In [2]:
```
import math
import matplotlib.pyplot as plt
import numpy as np
import scipy as sp
```

Wir wollen eine Funktion f lernen, von der wir annehmen, dass sie eine Überlagerung von Cosinus-Schwingungen zwischen 0 Hz und 10 Hz ist, also

$$f(x) = \lambda_0 + \sum_{k=1}^{10} \lambda_k \cos(kx).$$

Wir wissen nicht, welche Frequenzen k tatsächlich vorhanden sind und wie groß deren Amplitude λ_k ist.

Wir haben für $n+1$ Stellen $-\pi = x_0 < x_1 < x_2 < \cdots < x_n = \pi$ jeweils einen Messwert y_i von f, der aber durch leichtes gleichförmiges Rauschen überlagert ist, also

$$y_i = f(x_i) + r(x_i)$$

mit einer Rauschfunktion $r(x)$.

In [3]:
```
y_data = [-16.905934248183623,-16.744088965937685,-17.16373674978355,-16.9403837
```

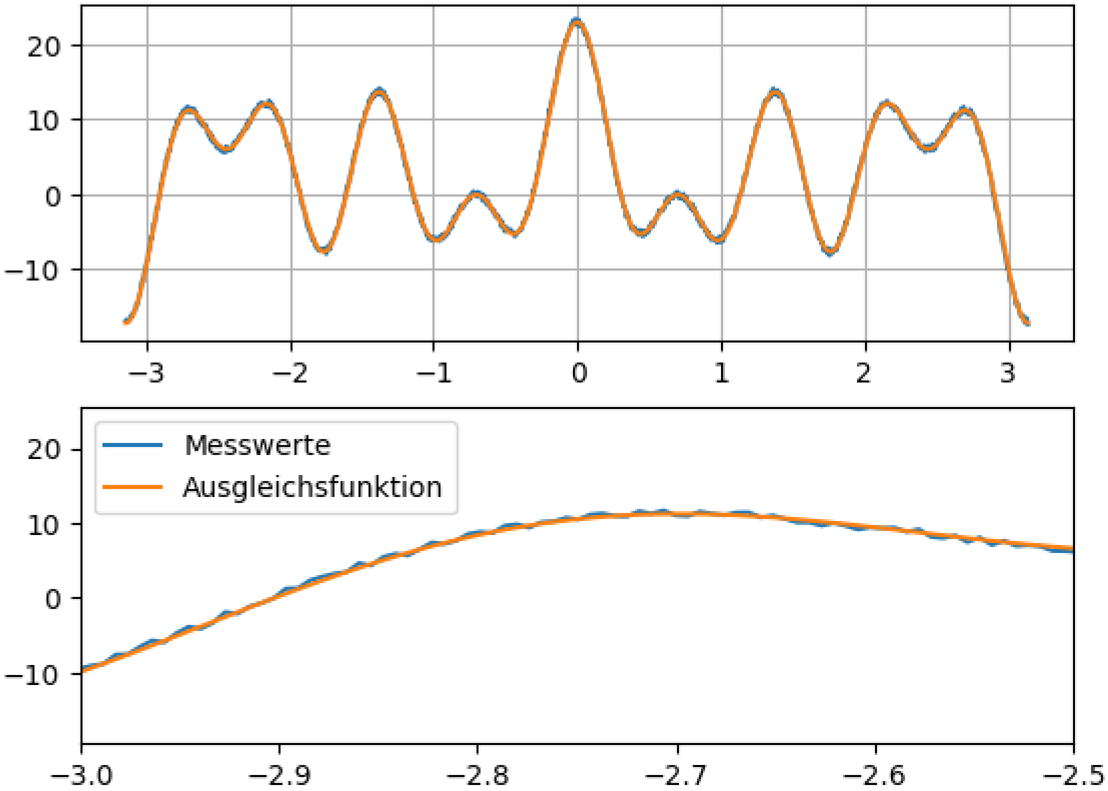
(a) Auf der Homepage der Vorlesung finden Sie einen Link zu einem Java-Quelltextfragment, das die Werte für y_i in einem Java-Array definiert ($n = 1024$). Die zugehörigen Stützstellen sind

$$x_i = -\pi + \frac{2\pi i}{n}, \quad i = 0, \ldots, n.$$

Schätzen Sie mithilfe der linearen Ausgleichsrechnung die Parameter $\lambda_0, \ldots, \lambda_{10}$ und geben Sie diese Werte an.

In [6]:
```
# 'y_data' data set hidden in exported notebook

data_size = len(y_data)
x_data = [-math.pi + ((2*math.pi*i) / data_size) for i in range(data_size)]


def model_cos (x, l0, l1, l2, l3, l4, l5, l6, l7, l8, l9 ,l10):
    return l0+l1*np.cos(x)+l2*np.cos(x*2)+l3*np.cos(x*3)+l4*np.cos(x*4)+l5*np.co
           +l6*np.cos(x*6)+l7*np.cos(x*7)+l8*np.cos(x*8)+l9*np.cos(x*9)+l10*np.c

reg = sp.optimize.curve_fit(model_cos, x_data, y_data)
l_fitted = [reg[0][i] for i in range (11)]
y_fitted = [model_cos(i, l_fitted[0],l_fitted[1],l_fitted[2],l_fitted[3],l_fitte
                      l_fitted[6],l_fitted[7],l_fitted[8],l_fitted[9],l_fitted[1

print("l0,...,l10= ", l_fitted)

fig, (ax1, ax2) = plt.subplots(2)
ax1.plot(x_data, y_data)
ax1.plot(x_data, y_fitted)
ax2.plot(x_data, y_data, label="Messwerte")
ax2.plot(x_data, y_fitted, label="Ausgleichsfunktion")
ax2.set_xlim(-3,-2.5)
ax1.grid()
ax2.grid()
plt.legend()
plt.show()
```

```
l0,...,l10=  [2.9724824145214326, 0.017887956285536855, -0.05152206120341085, 5.0
38762174399967, -0.039768217332452505, 8.009179069221794, 0.007348135592907479,
0.041965362360981806, -0.05404881946063256, 6.984182645732481, 0.0514826347620364
06]
```

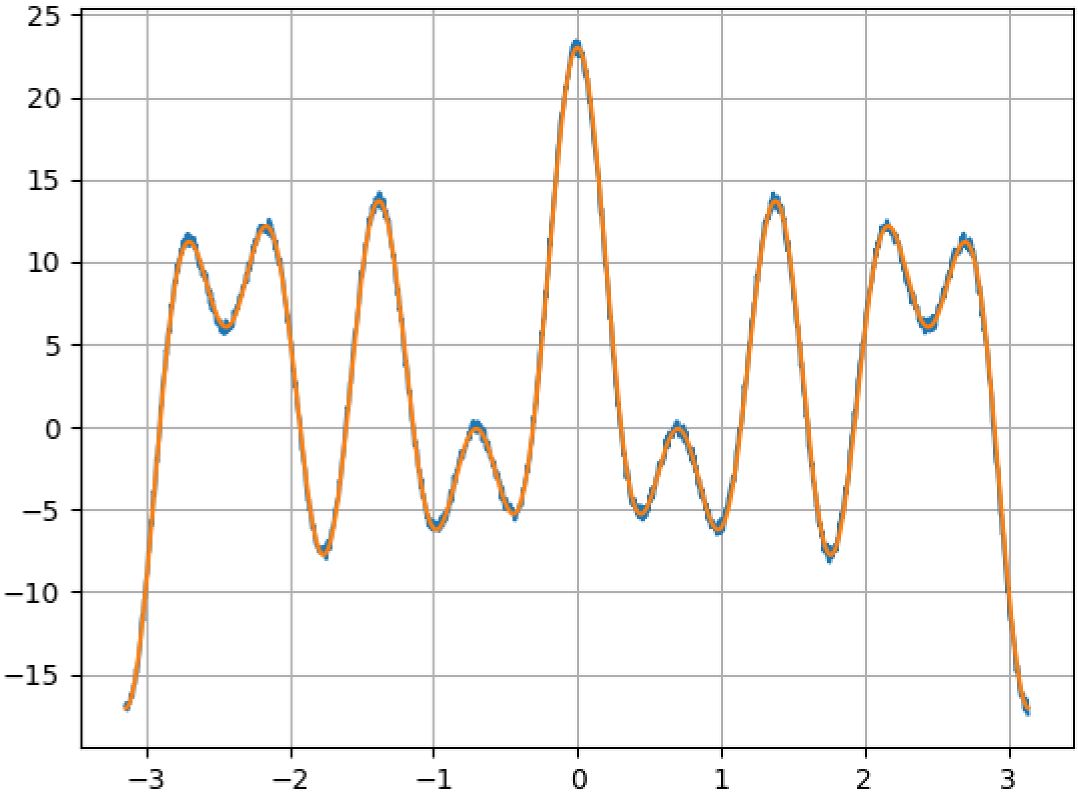


(b) Wenn Sie sich die Schätzung für die λ_k anschauen, werden Sie feststellen, dass einige recht nahe bei Null liegen. Wir können daher davon ausgehen, dass diese Frequenzen in der Funktion f tatsächlich nicht auftreten.

Entscheiden Sie auf Basis der Parameterschätzung von (a), welche Frequenzen in f tatsächlich vorhanden sind und berechnen Sie auf dieser Basis die Parameter λ_k neu.

In [7]:
```
def model2_cos(x, l0, l3, l5, l9):
    return l0+l3*np.cos(x*3)+l5*np.cos(x*5)+l9*np.cos(x*9)

reg2 = sp.optimize.curve_fit(model2_cos, x_data, y_data)
l2_fitted = [l_fitted[0], l_fitted[3], l_fitted[5], l_fitted[9]]
y2_fitted = [model2_cos(i, l2_fitted[0],l2_fitted[1],l2_fitted[2],l2_fitted[3])

print("Vorhandene Frequenzen: 0HZ:", l2_fitted[0], ", 3HZ:", l2_fitted[1], ", 5H
plt.plot(x_data, y_data)
plt.plot(x_data, y2_fitted)
plt.grid()
plt.show()
```

```
Vorhandene Frequenzen: 0HZ: 2.9724824145214326 , 3HZ: 5.038762174399967 , 5HZ: 8.
009179069221794 , 9HZ: 6.984182645732481
```


25
20
15
10
5
0
−5
−10
−15
−3
−2
−1
0
1
2
3