# Auswertung Versuch 232: Michelson-Interferometer

## Teil 1: Bestimmung der Wellenlänge des Lasers

In [31]:

```
#Importieren der benötigten Pakete
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
%matplotlib inline
import numpy as np
from scipy.optimize import curve_fit

sa=1e-3*np.array([1.0175,0.6065,0.6580,0.7280,1.5495]) #Anfangsposition
sa_err=0.007e-3*np.ones(5)
se=1e-3*np.array([3.9735,3.5670,3.6320,3.6700,4.5200]) #Endposition
se_err=0.007e-3*np.ones(5)

m=np.array([11077,11137,11174,11158,11175]) #Anzahl Impulse
m_err=50*np.ones(5)

lam=2*(se-sa)/m #Wellenlänge
lam_sys=(4/(m**2)*(sa_err**2+se_err**2)+(2*(se-sa)*m_err/(m**2))**2)**0.5  #Fehlerfortpflan

lam_mean=np.mean(lam) #Mittelwertbilung
lam_mean_sys=(np.sum(lam_sys**2)/len(lam_sys))**0.5 #syst. Fehler
lam_mean_std=np.std(lam)/np.sqrt(len(lam)) #stat. Fehler

print("Messergebnisse für Lambda [m]:")
print(lam)
print(lam_sys)
print()
print("Mittelwert mit syst. und stat. Fehler:")
print(str(np.mean(lam))+ ' +/- ' + str(lam_mean_sys)+ ' +/- ' +str(lam_mean_std) +' [m]')

lam_lit=532e-9 #Wert aus Anleitung
lam_lit_err=1e-9
```

```
Messergebnisse für Lambda [m]:
[  5.33718516e-07   5.31651253e-07   5.32307142e-07   5.27334648e-07
   5.31633110e-07]
[  2.99978167e-09   2.97617201e-09   2.96867142e-09   2.95508025e-09
   2.96598659e-09]

Mittelwert mit syst. und stat. Fehler:
5.31328933482e-07 +/- 2.97317593359e-09 +/- 9.55358066625e-10 [m]
```

In [32]:

```
#Prüfe auf Signifikanz
diff_1=np.abs(lam_mean-lam_lit)
diff_1_err=(lam_mean_sys**2+lam_lit_err**2)**0.5

diff_2=np.abs(lam_mean-lam_lit)
diff_2_err=(lam_mean_std**2+lam_lit_err**2)**0.5

print()
print("Differenz mit syst. Fehler: "+str(diff_1)+" +/- "+str(diff_1_err)+' [m]')
print("Differenz  mit stat. Fehler: "+str(diff_2)+" +/- "+str(diff_2_err)+' [m]')
```

```
Differenz mit syst. Fehler: 6.71066518036e-10 +/- 3.13684158543e-09 [m]
Differenz  mit stat. Fehler: 6.71066518036e-10 +/- 1.38300724346e-09 [m]
```

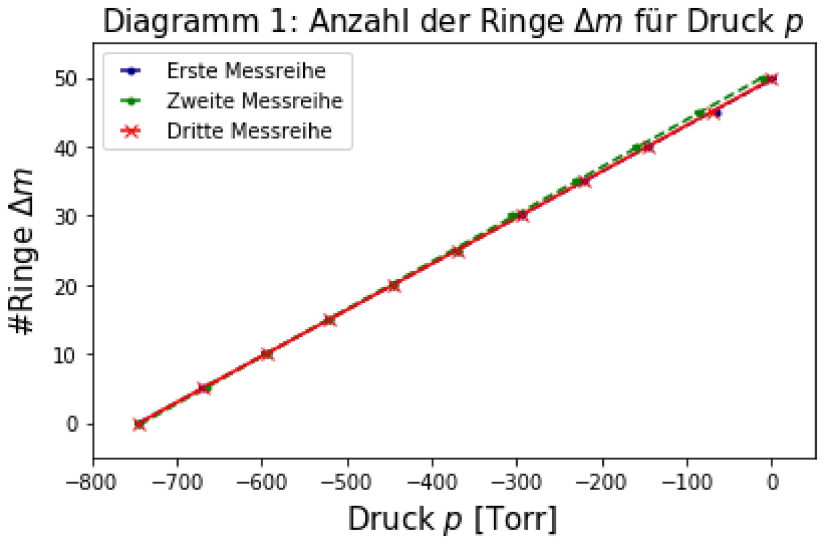
## Teil 2: Bestimmung des Brechungsindex von Luft

In [33]:

```
T_0=273.15
p_0=760 #Normaldruck
T=22.3+273.15 #Umrechnung in Kelvin
T_err=0.1
a=50e-3 #Innenmaß Küvette
a_err=0.05e-3

del_m=np.array([0,5,10,15,20,25,30,35,40,45,50])
p_1=-1*np.array([745,670,595,520,445,370,295,220,145,65,0])
p_2=-1*np.array([745,665,595,520,445,370,305,230,160,85,10])
p_3=-1*np.array([745,670,595,520,445,370,295,220,145,70,0])

p_1_err=5*np.ones(11)
p_2_err=5*np.ones(11)
p_3_err=5*np.ones(11)

plt.xlabel("Druck $p$ [Torr]",size=15)
plt.ylabel("#Ringe $\Delta m$",size=15)
plt.xlim(-800,50)
plt.ylim(-5,55)
plt.title("Diagramm 1: Anzahl der Ringe $\Delta m$ für Druck $p$", size=15)

def linear(x,a,b):
    return a*x+b

popt_1,pcov_1=curve_fit(linear,p_1,del_m)
popt_2,pcov_2=curve_fit(linear,p_2,del_m)
popt_3,pcov_3=curve_fit(linear,p_3,del_m)

plt.plot(p_1,linear(p_1,*popt_1),color="darkblue",linestyle='--')
plt.plot(p_2,linear(p_2,*popt_2),color="green",linestyle='--')
plt.plot(p_3,linear(p_3,*popt_3),color="red")

plt.errorbar(p_1,del_m,xerr=p_1_err,marker='.',color="darkblue",linestyle='',label="Erste M
plt.errorbar(p_2,del_m,xerr=p_2_err,marker='.',color="green",linestyle='',label="Zweite Mes
plt.errorbar(p_3,del_m,xerr=p_3_err,marker='x',color="red",linestyle='',label="Dritte Messr
plt.legend(frameon=True)

plt.tight_layout()
plt.savefig(r"C:\Users\Quirinus\Documents\GitHub\Praktikum\Praktikum\232 - Michelson Interf

n_0_1=lam_lit*p_0*T/(2*a*T_0)*popt_1[0]+1
n_0_2=lam_lit*p_0*T/(2*a*T_0)*popt_2[0]+1
n_0_3=lam_lit*p_0*T/(2*a*T_0)*popt_3[0]+1

popt_1_err=pcov_1[0,0]**0.5
popt_2_err=pcov_2[0,0]**0.5
popt_3_err=pcov_3[0,0]**0.5

n_0_1_err=(n_0_1-1)*np.sqrt((lam_lit_err/lam_lit)**2+(a_err/a)**2+(T_err/T)**2+(popt_1_err/
n_0_2_err=(n_0_2-1)*np.sqrt((lam_lit_err/lam_lit)**2+(a_err/a)**2+(T_err/T)**2+(popt_2_err/
n_0_3_err=(n_0_3-1)*np.sqrt((lam_lit_err/lam_lit)**2+(a_err/a)**2+(T_err/T)**2+(popt_3_err/

#Berechnung Mittelwert & dessen Fehler
n0=np.array([n_0_1,n_0_2,n_0_3])
n0_err=np.array([n_0_1_err,n_0_2_err,n_0_3_err])
n0_mean=np.mean(n0)
n0_mean_syst=(np.sum(n0_err**2)/len(n0_err))**0.5
n0_mean_std=np.std(n0)/np.sqrt(len(n0))
```

```
print("Brechungsindex erste Messung mit Fehler:")
print(str(n_0_1) + ' +/- ' + str(n_0_1_err))

print("Brechungsindex zweite Messung mit Fehler:")
print(str(n_0_2) + ' +/- ' + str(n_0_2_err))

print("Brechungsindex dritte Messung mit Fehler:")
print(str(n_0_3) + ' +/- ' + str(n_0_3_err))

n_0_lit=1.00028 #Literaturwert

diff_1=np.abs(n_0_1-n_0_lit)
diff_2=np.abs(n_0_2-n_0_lit)
diff_3=np.abs(n_0_3-n_0_lit)
#gesuchte Differenz
diff_mean=np.abs(n0_mean-n_0_lit)

print()
print("Differenz erster Messwert mit Literaturwert:")
print(str(diff_1)+" +/- "+str(n_0_1_err)+" ("+str(diff_1/n_0_1_err)+")")
print("Differenz zweiter Messwert mit Literaturwert:")
print(str(diff_2)+" +/- "+str(n_0_2_err)+" ("+str(diff_2/n_0_2_err)+")")
print("Differenz dritter Messwert mit Literaturwert:")
print(str(diff_3)+" +/- "+str(n_0_3_err)+" ("+str(diff_3/n_0_3_err)+")")

print()
print("Mittelwert von n0 mit syst. & stat. Fehler:")
print(str(n0_mean)+ ' +/- '+ str(n0_mean_syst)+ ' +/- '+str(n0_mean_std))
print("Differenz zwischen Mittelwert und Literaturwert:")
print(str(diff_mean)+ ' +/- '+str(n0_mean_syst)+ ' +/- '+str(n0_mean_std))
```

```
Brechungsindex erste Messung mit Fehler:
1.0002917058 +/- 1.0756208396e-06
Brechungsindex zweite Messung mit Fehler:
1.00029986663 +/- 1.40953922362e-06
Brechungsindex dritte Messung mit Fehler:
1.00029243056 +/- 8.12903148649e-07

Differenz erster Messwert mit Literaturwert:
1.17058022207e-05 +/- 1.0756208396e-06 (10.8828332343)
Differenz zweiter Messwert mit Literaturwert:
1.986663413e-05 +/- 1.40953922362e-06 (14.0944173791)
Differenz dritter Messwert mit Literaturwert:
1.24305628835e-05 +/- 8.12903148649e-07 (15.2915669033)

Mittelwert von n0 mit syst. & stat. Fehler:
1.00029466767 +/- 1.12613979928e-06 +/- 2.12933315117e-06
Differenz zwischen Mittelwert und Literaturwert:
1.46676664114e-05 +/- 1.12613979928e-06 +/- 2.12933315117e-06
```




## Teil 3: Messung der Kohärenzlänge einer Leuchtdiode

In [34]:

```
from scipy.signal import argrelextrema
from scipy import signal

#Auslesen der Messdaten
data = np.genfromtxt (r'Interferenz_Daten.csv', delimiter = ",",
                      skip_header =2, skip_footer =0, usecols = (0, 1))
x1 = data[0:, 0]
y1 = data[0:, 1]

#Um 0.02 sehr starkes "Rauschen" der Messwerte, deshalb werden diese Werte beim fitten der
mask1 = (y1>-0.002)

y2 = y1[mask1]
x2 = x1[mask1]

#Bestimmung der lokalen Maxima
mask2 = argrelextrema(y2, np.greater_equal, order = 1)
y3 = y2[mask2]
x3 = x2[mask2]

#Fitten der Gaußkurve
def gaussian(x, mu, sigma, A):
    return A / (sigma * np.sqrt(2 * np.pi)) * np.exp(-(x - mu)**2 / sigma**2 / 2)
p = [0.01, 0.01, 0.01]
popt, pcov = curve_fit(gaussian, x3, y3, p0 = p)
```

In [35]:

```
#Plotten der Messdaten
plt.plot(x1, y1, linewidth = 0.1, color = 'k', linestyle = '--')


#Plotten des Fits
x = np.linspace(x1[0],x1[-1], 1000)
plt.plot(x, gaussian(x, *popt), 'r-', linewidth = 1)
plt.legend(['Messwerte', 'Gauß-Fit'],frameon=True)
plt.xlabel(r'Verfahrdauer [$s$]',size=15)
plt.ylabel('Intensität',size=15)
plt.title('Diagramm 2: Gauß-Fit der Messwerte', size=15)
plt.xlim(-0.03,0.1)

#Berechnung & Plot der Halbwertsbreite
x_hwb=np.linspace(popt[0]-popt[1]*np.sqrt(2 * np.log(2)),popt[0]+popt[1]*np.sqrt(2 * np.log
y_hwb=0.5*gaussian(popt[0],*popt)*np.ones(100)

plt.plot(x_hwb,y_hwb,color='darkblue')
plt.annotate(r'Halbwertsbreite',
             xy=(0.025,y_hwb[0]), xycoords='data',
             xytext=(-120, 25), textcoords='offset points', fontsize=12,
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3, rad=-.2"))

#Markierung der Halbwertsbreite
hwb_marker=0.007
plt.errorbar(x_hwb[0],y_hwb[0],yerr=hwb_marker,color='darkblue')
plt.errorbar(x_hwb[-1],y_hwb[0],yerr=hwb_marker,color='darkblue')

#Allgemeine Berechung für die HWB einer Gaußkurve
halb=2*popt[1]*np.sqrt(2 * np.log(2))
halb_err=2*(pcov[1,1]**0.5)*np.sqrt(2 * np.log(2))
print("Die Halbwertsbreite beträgt:")
print(str(halb) + ' +/- ' + str(halb_err) + ' [s]')

#Umrechnung von Zeit in Länge -> Kohärenzlänge

v_verfahr=0.1e-3 #gemäß Anleitung

koh=halb*v_verfahr
koh_err=halb_err*v_verfahr

print()
print("Die Kohärenzlänge entspricht der Halbwertsbreite, umgerechnet in Längeneinheiten:")
print(str(koh)+ ' +/- '  + str(koh_err) + ' [m]')


#speichern des Graphen
plt.tight_layout()
plt.savefig('Diagramme/V232Diagramm2.pdf', format='PDF')
```

```
Die Halbwertsbreite beträgt:
0.0349433661189 +/- 7.9509729811e-05 [s]

Die Kohärenzlänge entspricht der Halbwertsbreite, umgerechnet in Längeneinhe
iten:
3.49433661189e-06 +/- 7.9509729811e-09 [m]
```

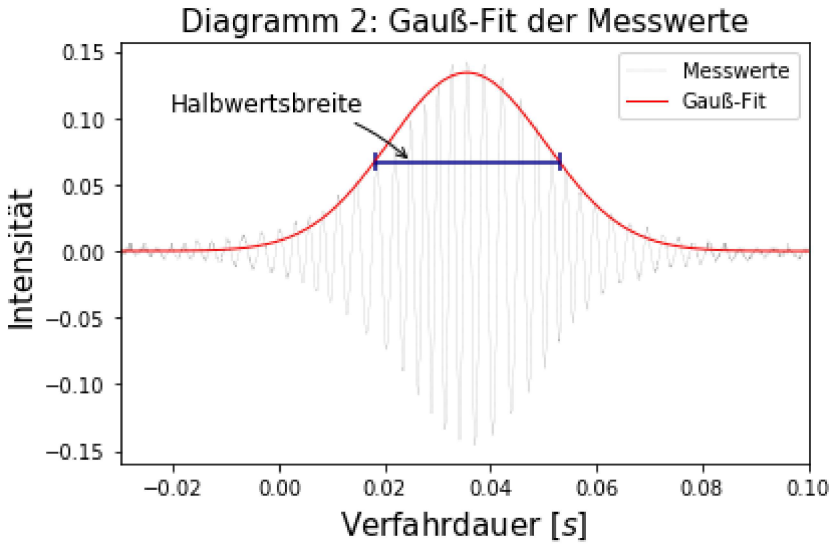


Diagramm 2: Gauß-Fit der Messwerte
Messwerte
Gauß-Fit
Halbwertsbreite
Intensität
Verfahrdauer [s]
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.02
0.00
0.02
0.04
0.06
0.08
0.10

In [ ]: