膜宇宙論 v4.4 解析スクリプト全集
Script Catalog with Full Source Code

著者: 坂口 忍 (坂口製麺所)

日付: 2026年4月7日

スクリプト数: 13本 (本編) + 3本 (中間版) = 16本

本文書は膜宇宙論v4.4検証作業で使用した全解析スクリプトの目的・解析対象データ・主要結果・ソースコード全文を収録する。全スクリプトは再現可能性を担保するためそのまま実行可能な形式で記載。

実行環境: Claude Code (VS Code拡張) on Windows, uv run --with [dependencies] python [script]

スクリプト一覧

#	スクリプト	章	目的
1	sparc_jackknife_extended.py	第23章	SPARC 175銀河の拡張ジャックナイフ検証
2	probes_vbar_pipeline.py	第24章前提	PROBES x S4G クロスマッチによる概算V_bar構築
3	probes_Yd_free.py	第24章	PROBES 290銀河でYdをフリーパラメータとして最適化
4	sparc_Yd_sensitivity.py	第25章v1	SPARC Yd感度テスト (Sigma0にYd混入版)
5	sparc_Yd_sensitivity_v2.py	第25章	SPARC Yd感度テスト v2 (G*Sigma0=Vflat^2/hR, Yd非依存)
6	sparc_gc_method_comparison.py	第26章前提	g_c定義の6手法比較 + TA3との直接比較
7	sparc_alpha_decomposition.py	第26章前提	TA3 CSVの構造解析と要素分離(初版)
8	sparc_alpha_reproduce.py	第26章(最終版)	alpha=0.545完全分解: 元パイプライン完全再現+要素置換
9	probes_vbar_v2_local.py	付録(V_barボトルネック)	VizieR HIソース系統探索 + 既存データ確認
10	cluster_stack_v3_fix2.py	付録(弱レンズ)	BCG同定(z_spec必須) + cl1 3中心シェアプロファイル比較
11	cl1_model_fit.py	付録(弱レンズ)	cl1単体: NFW/NFW+2halo/Gaussianリング 3モデルフィット
12	cl1_cl3_stack.py	付録(弱レンズ)	cl1+cl3スタック(cl4/cl27除外): Sigma_crit正規化
13	cl1_modelB_fit.py	付録(弱レンズ)	光伝播モデルB(式10a-10d, 提案G) cl1フィット

中間版 (参考):

#	スクリプト	章	目的
S1	cluster_stack_v3.py	中間版(v3_fix2に置換)	BCG同定(z_spec NULL許可版) + NFW+2haloモデル設計
S2	cluster_stack_v3_fix.py	中間版(v3_fix2に置換)	BCG同定修正版(z_spec必須+M_r<-21)
S3	probes_vbar_v2.py	中間版(v2_localに置換)	PROBES V_bar v2: WHISP HI + S4G + Yd最適化(VizieR版)

sparc_jackknife_extended.py

章: 第23章

目的: SPARC 175銀河の拡張ジャックナイフ検証

解析対象データ: SPARC Rotmod_LTG (Lelli+2016)

主要結果: 半分割x1000: alpha=0.545+/-0.037 (train), 0.550+/-0.038 (test)。Leave-50-out: alpha=0.546+/-0.024。Bootstrap 95%CI=[0.478,0.626]。内部頑健性は決定的。

ソースコード (365 lines, 10,851 bytes)

```
#!/usr/bin/env python3
"""
sparc_jackknife_extended.py
===========================
SPARC 175銀河での拡張ジャックナイフ検証。
半分割×1000回で内部独立性を徹底的に示す。

検証項目:
  1. train/test 各αの分布 → α=0.5の棄却率
  2. Δα = α(train) - α(test) の分布 → 系統差の有無
  3. 残差σの安定性
  4. Hubble type別・質量ビン別の層化ジャックナイフ
  5. Leave-N-out (N=10,20,50) の安定性

既存結果（再現対象）:
  α(train) = 0.545 ± 0.037
  α(test)  = 0.550 ± 0.038

実行: uv run --with scipy --with matplotlib --with numpy python sparc_jackknife_extended.py

前提: SPARCデータが Rotmod_LTG/ ディレクトリに存在すること。
      パスは環境に応じて変更してください。
"""

import numpy as np
import os
import sys
import json
from pathlib import Path
from scipy.stats import linregress, t as tdist
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

# === 設定 ===
# SPARCデータディレクトリ（ローカル環境に合わせて変更）
SPARC_DIR = Path(r"D:¥ドキュメント¥エントロピー¥新膜宇宙論¥これまでの軌跡¥パイソン¥Rotmod_LTG")
# 代替パス (Claude Code環境)
if not SPARC_DIR.exists():
    SPARC_DIR = Path("Rotmod_LTG")
if not SPARC_DIR.exists():
    SPARC_DIR = Path(".")

OUTDIR = Path("jackknife_output")
OUTDIR.mkdir(exist_ok=True)

# 物理定数
G_SI = 6.674e-11
Msun = 1.989e30
pc = 3.086e16
kpc = 1e3 * pc
a0 = 1.2e-10  # m/s²

N_JACK = 1000  # ジャックナイフ反復回数
RNG_SEED = 42


def load_sparc_galaxy(filepath):
    """
    SPARC Rotmod_LTG ファイルを読み込む。
    カラム: R(kpc), Vobs, errV, Vgas, Vdisk, Vbul
    """
    try:
        data = np.loadtxt(filepath, comments="#")
    except:
        return None

    if data.ndim != 2 or data.shape[1] &lt; 5:
        return None
```

```
    return {
        "R_kpc": data[:, 0],
        "V_obs": data[:, 1],
        "e_V": data[:, 2],
        "V_gas": data[:, 3],
        "V_disk": data[:, 4],
        "V_bul": data[:, 5] if data.shape[1] &gt; 5 else np.zeros(len(data)),
    }


def compute_gc_sparc(gal, Yd=0.5, Yb=0.7):
    """
    SPARC銀河からg_cとΣ□を測定。
    V_bar² = V_gas² + Yd×V_disk² + Yb×V_bul²
    g_c = V_obs²/R - V_bar²/R  (外側1/3の中央値)
    """
    R = gal["R_kpc"]
    Vo = gal["V_obs"]
    Vg = gal["V_gas"]
    Vd = gal["V_disk"]
    Vb = gal["V_bul"]

    n = len(R)
    if n &lt; 5:
        return None

    # V_bar
    V_bar2 = Vg**2 + Yd * Vd**2 + Yb * Vb**2
    V_bar = np.sqrt(np.maximum(V_bar2, 0))

    # 外側1/3
    outer = slice(2*n//3, n)
    R_m = R[outer] * kpc
    g_obs = (Vo[outer] * 1e3)**2 / R_m
    g_bar = (V_bar[outer] * 1e3)**2 / R_m

    gc_vals = g_obs - g_bar
    gc = np.median(gc_vals)

    if gc &lt;= 0:
        return None

    # Σ□: ディスクの面密度（中心値推定）
    # Σ□ ∝ V_disk(R_peak)² / (G × h) の近似
    # ここではg_c × a□ の幾何平均のΣ□部分を直接推定
    # SPARCのg_c vs a□×G×Σ□ の回帰ではΣ□は独立に推定する必要がある

    # 簡易推定: Σ□ = Yd × V_disk_max² / (2πG h)
    # h ≈ R(V_disk_max) / 2.2 (指数ディスクのピーク位置)
    idx_peak = np.argmax(np.abs(Vd))
    if idx_peak == 0:
        idx_peak = 1
    R_peak = R[idx_peak]
    h_kpc = R_peak / 2.2
    V_d_peak = abs(Vd[idx_peak])

    if h_kpc &gt; 0 and V_d_peak &gt; 0:
        h_m = h_kpc * kpc
        Sigma0 = Yd * (V_d_peak * 1e3)**2 / (2 * np.pi * G_SI * h_m)
        Sigma0_Mpc2 = Sigma0 * pc**2 / Msun  # M_sun/pc²
    else:
        return None

    if Sigma0_Mpc2 &lt;= 0:
        return None

    mad = np.median(np.abs(gc_vals - gc))
    e_gc = 1.4826 * mad / np.sqrt(len(gc_vals))

    return {
        "gc": gc,
        "e_gc": e_gc,
        "Sigma0": Sigma0_Mpc2,
        "h_kpc": h_kpc,
        "V_flat": np.median(Vo[outer]),
    }


def alpha_fit(gc_arr, Sigma0_arr):
    """log(g_c) = log(η) + α × log(a□·G·Σ□) の線形回帰"""
    x = np.log10(a0 * G_SI * Sigma0_arr * Msun / pc**2)
    y = np.log10(gc_arr)

    mask = np.isfinite(x) &amp;amp; np.isfinite(y)
```

```
    x, y = x[mask], y[mask]

    if len(x) &amp;lt; 5:
        return None

    slope, intercept, r, p, se = linregress(x, y)
    t_stat = (slope - 0.5) / se
    p_05 = 2 * tdist.sf(abs(t_stat), df=len(x)-2)

    return {
        "alpha": slope, "e_alpha": se, "p05": p_05,
        "r": r, "N": len(x), "intercept": intercept,
    }


def main():
    print("=" * 60)
    print("(B) SPARC拡張ジャックナイフ検証")
    print(f"    N_iterations = {N_JACK}")
    print("=" * 60)

    # === SPARC全銀河読み込み ===
    print(f"¥nSPARCデータ読み込み: {SPARC_DIR}")

    files = sorted(SPARC_DIR.glob("*.dat"))
    if not files:
        files = sorted(SPARC_DIR.glob("*_rotmod.dat"))
    if not files:
        print(f"  ☒ .datファイルが見つかりません: {SPARC_DIR}")
        print(f"  ディレクトリ内容: {list(SPARC_DIR.iterdir())[:10]}")
        sys.exit(1)

    print(f"  {len(files)} ファイル検出")

    galaxies = []
    for f in files:
        gal = load_sparc_galaxy(f)
        if gal is None:
            continue
        result = compute_gc_sparc(gal)
        if result is None:
            continue
        result["name"] = f.stem
        galaxies.append(result)

    print(f"  → {len(galaxies)} 銀河で g_c 測定成功")

    if len(galaxies) &amp;lt; 20:
        print("  ☒ 銀河数不足。パスを確認してください。")
        sys.exit(1)

    gc_all = np.array([g["gc"] for g in galaxies])
    S0_all = np.array([g["Sigma0"] for g in galaxies])
    names_all = [g["name"] for g in galaxies]
    N_gal = len(galaxies)

    # === 全体フィット ===
    print("¥n--- 全体フィット ---")
    full = alpha_fit(gc_all, S0_all)
    if full:
        print(f"  α = {full['alpha']:.3f} ± {full['e_alpha']:.3f}")
        print(f"  p(α=0.5) = {full['p05']:.4f}")
        print(f"  N = {full['N']}, r = {full['r']:.3f}")

    # === 半分割ジャックナイフ ×1000 ===
    print(f"¥n--- 半分割ジャックナイフ ×{N_JACK} ---")
    rng = np.random.RandomState(RNG_SEED)

    alpha_train = []
    alpha_test = []
    p05_train = []
    p05_test = []
    delta_alpha = []

    for i in range(N_JACK):
        idx = rng.permutation(N_gal)
        half = N_gal // 2
        tr = idx[:half]
        te = idx[half:]

        fit_tr = alpha_fit(gc_all[tr], S0_all[tr])
        fit_te = alpha_fit(gc_all[te], S0_all[te])

        if fit_tr and fit_te:
```

```
        alpha_train.append(fit_tr["alpha"])
        alpha_test.append(fit_te["alpha"])
        p05_train.append(fit_tr["p05"])
        p05_test.append(fit_te["p05"])
        delta_alpha.append(fit_tr["alpha"] - fit_te["alpha"])

alpha_train = np.array(alpha_train)
alpha_test = np.array(alpha_test)
p05_train = np.array(p05_train)
p05_test = np.array(p05_test)
delta_alpha = np.array(delta_alpha)

n_valid = len(alpha_train)
print(f"  有効反復: {n_valid}/{N_JACK}")
print(f"¥n  α(train): {np.mean(alpha_train):.3f} ± {np.std(alpha_train):.3f}")
print(f"  α(test):  {np.mean(alpha_test):.3f} ± {np.std(alpha_test):.3f}")
print(f"  Δα = α(train)-α(test): {np.mean(delta_alpha):.4f} ± {np.std(delta_alpha):.4f}")
print(f"  |Δα| &lt; 0.05 の割合: {np.mean(np.abs(delta_alpha)&lt;0.05)*100:.1f}%")
print(f"  α=0.5 棄却率 (train, p&lt;0.05): {np.mean(p05_train&lt;0.05)*100:.1f}%")
print(f"  α=0.5 棄却率 (test, p&lt;0.05):  {np.mean(p05_test&lt;0.05)*100:.1f}%")

# === Leave-N-out ===
print(f"¥n--- Leave-N-out 安定性 ---")
for N_leave in [10, 20, 50]:
    if N_leave &gt;= N_gal:
        continue
    alphas = []
    for i in range(500):
        keep = rng.choice(N_gal, N_gal - N_leave, replace=False)
        fit = alpha_fit(gc_all[keep], S0_all[keep])
        if fit:
            alphas.append(fit["alpha"])
    alphas = np.array(alphas)
    print(f"  Leave-{N_leave}-out (×500): α = {np.mean(alphas):.3f} ± {np.std(alphas):.3f}, "
          f"range [{np.min(alphas):.3f}, {np.max(alphas):.3f}]")

# === Bootstrap (参考)  ===
print(f"¥n--- Bootstrap ×{N_JACK} ---")
alpha_boot = []
for i in range(N_JACK):
    idx = rng.choice(N_gal, N_gal, replace=True)
    fit = alpha_fit(gc_all[idx], S0_all[idx])
    if fit:
        alpha_boot.append(fit["alpha"])
alpha_boot = np.array(alpha_boot)
print(f"  α(boot): {np.mean(alpha_boot):.3f} ± {np.std(alpha_boot):.3f}")
print(f"  95% CI: [{np.percentile(alpha_boot,2.5):.3f}, {np.percentile(alpha_boot,97.5):.3f}]")
print(f"  α=0.5 は CI 内: {'YES' if np.percentile(alpha_boot,2.5)&lt;=0.5&lt;=np.percentile(alpha_boot,97.5) else 'NO'}")

# === 図の生成 ===
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# (a) α(train) vs α(test) 散布図
ax = axes[0, 0]
ax.scatter(alpha_train, alpha_test, alpha=0.2, s=5)
ax.axhline(0.5, color="red", ls="--", label="alpha=0.5")
ax.axvline(0.5, color="red", ls="--")
ax.plot([0.2, 0.9], [0.2, 0.9], "k--", alpha=0.3)
ax.set_xlabel("alpha (train)")
ax.set_ylabel("alpha (test)")
ax.set_title(f"Half-split x{N_JACK}")
ax.legend()

# (b) Δα分布
ax = axes[0, 1]
ax.hist(delta_alpha, bins=50, edgecolor="black", alpha=0.7)
ax.axvline(0, color="red", ls="--")
ax.set_xlabel("Delta alpha = alpha(train) - alpha(test)")
ax.set_ylabel("Count")
ax.set_title(f"mean={np.mean(delta_alpha):.4f}, std={np.std(delta_alpha):.4f}")

# (c) α(train)分布
ax = axes[1, 0]
ax.hist(alpha_train, bins=50, alpha=0.6, label="train", color="blue")
ax.hist(alpha_test, bins=50, alpha=0.6, label="test", color="orange")
ax.axvline(0.5, color="red", ls="--", label="alpha=0.5")
ax.set_xlabel("alpha")
ax.set_ylabel("Count")
ax.set_title("alpha distribution")
ax.legend()

# (d) Bootstrap分布
ax = axes[1, 1]
ax.hist(alpha_boot, bins=50, edgecolor="black", alpha=0.7, color="green")
```

```
    ax.axvline(0.5, color="red", ls="--", label="alpha=0.5")
    ci = np.percentile(alpha_boot, [2.5, 97.5])
    ax.axvline(ci[0], color="red", ls=":", alpha=0.5)
    ax.axvline(ci[1], color="red", ls=":", alpha=0.5)
    ax.set_xlabel("alpha (bootstrap)")
    ax.set_ylabel("Count")
    ax.set_title(f"Bootstrap 95% CI: [{ci[0]:.3f}, {ci[1]:.3f}]")
    ax.legend()

    plt.tight_layout()
    fig_path = OUTDIR / "jackknife_extended.png"
    plt.savefig(fig_path, dpi=150)
    print(f"¥n図保存: {fig_path}")

    # === 結果JSON保存 ===
    summary = {
        "full_fit": full,
        "jackknife": {
            "N_iter": n_valid,
            "alpha_train_mean": float(np.mean(alpha_train)),
            "alpha_train_std": float(np.std(alpha_train)),
            "alpha_test_mean": float(np.mean(alpha_test)),
            "alpha_test_std": float(np.std(alpha_test)),
            "delta_alpha_mean": float(np.mean(delta_alpha)),
            "delta_alpha_std": float(np.std(delta_alpha)),
            "reject_rate_train": float(np.mean(p05_train &lt; 0.05)),
            "reject_rate_test": float(np.mean(p05_test &lt; 0.05)),
        },
        "bootstrap": {
            "alpha_mean": float(np.mean(alpha_boot)),
            "alpha_std": float(np.std(alpha_boot)),
            "CI95": [float(ci[0]), float(ci[1])],
        },
    }
    with open(OUTDIR / "jackknife_summary.json", "w") as f:
        json.dump(summary, f, indent=2)

    print(f"結果保存: {OUTDIR / 'jackknife_summary.json'}")


if __name__ == "__main__":
    main()
```

probes_vbar_pipeline.py

章: 第24章前提

目的: PROBES x S4G クロスマッチによる概算V_bar構築

解析対象データ: PROBES (Stone+2021, Zenodo), S4G (Salo+2015, VizieR)

主要結果: 290銀河マッチ(284 SPARC独立)。固定Yd=0.5でalpha=1.056(棄却)。概算V_barではalpha=0.5再現不可。

ソースコード (614 lines, 18,823 bytes)

```
#!/usr/bin/env python3
"""
probes_vbar_pipeline.py
=======================
PROBES V_obs回転曲線 + WISE/S4G 3.6μm表面輝度プロファイル + HI 21cmデータ
から自前V_barを構築し、幾何平均法則 α=0.5 を検証する。

実行: uv run --with scipy --with matplotlib --with astropy --with requests python probes_vbar_pipeline.py

戦略:
  Step 0: PROBES RC取得 (CADC)
  Step 1: S4G 3.6μm表面輝度プロファイル取得 (VizieR: J/PASP/127/299 Muñoz-Mateos+2015)
  Step 2: THINGS HI表面密度プロファイル取得 (VizieR: J/AJ/136/2563 Walter+2008)
  Step 3: PROBES × S4G × THINGS クロスマッチ (名前ベース)
  Step 4: V_disk(R) = sqrt(Υ_disk × GM_disk(&lt;R)/R) を表面輝度から計算
  Step 5: V_gas(R) = sqrt(1.33 × GM_gas(&lt;R)/R) をHI面密度から計算
  Step 6: V_bar = sqrt(V_disk² + V_gas²)
  Step 7: g_c測定 + α検定 (SPARC同一手法)

注意:
  - Υ_disk = 0.5 M_sun/L_sun (SPARC準拠)
  - HI質量には He補正 ×1.33
  - 薄いディスク近似で V_disk を計算 (厳密にはFreeman disk積分だが、
    累積質量 / R 近似を第一段階とする)
"""

import os
import sys
import json
import numpy as np
from pathlib import Path

# === 設定 ===
OUTDIR = Path("probes_vbar_output")
OUTDIR.mkdir(exist_ok=True)

# 物理定数
G_CGS = 6.674e-8        # cm³/(g·s²)
G_SI  = 6.674e-11       # m³/(kg·s²)
Msun  = 1.989e30        # kg
pc    = 3.086e16         # m
kpc   = 1e3 * pc        # m
Lsun_36 = 1.19e26       # W (3.6μm太陽光度、Vega系)

# SPARC準拠の仮定
Y_disk = 0.5  # M_sun / L_sun (3.6μm)


# ============================================================
# Step 0: PROBES回転曲線の取得
# ============================================================
def step0_fetch_probes():
    """PROBESカタログからV_obs回転曲線を取得"""
    import requests

    print("=" * 60)
    print("Step 0: PROBES回転曲線の取得")
    print("=" * 60)

    # PROBES on CADC — まず利用可能なファイルリストを確認
    # Stone+2021: https://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/en/community/PROBES/
    # 代替: VizieR J/ApJS/256/33

    vizier_url = "https://vizier.cds.unistra.fr/viz-bin/asu/tsv"

    # まずマスターテーブル (銀河パラメータ) を取得
    params_master = {
        "-source": "J/ApJS/256/33/table1",
        "-out.max": "5000",
        "-out": "Name,RAJ2000,DEJ2000,Dist,Vmax,incl,PA,Ref",
```

```
        "-out.form": "tsv",
    }

    print("  VizieRからPROBESマスターテーブル取得中...")
    try:
        r = requests.get(vizier_url, params=params_master, timeout=60)
        master_file = OUTDIR / "probes_master.tsv"
        master_file.write_text(r.text)

        # パース
        lines = [l for l in r.text.strip().split("¥n") if l and not l.startswith("#")]
        header_idx = None
        for i, l in enumerate(lines):
            if "Name" in l and "RAJ2000" in l:
                header_idx = i
                break

        if header_idx is None:
            print("  ☒ マスターテーブルのヘッダが見つかりません")
            print(f"  最初の5行:¥n{chr(10).join(lines[:5])}")
            return None

        headers = lines[header_idx].split("¥t")
        data_lines = [l for l in lines[header_idx+1:] if l.strip() and not l.startswith("-")]

        galaxies = []
        for l in data_lines:
            cols = l.split("¥t")
            if len(cols) &gt;= len(headers):
                d = dict(zip(headers, cols))
                galaxies.append(d)

        print(f"  → {len(galaxies)} 銀河取得")

        # JSON保存
        with open(OUTDIR / "probes_galaxies.json", "w") as f:
            json.dump(galaxies, f, indent=2)

        return galaxies

    except Exception as e:
        print(f"  ☒ PROBES取得失敗: {e}")
        return None


def step0b_fetch_probes_rc(galaxy_names):
    """PROBES個別回転曲線の取得 (VizieR rcテーブル) """
    import requests

    print("¥n  PROBES回転曲線テーブル取得中...")

    vizier_url = "https://vizier.cds.unistra.fr/viz-bin/ase/tsv"
    params_rc = {
        "-source": "J/ApJS/256/33/table2",
        "-out.max": "999999",
        "-out": "Name,Rad,Vrot,e_Vrot",
        "-out.form": "tsv",
    }

    try:
        r = requests.get(vizier_url, params=params_rc, timeout=120)
        rc_file = OUTDIR / "probes_rc.tsv"
        rc_file.write_text(r.text)

        lines = [l for l in r.text.strip().split("¥n") if l and not l.startswith("#")]
        header_idx = None
        for i, l in enumerate(lines):
            if "Name" in l and "Rad" in l:
                header_idx = i
                break

        if header_idx is None:
            print("  ☒ RCテーブルのヘッダが見つかりません")
            return None

        headers = lines[header_idx].split("¥t")
        data_lines = [l for l in lines[header_idx+1:] if l.strip() and not l.startswith("-")]

        # 銀河ごとにグループ化
        rc_data = {}
        for l in data_lines:
            cols = l.split("¥t")
            if len(cols) &gt;= 4:
                name = cols[0].strip()
```

```
                try:
                    rad = float(cols[1])
                    vrot = float(cols[2])
                    e_vrot = float(cols[3]) if cols[3].strip() else 0.0
                    if name not in rc_data:
                        rc_data[name] = {"R_arcsec": [], "V_obs": [], "e_V": []}
                    rc_data[name]["R_arcsec"].append(rad)
                    rc_data[name]["V_obs"].append(vrot)
                    rc_data[name]["e_V"].append(e_vrot)
                except ValueError:
                    pass

        print(f"  → {len(rc_data)} 銀河の回転曲線取得")

        # numpy配列に変換
        for name in rc_data:
            for k in rc_data[name]:
                rc_data[name][k] = np.array(rc_data[name][k])

        return rc_data

    except Exception as e:
        print(f"  ☒ RC取得失敗: {e}")
        return None


# ======================================================================
# Step 1: S4G 3.6μm表面輝度プロファイル
# ======================================================================
def step1_fetch_s4g():
    """S4G 3.6μm表面輝度プロファイル (Muñoz-Mateos+2015) """
    import requests

    print("¥n" + "=" * 60)
    print("Step 1: S4G 3.6μm表面輝度プロファイル取得")
    print("=" * 60)

    vizier_url = "https://vizier.cds.unistra.fr/viz-bin/ase/tsv"

    # S4G P4 プロファイルテーブル (Muñoz-Mateos+2015)
    # J/ApJ/799/213 にプロファイルデータあり
    # 代替: Salo+2015 J/ApJS/219/4 (分解モデル)

    # まず銀河リスト
    params = {
        "-source": "J/ApJS/219/4/table1",  # Salo+2015 S4G decomposition
        "-out.max": "5000",
        "-out": "Name,T,Dist,h,mu0,PA,b/a",
        "-out.form": "tsv",
    }

    print("  VizieRからS4G分解パラメータ取得中...")
    try:
        r = requests.get(vizier_url, params=params, timeout=60)
        s4g_file = OUTDIR / "s4g_params.tsv"
        s4g_file.write_text(r.text)

        lines = [l for l in r.text.strip().split("¥n") if l and not l.startswith("#")]
        header_idx = None
        for i, l in enumerate(lines):
            if "Name" in l:
                header_idx = i
                break

        if header_idx is None:
            # 別のテーブルを試す
            print("  Salo+2015見つからず、Muñoz-Mateos+2015を試行...")
            params2 = {
                "-source": "J/ApJ/799/213",
                "-out.max": "5000",
                "-out.form": "tsv",
            }
            r2 = requests.get(vizier_url, params=params2, timeout=60)
            s4g_file2 = OUTDIR / "s4g_munoz.tsv"
            s4g_file2.write_text(r2.text)
            print(f"  → レスポンス {len(r2.text)} bytes")
            print(f"  最初の5行:¥n{chr(10).join(r2.text.split(chr(10))[:5])}")
            return None

        headers = lines[header_idx].split("¥t")
        data_lines = [l for l in lines[header_idx+1:] if l.strip() and not l.startswith("-")]

        s4g_galaxies = {}
        for l in data_lines:
```

```
            cols = l.split("¥t")
            if len(cols) &gt;= len(headers):
                d = dict(zip(headers, cols))
                name = d.get("Name", "").strip()
                if name:
                    s4g_galaxies[name] = d

        print(f"  → {len(s4g_galaxies)} 銀河の分解パラメータ取得")
        return s4g_galaxies

    except Exception as e:
        print(f"  ☒ S4G取得失敗: {e}")
        return None


# ======================================================================
# Step 2: THINGS HI表面密度プロファイル
# ======================================================================
def step2_fetch_things_hi():
    """THINGS HI表面密度プロファイル (Walter+2008, de Blok+2008) """
    import requests

    print("¥n" + "=" * 60)
    print("Step 2: THINGS HI表面密度プロファイル取得")
    print("=" * 60)

    vizier_url = "https://vizier.cds.unistra.fr/viz-bin/ase/tsv"

    # de Blok+2008 J/AJ/136/2648 — THINGS回転曲線 (V_obsだけでなくV_gas, V_disk分解あり!)
    params = {
        "-source": "J/AJ/136/2648/table5",
        "-out.max": "99999",
        "-out.form": "tsv",
    }

    print("  VizieRからTHINGS分解回転曲線取得中...")
    try:
        r = requests.get(vizier_url, params=params, timeout=60)
        things_file = OUTDIR / "things_rc.tsv"
        things_file.write_text(r.text)

        print(f"  → レスポンス {len(r.text)} bytes")

        lines = [l for l in r.text.strip().split("¥n") if l and not l.startswith("#")]

        # ヘッダ探索
        header_idx = None
        for i, l in enumerate(lines):
            if "Name" in l or "Galaxy" in l or "Rad" in l:
                header_idx = i
                break

        if header_idx is not None:
            headers = lines[header_idx].split("¥t")
            print(f"  ヘッダ: {headers}")
            data_lines = [l for l in lines[header_idx+1:] if l.strip() and not l.startswith("-")]
            print(f"  → {len(data_lines)} 行")
        else:
            print(f"  最初の10行:¥n{chr(10).join(lines[:10])}")

        return things_file

    except Exception as e:
        print(f"  ☒ THINGS取得失敗: {e}")
        return None


def step2b_fetch_things_decomposed():
    """
    THINGS回転曲線のバリオン分解版を探す。
    de Blok+2008はV_obs, V_gas, V_disk, V_bulを個別に報告している可能性がある。
    代替: Frank+2016 (J/AJ/151/94) — THINGS HI面密度プロファイル
    """
    import requests

    vizier_url = "https://vizier.cds.unistra.fr/viz-bin/ase/tsv"

    # Frank+2016: THINGS HI面密度プロファイル
    params = {
        "-source": "J/AJ/151/94",
        "-out.max": "99999",
        "-out.form": "tsv",
    }
    print("¥n  Frank+2016 THINGS HI面密度プロファイル取得中...")
```

```
    try:
        r = requests.get(vizier_url, params=params, timeout=60)
        f = OUTDIR / "things_hi_profiles.tsv"
        f.write_text(r.text)
        print(f"  → レスポンス {len(r.text)} bytes")
        lines = r.text.strip().split("¥n")[:10]
        print(f"  最初の10行:¥n{chr(10).join(lines)}")
        return f
    except Exception as e:
        print(f"  ☒ 取得失敗: {e}")
        return None


# ======================================================================
# Step 3: クロスマッチ
# ======================================================================
def normalize_name(name):
    """銀河名を正規化してクロスマッチ"""
    import re
    n = name.strip().upper()
    # 共通の置換
    n = n.replace("NGC ", "NGC").replace("NGC-", "NGC")
    n = n.replace("UGC ", "UGC").replace("UGC-", "UGC")
    n = n.replace("IC ", "IC").replace("IC-", "IC")
    n = n.replace("DDO ", "DDO").replace("DDO-", "DDO")
    n = re.sub(r"¥s+", "", n)
    return n


def step3_crossmatch(probes_names, s4g_names, things_names):
    """3カタログの名前ベースクロスマッチ"""
    print("¥n" + "=" * 60)
    print("Step 3: クロスマッチ")
    print("=" * 60)

    p_norm = {normalize_name(n): n for n in probes_names}
    s_norm = {normalize_name(n): n for n in s4g_names}
    t_norm = {normalize_name(n): n for n in things_names}

    # PROBES ∩ S4G ∩ THINGS
    common_3 = set(p_norm.keys()) &amp;amp; set(s_norm.keys()) &amp;amp; set(t_norm.keys())
    # PROBES ∩ S4G (THINGSなしでもV_diskは計算可能)
    common_ps = set(p_norm.keys()) &amp;amp; set(s_norm.keys())

    print(f"  PROBES: {len(p_norm)}")
    print(f"  S4G: {len(s_norm)}")
    print(f"  THINGS: {len(t_norm)}")
    print(f"  PROBES ∩ S4G: {len(common_ps)}")
    print(f"  PROBES ∩ S4G ∩ THINGS: {len(common_3)}")

    matches = {
        "all_three": {k: (p_norm[k], s_norm[k], t_norm[k]) for k in common_3},
        "probes_s4g": {k: (p_norm[k], s_norm[k]) for k in common_ps},
    }

    if common_3:
        print(f"¥n  3カタログ共通銀河:")
        for k in sorted(common_3)[:20]:
            print(f"    {p_norm[k]}")

    return matches


# ======================================================================
# Step 4-6: V_bar構築
# ======================================================================
def compute_vdisk_exponential(R_kpc, h_kpc, mu0_36, dist_Mpc):
    """
    指数ディスクからV_disk(R)を計算。

    Freeman (1970) の薄いディスク公式:
    V_disk²(R) = 4π G Y_disk I☒ h × y² × [I☒(y)K☒(y) - I☒(y)K☒(y)]
    ただし y = R/(2h), I_n/K_n は修正ベッセル関数

    Parameters:
        R_kpc: 半径配列 [kpc]
        h_kpc: スケール長 [kpc]
        mu0_36: 中心表面輝度 [mag/arcsec²] (3.6μm AB or Vega)
        dist_Mpc: 距離 [Mpc]
    """
    from scipy.special import i0, i1, k0, k1

    # 表面輝度 → 面光度密度
    # μ(R) = mu0 + 1.0857 × R/h  [mag/arcsec²]
```

```
    # I□ [L_sun/pc²] from mu0
    # 3.6μm Vega zero point: M_sun(3.6μm) ≈ 3.24 (Oh+2008)
    M_sun_36 = 3.24  # AB系の場合は調整必要

    # I□ = 10^(-0.4*(mu0 - M_sun_36 - 21.572))  [L_sun/pc²]
    # 21.572 = 5*log10(206265) - 5  (arcsec² → pc² at 10pc)
    I0_Lpc2 = 10**(-0.4 * (mu0_36 - M_sun_36 - 21.572))

    # Σ□ = Υ_disk × I□ [M_sun/pc²]
    Sigma0 = Y_disk * I0_Lpc2

    # Freeman disk
    y = R_kpc / (2.0 * h_kpc)
    y = np.clip(y, 1e-6, 50)  # 数値安全

    bessel_term = i0(y) * k0(y) - i1(y) * k1(y)

    # V_disk² = 4π G Σ□ h × y² × bessel_term
    # 単位変換: Σ□ [M_sun/pc²], h [kpc]
    h_m = h_kpc * kpc
    Sigma0_SI = Sigma0 * Msun / (pc**2)

    V2 = 4 * np.pi * G_SI * Sigma0_SI * h_m * y**2 * bessel_term
    V_disk = np.sqrt(np.maximum(V2, 0)) / 1e3  # m/s → km/s

    return V_disk, Sigma0


def compute_vgas_from_sigma(R_kpc, Sigma_HI_Mpc2, h_gas_kpc=None):
    """
    HI面密度プロファイルからV_gas(R)を計算。

    簡易版: 累積質量近似 V_gas² = 1.33 × G × M_gas(&lt;R) / R
     (薄いディスクのFreeman公式を適用する方が正確だが、
      HIプロファイルは指数関数でないため累積質量近似を使用)

    Parameters:
        R_kpc: 半径配列 [kpc]
        Sigma_HI_Mpc2: HI面密度 [M_sun/pc²] at each R
    """
    R_m = R_kpc * kpc
    Sigma_SI = Sigma_HI_Mpc2 * Msun / (pc**2)

    # 累積質量 (台形積分)
    M_cum = np.zeros_like(R_kpc)
    for i in range(1, len(R_kpc)):
        dR = R_m[i] - R_m[i-1]
        R_mid = 0.5 * (R_m[i] + R_m[i-1])
        Sigma_mid = 0.5 * (Sigma_SI[i] + Sigma_SI[i-1])
        M_cum[i] = M_cum[i-1] + 2 * np.pi * R_mid * Sigma_mid * dR

    # He補正 × 1.33
    M_cum *= 1.33

    V2 = G_SI * M_cum / np.maximum(R_m, 1e-10)
    V_gas = np.sqrt(np.maximum(V2, 0)) / 1e3  # km/s

    return V_gas


# ======================================================================
# Step 7: g_c測定 + α検定
# ======================================================================
def measure_gc(R_kpc, V_obs, V_bar, dist_Mpc):
    """
    g_c = V_obs²(R_flat) / R_flat での遠心力加速度のうち
    バリオンで説明できない成分を測定。

    SPARC準拠の手法:
    g_obs = V_obs² / R
    g_bar = V_bar² / R
    g_c = g_obs - g_bar  (外側フラット領域で評価)
    """
    # フラット領域の特定 (外側1/3)
    n = len(R_kpc)
    if n &lt; 5:
        return None

    outer = slice(2*n//3, n)
    R_out = R_kpc[outer] * kpc  # m
    V_obs_out = V_obs[outer] * 1e3  # m/s
    V_bar_out = V_bar[outer] * 1e3  # m/s

    g_obs = V_obs_out**2 / R_out
```

```
    g_bar = V_bar_out**2 / R_out
    g_c = np.median(g_obs - g_bar)

    # 中央値の不確かさ (MADベース)
    gc_values = g_obs - g_bar
    mad = np.median(np.abs(gc_values - g_c))
    e_gc = 1.4826 * mad / np.sqrt(len(gc_values))

    return g_c, e_gc


def alpha_test(gc_array, a0, G_val, Sigma0_array):
    """
    g_c = η × (a□ × G × Σ□)^α のαをフィット。
    log(g_c) = log(η) + α × log(a□ × G × Σ□)
    """
    from scipy.stats import linregress

    x = np.log10(a0 * G_val * Sigma0_array)
    y = np.log10(gc_array)

    mask = np.isfinite(x) &amp; np.isfinite(y) &amp; (gc_array &gt; 0) &amp; (Sigma0_array &gt; 0)
    x, y = x[mask], y[mask]

    if len(x) &lt; 5:
        return None

    slope, intercept, r, p, se = linregress(x, y)

    # α=0.5検定
    from scipy.stats import t as tdist
    t_stat = (slope - 0.5) / se
    p_05 = 2 * tdist.sf(abs(t_stat), df=len(x)-2)

    return {
        "alpha": slope,
        "e_alpha": se,
        "log_eta": intercept,
        "r": r,
        "p_alpha05": p_05,
        "N": len(x),
    }


# ======================================================================
# メイン
# ======================================================================
def main():
    print("膜宇宙論 v4.1 — PROBES自前V_bar構築パイプライン")
    print("=" * 60)

    # --- Step 0: PROBES取得 ---
    galaxies = step0_fetch_probes()
    if galaxies is None:
        print("¥n□ PROBESマスターテーブル取得失敗。VizieRテーブル名の確認が必要。")
        print("  手動確認: https://vizier.cds.unistra.fr/cgi-bin/VizieR?-source=J/ApJS/256/33")
        return

    galaxy_names = [g.get("Name", "").strip() for g in galaxies if g.get("Name")]

    # --- Step 0b: PROBES RC取得 ---
    rc_data = step0b_fetch_probes_rc(galaxy_names)
    if rc_data is None:
        print("¥n□ PROBES RC取得失敗。")
        return

    # --- Step 1: S4G取得 ---
    s4g = step1_fetch_s4g()

    # --- Step 2: THINGS HI取得 ---
    things = step2_fetch_things_hi()
    step2b_fetch_things_decomposed()

    # --- Step 3: クロスマッチ ---
    s4g_names = list(s4g.keys()) if s4g else []
    things_names = []  # THINGSの銀河名はStep 2の結果から抽出する必要あり

    if s4g_names:
        matches = step3_crossmatch(list(rc_data.keys()), s4g_names, things_names)
    else:
        print("¥n□ S4Gデータ取得失敗。クロスマッチをスキップ。")
        matches = None

    # --- 結果サマリ ---
```

```
    print("¥n" + "=" * 60)
    print("パイプライン結果サマリ")
    print("=" * 60)
    print(f"  PROBES RC: {len(rc_data)} 銀河")
    print(f"  S4G パラメータ: {len(s4g) if s4g else 0} 銀河")

    if matches and matches.get("probes_s4g"):
        ps_matches = matches["probes_s4g"]
        print(f"  PROBES ∩ S4G: {len(ps_matches)} 銀河")

        # S4Gのスケール長hとmu0があればV_diskを計算可能
        print("¥n  → 次のステップ:")
        print("    1. マッチした銀河でV_disk(R)をFreeman disk公式で計算")
        print("    2. THINGSまたはWHISP HI データでV_gas(R)を計算")
        print("    3. V_bar = sqrt(V_disk² + V_gas²) を構築")
        print("    4. g_c測定 + α検定")

        # 試行: S4Gのhとmu0が利用可能な銀河でV_disk計算
        print("¥n  S4Gパラメータのサンプル:")
        for i, (norm_name, (p_name, s_name)) in enumerate(list(ps_matches.items())[:5]):
            s4g_data = s4g.get(s_name, {})
            print(f"    {p_name}: h={s4g_data.get('h','?')} arcsec, "
                  f"mu0={s4g_data.get('mu0','?')} mag/arcsec²")
    else:
        print("¥n  ☒ クロスマッチ結果なし。")
        print("  考えられる原因:")
        print("    - VizieRテーブル名の誤り")
        print("    - 銀河名の表記揺れ")
        print("  → 手動確認とデバッグが必要")

    # 取得データの保存先
    print(f"¥n  出力ディレクトリ: {OUTDIR.resolve()}")
    for f in sorted(OUTDIR.iterdir()):
        print(f"    {f.name}: {f.stat().st_size:,} bytes")


if __name__ == "__main__":
    main()
```

probes_Yd_free.py

章: 第24章

目的: PROBES 290銀河でYdをフリーパラメータとして最適化

解析対象データ: PROBES + S4G (VizieR)

主要結果: Yd=0.3: alpha=0.95, Yd=0.5: alpha=1.07, Free: alpha=0.76(N=18)。Ydがalphaを制御するトレンドを確認(Sigma0定義問題を含む)。

ソースコード (469 lines, 14,523 bytes)

```
#!/usr/bin/env python3
"""
probes_Yd_free.py
=================
PROBES 290銀河（概算V_bar構築済み）で Υ_disk をフリーパラメータとして
銀河ごとに最適化し、α=0.5 が再現されるかを検証する。

方法:
  各銀河について、Υ_d を 0.1-1.5 の範囲でグリッドサーチし、
  V_bar(R; Υ_d) = sqrt(Υ_d × V_disk² + V_gas²) として
  V_obs との残差 chi² を最小化する Υ_d^opt を決定。
  → Υ_d^opt で再計算した g_c と G·Σ□ で α を再フィット。

追加テスト:
  - Υ_d固定 (0.3, 0.5, 0.7) でのα比較（感度テスト）
  - Υ_d^opt の分布（SPARCの0.3-0.9と整合するか）

実行: uv run --with scipy --with matplotlib --with numpy python probes_Yd_free.py

前提: probes_vbar_pipeline.py の出力ファイルが probes_vbar_output/ に存在すること。
      存在しない場合は、VizieRから再取得してインメモリで処理する。
"""

import numpy as np
import json
import sys
from pathlib import Path
from scipy.optimize import minimize_scalar
from scipy.stats import linregress, t as tdist

# 物理定数
G_SI = 6.674e-11
Msun = 1.989e30
pc = 3.086e16
kpc = 1e3 * pc
a0 = 1.2e-10  # m/s² (MOND加速度)

OUTDIR = Path("probes_vbar_output")


def load_pipeline_results():
    """前回パイプラインの結果をロード（JSON or 再構築）"""

    # 前回の出力ファイルを探す
    results_file = OUTDIR / "vbar_results.json"
    rc_file = OUTDIR / "probes_rc_data.json"
    s4g_file = OUTDIR / "s4g_matched.json"

    # ファイルがあれば読む、なければ再構築が必要
    # ここでは前回パイプラインのデータ構造を期待
    # 実際にはパイプラインの中間結果を保存するよう修正が必要

    print("前回パイプライン結果の検索中...")

    found_files = list(OUTDIR.glob("*.json")) + list(OUTDIR.glob("*.tsv"))
    print(f"  {OUTDIR} 内のファイル:")
    for f in found_files:
        print(f"    {f.name}: {f.stat().st_size:,} bytes")

    return found_files


def fetch_and_build_vbar():
    """
    VizieRから直接取得して V_bar を構築（パイプライン統合版）。
    前回と同じデータを使うが、Υ_d をパラメータとして扱う。
    """
    import requests
    print("¥n=== VizieRからデータ再取得 ===")
```

```
vizier = "https://vizier.cds.unistra.fr/viz-bin/ase/tsv"

# --- PROBES RC ---
print("PROBES RC取得中...")
r = requests.get(vizier, params={
    "-source": "J/ApJS/256/33/table2",
    "-out.max": "999999",
    "-out": "Name,Rad,Vrot,e_Vrot",
    "-out.form": "tsv",
}, timeout=120)

rc_data = {}
lines = r.text.strip().split("¥n")
header_idx = next((i for i, l in enumerate(lines) if "Name" in l and "Rad" in l), None)
if header_idx is None:
    # ヘッダ形式が違う場合のフォールバック
    for i, l in enumerate(lines):
        if not l.startswith("#") and not l.startswith("-") and "¥t" in l:
            header_idx = i
            break

if header_idx is not None:
    data_lines = [l for l in lines[header_idx+1:]
                  if l.strip() and not l.startswith("-") and not l.startswith("#")]
    for l in data_lines:
        cols = l.split("¥t")
        if len(cols) &gt;= 4:
            name = cols[0].strip()
            try:
                rad, vrot = float(cols[1]), float(cols[2])
                e_v = float(cols[3]) if cols[3].strip() else 5.0
                if name not in rc_data:
                    rc_data[name] = {"R": [], "V": [], "eV": []}
                rc_data[name]["R"].append(rad)
                rc_data[name]["V"].append(vrot)
                rc_data[name]["eV"].append(e_v)
            except ValueError:
                pass

print(f"  → {len(rc_data)} 銀河")

# --- PROBES マスター（距離） ---
print("PROBESマスターテーブル取得中...")
r2 = requests.get(vizier, params={
    "-source": "J/ApJS/256/33/table1",
    "-out.max": "5000",
    "-out": "Name,Dist",
    "-out.form": "tsv",
}, timeout=60)

dist_map = {}
lines2 = r2.text.strip().split("¥n")
for l in lines2:
    if l.startswith("#") or l.startswith("-") or "Name" in l:
        continue
    cols = l.split("¥t")
    if len(cols) &gt;= 2:
        try:
            dist_map[cols[0].strip()] = float(cols[1])
        except ValueError:
            pass
print(f"  → {len(dist_map)} 銀河の距離")

# --- S4G (Salo+2015: Re, Tmag, n) ---
print("S4G (Salo+2015) 取得中...")
r3 = requests.get(vizier, params={
    "-source": "J/ApJS/219/4/table1",
    "-out.max": "5000",
    "-out": "Name,Re,Tmag,n,T",
    "-out.form": "tsv",
}, timeout=60)

s4g = {}
lines3 = r3.text.strip().split("¥n")
h_idx = next((i for i, l in enumerate(lines3) if "Name" in l and "Re" in l), None)
if h_idx is not None:
    for l in lines3[h_idx+1:]:
        if l.startswith("-") or l.startswith("#") or not l.strip():
            continue
        cols = l.split("¥t")
        if len(cols) &gt;= 4:
            name = cols[0].strip()
            try:
                Re = float(cols[1])  # arcsec
```

```
                Tmag = float(cols[2])  # total mag
                n_sersic = float(cols[3]) if cols[3].strip() else 1.0
                s4g[name] = {"Re": Re, "Tmag": Tmag, "n": n_sersic}
            except ValueError:
                pass
    print(f"  → {len(s4g)} 銀河")

    # --- SPARC名リスト（除外用） ---
    sparc_names = {
        "NGC7331", "NGC2403", "NGC3198", "NGC2841", "NGC6946", "NGC3521",
        "NGC925", "NGC2976", "NGC4736", "NGC5055", "NGC7793", "NGC3031",
        "NGC4826", "NGC2903", "NGC4559", "NGC1003", "NGC3109", "NGC4395",
        "DDO154", "DDO168", "IC2574", "NGC1560", "UGC2259",
        # 完全なリストではないが主要なものを含む
    }

    return rc_data, dist_map, s4g, sparc_names


def compute_vbar_components(R_arcsec, dist_Mpc, Re_arcsec, Tmag, n_sersic, Yd, f_gas=0.20):
    """
    V_disk と V_gas を計算（Y_d パラメータ付き）

    V_disk: Sersicプロファイル → 指数ディスク近似（n→h変換） → Freeman公式
    V_gas: f_gas × V_disk² の仮定（一律比率）
    """
    from scipy.special import i0, i1, k0, k1

    # Re → h (exponential scale length)
    # Sersic: Re = b_n × h  where b_n ≈ 1.678 for n=1 (exponential)
    # 一般: b_n ≈ 2n - 1/3 + 4/(405n) for n&gt;0.5
    if n_sersic &gt; 0.5:
        bn = 2 * n_sersic - 1.0/3 + 4.0/(405*n_sersic)
    else:
        bn = 1.678
    h_arcsec = Re_arcsec / bn

    # arcsec → kpc
    dist_kpc = dist_Mpc * 1e3
    h_kpc = h_arcsec * dist_kpc * np.pi / (180 * 3600)
    R_kpc = R_arcsec * dist_kpc * np.pi / (180 * 3600)

    if h_kpc &lt; 0.01 or h_kpc &gt; 100:
        return None

    # 中心表面輝度 from Tmag
    # L_total = 2π × I0 × h² for exponential disk
    # m_total = M_sun(3.6) - 2.5*log10(L_total/L_sun_at_10pc)
    # → I0 = L_total / (2π h²)
    # ここでは簡略化: mu0 = Tmag + 2.5*log10(2*pi*(Re_arcsec)²) + 0.7 (Sersic補正)
    area_arcsec2 = 2 * np.pi * Re_arcsec**2
    if area_arcsec2 &gt; 0:
        mu0 = Tmag + 2.5 * np.log10(area_arcsec2) + 0.7
    else:
        return None

    M_sun_36 = 3.24
    I0_Lpc2 = 10**(-0.4 * (mu0 - M_sun_36 - 21.572))
    Sigma0 = Yd * I0_Lpc2  # M_sun/pc²

    # Freeman disk V_disk
    y = R_kpc / (2.0 * h_kpc)
    y = np.clip(y, 1e-6, 50)
    bessel = i0(y) * k0(y) - i1(y) * k1(y)

    h_m = h_kpc * kpc
    Sigma0_SI = Sigma0 * Msun / (pc**2)

    V2_disk = 4 * np.pi * G_SI * Sigma0_SI * h_m * y**2 * bessel
    V2_disk = np.maximum(V2_disk, 0)

    # V_gas² = f_gas × V_disk² (一律仮定)
    V2_gas = f_gas * V2_disk

    # V_bar
    V2_bar = Yd * V2_disk + V2_gas  # 注: V2_disk already has Yd in Sigma0
    # 修正: Sigma0 = Yd * I0 なので V2_disk ∝ Yd already
    # V_bar² = V_disk²(Yd) + V_gas²(f_gas)
    # V_disk²(Yd) = Yd × (4πG I0 h y² B) ... I0は光度密度なのでY不含
    # 再計算:
    Sigma0_noYd = I0_Lpc2 * Msun / (pc**2)
    V2_disk_noYd = 4 * np.pi * G_SI * Sigma0_noYd * h_m * y**2 * bessel
    V2_disk_noYd = np.maximum(V2_disk_noYd, 0)
    V2_bar = Yd * V2_disk_noYd + f_gas * Yd * V2_disk_noYd
```

```
    V_bar = np.sqrt(V2_bar) / 1e3  # m/s → km/s
    V_disk = np.sqrt(Yd * V2_disk_noYd) / 1e3

    return {
        "V_bar": V_bar,
        "V_disk": V_disk,
        "R_kpc": R_kpc,
        "h_kpc": h_kpc,
        "Sigma0": Yd * I0_Lpc2,  # M_sun/pc²
        "I0_Lpc2": I0_Lpc2,
    }


def optimize_Yd(R_arcsec, V_obs, eV, dist_Mpc, Re, Tmag, n_sersic, f_gas=0.20):
    """銀河ごとにΥ_dを最適化（chi²最小化）"""

    def chi2(Yd):
        result = compute_vbar_components(R_arcsec, dist_Mpc, Re, Tmag, n_sersic, Yd, f_gas)
        if result is None:
            return 1e10
        V_bar = result["V_bar"]
        if len(V_bar) != len(V_obs):
            return 1e10
        residual = (V_obs - V_bar) / np.maximum(eV, 1.0)
        return np.sum(residual**2)

    try:
        res = minimize_scalar(chi2, bounds=(0.05, 2.0), method="bounded")
        if res.success or res.fun &lt; 1e9:
            return res.x, res.fun / max(len(V_obs) - 1, 1)
    except:
        pass
    return None, None


def measure_gc_from_vbar(R_kpc, V_obs_kms, V_bar_kms):
    """外側1/3でg_cを測定"""
    n = len(R_kpc)
    if n &lt; 5:
        return None, None

    outer = slice(2*n//3, n)
    R_m = R_kpc[outer] * kpc
    Vo = V_obs_kms[outer] * 1e3
    Vb = V_bar_kms[outer] * 1e3

    g_obs = Vo**2 / R_m
    g_bar = Vb**2 / R_m
    gc_vals = g_obs - g_bar
    gc = np.median(gc_vals)

    if gc &lt;= 0:
        return None, None

    mad = np.median(np.abs(gc_vals - gc))
    e_gc = 1.4826 * mad / np.sqrt(len(gc_vals))

    return gc, e_gc


def run_alpha_test(gc_arr, Sigma0_arr, label=""):
    """α検定"""
    x = np.log10(a0 * G_SI * Sigma0_arr * Msun / pc**2)
    y = np.log10(gc_arr)

    mask = np.isfinite(x) &amp; np.isfinite(y)
    x, y = x[mask], y[mask]

    if len(x) &lt; 10:
        print(f"  {label}: N={len(x)} &lt; 10, スキップ")
        return None

    slope, intercept, r, p, se = linregress(x, y)
    t_stat = (slope - 0.5) / se
    p_05 = 2 * tdist.sf(abs(t_stat), df=len(x)-2)

    print(f"  {label}:")
    print(f"    N = {len(x)}")
    print(f"    alpha = {slope:.3f} +/- {se:.3f}")
    print(f"    p(alpha=0.5) = {p_05:.4f}  {'棄却不可' if p_05 &gt; 0.05 else '棄却'}")
    print(f"    r = {r:.3f}")

    return {"alpha": slope, "e_alpha": se, "p05": p_05, "N": len(x), "r": r}
def main():
```

```
print("=" * 60)
print("(A) PROBES Υ_d フリーフィット検証")
print("=" * 60)

rc_data, dist_map, s4g, sparc_names = fetch_and_build_vbar()

# 名前正規化
import re
def norm(n):
    return re.sub(r"[¥s¥-_]", "", n.upper())

s4g_norm = {norm(k): k for k in s4g}
dist_norm = {norm(k): k for k in dist_map}
sparc_norm = {norm(n) for n in sparc_names}

# === メインループ: 3つのΥ_d戦略 ===
results = {"fixed_03": [], "fixed_05": [], "fixed_07": [], "free": []}
Yd_opt_list = []

n_processed = 0
n_matched = 0

for rc_name, rc in rc_data.items():
    nn = norm(rc_name)

    # S4Gマッチ
    if nn not in s4g_norm:
        continue
    s_name = s4g_norm[nn]
    s = s4g[s_name]

    # 距離マッチ
    if nn not in dist_norm:
        continue
    dist = dist_map[dist_norm[nn]]

    n_matched += 1

    # SPARC除外
    is_sparc = nn in sparc_norm

    R = np.array(rc["R"])
    V = np.array(rc["V"])
    eV = np.array(rc["eV"])

    if len(R) &lt; 5 or dist &lt;= 0:
        continue

    Re = s["Re"]
    Tmag = s["Tmag"]
    n_ser = s["n"]

    # --- 固定Υ_d ---
    for Yd_val, key in [(0.3, "fixed_03"), (0.5, "fixed_05"), (0.7, "fixed_07")]:
        res = compute_vbar_components(R, dist, Re, Tmag, n_ser, Yd_val)
        if res is None or len(res["V_bar"]) != len(V):
            continue
        gc, e_gc = measure_gc_from_vbar(res["R_kpc"], V, res["V_bar"])
        if gc is not None and gc &gt; 0 and res["Sigma0"] &gt; 0:
            results[key].append({
                "name": rc_name, "gc": gc, "Sigma0": res["Sigma0"],
                "is_sparc": is_sparc,
            })

    # --- フリーΥ_d ---
    Yd_best, chi2_dof = optimize_Yd(R, V, eV, dist, Re, Tmag, n_ser)
    if Yd_best is not None and chi2_dof is not None and chi2_dof &lt; 20:
        res = compute_vbar_components(R, dist, Re, Tmag, n_ser, Yd_best)
        if res is not None and len(res["V_bar"]) == len(V):
            gc, e_gc = measure_gc_from_vbar(res["R_kpc"], V, res["V_bar"])
            if gc is not None and gc &gt; 0 and res["Sigma0"] &gt; 0:
                results["free"].append({
                    "name": rc_name, "gc": gc, "Sigma0": res["Sigma0"],
                    "Yd": Yd_best, "chi2_dof": chi2_dof,
                    "is_sparc": is_sparc,
                })
                Yd_opt_list.append(Yd_best)

    n_processed += 1

print(f"¥nマッチ: {n_matched}, 処理完了: {n_processed}")

# === α検定 ===
print("¥n" + "=" * 60)
```

```
    print("α検定結果")
    print("=" * 60)

    for key, label in [
        ("fixed_03", "Y_d=0.3 固定"),
        ("fixed_05", "Y_d=0.5 固定"),
        ("fixed_07", "Y_d=0.7 固定"),
        ("free", "Y_d フリーフィット"),
    ]:
        data = results[key]
        if not data:
            print(f"  {label}: データなし")
            continue

        # 全体
        gc_arr = np.array([d["gc"] for d in data])
        S0_arr = np.array([d["Sigma0"] for d in data])
        run_alpha_test(gc_arr, S0_arr, f"{label} (全体)")

        # SPARC除外
        gc_indep = np.array([d["gc"] for d in data if not d["is_sparc"]])
        S0_indep = np.array([d["Sigma0"] for d in data if not d["is_sparc"]])
        if len(gc_indep) &gt;= 10:
            run_alpha_test(gc_indep, S0_indep, f"{label} (SPARC除外)")

    # === Y_d分布 ===
    if Yd_opt_list:
        Yd_arr = np.array(Yd_opt_list)
        print(f"¥n{'='*60}")
        print(f"Y_d最適値の分布 (N={len(Yd_arr)})")
        print(f"{'='*60}")
        print(f"  中央値: {np.median(Yd_arr):.3f}")
        print(f"  平均: {np.mean(Yd_arr):.3f}")
        print(f"  標準偏差: {np.std(Yd_arr):.3f}")
        print(f"  範囲: [{np.min(Yd_arr):.3f}, {np.max(Yd_arr):.3f}]")
        print(f"  IQR: [{np.percentile(Yd_arr,25):.3f}, {np.percentile(Yd_arr,75):.3f}]")
        print(f"  SPARC準拠範囲(0.3-0.9)内: {np.sum((Yd_arr&gt;=0.3)&amp;(Yd_arr&lt;=0.9))}/{len(Yd_arr)}")

    # === 結果保存 ===
    summary = {
        "n_matched": n_matched,
        "n_processed": n_processed,
        "Yd_opt_stats": {
            "median": float(np.median(Yd_arr)) if Yd_opt_list else None,
            "mean": float(np.mean(Yd_arr)) if Yd_opt_list else None,
            "std": float(np.std(Yd_arr)) if Yd_opt_list else None,
        },
        "results_count": {k: len(v) for k, v in results.items()},
    }
    with open(OUTDIR / "Yd_free_summary.json", "w") as f:
        json.dump(summary, f, indent=2)

    print(f"¥n結果保存: {OUTDIR / 'Yd_free_summary.json'}")


if __name__ == "__main__":
    main()
```

sparc_Yd_sensitivity.py

章: 第25章v1

目的: SPARC Yd感度テスト (Sigma0にYd混入版)

解析対象データ: SPARC Rotmod_LTG

主要結果: 全Ydでalpha<0.5。Sigma0定義にYdが混入したためアーティファクト。参考値のみ。v2で修正。

ソースコード (401 lines, 13,378 bytes)

```
#!/usr/bin/env python3
"""
sparc_Yd_sensitivity.py
=======================
SPARC 175銀河でΥ_dを0.1-1.5の範囲で系統的に変化させ、
alphaの応答を測定する感度テスト。

核心的問い:
  PROBESではΥ_d増加 -&gt; alpha減少の単調トレンドが確認された。
  SPARCでも同じトレンドが存在するか？
  alpha=0.5はΥ_d=0.5の「帰結」か、それとも「物理」か？

テスト内容:
  (1) Υ_d = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2, 1.5
      でalphaを測定 (Υ_bul=0.7固定)
  (2) Υ_bulも同時に振る2Dグリッド (Υ_d x Υ_b)
  (3) dalpha/dYd の数値微分 -&gt; Υ_d=0.5近傍での感度
  (4) alpha=0.5となるΥ_dの逆算 -&gt; 天文学的標準値と一致するか
  (5) MOND比較: g_c=a0 (alpha=1) が成立するΥ_dの特定

実行: uv run --with scipy --with matplotlib python sparc_Yd_sensitivity.py

前提: SPARCデータが Rotmod_LTG/ ディレクトリに存在すること。
"""

import numpy as np
import sys
import json
from pathlib import Path
from scipy.stats import linregress, t as tdist
from scipy.optimize import brentq
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib import font_manager as _fm
for _fp in ['/usr/share/fonts/opentype/ipafont-gothic/ipag.ttf',
            '/usr/share/fonts/opentype/ipafont-gothic/ipagp.ttf']:
    try: _fm.fontManager.addfont(_fp)
    except: pass
plt.rcParams['font.family'] = 'IPAGothic'
plt.rcParams['axes.unicode_minus'] = False

# === 設定 ===
SPARC_DIR = Path(r"D:¥ドキュメント¥エントロピー¥新膜宇宙論¥これまでの軌跡¥パイソン¥Rotmod_LTG")
if not SPARC_DIR.exists():
    SPARC_DIR = Path("Rotmod_LTG")
if not SPARC_DIR.exists():
    SPARC_DIR = Path(".")

OUTDIR = Path("sparc_Yd_sensitivity_output")
OUTDIR.mkdir(exist_ok=True)

# 物理定数
G_SI = 6.674e-11
Msun = 1.989e30
pc = 3.086e16
kpc = 1e3 * pc
a0 = 1.2e-10  # m/s^2

# Υ_dグリッド
YD_GRID = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2, 1.5]
YB_DEFAULT = 0.7


def load_sparc_galaxy(filepath):
    """SPARC Rotmod_LTGファイル読み込み"""
    try:
        data = np.loadtxt(filepath, comments='#')
    except:
        return None
```

```
    if data.ndim != 2 or data.shape[1] &lt; 5:
        return None
    return {
        'R_kpc': data[:, 0],
        'V_obs': data[:, 1],
        'e_V':   data[:, 2],
        'V_gas': data[:, 3],
        'V_disk': data[:, 4],
        'V_bul': data[:, 5] if data.shape[1] &gt; 5 else np.zeros(len(data)),
        'name':  filepath.stem,
    }


def compute_gc_sigma0(gal, Yd, Yb):
    """
    g_cとSigma_0を測定。
    V_bar^2 = V_gas^2 + Yd*V_disk^2 + Yb*V_bul^2
    """
    R = gal['R_kpc']
    Vo = gal['V_obs']
    Vg = gal['V_gas']
    Vd = gal['V_disk']
    Vb = gal['V_bul']

    n = len(R)
    if n &lt; 5:
        return None

    V_bar2 = Vg**2 + Yd * Vd**2 + Yb * Vb**2
    V_bar = np.sqrt(np.maximum(V_bar2, 0))

    # 外側1/3でg_c測定
    outer = slice(2*n//3, n)
    R_m = R[outer] * kpc
    if len(R_m) &lt; 2 or np.any(R_m &lt;= 0):
        return None

    g_obs = (Vo[outer] * 1e3)**2 / R_m
    g_bar = (V_bar[outer] * 1e3)**2 / R_m

    gc_vals = g_obs - g_bar
    gc = np.median(gc_vals)

    if gc &lt;= 0:
        return None

    # Sigma_0推定: V_disk peakからスケール長を推定
    idx_peak = np.argmax(np.abs(Vd))
    if idx_peak == 0:
        idx_peak = 1
    R_peak = R[idx_peak]
    h_kpc = R_peak / 2.2
    V_d_peak = abs(Vd[idx_peak])

    if h_kpc &lt;= 0 or V_d_peak &lt;= 0:
        return None

    h_m = h_kpc * kpc
    # Sigma_0 = Yd * V_d_peak^2 / (2*pi*G*h)  [SI] -&gt; [M_sun/pc^2]
    Sigma0_SI = Yd * (V_d_peak * 1e3)**2 / (2 * np.pi * G_SI * h_m)
    Sigma0 = Sigma0_SI * pc**2 / Msun

    if Sigma0 &lt;= 0:
        return None

    return {'gc': gc, 'Sigma0': Sigma0}


def alpha_fit(gc_arr, S0_arr):
    """線形回帰 log(g_c) = log(eta) + alpha * log(a0*G*Sigma_0)"""
    x = np.log10(a0 * G_SI * S0_arr * Msun / pc**2)
    y = np.log10(gc_arr)

    mask = np.isfinite(x) &amp;amp; np.isfinite(y)
    x, y = x[mask], y[mask]

    if len(x) &lt; 10:
        return None

    slope, intercept, r, p, se = linregress(x, y)
    t_stat = (slope - 0.5) / se
    p_05 = 2 * tdist.sf(abs(t_stat), df=len(x) - 2)

    return {
```

```
        'alpha': slope, 'e_alpha': se, 'p05': p_05,
        'r': r, 'N': len(x), 'intercept': intercept,
    }


def main():
    print('=' * 60)
    print('SPARC Y_d 感度テスト')
    print('=' * 60)

    # 全銀河読み込み
    files = sorted(SPARC_DIR.glob('*.dat'))
    if not files:
        files = sorted(SPARC_DIR.glob('*_rotmod.dat'))
    if not files:
        print(f'ERROR: .datファイルが見つかりません: {SPARC_DIR}')
        sys.exit(1)

    galaxies = []
    for f in files:
        gal = load_sparc_galaxy(f)
        if gal is not None:
            galaxies.append(gal)
    print(f'{len(galaxies)} 銀河読み込み')

    # ============================================================
    # テスト1: Y_d 1Dスキャン (Y_b=0.7固定)
    # ============================================================
    print(f'¥n--- テスト1: Y_d 1Dスキャン (Y_b={YB_DEFAULT}) ---')
    print(f'{"Y_d":&gt;6} {"N":&gt;5} {"alpha":&gt;8} {"SE":&gt;8} {"p(0.5)":&gt;10} {"r":&gt;7}')
    print('-' * 50)

    results_1d = []
    for Yd in YD_GRID:
        gc_list = []
        S0_list = []
        for gal in galaxies:
            res = compute_gc_sigma0(gal, Yd, YB_DEFAULT)
            if res:
                gc_list.append(res['gc'])
                S0_list.append(res['Sigma0'])

        gc_arr = np.array(gc_list)
        S0_arr = np.array(S0_list)
        fit = alpha_fit(gc_arr, S0_arr)

        if fit:
            print(f'{Yd:6.2f} {fit["N"]:5d} {fit["alpha"]:8.3f} {fit["e_alpha"]:8.3f} '
                  f'{fit["p05"]:10.4f} {fit["r"]:7.3f}')
            results_1d.append({'Yd': Yd, **fit})
        else:
            print(f'{Yd:6.2f}   -- fit failed --')

    # ============================================================
    # テスト2: Y_d x Y_b 2Dグリッド
    # ============================================================
    print(f'¥n--- テスト2: Y_d x Y_b 2Dグリッド ---')
    YD_2D = [0.3, 0.4, 0.5, 0.6, 0.7, 0.8]
    YB_2D = [0.3, 0.5, 0.7, 0.9, 1.1]

    print(f'{"":&gt;6}', end='')
    for Yb in YB_2D:
        print(f' Yb={Yb:.1f}', end='')
    print()
    print('-' * (6 + 9 * len(YB_2D)))

    results_2d = {}
    for Yd in YD_2D:
        print(f'Yd={Yd:.1f}', end='')
        for Yb in YB_2D:
            gc_list, S0_list = [], []
            for gal in galaxies:
                res = compute_gc_sigma0(gal, Yd, Yb)
                if res:
                    gc_list.append(res['gc'])
                    S0_list.append(res['Sigma0'])
            fit = alpha_fit(np.array(gc_list), np.array(S0_list))
            if fit:
                print(f'  {fit["alpha"]:6.3f}', end='')
                results_2d[(Yd, Yb)] = fit
            else:
                print(f'    --  ', end='')
        print()
    # ============================================================
```

```
# テスト3: dalpha/dYd (数値微分)
# ============================================================
print(f'¥n--- テスト3: dalpha/dYd (Y_d=0.5近傍) ---')
if len(results_1d) &gt;= 3:
    Yd_arr = np.array([r['Yd'] for r in results_1d])
    alpha_arr = np.array([r['alpha'] for r in results_1d])

    # Ｙ_d=0.5の前後で数値微分
    idx_05 = np.argmin(np.abs(Yd_arr - 0.5))
    if 0 &lt; idx_05 &lt; len(Yd_arr) - 1:
        dYd = Yd_arr[idx_05 + 1] - Yd_arr[idx_05 - 1]
        dalpha = alpha_arr[idx_05 + 1] - alpha_arr[idx_05 - 1]
        sensitivity = dalpha / dYd
        print(f'  dalpha/dYd at Y_d=0.5: {sensitivity:.3f}')
        print(f'  -&gt; Y_dを0.1変えるとalphaが{abs(sensitivity)*0.1:.3f}変化')
        print(f'  -&gt; Y_dの10%不確かさ(+/-0.05)でalphaは+/-{abs(sensitivity)*0.05:.3f}')

    # 全範囲の勾配
    from scipy.stats import linregress as lr
    sl, ic, _, _, _ = lr(Yd_arr, alpha_arr)
    print(f'  全範囲線形近似: alpha = {sl:.3f} x Y_d + {ic:.3f}')

# ============================================================
# テスト4: alpha=0.5を与えるＹ_dの逆算
# ============================================================
print(f'¥n--- テスト4: alpha=0.5を与えるY_dの逆算 ---')
if len(results_1d) &gt;= 3:
    Yd_arr = np.array([r['Yd'] for r in results_1d])
    alpha_arr = np.array([r['alpha'] for r in results_1d])

    # 補間でalpha=0.5となるＹ_dを探す
    from scipy.interpolate import interp1d
    try:
        f_interp = interp1d(Yd_arr, alpha_arr - 0.5, kind='linear')
        # alpha-0.5がゼロになる点を探す
        for i in range(len(Yd_arr) - 1):
            if (alpha_arr[i] - 0.5) * (alpha_arr[i+1] - 0.5) &lt; 0:
                Yd_cross = brentq(f_interp, Yd_arr[i], Yd_arr[i+1])
                print(f'  alpha=0.5 at Y_d = {Yd_cross:.3f}')
                if 0.3 &lt;= Yd_cross &lt;= 0.9:
                    print(f'  -&gt; 天文学的標準範囲 (0.3-0.9) 内: YES')
                else:
                    print(f'  -&gt; 天文学的標準範囲 (0.3-0.9) 内: NO')
                break
        else:
            # 交差しない場合
            if np.all(alpha_arr &gt; 0.5):
                print(f'  alpha &gt; 0.5 for all Y_d in [{Yd_arr[0]:.1f}, {Yd_arr[-1]:.1f}]')
                print(f'  alpha=0.5には Y_d &gt; {Yd_arr[-1]:.1f} が必要')
            elif np.all(alpha_arr &lt; 0.5):
                print(f'  alpha &lt; 0.5 for all Y_d in [{Yd_arr[0]:.1f}, {Yd_arr[-1]:.1f}]')
    except Exception as e:
        print(f'  補間エラー: {e}')

# ============================================================
# テスト5: MOND比較 (alpha=1を与えるＹ_d)
# ============================================================
print(f'¥n--- テスト5: alpha=1.0 (MOND相当) を与えるY_d ---')
if len(results_1d) &gt;= 3:
    try:
        f_interp1 = interp1d(Yd_arr, alpha_arr - 1.0, kind='linear')
        for i in range(len(Yd_arr) - 1):
            if (alpha_arr[i] - 1.0) * (alpha_arr[i+1] - 1.0) &lt; 0:
                Yd_mond = brentq(f_interp1, Yd_arr[i], Yd_arr[i+1])
                print(f'  alpha=1.0 at Y_d = {Yd_mond:.3f}')
                break
        else:
            if np.all(alpha_arr &lt; 1.0):
                print(f'  alpha &lt; 1.0 for all Y_d -&gt; MOND (alpha=1) は全Y_dで棄却')
            elif np.all(alpha_arr &gt; 1.0):
                print(f'  alpha &gt; 1.0 for all Y_d -&gt; 予想外の結果')
    except Exception as e:
        print(f'  補間エラー: {e}')

# ============================================================
# 図の生成
# ============================================================
fig, axes = plt.subplots(2, 2, figsize=(14, 11))

# (a) alpha vs Y_d (1D)
ax = axes[0, 0]
Yd_plot = [r['Yd'] for r in results_1d]
alpha_plot = [r['alpha'] for r in results_1d]
e_alpha_plot = [r['e_alpha'] for r in results_1d]
```

```
    ax.errorbar(Yd_plot, alpha_plot, yerr=e_alpha_plot, fmt='o-', color='#1a1a2e',
                capsize=3, markersize=6)
    ax.axhline(0.5, color='#e94560', ls='--', lw=2, label='alpha=0.5')
    ax.axhline(1.0, color='blue', ls=':', lw=1, label='alpha=1.0 (MOND)')
    ax.axvline(0.5, color='grey', ls=':', alpha=0.5, label='Y_d=0.5 (standard)')
    ax.axvspan(0.3, 0.9, alpha=0.1, color='green', label='standard range')
    ax.set_xlabel('Y_disk [M_sun/L_sun]')
    ax.set_ylabel('alpha')
    ax.set_title('(a) alpha vs Y_d (SPARC, Y_b=0.7)')
    ax.legend(fontsize=8)
    ax.grid(True, alpha=0.3)

    # (b) p(alpha=0.5) vs Y_d
    ax = axes[0, 1]
    p05_plot = [r['p05'] for r in results_1d]
    ax.semilogy(Yd_plot, p05_plot, 'o-', color='#1a1a2e', markersize=6)
    ax.axhline(0.05, color='#e94560', ls='--', label='p=0.05')
    ax.axvline(0.5, color='grey', ls=':', alpha=0.5)
    ax.set_xlabel('Y_disk [M_sun/L_sun]')
    ax.set_ylabel('p(alpha=0.5)')
    ax.set_title('(b) p(alpha=0.5) vs Y_d')
    ax.legend(fontsize=8)
    ax.grid(True, alpha=0.3)
    # p&gt;0.05の領域をマーク
    for i, (yd, pv) in enumerate(zip(Yd_plot, p05_plot)):
        if pv &gt; 0.05:
            ax.plot(yd, pv, 'o', color='#d4edda', markersize=12, zorder=0)

    # (c) N vs Y_d
    ax = axes[1, 0]
    N_plot = [r['N'] for r in results_1d]
    ax.bar(Yd_plot, N_plot, width=0.07, color='#1a1a2e', alpha=0.7)
    ax.set_xlabel('Y_disk [M_sun/L_sun]')
    ax.set_ylabel('N (valid galaxies)')
    ax.set_title('(c) Number of valid galaxies vs Y_d')
    ax.grid(True, alpha=0.3, axis='y')

    # (d) 2Dヒートマップ (Y_d x Y_b)
    ax = axes[1, 1]
    alpha_2d = np.full((len(YD_2D), len(YB_2D)), np.nan)
    for i, Yd in enumerate(YD_2D):
        for j, Yb in enumerate(YB_2D):
            if (Yd, Yb) in results_2d:
                alpha_2d[i, j] = results_2d[(Yd, Yb)]['alpha']

    im = ax.imshow(alpha_2d, aspect='auto', origin='lower',
                   extent=[YB_2D[0]-0.1, YB_2D[-1]+0.1, YD_2D[0]-0.05, YD_2D[-1]+0.05],
                   cmap='RdYlGn_r', vmin=0.3, vmax=0.8)
    ax.set_xlabel('Y_bul [M_sun/L_sun]')
    ax.set_ylabel('Y_disk [M_sun/L_sun]')
    ax.set_title('(d) alpha(Y_d, Y_b) 2D map')
    plt.colorbar(im, ax=ax, label='alpha')
    # alpha=0.5等高線
    try:
        ax.contour(np.linspace(YB_2D[0], YB_2D[-1], len(YB_2D)),
                   np.linspace(YD_2D[0], YD_2D[-1], len(YD_2D)),
                   alpha_2d, levels=[0.5], colors='#e94560', linewidths=2)
    except:
        pass

    plt.tight_layout()
    fig_path = OUTDIR / 'sparc_Yd_sensitivity.png'
    plt.savefig(fig_path, dpi=150)
    print(f'¥n figure: {fig_path}')

    # === 結果JSON保存 ===
    summary = {
        'test1_1d': results_1d,
        'test2_2d': {f'Yd={k[0]}_Yb={k[1]}': v for k, v in results_2d.items()},
    }
    with open(OUTDIR / 'Yd_sensitivity_summary.json', 'w') as f:
        json.dump(summary, f, indent=2, default=float)
    print(f'results: {OUTDIR / "Yd_sensitivity_summary.json"}')


if __name__ == '__main__':
    main()
```

sparc_Yd_sensitivity_v2.py

章: 第25章

目的: SPARC Yd感度テスト v2 (G*Sigma0=Vflat^2/hR, Yd非依存)

解析対象データ: SPARC Rotmod_LTG

主要結果: alpha=0.5が棄却されないYd範囲: 0.65-0.80。alpha=0.5を与えるYd=0.74(SPS標準範囲内)。Yd=0.5: alpha=0.670(棄却)。

ソースコード (477 lines, 15,580 bytes)

```
#!/usr/bin/env python3
"""
sparc_Yd_sensitivity_v2.py
==========================
修正版: 元パイプラインと同一の回帰変数定義を使用。

核心的違い:
  旧スクリプト (v1): Sigma_0 = Yd * V_peak^2 / (2*pi*G*h) -&gt; Yd依存
  本スクリプト (v2): G*Sigma_0 = V_flat^2 / h_R           -&gt; Yd非依存 (観測量のみ)

これにより、alpha vs Yd の真の依存性を測定できる。
g_c = g_obs - g_bar のみがΥ_dの影響を受ける。

テスト:
  (1) Υ_d = 0.1-1.5 でalpha測定 (G*Sigma_0はΥ_d固定)
  (2) dalpha/dYd の数値微分
  (3) alpha=0.5を与えるΥ_dの逆算
  (4) g_cが負になる (V_bar &gt; V_obs) Υ_dの閾値
  (5) Bootstrap信頼区間つきのΥ_dスキャン

実行: uv run --with scipy --with matplotlib python sparc_Yd_sensitivity_v2.py
"""

import numpy as np
import sys
import json
from pathlib import Path
from scipy.stats import linregress, t as tdist
from scipy.optimize import brentq
from scipy.interpolate import interp1d
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib import font_manager as _fm
for _fp in ['/usr/share/fonts/opentype/ipafont-gothic/ipag.ttf',
            '/usr/share/fonts/opentype/ipafont-gothic/ipagp.ttf']:
    try: _fm.fontManager.addfont(_fp)
    except: pass
plt.rcParams['font.family'] = 'IPAGothic'
plt.rcParams['axes.unicode_minus'] = False

# === 設定 ===
SPARC_DIR = Path(r"D:¥ドキュメント¥エントロピー¥新膜宇宙論¥これまでの軌跡¥パイソン¥Rotmod_LTG")
if not SPARC_DIR.exists():
    SPARC_DIR = Path("Rotmod_LTG")
if not SPARC_DIR.exists():
    SPARC_DIR = Path(".")

OUTDIR = Path("sparc_Yd_sensitivity_output")
OUTDIR.mkdir(exist_ok=True)

# 物理定数
G_SI = 6.674e-11
Msun = 1.989e30
pc = 3.086e16
kpc = 1e3 * pc
a0 = 1.2e-10

# Υ_dグリッド (細かく)
YD_GRID = [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5,
           0.55, 0.6, 0.65, 0.7, 0.8, 0.9, 1.0, 1.2, 1.5]
YB_DEFAULT = 0.7
N_BOOT = 500
RNG_SEED = 42


def load_sparc_galaxy(filepath):
    try:
        data = np.loadtxt(filepath, comments='#')
    except:
```

```
        return None
    if data.ndim != 2 or data.shape[1] &lt; 5:
        return None
    return {
        'R_kpc': data[:, 0],
        'V_obs': data[:, 1],
        'e_V':   data[:, 2],
        'V_gas': data[:, 3],
        'V_disk': data[:, 4],
        'V_bul': data[:, 5] if data.shape[1] &gt; 5 else np.zeros(len(data)),
        'name':  filepath.stem,
    }


def measure_observables(gal):
    """
    Υ_d非依存の観測量を測定:
    - V_flat: 外側1/3のV_obsの中央値 [km/s]
    - h_R: V_diskピーク位置 / 2.2 [kpc] (指数ディスクのピーク)
    - G*Sigma_0 = V_flat^2 / h_R [m/s^2 * m = m^2/s^2 ... ]
      正確には: G*Sigma_0 ~ V_flat^2 / (h_R [m]) [m/s^2]
    """
    R = gal['R_kpc']
    Vo = gal['V_obs']
    Vd = gal['V_disk']
    n = len(R)
    if n &lt; 5:
        return None

    # V_flat (外側1/3中央値)
    outer = slice(2*n//3, n)
    V_flat = np.median(Vo[outer])  # km/s
    if V_flat &lt;= 0:
        return None

    # h_R from V_disk peak
    idx_peak = np.argmax(np.abs(Vd))
    if idx_peak == 0:
        idx_peak = min(1, n-1)
    R_peak = R[idx_peak]
    h_R_kpc = R_peak / 2.2
    if h_R_kpc &lt;= 0.01:
        return None

    h_R_m = h_R_kpc * kpc

    # G*Sigma_0 = V_flat^2 / h_R [SI: m/s^2]
    # これは加速度の次元を持ち、a_0と直接比較可能
    G_Sigma0 = (V_flat * 1e3)**2 / h_R_m  # m/s^2

    return {
        'V_flat': V_flat,
        'h_R_kpc': h_R_kpc,
        'G_Sigma0': G_Sigma0,
    }


def measure_gc(gal, Yd, Yb):
    """
    g_c = g_obs - g_bar (外側1/3の中央値)
    g_barのみΥ_dに依存する。
    """
    R = gal['R_kpc']
    Vo = gal['V_obs']
    Vg = gal['V_gas']
    Vd = gal['V_disk']
    Vb = gal['V_bul']
    n = len(R)
    if n &lt; 5:
        return None

    V_bar2 = Vg**2 + Yd * Vd**2 + Yb * Vb**2
    V_bar = np.sqrt(np.maximum(V_bar2, 0))

    outer = slice(2*n//3, n)
    R_m = R[outer] * kpc
    if len(R_m) &lt; 2 or np.any(R_m &lt;= 0):
        return None

    g_obs = (Vo[outer] * 1e3)**2 / R_m
    g_bar = (V_bar[outer] * 1e3)**2 / R_m

    gc_vals = g_obs - g_bar
    gc = np.median(gc_vals)
```

```
    # g_c &lt; 0 の場合 (V_bar &gt; V_obs: oversubtraction)
    n_negative = np.sum(gc_vals &lt; 0)

    return {
        'gc': gc,
        'n_negative': n_negative,
        'n_points': len(gc_vals),
    }


def alpha_fit(gc_arr, GS0_arr):
    """
    log(g_c) = log(eta) + alpha * log(a_0 * G*Sigma_0)
    ここでG*Sigma_0は既にG込みの値なので、a_0 * G*Sigma_0 を横軸にする。

    注意: 元パイプラインの正確な定義を確認する必要あり。
    ここでは x = log10(G*Sigma_0) として回帰する。
    a_0との積は定数オフセットなのでalphaには影響しない。
    """
    x = np.log10(GS0_arr)
    y = np.log10(gc_arr)

    mask = np.isfinite(x) &amp;amp; np.isfinite(y)
    x, y = x[mask], y[mask]

    if len(x) &lt; 10:
        return None

    slope, intercept, r, p, se = linregress(x, y)
    t_stat = (slope - 0.5) / se
    p_05 = 2 * tdist.sf(abs(t_stat), df=len(x) - 2)

    return {
        'alpha': slope, 'e_alpha': se, 'p05': p_05,
        'r': r, 'N': len(x),
    }


def main():
    print('=' * 65)
    print('SPARC Y_d 感度テスト v2')
    print('G*Sigma_0 = V_flat^2/h_R (Y_d非依存)')
    print('=' * 65)

    # 全銀河読み込み
    files = sorted(SPARC_DIR.glob('*.dat'))
    if not files:
        files = sorted(SPARC_DIR.glob('*_rotmod.dat'))
    if not files:
        print(f'ERROR: {SPARC_DIR}')
        sys.exit(1)

    galaxies = []
    for f in files:
        gal = load_sparc_galaxy(f)
        if gal is not None:
            galaxies.append(gal)
    print(f'{len(galaxies)} galaxies loaded')

    # === 事前計算: Y_d非依存の観測量 ===
    obs_data = []
    valid_galaxies = []
    for gal in galaxies:
        obs = measure_observables(gal)
        if obs is not None:
            obs_data.append(obs)
            valid_galaxies.append(gal)
    print(f'{len(valid_galaxies)} galaxies with valid observables')

    GS0_all = np.array([o['G_Sigma0'] for o in obs_data])

    # ================================================================
    # テスト1: Y_d 1Dスキャン
    # ================================================================
    print(f'¥n{"="*65}')
    print('Test 1: Y_d scan (G*Sigma_0 = V_flat^2/h_R, Y_d-independent)')
    print(f'{"="*65}')
    print(f'{"Y_d":&gt;6} {"N":&gt;5} {"N_gc&gt;0":&gt;7} {"alpha":&gt;8} {"SE":&gt;8} '
          f'{"p(0.5)":&gt;10} {"r":&gt;7} {"reject":&gt;7}')
    print('-' * 65)

    results_1d = []
    for Yd in YD_GRID:
        gc_list = []
```

```
    gs0_list = []
    n_gc_pos = 0
    n_gc_neg = 0

    for gal, obs in zip(valid_galaxies, obs_data):
        res = measure_gc(gal, Yd, YB_DEFAULT)
        if res is None:
            continue
        if res['gc'] &gt; 0:
            gc_list.append(res['gc'])
            gs0_list.append(obs['G_Sigma0'])
            n_gc_pos += 1
        else:
            n_gc_neg += 1

    gc_arr = np.array(gc_list)
    gs0_arr = np.array(gs0_list)
    fit = alpha_fit(gc_arr, gs0_arr)

    entry = {
        'Yd': Yd,
        'N_total': n_gc_pos + n_gc_neg,
        'N_gc_pos': n_gc_pos,
        'N_gc_neg': n_gc_neg,
    }

    if fit:
        entry.update(fit)
        reject = 'YES' if fit['p05'] &lt; 0.05 else 'no'
        print(f'{Yd:6.2f} {fit["N"]:5d} {n_gc_pos:7d} {fit["alpha"]:8.3f} '
              f'{fit["e_alpha"]:8.3f} {fit["p05"]:10.4f} {fit["r"]:7.3f} {reject:&gt;7}')
    else:
        print(f'{Yd:6.2f}   -- fit failed (N_pos={n_gc_pos}, N_neg={n_gc_neg}) --')

    results_1d.append(entry)

# ============================================================
# テスト2: dalpha/dYd
# ============================================================
print(f'¥n{"="*65}')
print('Test 2: dalpha/dYd')
print(f'{"="*65}')

valid_1d = [r for r in results_1d if 'alpha' in r]
if len(valid_1d) &gt;= 3:
    Yd_arr = np.array([r['Yd'] for r in valid_1d])
    alpha_arr = np.array([r['alpha'] for r in valid_1d])

    # 中心差分
    for target_Yd in [0.3, 0.5, 0.7]:
        idx = np.argmin(np.abs(Yd_arr - target_Yd))
        if 0 &lt; idx &lt; len(Yd_arr) - 1:
            dYd = Yd_arr[idx+1] - Yd_arr[idx-1]
            dalpha = alpha_arr[idx+1] - alpha_arr[idx-1]
            sens = dalpha / dYd
            print(f'  dalpha/dYd at Y_d={target_Yd:.1f}: {sens:.3f}')
            print(f'    -&gt; Y_d +/- 0.05 (10%) -&gt; alpha +/- {abs(sens)*0.05:.4f}')

    # 全範囲線形フィット
    sl, ic, _, _, _ = linregress(Yd_arr, alpha_arr)
    print(f'¥n  Linear fit: alpha = {sl:.4f} * Y_d + {ic:.4f}')
    print(f'  Overall slope: {sl:.4f} per unit Y_d')

# ============================================================
# テスト3: alpha=0.5を与えるY_d
# ============================================================
print(f'¥n{"="*65}')
print('Test 3: Y_d that gives alpha=0.5')
print(f'{"="*65}')

if len(valid_1d) &gt;= 3:
    try:
        f_interp = interp1d(Yd_arr, alpha_arr - 0.5, kind='linear',
                            fill_value='extrapolate')
        # 交差点を探す
        found_crossing = False
        for i in range(len(Yd_arr) - 1):
            v0 = alpha_arr[i] - 0.5
            v1 = alpha_arr[i+1] - 0.5
            if v0 * v1 &lt; 0:
                Yd_cross = brentq(f_interp, Yd_arr[i], Yd_arr[i+1])
                print(f'  alpha=0.5 at Y_d = {Yd_cross:.4f}')
                in_range = 0.3 &lt;= Yd_cross &lt;= 0.9
                print(f'  Standard range (0.3-0.9): {"YES" if in_range else "NO"}')
```

```
                found_crossing = True
                break
        if not found_crossing:
            if np.all(alpha_arr &gt; 0.5):
                print(f'  alpha &gt; 0.5 for all Y_d in [{Yd_arr[0]:.2f}, {Yd_arr[-1]:.2f}]')
                print(f'  -&gt; alpha=0.5 requires Y_d &gt; {Yd_arr[-1]:.1f} (extrapolation)')
                # 外挿
                Yd_extrap = (0.5 - ic) / sl
                print(f'  -&gt; Linear extrapolation: Y_d ~= {Yd_extrap:.2f}')
            elif np.all(alpha_arr &lt; 0.5):
                print(f'  alpha &lt; 0.5 for all Y_d')
                print(f'  -&gt; alpha=0.5 is never reached (alpha always below)')
    except Exception as e:
        print(f'  Error: {e}')

# ============================================================
# テスト4: g_c &lt; 0 の割合 vs Y_d
# ============================================================
print(f'¥n{"="*65}')
print('Test 4: Oversubtraction (g_c &lt; 0) fraction vs Y_d')
print(f'{"="*65}')
print(f'{"Y_d":&gt;6} {"N_pos":&gt;6} {"N_neg":&gt;6} {"frac_neg":&gt;9}')
for r in results_1d:
    n_pos = r.get('N_gc_pos', 0)
    n_neg = r.get('N_gc_neg', 0)
    total = n_pos + n_neg
    frac = n_neg / total if total &gt; 0 else 0
    print(f'{r["Yd"]:6.2f} {n_pos:6d} {n_neg:6d} {frac:9.3f}')

# ============================================================
# テスト5: Bootstrap Y_dスキャン (主要3点)
# ============================================================
print(f'¥n{"="*65}')
print(f'Test 5: Bootstrap Y_d scan (N_boot={N_BOOT})')
print(f'{"="*65}')

rng = np.random.RandomState(RNG_SEED)

for Yd_target in [0.3, 0.5, 0.7]:
    # 全銀河のg_c計算
    gc_full = []
    gs0_full = []
    for gal, obs in zip(valid_galaxies, obs_data):
        res = measure_gc(gal, Yd_target, YB_DEFAULT)
        if res and res['gc'] &gt; 0:
            gc_full.append(res['gc'])
            gs0_full.append(obs['G_Sigma0'])
    gc_full = np.array(gc_full)
    gs0_full = np.array(gs0_full)
    N = len(gc_full)

    alpha_boot = []
    for _ in range(N_BOOT):
        idx = rng.choice(N, N, replace=True)
        fit = alpha_fit(gc_full[idx], gs0_full[idx])
        if fit:
            alpha_boot.append(fit['alpha'])
    alpha_boot = np.array(alpha_boot)

    ci = np.percentile(alpha_boot, [2.5, 97.5])
    contains_05 = ci[0] &lt;= 0.5 &lt;= ci[1]
    print(f'  Y_d={Yd_target:.1f}: alpha={np.mean(alpha_boot):.3f} +/- {np.std(alpha_boot):.3f}, '
          f'95%CI=[{ci[0]:.3f}, {ci[1]:.3f}], alpha=0.5 in CI: {"YES" if contains_05 else "NO"}')

# ============================================================
# 図の生成
# ============================================================
fig, axes = plt.subplots(2, 2, figsize=(14, 11))

valid_1d = [r for r in results_1d if 'alpha' in r]
Yd_plot = [r['Yd'] for r in valid_1d]
alpha_plot = [r['alpha'] for r in valid_1d]
e_alpha_plot = [r['e_alpha'] for r in valid_1d]
p05_plot = [r['p05'] for r in valid_1d]

# (a) alpha vs Y_d
ax = axes[0, 0]
ax.errorbar(Yd_plot, alpha_plot, yerr=e_alpha_plot, fmt='o-', color='#1a1a2e',
            capsize=3, markersize=6, lw=2)
ax.axhline(0.5, color='#e94560', ls='--', lw=2, label='alpha=0.5')
ax.axhline(1.0, color='blue', ls=':', lw=1, label='alpha=1.0 (MOND)')
ax.axvline(0.5, color='grey', ls=':', alpha=0.5)
ax.axvspan(0.3, 0.9, alpha=0.08, color='green', label='standard Y_d range')
ax.set_xlabel('Y_disk [M_sun/L_sun]', fontsize=11)
```

```
    ax.set_ylabel('alpha', fontsize=11)
    ax.set_title('(a) alpha vs Y_d¥n(G*Sigma_0 = V_flat^2/h_R: Y_d-independent)', fontsize=10)
    ax.legend(fontsize=8)
    ax.grid(True, alpha=0.3)
    ax.set_ylim(bottom=min(alpha_plot)-0.1, top=max(max(alpha_plot)+0.1, 0.7))

    # (b) p(alpha=0.5) vs Y_d
    ax = axes[0, 1]
    ax.semilogy(Yd_plot, [max(p, 1e-10) for p in p05_plot], 'o-', color='#1a1a2e',
                markersize=6, lw=2)
    ax.axhline(0.05, color='#e94560', ls='--', lw=2, label='p=0.05')
    ax.axvline(0.5, color='grey', ls=':', alpha=0.5)
    ax.set_xlabel('Y_disk [M_sun/L_sun]', fontsize=11)
    ax.set_ylabel('p(alpha=0.5)', fontsize=11)
    ax.set_title('(b) p(alpha=0.5) vs Y_d', fontsize=10)
    ax.legend(fontsize=8)
    ax.grid(True, alpha=0.3)
    # p&gt;0.05をマーク
    for yd, pv in zip(Yd_plot, p05_plot):
        if pv &gt; 0.05:
            ax.plot(yd, max(pv, 1e-10), 'o', color='#d4edda', markersize=14, zorder=0)

    # (c) oversubtraction割合
    ax = axes[1, 0]
    Yd_os = [r['Yd'] for r in results_1d]
    frac_neg = [r.get('N_gc_neg',0)/(r.get('N_gc_pos',0)+r.get('N_gc_neg',1))
                for r in results_1d]
    ax.bar(Yd_os, frac_neg, width=0.04, color='#e94560', alpha=0.7)
    ax.set_xlabel('Y_disk [M_sun/L_sun]', fontsize=11)
    ax.set_ylabel('Oversubtraction fraction (g_c &lt; 0)', fontsize=11)
    ax.set_title('(c) V_bar &gt; V_obs fraction vs Y_d', fontsize=10)
    ax.grid(True, alpha=0.3, axis='y')
    ax.axvline(0.5, color='grey', ls=':', alpha=0.5)

    # (d) dalpha/dYd (数値微分)
    ax = axes[1, 1]
    if len(Yd_plot) &gt;= 3:
        Yd_mid = []
        dalpha_dYd = []
        for i in range(1, len(Yd_plot)-1):
            Yd_mid.append(Yd_plot[i])
            da = (alpha_plot[i+1] - alpha_plot[i-1]) / (Yd_plot[i+1] - Yd_plot[i-1])
            dalpha_dYd.append(da)
        ax.plot(Yd_mid, dalpha_dYd, 'o-', color='#1a1a2e', markersize=6, lw=2)
        ax.axhline(0, color='grey', ls='-', alpha=0.3)
        ax.axvline(0.5, color='grey', ls=':', alpha=0.5)
    ax.set_xlabel('Y_disk [M_sun/L_sun]', fontsize=11)
    ax.set_ylabel('dalpha/dY_d', fontsize=11)
    ax.set_title('(d) Sensitivity dalpha/dY_d', fontsize=10)
    ax.grid(True, alpha=0.3)

    plt.suptitle('SPARC Y_d Sensitivity Test v2¥n'
                 '(G*Sigma_0 = V_flat^2/h_R: Y_d-independent regressor)',
                 fontsize=13, y=1.02)
    plt.tight_layout()

    fig_path = OUTDIR / 'sparc_Yd_sensitivity_v2.png'
    plt.savefig(fig_path, dpi=150, bbox_inches='tight')
    print(f'¥nFigure: {fig_path}')

    # JSON保存
    summary = {'results_1d': valid_1d}
    with open(OUTDIR / 'Yd_sensitivity_v2_summary.json', 'w') as f:
        json.dump(summary, f, indent=2, default=float)
    print(f'Results: {OUTDIR / "Yd_sensitivity_v2_summary.json"}')


if __name__ == '__main__':
    main()
```

sparc_gc_method_comparison.py

章: 第26章前提

目的: g_c定義の6手法比較 + TA3との直接比較

解析対象データ: SPARC Rotmod_LTG, TA3_gc_independent.csv

主要結果: 6手法でalpha=0.59-0.73。median/last_point/flat_regionでCI含0.5。TA3 g_cでhR統一: alpha=0.602。g_c比v2/TA3=0.223。

ソースコード (474 lines, 15,437 bytes)

```
#!/usr/bin/env python3
"""
sparc_gc_method_comparison.py
============================
元パイプライン（TA3_gc_independent.csv）と本スクリプトの
g_c測定手法の差を系統的に分析する。

核心的問い:
  同じSPARCデータ、同じΥ_d=0.5、同じG×Σ☒定義で、
  なぜ alpha が 0.545（TA3）vs 0.670（v2）と 0.125 異なるのか？

分析項目:
  (1) TA3_gc_independent.csv を読み込み、g_c値を銀河ごとに比較
  (2) g_c測定手法の違いを特定:
      - TA3: どのような最適化/フィッティングでg_cを決めているか？
      - v2: median(g_obs - g_bar) in outer 1/3
  (3) g_cの差分分析: どの銀河でどの程度ずれているか
  (4) g_c定義の変種テスト:
      a) 外側1/3の中央値（v2のデフォルト）
      b) 外側1/3の平均値
      c) 外側1/2の中央値
      d) 最外点のみ
      e) フラット領域検出（V_obs変動 &lt; 10%）
      f) V_obsの2乗/R at R_last
  (5) 各定義でのalpha測定 -&gt; どの定義がTA3のalpha=0.545に最も近いか
  (6) Bootstrap 95%CIで手法差の有意性評価

実行: uv run --with scipy --with matplotlib python sparc_gc_method_comparison.py

前提:
  - SPARC Rotmod_LTG/ ディレクトリ
  - TA3_gc_independent.csv（元パイプライン出力）
    存在しない場合はスキップし、g_c定義変種テストのみ実施
"""

import numpy as np
import sys
import json
from pathlib import Path
from scipy.stats import linregress, t as tdist
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib import font_manager as _fm
for _fp in ['/usr/share/fonts/opentype/ipafont-gothic/ipag.ttf',
            '/usr/share/fonts/opentype/ipafont-gothic/ipagp.ttf']:
    try: _fm.fontManager.addfont(_fp)
    except: pass
plt.rcParams['font.family'] = 'IPAGothic'
plt.rcParams['axes.unicode_minus'] = False

# === 設定 ===
SPARC_DIR = Path(r"D:¥ドキュメント¥エントロピー¥新膜宇宙論¥これまでの軌跡¥パイソン¥Rotmod_LTG")
if not SPARC_DIR.exists():
    SPARC_DIR = Path("Rotmod_LTG")
if not SPARC_DIR.exists():
    SPARC_DIR = Path(".")

# TA3ファイルの候補パス
TA3_CANDIDATES = [
    Path(r"D:¥ドキュメント¥エントロピー¥新膜宇宙論¥これまでの軌跡¥パイソン¥TA3_gc_independent.csv"),
    Path("TA3_gc_independent.csv"),
    Path("../TA3_gc_independent.csv"),
]

OUTDIR = Path("gc_method_comparison_output")
OUTDIR.mkdir(exist_ok=True)

# 物理定数
```

```
G_SI = 6.674e-11
Msun = 1.989e30
pc = 3.086e16
kpc = 1e3 * pc
a0 = 1.2e-10

YD = 0.5
YB = 0.7
N_BOOT = 1000
RNG_SEED = 42


def load_sparc(filepath):
    try:
        data = np.loadtxt(filepath, comments='#')
    except:
        return None
    if data.ndim != 2 or data.shape[1] &lt; 5:
        return None
    return {
        'R': data[:, 0], 'Vo': data[:, 1], 'eV': data[:, 2],
        'Vg': data[:, 3], 'Vd': data[:, 4],
        'Vb': data[:, 5] if data.shape[1] &gt; 5 else np.zeros(len(data)),
        'name': filepath.stem,
    }


def vbar(gal, Yd=YD, Yb=YB):
    return np.sqrt(np.maximum(
        gal['Vg']**2 + Yd * gal['Vd']**2 + Yb * gal['Vb']**2, 0))


def observables(gal):
    """V_flat, h_R, G*Sigma_0 (Yd-independent)"""
    R, Vo, Vd = gal['R'], gal['Vo'], gal['Vd']
    n = len(R)
    if n &lt; 5:
        return None
    outer = slice(2*n//3, n)
    V_flat = np.median(Vo[outer])
    idx_pk = max(np.argmax(np.abs(Vd)), 1)
    h_R = R[idx_pk] / 2.2
    if V_flat &lt;= 0 or h_R &lt;= 0.01:
        return None
    GS0 = (V_flat * 1e3)**2 / (h_R * kpc)
    return {'V_flat': V_flat, 'h_R': h_R, 'GS0': GS0}


# ============================================================
# g_c測定の6つの変種
# ============================================================
def gc_outer_third_median(gal):
    """v2デフォルト: 外側1/3の中央値"""
    R, Vo, Vb_ = gal['R'], gal['Vo'], vbar(gal)
    n = len(R)
    if n &lt; 5: return None
    s = slice(2*n//3, n)
    Rm = R[s] * kpc
    gc = np.median((Vo[s]*1e3)**2/Rm - (Vb_[s]*1e3)**2/Rm)
    return gc if gc &gt; 0 else None


def gc_outer_third_mean(gal):
    """外側1/3の平均値"""
    R, Vo, Vb_ = gal['R'], gal['Vo'], vbar(gal)
    n = len(R)
    if n &lt; 5: return None
    s = slice(2*n//3, n)
    Rm = R[s] * kpc
    gc = np.mean((Vo[s]*1e3)**2/Rm - (Vb_[s]*1e3)**2/Rm)
    return gc if gc &gt; 0 else None


def gc_outer_half_median(gal):
    """外側1/2の中央値"""
    R, Vo, Vb_ = gal['R'], gal['Vo'], vbar(gal)
    n = len(R)
    if n &lt; 5: return None
    s = slice(n//2, n)
    Rm = R[s] * kpc
    gc = np.median((Vo[s]*1e3)**2/Rm - (Vb_[s]*1e3)**2/Rm)
    return gc if gc &gt; 0 else None
def gc_last_point(gal):
    """最外点のみ"""
```

```
    R, Vo, Vb_ = gal['R'], gal['Vo'], vbar(gal)
    n = len(R)
    if n &lt; 3: return None
    Rm = R[-1] * kpc
    if Rm &lt;= 0: return None
    gc = (Vo[-1]*1e3)**2/Rm - (Vb_[-1]*1e3)**2/Rm
    return gc if gc &gt; 0 else None


def gc_flat_region(gal):
    """フラット領域検出: V_obsの変動が10%以内の最長連続区間"""
    R, Vo, Vb_ = gal['R'], gal['Vo'], vbar(gal)
    n = len(R)
    if n &lt; 5: return None

    # フラット領域: |V(i)-V(i-1)|/V(i) &lt; 0.1 が連続する区間
    best_start, best_len = 0, 0
    cur_start, cur_len = 0, 1
    for i in range(1, n):
        if Vo[i] &gt; 0 and abs(Vo[i] - Vo[i-1]) / Vo[i] &lt; 0.10:
            cur_len += 1
        else:
            if cur_len &gt; best_len:
                best_start, best_len = cur_start, cur_len
            cur_start, cur_len = i, 1
    if cur_len &gt; best_len:
        best_start, best_len = cur_start, cur_len

    if best_len &lt; 3:
        return None

    s = slice(best_start, best_start + best_len)
    Rm = R[s] * kpc
    gc = np.median((Vo[s]*1e3)**2/Rm - (Vb_[s]*1e3)**2/Rm)
    return gc if gc &gt; 0 else None


def gc_weighted_outer(gal):
    """誤差重み付き外側1/3"""
    R, Vo, eV, Vb_ = gal['R'], gal['Vo'], gal['eV'], vbar(gal)
    n = len(R)
    if n &lt; 5: return None
    s = slice(2*n//3, n)
    Rm = R[s] * kpc
    gc_pts = (Vo[s]*1e3)**2/Rm - (Vb_[s]*1e3)**2/Rm
    w = 1.0 / np.maximum(eV[s], 1.0)**2
    gc = np.average(gc_pts, weights=w)
    return gc if gc &gt; 0 else None


GC_METHODS = {
    'outer_1/3_median': gc_outer_third_median,
    'outer_1/3_mean':   gc_outer_third_mean,
    'outer_1/2_median': gc_outer_half_median,
    'last_point':       gc_last_point,
    'flat_region':      gc_flat_region,
    'weighted_outer':   gc_weighted_outer,
}


def alpha_fit(gc_arr, GS0_arr):
    x = np.log10(GS0_arr)
    y = np.log10(gc_arr)
    mask = np.isfinite(x) &amp;amp; np.isfinite(y)
    x, y = x[mask], y[mask]
    if len(x) &lt; 10: return None
    sl, ic, r, p, se = linregress(x, y)
    t_stat = (sl - 0.5) / se
    p05 = 2 * tdist.sf(abs(t_stat), df=len(x)-2)
    return {'alpha': sl, 'e_alpha': se, 'p05': p05, 'r': r, 'N': len(x)}


def main():
    print('=' * 70)
    print('SPARC g_c Method Comparison')
    print('=' * 70)

    # === 銀河読み込み ===
    files = sorted(SPARC_DIR.glob('*.dat'))
    if not files:
        files = sorted(SPARC_DIR.glob('*_rotmod.dat'))
    if not files:
        print(f'ERROR: {SPARC_DIR}')
        sys.exit(1)
```

```
galaxies = []
obs_list = []
for f in files:
    g = load_sparc(f)
    if g is None: continue
    o = observables(g)
    if o is None: continue
    galaxies.append(g)
    obs_list.append(o)
print(f'{len(galaxies)} galaxies loaded')

GS0_all = np.array([o['GS0'] for o in obs_list])

# === TA3比較（ファイルが存在する場合）===
ta3_path = None
for p in TA3_CANDIDATES:
    if p.exists():
        ta3_path = p
        break

ta3_gc = {}
if ta3_path:
    print(f'¥nTA3 found: {ta3_path}')
    import csv
    with open(ta3_path, 'r', encoding='utf-8-sig') as f:
        reader = csv.DictReader(f)
        for row in reader:
            name = row.get('Name', row.get('name', '')).strip()
            gc_val = row.get('gc', row.get('g_c', row.get('gc_indep', '')))
            try:
                ta3_gc[name] = float(gc_val)
            except (ValueError, TypeError):
                pass
    print(f'  {len(ta3_gc)} galaxies in TA3')

    # 銀河ごとのg_c比較
    matched = []
    for gal, obs in zip(galaxies, obs_list):
        name = gal['name']
        # TA3名マッチ（完全一致 or _Rotmod除去）
        ta3_name = name.replace('_rotmod', '').replace('_Rotmod', '')
        gc_ta3 = ta3_gc.get(name) or ta3_gc.get(ta3_name)
        if gc_ta3 is None:
            continue
        gc_v2 = gc_outer_third_median(gal)
        if gc_v2 is None:
            continue
        matched.append({
            'name': name,
            'gc_ta3': gc_ta3,
            'gc_v2': gc_v2,
            'ratio': gc_v2 / gc_ta3 if gc_ta3 &gt; 0 else np.nan,
            'GS0': obs['GS0'],
        })

    print(f'  Matched: {len(matched)} galaxies')

    if len(matched) &gt; 10:
        ratios = np.array([m['ratio'] for m in matched if np.isfinite(m['ratio'])])
        print(f'  g_c(v2) / g_c(TA3):')
        print(f'    median = {np.median(ratios):.3f}')
        print(f'    mean   = {np.mean(ratios):.3f}')
        print(f'    std    = {np.std(ratios):.3f}')
        print(f'    range  = [{np.min(ratios):.3f}, {np.max(ratios):.3f}]')

        # TA3のg_cでalpha測定
        gc_ta3_arr = np.array([m['gc_ta3'] for m in matched])
        gs0_ta3_arr = np.array([m['GS0'] for m in matched])
        fit_ta3 = alpha_fit(gc_ta3_arr, gs0_ta3_arr)
        if fit_ta3:
            print(f'¥n  TA3 g_c -&gt; alpha = {fit_ta3["alpha"]:.3f} +/- {fit_ta3["e_alpha"]:.3f}, '
                  f'p(0.5) = {fit_ta3["p05"]:.4f}, N={fit_ta3["N"]}')

        # v2のg_cでalpha測定（同じ銀河セット）
        gc_v2_arr = np.array([m['gc_v2'] for m in matched])
        fit_v2m = alpha_fit(gc_v2_arr, gs0_ta3_arr)
        if fit_v2m:
            print(f'  v2 g_c  -&gt; alpha = {fit_v2m["alpha"]:.3f} +/- {fit_v2m["e_alpha"]:.3f}, '
                  f'p(0.5) = {fit_v2m["p05"]:.4f}, N={fit_v2m["N"]}')

        # 差の有意性
        if fit_ta3 and fit_v2m:
            z = (fit_v2m['alpha'] - fit_ta3['alpha']) / np.sqrt(
                fit_v2m['e_alpha']**2 + fit_ta3['e_alpha']**2)
```

```
            print(f'  Delta-alpha z-score: {z:.2f}')
else:
    print('¥nTA3_gc_independent.csv not found. Skipping direct comparison.')
    print('  Searched:', [str(p) for p in TA3_CANDIDATES])

# ============================================================
# g_c定義変種テスト
# ============================================================
print(f'¥n{"="*70}')
print('g_c Definition Variants Test (Y_d=0.5, Y_b=0.7)')
print(f'{"="*70}')
print(f'{"Method":&lt;22} {"N":&gt;5} {"alpha":&gt;8} {"SE":&gt;8} {"p(0.5)":&gt;10} {"r":&gt;7} {"reject":&gt;7}')
print('-' * 70)

variant_results = {}
for method_name, method_func in GC_METHODS.items():
    gc_list = []
    gs0_list = []
    for gal, obs in zip(galaxies, obs_list):
        gc = method_func(gal)
        if gc is not None:
            gc_list.append(gc)
            gs0_list.append(obs['GS0'])

    fit = alpha_fit(np.array(gc_list), np.array(gs0_list))
    if fit:
        reject = 'YES' if fit['p05'] &lt; 0.05 else 'no'
        print(f'{method_name:&lt;22} {fit["N"]:5d} {fit["alpha"]:8.3f} {fit["e_alpha"]:8.3f} '
              f'{fit["p05"]:10.4f} {fit["r"]:7.3f} {reject:&gt;7}')
        variant_results[method_name] = fit
    else:
        print(f'{method_name:&lt;22}   -- fit failed --')

# ============================================================
# Bootstrap各手法
# ============================================================
print(f'¥n{"="*70}')
print(f'Bootstrap 95% CI (N_boot={N_BOOT})')
print(f'{"="*70}')

rng = np.random.RandomState(RNG_SEED)

for method_name, method_func in GC_METHODS.items():
    gc_list, gs0_list = [], []
    for gal, obs in zip(galaxies, obs_list):
        gc = method_func(gal)
        if gc is not None:
            gc_list.append(gc)
            gs0_list.append(obs['GS0'])
    gc_arr = np.array(gc_list)
    gs0_arr = np.array(gs0_list)
    N = len(gc_arr)
    if N &lt; 20:
        continue

    alphas = []
    for _ in range(N_BOOT):
        idx = rng.choice(N, N, replace=True)
        fit = alpha_fit(gc_arr[idx], gs0_arr[idx])
        if fit:
            alphas.append(fit['alpha'])
    alphas = np.array(alphas)
    ci = np.percentile(alphas, [2.5, 97.5])
    has_05 = ci[0] &lt;= 0.5 &lt;= ci[1]
    print(f'  {method_name:&lt;22}: {np.mean(alphas):.3f} +/- {np.std(alphas):.3f}, '
          f'95%CI=[{ci[0]:.3f}, {ci[1]:.3f}], 0.5 in CI: {"YES" if has_05 else "NO"}')

# ============================================================
# 図の生成
# ============================================================
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# (a) alpha by method
ax = axes[0]
methods = list(variant_results.keys())
alphas = [variant_results[m]['alpha'] for m in methods]
errors = [variant_results[m]['e_alpha'] for m in methods]
y_pos = np.arange(len(methods))

ax.barh(y_pos, alphas, xerr=errors, color='#1a1a2e', alpha=0.7, capsize=3)
ax.axvline(0.5, color='#e94560', ls='--', lw=2, label='alpha=0.5')
ax.axvline(0.545, color='green', ls=':', lw=2, label='alpha=0.545 (TA3)')
ax.set_yticks(y_pos)
ax.set_yticklabels(methods, fontsize=9)
```

```
    ax.set_xlabel('alpha')
    ax.set_title('(a) alpha by g_c definition')
    ax.legend(fontsize=8)
    ax.grid(True, alpha=0.3, axis='x')

    # (b) TA3 vs v2 scatter (if available)
    ax = axes[1]
    if ta3_path and len(matched) &gt; 10:
        gc_ta3_plot = [m['gc_ta3'] for m in matched]
        gc_v2_plot = [m['gc_v2'] for m in matched]
        ax.scatter(gc_ta3_plot, gc_v2_plot, s=15, alpha=0.5, color='#1a1a2e')
        lim = [min(min(gc_ta3_plot), min(gc_v2_plot)),
               max(max(gc_ta3_plot), max(gc_v2_plot))]
        ax.plot(lim, lim, 'k--', alpha=0.3, label='1:1')
        ax.set_xlabel('g_c (TA3)')
        ax.set_ylabel('g_c (v2: outer 1/3 median)')
        ax.set_title(f'(b) g_c comparison (N={len(matched)})')
        ax.set_xscale('log')
        ax.set_yscale('log')
        ax.legend()
    else:
        ax.text(0.5, 0.5, 'TA3 data not available¥n¥nRun gc_geometric_mean_test.py¥n'
                'to generate TA3_gc_independent.csv¥nthen re-run this script',
                ha='center', va='center', fontsize=11, transform=ax.transAxes)
        ax.set_title('(b) g_c: TA3 vs v2 (requires TA3 file)')

    plt.tight_layout()
    fig_path = OUTDIR / 'gc_method_comparison.png'
    plt.savefig(fig_path, dpi=150)
    print(f'¥nFigure: {fig_path}')

    # JSON
    summary = {
        'variant_results': {k: v for k, v in variant_results.items()},
        'ta3_found': ta3_path is not None,
        'ta3_matched': len(matched) if ta3_path else 0,
    }
    with open(OUTDIR / 'gc_method_comparison.json', 'w') as f:
        json.dump(summary, f, indent=2, default=float)
    print(f'Results: {OUTDIR / "gc_method_comparison.json"}')

    # ============================================================
    # 結果の解釈ガイド
    # ============================================================
    print(f'¥n{"="*70}')
    print('Interpretation Guide')
    print(f'{"="*70}')
    print('''
If all methods give alpha ~ 0.65-0.70:
  -&gt; The 0.125 gap vs TA3 is due to g_c definition, not Y_d.
  -&gt; TA3 likely uses a different approach (e.g., RC fitting, not simple subtraction).
  -&gt; Next step: read TA3 source code to understand its g_c definition.

If some method gives alpha ~ 0.54:
  -&gt; That method matches TA3. The gap is explained by g_c definition choice.
  -&gt; Document which definition is physically most appropriate.

If TA3 g_c values are systematically lower than v2:
  -&gt; TA3 may account for adiabatic contraction or non-circular motions,
     which reduce g_c and push alpha toward 0.5.
''')


if __name__ == '__main__':
    main()
```

sparc_alpha_decomposition.py

章: 第26章前提

目的: TA3 CSVの構造解析と要素分離(初版)

解析対象データ: TA3_gc_independent.csv, SPARC Rotmod_LTG

主要結果: TA3: gc_over_a0, upsilon_d(0.3-0.93), v_flat, rs_tanh
カラム特定。8組み合わせテスト実施。sparc_alpha_reproduce.pyで最終化。

ソースコード (465 lines, 16,531 bytes)

```
#!/usr/bin/env python3
"""
sparc_alpha_decomposition.py
============================
TA3_gc_independent.csv を直接読み込み、alpha=0.545 を復元し、
各方法論的要素の寄与を分離する。

分離する要素:
  (1) gc_over_a0 の定義 (TA3 vs 直接計算)
  (2) Y_d の最適化 (個別 vs 0.5固定)
  (3) h_R の定義 (rs_tanh vs V_disk peak/2.2)
  (4) v_flat の定義 (tanh fit vs 外側median)
  (5) 下限打切り (gc_over_a0=0.1 or v_flat=5.0)
  (6) gc_circular vs gc_over_a0

テスト:
  A) TA3の全パラメータ再現 -&gt; alpha確認
  B) 要素を1つずつv2方式に置換 -&gt; alphaの変化を測定
  C) v2の全パラメータで計算 -&gt; alpha=0.670確認
  -&gt; 差分がどの要素に帰属するかを定量化

実行: uv run --with scipy --with matplotlib python sparc_alpha_decomposition.py

前提: TA3_gc_independent.csv がカレントまたは指定パスに存在すること
"""

import numpy as np
import csv
import sys
import json
from pathlib import Path
from scipy.stats import linregress, t as tdist
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib import font_manager as _fm
for _fp in ['/usr/share/fonts/opentype/ipafont-gothic/ipag.ttf',
            '/usr/share/fonts/opentype/ipafont-gothic/ipagp.ttf']:
    try: _fm.fontManager.addfont(_fp)
    except: pass
plt.rcParams['font.family'] = 'IPAGothic'
plt.rcParams['axes.unicode_minus'] = False

# === 設定 ===
SPARC_DIR = Path(r"D:¥ドキュメント¥エントロピー¥新膜宇宙論¥これまでの軌跡¥パイソン¥Rotmod_LTG")
if not SPARC_DIR.exists():
    SPARC_DIR = Path("Rotmod_LTG")
if not SPARC_DIR.exists():
    SPARC_DIR = Path(".")

TA3_CANDIDATES = [
    Path(r"D:¥ドキュメント¥エントロピー¥新膜宇宙論¥これまでの軌跡¥パイソン¥TA3_gc_independent.csv"),
    Path("TA3_gc_independent.csv"),
    Path("../TA3_gc_independent.csv"),
]

OUTDIR = Path("alpha_decomposition_output")
OUTDIR.mkdir(exist_ok=True)

# 物理定数
G_SI = 6.674e-11
Msun = 1.989e30
pc = 3.086e16
kpc = 1e3 * pc
a0 = 1.2e-10  # m/s^2

N_BOOT = 1000
RNG_SEED = 42
def load_ta3():
```

```
    """TA3 CSV読み込み"""
    for p in TA3_CANDIDATES:
        if p.exists():
            print(f'TA3 found: {p}')
            data = []
            with open(p, 'r', encoding='utf-8-sig') as f:
                reader = csv.DictReader(f)
                for row in reader:
                    try:
                        name = row.get('galaxy', '').strip()
                        vflat = float(row.get('v_flat', row.get(' v_flat', '0')).strip())
                        rs = float(row.get('rs_tanh', row.get(' rs_tanh', '0')).strip())
                        ud = float(row.get('upsilon_d', row.get(' upsilon_d', '0.5')).strip())
                        gc_a0 = float(row.get('gc_over_a0', row.get(' gc_over_a0', '0')).strip())
                        gc_err = float(row.get('gc_err', row.get(' gc_err', '0')).strip())
                        chi2 = float(row.get('chi2_dof', row.get(' chi2_dof', '99')).strip())
                        gc_circ_str = row.get('gc_circular', row.get(' gc_circular', '')).strip()
                        gc_circ = float(gc_circ_str) if gc_circ_str else np.nan

                        if name and vflat &gt; 0 and gc_a0 &gt; 0:
                            data.append({
                                'name': name,
                                'v_flat_ta3': vflat,        # km/s
                                'rs_tanh': rs,               # kpc
                                'upsilon_d': ud,
                                'gc_over_a0': gc_a0,
                                'gc_ta3': gc_a0 * a0,        # m/s^2
                                'gc_err': gc_err,
                                'chi2_dof': chi2,
                                'gc_circular': gc_circ * a0 if np.isfinite(gc_circ) else np.nan,
                            })
                    except (ValueError, KeyError):
                        pass
            print(f'  {len(data)} galaxies loaded')
            return data
    print('ERROR: TA3_gc_independent.csv not found')
    sys.exit(1)


def load_sparc(filepath):
    try:
        d = np.loadtxt(filepath, comments='#')
    except:
        return None
    if d.ndim != 2 or d.shape[1] &lt; 5:
        return None
    return {
        'R': d[:, 0], 'Vo': d[:, 1], 'eV': d[:, 2],
        'Vg': d[:, 3], 'Vd': d[:, 4],
        'Vb': d[:, 5] if d.shape[1] &gt; 5 else np.zeros(len(d)),
        'name': filepath.stem,
    }


def name_norm(n):
    import re
    return re.sub(r'[¥s¥-_]', '', n.upper().replace('_ROTMOD', '').replace('_LSBALL', ''))


def gc_direct(gal, Yd, Yb=0.7):
    """直接計算: median(g_obs - g_bar) in outer 1/3"""
    R, Vo, Vg, Vd, Vb = gal['R'], gal['Vo'], gal['Vg'], gal['Vd'], gal['Vb']
    n = len(R)
    if n &lt; 5: return None
    Vbar = np.sqrt(np.maximum(Vg**2 + Yd*Vd**2 + Yb*Vb**2, 0))
    s = slice(2*n//3, n)
    Rm = R[s] * kpc
    if len(Rm) &lt; 2 or np.any(Rm &lt;= 0): return None
    gc = np.median((Vo[s]*1e3)**2/Rm - (Vbar[s]*1e3)**2/Rm)
    return gc if gc &gt; 0 else None


def hR_vdisk_peak(gal):
    """h_R = V_disk peak / 2.2"""
    Vd = gal['Vd']
    R = gal['R']
    idx = max(np.argmax(np.abs(Vd)), 1)
    h = R[idx] / 2.2
    return h if h &gt; 0.01 else None


def vflat_obs(gal):
    """外側1/3のV_obs中央値"""
    n = len(gal['R'])
```

```
    if n &lt; 5: return None
    return np.median(gal['Vo'][2*n//3:])


def alpha_fit(gc_arr, gs0_arr):
    x = np.log10(gs0_arr)
    y = np.log10(gc_arr)
    mask = np.isfinite(x) &amp;amp; np.isfinite(y)
    x, y = x[mask], y[mask]
    if len(x) &lt; 10: return None
    sl, ic, r, p, se = linregress(x, y)
    t_stat = (sl - 0.5) / se
    p05 = 2 * tdist.sf(abs(t_stat), df=len(x)-2)
    return {'alpha': sl, 'e_alpha': se, 'p05': p05, 'r': r, 'N': len(x)}


def bootstrap_alpha(gc_arr, gs0_arr, n_boot=N_BOOT):
    rng = np.random.RandomState(RNG_SEED)
    N = len(gc_arr)
    alphas = []
    for _ in range(n_boot):
        idx = rng.choice(N, N, replace=True)
        fit = alpha_fit(gc_arr[idx], gs0_arr[idx])
        if fit: alphas.append(fit['alpha'])
    return np.array(alphas)


def main():
    print('=' * 70)
    print('alpha=0.545 Decomposition Analysis')
    print('=' * 70)

    # === データ読み込み ===
    ta3 = load_ta3()

    files = sorted(SPARC_DIR.glob('*.dat'))
    if not files:
        files = sorted(SPARC_DIR.glob('*_rotmod.dat'))
    sparc = {}
    for f in files:
        g = load_sparc(f)
        if g: sparc[name_norm(g['name'])] = g
    print(f'SPARC: {len(sparc)} galaxies')

    # === TA3-SPARC クロスマッチ ===
    matched = []
    for t in ta3:
        nn = name_norm(t['name'])
        if nn in sparc:
            gal = sparc[nn]
            hR = hR_vdisk_peak(gal)
            vf = vflat_obs(gal)
            gc_v2_05 = gc_direct(gal, Yd=0.5)
            gc_v2_opt = gc_direct(gal, Yd=t['upsilon_d'])
            if hR and vf and gc_v2_05:
                matched.append({
                    **t,
                    'gal': gal,
                    'hR_vdpeak': hR,
                    'vflat_obs': vf,
                    'gc_v2_Yd05': gc_v2_05,
                    'gc_v2_Ydopt': gc_v2_opt,
                })
    print(f'Matched: {len(matched)}')

    # === 下限打切りの確認 ===
    gc_a0_vals = [m['gc_over_a0'] for m in matched]
    vflat_vals = [m['v_flat_ta3'] for m in matched]
    n_gc_floor = sum(1 for v in gc_a0_vals if v &lt;= 0.101)
    n_vf_floor = sum(1 for v in vflat_vals if v &lt;= 5.1)
    print(f'¥nFloor values:')
    print(f'  gc_over_a0 &lt;= 0.101: {n_gc_floor}/{len(matched)} ({100*n_gc_floor/len(matched):.1f}%)')
    print(f'  v_flat &lt;= 5.1: {n_vf_floor}/{len(matched)} ({100*n_vf_floor/len(matched):.1f}%)')

    # === Υ_d分布 ===
    ud_vals = np.array([m['upsilon_d'] for m in matched])
    print(f'¥nUpsilon_d distribution:')
    print(f'  median = {np.median(ud_vals):.3f}')
    print(f'  mean   = {np.mean(ud_vals):.3f}')
    print(f'  std    = {np.std(ud_vals):.3f}')
    print(f'  range  = [{np.min(ud_vals):.3f}, {np.max(ud_vals):.3f}]')
    print(f'  IQR    = [{np.percentile(ud_vals,25):.3f}, {np.percentile(ud_vals,75):.3f}]')

    # ============================================================
```

```
# Alpha測定: 組み合わせ分離テスト
# ============================================================
print(f'¥n{"="*70}')
print('Alpha Decomposition: Element-by-Element Substitution')
print(f'{"="*70}')

# G*Sigma_0 の3つの定義
def gs0_ta3_rs(m):
    """TA3: v_flat^2 / rs_tanh"""
    if m['rs_tanh'] &gt; 0:
        return (m['v_flat_ta3']*1e3)**2 / (m['rs_tanh']*kpc)
    return None

def gs0_ta3_vf_v2_hr(m):
    """TA3 v_flat + v2 h_R"""
    return (m['v_flat_ta3']*1e3)**2 / (m['hR_vdpeak']*kpc)

def gs0_v2(m):
    """v2: vflat_obs^2 / hR_vdpeak"""
    return (m['vflat_obs']*1e3)**2 / (m['hR_vdpeak']*kpc)

# 組み合わせテスト
tests = [
    # (label, gc_source, gs0_func, filter_desc)
    ('A: TA3全再現¥n  gc=TA3, GS0=vflat_ta3^2/rs_tanh',
     lambda m: m['gc_ta3'], gs0_ta3_rs, 'none'),

    ('B: TA3 gc + v2のGS0¥n  gc=TA3, GS0=vflat_obs^2/hR_vdpeak',
     lambda m: m['gc_ta3'], gs0_v2, 'none'),

    ('C: TA3 gc + TA3 vflat + v2 hR¥n  gc=TA3, GS0=vflat_ta3^2/hR_vdpeak',
     lambda m: m['gc_ta3'], gs0_ta3_vf_v2_hr, 'none'),

    ('D: v2 gc(Yd=opt) + TA3のGS0¥n  gc=direct(Yd_opt), GS0=vflat_ta3^2/rs_tanh',
     lambda m: m['gc_v2_Ydopt'], gs0_ta3_rs, 'none'),

    ('E: v2 gc(Yd=0.5) + TA3のGS0¥n  gc=direct(Yd=0.5), GS0=vflat_ta3^2/rs_tanh',
     lambda m: m['gc_v2_Yd05'], gs0_ta3_rs, 'none'),

    ('F: v2全パラメータ¥n  gc=direct(Yd=0.5), GS0=vflat_obs^2/hR_vdpeak',
     lambda m: m['gc_v2_Yd05'], gs0_v2, 'none'),

    ('G: v2 gc(Yd=opt) + v2のGS0¥n  gc=direct(Yd_opt), GS0=vflat_obs^2/hR_vdpeak',
     lambda m: m.get('gc_v2_Ydopt'), gs0_v2, 'none'),

    ('H: gc_circular + TA3のGS0¥n  gc=gc_circular, GS0=vflat_ta3^2/rs_tanh',
     lambda m: m.get('gc_circular'), gs0_ta3_rs, 'none'),
]

# 下限除外テスト
tests.append(
    ('A2: TA3全再現 (floor除外)¥n  gc_over_a0&gt;0.101 &amp;amp; vflat&amp;gt;5.1',
     lambda m: m['gc_ta3'], gs0_ta3_rs, 'no_floor'),
)

print(f'¥n{"Label":&lt;50} {"N":&gt;5} {"alpha":&gt;7} {"SE":&gt;7} {"p(0.5)":&gt;9} {"0.5?":&gt;5}')
print('-' * 85)

decomp_results = {}
for label, gc_func, gs0_func, filt in tests:
    gc_list, gs0_list = [], []
    for m in matched:
        # フィルタ
        if filt == 'no_floor':
            if m['gc_over_a0'] &lt;= 0.101 or m['v_flat_ta3'] &lt;= 5.1:
                continue

        gc = gc_func(m)
        gs0 = gs0_func(m)
        if gc and gs0 and gc &gt; 0 and gs0 &gt; 0 and np.isfinite(gc) and np.isfinite(gs0):
            gc_list.append(gc)
            gs0_list.append(gs0)

    gc_arr = np.array(gc_list)
    gs0_arr = np.array(gs0_list)
    fit = alpha_fit(gc_arr, gs0_arr)

    label_short = label.split('¥n')[0]
    if fit:
        ok = 'YES' if fit['p05'] &gt; 0.05 else 'no'
        print(f'{label_short:&lt;50} {fit["N"]:5d} {fit["alpha"]:7.3f} {fit["e_alpha"]:7.3f} '
              f'{fit["p05"]:9.4f} {ok:&gt;5}')
        decomp_results[label_short] = fit
    else:
```

```
        print(f'{label_short:&lt;50}  -- failed --')

# ============================================================
# 差分分析: どの要素がどれだけalphaを動かすか
# ============================================================
print(f'¥n{"="*70}')
print('Element Attribution (delta-alpha)')
print(f'{"="*70}')

def get_alpha(key):
    for k, v in decomp_results.items():
        if k.startswith(key):
            return v['alpha']
    return None

a_A = get_alpha('A:')
a_B = get_alpha('B:')
a_C = get_alpha('C:')
a_D = get_alpha('D:')
a_E = get_alpha('E:')
a_F = get_alpha('F:')
a_G = get_alpha('G:')

if all(v is not None for v in [a_A, a_B, a_C, a_D, a_E, a_F]):
    print(f'¥n  Base (A: TA3 full):        alpha = {a_A:.3f}')
    print(f'  Target (F: v2 full):       alpha = {a_F:.3f}')
    print(f'  Total gap:                 Delta = {a_F - a_A:+.3f}')
    print()

    # GS0の影響 (A-&gt;B: rs_tanh -&gt; hR_vdpeak, keeping gc=TA3)
    print(f'  A-&gt;B (GS0: rs_tanh -&gt; hR_vdpeak):  {a_B - a_A:+.3f}')

    # vflatの影響 (B-&gt;C or equivalent)
    if a_C is not None:
        print(f'  A-&gt;C (hR only: rs_tanh -&gt; hR_vdpeak, keep vflat_ta3): {a_C - a_A:+.3f}')
        print(f'  C-&gt;B (vflat: ta3 -&gt; obs, keep hR_vdpeak):  {a_B - a_C:+.3f}')

    # gc定義の影響 (A-&gt;D: TA3 gc -&gt; direct gc with Yd_opt)
    print(f'  A-&gt;D (gc: TA3 -&gt; direct(Yd_opt)):  {a_D - a_A:+.3f}')

    # Yd最適化の影響 (D-&gt;E: Yd_opt -&gt; Yd=0.5)
    print(f'  D-&gt;E (Yd: optimal -&gt; 0.5):         {a_E - a_D:+.3f}')

    # 残差
    print(f'  E-&gt;F (GS0: ta3 -&gt; v2):             {a_F - a_E:+.3f}')

    print(f'¥n  Decomposition check: A + sum = {a_A + (a_B-a_A) + (a_D-a_A) + (a_E-a_D) + (a_F-a_E):.3f} (should be ~{a_F:.3f})')
    print(f'  Note: elements are not strictly additive due to interactions')

# ============================================================
# 図の生成
# ============================================================
fig, axes = plt.subplots(2, 2, figsize=(14, 11))

# (a) alpha by combination
ax = axes[0, 0]
labels_short = [k.split(':')[0] for k in decomp_results.keys()]
alphas_plot = [v['alpha'] for v in decomp_results.values()]
errors_plot = [v['e_alpha'] for v in decomp_results.values()]
y_pos = np.arange(len(labels_short))
colors_bar = ['#1a1a2e'] * len(labels_short)
ax.barh(y_pos, alphas_plot, xerr=errors_plot, color=colors_bar, alpha=0.7, capsize=3)
ax.axvline(0.5, color='#e94560', ls='--', lw=2, label='alpha=0.5')
ax.axvline(0.545, color='green', ls=':', lw=2, label='alpha=0.545 (original)')
ax.set_yticks(y_pos)
ax.set_yticklabels(labels_short, fontsize=9)
ax.set_xlabel('alpha')
ax.set_title('(a) alpha by parameter combination')
ax.legend(fontsize=8, loc='lower right')
ax.grid(True, alpha=0.3, axis='x')

# (b) Y_d分布
ax = axes[0, 1]
ax.hist(ud_vals, bins=20, color='#1a1a2e', alpha=0.7, edgecolor='white')
ax.axvline(0.5, color='#e94560', ls='--', lw=2, label='Y_d=0.5 (fixed)')
ax.axvline(np.median(ud_vals), color='green', ls=':', lw=2,
           label=f'median={np.median(ud_vals):.2f}')
ax.set_xlabel('Y_d (TA3 optimized)')
ax.set_ylabel('Count')
ax.set_title('(b) TA3 per-galaxy Y_d distribution')
ax.legend(fontsize=8)
ax.grid(True, alpha=0.3)

# (c) gc_ta3 vs gc_v2(Yd=0.5)
```

```
    ax = axes[1, 0]
    gc_ta3_plot = [m['gc_ta3'] for m in matched if m['gc_v2_Yd05']]
    gc_v2_plot = [m['gc_v2_Yd05'] for m in matched if m['gc_v2_Yd05']]
    ax.scatter(gc_ta3_plot, gc_v2_plot, s=15, alpha=0.5, color='#1a1a2e')
    lim = [min(min(gc_ta3_plot), min(gc_v2_plot)),
           max(max(gc_ta3_plot), max(gc_v2_plot))]
    ax.plot(lim, lim, 'k--', alpha=0.3, label='1:1')
    ax.set_xlabel('g_c (TA3: tanh fit, Yd optimized)')
    ax.set_ylabel('g_c (v2: median, Yd=0.5)')
    ax.set_title(f'(c) g_c comparison (N={len(gc_ta3_plot)})')
    ax.set_xscale('log')
    ax.set_yscale('log')
    ax.legend()
    ax.grid(True, alpha=0.3)

    # (d) gc_ta3 vs gc_v2(Yd=opt)
    ax = axes[1, 1]
    gc_vopt = [m['gc_v2_Ydopt'] for m in matched if m['gc_v2_Ydopt'] and m['gc_v2_Ydopt'] &gt; 0]
    gc_ta3_opt = [m['gc_ta3'] for m in matched if m['gc_v2_Ydopt'] and m['gc_v2_Ydopt'] &gt; 0]
    if gc_vopt:
        ax.scatter(gc_ta3_opt, gc_vopt, s=15, alpha=0.5, color='#e94560')
        lim2 = [min(min(gc_ta3_opt), min(gc_vopt)),
                max(max(gc_ta3_opt), max(gc_vopt))]
        ax.plot(lim2, lim2, 'k--', alpha=0.3, label='1:1')
        ax.set_xlabel('g_c (TA3)')
        ax.set_ylabel('g_c (v2: median, Yd=TA3 optimized)')
        ax.set_title(f'(d) g_c with matched Yd (N={len(gc_vopt)})')
        ax.set_xscale('log')
        ax.set_yscale('log')
        ax.legend()
    ax.grid(True, alpha=0.3)

    plt.suptitle('alpha=0.545 Decomposition Analysis', fontsize=13, y=1.01)
    plt.tight_layout()
    fig_path = OUTDIR / 'alpha_decomposition.png'
    plt.savefig(fig_path, dpi=150, bbox_inches='tight')
    print(f'¥nFigure: {fig_path}')

    # JSON
    summary = {
        'n_matched': len(matched),
        'n_gc_floor': n_gc_floor,
        'n_vf_floor': n_vf_floor,
        'Yd_stats': {
            'median': float(np.median(ud_vals)),
            'mean': float(np.mean(ud_vals)),
            'std': float(np.std(ud_vals)),
        },
        'decomposition': {k: v for k, v in decomp_results.items()},
    }
    with open(OUTDIR / 'alpha_decomposition.json', 'w') as f:
        json.dump(summary, f, indent=2, default=float)
    print(f'Results: {OUTDIR / "alpha_decomposition.json"}')


if __name__ == '__main__':
    main()
```

sparc_alpha_reproduce.py

章: 第26章(最終版)

目的: alpha=0.545完全分解: 元パイプライン完全再現+要素置換

解析対象データ: SPARC Rotmod_LTG, TA3_gc_independent.csv, phase1/sparc_results.csv

主要結果: 完全再現: N=160, alpha=0.545, p=0.273。核心:
Yd個別最適化が本質(Delta=-0.113)。g_c手法は無関係(Delta=+0.001)。phase1=TA3は完全同一値。

ソースコード (577 lines, 19,814 bytes)

```
#!/usr/bin/env python3
"""
sparc_alpha_reproduce.py
========================
gc_geometric_mean_test.py の手法を完全に再現し、alpha=0.545を復元する。

元パイプラインの手法（4つの核心要素）:
  1. g_c = gc_over_a0 * a0  (TA3そのまま)
  2. h_R = r_pk / 2.15      (v2は2.2)
  3. r_pk = argmax(sqrt(ud) * |V_disk|)  (v2は単純V_diskピーク)
  4. r_pk &gt;= r.max()*0.9 の銀河を除外  (v2はフィルタなし)
  5. vflat = phase1/sparc_results.csv の値 (TA3のv_flat_tanhではない)

本スクリプトのテスト:
  Step 1: 全4要素を正確に再現 -&gt; alpha=0.545の復元確認
  Step 2: 要素を1つずつv2方式に置換 -&gt; 各要素のDelta-alpha測定
  Step 3: phase1 vflat vs TA3 v_flat vs obs v_flat の3種比較

実行: uv run --with scipy --with matplotlib python sparc_alpha_reproduce.py

前提:
  - Rotmod_LTG/ (SPARC回転曲線)
  - TA3_gc_independent.csv
  - phase1/sparc_results.csv (vflat, ud)
  パスは環境に合わせて修正してください。
"""

import numpy as np
import csv
import sys
import json
from pathlib import Path
from scipy.stats import linregress, t as tdist
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib import font_manager as _fm
for _fp in ['/usr/share/fonts/opentype/ipafont-gothic/ipag.ttf',
            '/usr/share/fonts/opentype/ipafont-gothic/ipagp.ttf']:
    try: _fm.fontManager.addfont(_fp)
    except: pass
plt.rcParams['font.family'] = 'IPAGothic'
plt.rcParams['axes.unicode_minus'] = False

# === パス設定 ===
BASE = Path(r"D:¥ドキュメント¥エントロピー¥新膜宇宙論¥これまでの軌跡¥パイソン")
SPARC_DIR = BASE / "Rotmod_LTG"
TA3_PATH = BASE / "TA3_gc_independent.csv"
PHASE1_PATH = BASE / "phase1" / "sparc_results.csv"

# フォールバック
if not SPARC_DIR.exists():
    SPARC_DIR = Path("Rotmod_LTG")
if not TA3_PATH.exists():
    TA3_PATH = Path("TA3_gc_independent.csv")
if not PHASE1_PATH.exists():
    PHASE1_PATH = Path("phase1/sparc_results.csv")

OUTDIR = Path("alpha_reproduce_output")
OUTDIR.mkdir(exist_ok=True)

a0 = 1.2e-10  # m/s^2
kpc = 3.086e19  # m

N_BOOT = 1000
RNG_SEED = 42


def name_norm(n):
```

```
    import re
    return re.sub(r'[¥s¥-_]', '', n.upper().replace('_ROTMOD', '').replace('_LSBALL', ''))


def load_ta3():
    data = {}
    with open(TA3_PATH, 'r', encoding='utf-8-sig') as f:
        reader = csv.DictReader(f)
        for row in reader:
            try:
                name = row.get('galaxy', '').strip()
                gc_a0 = float(row.get('gc_over_a0', row.get(' gc_over_a0', '0')).strip())
                vf_ta3 = float(row.get('v_flat', row.get(' v_flat', '0')).strip())
                ud_ta3 = float(row.get('upsilon_d', row.get(' upsilon_d', '0.5')).strip())
                if name and gc_a0 &gt; 0:
                    data[name_norm(name)] = {
                        'gc_a0': gc_a0,
                        'gc': gc_a0 * a0,
                        'vflat_ta3': vf_ta3,
                        'ud_ta3': ud_ta3,
                        'name_orig': name,
                    }
            except (ValueError, KeyError):
                pass
    print(f'TA3: {len(data)} galaxies')
    return data


def load_phase1():
    data = {}
    if not PHASE1_PATH.exists():
        print(f'WARNING: phase1 not found: {PHASE1_PATH}')
        print('  -&gt; Will use TA3 v_flat and ud as fallback')
        return None

    with open(PHASE1_PATH, 'r', encoding='utf-8-sig') as f:
        reader = csv.DictReader(f)
        headers = reader.fieldnames
        print(f'phase1 headers: {headers}')
        for row in reader:
            try:
                # 名前の候補カラム
                name = ''
                for key in ['galaxy', 'Galaxy', 'name', 'Name', 'GALAXY']:
                    if key in row and row[key].strip():
                        name = row[key].strip()
                        break
                if not name:
                    # 最初のカラムを試す
                    name = list(row.values())[0].strip()

                # vflatの候補カラム
                vflat = None
                for key in ['vflat', 'v_flat', 'Vflat', 'V_flat', 'vflat_kms']:
                    if key in row:
                        try:
                            vflat = float(row[key].strip())
                            break
                        except ValueError:
                            pass

                # udの候補カラム
                ud = None
                for key in ['ud', 'upsilon_d', 'Upsilon_d', 'Yd', 'ud_opt',
                            'upsilon_disk', 'Ud']:
                    if key in row:
                        try:
                            ud = float(row[key].strip())
                            break
                        except ValueError:
                            pass

                if name and vflat and vflat &gt; 0:
                    data[name_norm(name)] = {
                        'vflat_p1': vflat,
                        'ud_p1': ud if ud else 0.5,
                        'name_orig': name,
                    }
            except (ValueError, KeyError):
                pass
    print(f'phase1: {len(data)} galaxies')
    return data
def load_sparc(filepath):
    try:
```

```
        d = np.loadtxt(filepath, comments='#')
    except:
        return None
    if d.ndim != 2 or d.shape[1] &lt; 5:
        return None
    return {
        'R': d[:, 0], 'Vo': d[:, 1], 'eV': d[:, 2],
        'Vg': d[:, 3], 'Vd': d[:, 4],
        'Vb': d[:, 5] if d.shape[1] &gt; 5 else np.zeros(len(d)),
    }


def alpha_fit(gc_arr, gs0_arr):
    log_gc = np.log10(gc_arr)
    log_gs = np.log10(gs0_arr)
    log_a0 = np.log10(a0)

    # 元パイプラインの回帰: log(gc) vs log(GS0/a0) = log(GS0) - log(a0)
    x = log_gs - log_a0
    y = log_gc

    mask = np.isfinite(x) &amp;amp; np.isfinite(y)
    x, y = x[mask], y[mask]
    if len(x) &lt; 10:
        return None

    sl, ic, r, p, se = linregress(x, y)
    t_stat = (sl - 0.5) / se
    p05 = 2 * tdist.sf(abs(t_stat), df=len(x)-2)
    return {'alpha': sl, 'e_alpha': se, 'p05': p05, 'r': r, 'N': len(x)}


def main():
    print('=' * 70)
    print('alpha=0.545 Exact Reproduction')
    print('=' * 70)

    ta3 = load_ta3()
    phase1 = load_phase1()

    # SPARC読み込み
    files = sorted(SPARC_DIR.glob('*.dat'))
    if not files:
        files = sorted(SPARC_DIR.glob('*_rotmod.dat'))
    sparc = {}
    for f in files:
        g = load_sparc(f)
        if g:
            sparc[name_norm(f.stem)] = g
    print(f'SPARC: {len(sparc)} galaxies')

    # === クロスマッチ ===
    matched = []
    for nn, t in ta3.items():
        if nn not in sparc:
            continue
        gal = sparc[nn]
        R = gal['R']
        Vd = gal['Vd']
        n = len(R)
        if n &lt; 5:
            continue

        # phase1データ (あれば)
        p1 = phase1.get(nn, {}) if phase1 else {}
        vflat_p1 = p1.get('vflat_p1', t['vflat_ta3'])
        ud_p1 = p1.get('ud_p1', t['ud_ta3'])

        entry = {
            'nn': nn,
            'name': t['name_orig'],
            'gc': t['gc'],
            'gc_a0': t['gc_a0'],
            'vflat_ta3': t['vflat_ta3'],
            'ud_ta3': t['ud_ta3'],
            'vflat_p1': vflat_p1,
            'ud_p1': ud_p1,
            'R': R,
            'Vd': Vd,
            'Vo': gal['Vo'],
            'Vg': gal['Vg'],
            'Vb': gal['Vb'],
        }
        matched.append(entry)
```

```
print(f'Matched: {len(matched)}')

# ================================================================
# 各手法でのh_R計算
# ================================================================
for m in matched:
    R = m['R']
    Vd = m['Vd']
    n = len(R)

    # --- 元パイプライン方式 ---
    # vds = sqrt(ud) * |V_disk|
    vds_p1 = np.sqrt(m['ud_p1']) * np.abs(Vd)
    i_pk_p1 = np.argmax(vds_p1)
    r_pk_p1 = R[i_pk_p1]
    m['r_pk_original'] = r_pk_p1
    m['hR_original'] = r_pk_p1 / 2.15 if r_pk_p1 &gt; 0 else None
    m['r_pk_at_edge'] = (r_pk_p1 &gt;= R.max() * 0.9) or (r_pk_p1 &lt; 0.01)

    # TA3のudで同じ計算
    vds_ta3 = np.sqrt(m['ud_ta3']) * np.abs(Vd)
    i_pk_ta3 = np.argmax(vds_ta3)
    r_pk_ta3 = R[i_pk_ta3]
    m['hR_ta3_ud'] = r_pk_ta3 / 2.15 if r_pk_ta3 &gt; 0 else None
    m['r_pk_ta3_edge'] = (r_pk_ta3 &gt;= R.max() * 0.9) or (r_pk_ta3 &lt; 0.01)

    # --- v2方式 ---
    i_pk_v2 = max(np.argmax(np.abs(Vd)), 1)
    r_pk_v2 = R[i_pk_v2]
    m['hR_v2'] = r_pk_v2 / 2.2 if r_pk_v2 &gt; 0 else None

    # --- v_flat ---
    outer = slice(2*n//3, n)
    m['vflat_obs'] = np.median(m['Vo'][outer]) if n &gt;= 5 else None

# ================================================================
# テスト: 組み合わせマトリクス
# ================================================================
print(f'¥n{"="*70}')
print('Combination Matrix')
print(f'{"="*70}')

def compute_alpha(label, gc_func, gs0_func, filter_func):
    gc_list, gs0_list = [], []
    n_filtered = 0
    for m in matched:
        if not filter_func(m):
            n_filtered += 1
            continue
        gc = gc_func(m)
        gs0 = gs0_func(m)
        if gc and gs0 and gc &gt; 0 and gs0 &gt; 0:
            gc_list.append(gc)
            gs0_list.append(gs0)

    gc_arr = np.array(gc_list)
    gs0_arr = np.array(gs0_list)
    fit = alpha_fit(gc_arr, gs0_arr)
    if fit:
        ok = 'YES' if fit['p05'] &gt; 0.05 else 'no'
        print(f'{label:&lt;55} N={fit["N"]:3d} (filt={n_filtered:2d}) '
              f'alpha={fit["alpha"]:.3f}+/-{fit["e_alpha"]:.3f} '
              f'p={fit["p05"]:.4f} {ok}')
        return fit, gc_arr, gs0_arr
    else:
        print(f'{label:&lt;55} FAILED')
        return None, None, None

# フィルタ関数
def filt_original(m):
    return not m['r_pk_at_edge'] and m['hR_original'] and m['hR_original'] &gt; 0
def filt_ta3_edge(m):
    return not m['r_pk_ta3_edge'] and m['hR_ta3_ud'] and m['hR_ta3_ud'] &gt; 0
def filt_none(m):
    return True

# GS0関数
def gs0_orig(m):
    if m['hR_original'] and m['hR_original'] &gt; 0:
        return (m['vflat_p1'] * 1e3)**2 / (m['hR_original'] * kpc)
    return None
def gs0_orig_ta3vf(m):
    if m['hR_original'] and m['hR_original'] &gt; 0:
        return (m['vflat_ta3'] * 1e3)**2 / (m['hR_original'] * kpc)
```

```
    return None
def gs0_orig_obsvf(m):
    if m['hR_original'] and m['hR_original'] &gt; 0 and m['vflat_obs']:
        return (m['vflat_obs'] * 1e3)**2 / (m['hR_original'] * kpc)
    return None
def gs0_v2(m):
    if m['hR_v2'] and m['hR_v2'] &gt; 0 and m['vflat_obs']:
        return (m['vflat_obs'] * 1e3)**2 / (m['hR_v2'] * kpc)
    return None
def gs0_hybrid_215(m):
    """v2 peak位置だがdivisor=2.15"""
    if m['hR_v2'] and m['hR_v2'] &gt; 0 and m['vflat_obs']:
        hR_215 = m['hR_v2'] * 2.2 / 2.15  # rescale
        return (m['vflat_obs'] * 1e3)**2 / (hR_215 * kpc)
    return None

# gc関数
def gc_ta3(m):
    return m['gc']
def gc_direct_05(m):
    R, Vo, Vg, Vd, Vb = m['R'], m['Vo'], m['Vg'], m['Vd'], m['Vb']
    n = len(R)
    if n &lt; 5: return None
    Vbar = np.sqrt(np.maximum(Vg**2 + 0.5*Vd**2 + 0.7*Vb**2, 0))
    s = slice(2*n//3, n)
    Rm = R[s] * kpc
    if len(Rm) &lt; 2: return None
    gc = np.median((Vo[s]*1e3)**2/Rm - (Vbar[s]*1e3)**2/Rm)
    return gc if gc &gt; 0 else None
def gc_direct_opt(m):
    R, Vo, Vg, Vd, Vb = m['R'], m['Vo'], m['Vg'], m['Vd'], m['Vb']
    ud = m['ud_p1']
    n = len(R)
    if n &lt; 5: return None
    Vbar = np.sqrt(np.maximum(Vg**2 + ud*Vd**2 + 0.7*Vb**2, 0))
    s = slice(2*n//3, n)
    Rm = R[s] * kpc
    if len(Rm) &lt; 2: return None
    gc = np.median((Vo[s]*1e3)**2/Rm - (Vbar[s]*1e3)**2/Rm)
    return gc if gc &gt; 0 else None

print('¥n--- Step 1: Exact Reproduction ---')
fit_exact, _, _ = compute_alpha(
    'EXACT: gc=TA3, GS0=vflat_p1^2/(rpk_p1/2.15), edge_filter',
    gc_ta3, gs0_orig, filt_original)

print('¥n--- Step 2: Element Substitution ---')

# 2a: vflatをTA3に置換
compute_alpha(
    'vflat: p1 -&gt; TA3 (rest=original)',
    gc_ta3, gs0_orig_ta3vf, filt_original)

# 2b: vflatをobs中央値に置換
compute_alpha(
    'vflat: p1 -&gt; obs_median (rest=original)',
    gc_ta3, gs0_orig_obsvf, filt_original)

# 2c: h_R計算を2.15-&gt;2.2に変更
compute_alpha(
    'divisor: 2.15 -&gt; 2.2 (rest=original)',
    gc_ta3, gs0_hybrid_215, filt_original)

# 2d: edgeフィルタを外す
compute_alpha(
    'filter: edge_filter OFF (rest=original)',
    gc_ta3, gs0_orig, filt_none)

# 2e: r_pkをud加重からVdピークに変更
def gs0_v2peak_p1vf(m):
    if m['hR_v2'] and m['hR_v2'] &gt; 0:
        return (m['vflat_p1'] * 1e3)**2 / (m['hR_v2'] * kpc)
    return None
compute_alpha(
    'r_pk: sqrt(ud)*Vd -&gt; simple Vd (rest=original)',
    gc_ta3, gs0_v2peak_p1vf, filt_original)

# 2f: gc定義をdirect(Yd=0.5)に置換
compute_alpha(
    'gc: TA3 -&gt; direct(Yd=0.5) (rest=original)',
    gc_direct_05, gs0_orig, filt_original)

# 2g: gc定義をdirect(Yd=opt)に置換
compute_alpha(
```

```
    'gc: TA3 -&gt; direct(Yd=p1_opt) (rest=original)',
    gc_direct_opt, gs0_orig, filt_original)

print('¥n--- Step 3: Full v2 Method ---')
compute_alpha(
    'FULL_V2: gc=direct(0.5), GS0=obs^2/(Vd_pk/2.2), no_filter',
    gc_direct_05, gs0_v2, filt_none)

# ============================================================
# vflat比較
# ============================================================
print(f'¥n{"="*70}')
print('v_flat Comparison')
print(f'{"="*70}')

vf_p1_list, vf_ta3_list, vf_obs_list = [], [], []
for m in matched:
    if m['vflat_obs'] and m['vflat_p1'] &gt; 0:
        vf_p1_list.append(m['vflat_p1'])
        vf_ta3_list.append(m['vflat_ta3'])
        vf_obs_list.append(m['vflat_obs'])

vf_p1 = np.array(vf_p1_list)
vf_ta3 = np.array(vf_ta3_list)
vf_obs = np.array(vf_obs_list)

print(f'  N = {len(vf_p1)}')
print(f'  phase1 vs TA3:   median ratio = {np.median(vf_p1/vf_ta3):.3f}, '
      f'corr = {np.corrcoef(vf_p1, vf_ta3)[0,1]:.4f}')
print(f'  phase1 vs obs:   median ratio = {np.median(vf_p1/vf_obs):.3f}, '
      f'corr = {np.corrcoef(vf_p1, vf_obs)[0,1]:.4f}')
print(f'  TA3 vs obs:      median ratio = {np.median(vf_ta3/vf_obs):.3f}, '
      f'corr = {np.corrcoef(vf_ta3, vf_obs)[0,1]:.4f}')

# ============================================================
# Y_d比較 (phase1 vs TA3)
# ============================================================
if phase1:
    print(f'¥n{"="*70}')
    print('Upsilon_d Comparison (phase1 vs TA3)')
    print(f'{"="*70}')
    ud_p1_list, ud_ta3_list = [], []
    for m in matched:
        ud_p1_list.append(m['ud_p1'])
        ud_ta3_list.append(m['ud_ta3'])
    ud_p1_arr = np.array(ud_p1_list)
    ud_ta3_arr = np.array(ud_ta3_list)
    print(f'  phase1: median={np.median(ud_p1_arr):.3f}, '
          f'mean={np.mean(ud_p1_arr):.3f}, range=[{np.min(ud_p1_arr):.3f},{np.max(ud_p1_arr):.3f}]')
    print(f'  TA3:    median={np.median(ud_ta3_arr):.3f}, '
          f'mean={np.mean(ud_ta3_arr):.3f}, range=[{np.min(ud_ta3_arr):.3f},{np.max(ud_ta3_arr):.3f}]')
    print(f'  corr:   {np.corrcoef(ud_p1_arr, ud_ta3_arr)[0,1]:.4f}')

# ============================================================
# エッジフィルタの影響
# ============================================================
print(f'¥n{"="*70}')
print('Edge Filter Analysis')
print(f'{"="*70}')
n_edge = sum(1 for m in matched if m['r_pk_at_edge'])
n_ok = sum(1 for m in matched if not m['r_pk_at_edge'])
print(f'  Edge filtered: {n_edge}/{len(matched)} ({100*n_edge/len(matched):.1f}%)')
print(f'  Kept: {n_ok}')

# エッジ銀河のg_c分布
gc_edge = [m['gc_a0'] for m in matched if m['r_pk_at_edge']]
gc_ok = [m['gc_a0'] for m in matched if not m['r_pk_at_edge']]
if gc_edge:
    print(f'  Edge galaxies gc/a0: median={np.median(gc_edge):.3f}, '
          f'mean={np.mean(gc_edge):.3f}')
print(f'  Kept galaxies gc/a0: median={np.median(gc_ok):.3f}, '
      f'mean={np.mean(gc_ok):.3f}')

# ============================================================
# 図の生成
# ============================================================
fig, axes = plt.subplots(2, 2, figsize=(14, 11))

# (a) vflat比較
ax = axes[0, 0]
if len(vf_p1) &gt; 0:
    ax.scatter(vf_obs, vf_p1, s=15, alpha=0.5, label='phase1', color='#1a1a2e')
    ax.scatter(vf_obs, vf_ta3, s=15, alpha=0.5, label='TA3', color='#e94560')
    lim = [0, max(max(vf_obs), max(vf_p1), max(vf_ta3)) * 1.05]
```

```
        ax.plot(lim, lim, 'k--', alpha=0.3)
        ax.set_xlabel('V_flat (obs median) [km/s]')
        ax.set_ylabel('V_flat (phase1 / TA3) [km/s]')
        ax.set_title('(a) V_flat comparison')
        ax.legend()
    ax.grid(True, alpha=0.3)

    # (b) h_R比較
    ax = axes[0, 1]
    hR_orig = [m['hR_original'] for m in matched if m['hR_original'] and m['hR_v2']]
    hR_v2 = [m['hR_v2'] for m in matched if m['hR_original'] and m['hR_v2']]
    if hR_orig and hR_v2:
        colors_scat = ['#e94560' if m['r_pk_at_edge'] else '#1a1a2e'
                       for m in matched if m['hR_original'] and m['hR_v2']]
        ax.scatter(hR_v2, hR_orig, s=15, alpha=0.5, c=colors_scat)
        lim = [0, max(max(hR_v2), max(hR_orig)) * 1.05]
        ax.plot(lim, lim, 'k--', alpha=0.3, label='1:1')
        ax.set_xlabel('h_R (v2: Vd_peak/2.2) [kpc]')
        ax.set_ylabel('h_R (original: sqrt(ud)*Vd_peak/2.15) [kpc]')
        ax.set_title('(b) h_R comparison (red=edge-filtered)')
        ax.legend()
    ax.grid(True, alpha=0.3)

    # (c) Ud分布比較
    ax = axes[1, 0]
    if phase1:
        ax.hist(ud_p1_arr, bins=20, alpha=0.6, label='phase1', color='#1a1a2e')
        ax.hist(ud_ta3_arr, bins=20, alpha=0.6, label='TA3', color='#e94560')
        ax.axvline(0.5, color='green', ls='--', lw=2, label='Y_d=0.5')
        ax.set_xlabel('Y_d')
        ax.set_title('(c) Y_d: phase1 vs TA3')
        ax.legend()
    ax.grid(True, alpha=0.3)

    # (d) g_c(TA3) vs G*S0 回帰プロット
    ax = axes[1, 1]
    gc_plot, gs0_plot, edge_plot = [], [], []
    for m in matched:
        gs0 = gs0_orig(m)
        if gs0 and gs0 &gt; 0 and m['gc'] &gt; 0:
            gc_plot.append(m['gc'])
            gs0_plot.append(gs0)
            edge_plot.append(m['r_pk_at_edge'])

    if gc_plot:
        gc_p = np.array(gc_plot)
        gs0_p = np.array(gs0_plot)
        edge_p = np.array(edge_plot)

        ax.scatter(np.log10(gs0_p[~edge_p]/a0), np.log10(gc_p[~edge_p]),
                   s=15, alpha=0.5, color='#1a1a2e', label='kept')
        ax.scatter(np.log10(gs0_p[edge_p]/a0), np.log10(gc_p[edge_p]),
                   s=15, alpha=0.5, color='#e94560', label='edge-filtered')

        # 回帰線 (kept only)
        x_kept = np.log10(gs0_p[~edge_p]/a0)
        y_kept = np.log10(gc_p[~edge_p])
        mask = np.isfinite(x_kept) &amp; np.isfinite(y_kept)
        if np.sum(mask) &gt; 10:
            sl, ic, _, _, _ = linregress(x_kept[mask], y_kept[mask])
            x_line = np.linspace(np.min(x_kept[mask]), np.max(x_kept[mask]), 50)
            ax.plot(x_line, sl*x_line + ic, 'g-', lw=2,
                    label=f'alpha={sl:.3f}')

        ax.set_xlabel('log(G*Sigma_0 / a_0)')
        ax.set_ylabel('log(g_c)')
        ax.set_title('(d) Regression (green=kept, red=filtered)')
        ax.legend(fontsize=8)
    ax.grid(True, alpha=0.3)

    plt.suptitle('alpha=0.545 Reproduction &amp; Decomposition', fontsize=13, y=1.01)
    plt.tight_layout()
    fig_path = OUTDIR / 'alpha_reproduce.png'
    plt.savefig(fig_path, dpi=150, bbox_inches='tight')
    print(f'¥nFigure: {fig_path}')

    # JSON
    summary = {'n_matched': len(matched), 'n_edge_filtered': n_edge}
    with open(OUTDIR / 'alpha_reproduce.json', 'w') as f:
        json.dump(summary, f, indent=2, default=float)


if __name__ == '__main__':
    main()
```

probes_vbar_v2_local.py

章: 付録(V_barボトルネック)

目的: VizieR HIソース系統探索 + 既存データ確認

解析対象データ: VizieR 8カタログ探索

主要結果: V_gas(R)プロファイルを持つカタログはVizieRに存在しない。V_barボトルネック確定。

ソースコード (388 lines, 12,998 bytes)

```
#!/usr/bin/env python3
"""
probes_vbar_v2_local.py
=======================
修正版: 既存のローカルデータ（前回パイプラインの出力）を活用し、
WHISP/THINGS HIデータの取得を正しいソースから試みる。

前提:
  - probes_vbar_output/ に前回パイプライン出力が存在
  - または PROBES/S4G データを Zenodo/VizieR から再取得

HI データソース探索:
  1. THINGS de Blok+2008: VizieR のテーブル名を自動探索
  2. WHISP: 複数の関連カタログを試行
  3. Noordermeer+2007 (J/MNRAS/376/1513): 早期型銀河HI
  4. Lelli+2016 SPARC自体のV_gasカラム（参考値として）

実行: uv run --with scipy --with matplotlib --with requests python probes_vbar_v2_local.py
"""

import numpy as np
import sys
import json
import csv
import re
from pathlib import Path
from scipy.optimize import minimize_scalar
from scipy.special import i0, i1, k0, k1
from scipy.stats import linregress, t as tdist

OUTDIR = Path("probes_vbar_v2_output")
OUTDIR.mkdir(exist_ok=True)

# 物理定数
G_SI = 6.674e-11; Msun = 1.989e30; pc = 3.086e16; kpc = 1e3*pc; a0 = 1.2e-10
M_sun_W1 = 3.24

def norm(n):
    return re.sub(r'[¥s¥-_]', '', n.upper())


# ============================================================
# Part 1: VizieR テーブル探索（WHISP/THINGS の正しいテーブル名特定）
# ============================================================
def vizier_list_tables(catalog):
    """VizieRカタログ内のテーブル一覧を取得"""
    import requests
    url = f"https://vizier.cds.unistra.fr/viz-bin/VizieR-3?-source={catalog}&amp;-to=1"
    try:
        r = requests.get(url, timeout=30)
        # HTMLからテーブル名を抽出
        tables = re.findall(r'name="([^"]*table[^"]*)"', r.text, re.I)
        if not tables:
            tables = re.findall(rf'{re.escape(catalog)}/(¥w+)', r.text)
        return tables
    except:
        return []


def vizier_fetch(source, columns="all", max_rows=50000):
    """VizieR汎用取得（エラーハンドリング付き）"""
    import requests
    url = "https://vizier.cds.unistra.fr/viz-bin/asu-tsv"
    params = {
        "-source": source,
        "-out.max": str(max_rows),
        "-out.form": "tsv",
    }
    if columns != "all":
        params["-out"] = columns
```

```
    r = requests.get(url, params=params, timeout=60)

    if "does not exist" in r.text or len(r.text) &lt; 100:
        return None, r.text[:200]

    lines = [l for l in r.text.strip().split('¥n')
             if l and not l.startswith('#')]

    # ヘッダ行探索
    header_idx = None
    for i, l in enumerate(lines):
        if '¥t' in l and not l.startswith('-'):
            # 数値でない行をヘッダ候補とする
            cols = l.split('¥t')
            if cols and not cols[0].replace('.','').replace('-','').replace('+','').isdigit():
                header_idx = i
                break

    if header_idx is None:
        # 最初の非ダッシュ行をヘッダとする
        for i, l in enumerate(lines):
            if not l.startswith('-') and '¥t' in l:
                header_idx = i
                break

    if header_idx is None:
        return None, "No header found"

    headers = [h.strip() for h in lines[header_idx].split('¥t')]
    data = []
    for l in lines[header_idx+1:]:
        if l.startswith('-') or not l.strip():
            continue
        cols = l.split('¥t')
        if len(cols) &gt;= len(headers):
            data.append(dict(zip(headers, [c.strip() for c in cols])))

    return {'headers': headers, 'data': data, 'n': len(data)}, None


def search_hi_sources():
    """HI回転曲線/面密度のVizieRソースを系統的に探索"""
    import requests

    print('=' * 60)
    print('HI Data Source Search')
    print('=' * 60)

    # 候補カタログ
    candidates = [
        # (catalog_base, description, expected_columns)
        ("J/AJ/136/2648", "THINGS de Blok+2008", "Vrot,Vgas"),
        ("J/AJ/141/193",  "Swaters+2011 WHISP dwarfs", "Vrot,Vgas"),
        ("J/A+A/390/829",  "Swaters+2002 dwarf RC", "Vrot"),
        ("J/MNRAS/376/1513", "Noordermeer+2007 early-type HI", "Vrot"),
        ("J/AJ/149/180",  "Oh+2015 LITTLE THINGS", "Vrot"),
        ("J/A+A/526/A118", "WHISP van der Hulst+2001", "Vrot"),
        ("J/AJ/136/2563",  "Walter+2008 THINGS survey", "Vrot"),
        ("J/A+A/582/A90",  "Frank+2016 THINGS HI profiles", "col"),
    ]

    found = {}

    for cat, desc, expected in candidates:
        print(f'¥n  {cat} ({desc}):')

        # まずテーブル一覧を試す
        result, err = vizier_fetch(cat, "all", max_rows=5)

        if result:
            print(f'    OK: {result["n"]} rows, headers={result["headers"][:8]}')

            # V_gasカラムがあるか確認
            has_vgas = any('gas' in h.lower() or 'vhi' in h.lower()
                           for h in result['headers'])
            has_name = any('name' in h.lower() or 'galaxy' in h.lower()
                           for h in result['headers'])
            has_rad = any('rad' in h.lower() or 'r' == h.lower()
                          for h in result['headers'])

            print(f'    V_gas column: {"YES" if has_vgas else "NO"}')
            print(f'    Name column:  {"YES" if has_name else "NO"}')
            print(f'    Radius column: {"YES" if has_rad else "NO"}')
            if has_vgas or has_rad:
```

```
                found[cat] = {
                    'desc': desc,
                    'headers': result['headers'],
                    'has_vgas': has_vgas,
                    'sample_n': result['n'],
                }
            else:
                print(f'    FAILED: {err}')

                # テーブル名を変えて再試行
                for suffix in ['', '/table1', '/table2', '/table3',
                               '/table4', '/table5', '/tablea1', '/stars']:
                    alt = cat + suffix
                    result2, err2 = vizier_fetch(alt, "all", max_rows=3)
                    if result2:
                        print(f'    ALT {alt}: OK, headers={result2["headers"][:6]}')
                        found[alt] = {
                            'desc': desc + f' ({suffix})',
                            'headers': result2['headers'],
                            'sample_n': result2['n'],
                        }
                        break

    print(f'¥n  Found {len(found)} working sources')
    return found


# ============================================================
# Part 2: 既存データの読み込み
# ============================================================
def load_previous_pipeline():
    """前回パイプラインの出力を読み込む"""
    prev_dir = Path("probes_vbar_output")

    print(f'¥n{"="*60}')
    print('Loading Previous Pipeline Data')
    print(f'{"="*60}')

    if not prev_dir.exists():
        print(f'  {prev_dir} not found')
        return None

    files = list(prev_dir.iterdir())
    print(f'  {prev_dir}: {len(files)} files')
    for f in sorted(files):
        print(f'    {f.name}: {f.stat().st_size:,} bytes')

    # TSVファイルを読む
    data = {}
    for f in files:
        if f.suffix in ('.tsv', '.csv', '.json'):
            try:
                content = f.read_text(encoding='utf-8')
                data[f.stem] = content
                print(f'    Loaded: {f.name} ({len(content):,} chars)')
            except:
                pass

    return data


# ============================================================
# Part 3: Ｙ_d最適化エンジン (Part 1-2の結果を統合)
# ============================================================
def freeman_vdisk(R_kpc, h_kpc, I0_Lpc2, Yd):
    """Freeman disk V_disk(R)"""
    y = R_kpc / (2.0 * h_kpc)
    y = np.clip(y, 1e-6, 50)
    b = i0(y)*k0(y) - i1(y)*k1(y)
    S0_SI = Yd * I0_Lpc2 * Msun / (pc**2)
    V2 = 4 * np.pi * G_SI * S0_SI * (h_kpc*kpc) * y**2 * b
    return np.sqrt(np.maximum(V2, 0)) / 1e3


def optimize_Yd_chi2(R_kpc, V_obs, eV, V_disk_unit, V_gas):
    """chi^2最小化でＹ_dを最適化"""
    def chi2(Yd):
        Vbar = np.sqrt(np.maximum(Yd * V_disk_unit**2 + V_gas**2, 0))
        return np.sum(((V_obs - Vbar) / np.maximum(eV, 1.0))**2) / max(len(V_obs)-1, 1)

    try:
        res = minimize_scalar(chi2, bounds=(0.05, 2.0), method='bounded')
        if res.fun &lt; 50:
            return res.x, res.fun
```

```
    except:
        pass
    return None, None


def alpha_fit(gc, gs0):
    x = np.log10(gs0) - np.log10(a0)
    y = np.log10(gc)
    m = np.isfinite(x) &amp;amp; np.isfinite(y)
    x, y = x[m], y[m]
    if len(x) &amp;lt; 5: return None
    sl, ic, r, p, se = linregress(x, y)
    t_stat = (sl - 0.5) / se
    p05 = 2 * tdist.sf(abs(t_stat), df=len(x)-2)
    return {'alpha': sl, 'e_alpha': se, 'p05': p05, 'r': r, 'N': len(x)}


# ============================================================
# Main
# ============================================================
def main():
    print('=' * 60)
    print('PROBES V_bar Pipeline v2 (Local + HI Search)')
    print('=' * 60)

    # Part 1: HI ソース探索
    hi_sources = search_hi_sources()

    # Part 2: 既存データ読み込み
    prev_data = load_previous_pipeline()

    # Part 3: 結果のサマリと次のステップ
    print(f'¥n{"="*60}')
    print('Summary and Next Steps')
    print(f'{"="*60}')

    print(f'¥n  HI sources found: {len(hi_sources)}')
    for cat, info in hi_sources.items():
        print(f'    {cat}: {info["desc"]}')
        print(f'      headers: {info["headers"][:6]}')
        print(f'      V_gas: {"YES" if info.get("has_vgas") else "check manually"}')

    if prev_data:
        print(f'¥n  Previous pipeline data available: {list(prev_data.keys())}')

    # 実行可能なHIソースがあれば、全データ取得+V_bar構築を試みる
    if hi_sources:
        print('¥n  -&gt; HI sources found. Running full pipeline with best source...')
        run_full_pipeline(hi_sources, prev_data)
    else:
        print('¥n  -&gt; No HI sources found via VizieR.')
        print('  Alternative approaches:')
        print('    1. WHISP data: https://www.astron.nl/~whisp/ (manual download)')
        print('    2. THINGS: https://www.mpia.de/THINGS/ (manual download)')
        print('    3. Use SPARC V_gas as reference for cross-matched galaxies')
        print('    4. Estimate V_gas from HI mass (MHI from HyperLeda or ALFALFA)')

    # 結果保存
    summary = {
        'hi_sources': {k: {kk: vv for kk, vv in v.items() if kk != 'headers'}
                       for k, v in hi_sources.items()},
        'prev_data_available': prev_data is not None,
    }
    with open(OUTDIR / 'v2_search_results.json', 'w') as f:
        json.dump(summary, f, indent=2)
    print(f'¥nResults: {OUTDIR / "v2_search_results.json"}')


def run_full_pipeline(hi_sources, prev_data):
    """HIソースが見つかった場合のフルパイプライン"""
    import requests

    # 最も有望なHIソースを選択 (V_gasカラムがあるもの優先)
    best_hi = None
    for cat, info in hi_sources.items():
        if info.get('has_vgas'):
            best_hi = cat
            break
    if best_hi is None and hi_sources:
        best_hi = list(hi_sources.keys())[0]

    if best_hi is None:
        print('  No usable HI source.')
        return
```

```
    print(f'¥n  Using HI source: {best_hi} ({hi_sources[best_hi]["desc"]})')

    # HI全データ取得
    result, err = vizier_fetch(best_hi, "all", max_rows=999999)
    if not result:
        print(f'  Failed to fetch HI data: {err}')
        return

    print(f'  HI data: {result["n"]} rows')
    print(f'  Headers: {result["headers"]}')

    # ヘッダ解析: Name/Galaxy, Rad/R, Vgas/VHI カラムを特定
    headers = result['headers']
    name_col = next((h for h in headers if h.lower() in ['name','galaxy','gal']), None)
    rad_col = next((h for h in headers if h.lower() in ['rad','r','radius']), None)
    vgas_col = next((h for h in headers if 'gas' in h.lower() or 'vhi' in h.lower()), None)
    vobs_col = next((h for h in headers if 'vrot' in h.lower() or 'vobs' in h.lower()), None)

    print(f'  Identified: name={name_col}, rad={rad_col}, vgas={vgas_col}, vobs={vobs_col}')

    if not name_col or not rad_col:
        print('  Cannot identify required columns.')
        return

    # HI RC をパース
    hi_rc = {}
    for row in result['data']:
        name = row.get(name_col, '').strip()
        if not name:
            continue
        try:
            rad = float(row.get(rad_col, '0'))
            vg = float(row.get(vgas_col, '0')) if vgas_col and row.get(vgas_col, '').strip() else None
            vo = float(row.get(vobs_col, '0')) if vobs_col and row.get(vobs_col, '').strip() else None

            if name not in hi_rc:
                hi_rc[name] = {'R': [], 'Vgas': [], 'Vobs': []}
            hi_rc[name]['R'].append(rad)
            if vg is not None:
                hi_rc[name]['Vgas'].append(vg)
            if vo is not None:
                hi_rc[name]['Vobs'].append(vo)
        except ValueError:
            pass

    for name in hi_rc:
        for k in hi_rc[name]:
            hi_rc[name][k] = np.array(hi_rc[name][k]) if hi_rc[name][k] else np.array([])

    n_with_vgas = sum(1 for v in hi_rc.values() if len(v['Vgas']) &gt;= 3)
    print(f'  HI galaxies: {len(hi_rc)}, with V_gas profile: {n_with_vgas}')

    # TODO: PROBES RC との クロスマッチ + S4G V_disk + Υ_d最適化
    # これはPROBES/S4Gデータの取得方法に依存する
    # 前回パイプラインのローカルデータがあればそこから読む

    print(f'¥n  HI galaxies (sample): {list(hi_rc.keys())[:10]}')
    print(f'¥n  -&gt; Next: Cross-match with PROBES RC and S4G, then run Yd optimization.')


if __name__ == '__main__':
    main()
```

cluster_stack_v3_fix2.py

章: 付録(弱レンズ)

目的: BCG同定(z_spec必須) + cl1 3中心シェアプロファイル比較

解析対象データ: SDSS DR17, HSC-SSP cl1_sources_photoz.csv

主要結果: 元中心(140.45,-0.25): S/N=8.0, |gx|=0.014 <- 最良。BCG候補はS/N=7.1-7.5で劣る。元中心の正当性確認。

ソースコード (376 lines, 12,682 bytes)

```
#!/usr/bin/env python3
"""
cluster_stack_v3_fix2.py
========================
cl1_sources_photoz.csv を正しく読み込み、
元中心 vs BCG中心 vs BCG候補#3 のシェアプロファイルを比較する。

修正点:
  1. ファイル選択: 'sources_photoz' を優先
  2. HSCカラム: i_hsmshaperegauss_e1/e2, derived_weight, photoz_best
  3. '#' 付きヘッダ対応
  4. BCG候補複数での比較

実行: uv run --with scipy --with matplotlib python cluster_stack_v3_fix2.py
"""

import numpy as np
import csv
import json
from pathlib import Path
from scipy.integrate import quad

OUTDIR = Path("cluster_stack_v3_output")
OUTDIR.mkdir(exist_ok=True)

c_light = 2.998e5; H0 = 70.0; Omega_m = 0.3; Omega_L = 0.7

# cl1の中心候補
CENTERS = {
    'original':   {'ra': 140.45,   'dec': -0.25,    'label': 'Original (140.45, -0.25)'},
    'bcg_top1':   {'ra': None,     'dec': None,     'label': 'BCG #1 (from SDSS)'},
    'bcg_top3':   {'ra': None,     'dec': None,     'label': 'BCG #3 (offset=1.16¥')'},
}
Z_CL = 0.313


def D_A(z):
    f = lambda zp: 1.0 / np.sqrt(Omega_m*(1+zp)**3 + Omega_L)
    chi, _ = quad(f, 0, z)
    return c_light / H0 * chi / (1+z)


def find_sources_file():
    """cl1_sources_photoz.csv を探す"""
    candidates = []
    for d in [Path('.'), Path('..'), OUTDIR, Path('cluster_stack_output')]:
        if not d.exists():
            continue
        for f in d.rglob('*sources_photoz*.csv'):
            candidates.append(f)
        for f in d.rglob('*sources*.csv'):
            if 'individual' not in f.name.lower():
                candidates.append(f)

    # サイズ順（大きいほうが生データ）
    candidates = list(set(candidates))
    candidates.sort(key=lambda f: f.stat().st_size, reverse=True)

    print(f'Source file candidates:')
    for f in candidates[:5]:
        print(f'  {f}: {f.stat().st_size/1e6:.1f} MB')

    return candidates[0] if candidates else None


def load_sources(filepath):
    """HSC弱レンズソースカタログ読み込み"""
    print(f'¥nLoading: {filepath}')
    print(f'  Size: {filepath.stat().st_size/1e6:.1f} MB')
    # ヘッダ確認（#付き対応）
```

```
with open(filepath, 'r', encoding='utf-8-sig') as f:
    first_line = f.readline().strip()

# '#' を除去してヘッダ解析
if first_line.startswith('#'):
    first_line = first_line[1:].strip()

raw_headers = [h.strip() for h in first_line.split(',')]
print(f'  Raw headers ({len(raw_headers)}): {raw_headers[:10]}...')

# カラム検出
col_map = {}
for h in raw_headers:
    hl = h.lower()
    if hl in ('i_ra', 'ra'): col_map['ra'] = h
    elif hl in ('i_dec', 'dec'): col_map['dec'] = h
    elif 'e1' in hl and 'hsm' in hl: col_map['e1'] = h
    elif 'e2' in hl and 'hsm' in hl: col_map['e2'] = h
    elif 'e1' in hl and 'e1' not in col_map: col_map.setdefault('e1', h)
    elif 'e2' in hl and 'e2' not in col_map: col_map.setdefault('e2', h)
    elif 'derived_shape_weight' in hl or hl == 'weight': col_map['w'] = h
    elif 'derived_weight' in hl: col_map.setdefault('w', h)
    elif hl in ('photoz_best', 'photoz_mean', 'photo_z', 'z_phot'):
        col_map['zph'] = h
    elif 'photoz' in hl and 'zph' not in col_map:
        col_map.setdefault('zph', h)

print(f'  Column map: {col_map}')

missing = [k for k in ['ra', 'dec', 'e1', 'e2'] if k not in col_map]
if missing:
    print(f'  ERROR: Missing columns: {missing}')
    print(f'  All headers: {raw_headers}')
    return None

# データ読み込み
data = []
n_total = 0
n_bad = 0

with open(filepath, 'r', encoding='utf-8-sig') as f:
    # '#'付きヘッダをスキップ
    line = f.readline()
    if line.startswith('#'):
        # ヘッダ行のカラム名を '#' なしで取得
        clean_headers = [h.strip() for h in line[1:].strip().split(',')]
    else:
        clean_headers = [h.strip() for h in line.strip().split(',')]
        f.seek(0)  # '#'なしなら巻き戻し

    # カラムインデックスを特定
    col_idx = {}
    for key, col_name in col_map.items():
        # clean_headers or raw_headers でインデックス検索
        for i, h in enumerate(clean_headers):
            if h.strip() == col_name.strip():
                col_idx[key] = i
                break
        if key not in col_idx:
            for i, h in enumerate(raw_headers):
                if h.strip() == col_name.strip():
                    col_idx[key] = i
                    break

    print(f'  Column indices: {col_idx}')

    if not all(k in col_idx for k in ['ra', 'dec', 'e1', 'e2']):
        print(f'  ERROR: Could not map columns to indices')
        return None

    # CSVとして再読み込み
    f.seek(0)
    first = f.readline()  # ヘッダスキップ

    for line in f:
        n_total += 1
        cols = line.strip().split(',')
        try:
            ra = float(cols[col_idx['ra']])
            dec = float(cols[col_idx['dec']])
            e1 = float(cols[col_idx['e1']])
            e2 = float(cols[col_idx['e2']])
            w = float(cols[col_idx['w']]) if 'w' in col_idx else 1.0
            zph = float(cols[col_idx['zph']]) if 'zph' in col_idx else 1.0
```

```
                if w &gt; 0 and zph &gt; Z_CL + 0.1:
                    data.append((ra, dec, e1, e2, w, zph))
            except (ValueError, IndexError):
                n_bad += 1

    print(f'  Total rows: {n_total}, Valid bg: {len(data)}, Bad: {n_bad}')
    return data


def shear_profile(sources, ra_c, dec_c, z_cl, label=''):
    """シェアプロファイル計算"""
    D_l = D_A(z_cl)

    R_bins = np.array([0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1.0, 1.5, 2.0, 3.0, 5.0])
    R_mid = np.sqrt(R_bins[:-1] * R_bins[1:])
    n_bins = len(R_mid)

    gt_sum = np.zeros(n_bins)
    gx_sum = np.zeros(n_bins)
    w_sum = np.zeros(n_bins)
    n_count = np.zeros(n_bins, dtype=int)

    for ra_g, dec_g, e1, e2, w, zph in sources:
        dra = (ra_g - ra_c) * np.cos(np.radians(dec_c))
        ddec = dec_g - dec_c
        R_Mpc = np.sqrt(dra**2 + ddec**2) * np.pi / 180 * D_l

        if R_Mpc &lt; R_bins[0] or R_Mpc &gt;= R_bins[-1]:
            continue

        phi = np.arctan2(ddec, dra)
        gt = -(e1 * np.cos(2*phi) + e2 * np.sin(2*phi))
        gx = +(e2 * np.cos(2*phi) - e1 * np.sin(2*phi))

        idx = np.searchsorted(R_bins, R_Mpc) - 1
        if 0 &lt;= idx &lt; n_bins:
            gt_sum[idx] += gt * w
            gx_sum[idx] += gx * w
            w_sum[idx] += w
            n_count[idx] += 1

    mask = w_sum &gt; 0
    gamma_t = np.zeros(n_bins)
    gamma_x = np.zeros(n_bins)
    gamma_t[mask] = gt_sum[mask] / w_sum[mask]
    gamma_x[mask] = gx_sum[mask] / w_sum[mask]

    sigma_e = 0.26
    e_gamma = np.full(n_bins, np.inf)
    e_gamma[mask] = sigma_e / np.sqrt(n_count[mask])

    # 出力
    print(f'¥n  Profile [{label}]:')
    print(f'  {"R[Mpc]":&gt;8} {"N":&gt;7} {"gamma_t":&gt;10} {"gamma_x":&gt;10} {"S/N":&gt;7}')
    for i in range(n_bins):
        sn = gamma_t[i] / e_gamma[i] if e_gamma[i] &lt; np.inf else 0
        print(f'  {R_mid[i]:8.3f} {n_count[i]:7d} {gamma_t[i]:10.6f} '
              f'{gamma_x[i]:10.6f} {sn:7.2f}')

    total_sn = np.sqrt(np.sum(np.where(mask, (gamma_t/e_gamma)**2, 0)))
    gx_med = np.median(np.abs(gamma_x[mask])) if np.any(mask) else 0
    print(f'  Total S/N = {total_sn:.1f}, |gamma_x| median = {gx_med:.6f}')

    return {
        'R': R_mid, 'gt': gamma_t, 'gx': gamma_x, 'eg': e_gamma,
        'n': n_count, 'SN': total_sn, 'gx_med': gx_med,
    }


def main():
    import matplotlib
    matplotlib.use('Agg')
    import matplotlib.pyplot as plt

    print('=' * 60)
    print('cl1 Shear Profile: Multi-Center Comparison')
    print('=' * 60)

    # ファイル検索
    src_file = find_sources_file()
    if src_file is None:
        print('ERROR: No source file found.')
        print('Expected: cl1_sources_photoz.csv')
        return
```

```
# データ読み込み
sources = load_sources(src_file)
if sources is None:
    return

# BCG座標の手動入力 (v3_fixの結果から)
# これらは前回のSDSS結果から。実際の値に置き換えてください。
CENTERS['bcg_top1']['ra'] = 140.45  # TODO: 前回の結果を入力
CENTERS['bcg_top1']['dec'] = -0.25  # TODO: 前回の結果を入力

# 前回の結果がない場合: SDSSから再取得
try:
    import requests
    sql = """
    SELECT TOP 5 p.ra, p.dec, p.r, s.z as z_spec
    FROM PhotoObj AS p
    JOIN SpecObj AS s ON p.objID = s.bestObjID
    WHERE p.ra BETWEEN 140.3 AND 140.6
      AND p.dec BETWEEN -0.35 AND -0.15
      AND p.type = 3
      AND s.z BETWEEN 0.298 AND 0.328
      AND s.zWarning = 0
      AND p.r &lt; 20
    ORDER BY p.r ASC
    """
    r = requests.get(
        "https://skyserver.sdss.org/dr17/SkyServerWS/SearchTools/SqlSearch",
        params={'cmd': sql, 'format': 'json'}, timeout=30)
    data = r.json()
    rows = []
    if isinstance(data, list):
        for item in data:
            if isinstance(item, dict) and 'Rows' in item:
                rows = item['Rows']
                break
            elif isinstance(item, dict) and 'ra' in item:
                rows = data
                break

    if rows:
        print(f'¥nSDSS BCG candidates for cl1:')
        for i, row in enumerate(rows):
            off = np.sqrt(
                ((float(row['ra'])-140.45)*np.cos(np.radians(-0.25))*60)**2 +
                ((float(row['dec'])+0.25)*60)**2)
            print(f'  #{i+1}: RA={float(row["ra"]):.5f} Dec={float(row["dec"]):.5f} '
                  f'r={float(row["r"]):.2f} z={float(row["z_spec"]):.4f} '
                  f'offset={off:.2f}¥'')

        CENTERS['bcg_top1']['ra'] = float(rows[0]['ra'])
        CENTERS['bcg_top1']['dec'] = float(rows[0]['dec'])
        if len(rows) &gt;= 3:
            CENTERS['bcg_top3']['ra'] = float(rows[2]['ra'])
            CENTERS['bcg_top3']['dec'] = float(rows[2]['dec'])
            CENTERS['bcg_top3']['label'] = f'BCG #3 ({float(rows[2]["ra"]):.4f}, {float(rows[2]["dec"]):.4f})'
except Exception as e:
    print(f'SDSS query failed: {e}')

# プロファイル計算: 各中心
profiles = {}
for key, center in CENTERS.items():
    if center['ra'] is None:
        continue
    print(f'¥n{"="*60}')
    print(f'Center: {center["label"]}')
    print(f'  RA={center["ra"]:.5f}, Dec={center["dec"]:.5f}')
    print(f'{"="*60}')
    prof = shear_profile(sources, center['ra'], center['dec'], Z_CL, center['label'])
    profiles[key] = prof

# 比較サマリ
print(f'¥n{"="*60}')
print('Summary: S/N and |gamma_x| by center')
print(f'{"="*60}')
print(f'{"Center":&lt;35} {"S/N":&gt;7} {"|gx|":&gt;10} {"Best?":&gt;6}')
best_sn = max(p['SN'] for p in profiles.values())
for key, prof in profiles.items():
    is_best = 'YES' if prof['SN'] == best_sn else ''
    print(f'{CENTERS[key]["label"]:&lt;35} {prof["SN"]:7.1f} {prof["gx_med"]:10.6f} {is_best:&gt;6}')

# 図の生成
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

colors = {'original': '#1a1a2e', 'bcg_top1': '#e94560', 'bcg_top3': '#2ecc71'}
```

```
    # (a) gamma_t profiles
    ax = axes[0]
    for key, prof in profiles.items():
        mask = prof['n'] &gt; 0
        c = colors.get(key, 'grey')
        ax.errorbar(prof['R'][mask], prof['gt'][mask], yerr=prof['eg'][mask],
                    fmt='o-', color=c, capsize=3, markersize=5,
                    label=f'{CENTERS[key]["label"]} (S/N={prof["SN"]:.1f})')
    ax.axhline(0, color='grey', ls='--', alpha=0.3)
    ax.set_xlabel('R [Mpc]')
    ax.set_ylabel('gamma_t')
    ax.set_xscale('log')
    ax.set_title('(a) Tangential shear profile')
    ax.legend(fontsize=7)
    ax.grid(True, alpha=0.3)

    # (b) gamma_x profiles (null test)
    ax = axes[1]
    for key, prof in profiles.items():
        mask = prof['n'] &gt; 0
        c = colors.get(key, 'grey')
        ax.errorbar(prof['R'][mask], prof['gx'][mask], yerr=prof['eg'][mask],
                    fmt='o-', color=c, capsize=3, markersize=5,
                    label=CENTERS[key]['label'])
    ax.axhline(0, color='grey', ls='--', alpha=0.3)
    ax.set_xlabel('R [Mpc]')
    ax.set_ylabel('gamma_x (null test)')
    ax.set_xscale('log')
    ax.set_title('(b) Cross shear (should be ~0)')
    ax.legend(fontsize=7)
    ax.grid(True, alpha=0.3)

    plt.suptitle('cl1 Weak Lensing: Original vs BCG Center', fontsize=13)
    plt.tight_layout()
    fig_path = OUTDIR / 'cl1_center_comparison.png'
    plt.savefig(fig_path, dpi=150)
    print(f'¥nFigure: {fig_path}')

    # JSON保存
    summary = {k: {'SN': p['SN'], 'gx_med': p['gx_med'],
                   'gt_max': float(np.max(p['gt'])),
                   'center': CENTERS[k]}
               for k, p in profiles.items()}
    with open(OUTDIR / 'center_comparison.json', 'w') as f:
        json.dump(summary, f, indent=2, default=str)


if __name__ == '__main__':
    main()
```

cl1_model_fit.py

章: 付録(弱レンズ)

目的: cl1単体: NFW/NFW+2halo/Gaussianリング 3モデルフィット

解析対象データ: cl1_individual_shear.csv (ビニング済み)

主要結果: NFW: chi2/dof=1.47(logM=16,c=1境界)。Gaussianリング: DAIC=+20.5(棄却)。NFW+2halo: DAIC=+2.0(検出不可)。

ソースコード (471 lines, 15,640 bytes)

```
#!/usr/bin/env python3
"""
cl1_model_fit.py
================
cl1の弱レンズシェアプロファイルに3モデルをフィットしてAIC比較。

モデル:
  A) NFW (2パラメータ: M200, c200)
  B) NFW + 2-halo (3パラメータ: M200, c200, b_lin)
  C) Membrane delta-layer (3パラメータ: M200, f_delta, r_delta)

入力: cluster_stack_v3_fix2.py で計算されたプロファイル
      または cl1_individual_shear.csv (既存ビニング済み)

実行: uv run --with scipy --with matplotlib python cl1_model_fit.py
"""

import numpy as np
import json
from pathlib import Path
from scipy.optimize import minimize, differential_evolution
from scipy.integrate import quad
from scipy.special import hyp2f1

import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib import font_manager as _fm
for _fp in ['/usr/share/fonts/opentype/ipafont-gothic/ipag.ttf',
            '/usr/share/fonts/opentype/ipafont-gothic/ipagp.ttf']:
    try: _fm.fontManager.addfont(_fp)
    except: pass
plt.rcParams['font.family'] = 'IPAGothic'
plt.rcParams['axes.unicode_minus'] = False

OUTDIR = Path("cluster_stack_v3_output")
OUTDIR.mkdir(exist_ok=True)

# 宇宙論
c_light = 2.998e5; H0 = 70.0; Omega_m = 0.3; Omega_L = 0.7
G_N = 4.301e-3  # pc Msun^-1 (km/s)^2

Z_CL = 0.313

def E(z):
    return np.sqrt(Omega_m*(1+z)**3 + Omega_L)

def D_A(z):
    f = lambda zp: 1.0 / E(zp)
    chi, _ = quad(f, 0, z)
    return c_light / H0 * chi / (1+z)  # Mpc

def rho_crit(z):
    """Critical density [Msun/Mpc^3]"""
    Hz = H0 * E(z)  # km/s/Mpc
    return 3 * Hz**2 / (8 * np.pi * G_N * 1e-6)  # convert

# 正確な rho_crit: 3H^2/(8piG)
# H0 = 70 km/s/Mpc -&gt; rho_crit = 9.47e10 Msun/Mpc^3
RHO_CRIT_0 = 1.36e11  # Msun/Mpc^3 (H0=70)


# ============================================================
# NFW プロファイル
# ============================================================
def nfw_Sigma(R, M200, c, z):
    """NFW projected surface mass density [Msun/Mpc^2]"""
    rho_c = RHO_CRIT_0 * E(z)**2
    r200 = (3*M200 / (4*np.pi*200*rho_c))**(1./3)
    rs = r200 / c
```

```
    rho_s = M200 / (4*np.pi*rs**3 * (np.log(1+c) - c/(1+c)))

    x = np.atleast_1d(R / rs).astype(float)
    result = np.zeros_like(x)

    lt1 = (x &amp;lt; 1) &amp;amp; (x &amp;gt; 0)
    gt1 = x &amp;gt; 1
    eq1 = np.abs(x - 1) &amp;lt; 1e-4

    if np.any(lt1):
        t = np.sqrt(1 - x[lt1]**2)
        result[lt1] = 1/(x[lt1]**2 - 1) * (1 - np.arctanh(t)/t)
    if np.any(gt1):
        t = np.sqrt(x[gt1]**2 - 1)
        result[gt1] = 1/(x[gt1]**2 - 1) * (1 - np.arctan(t)/t)
    if np.any(eq1):
        result[eq1] = 1./3

    return 2 * rs * rho_s * result  # Msun/Mpc^2


def nfw_DeltaSigma(R, M200, c, z):
    """NFW excess surface mass density [Msun/Mpc^2]"""
    R = np.atleast_1d(R)
    Sigma_R = nfw_Sigma(R, M200, c, z)

    # Mean Sigma(&amp;lt;R) by integration
    Sigma_mean = np.zeros_like(R)
    for i, Ri in enumerate(R):
        r_int = np.linspace(0.001*Ri, Ri, 500)
        S_int = nfw_Sigma(r_int, M200, c, z)
        Sigma_mean[i] = 2 * np.trapz(r_int * S_int, r_int) / Ri**2

    return Sigma_mean - Sigma_R


# ============================================================
# 2-halo項
# ============================================================
def two_halo_DeltaSigma(R, M200, b, z):
    """
    Simplified 2-halo term.
    DeltaSigma_2h ~ b * rho_m_bar * Sigma_lin(R)

    Using power-law matter correlation: xi(r) ~ (r/r0)^-gamma
    r0 = 5 h^-1 Mpc, gamma = 1.8

    Projected: Sigma_2h(R) ~ rho_m * r0 * (R/r0)^(1-gamma) * B(0.5, (gamma-1)/2)
    """
    rho_m = RHO_CRIT_0 * Omega_m * (1+z)**3
    r0 = 5.0 / 0.7  # Mpc (h=0.7)
    gamma = 1.8

    # Sigma_lin ~ integral of xi along los
    # Approximate: Sigma(R) ~ rho_m * r0 * (R/r0)^(1-gamma) * pi * Gamma((gamma-1)/2) / Gamma(gamma/2)
    from scipy.special import gamma as Gamma
    prefactor = rho_m * r0 * np.sqrt(np.pi) * Gamma((gamma-1)/2) / Gamma(gamma/2)
    Sigma_2h = b * prefactor * (R/r0)**(1-gamma)

    # DeltaSigma_2h = Sigma_mean_2h(&amp;lt;R) - Sigma_2h(R)
    # For power-law: Sigma_mean(&amp;lt;R) = 2/(2+(1-gamma)) * Sigma(R) * (3-gamma)/(3-gamma) ...
    # Simplified: DeltaSigma ~ (gamma-1)/(3-gamma) * Sigma
    DS_2h = (gamma - 1) / (3 - gamma) * Sigma_2h

    return DS_2h


# ============================================================
# 膜宇宙論 delta-layer モデル
# ============================================================
def membrane_DeltaSigma(R, M200, f_delta, r_delta, z):
    """
    NFW + delta-layer (rs近傍の集中質量殻)

    物理的解釈: 膜の折り畳み構造による追加的な重力レンズ効果
    Dark Matter P (Psi_0) がrs近傍で局在
    """
    c = 5.0  # typical concentration
    DS_nfw = nfw_DeltaSigma(R, M200 * (1 - f_delta), c, z)

    # Delta layer: Gaussian ring at r_delta with width sigma_ring
    sigma_ring = 0.05  # Mpc (50 kpc width)
    M_ring = f_delta * M200
    # Ring surface density
```

```
    Sigma_ring = M_ring / (2*np.pi*r_delta*sigma_ring*np.sqrt(2*np.pi)) * ¥
                 np.exp(-0.5*((R - r_delta)/sigma_ring)**2)

    # Mean Sigma_ring(&lt;R)
    Sigma_ring_mean = np.zeros_like(R)
    for i, Ri in enumerate(R):
        r_int = np.linspace(0.001, Ri, 300)
        S_int = M_ring / (2*np.pi*r_delta*sigma_ring*np.sqrt(2*np.pi)) * ¥
                np.exp(-0.5*((r_int - r_delta)/sigma_ring)**2)
        Sigma_ring_mean[i] = 2*np.trapz(r_int*S_int, r_int) / Ri**2

    DS_ring = Sigma_ring_mean - Sigma_ring

    return DS_nfw + DS_ring


# ============================================================
# プロファイル読み込み
# ============================================================
def load_profile():
    """プロファイルデータの読み込み"""
    # 1. v3_fix2の出力 (JSON)
    json_file = OUTDIR / 'center_comparison.json'
    if json_file.exists():
        with open(json_file) as f:
            data = json.load(f)
        if 'original' in data:
            # JSONには要約のみ。フルプロファイルは別途
            pass

    # 2. cl1_individual_shear.csv (既存ビニング済み)
    for d in [Path('.'), Path('..'), OUTDIR, Path('cluster_stack_output')]:
        if not d.exists():
            continue
        for f in d.glob('*individual*shear*.csv'):
            print(f'Found binned profile: {f}')
            return load_binned_csv(f)

    # 3. ハードコードの代替値 (前回v2結果から)
    print('Using hardcoded cl1 profile from v2 analysis')
    return hardcoded_cl1_profile()


def load_binned_csv(filepath):
    """ビニング済みCSVを読み込む"""
    import csv
    print(f'Loading: {filepath}')

    with open(filepath, 'r', encoding='utf-8-sig') as f:
        first = f.readline()
        print(f'  Header: {first.strip()[:100]}')

    # カラム検出
    with open(filepath, 'r', encoding='utf-8-sig') as f:
        line = f.readline().strip()
        if line.startswith('#'):
            line = line[1:].strip()
        headers = [h.strip() for h in line.split(',')]
        print(f'  Columns: {headers}')

        R, gt, gx, eg, N = [], [], [], [], []
        for row_line in f:
            cols = row_line.strip().split(',')
            try:
                # R: r_kpc or r_Mpc or r_arcmin
                r_col = None
                for i, h in enumerate(headers):
                    if 'kpc' in h.lower():
                        r_col = i; r_unit = 'kpc'; break
                    elif 'mpc' in h.lower():
                        r_col = i; r_unit = 'Mpc'; break
                    elif 'arcmin' in h.lower():
                        r_col = i; r_unit = 'arcmin'; break
                    elif h.lower() in ('r', 'rad', 'radius'):
                        r_col = i; r_unit = 'unknown'; break

                if r_col is None:
                    r_col = 0; r_unit = 'unknown'

                r_val = float(cols[r_col])

                # Convert to Mpc
                if r_unit == 'kpc':
                    r_Mpc = r_val / 1000
```

```
                elif r_unit == 'arcmin':
                    r_Mpc = r_val / 60 * np.pi / 180 * D_A(Z_CL)
                else:
                    r_Mpc = r_val  # assume Mpc

                # gamma_t
                gt_col = next((i for i, h in enumerate(headers) if 'gamma_t' in h.lower() or 'gt' == h.lower()), 1)
                gt_val = float(cols[gt_col])

                # error
                eg_col = next((i for i, h in enumerate(headers) if 'err' in h.lower() or 'e_g' in h.lower()), None)
                eg_val = float(cols[eg_col]) if eg_col else 0.01

                R.append(r_Mpc)
                gt.append(gt_val)
                eg.append(eg_val)
            except (ValueError, IndexError):
                pass

    print(f'  {len(R)} bins loaded')
    return np.array(R), np.array(gt), np.array(eg)


def hardcoded_cl1_profile():
    """前回解析のcl1プロファイル（代替値）"""
    # v2解析からの典型的なプロファイル（元中心, S/N~8）
    R_Mpc = np.array([0.07, 0.14, 0.24, 0.39, 0.59, 0.84, 1.22, 1.73, 2.45])
    gamma_t = np.array([0.035, 0.025, 0.018, 0.012, 0.008, 0.005, 0.003, 0.002, 0.001])
    e_gamma = np.array([0.012, 0.006, 0.004, 0.003, 0.002, 0.002, 0.001, 0.001, 0.001])
    return R_Mpc, gamma_t, e_gamma


# ============================================================
# フィッティング
# ============================================================
def fit_nfw(R, gt, eg):
    """NFW fit (M200, c200)"""
    # DeltaSigma -&gt; gamma_t 変換: gamma_t ~ DeltaSigma / Sigma_crit
    # Sigma_crit is unknown -&gt; fit amplitude
    # gamma_t = A * DeltaSigma_NFW(R; M200, c) where A = 1/Sigma_crit

    def model(params):
        logM, c, logA = params
        M200 = 10**logM
        A = 10**logA
        DS = nfw_DeltaSigma(R, M200, c, Z_CL)
        pred = A * DS
        chi2 = np.sum(((gt - pred) / eg)**2)
        return chi2

    try:
        result = differential_evolution(model,
            bounds=[(13, 16), (1, 15), (-18, -12)],
            seed=42, maxiter=500, tol=1e-6)
        logM, c, logA = result.x
        chi2 = result.fun
        dof = len(R) - 3

        DS_best = nfw_DeltaSigma(R, 10**logM, c, Z_CL)
        pred_best = 10**logA * DS_best

        return {
            'name': 'NFW',
            'logM200': logM, 'c200': c, 'logA': logA,
            'chi2': chi2, 'dof': dof, 'chi2_dof': chi2/max(dof,1),
            'AIC': chi2 + 2*3,
            'pred': pred_best,
            'n_param': 3,
        }
    except Exception as e:
        print(f'  NFW fit failed: {e}')
        return None


def fit_nfw_2halo(R, gt, eg):
    """NFW + 2-halo fit"""
    def model(params):
        logM, c, logA, b = params
        M200 = 10**logM
        A = 10**logA
        DS_1h = nfw_DeltaSigma(R, M200, c, Z_CL)
        DS_2h = two_halo_DeltaSigma(R, M200, b, Z_CL)
        pred = A * (DS_1h + DS_2h)
        return np.sum(((gt - pred) / eg)**2)
```

```
    try:
        result = differential_evolution(model,
            bounds=[(13, 16), (1, 15), (-18, -12), (0.1, 10)],
            seed=42, maxiter=500, tol=1e-6)
        logM, c, logA, b = result.x
        chi2 = result.fun
        dof = len(R) - 4

        DS_1h = nfw_DeltaSigma(R, 10**logM, c, Z_CL)
        DS_2h = two_halo_DeltaSigma(R, 10**logM, b, Z_CL)
        pred = 10**logA * (DS_1h + DS_2h)

        return {
            'name': 'NFW+2halo',
            'logM200': logM, 'c200': c, 'logA': logA, 'b_lin': b,
            'chi2': chi2, 'dof': dof, 'chi2_dof': chi2/max(dof,1),
            'AIC': chi2 + 2*4,
            'pred': pred,
            'n_param': 4,
        }
    except Exception as e:
        print(f'  NFW+2halo fit failed: {e}')
        return None


def fit_membrane(R, gt, eg):
    """Membrane delta-layer fit"""
    def model(params):
        logM, f_d, r_d, logA = params
        M200 = 10**logM
        A = 10**logA
        DS = membrane_DeltaSigma(R, M200, f_d, r_d, Z_CL)
        pred = A * DS
        return np.sum(((gt - pred) / eg)**2)

    try:
        result = differential_evolution(model,
            bounds=[(13, 16), (0.01, 0.5), (0.05, 1.0), (-18, -12)],
            seed=42, maxiter=500, tol=1e-6)
        logM, f_d, r_d, logA = result.x
        chi2 = result.fun
        dof = len(R) - 4

        DS = membrane_DeltaSigma(R, 10**logM, f_d, r_d, Z_CL)
        pred = 10**logA * DS

        return {
            'name': 'Membrane',
            'logM200': logM, 'f_delta': f_d, 'r_delta': r_d, 'logA': logA,
            'chi2': chi2, 'dof': dof, 'chi2_dof': chi2/max(dof,1),
            'AIC': chi2 + 2*4,
            'pred': pred,
            'n_param': 4,
        }
    except Exception as e:
        print(f'  Membrane fit failed: {e}')
        return None


# ================================================================
# Main
# ================================================================
def main():
    print('=' * 60)
    print('cl1 Model Fitting: NFW vs NFW+2halo vs Membrane')
    print('=' * 60)

    R, gt, eg = load_profile()

    # フィルタ: 有効なビンのみ
    mask = (gt != 0) &amp;amp; (eg &amp;gt; 0) &amp;amp; (eg &amp;lt; 1) &amp;amp; (R &amp;gt; 0)
    R, gt, eg = R[mask], gt[mask], eg[mask]
    print(f'¥nValid bins: {len(R)}')
    print(f'R range: [{R.min():.3f}, {R.max():.3f}] Mpc')

    # フィット
    print(f'¥n--- Fitting ---')
    fits = {}

    print('¥n  Model A: NFW')
    f_nfw = fit_nfw(R, gt, eg)
    if f_nfw:
        fits['NFW'] = f_nfw
        print(f'    logM={f_nfw["logM200"]:.2f}, c={f_nfw["c200"]:.1f}, '
```

```
            f'chi2/dof={f_nfw["chi2_dof"]:.2f}')

    print('¥n  Model B: NFW + 2-halo')
    f_2h = fit_nfw_2halo(R, gt, eg)
    if f_2h:
        fits['NFW+2halo'] = f_2h
        print(f'    logM={f_2h["logM200"]:.2f}, c={f_2h["c200"]:.1f}, '
              f'b={f_2h["b_lin"]:.2f}, chi2/dof={f_2h["chi2_dof"]:.2f}')

    print('¥n  Model C: Membrane delta-layer')
    f_mem = fit_membrane(R, gt, eg)
    if f_mem:
        fits['Membrane'] = f_mem
        print(f'    logM={f_mem["logM200"]:.2f}, f_d={f_mem["f_delta"]:.3f}, '
              f'r_d={f_mem["r_delta"]:.3f}, chi2/dof={f_mem["chi2_dof"]:.2f}')

    # AIC比較
    if fits:
        aic_min = min(f['AIC'] for f in fits.values())
        print(f'¥n--- AIC Comparison ---')
        print(f'{"Model":&lt;15} {"chi2":&gt;7} {"dof":&gt;4} {"chi2/dof":&gt;9} {"AIC":&gt;7} {"DAIC":&gt;7} {"Best?":&gt;6}')
        for name, f in sorted(fits.items(), key=lambda x: x[1]['AIC']):
            daic = f['AIC'] - aic_min
            best = '&lt;--' if daic == 0 else ''
            print(f'{name:&lt;15} {f["chi2"]:7.2f} {f["dof"]:4d} {f["chi2_dof"]:9.2f} '
                  f'{f["AIC"]:7.2f} {daic:7.2f} {best:&gt;6}')

    # 図
    fig, ax = plt.subplots(figsize=(10, 7))

    ax.errorbar(R, gt, yerr=eg, fmt='ko', capsize=3, markersize=6, label='cl1 data')

    R_fine = np.linspace(R.min()*0.8, R.max()*1.2, 200)
    colors_m = {'NFW': '#1a1a2e', 'NFW+2halo': '#e94560', 'Membrane': '#2ecc71'}

    for name, f in fits.items():
        ax.plot(R, f['pred'], '-', color=colors_m.get(name, 'grey'), lw=2,
                label=f'{name} (chi2/dof={f["chi2_dof"]:.2f}, DAIC={f["AIC"]-aic_min:.1f})')

    ax.set_xlabel('R [Mpc]', fontsize=12)
    ax.set_ylabel('gamma_t', fontsize=12)
    ax.set_xscale('log')
    ax.set_title('cl1 Weak Lensing: Model Comparison', fontsize=13)
    ax.legend(fontsize=9)
    ax.grid(True, alpha=0.3)
    ax.axhline(0, color='grey', ls='--', alpha=0.3)

    plt.tight_layout()
    fig_path = OUTDIR / 'cl1_model_fit.png'
    plt.savefig(fig_path, dpi=150)
    print(f'¥nFigure: {fig_path}')

    # JSON
    summary = {name: {k: v for k, v in f.items() if k != 'pred'}
               for name, f in fits.items()}
    with open(OUTDIR / 'cl1_model_fit.json', 'w') as f:
        json.dump(summary, f, indent=2, default=float)
    print(f'Results: {OUTDIR / "cl1_model_fit.json"}')


if __name__ == '__main__':
    main()
```

cl1_cl3_stack.py

章: 付録(弱レンズ)

目的: cl1+cl3スタック(cl4/cl27除外): Sigma_crit正規化

解析対象データ: cl1/cl3_sources_photoz.csv

主要結果: スタックS/N=17.2。NFW: chi2/dof=24.3。全モデル不可。内側ビン符号反転。cl1単体が最信頼。

ソースコード (513 lines, 17,081 bytes)

```
#!/usr/bin/env python3
"""
cl1_cl3_stack.py
================
cl1 + cl3 のみのスタック弱レンズ解析。
cl4/cl27 はSDSS分光メンバーなし → 除外。

手順:
  1. cl1, cl3 のソースカタログを読み込み
  2. 各クラスターで元中心を使用（BCG探索で元中心が最良と確認済み）
  3. Sigma_crit 正規化でスタック
  4. NFW / NFW+2halo / Abel変換membrane でフィット
  5. AIC比較

実行: uv run --with scipy --with matplotlib python cl1_cl3_stack.py
"""

import numpy as np
import csv
import json
from pathlib import Path
from scipy.optimize import differential_evolution
from scipy.integrate import quad

import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib import font_manager as _fm
for _fp in ['/usr/share/fonts/opentype/ipafont-gothic/ipag.ttf',
            '/usr/share/fonts/opentype/ipafont-gothic/ipagp.ttf']:
    try: _fm.fontManager.addfont(_fp)
    except: pass
plt.rcParams['font.family'] = 'IPAGothic'
plt.rcParams['axes.unicode_minus'] = False

OUTDIR = Path("cluster_stack_v3_output")
OUTDIR.mkdir(exist_ok=True)

c_light = 2.998e5; H0 = 70.0; Omega_m = 0.3; Omega_L = 0.7
RHO_CRIT_0 = 1.36e11  # Msun/Mpc^3

CLUSTERS = {
    'cl1': {'ra': 140.45, 'dec': -0.25, 'z': 0.313},
    'cl3': {'ra': 139.80, 'dec': -0.10, 'z': 0.318},
}


def E(z):
    return np.sqrt(Omega_m*(1+z)**3 + Omega_L)

def D_A(z):
    f = lambda zp: 1.0 / E(zp)
    chi, _ = quad(f, 0, z)
    return c_light / H0 * chi / (1+z)

def Sigma_crit(z_l, z_s):
    """Sigma_crit [Msun/Mpc^2]"""
    D_l = D_A(z_l)
    D_s = D_A(z_s)
    # D_ls for flat LCDM
    f = lambda zp: 1.0 / E(zp)
    chi_ls, _ = quad(f, z_l, z_s)
    D_ls = c_light / H0 * chi_ls / (1+z_s)

    if D_ls &lt;= 0 or D_l &lt;= 0 or D_s &lt;= 0:
        return None

    # Sigma_crit = c^2/(4piG) * D_s/(D_l*D_ls) [Msun/Mpc^2]
    # c^2/(4piG) in Msun/Mpc units
    c2_over_4piG = (c_light * 1e3)**2 / (4 * np.pi * 6.674e-11) / (1.989e30) * (3.086e22)
```

```
    return c2_over_4piG * D_s / (D_l * D_ls)


# ============================================================
# ソースカタログ読み込み
# ============================================================
def find_source_file(cl_name):
    """クラスターのソースCSVファイルを探す"""
    patterns = [
        f'{cl_name}_sources_photoz.csv',
        f'{cl_name}_sources.csv',
        f'{cl_name}_background.csv',
        f'{cl_name}_photoz.csv',
    ]
    for d in [Path('.'), Path('..'), OUTDIR, Path('cluster_stack_output')]:
        if not d.exists():
            continue
        for p in patterns:
            f = d / p
            if f.exists():
                return f
        # glob
        for f in d.glob(f'*{cl_name}*source*photoz*.csv'):
            return f
        for f in d.glob(f'*{cl_name}*source*.csv'):
            if 'individual' not in f.name.lower():
                return f
    return None


def load_sources(filepath, z_cl):
    """HSCソースカタログ読み込み"""
    print(f'  Loading: {filepath} ({filepath.stat().st_size/1e6:.1f} MB)')

    with open(filepath, 'r', encoding='utf-8-sig') as f:
        first = f.readline().strip()

    if first.startswith('#'):
        first = first[1:].strip()
    headers = [h.strip() for h in first.split(',')]

    # カラムインデックス検出
    def find_col(candidates):
        for c in candidates:
            for i, h in enumerate(headers):
                if h.lower() == c.lower() or c.lower() in h.lower():
                    return i
        return None

    i_ra = find_col(['i_ra', 'ra', 'RA'])
    i_dec = find_col(['i_dec', 'dec', 'DEC'])
    i_e1 = find_col(['i_hsmshaperegauss_e1', 'e1', 'hsmshaperegauss_e1'])
    i_e2 = find_col(['i_hsmshaperegauss_e2', 'e2', 'hsmshaperegauss_e2'])
    i_w = find_col(['derived_shape_weight', 'weight', 'derived_weight', 'ishape_hsm_regauss_derived_shape_weight'])
    i_zph = find_col(['photoz_best', 'photoz_mean', 'photo_z', 'pz_best_z', 'mizuki_photoz_best'])

    print(f'    Columns: ra={i_ra}, dec={i_dec}, e1={i_e1}, e2={i_e2}, w={i_w}, zph={i_zph}')

    if i_ra is None or i_dec is None or i_e1 is None or i_e2 is None:
        print(f'    ERROR: Missing columns. Headers: {headers[:15]}')
        return None

    sources = []
    n_total = 0
    with open(filepath, 'r', encoding='utf-8-sig') as f:
        f.readline()  # skip header
        for line in f:
            n_total += 1
            cols = line.strip().split(',')
            try:
                ra = float(cols[i_ra])
                dec = float(cols[i_dec])
                e1 = float(cols[i_e1])
                e2 = float(cols[i_e2])
                w = float(cols[i_w]) if i_w is not None else 1.0
                zph = float(cols[i_zph]) if i_zph is not None else 1.0

                if w &gt; 0 and zph &gt; z_cl + 0.1:
                    sources.append((ra, dec, e1, e2, w, zph))
            except (ValueError, IndexError):
                pass

    print(f'    Total: {n_total}, Background: {len(sources)}')
    return sources
```

```
# ============================================================
# Delta-Sigma プロファイル計算
# ============================================================
def compute_DeltaSigma_profile(sources, ra_c, dec_c, z_cl, R_bins_Mpc):
    """
    物理単位の Delta-Sigma プロファイルを計算。
    gamma_t * Sigma_crit = Delta-Sigma [Msun/Mpc^2]
    """
    D_l = D_A(z_cl)
    n_bins = len(R_bins_Mpc) - 1
    R_mid = np.sqrt(R_bins_Mpc[:-1] * R_bins_Mpc[1:])

    DS_sum = np.zeros(n_bins)
    w_sum = np.zeros(n_bins)
    n_count = np.zeros(n_bins, dtype=int)

    # Cross-component for null test
    DX_sum = np.zeros(n_bins)

    for ra_g, dec_g, e1, e2, w, zph in sources:
        # 角度距離 -&gt; 物理距離
        dra = (ra_g - ra_c) * np.cos(np.radians(dec_c))
        ddec = dec_g - dec_c
        R_Mpc = np.sqrt(dra**2 + ddec**2) * np.pi / 180 * D_l

        if R_Mpc &lt; R_bins_Mpc[0] or R_Mpc &gt;= R_bins_Mpc[-1]:
            continue

        # Sigma_crit
        Sc = Sigma_crit(z_cl, zph)
        if Sc is None or Sc &lt;= 0:
            continue

        # Tangential/Cross shear
        phi = np.arctan2(ddec, dra)
        gt = -(e1 * np.cos(2*phi) + e2 * np.sin(2*phi))
        gx = +(e2 * np.cos(2*phi) - e1 * np.sin(2*phi))

        # Delta-Sigma = gamma_t * Sigma_crit
        DS = gt * Sc
        DX = gx * Sc

        # 重み: w_ls = w / Sc^2 (Sigma_crit optimal weighting)
        w_ls = w / Sc**2

        idx = np.searchsorted(R_bins_Mpc, R_Mpc) - 1
        if 0 &lt;= idx &lt; n_bins:
            DS_sum[idx] += DS * w_ls
            DX_sum[idx] += DX * w_ls
            w_sum[idx] += w_ls
            n_count[idx] += 1

    mask = w_sum &gt; 0
    DeltaSigma = np.zeros(n_bins)
    DeltaSigma_x = np.zeros(n_bins)
    DeltaSigma[mask] = DS_sum[mask] / w_sum[mask]
    DeltaSigma_x[mask] = DX_sum[mask] / w_sum[mask]

    # 誤差: shape noise
    sigma_e = 0.26
    e_DS = np.full(n_bins, np.inf)
    for i in range(n_bins):
        if n_count[i] &gt; 0 and w_sum[i] &gt; 0:
            # Mean Sigma_crit^-2 weighted error
            Sc_eff = np.sqrt(w_sum[i] / n_count[i]) if n_count[i] &gt; 0 else 1
            e_DS[i] = sigma_e / np.sqrt(n_count[i]) * np.sqrt(1.0 / w_sum[i]) * w_sum[i]
            # 簡易版: sigma_e * Sc_mean / sqrt(N)
            e_DS[i] = sigma_e * 1e15 / np.sqrt(n_count[i])  # scale

    return {
        'R': R_mid, 'DS': DeltaSigma, 'DS_x': DeltaSigma_x,
        'e_DS': e_DS, 'n': n_count, 'w': w_sum,
    }


# ============================================================
# スタッキング
# ============================================================
def stack_profiles(profiles):
    """複数プロファイルの加重平均スタック"""
    R = profiles[0]['R']
    n_bins = len(R)

    DS_sum = np.zeros(n_bins)
```

```
    DX_sum = np.zeros(n_bins)
    w_sum = np.zeros(n_bins)
    n_total = np.zeros(n_bins, dtype=int)

    for p in profiles:
        mask = p['w'] &gt; 0
        for i in range(n_bins):
            if mask[i]:
                DS_sum[i] += p['DS'][i] * p['w'][i]
                DX_sum[i] += p['DS_x'][i] * p['w'][i]
                w_sum[i] += p['w'][i]
                n_total[i] += p['n'][i]

    mask = w_sum &gt; 0
    DS_stack = np.zeros(n_bins)
    DX_stack = np.zeros(n_bins)
    DS_stack[mask] = DS_sum[mask] / w_sum[mask]
    DX_stack[mask] = DX_sum[mask] / w_sum[mask]

    sigma_e = 0.26
    e_DS = np.full(n_bins, np.inf)
    e_DS[mask] = sigma_e * 1e15 / np.sqrt(n_total[mask])

    return {
        'R': R, 'DS': DS_stack, 'DS_x': DX_stack,
        'e_DS': e_DS, 'n': n_total, 'w': w_sum,
    }


# ============================================================
# NFW モデル
# ============================================================
def nfw_DeltaSigma(R, M200, c, z):
    rho_c = RHO_CRIT_0 * E(z)**2
    r200 = (3*M200/(4*np.pi*200*rho_c))**(1./3)
    rs = r200/c
    rho_s = M200/(4*np.pi*rs**3*(np.log(1+c)-c/(1+c)))

    x = np.atleast_1d(R/rs).astype(float)
    Sig = np.zeros_like(x)

    lt = (x&gt;0)&amp;(x&lt;1); gt = x&gt;1; eq = np.abs(x-1)&lt;1e-4
    if np.any(lt):
        t = np.sqrt(1-x[lt]**2)
        Sig[lt] = 1/(x[lt]**2-1)*(1-np.arctanh(t)/t)
    if np.any(gt):
        t = np.sqrt(x[gt]**2-1)
        Sig[gt] = 1/(x[gt]**2-1)*(1-np.arctan(t)/t)
    if np.any(eq):
        Sig[eq] = 1./3

    Sigma = 2*rs*rho_s*Sig

    Sigma_mean = np.zeros_like(R)
    for i, Ri in enumerate(R):
        r_int = np.linspace(0.001*Ri, Ri, 300)
        x_int = r_int/rs
        S_int = np.zeros_like(x_int)
        lt_i = (x_int&gt;0)&amp;(x_int&lt;1); gt_i = x_int&gt;1; eq_i = np.abs(x_int-1)&lt;1e-4
        if np.any(lt_i):
            t_i = np.sqrt(1-x_int[lt_i]**2)
            S_int[lt_i] = 1/(x_int[lt_i]**2-1)*(1-np.arctanh(t_i)/t_i)
        if np.any(gt_i):
            t_i = np.sqrt(x_int[gt_i]**2-1)
            S_int[gt_i] = 1/(x_int[gt_i]**2-1)*(1-np.arctan(t_i)/t_i)
        if np.any(eq_i):
            S_int[eq_i] = 1./3
        S_int *= 2*rs*rho_s
        Sigma_mean[i] = 2*np.trapz(r_int*S_int, r_int)/Ri**2

    return Sigma_mean - Sigma


def two_halo_DS(R, b, z):
    from scipy.special import gamma as Gamma
    rho_m = RHO_CRIT_0*Omega_m*(1+z)**3
    r0 = 7.0; gam = 1.8
    pf = rho_m*r0*np.sqrt(np.pi)*Gamma((gam-1)/2)/Gamma(gam/2)
    S2h = b*pf*(R/r0)**(1-gam)
    return (gam-1)/(3-gam)*S2h


# ============================================================
# フィット
```

```
# ============================================================
def fit_all(R, DS, eDS, z_stack):
    """3モデルフィット"""
    mask = np.isfinite(DS) &amp;amp; np.isfinite(eDS) &amp;amp; (eDS &amp;gt; 0) &amp;amp; (eDS &amp;lt; 1e20)
    R_f, DS_f, eDS_f = R[mask], DS[mask], eDS[mask]

    if len(R_f) &amp;lt; 4:
        print('  Too few valid bins')
        return {}

    results = {}

    # NFW
    def chi2_nfw(p):
        logM, c = p
        pred = nfw_DeltaSigma(R_f, 10**logM, c, z_stack)
        return np.sum(((DS_f - pred)/eDS_f)**2)

    try:
        res = differential_evolution(chi2_nfw, [(12,16),(1,20)], seed=42, maxiter=1000)
        logM, c = res.x; chi2 = res.fun; dof = len(R_f)-2
        pred = nfw_DeltaSigma(R_f, 10**logM, c, z_stack)
        results['NFW'] = {
            'logM': logM, 'c': c,
            'chi2': chi2, 'dof': dof, 'chi2_dof': chi2/max(dof,1),
            'AIC': chi2+2*2, 'pred': pred,
        }
        print(f'  NFW: logM={logM:.2f}, c={c:.1f}, chi2/dof={chi2/max(dof,1):.2f}')
    except Exception as e:
        print(f'  NFW failed: {e}')

    # NFW + 2halo
    def chi2_2h(p):
        logM, c, b = p
        pred = nfw_DeltaSigma(R_f, 10**logM, c, z_stack) + two_halo_DS(R_f, b, z_stack)
        return np.sum(((DS_f - pred)/eDS_f)**2)

    try:
        res = differential_evolution(chi2_2h, [(12,16),(1,20),(0.1,10)], seed=42, maxiter=1000)
        logM, c, b = res.x; chi2 = res.fun; dof = len(R_f)-3
        pred = nfw_DeltaSigma(R_f, 10**logM, c, z_stack) + two_halo_DS(R_f, b, z_stack)
        results['NFW+2halo'] = {
            'logM': logM, 'c': c, 'b': b,
            'chi2': chi2, 'dof': dof, 'chi2_dof': chi2/max(dof,1),
            'AIC': chi2+2*3, 'pred': pred,
        }
        print(f'  NFW+2h: logM={logM:.2f}, c={c:.1f}, b={b:.2f}, chi2/dof={chi2/max(dof,1):.2f}')
    except Exception as e:
        print(f'  NFW+2h failed: {e}')

    # AIC比較
    if results:
        aic_min = min(r['AIC'] for r in results.values())
        for r in results.values():
            r['DAIC'] = r['AIC'] - aic_min

    return results


# ============================================================
# Main
# ============================================================
def main():
    print('=' * 60)
    print('cl1 + cl3 Stack (cl4/cl27 excluded)')
    print('=' * 60)

    R_bins = np.array([0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1.0, 1.5, 2.0, 3.0, 5.0])

    profiles = {}
    for cl_name, cl in CLUSTERS.items():
        print(f'¥n--- {cl_name} (z={cl["z"]}) ---')

        src_file = find_source_file(cl_name)
        if src_file is None:
            print(f'  Source file not found for {cl_name}')
            print(f'  Expected: {cl_name}_sources_photoz.csv')
            continue

        sources = load_sources(src_file, cl['z'])
        if sources is None or len(sources) &amp;lt; 100:
            print(f'  Insufficient sources')
            continue
        prof = compute_DeltaSigma_profile(sources, cl['ra'], cl['dec'], cl['z'], R_bins)
```

```
    profiles[cl_name] = prof

    # 個別プロファイル表示
    total_sn = 0
    print(f'¥n  {"R[Mpc]":&gt;8} {"N":&gt;7} {"DS":&gt;12} {"DS_x":&gt;12}')
    for i in range(len(prof['R'])):
        print(f'  {prof["R"][i]:8.3f} {prof["n"][i]:7d} '
              f'{prof["DS"][i]:12.2f} {prof["DS_x"][i]:12.2f}')
        if prof['e_DS'][i] &lt; 1e18 and prof['e_DS'][i] &gt; 0:
            total_sn += (prof['DS'][i]/prof['e_DS'][i])**2
    print(f'  S/N = {np.sqrt(total_sn):.1f}')

if len(profiles) &lt; 1:
    print('¥nERROR: No profiles computed.')
    return

# スタック
if len(profiles) &gt;= 2:
    print(f'¥n{"="*60}')
    print(f'Stacking {len(profiles)} clusters')
    print(f'{"="*60}')
    stacked = stack_profiles(list(profiles.values()))
    z_stack = np.mean([cl['z'] for cl in CLUSTERS.values()])
else:
    cl_name = list(profiles.keys())[0]
    print(f'¥n  Only {cl_name} available, using single profile')
    stacked = profiles[cl_name]
    z_stack = CLUSTERS[cl_name]['z']

# スタックプロファイル表示
print(f'¥n  Stacked profile:')
print(f'  {"R[Mpc]":&gt;8} {"N_total":&gt;8} {"DS":&gt;12} {"DS_x":&gt;12} {"e_DS":&gt;12}')
for i in range(len(stacked['R'])):
    print(f'  {stacked["R"][i]:8.3f} {stacked["n"][i]:8d} '
          f'{stacked["DS"][i]:12.2f} {stacked["DS_x"][i]:12.2f} '
          f'{stacked["e_DS"][i]:12.2f}')

# モデルフィット
print(f'¥n{"="*60}')
print('Model Fitting')
print(f'{"="*60}')
fits = fit_all(stacked['R'], stacked['DS'], stacked['e_DS'], z_stack)

if fits:
    print(f'¥n--- AIC Comparison ---')
    print(f'{"Model":&lt;15} {"chi2":&gt;8} {"dof":&gt;4} {"chi2/dof":&gt;9} {"AIC":&gt;8} {"DAIC":&gt;7}')
    for name, f in sorted(fits.items(), key=lambda x: x[1]['AIC']):
        print(f'{name:&lt;15} {f["chi2"]:8.2f} {f["dof"]:4d} {f["chi2_dof"]:9.2f} '
              f'{f["AIC"]:8.2f} {f["DAIC"]:7.2f}')

# 図
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

ax = axes[0]
mask = stacked['n'] &gt; 0
ax.errorbar(stacked['R'][mask], stacked['DS'][mask],
            yerr=stacked['e_DS'][mask], fmt='ko', capsize=3, label='cl1+cl3 stack')

colors_m = {'NFW': '#1a1a2e', 'NFW+2halo': '#e94560'}
for name, f in fits.items():
    ax.plot(stacked['R'][mask], f['pred'], '-', color=colors_m.get(name, 'grey'),
            lw=2, label=f'{name} (DAIC={f["DAIC"]:.1f})')

ax.set_xlabel('R [Mpc]')
ax.set_ylabel('Delta-Sigma [Msun/Mpc^2]')
ax.set_xscale('log')
ax.set_title('(a) cl1+cl3 Stack: Delta-Sigma')
ax.legend(fontsize=8)
ax.grid(True, alpha=0.3)

# Null test
ax = axes[1]
ax.errorbar(stacked['R'][mask], stacked['DS_x'][mask],
            yerr=stacked['e_DS'][mask], fmt='ko', capsize=3)
ax.axhline(0, color='#e94560', ls='--', lw=2)
ax.set_xlabel('R [Mpc]')
ax.set_ylabel('Delta-Sigma_x [Msun/Mpc^2]')
ax.set_xscale('log')
ax.set_title('(b) Cross component (null test)')
ax.grid(True, alpha=0.3)

plt.suptitle('cl1+cl3 Stack Weak Lensing (cl4/cl27 excluded)', fontsize=13)
plt.tight_layout()
fig_path = OUTDIR / 'cl1_cl3_stack.png'
```

```
    plt.savefig(fig_path, dpi=150)
    print(f'¥nFigure: {fig_path}')

    # JSON
    summary = {
        'clusters_used': list(CLUSTERS.keys()),
        'clusters_excluded': ['cl4', 'cl27'],
        'exclusion_reason': 'No SDSS spectroscopic members found',
        'n_bins': int(len(stacked['R'])),
        'fits': {name: {k: v for k, v in f.items() if k != 'pred'}
                 for name, f in fits.items()},
    }
    with open(OUTDIR / 'cl1_cl3_stack.json', 'w') as f:
        json.dump(summary, f, indent=2, default=float)
    print(f'Results: {OUTDIR / "cl1_cl3_stack.json"}')


if __name__ == '__main__':
    main()
```

cl1_modelB_fit.py

章: 付録(弱レンズ)

目的: 光伝播モデルB(式10a-10d, 提案G) cl1フィット

解析対象データ: cl1_individual_shear.csv

主要結果: モデルB: DAIC=+2.0(パラメータペナルティのみ、chi2同一)。delta->0.001
Mpc(下限)に収束。NFWと区別不能。「局在リング棄却、正式モデルBは区別不能」が正しい結論。

ソースコード (424 lines, 14,139 bytes)

```
#!/usr/bin/env python3
"""
cl1_modelB_fit.py
=================
光伝播モデルB（式10a-10d, 提案G）をcl1弱レンズに組み込む。

前回棄却されたのは「Gaussianリング」パラメトリックモデル（DAIC=+20.5）であり、
膜宇宙論の正式な光伝播モデルB（提案G）とは根本的に異なる。

モデルBの物理:
  光は膜から距離δの層を伝播し、折り目に接近すると測地線が再配線される。
  これにより、NFWプロファイルに対して以下の2つの修正が加わる:
    1. Fold attenuation: κ_fold = exp(-δ/ξ_eff) → κを減衰
    2. Geodesic rewiring: P_rewire寄与 → κを増強

式(10a-10d):
  (10a) κ_fold = exp(-δ/ξ(r))
  (10b) n_fold(r) = k_cross × (T(r,r_s)/r)²
  (10c) P_rewire(r) = 1 - exp(-π(2δ)²·n_fold(r))
  (10d★) κ_total/κ_A = [1 + α·β_AB·q] × [q + (1-q)·exp(-δ/ξ)]
         ここで q = ∫ρ·P_rewire dl / ∫ρ dl

フィットするモデル:
  A) NFW (baseline: M200, c200)
  B) MOND Abel変換 (M200, c200_mond)
  C) Membrane モデルB 提案G (M200, c200, δ, k_cross, α_rewire)
  D) Membrane モデルB 簡易版 (M200, c200, δ) ← k_cross, α固定

実行: uv run --with scipy --with matplotlib python cl1_modelB_fit.py
"""

import numpy as np
import csv
import json
from pathlib import Path
from scipy.optimize import differential_evolution
from scipy.integrate import quad

import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib import font_manager as _fm
for _fp in ['/usr/share/fonts/opentype/ipafont-gothic/ipag.ttf',
            '/usr/share/fonts/opentype/ipafont-gothic/ipagp.ttf']:
    try: _fm.fontManager.addfont(_fp)
    except: pass
plt.rcParams['font.family'] = 'IPAGothic'
plt.rcParams['axes.unicode_minus'] = False

OUTDIR = Path("cluster_stack_v3_output")
OUTDIR.mkdir(exist_ok=True)

# 宇宙論
c_light = 2.998e5; H0 = 70.0; Omega_m = 0.3; Omega_L = 0.7
RHO_CRIT_0 = 1.36e11  # Msun/Mpc^3

Z_CL = 0.313

def E(z): return np.sqrt(Omega_m*(1+z)**3 + Omega_L)

def D_A(z):
    f = lambda zp: 1.0/E(zp)
    chi, _ = quad(f, 0, z)
    return c_light/H0*chi/(1+z)

# cl1 パラメータ（前回解析から）
# rs はtanhフィットから（TA3_gc_independent.csvから取得すべき）
RS_CL1 = 0.36  # Mpc (typical for z~0.3 cluster)
# ============================================================
```

```
# NFW Sigma, DeltaSigma
# ================================================================
def nfw_profiles(R, M200, c, z):
    """NFWのΣ(R)とΔΣ(R)を返す"""
    rho_c = RHO_CRIT_0 * E(z)**2
    r200 = (3*M200/(4*np.pi*200*rho_c))**(1./3)
    rs = r200/c
    rho_s = M200/(4*np.pi*rs**3*(np.log(1+c)-c/(1+c)))

    x = np.atleast_1d(R/rs).astype(float)
    Sig = np.zeros_like(x)

    lt = (x&gt;0)&amp;(x&lt;1); gt = x&gt;1; eq = np.abs(x-1)&lt;1e-4
    if np.any(lt):
        t = np.sqrt(1-x[lt]**2)
        Sig[lt] = 1/(x[lt]**2-1)*(1-np.arctanh(t)/t)
    if np.any(gt):
        t = np.sqrt(x[gt]**2-1)
        Sig[gt] = 1/(x[gt]**2-1)*(1-np.arctan(t)/t)
    if np.any(eq):
        Sig[eq] = 1./3

    Sigma = 2*rs*rho_s*Sig  # Msun/Mpc^2

    # Mean Sigma(&lt;R)
    Sigma_mean = np.zeros_like(R)
    for i, Ri in enumerate(R):
        r_int = np.linspace(0.001*Ri, Ri, 300)
        x_int = r_int/rs
        S_int = np.zeros_like(x_int)
        lt_i = (x_int&gt;0)&amp;(x_int&lt;1); gt_i = x_int&gt;1; eq_i = np.abs(x_int-1)&lt;1e-4
        if np.any(lt_i):
            t_i = np.sqrt(1-x_int[lt_i]**2)
            S_int[lt_i] = 1/(x_int[lt_i]**2-1)*(1-np.arctanh(t_i)/t_i)
        if np.any(gt_i):
            t_i = np.sqrt(x_int[gt_i]**2-1)
            S_int[gt_i] = 1/(x_int[gt_i]**2-1)*(1-np.arctan(t_i)/t_i)
        if np.any(eq_i):
            S_int[eq_i] = 1./3
        S_int *= 2*rs*rho_s
        Sigma_mean[i] = 2*np.trapz(r_int*S_int, r_int)/Ri**2

    DeltaSigma = Sigma_mean - Sigma
    return Sigma, DeltaSigma, rs, rho_s


# ================================================================
# 光伝播モデルB (式10a-10d)
# ================================================================
def T_tanh(r, rs, w=3.0):
    """tanh遷移関数"""
    return 0.5*(1 + np.tanh(w*(r - rs)/rs))


def fold_density(r, rs, k_cross):
    """式(10b): n_fold(r) = k_cross * (T(r,rs)/r)^2"""
    T = T_tanh(r, rs)
    if r &lt; 0.001:
        return 0.0
    return k_cross * (T/r)**2


def P_rewire(r, rs, delta, k_cross):
    """式(10c): P_rewire(r) = 1 - exp(-pi*(2*delta)^2 * n_fold(r))"""
    n = fold_density(r, rs, k_cross)
    return 1 - np.exp(-np.pi * (2*delta)**2 * n)


def q_factor(R, rs, delta, k_cross, rho_s, rs_nfw):
    """
    q = ∫ ρ·P_rewire dl / ∫ ρ dl
    ρ = NFW 3D density at r = sqrt(R^2 + l^2)
    積分は視線方向 (l) に沿って実施
    """
    def rho_nfw(r):
        x = r / rs_nfw
        if x &lt; 1e-6: return rho_s
        return rho_s / (x * (1+x)**2)

    l_max = 10.0  # Mpc
    n_pts = 200
    l_arr = np.linspace(-l_max, l_max, n_pts)
    dl = l_arr[1] - l_arr[0]
    rho_arr = np.array([rho_nfw(np.sqrt(R**2 + l**2)) for l in l_arr])
```

```
    Prw_arr = np.array([P_rewire(np.sqrt(R**2 + l**2), rs, delta, k_cross) for l in l_arr])

    num = np.trapz(rho_arr * Prw_arr, l_arr)
    den = np.trapz(rho_arr, l_arr)

    if den &lt; 1e-30:
        return 0.0
    return num / den


def kappa_ratio_modelB(R, rs_membrane, delta, k_cross, alpha_rw,
                       xi_eff, beta_AB, rho_s, rs_nfw):
    """
    式(10d★): κ_total/κ_A = [1 + α·β_AB·q] × [q + (1-q)·exp(-δ/ξ)]

    Returns: κ_total / κ_NFW (modification factor)
    """
    R = np.atleast_1d(R)
    ratio = np.ones_like(R)

    for i, Ri in enumerate(R):
        q = q_factor(Ri, rs_membrane, delta, k_cross, rho_s, rs_nfw)
        kf = np.exp(-delta / xi_eff)  # 式(10a)

        # 提案G
        factor = (1 + alpha_rw * beta_AB * q) * (q + (1 - q) * kf)
        ratio[i] = factor

    return ratio


def membrane_modelB_DeltaSigma(R, M200, c_nfw, delta, k_cross, alpha_rw, z):
    """
    膜宇宙論モデルB: NFW × κ修正因子

    DeltaSigma_membrane(R) = kappa_ratio(R) × DeltaSigma_NFW(R)

    パラメータ:
      M200: NFW質量 [Msun]
      c_nfw: NFW concentration
      delta: 膜-光路間距離 [Mpc]
      k_cross: 折り目密度振幅
      alpha_rw: 再配線強度パラメータ (段階2b標準: α=2.0)
    """
    Sigma, DS_nfw, rs_nfw, rho_s = nfw_profiles(R, M200, c_nfw, z)

    # 膜パラメータ
    rs_membrane = RS_CL1  # クラスターのtanhスケール半径
    xi_eff = rs_membrane * 0.5  # ξ_eff ~ rs/2 (典型値)
    beta_AB = 4.0 / (3*np.pi)  # ≈ 0.424 (v3.0 標準)

    # κ修正因子
    ratio = kappa_ratio_modelB(R, rs_membrane, delta, k_cross, alpha_rw,
                               xi_eff, beta_AB, rho_s, rs_nfw)

    # ΔΣ_membrane = ratio × ΔΣ_NFW
    # 注: 厳密にはΣ修正→ΔΣ再計算が必要だが、ratio~1近傍ではこの近似で十分
    DS_membrane = ratio * DS_nfw

    return DS_membrane


# ============================================================
# プロファイル読み込み
# ============================================================
def load_profile():
    """cl1のシェアプロファイルを読み込み"""
    # cl1_individual_shear.csv を探す
    for d in [Path('.'), Path('..'), OUTDIR, Path('cluster_stack_output')]:
        if not d.exists(): continue
        for f in d.glob('*individual*shear*.csv'):
            return load_binned(f)

    # ハードコード代替
    print('Using hardcoded cl1 profile')
    R = np.array([0.07, 0.14, 0.24, 0.39, 0.59, 0.84, 1.22, 1.73, 2.45])
    gt = np.array([0.035, 0.025, 0.018, 0.012, 0.008, 0.005, 0.003, 0.002, 0.001])
    eg = np.array([0.012, 0.006, 0.004, 0.003, 0.002, 0.002, 0.001, 0.001, 0.001])
    return R, gt, eg


def load_binned(filepath):
    print(f'Loading: {filepath}')
    with open(filepath, 'r', encoding='utf-8-sig') as f:
```

```
        first = f.readline().strip()
        if first.startswith('#'): first = first[1:].strip()
        headers = [h.strip() for h in first.split(',')]
        print(f'  Headers: {headers}')

        R, gt, eg = [], [], []
        for line in f:
            cols = line.strip().split(',')
            try:
                # R
                r_col = next((i for i, h in enumerate(headers) if 'kpc' in h.lower()), 0)
                r_val = float(cols[r_col])
                if 'kpc' in headers[r_col].lower():
                    r_val /= 1000  # kpc -&gt; Mpc

                gt_col = next((i for i, h in enumerate(headers) if 'gamma_t' in h.lower()), 1)
                eg_col = next((i for i, h in enumerate(headers) if 'err' in h.lower()), None)

                gt_val = float(cols[gt_col])
                eg_val = float(cols[eg_col]) if eg_col else 0.01

                R.append(r_val); gt.append(gt_val); eg.append(eg_val)
            except (ValueError, IndexError):
                pass
    print(f'  {len(R)} bins')
    return np.array(R), np.array(gt), np.array(eg)


# ============================================================
# フィッティング
# ============================================================
def fit_models(R, gt, eg):
    """4モデルフィット"""
    mask = (eg &gt; 0) &amp; (eg &lt; 1) &amp; np.isfinite(gt)
    R_f, gt_f, eg_f = R[mask], gt[mask], eg[mask]
    if len(R_f) &lt; 4:
        print('Too few bins')
        return {}

    results = {}

    # (A) NFW
    def chi2_nfw(p):
        logM, c, logA = p
        DS = nfw_profiles(R_f, 10**logM, c, Z_CL)[1]
        pred = 10**logA * DS
        return np.sum(((gt_f - pred)/eg_f)**2)

    try:
        res = differential_evolution(chi2_nfw, [(13,16),(1,15),(-18,-12)],
                                     seed=42, maxiter=1000)
        logM, c, logA = res.x
        DS = nfw_profiles(R_f, 10**logM, c, Z_CL)[1]
        pred = 10**logA * DS
        chi2 = res.fun; dof = len(R_f)-3
        results['NFW'] = {
            'logM': logM, 'c': c, 'logA': logA,
            'chi2': chi2, 'dof': dof, 'chi2_dof': chi2/max(dof,1),
            'AIC': chi2+2*3, 'pred': pred, 'n_param': 3,
        }
        print(f'  NFW: logM={logM:.2f}, c={c:.1f}, chi2/dof={chi2/max(dof,1):.2f}')
    except Exception as e:
        print(f'  NFW failed: {e}')

    # (C) Membrane Model B 簡易版 (delta, k_cross=0.1固定, alpha=2.0固定)
    def chi2_memB_simple(p):
        logM, c, logA, delta = p
        DS_mem = membrane_modelB_DeltaSigma(R_f, 10**logM, c, delta,
                                            k_cross=0.1, alpha_rw=2.0, z=Z_CL)
        pred = 10**logA * DS_mem
        return np.sum(((gt_f - pred)/eg_f)**2)

    try:
        res = differential_evolution(chi2_memB_simple,
            [(13,16),(1,15),(-18,-12),(0.001,0.5)],
            seed=42, maxiter=1000, tol=1e-5)
        logM, c, logA, delta = res.x
        DS_mem = membrane_modelB_DeltaSigma(R_f, 10**logM, c, delta,
                                            k_cross=0.1, alpha_rw=2.0, z=Z_CL)
        pred = 10**logA * DS_mem
        chi2 = res.fun; dof = len(R_f)-4
        results['MembraneB_simple'] = {
            'logM': logM, 'c': c, 'logA': logA, 'delta': delta,
            'k_cross': 0.1, 'alpha_rw': 2.0,
```

```
            'chi2': chi2, 'dof': dof, 'chi2_dof': chi2/max(dof,1),
            'AIC': chi2+2*4, 'pred': pred, 'n_param': 4,
        }
        print(f'  MemB_simple: logM={logM:.2f}, c={c:.1f}, delta={delta:.4f}, '
              f'chi2/dof={chi2/max(dof,1):.2f}')
    except Exception as e:
        print(f'  MemB_simple failed: {e}')

    # (D) Membrane Model B フル (delta, k_cross, alpha_rw全てフリー)
    def chi2_memB_full(p):
        logM, c, logA, delta, k_cross, alpha_rw = p
        DS_mem = membrane_modelB_DeltaSigma(R_f, 10**logM, c, delta,
                                            k_cross=k_cross, alpha_rw=alpha_rw, z=Z_CL)
        pred = 10**logA * DS_mem
        return np.sum(((gt_f - pred)/eg_f)**2)

    try:
        res = differential_evolution(chi2_memB_full,
            [(13,16),(1,15),(-18,-12),(0.001,0.5),(0.01,10),(0.1,5)],
            seed=42, maxiter=1500, tol=1e-5)
        logM, c, logA, delta, k_cross, alpha_rw = res.x
        DS_mem = membrane_modelB_DeltaSigma(R_f, 10**logM, c, delta,
                                            k_cross=k_cross, alpha_rw=alpha_rw, z=Z_CL)
        pred = 10**logA * DS_mem
        chi2 = res.fun; dof = len(R_f)-6
        results['MembraneB_full'] = {
            'logM': logM, 'c': c, 'logA': logA,
            'delta': delta, 'k_cross': k_cross, 'alpha_rw': alpha_rw,
            'chi2': chi2, 'dof': dof, 'chi2_dof': chi2/max(dof,1),
            'AIC': chi2+2*6, 'pred': pred, 'n_param': 6,
        }
        print(f'  MemB_full: delta={delta:.4f}, k={k_cross:.3f}, alpha={alpha_rw:.2f}, '
              f'chi2/dof={chi2/max(dof,1):.2f}')
    except Exception as e:
        print(f'  MemB_full failed: {e}')

    # AIC比較
    if results:
        aic_min = min(r['AIC'] for r in results.values())
        for r in results.values():
            r['DAIC'] = r['AIC'] - aic_min

    return results


def main():
    print('=' * 60)
    print('cl1 Weak Lensing: Model B (Eqs. 10a-10d) Fit')
    print('=' * 60)

    R, gt, eg = load_profile()
    mask = (eg &amp;gt; 0) &amp;amp; (eg &amp;lt; 1) &amp;amp; np.isfinite(gt) &amp;amp; (R &amp;gt; 0)
    R, gt, eg = R[mask], gt[mask], eg[mask]
    print(f'Valid bins: {len(R)}, R range: [{R.min():.3f}, {R.max():.3f}] Mpc')

    fits = fit_models(R, gt, eg)

    if fits:
        print(f'¥n{"="*60}')
        print('AIC Comparison')
        print(f'{"="*60}')
        print(f'{"Model":&amp;lt;20} {"chi2":&amp;gt;7} {"dof":&amp;gt;4} {"chi2/dof":&amp;gt;9} '
              f'{"AIC":&amp;gt;7} {"DAIC":&amp;gt;7} {"n_p":&amp;gt;4}')
        for name, f in sorted(fits.items(), key=lambda x: x[1]['AIC']):
            print(f'{name:&amp;lt;20} {f["chi2"]:7.2f} {f["dof"]:4d} {f["chi2_dof"]:9.2f} '
                  f'{f["AIC"]:7.2f} {f["DAIC"]:7.2f} {f["n_param"]:4d}')

    # 図
    fig, ax = plt.subplots(figsize=(10, 7))
    ax.errorbar(R, gt, yerr=eg, fmt='ko', capsize=3, markersize=6, label='cl1 data')

    colors_m = {'NFW': '#1a1a2e', 'MembraneB_simple': '#e94560', 'MembraneB_full': '#2ecc71'}
    for name, f in fits.items():
        ax.plot(R, f['pred'], '-', color=colors_m.get(name, 'grey'), lw=2,
                label=f'{name} (DAIC={f["DAIC"]:.1f})')

    ax.set_xlabel('R [Mpc]', fontsize=12)
    ax.set_ylabel('gamma_t', fontsize=12)
    ax.set_xscale('log')
    ax.set_title('cl1: NFW vs Membrane Model B (Eqs. 10a-10d)', fontsize=13)
    ax.legend(fontsize=9)
    ax.grid(True, alpha=0.3)
    ax.axhline(0, color='grey', ls='--', alpha=0.3)
    plt.tight_layout()
```

```
    fig_path = OUTDIR / 'cl1_modelB_fit.png'
    plt.savefig(fig_path, dpi=150)
    print(f'¥nFigure: {fig_path}')

    # JSON
    summary = {name: {k: v for k, v in f.items() if k != 'pred'}
               for name, f in fits.items()}
    with open(OUTDIR / 'cl1_modelB_fit.json', 'w') as f:
        json.dump(summary, f, indent=2, default=float)
    print(f'Results: {OUTDIR / "cl1_modelB_fit.json"}')


if __name__ == '__main__':
    main()
```

cluster_stack_v3.py

章: 中間版(v3_fix2に置換)

目的: BCG同定(z_spec NULL許可版) + NFW+2haloモデル設計

解析対象データ: SDSS DR17

主要結果: z_spec NULLで前景銀河誤認(cl27 r=8.85)。v3_fixで修正。

ソースコード (569 lines, 18,866 bytes)

```
#!/usr/bin/env python3
"""
cluster_stack_v3.py
==================
スタック弱レンズ解析の改善版（v3）

v2からの改善点:
  1. BCG同定: SDSSから各クラスターのBCG（最輝星銀河）を特定し、正確な中心座標を使用
  2. 2-halo項: NFW 1-halo + linear bias 2-halo モデル
  3. 非弛緩系の追加フィルタ: BCG-X線中心オフセット, DS統計量
  4. photo-z改善: P(z)分布を使用した背景銀河選択
  5. 膜宇宙論モデル（delta-layer）との比較

データ:
  - HSC-SSP PDR3 弱レンズシェアカタログ
  - SDSS DR17 分光/測光カタログ（BCG同定）
  - Planck PSZ2 / redMaPPer クラスターカタログ

クラスター:
  - cl1: (140.45, -0.25), z=0.313 — 主要クラスター
  - cl3, cl4, cl27: z~0.32 — スタック候補

実行: uv run --with scipy --with matplotlib --with astropy --with requests python cluster_stack_v3.py

注意: HSCデータはhsc-release.mtk.nao.ac.jp CASからの取得が必要。
      Claude Code（ネットワーク制限なし）で実行。
"""

import numpy as np
import sys
import json
from pathlib import Path
from scipy.optimize import minimize, curve_fit
from scipy.stats import chi2 as chi2_dist
from scipy.integrate import quad
from scipy.interpolate import interp1d

OUTDIR = Path("cluster_stack_v3_output")
OUTDIR.mkdir(exist_ok=True)

# 物理定数・宇宙論パラメータ
c_light = 2.998e5  # km/s
G_SI = 6.674e-11
Msun = 1.989e30
pc = 3.086e16
Mpc = 1e6 * pc
H0 = 70.0  # km/s/Mpc
Omega_m = 0.3
Omega_L = 0.7
rho_crit_0 = 3 * H0**2 / (8 * np.pi * G_SI) * (1e3/(Mpc))**2 / Msun * Mpc**3
# rho_crit_0 in Msun/Mpc^3

# クラスター情報（前回解析から）
CLUSTERS = {
    'cl1': {'ra': 140.45, 'dec': -0.25, 'z': 0.313, 'sigma_v': 527,
            'n_member': 22, 'notes': 'primary cluster'},
    'cl3': {'ra': 139.80, 'dec': -0.10, 'z': 0.318, 'sigma_v': None,
            'n_member': None, 'notes': 'stack candidate'},
    'cl4': {'ra': 140.90, 'dec': -0.50, 'z': 0.315, 'sigma_v': None,
            'n_member': None, 'notes': 'stack candidate'},
    'cl27': {'ra': 141.20, 'dec': 0.10, 'z': 0.322, 'sigma_v': None,
             'n_member': None, 'notes': 'stack candidate'},
}


# ============================================================
# Step 1: BCG同定
# ============================================================
def step1_bcg_identification():
    """
```

```
SDSSから各クラスター中心付近のBCG候補を特定。
BCG = クラスターredshift付近で最も明るい銀河
"""
import requests

print('=' * 60)
print('Step 1: BCG Identification')
print('=' * 60)

bcg_results = {}

for cl_name, cl in CLUSTERS.items():
    print(f'¥n  {cl_name}: (RA={cl["ra"]}, Dec={cl["dec"]}, z={cl["z"]})')

    # SDSS CasJobs/SkyServer SQL query
    # BCG候補: クラスター中心から1 Mpc以内、z_spec ~ z_cl、最も明るいr-band
    search_radius_arcmin = 5.0  # ~1 Mpc at z=0.3
    z_lo = cl['z'] - 0.01
    z_hi = cl['z'] + 0.01

    sql = f"""
    SELECT TOP 20
        p.objID, p.ra, p.dec, p.r, p.g, p.i,
        s.z as z_spec, s.zErr,
        p.type, p.petroRad_r
    FROM PhotoObj AS p
    LEFT JOIN SpecObj AS s ON p.objID = s.bestObjID
    WHERE
        p.ra BETWEEN {cl['ra'] - search_radius_arcmin/60} AND {cl['ra'] + search_radius_arcmin/60}
        AND p.dec BETWEEN {cl['dec'] - search_radius_arcmin/60} AND {cl['dec'] + search_radius_arcmin/60}
        AND p.type = 3
        AND p.r &lt; 19.0
        AND (s.z BETWEEN {z_lo} AND {z_hi} OR s.z IS NULL)
    ORDER BY p.r ASC
    """

    # SDSS SkyServer API
    url = "https://skyserver.sdss.org/dr17/SkyServerWS/SearchTools/SqlSearch"
    try:
        r = requests.get(url, params={'cmd': sql, 'format': 'json'}, timeout=30)
        data = r.json()

        if isinstance(data, list) and len(data) &gt; 0:
            rows = data[0].get('Rows', data) if isinstance(data[0], dict) else data
            if rows:
                bcg = rows[0]  # 最も明るい
                bcg_ra = float(bcg.get('ra', cl['ra']))
                bcg_dec = float(bcg.get('dec', cl['dec']))
                bcg_r = float(bcg.get('r', 99))
                bcg_z = bcg.get('z_spec', None)

                # BCGとクラスター中心のオフセット
                offset_arcmin = np.sqrt(
                    ((bcg_ra - cl['ra']) * np.cos(np.radians(cl['dec'])))**2 +
                    (bcg_dec - cl['dec'])**2
                ) * 60

                bcg_results[cl_name] = {
                    'bcg_ra': bcg_ra,
                    'bcg_dec': bcg_dec,
                    'bcg_r_mag': bcg_r,
                    'bcg_z_spec': bcg_z,
                    'offset_arcmin': offset_arcmin,
                    'n_candidates': len(rows),
                }
                print(f'    BCG: RA={bcg_ra:.4f}, Dec={bcg_dec:.4f}, r={bcg_r:.2f}')
                print(f'    Offset: {offset_arcmin:.2f} arcmin')
                if bcg_z:
                    print(f'    z_spec: {bcg_z}')
            else:
                print(f'    No candidates found')
        else:
            print(f'    SDSS query returned no data')

    except Exception as e:
        print(f'    SDSS query failed: {e}')
        # フォールバック: 元の中心を使用
        bcg_results[cl_name] = {
            'bcg_ra': cl['ra'],
            'bcg_dec': cl['dec'],
            'bcg_r_mag': None,
            'bcg_z_spec': None,
            'offset_arcmin': 0.0,
            'n_candidates': 0,
```

```
                'fallback': True,
            }

    return bcg_results


# ============================================================
# Step 2: NFW + 2-halo モデル
# ============================================================
def nfw_sigma(R_Mpc, M200, c200, z):
    """
    NFW profile projected surface mass density Sigma(R).
    M200: M_sun, c200: concentration, z: redshift
    """
    # rho_crit at z
    Ez2 = Omega_m * (1+z)**3 + Omega_L
    rho_c = rho_crit_0 * Ez2  # Msun/Mpc^3

    # r200, rs
    r200 = (3 * M200 / (4 * np.pi * 200 * rho_c))**(1.0/3)  # Mpc
    rs = r200 / c200  # Mpc

    x = R_Mpc / rs
    x = np.clip(x, 1e-6, 1e4)

    # Sigma(x) for NFW (Bartelmann 1996, Wright &amp;amp; Brainerd 2000)
    sigma = np.zeros_like(x)

    # x &lt; 1
    m1 = x &lt; 1
    if np.any(m1):
        t = np.sqrt(1 - x[m1]**2)
        sigma[m1] = 1.0 / (x[m1]**2 - 1) * (1 - np.arctanh(t) / t)

    # x &gt; 1
    m2 = x &gt; 1
    if np.any(m2):
        t = np.sqrt(x[m2]**2 - 1)
        sigma[m2] = 1.0 / (x[m2]**2 - 1) * (1 - np.arctan(t) / t)

    # x = 1
    m3 = np.abs(x - 1) &lt; 1e-6
    sigma[m3] = 1.0 / 3

    # Physical normalization
    rho_s = M200 / (4 * np.pi * rs**3 * (np.log(1+c200) - c200/(1+c200)))
    Sigma_phys = 2 * rs * rho_s * sigma  # Msun/Mpc^2

    return Sigma_phys


def nfw_delta_sigma(R_Mpc, M200, c200, z):
    """
    NFW excess surface mass density Delta-Sigma(R) = Sigma_mean(&lt;R) - Sigma(R)
    """
    Sigma_R = nfw_sigma(R_Mpc, M200, c200, z)

    # Sigma_mean(&lt;R) by numerical integration
    Sigma_mean = np.zeros_like(R_Mpc)
    for i, Ri in enumerate(R_Mpc):
        r_int = np.linspace(0.001, Ri, 200)
        S_int = nfw_sigma(r_int, M200, c200, z)
        Sigma_mean[i] = 2 * np.trapz(r_int * S_int, r_int) / Ri**2

    return Sigma_mean - Sigma_R


def two_halo_delta_sigma(R_Mpc, M200, b_lin, z):
    """
    2-halo term: Delta-Sigma_2h(R) = b_lin * rho_m * xi_mm(R)
    Simplified: power-law approximation for large R

    Delta-Sigma_2h ~ b_lin * rho_m_bar * (R/R0)^(-1)
    where R0 ~ 1 Mpc (correlation length)
    """
    Ez2 = Omega_m * (1+z)**3 + Omega_L
    rho_m = rho_crit_0 * Omega_m * (1+z)**3  # Msun/Mpc^3 (physical)

    # Simplified 2-halo: power-law
    R0 = 5.0  # Mpc (correlation length)
    xi_mm = (R_Mpc / R0)**(-1.8)  # matter correlation function approximation

    # Project along line of sight (simplified)
    Delta_Sigma_2h = b_lin * rho_m * R0 * xi_mm / 1e6  # scale factor
```

```
    return Delta_Sigma_2h


def model_nfw_2halo(R_Mpc, M200, c200, b_lin, z):
    """NFW 1-halo + 2-halo combined"""
    ds_1h = nfw_delta_sigma(R_Mpc, M200, c200, z)
    ds_2h = two_halo_delta_sigma(R_Mpc, M200, b_lin, z)
    return ds_1h + ds_2h


def model_membrane(R_Mpc, M_lens, f_delta, r_delta, z):
    """
    膜宇宙論 delta-layer model:
    Sigma(R) = Sigma_NFW(R) + f_delta * delta(R - r_delta)

    Simplified: NFW + concentrated mass shell
    """
    # NFW component (M200 = M_lens, c=5 typical)
    c200 = 5.0
    ds_nfw = nfw_delta_sigma(R_Mpc, M_lens, c200, z)

    # Delta-layer: sharp feature at r_delta
    # In projection: ring-like excess
    sigma_ring = f_delta * M_lens / (2 * np.pi * r_delta * 0.05)  # width=50kpc
    ring = np.exp(-0.5 * ((R_Mpc - r_delta) / 0.05)**2) * sigma_ring

    # Delta-Sigma from ring (approximate)
    ring_mean = np.zeros_like(R_Mpc)
    for i, Ri in enumerate(R_Mpc):
        r_int = np.linspace(0.001, Ri, 200)
        r_int_ring = np.exp(-0.5 * ((r_int - r_delta) / 0.05)**2) * sigma_ring
        ring_mean[i] = 2 * np.trapz(r_int * r_int_ring, r_int) / Ri**2
    ds_ring = ring_mean - ring

    return ds_nfw + ds_ring


# ================================================================
# Step 3: HSC弱レンズデータ取得
# ================================================================
def step3_hsc_shear(bcg_results):
    """
    HSC-SSP PDR3 弱レンズシェアカタログから背景銀河を取得。

    注: HSC CASは認証が必要な場合あり。
    このスクリプトはクエリを生成し、手動実行の手順を提供する。
    既にcl1のデータがローカルにある場合はそこから読み込む。
    """
    print('¥n' + '=' * 60)
    print('Step 3: HSC Shear Data')
    print('=' * 60)

    for cl_name, bcg in bcg_results.items():
        ra_c = bcg['bcg_ra']
        dec_c = bcg['bcg_dec']
        z_cl = CLUSTERS[cl_name]['z']

        # HSC SQL query
        sql = f"""
        SELECT
            object_id, i_ra, i_dec,
            e1, e2, weight, m_bias,
            photoz_mean, photoz_err,
            b_mode_mask
        FROM s21a_wide.weaklensing_hsm_regauss
        WHERE
            b_mode_mask = 1
            AND coneSearch(i_ra, i_dec, {ra_c}, {dec_c}, 30.0)
            AND photoz_mean &gt; {z_cl + 0.1}
            AND weight &gt; 0
        """

        print(f'¥n  {cl_name} (center: BCG at {ra_c:.4f}, {dec_c:.4f}):')
        print(f'  HSC SQL query saved to: {OUTDIR / f"hsc_query_{cl_name}.sql"}')

        (OUTDIR / f"hsc_query_{cl_name}.sql").write_text(sql)

    print('¥n  手動実行手順:')
    print('  1. https://hsc-release.mtk.nao.ac.jp/datasearch/ にアクセス')
    print('  2. 上記SQLクエリを実行')
    print('  3. 結果をCSVで保存: hsc_shear_{cl_name}.csv')
    print('  4. 本スクリプトを再実行')

    # 既存データの検索
```

```
    existing = {}
    for cl_name in CLUSTERS:
        for pattern in [f'hsc_shear_{cl_name}.csv', f'cl1_shear.csv',
                        f'{cl_name}_background.csv']:
            for search_dir in [OUTDIR, Path('.'), Path('..'),
                               Path('cluster_stack_output')]:
                f = search_dir / pattern
                if f.exists():
                    print(f'  Found existing: {f}')
                    existing[cl_name] = f
                    break

    return existing


# ============================================================
# Step 4: シェアプロファイル計算 + モデルフィット
# ============================================================
def compute_shear_profile(shear_file, ra_c, dec_c, z_cl, z_s_min=None):
    """背景銀河のシェアプロファイルを計算"""
    import csv

    data = []
    with open(shear_file, 'r') as f:
        reader = csv.DictReader(f)
        for row in reader:
            try:
                ra = float(row.get('i_ra', row.get('ra', '0')))
                dec = float(row.get('i_dec', row.get('dec', '0')))
                e1 = float(row.get('e1', '0'))
                e2 = float(row.get('e2', '0'))
                w = float(row.get('weight', '1'))
                z_ph = float(row.get('photoz_mean', row.get('z_phot', '1.0')))

                if w &gt; 0 and z_ph &gt; z_cl + 0.1:
                    data.append({
                        'ra': ra, 'dec': dec, 'e1': e1, 'e2': e2,
                        'w': w, 'z_ph': z_ph,
                    })
            except (ValueError, TypeError):
                pass

    if len(data) &lt; 50:
        print(f'  Warning: only {len(data)} background galaxies')
        return None

    print(f'  {len(data)} background galaxies')

    # 角度距離
    # D_A(z) simplified (flat LCDM)
    def D_A(z):
        from scipy.integrate import quad
        f = lambda zp: 1.0 / np.sqrt(Omega_m*(1+zp)**3 + Omega_L)
        chi, _ = quad(f, 0, z)
        return c_light / H0 * chi / (1+z)  # Mpc

    D_l = D_A(z_cl)

    # ビニング
    R_bins = np.logspace(np.log10(0.1), np.log10(5.0), 12)  # Mpc
    R_mid = np.sqrt(R_bins[:-1] * R_bins[1:])

    gamma_t = np.zeros(len(R_mid))
    gamma_x = np.zeros(len(R_mid))
    w_sum = np.zeros(len(R_mid))
    n_count = np.zeros(len(R_mid), dtype=int)

    for gal in data:
        # 角度距離
        dra = (gal['ra'] - ra_c) * np.cos(np.radians(dec_c))
        ddec = gal['dec'] - dec_c
        theta = np.sqrt(dra**2 + ddec**2) * np.pi / 180  # radians
        R_phys = theta * D_l  # Mpc (物理距離の近似)

        # phi (position angle)
        phi = np.arctan2(ddec, dra)

        # tangential/cross shear
        gt = -(gal['e1'] * np.cos(2*phi) + gal['e2'] * np.sin(2*phi))
        gx = -(gal['e2'] * np.cos(2*phi) - gal['e1'] * np.sin(2*phi))

        # Sigma_crit
        D_s = D_A(gal['z_ph'])
        D_ls = D_A(gal['z_ph']) - D_A(z_cl)  # approximate
```

```
        if D_ls &lt;= 0:
            continue
        Sigma_crit = c_light**2 / (4 * np.pi * G_SI) * D_s / (D_l * D_ls)
        # 単位変換は省略（相対比較のため）

        # ビンに振り分け
        idx = np.searchsorted(R_bins, R_phys) - 1
        if 0 &lt;= idx &lt; len(R_mid):
            gamma_t[idx] += gt * gal['w']
            gamma_x[idx] += gx * gal['w']
            w_sum[idx] += gal['w']
            n_count[idx] += 1

    # 加重平均
    mask = w_sum &gt; 0
    gamma_t[mask] /= w_sum[mask]
    gamma_x[mask] /= w_sum[mask]
    e_gamma = np.where(n_count &gt; 1, 1.0 / np.sqrt(n_count * w_sum / np.maximum(w_sum, 1)), 1e10)

    return {
        'R_Mpc': R_mid,
        'gamma_t': gamma_t,
        'gamma_x': gamma_x,
        'e_gamma': e_gamma,
        'n_count': n_count,
        'n_total': len(data),
    }


# ============================================================
# Step 5: モデルフィット + AIC比較
# ============================================================
def fit_models(profile, z_cl):
    """NFW, NFW+2halo, membrane モデルのフィット"""
    R = profile['R_Mpc']
    gt = profile['gamma_t']
    eg = profile['e_gamma']

    mask = (gt != 0) &amp;amp; (eg &lt; 1)
    R_fit = R[mask]
    gt_fit = gt[mask]
    eg_fit = eg[mask]

    if len(R_fit) &lt; 4:
        return None

    results = {}

    # (a) NFW only
    try:
        def nfw_model(R, logM, c):
            return nfw_delta_sigma(R, 10**logM, c, z_cl) * 1e-15  # scale

        popt, pcov = curve_fit(nfw_model, R_fit, gt_fit, p0=[14.5, 5],
                               sigma=eg_fit, maxfev=5000)
        gt_pred = nfw_model(R_fit, *popt)
        chi2_nfw = np.sum(((gt_fit - gt_pred) / eg_fit)**2)
        dof_nfw = len(R_fit) - 2
        results['NFW'] = {
            'logM200': popt[0], 'c200': popt[1],
            'chi2': chi2_nfw, 'dof': dof_nfw,
            'chi2_dof': chi2_nfw / dof_nfw,
            'AIC': chi2_nfw + 2*2,
        }
    except Exception as e:
        print(f'  NFW fit failed: {e}')

    # (b) NFW + 2-halo
    try:
        def nfw_2h_model(R, logM, c, b):
            return model_nfw_2halo(R, 10**logM, c, b, z_cl) * 1e-15

        popt2, pcov2 = curve_fit(nfw_2h_model, R_fit, gt_fit, p0=[14.5, 5, 3],
                                 sigma=eg_fit, maxfev=5000)
        gt_pred2 = nfw_2h_model(R_fit, *popt2)
        chi2_2h = np.sum(((gt_fit - gt_pred2) / eg_fit)**2)
        dof_2h = len(R_fit) - 3
        results['NFW+2halo'] = {
            'logM200': popt2[0], 'c200': popt2[1], 'b_lin': popt2[2],
            'chi2': chi2_2h, 'dof': dof_2h,
            'chi2_dof': chi2_2h / dof_2h,
            'AIC': chi2_2h + 2*3,
        }
    except Exception as e:
```

```
        print(f'  NFW+2halo fit failed: {e}')

    # AIC比較
    if results:
        aic_min = min(r['AIC'] for r in results.values())
        for name, r in results.items():
            r['DAIC'] = r['AIC'] - aic_min

    return results


# ============================================================
# Main
# ============================================================
def main():
    print('=' * 60)
    print('Cluster Stack Weak Lensing v3')
    print('BCG identification + NFW+2halo model')
    print('=' * 60)

    # Step 1: BCG同定
    bcg = step1_bcg_identification()

    # BCG結果サマリ
    print(f'¥n--- BCG Summary ---')
    for cl_name, b in bcg.items():
        offset = b.get('offset_arcmin', 0)
        print(f'  {cl_name}: offset={offset:.2f} arcmin, '
              f'r_mag={b.get("bcg_r_mag", "N/A")}')

    # Step 3: HSCデータ
    existing = step3_hsc_shear(bcg)

    if existing:
        print(f'¥n--- Processing existing shear data ---')
        for cl_name, shear_file in existing.items():
            b = bcg.get(cl_name, {})
            ra_c = b.get('bcg_ra', CLUSTERS[cl_name]['ra'])
            dec_c = b.get('bcg_dec', CLUSTERS[cl_name]['dec'])
            z_cl = CLUSTERS[cl_name]['z']

            print(f'¥n  {cl_name}:')
            profile = compute_shear_profile(shear_file, ra_c, dec_c, z_cl)
            if profile:
                print(f'    S/N = {np.sum(profile["gamma_t"]**2 / profile["e_gamma"]**2)**0.5:.1f}')
                print(f'    |gamma_x| = {np.median(np.abs(profile["gamma_x"])):.4f}')

                # モデルフィット
                results = fit_models(profile, z_cl)
                if results:
                    print(f'¥n    Model fits:')
                    for name, r in results.items():
                        print(f'      {name}: chi2/dof={r["chi2_dof"]:.2f}, '
                              f'DAIC={r["DAIC"]:.1f}')
    else:
        print('¥n  No existing shear data found.')
        print('  Run HSC queries manually, then re-run this script.')

    # 結果保存
    summary = {
        'bcg': {k: {kk: vv for kk, vv in v.items()} for k, v in bcg.items()},
        'existing_data': {k: str(v) for k, v in existing.items()} if existing else {},
    }
    with open(OUTDIR / 'stack_v3_summary.json', 'w') as f:
        json.dump(summary, f, indent=2, default=str)

    print(f'¥nResults: {OUTDIR}')
    print(f'¥n--- Next Steps ---')
    print(f'  1. If BCG offsets &gt; 0.5 arcmin: re-center on BCG')
    print(f'  2. Run HSC queries for all 4 clusters')
    print(f'  3. Stack profiles with Sigma_crit normalization')
    print(f'  4. Compare NFW vs NFW+2halo vs membrane models')
    print(f'  5. Report DAIC and chi2/dof for each model')


if __name__ == '__main__':
    main()
```

cluster_stack_v3_fix.py

章: 中間版(v3_fix2に置換)

目的: BCG同定修正版(z_spec必須+M_r<-21)

解析対象データ: SDSS DR17

主要結果: cl1 BCG offset=3.18'。cl4/cl27メンバーなし。v3_fix2で中心比較追加。

ソースコード (429 lines, 14,335 bytes)

```
#!/usr/bin/env python3
"""
cluster_stack_v3_fix.py
=======================
v3の2つのバグを修正:
  1. BCG同定: z_spec NULL除外 + 絶対等級フィルタ (M_r &lt; -21)
  2. シェア計算: CSVカラム自動検出 + デバッグ出力

実行: uv run --with scipy --with matplotlib --with requests python cluster_stack_v3_fix.py
"""

import numpy as np
import csv
import sys
import json
from pathlib import Path
from scipy.optimize import curve_fit
from scipy.stats import linregress

OUTDIR = Path("cluster_stack_v3_output")
OUTDIR.mkdir(exist_ok=True)

c_light = 2.998e5
G_SI = 6.674e-11
Msun = 1.989e30
Mpc = 3.086e22
H0 = 70.0
Omega_m = 0.3
Omega_L = 0.7

CLUSTERS = {
    'cl1':  {'ra': 140.45, 'dec': -0.25, 'z': 0.313},
    'cl3':  {'ra': 139.80, 'dec': -0.10, 'z': 0.318},
    'cl4':  {'ra': 140.90, 'dec': -0.50, 'z': 0.315},
    'cl27': {'ra': 141.20, 'dec':  0.10, 'z': 0.322},
}


# ============================================================
# Fix 1: BCG同定 (z_spec必須 + 絶対等級フィルタ)
# ============================================================
def bcg_identification():
    import requests

    print('=' * 60)
    print('BCG Identification (z_spec required, M_r &lt; -21)')
    print('=' * 60)

    results = {}

    for cl_name, cl in CLUSTERS.items():
        z = cl['z']
        ra, dec = cl['ra'], cl['dec']
        search_r = 5.0  # arcmin

        # 距離モジュール (簡易)
        # D_L ~ c*z/H0 * (1 + z/2) for flat LCDM approximation
        D_L_Mpc = c_light * z / H0 * (1 + z/2)
        dist_mod = 5 * np.log10(D_L_Mpc * 1e6 / 10)  # pc

        # 絶対等級 M_r &lt; -21 → 見かけ r &lt; dist_mod - 21
        r_limit = dist_mod - 21
        print(f'¥n  {cl_name}: z={z:.3f}, D_L={D_L_Mpc:.0f} Mpc, '
              f'dist_mod={dist_mod:.1f}, r_limit={r_limit:.1f}')

        # SDSS SQL: z_spec 必須、z_cl +/- 0.01、r &lt; r_limit
        sql = f"""
        SELECT TOP 10
            p.objID, p.ra, p.dec, p.r, p.g, p.i,
            s.z as z_spec, s.zErr,
```

```
            p.petroRad_r, p.petroR50_r
        FROM PhotoObj AS p
        JOIN SpecObj AS s ON p.objID = s.bestObjID
        WHERE
            p.ra BETWEEN {ra - search_r/60} AND {ra + search_r/60}
            AND p.dec BETWEEN {dec - search_r/60} AND {dec + search_r/60}
            AND p.type = 3
            AND s.z BETWEEN {z - 0.015} AND {z + 0.015}
            AND s.zWarning = 0
            AND p.r &lt; {r_limit}
        ORDER BY p.r ASC
        """

        url = "https://skyserver.sdss.org/dr17/SkyServerWS/SearchTools/SqlSearch"
        try:
            r_resp = requests.get(url, params={'cmd': sql, 'format': 'json'}, timeout=30)
            data = r_resp.json()

            rows = []
            if isinstance(data, list):
                for item in data:
                    if isinstance(item, dict) and 'Rows' in item:
                        rows = item['Rows']
                        break
                    elif isinstance(item, dict) and 'ra' in item:
                        rows = data
                        break

            if rows:
                bcg = rows[0]
                bcg_ra = float(bcg['ra'])
                bcg_dec = float(bcg['dec'])
                bcg_r = float(bcg['r'])
                bcg_z = float(bcg['z_spec'])
                M_r = bcg_r - dist_mod

                offset_arcmin = np.sqrt(
                    ((bcg_ra - ra) * np.cos(np.radians(dec)) * 60)**2 +
                    ((bcg_dec - dec) * 60)**2
                )
                offset_Mpc = offset_arcmin / 60 * np.pi / 180 * D_L_Mpc / (1+z)

                results[cl_name] = {
                    'bcg_ra': bcg_ra, 'bcg_dec': bcg_dec,
                    'bcg_r_mag': bcg_r, 'bcg_z_spec': bcg_z,
                    'M_r': M_r,
                    'offset_arcmin': offset_arcmin,
                    'offset_Mpc': offset_Mpc,
                    'n_candidates': len(rows),
                }
                print(f'    BCG: RA={bcg_ra:.5f} Dec={bcg_dec:.5f}')
                print(f'    r={bcg_r:.2f}, M_r={M_r:.2f}, z_spec={bcg_z:.4f}')
                print(f'    Offset: {offset_arcmin:.2f} arcmin = {offset_Mpc:.3f} Mpc')
                print(f'    Candidates: {len(rows)}')

                # 全候補表示
                for i, row in enumerate(rows[:5]):
                    off = np.sqrt(
                        ((float(row['ra'])-ra)*np.cos(np.radians(dec))*60)**2 +
                        ((float(row['dec'])-dec)*60)**2)
                    print(f'      #{i+1}: r={float(row["r"]):.2f}, '
                          f'z={float(row["z_spec"]):.4f}, '
                          f'offset={off:.2f}¥'')
            else:
                print(f'    No spectroscopic members found')
                print(f'    -&gt; Using original center as fallback')
                results[cl_name] = {
                    'bcg_ra': ra, 'bcg_dec': dec,
                    'offset_arcmin': 0, 'offset_Mpc': 0,
                    'fallback': True,
                }

        except Exception as e:
            print(f'    SDSS query failed: {e}')
            results[cl_name] = {
                'bcg_ra': ra, 'bcg_dec': dec,
                'offset_arcmin': 0, 'offset_Mpc': 0,
                'error': str(e),
            }

    return results


# ============================================================
```

```
# Fix 2: シェア計算 (CSVカラム自動検出 + デバッグ)
# ============================================================
def find_shear_files():
    """既存のシェアCSVファイルを探索"""
    candidates = []
    for d in [OUTDIR, Path('.'), Path('..'), Path('cluster_stack_output'),
              Path('probes_vbar_output')]:
        if d.exists():
            for f in d.glob('*.csv'):
                if any(k in f.name.lower() for k in ['shear', 'source', 'photoz',
                                                     'background', 'cl1', 'weak']):
                    candidates.append(f)
    return candidates


def inspect_csv(filepath):
    """CSVのカラム名と先頭行を調査"""
    print(f'¥n  Inspecting: {filepath}')
    print(f'    Size: {filepath.stat().st_size:,} bytes')

    with open(filepath, 'r', encoding='utf-8-sig') as f:
        reader = csv.reader(f)
        header = next(reader, None)
        if header:
            print(f'    Columns ({len(header)}): {header[:15]}')
            # 先頭3行
            for i, row in enumerate(reader):
                if i &gt;= 3:
                    break
                print(f'    Row {i}: {row[:10]}')
            return header
    return None


def compute_shear_profile_v2(filepath, ra_c, dec_c, z_cl):
    """修正版シェアプロファイル計算"""
    # カラム名の候補マッピング
    col_maps = {
        'ra':  ['i_ra', 'ra', 'RA', 'RAJ2000', 'ra_gal', 'alpha'],
        'dec': ['i_dec', 'dec', 'DEC', 'DEJ2000', 'dec_gal', 'delta'],
        'e1':  ['e1', 'e1_regauss', 'g1', 'e1_hsm', 'ishape_hsm_regauss_e1'],
        'e2':  ['e2', 'e2_regauss', 'g2', 'e2_hsm', 'ishape_hsm_regauss_e2'],
        'w':   ['weight', 'w', 'ishape_hsm_regauss_derived_shape_weight', 'wt'],
        'zph': ['photoz_mean', 'z_phot', 'photo_z', 'z_best', 'photoz_best',
                'pz_best_z', 'mizuki_photoz_best'],
    }

    # ヘッダ読み込み
    with open(filepath, 'r', encoding='utf-8-sig') as f:
        reader = csv.DictReader(f)
        headers = reader.fieldnames
        print(f'    CSV columns: {headers[:15]}')

        # カラムマッチ
        matched = {}
        for key, candidates in col_maps.items():
            for c in candidates:
                if c in headers:
                    matched[key] = c
                    break
            if key not in matched:
                # 部分一致
                for c in candidates:
                    for h in headers:
                        if c.lower() in h.lower():
                            matched[key] = h
                            break
                    if key in matched:
                        break

        print(f'    Matched columns: {matched}')

        missing = [k for k in ['ra', 'dec', 'e1', 'e2'] if k not in matched]
        if missing:
            print(f'    ERROR: Missing required columns: {missing}')
            print(f'    Available: {headers}')
            return None

        # デフォルト値
        if 'w' not in matched:
            print(f'    Warning: no weight column, using w=1')
        if 'zph' not in matched:
            print(f'    Warning: no photo-z column, using z=1.0')
        # データ読み込み
```

```
    f.seek(0)
    reader = csv.DictReader(f)

    gal_data = []
    n_read = 0
    n_skip_parse = 0
    n_skip_zph = 0

    for row in reader:
        n_read += 1
        try:
            ra_g = float(row[matched['ra']])
            dec_g = float(row[matched['dec']])
            e1 = float(row[matched['e1']])
            e2 = float(row[matched['e2']])
            w = float(row[matched['w']]) if 'w' in matched else 1.0
            zph = float(row[matched['zph']]) if 'zph' in matched else 1.0

            if w &lt;= 0:
                continue
            if zph &lt;= z_cl + 0.1:
                n_skip_zph += 1
                continue

            gal_data.append((ra_g, dec_g, e1, e2, w, zph))
        except (ValueError, KeyError):
            n_skip_parse += 1

print(f'    Read: {n_read}, Valid: {len(gal_data)}, '
      f'Skip(z): {n_skip_zph}, Skip(parse): {n_skip_parse}')

if len(gal_data) &lt; 50:
    print(f'    ERROR: Too few galaxies ({len(gal_data)})')
    return None

# 角度距離
from scipy.integrate import quad
def D_A(z):
    f = lambda zp: 1.0 / np.sqrt(Omega_m*(1+zp)**3 + Omega_L)
    chi, _ = quad(f, 0, z)
    return c_light / H0 * chi / (1+z)

D_l = D_A(z_cl)
print(f'    D_A(z={z_cl}) = {D_l:.1f} Mpc')
arcmin_to_Mpc = D_l * np.pi / (180 * 60)
print(f'    1 arcmin = {arcmin_to_Mpc:.4f} Mpc')

# ビニング
R_bins_Mpc = np.array([0.1, 0.2, 0.3, 0.5, 0.7, 1.0, 1.5, 2.0, 3.0, 5.0])
R_mid = np.sqrt(R_bins_Mpc[:-1] * R_bins_Mpc[1:])
n_bins = len(R_mid)

gt_sum = np.zeros(n_bins)
gx_sum = np.zeros(n_bins)
w_sum = np.zeros(n_bins)
n_count = np.zeros(n_bins, dtype=int)

for ra_g, dec_g, e1, e2, w, zph in gal_data:
    dra = (ra_g - ra_c) * np.cos(np.radians(dec_c))
    ddec = dec_g - dec_c
    theta_deg = np.sqrt(dra**2 + ddec**2)
    R_Mpc = theta_deg * D_l * np.pi / 180

    if R_Mpc &lt; R_bins_Mpc[0] or R_Mpc &gt;= R_bins_Mpc[-1]:
        continue

    phi = np.arctan2(ddec, dra)
    gt = -(e1 * np.cos(2*phi) + e2 * np.sin(2*phi))
    gx = +(e2 * np.cos(2*phi) - e1 * np.sin(2*phi))

    idx = np.searchsorted(R_bins_Mpc, R_Mpc) - 1
    if 0 &lt;= idx &lt; n_bins:
        gt_sum[idx] += gt * w
        gx_sum[idx] += gx * w
        w_sum[idx] += w
        n_count[idx] += 1

# 加重平均
mask = w_sum &gt; 0
gamma_t = np.zeros(n_bins)
gamma_x = np.zeros(n_bins)
e_gamma = np.full(n_bins, np.inf)

gamma_t[mask] = gt_sum[mask] / w_sum[mask]
```

```
    gamma_x[mask] = gx_sum[mask] / w_sum[mask]

    # シェア分散 (shape noise ~= 0.26 per component)
    sigma_e = 0.26
    e_gamma[mask] = sigma_e / np.sqrt(n_count[mask])

    # デバッグ出力
    print(f'¥n    Radial profile:')
    print(f'    {"R[Mpc]":&gt;8} {"N":&gt;6} {"gamma_t":&gt;10} {"gamma_x":&gt;10} {"S/N":&gt;8}')
    for i in range(n_bins):
        sn = gamma_t[i] / e_gamma[i] if e_gamma[i] &lt; np.inf else 0
        print(f'    {R_mid[i]:8.3f} {n_count[i]:6d} {gamma_t[i]:10.5f} '
              f'{gamma_x[i]:10.5f} {sn:8.2f}')

    total_sn = np.sqrt(np.sum((gamma_t[mask] / e_gamma[mask])**2))
    gx_median = np.median(np.abs(gamma_x[mask])) if np.any(mask) else 0

    print(f'¥n    Total S/N = {total_sn:.1f}')
    print(f'    |gamma_x| median = {gx_median:.5f}')

    return {
        'R_Mpc': R_mid,
        'gamma_t': gamma_t,
        'gamma_x': gamma_x,
        'e_gamma': e_gamma,
        'n_count': n_count,
        'S_N': total_sn,
    }


# ============================================================
# Main
# ============================================================
def main():
    print('=' * 60)
    print('Cluster Stack v3 Fix')
    print('=' * 60)

    # Step 1: BCG (修正版)
    bcg = bcg_identification()

    # Step 2: シェアファイル探索
    print(f'¥n{"="*60}')
    print('Shear File Search')
    print(f'{"="*60}')

    shear_files = find_shear_files()
    print(f'  Found {len(shear_files)} candidate files:')
    for f in shear_files:
        print(f'    {f}')

    # Step 3: 各ファイルを調査
    if shear_files:
        for sf in shear_files:
            header = inspect_csv(sf)

    # Step 4: cl1のシェアプロファイル計算 (BCG中心 vs 元中心)
    # cl1用のファイルを見つける
    cl1_file = None
    for sf in shear_files:
        if 'cl1' in sf.name.lower() or 'source' in sf.name.lower():
            cl1_file = sf
            break
    if cl1_file is None and shear_files:
        cl1_file = shear_files[0]  # 最初のファイルを試す

    if cl1_file:
        cl1_bcg = bcg.get('cl1', {})

        # (A) 元中心でのプロファイル
        print(f'¥n{"="*60}')
        print(f'cl1 Shear Profile: Original Center ({CLUSTERS["cl1"]["ra"]}, {CLUSTERS["cl1"]["dec"]})')
        print(f'{"="*60}')
        prof_orig = compute_shear_profile_v2(
            cl1_file, CLUSTERS['cl1']['ra'], CLUSTERS['cl1']['dec'], CLUSTERS['cl1']['z'])

        # (B) BCG中心でのプロファイル
        if not cl1_bcg.get('fallback') and cl1_bcg.get('bcg_ra'):
            print(f'¥n{"="*60}')
            print(f'cl1 Shear Profile: BCG Center ({cl1_bcg["bcg_ra"]:.5f}, {cl1_bcg["bcg_dec"]:.5f})')
            print(f'{"="*60}')
            prof_bcg = compute_shear_profile_v2(
                cl1_file, cl1_bcg['bcg_ra'], cl1_bcg['bcg_dec'], CLUSTERS['cl1']['z'])
            # 比較
```

```
            if prof_orig and prof_bcg:
                print(f'¥n  Comparison:')
                print(f'    Original center: S/N = {prof_orig["S_N"]:.1f}')
                print(f'    BCG center:      S/N = {prof_bcg["S_N"]:.1f}')
                print(f'    -&gt; {"BCG is better" if prof_bcg["S_N"] &gt; prof_orig["S_N"] else "Original is better"}')
    else:
        print('¥n  No shear data file found.')
        print('  Expected files: cl1_sources_photoz.csv or similar')
        print('  Please provide the filename or path.')

    # 結果保存
    summary = {
        'bcg': {k: v for k, v in bcg.items()},
    }
    with open(OUTDIR / 'v3_fix_results.json', 'w') as f:
        json.dump(summary, f, indent=2, default=str)


if __name__ == '__main__':
    main()
```

probes_vbar_v2.py

章: 中間版(v2_localに置換)

目的: PROBES V_bar v2: WHISP HI + S4G + Yd最適化(VizieR版)

解析対象データ: VizieR(取得失敗)

主要結果: VizieRテーブル名誤りで取得失敗。v2_localでHIソース探索に切替。

ソースコード (603 lines, 20,145 bytes)

```
#!/usr/bin/env python3
"""
probes_vbar_v2.py
=================
PROBES自前V_bar構築パイプライン v2

第26章の知見を反映:
  - Y_d を銀河ごとに最適化（RC全体フィット）
  - 実測HI面密度プロファイル（WHISP優先）
  - S4G 3.6μm表面輝度プロファイル

データソース:
  1. PROBES V_obs(R): VizieR J/ApJS/256/33 (Stone+2021) — 取得済み
  2. S4G 3.6μm: VizieR J/ApJS/219/4 (Salo+2015) — Re, Tmag, n
     + IRSA S4G P3 profiles (Muñoz-Mateos+2015) — radial SB profiles
  3. HI面密度:
     a) WHISP: VizieR J/A+A/526/A118 (Swaters+2002) — ~300銀河 HI RC
     b) THINGS: VizieR J/AJ/136/2648 (de Blok+2008) — 19銀河 分解RC
     c) LITTLE THINGS: 取得済み (Oh+2015)

パイプライン:
  Step 0: 全データ取得 (VizieR + IRSA)
  Step 1: PROBES x (S4G or WISE) x (WHISP or THINGS) 3重クロスマッチ
  Step 2: V_disk(R) 計算 (Freeman disk, S4Gパラメータ or プロファイル)
  Step 3: V_gas(R) 計算 (HI面密度 x 1.33 He補正)
  Step 4: Y_d 個別最適化: min chi²(V_obs - V_bar(Y_d))
  Step 5: g_c測定 (外側1/3中央値, Y_d=最適値)
  Step 6: G×Σ☒ = V_flat²/h_R (Y_d非依存)
  Step 7: α検定 + SPARC結果との比較

実行: uv run --with scipy --with matplotlib --with astropy --with requests python probes_vbar_v2.py

注意: ネットワークアクセスが必要 (VizieR, IRSA) 。Claude Code で実行。
"""

import numpy as np
import sys
import json
import os
from pathlib import Path
from scipy.optimize import minimize_scalar
from scipy.special import i0, i1, k0, k1
from scipy.stats import linregress, t as tdist
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib import font_manager as _fm
for _fp in ['/usr/share/fonts/opentype/ipafont-gothic/ipag.ttf',
            '/usr/share/fonts/opentype/ipafont-gothic/ipagp.ttf']:
    try: _fm.fontManager.addfont(_fp)
    except: pass
plt.rcParams['font.family'] = 'IPAGothic'
plt.rcParams['axes.unicode_minus'] = False

# === 物理定数 ===
G_SI = 6.674e-11
Msun = 1.989e30
pc = 3.086e16
kpc = 1e3 * pc
a0 = 1.2e-10
M_sun_W1 = 3.24  # 3.6μm Vega太陽絶対等級

OUTDIR = Path("probes_vbar_v2_output")
OUTDIR.mkdir(exist_ok=True)


# ============================================================
# データ取得
# ============================================================
def fetch_vizier_tsv(source, out_params, max_rows=999999, outfile=None):
```

```
    """VizieR汎用取得"""
    import requests
    url = "https://vizier.cds.unistra.fr/viz-bin/ase/tsv"
    params = {
        "-source": source,
        "-out.max": str(max_rows),
        "-out": out_params,
        "-out.form": "tsv",
    }
    print(f'  VizieR: {source} ...')
    r = requests.get(url, params=params, timeout=120)
    if outfile:
        Path(outfile).write_text(r.text)
    return r.text


def parse_tsv(text, required_cols=None):
    """TSVパース (ヘッダ自動検出) """
    lines = [l for l in text.strip().split('¥n') if l and not l.startswith('#')]
    header_idx = None
    for i, l in enumerate(lines):
        if required_cols and all(c in l for c in required_cols[:2]):
            header_idx = i
            break
        elif not l.startswith('-') and '¥t' in l and i &lt; 10:
            header_idx = i
            break
    if header_idx is None:
        return [], []
    headers = [h.strip() for h in lines[header_idx].split('¥t')]
    data = []
    for l in lines[header_idx+1:]:
        if l.startswith('-') or not l.strip():
            continue
        cols = l.split('¥t')
        if len(cols) &gt;= len(headers):
            data.append(dict(zip(headers, [c.strip() for c in cols])))
    return headers, data


def step0_fetch_all():
    """全データ取得"""
    print('=' * 60)
    print('Step 0: Data Retrieval')
    print('=' * 60)

    results = {}

    # --- PROBES RC ---
    print('¥n[PROBES RC]')
    txt = fetch_vizier_tsv(
        "J/ApJS/256/33/table2",
        "Name,Rad,Vrot,e_Vrot",
        outfile=OUTDIR / "probes_rc.tsv")
    _, rc_rows = parse_tsv(txt, ['Name', 'Rad'])

    rc_data = {}
    for row in rc_rows:
        try:
            name = row.get('Name', '').strip()
            rad = float(row.get('Rad', '0'))
            vrot = float(row.get('Vrot', '0'))
            ev = float(row.get('e_Vrot', '5')) if row.get('e_Vrot', '').strip() else 5.0
            if name:
                if name not in rc_data:
                    rc_data[name] = {'R': [], 'V': [], 'eV': []}
                rc_data[name]['R'].append(rad)
                rc_data[name]['V'].append(vrot)
                rc_data[name]['eV'].append(ev)
        except ValueError:
            pass

    for name in rc_data:
        for k in rc_data[name]:
            rc_data[name][k] = np.array(rc_data[name][k])
    print(f'  -&gt; {len(rc_data)} galaxies')
    results['probes_rc'] = rc_data

    # --- PROBES master (距離) ---
    print('¥n[PROBES Master]')
    txt = fetch_vizier_tsv(
        "J/ApJS/256/33/table1",
        "Name,Dist,incl,Vmax",
        outfile=OUTDIR / "probes_master.tsv")
```

```
_, master_rows = parse_tsv(txt, ['Name', 'Dist'])

dist_map = {}
for row in master_rows:
    try:
        name = row.get('Name', '').strip()
        dist = float(row.get('Dist', '0'))
        if name and dist &gt; 0:
            dist_map[name] = dist
    except ValueError:
        pass
print(f'  -&gt; {len(dist_map)} galaxies with distance')
results['dist'] = dist_map

# --- S4G (Salo+2015) ---
print('¥n[S4G Salo+2015]')
txt = fetch_vizier_tsv(
    "J/ApJS/219/4/table1",
    "Name,Re,Tmag,n,T,PA,b/a",
    outfile=OUTDIR / "s4g_params.tsv")
_, s4g_rows = parse_tsv(txt, ['Name', 'Re'])

s4g = {}
for row in s4g_rows:
    try:
        name = row.get('Name', '').strip()
        Re = float(row.get('Re', '0'))
        Tmag = float(row.get('Tmag', '99'))
        n = float(row.get('n', '1')) if row.get('n', '').strip() else 1.0
        if name and Re &gt; 0 and Tmag &lt; 30:
            s4g[name] = {'Re': Re, 'Tmag': Tmag, 'n': n}
    except ValueError:
        pass
print(f'  -&gt; {len(s4g)} galaxies')
results['s4g'] = s4g

# --- WHISP HI rotation curves (Swaters+2002 or van der Hulst+2002) ---
print('¥n[WHISP HI]')
# Swaters+2002: J/A+A/390/829 (dwarf galaxies RC with HI)
# van Eymeren+2011: J/A+A/530/A29 (WHISP kinematic analysis)
# Try multiple WHISP-related catalogs
whisp_sources = [
    ("J/A+A/390/829", "Name,Rad,Vrot,e_Vrot,Vgas"),  # Swaters+2002
    ("J/AJ/141/193", "Name,Rad,Vobs,e_Vobs,Vgas,Vdisk"),  # Swaters+2011
]

whisp_rc = {}
for src, cols in whisp_sources:
    try:
        txt = fetch_vizier_tsv(src, cols, outfile=OUTDIR / f"whisp_{src.replace('/','-')}.tsv")
        _, rows = parse_tsv(txt)
        if rows:
            print(f'  {src}: {len(rows)} rows')
            # パース（カラム名はソースにより異なる）
            for row in rows:
                name = (row.get('Name', '') or row.get('Galaxy', '')).strip()
                try:
                    rad = float(row.get('Rad', row.get('R', '0')))
                    vgas_str = row.get('Vgas', row.get('VHI', ''))
                    if name and vgas_str.strip():
                        vgas = float(vgas_str)
                        if name not in whisp_rc:
                            whisp_rc[name] = {'R': [], 'Vgas': []}
                        whisp_rc[name]['R'].append(rad)
                        whisp_rc[name]['Vgas'].append(vgas)
                except ValueError:
                    pass
    except Exception as e:
        print(f'  {src}: failed ({e})')

for name in whisp_rc:
    for k in whisp_rc[name]:
        whisp_rc[name][k] = np.array(whisp_rc[name][k])
print(f'  -&gt; {len(whisp_rc)} galaxies with HI data')
results['whisp'] = whisp_rc

# --- THINGS decomposed RC (de Blok+2008) ---
print('¥n[THINGS de Blok+2008]')
txt = fetch_vizier_tsv(
    "J/AJ/136/2648/table5",
    "Name,Rad,Vobs,Vgas,Vdis,Vbul",
    outfile=OUTDIR / "things_rc.tsv")
_, things_rows = parse_tsv(txt)
things_rc = {}
```

```
    for row in things_rows:
        name = (row.get('Name', '') or row.get('Galaxy', '')).strip()
        try:
            rad = float(row.get('Rad', '0'))
            vgas_str = row.get('Vgas', '')
            vdisk_str = row.get('Vdis', row.get('Vdisk', ''))
            if name and vgas_str.strip():
                vgas = float(vgas_str)
                vdisk = float(vdisk_str) if vdisk_str.strip() else 0.0
                if name not in things_rc:
                    things_rc[name] = {'R': [], 'Vgas': [], 'Vdisk': []}
                things_rc[name]['R'].append(rad)
                things_rc[name]['Vgas'].append(vgas)
                things_rc[name]['Vdisk'].append(vdisk)
        except ValueError:
            pass

    for name in things_rc:
        for k in things_rc[name]:
            things_rc[name][k] = np.array(things_rc[name][k])
    print(f'  -&gt; {len(things_rc)} galaxies with decomposed RC')
    results['things'] = things_rc

    return results


# ============================================================
# V_disk計算
# ============================================================
def compute_vdisk_freeman(R_kpc, h_kpc, I0_Lpc2, Yd):
    """Freeman (1970) thin disk V_disk(R)"""
    y = R_kpc / (2.0 * h_kpc)
    y = np.clip(y, 1e-6, 50)
    bessel = i0(y)*k0(y) - i1(y)*k1(y)

    Sigma0_SI = Yd * I0_Lpc2 * Msun / (pc**2)
    h_m = h_kpc * kpc
    V2 = 4 * np.pi * G_SI * Sigma0_SI * h_m * y**2 * bessel
    return np.sqrt(np.maximum(V2, 0)) / 1e3  # km/s


def s4g_to_disk_params(s4g_entry, dist_Mpc):
    """S4Gパラメータからディスクパラメータを計算"""
    Re_arcsec = s4g_entry['Re']
    Tmag = s4g_entry['Tmag']
    n = s4g_entry['n']

    # Re -&gt; h
    bn = 2*n - 1.0/3 + 4.0/(405*n) if n &gt; 0.5 else 1.678
    h_arcsec = Re_arcsec / bn

    dist_kpc = dist_Mpc * 1e3
    h_kpc = h_arcsec * dist_kpc * np.pi / (180 * 3600)

    if h_kpc &lt; 0.01 or h_kpc &gt; 100:
        return None

    # 中心表面輝度
    area = 2 * np.pi * Re_arcsec**2
    if area &lt;= 0:
        return None
    mu0 = Tmag + 2.5 * np.log10(area) + 0.7
    I0_Lpc2 = 10**(-0.4 * (mu0 - M_sun_W1 - 21.572))

    return {'h_kpc': h_kpc, 'I0_Lpc2': I0_Lpc2}


# ============================================================
# Υ_d最適化
# ============================================================
def optimize_Yd(R_kpc, V_obs, eV, V_disk_unit, V_gas):
    """
    Υ_dを最適化: V_bar(Yd) = sqrt(Yd * V_disk_unit² + V_gas²)
    V_disk_unit: Yd=1でのV_disk (Ydを含まない)
    """
    def chi2(Yd):
        V_bar = np.sqrt(np.maximum(Yd * V_disk_unit**2 + V_gas**2, 0))
        resid = (V_obs - V_bar) / np.maximum(eV, 1.0)
        return np.sum(resid**2) / max(len(V_obs) - 1, 1)

    try:
        res = minimize_scalar(chi2, bounds=(0.05, 2.0), method='bounded')
        if res.fun &lt; 50:
            return res.x, res.fun
```

```
    except:
        pass
    return None, None


# ============================================================
# g_c測定 + α検定
# ============================================================
def measure_gc_gs0(R_kpc, V_obs, V_bar, V_disk_unit, Yd_opt):
    """g_c (外側1/3中央値) + G*Sigma_0 (Yd非依存)"""
    n = len(R_kpc)
    if n &lt; 5:
        return None

    outer = slice(2*n//3, n)
    R_m = R_kpc[outer] * kpc
    gc_vals = (V_obs[outer]*1e3)**2/R_m - (V_bar[outer]*1e3)**2/R_m
    gc = np.median(gc_vals)

    if gc &lt;= 0:
        return None

    # V_flat (Yd非依存)
    V_flat = np.median(V_obs[outer])

    # h_R (V_disk_unitのピーク / 2.15)
    vds = np.sqrt(Yd_opt) * np.abs(V_disk_unit)
    i_pk = np.argmax(vds)
    if i_pk == 0:
        i_pk = 1
    r_pk = R_kpc[i_pk]
    if r_pk &gt;= R_kpc.max() * 0.9 or r_pk &lt; 0.01:
        return None  # edgeフィルタ (元パイプライン準拠)
    h_R = r_pk / 2.15

    GS0 = (V_flat * 1e3)**2 / (h_R * kpc)

    return {'gc': gc, 'GS0': GS0, 'V_flat': V_flat, 'h_R': h_R, 'Yd': Yd_opt}


def alpha_fit(gc_arr, gs0_arr):
    log_gc = np.log10(gc_arr)
    log_gs = np.log10(gs0_arr)
    x = log_gs - np.log10(a0)
    y = log_gc
    mask = np.isfinite(x) &amp;amp; np.isfinite(y)
    x, y = x[mask], y[mask]
    if len(x) &lt; 5:
        return None
    sl, ic, r, p, se = linregress(x, y)
    t_stat = (sl - 0.5) / se
    p05 = 2 * tdist.sf(abs(t_stat), df=len(x)-2)
    return {'alpha': sl, 'e_alpha': se, 'p05': p05, 'r': r, 'N': len(x)}


# ============================================================
# 名前正規化
# ============================================================
def norm_name(n):
    import re
    return re.sub(r'[¥s¥-_]', '', n.upper())


# ============================================================
# SPARC除外リスト
# ============================================================
SPARC_NAMES = {norm_name(n) for n in [
    "NGC7331","NGC2403","NGC3198","NGC2841","NGC6946","NGC3521",
    "NGC925","NGC2976","NGC4736","NGC5055","NGC7793","NGC3031",
    "NGC4826","NGC2903","NGC4559","NGC1003","NGC3109","NGC4395",
    "DDO154","DDO168","IC2574","NGC1560","UGC2259","DDO052","DDO087",
    "DDO101","DDO126","DDO133","DDO047","DDO050","DDO053","DDO046",
    "CVnIdwA","Haro29","Haro36","WLM","NGC6822","NGC3738","NGC4214",
    "NGC1569","NGC2366","SagDIG","DDO210","DDO216","UGC8508","UGCA281",
]}


# ============================================================
# メインパイプライン
# ============================================================
def main():
    print('=' * 60)
    print('PROBES V_bar Pipeline v2')
    print('(Yd individual optimization + HI profiles)')
```

```
print('=' * 60)

# Step 0: データ取得
data = step0_fetch_all()
rc_data = data['probes_rc']
dist_map = data['dist']
s4g = data['s4g']
whisp = data['whisp']
things = data['things']

# Step 1: クロスマッチ
print(f'¥n{"="*60}')
print('Step 1: Cross-match')
print(f'{"="*60}')

# 名前正規化マップ
rc_norm = {norm_name(n): n for n in rc_data}
s4g_norm = {norm_name(n): n for n in s4g}
whisp_norm = {norm_name(n): n for n in whisp}
things_norm = {norm_name(n): n for n in things}
dist_norm = {norm_name(n): n for n in dist_map}

# HI統合 (THINGS優先 &gt; WHISP)
hi_all = {}
for nn, orig in whisp_norm.items():
    hi_all[nn] = {'source': 'WHISP', 'data': whisp[orig]}
for nn, orig in things_norm.items():
    hi_all[nn] = {'source': 'THINGS', 'data': things[orig]}  # THINGS上書き

# 3重マッチ: PROBES ∩ S4G ∩ HI
match_3 = set(rc_norm.keys()) &amp; set(s4g_norm.keys()) &amp; set(hi_all.keys()) &amp; set(dist_norm.keys())
# 2重マッチ: PROBES ∩ S4G (HIなし — V_gas推定で代替)
match_2 = (set(rc_norm.keys()) &amp; set(s4g_norm.keys()) &amp; set(dist_norm.keys())) - match_3

print(f'  PROBES: {len(rc_norm)}')
print(f'  S4G: {len(s4g_norm)}')
print(f'  HI (WHISP+THINGS): {len(hi_all)}')
print(f'  PROBES x S4G x HI x Dist: {len(match_3)}')
print(f'  PROBES x S4G x Dist (no HI): {len(match_2)}')

# SPARC除外
match_3_indep = match_3 - SPARC_NAMES
match_2_indep = match_2 - SPARC_NAMES
print(f'  3-way independent of SPARC: {len(match_3_indep)}')
print(f'  2-way independent of SPARC: {len(match_2_indep)}')

# Step 2-6: V_bar構築 + α測定
print(f'¥n{"="*60}')
print('Step 2-6: V_bar construction + alpha test')
print(f'{"="*60}')

results_3way = []  # 3重マッチ (実測HI)
results_2way = []  # 2重マッチ (推定HI)
Yd_list = []

for nn in sorted(match_3_indep | match_2_indep):
    rc_name = rc_norm[nn]
    s4g_name = s4g_norm[nn]
    dist_name = dist_norm[nn]
    dist_Mpc = dist_map[dist_name]

    rc = rc_data[rc_name]
    R_arcsec = rc['R']
    V_obs = rc['V']
    eV = rc['eV']

    # R_arcsec -&gt; R_kpc
    dist_kpc = dist_Mpc * 1e3
    R_kpc = R_arcsec * dist_kpc * np.pi / (180 * 3600)

    if len(R_kpc) &lt; 8:
        continue

    # V_disk (S4G Freeman disk, Yd=1)
    dp = s4g_to_disk_params(s4g[s4g_name], dist_Mpc)
    if dp is None:
        continue

    V_disk_unit = compute_vdisk_freeman(R_kpc, dp['h_kpc'], dp['I0_Lpc2'], Yd=1.0)

    if len(V_disk_unit) != len(V_obs):
        continue

    # V_gas
```

```
    has_hi = nn in hi_all
    if has_hi:
        hi = hi_all[nn]['data']
        # HIのRをPROBESのRに補間
        hi_R = hi.get('R', np.array([]))
        hi_Vgas = hi.get('Vgas', np.array([]))
        if len(hi_R) &gt;= 3 and len(hi_Vgas) &gt;= 3:
            from scipy.interpolate import interp1d
            try:
                f_gas = interp1d(hi_R, hi_Vgas, kind='linear',
                                 fill_value='extrapolate', bounds_error=False)
                V_gas = np.maximum(f_gas(R_kpc), 0)
            except:
                V_gas = np.zeros_like(R_kpc)
                has_hi = False
        else:
            V_gas = np.zeros_like(R_kpc)
            has_hi = False

    if not has_hi:
        # f_gas=15%推定 (矮小銀河は高め、大質量は低め)
        V_flat_est = np.median(V_obs[2*len(V_obs)//3:])
        if V_flat_est &lt; 80:
            f_gas = 0.30  # 矮小銀河
        elif V_flat_est &lt; 150:
            f_gas = 0.15  # 中間
        else:
            f_gas = 0.05  # 大質量
        V_gas = np.sqrt(f_gas) * V_disk_unit

    # Υ_d最適化
    Yd_opt, chi2_dof = optimize_Yd(R_kpc, V_obs, eV, V_disk_unit, V_gas)
    if Yd_opt is None:
        continue

    # V_bar (最適Υ_d)
    V_bar = np.sqrt(np.maximum(Yd_opt * V_disk_unit**2 + V_gas**2, 0))

    # g_c + GS0 測定
    result = measure_gc_gs0(R_kpc, V_obs, V_bar, V_disk_unit, Yd_opt)
    if result is None:
        continue

    result['name'] = rc_name
    result['has_hi'] = has_hi
    result['chi2_dof'] = chi2_dof
    result['dist'] = dist_Mpc

    if has_hi:
        results_3way.append(result)
    else:
        results_2way.append(result)
    Yd_list.append(Yd_opt)

print(f'¥n  3-way results (real HI): {len(results_3way)}')
print(f'  2-way results (est HI):  {len(results_2way)}')
print(f'  Total: {len(results_3way) + len(results_2way)}')

# Υ_d分布
if Yd_list:
    Yd_arr = np.array(Yd_list)
    print(f'¥n  Yd distribution:')
    print(f'    median={np.median(Yd_arr):.3f}, mean={np.mean(Yd_arr):.3f}, '
          f'std={np.std(Yd_arr):.3f}')
    print(f'    range=[{np.min(Yd_arr):.3f}, {np.max(Yd_arr):.3f}]')
    print(f'    IQR=[{np.percentile(Yd_arr,25):.3f}, {np.percentile(Yd_arr,75):.3f}]')

# Step 7: α検定
print(f'¥n{"="*60}')
print('Step 7: Alpha Test')
print(f'{"="*60}')

for label, res_list in [
    ('3-way (real HI)', results_3way),
    ('2-way (est HI)', results_2way),
    ('All combined', results_3way + results_2way),
]:
    if len(res_list) &lt; 5:
        print(f'  {label}: N={len(res_list)} &lt; 5, skip')
        continue
    gc_arr = np.array([r['gc'] for r in res_list])
    gs0_arr = np.array([r['GS0'] for r in res_list])
    fit = alpha_fit(gc_arr, gs0_arr)
    if fit:
```

```
            ok = 'YES' if fit['p05'] &gt; 0.05 else 'no'
            print(f'  {label}: N={fit["N"]}, alpha={fit["alpha"]:.3f}+/-{fit["e_alpha"]:.3f}, '
                  f'p(0.5)={fit["p05"]:.4f}, r={fit["r"]:.3f}, 0.5棄却不可={ok}')

    # 結果保存
    summary = {
        'n_3way': len(results_3way),
        'n_2way': len(results_2way),
        'Yd_stats': {
            'median': float(np.median(Yd_arr)) if Yd_list else None,
            'mean': float(np.mean(Yd_arr)) if Yd_list else None,
        },
        'results_3way': [{'name': r['name'], 'gc': float(r['gc']),
                          'GS0': float(r['GS0']), 'Yd': float(r['Yd'])}
                         for r in results_3way],
        'results_2way_count': len(results_2way),
    }
    with open(OUTDIR / 'vbar_v2_summary.json', 'w') as f:
        json.dump(summary, f, indent=2)

    print(f'¥nResults: {OUTDIR / "vbar_v2_summary.json"}')
    print(f'¥nNext: If N &gt;= 20, this constitutes an independent verification.')
    print(f'      Compare Yd distribution with SPARC (median~0.50, IQR 0.38-0.66).')


if __name__ == '__main__':
    main()
```