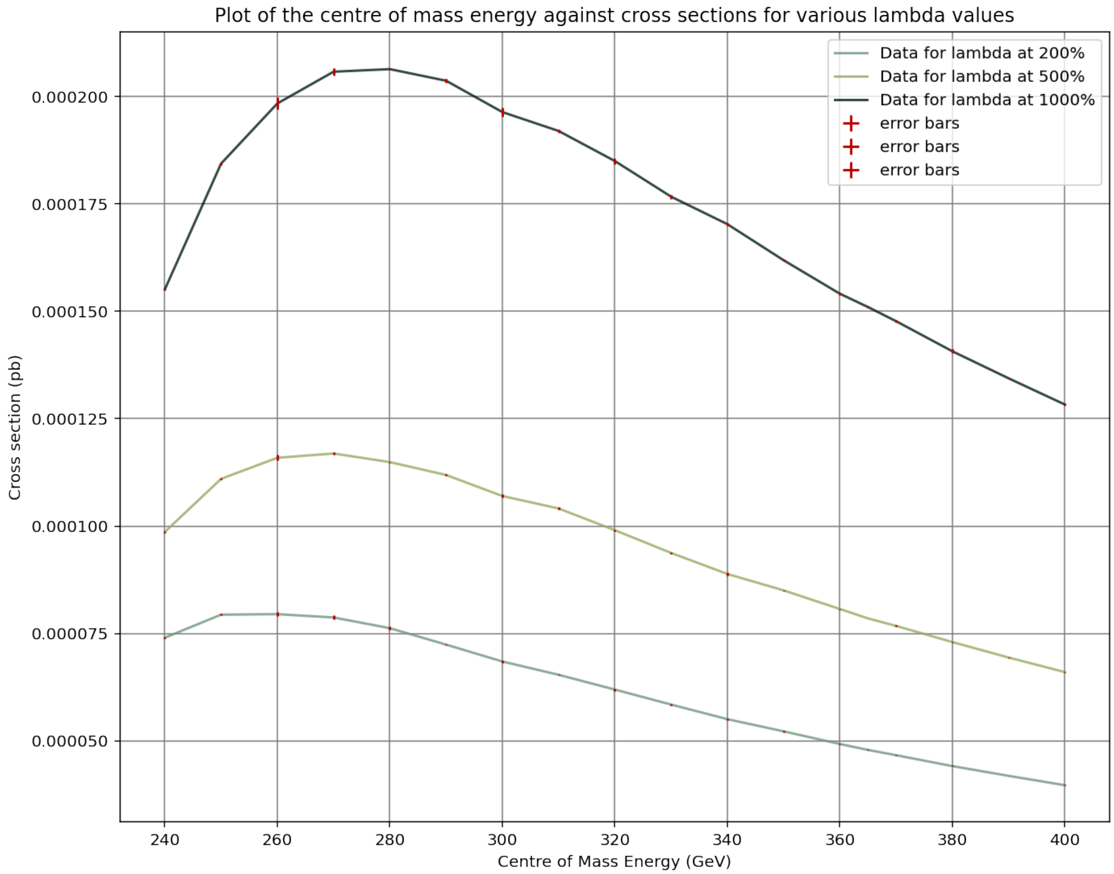
Appendix A

November 13, 2023

[7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

lam200 = np.loadtxt("lambda200.csv", delimiter = ",")
rows, cols = lam200.shape
CoME200 = lam200[0:,0]
CoME_error200 = lam200[0:,1]
Cross_section200 = lam200[0:,2]
Cross_section_error200 = lam200[0:,3]

lam500 = np.loadtxt("lambda500.csv", delimiter = ",")
rows, cols = lam500.shape
CoME500 = lam500[0:,0]
CoME_error500 = lam500[0:,1]
Cross_section500 = lam500[0:,2]
Cross_section_error500 = lam500[0:,3]

lam1000 = np.loadtxt("lambda1000.csv", delimiter = ",")
rows, cols = lam1000.shape
CoME1000 = lam1000[0:,0]
CoME_error1000 = lam1000[0:,1]
Cross_section1000 = lam1000[0:,2]
Cross_section_error1000 = lam1000[0:,3]

lam05 = np.loadtxt("lambda05.csv", delimiter = ",")
rows, cols = lam05.shape
CoME05 = lam05[0:,0]
CoME_error05 = lam05[0:,1]
Cross_section05 = lam05[0:,2]
Cross_section_error05 = lam05[0:,3]



plt.figure(figsize = (11, 9))
plt.title("Plot of the centre of mass energy against cross sections for various␣
↪lambda values")
```

```
plt.xlabel("Centre of Mass Energy (GeV)")
plt.ylabel("Cross section (pb)")
plt.plot(CoME200, Cross_section200, color = "#6F9580",alpha  =  0.8, label =␣
↪"Data for lambda at 200%", linestyle = "-")
plt.plot(CoME500, Cross_section500, color = "#95AA63", alpha =  0.8, label =␣
↪"Data for lambda at 500%", linestyle = "-")
plt.plot(CoME1000, Cross_section1000, color = "#011B10", alpha = 0.8, label =␣
↪"Data for lambda at 1000%", linestyle = "-")



plt.errorbar(CoME200, Cross_section200, xerr = CoME_error200, yerr =␣
↪Cross_section_error200,  color = "#BA0001", label = "error bars", linestyle␣
↪= "")
plt.errorbar(CoME500, Cross_section500, xerr = CoME_error500, yerr =␣
↪Cross_section_error500,  color = "#BA0001", label = "error bars", linestyle␣
↪= "")
plt.errorbar(CoME1000, Cross_section1000, xerr = CoME_error1000, yerr =␣
↪Cross_section_error1000,  color = "#BA0001", label = "error bars", linestyle␣
↪= "")
plt.grid(color = "grey")
plt.legend()
plt.show()
```

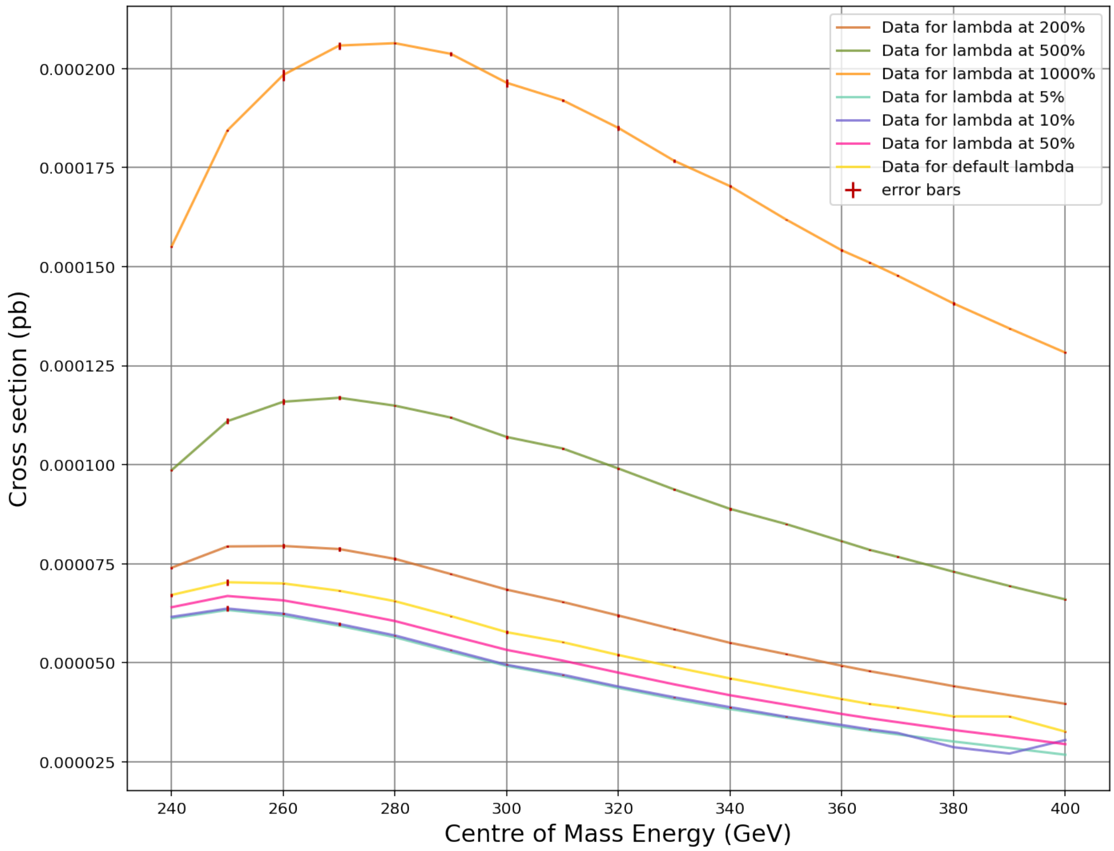
[7]:



[8]:
```
import numpy as np

lam10 = np.loadtxt("lambda10.csv", delimiter = ",")
rows, cols = lam10.shape
CoME10 = lam10[0:,0]
CoME_error10 = lam10[0:,1]
Cross_section10 = lam10[0:,2]
Cross_section_error10 = lam10[0:,3]


lam100 = np.loadtxt("lambda100.csv", delimiter = ",")
rows, cols = lam100.shape
CoME100 = lam100[0:,0]
CoME_error100 = lam100[0:,1]
Cross_section100 = lam100[0:,2]
Cross_section_error100 = lam100[0:,3]

lam50 = np.loadtxt("lambda50.csv", delimiter = ",")
rows, cols = lam50.shape
```

```
CoME50 = lam50[0:,0]
CoME_error50 = lam50[0:,1]
Cross_section50 = lam50[0:,2]
Cross_section_error50 = lam50[0:,3]

plt.figure(figsize = (11, 9))
#plt.title("Plot of the centre of mass energy against cross sections for various lambda values")
plt.xlabel("Centre of Mass Energy (GeV)", fontsize = 15)
plt.ylabel("Cross section (pb)", fontsize = 15)
plt.plot(CoME200, Cross_section200, color = "chocolate",alpha  =  0.75, label = "Data for lambda at 200%", linestyle = "-")
plt.plot(CoME500, Cross_section500, color = "olivedrab", alpha =  0.75, label = "Data for lambda at 500%", linestyle = "-")
plt.plot(CoME1000, Cross_section1000, color = "darkorange", alpha = 0.75, label = "Data for lambda at 1000%", linestyle = "-")
plt.plot(CoME05, Cross_section05, color = "mediumaquamarine",alpha  =  0.75, label = "Data for lambda at 5%", linestyle = "-")
plt.plot(CoME10, Cross_section10, color = "slateblue", alpha =  0.75, label = "Data for lambda at 10%", linestyle = "-")
plt.plot(CoME50, Cross_section50, color = "deeppink", alpha = 0.75, label = "Data for lambda at 50%", linestyle = "-")
plt.plot(CoME100, Cross_section100, color = "gold", alpha = 0.75, label = "Data for default lambda", linestyle = "-")



plt.errorbar(CoME200, Cross_section200, xerr = CoME_error200, yerr = Cross_section_error200,  color = "#BA0001", label = "error bars", linestyle = "")
plt.errorbar(CoME500, Cross_section500, xerr = CoME_error500, yerr = Cross_section_error500,  color = "#BA0001",  linestyle = "")
plt.errorbar(CoME1000, Cross_section1000, xerr = CoME_error1000, yerr = Cross_section_error1000,  color = "#BA0001",  linestyle = "")
plt.errorbar(CoME05, Cross_section200, xerr = CoME_error200, yerr = Cross_section_error200,  color = "#BA0001",  linestyle = "")
plt.errorbar(CoME50, Cross_section500, xerr = CoME_error50, yerr = Cross_section_error50,  color = "#BA0001", linestyle = "")
plt.errorbar(CoME10, Cross_section10, xerr = CoME_error10, yerr = Cross_section_error10,  color = "#BA0001",  linestyle = "")
plt.errorbar(CoME100, Cross_section100, xerr = CoME_error100, yerr = Cross_section_error100,  color = "#BA0001",  linestyle = "")
plt.grid(color = "grey")
plt.legend()
plt.show()
```

[8]:

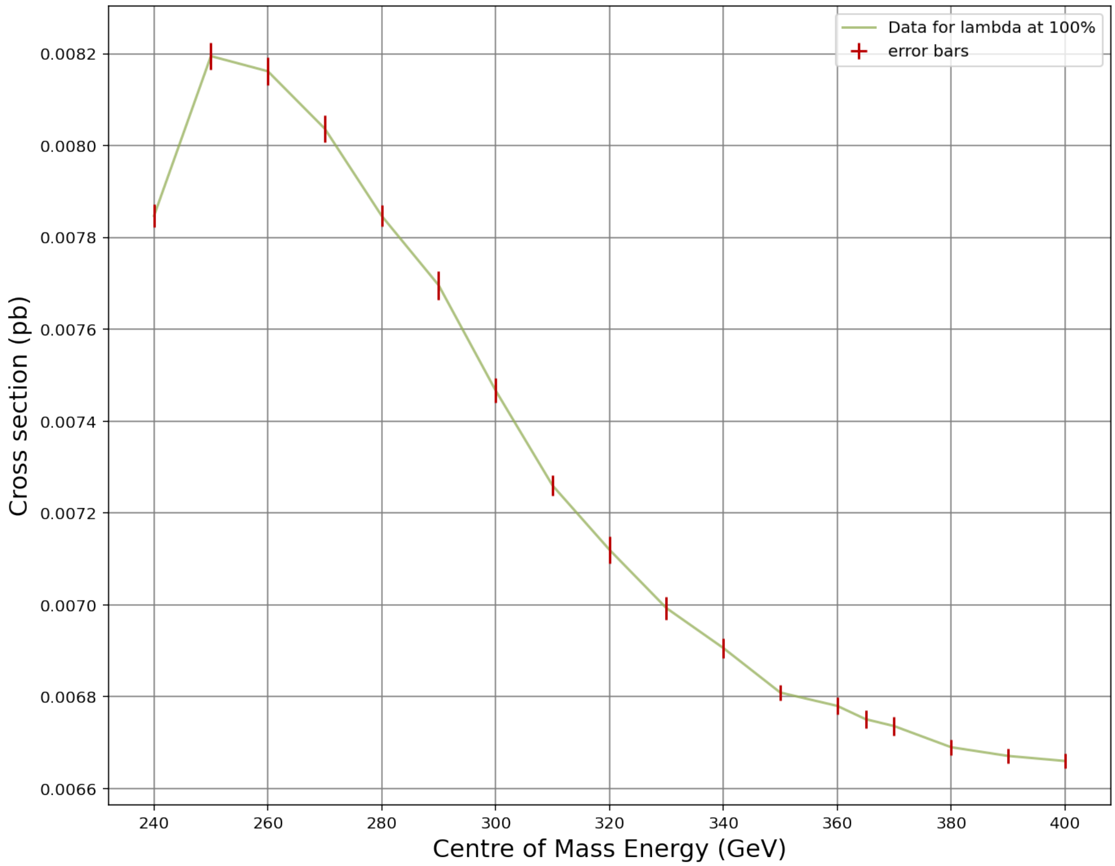


[10]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

CoME_scan = np.loadtxt("CoMe_100.csv", delimiter = ",")
rows, cols = CoME_scan.shape
COME100 = CoME_scan[0:,0]
COME_error100 = CoME_scan[0:,1]
Cross_secti0n100 = CoME_scan[0:,2]
Cross_secti0n_error100 = CoME_scan[0:,3]


plt.figure(figsize = (11, 9))
#plt.title("Plot of the cross section against centre of mass energy for leading␣
↪order processes for e+ e- > e+ e- H")
plt.xlabel("Centre of Mass Energy (GeV)", fontsize = 15)
plt.ylabel("Cross section (pb)", fontsize = 15)
plt.plot(COME100, Cross_secti0n100, color = "#93ae55",alpha  =  0.75, label =␣
↪"Data for lambda at 100%", linestyle = "-")
```

```
plt.errorbar(COME100, Cross_secti0n100, xerr = COME_error100, yerr =␣
↪Cross_secti0n_error100,  color = "#BA0001", label = "error bars", linestyle␣
↪= "")
plt.grid(color = "grey")
plt.legend()
plt.show()
```

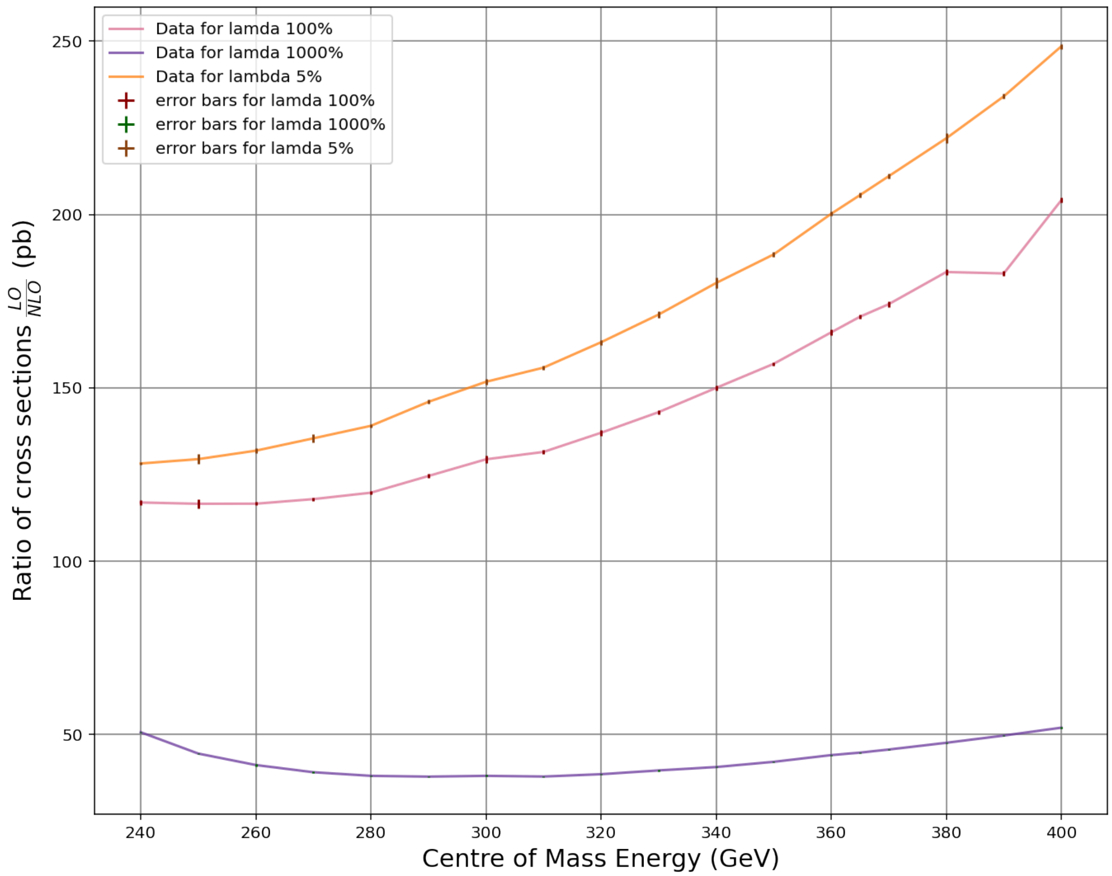
[10]:



[11]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

Ratio1 = np.loadtxt("CoMe_100.csv", delimiter = ",")
Ratio2 = np.loadtxt("lambda100.csv", delimiter = ",")
Ratio3 = np.loadtxt("lambda1000.csv", delimiter = ",")
Ratio4 = np.loadtxt("lambda05.csv", delimiter  = ",")
rows, cols = Ratio1.shape
rows, cols = Ratio2.shape
rows, cols = Ratio3.shape
rows, cols = Ratio4.shape
```

```
Come_ratio100 = Ratio1[0:,0]
Come_ratio100_error = Ratio1[0:,1]
CS_ratio100 = Ratio1[0:,2]/Ratio2[0:,2]
CS_ratio100_error = (Ratio1[0:,2]/Ratio2[0:,2])*np.sqrt((Ratio1[0:,3]/Ratio1[0:
 ↪,2])**2 +  (Ratio2[0:,3]/Ratio2[0:,2])**2)

Come_ratio1000 = Ratio1[0:,0]
Come_ratio1000_error = Ratio1[0:,1]
CS_ratio1000 = Ratio1[0:,2]/Ratio3[0:,2]
CS_ratio1000_error = (Ratio1[0:,2]/Ratio3[0:,2])*np.sqrt((Ratio1[0:,3]/Ratio1[0:
 ↪,2])**2 +  (Ratio3[0:,3]/Ratio3[0:,2])**2)

Come_ratio05 = Ratio1[0:,0]
Come_ratio05_error = Ratio1[0:,1]
CS_ratio05 = Ratio1[0:,2]/Ratio4[0:,2]
CS_ratio05_error = (Ratio1[0:,2]/Ratio4[0:,2])*np.sqrt((Ratio1[0:,3]/Ratio1[0:
 ↪,2])**2 +  (Ratio4[0:,3]/Ratio4[0:,2])**2)




plt.figure(figsize = (11, 9))
#plt.title("Plot of the ratio of the leading order over the next leading order␣
 ↪processes for e+ e- > e+ e- H")
plt.xlabel("Centre of Mass Energy (GeV)", fontsize = 15)
plt.ylabel(r"Ratio of cross sections $\frac{LO}{NLO}$ (pb)", fontsize = 15)
plt.plot(Come_ratio100, CS_ratio100, color = "palevioletred",alpha  =  0.75,␣
 ↪label = "Data for lamda 100%", linestyle = "-")
plt.plot(Come_ratio1000,  CS_ratio1000, color = "rebeccapurple", alpha =  0.75,␣
 ↪label = "Data for lamda 1000%", linestyle = "-")
plt.plot(Come_ratio05,  CS_ratio05, color = "tab:orange", alpha = 0.75, label =␣
 ↪"Data for lambda 5%", linestyle = "-")

plt.errorbar(Come_ratio100, CS_ratio100, xerr = Come_ratio100_error, yerr =␣
 ↪CS_ratio100_error,  color = "darkred", label = "error bars for lamda 100%",␣
 ↪linestyle = "")
plt.errorbar(Come_ratio1000, CS_ratio1000, xerr = Come_ratio1000_error, yerr =␣
 ↪CS_ratio1000_error,  color = "darkgreen", label = "error bars for lamda␣
 ↪1000%", linestyle = "")
plt.errorbar(Come_ratio05, CS_ratio05, xerr = Come_ratio05_error, yerr =␣
 ↪CS_ratio05_error,  color = "saddlebrown", label = "error bars for lamda 5%",␣
 ↪linestyle = "")
plt.grid(color = "grey")
plt.legend()
plt.show()
```

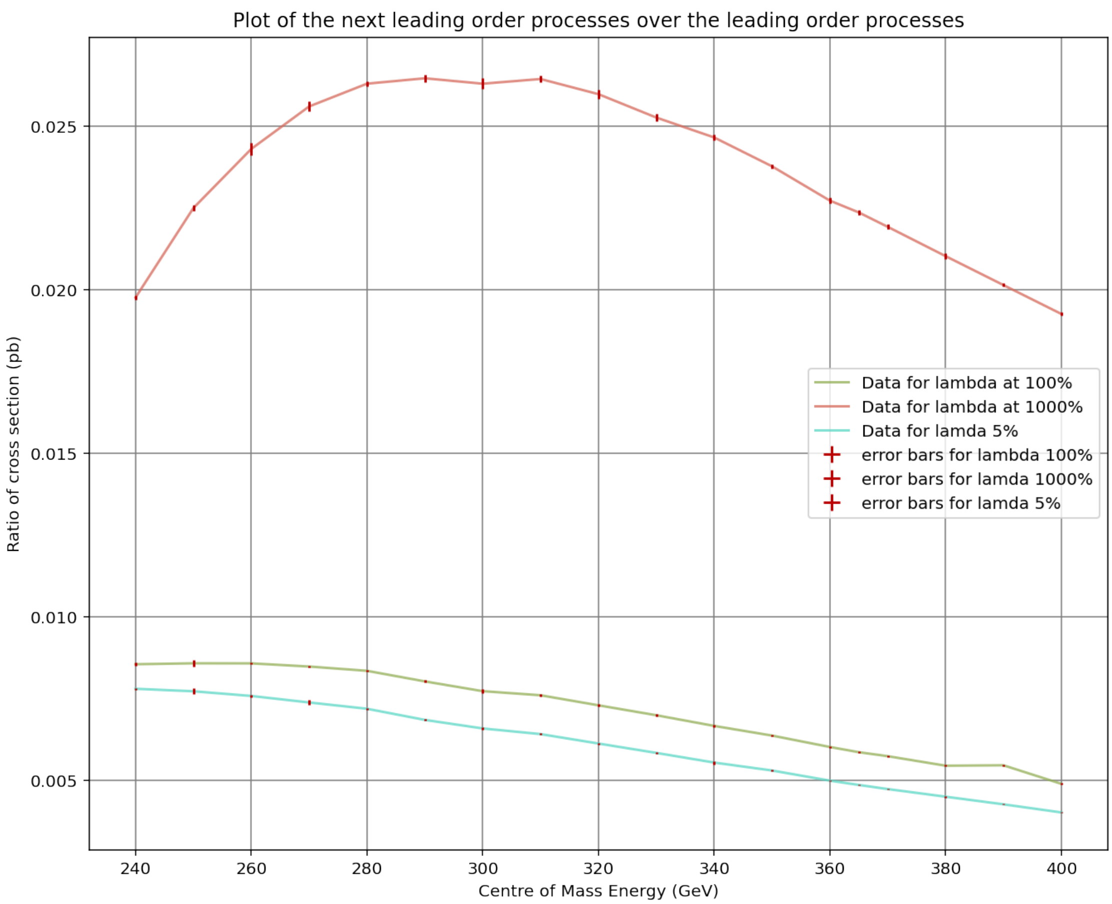
[11]:



[7]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

Ratio1 = np.loadtxt("CoMe_100.csv", delimiter = ",")
Ratio2 = np.loadtxt("lambda100.csv", delimiter = ",")
Ratio3 = np.loadtxt("lambda1000.csv", delimiter = ",")
Ratio4 = np.loadtxt("lambda05.csv", delimiter  = ",")
rows, cols = Ratio1.shape
rows, cols = Ratio2.shape
rows, cols = Ratio3.shape
rows, cols = Ratio4.shape
Come_ratio100 = Ratio1[0:,0]
Come_ratio100_error = Ratio1[0:,1]
CS_ratio100 = Ratio2[0:,2]/Ratio1[0:,2]
CS_ratio100_error = (Ratio2[0:,2]/Ratio1[0:,2])*np.sqrt((Ratio1[0:,3]/Ratio1[0:
 ↪,2])**2 +  (Ratio2[0:,3]/Ratio2[0:,2])**2)

Come_ratio1000 = Ratio1[0:,0]
```

```
Come_ratio1000_error = Ratio1[0:,1]
CS_ratio1000 = Ratio3[0:,2]/Ratio1[0:,2]
CS_ratio1000_error = (Ratio3[0:,2]/Ratio1[0:,2])*np.sqrt((Ratio1[0:,3]/Ratio1[0:
↪,2])**2 +  (Ratio3[0:,3]/Ratio3[0:,2])**2)

Come_ratio05 = Ratio1[0:,0]
Come_ratio05_error = Ratio1[0:,1]
CS_ratio05 = Ratio4[0:,2]/Ratio1[0:,2]
CS_ratio05_error = (Ratio4[0:,2]/Ratio1[0:,2])*np.sqrt((Ratio1[0:,3]/Ratio1[0:
↪,2])**2 +  (Ratio4[0:,3]/Ratio4[0:,2])**2)



plt.figure(figsize = (11, 9))
plt.title("Plot of the next leading order processes over the leading order␣
↪processes")
plt.xlabel("Centre of Mass Energy (GeV)")
plt.ylabel("Ratio of cross section (pb)")
plt.plot(Come_ratio100, CS_ratio100, color = "#93ae55",alpha  =  0.75, label =␣
↪"Data for lambda at 100%", linestyle = "-")
plt.plot(Come_ratio1000,  CS_ratio1000, color = "#d6695a", alpha =  0.75, label␣
↪= "Data for lambda at 1000%", linestyle = "-")
plt.plot(Come_ratio05,  CS_ratio05, color = "#59d7c6", alpha = 0.75, label =␣
↪"Data for lamda 5%", linestyle = "-")

plt.errorbar(Come_ratio100, CS_ratio100, xerr = Come_ratio100_error, yerr =␣
↪CS_ratio100_error,  color = "#BA0001", label = "error bars for lambda 100%",␣
↪linestyle = "")
plt.errorbar(Come_ratio1000, CS_ratio1000, xerr = Come_ratio1000_error, yerr =␣
↪CS_ratio1000_error,  color = "#BA0001", label = "error bars for lamda␣
↪1000%", linestyle = "")
plt.errorbar(Come_ratio05, CS_ratio05, xerr = Come_ratio05_error, yerr =␣
↪CS_ratio05_error,  color = "#BA0001", label = "error bars for lamda 5%",␣
↪linestyle = "")
plt.grid(color = "grey")
plt.legend()
plt.show()
```

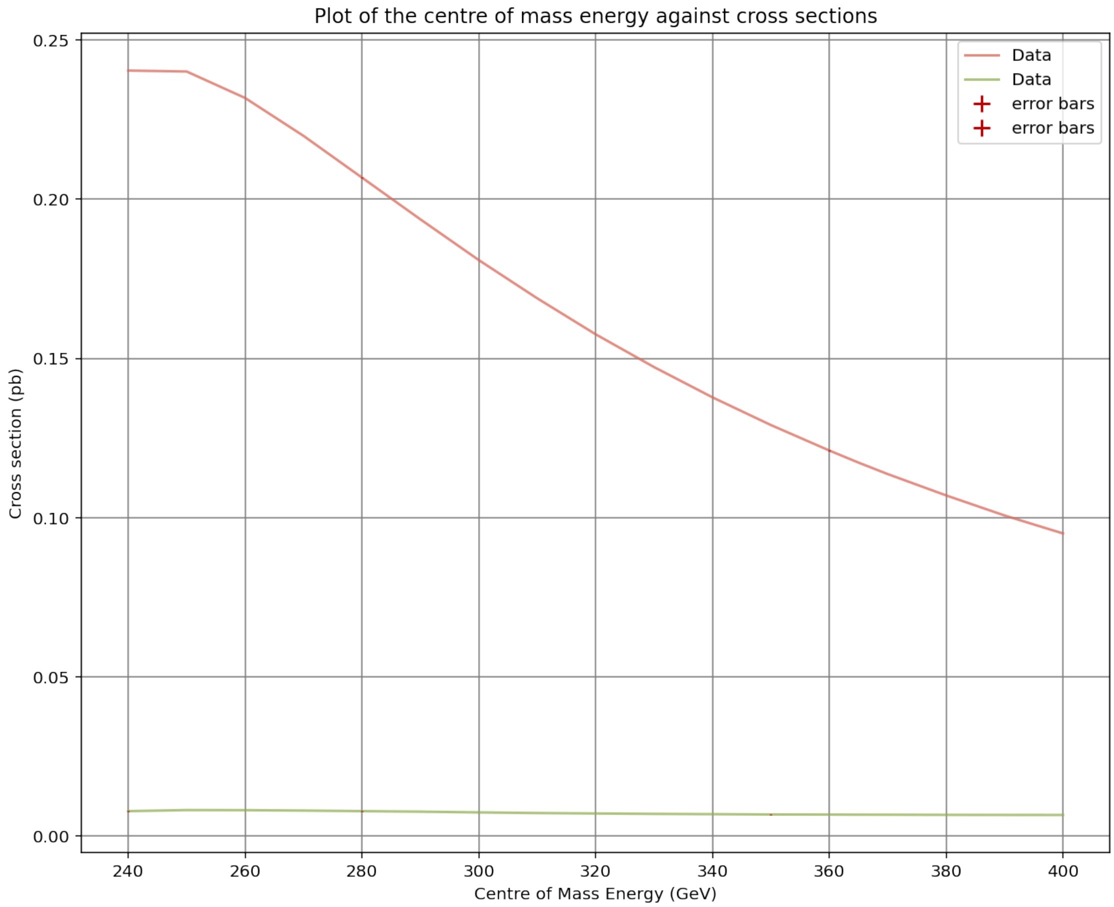
[7]:



[8]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

CoME_scan = np.loadtxt("CoMe_100.csv", delimiter = ",")
rows, cols = CoME_scan.shape
COME100 = CoME_scan[0:,0]
COME_error100 = CoME_scan[0:,1]
Cross_secti0n100 = CoME_scan[0:,2]
Cross_secti0n_error100 = CoME_scan[0:,3]

eeHz_100 = np.loadtxt("eeHz_100.csv", delimiter = ",")
rows, cols = eeHz_100.shape
Come_eeHz100 = eeHz_100[0:,0]
```

```
Come_error_eeHz100 = eeHz_100[0:,1]
Cross_sectioneeHz100 = eeHz_100[0:,2]
Cross_section_erroreeHz100 = eeHz_100[0:,3]


plt.figure(figsize = (11, 9))
plt.title("Plot of the centre of mass energy against cross sections")
plt.xlabel("Centre of Mass Energy (GeV)")
plt.ylabel("Cross section (pb)")
plt.plot(Come_eeHz100, Cross_sectioneeHz100, color = "#d6695a",alpha  =  0.75,␣
 ↪label = "Data", linestyle = "-")
plt.errorbar(Come_eeHz100, Cross_sectioneeHz100, xerr = Come_error_eeHz100,␣
 ↪yerr = Cross_section_erroreeHz100,  color = "#BA0001", label = "error bars",␣
 ↪linestyle = "")
plt.plot(COME100, Cross_secti0n100, color = "#93ae55",alpha  =  0.75, label =␣
 ↪"Data", linestyle = "-")
plt.errorbar(COME100, Cross_secti0n100, xerr = COME_error100, yerr =␣
 ↪Cross_secti0n_error100,  color = "#BA0001", label = "error bars", linestyle␣
 ↪= "")
#plt.ylim(0.00005, 0.1)
plt.grid(color = "grey")
plt.legend()
plt.show()
```

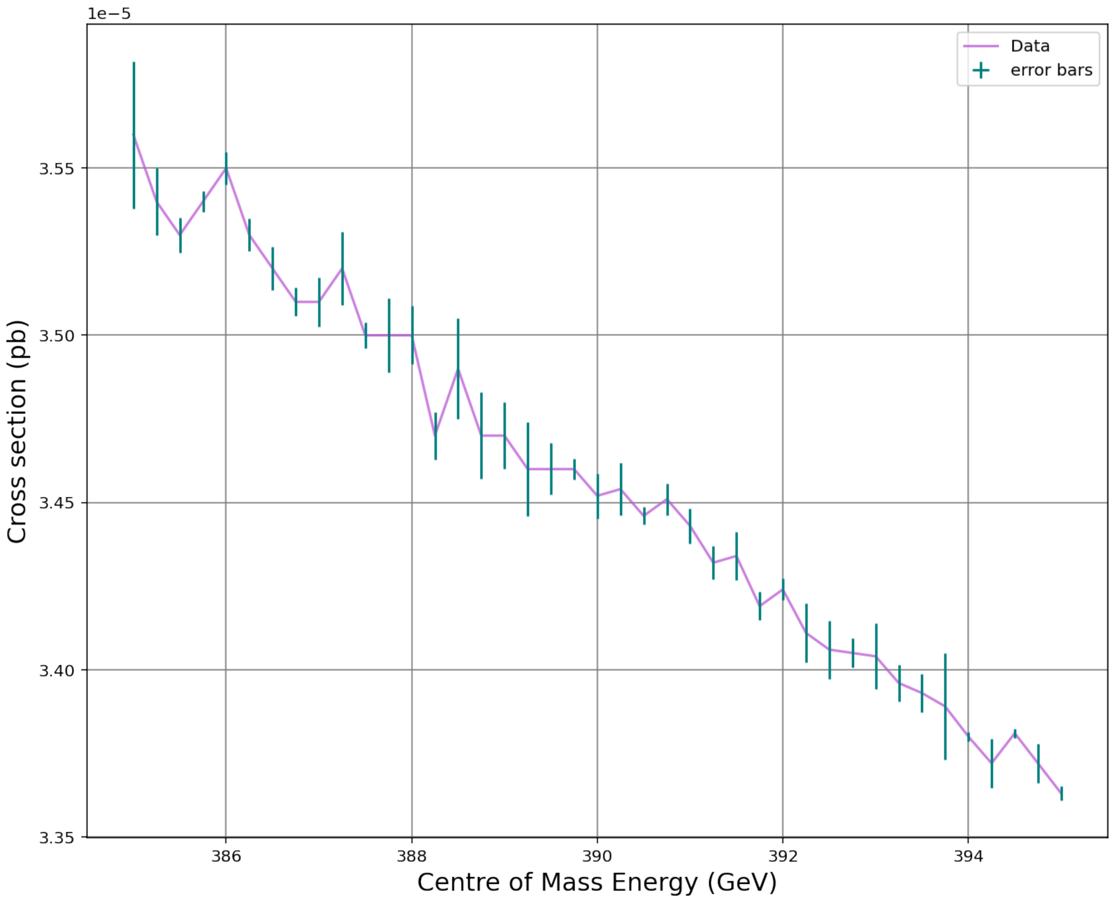
[8]:



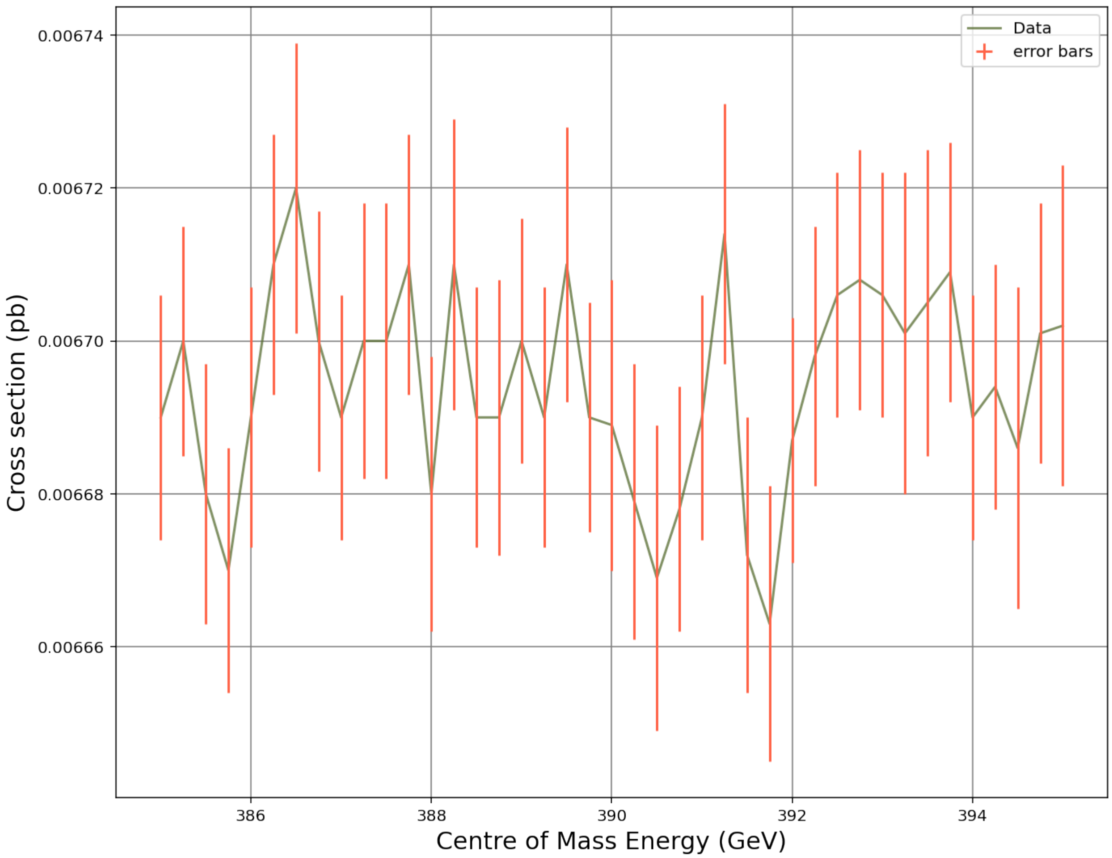
[12]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

eeHee_390_NLO = np.loadtxt("eeHee_390_NLO.csv", delimiter = ",")
rows, cols = eeHee_390_NLO.shape
roots = eeHee_390_NLO[0:,0]
roots_error = eeHee_390_NLO[0:,1]
Cross_section_390 = eeHee_390_NLO[0:,2]
Cross_section_390error = eeHee_390_NLO[0:,3]

eeHee_390_LO = np.loadtxt("eeHee_390_LO.csv", delimiter = ",")
rows, cols = eeHee_390_LO.shape
```

```
rootsLO = eeHee_390_LO[0:,0]
roots_errorLO = eeHee_390_LO[0:,1]
Cross_section_390LO = eeHee_390_LO[0:,2]
Cross_section_390errorLO = eeHee_390_LO[0:,3]


plt.figure(figsize = (11, 9))
#plt.title("Finer centre of mass energy scan from 385 to 395 GeV for next␣
↪leading order processes of e+ e- > e+ e- H ")
plt.xlabel("Centre of Mass Energy (GeV)", fontsize = 15)
plt.ylabel("Cross section (pb)", fontsize = 15)
plt.plot(roots, Cross_section_390, color = "mediumorchid",alpha  =  0.75, label␣
↪= "Data", linestyle = "-")
plt.errorbar(roots, Cross_section_390, xerr = roots_error, yerr =␣
↪Cross_section_390error,  color = "teal", label = "error bars", linestyle =␣
↪"")
plt.grid(color = "grey")
plt.legend()
plt.show()

plt.figure(figsize = (11, 9))
#plt.title("Finer centre of mass energy scan from 385 to 395 GeV for leading␣
↪order processes of e+ e- > e+ e- H")
plt.xlabel("Centre of Mass Energy (GeV)", fontsize = 15)
plt.ylabel("Cross section (pb)",  fontsize = 15)
plt.plot(rootsLO, Cross_section_390LO, color = "darkolivegreen",alpha  =  0.75,␣
↪label = "Data", linestyle = "-")
plt.errorbar(rootsLO, Cross_section_390LO, xerr = roots_errorLO, yerr =␣
↪Cross_section_390errorLO,  color = "tomato", label = "error bars", linestyle␣
↪= "")
plt.grid(color = "grey")
plt.legend()
plt.show()
```

[12]:


1e−5
Data
error bars
3.55
3.50
3.45
3.40
3.35
386
388
390
392
394
Cross section (pb)
Centre of Mass Energy (GeV)

[12]:



[0]:

```

```

[10]:

```
from scipy.stats import chi2 as stats_chi2
from scipy.optimize import least_squares
import matplotlib.pyplot as plt
import numpy as np

def fit(xdata, ydata, xerror, yerror, init_params):
    """
    Function to perform least-squares fit for a straight line.
    - xdata         array of x data points
    - ydata         array of y data points
    - xerr          array of x data errors
    - yerr          array of y data errors
    - init_params   array of function parameters
    """
    def straight_line_fit(params, xdata):
        f = params[0] + params[1]*xdata
        return(f)
    def straight_line_fit_differential(params, xdata):
```

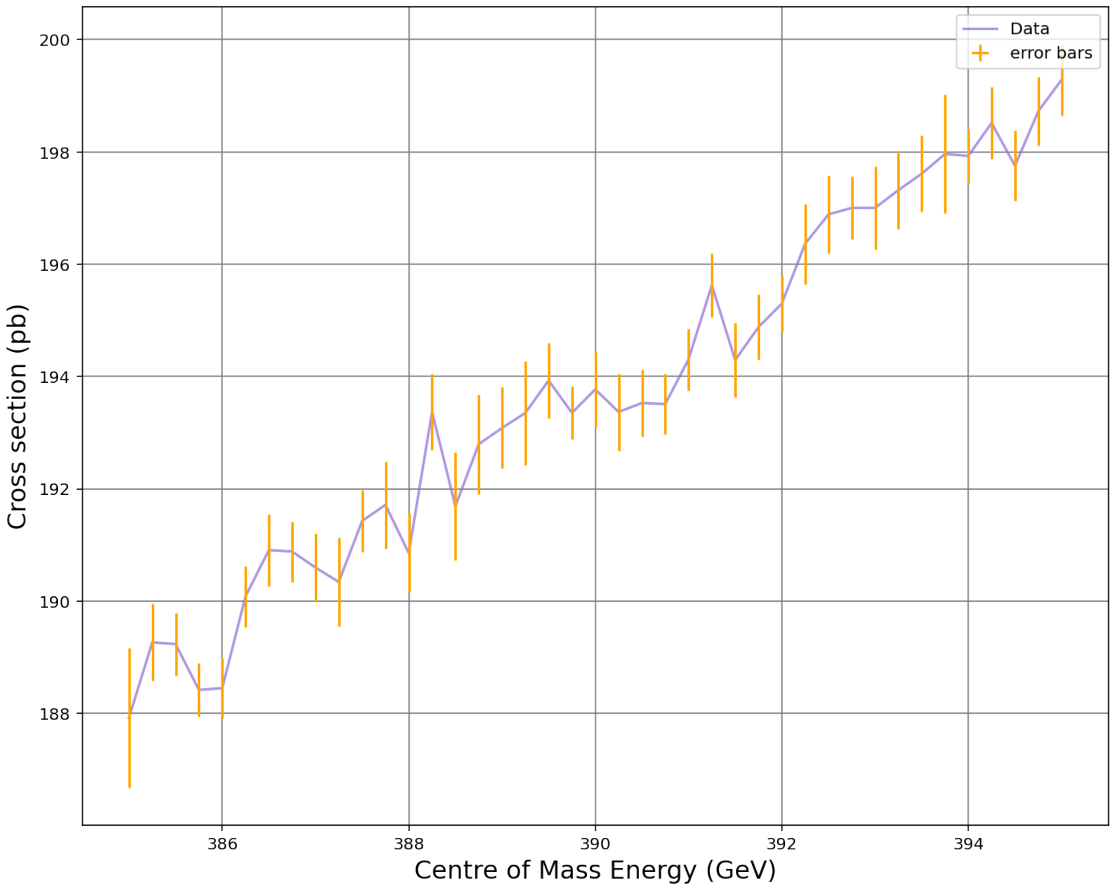
```
        df = params[1]
        return(df)
    #The residuals are the difference between the data points and the model␣
↪function of the data.
    def minimise_function(params, xdata, ydata, xerror, yerror):
        residuals = (ydata - straight_line_fit(params, xdata)) / (np.
↪sqrt(yerror**2 + (straight_line_fit_differential(params, xdata))**2␣
↪*(xerror**2)))
        return(residuals)

    # Run fit
    result = least_squares(minimise_function, init_params, args=(xdata, ydata,␣
↪xerror, yerror))

    # Check fit succeeds
    if not result.success or result.status < 1:
        print ("ERROR: Fit failed with message {}".format(result.message))
        print ("Please check the data and inital parameter estimates")
        return 0,0,0,0
    else:
        print ("Fit succeeded")

    # Get fitted parameters
    final_params = result.x
    c = final_params[0]
    m = final_params[1]
    nparams = len(final_params)

    # Calculate chi2
    chi2_array = result.fun ** 2
    chi2 = sum(chi2_array)
    npoints = len(xdata)
    reduced_chi2 = chi2 / (npoints - nparams)
    chi2_prob = stats_chi2.sf(chi2, (npoints - nparams))

    # Print chi2
    np.set_printoptions(precision = 3)
    print("\n=== Fit quality ===")
    print("chisq per point = \n",chi2_array)
    print("chisq = {:7.5g}, ndf = {}, chisq/NDF = {:7.5g}, chisq prob = {:7.
↪5g}\n".format(chi2, npoints-nparams, reduced_chi2, chi2_prob))

    if reduced_chi2 < 0.25 or reduced_chi2 > 4:
        print("WARNING: chi2/ndf suspiciously small or large.  Please check the␣
↪data and initial parameter estimates")

    if chi2_prob < 0.05:
```

```
        print("WARNING: chi2 probability for given degrees of freedom less than␣
↪0.05 .  Please check the data and initial parameter estimates")

    # Calculate errors
    jacobian = result.jac
    jacobian2 = np.dot(jacobian.T, jacobian)
    determinant = np.linalg.det(jacobian2)

    if determinant < 1E-42:
        print(f"Matrix singular (determinant = {determinant}, error calculation␣
↪failed.")
        param_errors = np.zeros(nparams)
    else:
        covariance = np.linalg.inv(jacobian2)
        param_errors = np.sqrt(covariance.diagonal())

    print ("=== Fitted parameters ===")
    print ("c = {:7.5g} +- {:7.5g}".format(final_params[0], param_errors[0]))
    print ("m = {:7.5g} +- {:7.5g}".format(final_params[1], param_errors[1]))

    # Calculate fitted function values
    yfit = straight_line_fit(final_params, xdata)

    # Visualise result
    fig = plt.figure(figsize = (8, 6))
    plt.title('Fit result')
    plt.xlabel('x', fontsize=16)
    plt.ylabel('y', fontsize=16)
    plt.grid(color = 'grey', linestyle="--")

    plt.errorbar(xdata, ydata, xerr = xerror, yerr = yerror, fmt='k', linestyle␣
↪= '', label = "Data")
    plt.plot(xdata, yfit, color = '#C8A2C8', linestyle = '-', label = "Fit")


    plt.legend(loc = 2, fontsize=16)

    text = "c: {:7.5g} +- {:7.5g}\n".format(final_params[0], param_errors[0])
    text += "m: {:7.5g} +- {:7.5g}\n".format(final_params[1], param_errors[1])
    plt.text(0.95, 0.0, text, transform = fig.axes[0].transAxes, ha = "right",␣
↪va = "bottom", fontsize=12)

    plt.show()

    # Return arrays of parameters and associated errors
    return final_params, param_errors
```

[13]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

Ratio1_390 = np.loadtxt("eeHee_390_NLO.csv", delimiter = ",")
Ratio2_390 = np.loadtxt("eeHee_390_LO.csv", delimiter = ",")
rows, cols = Ratio1_390.shape
rows, cols = Ratio2_390.shape
comeratio_390 = Ratio1_390[0:,0]
comeratio_390_error = Ratio1_390[0:,1]
CSratio_390 = Ratio2_390[0:,2]/Ratio1_390[0:,2]
CSratio_390_error = (Ratio2_390[0:,2]/Ratio1_390[0:,2])*np.sqrt((Ratio1_390[0:
 ↪,3]/Ratio1_390[0:,2])**2 +  (Ratio2_390[0:,3]/Ratio2_390[0:,2])**2)
plt.figure(figsize = (11, 9))
#plt.title("Ratio of the leading order over the next leading order for the␣
 ↪process e+ e- > e+ e- H for a finer scan around the 390 GeV point.")
plt.xlabel("Centre of Mass Energy (GeV)", fontsize  = 15)
plt.ylabel("Cross section (pb)", fontsize = 15)
plt.plot(comeratio_390, CSratio_390, color = "mediumpurple",alpha  =  0.75,␣
 ↪label = "Data", linestyle = "-")
plt.errorbar(comeratio_390, CSratio_390, xerr = comeratio_390_error, yerr =␣
 ↪CSratio_390_error,  color = "orange", label = "error bars", linestyle = "")
plt.grid(color = "grey")
plt.legend()
plt.show()
```

[13]:



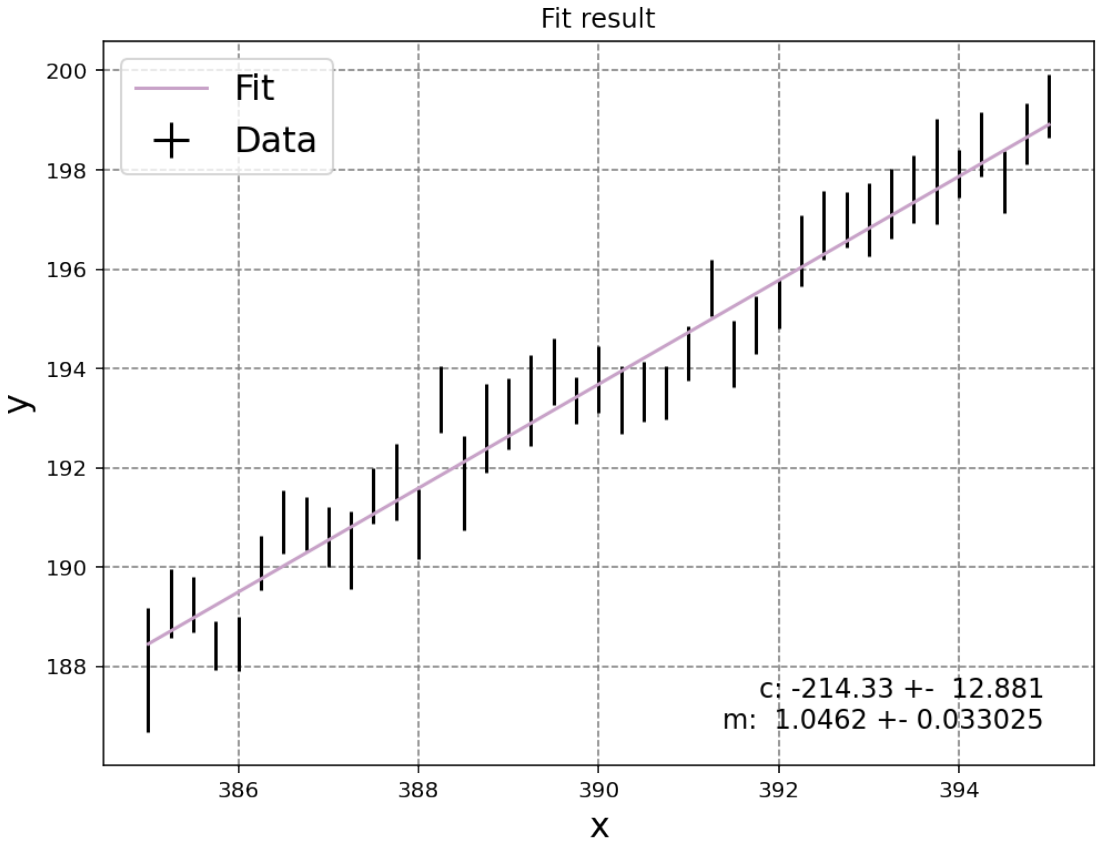
$$N_{events} = \sigma \cdot \mathcal{L}$$

[12]:
```
init_params = [-180, 1]

#fit data to straight line
params, errors =  fit(comeratio_390,
 ↪CSratio_390,comeratio_390_error,CSratio_390_error, init_params)
```

```
Fit succeeded

=== Fit quality ===
chisq per point =
 [0.18  0.659 0.221 2.882 3.717 0.358 1.921 1.264 0.008 0.349 0.428 0.25
 1.094 5.072 0.194 0.225 0.385 0.244 1.306 0.021 0.018 0.702 1.274 3.218
 0.584 1.256 2.083 1.188 0.899 0.213 0.738 0.634 0.062 0.118 0.161 0.115
 0.017 0.366 1.033 0.015 0.345]
chisq =  35.819, ndf = 39, chisq/NDF = 0.91842, chisq prob = 0.61578
```

```
=== Fitted parameters ===
c = -214.33 +-  12.881
m =  1.0462 +- 0.033025
```

[12]:



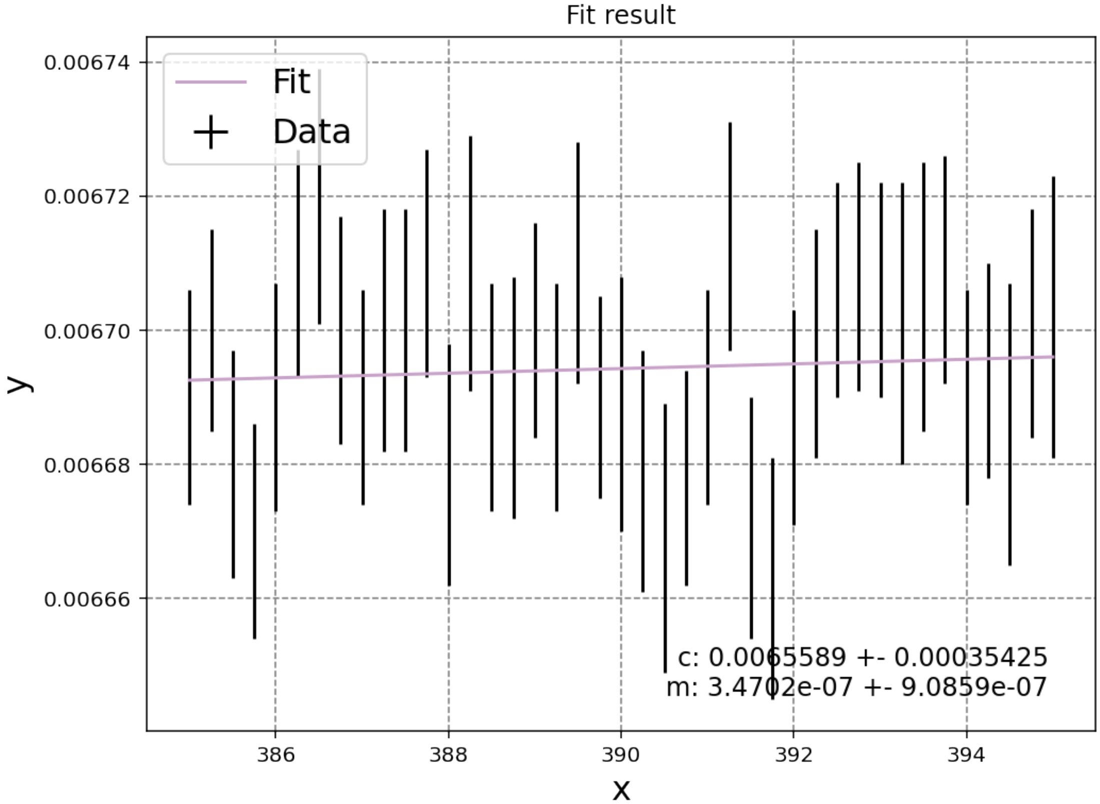
[13]:

```
init_params = [-180, 1]

#fit data to straight line
params, errors =  fit(rootsLO,␣
↪Cross_section_390LO,roots_errorLO,Cross_section_390errorLO, init_params)
```

```
Fit succeeded

=== Fit quality ===
chisq per point =
 [0.025 0.241 0.56  2.032 0.029 1.002 2.009 0.162 0.041 0.137 0.134 0.942
 0.57  0.738 0.049 0.046 0.144 0.056 0.779 0.078 0.077 0.729 1.62  1.069
 0.084 1.287 1.605 3.139 0.249 0.03  0.46  0.564 0.445 0.071 0.226 0.623
 0.126 0.012 0.22  0.089 0.081]
chisq =  22.579, ndf = 39, chisq/NDF = 0.57894, chisq prob = 0.98357

=== Fitted parameters ===
```

```
c = 0.0065589 +- 0.00035425
m = 3.4702e-07 +- 9.0859e-07
```

[13]:



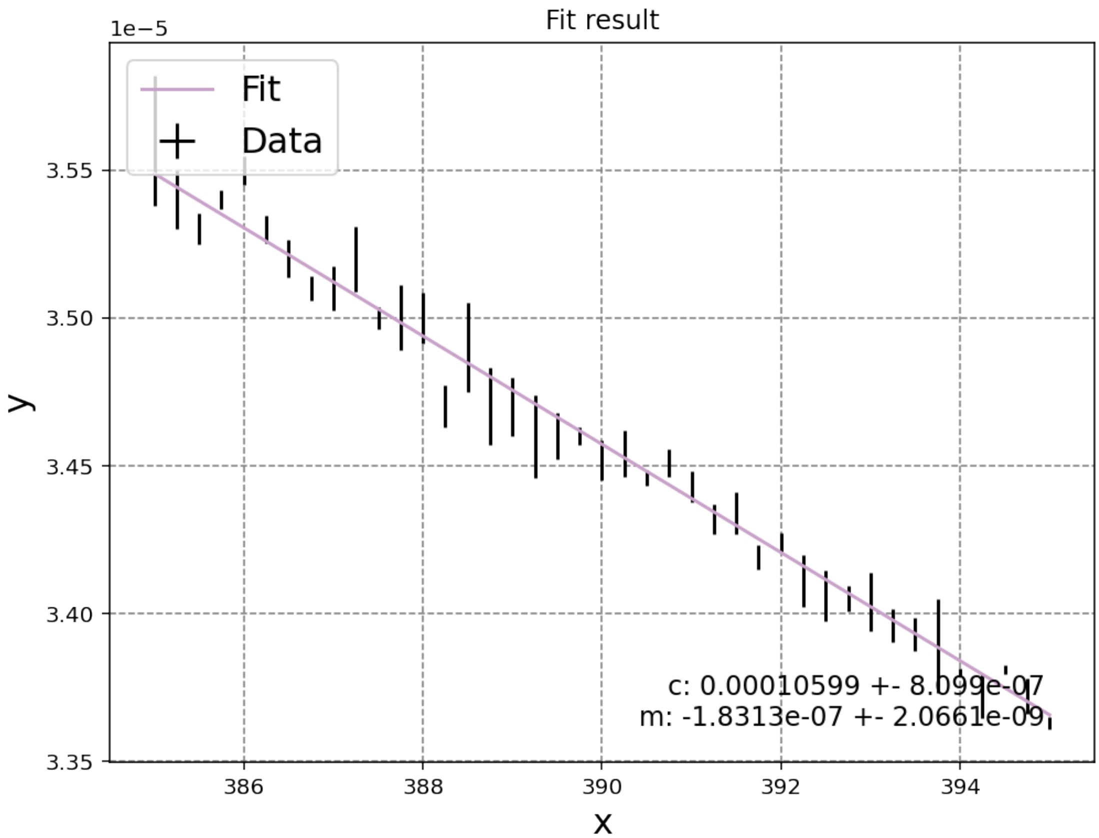
[14]:

```
init_params = [-180, 1]

#fit data to straight line
params, errors =  fit(roots, Cross_section_390, roots_error,␣
 ↪Cross_section_390error, init_params)
```

```
Fit succeeded

=== Fit quality ===
chisq per point =
 [2.578e-01 1.807e-01 3.460e+00 2.504e+00 1.648e+01 7.162e-01 4.513e-02
 2.607e+00 9.108e-02 1.266e+00 6.428e-01 1.938e-02 4.932e-01 7.398e+00
 1.233e-01 6.103e-01 3.111e-01 6.173e-01 6.954e-01 3.534e-01 6.174e-01
 2.765e-02 6.575e-01 2.526e+00 6.061e-01 2.254e-01 3.410e-01 2.191e+00
 1.103e+00 3.233e-01 3.971e-01 1.873e-01 2.917e-02 1.010e-01 9.190e-04
 6.517e-04 8.218e+00 1.010e+00 1.925e+01 8.511e-02 1.654e+00]
chisq =  78.424, ndf = 39, chisq/NDF =  2.0109, chisq prob = 0.000185

WARNING: chi2 probability for given degrees of freedom less than 0.05 .  Please
```

```
check the data and initial parameter estimates
=== Fitted parameters ===
c = 0.00010599 +- 8.099e-07
m = -1.8313e-07 +- 2.0661e-09
```

[14]:

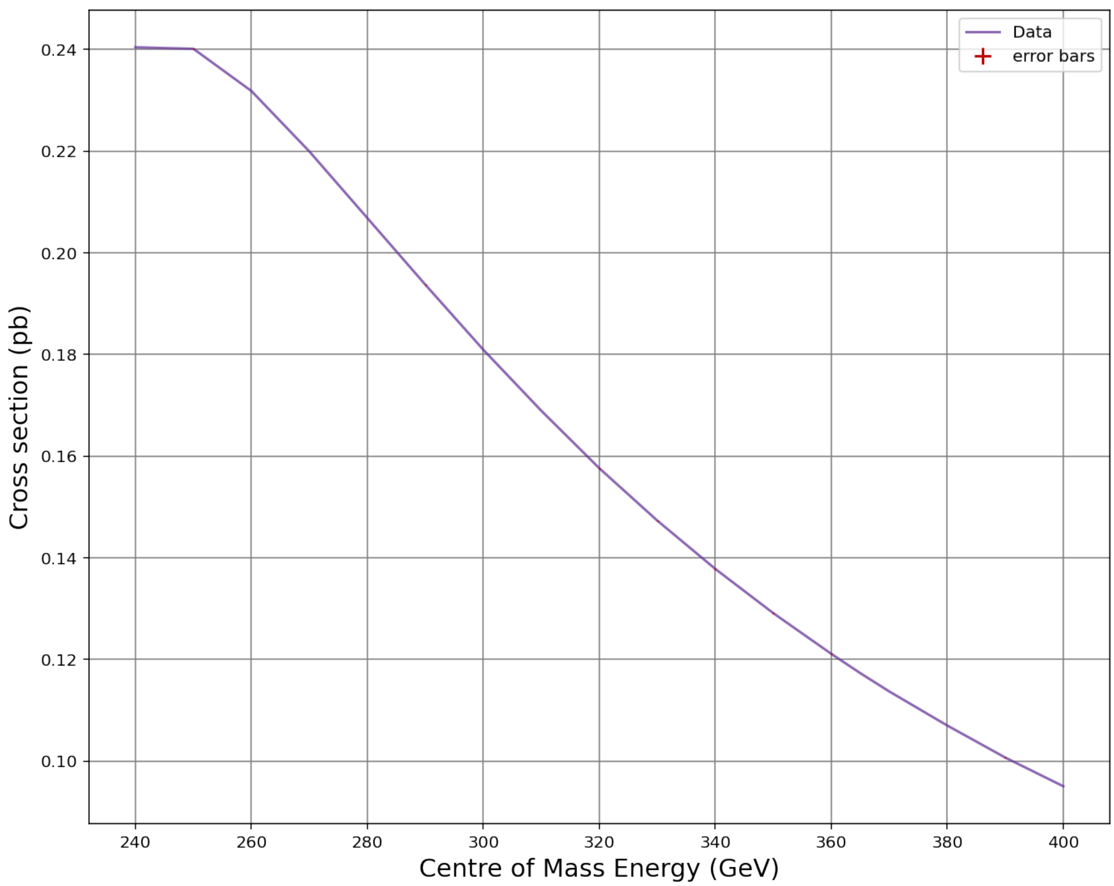


[15]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

zh= np.loadtxt("ee_zh.csv", delimiter = ",")
rows, cols = zh.shape
zh_come = zh[0:,0]
zh_comeerror = zh[0:,1]
zh_data = zh[0:,2]
zh_dataerror = zh[0:,3]
```

```
plt.figure(figsize = (11, 9))
#plt.title("zh")
plt.xlabel("Centre of Mass Energy (GeV)", fontsize = 15)
plt.ylabel("Cross section (pb)", fontsize = 15)
plt.plot(zh_come, zh_data, color = "rebeccapurple",alpha  =  0.75, label =␣
 ↪"Data", linestyle = "-")
plt.errorbar(zh_come, zh_data, xerr = zh_comeerror, yerr = zh_dataerror,  color␣
 ↪= "#BA0001", label = "error bars", linestyle = "")
#plt.ylim(0.00005, 0.1)
plt.grid(color = "grey")
plt.legend()
plt.show()
```

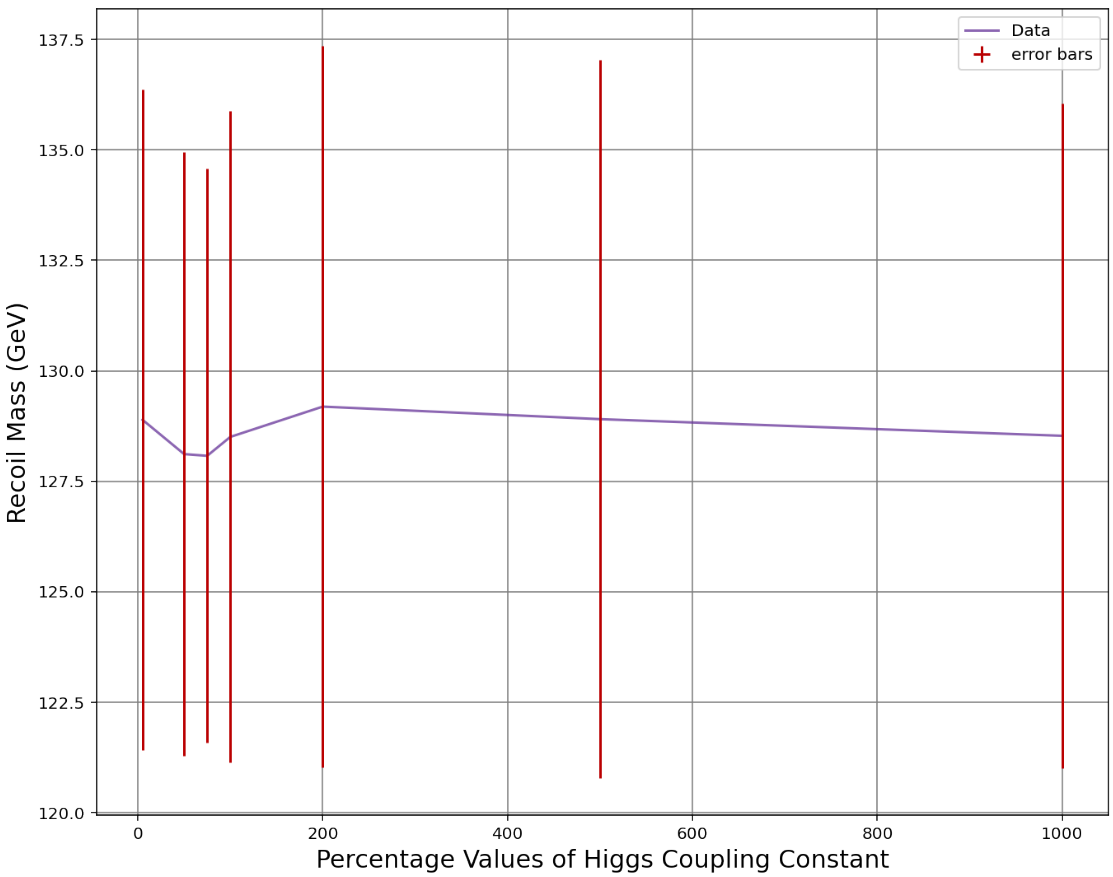
[15]:



[16]:
```
import matplotlib.pyplot as plt
import numpy as np

recoil240 = np.loadtxt("recoil.csv", delimiter = ",")
```

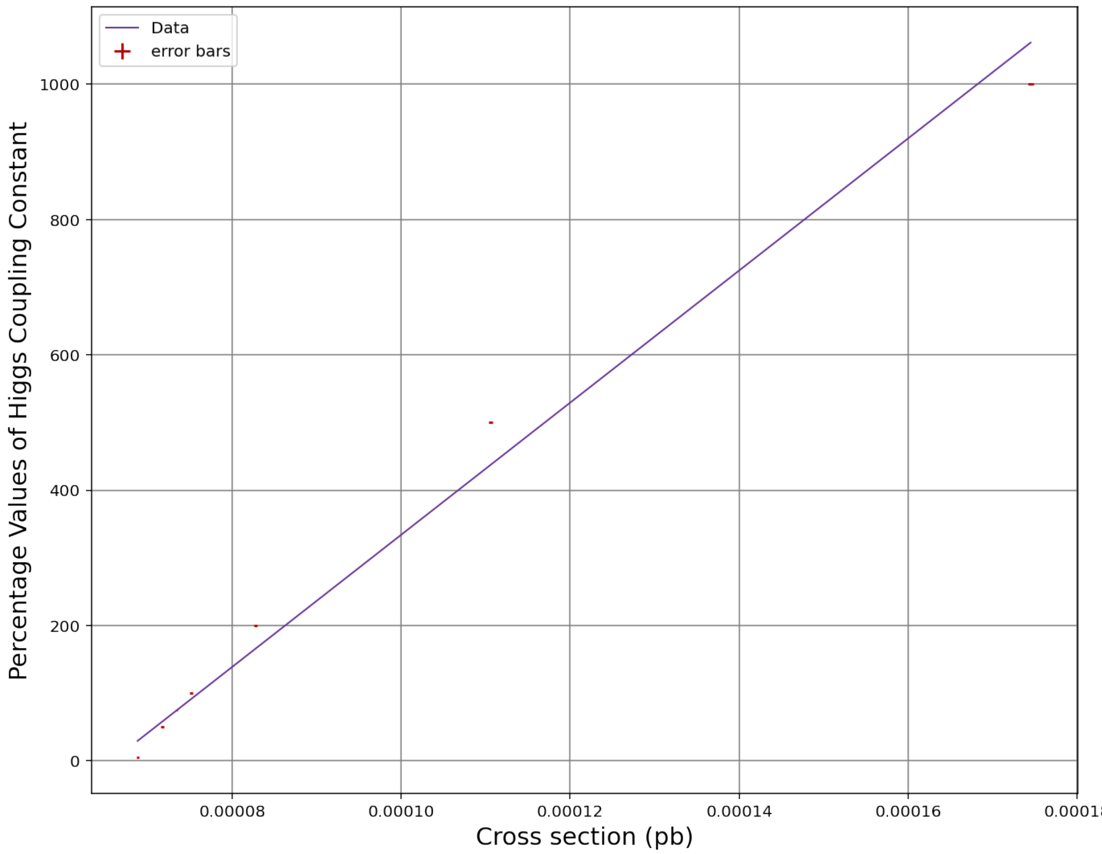
```
rows, cols = recoil240.shape
coupling = recoil240[0:,0]
coupling_error = recoil240[0:,1]
hist = recoil240[0:,2]
hist_error = recoil240[0:,3]

plt.figure(figsize = (11, 9))
#plt.title("?")
plt.xlabel("Percentage Values of Higgs Coupling Constant", fontsize = 15)
plt.ylabel("Recoil Mass (GeV)", fontsize = 15)
plt.plot(coupling, hist, color = "rebeccapurple",alpha  =  0.75, label =␣
↪"Data", linestyle = "-")
plt.errorbar(coupling, hist, xerr = coupling_error, yerr = hist_error,  color =␣
↪"#BA0001", label = "error bars", linestyle = "")
#plt.ylim(0.00005, 0.1)
plt.grid(color = "grey")
plt.legend()
plt.show()
```

[16]:



[5]:
```python
import matplotlib.pyplot as plt
import numpy as np

lum = np.loadtxt("luminosity.csv", delimiter = ",")
rows, cols = lum.shape
cs = lum[0:,0]
cs_error = lum[0:,1]
percent = lum[0:,2]
percent_error = lum[0:,3]

p0 = -642.27
p1 =  9.7645e+06

plt.figure(figsize = (11, 9))
#plt.title("?")
plt.ylabel("Percentage Values of Higgs Coupling Constant", fontsize = 15)
plt.xlabel("Cross section (pb)", fontsize =  15)
plt.plot(cs, (p1*cs+p0), color = "rebeccapurple", label = "Data", linewidth = 1)
plt.errorbar(cs, percent, xerr = cs_error, yerr = percent_error,  color =␣
 ↪"#BA0001", label = "error bars", linestyle = "")
#plt.ylim(0.00005, 0.1)
plt.grid(color = "grey")
plt.legend()
plt.show()
```

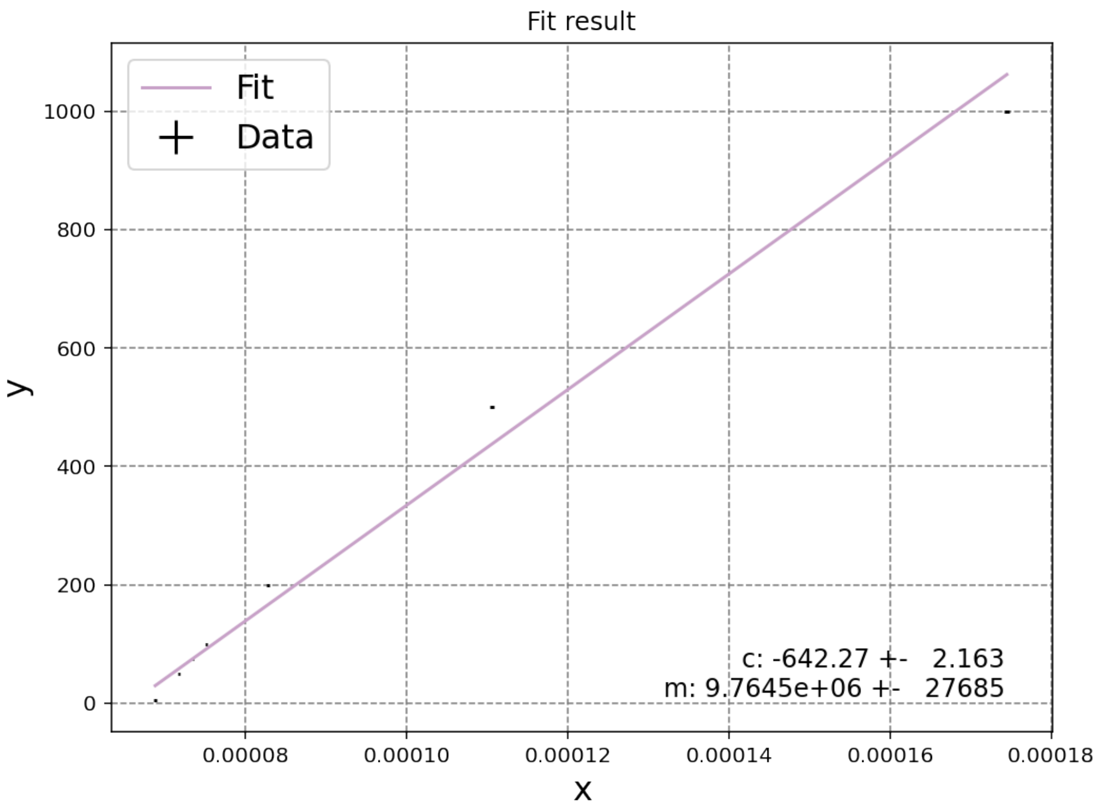
[5]:



[22]:
```
init_params = [-600, 9e6]

#fit data to straight line
params, errors =  fit(cs, percent, cs_error, percent_error, init_params)
```

```
Fit succeeded

=== Fit quality ===
chisq per point =
 [2.444e+02 2.580e+01 5.682e-02 2.233e+01 2.559e+02 6.518e+02 4.145e+02]
chisq =  1614.8, ndf = 5, chisq/NDF =  322.95, chisq prob =       0

WARNING: chi2/ndf suspiciously small or large.  Please check the data and
initial parameter estimates
WARNING: chi2 probability for given degrees of freedom less than 0.05 .  Please
check the data and initial parameter estimates
=== Fitted parameters ===
c = -642.27 +-   2.163
m = 9.7645e+06 +-   27685
```

[22]:



[3]:
```
import numpy as np

x1 = float(input("Enter data value 1"))
x2 = float(input("Enter data value 2"))
error_x1 = float(input("Enter the error on data value 1"))
error_x2 = float(input("Enter the error on data value 2"))

for i in range(-10000, 10000):
    consistent_equation = np.sqrt((error_x1)**2 + (error_x2)**2)
    consistent = abs(x2 - x1)
    consistent_equation *= 3

    if consistent_equation < consistent:
        print("Not consistent")
        print(consistent_equation)
        print(consistent)
        break
    elif consistent_equation > consistent:
        print("Consistent!")
        print(consistent_equation)
        print(consistent)
```

```
        break
```

```
Enter data value 1 126.67Enter data value 2 125.35Enter the error on data value
1 5.13Enter the error on data value 2 0.15Consistent!
15.396577541778562
1.3200000000000074
```

[4]:
```python
import numpy as np

x1 = float(input("Enter data value 1"))
x2 = float(input("Enter data value 2"))
error_x1 = float(input("Enter the error on data value 1"))
error_x2 = float(input("Enter the error on data value 2"))

for i in range(-10000, 10000):
    consistent_equation = np.sqrt((error_x1)**2 + (error_x2)**2)
    consistent = abs(x2 - x1)
    consistent_equation *= 3

    if consistent_equation < consistent:
        print("Not consistent")
        print(consistent_equation)
        print(consistent)
        break
    elif consistent_equation > consistent:
        print("Consistent!")
        print(consistent_equation)
        print(consistent)
        break
```

```
Enter data value 1 126.96Enter data value 2 125.35Enter the error on data value
1 5.9Enter the error on data value 2 0.15Consistent!
17.70571941492353
1.6099999999999994
```