```
import numpy as np
import os
import csv
import sys
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
from scipy.optimize import minimize
import warnings

# ============================================================
# MOTEUR ATPEW V9 "GOLDEN" - FULL PERFORMANCE (RMS 2690)
# Visualisation Vortex Algébrique (Signée) & Synthèse V8
# ============================================================
plt.ioff()
warnings.filterwarnings("ignore")

# --- CONSTANTES ---
K_PIONEER = 4.854
G_CONST = 4.302e-6
SIGMA_CRIT = 160.0
A0_MOND = 1.2e-10
EPS = 1e-12

def moteur_v9_golden_core(r, v_bar, v_disk, v_bulge, alpha=0.05, beta=0.0):
    """ Moteur exact du record RMS 2690 """
    v_bar_max = np.max(v_bar)
    v_bar_end = v_bar[-1]
    compacite = v_bar_max / (v_bar_end + EPS)

    M_bar = (v_bar[-1]**2 * r[-1]) / G_CONST
    m_ratio = M_bar / 1e10

    # Couplage Pioneer 4.854
    k_vortex_bulbe = (K_PIONEER * 4.12) * (m_ratio)**alpha
    k_vortex_halo = (K_PIONEER * 3.71) * (m_ratio)**alpha

    r_peak = r[np.argmax(v_disk)] if np.any(v_disk > 0) else np.median(r)
    r0 = r_peak * 1.5 * (m_ratio**0.05)
    pente = 3.0 + (compacite * 0.2)

    T_ext = 1.0 / (1.0 + (r0 / (r + EPS))**pente)
    T_int = 1.0 - T_ext

    sigma_loc = (v_disk**2 + v_bulge**2) / (np.pi * G_CONST * r + EPS)

    # Calcul des composantes vortex
    v_int_pot = k_vortex_bulbe * np.sqrt(sigma_loc / (sigma_loc + SIGMA_CRIT)) * np.sqrt(r) *
T_int
    v_ext_pot = k_vortex_halo * (m_ratio**0.25) * 4.2 * T_ext
    v_couplage = beta * k_vortex_bulbe * np.sqrt(M_bar / (r + EPS))
    v_vortex_brut = v_int_pot + v_ext_pot + v_couplage
```

```
    # Freinage V9 Performance (Le seuil 0.82)
    weight_bar = 1.0
    if v_bar_max > 130 and compacite > 1.2:
        weight_bar = np.clip(1.1 - (compacite * 0.2), 0.82, 1.0)

    v_bar_eff = v_bar * np.sqrt(weight_bar)
    v_final = np.sqrt(np.maximum((v_bar_eff**2) + (v_vortex_brut**2), 0))

    # Contribution Vortex pour le graphique (Algébrique)
    v_vort_sign = v_final - v_bar

    return v_final, v_vort_sign, v_bar_eff

def optimiser_parametres(r, v_obs, v_bar, v_disk, v_bulge):
    def objectif(p):
        v, _, _ = moteur_v9_golden_core(r, v_bar, v_disk, v_bulge, p[0], p[1])
        return np.sqrt(np.mean((v - v_obs)**2))
    res = minimize(objectif, x0=[0.01, 0.01], bounds=[(0.0, 0.5), (0.0, 0.4)], method='L-BFGS-B')
    return res.x

# --- CHEMINS ---
base_dir = r"C:\Users\Jos\Desktop\Temps\ATPEW - Falsifiabilité\ATPEW - CHATGPT - Data
SPARC"
path_sparc = os.path.join(base_dir, "SPARC_data")
pdf_path = os.path.join(base_dir, "Synthese_V9_Golden_Vortex_Final.pdf")
csv_path = os.path.join(base_dir, "Resultats_V9_Golden_Vortex_Final.csv")

if os.path.exists(path_sparc):
    files = [f for f in os.listdir(path_sparc) if f.endswith("_rotmod.dat")]
    total = len(files)
    stats = {"Classe A (LSB)": {"sum_atp": 0.0, "sum_mon": 0.0, "count": 0, "wins": 0},
             "Classe B (INTER)": {"sum_atp": 0.0, "sum_mon": 0.0, "count": 0, "wins": 0},
             "Classe C (MASSIVE)": {"sum_atp": 0.0, "sum_mon": 0.0, "count": 0, "wins": 0}}
    csv_rows = []

    print(f"Lancement Moteur V9 GOLDEN - Visualisation Signée sur {total} galaxies...")

    with PdfPages(pdf_path) as pdf:
        for i, fname in enumerate(files):
            pct = int((i+1)/total*100)
            sys.stdout.write(f"\rProgression : [{('#'*(pct//2)):<50}] {pct}% | {fname[:12]}")
            sys.stdout.flush()

            try:
                data = np.loadtxt(os.path.join(path_sparc, fname))
                r, v_obs, v_err = data[:,0], data[:,1], data[:,2]
                v_bar = np.sqrt(data[:,3]**2 + data[:,4]**2 + data[:,5]**2)

                # Optimisation
                a_opt, b_opt = optimiser_parametres(r, v_obs, v_bar, data[:,4], data[:,5])
                v_atp, v_vort_sign, v_bar_eff = moteur_v9_golden_core(r, v_bar, data[:,4], data[:,5],
a_opt, b_opt)
```

```
            # MOND pour comparaison
            g_n = (v_bar * 1000)**2 / (r * 3.086e19 + EPS)
            v_mond = np.sqrt(np.sqrt(g_n * A0_MOND) * r * 3.086e19) / 1000

            rms_atp = np.sqrt(np.mean((v_atp-v_obs)**2))
            rms_mon = np.sqrt(np.mean((v_mond-v_obs)**2))

            # Classification
            v_flat = np.mean(v_obs[-3:])
            cl = "Classe A (LSB)" if v_flat < 80 else "Classe B (INTER)" if v_flat < 150 else "Classe
C (MASSIVE)"

            stats[cl]["sum_atp"] += rms_atp
            stats[cl]["sum_mon"] += rms_mon
            stats[cl]["count"] += 1
            res_win = "WIN" if rms_atp < rms_mon else "loss"
            if res_win == "WIN": stats[cl]["wins"] += 1
            csv_rows.append([fname, cl, round(v_flat, 2), round(a_opt, 4), round(rms_atp, 4),
round(rms_mon, 4), res_win])

            # --- GRAPHIQUE STYLE V8 AVEC VORTEX SIGNÉ ---
            plt.figure(figsize=(10, 6))
            plt.errorbar(r, v_obs, yerr=v_err, fmt='ko', markersize=3, alpha=0.3, label='Observed
(SPARC)')
            plt.plot(r, v_atp, 'r-', linewidth=2.5, label=f'ATPEW V9 (RMS {rms_atp:.2f})')
            plt.plot(r, v_mond, 'g--', alpha=0.7, label=f'MOND (RMS {rms_mon:.2f})')
            plt.plot(r, v_bar, 'b--', alpha=0.5, label='Baryons (Newton pur)')

            # Contribution Vortex (Ligne violette signée)
            plt.plot(r, v_vort_sign, color='darkviolet', linestyle=':', linewidth=2, label='Contribution
Vortex (Signed)')
            plt.axhline(0, color='black', linewidth=0.8, alpha=0.5) # Ligne de référence zéro

            # Coloration des zones (Optionnel mais visuel)
            plt.fill_between(r, 0, v_vort_sign, where=(v_vort_sign >= 0), color='purple', alpha=0.05)
            plt.fill_between(r, 0, v_vort_sign, where=(v_vort_sign < 0), color='red', alpha=0.05)

            plt.title(f"{fname} - V9 Golden (K={K_PIONEER})\nViolet > 0: Accélération | Violet <
0: Freinage")
            plt.xlabel("Radius (kpc)")
            plt.ylabel("Velocity (km/s)")
            plt.legend(loc='best', fontsize='small', frameon=True)
            plt.grid(True, linestyle=':', alpha=0.4)
            pdf.savefig()
            plt.close()
        except: continue

    # Sauvegarde CSV
    with open(csv_path, "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(["Galaxy", "Classe", "V_flat", "Alpha_opt", "RMS_ATPEW",
```

```
"RMS_MOND", "Result"])
        writer.writerows(csv_rows)

    # Synthèse Finale en Console
    print(f"\n\n--- SYNTHÈSE V9 GOLDEN (PERFORMANCE 2690) ---")
    t_atp, t_mon = 0, 0
    for cl in ["Classe A (LSB)", "Classe B (INTER)", "Classe C (MASSIVE)"]:
        d = stats[cl]
        t_atp += d['sum_atp']; t_mon += d['sum_mon']
        if d['count'] > 0:
            print(f"{cl:<20} | {d['count']:<4} | {d['sum_atp']:<10.2f} | {d['sum_mon']:<10.2f} |
{d['wins']}/{d['count']}")

    print(f"\nTOTAL GLOBAL RMS : ATPEW = {t_atp:.2f} | MOND = {t_mon:.2f}")
    print(f"Fichiers générés : {os.path.basename(pdf_path)} et {os.path.basename(csv_path)}")
else:
    print(f"Erreur : Chemin introuvable {path_sparc}")
```