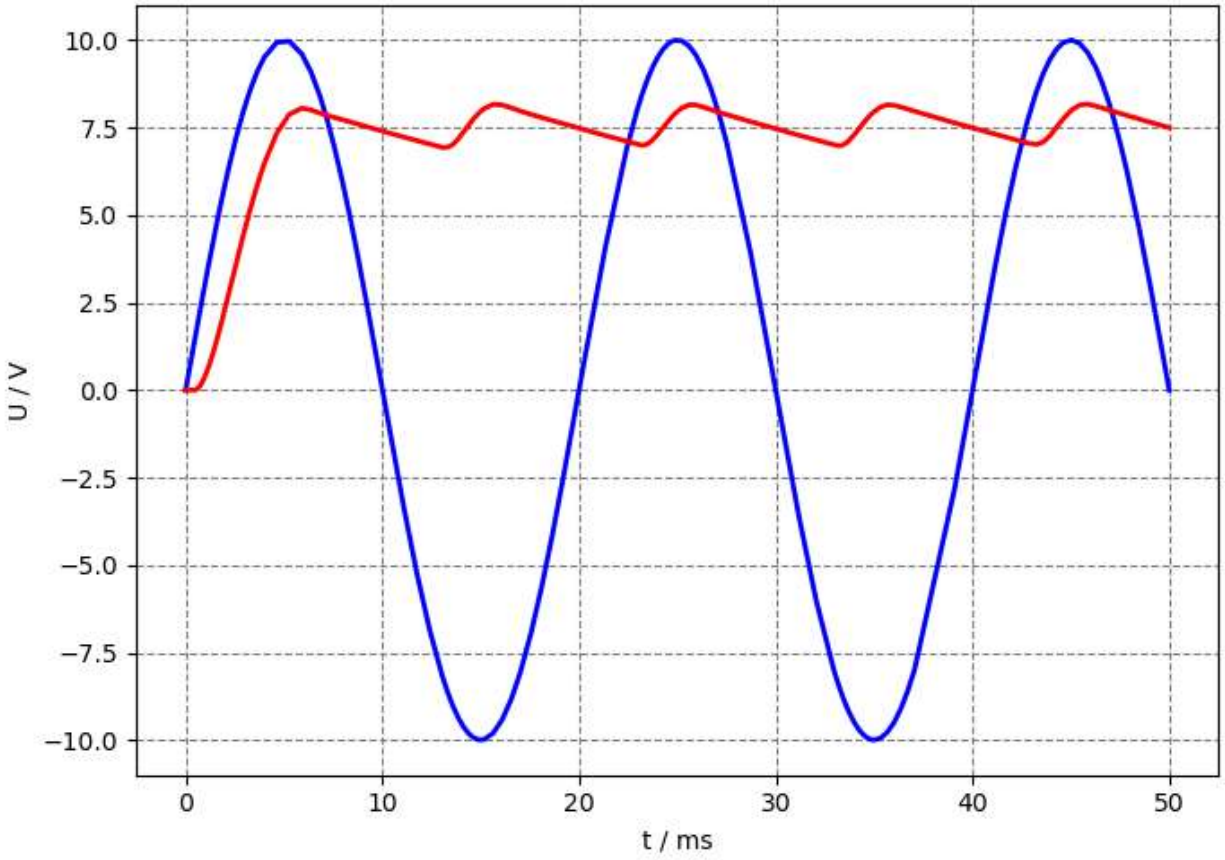
Simulation einer B2-Gleichrichterschaltung.

```
"""Unabhängig von der Polarität der Eingangsspannung ue(t) befinden sich
stets zwei Dioden in Durchlassrichtung. Der Glättungskondensator am Ausgang
des Gleichrichters wird daher in jeder Periode der Eingangswechselspannung
zwei mal wieder aufgeladen: Immer dann, wenn |ue(t)| die Kondensatorspannung
ua(t) um mehr als zwei Dioden-Schwellenspannungen US überschreitet.

Für weitere Details zur Schaltungssimulation mit Python siehe:
https://kuepper.userweb.mwn.de/elektronik/formelsammlung-elektronik.pdf

Tilman Küpper, https://kuepper.userweb.mwn.de, 2022-06-01
"""

from scipy.integrate import solve_ivp
from numpy import pi, sin
import matplotlib.pyplot as plt

# pylint: disable=invalid-name

C = 0.000470    # Glättungskapazität (in F)
RA = 100.0      # Lastwiderstand (in Ohm)
US = 0.7        # Schwellenspannung Diode (in V)
RF = 1.0        # diff. Widerstand Diode (in Ohm)
AMPL = 10.0     # Amplitude (in V)
FREQ = 50.0     # Frequenz (in Hz)
T_STOP = 0.05   # Simulation läuft bis zu diesem Zeitpunkt
UA0 = 0.0       # Startwert für die Ausgangsspannung


def ue(t):
    """Berechne die Eingangsspannung zum Zeitpunkt t."""
    return AMPL * sin(2.0 * pi * FREQ * t)


def b2_dgl(t, ua):
    """Differentialgleichung zur Berechnung der Ausgangsspannung."""
    ue_b = abs(ue(t))
    if ue_b - 2 * US <= ua:
        # Entladephase
        ddt_ua = -ua / (RA * C)
    else:
        # Aufladephase
        ddt_ua = -ua / (RA * C) + (ue_b - 2 * US - ua) / (2 * RF * C)
    return ddt_ua


def main():
    """Berechne die Ausgangsspannung, plotte die Ergebnisse."""
    ms = 0.1 / FREQ

    # Das BDF-Lösungsverfahren erkennt die Übergänge zwischen Lade-
    # und Entladephasen recht sicher und passt in diesen Bereichen die
    # Schrittweite bei der numerischen DGL-Berechnung entsprechend an.
    # Per max_step wird zusätzlich dafür gesorgt, dass in jedem Fall
    # mindestens zehn Zeitschritte pro Periode berechnet werden.
    sol = solve_ivp(b2_dgl, [0, T_STOP], [UA0], method="BDF", max_step=ms)

    plt.plot(sol.t * 1000, ue(sol.t), "b-", linewidth=2)
    plt.plot(sol.t * 1000, sol.y[0], "r-", linewidth=2)
    plt.grid(True, color="gray", linestyle="dashed")
    plt.xlabel("t / ms")
    plt.ylabel("U / V")
    plt.show()


# Berechne die Ausgangsspannung, plotte die Ergebnisse.
if __name__ == "__main__":
    main()
```


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
U / V
0
10
20
30
40
50
t / ms