

# Blatt 9

Vincent Kümmerle und Elvis Gnaglo

19. Dezember 2025

## 1 Fitten von Funktionen

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.optimize import curve_fit
4
5 # Daten mit delimiter "," laden
6 try:
7     data = np.loadtxt('C:/Studium/5. Semester/AC II lab/Protokolle/
8         CGL/Blatt 9/data.txt', delimiter=',')
9     x_data = data[:, 0]
10    y_data = data[:, 1]
11 except Exception as e:
12     print(f"Fehler: {e}")
13     # Fallback simulation, damit der Code hier läuft
14     x_data = np.linspace(0, 3, 300)
15     y_data = 1.0 * np.exp(-0.8 * x_data) * np.sin(10 * x_data) + np.
16         random.normal(0, 0.1, 300)
17
18 # Fitfunktion definieren
19 def damped_sine(x, amplitude, decay, omega, phase, offset):
20     """
21     amplitude: Start-Höhe der Welle
22     decay:      Wie schnell die Welle abklingt (Dämpfung)
23     omega:      Kreisfrequenz (bestimmt den Abstand der Wellenberge)
24     phase:      Verschiebung nach links/rechts
25     offset:     Verschiebung nach oben/unten
26     """
27     return amplitude * np.exp(-decay * x) * np.sin(omega * x + phase)
28     + offset
29
30 p0_guess = [1.0, 0.8, 10.0, 0.0, 0.0]
```

```

29 try:
30     params, covariance = curve_fit(damped_sine, x_data, y_data, p0=
        p0_guess)
31     # Parameter ausgeben
32     labels = ["Amplitude", "Decay (Dämpfung)", "Omega (Frequenz)", "
        Phase", "Offset"]
33     print("Gefundene Parameter:")
34     for label, val in zip(labels, params):
35         print(f" {label}: {val:.4f}")
36
37     # Plotten
38     plt.figure(figsize=(10, 6))
39
40     # Messdaten und Fit-Kurve
41     plt.scatter(x_data, y_data, label='Messdaten', color='black',
        alpha=0.5, s=15)
42     x_fit = np.linspace(min(x_data), max(x_data), 1000)
43     y_fit = damped_sine(x_fit, *params)
44
45     plt.plot(x_fit, y_fit, 'r-', linewidth=2, label='Fit: Gedämpfte
        Schwingung')
46     plt.xlabel('x')
47     plt.ylabel('y')
48     plt.savefig('fit_plot.pdf')
49     plt.show()
50
51 except RuntimeError:
52     print("Der Fit hat nicht konvergiert. Versuche, die p0_guess
        Werte anzupassen.")

```

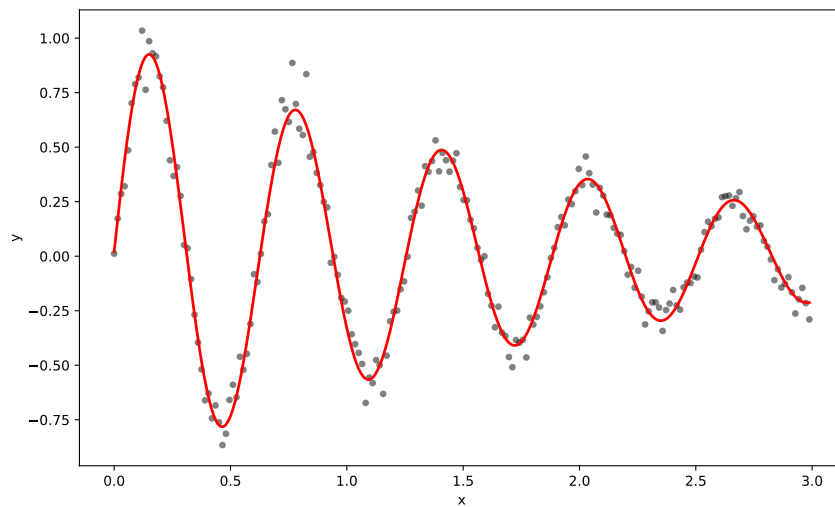


Abbildung 1: Fitfunktion der gegebenen Messdaten aus data.txt

## 2 SymPy

```

1 import sympy as sp
2 import numpy as np
3 import matplotlib.pyplot as plt
4 # ---Symbolische Definition---
5 t = sp.Symbol('t', real=True)
6 A = sp.Symbol('A', real=True)
7 lam = sp.Symbol('lambda', positive=True, real=True)
8 omega = sp.Symbol('omega', positive=True, real=True)
9
10 f = A * sp.exp(-lam * t) * sp.cos(omega * t)
11
12 print("--- Symbolische Analyse ---")
13 print(f"Funktion f(t): {f}")
14
15 # ---Analysis---
16 # Erste Ableitung
17 f_prime = sp.diff(f, t)
18 print(f"Ableitung f'(t): {f_prime}")
19
20 # Unbestimmtes Integral
21 f_int = sp.integrate(f, t)
22 print(f"Stammfunktion F(t): {f_int}")
23
24 # Grenzwert für t -> unendlich
25 limit_inf = sp.limit(f, t, sp.oo)

```

```

26 print(f"Grenzwert für t->oo: {limit_inf}")
27
28 # ---Numerische Konvertierung---
29 # Erstellen von Python-Funktionen, die mit NumPy-Arrays arbeiten können und Definition der Variablen
30 func_f = sp.lambdify((t, A, lam, omega), f, modules='numpy')
31 func_f_prime = sp.lambdify((t, A, lam, omega), f_prime, modules='numpy')
32
33 # ---Visualisierung---
34 # Festlegung Parameterwerte und Zeitbereich
35 A_val = 2
36 lam_val = 0.5
37 omega_val = 3
38 t_vals = np.linspace(0, 10, 500)
39
40 # Berechnung der y-Werte durch Aufruf der erstellten Funktionen
41 y_vals = func_f(t_vals, A_val, lam_val, omega_val)
42 y_prime_vals = func_f_prime(t_vals, A_val, lam_val, omega_val)
43
44 plt.figure(figsize=(10, 6))
45
46 plt.plot(t_vals, y_vals, label=r'$f(t) = A \cdot e^{-\lambda t} \cdot \cos(\omega t)$', color='blue')
47 plt.plot(t_vals, y_prime_vals, label=r"$f'(t)$ (Ableitung)", color='orange', linestyle='--')
48
49 plt.xlabel('Zeit t')
50 plt.ylabel('Amplitude')
51 plt.legend(loc='upper right')
52 plt.savefig('Sympy.pdf')
53 plt.show()

```

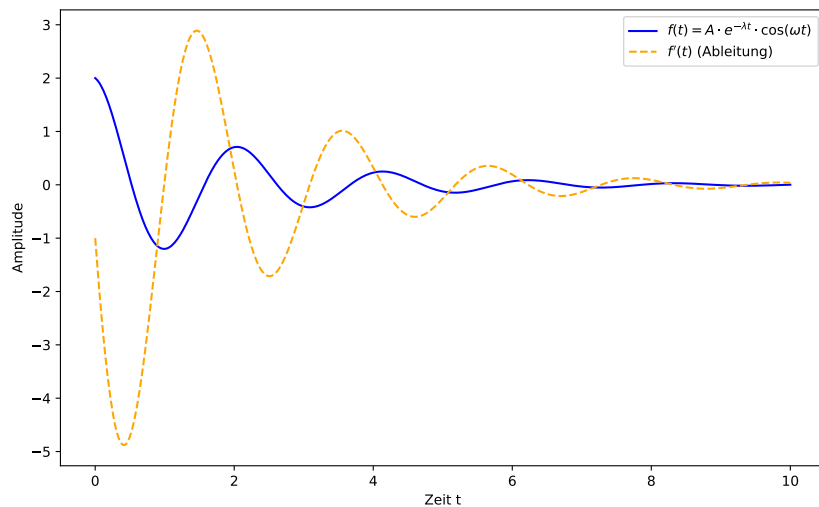


Abbildung 2: Plot der Funktion und ihrer Ableitung

### 3 Primzahlen

```

1 N = int(input("N eingeben: "))
2 gestrichen = [False] * (N + 1) # False = noch Primzahl
3
4 # Jede zusammengesetzte Zahl j=a*b hat mindestens einen Faktor
  kleiner gleich sqrt(j)
5 for i in range(2, int(N**0.5) + 1): # Hat eine Zahl keinen Teiler
  kleiner gleich sqrt(), dann ist sie prim
6     if not gestrichen[i]: # i ist Primzahl
7         print(i, end=", ")
8         for j in range(i*i, N + 1, i): # Vielfache streichen
9             gestrichen[j] = True
10
11 # Restliche Primzahlen ausgeben:
12 for i in range(int(N**0.5) + 1, N + 1):
13     if not gestrichen[i]:
14         print(i, end=", ")

```