Blatt 9

Vincent Kümmerle und Elvis Gnaglo

17. Dezember 2025

1 Fitten von Funktionen

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Daten mit delimiter , laden
try:
    data = np.loadtxt('C:/Studium/5. Semester/AC II lab/Protokolle/CGL/Blatt 9/data.txt', delimiter=',')
    x_data = data[:, 0]
    y_data = data[:, 1]
except Exception as e:
    print(f"Fehler: {e}")
    # Fallback simulation, damit der Code hier läuft
    x_data = np.linspace(0, 3, 300)
    y_data = 1.0 * np.exp(-0.8 * x_data) * np.sin(10 * x_data) + np.random.normal(0, 0.1, 300)


# Fitfunktion definieren
def damped_sine(x, amplitude, decay, omega, phase, offset):
    """
    amplitude: Start-Höhe der Welle
    decay:     Wie schnell die Welle abklingt (Dämpfung)
    omega:     Kreisfrequenz (bestimmt den Abstand der Wellenberge)
    phase:     Verschiebung nach links/rechts
    offset:    Verschiebung nach oben/unten
    """
    return amplitude * np.exp(-decay * x) * np.sin(omega * x + phase) + offset


```

```
p0_guess = [1.0, 0.8, 10.0, 0.0, 0.0]

try:
    params, covariance = curve_fit(damped_sine, x_data, y_data, p0=
        p0_guess)

    # Parameter ausgeben
    labels = ["Amplitude", "Decay (Dämpfung)", "Omega (Frequenz)", "
        Phase", "Offset"]
    print("Gefundene Parameter:")
    for label, val in zip(labels, params):
        print(f"  {label}: {val:.4f}")

    # Plotten
    plt.figure(figsize=(10, 6))

    # Messdaten
    plt.scatter(x_data, y_data, label='Messdaten', color='black',
        alpha=0.5, s=15)

    # Fit-Kurve
    x_fit = np.linspace(min(x_data), max(x_data), 1000)
    y_fit = damped_sine(x_fit, *params)

    plt.plot(x_fit, y_fit, 'r-', linewidth=2, label='Fit: Gedämpfte
        Schwingung')
    # Plot anzeigen und speichern
    plt.xlabel('x')
    plt.ylabel('y')
    plt.savefig('fit_plot.pdf')
    plt.show()

except RuntimeError:
    print("Der Fit hat nicht konvergiert. Versuche, die p0_guess
        Werte anzupassen.")
```

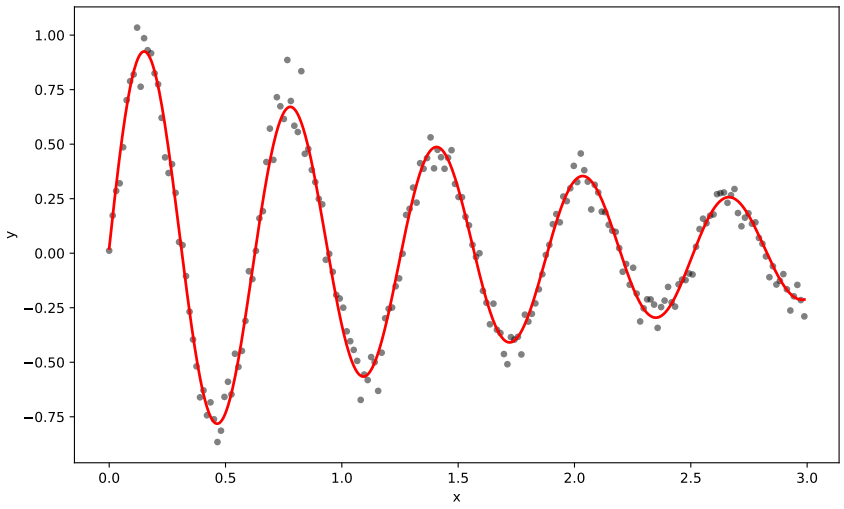

Abbildung 1: Fitfunktion der gegebenen Messdaten aus data.txt

2 SymPy

```python
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
#--------Definition-----------
# Definition der Symbole
t = sp.Symbol('t', real=True)
A = sp.Symbol('A', real=True)
lam = sp.Symbol('lambda', positive=True, real=True)
omega = sp.Symbol('omega', positive=True, real=True)

# Definition der Funktion
f = A * sp.exp(-lam * t) * sp.cos(omega * t)

print("--- Symbolische Analyse ---")
print(f"Funktion f(t): {f}")

#------Analysis------
# Erste Ableitung
f_prime = sp.diff(f, t)
print(f"Ableitung f'(t): {f_prime}")

# Unbestimmtes Integral
f_int = sp.integrate(f, t)
print(f"Stammfunktion F(t): {f_int}")

```

```
# Grenzwert für t -> unendlich
limit_inf = sp.limit(f, t, sp.oo)
print(f"Grenzwert für t->oo: {limit_inf}")

# ---Numerische Konvertierung---
# Erstellen von Python-Funktionen die mit NumPy-Arrays arbeiten kö
    nnen und definieren der Variablen
func_f = sp.lambdify((t, A, lam, omega), f, modules='numpy')
func_f_prime = sp.lambdify((t, A, lam, omega), f_prime, modules='
    numpy')

# ---Visualisierung---
# Festlegung der Parameterwerte
A_val = 2
lam_val = 0.5
omega_val = 3

# Festlegung des Zeitbereichs
t_vals = np.linspace(0, 10, 500)

# Berechnung der y-Werte durch Aufruf der erstellten Funktionen
y_vals = func_f(t_vals, A_val, lam_val, omega_val)
y_prime_vals = func_f_prime(t_vals, A_val, lam_val, omega_val)

plt.figure(figsize=(10, 6))

plt.plot(t_vals, y_vals, label=r'$f(t) = A \cdot e^{-\lambda t} \cdot
     \cos(\omega t)$', color='blue')
plt.plot(t_vals, y_prime_vals, label=r"$f'(t)$ (Ableitung)", color='
    orange', linestyle='--')

#plt.title('Analyse einer gedämpften Schwingung')
plt.xlabel('Zeit t')
plt.ylabel('Amplitude')
plt.legend(loc='upper right')
#plt.grid(True, alpha=0.5)
plt.savefig('Sympy.pdf')
plt.show()
```

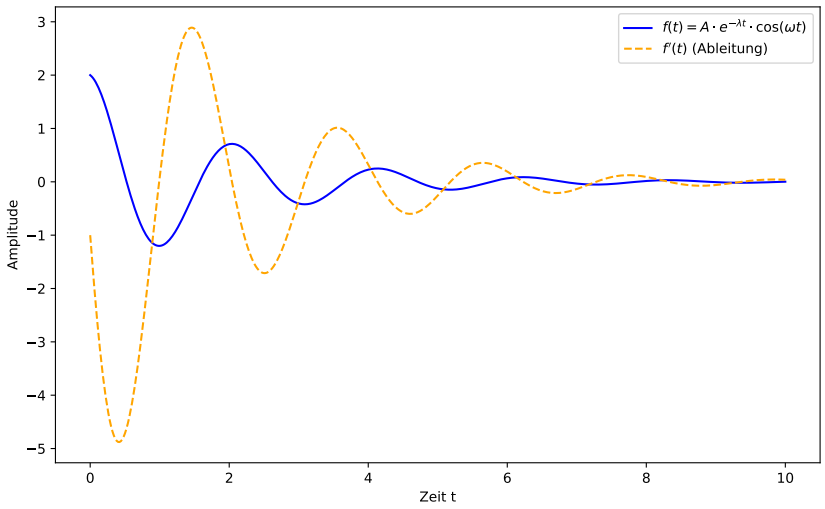

Abbildung 2: Plot der Funktion und ihrer Ableitung

3 Primzahlen

```python
N = int(input("N eingeben: "))
gestrichen = [False] * (N + 1) # False = noch Primzahl

# Jede zusammengesetzte Zahl j=a*b hat mindestens einen Faktor kleiner gleich sqrt(j)
for i in range(2, int(N**0.5) + 1): # Hat eine Zahl keinen Teiler kleiner gleich sqrt(), dann ist sie prim
    if not gestrichen[i]: # i ist Primzahl
        print(i, end=", ")
        for j in range(i*i, N + 1, i): # Vielfache streichen
            gestrichen[j] = True

# Restliche Primzahlen ausgeben:
for i in range(int(N**0.5) + 1, N + 1):
    if not gestrichen[i]:
        print(i, end=", ")
```