

# 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/
    CGL/Blatt 9/data.txt', delimiter=',')
8     x_data = data[:, 0]
9     y_data = data[:, 1]
10 except Exception as e:
11     print(f" Fehler: {e}")
12 # Fallback simulation, damit der Code hier läuft
13 x_data = np.linspace(0, 3, 300)
14 y_data = 1.0 * np.exp(-0.8 * x_data) * np.sin(10 * x_data) + np.
    random.normal(0, 0.1, 300)
15
16 # Fitfunktion definieren
17 def damped_sine(x, amplitude, decay, omega, phase, offset):
18     """
19         amplitude: Start-Höhe der Welle
20         decay: Wie schnell die Welle abklingt (Dämpfung)
21         omega: Kreisfrequenz (bestimmt den Abstand der Wellenberge)
22         phase: Verschiebung nach links/rechts
23         offset: Verschiebung nach oben/unten
24     """
25     return amplitude * np.exp(-decay * x) * np.sin(omega * x + phase)
26             + offset
27
28 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=
31         p0_guess)
32     # Parameter ausgeben
33     labels = ["Amplitude", "Decay (Dämpfung)", "Omega (Frequenz)", "
34         Phase", "Offset"]
35     print("Gefundene Parameter:")
36     for label, val in zip(labels, params):
37         print(f" {label}: {val:.4f}")
38
39     # Plotten
40     plt.figure(figsize=(10, 6))
41
42     # Messdaten und Fit-Kurve
43     plt.scatter(x_data, y_data, label='Messdaten', color='black',
44                 alpha=0.5, s=15)
45     x_fit = np.linspace(min(x_data), max(x_data), 1000)
46     y_fit = damped_sine(x_fit, *params)
47
48     plt.plot(x_fit, y_fit, 'r-', linewidth=2, label='Fit: Gedämpfte
49         Schwingung')
50     plt.xlabel('x')
51     plt.ylabel('y')
52     plt.savefig('fit_plot.pdf')
53     plt.show()
54
55 except RuntimeError:
56     print("Der Fit hat nicht konvergiert. Versuche, die p0_guess
57         Werte anzupassen.")

```

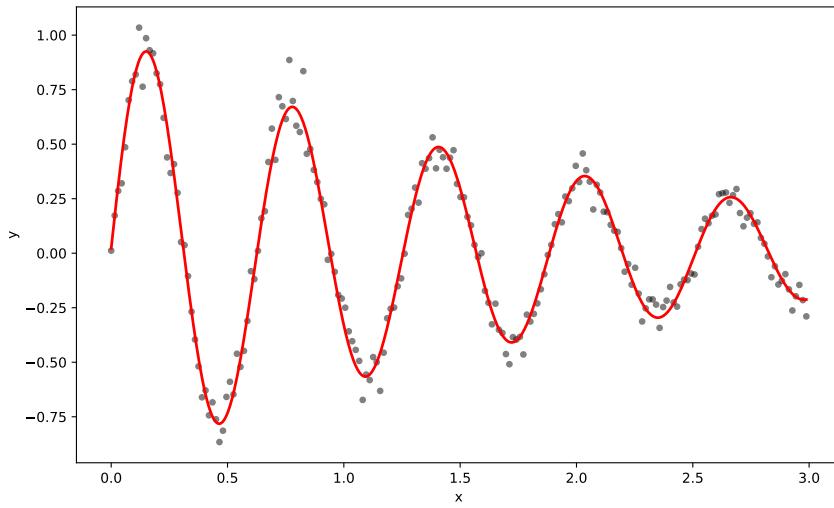


Abbildung 1: Fitfunktion der gegebenen Messdaten aus data.txt

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

```

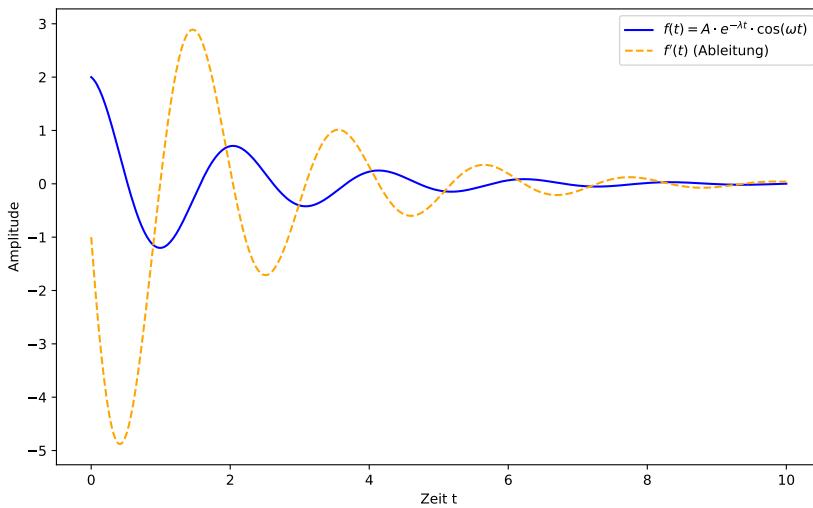


Abbildung 2: Plot der Funktion und ihrer Ableitung

### 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
5     kleiner gleich sqrt(j)
6 for i in range(2, int(N**0.5) + 1): # Hat eine Zahl keinen Teiler
7     kleiner gleich sqrt(), dann ist sie prim
8     if not gestrichen[i]: # i ist Primzahl
9         print(i, end=", ")
10    for j in range(i*i, N + 1, i): # Vielfache streichen
11        gestrichen[j] = True
12
13 # Restliche Primzahlen ausgeben:
14 for i in range(int(N**0.5) + 1, N + 1):
15     if not gestrichen[i]:
16         print(i, end=", ")

```