to 0.01. This tune successfully minimised all three variables, as can be seen in figures 6 and **??**. For this tune, a goodness of fit of 2086, much the same as the previous tune was obtained.

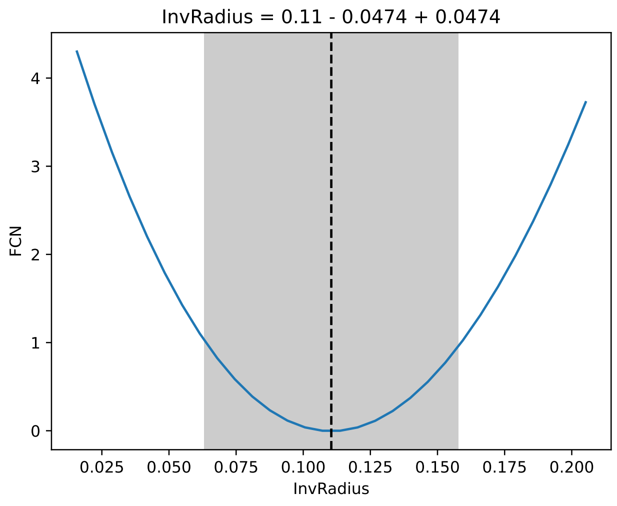

Figure 6: A successful Minimum for the Inverse Hadron Radius, at the dashed line with a value of 0.1104

Both the first and second tunes agree that the Inverse Hadron Radius has a small value. As this is analagous to an inverse cross section - a lower value referring to a higher likelihood of collisions - it is possible that Professor is attempting to maximise collisions. This may be simply to fit with the Pythia8 data, or it is possible that more data simply provides a stronger, and therefore more collisions will simply fit better. To an extent at least, this may be true, as the minimum is low but not zero when actually achieved.

The flipping of correlation in figure **??** from 5 is an unusual and unexpected occurrence. An explanation for this may be that since the inverse hadron radius has now achieved a minimum, this minimum also agrees with a minimum in the result of 1, $p_{\perp}^{min}$, or at least a fixed value. If $p_{\perp}^{min}$ is a fixed value, this would possibly explain the now negative correlation between to contributors to it, increase one paramter must decrease the other to maintain a constant result.

A final results of importance from this second tune is a graph from Rivet. This is the "Eta Sum Et" variable, or Pseudorapidity distribution of $\sum E_{\perp}$. This variable should cover the total energy of particles in directions orthogonal to the beam angle. As can be seen, it is the same value across all pseudorapidities. The graph suggests all pseudorapidities have the same total transverse energy, but provides no information of this distribution among particles at these pseudorapidities. Qualitatively this makes some intuitive sense when considering conservation laws, however a quantitative analysis of this is something that should be looked into and will likely require some investigation into the physics inside Herwig. - unfortunately due to time constraints, not something possible for this report. This data clearly has little importance to the tune, being constant between

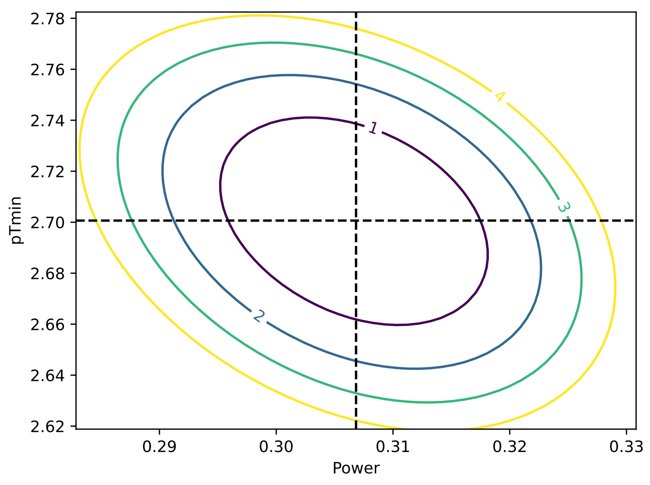

Figure 7: Another graphing of the fitting and correlation of both the Power $b$ and $p_{\perp,0}^{min}$. Interestingly enough the correlation is negative, from the now $y = -x$ diagonal slant, a change from the first tune. $p_{\perp,0}^{min}$ is fit to 2.7006, and power $b$ to 0.3068

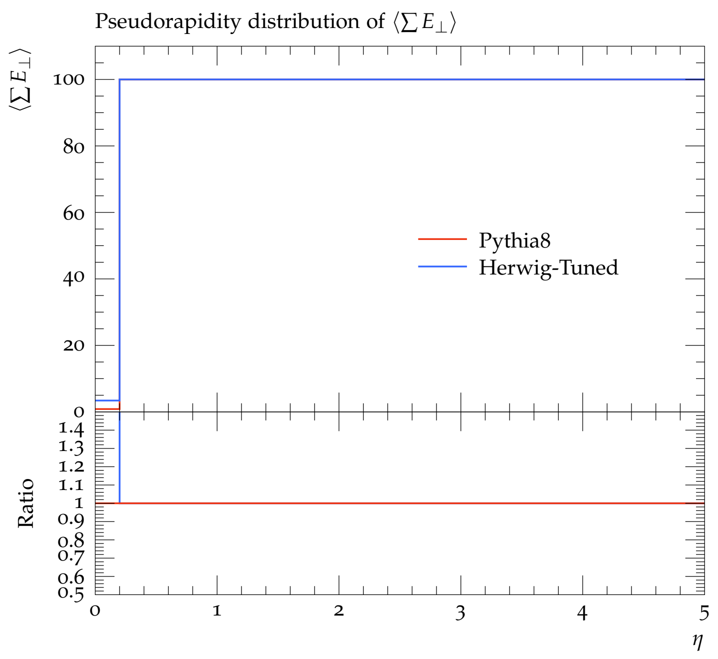

Figure 8: The "Eta Sum Et" ($\eta \sum E_\perp$) Rivet generated histogram. Clearly the data is not something which should be contributing significantly to the tune, being the same as the Pythia8 Data, and a consistent value across the pseudorapidity.

both generators and across all pseudorapidity values, but despite this, it has a mean contribution to the tune of 217, two orders of magnitude higher than the next highest contribution. Subsequently, the weighting of this variable in Professor's consideration for tuning was reduced to zero, acquiring the Final Tune of the project.

### 4.2.3 Final Tune

The final tune of the Herwig7 MCEG completed in this project was quite successful. From previous tunes, it was seen that the "Eta Sum Et" variable was contributing with a high significance to the tunes, despite the data being quite irrelevant to the tune. This variable had its weighting removed from the tune, producing significantly different results from the previous two. All three parameters tuned successfully. The goodness of fit was the best of the three tunes, at 1117, just over half the previous tunes.

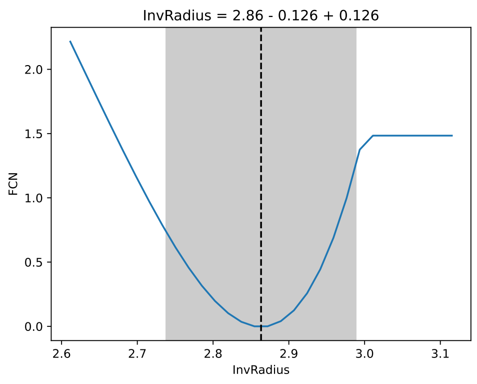

Figure 9: A clear minimum at 2.8633 the dashed line for the Inverse Hadron Radius. The plateau occurs outside sample space. Interestingly, the removal of the "Eta Sum ET" variable has caused the minimum to move to the opposite end of the range.

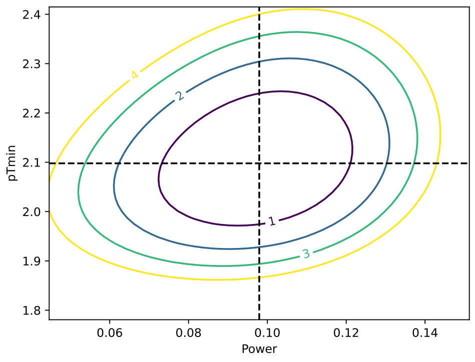

Figure 10: The Power $b$ and $p_{\perp,0}^{min}$ correlation once again becomes negative after the "Eta Sum Et" variable is prevented from contributing to the tune. It is not quite as round as the graph obtained in the first Tune, but its minimum is in a similar enough position at 2.0979 for $p_{\perp,0}^{min}$ and 0.0979 for the power $b$. The grey shaded area displays a $1\sigma$ confidence region.

It is shown in figures 9 and 10 that a minumum was reached for all three parameters. It appears that the Power $b$ and $p_{\perp,0}^{min}$ have returned to similar values to the firtst tune, 0.098 and 2.098. They have also regained their positive correlation, now at 38. This is somewhat more expected than the negative correlation obtained in the second tune, due to the nature of equation 1. As $p_{\perp,0}^{min}$ is the value of $p_{\perp}^{min}$ when $\sqrt{s} = E_0$, an increase in $b$ should be responsible for the scaling of $p_{\perp,0}^{min}$ and $\sqrt{s}$ to reach $p_{\perp}^{min}$. Thus an increase or decrease in $p_{\perp}^{min}$ would be expected to follow increases and decreases respectively in both $p_{\perp,0}^{min}$ and $b$. Additionally the minimum of the inverse hadron radius has a much higher value of 2.863. However, the confidence region does reach the top of the range of the parameter space samples. For this reason it may be prudent to extent the range above 3.0 in further samples, but as a minimum has been reached, this is not as necessary as it was for the first tune. This higher vlaue of the inverse hadron radius implies that Professor is no longer attempting to maximise collisions in order to fit the data most accurately, and that the cause of this behaviour was the "Eta Sum Et" variable. It is possible that since this variable was perfectly aligned with the Pythia8 data, professor was minimising the errors on this particular fit by increasing the amount of collisions and therefore particles.

The multiplicity in figure 11 represents the number of particles produced from collisions. Figure 11 contain the largest difference between the Herwig7 and Pythia8 data. Herwig7

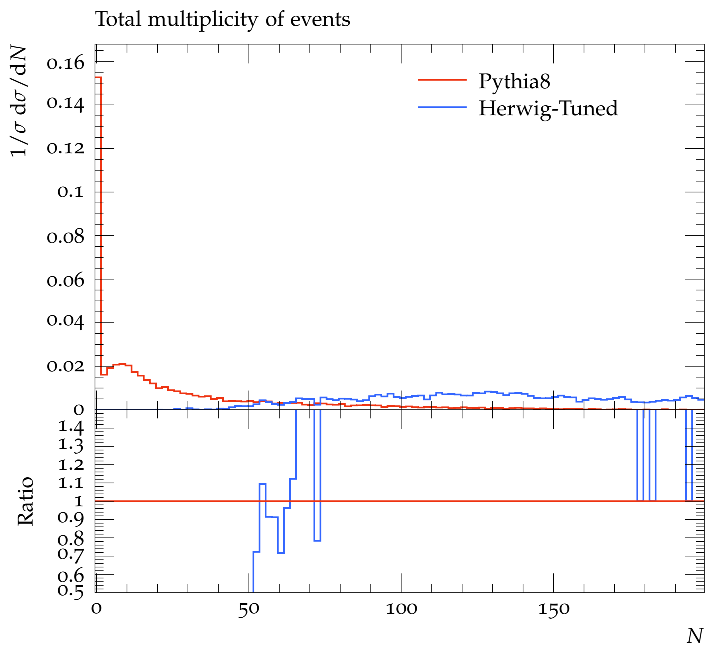

Figure 11: The multiplicity shows the biggest difference between Pythia8 and the tuned Herwig7, although this is likely due to Pythia8 not being specified an interation, and a difference in cuts.

here produces by far a greater range and quantity of multiplicities of events than Pythia8. There are two obvious conclusions from this, the first being that Herwig7 and and Pythia8 use significantly different physics models with regards to quantities of partilce produced by collisions, and the second possibility is that Herwig7 is using higher energy particles as this would allow for much higher amounts of particles to be produced from collisions.

The energy distribution of Pythia8 and Herwig7 in 12 are very similar at lover energies, and have a high frequency. As this frequency decreases at higher energies, there is a larger amount of variation deriving from the statistical nature of MCEGs. Despite this, there is still a clear tendency of Herwig7 to produce more high energy particles than Pythia8. These conclusions fit well with expectations from Professor as a tuning program, being able to tune the generator better in areas with higher frequency due to lower statistical uncertainty.

The transverse momentum in figure 13 shows a similar pattern to the particle energy in figure 12, although the fact that Herwig7 produces less low energy particles is slightly more apparent. This is also consistent with figure 11, which is a figure that may benefit from a log scale in future. From these three graphs, it seems safe to conclude that Herwig7 is producing more higher rest-mass particles.

This makes sense with the collected data for both generators. Herwig7 has been configured to collect b-bbar events for the majority of this project, and this clearly indicates the comparison pythia8 data collected using default settings does not favour this interaction in the same way.

Figures 12 and 13 correlated with 11 all show Herwig7 producing particles with higher energies.

The Pseudorapidity charge ratio in figure 14 shows another successful tune. The ratio between the two generator data sets at the bottom of the graph clearly shows the

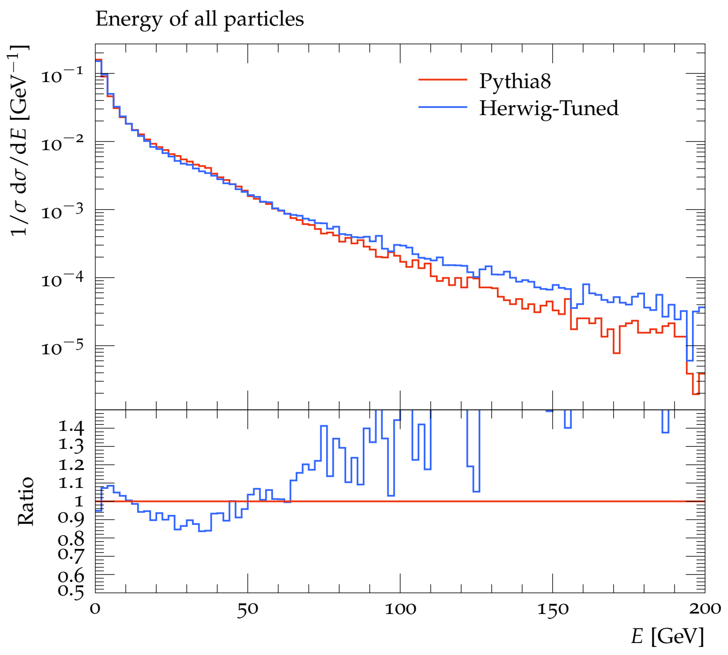

Figure 12: A good demonstration of tuning ability of Professor. Quite an accurate distribution at lower energies. Noise increases as energy gets higher, and Herwig appeaers to have a generally higher frequency of high-energy particles.

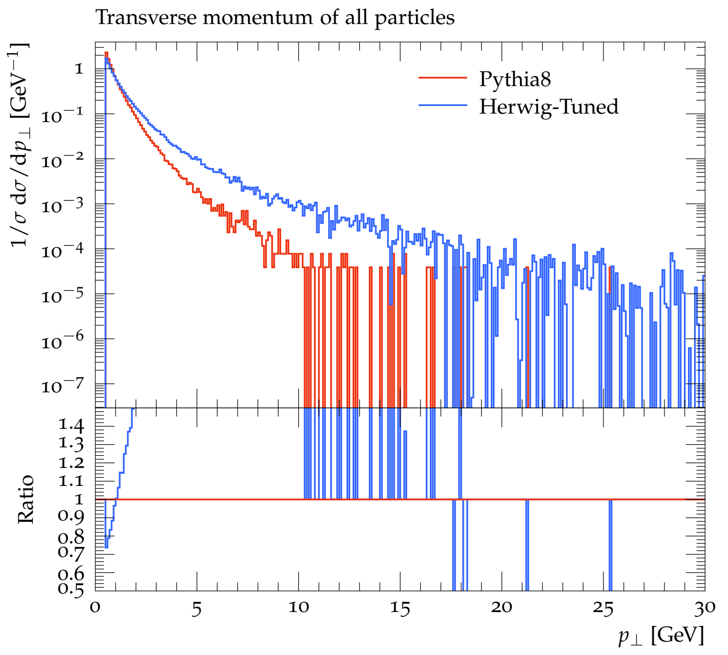

Figure 13: Herwig has a statistically higher frequency of high momentum particles. Comparing with figures 12 and 11, it is consistent with Herwig producing more high energy particles.

deviations are quite small, and likely down to statistical variance. Another point of information about this graph is the range across the full range of pseudorapidities. This is something that should not be present in fully tuned and cut data fit for LHCb, as LHCb does not record events below $\eta = 2.5$.

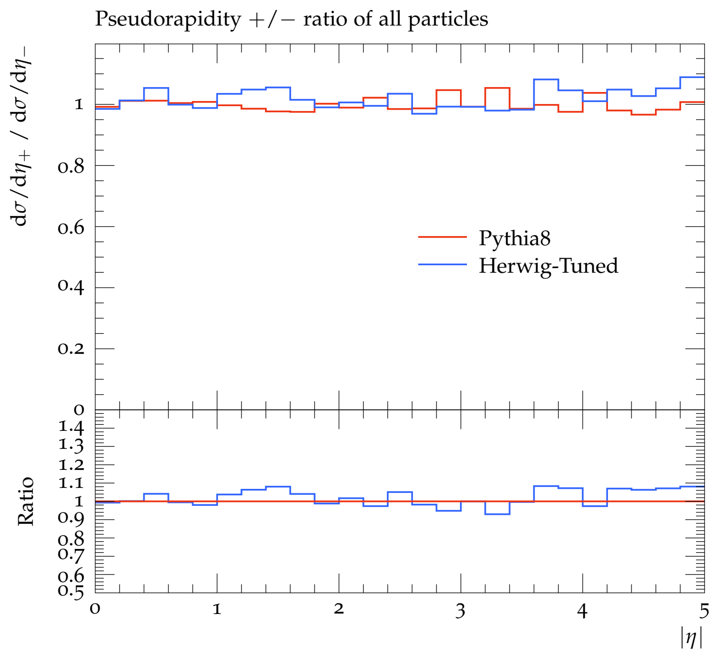

Figure 14: Another nice demonstration of a good fit. Variations mostly due to statistical nature of MCEGs. Worth noting a higher range of Pseudorapidities here than would be present for real LHCb data [10]

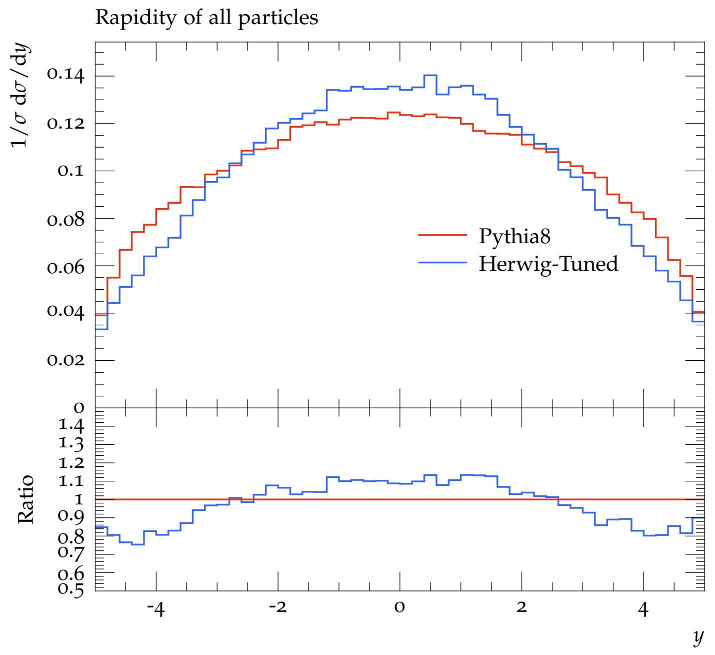

Figure 15: Whilst following the same general trends, there is a distinct difference in the shape of the two rapidity distributions.

The final Rivet graph that will be discussed in this report is the distribution of rapidities in figure 15. This value is analagous to a velocity, accounting for relativistic physics. This plot shows similar overall approximately gaussian structure between the Pythia8 and Herwig7 data, but with a few significant differences. The Herwig7 data has more particles produced of the most common middling rapidities, and lower freqencies of rapidities at both extremes, away from the centre of the distribution. This agrees with the fact that both datasets have the same number of events. Additionally, the Herwig7 data has more linear gradients at the edges of the graph. A similar overall rapidity structure agrees with the conclusion that Herwig7 is producing more massive particles, as the rapidity suggests overall similar velocities, and even fewer high velocities than Pythia8.

## 5 Encountered Issues

This section aims to outline some of the problems encountered over the course of the project in order to aid in completion of similar projects and future work.

One large issue encountered during this project was the passing of environment variables correctly through the system and into the Grid system used for parallel job processing. It is necessary for Gauss to be able to locate the Herwig7 Generator Files, and the Library containing the production file instructing Gauss how to interface with Herwig. These have been set as environment variables over the course of the project, however this caused issues with getting the jobs into the Grid.

A job is an object within Ganga, and has a specified variable of application. In the basic structure of an application variable, there is room to set the environment variables. However, once this application variable is filled with a ”GaudiExec” or Gaudi Executable type variable, this removes the ability to set any environment variables, and overrides any previous set. Unfortunately, Gauss is a Gaudi Executable. This proved a somewhat significant roadblock to this part of the project. It was discovered that the Gaudi Executable contains a variable for additional parameters, which will then be passed though lb-run to Gauss when the job is initiated and run through the Grid. Using this variable, it was eventually made possible to pass the environment variables through to the Gauss system being run on the Grid. Removing the requirements for these environment variables to be set in the future would remove this requirement for Grid jobs, and make data processing using Herwig7 and Gauss much easier.

The collection of timing data also proved a minor issue. Gauss outputs this information, but as part of its general output. This means that in order to pull this data, a program would have to be written to scrape the data out of the output. It may be possible to output this into a separate file for collection when using the grid. Under the current magnitude of data collection, manual collection proved quicker than writing this code, but is optimisation is to be seriously considered, the ability to acquire timing information automatically is something that would be extremely helpful to have.

There was also the quite unavoidable issue of the final data collection occurring extreme close to the end of the project. This reduced the amount of time that could be spent on full analysis of results, only really leaving enough time to gain a qualitative understanding of the data. Whilst this is be functionally enough, given that the results are extremely early data, intending to provide information for how to move further forward with the tuning, quantitative analysis is always useful to have as well.

# 6 Next Steps

## 6.1 Improving Efficiency

For further inspection of the efficiency of Herwig7, collection of larger number of events would be useful to determine whether the use of caching proves useful at higher numbers of events. Additionally, an inspection of internal code to investigate the apparent increase in timing variance would be a good idea should this behaviour continue with a larger amount of repeats.

## 6.2 Improving Tuning

Firstly, many more tunes should be taken for Herwig7. This project has successfully set up and demonstrated this process, and now this should be made use of to fully adapt Herwig7 for LHCb. The relevance of the varying variables to LHCb should be investigated, and the weightings for contributions to the tunings altered accordingly. New Rivet Analyses should be used and written in order to cut and bin data appropriately for LHCb, a key example being pseudorapidity. The addition of default Herwig7 data would also aid in comparison, and provide the ability to demonstrate the effects of the tuning on Herwig7 itself.

The collection of more data at each point in the parameter space, and increasing this amount for the comparison data should also improve the tuning, ideally improving the goodness of fit further. Extending the comparison data to LHCb data would also help to bring Herwig7 in line with LHCb.

Investigation of the relevance of additional parameters should also be considered, following on first with those also discusses in the CMS paper [9], and then all those listed on the Herwig7 MPI parameter page [8].

As for variables, the relevance of certain variables to LHCb should also be investigated, along with how they are altered by the tuning of varying parameters.

# 7 Conclusion

The project has been a great success. Herwig7 has been successfully implemented into Gauss, with the library now requiring a review of the code before submission to the LHCb Simulation Group for merging with Gauss. The optimisation of Herwig7 has been looked into, and compared with Pythia8. While as expected it is not as well optimised as Pythia, it is not by orders of magnitude significant, especially when being run on the LHCb Grid, another purpose Herwig7 has now been adapted for during the course of this project. Further use of Herwig7 with software such as Professor has also been demonstrated, with tunings achieved and analysed. The results have achieve reasonable fits, with a final goodness of fit of 1117, and a reasonable match to Pythia8 data with explainable differences. This should leave Herwig7 in a good place to be tuned to produce data useful to the LHCb Group.

## References

[1] Z. Harper, *Opening the door for the use of herwig in lhcb*, 2021.

[2] Müller, Dominik, *Adopting new technologies in the lhcb gauss simulation framework*, EPJ Web Conf. **214** (2019) 02004.

[3] G. Collaboration, *Gauss*, `https://gitlab.cern.ch/lhcb/Gauss`. Accessed: 05/2022.

[4] J. Bellm *et al.*, *Herwig 7.0/herwig++ 3.0 release note*, The European Physical Journal C **76** (2016) .

[5] M. Bähr, S. Gieseke, and M. H. Seymour, *Simulation of multiple partonic interactions in herwig*, Journal of High Energy Physics **2008** (2008) 076.

[6] M. Bähr, S. Gieseke, and M. H. Seymour, *Multiple interactions in herwig++*, 2008. doi: 10.48550/ARXIV.0806.4250.

[7] A. Buckley *et al.*, *Systematic event generator tuning for the LHC*, The European Physical Journal C **65** (2009) 331.

[8] T. H. Collaboration, *Multiple-parton interactions model*, `https://herwig.hepforge.org/tutorials/mpi/parameters.html#main-parameters-of-the-model`. Accessed: 05/2022.

[9] T. C. Collaboration, *Event generator tunes obtained from underlying event and multiparton scattering measurements*, The European Physical Journal C **76** (2016) .

[10] LHCb Collaboration, *Graphical comparison of the LHCb measurements of $W$ and $Z$ boson production with ATLAS and CMS*, .

[11] C. Bierlich *et al.*, *Robust independent validation of experiment and theory: Rivet version 3*, SciPost Physics **8** (2020) .

[12] A. Buckley and I. Bruce, *Mcfsparticles analysis*, `https://rivet.hepforge.org/analyses/MC_FSPARTICLES.html`. Accessed: 05/2022.

[13] A. Buckley and C. H. Christensen, *Hepforge*, `https://yoda.hepforge.org/`. Accessed: 05/2022.

[14] F. James, *Minuit: Function minimization and error analysis*, `https://root.cern.ch/download/minuit.pdf`, 1994.

[15] H. S. Andy Buckley, *Professor documentation*, `https://professor.hepforge.org/docs.sphinx220/index.html`. Accessed: 05/2022.

[16] J. T. Mościcki *et al.*, *Ganga : A tool for computational-task management and easy access to Grid resources*, Comput. Phys. Commun. **180** (2009) 2303, `arXiv:0902.2685`.

[17] A. Tsaregorodtsev *et al.*, *DIRAC: a community grid solution*, Journal of Physics: Conference Series **119** (2008) 062048.