```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

U = np.array([0, 0.1, 0.2, 0.3, 0.5, 0.7, 1, 1.5, 2, 3.5, 5, 8, 10, 15, 20, 25, 30])
C_pF = np.array([122.6, 110.1, 102.4, 96.5, 87.8, 81.6, 74.8, 67.2, 62, 52.5, 47, 40.5, 37.7, 33.1, 30.1, 28, 26.4])
C = C_pF * 1e-12

M_values = np.linspace(0.2, 1.0, 100)
best_r2 = -np.inf
best_M = None
best_Y = None

for M in M_values:
    Y = (1 / C)**(1 / M)
    model = LinearRegression()
    model.fit(U.reshape(-1, 1), Y)
    Y_pred = model.predict(U.reshape(-1, 1))
    r2 = r2_score(Y, Y_pred)

    if r2 > best_r2:
        best_r2 = r2
        best_M = M
        best_Y = Y
        best_Y_pred = Y_pred


#Regressionskoeffizienten
model = LinearRegression()
model.fit(U.reshape(-1, 1), best_Y)
a = model.coef_[0]
b = model.intercept_


#UD = b / a
print(f"Diffusionsspannung U_D ≈ {UD:.3f} V")

print(f"Optimaler Stufenfaktor M: {best_M:.3f} mit R² = {best_r2:.5f}")

# Plot
plt.scatter(U, best_Y, label="Daten transformiert")
plt.plot(U, best_Y_pred, color="red", label="Lineare Regression")
plt.xlabel("Sperrspannung U (V)")
plt.ylabel("$(1/C)^{1/M}$ (1/F)⁽¹/ᴹ⁾")
plt.title(f"Linearisierung für besten M = {best_M:.3f}")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```