

```
#!/usr/bin/env python3
"""
CEA QuTiP-Simulation
Circle-Exponential Equivalence (Hirschl 2026)
 $T/\tau = \pi/e$  bzw. 52/45
 $Q = e$  (Artikelkonvention:  $Q = \omega/(2\gamma_{\text{amp}}) = \omega/\kappa_{\text{Energie}}$ )

Quantenoptisches Modell: gedämpfte Kavität / Quanten-Oszillator
 $H = \omega a^\dagger a$ 
 $c_{\text{ops}} = [\sqrt{\kappa} a] \rightarrow \text{Amplitude} \sim e^{-\kappa t / 2}, \text{Energie} \sim e^{-\kappa t}$ 

Artikel:
 $x(t) \propto e^{-\gamma t} \cos(\omega t), \tau = 1/\gamma, \gamma = \kappa/2$ 
 $T/\tau = \pi/e \Rightarrow \kappa = \omega/e \Rightarrow Q_{\text{Energie}} = \omega/\kappa = e$ 

Autor der Hypothese: Dr. Naftali Hirschl
Simulation: Eitan / QuTiP 5.x
Lizenzbezug: CC BY-NC-ND 4.0 (Artikel); dieses Skript zur Prüfung
der Dynamik.
"""
```

```
from __future__ import annotations

import numpy as np
from numpy.polynomial.polynomial import polyfit
from qutip import (basis, coherent, destroy, expect, ket2dm,
                   mesolve,
                   num, sigmax, sigmaz)
import matplotlib.pyplot as plt

#
-----
# Konstanten
#
-----
PI = np.pi
E = np.e
KAPPA0 = PI / E #  $\approx 1.15572734979 = T/\tau$ 
(irrational)
KAPPA_R = 52.0 / 45.0 #  $= 1.1555... = T/\tau$ 
(rational, 5. Konvergente)
Q_TARGET = E # Artikel-Q

print("=" * 72)
print("CEA QuTiP-Simulation – Circle-Exponential Equivalence")
print(f" $\pi/e = \{KAPPA0:.12f\}$ ")
print(f" $52/45 = \{KAPPA_R:.12f\}$ ")
print(f" $|\Delta\kappa|/\kappa = \{abs(KAPPA_R - KAPPA0) / KAPPA0:.6e\}$ ")
print("=" * 72)

def simulate_cavity(
```

```

    omega: float = 2 * PI,          #  $\omega = 2\pi \rightarrow T = 1$  in natürlichen
Einheiten
    q_energy: float = E,           #  $\omega/\kappa$ 
    n_periods: float = 8.0,
    n_points: int = 4000,
    alpha0: complex = 2.0 + 0j,
    n_fock: int = 24,
    label: str = "CEA  $\pi/e$ ",
):
    """Gedämpfte Kavität. Gibt Zeitreihe von <x> und Metadaten
zurück."""
    T = 2 * PI / omega
    kappa = omega / q_energy        # Energiezerfallsrate
    gamma_amp = kappa / 2.0        # Artikel- $\gamma$ 
    tau = 1.0 / gamma_amp          # Artikel- $\tau$ 
    t_over_tau = T / tau           # sollte  $\pi/e$  bzw. 52/45 sein

    a = destroy(n_fock)
    H = omega * a.dag() * a
    c_ops = [np.sqrt(kappa) * a]
    psi0 = coherent(n_fock, alpha0)
    tlist = np.linspace(0.0, n_periods * T, n_points)

    x_op = (a + a.dag()) / np.sqrt(2.0)
    p_op = -1j * (a - a.dag()) / np.sqrt(2.0)
    n_op = num(n_fock)

    result = mesolve(H, psi0, tlist, c_ops, [x_op, p_op, n_op],
options={"nsteps": 50_000, "atol": 1e-10,
"rtol": 1e-8})
    x = np.real(result.expect[0])
    p = np.real(result.expect[1])
    n = np.real(result.expect[2])

    meta = dict(
        label=label,
        omega=omega, T=T, kappa=kappa, gamma_amp=gamma_amp, tau=tau,
        t_over_tau_theory=t_over_tau, q_energy=q_energy,
        q_article=omega / (2 * gamma_amp),
        alpha0=alpha0,
    )
    return tlist, x, p, n, meta

def measure_period_zero_cross(t: np.ndarray, x: np.ndarray,
skip_early: float = 0.05):
    """Periode aus positiven Nulldurchgängen (nach
Einschwinganteil)."""
    t0 = t[0] + skip_early * (t[-1] - t[0])
    mask = t >= t0
    tt, xx = t[mask], x[mask]
    # Vorzeichenwechsel -  $\rightarrow$  +
    s = np.sign(xx)
    s[s == 0] = 1

```

```

    idx = np.where((s[:-1] < 0) & (s[1:] > 0))[0]
    if len(idx) < 2:
        return np.nan, np.nan
    crossings = tt[idx] + (0 - xx[idx]) / (xx[idx + 1] - xx[idx]) *
    (tt[idx + 1] - tt[idx])
    periods = np.diff(crossings)
    return float(np.mean(periods)), float(np.std(periods, ddof=1) if
    len(periods) > 1 else 0.0)

```

```

def measure_tau_logfit(t: np.ndarray, x: np.ndarray, p: np.ndarray,
    skip_early: float = 0.02, skip_late: float =
0.15):
    """

```

Amplitude-Hülle aus $r = \sqrt{x^2 + p^2}$ (kohärente Quadraturen).
 Lineare Regression $\ln r = \ln A - t/\tau$.
 Späten Rauschboden und ganz frühen Punkt weglassen.

```

    """
    r = np.sqrt(x**2 + p**2)
    tspan = t[-1] - t[0]
    mask = (t >= t[0] + skip_early * tspan) & (t <= t[-1] -
skip_late * tspan) & (r > 1e-8)
    tt, rr = t[mask], r[mask]
    y = np.log(rr)
    # polyfit:  $y = c_0 + c_1 t \rightarrow c_1 = -1/\tau$ 
    c0, c1 = polyfit(tt, y, 1)
    tau_fit = -1.0 / c1
    # Residuum
    yhat = c0 + c1 * tt
    rms = float(np.sqrt(np.mean((y - yhat) ** 2)))
    return float(tau_fit), rms, r

```

```

def run_and_report(omega=2 * PI, q_energy=E, n_periods=8.0,
    label="CEA  $\pi/e$ ):

```

```

    t, x, p, n, meta = simulate_cavity(
        omega=omega, q_energy=q_energy, n_periods=n_periods,
    label=label
    )

```

```

    T_meas, T_std = measure_period_zero_cross(t, x)
    tau_meas, rms, r = measure_tau_logfit(t, x, p)

```

```

    kappa_meas = T_meas / tau_meas if np.isfinite(T_meas) and
tau_meas > 0 else np.nan
    q_art_meas = PI * tau_meas / T_meas if np.isfinite(T_meas) and
tau_meas > 0 else np.nan
    # Artikel:  $Q = \pi \tau / T$  (weil  $Q = \omega/(2\gamma) = (2\pi/T)/(2/\tau) = \pi \tau / T$ )

```

```

    print(f"\n--- {label} ---")
    print(f"Theorie: T      = {meta['T']:.8f}")
    print(f"Theorie:  $\tau$     = {meta['tau']:.8f}")
    print(f"Theorie:  $T/\tau$   = {meta['t_over_tau_theory']:.10f}")
    print(f"Theorie:  $Q_{\text{art}}$  = {meta['q_article']:.10f} (Soll: e =

```

```

{E:.10f}))")
    print(f"Theorie: Q_E      = {meta['q_energy']:.10f}    ( $\omega/\kappa$ )")
    print(f"Messung: T       = {T_meas:.8f}  $\pm$  {T_std:.2e}")
    print(f"Messung:  $\tau$       = {tau_meas:.8f}    (log-Fit RMS  $\ln r =$ 
{rms:.3e}))")
    print(f"Messung: T/ $\tau$     = {kappa_meas:.10f}")
    print(f"Messung: Q_art    = {q_art_meas:.10f}")
    print(f"rel. Fehler T/ $\tau$  = {abs(kappa_meas -
meta['t_over_tau_theory']) / meta['t_over_tau_theory']:.3e}")

    return t, x, p, n, r, meta, dict(
        T_meas=T_meas, T_std=T_std, tau_meas=tau_meas,
        kappa_meas=kappa_meas, q_art_meas=q_art_meas,
    )

```

#

```

# Drei Regime: CEA ( $\pi/e$ ), CEA rational (52/45), hoher Q
(Negativtest-Idee)
#

```

#

```

#  $\omega = 2\pi \Rightarrow T = 1$ 
#  $Q_E = e \Rightarrow T/\tau = \pi/e$ 
# Für rationale Form:  $T/\tau = 52/45 \Rightarrow Q_{art} = \pi / (52/45) = 45\pi/52$ 
#  $Q_E = Q_{art} = \omega/(2\gamma)$  likewise.

```

```

q_e_irr = E
#  $T/\tau = \pi/e \Leftrightarrow Q_{art} = e$ 
#  $T/\tau = 52/45 \Leftrightarrow Q_{art} = \pi * 45/52$ 
q_e_rat =  $\pi * 45.0 / 52.0$ 
q_e_high = 200.0 # Quarz-ähnlicher Negativtest ( $\kappa$  klein)

```

```

results = {}
t1, x1, p1, n1, r1, m1, f1 = run_and_report(q_energy=q_e_irr,
label="CEA irrational T/ $\tau = \pi/e$ ")
results["irr"] = (t1, x1, p1, n1, r1, m1, f1)

t2, x2, p2, n2, r2, m2, f2 = run_and_report(q_energy=q_e_rat,
label="CEA rational T/ $\tau = 52/45$ ")
results["rat"] = (t2, x2, p2, n2, r2, m2, f2)

t3, x3, p3, n3, r3, m3, f3 = run_and_report(
    q_energy=q_e_high, n_periods=4.0, label="Negativtest Q_E =
200"
)
results["high"] = (t3, x3, p3, n3, r3, m3, f3)

```

#

```

# Zusätzlich: klassische Zweiniveau-Bloch / Rabi mit Amplitude-

```

Dämpfung

```
# (optionalen Quanten-Zwei-Niveau-Check, schwaches Drive aus, nur
Zerfall
# eines Kohärenzterms – hier zur Illustration der Hülle)
#
```

```
-----
-----
```

```
def simulate_tls_coherence(omega=2 * PI, q_energy=E, n_periods=8.0,
n_points=4000):
```

```
    """
```

```
    TLS:  $H = (\omega/2) \sigma_z$ , Kollaps  $\sqrt{\gamma} \sigma_-$  (Amplitudendämpfung).
    Kohärenz  $\rho_{01} \sim e^{-\gamma t / 2} e^{-i \omega t}$  in manchen Konventionen.
    Wir setzen  $\gamma_{\text{energy}} = \omega/q_{\text{energy}}$  analog,  $\sigma_-$ -Rate =  $\gamma_{\text{energy}}$ .
    """
```

```
    T = 2 * PI / omega
    kappa = omega / q_energy
    sm = (sigmax() - 1j * sigmaz() * 0) # dummy, use destroy-like
    from qutip import sigmam, sigmap
    H = 0.5 * omega * sigmaz()
    c_ops = [np.sqrt(kappa) * sigmam()]
    # Start: Superposition
    psi0 = (basis(2, 0) + basis(2, 1)).unit()
    tlist = np.linspace(0.0, n_periods * T, n_points)
    sx, sy = sigmax(), (sigmam() - sigmap()) * 1j #  $\sigma_y$ 
    from qutip import sigmay
    result = mesolve(H, psi0, tlist, c_ops, [sigmax(), sigmay()],
                    options={"nsteps": 20_000})
    return tlist, np.real(result.expect[0]),
    np.real(result.expect[1])
```

```
#
```

```
-----
-----
```

```
# Abbildungen
```

```
#
```

```
-----
-----
```

```
fig, axes = plt.subplots(3, 1, figsize=(11, 10), sharex=False)
```

```
def panel(ax, t, x, r, meta, fit, title_extra=""):
```

```
    T = meta["T"]
```

```
    tau = meta["tau"]
```

```
    env_th = r[0] * np.exp(-t / tau) if r[0] > 0 else np.exp(-t /
tau) * np.max(np.abs(x))
```

```
    # besser: Hülle an max |x| nahe t=0 anpassen
```

```
    a0 = np.max(np.abs(x[: max(5, len(x) // 80)]))
```

```
    env_th = a0 * np.exp(-t / tau)
```

```
    ax.plot(t / T, x, color="#1f4e79", lw=1.1, label=r"$\langle
```

```
x\rangle(t)$")
```

```
    ax.plot(t / T, env_th, color="#c0392b", lw=1.3, ls="--",
```

```

        label=fr"Theorie  $A_0 e^{\{-t/\tau\}}$ ,  $\tau/T=\{\tau/T:.5f\}$ "
    ax.plot(t / T, -env_th, color="#c0392b", lw=1.3, ls="--")
    ax.axhline(0, color="0.5", lw=0.4)
    ax.set_ylabel(r"$\langle x \rangle$")
    ax.set_title(
        f"{meta['label']} | T/ $\tau$ 
theo={meta['t_over_tau_theory']:.6f} "
        f"gemessen={fit['kappa_meas']:.6f} {title_extra}",
        fontsize=10,
    )
    ax.legend(loc="upper right", fontsize=8, frameon=False)
    ax.set_xlim(0, t[-1] / T)
    ax.grid(True, alpha=0.25)

panel(axes[0], t1, x1, r1, m1, f1)
panel(axes[1], t2, x2, r2, m2, f2)
panel(axes[2], t3, x3, r3, m3, f3)
axes[2].set_xlabel(r"Zeit in Perioden  $t/T$ ")

fig.suptitle(
    "QuTiP: gedämpfter Quantenoszillator (Kavität) – Circle-Exponential Equivalence\n"
    r"$x(t) \propto e^{\{-t/\tau\}} \cos(\omega t)$,  $T/\tau=\pi/e$ ;  

 $\mathit{oder}$ ; 52/45,  $Q=e$ ",
    fontsize=12,
)
fig.tight_layout(rect=[0, 0, 1, 0.96])
fig_path = "/home/workdir/artifacts/cea_qutip_waveforms.png"
fig.savefig(fig_path, dpi=160)
print(f"\nAbbildung gespeichert: {fig_path}")

# Zweites Bild: Energiezerfall  $n(t)$  vs.  $e^{-\kappa t}$ 
fig2, ax = plt.subplots(figsize=(10, 4.5))
for key, color, ls in [("irr", "#1f4e79", "-"), ("rat", "#c0392b", "--")]:
    t, x, p, n, r, meta, fit = results[key]
    T = meta["T"]
    n0 = n[0]
    ax.plot(t / T, n / n0, color=color, ls=ls, lw=1.6,
            label=fr"{meta['label']}:  $\langle n \rangle / \langle n \rangle_0$ "
            n\langle n \rangle_0$")
    ax.plot(t / T, np.exp(-meta["kappa"] * t), color=color, ls=":",
            lw=1.2,
            label=fr"  $e^{\{-\kappa t\}}$ ,  $\kappa=\omega/Q_E$ ")
ax.set_xlabel(r"$t/T$")
ax.set_ylabel(r"normierte Photonenzahl")
ax.set_yscale("log")
ax.set_title(r"Energiehülle:  $\langle n \rangle(t) \propto e^{-\kappa t}$  (CEA:  $Q_E=e$  bzw.  $45\pi/52$ )")
ax.legend(fontsize=8, frameon=False)
ax.grid(True, which="both", alpha=0.3)
fig2.tight_layout()
fig2_path = "/home/workdir/artifacts/cea_qutip_energy.png"

```

```

fig2.savefig(fig2_path, dpi=160)
print(f"Abbildung gespeichert: {fig2_path}")

# Kurze TLS-Kohärenz nur als Zahlencheck
print("\n--- TLS-Kohärenz ( $\sigma$ --Dämpfung), nur Fit-Check CEA  $\pi/e$  ---")
tt, sx, sy = simulate_tls_coherence()
r_tls = np.sqrt(sx**2 + sy**2)
# log-Fit
tspan = tt[-1] - tt[0]
mask = (tt > 0.05 * tspan) & (tt < 0.85 * tspan) & (r_tls > 1e-6)
c0, c1 = polyfit(tt[mask], np.log(r_tls[mask]), 1)
tau_tls = -1.0 / c1
T_tls = 1.0 #  $\omega=2\pi$ 
print(f"TLS  $T/\tau$  gemessen = {T_tls / tau_tls:.8f} Theorie  $\pi/e =$ 
{KAPPA0:.8f}")
# Bei  $\sigma$ :- Bloch-Vektor-Transversalkomponente zerfällt mit  $\kappa/2$ ,
analog Amplitude.

print("\nFertig. Skript: /home/workdir/artifacts/
cea_qutip_simulation.py")

```