International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Exploring the Labyrinth of Intertwined Theories: Probing Gravitons and Dark Matter through the Decay Mode of Graviton-> ZZ using ATLAS Open Data

Saarthak Jain

Grade 12, Modern School, Vasant Vihar, Delhi
Email: *saarthak3007[at]gmail.com*

Abstract: ***This article delves into the intricacies of the Standard Model of Particle Physics, a fundamental theory in particle physics describing the 17 known fundamental particles and their interactions. It highlights the models successes in explaining a wide range of experimental data, including the behavior of matter and energy at high energies, while acknowledging its limitations, such as the inability to explain dark matter, dark energy, and the matter-antimatter asymmetry. The article explores the role of the Large Hadron Collider LHC in testing the Standard Model through the observation of particle behaviors and interactions. It also touches on the structure of atoms, the discovery of quarks, and the four fundamental forces: gravity, electromagnetic, strong, and weak forces. Additionally, the piece discusses the Higgs fields role in imparting mass to particles, the theoretical existence of gravitons and gravitinos, and the use of particle accelerators to recreate conditions of the early universe, providing insights into its initial moments.***

Keywords: Standard Model, Particle Physics, Large Hadron Collider, Higgs Field, Fundamental Forces

1. Introduction

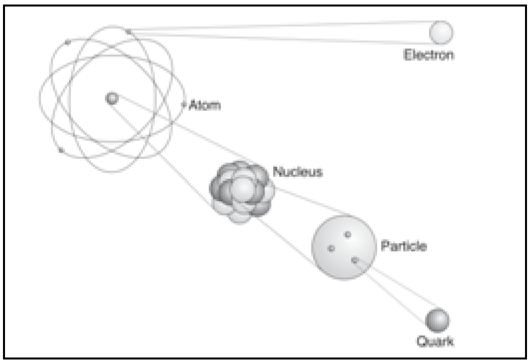
Over the past several decades, the Standard Model has been the prevailing theory in the field of particle physics. [2] The Standard Model of Particle Physics describes the 17 known fundamental particles, their interactions, and a detailed set of predictions for how they should behave and interact. [3] The Standard Model has been incredibly successful in explaining a wide range of experimental data, but it is not without its shortcomings. For example, it does not explain dark matter or dark energy [9], which makes up the vast majority of the universe. It also does not explain why there is more matter than antimatter in the universe. These are just some of the mysteries that physicists hope to solve by going beyond the Standard Model.

Physicists at the LHC have compared the predictions of the Standard Model with their data sets in countless ways. By precisely measuring the ways in which each of the known particles is created, decays, and otherwise interacts, we can test the validity of the Standard Model. [2] To date, the Standard Model has been very successful in explaining the behavior of matter and energy at high energies. It has passed all experimental tests to date, and it appears to provide a very good description of our universe at energies as high as thousands of GeV. [2] Consequently, the Standard Model provides us with a accurate description of how our universe behaved when it was only a trillionth of a second old and at a temperature of nearly 1017 degrees. [2]

		Mass	Electric Charge	Color
Leptons	Electron	$0.0054\ m_{proton}$	Yes	No
	Muon	$0.11\ m_{proton}$		
	Tau	$1.9\ m_{proton}$		
	Electron Neutrino	$< 10^{-10}\ m_{proton}$	No	No
	Muon Neutrino	$< 10^{-10}\ m_{proton}$		
	Tau Neutrino	$< 10^{-10}\ m_{proton}$		
Quarks	Up	$0.002\ m_{proton}$	Yes	Yes
	Down	$0.005\ m_{proton}$		
	Strange	$0.10\ m_{proton}$		
	Charm	$1.4\ m_{proton}$		
	Bottom	$4.5\ m_{proton}$		
	Top	$185\ m_{proton}$		
Bosons	Photon	0	No	No
	Gluon	0	No	Yes
	W Boson	$86\ m_{proton}$	Yes	No
	Z Boson	$97\ m_{proton}$	No	No
	Higgs Boson	$133\ m_{proton}$	No	No

The seventeen fundamental forms of matter and energy that make up the Standard Model of Particle Physics.

Each of the seventeen kinds of matter and energy specified by the Standard Model, as well as a wide range of other particles, can be produced in the collisions seen at the LHC. [4]

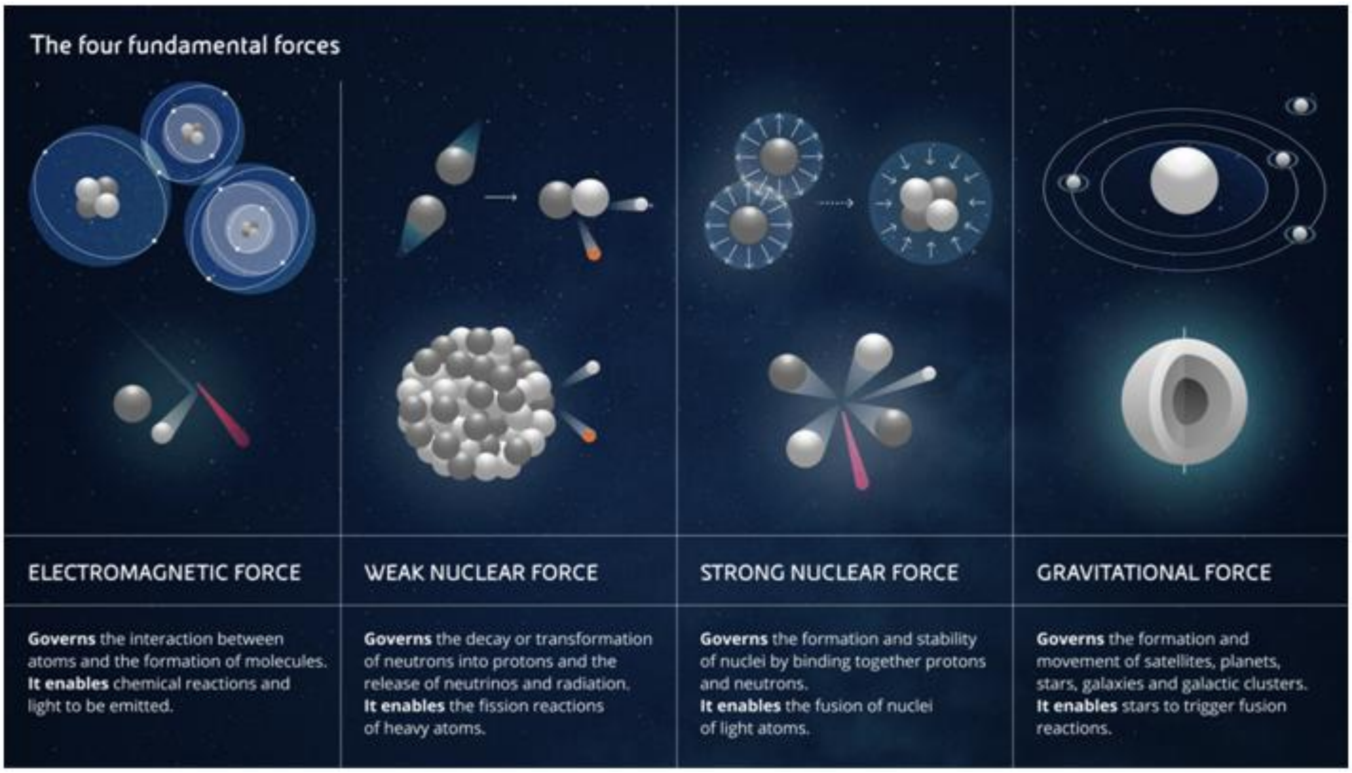


2. Background

But at first, at the heart of matter lies the atom. Atoms are composed of a central nucleus surrounded by a cloud of electrons. The nucleus contains protons and neutrons. [5] Electrons exist in energy levels called electron shells, and their behavior determines the properties of different elements. The nucleus, made up of protons and neutrons, is held together by a strong nuclear force. [1]

Magnify a neutron or proton a thousand times and you will discern that they too have a rich internal structure. [6] A neutron or proton, like a swarm of bees, appears as a dark spot from afar but is revealed to be a buzzing cloud of energy upon closer inspection. [1] On a low-powered image they appear like simple spots, but when viewed with a high-resolution microscope, they are found to be clusters of smaller particles called quarks. There are six types of quarks: up, down, strange, charm, bottom, and top. [7] The key evidence for their existence came from a series of inelastic electron-nucleon scattering experiments conducted between 1967 and 1973 at the Stanford Linear Accelerator Center. [8] The strong nuclear force, one of the four fundamental forces, plays a crucial role in holding quarks together within protons and neutrons.

If the electrons and quarks are like the letters, then there are also analogues of the grammar: the rules that glue the letters into words, sentences, and literature. For the universe, this glue is what we call the fundamental forces. There are four fundamental forces in nature, of which gravity is the most familiar. Gravity is the force that governs the motion of large objects, such as planets and stars. Matter is held together by the electromagnetic force, which is responsible for the attraction between electrons and protons in atoms, as well as the attraction between atoms in molecules. The strong force holds quarks together to form protons and neutrons, the building blocks of atomic nuclei. The weak force is responsible for radioactive decay, which is the process by which unstable atomic nuclei transform into other nuclei. It can also change one variety of particle into another, such as a proton into a neutron. [1]

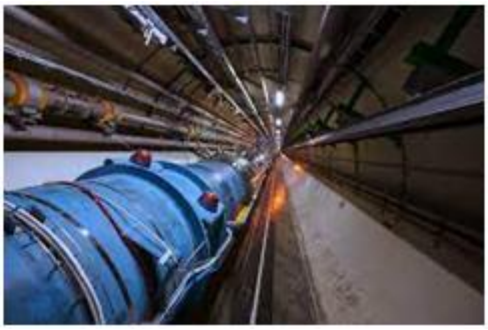


3. Discussion

The Standard Model of the fundamental particles and the forces that act among them explains mass by proposing that it is due to a new field, named the Higgs field after Peter Higgs who in 1964 was one of the first to recognize this theoretical possibility. [10] The Higgs field also permeates all of space. Were there no Higgs field, according to the theory, the fundamental particles would have no mass. What we recognize as mass is, in part, the effect of the interaction between particles and the Higgs field. Photons do not interact with the Higgs field and so are massless; the W and Z bosons do interact and thereby acquire their large masses. The building blocks of matter, the quarks and leptons, are also presumed to gain their masses by interacting with the Higgs Field. [11] Just as electromagnetic fields produce the quantum bundles known as photons, the Higgs field should manifest itself in Higgs bosons. In Higgs' original theory, there was only one type of Higgs boson, but if supersymmetry is correct, there should be a family of such particles. [1]

Some theorists suggest that a particle called the “graviton” is associated with gravity in the same way as the photon is associated with the electromagnetic force. [9] It is a spin-2 boson with mass zero. They have not been directly observed, but are predicted by general relativity. Gravitons are also thought to be responsible for the expansion of the universe. As the universe expands, the space between objects increases. This causes the gravitons between these objects to stretch, which weakens the gravitational force between them. This is thought to be one of the main causes of the accelerating expansion of the universe. The hypothetical graviton, the carrier of gravity, is predicted to have a partner, the gravitino. [12]

To understand what our universe was like in its earliest moments, we rely on machines that recreate the physical conditions of the distant past. [2] Using particle accelerators, we study how matter and other forms of energy behave at the highest of temperatures—those that were found throughout our universe as early as a trillionth of a second after the Big Bang. Since we cannot travel back in time or otherwise directly witness the first instants of our universe's history, we recreate a microcosm of the Big Bang here on Earth. The largest accelerator currently active is the Large Hadron Collider (LHC) near Geneva, Switzerland, operated by CERN. [13]

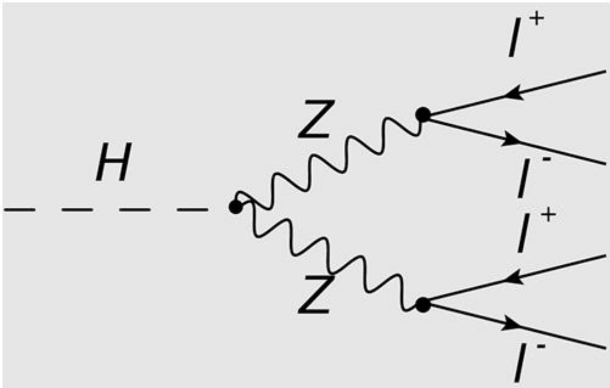


At the Large Hadron Collider, bunches of particles pass through each other 40 million times per second. Each time they cross, there are up to 25 collisions, resulting in nearly a billion collisions per second. The data collection rate of the detectors is equivalent to the information processing of 20 simultaneous telephone conversations by every man, woman, and child on Earth. [1]

Huge detectors are housed at the collision points. CMS (Compact Muon Solenoid) and ATLAS (A Toroidal LHC ApparatuS) explore the new energy region looking for all kinds of new effects - both expected and unexpected. [14]The ATLAS detector is five stories high (20 m) and yet able to measure particle tracks to a precision of 0.01 mm. The traditional design for contemporary particle detectors is followed by both ATLAS and CMS. The first is the aptly titled "inner tracker, " which accurately measures the locations of electrically charged particles to within a tenth of a millimeter, allowing computers to recreate their paths as they curve through the powerful magnetic fields. The next layer is a two-part calorimeter, designed to capture all the energy of many types of particle. The inner part is the electromagnetic calorimeter, which traps and records the energies of electrons and photons.

ATLAS Open Data offers accessible access to proton-proton collision data conducted at the Large Hadron Collider, primarily geared towards educational purposes. This initiative has been carefully developed through collaboration with students and educators. ATLAS Open Data provides a comprehensive and thorough repository of information by harnessing data obtained from the Large Hadron Collider at CERN, which is made readily accessible to the public. [15]

Utilizing the open access to proton-proton collision data at the Large Hadron Collider, the following code probes the new physics signal model where graviton goes to 2 ZZ bosons which further on decay to 4 Leptons.



Source Code:

```
import sys
!{sys.executable} -m pip install --upgrade --user pip
!{sys.executable} -m pip install -U numpy pandas uproot3 matplotlib --user
```

```
import uproot3 # for reading .root files
import pandas as pd # to store data as dataframe
import time # to measure time to analyse
import math # for mathematical functions such as square root
import numpy as np # for numerical calculations such as histogramming
import matplotlib.pyplot as plt # for plotting
from matplotlib.ticker import AutoMinorLocator,LogLocator,LogFormatterSciNotation # for minor ticks

import infofile # local file containing cross-sections, sums of weights, dataset IDs
```

```
#lumi = 0.5 # fb-1 # data_A only
#lumi = 1.9 # fb-1 # data_B only
#lumi = 2.9 # fb-1 # data_C only
#lumi = 4.7 # fb-1 # data_D only
lumi = 0.2 # fb-1 # data_A,data_B,data_C,data_D

fraction = 0.09 # reduce this is you want the code to run quicker

#tuple_path = "Input/4lep/" # local
tuple_path = "https://atlas-opendata.web.cern.ch/atlas-opendata/samples/2020/4lep/" # web address
```

```
samples = {

    'data': {
        'list' : ['data_A','data_B','data_C','data_D']
    },

    r'$Z,t\bar{t}$' : { # Z + ttbar
        'list' : ['Zee','Zmumu','ttbar_lep'],
        'color' : "#6b59d3" # purple
    },

    r'$t\bar{t}V$' : { # ttV
        'list' : ['ttW','ttee','ttmumu'], # ttW, ttZ(->ee), ttZ(->mm)
        'color' : "#f0f423" # yellow
    },

    'ZZ' : { # ZZ->llll
        'list' : ['llll'],
        'color' : "#ff0000" # red
    },

    'Graviton' : {
        'list' : ['RS_G_ZZ_llll_c10_m0500'], # mG = 500 GeV
        'color' : "#baff8d" # green
    },

}
```

```
def get_data_from_files():

    data = {} # define empty dictionary to hold dataframes
    for s in samples: # loop over samples
        print('Processing '+s+' samples') # print which sample
        frames = [] # define empty list to hold data
        for val in samples[s]['list']: # loop over each file
            if s == 'data': prefix = "Data/" # Data prefix
            else: # MC prefix
                prefix = "MC/mc_"+str(infofile.infos[val]["DSID"])+"."
            fileString = tuple_path+prefix+val+".4lep.root" # file name to open
            temp = read_file(fileString,val) # call the function read_file defined below
            frames.append(temp) # append dataframe returned from read_file to list of dataframes
        data[s] = pd.concat(frames) # dictionary entry is concatenated dataframes

    return data # return dictionary of dataframes
```

```
def calc_weight(xsec_weight, mcWeight, scaleFactor_PILEUP,
                scaleFactor_ELE, scaleFactor_MUON,
                scaleFactor_LepTRIGGER ):
    return xsec_weight*mcWeight*scaleFactor_PILEUP*scaleFactor_ELE*scaleFactor_MUON*scaleFactor_LepTRIGGER
```

```
def get_xsec_weight(sample):
    info = infofile.infos[sample] # open infofile
    xsec_weight = (lumi*1000*info["xsec"])/(info["sumw"]*info["red_eff"]) #*1000 to go from fb-1 to pb-1
    return xsec_weight # return cross-section weight
```

```
def calc_lep_pt_i(lep_pt,i):
    return lep_pt[i]/1000 # /1000 to go from MeV to GeV
```

```
def calc_mllll(lep_pt,lep_eta,lep_phi,lep_E):
    # first lepton is [0], 2nd lepton is [1] etc
    px_0 = lep_pt[0]*math.cos(lep_phi[0]) # x-component of lep[0] momentum
    py_0 = lep_pt[0]*math.sin(lep_phi[0]) # y-component of lep[0] momentum
    pz_0 = lep_pt[0]*math.sinh(lep_eta[0]) # z-component of lep[0] momentum
    px_1 = lep_pt[1]*math.cos(lep_phi[1]) # x-component of lep[1] momentum
    py_1 = lep_pt[1]*math.sin(lep_phi[1]) # y-component of lep[1] momentum
    pz_1 = lep_pt[1]*math.sinh(lep_eta[1]) # z-component of lep[1] momentum
    px_2 = lep_pt[2]*math.cos(lep_phi[2]) # x-component of lep[2] momentum
    py_2 = lep_pt[2]*math.sin(lep_phi[2]) # y-component of lep[2] momentum
    pz_2 = lep_pt[2]*math.sinh(lep_eta[2]) # z-component of lep[3] momentum
    px_3 = lep_pt[3]*math.cos(lep_phi[3]) # x-component of lep[3] momentum
    py_3 = lep_pt[3]*math.sin(lep_phi[3]) # y-component of lep[3] momentum
    pz_3 = lep_pt[3]*math.sinh(lep_eta[3]) # z-component of lep[3] momentum
    sumpx = px_0 + px_1 + px_2 + px_3 # x-component of 4-lepton momentum
    sumpy = py_0 + py_1 + py_2 + py_3 # y-component of 4-lepton momentum
    sumpz = pz_0 + pz_1 + pz_2 + pz_3 # z-component of 4-lepton momentum
    sumE = lep_E[0] + lep_E[1] + lep_E[2] + lep_E[3] # energy of 4-lepton system
    return math.sqrt(sumE**2 - sumpx**2 - sumpy**2 - sumpz**2)/1000 #/1000 to go from MeV to GeV
```

```
# cut on lepton charge
# paper: "selecting two pairs of isolated leptons, each of which is comprised of two leptons with the same flavour and opposite charge"
def cut_lep_charge(lep_charge):
# throw away when sum of lepton charges is not equal to 0
# first lepton is [0], 2nd lepton is [1] etc
    return lep_charge[0] + lep_charge[1] + lep_charge[2] + lep_charge[3] != 0

# cut on lepton type
# paper: "selecting two pairs of isolated leptons, each of which is comprised of two leptons with the same flavour and opposite charge"
def cut_lep_type(lep_type):
# for an electron lep_type is 11
# for a muon lep_type is 13
# throw away when none of eeee, mumumumu, eemumu
    sum_lep_type = lep_type[0] + lep_type[1] + lep_type[2] + lep_type[3]
    return (sum_lep_type != 44) and (sum_lep_type != 48) and (sum_lep_type != 52)
```

```
def read_file(path,sample):
    start = time.time() # start the clock
    print("\tProcessing: "+sample) # print which sample is being processed
    data_all = pd.DataFrame() # define empty pandas DataFrame to hold all data for this sample
    tree = uproot3.open(path)["mini"] # open the tree called mini
    numevents = uproot3.numentries(path, "mini") # number of events
    if 'data' not in sample: xsec_weight = get_xsec_weight(sample) # get cross-section weight
    for data in tree.iterate(['lep_pt','lep_eta','lep_phi',
                              'lep_E','lep_charge','lep_type',
                              # add more variables here if you make cuts on them
                              'mcWeight','scaleFactor_PILEUP',
                              'scaleFactor_ELE','scaleFactor_MUON',
                              'scaleFactor_LepTRIGGER'], # variables to calculate Monte Carlo weight
                             outputtype=pd.DataFrame, # choose output type as pandas DataFrame
                             entrystop=numevents*fraction): # process up to numevents*fraction

        nIn = len(data.index) # number of events in this batch

        if 'data' not in sample: # only do this for Monte Carlo simulation files
            # multiply all Monte Carlo weights and scale factors together to give total weight
            data['totalWeight'] = np.vectorize(calc_weight)(xsec_weight,
                                                            data.mcWeight,
                                                            data.scaleFactor_PILEUP,
                                                            data.scaleFactor_ELE,
                                                            data.scaleFactor_MUON,
                                                            data.scaleFactor_LepTRIGGER)

        # cut on lepton charge using the function cut_lep_charge defined above
        fail = data[ np.vectorize(cut_lep_charge)(data.lep_charge) ].index
        data.drop(fail, inplace=True)

        # cut on lepton type using the function cut_lep_type defined above
        fail = data[ np.vectorize(cut_lep_type)(data.lep_type) ].index
        data.drop(fail, inplace=True)

        # calculation of 4-lepton invariant mass using the function calc_mllll defined above
        data['mllll'] = np.vectorize(calc_mllll)(data.lep_pt,data.lep_eta,data.lep_phi,data.lep_E)

        # dataframe contents can be printed at any stage like this
        #print(data)

        # dataframe column can be printed at any stage like this
        print(data['lep_pt'])

        # return the individual lepton transverse momenta in GeV
        data['lep_pt_1'] = np.vectorize(calc_lep_pt_i)(data.lep_pt,1)
        data['lep_pt_2'] = np.vectorize(calc_lep_pt_i)(data.lep_pt,2)

        # multiple dataframe columns can be printed at any stage like this
        #print(data[['lep_pt','lep_eta']])

        nOut = len(data.index) # number of events passing cuts in this batch
        data_all = pd. concat ([data], ignore_index=True) # append dataframe from this batch to the dataframe for the whole sample
        elapsed = time.time() - start # time taken to process
        print("\t\t nIn: "+str(nIn)+",\t nOut: \t"+str(nOut)+"\t in "+str(round(elapsed,1))+"s") # events before and after

    return data_all # return dataframe containing events passing all cuts
```

```
start = time.time() # time at start of whole processing
data = get_data_from_files() # process all files
elapsed = time.time() - start # time after whole processing
print("Time taken: "+str(round(elapsed,1))+"s") # print total time taken to process every file
```

```
lep_pt_2 = { # dictionary containing plotting parameters for the lep_pt_2 histogram
    # change plotting parameters
    'bin_width':1, # width of each histogram bin
    'num_bins':13, # number of histogram bins
    'xrange_min':7, # minimum on x-axis
    'xlabel':r'$lep\_pt$[2] [GeV]', # x-axis label
}

lep_pt_1 = { # dictionary containing plotting parameters for the lep_pt_1 histogram
    # change plotting parameters
    'bin_width':1, # width of each histogram bin
    'num_bins':28, # number of histogram bins
    'xrange_min':7, # minimum on x-axis
    'xlabel':r'$lep\_pt$[1] [GeV]', # x-axis label
}

SoverB_hist_dict = {'lep_pt_2':lep_pt_2,'lep_pt_1':lep_pt_1}
# add a histogram here if you want it plotted
```

```
def plot_SoverB(data):

    signal = 'Graviton' # which sample is the signal

    # *******************
    # general definitions (shouldn't need to change)

    for x_variable,hist in SoverB_hist_dict.items(): # access the dictionary of histograms defined in the cell above

        h_bin_width = hist['bin_width'] # get the bin width defined in the cell above
        h_num_bins = hist['num_bins'] # get the number of bins defined in the cell above
        h_xrange_min = hist['xrange_min'] # get the x-range minimum defined in the cell above
        h_xlabel = hist['xlabel'] # get the x-axis label defined in the cell above

        bin_edges = [ h_xrange_min + x*h_bin_width for x in range(h_num_bins+1) ] # bin limits
        bin_centres = [ h_xrange_min+h_bin_width/2 + x*h_bin_width for x in range(h_num_bins) ] # bin centres

        signal_x = data[signal][x_variable] # histogram the signal

        mc_x = [] # define list to hold the Monte Carlo histogram entries

        for s in samples: # loop over samples
            if s not in ['data', signal]: # if not data nor signal
                mc_x = [*mc_x, *data[s][x_variable] ] # append to the list of Monte Carlo histogram entries



        # **************
        # Signal and background distributions
        # **************
        distributions_axes = plt.gca() # get current axes

        mc_heights = distributions_axes.hist(mc_x, bins=bin_edges, color='red',
                                             label='Total background',
                                             histtype='step', # lineplot that's unfilled
                                             density=True ) # normalize to form probability density
        signal_heights = distributions_axes.hist(signal_x, bins=bin_edges, color='blue',
                                                 label=signal,
                                                 histtype='step', # lineplot that's unfilled
                                                 density=True, # normalize to form probability density
                                                 linestyle='--' ) # dashed line

        distributions_axes.set_xlim( left=bin_edges[0], right=bin_edges[-1] ) # x-limits of the distributions axes
        distributions_axes.set_ylabel('Arbitrary units' ) # y-axis label for distributions axes
        distributions_axes.set_ylim( top=max(signal_heights[0])*1.3 ) # set y-axis limits
        plt.title('Signal and background '+x_variable+' distributions') # add title
        distributions_axes.legend() # draw the legend
        distributions_axes.set_xlabel( h_xlabel ) # x-axis label

        # Add text 'ATLAS Open Data' on plot
        plt.text(0.05, # x
                 0.93, # y
                 'ATLAS Open Data', # text
                 transform=distributions_axes.transAxes, # coordinate system used is that of distributions_axes
                 fontsize=13 )
        # Add text 'for education' on plot
        plt.text(0.05, # x
                 0.88, # y
                 'for education', # text
                 transform=distributions_axes.transAxes, # coordinate system used is that of distributions_axes
                 style='italic',
                 fontsize=8 )

        plt.show() # show the Signal and background distributions


        # **************
        # Signal to background ratio
        # **************
        plt.figure() # start new figure
        SoverB = [] # list to hold S/B values
        for cut_value in bin_edges: # loop over bins
            signal_weights_passing_cut = sum(data[signal][data[signal][x_variable]>cut_value].totalWeight)
            background_weights_passing_cut = 0 # start counter for background weights passing cut
            for s in samples: # loop over samples
                if s not in ['data', signal]: # if not data nor signal
                    background_weights_passing_cut += sum(data[s][data[s][x_variable]>cut_value].totalWeight)
            if background_weights_passing_cut!=0: # some background passes cut
                SoverB_value = signal_weights_passing_cut/background_weights_passing_cut
                SoverB_percent = 100*SoverB_value # multiply by 100 for percentage
                SoverB.append(SoverB_percent) # append to list of S/B values

        SoverB_axes = plt.gca() # get current axes
        SoverB_axes.plot( bin_edges[:len(SoverB)], SoverB ) # plot the data points
        SoverB_axes.set_xlim( left=bin_edges[0], right=bin_edges[-1] ) # set the x-limit of the main axes
        SoverB_axes.set_ylabel( 'S/B (%)' ) # write y-axis label for main axes
        plt.title('Signal to background ratio for different '+x_variable+' cut values', family='sans-serif')
        SoverB_axes.set_xlabel( h_xlabel ) # x-axis label

        plt.show() # show S/B plot

    return
```

```
plot_SoverB(data)
```

```
# define class to display 1 and 10 normally
class CustomTicker(LogFormatterSciNotation):
    def __call__(self, x, pos=None):
        if x not in [1,10]: # not 1 or 10
            return LogFormatterSciNotation.__call__(self,x, pos=None)
        else: # 1 or 10
            return "{x:g}".format(x=x) # standard notation
```

```python
def plot_data(data):

    xmin = 130 # GeV
    xmax = 1230 # GeV
    step_size = 55 # GeV

    bin_edges = np.arange(start=xmin, # The interval includes this value
                    stop=xmax+step_size, # The interval doesn't include this value
                    step=step_size ) # Spacing between values
    bin_centres = np.arange(start=xmin+step_size/2, # The interval includes this value
                        stop=xmax+step_size/2, # The interval doesn't include this value
                        step=step_size ) # Spacing between values

    data_x,_ = np.histogram(data['data']['mllll'],
                        bins=bin_edges ) # histogram the data
    data_x_errors = np.sqrt( data_x ) # statistical error on the data

    signal_x = data['Graviton']['mllll'] # histogram the signal
    signal_weights = data['Graviton'].totalWeight # get the weights of the signal events
    signal_color = samples['Graviton']['color'] # get the colour for the signal bar

    mc_x = [] # define list to hold the Monte Carlo histogram entries
    mc_weights = [] # define list to hold the Monte Carlo weights
    mc_colors = [] # define list to hold the colors of the Monte Carlo bars
    mc_labels = [] # define list to hold the legend labels of the Monte Carlo bars

    for s in samples: # loop over samples
        if s not in ['data', 'Graviton']: # if not data nor signal
            mc_x.append( data[s]['mllll'] ) # append to the list of Monte Carlo histogram entries
            mc_weights.append( data[s].totalWeight ) # append to the list of Monte Carlo weights
            mc_colors.append( samples[s]['color'] ) # append to the list of Monte Carlo bar colors
            mc_labels.append( s ) # append to the list of Monte Carlo legend labels



    # *************
    # Main plot
    # *************
    main_axes = plt.gca() # get current axes

    # plot the data points
    main_axes.errorbar(x=bin_centres, y=data_x, yerr=data_x_errors,
                    fmt='ko', # 'k' means black and 'o' is for circles
                    label='Data')

    # plot the Monte Carlo bars
    mc_heights = main_axes.hist(mc_x, bins=bin_edges,
                            weights=mc_weights, stacked=True,
                            color=mc_colors, label=mc_labels )

    mc_x_tot = mc_heights[0][-1] # stacked background MC y-axis value

    # calculate MC statistical uncertainty: sqrt(sum w^2)
    mc_x_err = np.sqrt(np.histogram(np.hstack(mc_x), bins=bin_edges, weights=np.hstack(mc_weights)**2)[0])

    # plot the signal bar
    main_axes.hist(signal_x, bins=bin_edges, bottom=mc_x_tot,
                weights=signal_weights, color=signal_color,
                label='Graviton')

    # plot the statistical uncertainty
    main_axes.bar(bin_centres, # x
               2*mc_x_err, # heights
               alpha=0.5, # half transparency
               bottom=mc_x_tot-mc_x_err, color='none',
               hatch="////", width=step_size, label='Stat. Unc.' )

    # set the x-limit of the main axes
    main_axes.set_xlim( left=xmin, right=xmax )

    # separation of x axis minor ticks
    main_axes.xaxis.set_minor_locator( AutoMinorLocator() )

    # set the axis tick parameters for the main axes
    main_axes.tick_params(which='both', # ticks on both x and y axes
                        direction='in', # Put ticks inside and outside the axes
                        top=True, # draw ticks on the top axis
                        right=True ) # draw ticks on right axis

    # x-axis label
    main_axes.set_xlabel(r'4-lepton invariant mass $\mathrm{m_{4l}}$ [GeV]',
                    fontsize=13, x=1, horizontalalignment='right' )

    # write y-axis label for main axes
    main_axes.set_ylabel('Events / '+str(step_size)+' GeV',
                     y=1, horizontalalignment='right')

    # add minor ticks on y-axis for main axes
    main_axes.yaxis.set_minor_locator( AutoMinorLocator() )

    main_axes.set_yscale('log') # set y-scale
    smallest_contribution = mc_heights[0][0] # get smallest contribution
    smallest_contribution.sort() # sort smallest contribution
    bottom = np.amax(data_x)/1000 # set bottom limit on y-axis
    top = np.amax(data_x)*100 # set top limit on y-axis
    main_axes.set_ylim( bottom=bottom, top=top ) # y-axis limits
    main_axes.yaxis.set_major_formatter( CustomTicker() )
    locmin = LogLocator(base=10.0, # log base 10
                    subs=(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9) ) # minor tick every 0.1
    main_axes.yaxis.set_minor_locator( locmin ) # set minor ticks

    # [illegible] plot
    plt.text(0.05, # x
```

```
plot_data(data)
```

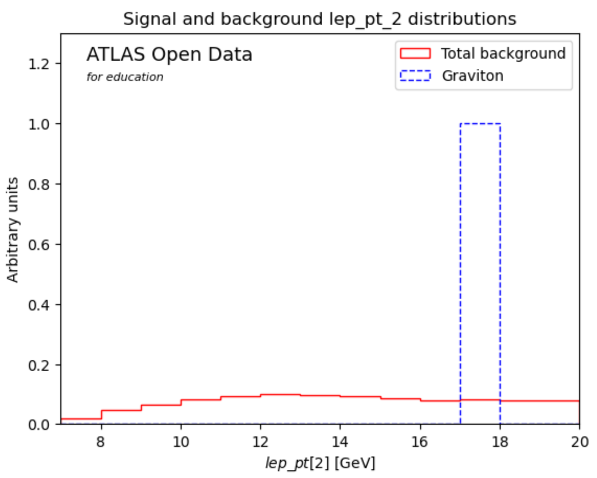
Resulting Graphs:

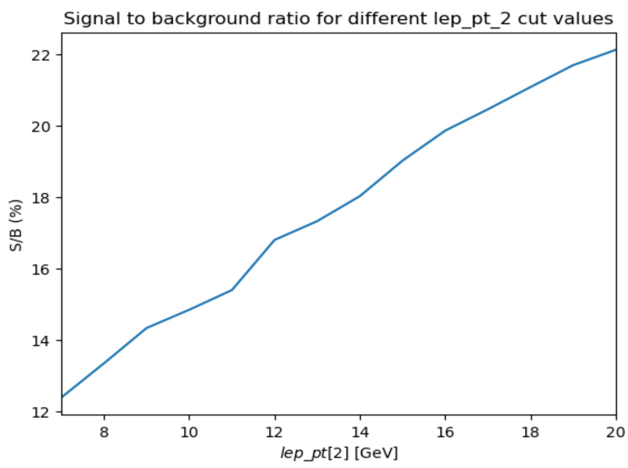
From the Following Source Code we obtain 5 graphs for physical processes of varying luminosities and a 0.09 fraction of event. First, We option a graph of signal and background of Lep_pt_2 Distribution, Secondly, we option a graph of signal and background of Lep_pt_1 Distribution, Thirdly, we option a graph of signal to background ratio for different Lep_pt_2 cut values, fourthly, we option a graph of signal to background ratio for different Lep_pt_1 cut values, and lastly, we obtain the graph of the 4 lepton invariant graph.

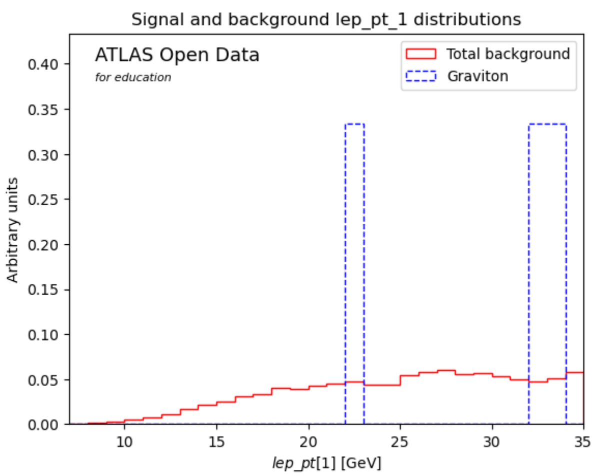
Instantaneous luminosity measures how tightly particles are packed into a given space, such as the LHC's proton beam. A higher luminosity means a greater likelihood particles will collide and result in a desired interaction. This can be achieved by packing more particles in the beam, or by focusing the beam more tightly. [16]

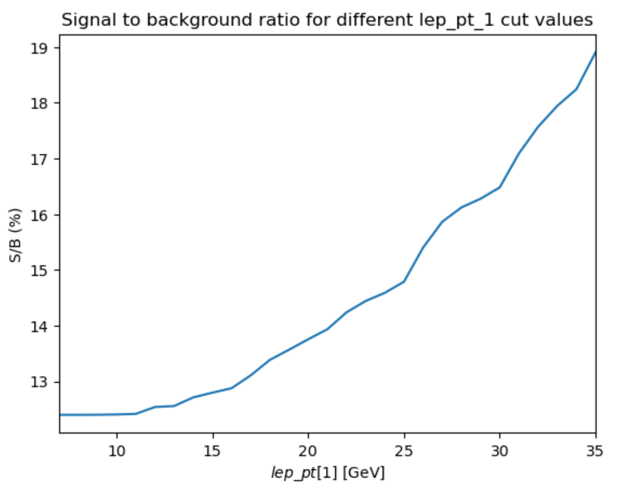
Lumi = 0.5

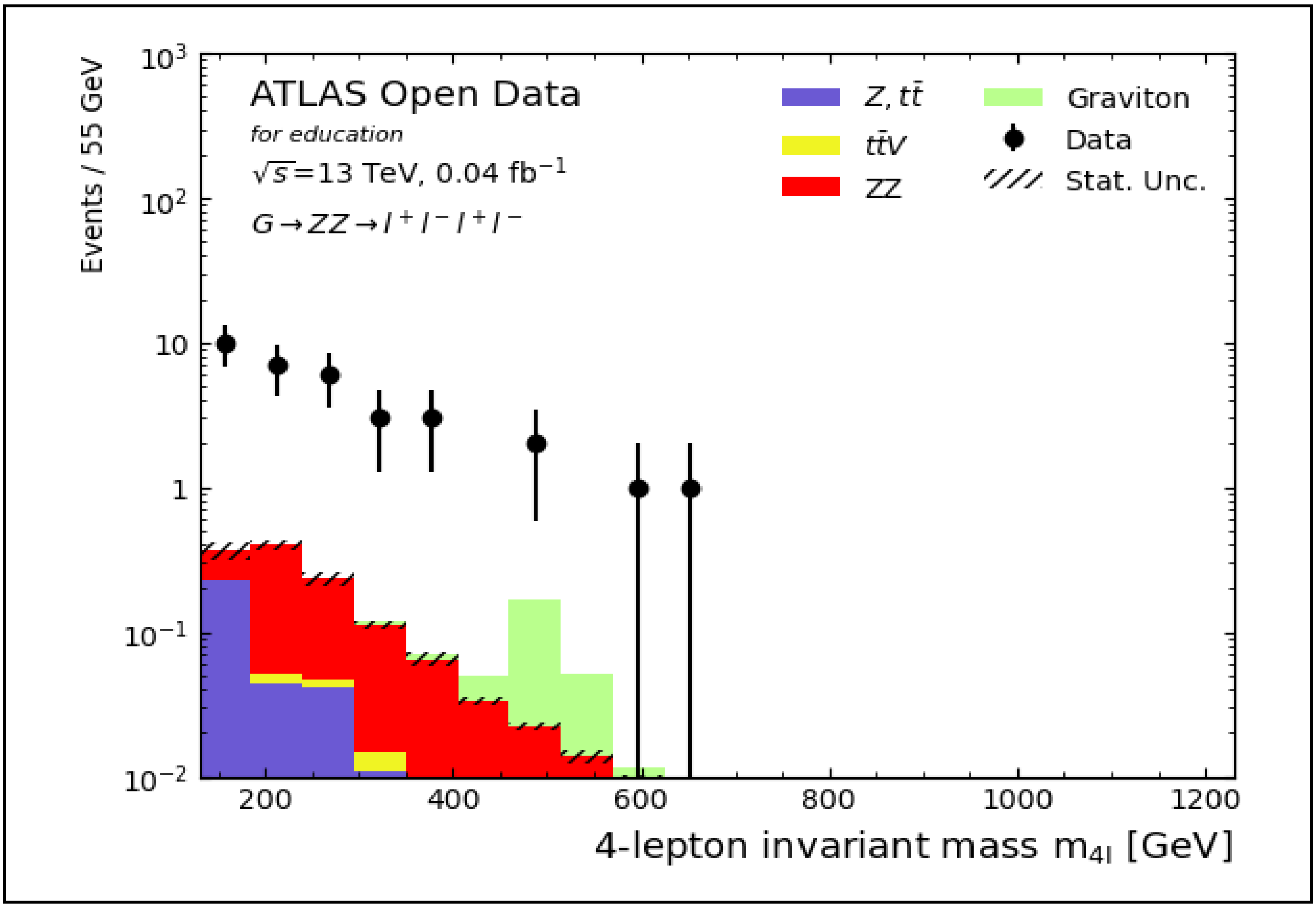
Fraction = 0.09

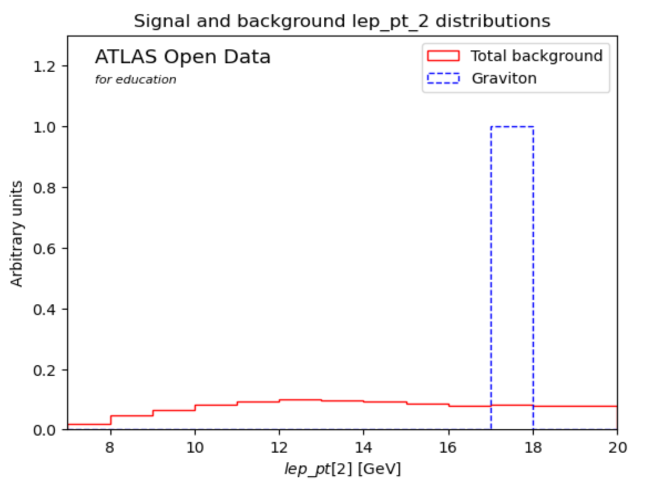
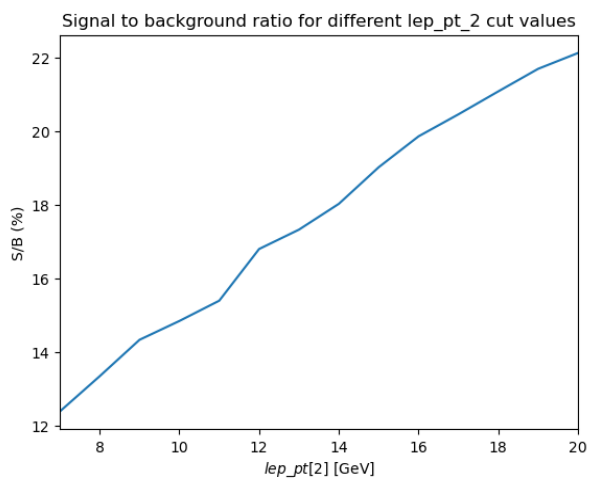


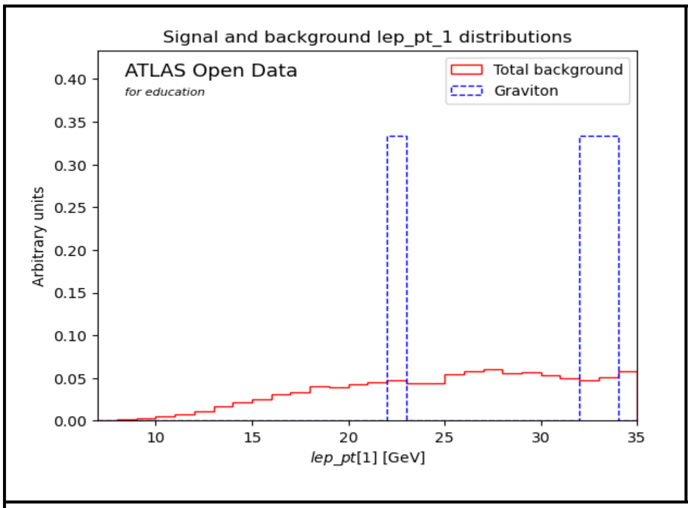


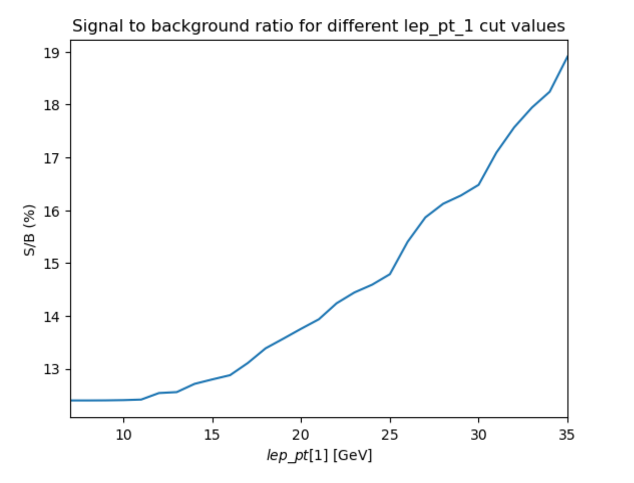


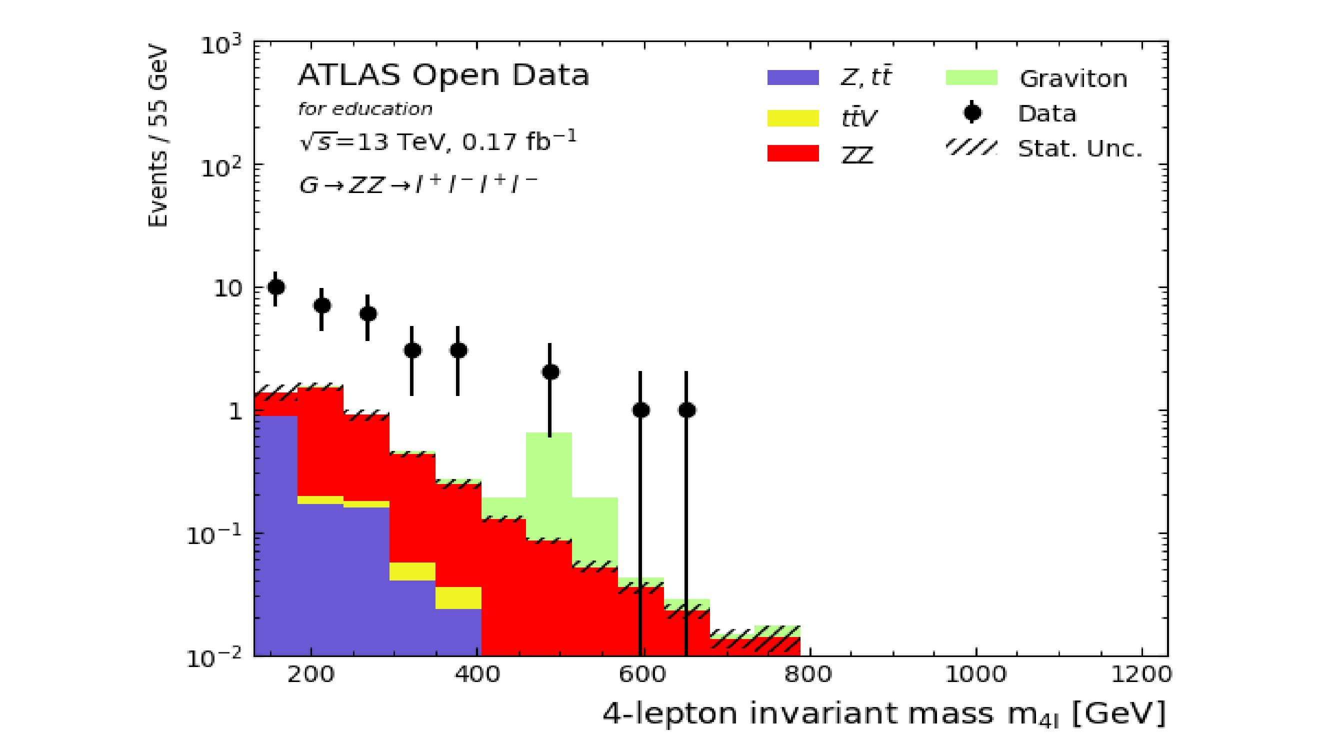




Lumi = 1.9	
Fraction = 0.09	
	

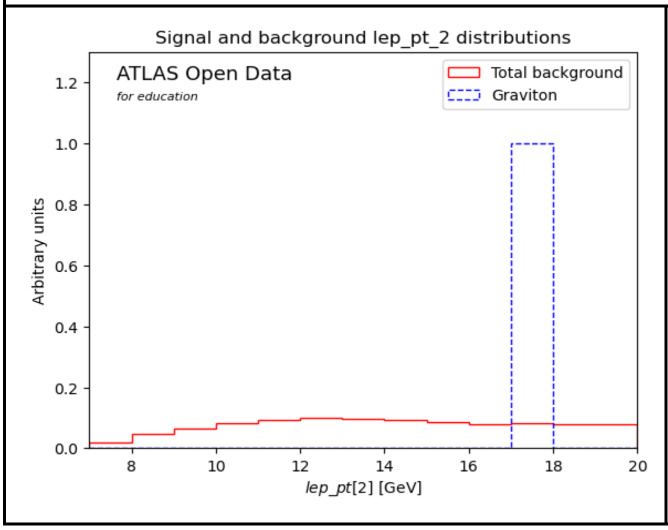


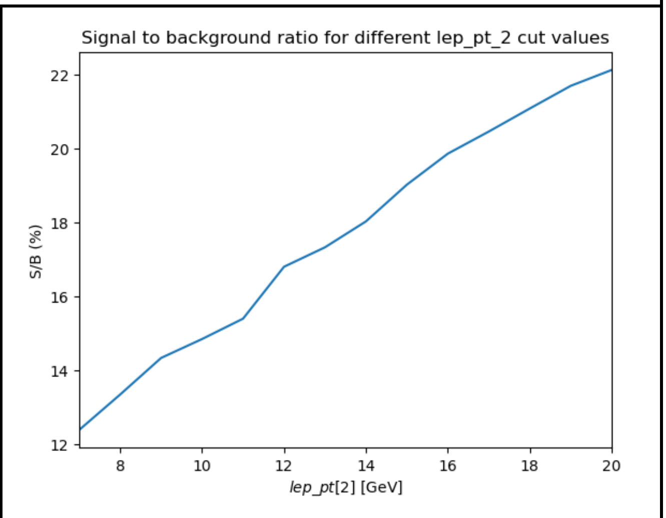


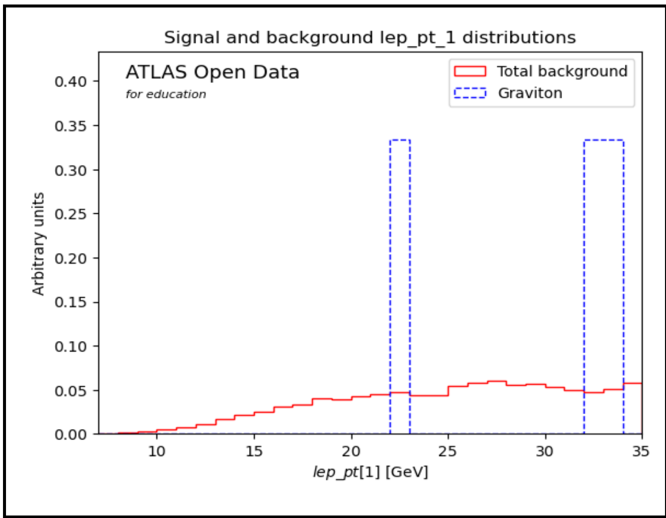


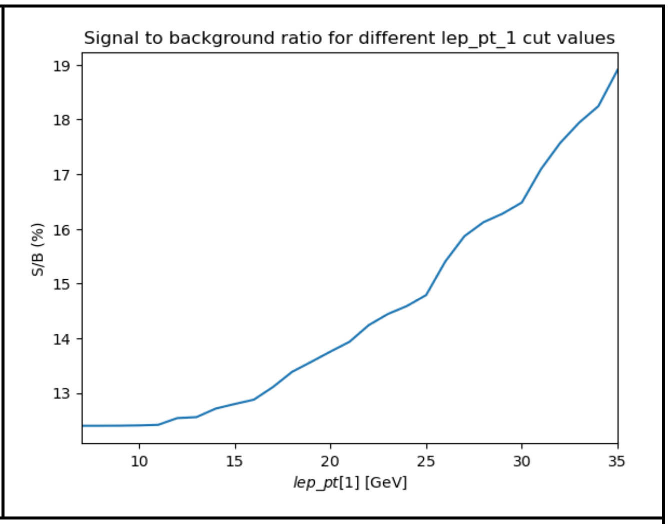
Lumi =2.9

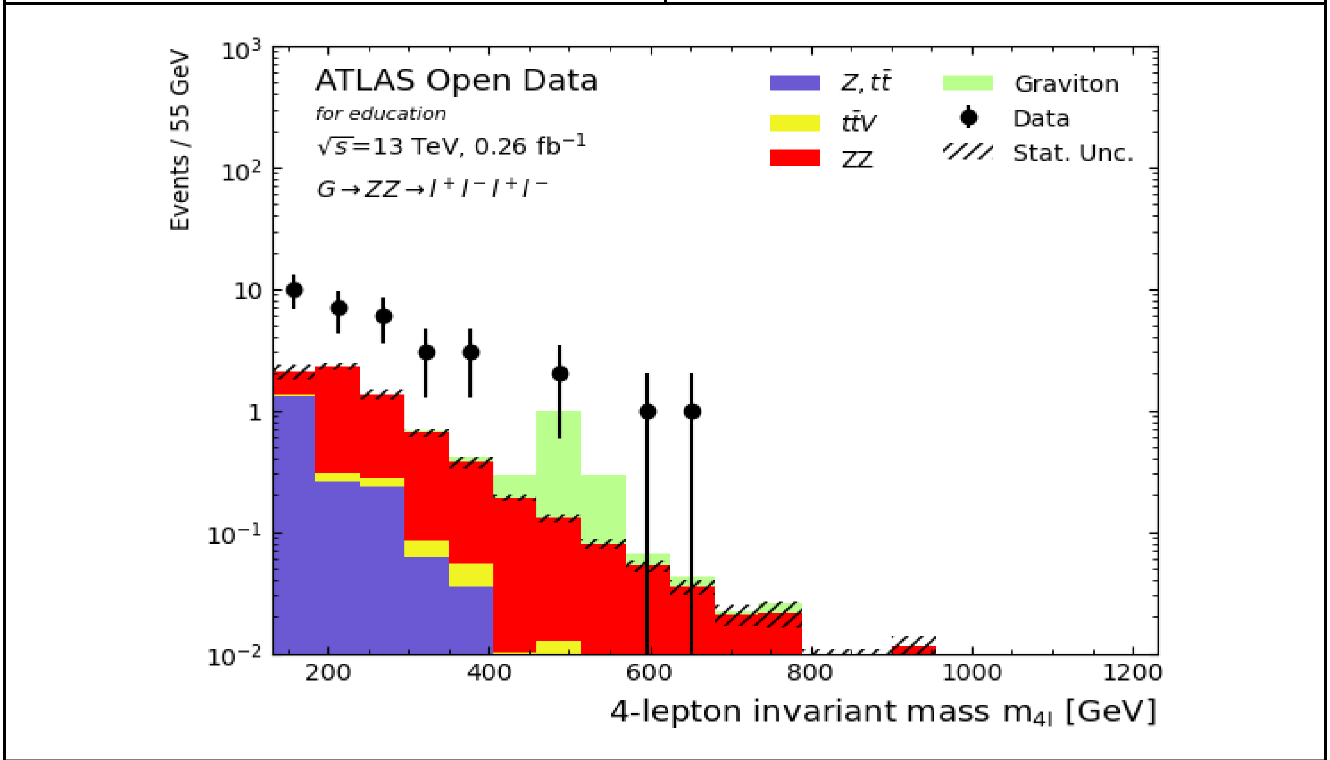
Fraction = 0.09



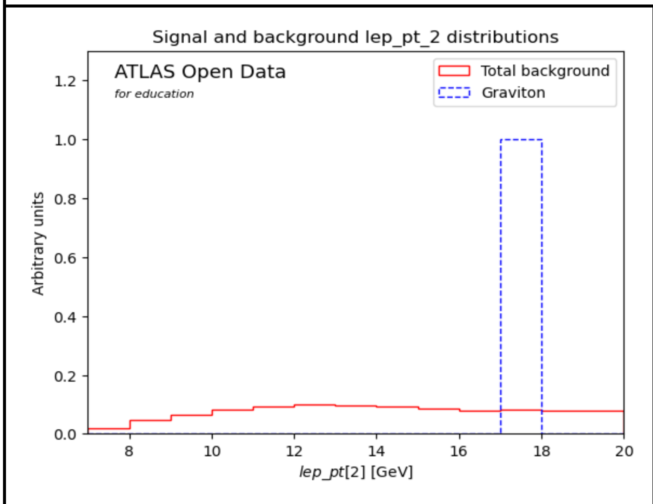


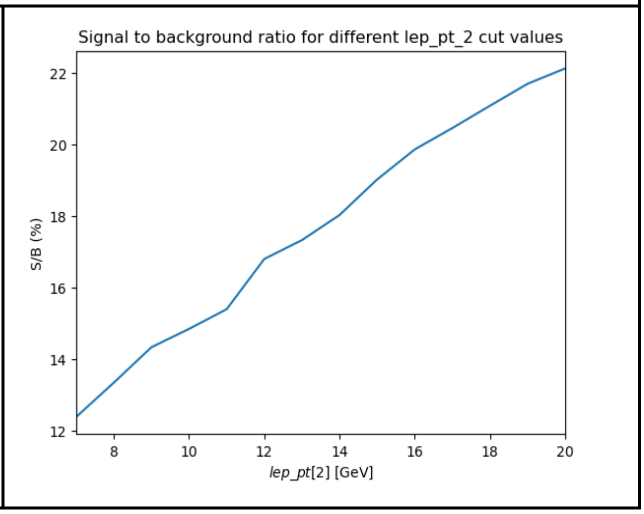




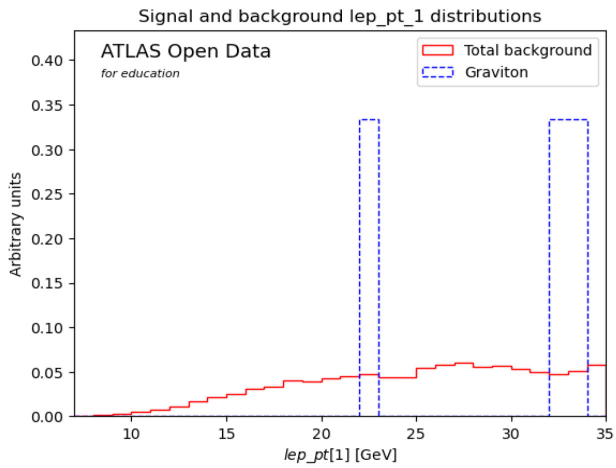


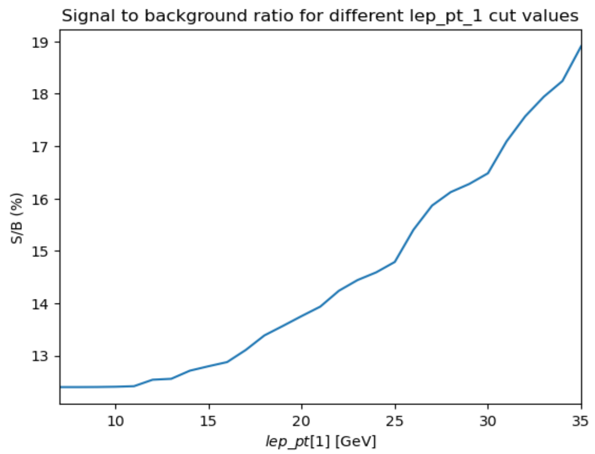
Lumi =4.7
Fraction = 0.09

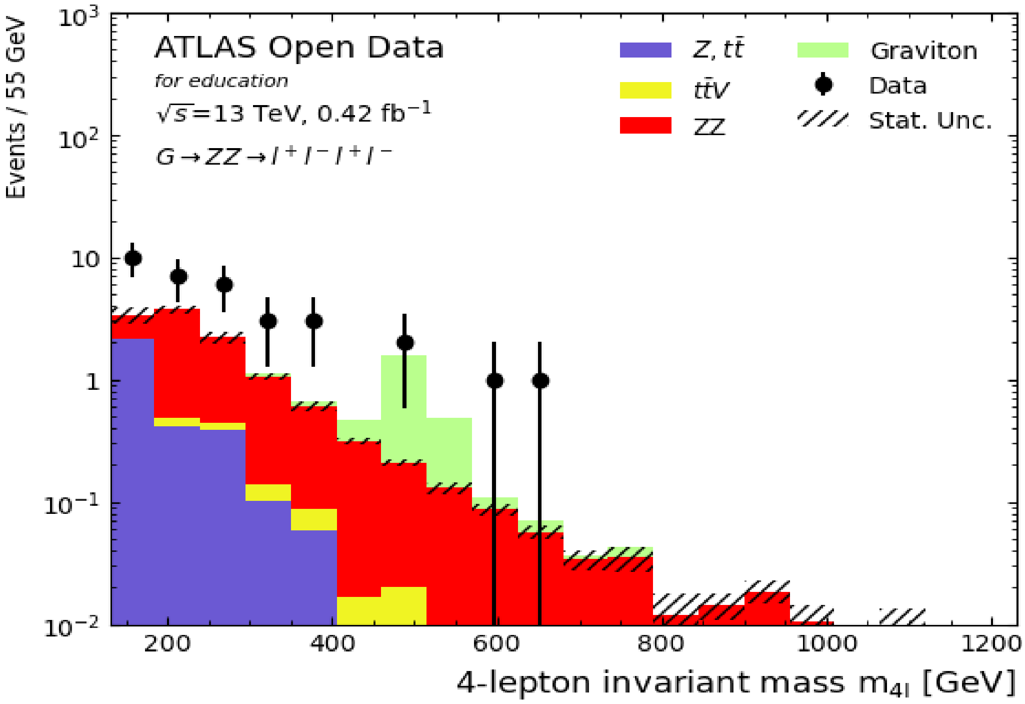




International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

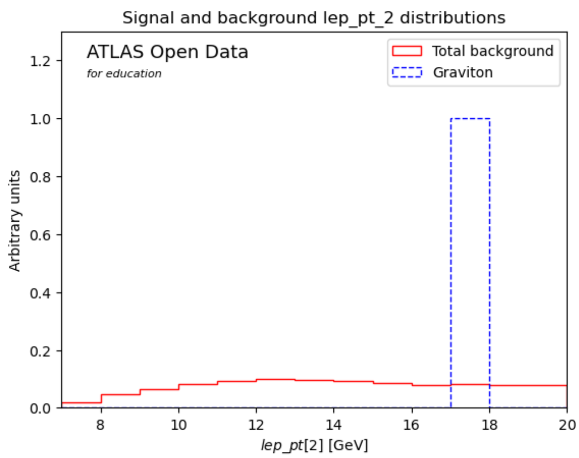


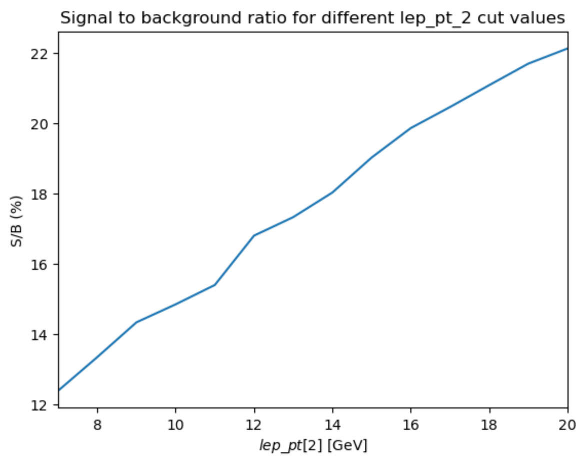


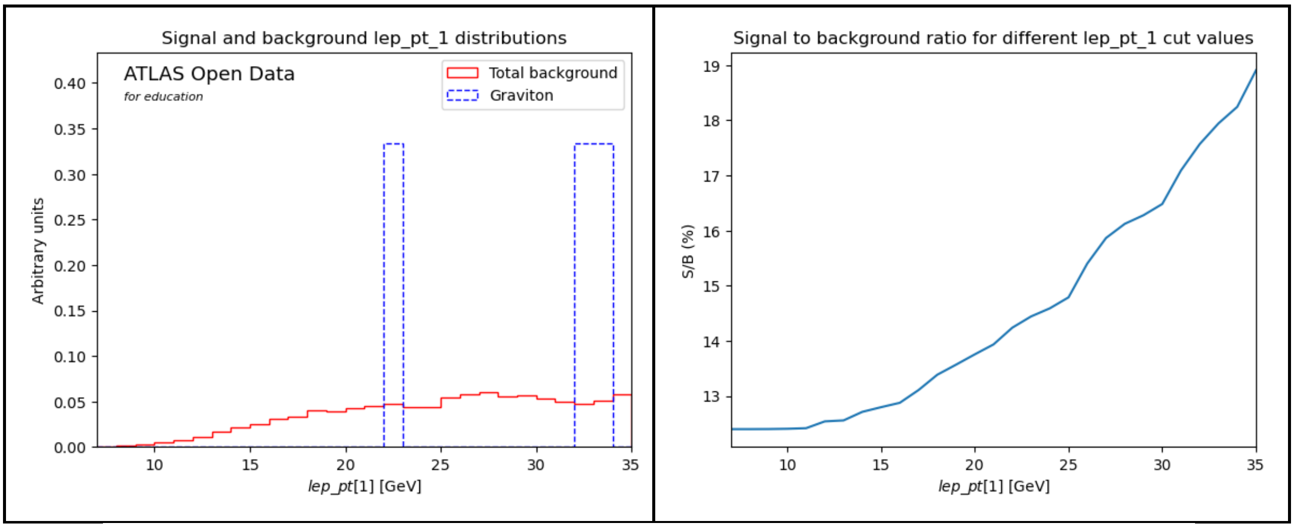


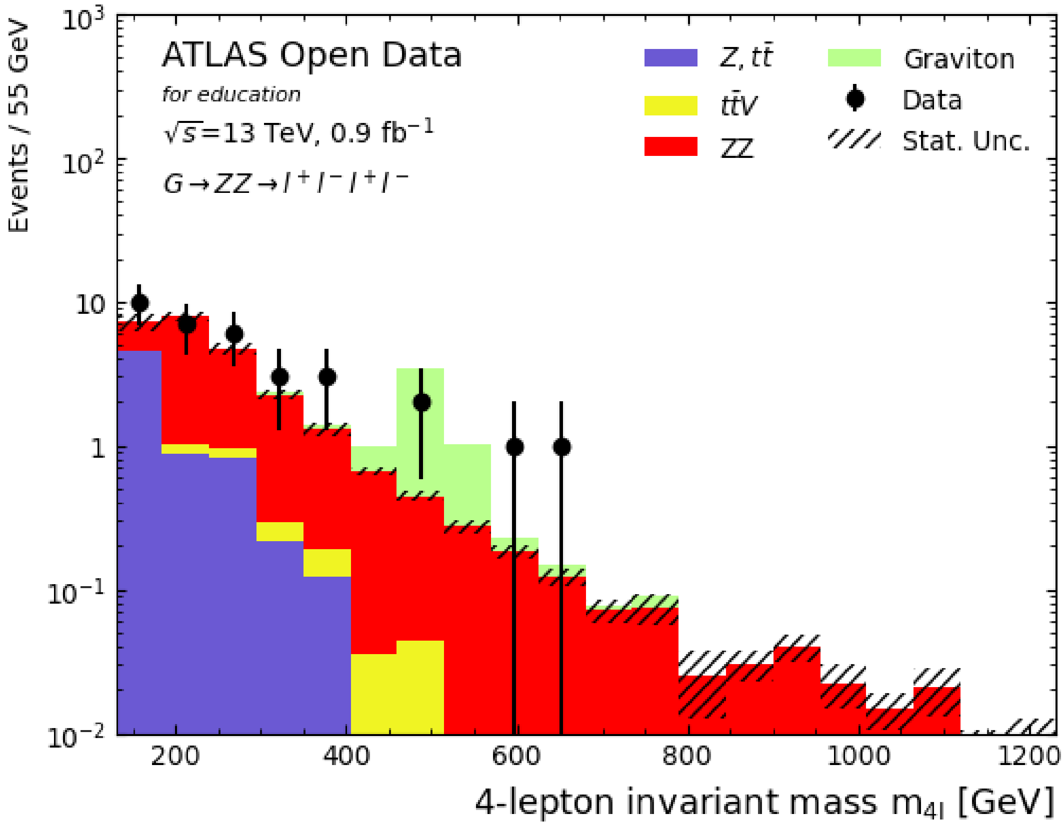
Lumi =10

Fraction = 0.09









4. Conclusion

In conclusion, this research paper delves into the fascinating world of particle physics, with a focus on exploring the decay mode of Graviton-> ZZ using ATLAS Open Data. The study encompasses a comprehensive feasibility analysis, examining the potential for discovering Gravitons and their implications for dark matter. The paper investigates the variations in luminosity, and the key background samples of Z, TT, TTV, and ZZ in the context of the Graviton-Z boson pair decay, ultimately leading to 4-lepton final states. The research has presented intriguing findings, particularly in the comparison of simulated backgrounds and actual data for a luminosity of 0.9 fb^{-1}. While the simulated background accounts for most of the observed data, there is an intriguing deviation observed in the mass distribution of 4 leptons in the energy range of 450-500 GeV. This deviation raises important questions and suggests the need for further studies and data collection. To ascertain the root cause of this observed difference between the Monte Carlo simulations and actual data, it is imperative to consider whether it is due to a flaw in the considered signal model or if it is merely a statistical fluctuation. Therefore, this research underlines the necessity for continued exploration, additional data collection, and the refinement of theoretical models in the quest to uncover the mysteries of the universe beyond the Standard Model.

References

[1] Close, F. (2004b). Particle Physics: A very short introduction. Oxford University Press.

[2] Hooper, D. (2021). At the Edge of Time: Exploring the Mysteries of Our Universe's First Seconds. Princeton University Press.

[3] The Standard Model. (n.d.). Institute of Physics. https://www.iop.org/explore-physics/big-ideas-physics/standard-model

[4] Facts and figures about the LHC | CERN. (2023, October 11). https://home.cern/resources/faqs/facts-and-figures-about-lhc

[5] Siegel, E. (2020, April 16). You Are Not Mostly Empty Space. Forbes. https://www.forbes.com/sites/startswithabang/2020/04/16/you-are-not-mostly-empty-space/?sh=21ab48912c2b

[6] DOE Explains. . .Quarks and Gluons. (n.d.). Energy.gov. https://www.energy.gov/science/doe-explainsquarks-and-gluons

[7] Simple science: particles. (n.d.). https://www.imperial.ac.uk/humanities/webdesign/2012/nickyguttridge/html/page4.html

[8] Riordan, M. (1992). The discovery of quarks. Science, 256(5061), 1287–1293. https://doi.org/10.1126/science.256.5061.1287

[9] Extra dimensions, gravitons, and tiny black holes. (2023, June 13). CERN. https://home.cern/science/physics/extra-dimensions-gravitons-and-tiny-black-holes

[10] DOE explains. . .the Higgs boson. (n.d.). Energy.gov. https://www.energy.gov/science/doe-explainsthe-higgs-boson

[11] The fundamental building blocks of matter. (n.d.). University of Illinois. https://courses.physics.illinois.edu/phys150/fa2003/slides/lect24.pdf

[12] Magazine, Z. M. N. (2013, September 11). Fat gravity particle gives clues to dark energy. Scientific American. https://www.scientificamerican.com/article/fat-gravity-particle-gives-clues-to-dark-energy/

[13] The large Hadron collider. (2023, October 11). CERN. https://home.cern/science/accelerators/large-hadron-collider#:~:text=The%20Large%20Hadron%20Collider%20(LHC,the%20particles%20along%20the%20way.

[14] ATLAS and CMS | University of Oxford Department of Physics. (n.d.). https://www2.physics.ox.ac.uk/accelerate/resources/background/atlas-and-cms

[15] ATLAS. (2023, October 11). CERN. https://home.cern/science/experiments/atlas

[16] CROSS SECTION AND LUMINOSITY. (n.d.). CERN Document Server. https://cds.cern.ch/record/2800578/files/Cross%20Section%20and%20Luminosity%20Physics%20Cheat%20Sheet.pdf