Measurement of matter antimatter differences with the LHCb experiment

Introduction

Hit the grey arrow to extend a section

Welcome to the HEP lab course of the TU Dortmund LHCb group

This notebook is going to help you analysing data from the Large Hadron Collider (LHC) at CERN. The goal of your analysis is to make differences visible in the interactions of matter and antimatter. This lab course is designed to make you capable to perform data analysis of HEP data on your onw, on a level similiar to basic researches performed at CERN. Basic understanding of particle physics used to be sufficient and no very sophisticated knowledge is required.

First steps

Goal:

- Get familiar with the principles of this notebook
- Read simulated data ____

Like researchers at CERN you are going to write your own analysis code. To do so you are going to utilize `python`. Though, no experiences in using `python` are expected. There will be a

sufficient amount of hints and examples with the exercises. If you need more help using python, there are tutorials in the internet.

Examples of specific code that could be helpful for this analysis could be found in the example of an analysis of Nobel laureates. In this analysis utilizes similiar techniques as you will need for this course. Reading and understanding the code of the nobel laureates analysis could therefore make this lab course significantly easier.

Important: Each of the following code boxes will be executed by hitting Shift+Enter, after selecting the specific box. The replacement of `In [x]:` (x is empty or a integer) by `In [*]:` next to a code box indicates, that the code is currently running.

Reading simulated data

Simulated data are a great tool to develop your code and ensure it works correctly before analysing real data from LHCb. The latter are more difficult to analyse, since they contain two types of events. One being the events you are actually interested in, referred to as signal, and the ones that only look or behave similar, called background. In reality, the recorded data are also limited by the resolution of the detector. The data samples provided for this exersice only contain signal events under the assumption of a perfect detector.

```
# IMPORT PACKAGES CELL

from __future__ import print_function
from __future__ import division

# setup the notebook
%pylab inline

import functools
import os
import warnings
from subprocess import check_output

# python scientific computing
import numpy as np

# python plotting library
import matplotlib.pyplot as plt
import matplotlib.colors as colors
from matplotlib.patches import Rectangle
%matplotlib inline

from ipywidgets import interact
import pandas as pd
from root_pandas import read_root
from scipy import stats as st

warnings.filterwarnings("ignore")
rcParams['image.cmap'] = 'Blues'  # change default colormap
```

```
pd.set_option('display.max_columns', None)

# matplotlib styles for scientific figures
# https://github.com/garrettj403/SciencePlots
# https://pypi.org/project/SciencePlots/
import scienceplots
plt.style.use(['science', 'ieee', 'grid'])
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
# default figure size
plt.rcParams['figure.figsize'] = [8, 6]

# general font size (affects all text)
plt.rcParams['font.size'] = 16

# axis label font size
plt.rcParams['axes.labelsize'] = 16

# x-axis tick label size
plt.rcParams['xtick.labelsize'] = 14

# y-axis tick label size
plt.rcParams['ytick.labelsize'] = 14

# title font size
plt.rcParams['axes.titlesize'] = 18

# SAVE PLOT

# dpi resolution to save images
image_save_dpi = 1200
path='graphs/'
format='.pdf'

save = 0
```

If you need help with coding there are, in addition to the example code, hints in each section and an overview of useful functions in the instructions. Alternatively you can ask your supervisor for advice.

```
# load the simulated data
sim_data = read_root('/data/PhaseSpaceSimulation.root')

##---------------------------##

# preselection over real data

preselection = ('!H1_isMuon & H1_ProbPi<0.5 & H1_ProbK>0.5 & '+
                '!H2_isMuon & H2_ProbPi<0.5 & H2_ProbK>0.5 & '+
                '!H3_isMuon & H3_ProbPi<0.5 & H3_ProbK>0.5')
```

```
real_data = read_root(['/data/B2HHH_MagnetUp.root',
                       '/data/B2HHH_MagnetDown.root'],
                      where=preselection)
```

The data contains information about 'events' that were observed in the detector. An event refers to the particles produced when two proton bunches of the beams collided at the LHC. If you think of the data as a table, each row of the table is the results from a different collision and each column is a different measured quantity.

We are interested in analysing the decays of particles called B+ or B- mesons decaying into three other mesons called kaons (K+ or K-). The events you have been given are those in which this process may have occurred. The detector has been used to reconstruct tracks that may have come from the kaons. You are given the measured momenta, charge, and likelihood of the tracks being kaons. You are given information for three tracks in each event, the ones that could be the three kaons that a B+ or B- meson has decayed into. The instructions give an overview about the recorded information for each event.

```
# make a table of the data variables here
sim_data.head()
```

```
   B_FlightDistance  B_VertexChi2    H1_PX    H1_PY      H1_PZ
H1_ProbK  \
0               0.0           1.0  3551.84  1636.96   23904.14
1.0
1               0.0           1.0 -2525.98 -5284.05   35822.00
1.0
2               0.0           1.0  -700.67  1299.73    8127.76
1.0
3               0.0           1.0  3364.63  1397.30  222815.29
1.0
4               0.0           1.0  -581.66 -1305.24   22249.59
1.0

   H1_ProbPi  H1_Charge  H1_IPChi2  H1_isMuon     H2_PX     H2_PY
H2_PZ  \
0        0.0         -1        1.0          0  41507.15  15980.59
331663.64
1        0.0          1        1.0          0 -43182.91 -96553.03
585289.31
2        0.0         -1        1.0          0  -1411.99   3550.96
16120.27
3        0.0          1        1.0          0   2192.42   1369.87
161924.75
4        0.0         -1        1.0          0   1153.05  -1998.17
66134.22

   H2_ProbK  H2_ProbPi  H2_Charge  H2_IPChi2  H2_isMuon     H3_PX
H3_PY  \
```

```
0       1.0        0.0           1        1.0           0  36100.40
16546.83
1       1.0        0.0          -1        1.0           0  -8648.32 -
16617.56
2       1.0        0.0           1        1.0           0 -13483.34
10860.77
3       1.0        0.0          -1        1.0           0   1925.16   -
551.12
4       1.0        0.0           1        1.0           0  -2820.04  -
8305.43

       H3_PZ  H3_ProbK  H3_ProbPi  H3_Charge  H3_IPChi2  H3_isMuon
0  295600.61       1.0        0.0         -1        1.0          0
1   98535.13       1.0        0.0         -1        1.0          0
2   79787.59       1.0        0.0          1        1.0          0
3   40420.96       1.0        0.0          1        1.0          0
4  250130.00       1.0        0.0         -1        1.0          0
```

Hints

Creating a table - Use your `head()` - remember to look at the example analysis to see how it was done there.

Reconstruction of the invariant mass

Aims:

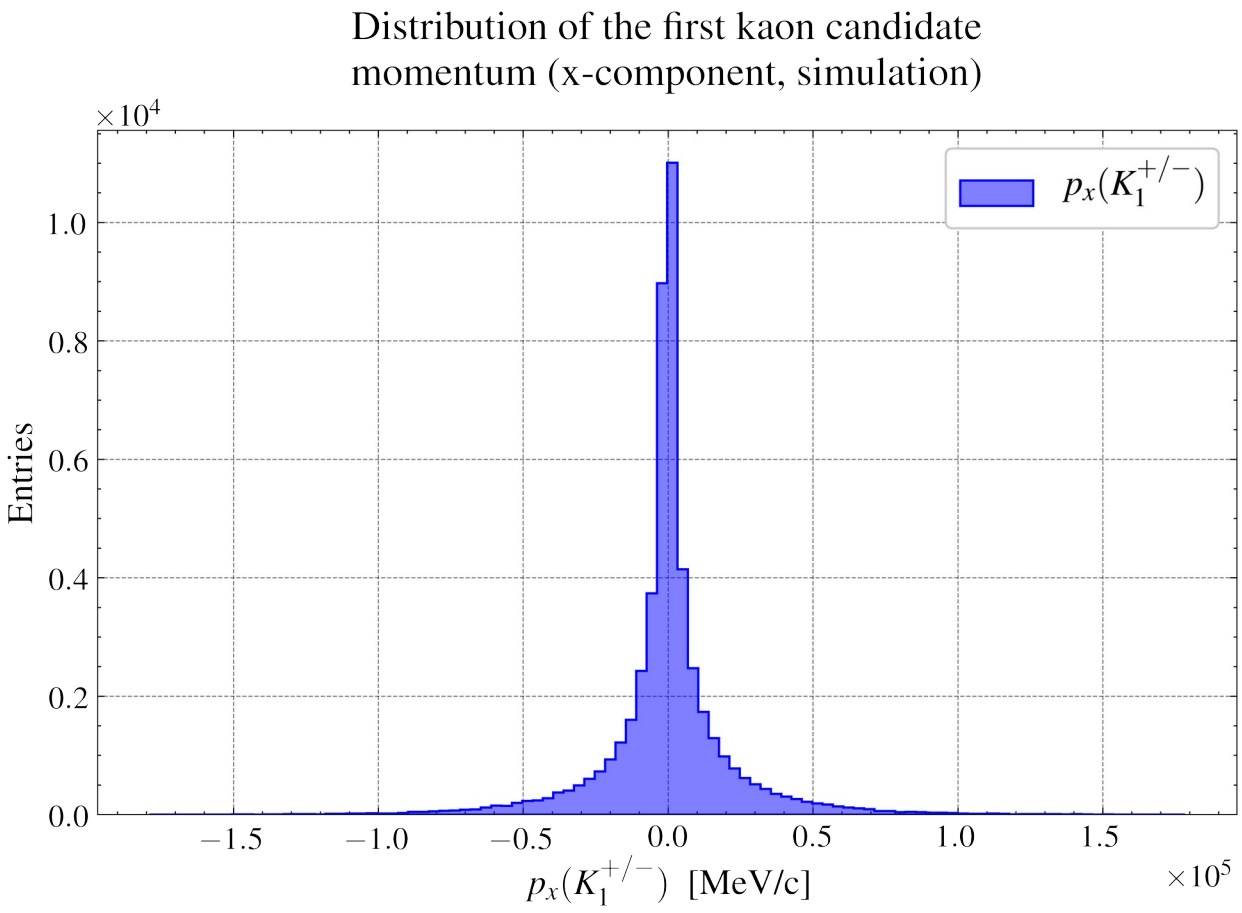
- Plot histograms of the momenta of one of the kaon candidates
- Calculate the energy of each of the kaon candidates
- Plot the invariant masses of the B+ or B- mesons

Plotting a feature:

You can plot any features of the data in a histogram. Choose any suitable binning that allows you to observ the distribution of the variable clearly. Try making a histogram for the first kaon candidate's momentum x-component (H1_PX):

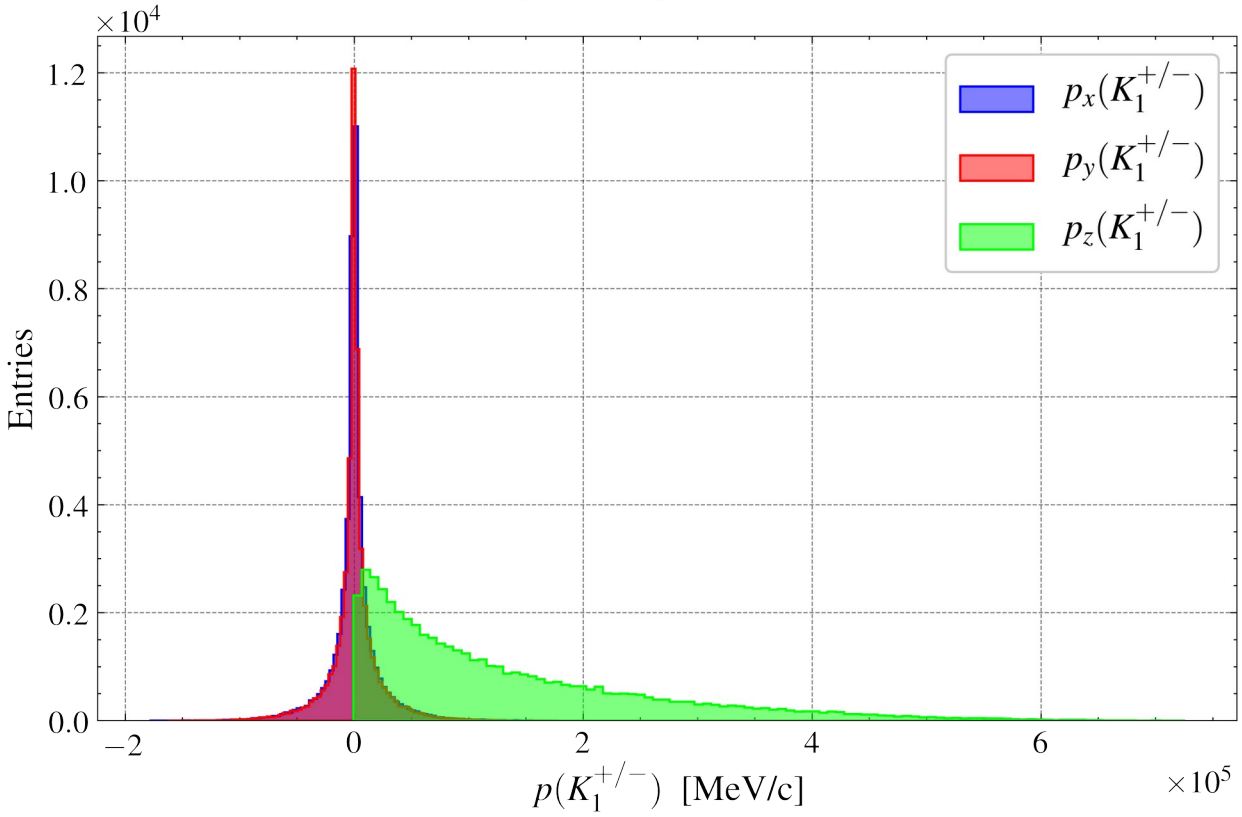
```
# make a histogram of the H1_PX variable here

# plot histogram of H1_PX variable
plt.figure()

plt.hist(sim_data['H1_PX'],
         bins=100,
         density=False,
         histtype='stepfilled',
         facecolor=(0,0,1,0.5),
```

```
          edgecolor=(0,0,1,1.0),
          lw=1,
          ls='solid',
          label='$p_{x}(K_{1}^{+/-})$')

plt.title('Distribution of the first kaon candidate'
          '\n'
          'momentum (x-component, simulation)',
          pad=20)
plt.xlabel('$p_{x}(K_{1}^{+/-})\;$ [MeV/c]')
plt.ylabel('Entries')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='best')
plt.tight_layout()
if save==1: plt.savefig(path+'H1_PX'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



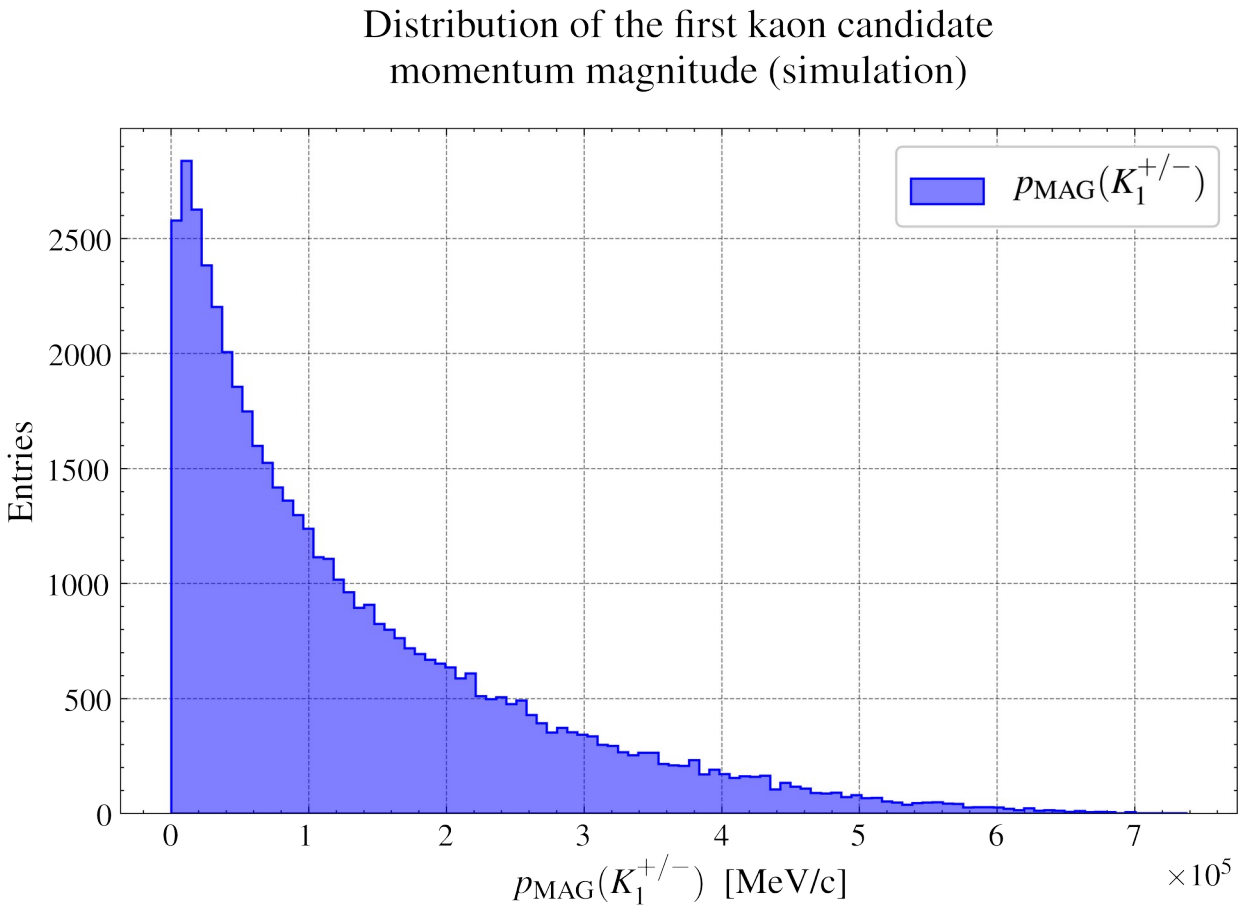
Momentum is a **vector** quantity, it has x,y, and z components. Compare the distributions of the different momentum components. Try calculating the **magnitude** of the momentum of the first kaon candidate and plot a histogram of the resulting distribution. (You'll need the `H1_PX`, `H1_PY` and `H1_PZ` variables.)

```
# make a histogram of the H1_PX, H1_PY and H1_PZ variables here
plt.figure()

plt.hist(sim_data['H1_PX'],
         bins=100,
         density=False,
         histtype='stepfilled',
         facecolor=(0,0,1,0.5),
         edgecolor=(0,0,1,1.0),
         lw=1,
         ls='solid',
         label='$p_{x}(K_{1}^{+/-})$')

plt.hist(sim_data['H1_PY'],
         bins=100,
         density=False,
         histtype='stepfilled',
         facecolor=(1,0,0,0.5),
         edgecolor=(1,0,0,1.0),
         lw=1,
         ls='solid',
         label='$p_{y}(K_{1}^{+/-})$')

plt.hist(sim_data['H1_PZ'],
         bins=100,
         density=False,
         histtype='stepfilled',
         facecolor=(0,1,0,0.5),
         edgecolor=(0,1,0,1.0),
         lw=1,
         ls='solid',
         label='$p_{z}(K_{1}^{+/-})$')

plt.title('Distribution of the first kaon candidate'
          '\n'
          'momenta (per components, simulation)',
          pad=20)
plt.xlabel('$p(K_{1}^{+/-})\;$ [MeV/c]')
plt.ylabel('Entries')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='best')
plt.tight_layout()
if save==1: plt.savefig(path+'H1_PXYZ'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```




```
# calculate a variable for the magnitude
#of the momentum of the first kaon
H1_MAG = np.sqrt((sim_data['H1_PX']**2)+
                 (sim_data['H1_PY']**2)+
                 (sim_data['H1_PZ']**2))

# plot histogram of H1_MAG variable
plt.figure()

plt.hist(H1_MAG,
         bins=100,
         density=False,
         histtype='stepfilled',
         facecolor=(0,0,1,0.5),
         edgecolor=(0,0,1,1.0),
         lw=1,
         ls='solid',
         label='$p_{\mathrm{MAG}}(K_{1}^{+/-})$')

plt.title('Distribution of the first kaon candidate'
          '\n'
          'momentum magnitude (simulation)',
```

```
          pad=20)
plt.xlabel(r'$p_{\mathrm{MAG}}(K_{1}^{+/-})\;$ [MeV/c]')
plt.ylabel('Entries')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='best')
plt.tight_layout()
if save==1: plt.savefig(path+'H1_MAG'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



Hints

Histogram plotting - You can use the `hist()` function. The parameters `bins(n)` and `range(x,y)` allow you to chose the binning and the range of the histogram.

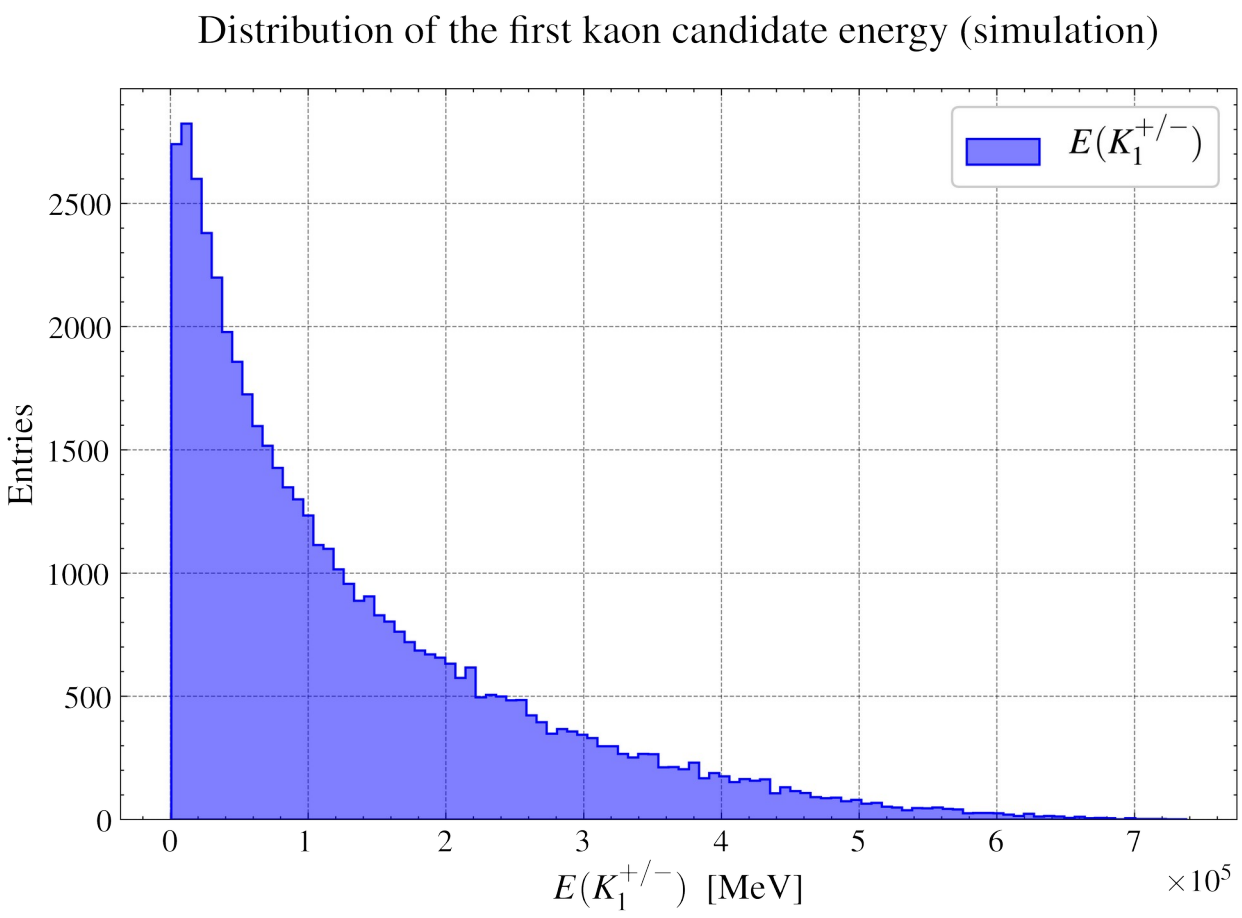
Energy and mass

Einstein's theory of special relativity relates energy, mass and momentum. The momentum of the kaon candidates is measured with the detector (For one candidate you already plotted the components and the magnitude). The invariant mass of the kaon is well known and you can look

it up. We wish to determine the energy of the kaons. With all these information you can calculate the energy of the kaon candidates easily using:

$$E^2 = p^2 + m^2$$

```
# calculate the energy of the first kaon
K_PDG_M = 493.677  # [MeV/c^2], from PDG
H1_E = np.sqrt((H1_MAG**2)+(K_PDG_M**2))

# plot a histogram of this variable

# plot histogram of H1_E variable
plt.figure()

plt.hist(H1_E,
         bins=100,
         density=False,
         histtype='stepfilled',
         facecolor=(0,0,1,0.5),
         edgecolor=(0,0,1,1.0),
         lw=1,
         ls='solid',
         label='$E(K_{1}^{+/-})$')

plt.title('Distribution of the first kaon candidate '+
          'energy (simulation)',
          pad=20)
plt.xlabel('$E(K_{1}^{+/-})\;$ [MeV]')
plt.ylabel('Entries')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='best')
plt.tight_layout()
if save==1: plt.savefig(path+'H1_E'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



Hints

Energy calculation - Use the magnitude of momentum variable you calculated above and the known invariant mass of the kaon to work out the energy of the first hadron. Calculate the energy squared, and then the energy and plot this.

Kaon mass - you can find the kaon mass on wikipedia or in physics textbooks. A commonly used reference for these kind of information is the Particle Data Group (PDG). All knowledge about the properties of discovered particles are collected there.

Calculate the momenta and energies of the second and third kaon candidates as well.

```
# calculate variables for the energy of the other two kaons

# magnitude of the momentum and energy of the second kaon
H2_MAG = np.sqrt((sim_data['H2_PX']**2)+
                 (sim_data['H2_PY']**2)+
                 (sim_data['H2_PZ']**2))
H2_E = np.sqrt((H2_MAG**2)+(K_PDG_M**2))

# magnitude of the momentum and energy of the third kaon
H3_MAG = np.sqrt((sim_data['H3_PX']**2)+
```

```
                  (sim_data['H3_PY']**2)+
                  (sim_data['H3_PZ']**2))
H3_E = np.sqrt((H3_MAG**2)+(K_PDG_M**2))
```

Adding features of the B meson

In this analysis we are looking for B+ or B- mesons (see B meson) that have decayed into the three charged kaons.

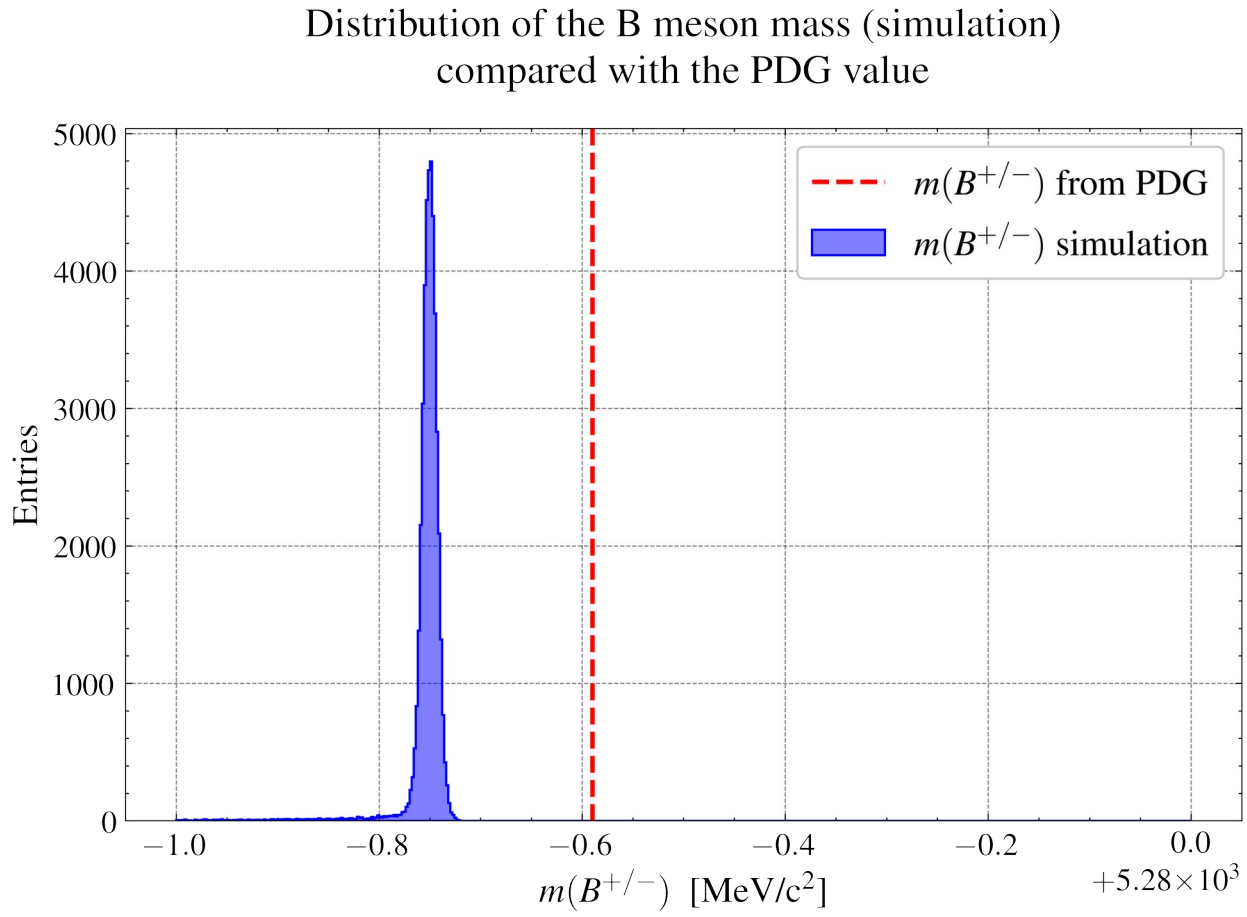
Energy is a conserved quantities. This means that you can use the energy of the three 'daughter' kaons, which you have calculated above, to calculate the energy that the B meson that decayed into them must have.

Momentum is also a conserved quantity. Hence you can also use the momenta of the 'daughter' kaons to calculate the momentum of the B meson. But be careful - momentum is a **vector** quantity.

Using the energy and the magnitude of the momentum you can use the energy-momentum relationship to calculate the invariant mass of the B meson.

```
# calculate the energy of the B meson

B_E = H1_E+H2_E+H3_E

# calculate the momentum components of the B meson

B_PX = sim_data['H1_PX']+sim_data['H2_PX']+sim_data['H3_PX']
B_PY = sim_data['H1_PY']+sim_data['H2_PY']+sim_data['H3_PY']
B_PZ = sim_data['H1_PZ']+sim_data['H2_PZ']+sim_data['H3_PZ']

# and the magnitude of the momentum of the B meson

B_MAG = np.sqrt((B_PX**2)+(B_PY**2)+(B_PZ**2))

# calculate the B meson invariant mass

B_PDG_M = 5279.41  # [MeV/c^2], from PDG
B_M = np.sqrt((B_E**2)-(B_MAG**2))

# plot the B meson invariant mass in a histogram

# plot histogram of B_M variable
plt.figure()

plt.hist(B_M,
         bins=500,
         range=[5279, 5280],
         density=False,
         histtype='stepfilled',
         facecolor=(0,0,1,0.5),
```

```
          edgecolor=(0,0,1,1.0),
          lw=1,
          ls='solid',
          label='$m(B^{+/-})$ simulation')

plt.axvline(B_PDG_M, color='r', lw=2, ls='--',
            label='$m(B^{+/-})$ from PDG')

plt.title('Distribution of the B meson mass (simulation)'
          '\n'
          'compared with the PDG value',
          pad=20)
plt.xlabel('$m(B^{+/-})\;$ [MeV/c$^2$]')
plt.ylabel('Entries')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='best')
plt.tight_layout()
if save==1: plt.savefig(path+'B_M'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



You should have a graph that sharply peaks at the mass of the B+ meson. The mass of the B+ and B- meson are the same. Check that the peak of your graph is at the known mass of the B meson.

Recall that you have made this plot for simulated data. How might you expect the plots for real data to look different ?

Congratulations, you finished your first analysis steps! In the next section you will start to work with the real LHC data.

Hints

B Meson Energy - From energy conservation, the energy of the B meson will be the sum of the energies of the three kaons: $E_B = E_{K1} + E_{K2} + E_{K3}$, where E_B is the energy of the B meson, E_{K1}, E_{K2}, E_{K3} are the energies of each of the kaons.

B meson momentum - From momentum conservation, the X component of the momentum of the B meson will be the sum of the X momentum components of the three Kaons : $p\,x_B = p\,x_{K1} + p\,x_{K2} + p\,x_{K3}$, where $p\,x$ is the X direction component of the momentum of the B meson, $p\,x_{K1}, p\,x_{K2}, p\,x_{K3}$ are the X direction components of the momenta of the three kaons. You can then do the same with the Y and Z components. Having obtained the X,Y, and z components of the B momentum you can find the magnitude of the momentum of the B meson.

B meson invariant mass* - Rearrange the equation $E^2 = p^2 + m^2$ to find m^2. Using the values of the magnitude of the momentum of the B meson and the B meson Energy, find the mass of the B meson.

Histogram plotting - Take care that the range of your mass plot is set suitably that you can see the mass peak.

Units - The data you are provided use the unit 'MeV' (106 electron volts) for momenta.

Working with real data

Aims:

- Filter out data that is not coming from the B+ → K+K+K− decay channel, or the antiparticle equivalent B- → K+K-K−
- Plot a histogram of B-meson mass for the real data and observe how different cuts affect the data

In the previous section simulated data have been used to determine the invariant B± mass. Now you are going to utilize real LHCb data, recorded in 2011 during the first major data taking period of the LHC.

The data you are given has been filtered to select only events that are likely to have come from B+ or B- mesons decaying into three final state charged particles. You are interested in the case where these three final state paticles are charged kaons K+ or K-.

Preselection

You want to apply a preselection to the data to select only events that have a high probability to contain three kaons in the final state. Therefore, you will need to:

- Ensure that they are not muons (use `H1_isMuon` and the equivilant for `H2` and `H3`)
- Require that they each have a low probability of being pions (e.g. `H1_ProbPi < 0.5`)
- Require that they each have a high probability of being a kaon (e.g. `H1_ProbK > 0.5`)

You need to find a balance between making cuts that are too loose and include too many background events and too tight and reject too many of your signal events. Feel free to come back to this stage later and adjust your cuts to see the impact.

Now develop your preselection and name it `'preselection'`. We have provided an example preselection in the hints, so feel free to use that to get started if you wish. Redefine the preselection after you have studied the plots and check the impact.

```
# already done above
```

This next line of code just loads the real data into a new DataFrame and applies your created preselection `'preselection'`. This may take a few minutes due to the size of the data sample.

```
# already done above

real_data.head()
```

```
   B_FlightDistance  B_VertexChi2        H1_PX        H1_PY
H1_PZ  \
0         10.428140      4.056947 -4168.055539  -704.953919
24548.766576
1         33.591307      2.220242  1295.910334   -61.956920
35569.606902
2         16.531858     11.593388  1493.048615  1944.600925
33003.915368
3          4.102797     11.321708  -236.694032   843.809179
28207.109021
4          3.509346      5.739399   123.873807   856.290459
51956.566616

   H1_ProbK  H1_ProbPi  H1_Charge    H1_IPChi2  H1_isMuon        H2_PX
\
0  0.771199   0.051726         -1  1002.657075          0 -3393.636093

1  0.932006   0.058690         -1  6672.092707          0   613.484785

2  0.953512   0.104768         -1  1731.923563          0   373.613994

3  0.655129   0.045339         -1     5.033764          0  -561.905858

4  0.940484   0.078221         -1     7.005894          0   110.753386
```

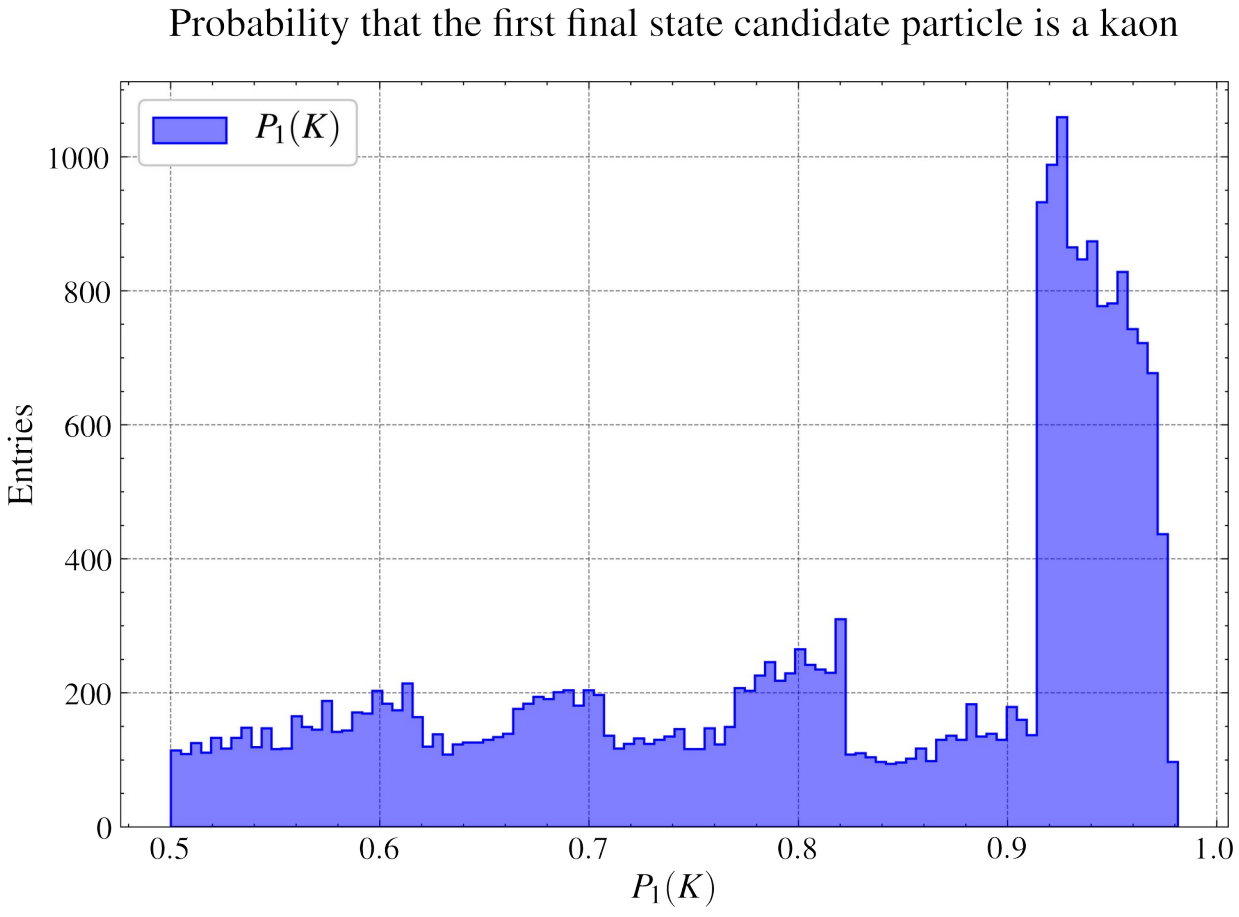
```
         H2_PY         H2_PZ  H2_ProbK  H2_ProbPi  H2_Charge
H2_IPChi2  \
0  -706.874508  19358.787446  0.917991   0.038719          1
714.174519
1   464.017357  17701.755537  0.515930   0.043657          1
737.219732
2   843.614548  30062.084752  0.868579   0.039157          1
499.998835
3  -189.352668  30100.030163  0.601544   0.040908          1
6.588705
4  1070.975864  24974.211721  0.920603   0.041739          1
2.090424

   H2_isMuon        H3_PX        H3_PY         H3_PZ  H3_ProbK
H3_ProbPi  \
0          0 -8864.962492 -7428.640292  77446.175923  0.934580
0.128720
1          0  -341.750434  3317.423370  24380.200353  0.968883
0.129317
2          0  4779.029220  -287.829864  53908.496894  0.844950
0.064803
3          0 -1085.332141 -3076.759324  30036.196827  0.912293
0.080841
4          0  1639.997962  2660.275891  19441.349314  0.702135
0.028641

   H3_Charge     H3_IPChi2  H3_isMuon
0          1    536.585115          0
1          1  23332.453390          0
2          1   3484.193030          0
3          1    489.046805          0
4          1    888.535134          0
```

Make histograms of the probability of a final state particle being a kaon or a pion. These will help to guide you on suitable probability values to cut on.

You can also consider more sophisticated options like 2-D plots of kaon and pion probabilities or different values of the cuts for the different final state particles.

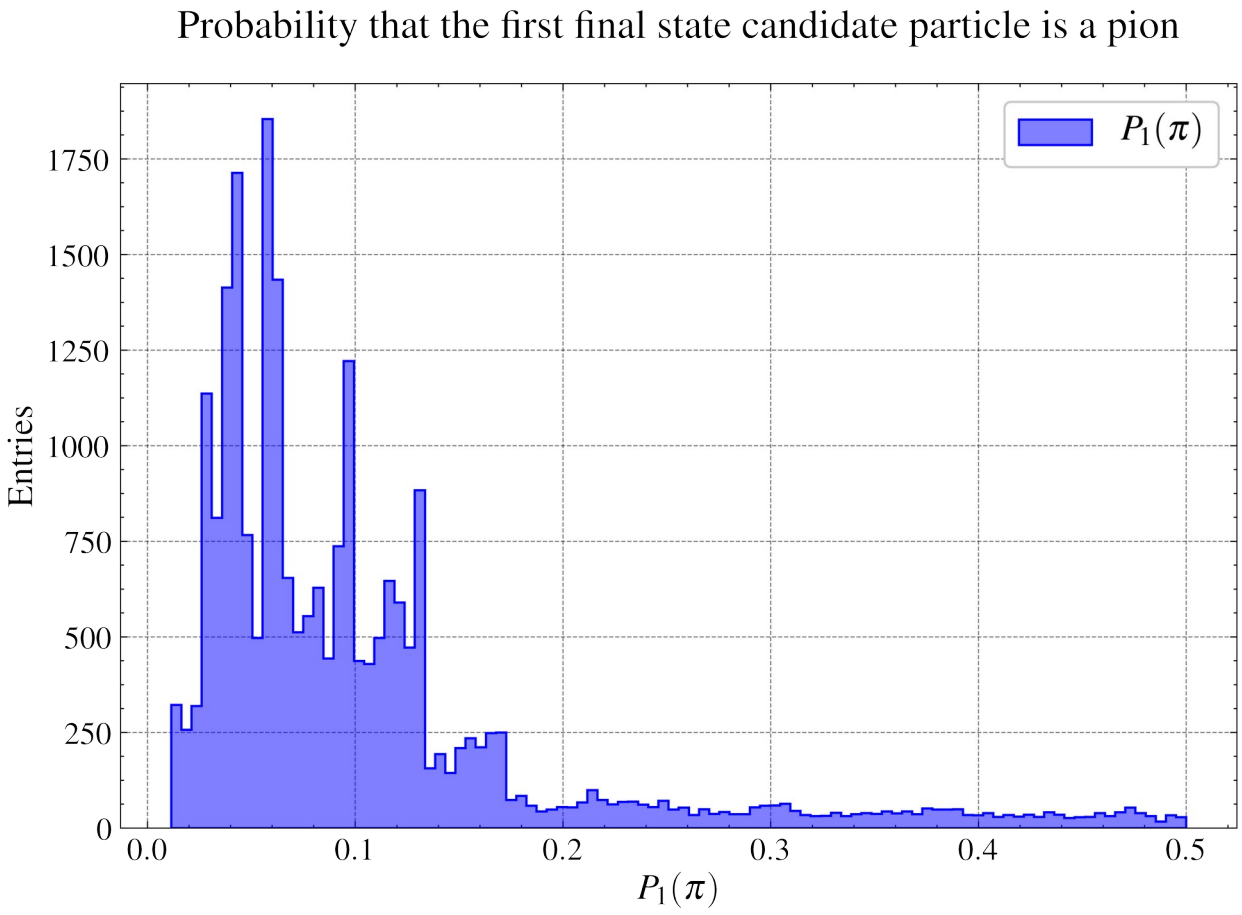
```
# plot the probability that a final state particle is a kaon
plt.figure()

plt.hist(real_data['H1_ProbK'],
         bins=100,
         density=False,
         histtype='stepfilled',
         facecolor=(0,0,1,0.5),
         edgecolor=(0,0,1,1.0),
```

```
         lw=1,
         ls='solid',
         label='$P_{1}(K)$')

plt.title('Probability that the first final state candidate '+
          'particle is a kaon',
          pad=20)
plt.xlabel('$P_{1}(K)$')
plt.ylabel('Entries')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='upper left')
plt.tight_layout()
if save==1: plt.savefig(path+'H1_ProbK'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```




```
# plot the probability that a final state particle is a pion
plt.figure()

plt.hist(real_data['H1_ProbPi'],
         bins=100,
```

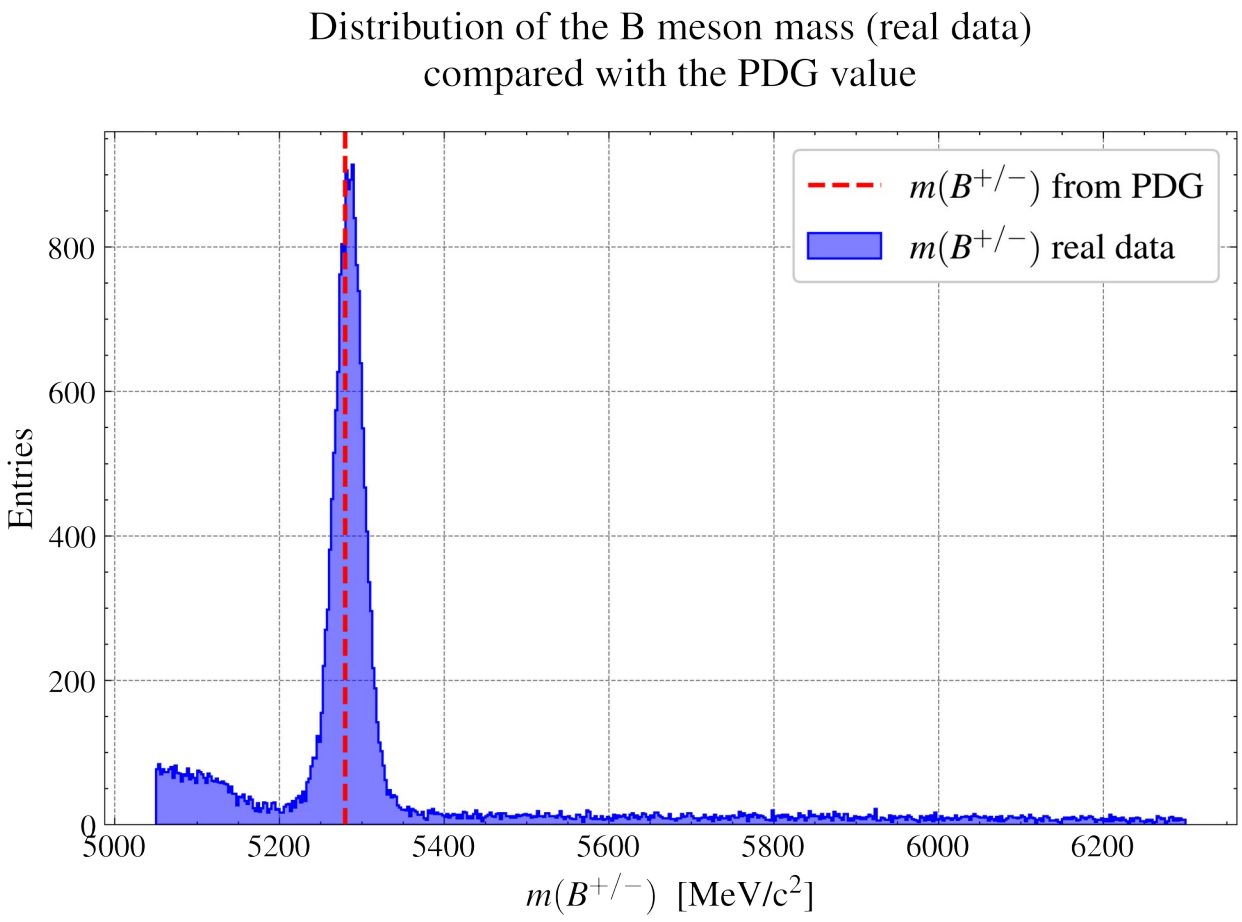
```
          density=False,
          histtype='stepfilled',
          facecolor=(0,0,1,0.5),
          edgecolor=(0,0,1,1.0),
          lw=1,
          ls='solid',
          label='$P_{1}(\pi)$')

plt.title('Probability that the first final state candidate '+
           'particle is a pion',
           pad=20)
plt.xlabel('$P_{1}(\pi)$')
plt.ylabel('Entries')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='best')
plt.tight_layout()
if save==1: plt.savefig(path+'H1_ProbPi'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



Now calculate the invariant mass of the B meson for the real data and plot a histogram of this. Which differences do you observe compared to the mass plotted with simulated data? Try to explain the differences.

```
B_PX_data = real_data['H1_PX']+real_data['H2_PX']+real_data['H3_PX']
B_PY_data = real_data['H1_PY']+real_data['H2_PY']+real_data['H3_PY']
B_PZ_data = real_data['H1_PZ']+real_data['H2_PZ']+real_data['H3_PZ']

B_MAG_data = np.sqrt((B_PX_data**2)+(B_PY_data**2)+(B_PZ_data**2))

##---------------------------##

H1_MAG_data = np.sqrt((real_data['H1_PX']**2)+
                      (real_data['H1_PY']**2)+
                      (real_data['H1_PZ']**2))
H1_E_data = np.sqrt((H1_MAG_data**2)+(K_PDG_M**2))

H2_MAG_data = np.sqrt((real_data['H2_PX']**2)+
                      (real_data['H2_PY']**2)+
                      (real_data['H2_PZ']**2))
H2_E_data = np.sqrt((H2_MAG_data**2)+(K_PDG_M**2))

H3_MAG_data = np.sqrt((real_data['H3_PX']**2)+
                      (real_data['H3_PY']**2)+
                      (real_data['H3_PZ']**2))
H3_E_data = np.sqrt((H3_MAG_data**2)+(K_PDG_M**2))

B_E_data = H1_E_data+H2_E_data+H3_E_data

##---------------------------##

B_M_data = np.sqrt((B_E_data**2)-(B_MAG_data**2))

# plot histogram of B_M_data variable
plt.figure()

plt.hist(B_M_data,
         bins=500,
         #range=[5279, 5280],
         density=False,
         histtype='stepfilled',
         facecolor=(0,0,1,0.5),
         edgecolor=(0,0,1,1.0),
         lw=1,
         ls='solid',
         label='$m(B^{+/-})$ real data')

plt.axvline(B_PDG_M, color='r', lw=2, ls='--',
            label='$m(B^{+/-})$ from PDG')

plt.title('Distribution of the B meson mass (real data)'
```

```
          '\n'
          'compared with the PDG value',
          pad=20)
plt.xlabel('$m(B^{+/-})\;$ [MeV/c$^2$]')
plt.ylabel('Entries')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='best')
plt.tight_layout()
if save==1: plt.savefig(path+'B_M_data'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



Experiment with different cuts and observe the impact of them on the invariant mass plot. You should select a set of cuts which preserves most of the signal while removing as much as possible of the background at the same time. Once you have finalised the selection on particle identification, make cuts on the reconstructed particle mass to select the events in the B meson mass peak, removing the background events which lie at lower and higher invariant masses.

Searching for global matter anti-matter differences

In this section you will start to study matter antimatter differences (CP Violation). The attribute 'global' means that you are looking for differences across all ranges of energy and momentum (the kinematics) of the kaons into which the charged B mesons have decayed. Later you will look at 'local' differences in different regions of the kinematics.

Aims:

- Calculate the global CP asymmetry
- Work out the statistical uncertainty
- Determine, if there is evidence for CP violation in this decay

In order to quantify the matter antimatter asymmetry in this process, we will compare the B+ meson with its anti-particle B-. Use the provided charge information (`H1_Charge`, `H2_Charge` and `H3_Charge`) to distinguish B+ and B-.

```
# make a variable for the charge of the B mesons
cha = (real_data['H1_Charge']
       +real_data['H2_Charge']
       +real_data['H3_Charge'])
```

Now count the numbers of events of each of the two types (N+ and N-) and the difference between those.

```
# make variables for the numbers of positive and negative B mesons
pos = cha[cha>0].size
neg = cha[cha<0].size
```

Use the following equation to determine the Asmmetry:

$$A = \frac{N^{+¿} - N^{-}}{N^{+¿} + N^{-}} ¿$$

```
# calculate the value of the asymmetry, by using the formula above,
# and then print it
A = (pos-neg)/(pos+neg)
print('Asymmetry'+
      '\n'+
      f'A = {A:.4f}')
```

```
Asymmetry
A = 0.0370
```

Hints

Differentiating between B+ and B-

- Charge is a conserved quantity. The charge of the B meson is equal to the sum of the charges of the three daughter particles.
- You can use `len(real_data.query('B_Charge == charge'))` to count the number of mesons, where `B_Charge` is the variable you created and `charge` is `1` or `-1`.

Estimating the significance of the deviation

You will now need to calculate the statistical uncertainty of the asymmetry. You can do so using the formula:

$$\sigma_A = \sqrt{\frac{1 - A^2}{N^{+¿+N^-}}¿}$$

The significance of the result, sigma, is found by dividing the value for asymmetry by its uncertainty. A value exceeding three sigma is considered "evidence" by particle physicists while a value of five sigma or more can be called an "observation" or "discovery".

```
# calculate the statistical significance of your result and print it
sig = np.sqrt((1-A**2)/(pos+neg))
s = A/sig

print('Standard deviation (statistical uncertainty)'+
      '\n'+
      f'\u03C3\u1D00 = {sig:.4f}\n')
print('Significance (statistical uncertainty)'+
      '\n'+
      f'S = {s:.4f}')
```

```
Standard deviation (statistical uncertainty)
σA = 0.0065

Significance (statistical uncertainty)
S = 5.7291
```

Congratulations! You have performed your first search for CP violaton.

Till now you have only considered the statistical uncertainty. Your measurement will also have other sources of uncertainty known as systematic uncertainties. Try to include one systematic uncertainty for the production asymmetry of the B meson and recalculate the significance.

```
sig1 = np.sqrt((1/100)**2+(sig)**2)
s1 = A/sig1

print('Standard deviation (statistical uncertainty)'+
      '\n'+
      f'\u03C3\u1D00 = {sig1:.4f}\n')
```

```
print('Significance (statistical and systematic uncertainty)'+
      '\n'+
      f'S = {s1:.4f}')

# just by looking at the significances, one can deduce that
# systematic uncertainties dominate over statistical ones:
# by taking into consideration the former, a measurement that could
# have been considered a discovery becomes merely evidence
```

```
Standard deviation (statistical uncertainty)
σA = 0.0119

Significance (statistical and systematic uncertainty)
S = 3.1104
```

Dalitz plots and two body resonances

Aims:

- Produce Dalitz plots of the simulation and real data sample
- Create ordered and binned dalitz plots to improve the visibility of resonances
- Identify two body resonances in the Dalitz plots

In this stage we introduce you to an important technique for analysing three body decays, in our case the decay of charged B mesons to three kaons. This is known as a Dalitz plot.

The decay of the B meson can proceed either directly to the three-body final state or via an intermediate particle. For example, B+ → K+K+K−, could proceed through the decay B+ → K+R0, where R0 is a neutral particle resonance which can decay R0 → K+K-. Dalitz plots can be used to identify these resonances which are visible as bands on the Dalitz plot.

The kinematics of a three-body decay can be fully described using only two variables. The energies and momenta of the three kaons are not independent of each other as they all come from the decay of a B meson and energy and momentum are conserved. The axes of the plots conventionally are the squared invariant masses of two pairs of the decay products. The created 2D plots can display structures in their density distribution.

In our decay B+ → K+1K+2K−3, the kaons are nummerated to distinguish them. Therefore, we can calculate the invariant mass of three possible combinations that could correspond to intermediate resonances: R01 → K+1K-2, R++2 → K+1K+3 und R03 → K-2K+3.

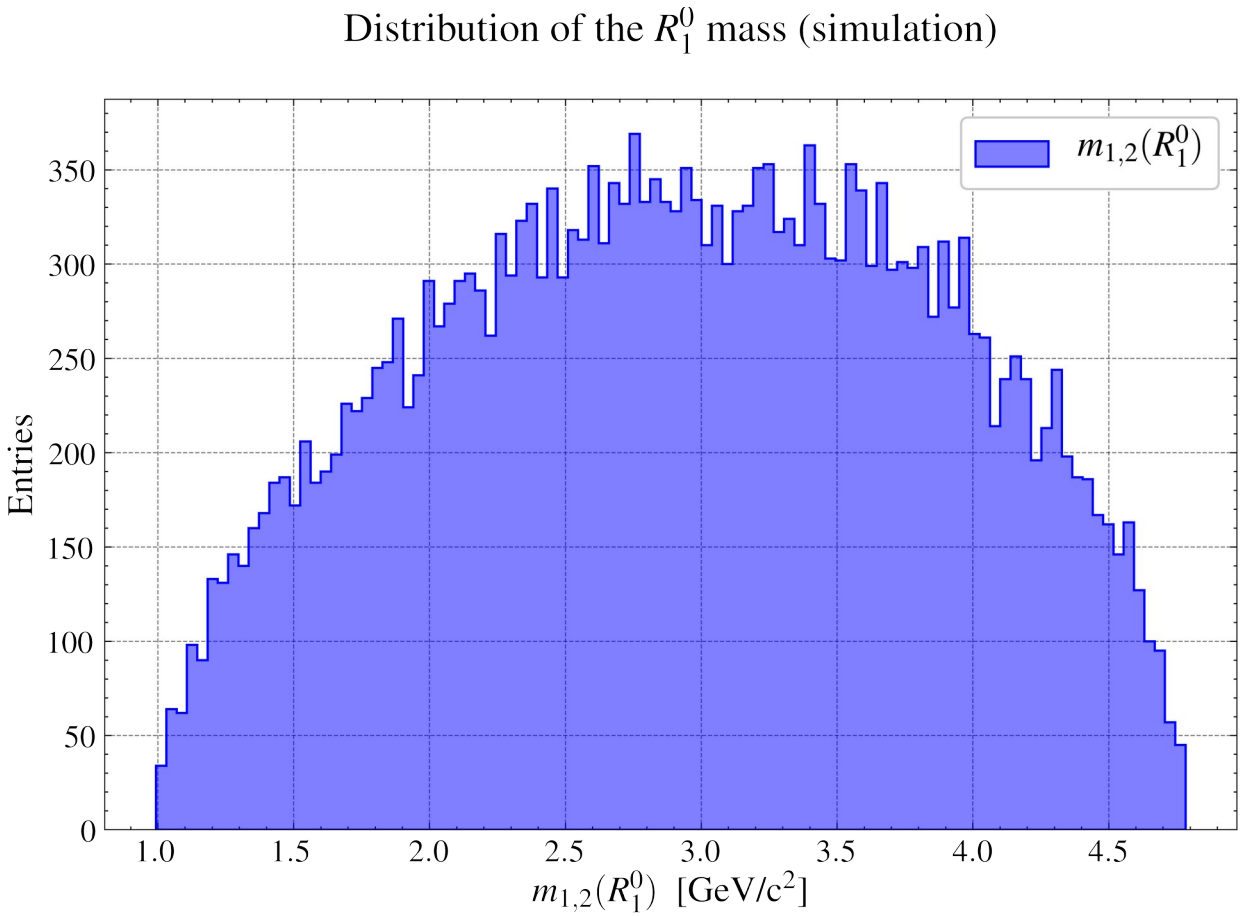
The potential resonance R++2 would be a doubly charged resonance. We would not expect to see any resonances corresponding to this as mesons are composed of one quark and one anti-quark and their charges cannot add up to two units.

But the uncharged resonances, R01 and R03, could be observed. Hence you should compute the invariant mass combinations for these. The square of these masses should be used as the Dalitz variables for a 2D plot.

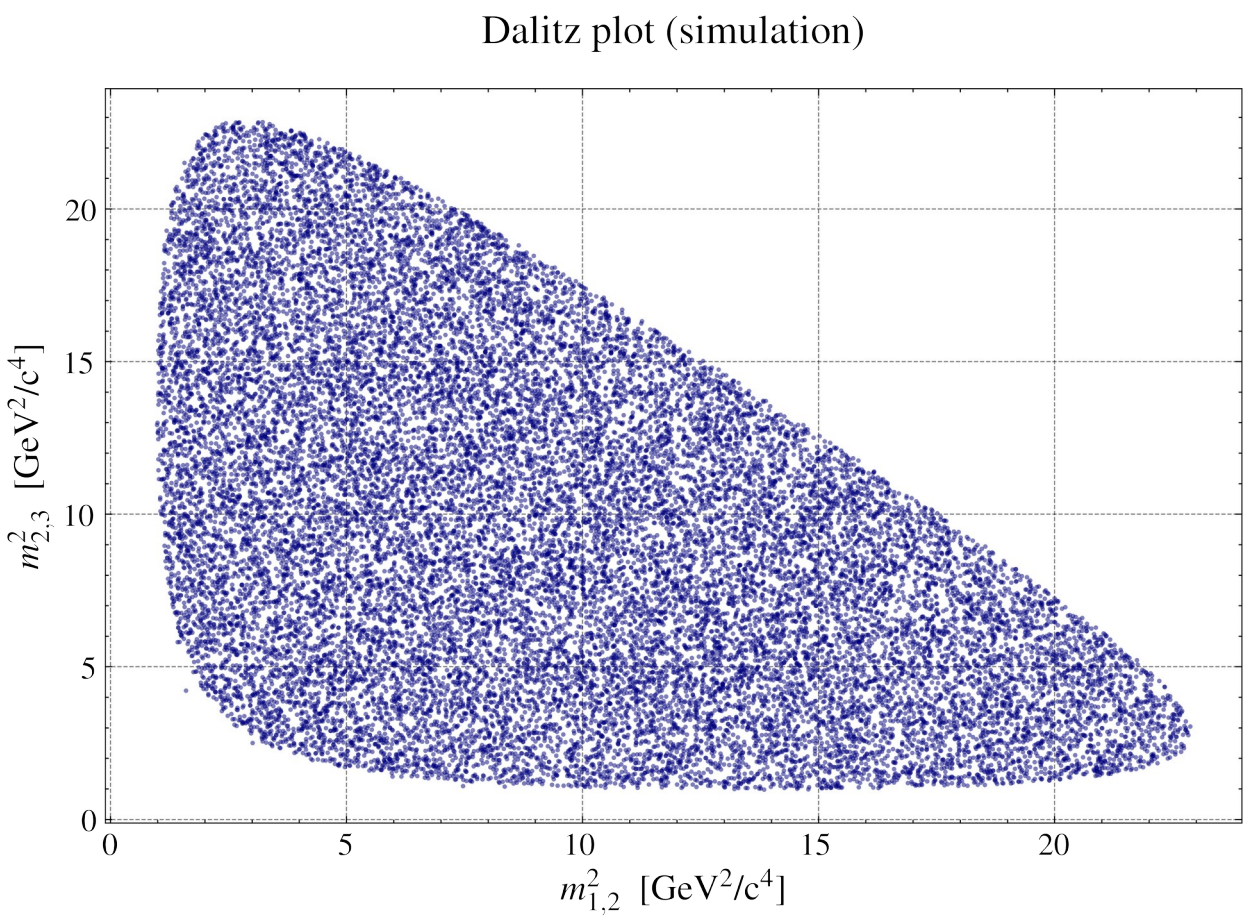
We suggest you make these plots first for the simulation data. In the simulation there are no intermediate resonances and your plot should be of uniform density inside the range physically allowed by energy and momentum conservation.

```
# calculate the invariant masses for each possible hadron pair
combination
mask = (((sim_data['H1_Charge']== 1) &
         (sim_data['H2_Charge']==-1) &
         (sim_data['H3_Charge']== 1)
        ) |
        ((sim_data['H1_Charge']==-1) &
         (sim_data['H2_Charge']== 1) &
         (sim_data['H3_Charge']==-1)
        ))

B_PX_12 = sim_data['H1_PX'][mask]+sim_data['H2_PX'][mask]
B_PY_12 = sim_data['H1_PY'][mask]+sim_data['H2_PY'][mask]
B_PZ_12 = sim_data['H1_PZ'][mask]+sim_data['H2_PZ'][mask]

B_PX_23 = sim_data['H2_PX'][mask]+sim_data['H3_PX'][mask]
B_PY_23 = sim_data['H2_PY'][mask]+sim_data['H3_PY'][mask]
B_PZ_23 = sim_data['H2_PZ'][mask]+sim_data['H3_PZ'][mask]

B_MAG_12 = np.sqrt((B_PX_12**2)+(B_PY_12**2)+(B_PZ_12**2))
B_MAG_23 = np.sqrt((B_PX_23**2)+(B_PY_23**2)+(B_PZ_23**2))

H1_MAG = np.sqrt((sim_data['H1_PX'][mask]**2)+
                 (sim_data['H1_PY'][mask]**2)+
                 (sim_data['H1_PZ'][mask]**2))
H1_E = np.sqrt((H1_MAG**2)+(K_PDG_M**2))

H2_MAG = np.sqrt((sim_data['H2_PX'][mask]**2)+
                 (sim_data['H2_PY'][mask]**2)+
                 (sim_data['H2_PZ'][mask]**2))
H2_E = np.sqrt((H2_MAG**2)+(K_PDG_M**2))

H3_MAG = np.sqrt((sim_data['H3_PX'][mask]**2)+
                 (sim_data['H3_PY'][mask]**2)+
                 (sim_data['H3_PZ'][mask]**2))
H3_E = np.sqrt((H3_MAG**2)+(K_PDG_M**2))

B_E_12 = H1_E+H2_E
B_E_23 = H2_E+H3_E

B_M_12 = np.sqrt((B_E_12**2)-(B_MAG_12**2))
B_M_23 = np.sqrt((B_E_23**2)-(B_MAG_23**2))

# conversion Mev to Gev
B_M_12 = B_M_12/(10**3)
B_M_23 = B_M_23/(10**3)
```

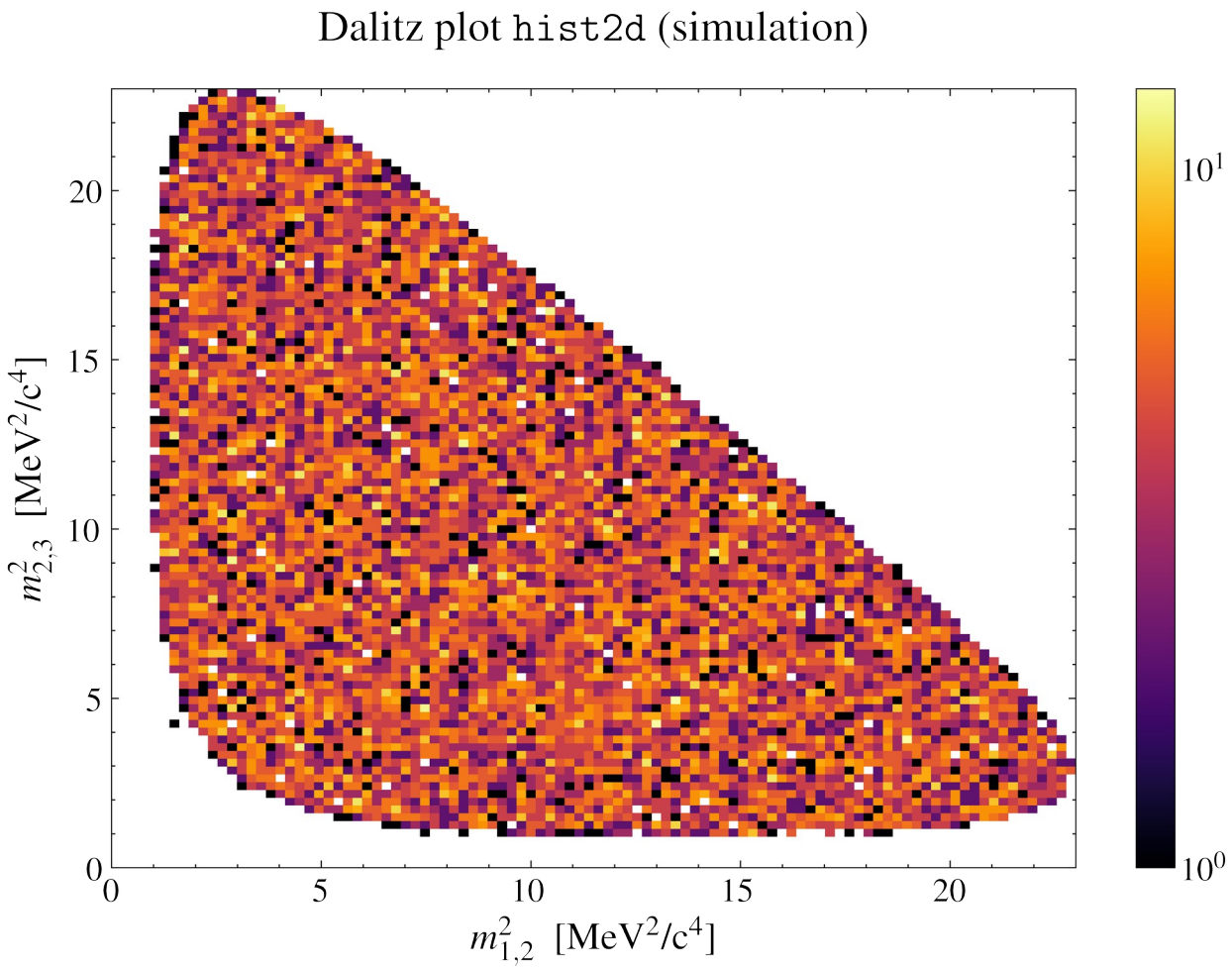
```
x_max = np.int(np.max(B_M_12**2))+1
y_max = np.int(np.max(B_M_23**2))+1

# plot the invariant mass for one of these combinations
plt.figure()

plt.hist(B_M_12,
         bins=100,
         density=False,
         histtype='stepfilled',
         facecolor=(0,0,1,0.5),
         edgecolor=(0,0,1,1.0),
         lw=1,
         ls='solid',
         label='$m_{1,2}(R_{1}^{0})$')

plt.title('Distribution of the $R_{1}^{0}$ mass (simulation)',
          pad=20)
plt.xlabel('$m_{1,2}(R_{1}^{0})\;$ [GeV/c$^2$]')
plt.ylabel('Entries')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='best')
plt.tight_layout()
if save==1: plt.savefig(path+'B_M_12'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```




```
# make a Dalitz plot with labelled axes for the simulation data
plt.figure()

plt.scatter(B_M_12**2, B_M_23**2,
            s=4,
            c='navy',
            marker='o',
            cmap='viridis',
            alpha=0.5,
            edgecolors='none')

plt.title('Dalitz plot (simulation)',
          pad=20)
plt.xlabel('$m_{1,2}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{2,3}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'dalitz_sim'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```




```
# make a Dalitz plot with hist2d for the simulation data
plt.figure()

plt.hist2d(B_M_12**2, B_M_23**2,
           bins=100,
           range=[[0, x_max], [0, y_max]],
           cmap=mpl.cm.inferno,
           norm=matplotlib.colors.LogNorm())

plt.colorbar()

plt.title('Dalitz plot $\mathtt{hist2d}$ (simulation)',
          pad=20)
plt.xlabel('$m_{1,2}^2\;$ [MeV$^2$/c$^4$]')
plt.ylabel('$m_{2,3}^2\;$ [MeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'hist2d_sim'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



Hint

Calculating the invariant mass - Use the same technique as you did above for the B meson. But instead of using all three finalstate particles use only two. **Plotting the Dalitz plot** - You can use the `scatter()` function from the `matplotlib` library or use the example given in the example analysis. Remember to use the square of each two-body mass.

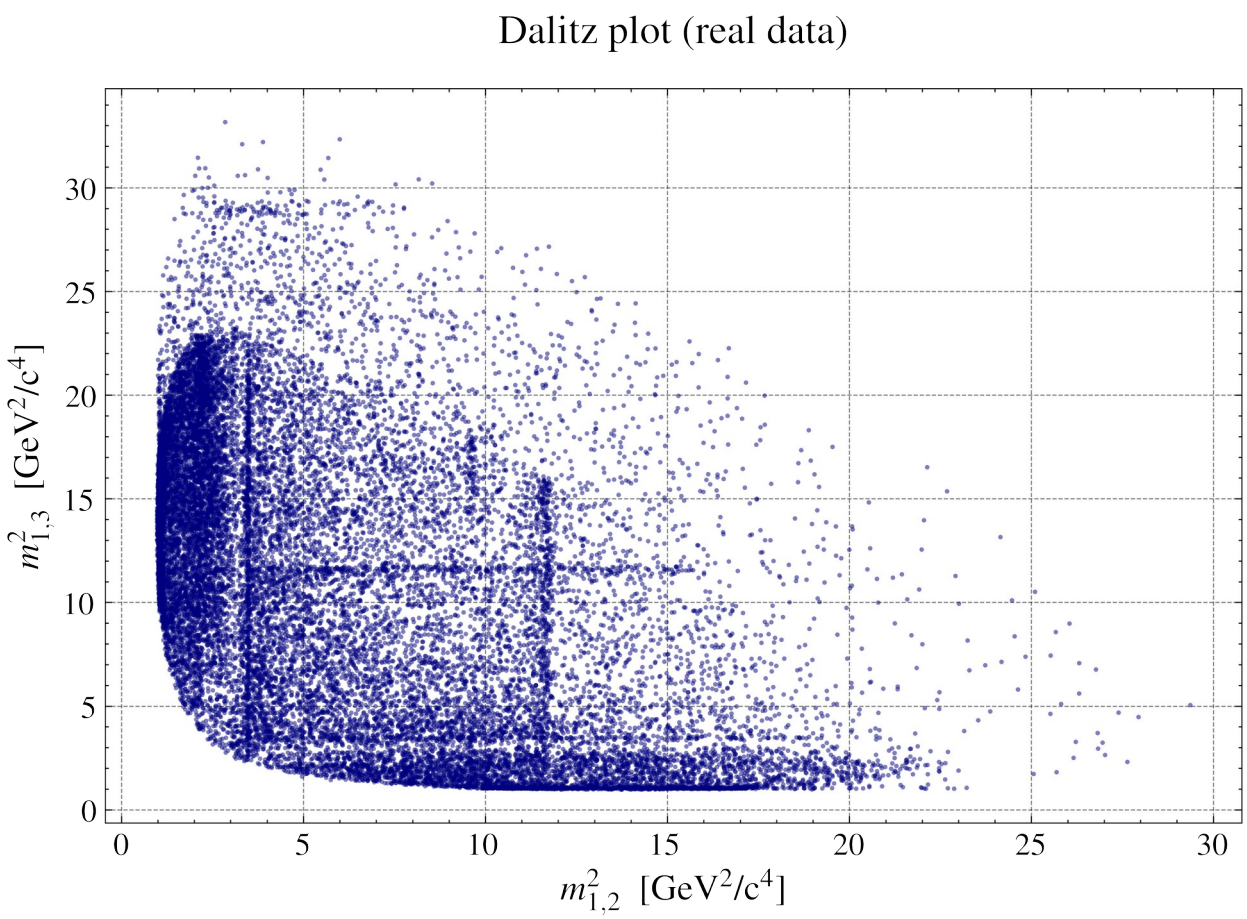
Adding Dalitz plot for real data

Now draw a Dalitz plot for the real data. Check that your two constructed resonances R02 and R03 have a neutral charge.

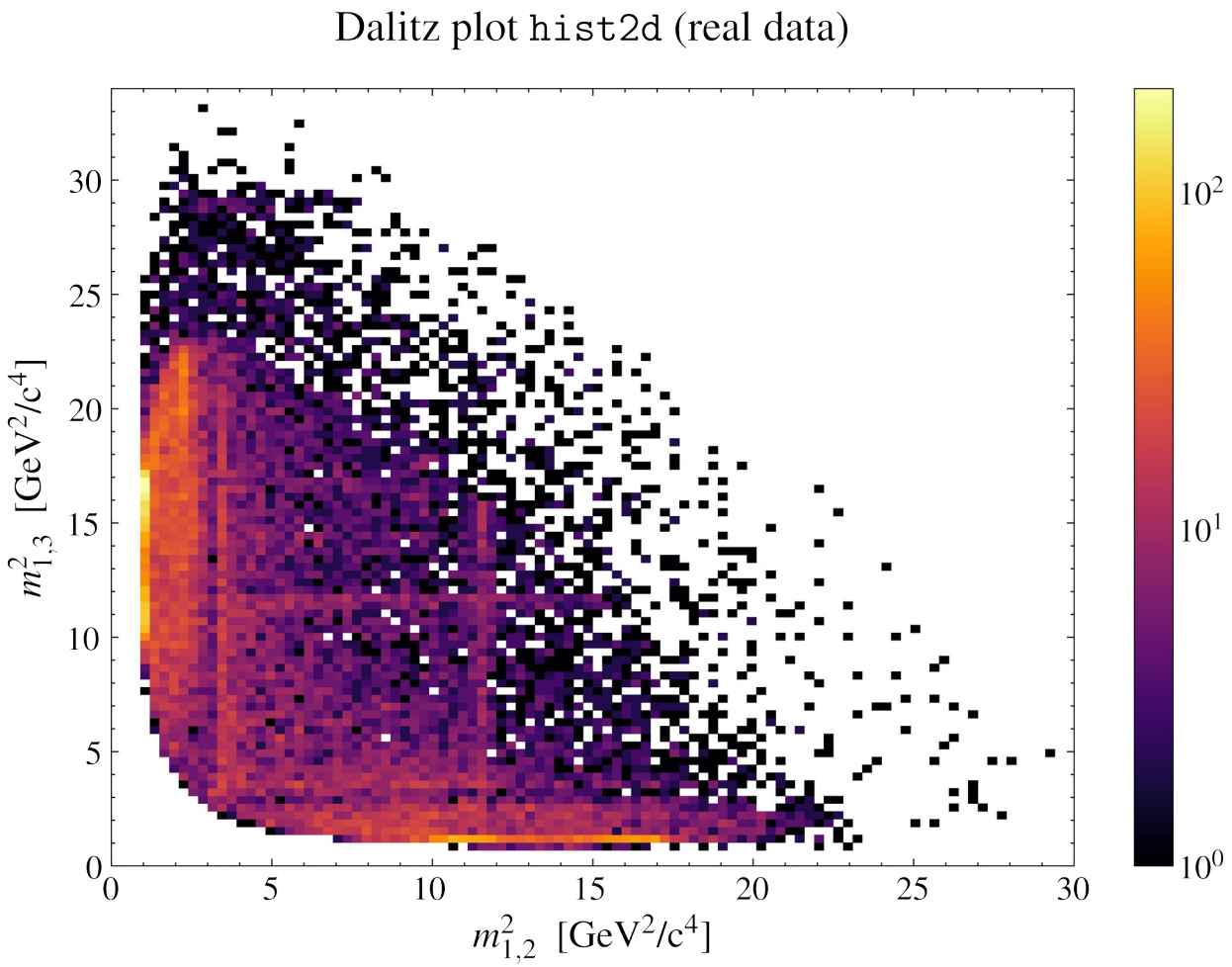
!!! NOTE: in the Dalitz plot of the simulated data you used $(m_{12})^2$ and $(m_{23})^2$, imposing the `mask conditions: + | - | + or - | + | -`. For the real data this cannot be repeated because the charge of H_2 is equal to that of H_3, so I choose to use $(m_{12})^2$ and $(m_{13})^2$, imposing the conditions `mask: + | - | - or - | + | +`. I wonder if I should then also change the chosen variables, and the conditions on the charge, for the simulated data → try!

```
# calculate the invariant masses for each possible hadron pair
combination in the real data
mask = (((real_data['H1_Charge']== 1) &
         (real_data['H2_Charge']==-1) &
         (real_data['H3_Charge']==-1)
        ) |
        ((real_data['H1_Charge']==-1) &
         (real_data['H2_Charge']== 1) &
         (real_data['H3_Charge']== 1)
        ))

B_PX_data_12 = real_data['H1_PX'][mask]+real_data['H2_PX'][mask]
B_PY_data_12 = real_data['H1_PY'][mask]+real_data['H2_PY'][mask]
B_PZ_data_12 = real_data['H1_PZ'][mask]+real_data['H2_PZ'][mask]

B_PX_data_13 = real_data['H1_PX'][mask]+real_data['H3_PX'][mask]
B_PY_data_13 = real_data['H1_PY'][mask]+real_data['H3_PY'][mask]
B_PZ_data_13 = real_data['H1_PZ'][mask]+real_data['H3_PZ'][mask]

B_MAG_data_12 = np.sqrt((B_PX_data_12**2)+
                        (B_PY_data_12**2)+
                        (B_PZ_data_12**2))
B_MAG_data_13 = np.sqrt((B_PX_data_13**2)+
                        (B_PY_data_13**2)+
                        (B_PZ_data_13**2))

H1_MAG_data = np.sqrt((real_data['H1_PX'][mask]**2)+
                      (real_data['H1_PY'][mask]**2)+
                      (real_data['H1_PZ'][mask]**2))
H1_E_data = np.sqrt((H1_MAG_data**2)+(K_PDG_M**2))

H2_MAG_data = np.sqrt((real_data['H2_PX'][mask]**2)+
                      (real_data['H2_PY'][mask]**2)+
                      (real_data['H2_PZ'][mask]**2))
H2_E_data = np.sqrt((H2_MAG_data**2)+(K_PDG_M**2))

H3_MAG_data = np.sqrt((real_data['H3_PX'][mask]**2)+
                      (real_data['H3_PY'][mask]**2)+
                      (real_data['H3_PZ'][mask]**2))
H3_E_data = np.sqrt((H3_MAG_data**2)+(K_PDG_M**2))

B_E_data_12 = H1_E_data+H2_E_data
B_E_data_13 = H1_E_data+H3_E_data

B_M_data_12 = np.sqrt((B_E_data_12**2)-(B_MAG_data_12**2))
B_M_data_13 = np.sqrt((B_E_data_13**2)-(B_MAG_data_13**2))

# conversion Mev to Gev
B_M_data_12 = B_M_data_12/(10**3)
B_M_data_13 = B_M_data_13/(10**3)
```

```
x_max = np.int(np.max(B_M_data_12**2))+1
y_max = np.int(np.max(B_M_data_13**2))+1

# make a Dalitz plot for the real data
# (with your preselection cuts applied)
plt.figure()

plt.scatter(B_M_data_12**2, B_M_data_13**2,
            s=4,
            c='navy',
            marker='o',
            cmap='viridis',
            alpha=0.5,
            edgecolors='none')

plt.title('Dalitz plot (real data)',
          pad=20)
plt.xlabel('$m_{1,2}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{1,3}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'dalitz_real'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```




```
# make a Dalitz plot with hist2d for the real data
plt.figure()

plt.hist2d(B_M_data_12**2, B_M_data_13**2,
           bins=100,
           range=[[0, x_max], [0, y_max]],
           cmap=mpl.cm.inferno,
           norm=matplotlib.colors.LogNorm())

plt.colorbar()

plt.title('Dalitz plot $\mathtt{hist2d}$ (real data)',
          pad=20)
plt.xlabel('$m_{1,2}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{1,3}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'hist2d_real'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```

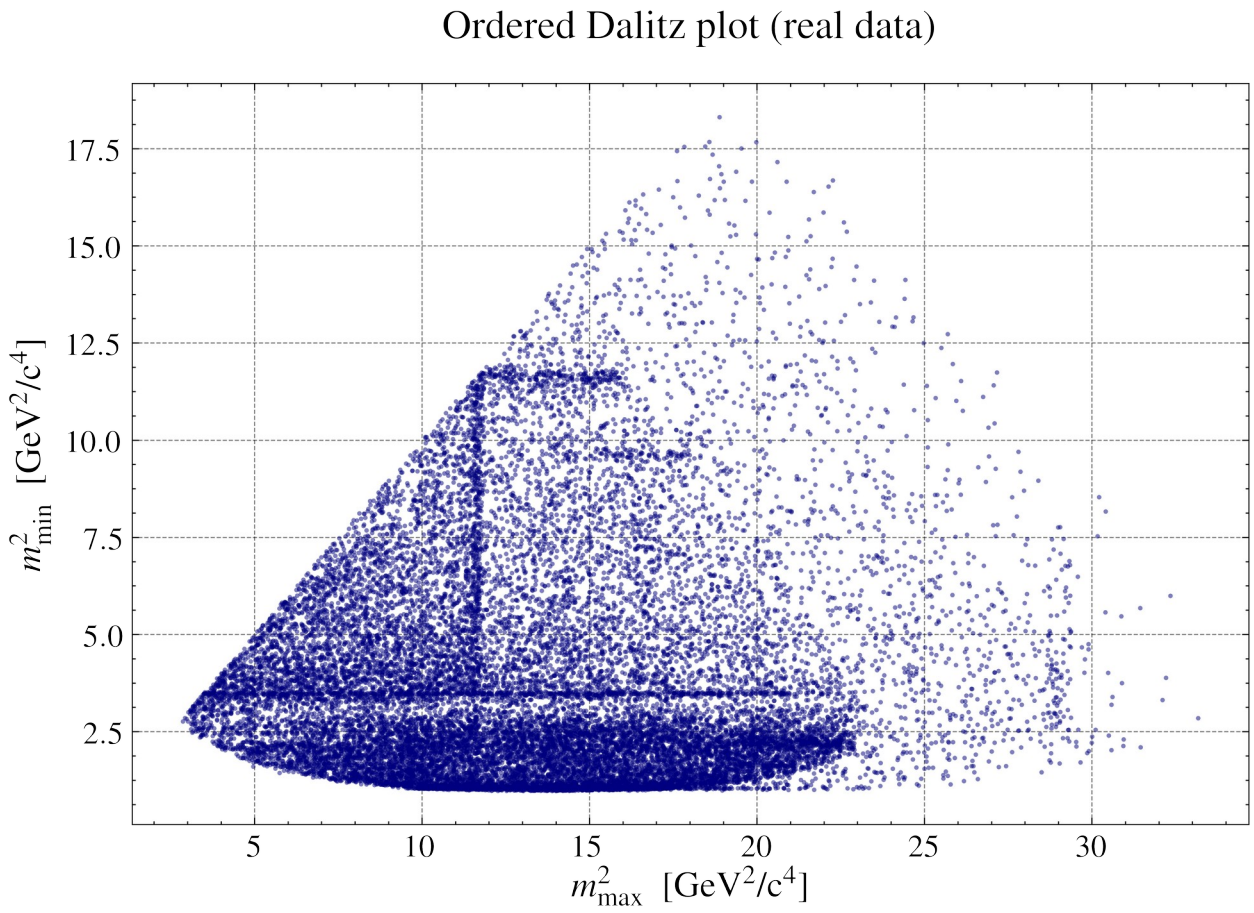


Ordering the Dalitz variables

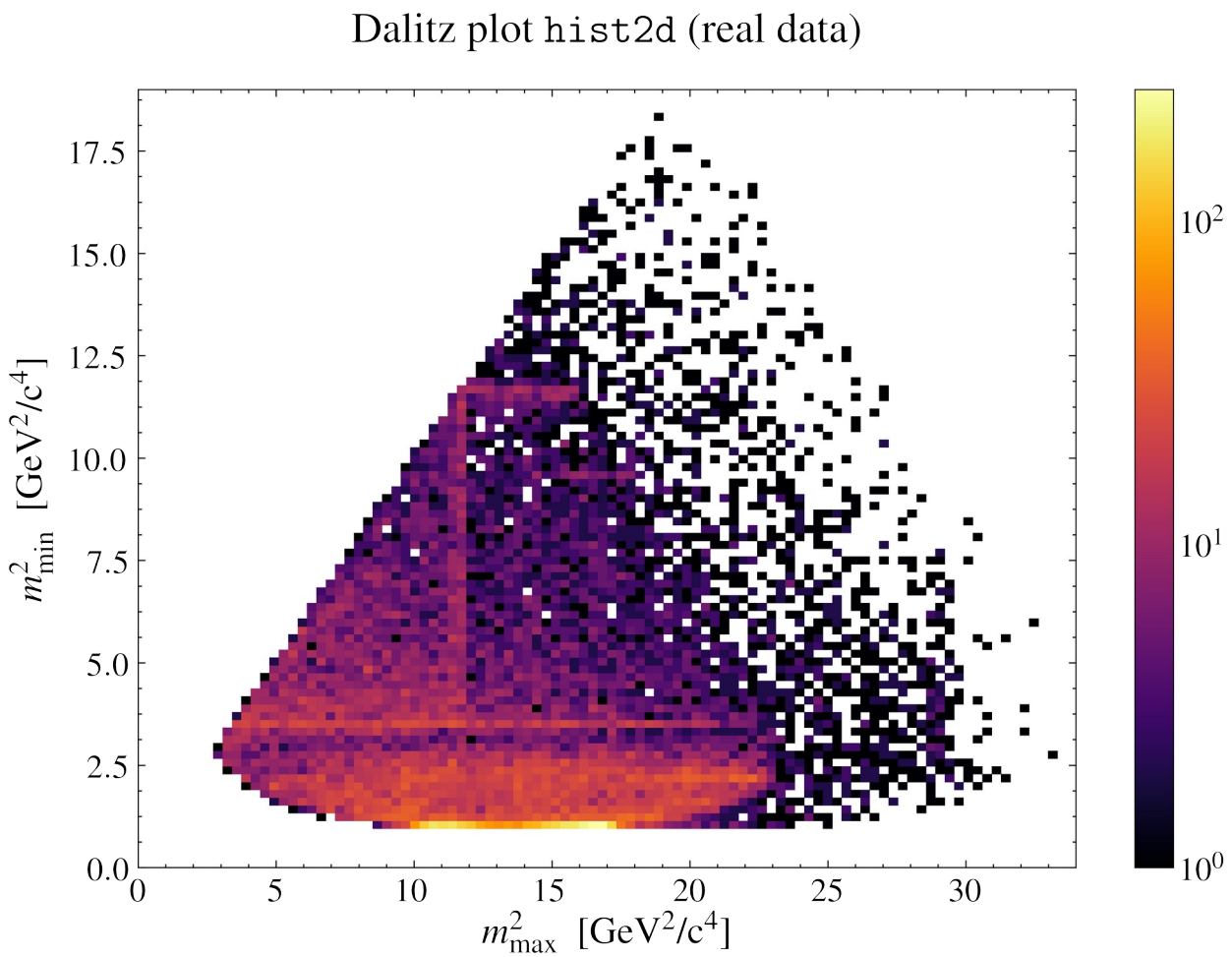
The visibility of the resonances in the Dalitz plot can be improved further. The resonances R01 and R03 are both composed of the same particle types, K+K-, and hence have the same distributions. It is useful to impose an ordering which distinguishes the resonances to access more information. Therefore, we sort the two resonances in a combination of kaons with the respectively higher mass R0High and one with the corresponding lower mass R0Low. You can now use the mass of these ordered resonances as your Dalitz plot variables, thus effectively "folding" your Dalitz plot so that one axis always has a higher value than the other. The total energy range is now reduced, while still remaining with the same statistics. This leads to a higher event density and therefore much clearer structures in the Dalitz plots.

```
# make a new Dalitz plot with a mass ordering of the axes
R_min = np.min(np.matrix([B_M_data_12,B_M_data_13]), axis=0).getA1()
R_max = np.max(np.matrix([B_M_data_12,B_M_data_13]), axis=0).getA1()

x_max = np.int(np.max(R_max**2))+1
y_max = np.int(np.max(R_min**2))+1
```

```
plt.figure()

plt.scatter(R_max**2,R_min**2,
            s=4,
            c='navy',
            marker='o',
            cmap='viridis',
            alpha=0.5,
            edgecolors='none')

plt.title('Ordered Dalitz plot (real data)',
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'dalitz_ordered'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



```
plt.figure()

plt.hist2d(R_max**2,R_min**2,
           bins=100,
           range=[[0, x_max], [0, y_max]],
           cmap=mpl.cm.inferno,
           norm=matplotlib.colors.LogNorm())

plt.colorbar()

plt.title('Dalitz plot $\mathtt{hist2d}$ (real data)',
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'hist2d_ordered'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



Hint

Sorting - You can find the lower/higher mass by elementwise comparison of the determined masses R01 and R03. The functions `numpy.min(a,b)` and `numpy.max(a,b)` from the `numpy` library can be helpful. Both functions perform an elementwise comparison between the arrays `a` and `b` and return one array filled by either the individual min/max element.

Binned Dalitz plot

In addition, the representation of the Dalitz plots can be improved by using a binning scheme. Therefore, you can use the `hist2d()` function to create a 2D histogram.

```
# plot a binned Dalitz Plot
# use colorbar() to make a legend for your plot at the side

# already done above
```

Two body resonances

You can now use your Dalitz plot to identify the intermediate resonances. These resonances create lines with higher event densities in the Dalitz plots. You can use PDG page to identify the observed resonances. The tables give the masses and widths of the particles and their decay modes. You are looking for mesons with the masses corresponding to where you see the bands and that decay into K+K-.

Searching for local matter anti-matter differences

Aims:

- Observe matter antimatter differences (CP violation) in regions of the Dalitz plots of the B+ and B- mesons
- Produce plots to display the CP violation in certain regions

In a section above you searched for global CP violation. You probably did not find a result with very high significance.

CP violation arises from interference between different decay chains with different intermediate resonances to a common final state. Therefore, the strength and the sign of the CP violation might vary in different kinematic regions. Hence, we can use the same equation that we already used for the global CP violation

$$A = \frac{N^{+¿} - N^{-}}{N^{+¿} + N^{-}} ¿$$

But we will use it in different kinematic regions independently.

Removing charm resonances

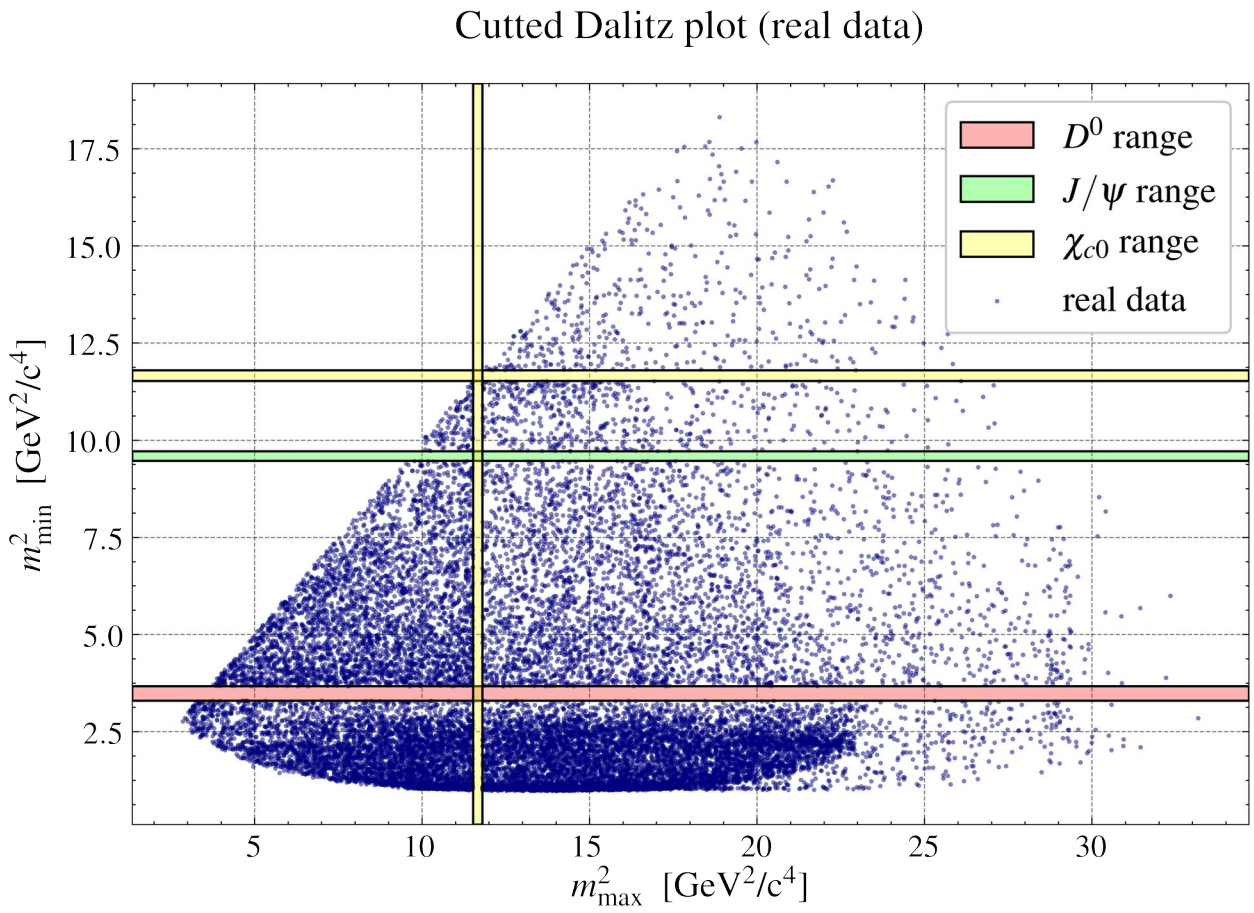
The analysis performed in this exercise is to study the CP violation in charmless B meson decays to kaons. "Charmless" means that the decay does not proceed through a charm quark. However, the most frequent decay of the B mesons occur through the *b* quark decaying into a *c* quark. The majority of these events can be removed by rejecting the events that are proceeding through a D0 meson (which contains the charm quark).

In the previous section you were most likely able to already identify a D0 meson resonance in the Dalitz plots. As a next step you should reject all events in this kinematic region to surpress the charm resonance contribution. You could add this step to your already existing preselection. By simply re-executing the previous analysis steps, you could observe the effects this change has on the Dalitz plots.

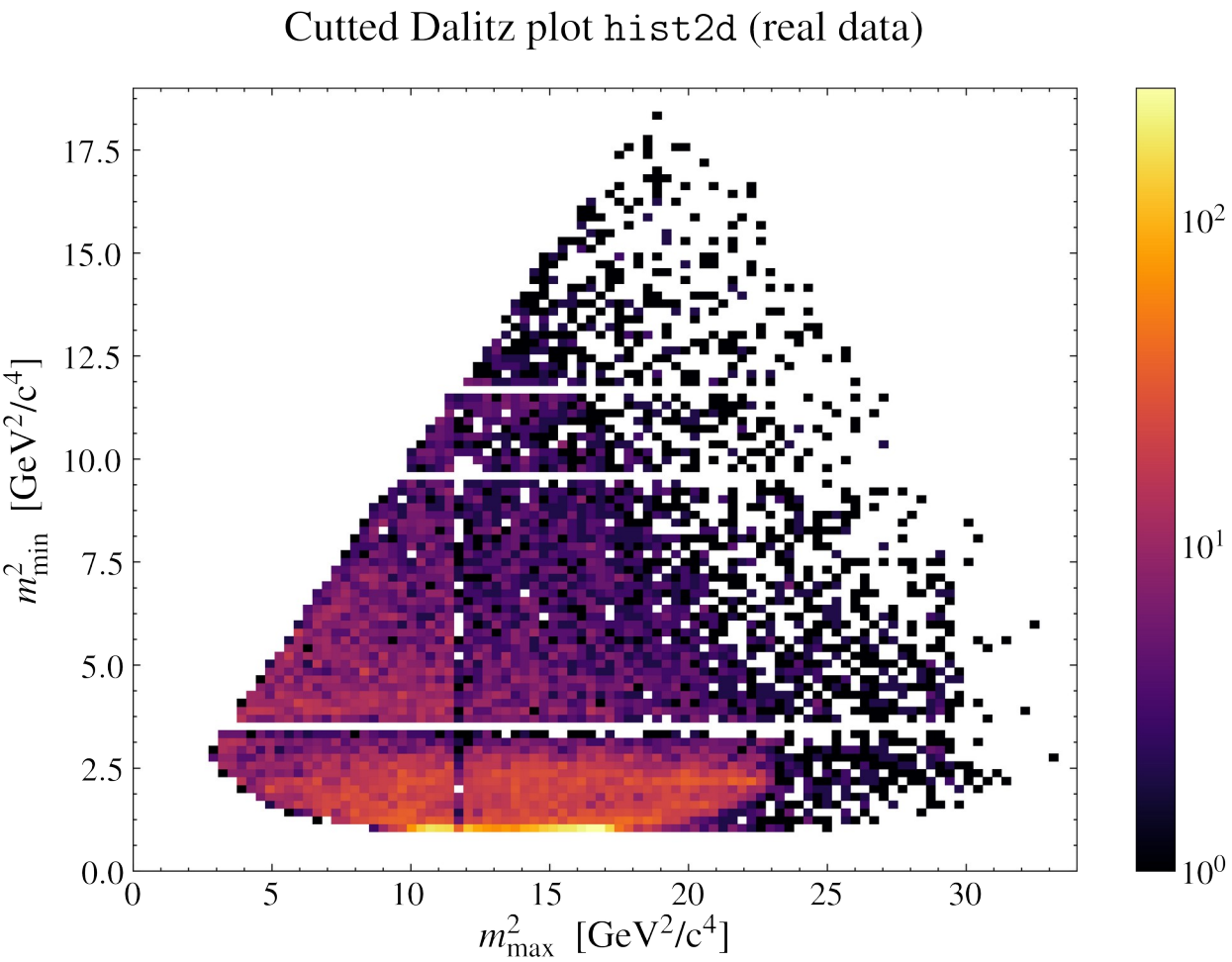
```
new_data = pd.DataFrame({'B_M_data_12': B_M_data_12,
                         'B_M_data_13': B_M_data_13})
new_data['R_min'] = R_min
new_data['R_max'] = R_max

new_data['H1_Charge'] = real_data['H1_Charge'][mask]
new_data['H2_Charge'] = real_data['H2_Charge'][mask]
new_data['H3_Charge'] = real_data['H2_Charge'][mask]
new_data['Charge'] = (new_data['H1_Charge']+
                      new_data['H2_Charge']+
                      new_data['H3_Charge'])

new_data['B_M_data'] = B_M_data

# remove D^0, JPSI and CHIc0 resonance
D_M_PDG = 1.86484  # [GeV/c^2]
cut_D = 0.05  # 0.05 [GeV/c^2] = 50 [MeV/c^2]

JPSI_M_PDG = 3.09690
cut_JPSI = 0.02

CHIc0_M_PDG = 3.41471
cut_CHIc0 = 0.02

for i in range(len(R_max)):
    if (((R_min[i]>D_M_PDG-cut_D) and
         (R_min[i]<D_M_PDG+cut_D)
        ) or
        ((R_min[i]>JPSI_M_PDG-cut_JPSI) and
         (R_min[i]<JPSI_M_PDG+cut_JPSI)
        ) or
        ((R_min[i]>CHIc0_M_PDG-cut_CHIc0) and
         (R_min[i]<CHIc0_M_PDG+cut_CHIc0)
        ) or
        ((R_max[i]>CHIc0_M_PDG-cut_CHIc0) and
         (R_max[i]<CHIc0_M_PDG+cut_CHIc0)
```

```
        )):
        new_data = new_data.drop([i])

new_data = new_data.reset_index(drop=True)

x_max = np.int(np.max(new_data['R_max']**2))+1
y_max = np.int(np.max(new_data['R_min']**2))+1

D0_int = [(D_M_PDG-cut_D)**2, (D_M_PDG+cut_D)**2]
JPSI_int = [(JPSI_M_PDG-cut_JPSI)**2, (JPSI_M_PDG+cut_JPSI)**2]
CHIc0_int = [(CHIc0_M_PDG-cut_CHIc0)**2, (CHIc0_M_PDG+cut_CHIc0)**2]

##---------------------------##

plt.figure()

plt.scatter(new_data['R_max']**2,new_data['R_min']**2,
            s=4,
            c='navy',
            marker='o',
            cmap='viridis',
            alpha=0.5,
            edgecolors='none',
            label='real data')

plt.axhspan(*D0_int,
            facecolor=(1,0,0,0.3),
            edgecolor=(0,0,0,1.0),
            lw=1,
            ls='solid',
            label='$D^{0}$ range')

plt.axhspan(*JPSI_int,
            facecolor=(0,1,0,0.3),
            edgecolor=(0,0,0,1.0),
            lw=1,
            ls='solid',
            label='$J/\psi$ range')

plt.axhspan(*CHIc0_int,
            facecolor=(1,1,0,0.3),
            edgecolor=(0,0,0,1.0),
            lw=1,
            ls='solid',
            label='$\chi_{c0}$ range')

plt.axvspan(*CHIc0_int,
            facecolor=(1,1,0,0.3),
            edgecolor=(0,0,0,1.0),
            lw=1,
```

```
            ls='solid')

plt.title('Cutted Dalitz plot (real data)',
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='best')
plt.tight_layout()
if save==1: plt.savefig(path+'dalitz_cutted'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()

print(f'D^0 range:    [{D0_int[0]:.1f} - {D0_int[1]:.1f}]    '+
      '[GeV^2/c^4]')
print(f'JPSI range:   [{JPSI_int[0]:.1f} - {JPSI_int[1]:.1f}]    '+
      '[GeV^2/c^4]')
print(f'CHIc0 range:  [{CHIc0_int[0]:.1f} - {CHIc0_int[1]:.1f}] '+
      '[GeV^2/c^4]')
```



```
D^0 range:   [3.3 - 3.7]   [GeV^2/c^4]
JPSI range:  [9.5 - 9.7]   [GeV^2/c^4]
CHIc0 range: [11.5 - 11.8] [GeV^2/c^4]

plt.figure()

plt.hist2d(new_data['R_max']**2,new_data['R_min']**2,
           bins=100,
           range=[[0, x_max], [0, y_max]],
           cmap=mpl.cm.inferno,
           norm=matplotlib.colors.LogNorm())

plt.colorbar()

plt.title('Cutted Dalitz plot $\mathtt{hist2d}$ (real data)',
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'hist2d_cutted'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



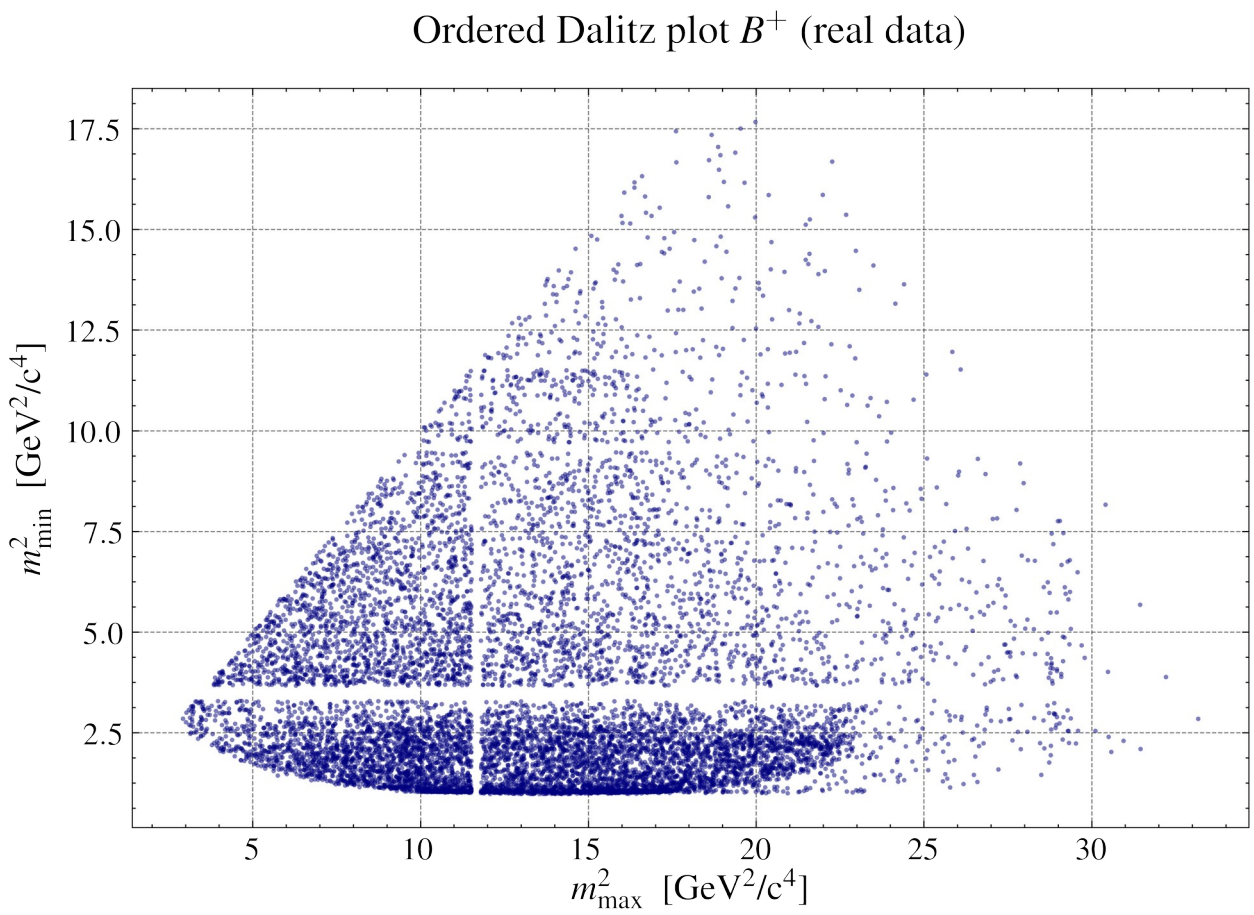
Comparing Dalitz plots

Make separate Dalitz plots for the B+ and the B- decays. Local CP violation will show up as asymmetry between both plots.

In order that the statistical error on the asymmetry in each bin is not overly large, the bins need to contain a reasonable number of entries. Therefore, a coarser binning is needed compared to the Dalitz plots that you used to search for the resonances. A suitable initial bin width should be in the order of a few GeV^2/c^4.

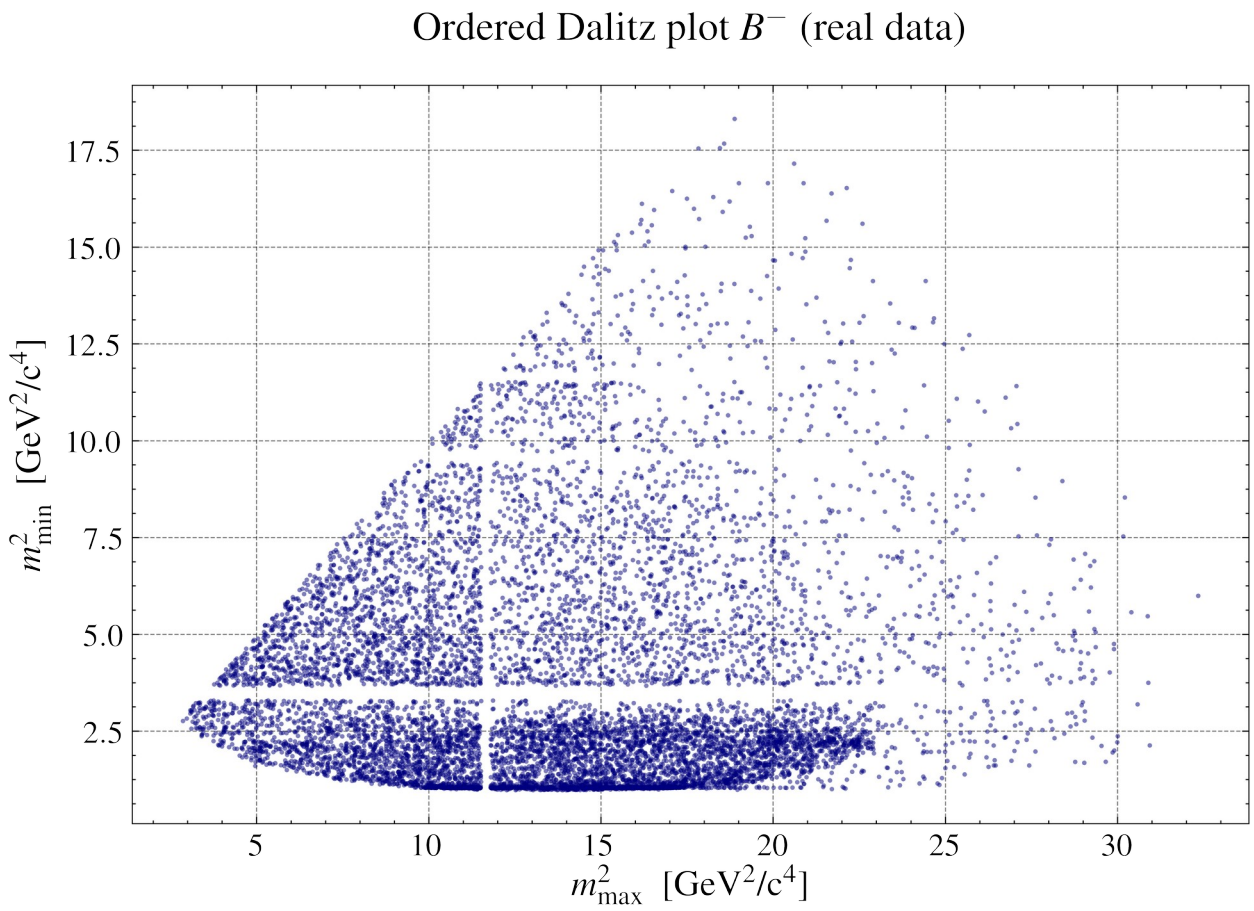
```
# make a Dalitz plot for the B+ events
plt.figure()

plt.scatter(new_data['R_max'][new_data['Charge']==1]**2,
            new_data['R_min'][new_data['Charge']==1]**2,
            s=4,
            c='navy',
            marker='o',
            cmap='viridis',
            alpha=0.5,
```

```
            edgecolors='none')

plt.title('Ordered Dalitz plot $B^{+}$ (real data)',
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'dalitz_orderedBplus'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



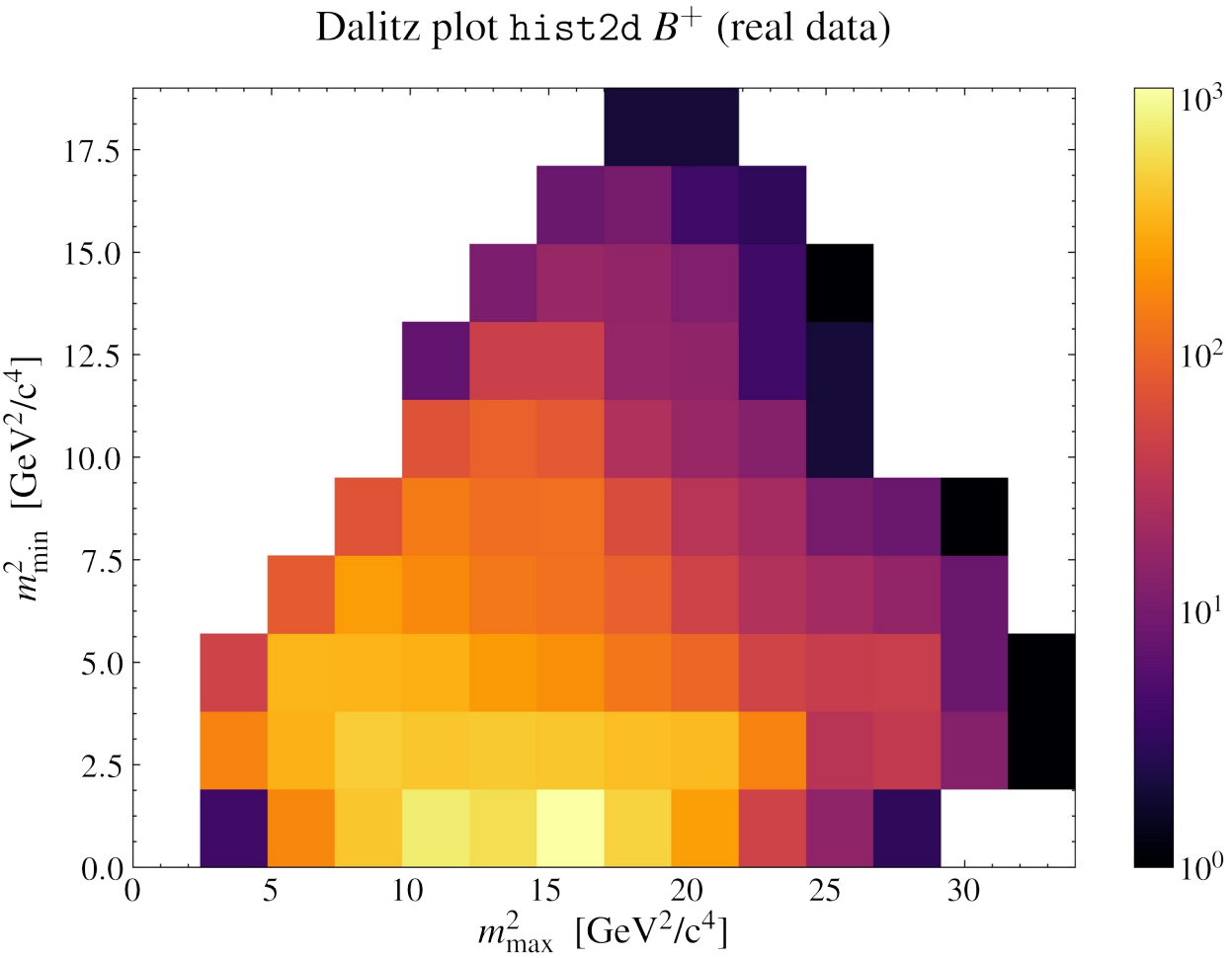

```
# make a Dalitz plot for the B+ events
plt.figure()

plt.scatter(new_data['R_max'][new_data['Charge']==-1]**2,
            new_data['R_min'][new_data['Charge']==-1]**2,
            s=4,
            c='navy',
            marker='o',
            cmap='viridis',
```

```
            alpha=0.5,
            edgecolors='none')

plt.title('Ordered Dalitz plot $B^{-}$ (real data)',
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'dalitz_orderedBminus'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```



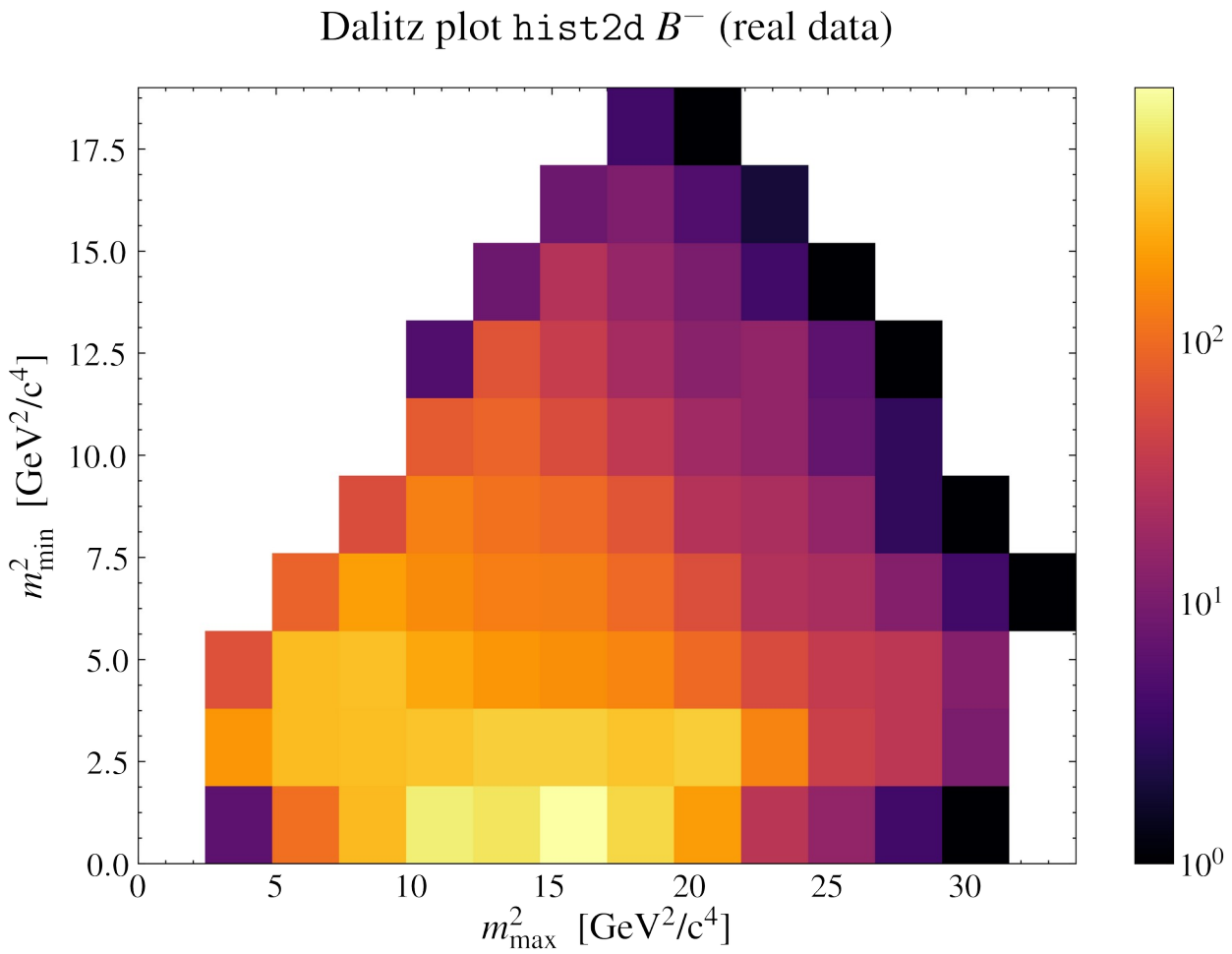

```
# use same binning for both hist2d B^{+} and b^{-}
x_bin = 14  # 14
y_bin = 10  # 10

x_maxPlus = (np.int
             (np.max(new_data['R_max'][new_data['Charge']==1]**2))+1)
y_maxPlus = (np.int
             (np.max(new_data['R_min'][new_data['Charge']==1]**2))+1)
```

```
x_maxMinus = (np.int
              (np.max(new_data['R_max'][new_data['Charge']==-1]**2))
+1)
y_maxMinus = (np.int
              (np.max(new_data['R_min'][new_data['Charge']==-1]**2))
+1)

x_max = max(x_maxPlus, x_maxMinus)
y_max = max(y_maxPlus, y_maxMinus)

plt.figure()

(hist2d_B_Plus,
 _,
 _,
 _) = plt.hist2d(new_data['R_max'][new_data['Charge']==1]**2,
                 new_data['R_min'][new_data['Charge']==1]**2,
                 bins=[x_bin,y_bin],
                 range=[[0, x_max], [0, y_max]],
                 cmap=mpl.cm.inferno,
                 norm=matplotlib.colors.LogNorm())

plt.colorbar()

plt.title('Dalitz plot $\mathtt{hist2d}$ $B^{+}$ (real data)',
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'hist2d_orderedBplus'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()

print(f'Bin area = {((x_max/x_bin)*(y_max/y_bin)):.4f} [GeV^2/c^4]')
```




```
Bin area = 4.6143 [GeV^2/c^4]

plt.figure()

(hist2d_B_Minus,
 xedges,
 yedges,
 _) = plt.hist2d(new_data['R_max'][new_data['Charge']==-1]**2,
                 new_data['R_min'][new_data['Charge']==-1]**2,
                 bins=[x_bin,y_bin],
                 range=[[0, x_max], [0, y_max]],
                 cmap=mpl.cm.inferno,
                 norm=matplotlib.colors.LogNorm())

plt.colorbar()

plt.title('Dalitz plot $\mathtt{hist2d}$ $B^{-}$ (real data)',
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
```

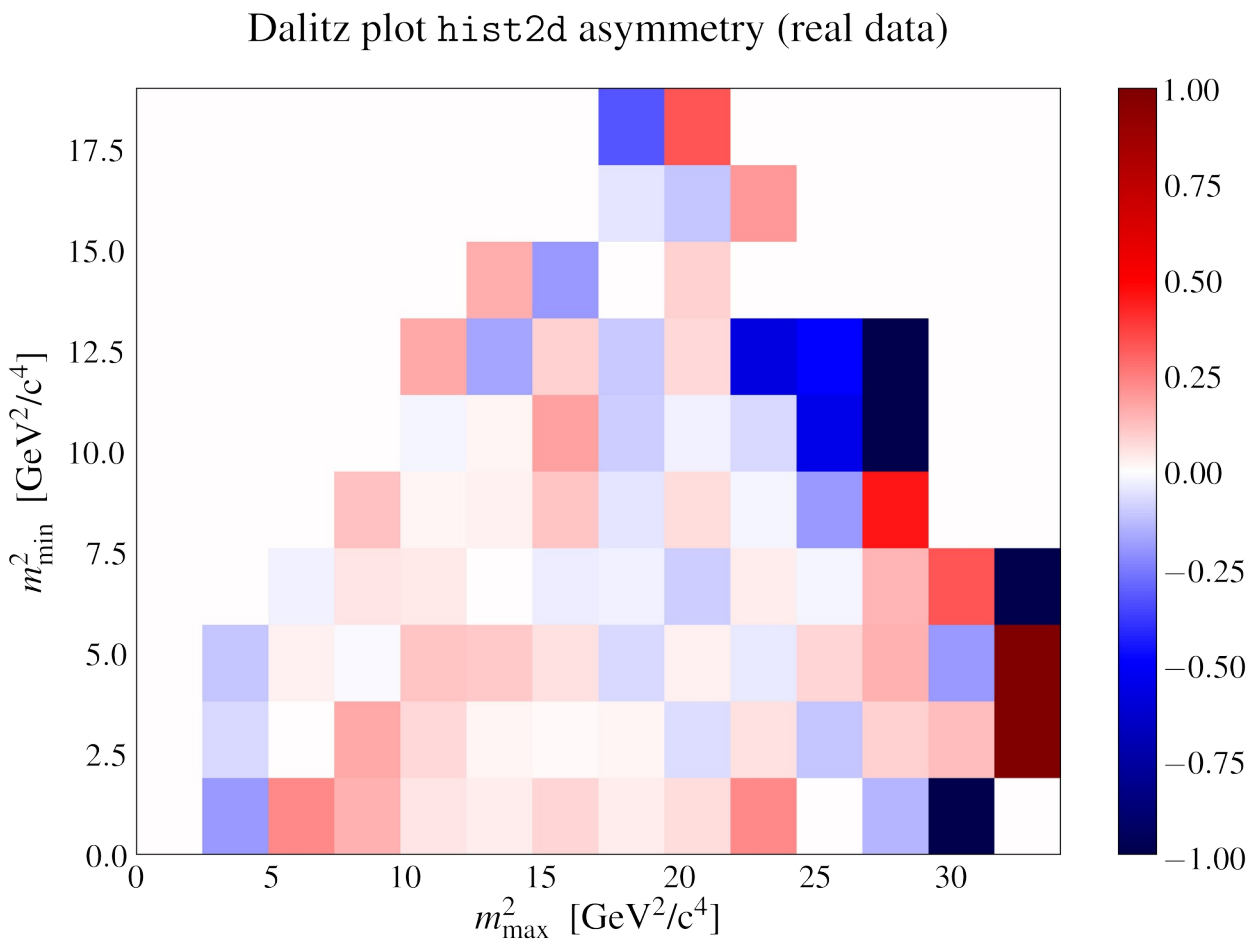
```
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'hist2d_orderedBminus'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()

print(f'Bin area = {((x_max/x_bin)*(y_max/y_bin)):.4f} [GeV^2/c^4]')
```




```
Bin area = 4.6143 [GeV^2/c^4]
```

```
# make a plot showing the asymmetry between these two Daltz plots
# i.e. calculate the asymmetry between each bin of the B+ and B-
# Dalitz plots and show the result in another 2D plot
A = (hist2d_B_Plus-hist2d_B_Minus)/(hist2d_B_Plus+hist2d_B_Minus)
A[np.isnan(A)] = 0

##---------------------------##

plt.figure()
```

```
plt.pcolor(xedges, yedges,
           A.T,
           cmap='seismic')

plt.colorbar()

plt.title('Dalitz plot $\mathtt{hist2d}$ asymmetry (real data)',
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'hist2d_asymmetry'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```

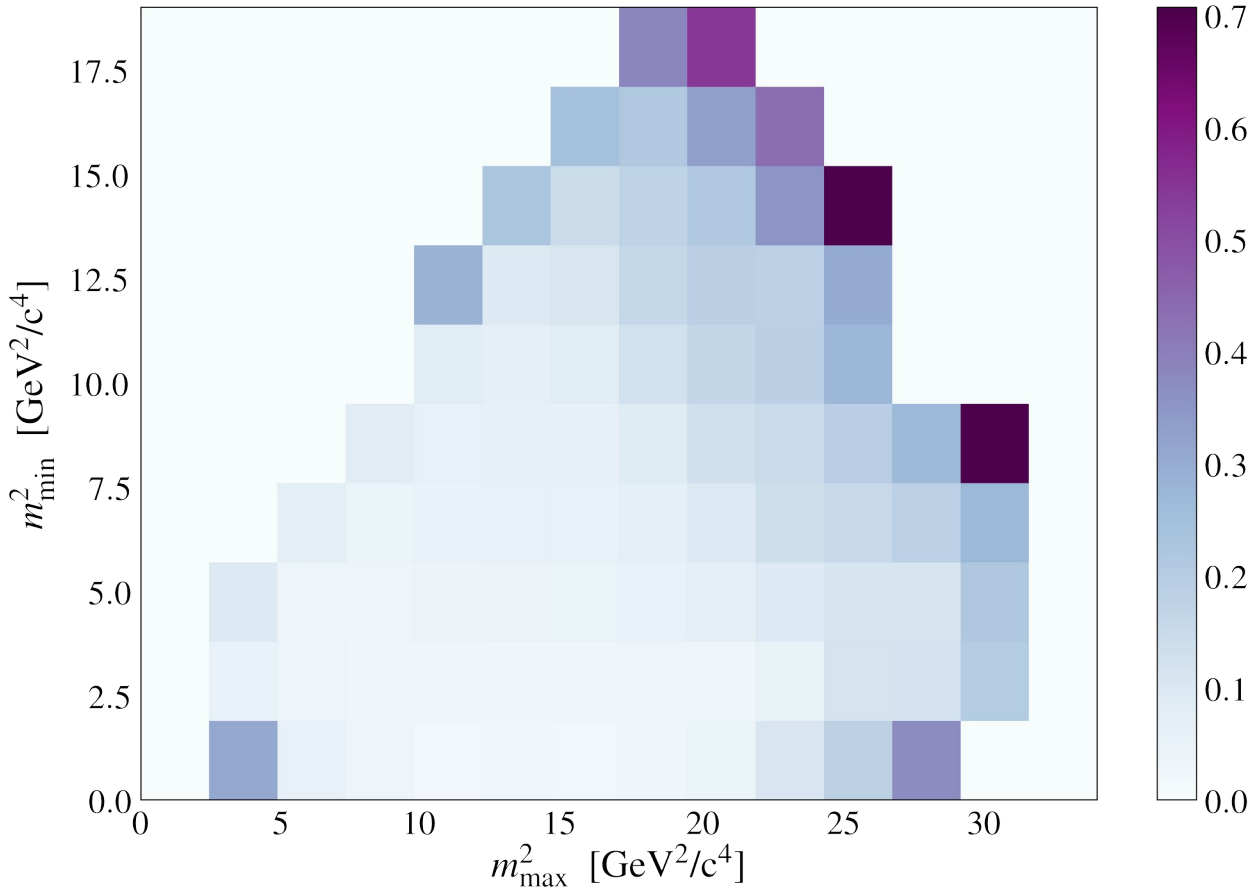


Observing a large asymmetry in some regions of the plot does not necessarily mean you have observed CP violation. If there are only very few events in that region of the plot the uncertainty on that large asymmetry may be large, too. Hence, the value may still be compatible with zero.

You can calculate the statistical uncertainty on the asymmetry, for each bin of the plot, using the same formulas as you used in the global asymmetry section. You can then make a plot showing the uncertainty on the asymmetry.

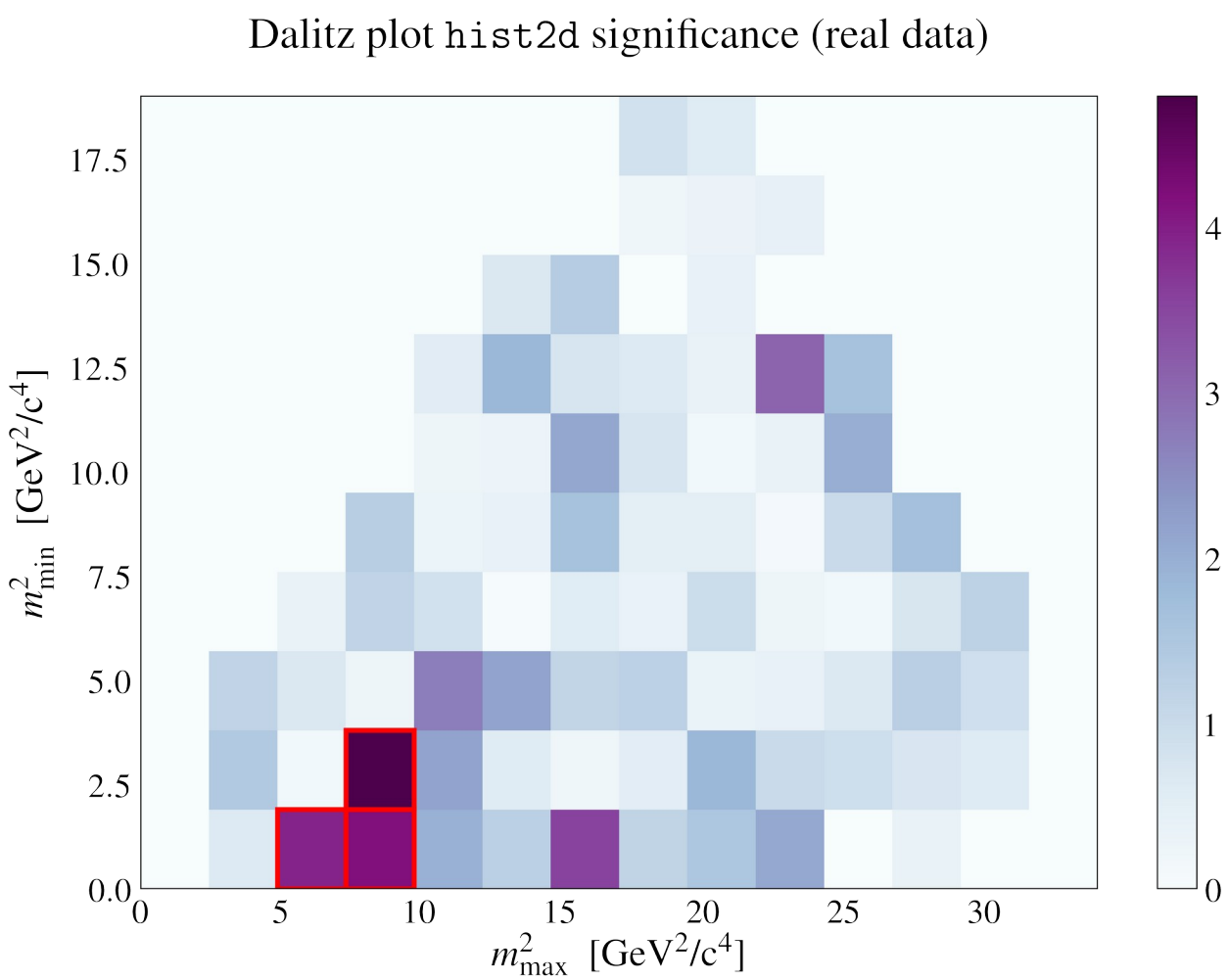
By dividing the plot showing the asymmetry by the plot showing the statistical uncertainty you can then obtain the significance of the asymmetry in each bin. Plotting the significance provides a way to select regions with CP violation.

```
sig = np.sqrt((1-(A**2))/(hist2d_B_Plus+hist2d_B_Minus))
sig[np.isinf(sig)] = 0
sig[np.isnan(sig)] = 0

s = np.abs(A/sig)
s[np.isinf(s)] = 0
s[np.isnan(s)] = 0

# make a plot showing the uncertainty on the asymmetry
plt.figure()

plt.pcolor(xedges, yedges,
           sig.T,
           cmap='BuPu')

plt.colorbar()

plt.title('Dalitz plot $\mathtt{hist2d}$ uncertainty (real data)',
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'hist2d_sigma'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```




```
x_step = xedges[4]-xedges[3]
y_step = yedges[4]-yedges[3]

# make a plot showing the statistical significance of the asymmetry
plt.figure()

plt.pcolor(xedges, yedges,
           s.T,
           cmap='BuPu')

plt.colorbar()

plt.gca().add_patch(Rectangle((4.85714286,0), x_step, y_step,
                              lw=2, edgecolor='r', facecolor='none'))
plt.gca().add_patch(Rectangle((7.28571429,0), x_step, y_step,
                              lw=2, edgecolor='r', facecolor='none'))
plt.gca().add_patch(Rectangle((7.28571429,1.9), x_step, y_step,
                              lw=2, edgecolor='r', facecolor='none'))

plt.title('Dalitz plot $\mathtt{hist2d}$ significance (real data)',
```

```
          pad=20)
plt.xlabel('$m_{\mathrm{max}}^2\;$ [GeV$^2$/c$^4$]')
plt.ylabel('$m_{\mathrm{min}}^2\;$ [GeV$^2$/c$^4$]')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.tight_layout()
if save==1: plt.savefig(path+'hist2d_signif'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```




```
print(f's.T[0,2] = {s.T[0,2]:.4f}')
print(f's.T[0,3] = {s.T[0,3]:.4f}')
print(f's.T[1,3] = {s.T[1,3]:.4f}')

s.T[0,2] = 3.9444
s.T[0,3] = 4.1711
s.T[1,3] = 4.7866

A = (
    ((hist2d_B_Plus[2,0]+
```

```
      hist2d_B_Plus[3,0]+
      hist2d_B_Plus[3,1])-
     (hist2d_B_Minus[2,0]+
      hist2d_B_Minus[3,0]+
      hist2d_B_Minus[3,1])
    )/
    ((hist2d_B_Plus[2,0]+
      hist2d_B_Plus[3,0]+
      hist2d_B_Plus[3,1])+
     (hist2d_B_Minus[2,0]+
      hist2d_B_Minus[3,0]+
      hist2d_B_Minus[3,1])
    ))
#A[np.isnan(A)] = 0

A_mean = A#.mean()
print('Asymmetry'+
      '\n'+
      f'A = {A_mean:.4f}\n')

sig = np.sqrt((1-(A**2))/
              ((hist2d_B_Plus[2,0]+
                hist2d_B_Plus[3,0]+
                hist2d_B_Plus[3,1])+
               (hist2d_B_Minus[2,0]+
                hist2d_B_Minus[3,0]+
                hist2d_B_Minus[3,1]
               ))
             )
#sig[np.isinf(sig)] = 0
#sig[np.isnan(sig)] = 0
sig_mean = sig#.mean()
print('Standard deviation'+
      '\n'+
      f'\u03C3\u1D00 = {sig_mean:.4f}\n')

signif = np.abs(A/sig)
#signif[np.isinf(signif)] = 0
#signif[np.isnan(signif)] = 0
signif_mean = signif#.mean()
print('Significance'+
      '\n'+
      f'S = {signif_mean:.4f}')
```

```
Asymmetry
A = 0.1701

Standard deviation
σA = 0.0231
```

```
Significance
S = 7.3700
```

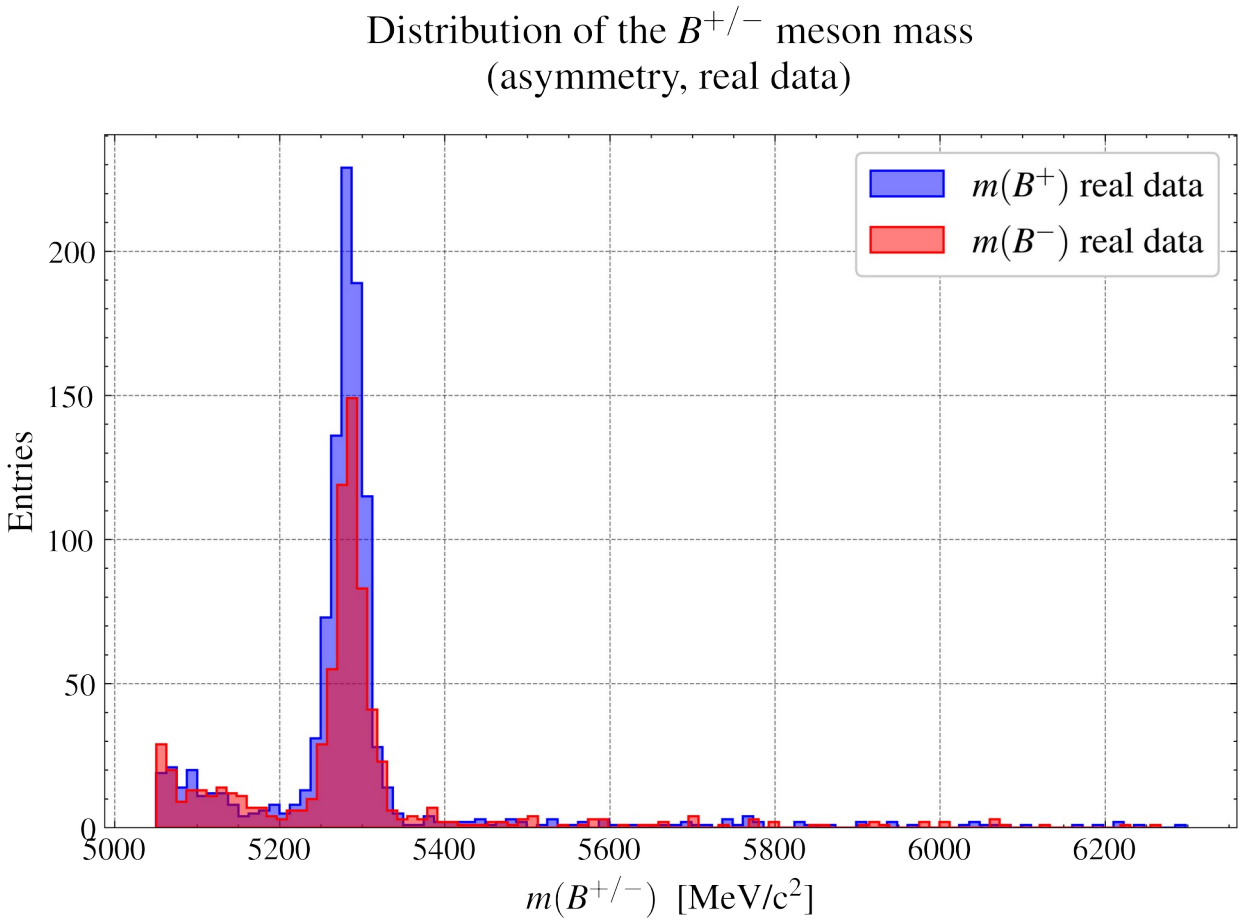
Observing CP violation

The previous investigations of the asymmetry and its significance should also now make it possible to select a region in the Dalitz plot that shows signs of significant CP violation of a certain size. This can be identified by a collection of contiguous bins with significant positive (or negative) asymmetry. It may help to vary the binning to select the optimal region.

You can now select the corresponding kinematic region in your data. Then, using a simple 1D histogram, plot the invariant mass distributions for the B+ and B- events, just as you did it for the global asymmetry. However, this time the events are exclusively in the kinematic region of interest. Now repeat the procedure to determine the CP violation for the selected data, count the respective event rate, calculate the asymmetry, its statistical uncertainty and the resulting significance of the measured CP violation.

```
maskPlus = ((new_data['Charge']==1) &
             (
                 ((4.85714286<new_data['R_max']**2) &
                  (new_data['R_max']**2<7.28571429) &
                  (0.<new_data['R_min']**2) &
                  (new_data['R_min']**2<1.9)
                 ) |
                 ((7.28571429<new_data['R_max']**2) &
                  (new_data['R_max']**2<9.71428571) &
                  (0.<new_data['R_min']**2) &
                  (new_data['R_min']**2<1.9)
                 ) |
                 ((7.28571429<new_data['R_max']**2) &
                  (new_data['R_max']**2<9.71428571) &
                  (1.9<new_data['R_min']**2) &
                  (new_data['R_min']**2<3.8)
                 )
             )
            )

maskMinus = ((new_data['Charge']==-1) &
              (
                  ((4.85714286<new_data['R_max']**2) &
                   (new_data['R_max']**2<7.28571429) &
                   (0.<new_data['R_min']**2) &
                   (new_data['R_min']**2<1.9)
                  ) |
                  ((7.28571429<new_data['R_max']**2) &
                   (new_data['R_max']**2<9.71428571) &
                   (0.<new_data['R_min']**2) &
                   (new_data['R_min']**2<1.9)
                  ) |
```

```
                ((7.28571429<new_data['R_max']**2) &
                 (new_data['R_max']**2<9.71428571) &
                 (1.9<new_data['R_min']**2**2) &
                 (new_data['R_min']**2<3.8)
                )
             )
            )

# make a plot showing the invariant mass of the B+ meson particles
# using events from a region of the Dalitz plot
# showing sizeable CP asymmetries
plt.figure()

plt.hist(new_data['B_M_data'][maskPlus],
         bins=100,
         #range=[5279, 5280],
         density=False,
         histtype='stepfilled',
         facecolor=(0,0,1,0.5),
         edgecolor=(0,0,1,1.0),
         lw=1,
         ls='solid',
         label='$m(B^{+})$ real data')

plt.hist(new_data['B_M_data'][maskMinus],
         bins=100,
         #range=[5279, 5280],
         density=False,
         histtype='stepfilled',
         facecolor=(1,0,0,0.5),
         edgecolor=(1,0,0,1.0),
         lw=1,
         ls='solid',
         label='$m(B^{-})$ real data')

plt.title('Distribution of the $B^{+/-}$ meson mass'
          '\n'
          '(asymmetry, real data)',
          pad=20)
plt.xlabel(r'$m(B^{+/-})\;$ [MeV/c$^2$]')
plt.ylabel('Entries')
plt.ticklabel_format(style='sci', axis='both', scilimits=[-4,4])
plt.legend(loc='best')
plt.tight_layout()
if save==1: plt.savefig(path+'B_M_asym'+format,
                        dpi=image_save_dpi, bbox_inches='tight')
plt.show()
```




```
# make a plot showing the invariant mass of the B- meson particles
using events from the same region

# already done above
```

Congratulations! You should now have succesfully observed significant evidence for CP Violation. Your Dalitz plots should show, that the decay widths of particles and anti-particles differ in different kinematic regions. You may wish to comapre your results with those published by the LHCb collaboration in this Paper.

Further analyses

The dataset used in your analysis contains an inclusive set of charged B meson decays into three charged tracks recorded at the LHCb experiment. Hence, this dataset has been used for further publicated measurements (you can fin some here and here).

There are further possibilities to extend this analysis. The following two possibilities will be allobrated in more details: Additional elements that you could add to your analysis of B+ → K+K+K− Similar analyses on this dataset

Adding extra sophistication

Systematic Uncertainties

In this analysis you considered the statistical uncertainty on the result. This occurs as a result of having only a limited number of events. In addition there are systematic uncertainties, these arise from biases in your measurement. Here we discuss three possible sources of those for this analysis. Production asymmetry. The LHC is a proton-proton collider and hence the initial state of the collision is not matter antimatter symmetric. Consequently B+ and B- mesons may not be produced at exactly the same rates. This small production asymmetry it is estimated to be approximately 1%. It can also be measured from the data, as discussed in the LHCb paper. Detection asymmetry. The LHCb detector could be more efficient for detecting either the B+ or the B- final states. This is because the positive and negative kaons will be bent by the magnet indifferent directions in the detector. If the efficiency of the detector is higher in one region than in another this will lead to higher efficiencies for K+ or K- and hence for B+ or B-. For this reason the magnetic field of the LHCb detector is regularly reversed. You used data in this analysis in which the magnetic field was both up and down and hence the effect will (partially) cancel. By comparing results for the two magnet polarities separately you can check the size of this effect. When loading the data above both polarities were combined, you can instead load them independently to measure the difference between the two datasets. Analysis technique. The analysis technique you have used may bias the result. A major simplification we made in the analysis above was to neglect 'background' events. We imposed a selection to select a sample of predominantly signal events but have not accounted for the effect of the residual background events.

Using mass sidebands

One source of 'background' events arises from random combinations of tracks in events that happen to fake the 'signal' characteristics. These events will not peak in the mass distribution at the mass of the B meson but will have a rather smoothly varying distribution. Looking at the number and distribution of events away from the mass peak can allow you to estimate the number of background events under the mass peak.

Fitting distributions

The next level of sophistication in the analysis requires fitting the distributions of events that are observed in the B mass distribution in order to estimate the yield of signal events and background events. You can see how this is done in the LHCb paper on the analysis. Fitting can be performed using for example the CERN root framework, the zfit or the iminuit libaries.

Further analyses

The LHCb papers using the data set, that you are using, analysed four decay channels of the charged B mesons. You can perform any of these analyses. $B^+ \to K^+K^+K^-$ (and anti-particle equivalent). This is the analysis you have performed here. It has the lowest background of the four channels and hence the approximation we made of neglecting the background events will give the least bias to this channel. $B^+ \to \pi^+\pi^+\pi^-$ (and anti-particle equivalent). In this analysis the final state is three charged pions. The level of background events compared to the signal is

significantly higher as pions are the most commonly produced particle at the LHC. Hence, a method of estimating the background level should be added to complete this analysis. $B^+ \to K^+\pi^+\pi^-$ (and anti-particle equivalent). In this analysis the final state is a mixture of one kaon and two pions. This means that the analysis needs to determine in each event which track is the best candidate kaon and apply selection cuts appropriately to select the events. $B^+ \to \pi^+K^+K^-$ (and anti-particle equivalent). This channel has a higher level of background compared to the signal.