

```

# =====
# PEMFC DEGRADATION MODELLING WITH:
# 1) Linear voltage baseline
# 2) Exponential model
# 3) Weibull model
# 4) SVR
# 5) ARIMA
# =====

import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from scipy.optimize import curve_fit
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.arima.model import ARIMA

# =====
# 1. CONFIGURATION
# =====
DATA_PATH = "/content/FC1_Ageing_merged.csv" # change if needed

COL_TIME = "Time"
COL_U = "Utot"

N_REF = 1000
EPSILON = 1e-6
U_EOL = 3.093

# Reliability / RUL settings
R_FAILURE = 0.2 # reliability threshold for EOL estimate
HI_THRESHOLD = 0.3 # health-index threshold for RUL estimate
FORECAST_MAX_NORM = 2.5 # normalized-time forecast horizon
FORECAST_POINTS = 800 # extra points for extrapolation

# =====
# 2. LOAD DATA
# =====
df = pd.read_csv(DATA_PATH)
df = df.sort_values(COL_TIME).reset_index(drop=True)

print("Total samples:", len(df))

```

```

# =====
# 3. HEALTHY REFERENCE PERIOD
# =====
ref = df.iloc[:N_REF]
U_ref = ref[COL_U].mean()

print("Reference voltage (U_ref):", U_ref)
print("End-of-life voltage (U_EOL):", U_EOL)

# =====
# 4. VOLTAGE DEGRADATION INDEX
#  $D_V(t) = (U_{ref} - U(t)) / (U_{ref} - U_{EOL})$ 
# =====
den_V = max(abs(U_ref - U_EOL), EPSILON)
df["D_V"] = ((U_ref - df[COL_U]) / den_V).clip(0, 1)

# =====
# 5. TIME AGGREGATION
# =====
df["Hour"] = df[COL_TIME].astype(int)
df_h = df.groupby("Hour")["D_V"].mean().reset_index()

# =====
# 6. MONOTONIC DEGRADATION
#  $D_V_{mono}(t) = \max_{\tau \leq t} D_V(\tau)$ 
# =====
df_h["D_V_mono"] = df_h["D_V"].cummax()

# =====
# 7. NORMALIZED TIME
# =====
df_h["t_norm"] = df_h["Hour"] / df_h["Hour"].max()

t = df_h["t_norm"].values
y = df_h["D_V_mono"].values

# =====
# 8. MODEL DEFINITIONS
# =====
def linear_model(t, m, c):
    return m * t + c

```

```

def exp_model(t, a, k):
    return a * (1 - np.exp(-k * t))

def weibull_model(t, a, k, beta):
    return a * (1 - np.exp(-(k * t) ** beta))

# =====
# 9. FIT LINEAR VOLTAGE BASELINE
# =====
params_lin, cov_lin = curve_fit(
    linear_model,
    t,
    y,
    bounds=([0.0, -0.2], [2.0, 1.0])
)

m_lin, c_lin = params_lin
y_lin = linear_model(t, m_lin, c_lin)
y_lin = np.clip(y_lin, 0, 1)

# =====
# 10. FIT EXPONENTIAL MODEL
# =====
params_exp, cov_exp = curve_fit(
    exp_model,
    t,
    y,
    bounds=(0, [1.5, 50])
)

a_exp, k_exp = params_exp
y_exp = exp_model(t, a_exp, k_exp)
y_exp = np.clip(y_exp, 0, 1)

# =====
# 11. FIT WEIBULL MODEL
# =====
params_w, cov_w = curve_fit(
    weibull_model,
    t,
    y,
    p0=[0.2, 2.0, 2.0],
    bounds=(0, [1.5, 50, 5])
)

a_w, k_w, beta_w = params_w

```

```

y_w = weibull_model(t, a_w, k_w, beta_w)
y_w = np.clip(y_w, 0, 1)

# =====
# 12. APPROXIMATE 95% CONFIDENCE BAND FOR WEIBULL MODEL
# =====
std_errors = np.sqrt(np.diag(cov_w))

a_low = max(0, a_w - 1.96 * std_errors[0])
k_low = max(0, k_w - 1.96 * std_errors[1])
beta_low = max(0, beta_w - 1.96 * std_errors[2])

a_high = a_w + 1.96 * std_errors[0]
k_high = k_w + 1.96 * std_errors[1]
beta_high = beta_w + 1.96 * std_errors[2]

y_low = weibull_model(t, a_low, k_low, beta_low)
y_high = weibull_model(t, a_high, k_high, beta_high)

y_low = np.clip(y_low, 0, 1)
y_high = np.clip(y_high, 0, 1)

# =====
# 13. SVR MODEL
# =====
t_resaped = t.reshape(-1, 1)

scaler = StandardScaler()
t_scaled = scaler.fit_transform(t_resaped)

svr_model = SVR(kernel="rbf", C=100, gamma=0.1, epsilon=0.01)
svr_model.fit(t_scaled, y)
y_svr = svr_model.predict(t_scaled)
y_svr = np.clip(y_svr, 0, 1)

# =====
# 14. ARIMA MODEL
# =====
arima_model = ARIMA(y, order=(2, 1, 2))
arima_fit = arima_model.fit()
y_arima = arima_fit.predict(start=0, end=len(y) - 1)
y_arima = np.clip(y_arima, 0, 1)

# =====
# 15. MODEL PERFORMANCE

```

```

# =====
r2_lin = r2_score(y, y_lin)
rmse_lin = np.sqrt(mean_squared_error(y, y_lin))

r2_exp = r2_score(y, y_exp)
rmse_exp = np.sqrt(mean_squared_error(y, y_exp))

r2_w = r2_score(y, y_w)
rmse_w = np.sqrt(mean_squared_error(y, y_w))

r2_svr = r2_score(y, y_svr)
rmse_svr = np.sqrt(mean_squared_error(y, y_svr))

r2_arima = r2_score(y, y_arima)
rmse_arima = np.sqrt(mean_squared_error(y, y_arima))

results_df = pd.DataFrame({
    "Model": ["Linear", "Exponential", "Weibull", "SVR", "ARIMA"],
    "R2": [r2_lin, r2_exp, r2_w, r2_svr, r2_arima],
    "RMSE": [rmse_lin, rmse_exp, rmse_w, rmse_svr, rmse_arima]
})

print("\nMODEL COMPARISON")
print("-" * 60)
print(f"{'Model':<15}{'R²':<12}{'RMSE':<12}")
print("-" * 60)
print(f"{'Linear':<15}{r2_lin:<12.4f}{rmse_lin:<12.4f}")
print(f"{'Exponential':<15}{r2_exp:<12.4f}{rmse_exp:<12.4f}")
print(f"{'Weibull':<15}{r2_w:<12.4f}{rmse_w:<12.4f}")
print(f"{'SVR':<15}{r2_svr:<12.4f}{rmse_svr:<12.4f}")
print(f"{'ARIMA':<15}{r2_arima:<12.4f}{rmse_arima:<12.4f}")

print("\nLINEAR BASELINE PARAMETERS")
print("m :", m_lin)
print("c :", c_lin)

print("\nEXPONENTIAL PARAMETERS")
print("a :", a_exp)
print("k :", k_exp)

print("\nWEIBULL PARAMETERS")
print("a      :", a_w)
print("k      :", k_w)
print("beta   :", beta_w)

# =====
# 16. HEALTH INDEX
# =====

```

```

df_h["Health_Index"] = (1 - y_w).clip(0, 1)

# =====
# 17. RELIABILITY AND FAILURE PROBABILITY
# =====
# Weibull reliability function:
#  $R(t) = \exp(-(k \cdot t)^\beta)$ 
#  $F(t) = 1 - R(t)$ 
df_h["Reliability"] = np.exp(-((k_w * t) ** beta_w))
df_h["Failure_Probability"] = 1 - df_h["Reliability"]

# =====
# 18. DEGRADATION RATE
#  $dD/dt = a * \beta * k^\beta * t^{(\beta-1)} * \exp(-(k \cdot t)^\beta)$ 
# =====
t_safe = np.clip(t, 1e-6, None)

df_h["Degradation_Rate"] = (
    a_w
    * beta_w
    * (k_w ** beta_w)
    * (t_safe ** (beta_w - 1))
    * np.exp(-((k_w * t_safe) ** beta_w))
)

# =====
# 19. PLOT 1: MODEL COMPARISON
# =====
plt.figure(figsize=(8, 5))

plt.plot(t, y, linewidth=1.5, label="Measured degradation")
plt.plot(t, y_lin, linewidth=1.8, linestyle="-. ", label="Linear voltage baseline")
plt.plot(t, y_exp, linewidth=1.8, label="Exponential model")
plt.plot(t, y_w, linewidth=2.4, label="Weibull model")
plt.plot(t, y_svr, linewidth=1.8, linestyle="--", label="SVR model")
plt.plot(t, y_arima, linewidth=1.8, linestyle=":", label="ARIMA model")

plt.fill_between(
    t, y_low, y_high,
    alpha=0.20,
    label="Approx. 95% confidence band"
)

plt.xlabel("Normalized operating time")
plt.ylabel("Degradation index")

```

```

plt.title("Degradation modelling of PEMFC durability data")
plt.xlim(0, 1.02)
plt.ylim(0, min(1.0, max(y_high.max(), y.max()) + 0.03))
plt.legend(frameon=True, fontsize=9)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig("Figure_2_Degradation_Modelling_All_Models.png", dpi=600,
bbox_inches="tight")
plt.show()

# =====
# 20. PLOT 2: HEALTH INDEX
# =====

plt.figure(figsize=(8, 5))

plt.plot(t, df_h["Health_Index"], linewidth=2)

plt.xlabel("Normalized operating time")
plt.ylabel("Health index")
plt.title("Health index evolution of the PEMFC system")
plt.xlim(0, 1.02)
plt.ylim(0, 1.02)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig("Figure_3_Health_Index.png", dpi=600, bbox_inches="tight")
plt.show()

# =====
# 21. RESIDUAL ANALYSIS
# =====

res_lin = y - y_lin
res_exp = y - y_exp
res_w = y - y_w
res_svr = y - y_svr
res_arima = y - y_arima

plt.figure(figsize=(8, 4.5))

plt.scatter(t, res_lin, s=16, label="Linear")
plt.scatter(t, res_exp, s=16, label="Exponential")
plt.scatter(t, res_w, s=16, label="Weibull")
plt.scatter(t, res_svr, s=16, label="SVR")
plt.scatter(t, res_arima, s=16, label="ARIMA")

plt.axhline(0, linewidth=1)

plt.xlabel("Normalized operating time")

```

```

plt.ylabel("Residual")
plt.title("Residual comparison of degradation models")
plt.legend(frameon=True, fontsize=9)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig("Figure_4_Residual_Comparison_All_Models.png", dpi=600,
bbox_inches="tight")
plt.show()

# =====
# 22. RESIDUAL HISTOGRAM
# =====
plt.figure(figsize=(8, 4.5))

plt.hist(res_lin, bins=20, alpha=0.5, label="Linear")
plt.hist(res_exp, bins=20, alpha=0.5, label="Exponential")
plt.hist(res_w, bins=20, alpha=0.5, label="Weibull")
plt.hist(res_svr, bins=20, alpha=0.5, label="SVR")
plt.hist(res_arima, bins=20, alpha=0.5, label="ARIMA")

plt.xlabel("Residual")
plt.ylabel("Frequency")
plt.title("Residual histogram comparison of degradation models")
plt.legend(frameon=True, fontsize=9)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig("Figure_5_Residual_Histogram_All_Models.png", dpi=600,
bbox_inches="tight")
plt.show()

# =====
# 23. PLOT 3: RELIABILITY AND FAILURE PROBABILITY
# =====
plt.figure(figsize=(8, 5))

plt.plot(t, df_h["Reliability"], linewidth=2, label="Reliability
function R(t)")
plt.plot(t, df_h["Failure_Probability"], linewidth=2, label="Failure
probability F(t)")

plt.xlabel("Normalized operating time")
plt.ylabel("Probability")
plt.title("Reliability and failure probability evolution")
plt.xlim(0, 1.02)
plt.ylim(0, 1.02)
plt.legend(frameon=True, fontsize=9)
plt.grid(True, alpha=0.3)

```



```

plt.tight_layout()
plt.savefig("Figure_6_Reliability_Failure_Probability.png", dpi=600,
bbox_inches="tight")
plt.show()

```

```

# =====
# 24. PLOT 4: DEGRADATION RATE
# =====
plt.figure(figsize=(8, 5))

```

```

plt.plot(t, df_h["Degradation_Rate"], linewidth=2)

```

```

plt.xlabel("Normalized operating time")
plt.ylabel("dD/dt")
plt.title("Weibull-based degradation rate curve")
plt.xlim(0, 1.02)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig("Figure_7_Degradation_Rate.png", dpi=600,
bbox_inches="tight")
plt.show()

```

```

# =====
# 25. RUL / EOL ESTIMATION FROM HEALTH INDEX THRESHOLD
# =====
# Extrapolate Weibull degradation beyond observed time
t_ext = np.linspace(0, FORECAST_MAX_NORM, len(t) + FORECAST_POINTS)
y_w_ext = weibull_model(t_ext, a_w, k_w, beta_w)
y_w_ext = np.clip(y_w_ext, 0, 1)

```

```

hi_ext = 1 - y_w_ext

```

```

# Find first time where health index drops below threshold
idx_hi = np.where(hi_ext <= HI_THRESHOLD)[0]

```

```

if len(idx_hi) > 0:
    t_eol_hi_norm = t_ext[idx_hi[0]]
    t_eol_hi_hour = t_eol_hi_norm * df_h["Hour"].max()
    current_hour = df_h["Hour"].max()
    rul_hi_hour = max(0, t_eol_hi_hour - current_hour)

```

```

    hi_msg_eol = f"{t_eol_hi_hour:.2f} hours"
    hi_msg_rul = f"{rul_hi_hour:.2f} hours"

```

```

else:
    t_eol_hi_norm = np.nan
    t_eol_hi_hour = np.nan
    rul_hi_hour = np.nan

```

```

hi_msg_eol = "Not reached within forecast horizon"
hi_msg_rul = "Not reached within forecast horizon"

print("\nHEALTH-INDEX-BASED RUL ESTIMATE")
print("HI threshold      :", HI_THRESHOLD)
print("Estimated EOL      :", hi_msg_eol)
print("Estimated RUL      :", hi_msg_rul)

# =====
# 26. EOL ESTIMATION FROM RELIABILITY THRESHOLD
#  $t_{EOL} = ((-\ln(R_f))^{(1/\beta)}) / k$ 
# =====
if k_w > 0 and beta_w > 0 and 0 < R_FAILURE < 1:
    t_eol_rel_norm = ((-np.log(R_FAILURE)) ** (1 / beta_w)) / k_w
    t_eol_rel_hour = t_eol_rel_norm * df_h["Hour"].max()
    rel_msg_eol = f"{t_eol_rel_hour:.2f} hours"
else:
    t_eol_rel_norm = np.nan
    t_eol_rel_hour = np.nan
    rel_msg_eol = "Could not be computed"

print("\nRELIABILITY-BASED EOL ESTIMATE")
print("Failure threshold R_f :", R_FAILURE)
print("Estimated EOL          :", rel_msg_eol)

# =====
# 27. PLOT 5: RUL / FUTURE DEGRADATION
# =====
plt.figure(figsize=(8, 5))

plt.plot(t, y, linewidth=1.5, label="Measured degradation")
plt.plot(t_ext, y_w_ext, linewidth=2.2, label="Weibull extrapolation")
plt.axhline(1 - HI_THRESHOLD, linewidth=1.5, linestyle="--", label="EOL degradation threshold")

if not np.isnan(t_eol_hi_norm):
    plt.axvline(t_eol_hi_norm, linewidth=1.5, linestyle=":", label="Predicted EOL")

plt.xlabel("Normalized operating time")
plt.ylabel("Degradation index")
plt.title("Future degradation extrapolation and RUL estimation")
plt.xlim(0, max(t_ext))
plt.ylim(0, 1.02)
plt.legend(frameon=True, fontsize=9)
plt.grid(True, alpha=0.3)

```

```

plt.tight_layout()
plt.savefig("Figure_8_RUL_Estimation.png", dpi=600,
bbox_inches="tight")
plt.show()
# =====
# 28. SAVE RESULTS
# =====
df_h.to_csv("PEMFC_Degradation_Processed_Results.csv", index=False)
results_df.to_csv("PEMFC_Model_Comparison.csv", index=False)

print("\nFiles saved:")
print("- PEMFC_Degradation_Processed_Results.csv")
print("- PEMFC_Model_Comparison.csv")
print("- Figure_2_Degradation_Modelling_All_Models.png")
print("- Figure_3_Health_Index.png")
print("- Figure_4_Residual_Comparison_All_Models.png")
print("- Figure_5_Residual_Histogram_All_Models.png")
print("- Figure_6_Reliability_Failure_Probability.png")
print("- Figure_7_Degradation_Rate.png")
print("- Figure_8_RUL_Estimation.png")

```