KODE PYTHON SKRIPSI

October 2, 2025

[1]:

```
#!/usr/bin/env python3  # Menentukan interpreter Python versi 3
# -*- coding: utf-8 -*-  # Menentukan encoding file untuk mendukung karakter␣
 ↪Unicode

import numpy as np  # Mengimpor NumPy untuk operasi array numerik
import matplotlib.pyplot as plt  # Mengimpor Matplotlib untuk plotting dan␣
 ↪visualisasi
from scipy.optimize import curve_fit, root_scalar  # Mengimpor fungsi optimasi␣
 ↪dari SciPy untuk fitting dan pencarian akar
from scipy.integrate import quad  # Mengimpor fungsi integrasi numerik dari␣
 ↪SciPy
from scipy.special import iv, kv  # Mengimpor fungsi Bessel modified dari SciPy␣
 ↪untuk perhitungan disk
from functools import lru_cache  # Mengimpor decorator untuk caching fungsi␣
 ↪agar lebih efisien
from sklearn.metrics import r2_score  # Mengimpor metrik R² dari Scikit-learn␣
 ↪untuk evaluasi model
import scipy.stats as stats  # Mengimpor modul statistik dari SciPy untuk uji␣
 ↪signifikansi
import warnings  # Mengimpor modul warnings untuk mengelola peringatan

# ================================ KONFIGURASI GLOBAL␣
 ↪=====================================
warnings.filterwarnings('ignore', category=RuntimeWarning)  # Menekan␣
 ↪peringatan RuntimeWarning agar output bersih
warnings.filterwarnings('ignore', category=UserWarning)  # Menekan peringatan␣
 ↪UserWarning agar output bersih

# Konstanta fisika dan astronomi
G_ASTRO = 4.30091e-6  # Konstanta gravitasi dalam satuan astronomi (kpc, km/s,␣
 ↪M_sun)
A0_STD_SI = 1.2e-10  # Skala percepatan MOND standar dalam SI (m/s²)
A0_ARCTAN_SI = A0_STD_SI  # Skala percepatan MOND arctan sama dengan standar␣
 ↪dalam SI (m/s²)
KM2_S2_PER_KPC_TO_SI = 3.2408e-14  # Faktor konversi dari (km/s)²/kpc ke SI
A0_STD_ASTRO = A0_STD_SI / KM2_S2_PER_KPC_TO_SI  # Konversi A0 standar ke␣
 ↪satuan astronomi
```

```python
A0_ARCTAN_ASTRO = A0_ARCTAN_SI / KM2_S2_PER_KPC_TO_SI  # Konversi A0 arctan ke␣
↪satuan astronomi
KAPPA = 7.6695  # Konstanta untuk profil de Vaucouleurs bulge

# =================================== KELAS UTAMA␣
↪========================================
class GalaxyRotationAnalyzer:  # Mendefinisikan kelas utama untuk analisis␣
↪kurva rotasi galaksi
    """Kelas utama untuk analisis empiris kurva rotasi galaksi, membandingkan␣
↪MOND (standar dan arctangent) dengan ΛCDM (NFW dan Core)."""  # Docstring:␣
↪Deskripsi kelas

    def __init__(self):  # Metode inisialisasi kelas
        self.data = self.load_observational_data()  # Memuat data observasi␣
↪galaksi
        self.models = {  # Dictionary untuk menyimpan fungsi model
            'MOND Std': self.v_total_mond_std,  # Model MOND standar
            'MOND Arctan': self.v_total_mond_arctan,  # Model MOND arctan
            'ΛCDM NFW': self.v_total_nfw,  # Model ΛCDM dengan halo NFW
            'ΛCDM Core': self.v_total_core  # Model ΛCDM dengan halo Core
        }
        self.initial_params = self.set_initial_parameters()  # Mengatur␣
↪parameter awal untuk fitting

    # ============================== FUNGSI DATA OBSERVASI␣
↪================================
    @staticmethod  # Dekorator staticmethod: Fungsi ini tidak memerlukan␣
↪instance kelas
    def load_observational_data():  # Fungsi statis untuk memuat data observasi
        """Memuat data observasi kurva rotasi untuk Bima Sakti dan M31 dari␣
↪sumber empiris terkini."""  # Docstring: Deskripsi fungsi
        # Data Bima Sakti (Milky Way) - Digabungkan dari bagian 1 dan 2
        r_mw = np.concatenate([  # Menggabungkan array radius untuk Bima Sakti
            np.array([0.100, 0.110, 0.123, 0.133, 0.146, 0.161, 0.177, 0.195, 0.
↪214, 0.236,  # Bagian 1: Radius dalam kpc
                      0.258, 0.285, 0.314, 0.345, 0.380, 0.418, 0.459, 0.505, 0.
↪556, 0.612,
                      0.673, 0.741, 0.814, 0.895, 0.985, 1.083, 1.192, 1.311, 1.
↪442, 1.586,
                      1.745, 1.919, 2.111, 2.323, 2.555, 2.811, 3.091]),
            np.array([3.400, 3.740, 4.114, 4.526, 4.979, 5.476, 6.024, 6.626, 7.
↪289, 8.018,  # Bagian 2: Radius dalam kpc
                      8.820, 9.702, 10.672, 11.739, 12.913, 14.204, 15.625, 17.
↪187, 18.906,
                      20.797, 22.876, 25.164, 27.680, 30.448, 33.493, 36.842,␣
↪40.527, 44.579,
```

```
                     49.037, 53.941, 59.335, 65.268, 71.795, 78.975, 86.872,␣
↪95.561])
      ])
      v_mw = np.concatenate([  # Menggabungkan array kecepatan untuk Bima␣
↪Sakti
          np.array([144.9, 147.4, 150.4, 153.8, 158.9, 162.4, 180.1, 196.6,␣
↪213.6, 227.8,  # Bagian 1: Kecepatan dalam km/s
                     234.4, 244.1, 248.2, 250.2, 251.0, 250.7, 249.7, 248.0,␣
↪245.0, 243.2,
                     239.8, 235.3, 231.7, 227.8, 224.5, 221.7, 210.1, 216.8,␣
↪214.7, 212.7,
                     211.2, 209.5, 208.5, 208.2, 208.9, 210.4, 213.4]),
          np.array([217.2, 222.0, 226.6, 229.5, 231.6, 234.1, 237.2, 240.5,␣
↪241.1, 236.0,  # Bagian 2: Kecepatan dalam km/s
                     236.7, 234.5, 234.2, 237.1, 242.8, 248.5, 240.7, 246.2,␣
↪246.5, 243.9,
                     241.6, 243.7, 237.3, 229.6, 222.5, 218.0, 207.1, 201.5,␣
↪194.2, 189.8,
                     186.2, 184.7, 183.9, 181.4, 179.5, 167.7])
      ])
      sigma_v_mw = np.concatenate([  # Menggabungkan array kesalahan␣
↪kecepatan untuk Bima Sakti
          np.array([3.7, 4.2, 4.8, 6.1, 10.3, 16.1, 23.4, 27.1, 26.9, 22.7,␣
↪17.0,  # Bagian 1: Sigma v dalam km/s
                     11.8, 7.6, 4.7, 2.9, 2.1, 2.3, 2.9, 3.7, 4.6, 5.4, 6.7, 6.
↪5,
                     6.0, 5.2, 4.5, 4.0, 3.7, 3.4, 3.1, 2.9, 2.3, 1.8, 1.6, 2.
↪2, 3.6, 4.8]),
          np.array([5.9, 6.6, 5.7, 4.1, 4.3, 5.3, 5.7, 5.0, 4.1, 4.4, 5.1, 6.
↪0,  # Bagian 2: Sigma v dalam km/s
                     7.1, 9.8, 12.4, 13.3, 14.8, 17.4, 17.3, 17.5, 15.6, 15.2,␣
↪16.1,
                     15.3, 14.1, 14.0, 13.8, 12.7, 12.9, 11.1, 10.4, 9.6, 9.3,␣
↪13.0, 14.6, 16.5])
      ])

      # Data M31 (Andromeda) - Digabungkan dari bagian 1 dan 2
      r_m31 = np.concatenate([  # Menggabungkan array radius untuk M31
          np.array([0.100, 0.120, 0.144, 0.173, 0.207, 0.249, 0.299, 0.358, 0.
↪430, 0.516,  # Bagian 1: Radius dalam kpc
                     0.619, 0.743, 0.892, 1.070, 1.284, 1.541, 1.849, 2.219, 2.
↪662, 3.195,
                     3.834, 4.601, 5.521, 6.625, 7.950, 9.540, 11.448, 13.
↪737]),
          np.array([16.5, 19.8, 23.7, 28.5, 34.2, 41.0, 49.2, 59.1, 70.9, 85.
↪1,  # Bagian 2: Radius dalam kpc
```

```
                    102.1, 122.5, 147.0, 176.4, 211.6, 254.0, 304.8, 365.7,␣
↪438.9, 526.6,
                    632.0, 758.4, 910.0, 1092.1, 1310.5, 1572.6, 1887.1])
        ])
        v_m31 = np.concatenate([  # Menggabungkan array kecepatan untuk M31
            np.array([183.9, 190.7, 204.1, 207.6, 210.2, 213.5, 217.2, 219.7,␣
↪219.4, 216.2,  # Bagian 1: Kecepatan dalam km/s
                    213.7, 219.5, 221.3, 235.3, 242.8, 248.3, 252.9, 245.0,␣
↪239.0, 236.6,
                    228.4, 222.7, 229.1, 243.2, 255.4, 261.4, 263.5, 262.8]),
            np.array([257.3, 246.1, 236.5, 232.3, 233.5, 230.4, 237.1, 249.4,␣
↪218.5, 206.6,  # Bagian 2: Kecepatan dalam km/s
                    213.0, 197.4, 177.5, 165.3, 165.1, 160.8, 151.1, 122.0,␣
↪168.3, 200.5,
                    239.0, 248.1, 250.4, 241.9, 235.0, 332.7, 414.9])
        ])
        sigma_v_m31 = np.concatenate([  # Menggabungkan array kesalahan␣
↪kecepatan untuk M31
            np.array([15.0, 10.3, 22.6, 4.6, 3.9, 4.2, 3.8, 2.2, 2.2, 3.7, 10.
↪9, 22.6,  # Bagian 1: Sigma v dalam km/s
                    30.7, 55.7, 57.0, 52.4, 50.6, 35.2, 16.5, 9.2, 21.1, 30.
↪4, 29.0,
                    23.6, 18.5, 14.7, 11.4, 10.0]),
            np.array([12.2, 13.4, 11.7, 16.6, 27.4, 52.4, 109.4, 134.6, 132.4,␣
↪135.0,  # Bagian 2: Sigma v dalam km/s
                    132.5, 105.7, 81.8, 74.7, 76.9, 77.4, 74.8, 96.5, 125.7,␣
↪123.4,
                    147.8, 145.0, 141.4, 138.7, 103.5, 109.5, 69.6])
        ])

        return {  # Mengembalikan dictionary berisi data untuk kedua galaksi
            'Bima Sakti': {'r': r_mw, 'v': v_mw, 'sigma_v': sigma_v_mw},  #␣
↪Data Bima Sakti
            'M31': {'r': r_m31, 'v': v_m31, 'sigma_v': sigma_v_m31}  # Data M31
        }

    # ============================== FUNGSI MODEL KOMPONEN GALAKSI␣
↪================================
    @staticmethod  # Dekorator staticmethod
    def sigma_b(r, Sigma_be, a_b):  # Fungsi densitas permukaan bulge de␣
↪Vaucouleurs
        """Densitas permukaan bulge dengan profil de Vaucouleurs yang empiris.
↪"""  # Docstring
        return Sigma_be * np.exp(-KAPPA * ((r / a_b)**0.25 - 1))  # Rumus␣
↪densitas permukaan bulge
```

```
    @staticmethod  # Dekorator staticmethod
    def dsigma_b_dx(x, Sigma_be, a_b):  # Fungsi turunan densitas permukaan bulge
        """Turunan densitas permukaan bulge untuk integrasi densitas 3D."""  # Docstring
        return GalaxyRotationAnalyzer.sigma_b(x, Sigma_be, a_b) * (-KAPPA / (4 * a_b)) * (x / a_b)**(-0.75)  # Rumus turunan

    @lru_cache(maxsize=None)  # Dekorator caching untuk efisiensi
    def rho_b_scalar(self, r_val, Sigma_be, a_b):  # Fungsi densitas 3D bulge scalar (cached)
        """Densitas 3D bulge menggunakan integrasi numerik (di-cache untuk efisiensi)."""  # Docstring
        if r_val == 0:  # Kondisi khusus untuk r=0
            integrand = lambda x: self.dsigma_b_dx(x, Sigma_be, a_b) / x  # Integrand untuk r=0
            integral, _ = quad(integrand, 1e-10, np.inf, epsabs=1e-4, epsrel=1e-4, limit=100)  # Integrasi numerik
        else:  # Untuk r > 0
            integrand = lambda x: self.dsigma_b_dx(x, Sigma_be, a_b) / np.sqrt(x**2 - r_val**2)  # Integrand untuk r>0
            integral, _ = quad(integrand, r_val, np.inf, epsabs=1e-4, epsrel=1e-4, limit=100)  # Integrasi numerik
        return -1 / np.pi * integral  # Mengembalikan densitas 3D

    def rho_b(self, r, Sigma_be, a_b):  # Fungsi vektorisasi densitas 3D bulge
        """Vektorisasi densitas 3D bulge."""  # Docstring
        return np.array([self.rho_b_scalar(ri, Sigma_be, a_b) for ri in np.atleast_1d(r)])  # Vektorisasi dengan list comprehension

    @lru_cache(maxsize=None)  # Dekorator caching
    def M_b_scalar(self, r_val, Sigma_be, a_b):  # Fungsi massa terlingkup bulge scalar (cached)
        """Massa terlingkup bulge (di-cache)."""  # Docstring
        if r_val == 0:  # Kondisi untuk r=0
            return 0.0  # Massa nol di pusat
        integrand = lambda s: 4 * np.pi * s**2 * self.rho_b_scalar(s, Sigma_be, a_b)  # Integrand massa
        mass, _ = quad(integrand, 1e-10, r_val, epsabs=1e-4, epsrel=1e-4, limit=100)  # Integrasi massa
        return mass  # Mengembalikan massa terlingkup

    def M_b(self, r, Sigma_be, a_b):  # Fungsi vektorisasi massa terlingkup bulge
        """Vektorisasi massa terlingkup bulge."""  # Docstring
```

```
        return np.array([self.M_b_scalar(ri, Sigma_be, a_b) for ri in np.atleast_1d(r)])  # Vektorisasi

    def v_bulge_sq(self, r, Sigma_be, a_b):  # Fungsi kuadrat kecepatan rotasi bulge
        """Kuadrat kecepatan rotasi dari bulge."""  # Docstring
        r_arr = np.atleast_1d(r)  # Memastikan r adalah array 1D
        mass = self.M_b(r_arr, Sigma_be, a_b)  # Menghitung massa terlingkup
        v_sq = np.zeros_like(r_arr)  # Inisialisasi array v² nol
        mask = r_arr > 0  # Mask untuk r > 0
        v_sq[mask] = G_ASTRO * mass[mask] / r_arr[mask]  # v² = G * M / r untuk r > 0
        return v_sq  # Mengembalikan v²

    @staticmethod  # Dekorator staticmethod
    def v_disk_sq(r, Sigma_d0, a_d):  # Fungsi kuadrat kecepatan rotasi disk eksponensial
        """Kuadrat kecepatan rotasi dari disk eksponensial."""  # Docstring
        r_arr = np.atleast_1d(r)  # Memastikan r adalah array 1D
        v_sq = np.zeros_like(r_arr)  # Inisialisasi array v² nol
        mask = r_arr > 0  # Mask untuk r > 0
        r_nonzero = r_arr[mask]  # Radius non-nol
        y = r_nonzero / (2 * a_d)  # Parameter y untuk fungsi Bessel
        bessel_term = iv(0, y) * kv(0, y) - iv(1, y) * kv(1, y)  # Istilah Bessel untuk disk
        v_sq[mask] = np.pi * G_ASTRO * Sigma_d0 * (r_nonzero**2 / a_d) * bessel_term  # Rumus v² disk
        return v_sq  # Mengembalikan v²

    @staticmethod  # Dekorator staticmethod
    def v_gas_sq(r):  # Fungsi kuadrat kecepatan rotasi gas (diabaikan)
        """Kuadrat kecepatan rotasi dari gas (diabaikan dalam analisis ini)."""  # Docstring
        return np.zeros_like(np.atleast_1d(r))  # Mengembalikan array nol (gas diabaikan)

    @staticmethod  # Dekorator staticmethod
    def v_nfw_sq(r, rho_0, h):  # Fungsi kuadrat kecepatan rotasi halo NFW
        """Kuadrat kecepatan rotasi dari halo NFW."""  # Docstring
        r_arr = np.atleast_1d(r)  # Memastikan r adalah array 1D
        v_sq = np.zeros_like(r_arr)  # Inisialisasi array v² nol
        mask = r_arr > 0  # Mask untuk r > 0
        r_nonzero = r_arr[mask]  # Radius non-nol
        x = r_nonzero / h  # Parameter x = r/h
        log_term = np.log(1 + x)  # Istilah logaritmik
        fraction = x / (1 + x)  # Fraksi
```

```
        v_sq[mask] = 4 * np.pi * G_ASTRO * rho_0 * h**2 * (log_term - fraction) / x  # Rumus v² NFW
        return v_sq  # Mengembalikan v²

    @staticmethod  # Dekorator staticmethod
    def v_core_sq(r, rho_0, h):  # Fungsi kuadrat kecepatan rotasi halo Core
        """Kuadrat kecepatan rotasi dari halo Core."""  # Docstring
        r_arr = np.atleast_1d(r)  # Memastikan r adalah array 1D
        v_sq = np.zeros_like(r_arr)  # Inisialisasi array v² nol
        mask = r_arr > 0  # Mask untuk r > 0
        r_nonzero = r_arr[mask]  # Radius non-nol
        r3_ratio = (r_nonzero / h)**3  # Rasio (r/h)^3
        log_term = np.log1p(r3_ratio)  # Log(1 + (r/h)^3)
        v_sq[mask] = (4 * np.pi * G_ASTRO / 3) * rho_0 * (h**3 / r_nonzero) * log_term  # Rumus v² Core
        return v_sq  # Mengembalikan v²

    def v_baryon_sq(self, r, Sigma_be, a_b, Sigma_d0, a_d):  # Fungsi total kuadrat kecepatan baryonik
        """Total kuadrat kecepatan dari komponen baryonik (bulge + disk)."""  # Docstring
        return self.v_bulge_sq(r, Sigma_be, a_b) + self.v_disk_sq(r, Sigma_d0, a_d)  # Penjumlahan v² bulge + disk

    # ============================== FUNGSI MODEL MOND ================================
    def v_mond_std_sq(self, r, v_b_sq):  # Fungsi kuadrat kecepatan MOND standar
        """Kuadrat kecepatan MOND standar dengan interpolasi  (x) = x / sqrt(1 + x²)."""  # Docstring
        r_arr = np.atleast_1d(r)  # Memastikan r adalah array 1D
        v_b_sq_arr = np.atleast_1d(v_b_sq)  # Memastikan v_b² adalah array 1D
        v_m_sq = np.zeros_like(r_arr)  # Inisialisasi array v_m² nol
        mask = v_b_sq_arr > 1e-9  # Mask untuk v_b² > threshold
        v_b_sq_valid = v_b_sq_arr[mask]  # v_b² valid
        r_valid = r_arr[mask]  # r valid
        term_in_sqrt = 1 + (2 * A0_STD_ASTRO * r_valid / v_b_sq_valid)**2  # Istilah dalam sqrt untuk interpolasi
        v_m_sq[mask] = (v_b_sq_valid / np.sqrt(2)) * np.sqrt(1 + np.sqrt(term_in_sqrt))  # Rumus v_m² MOND std
        return v_m_sq  # Mengembalikan v_m²

    def solve_a_m_arctan(self, a_b, tol=1e-6, verbose=False):  # Fungsi solusi numerik a_M untuk MOND arctan
        """Solusi numerik untuk a_M dalam MOND arctangent menggunakan Newton-Raphson."""  # Docstring
        def f(u):  # Fungsi f(u) untuk persamaan Newton-Raphson
```

```
            return u * (2 / np.pi) * np.arctan((np.pi * u) / 2) - a_b /␣
↪A0_ARCTAN_ASTRO  # Persamaan f(u) = 0

        def f_prime(u):  # Turunan f'(u)
            return (2 / np.pi) * np.arctan((np.pi * u) / 2) + u / (1 + ((np.pi␣
↪* u) / 2)**2)  # Rumus turunan

        # Estimasi awal: campuran antara rezim Newtonian dan deep-MOND
        u_guess = a_b / A0_ARCTAN_ASTRO + np.sqrt(a_b / A0_ARCTAN_ASTRO)  #␣
↪Estimasi awal u
        # Iterasi Newton-Raphson
        u = u_guess  # Inisialisasi u dengan guess
        if verbose:  # Jika verbose=True, cetak info iterasi
            print("\nIterasi Newton-Raphson untuk a_M (arctan):")  # Header␣
↪iterasi
            print(f"Estimasi awal u: {u:.6e}")  # Cetak estimasi awal

        converged = False  # Flag konvergensi

        for i in range(50):  # Loop iterasi maksimal 50 kali
            delta = f(u) / f_prime(u)  # Hitung delta = f(u)/f'(u)
            if verbose:  # Jika verbose, cetak iterasi
                print(f"Iterasi {i+1}: u = {u:.6e}, delta = {delta:.6e}")  #␣
↪Cetak u dan delta
            u -= delta  # Update u = u - delta
            if abs(delta) < tol:  # Cek konvergensi
                if verbose:  # Jika verbose, cetak konvergensi
                    print(f"Konvergen setelah {i+1} iterasi.")  # Pesan␣
↪konvergensi
                converged = True  # Set flag True
                break  # Keluar loop

        if not converged:  # Jika tidak konvergen
            warnings.warn(f"\nPeringatan: Iterasi Newton-Raphson tidak␣
↪konvergen setelah 50 iterasi untuk a_b = {a_b:.6e}. "  # Peringatan␣
↪non-konvergensi
                          f"Menggunakan nilai terakhir u = {u:.6e} dengan error␣
↪= {abs(delta):.6e} (tol = {tol}).")  # Detail error

        a_m = u * A0_ARCTAN_ASTRO  # Hitung a_M = u * A0
        if verbose:  # Jika verbose, cetak a_M akhir
            print(f"a_M akhir: {a_m:.6e} (dalam satuan astro)")  # Pesan a_M
        return a_m  # Mengembalikan a_M

    def v_mond_arctan_sq(self, r, v_b_sq):  # Fungsi kuadrat kecepatan MOND␣
↪arctan
```

```
        """Kuadrat kecepatan MOND arctangent dengan solusi numerik."""  #␣
↪Docstring
        r_arr = np.atleast_1d(r)  # Memastikan r adalah array 1D
        v_b_sq_arr = np.atleast_1d(v_b_sq)  # Memastikan v_b² adalah array 1D
        v_m_sq = np.zeros_like(r_arr)  # Inisialisasi array v_m² nol
        mask = v_b_sq_arr > 1e-9  # Mask untuk v_b² > threshold
        for i in np.where(mask)[0]:  # Loop untuk setiap indeks valid
            a_b = v_b_sq_arr[i] / r_arr[i]  # Hitung percepatan baryonik a_b
            a_m = self.solve_a_m_arctan(a_b)  # Solusi numerik a_M
            v_m_sq[i] = a_m * r_arr[i]  # v_m² = a_M * r
        return v_m_sq  # Mengembalikan v_m²

    def v_total_mond_std(self, r, Sigma_be, a_b, Sigma_d0, a_d):  # Fungsi␣
↪total kecepatan MOND standar
        """Total kecepatan rotasi MOND standar."""  # Docstring
        v_b_sq = self.v_baryon_sq(r, Sigma_be, a_b, Sigma_d0, a_d)  # Hitung␣
↪v_b² baryonik
        v_total_sq = self.v_mond_std_sq(r, v_b_sq)  # Hitung v_total² MOND std
        return np.sqrt(np.maximum(0, v_total_sq))  # Mengembalikan v_total␣
↪(akar kuadrat, clip nol)

    def v_total_mond_arctan(self, r, Sigma_be, a_b, Sigma_d0, a_d):  # Fungsi␣
↪total kecepatan MOND arctan
        """Total kecepatan rotasi MOND arctangent."""  # Docstring
        v_b_sq = self.v_baryon_sq(r, Sigma_be, a_b, Sigma_d0, a_d)  # Hitung␣
↪v_b² baryonik
        v_total_sq = self.v_mond_arctan_sq(r, v_b_sq)  # Hitung v_total² MOND␣
↪arctan
        return np.sqrt(np.maximum(0, v_total_sq))  # Mengembalikan v_total␣
↪(akar kuadrat, clip nol)

    def v_total_nfw(self, r, Sigma_be, a_b, Sigma_d0, a_d, rho_0, h):  # Fungsi␣
↪total kecepatan ΛCDM NFW
        """Total kecepatan rotasi ΛCDM NFW."""  # Docstring
        v_b2 = self.v_bulge_sq(r, Sigma_be, a_b)  # v² bulge
        v_d2 = self.v_disk_sq(r, Sigma_d0, a_d)  # v² disk
        v_g2 = self.v_gas_sq(r)  # v² gas (nol)
        v_h2 = self.v_nfw_sq(r, rho_0, h)  # v² halo NFW
        v_total_sq = v_b2 + v_d2 + v_g2 + v_h2  # Total v²
        return np.sqrt(np.maximum(0, v_total_sq))  # Mengembalikan v_total

    def v_total_core(self, r, Sigma_be, a_b, Sigma_d0, a_d, rho_0, h):  #␣
↪Fungsi total kecepatan ΛCDM Core
        """Total kecepatan rotasi ΛCDM Core."""  # Docstring
        v_b2 = self.v_bulge_sq(r, Sigma_be, a_b)  # v² bulge
        v_d2 = self.v_disk_sq(r, Sigma_d0, a_d)  # v² disk
```

```
        v_g2 = self.v_gas_sq(r)  # v² gas (nol)
        v_h2 = self.v_core_sq(r, rho_0, h)  # v² halo Core
        v_total_sq = v_b2 + v_d2 + v_g2 + v_h2  # Total v²
        return np.sqrt(np.maximum(0, v_total_sq))  # Mengembalikan v_total

    # ============================== PARAMETER AWAL DAN BATASAN␣
↪================================
    def set_initial_parameters(self):  # Fungsi pengaturan parameter awal dan␣
↪batasan fitting
        """Parameter awal dan batasan untuk fitting, disesuaikan untuk model␣
↪baru MOND Arctan."""  # Docstring
        return {  # Mengembalikan dictionary parameter untuk setiap galaksi dan␣
↪model
            'Bima Sakti': {  # Parameter untuk Bima Sakti
                'MOND Std': {'p0': [0.66e9, 1.60, 0.61e9, 8.23], 'bounds': ([0.
↪62e9, 1.51, 0.26e9, 7.59], [0.70e9, 1.69, 0.96e9, 8.87])},  # p0: initial␣
↪params, bounds: batasan
                'MOND Arctan': {'p0': [0.66e9, 1.60, 0.61e9, 8.23], 'bounds':␣
↪([0.62e9, 1.51, 0.26e9, 7.59], [0.70e9, 1.69, 0.96e9, 8.87])},  # Sama untuk␣
↪Arctan
                'ΛCDM NFW': {'p0': [0.66e9, 1.60, 0.61e9, 8.23, 0.71e4, 1.
↪07e3], 'bounds': ([0.62e9, 1.51, 0.26e9, 7.59, 0.37e4, 0.73e3], [0.70e9, 1.
↪69, 0.96e9, 8.87, 1.05e4, 1.41e3])},  # Tambah rho_0, h
                'ΛCDM Core': {'p0': [0.66e9, 1.60, 0.61e9, 8.23, 0.71e4, 1.
↪07e3], 'bounds': ([0.62e9, 1.51, 0.26e9, 7.59, 0.37e4, 0.73e3], [0.70e9, 1.
↪69, 0.96e9, 8.87, 1.05e4, 1.41e3])}  # Sama untuk Core
            },
            'M31': {  # Parameter untuk M31
                'MOND Std': {'p0': [0.70e9, 1.31, 0.61e9, 4.52], 'bounds': ([0.
↪55e9, 1.00, 0.29e9, 2.63], [0.85e9, 1.62, 0.93e9, 6.41])},  # p0 dan bounds␣
↪untuk MOND Std
                'MOND Arctan': {'p0': [0.70e9, 1.31, 0.61e9, 4.52], 'bounds':␣
↪([0.55e9, 1.00, 0.29e9, 2.63], [0.85e9, 1.62, 0.93e9, 6.41])},  # Sama untuk␣
↪Arctan
                'ΛCDM NFW': {'p0': [0.70e9, 1.31, 0.61e9, 4.52, 1.70e7, 14.8],␣
↪'bounds': ([0.55e9, 1.00, 0.29e9, 2.63, 0.25e7, 7.6], [0.85e9, 1.62, 0.93e9,␣
↪6.41, 3.15e7, 22.0])},  # Tambah rho_0, h
                'ΛCDM Core': {'p0': [0.70e9, 1.31, 0.61e9, 4.52, 1.70e7, 14.8],␣
↪'bounds': ([0.55e9, 1.00, 0.29e9, 2.63, 0.25e7, 7.6], [0.85e9, 1.62, 0.93e9,␣
↪6.41, 3.15e7, 22.0])}  # Sama untuk Core
            }
        }

    # ============================== FUNGSI ANALISIS DAN FITTING␣
↪================================
```

```
    def analyze_model(self, galaxy_name, model_name):  # Fungsi fitting model␣
↪ke data galaksi
        """Fitting model ke data observasi galaksi dengan pendekatan empiris.
↪"""  # Docstring
        data = self.data[galaxy_name]  # Ekstrak data galaksi
        r_data, v_data, sigma_v_data = data['r'], data['v'], data['sigma_v']  #␣
↪Pisahkan r, v, sigma_v
        fit_function = self.models[model_name]  # Ambil fungsi fitting
        p0 = self.initial_params[galaxy_name][model_name]['p0']  # Parameter␣
↪awal
        bounds = self.initial_params[galaxy_name][model_name]['bounds']  #␣
↪Batasan parameter

        print(f"\n{'='*80}")  # Cetak header analisis
        print(f"ANALISIS: Model {model_name} untuk Galaksi {galaxy_name}")  #␣
↪Judul analisis
        print(f"{'='*80}")  # Cetak footer header

        try:  # Blok try untuk fitting
            params, covariance = curve_fit(fit_function, r_data, v_data, p0=p0,␣
↪sigma=sigma_v_data,  # Fitting dengan curve_fit
                                           bounds=bounds, maxfev=15000,␣
↪method='trf', jac='3-point')  # Opsi fitting
            errors = np.sqrt(np.diag(covariance))  # Hitung error dari␣
↪kovariansi
            print("[SUCCESS] Fitting berhasil.")  # Pesan sukses

            if 'MOND' in model_name:  # Jika model MOND
                Sigma_be, a_b, Sigma_d0, a_d = params  # Ekstrak parameter MOND
                print("\n--- PARAMETER FITTING (MOND) ---")  # Header parameter
                print(f"Sigma_be (M_sun/kpc^2): {Sigma_be:.2e} ± {errors[0]:.
↪2e}")  # Cetak Sigma_be
                print(f"a_b (kpc): {a_b:.2f} ± {errors[1]:.2f}")  # Cetak a_b
                print(f"Sigma_d0 (M_sun/kpc^2): {Sigma_d0:.2e} ± {errors[2]:.
↪2e}")  # Cetak Sigma_d0
                print(f"a_d (kpc): {a_d:.2f} ± {errors[3]:.2f}")  # Cetak a_d
                param_names = ['Sigma_be', 'a_b', 'Sigma_d0', 'a_d']  # Nama␣
↪parameter
                if model_name == 'MOND Arctan':  # Khusus untuk Arctan
                    # Tambahan fitur: Hitung dan tampilkan a_M arctangent dari␣
↪metode iterasi untuk sampel radius
                    print("\n--- NILAI a_M ARCTANGENT DARI METODE ITERASI␣
↪(untuk 5 radius sampel) ---")  # Header a_M
                    sample_indices = np.linspace(0, len(r_data)-1, 5,␣
↪dtype=int)  # Indeks sampel
```

```
                    v_b_sq = self.v_baryon_sq(r_data, Sigma_be, a_b, Sigma_d0,␣
↪a_d)  # Hitung v_b²
                    for idx in sample_indices:  # Loop sampel
                        r_sample = r_data[idx]  # Radius sampel
                        a_b_sample = v_b_sq[idx] / r_sample  # a_b sampel
                        # Panggil dengan verbose=True untuk menampilkan proses␣
↪iterasi
                        a_m_sample = self.solve_a_m_arctan(a_b_sample,␣
↪verbose=True)  # Hitung a_M dengan verbose
                        print(f"Radius {r_sample:.3f} kpc: a_b = {a_b_sample:.
↪6e}, a_M = {a_m_sample:.6e}")  # Cetak hasil
            else:  # Jika model ΛCDM
                Sigma_be, a_b, Sigma_d0, a_d, rho_0, h = params  # Ekstrak␣
↪parameter ΛCDM
                halo_type = 'NFW' if 'NFW' in model_name else 'Core'  # Jenis␣
↪halo
                print(f"\n--- PARAMETER FITTING (ΛCDM {halo_type}) ---")  #␣
↪Header parameter
                print(f"Sigma_be (M_sun/kpc^2): {Sigma_be:.2e} ± {errors[0]:.
↪2e}")  # Cetak Sigma_be
                print(f"a_b (kpc): {a_b:.2f} ± {errors[1]:.2f}")  # Cetak a_b
                print(f"Sigma_d0 (M_sun/kpc^2): {Sigma_d0:.2e} ± {errors[2]:.
↪2e}")  # Cetak Sigma_d0
                print(f"a_d (kpc): {a_d:.2f} ± {errors[3]:.2f}")  # Cetak a_d
                print(f"rho_0 (M_sun/kpc^3): {rho_0:.2e} ± {errors[4]:.2e}")  #␣
↪Cetak rho_0
                print(f"h (kpc): {h:.2f} ± {errors[5]:.2f}")  # Cetak h
                param_names = ['Sigma_be', 'a_b', 'Sigma_d0', 'a_d', 'rho_0',␣
↪'h']  # Nama parameter

            self.plot_fit_components(galaxy_name, model_name, params, data)  #␣
↪Plot komponen fitting
            return params, covariance, param_names  # Kembalikan hasil

        except (RuntimeError, ValueError) as e:  # Tangkap error fitting
            print(f"[ERROR] Fitting gagal: {e}")  # Cetak error
            return None, None, None  # Kembalikan None

    def plot_fit_components(self, galaxy_name, model_name, params, data):  #␣
↪Fungsi plotting komponen fitting
        """Visualisasi sistematis hasil fitting dengan dekomposisi komponen."""␣
↪ # Docstring
        r_data = data['r']  # Radius data
        r_plot = np.logspace(np.log10(max(1e-2, r_data.min())), np.log10(r_data.
↪max()), 200)  # Grid radius logaritmik untuk plot
        fit_function = self.models[model_name]  # Fungsi fitting
```

```
        v_fit = fit_function(r_plot, *params)  # Hitung v_fit

        plt.figure(figsize=(12, 8))  # Buat figure baru
        plt.errorbar(r_data, data['v'], yerr=data['sigma_v'], fmt='ko', label='Data Observasi', capsize=3, markersize=5, zorder=5)  # Plot data dengan error bar

        if 'MOND' in model_name:  # Jika model MOND
            Sigma_be, a_b, Sigma_d0, a_d = params  # Ekstrak parameter
            v_bulge_fit = np.sqrt(self.v_bulge_sq(r_plot, Sigma_be, a_b))  # v bulge
            v_disk_fit = np.sqrt(self.v_disk_sq(r_plot, Sigma_d0, a_d))  # v disk
            v_baryon_fit = np.sqrt(self.v_baryon_sq(r_plot, Sigma_be, a_b, Sigma_d0, a_d))  # v baryonik total

            label = 'MOND Std' if model_name == 'MOND Std' else 'MOND Arctan'  # Label model
            plt.plot(r_plot, v_fit, '-', color='red', linewidth=2.5, label=f'Prediksi Total {label}')  # Plot total
            plt.plot(r_plot, v_baryon_fit, '--', color='blue', label='Total Baryonik')  # Plot baryonik
            plt.plot(r_plot, v_bulge_fit, ':', color='orange', label='Bulge')  # Plot bulge
            plt.plot(r_plot, v_disk_fit, ':', color='green', label='Disk')  # Plot disk
        else:  # Jika model ΛCDM
            Sigma_be, a_b, Sigma_d0, a_d, rho_0, h = params  # Ekstrak parameter
            halo_type = 'NFW' if 'NFW' in model_name else 'Core'  # Jenis halo
            v_bulge_fit = np.sqrt(self.v_bulge_sq(r_plot, Sigma_be, a_b))  # v bulge
            v_disk_fit = np.sqrt(self.v_disk_sq(r_plot, Sigma_d0, a_d))  # v disk
            v_halo_func = self.v_nfw_sq if 'NFW' in model_name else self.v_core_sq  # Fungsi halo
            v_halo_fit = np.sqrt(v_halo_func(r_plot, rho_0, h))  # v halo

            plt.plot(r_plot, v_fit, '-', color='red', linewidth=2.5, label=f'Prediksi Total ΛCDM {halo_type}')  # Plot total
            plt.plot(r_plot, v_bulge_fit, '--', color='orange', label='Bulge')  # Plot bulge
            plt.plot(r_plot, v_disk_fit, '--', color='green', label='Disk')  # Plot disk
            plt.plot(r_plot, v_halo_fit, '--', color='blue', label=f'Halo {halo_type}')  # Plot halo
```

```python
        plt.xscale('log')  # Skala x logaritmik
        plt.xlabel('Radius (kpc)', fontsize=14)  # Label x
        plt.ylabel('Kecepatan Rotasi (km/s)', fontsize=14)  # Label y
        plt.title(f'Kurva Rotasi {galaxy_name} - Model {model_name}', fontsize=16)  # Judul plot
        plt.legend(loc='best', fontsize=12)  # Legenda
        plt.grid(True, which="both", ls="--", alpha=0.5)  # Grid
        plt.ylim(bottom=0)  # Batas y bawah nol
        plt.tight_layout()  # Atur layout
        plt.show()  # Tampilkan plot

    @staticmethod  # Dekorator staticmethod
    def calculate_statistics(v_obs, sigma_v, v_model, k):  # Fungsi perhitungan statistik model
        """Perhitungan statistik empiris:  ², AIC, BIC, R²."""  # Docstring
        residuals = v_obs - v_model  # Hitung residual
        chi_sq = np.sum((residuals / sigma_v) ** 2)  # Hitung  ²
        n = len(v_obs)  # Jumlah data
        dof = n - k  # Derajat kebebasan
        chi_sq_reduced = chi_sq / dof if dof > 0 else float('inf')  #  ² reduced
        aic = 2 * k + chi_sq  # AIC
        bic = np.log(n) * k + chi_sq  # BIC
        r_squared = r2_score(v_obs, v_model)  # R²
        return chi_sq, chi_sq_reduced, aic, bic, r_squared  # Kembalikan statistik

    def compare_model_statistics(self, galaxy, results):  # Fungsi perbandingan statistik antar model
        """Perbandingan statistik sistematis antar-model."""  # Docstring
        print(f"\n{'='*80}")  # Header evaluasi
        print(f"EVALUASI STATISTIK UNTUK {galaxy.upper()}")  # Judul
        print(f"{'='*80}")  # Footer header
        print(f"{'Model':<20} | {' ²':>10} | {' ²/dof':>10} | {'AIC':>10} | {'BIC':>10} | {'R²':>10}")  # Header tabel
        print(f"{'-'*20}-|{'-'*12}|{'-'*12}|{'-'*12}|{'-'*12}|{'-'*12}")  # Garis pemisah

        data = self.data[galaxy]  # Data galaksi
        v_data, sigma_v_data = data['v'], data['sigma_v']  # v dan sigma_v
        model_stats = {}  # Dictionary statistik

        for model_name, result in results.items():  # Loop setiap model
            if result['params'] is not None:  # Jika fitting sukses
                k = len(result['params'])  # Jumlah parameter
                v_model = result['fit_function'](data['r'], *result['params']) # Prediksi v_model
```

```
            chi2, chi2_red, aic, bic, r2 = self.
↪calculate_statistics(v_data, sigma_v_data, v_model, k)  # Hitung statistik
            model_stats[model_name] = {'aic': aic, 'bic': bic}  # Simpan␣
↪AIC, BIC
            print(f"{model_name:<20} | {chi2:>10.2f} | {chi2_red:>10.2f} |␣
↪{aic:>10.2f} | {bic:>10.2f} | {r2:>10.4f}")  # Cetak baris tabel

    print(f"{'-'*80}\n")  # Garis pemisah

    # Contoh perbandingan AIC antara MOND Std dan MOND Arctan
    if 'MOND Std' in model_stats and 'MOND Arctan' in model_stats:  # Jika␣
↪kedua model ada
        delta_aic = model_stats['MOND Arctan']['aic'] - model_stats['MOND␣
↪Std']['aic']  # Delta AIC
        print(f"Perbandingan AIC (Arctan - Std): {delta_aic:.2f}")  # Cetak␣
↪delta
        if delta_aic > 10:  # Interpretasi delta >10
            print("-> Bukti sangat kuat mendukung MOND Std.")  # Kesimpulan
        elif delta_aic > 2:  # Interpretasi 2 < delta <=10
            print("-> Bukti positif mendukung MOND Std.")
        elif delta_aic < -10:  # Interpretasi delta <-10
            print("-> Bukti sangat kuat mendukung MOND Arctan.")
        elif delta_aic < -2:  # Interpretasi -10 <= delta <-2
            print("-> Bukti positif mendukung MOND Arctan.")
        else:  # Interpretasi |delta| <=2
            print("-> Model setara.")

    # Perbandingan AIC antara ΛCDM NFW dan Core
    if 'ΛCDM NFW' in model_stats and 'ΛCDM Core' in model_stats:  # Jika␣
↪kedua model ada
        delta_aic = model_stats['ΛCDM Core']['aic'] - model_stats['ΛCDM␣
↪NFW']['aic']  # Delta AIC
        print(f"Perbandingan AIC (Core - NFW): {delta_aic:.2f}")  # Cetak␣
↪delta
        if delta_aic > 10:  # Interpretasi delta >10
            print("-> Bukti sangat kuat mendukung ΛCDM NFW.")
        elif delta_aic > 2:  # Interpretasi 2 < delta <=10
            print("-> Bukti positif mendukung ΛCDM NFW.")
        elif delta_aic < -10:  # Interpretasi delta <-10
            print("-> Bukti sangat kuat mendukung ΛCDM Core.")
        elif delta_aic < -2:  # Interpretasi -10 <= delta <-2
            print("-> Bukti positif mendukung ΛCDM Core.")
        else:  # Interpretasi |delta| <=2
            print("-> Model setara.")
```

```
    def plot_model_comparison(self, galaxy, results):  # Fungsi plot␣
↪perbandingan model (NFW vs Core, Std vs Arctan)
        """Plot perbandingan logis antar-model: ΛCDM NFW vs Core, MOND Std vs␣
↪Arctan."""  # Docstring
        data = self.data[galaxy]  # Data galaksi
        r_data = data['r']  # Radius data
        r_plot = np.logspace(np.log10(max(1e-2, r_data.min())), np.log10(r_data.
↪max()), 200)  # Grid r plot

        # Plot 1: Perbandingan ΛCDM NFW vs Core
        plt.figure(figsize=(12, 8))  # Figure baru
        plt.errorbar(r_data, data['v'], yerr=data['sigma_v'], fmt='ko',␣
↪label='Observasi', capsize=3, markersize=5)  # Plot observasi

        if 'ΛCDM NFW' in results and results['ΛCDM NFW']['params'] is not None:␣
↪ # Jika NFW sukses
            v_nfw = self.v_total_nfw(r_plot, *results['ΛCDM NFW']['params'])  #␣
↪Hitung v NFW
            plt.plot(r_plot, v_nfw, 'b--', label='ΛCDM NFW')  # Plot NFW

        if 'ΛCDM Core' in results and results['ΛCDM Core']['params'] is not␣
↪None:  # Jika Core sukses
            v_core = self.v_total_core(r_plot, *results['ΛCDM Core']['params'])␣
↪ # Hitung v Core
            plt.plot(r_plot, v_core, 'g:', label='ΛCDM Core')  # Plot Core

        plt.xscale('log')  # Skala log x
        plt.xlabel('Radius (kpc)')  # Label x
        plt.ylabel('Kecepatan Rotasi (km/s)')  # Label y
        plt.title(f'Perbandingan ΛCDM NFW vs Core untuk {galaxy}')  # Judul
        plt.legend()  # Legenda
        plt.grid(True, which='both')  # Grid
        plt.ylim(bottom=0)  # Batas y
        plt.show()  # Tampilkan

        # Plot 2: Perbandingan MOND Std vs Arctan
        plt.figure(figsize=(12, 8))  # Figure baru
        plt.errorbar(r_data, data['v'], yerr=data['sigma_v'], fmt='ko',␣
↪label='Observasi', capsize=3, markersize=5)  # Plot observasi

        if 'MOND Std' in results and results['MOND Std']['params'] is not None:␣
↪ # Jika Std sukses
            v_std = self.v_total_mond_std(r_plot, *results['MOND␣
↪Std']['params'])  # Hitung v Std
            plt.plot(r_plot, v_std, 'r-', label='MOND Std')  # Plot Std
```

```
        if 'MOND Arctan' in results and results['MOND Arctan']['params'] is not␣
↪None:  # Jika Arctan sukses
            v_arctan = self.v_total_mond_arctan(r_plot, *results['MOND␣
↪Arctan']['params'])  # Hitung v Arctan
            plt.plot(r_plot, v_arctan, 'm-.', label='MOND Arctan')  # Plot␣
↪Arctan

        plt.xscale('log')  # Skala log x
        plt.xlabel('Radius (kpc)')  # Label x
        plt.ylabel('Kecepatan Rotasi (km/s)')  # Label y
        plt.title(f'Perbandingan MOND Std vs Arctan untuk {galaxy}')  # Judul
        plt.legend()  # Legenda
        plt.grid(True, which='both')  # Grid
        plt.ylim(bottom=0)  # Batas y
        plt.show()  # Tampilkan

    def plot_all_models_comparison(self, galaxy, results):  # Fungsi plot semua␣
↪model dalam satu plot
        """Plot perbandingan semua model dalam satu plot untuk galaksi tertentu.
↪"""  # Docstring
        data = self.data[galaxy]  # Data galaksi
        r_data = data['r']  # Radius data
        r_plot = np.logspace(np.log10(max(1e-2, r_data.min())), np.log10(r_data.
↪max()), 200)  # Grid r

        plt.figure(figsize=(12, 8))  # Figure baru
        plt.errorbar(r_data, data['v'], yerr=data['sigma_v'], fmt='ko',␣
↪label='Data Observasi', capsize=3, markersize=5)  # Plot data

        # Warna dan gaya garis untuk setiap model
        styles = {  # Dictionary gaya plot
            'MOND Std': {'color': 'red', 'linestyle': '-'},
            'MOND Arctan': {'color': 'magenta', 'linestyle': '-.'},
            'ΛCDM NFW': {'color': 'blue', 'linestyle': '--'},
            'ΛCDM Core': {'color': 'green', 'linestyle': ':'}
        }

        for model_name, result in results.items():  # Loop model
            if result['params'] is not None:  # Jika sukses
                v_fit = result['fit_function'](r_plot, *result['params'])  #␣
↪Hitung v_fit
                plt.plot(r_plot, v_fit, label=model_name, **styles[model_name])␣
↪ # Plot dengan gaya

        plt.xscale('log')  # Skala log x
        plt.xlabel('Radius (kpc)', fontsize=14)  # Label x
```

```
        plt.ylabel('Kecepatan Rotasi (km/s)', fontsize=14)  # Label y
        plt.title(f'Perbandingan Semua Model untuk {galaxy}', fontsize=16)  # Judul
        plt.legend(loc='best', fontsize=12)  # Legenda
        plt.grid(True, which="both", ls="--", alpha=0.5)  # Grid
        plt.ylim(bottom=0)  # Batas y
        plt.tight_layout()  # Layout
        plt.show()  # Tampilkan

    def plot_residuals(self, galaxy, results):  # Fungsi plot residual untuk evaluasi model
        """Plot residual untuk evaluasi logis kesesuaian model."""  # Docstring
        data = self.data[galaxy]  # Data galaksi
        r_data, v_data, sigma_v_data = data['r'], data['v'], data['sigma_v']  # Pisahkan data

        fig, axes = plt.subplots(len(results), 1, figsize=(12, 10), sharex=True)  # Subplots untuk residual
        if len(results) == 1:  # Jika satu model, wrap axes
            axes = [axes]
        fig.suptitle(f'Residual Analisis untuk {galaxy}', fontsize=16)  # Judul sup

        colors = {'MOND Std': 'red', 'MOND Arctan': 'magenta', 'ΛCDM NFW': 'blue', 'ΛCDM Core': 'green'}  # Warna model

        for i, (model_name, result) in enumerate(results.items()):  # Loop model
            if result['params'] is not None:  # Jika sukses
                v_model = result['fit_function'](r_data, *result['params'])  # Prediksi v_model
                residuals = v_data - v_model  # Hitung residual
                axes[i].errorbar(r_data, residuals, yerr=sigma_v_data, fmt='o',  # Plot residual dengan error
                                 color=colors[model_name], ecolor='lightgray', capsize=3)
                axes[i].axhline(0, color='black', linestyle='--')  # Garis nol
                axes[i].set_ylabel('Residual (km/s)')  # Label y
                axes[i].set_title(model_name)  # Judul subplot
                axes[i].grid(True)  # Grid

        plt.xscale('log')  # Skala log x
        plt.xlabel('Radius (kpc)')  # Label x
        plt.tight_layout()  # Layout
        plt.show()  # Tampilkan
```

```python
    def compare_models(self, galaxy, model1, model2, results):  # Fungsi perbandingan dua model secara komprehensif
        """
        Membandingkan performa statistik dua model kurva rotasi galaksi secara komprehensif.
        """  # Docstring
        # ====================== EKSTRAKSI DATA OBSERVASI ======================
        data = self.data[galaxy]  # Data galaksi
        r_data, v_data, sigma_v_data = data['r'], data['v'], data['sigma_v']  # Pisahkan data

        # ====================== VERIFIKASI KETERSEDIAAN MODEL ======================
        if model1 not in results or model2 not in results:  # Cek ketersediaan model
            print(f"Data tidak tersedia untuk {model1} vs {model2}")  # Pesan error
            return  # Keluar

        # ====================== EKSTRAKSI PARAMETER MODEL ======================
        params1 = results[model1]['params']  # Parameter model 1
        params2 = results[model2]['params']  # Parameter model 2
        if params1 is None or params2 is None:  # Cek jika gagal
            print(f"Fitting gagal untuk {model1} atau {model2}. Perbandingan dilewati.")  # Pesan skip
            return  # Keluar

        func1 = results[model1]['fit_function']  # Fungsi model 1
        func2 = results[model2]['fit_function']  # Fungsi model 2

        # ====================== PREDIKSI KECEPATAN ROTASI ======================
        v_pred1 = func1(r_data, *params1)  # Prediksi model 1
        v_pred2 = func2(r_data, *params2)  # Prediksi model 2

        # ====================== PERHITUNGAN METRIK STATISTIK ======================
        k1 = len(params1)  # Jumlah param 1
        k2 = len(params2)  # Jumlah param 2

        chi_sq1, _, aic1, bic1, r2_1 = self.calculate_statistics(v_data, sigma_v_data, v_pred1, k1)  # Statistik model 1
        chi_sq2, _, aic2, bic2, r2_2 = self.calculate_statistics(v_data, sigma_v_data, v_pred2, k2)  # Statistik model 2
```

```python
    # ====================== ANALISIS PERBEDAAN MODEL ======================
    delta_chi_sq = chi_sq1 - chi_sq2  # Delta  ²
    delta_aic = aic1 - aic2  # Delta AIC
    delta_bic = bic1 - bic2  # Delta BIC

    # ====================== TAMPILAN HASIL PERBANDINGAN ======================
    print(f"\n{'='*80}")  # Header perbandingan
    print(f"PERBANDINGAN {galaxy.upper()}: {model1} vs {model2}")  # Judul
    print(f"{'='*80}")  # Footer
    print(f"{'Model':<20} | {' ²':>10} | {'AIC':>10} | {'BIC':>10} | {'R²':>10}")  # Header tabel
    print(f"{'-'*20}-|{'-'*12}|{'-'*12}|{'-'*12}|{'-'*12}")  # Garis

    print(f"{model1:<20} | {chi_sq1:>10.2f} | {aic1:>10.2f} | {bic1:>10.2f} | {r2_1:>10.4f}")  # Baris model 1
    print(f"{model2:<20} | {chi_sq2:>10.2f} | {aic2:>10.2f} | {bic2:>10.2f} | {r2_2:>10.4f}")  # Baris model 2

    # ====================== INTERPRETASI PERBEDAAN STATISTIK ======================
    print("\nPerbedaan:")  # Header perbedaan
    print(f"Δ ² ({model1} - {model2}): {delta_chi_sq:.2f}")  # Cetak delta  ²
    print(f"ΔAIC ({model1} - {model2}): {delta_aic:.2f}")  # Cetak delta AIC
    print(f"ΔBIC ({model1} - {model2}): {delta_bic:.2f}")  # Cetak delta BIC

    print("\nInterpretasi Berdasarkan AIC:")  # Header interpretasi
    if delta_aic > 10:  # Interpretasi delta AIC >10
        print(f"  Bukti sangat kuat {model2} lebih baik.")
    elif delta_aic > 6:  # 6 < delta <=10
        print(f"  Bukti kuat {model2} lebih baik.")
    elif delta_aic > 2:  # 2 < delta <=6
        print(f"  Bukti sedang {model2} lebih baik.")
    elif delta_aic > 0:  # 0 < delta <=2
        print(f"  Bukti lemah {model2} lebih baik.")
    elif delta_aic < -10:  # delta < -10
        print(f"  Bukti sangat kuat {model1} lebih baik.")
    elif delta_aic < -6:  # -10 <= delta < -6
        print(f"  Bukti kuat {model1} lebih baik.")
    elif delta_aic < -2:  # -6 <= delta < -2
        print(f"  Bukti sedang {model1} lebih baik.")
    elif delta_aic < 0:  # -2 <= delta < 0
        print(f"  Bukti lemah {model1} lebih baik.")
    else:  # delta == 0
        print("  Kedua model setara berdasarkan AIC.")
```

```
        # ====================== UJI SIGNIFIKANSI STATISTIK␣
↪======================
        df = abs(k1 - k2)  # Perbedaan derajat kebebasan
        p_value = stats.chi2.sf(abs(delta_chi_sq), df) if df > 0 else 1.0  #␣
↪P-value dari chi2 sf

        print(f"\nUji Signifikansi Chi-kuadrat:")  # Header uji
        print(f"P-value: {p_value:.4f}")  # Cetak p-value

        if p_value < 0.05:  # Jika signifikan
            better_model = model1 if delta_chi_sq < 0 else model2  # Model␣
↪lebih baik
            print(f"  Perbedaan signifikan (p < 0.05). Model terbaik:␣
↪{better_model}")  # Kesimpulan
        else:  # Tidak signifikan
            print("  Tidak ada perbedaan signifikan (p   0.05)")  # Kesimpulan

        print(f"{'='*80}\n")  # Footer

    # ============================== FUNGSI BANTU UNTUK PLOT BARU␣
↪================================
    def compute_aB_aobs_for_model(self, r, model_name, params):  # Fungsi bantu␣
↪untuk hitung a_B dan a_obs model
        """Menghitung pasangan (a_B, a_obs_model) untuk suatu model pada grid␣
↪radius r."""  # Docstring fungsi
        Sigma_be, a_b, Sigma_d0, a_d = params[:4]  # Ambil parameter baryonik␣
↪pertama
        v_b_sq = self.v_baryon_sq(r, Sigma_be, a_b, Sigma_d0, a_d)  # Hitung␣
↪v_b_sq
        a_B = v_b_sq / r  # Hitung a_B = v_b_sq / r
        if model_name == 'MOND Std':  # Jika MOND Std
            v_model = self.v_total_mond_std(r, *params)  # Hitung v_model Std
        elif model_name == 'MOND Arctan':  # Jika MOND Arctan
            v_model = self.v_total_mond_arctan(r, *params)  # Hitung v_model␣
↪Arctan
        elif model_name == 'ΛCDM NFW':  # Jika ΛCDM NFW
            v_model = self.v_total_nfw(r, *params)  # Hitung v_model NFW
        elif model_name == 'ΛCDM Core':  # Jika ΛCDM Core
            v_model = self.v_total_core(r, *params)  # Hitung v_model Core
        else:  # Jika model lain
            v_model = np.zeros_like(r)  # Set v_model ke nol
        a_obs_model = (v_model**2) / r  # Hitung a_obs = v_model^2 / r
        return a_B, a_obs_model  # Mengembalikan a_B dan a_obs

    def plot_baryonic_hypothesis_test(self, galaxy, results):  # Fungsi untuk␣
↪plot uji hipotesis baryonik
```

```
    """Menggabungkan fitur-fitur dua plot relasi percepatan menjadi satu:␣
↪Plot Uji Hipotesis Baryonik."""  # Docstring fungsi
    data = self.data[galaxy]  # Ambil data galaksi
    r_data, v_data = data['r'], data['v']  # Ambil radius dan kecepatan
    a_obs_data = (v_data**2) / r_data  # Hitung a_obs dari data

    # Pilih parameter baryonik untuk menghitung a_B data (prioritas MOND␣
↪Std, lalu MOND Arctan, lalu ΛCDM)
    base_model_for_data = None  # Inisialisasi base model
    for candidate in ['MOND Std', 'MOND Arctan', 'ΛCDM NFW', 'ΛCDM Core']: ␣
↪# Loop prioritas model
        if candidate in results and results[candidate]['params'] is not␣
↪None:  # Cek jika model sukses
            base_model_for_data = candidate  # Set base model
            break  # Keluar loop
    if base_model_for_data is None:  # Jika tidak ada model sukses
        print(f"[INFO] Tidak ada model dengan parameter baryonik untuk␣
↪{galaxy}. Plot Uji Hipotesis Baryonik dilewati.")  # Cetak info skip
        return  # Keluar fungsi

    Sigma_be_d, a_b_d, Sigma_d0_d, a_d_d =␣
↪results[base_model_for_data]['params'][:4]  # Ambil parameter base
    v_b_sq_data = self.v_baryon_sq(r_data, Sigma_be_d, a_b_d, Sigma_d0_d,␣
↪a_d_d)  # Hitung v_b_sq data
    a_B_data = v_b_sq_data / r_data  # Hitung a_B data

    # Grid r dan style
    r_plot = np.logspace(np.log10(max(1e-2, r_data.min())), np.log10(r_data.
↪max()), 300)  # Grid halus untuk plot
    styles = {  # Dictionary gaya plot
        'MOND Std': {'color': 'red', 'linestyle': '-'},
        'MOND Arctan': {'color': 'magenta', 'linestyle': '-.'},
        'ΛCDM NFW': {'color': 'blue', 'linestyle': '--'},
        'ΛCDM Core': {'color': 'green', 'linestyle': ':'}
    }

    plt.figure(figsize=(12, 9))  # Buat figure plot
    # Data Observasi
    plt.loglog(a_B_data, a_obs_data, 'ko', label=f'Data Galaksi␣
↪({galaxy})', markersize=5, zorder=6)  # Plot data di skala log-log

    # Garis Newtonian dan deep-MOND
    a_B_line = np.logspace(np.log10(max(a_B_data.min(), 1e-12)), np.
↪log10(a_B_data.max()), 300)  # Grid a_B untuk garis referensi
    a_obs_newton = a_B_line  # Garis Newtonian: a_obs = a_B
```

```
        a_obs_deep_mond = np.sqrt(a_B_line * A0_STD_ASTRO)  # Garis deep-MOND:␣
↪a_obs = sqrt(a_B * a0)
        plt.loglog(a_B_line, a_obs_newton, 'k--', label='Garis Newtonian␣
↪($a_{obs} = a_B$)', linewidth=1.8, zorder=3)  # Plot garis Newtonian
        plt.loglog(a_B_line, a_obs_deep_mond, 'g:', label='Garis Deep-MOND␣
↪($a_{obs} = \\sqrt{a_B a_0}$)', linewidth=2.0, zorder=3)  # Plot garis␣
↪deep-MOND
        plt.axvline(A0_STD_ASTRO, color='gray', linestyle='-.', linewidth=1.5,␣
↪label=f'$a_0 = {A0_STD_SI:.1e}$ m/s²', zorder=2)  # Garis vertikal a0

        # Prediksi setiap model pada ruang (a_B, a_obs)
        for model_name in ['ΛCDM NFW', 'ΛCDM Core', 'MOND Std', 'MOND Arctan']:␣
↪ # Loop model
            if model_name in results and results[model_name]['params'] is not␣
↪None:  # Cek jika sukses
                params = results[model_name]['params']  # Ambil parameter
                aB_model, aobs_model = self.compute_aB_aobs_for_model(r_plot,␣
↪model_name, params)  # Hitung aB dan aobs model
                plt.loglog(aB_model, aobs_model, label=f'Prediksi␣
↪{model_name}', **styles[model_name], linewidth=2.5, zorder=4)  # Plot␣
↪prediksi model

        # Format sumbu, label, dan judul
        plt.xlabel(r'Percepatan Baryonik, $a_B$ [$(\mathrm{km/
↪s})^2\,\mathrm{kpc}^{-1}$]', fontsize=14)  # Label x
        plt.ylabel(r'Percepatan Observasi, $a_{obs}$ [$(\mathrm{km/
↪s})^2\,\mathrm{kpc}^{-1}$]', fontsize=14)  # Label y
        plt.title(f'Plot Uji Hipotesis Baryonik - {galaxy}', fontsize=16)  #␣
↪Judul plot
        plt.grid(True, which='both', ls='--', alpha=0.6)  # Tambahkan grid
        plt.legend(loc='best', fontsize=12)  # Tambahkan legenda
        plt.tight_layout()  # Atur layout
        plt.show()  # Tampilkan plot

    # ============================== FUNGSI PLOT DAN ANALISIS STATISTIK BARU␣
↪================================

    def plot_mond_std_specific_test(self, galaxy, results):  # Fungsi untuk␣
↪plot uji spesifik MOND Standar
        """Plot uji spesifik MOND Standar: a_obs/a_M vs a_B."""  # Docstring␣
↪fungsi
        data = self.data[galaxy]  # Ambil data galaksi
        r_data, v_data = data['r'], data['v']  # Ambil radius dan kecepatan
        a_obs_data = (v_data**2) / r_data  # Hitung a_obs data
```

```python
        # Gunakan parameter baryonik dari model mana pun untuk menghitung a_B data
        base_model_for_data = None  # Inisialisasi base model
        for candidate in ['MOND Std', 'MOND Arctan', 'ΛCDM NFW', 'ΛCDM Core']: # Loop prioritas
            if candidate in results and results[candidate]['params'] is not None:  # Cek sukses
                base_model_for_data = candidate  # Set base
                break  # Keluar
        if base_model_for_data is None:  # Jika gagal
            print(f"[INFO] Tidak ada model dengan parameter baryonik untuk {galaxy}. Uji Spesifik MOND Std dilewati.")  # Cetak info
            return  # Keluar

        Sigma_be_d, a_b_d, Sigma_d0_d, a_d_d = results[base_model_for_data]['params'][:4]  # Ambil parameter
        v_b_sq_data = self.v_baryon_sq(r_data, Sigma_be_d, a_b_d, Sigma_d0_d, a_d_d)  # Hitung v_b_sq
        a_B_data = v_b_sq_data / r_data  # Hitung a_B

        # Hitung a_M (MOND Std) untuk data
        term_std_data = 1 + (2 * A0_STD_ASTRO / a_B_data)**2  # Hitung istilah
        a_M_std_data = (a_B_data / np.sqrt(2)) * np.sqrt(1 + np.sqrt(term_std_data))  # Hitung a_M Std

        # Rasio data
        ratio_data_std = a_obs_data / a_M_std_data  # Hitung rasio a_obs / a_M

        # Grid untuk garis prediksi
        r_plot = np.logspace(np.log10(max(1e-2, r_data.min())), np.log10(r_data.max()), 300)  # Grid halus

        # Siapkan gaya plot
        styles = {  # Dictionary gaya
            'MOND Std': {'color': 'red', 'linestyle': '-'},
            'MOND Arctan': {'color': 'magenta', 'linestyle': '-.'},
            'ΛCDM NFW': {'color': 'blue', 'linestyle': '--'},
            'ΛCDM Core': {'color': 'green', 'linestyle': ':'}
        }

        # Buat figure
        plt.figure(figsize=(12, 9))  # Figure plot

        # Plot data rasio
        plt.semilogx(a_B_data, ratio_data_std, 'ko', label=f'Data Galaksi ({galaxy})', markersize=5, zorder=6)  # Plot rasio data
```

```
    # Garis referensi MOND (a_obs = a_M)
    a_B_line = np.logspace(np.log10(max(a_B_data.min(), 1e-12)), np.
↪log10(a_B_data.max()), 300)  # Grid a_B
    plt.semilogx(a_B_line, np.ones_like(a_B_line), color='black',␣
↪linestyle='--', linewidth=1.8,  # Plot garis referensi y=1
                 label='Garis Referensi MOND ($a_{obs} = a_M$)', zorder=3) ␣
↪# Label garis

    # Garis a0
    plt.axvline(A0_STD_ASTRO, color='gray', linestyle='-.', linewidth=1.5,␣
↪label=f'$a_0 = {A0_STD_SI:.1e}$ m/s²', zorder=2)  # Garis vertikal a0

    # PERBAIKAN: Prediksi MOND Arctan dari parameter fitting aktual (bukan␣
↪MOND Std)
    if 'MOND Arctan' in results and results['MOND Arctan']['params'] is not␣
↪None:  # Cek Arctan sukses
        params_arctan = results['MOND Arctan']['params']  # Ambil parameter␣
↪Arctan
        Sigma_be, a_b, Sigma_d0, a_d = params_arctan[:4]  # Parameter␣
↪baryonik

        # Hitung a_B dan a_obs dari model MOND Arctan
        v_b_sq_arctan = self.v_baryon_sq(r_plot, Sigma_be, a_b, Sigma_d0,␣
↪a_d)  # v_b_sq Arctan
        a_B_arctan = v_b_sq_arctan / r_plot  # a_B Arctan

        v_model_arctan = self.v_total_mond_arctan(r_plot, *params_arctan) ␣
↪# v_model Arctan
        a_obs_arctan = (v_model_arctan**2) / r_plot  # a_obs Arctan

        # Hitung a_M_std dari a_B menggunakan rumus MOND Std
        term_std = 1 + (2 * A0_STD_ASTRO / a_B_arctan)**2  # Istilah Std
        a_M_std = (a_B_arctan / np.sqrt(2)) * np.sqrt(1 + np.
↪sqrt(term_std))  # a_M Std
        ratio_arctan = a_obs_arctan / a_M_std  # Rasio untuk Arctan
        plt.semilogx(a_B_arctan, ratio_arctan, label='Prediksi MOND Arctan␣
↪(fitting)', **styles['MOND Arctan'], linewidth=2.5, zorder=4)  # Plot rasio␣
↪Arctan

    # Prediksi ΛCDM: rasio a_obs_LCDM / a_M_std (pemetaan terhadap a_B␣
↪model masing-masing)
    for model_name in ['ΛCDM NFW', 'ΛCDM Core']:  # Loop ΛCDM
        if model_name in results and results[model_name]['params'] is not␣
↪None:  # Cek sukses
            params = results[model_name]['params']  # Ambil parameter
```

```
                aB_model, aobs_model = self.compute_aB_aobs_for_model(r_plot, model_name, params)  # Hitung aB dan aobs

                # Hitung a_M_std untuk a_B model
                term_std = 1 + (2 * A0_STD_ASTRO / aB_model)**2  # Istilah Std
                aM_std_model = (aB_model / np.sqrt(2)) * np.sqrt(1 + np.sqrt(term_std))  # a_M Std
                ratio_model = aobs_model / aM_std_model  # Rasio model
                plt.semilogx(aB_model, ratio_model, label=f'Prediksi {model_name}', **styles[model_name], linewidth=2.5, zorder=4)  # Plot rasio model

        # Format sumbu, label, dan judul
        plt.xlabel(r'Percepatan Baryonik, $a_B$ [$(\mathrm{km/s})^2\,\mathrm{kpc}^{-1}$]', fontsize=14)  # Label x
        plt.ylabel(r'Rasio $a_{obs}/a_M$ (MOND Std)', fontsize=14)  # Label y
        plt.title(f'Uji Spesifik MOND Standar - {galaxy}', fontsize=16)  # Judul
        plt.grid(True, which='both', ls='--', alpha=0.6)  # Grid
        plt.legend(loc='best', fontsize=12)  # Legenda
        plt.tight_layout()  # Layout rapi
        plt.show()  # Tampilkan

    def plot_mond_arctan_specific_test(self, galaxy, results):  # Fungsi untuk plot uji spesifik MOND Arctan
        """Plot uji spesifik MOND Arctan: a_obs/a_M vs a_B."""  # Docstring fungsi
        data = self.data[galaxy]  # Ambil data
        r_data, v_data = data['r'], data['v']  # Radius dan v
        a_obs_data = (v_data**2) / r_data  # a_obs data

        # Gunakan parameter baryonik dari model mana pun untuk menghitung a_B data
        base_model_for_data = None  # Inisialisasi
        for candidate in ['MOND Std', 'MOND Arctan', 'ΛCDM NFW', 'ΛCDM Core']:  # Loop
            if candidate in results and results[candidate]['params'] is not None:  # Cek
                base_model_for_data = candidate  # Set
                break  # Keluar
        if base_model_for_data is None:  # Gagal
            print(f"[INFO] Tidak ada model dengan parameter baryonik untuk {galaxy}. Uji Spesifik MOND Arctan dilewati.")  # Info
            return  # Keluar

        Sigma_be_d, a_b_d, Sigma_d0_d, a_d_d = results[base_model_for_data]['params'][:4]  # Parameter
```

```
        v_b_sq_data = self.v_baryon_sq(r_data, Sigma_be_d, a_b_d, Sigma_d0_d,␣
↪a_d_d)  # v_b_sq
        a_B_data = v_b_sq_data / r_data  # a_B

        # Hitung a_M (MOND Arctan) untuk data
        a_M_arctan_data = np.array([self.solve_a_m_arctan(ab) for ab in␣
↪a_B_data])  # Hitung a_M Arctan untuk setiap a_B

        # Rasio data
        ratio_data_arctan = a_obs_data / a_M_arctan_data  # Rasio

        # Grid untuk garis prediksi
        r_plot = np.logspace(np.log10(max(1e-2, r_data.min())), np.log10(r_data.
↪max()), 300)  # Grid

        # Siapkan gaya plot
        styles = {  # Gaya
            'MOND Std': {'color': 'red', 'linestyle': '-'},
            'MOND Arctan': {'color': 'magenta', 'linestyle': '-.'},
            'ΛCDM NFW': {'color': 'blue', 'linestyle': '--'},
            'ΛCDM Core': {'color': 'green', 'linestyle': ':'}
        }

        # Buat figure
        plt.figure(figsize=(12, 9))  # Figure

        # Plot data rasio
        plt.semilogx(a_B_data, ratio_data_arctan, 'ko', label=f'Data Galaksi␣
↪({galaxy})', markersize=5, zorder=6)  # Plot rasio

        # Garis referensi MOND (a_obs = a_M)
        a_B_line = np.logspace(np.log10(max(a_B_data.min(), 1e-12)), np.
↪log10(a_B_data.max()), 300)  # Grid
        plt.semilogx(a_B_line, np.ones_like(a_B_line), color='black',␣
↪linestyle='--', linewidth=1.8,  # Garis y=1
                     label='Garis Referensi MOND ($a_{obs} = a_M$)', zorder=3) ␣
↪# Label

        # Garis a0
        plt.axvline(A0_STD_ASTRO, color='gray', linestyle='-.', linewidth=1.5,␣
↪label=f'$a_0 = {A0_STD_SI:.1e}$ m/s²', zorder=2)  # Garis a0

        # PERBAIKAN: Prediksi MOND Std dari parameter fitting aktual (bukan␣
↪MOND Arctan)
        if 'MOND Std' in results and results['MOND Std']['params'] is not None:␣
↪ # Cek Std
```

```
            params_std = results['MOND Std']['params']  # Parameter Std
            Sigma_be, a_b, Sigma_d0, a_d = params_std[:4]  # Baryonik

            # Hitung a_B dan a_obs dari model MOND Std
            v_b_sq_std = self.v_baryon_sq(r_plot, Sigma_be, a_b, Sigma_d0, a_d)␣
↪ # v_b_sq Std
            a_B_std = v_b_sq_std / r_plot  # a_B Std

            v_model_std = self.v_total_mond_std(r_plot, *params_std)  # v_model␣
↪Std
            a_obs_std = (v_model_std**2) / r_plot  # a_obs Std

            # Hitung a_M_arctan dari a_B menggunakan rumus MOND Arctan
            a_M_arctan = np.array([self.solve_a_m_arctan(ab) for ab in␣
↪a_B_std])  # a_M Arctan
            ratio_std = a_obs_std / a_M_arctan  # Rasio Std
            plt.semilogx(a_B_std, ratio_std, label='Prediksi MOND Std␣
↪(fitting)', **styles['MOND Std'], linewidth=2.5, zorder=4)  # Plot Std

        # Prediksi ΛCDM: rasio a_obs_LCDM / a_M_arctan (pemetaan terhadap a_B␣
↪model masing-masing)
        for model_name in ['ΛCDM NFW', 'ΛCDM Core']:  # Loop ΛCDM
            if model_name in results and results[model_name]['params'] is not␣
↪None:  # Cek
                params = results[model_name]['params']  # Parameter
                aB_model, aobs_model = self.compute_aB_aobs_for_model(r_plot,␣
↪model_name, params)  # Hitung

                # Hitung a_M_arctan untuk a_B model
                aM_arctan_model = np.array([self.solve_a_m_arctan(ab) for ab in␣
↪aB_model])  # a_M
                ratio_model = aobs_model / aM_arctan_model  # Rasio
                plt.semilogx(aB_model, ratio_model, label=f'Prediksi␣
↪{model_name}', **styles[model_name], linewidth=2.5, zorder=4)  # Plot

        # Format sumbu, label, dan judul
        plt.xlabel(r'Percepatan Baryonik, $a_B$ [$(\mathrm{km/
↪s})^2\,\mathrm{kpc}^{-1}$]', fontsize=14)  # Label x
        plt.ylabel(r'Rasio $a_{obs}/a_M$ (MOND Arctan)', fontsize=14)  # Label y
        plt.title(f'Uji Spesifik MOND Arctan - {galaxy}', fontsize=16)  # Judul
        plt.grid(True, which='both', ls='--', alpha=0.6)  # Grid
        plt.legend(loc='best', fontsize=12)  # Legenda
        plt.tight_layout()  # Layout
        plt.show()  # Tampilkan
```

```
    def plot_baryonic_hypothesis_test_with_stats(self, galaxy, results):  #␣
↪Fungsi untuk plot uji baryonik dengan statistik
        """Plot uji baryonik dengan analisis statistik deviasi lengkap."""  #␣
↪Docstring fungsi
        self.plot_baryonic_hypothesis_test(galaxy, results)  # Panggil plot␣
↪dasar

        # === ANALISIS STATISTIK DEVIASI UNTUK SETIAP MODEL ===
        print(f"\n{'='*60}")  # Header
        print(f"ANALISIS STATISTIK DEVIASI - UJI HIPOTESIS BARYONIK - {galaxy.
↪upper()}")  # Judul
        print(f"{'='*60}")  # Pemisah

        data = self.data[galaxy]  # Data
        r_data, v_data, sigma_v_data = data['r'], data['v'], data['sigma_v']  #␣
↪Pisah
        a_obs_data = (v_data**2) / r_data  # a_obs

        # Gunakan parameter baryonik dari model mana pun untuk menghitung a_B␣
↪data
        base_model_for_data = None  # Inisialisasi
        for candidate in ['MOND Std', 'MOND Arctan', 'ΛCDM NFW', 'ΛCDM Core']: ␣
↪# Loop
            if candidate in results and results[candidate]['params'] is not␣
↪None:  # Cek
                base_model_for_data = candidate  # Set
                break  # Keluar
        if base_model_for_data is None:  # Gagal
            print("Tidak dapat menghitung a_B data: tidak ada model yang␣
↪berhasil difitting.")  # Info
            return  # Keluar

        Sigma_be_d, a_b_d, Sigma_d0_d, a_d_d =␣
↪results[base_model_for_data]['params'][:4]  # Parameter
        v_b_sq_data = self.v_baryon_sq(r_data, Sigma_be_d, a_b_d, Sigma_d0_d,␣
↪a_d_d)  # v_b_sq
        a_B_data = v_b_sq_data / r_data  # a_B

        # Analisis untuk setiap model
        for model_name in ['ΛCDM NFW', 'ΛCDM Core', 'MOND Std', 'MOND Arctan']:␣
↪ # Loop model
            if model_name in results and results[model_name]['params'] is not␣
↪None:  # Cek
                params = results[model_name]['params']  # Parameter

                # Hitung prediksi a_obs dari model
```

```
            if 'MOND' in model_name:  # MOND
                if model_name == 'MOND Std':  # Std
                    v_pred = self.v_total_mond_std(r_data, *params)  #␣
↪v_pred Std
                else:  # Arctan
                    v_pred = self.v_total_mond_arctan(r_data, *params)  #␣
↪v_pred Arctan
            else:  # ΛCDM
                if model_name == 'ΛCDM NFW':  # NFW
                    v_pred = self.v_total_nfw(r_data, *params)  # v_pred NFW
                else:  # Core
                    v_pred = self.v_total_core(r_data, *params)  # v_pred␣
↪Core

            a_obs_pred = (v_pred**2) / r_data  # a_obs pred

            # Hitung a_B dari model ini (untuk plotting)
            Sigma_be, a_b, Sigma_d0, a_d = params[:4]  # Baryonik
            v_b_sq_model = self.v_baryon_sq(r_data, Sigma_be, a_b,␣
↪Sigma_d0, a_d)  # v_b_sq model
            a_B_model = v_b_sq_model / r_data  # a_B model

            # === METRIK STATISTIK UTAMA ===

            # 1. Deviasi absolut
            deviations = a_obs_data - a_obs_pred  # Deviasi
            abs_deviations = np.abs(deviations)  # Absolut

            # 2. Deviasi relatif (%)
            rel_deviations = (deviations / a_obs_data) * 100  # Relatif
            abs_rel_deviations = np.abs(rel_deviations)  # Absolut relatif

            # 3. Residual terstandarisasi (menggunakan error observasi)
            # Konversi error kecepatan ke error percepatan
            sigma_a_obs = (2 * v_data * sigma_v_data) / r_data  # Propagasi␣
↪error

            # Hindari division by zero
            mask = sigma_a_obs > 0  # Mask error >0
            if np.sum(mask) > 0:  # Jika ada data valid
                standardized_residuals = deviations[mask] /␣
↪sigma_a_obs[mask]  # Residual standar
                chi_sq = np.sum(standardized_residuals**2)  # Chi sq
                chi_sq_reduced = chi_sq / (np.sum(mask) - len(params))  #␣
↪Reduced
            else:  # Tidak ada
```

```
                chi_sq = float('nan')  # Nan
                chi_sq_reduced = float('nan')  # Nan

            print(f"\n--- {model_name} ---")  # Header model
            print(f"Jumlah data points: {len(a_obs_data)}")  # Jumlah point
            print(f"Rata-rata deviasi absolut: {np.mean(abs_deviations):.
↪3e}")  # Rata absolut
            print(f"Std dev deviasi absolut: {np.std(deviations):.3e}")  #␣
↪Std dev
            print(f"Deviasi maksimum absolut: {np.max(abs_deviations):.
↪3e}")  # Max absolut
            print(f"Rata-rata deviasi relatif: {np.mean(abs_rel_deviations):
↪.2f}%")  # Rata relatif
            print(f"Deviasi relatif maksimum: {np.max(abs_rel_deviations):.
↪2f}%")  # Max relatif

            if not np.isnan(chi_sq):  # Jika chi_sq valid
                print(f"Chi-square: {chi_sq:.2f}")  # Cetak chi sq
                print(f"Reduced chi-square: {chi_sq_reduced:.2f}")  #␣
↪Reduced

            # 4. Korelasi a_B vs deviasi
            correlation = np.corrcoef(a_B_data, abs_rel_deviations)[0, 1] ␣
↪# Korelasi
            print(f"Korelasi |a_B| vs |deviasi|: {correlation:.3f}")  #␣
↪Cetak korelasi

            # 5. Fraksi data dalam toleransi
            within_5_percent = np.sum(abs_rel_deviations <= 5) /␣
↪len(abs_rel_deviations) * 100  # Within 5%
            within_10_percent = np.sum(abs_rel_deviations <= 10) /␣
↪len(abs_rel_deviations) * 100  # Within 10%
            within_20_percent = np.sum(abs_rel_deviations <= 20) /␣
↪len(abs_rel_deviations) * 100  # Within 20%

            print(f"Data within 5%: {within_5_percent:.1f}%")  # Cetak 5%
            print(f"Data within 10%: {within_10_percent:.1f}%")  # Cetak 10%
            print(f"Data within 20%: {within_20_percent:.1f}%")  # Cetak 20%

            # 6. Deviasi berdasarkan regime percepatan
            mask_high_accel = a_B_data > A0_STD_ASTRO  # High accel
            mask_low_accel = a_B_data <= A0_STD_ASTRO  # Low accel

            if np.sum(mask_high_accel) > 0:  # Jika ada high
                dev_high = np.mean(abs_rel_deviations[mask_high_accel])  #␣
↪Dev high
```

```
                    print(f"Deviasi rata-rata (a_B > a0): {dev_high:.2f}%")  #␣
↪Cetak

                if np.sum(mask_low_accel) > 0:  # Jika ada low
                    dev_low = np.mean(abs_rel_deviations[mask_low_accel])  #␣
↪Dev low
                    print(f"Deviasi rata-rata (a_B   a0): {dev_low:.2f}%")  #␣
↪Cetak

                # 7. Uji normalitas residual
                if len(deviations) > 3:  # Minimal data
                    shapiro_stat, shapiro_p = stats.shapiro(deviations)  # Uji␣
↪Shapiro
                    print(f"Shapiro-Wilk test (normalitas): p-value =␣
↪{shapiro_p:.4f}")  # Cetak p-value
                    if shapiro_p > 0.05:  # Interpretasi
                        print("  → Residual terdistribusi normal (p > 0.05)") ␣
↪# Normal
                    else:  # Tidak normal
                        print("  → Residual TIDAK normal (p   0.05)")  # Tidak␣
↪normal

        print(f"\n{'='*60}")  # Footer

    def plot_mond_std_specific_test_with_stats(self, galaxy, results):  #␣
↪Fungsi untuk plot uji MOND Std dengan statistik
        """Plot uji spesifik MOND Standar dengan analisis statistik deviasi."""␣
↪ # Docstring fungsi
        self.plot_mond_std_specific_test(galaxy, results)  # Panggil plot dasar

        # === ANALISIS STATISTIK DEVIASI ===
        print(f"\n{'='*60}")  # Header
        print(f"ANALISIS STATISTIK DEVIASI - UJI SPESIFIK MOND STD - {galaxy.
↪upper()}")  # Judul
        print(f"{'='*60}")  # Pemisah

        data = self.data[galaxy]  # Data
        r_data, v_data, sigma_v_data = data['r'], data['v'], data['sigma_v']  #␣
↪Pisah
        a_obs_data = (v_data**2) / r_data  # a_obs

        # Gunakan parameter baryonik dari model mana pun untuk menghitung a_B␣
↪data
        base_model_for_data = None  # Inisialisasi
        for candidate in ['MOND Std', 'MOND Arctan', 'ΛCDM NFW', 'ΛCDM Core']: ␣
↪# Loop
```

```
            if candidate in results and results[candidate]['params'] is not␣
↪None:  # Cek
                base_model_for_data = candidate  # Set
                break  # Keluar
        if base_model_for_data is None:  # Gagal
            print("Tidak dapat menghitung a_B data: tidak ada model yang␣
↪berhasil difitting.")  # Info
            return  # Keluar

        Sigma_be_d, a_b_d, Sigma_d0_d, a_d_d =␣
↪results[base_model_for_data]['params'][:4]  # Parameter
        v_b_sq_data = self.v_baryon_sq(r_data, Sigma_be_d, a_b_d, Sigma_d0_d,␣
↪a_d_d)  # v_b_sq
        a_B_data = v_b_sq_data / r_data  # a_B

        # Hitung a_M (MOND Std) untuk data
        term_std_data = 1 + (2 * A0_STD_ASTRO / a_B_data)**2  # Istilah
        a_M_std_data = (a_B_data / np.sqrt(2)) * np.sqrt(1 + np.
↪sqrt(term_std_data))  # a_M

        # Rasio data
        ratio_data_std = a_obs_data / a_M_std_data  # Rasio

        # Analisis untuk SEMUA model (MOND Std, MOND Arctan, ΛCDM NFW, ΛCDM␣
↪Core)
        for model_name in ['MOND Std', 'MOND Arctan', 'ΛCDM NFW', 'ΛCDM Core']:␣
↪ # Loop
            if model_name in results and results[model_name]['params'] is not␣
↪None:  # Cek
                params = results[model_name]['params']  # Parameter

                # Hitung prediksi a_obs dari model
                if model_name == 'MOND Std':  # Std
                    v_pred = self.v_total_mond_std(r_data, *params)  # v_pred
                elif model_name == 'MOND Arctan':  # Arctan
                    v_pred = self.v_total_mond_arctan(r_data, *params)  # v_pred
                elif model_name == 'ΛCDM NFW':  # NFW
                    v_pred = self.v_total_nfw(r_data, *params)  # v_pred
                else:  # Core
                    v_pred = self.v_total_core(r_data, *params)  # v_pred

                a_obs_pred = (v_pred**2) / r_data  # a_obs pred

                # Hitung a_B dari model ini
                Sigma_be, a_b, Sigma_d0, a_d = params[:4]  # Baryonik
```

```
                v_b_sq_model = self.v_baryon_sq(r_data, Sigma_be, a_b,␣
↪Sigma_d0, a_d)  # v_b_sq
                a_B_model = v_b_sq_model / r_data  # a_B

                # Hitung a_M_std untuk model ini (menggunakan rumus MOND Std)
                term_std_model = 1 + (2 * A0_STD_ASTRO / a_B_model)**2  #␣
↪Istilah
                a_M_std_model = (a_B_model / np.sqrt(2)) * np.sqrt(1 + np.
↪sqrt(term_std_model))  # a_M

                # Rasio prediksi
                ratio_pred = a_obs_pred / a_M_std_model  # Rasio

                # Deviasi dari garis referensi (y = 1)
                deviations = ratio_pred - 1  # Deviasi
                abs_deviations = np.abs(deviations)  # Absolut
                rel_deviations = deviations * 100  # Relatif (%)

                print(f"\n--- {model_name} (vs MOND Std) ---")  # Header
                print(f"Jumlah data points: {len(ratio_pred)}")  # Jumlah
                print(f"Rata-rata deviasi absolut dari 1: {np.
↪mean(abs_deviations):.4f}")  # Rata absolut
                print(f"Std dev deviasi: {np.std(deviations):.4f}")  # Std dev
                print(f"Deviasi maksimum: {np.max(abs_deviations):.4f}")  # Max
                print(f"Rata-rata deviasi relatif: {np.mean(np.
↪abs(rel_deviations)):.2f}%")  # Rata relatif
                print(f"Deviasi relatif maksimum: {np.max(np.
↪abs(rel_deviations)):.2f}%")  # Max relatif

                # Fraksi data dalam toleransi
                within_5_percent = np.sum(np.abs(rel_deviations) <= 5) /␣
↪len(rel_deviations) * 100  # 5%
                within_10_percent = np.sum(np.abs(rel_deviations) <= 10) /␣
↪len(rel_deviations) * 100  # 10%
                within_20_percent = np.sum(np.abs(rel_deviations) <= 20) /␣
↪len(rel_deviations) * 100  # 20%

                print(f"Data within 5% dari y=1: {within_5_percent:.1f}%")  #␣
↪Cetak 5%
                print(f"Data within 10% dari y=1: {within_10_percent:.1f}%")  #␣
↪Cetak 10%
                print(f"Data within 20% dari y=1: {within_20_percent:.1f}%")  #␣
↪Cetak 20%

                # Uji normalitas
                if len(deviations) > 3:  # Minimal
```

```
                    shapiro_stat, shapiro_p = stats.shapiro(deviations)  # Uji
                    print(f"Shapiro-Wilk test (normalitas): p-value =␣
↪{shapiro_p:.4f}")  # p-value
                    if shapiro_p > 0.05:  # Interpretasi
                        print("  → Residual terdistribusi normal (p > 0.05)") ␣
↪# Normal
                    else:  # Tidak
                        print("  → Residual TIDAK normal (p   0.05)")  # Tidak␣
↪normal

        print(f"\n{'='*60}")  # Footer

    def plot_mond_arctan_specific_test_with_stats(self, galaxy, results):  #␣
↪Fungsi untuk plot uji MOND Arctan dengan statistik
        """Plot uji spesifik MOND Arctan dengan analisis statistik deviasi.""" ␣
↪# Docstring fungsi
        self.plot_mond_arctan_specific_test(galaxy, results)  # Panggil plot␣
↪dasar

        # === ANALISIS STATISTIK DEVIASI ===
        print(f"\n{'='*60}")  # Header
        print(f"ANALISIS STATISTIK DEVIASI - UJI SPESIFIK MOND ARCTAN - {galaxy.
↪upper()}")  # Judul
        print(f"{'='*60}")  # Pemisah

        data = self.data[galaxy]  # Data
        r_data, v_data, sigma_v_data = data['r'], data['v'], data['sigma_v']  #␣
↪Pisah
        a_obs_data = (v_data**2) / r_data  # a_obs

        # Gunakan parameter baryonik dari model mana pun untuk menghitung a_B␣
↪data
        base_model_for_data = None  # Inisialisasi
        for candidate in ['MOND Std', 'MOND Arctan', 'ΛCDM NFW', 'ΛCDM Core']: ␣
↪# Loop
            if candidate in results and results[candidate]['params'] is not␣
↪None:  # Cek
                base_model_for_data = candidate  # Set
                break  # Keluar
        if base_model_for_data is None:  # Gagal
            print("Tidak dapat menghitung a_B data: tidak ada model yang␣
↪berhasil difitting.")  # Info
            return  # Keluar

        Sigma_be_d, a_b_d, Sigma_d0_d, a_d_d =␣
↪results[base_model_for_data]['params'][:4]  # Parameter
```

```
        v_b_sq_data = self.v_baryon_sq(r_data, Sigma_be_d, a_b_d, Sigma_d0_d,␣
↪a_d_d)  # v_b_sq
        a_B_data = v_b_sq_data / r_data  # a_B

        # Hitung a_M (MOND Arctan) untuk data
        a_M_arctan_data = np.array([self.solve_a_m_arctan(ab) for ab in␣
↪a_B_data])  # a_M

        # Rasio data
        ratio_data_arctan = a_obs_data / a_M_arctan_data  # Rasio

        # Analisis untuk SEMUA model (MOND Std, MOND Arctan, ΛCDM NFW, ΛCDM␣
↪Core)
        for model_name in ['MOND Std', 'MOND Arctan', 'ΛCDM NFW', 'ΛCDM Core']:␣
↪ # Loop
            if model_name in results and results[model_name]['params'] is not␣
↪None:  # Cek
                params = results[model_name]['params']  # Parameter

                # Hitung prediksi a_obs dari model
                if model_name == 'MOND Std':  # Std
                    v_pred = self.v_total_mond_std(r_data, *params)  # v_pred
                elif model_name == 'MOND Arctan':  # Arctan
                    v_pred = self.v_total_mond_arctan(r_data, *params)  # v_pred
                elif model_name == 'ΛCDM NFW':  # NFW
                    v_pred = self.v_total_nfw(r_data, *params)  # v_pred
                else:  # Core
                    v_pred = self.v_total_core(r_data, *params)  # v_pred

                a_obs_pred = (v_pred**2) / r_data  # a_obs pred

                # Hitung a_B dari model ini
                Sigma_be, a_b, Sigma_d0, a_d = params[:4]  # Baryonik
                v_b_sq_model = self.v_baryon_sq(r_data, Sigma_be, a_b,␣
↪Sigma_d0, a_d)  # v_b_sq
                a_B_model = v_b_sq_model / r_data  # a_B

                # Hitung a_M_arctan untuk model ini (menggunakan rumus MOND␣
↪Arctan)
                a_M_arctan_model = np.array([self.solve_a_m_arctan(ab) for ab␣
↪in a_B_model])  # a_M

                # Rasio prediksi
                ratio_pred = a_obs_pred / a_M_arctan_model  # Rasio

                # Deviasi dari garis referensi (y = 1)
```

```
                deviations = ratio_pred - 1  # Deviasi
                abs_deviations = np.abs(deviations)  # Absolut
                rel_deviations = deviations * 100  # Relatif (%)

                print(f"\n--- {model_name} (vs MOND Arctan) ---")  # Header
                print(f"Jumlah data points: {len(ratio_pred)}")  # Jumlah
                print(f"Rata-rata deviasi absolut dari 1: {np.mean(abs_deviations):.4f}")  # Rata absolut
                print(f"Std dev deviasi: {np.std(deviations):.4f}")  # Std dev
                print(f"Deviasi maksimum: {np.max(abs_deviations):.4f}")  # Max
                print(f"Rata-rata deviasi relatif: {np.mean(np.abs(rel_deviations)):.2f}%")  # Rata relatif
                print(f"Deviasi relatif maksimum: {np.max(np.abs(rel_deviations)):.2f}%")  # Max relatif

                # Fraksi data dalam toleransi
                within_5_percent = np.sum(np.abs(rel_deviations) <= 5) / len(rel_deviations) * 100  # 5%
                within_10_percent = np.sum(np.abs(rel_deviations) <= 10) / len(rel_deviations) * 100  # 10%
                within_20_percent = np.sum(np.abs(rel_deviations) <= 20) / len(rel_deviations) * 100  # 20%

                print(f"Data within 5% dari y=1: {within_5_percent:.1f}%")  # Cetak 5%
                print(f"Data within 10% dari y=1: {within_10_percent:.1f}%")  # Cetak 10%
                print(f"Data within 20% dari y=1: {within_20_percent:.1f}%")  # Cetak 20%

                # Uji normalitas
                if len(deviations) > 3:  # Minimal
                    shapiro_stat, shapiro_p = stats.shapiro(deviations)  # Uji
                    print(f"Shapiro-Wilk test (normalitas): p-value = {shapiro_p:.4f}")  # p-value
                    if shapiro_p > 0.05:  # Interpretasi
                        print("  → Residual terdistribusi normal (p > 0.05)")  # Normal
                    else:  # Tidak
                        print("  → Residual TIDAK normal (p  0.05)")  # Tidak normal

        print(f"\n{'='*60}")  # Footer

    def plot_residuals_with_stats(self, galaxy, results):  # Fungsi untuk plot residual dengan statistik
```

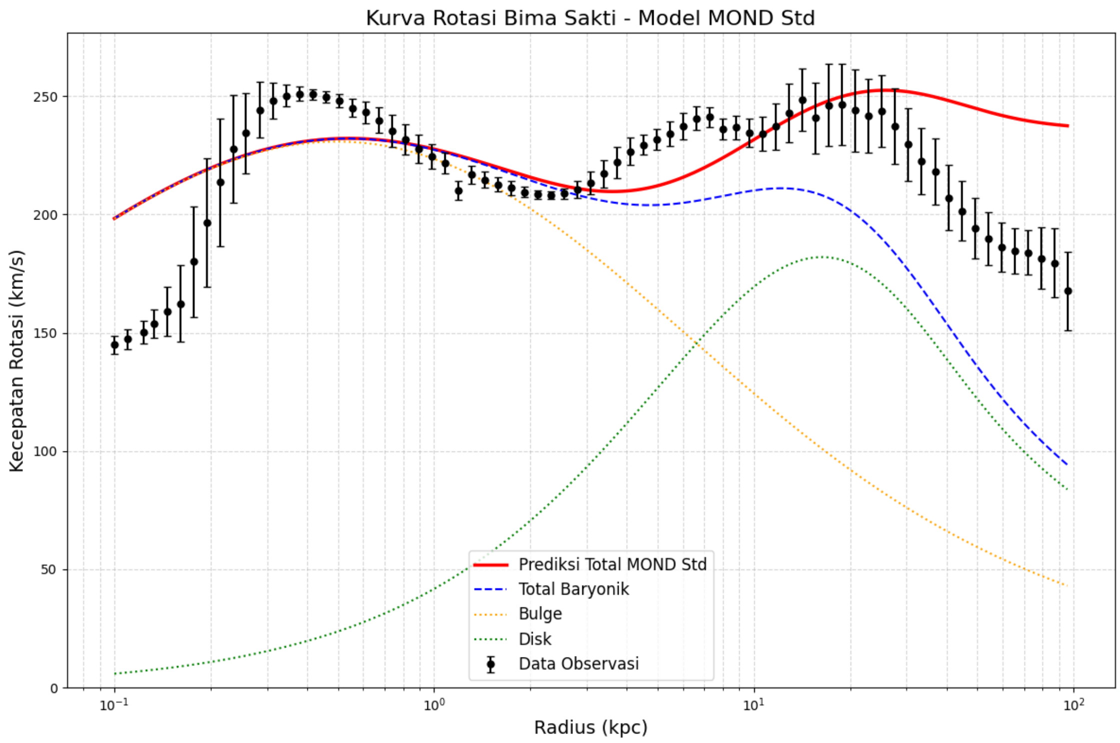
```python
        """Plot residual dengan analisis statistik deviasi lengkap."""  # Docstring fungsi
        self.plot_residuals(galaxy, results)  # Panggil plot residual dasar (asumsi fungsi ini ada di kode asli, tapi tidak lengkap; diasumsikan)

        # === ANALISIS STATISTIK RESIDUAL UNTUK SETIAP MODEL ===
        print(f"\n{'='*60}")  # Header
        print(f"ANALISIS STATISTIK RESIDUAL - {galaxy.upper()}")  # Judul
        print(f"{'='*60}")  # Pemisah

        data = self.data[galaxy]  # Data
        r_data, v_data, sigma_v_data = data['r'], data['v'], data['sigma_v']  # Pisah

        for model_name in ['ΛCDM NFW', 'ΛCDM Core', 'MOND Std', 'MOND Arctan']:  # Loop
            if model_name in results and results[model_name]['params'] is not None:  # Cek
                params = results[model_name]['params']  # Parameter

                # Hitung prediksi v dari model
                if 'MOND' in model_name:  # MOND
                    if model_name == 'MOND Std':  # Std
                        v_pred = self.v_total_mond_std(r_data, *params)  # v_pred
                    else:  # Arctan
                        v_pred = self.v_total_mond_arctan(r_data, *params)  # v_pred
                else:  # ΛCDM
                    if model_name == 'ΛCDM NFW':  # NFW
                        v_pred = self.v_total_nfw(r_data, *params)  # v_pred
                    else:  # Core
                        v_pred = self.v_total_core(r_data, *params)  # v_pred

                residuals = v_data - v_pred  # Residual
                abs_residuals = np.abs(residuals)  # Absolut
                rel_residuals = (residuals / v_data) * 100  # Relatif (%)
                abs_rel_residuals = np.abs(rel_residuals)  # Absolut relatif

                print(f"\n--- {model_name} ---")  # Header
                print(f"Jumlah data points: {len(v_data)}")  # Jumlah
                print(f"Rata-rata residual: {np.mean(residuals):.2f} km/s")  # Rata residual
                print(f"Std dev residual: {np.std(residuals):.2f} km/s")  # Std dev
```

```
                print(f"Rata-rata residual absolut: {np.mean(abs_residuals):.
↪2f} km/s")  # Rata absolut
                print(f"Residual maksimum: {np.max(abs_residuals):.2f} km/s") ␣
↪# Max
                print(f"Rata-rata residual relatif: {np.mean(abs_rel_residuals):
↪.2f}%")  # Rata relatif
                print(f"Residual relatif maksimum: {np.max(abs_rel_residuals):.
↪2f}%")  # Max relatif

                # Residual terstandarisasi
                standardized_residuals = residuals / sigma_v_data  # Standar
                chi_sq = np.sum(standardized_residuals**2)  # Chi sq
                chi_sq_reduced = chi_sq / (len(v_data) - len(params))  # Reduced

                print(f"Chi-square: {chi_sq:.2f}")  # Cetak
                print(f"Reduced chi-square: {chi_sq_reduced:.2f}")  # Cetak

                # Fraksi data dalam toleransi
                within_5_percent = np.sum(abs_rel_residuals <= 5) /␣
↪len(abs_rel_residuals) * 100  # 5%
                within_10_percent = np.sum(abs_rel_residuals <= 10) /␣
↪len(abs_rel_residuals) * 100  # 10%
                within_20_percent = np.sum(abs_rel_residuals <= 20) /␣
↪len(abs_rel_residuals) * 100  # 20%

                print(f"Data within 5%: {within_5_percent:.1f}%")  # Cetak
                print(f"Data within 10%: {within_10_percent:.1f}%")  # Cetak
                print(f"Data within 20%: {within_20_percent:.1f}%")  # Cetak

                # Uji normalitas
                if len(residuals) > 3:  # Minimal
                    shapiro_stat, shapiro_p = stats.shapiro(residuals)  # Uji
                    print(f"Shapiro-Wilk test (normalitas): p-value =␣
↪{shapiro_p:.4f}")  # p-value
                    if shapiro_p > 0.05:  # Interpretasi
                        print("  → Residual terdistribusi normal (p > 0.05)") ␣
↪# Normal
                    else:  # Tidak
                        print("  → Residual TIDAK normal (p   0.05)")  # Tidak␣
↪normal

        print(f"\n{'='*60}")  # Footer

    # Modifikasi fungsi run_analysis untuk menggunakan fungsi-fungsi baru
    def run_analysis(self):  # Fungsi utama untuk jalankan analisis
```

```
    """Eksekusi utama analisis tesis: fitting, perbandingan, dan␣
↪visualisasi."""  # Docstring fungsi
    print("Memulai analisis kurva rotasi galaksi...")  # Cetak pesan mulai
    all_results = {}  # Dictionary untuk simpan semua hasil

    for galaxy in ['Bima Sakti', 'M31']:  # Loop untuk dua galaksi
        print(f"\n{'#' * 50}")  # Header galaksi
        print(f" MEMPROSES GALAKSI: {galaxy.upper()} ")  # Judul galaksi
        print(f"{'#' * 50}")  # Pemisah

        galaxy_results = {}  # Hasil per galaksi
        for model_name in self.models.keys():  # Loop model
            params, cov, param_names = self.analyze_model(galaxy,␣
↪model_name)  # Analisis model
            galaxy_results[model_name] = {  # Simpan hasil
                'params': params,
                'covariance': cov,
                'param_names': param_names,
                'fit_function': self.models[model_name]
            }

        all_results[galaxy] = galaxy_results  # Simpan ke all_results

        # Tambahkan pemanggilan compare_models untuk pasangan model yang␣
↪diinginkan
        self.compare_models(galaxy, 'ΛCDM NFW', 'MOND Std', galaxy_results)␣
↪ # Bandingkan NFW vs Std
        self.compare_models(galaxy, 'ΛCDM Core', 'MOND Std',␣
↪galaxy_results)  # Bandingkan Core vs Std
        self.compare_models(galaxy, 'ΛCDM NFW', 'ΛCDM Core',␣
↪galaxy_results)  # Bandingkan NFW vs Core
        self.compare_models(galaxy, 'MOND Std', 'MOND Arctan',␣
↪galaxy_results)  # Bandingkan Std vs Arctan
        self.compare_models(galaxy, 'ΛCDM NFW', 'MOND Arctan',␣
↪galaxy_results)  # Bandingkan NFW vs Arctan
        self.compare_models(galaxy, 'ΛCDM Core', 'MOND Arctan',␣
↪galaxy_results)  # Bandingkan Core vs Arctan

        # Analisis komparatif untuk MOND
        mond_std_params = galaxy_results['MOND Std']['params']  # Params Std
        mond_arctan_params = galaxy_results['MOND Arctan']['params']  #␣
↪Params Arctan

        # Tampilkan hasil visualisasi dan statistik
        if any(res['params'] is not None for res in galaxy_results.
↪values()):  # Cek jika ada model sukses
```

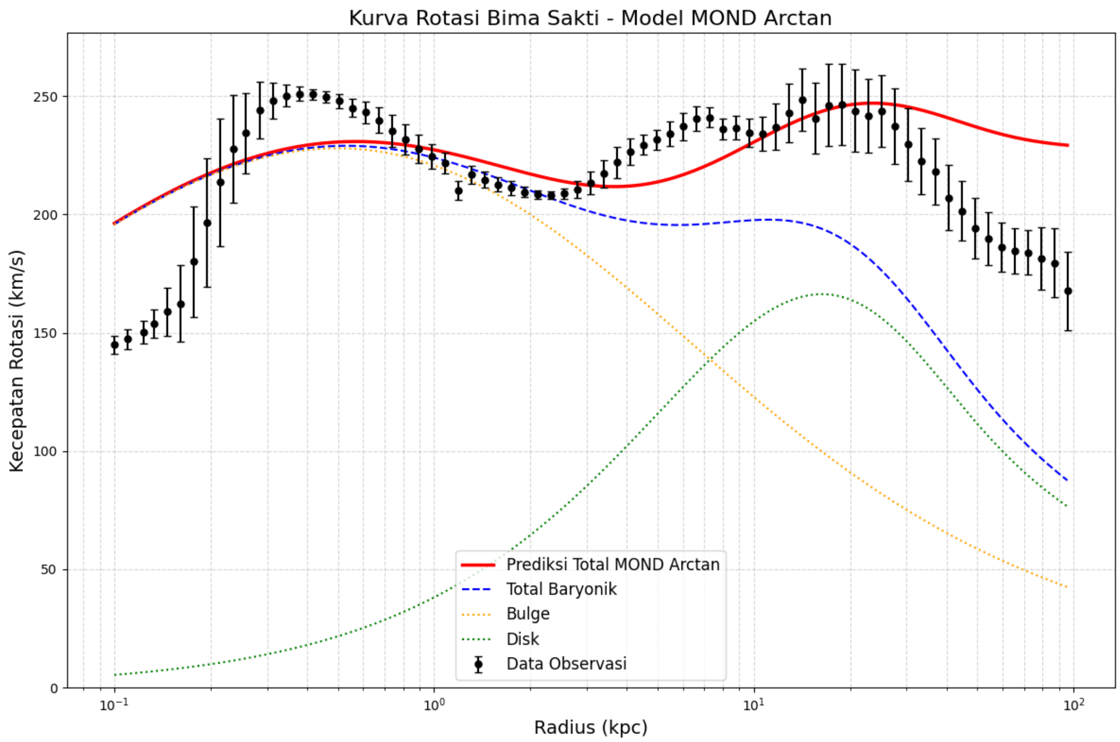
```
                print(f"\n--- VISUALISASI DAN STATISTIK UNTUK {galaxy.upper()}␣
↪---")  # Header visualisasi
                self.compare_model_statistics(galaxy, galaxy_results)  #␣
↪Bandingkan statistik
                self.plot_all_models_comparison(galaxy, galaxy_results)  # Plot␣
↪semua model

                # Gunakan fungsi-fungsi baru dengan analisis statistik
                self.plot_baryonic_hypothesis_test_with_stats(galaxy,␣
↪galaxy_results)  # Plot baryonik dengan stats
                self.plot_residuals_with_stats(galaxy, galaxy_results)  # Plot␣
↪residual dengan stats

                # Plot uji spesifik MOND yang terpisah
                self.plot_mond_std_specific_test_with_stats(galaxy,␣
↪galaxy_results)  # Plot Std dengan stats
                self.plot_mond_arctan_specific_test_with_stats(galaxy,␣
↪galaxy_results)  # Plot Arctan dengan stats
            else:  # Tidak ada sukses
                print(f"\n[INFO] Tidak ada model yang berhasil difitting untuk␣
↪{galaxy}. Visualisasi dilewati.")  # Info skip

        print("\nAnalisis selesai!")  # Pesan selesai

# ================================= EKSEKUSI PROGRAM␣
↪=====================================
if __name__ == "__main__":  # Cek jika skrip dijalankan langsung
    analyzer = GalaxyRotationAnalyzer()  # Buat instance analyzer
    analyzer.run_analysis()  # Jalankan analisis
```

```
Memulai analisis kurva rotasi galaksi…

##################################################
 MEMPROSES GALAKSI: BIMA SAKTI
##################################################

================================================================================
ANALISIS: Model MOND Std untuk Galaksi Bima Sakti
================================================================================
[SUCCESS] Fitting berhasil.

--- PARAMETER FITTING (MOND) ---
Sigma_be (M_sun/kpc^2): 6.36e+08 ± 9.59e+07
a_b (kpc): 1.69 ± 0.24
Sigma_d0 (M_sun/kpc^2): 4.17e+08 ± 9.07e+07
a_d (kpc): 7.59 ± 1.30
```

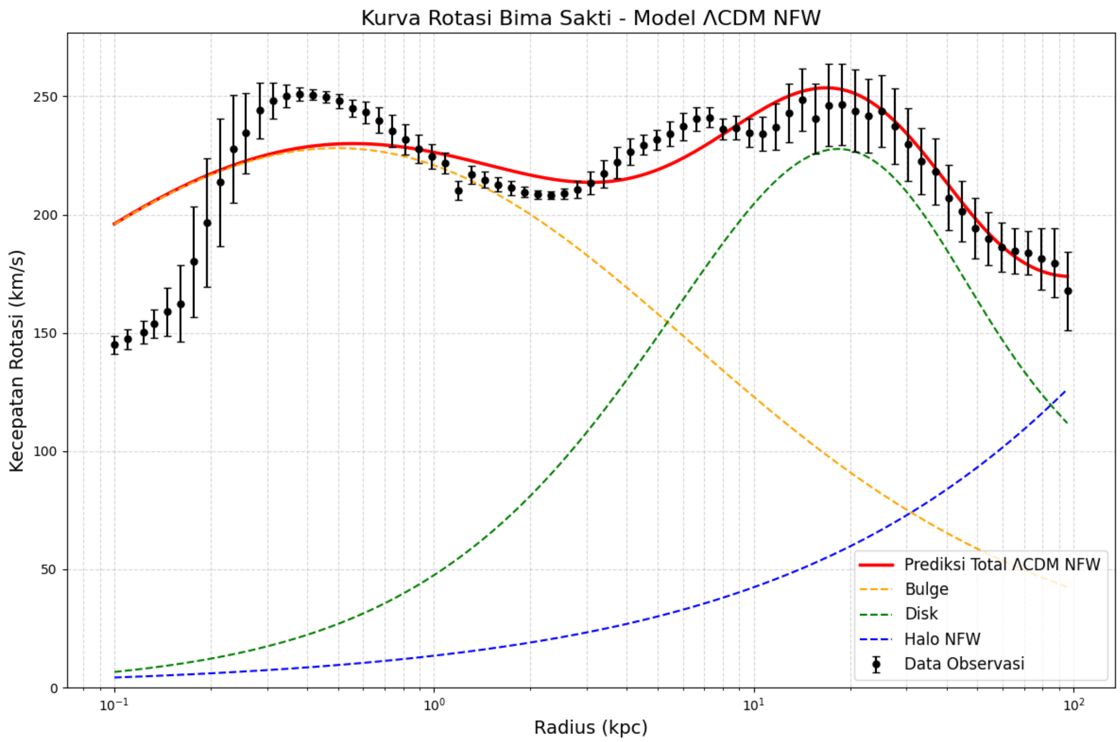



```
================================================================================
ANALISIS: Model MOND Arctan untuk Galaksi Bima Sakti
================================================================================
[SUCCESS] Fitting berhasil.

--- PARAMETER FITTING (MOND) ---
Sigma_be (M_sun/kpc^2): 6.21e+08 ± 9.21e+07
a_b (kpc): 1.69 ± 0.23
Sigma_d0 (M_sun/kpc^2): 3.48e+08 ± 8.24e+07
a_d (kpc): 7.59 ± 1.37

--- NILAI a_M ARCTANGENT DARI METODE ITERASI (untuk 5 radius sampel) ---

Iterasi Newton-Raphson untuk a_M (arctan):
Estimasi awal u: 1.137840e+02
Iterasi 1: u = 1.137840e+02, delta = 9.773390e+00
Iterasi 2: u = 1.040106e+02, delta = 1.055978e-07
Konvergen setelah 2 iterasi.
a_M akhir: 3.851294e+05 (dalam satuan astro)
Radius 0.100 kpc: a_b = 3.836287e+05, a_M = 3.851294e+05

Iterasi Newton-Raphson untuk a_M (arctan):
Estimasi awal u: 3.052007e+01
```

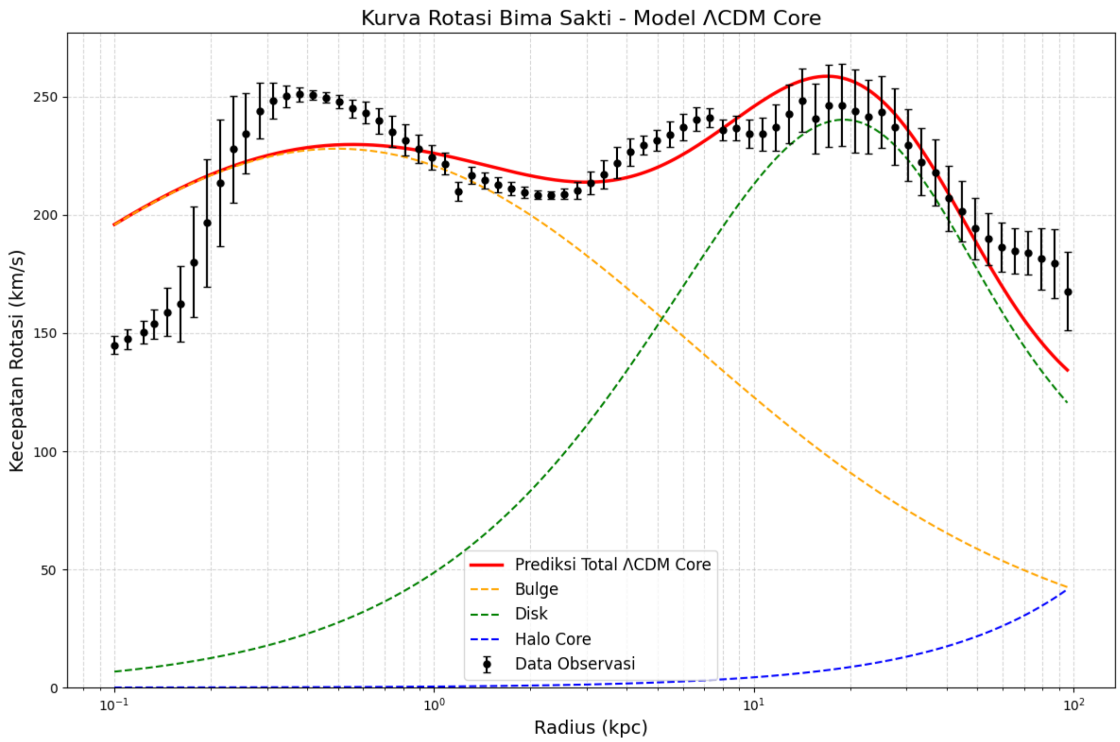
```
Iterasi 1: u = 3.052007e+01, delta = 4.641869e+00
Iterasi 2: u = 2.587820e+01, delta = 5.093456e-06
Iterasi 3: u = 2.587819e+01, delta = -3.552736e-15
Konvergen setelah 3 iterasi.
a_M akhir: 9.582149e+04 (dalam satuan astro)
Radius 0.556 kpc: a_b = 9.432111e+04, a_M = 9.582149e+04

Iterasi Newton-Raphson untuk a_M (arctan):
Estimasi awal u: 5.414400e+00
Iterasi 1: u = 5.414400e+00, delta = 1.477570e+00
Iterasi 2: u = 3.936830e+00, delta = 6.243516e-04
Iterasi 3: u = 3.936205e+00, delta = 2.536403e-10
Konvergen setelah 3 iterasi.
a_M akhir: 1.457494e+04 (dalam satuan astro)
Radius 3.091 kpc: a_b = 1.308714e+04, a_M = 1.457494e+04

Iterasi Newton-Raphson untuk a_M (arctan):
Estimasi awal u: 1.346656e+00
Iterasi 1: u = 1.346656e+00, delta = 3.990241e-01
Iterasi 2: u = 9.476318e-01, delta = 8.380743e-03
Iterasi 3: u = 9.392510e-01, delta = 7.471908e-06
Iterasi 4: u = 9.392435e-01, delta = 6.036710e-12
Konvergen setelah 4 iterasi.
a_M akhir: 3.477821e+03 (dalam satuan astro)
Radius 17.187 kpc: a_b = 2.158976e+03, a_M = 3.477821e+03

Iterasi Newton-Raphson untuk a_M (arctan):
Estimasi awal u: 1.689186e-01
Iterasi 1: u = 1.689186e-01, delta = 1.923505e-02
Iterasi 2: u = 1.496836e-01, delta = 1.128885e-03
Iterasi 3: u = 1.485547e-01, delta = 3.991821e-06
Iterasi 4: u = 1.485507e-01, delta = 4.996675e-11
Konvergen setelah 4 iterasi.
a_M akhir: 5.500519e+02 (dalam satuan astro)
Radius 95.561 kpc: a_b = 8.027421e+01, a_M = 5.500519e+02
```




```
================================================================================
ANALISIS: Model ΛCDM NFW untuk Galaksi Bima Sakti
================================================================================
[SUCCESS] Fitting berhasil.

--- PARAMETER FITTING (ΛCDM NFW) ---
Sigma_be (M_sun/kpc^2): 6.21e+08 ± 9.47e+07
a_b (kpc): 1.69 ± 0.25
Sigma_d0 (M_sun/kpc^2): 5.81e+08 ± 1.06e+08
a_d (kpc): 8.54 ± 2.24
rho_0 (M_sun/kpc^3): 4.77e+03 ± 3.43e+05
h (kpc): 1410.00 ± 93951.30
```




```
================================================================================
ANALISIS: Model ΛCDM Core untuk Galaksi Bima Sakti
================================================================================
[SUCCESS] Fitting berhasil.

--- PARAMETER FITTING (ΛCDM Core) ---
Sigma_be (M_sun/kpc^2): 6.21e+08 ± 8.74e+07
a_b (kpc): 1.69 ± 0.22
Sigma_d0 (M_sun/kpc^2): 6.22e+08 ± 7.56e+07
a_d (kpc): 8.87 ± 1.46
rho_0 (M_sun/kpc^3): 1.05e+04 ± 2.09e+05
h (kpc): 1410.00 ± 72176978.20
```




```
================================================================================
PERBANDINGAN BIMA SAKTI: ΛCDM NFW vs MOND Std
================================================================================
Model                     |            ² |          AIC |          BIC |           R²
--------------------------|-------------|-------------|-------------|-------------
ΛCDM NFW                  |     1064.76 |     1076.76 |     1090.50 |      0.5879
MOND Std                  |     1332.56 |     1340.56 |     1349.72 |     -0.0165

Perbedaan:
Δ ² (ΛCDM NFW - MOND Std): -267.80
ΔAIC (ΛCDM NFW - MOND Std): -263.80
ΔBIC (ΛCDM NFW - MOND Std): -259.22

Interpretasi Berdasarkan AIC:
  Bukti sangat kuat ΛCDM NFW lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 0.0000
  Perbedaan signifikan (p < 0.05). Model terbaik: ΛCDM NFW
================================================================================


================================================================================
```

PERBANDINGAN BIMA SAKTI: ΛCDM Core vs MOND Std

Model	^{2}	AIC	BIC	R^2
ΛCDM Core	1101.47	1113.47	1127.21	0.4793
MOND Std	1332.56	1340.56	1349.72	-0.0165

Perbedaan:
Δ ^{2} (ΛCDM Core - MOND Std): -231.10
ΔAIC (ΛCDM Core - MOND Std): -227.10
ΔBIC (ΛCDM Core - MOND Std): -222.51

Interpretasi Berdasarkan AIC:
Bukti sangat kuat ΛCDM Core lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 0.0000
Perbedaan signifikan ($p < 0.05$). Model terbaik: ΛCDM Core

PERBANDINGAN BIMA SAKTI: ΛCDM NFW vs ΛCDM Core

Model	^{2}	AIC	BIC	R^2
ΛCDM NFW	1064.76	1076.76	1090.50	0.5879
ΛCDM Core	1101.47	1113.47	1127.21	0.4793

Perbedaan:
Δ ^{2} (ΛCDM NFW - ΛCDM Core): -36.70
ΔAIC (ΛCDM NFW - ΛCDM Core): -36.70
ΔBIC (ΛCDM NFW - ΛCDM Core): -36.70

Interpretasi Berdasarkan AIC:
Bukti sangat kuat ΛCDM NFW lebih baik.

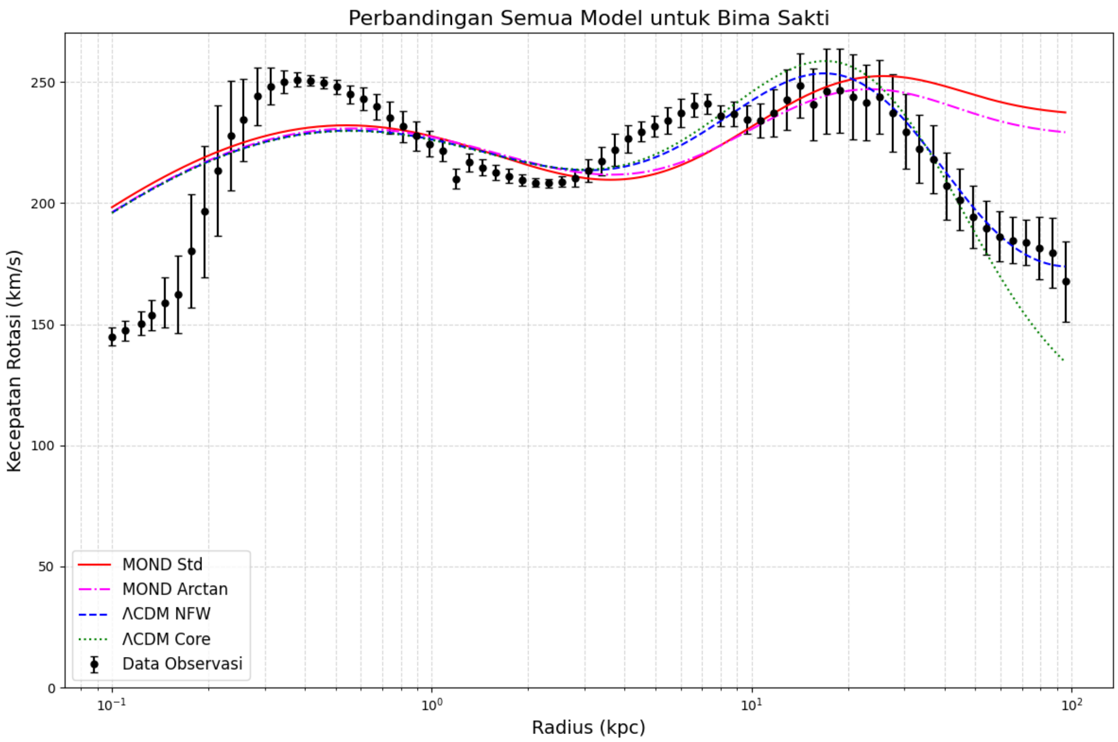
Uji Signifikansi Chi-kuadrat:
P-value: 1.0000
Tidak ada perbedaan signifikan (p 0.05)

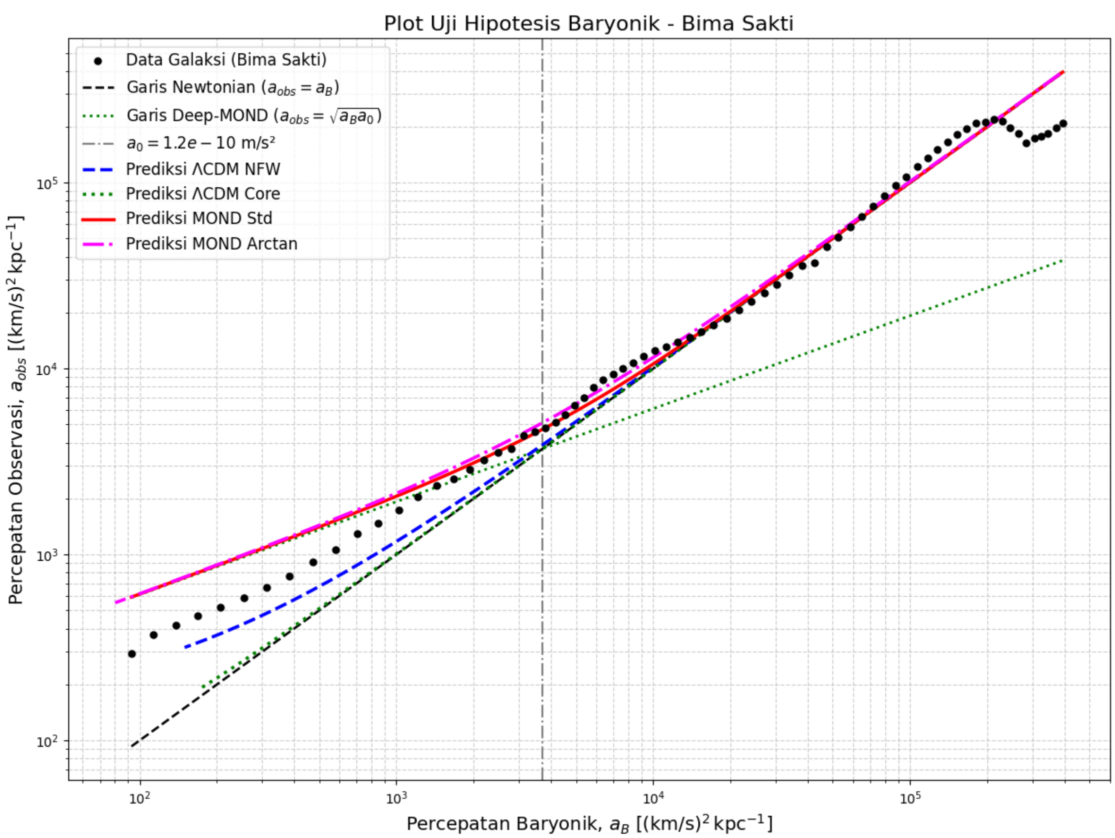
PERBANDINGAN BIMA SAKTI: MOND Std vs MOND Arctan

Model	^{2}	AIC	BIC	R^2

```
MOND Std             |    1332.56 |    1340.56 |    1349.72 |    -0.0165
MOND Arctan          |    1282.76 |    1290.76 |    1299.93 |     0.1654

Perbedaan:
Δ ² (MOND Std - MOND Arctan): 49.80
ΔAIC (MOND Std - MOND Arctan): 49.80
ΔBIC (MOND Std - MOND Arctan): 49.80

Interpretasi Berdasarkan AIC:
  Bukti sangat kuat MOND Arctan lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 1.0000
  Tidak ada perbedaan signifikan (p   0.05)
================================================================================


================================================================================
PERBANDINGAN BIMA SAKTI: ΛCDM NFW vs MOND Arctan
================================================================================
Model                |          ² |        AIC |        BIC |         R²
---------------------|------------|------------|------------|------------
ΛCDM NFW             |    1064.76 |    1076.76 |    1090.50 |     0.5879
MOND Arctan          |    1282.76 |    1290.76 |    1299.93 |     0.1654

Perbedaan:
Δ ² (ΛCDM NFW - MOND Arctan): -218.00
ΔAIC (ΛCDM NFW - MOND Arctan): -214.00
ΔBIC (ΛCDM NFW - MOND Arctan): -209.42

Interpretasi Berdasarkan AIC:
  Bukti sangat kuat ΛCDM NFW lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 0.0000
  Perbedaan signifikan (p < 0.05). Model terbaik: ΛCDM NFW
================================================================================


================================================================================
PERBANDINGAN BIMA SAKTI: ΛCDM Core vs MOND Arctan
================================================================================
Model                |          ² |        AIC |        BIC |         R²
---------------------|------------|------------|------------|------------
ΛCDM Core            |    1101.47 |    1113.47 |    1127.21 |     0.4793
MOND Arctan          |    1282.76 |    1290.76 |    1299.93 |     0.1654

Perbedaan:
```

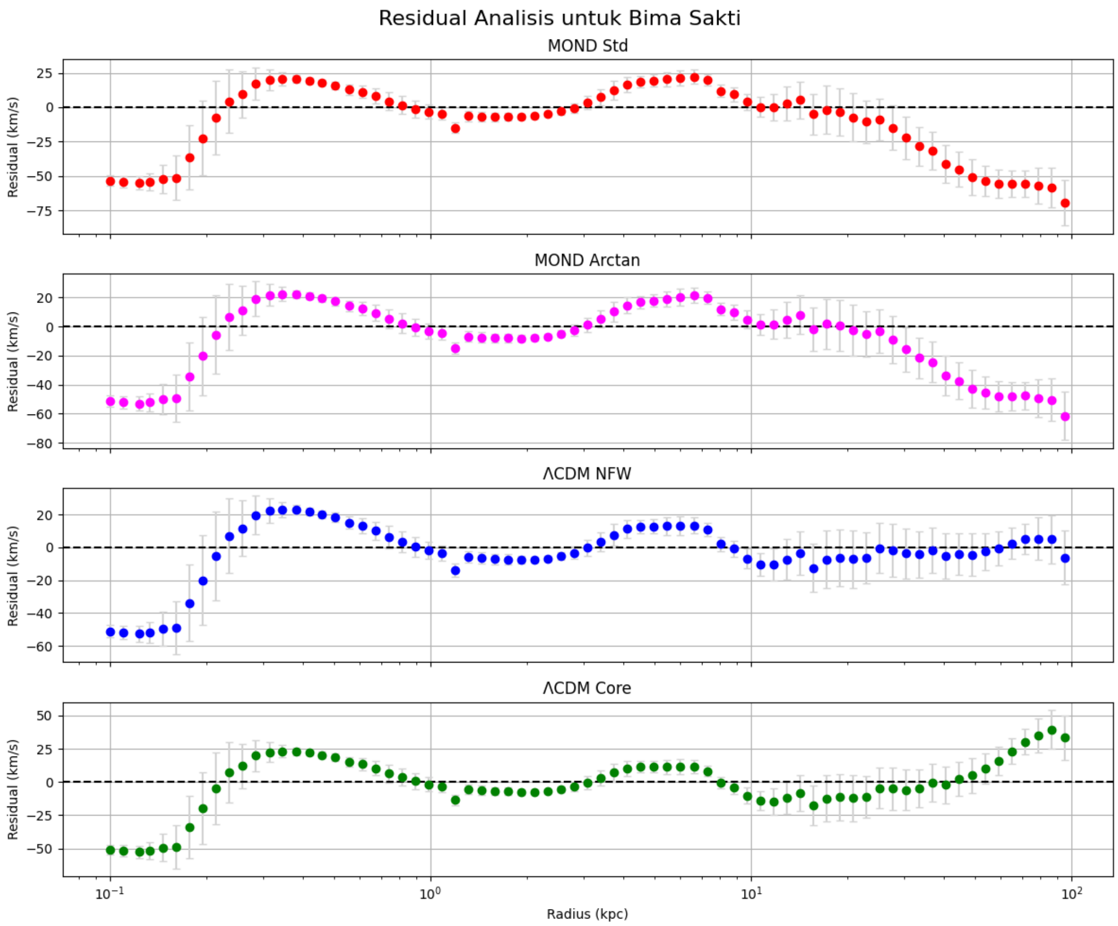
```
Δ ² (ΛCDM Core - MOND Arctan): -181.30
ΔAIC (ΛCDM Core - MOND Arctan): -177.30
ΔBIC (ΛCDM Core - MOND Arctan): -172.72

Interpretasi Berdasarkan AIC:
  Bukti sangat kuat ΛCDM Core lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 0.0000
  Perbedaan signifikan (p < 0.05). Model terbaik: ΛCDM Core
================================================================================


--- VISUALISASI DAN STATISTIK UNTUK BIMA SAKTI ---

================================================================================
EVALUASI STATISTIK UNTUK BIMA SAKTI
================================================================================
Model                |          ² |      ²/dof |        AIC |        BIC |
R²
---------------------|------------|------------|------------|------------|------
------
MOND Std             |    1332.56 |      19.31 |    1340.56 |    1349.72 |
-0.0165
MOND Arctan          |    1282.76 |      18.59 |    1290.76 |    1299.93 |
0.1654
ΛCDM NFW             |    1064.76 |      15.89 |    1076.76 |    1090.50 |
0.5879
ΛCDM Core            |    1101.47 |      16.44 |    1113.47 |    1127.21 |
0.4793
--------------------------------------------------------------------------------

Perbandingan AIC (Arctan - Std): -49.80
-> Bukti sangat kuat mendukung MOND Arctan.
Perbandingan AIC (Core - NFW): 36.70
-> Bukti sangat kuat mendukung ΛCDM NFW.
```


Perbandingan Semua Model untuk Bima Sakti
Kecepatan Rotasi (km/s)
Radius (kpc)
MOND Std
MOND Arctan
ΛCDM NFW
ΛCDM Core
Data Observasi
0
50
100
150
200
250
10^{-1}
10^{0}
10^{1}
10^{2}


Plot Uji Hipotesis Baryonik - Bima Sakti
Percepatan Observasi, a_{obs} $[(\mathrm{km/s})^2\,\mathrm{kpc}^{-1}]$
Percepatan Baryonik, a_B $[(\mathrm{km/s})^2\,\mathrm{kpc}^{-1}]$
Data Galaksi (Bima Sakti)
Garis Newtonian ($a_{obs} = a_B$)
Garis Deep-MOND ($a_{obs} = \sqrt{a_B a_0}$)
$a_0 = 1.2e-10$ m/s²
Prediksi ΛCDM NFW
Prediksi ΛCDM Core
Prediksi MOND Std
Prediksi MOND Arctan
10^{2}
10^{3}
10^{4}
10^{5}

```
============================================================
ANALISIS STATISTIK DEVIASI - UJI HIPOTESIS BARYONIK - BIMA SAKTI
============================================================

--- ΛCDM NFW ---
Jumlah data points: 73
Rata-rata deviasi absolut: 1.768e+04
Std dev deviasi absolut: 4.297e+04
Deviasi maksimum absolut: 1.744e+05
Rata-rata deviasi relatif: 13.50%
Deviasi relatif maksimum: 83.06%
Chi-square: 1253.61
Reduced chi-square: 18.71
Korelasi |a_B| vs |deviasi|: 0.862
Data within 5%: 28.8%
Data within 10%: 67.1%
Data within 20%: 89.0%
Deviasi rata-rata (a_B > a0): 17.41%
Deviasi rata-rata (a_B   a0): 4.43%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- ΛCDM Core ---
Jumlah data points: 73
Rata-rata deviasi absolut: 1.769e+04
Std dev deviasi absolut: 4.289e+04
Deviasi maksimum absolut: 1.740e+05
Rata-rata deviasi relatif: 16.16%
Deviasi relatif maksimum: 82.87%
Chi-square: 1283.15
Reduced chi-square: 19.15
Korelasi |a_B| vs |deviasi|: 0.768
Data within 5%: 23.3%
Data within 10%: 57.5%
Data within 20%: 82.2%
Deviasi rata-rata (a_B > a0): 17.45%
Deviasi rata-rata (a_B   a0): 13.15%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- MOND Std ---
Jumlah data points: 73
Rata-rata deviasi absolut: 1.818e+04
Std dev deviasi absolut: 4.481e+04
```

```
Deviasi maksimum absolut: 1.833e+05
Rata-rata deviasi relatif: 23.87%
Deviasi relatif maksimum: 100.48%
Chi-square: 1621.49
Reduced chi-square: 23.50
Korelasi |a_B| vs |deviasi|: 0.415
Data within 5%: 24.7%
Data within 10%: 46.6%
Data within 20%: 71.2%
Deviasi rata-rata (a_B > a0): 18.43%
Deviasi rata-rata (a_B   a0): 36.50%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- MOND Arctan ---
Jumlah data points: 73
Rata-rata deviasi absolut: 1.777e+04
Std dev deviasi absolut: 4.312e+04
Deviasi maksimum absolut: 1.752e+05
Rata-rata deviasi relatif: 21.69%
Deviasi relatif maksimum: 86.90%
Chi-square: 1514.33
Reduced chi-square: 21.95
Korelasi |a_B| vs |deviasi|: 0.488
Data within 5%: 26.0%
Data within 10%: 49.3%
Data within 20%: 72.6%
Deviasi rata-rata (a_B > a0): 18.20%
Deviasi rata-rata (a_B   a0): 29.80%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

============================================================
```

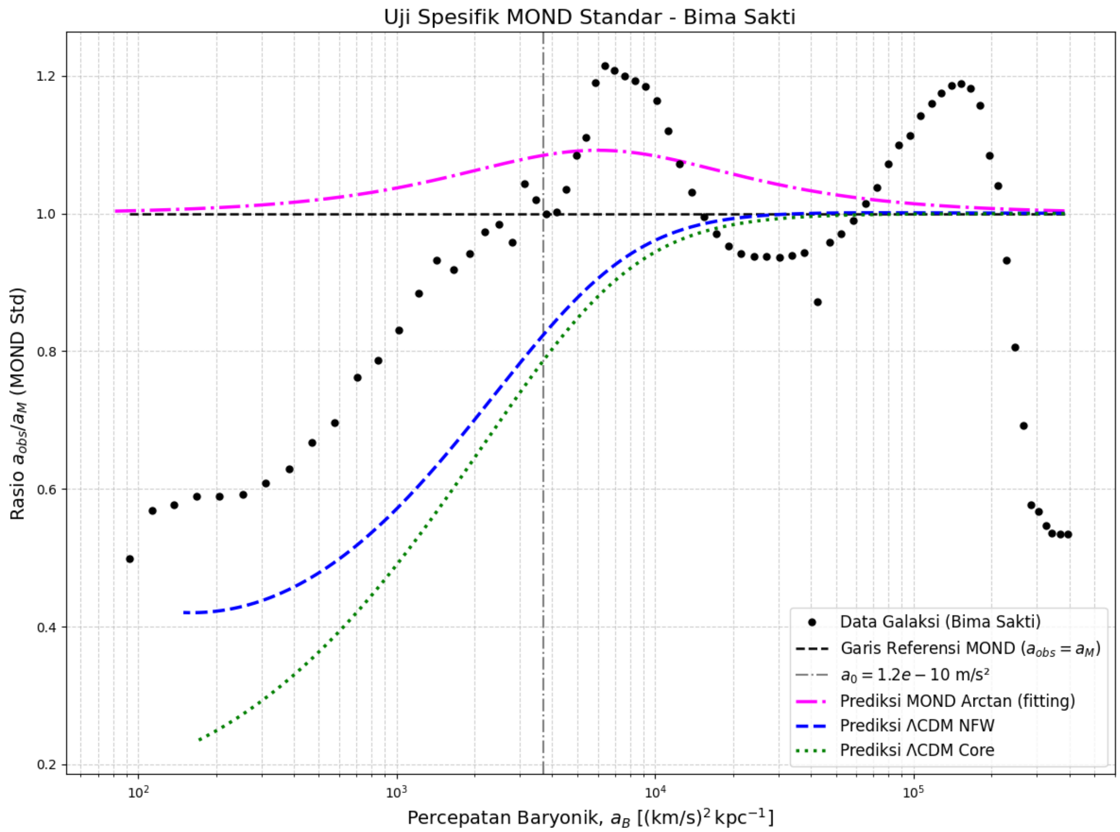



```
================================================================
ANALISIS STATISTIK RESIDUAL - BIMA SAKTI
================================================================

--- ΛCDM NFW ---
Jumlah data points: 73
Rata-rata residual: -3.10 km/s
Std dev residual: 17.79 km/s
Rata-rata residual absolut: 12.20 km/s
Residual maksimum: 52.76 km/s
Rata-rata residual relatif: 6.30%
Residual relatif maksimum: 35.30%
Chi-square: 1064.76
Reduced chi-square: 15.89
Data within 5%: 65.8%
Data within 10%: 89.0%
Data within 20%: 91.8%
Shapiro-Wilk test (normalitas): p-value = 0.0000
```

```
  → Residual TIDAK normal (p   0.05)

--- ΛCDM Core ---
Jumlah data points: 73
Rata-rata residual: -1.31 km/s
Std dev residual: 20.25 km/s
Rata-rata residual absolut: 14.95 km/s
Residual maksimum: 52.65 km/s
Rata-rata residual relatif: 7.74%
Residual relatif maksimum: 35.23%
Chi-square: 1101.47
Reduced chi-square: 16.44
Data within 5%: 57.5%
Data within 10%: 82.2%
Data within 20%: 90.4%
Shapiro-Wilk test (normalitas): p-value = 0.0003
  → Residual TIDAK normal (p   0.05)

--- MOND Std ---
Jumlah data points: 73
Rata-rata residual: -10.39 km/s
Std dev residual: 26.38 km/s
Rata-rata residual absolut: 20.78 km/s
Residual maksimum: 69.75 km/s
Rata-rata residual relatif: 10.81%
Residual relatif maksimum: 41.59%
Chi-square: 1332.56
Reduced chi-square: 19.31
Data within 5%: 45.2%
Data within 10%: 72.6%
Data within 20%: 78.1%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- MOND Arctan ---
Jumlah data points: 73
Rata-rata residual: -8.28 km/s
Std dev residual: 24.32 km/s
Rata-rata residual absolut: 19.14 km/s
Residual maksimum: 61.57 km/s
Rata-rata residual relatif: 9.95%
Residual relatif maksimum: 36.71%
Chi-square: 1282.76
Reduced chi-square: 18.59
Data within 5%: 46.6%
Data within 10%: 74.0%
Data within 20%: 80.8%
Shapiro-Wilk test (normalitas): p-value = 0.0000
```

→ Residual TIDAK normal (p 0.05)

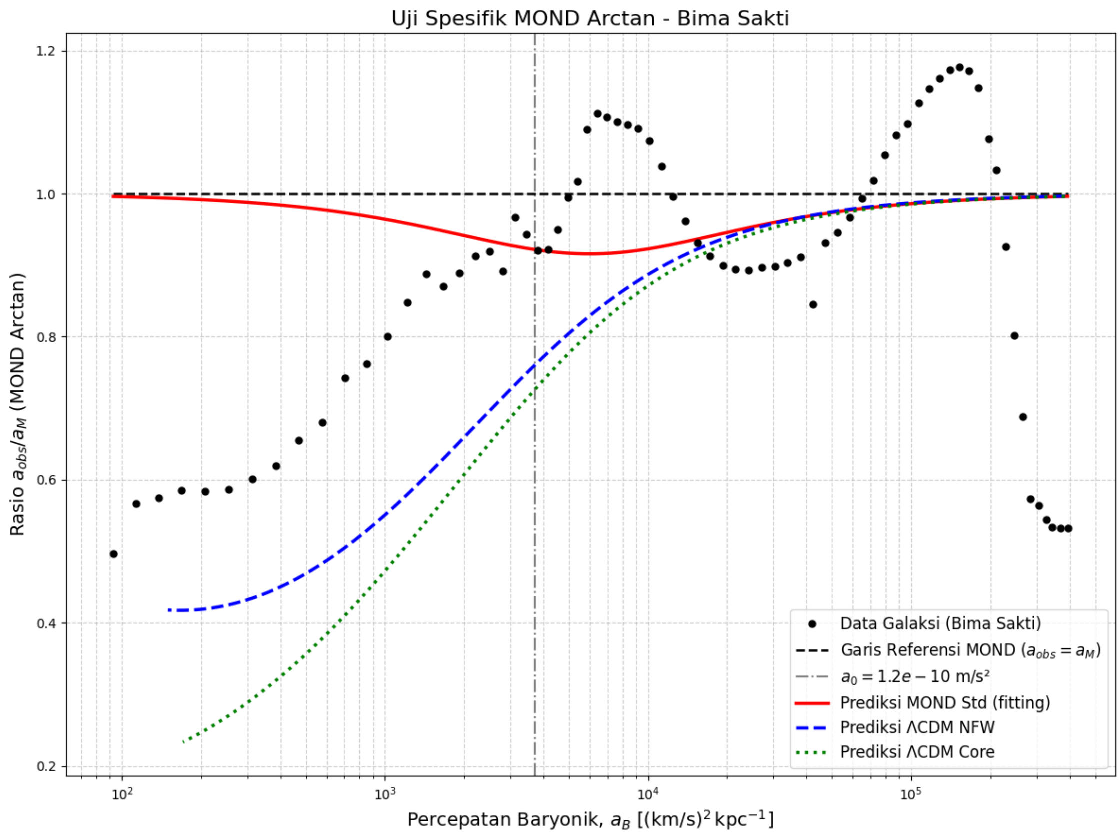
==



==
ANALISIS STATISTIK DEVIASI - UJI SPESIFIK MOND STD - BIMA SAKTI
==

--- MOND Std (vs MOND Std) ---
Jumlah data points: 73
Rata-rata deviasi absolut dari 1: 0.0000
Std dev deviasi: 0.0000
Deviasi maksimum: 0.0000
Rata-rata deviasi relatif: 0.00%
Deviasi relatif maksimum: 0.00%
Data within 5% dari y=1: 100.0%
Data within 10% dari y=1: 100.0%
Data within 20% dari y=1: 100.0%
Shapiro-Wilk test (normalitas): p-value = 0.0000
→ Residual TIDAK normal (p 0.05)

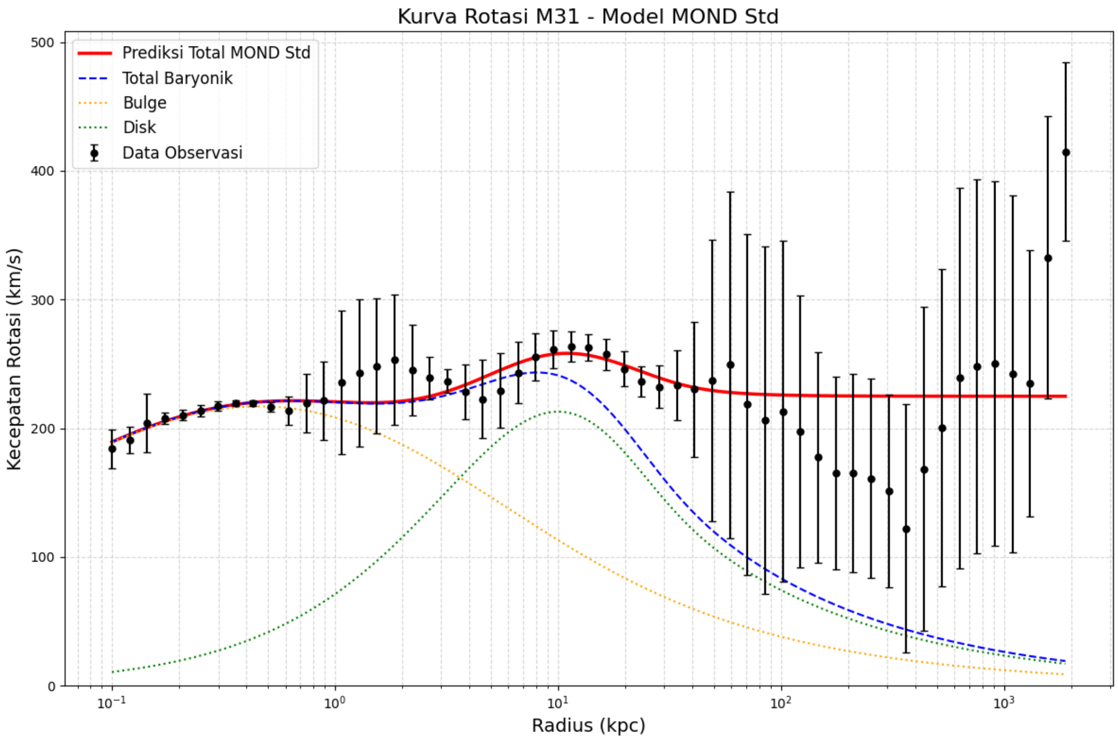
```
--- MOND Arctan (vs MOND Std) ---
Jumlah data points: 73
Rata-rata deviasi absolut dari 1: 0.0387
Std dev deviasi: 0.0314
Deviasi maksimum: 0.0919
Rata-rata deviasi relatif: 3.87%
Deviasi relatif maksimum: 9.19%
Data within 5% dari y=1: 64.4%
Data within 10% dari y=1: 100.0%
Data within 20% dari y=1: 100.0%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- ΛCDM NFW (vs MOND Std) ---
Jumlah data points: 73
Rata-rata deviasi absolut dari 1: 0.1279
Std dev deviasi: 0.1886
Deviasi maksimum: 0.5795
Rata-rata deviasi relatif: 12.79%
Deviasi relatif maksimum: 57.95%
Data within 5% dari y=1: 57.5%
Data within 10% dari y=1: 67.1%
Data within 20% dari y=1: 75.3%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- ΛCDM Core (vs MOND Std) ---
Jumlah data points: 73
Rata-rata deviasi absolut dari 1: 0.1522
Std dev deviasi: 0.2262
Deviasi maksimum: 0.7652
Rata-rata deviasi relatif: 15.22%
Deviasi relatif maksimum: 76.52%
Data within 5% dari y=1: 54.8%
Data within 10% dari y=1: 64.4%
Data within 20% dari y=1: 74.0%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

============================================================
```




```
============================================================
ANALISIS STATISTIK DEVIASI - UJI SPESIFIK MOND ARCTAN - BIMA SAKTI
============================================================

--- MOND Std (vs MOND Arctan) ---
Jumlah data points: 73
Rata-rata deviasi absolut dari 1: 0.0370
Std dev deviasi: 0.0290
Deviasi maksimum: 0.0842
Rata-rata deviasi relatif: 3.70%
Deviasi relatif maksimum: 8.42%
Data within 5% dari y=1: 64.4%
Data within 10% dari y=1: 100.0%
Data within 20% dari y=1: 100.0%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- MOND Arctan (vs MOND Arctan) ---
Jumlah data points: 73
Rata-rata deviasi absolut dari 1: 0.0000
```

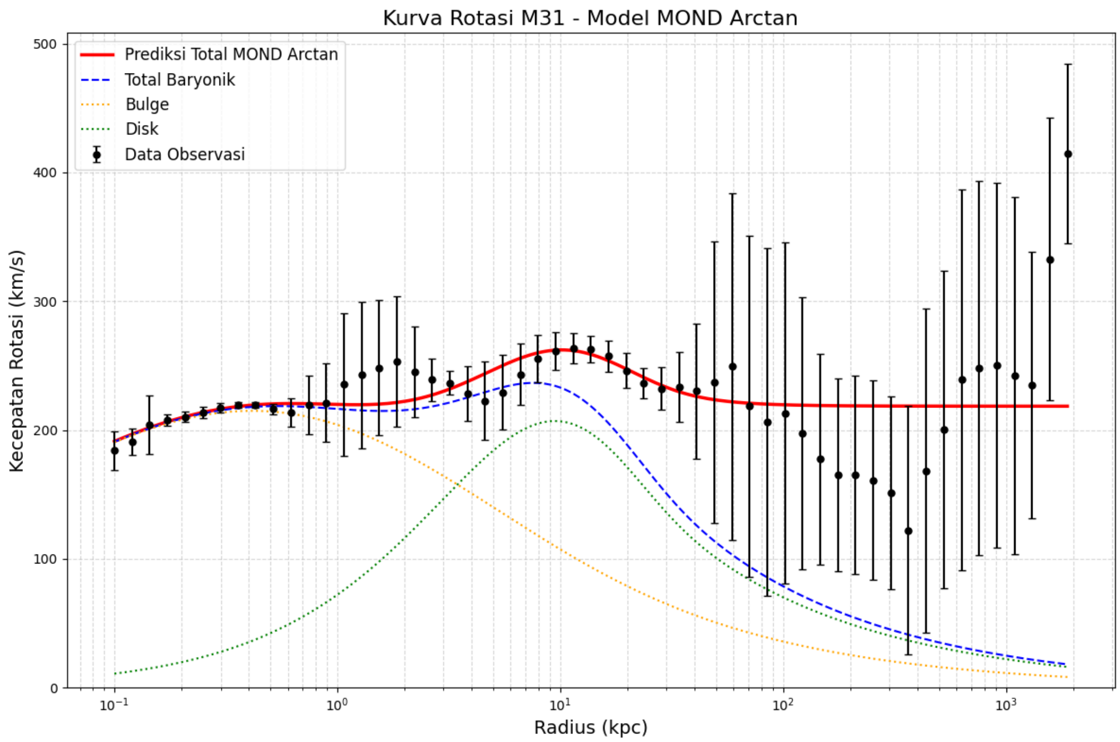
```
Std dev deviasi: 0.0000
Deviasi maksimum: 0.0000
Rata-rata deviasi relatif: 0.00%
Deviasi relatif maksimum: 0.00%
Data within 5% dari y=1: 100.0%
Data within 10% dari y=1: 100.0%
Data within 20% dari y=1: 100.0%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- ΛCDM NFW (vs MOND Arctan) ---
Jumlah data points: 73
Rata-rata deviasi absolut dari 1: 0.1623
Std dev deviasi: 0.1851
Deviasi maksimum: 0.5823
Rata-rata deviasi relatif: 16.23%
Deviasi relatif maksimum: 58.23%
Data within 5% dari y=1: 42.5%
Data within 10% dari y=1: 53.4%
Data within 20% dari y=1: 69.9%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- ΛCDM Core (vs MOND Arctan) ---
Jumlah data points: 73
Rata-rata deviasi absolut dari 1: 0.1867
Std dev deviasi: 0.2219
Deviasi maksimum: 0.7669
Rata-rata deviasi relatif: 18.67%
Deviasi relatif maksimum: 76.69%
Data within 5% dari y=1: 41.1%
Data within 10% dari y=1: 52.1%
Data within 20% dari y=1: 67.1%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

============================================================

##################################################
 MEMPROSES GALAKSI: M31
##################################################

================================================================================
ANALISIS: Model MOND Std untuk Galaksi M31
================================================================================
[SUCCESS] Fitting berhasil.

--- PARAMETER FITTING (MOND) ---
```

```
Sigma_be (M_sun/kpc^2): 6.09e+08 ± 5.07e+07
a_b (kpc): 1.56 ± 0.14
Sigma_d0 (M_sun/kpc^2): 9.30e+08 ± 9.19e+07
a_d (kpc): 4.66 ± 0.28
```

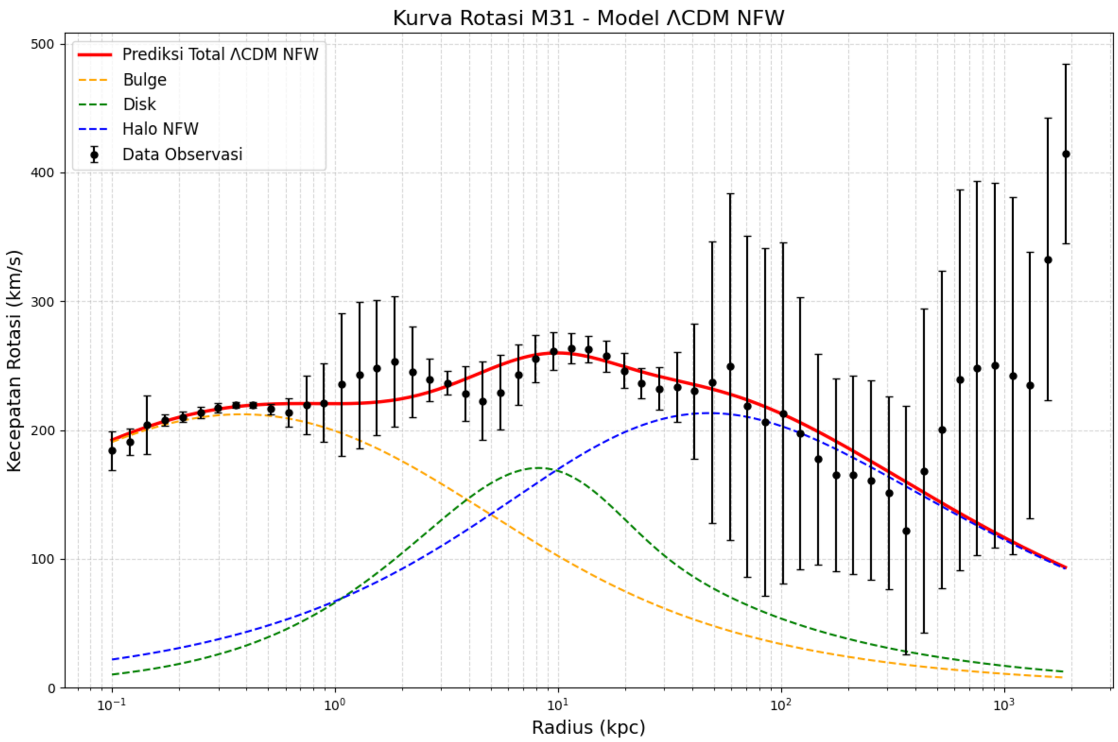



```
================================================================================
ANALISIS: Model MOND Arctan untuk Galaksi M31
================================================================================
[SUCCESS] Fitting berhasil.

--- PARAMETER FITTING (MOND) ---
Sigma_be (M_sun/kpc^2): 6.62e+08 ± 5.44e+07
a_b (kpc): 1.41 ± 0.12
Sigma_d0 (M_sun/kpc^2): 9.30e+08 ± 8.87e+07
a_d (kpc): 4.40 ± 0.25

--- NILAI a_M ARCTANGENT DARI METODE ITERASI (untuk 5 radius sampel) ---

Iterasi Newton-Raphson untuk a_M (arctan):
Estimasi awal u: 1.082172e+02
Iterasi 1: u = 1.082172e+02, delta = 9.509481e+00
Iterasi 2: u = 9.870775e+01, delta = 1.225432e-07
Konvergen setelah 2 iterasi.
```

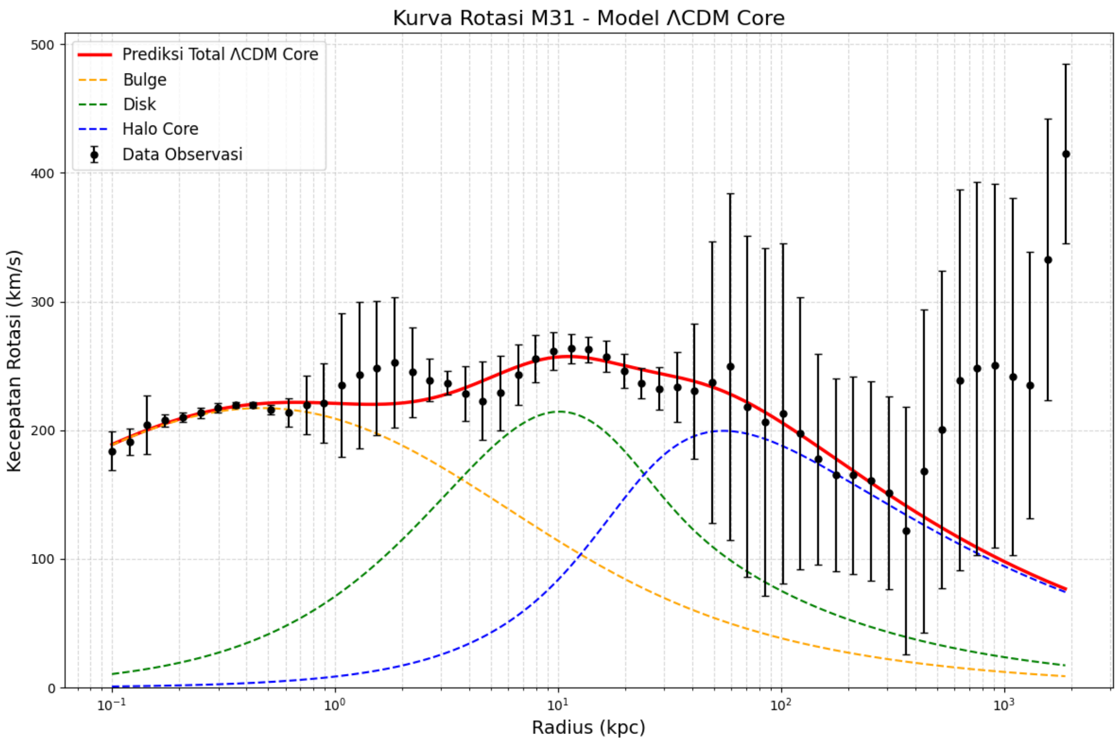
```
a_M akhir: 3.654940e+05 (dalam satuan astro)
Radius 0.100 kpc: a_b = 3.639933e+05, a_M = 3.654940e+05

Iterasi Newton-Raphson untuk a_M (arctan):
Estimasi awal u: 1.521368e+01
Iterasi 1: u = 1.521368e+01, delta = 3.027434e+00
Iterasi 2: u = 1.218625e+01, delta = 3.783367e-05
Iterasi 3: u = 1.218621e+01, delta = 1.065878e-14
Konvergen setelah 3 iterasi.
a_M akhir: 4.512296e+04 (dalam satuan astro)
Radius 1.070 kpc: a_b = 4.362364e+04, a_M = 4.512296e+04

Iterasi Newton-Raphson untuk a_M (arctan):
Estimasi awal u: 1.907561e+00
Iterasi 1: u = 1.907561e+00, delta = 5.858328e-01
Iterasi 2: u = 1.321728e+00, delta = 5.542330e-03
Iterasi 3: u = 1.316186e+00, delta = 1.136543e-06
Iterasi 4: u = 1.316185e+00, delta = 4.808814e-14
Konvergen setelah 4 iterasi.
a_M akhir: 4.873554e+03 (dalam satuan astro)
Radius 13.737 kpc: a_b = 3.475798e+03, a_M = 4.873554e+03

Iterasi Newton-Raphson untuk a_M (arctan):
Estimasi awal u: 9.555193e-02
Iterasi 1: u = 9.555193e-02, delta = 7.155132e-03
Iterasi 2: u = 8.839680e-02, delta = 2.811119e-04
Iterasi 3: u = 8.811569e-02, delta = 4.371296e-07
Konvergen setelah 3 iterasi.
a_M akhir: 3.262722e+02 (dalam satuan astro)
Radius 147.000 kpc: a_b = 2.856805e+01, a_M = 3.262722e+02

Iterasi Newton-Raphson untuk a_M (arctan):
Estimasi awal u: 6.877907e-03
Iterasi 1: u = 6.877907e-03, delta = 4.637737e-05
Iterasi 2: u = 6.831530e-03, delta = 1.573971e-07
Konvergen setelah 2 iterasi.
a_M akhir: 2.529513e+01 (dalam satuan astro)
Radius 1887.100 kpc: a_b = 1.727939e-01, a_M = 2.529513e+01
```




```
================================================================================
ANALISIS: Model ΛCDM NFW untuk Galaksi M31
================================================================================
[SUCCESS] Fitting berhasil.

--- PARAMETER FITTING (ΛCDM NFW) ---
Sigma_be (M_sun/kpc^2): 7.03e+08 ± 9.46e+07
a_b (kpc): 1.29 ± 0.19
Sigma_d0 (M_sun/kpc^2): 7.30e+08 ± 2.40e+08
a_d (kpc): 3.81 ± 0.82
rho_0 (M_sun/kpc^3): 8.03e+06 ± 9.73e+06
h (kpc): 22.00 ± 12.48
```




```
================================================================================
ANALISIS: Model ΛCDM Core untuk Galaksi M31
================================================================================
[SUCCESS] Fitting berhasil.

--- PARAMETER FITTING (ΛCDM Core) ---
Sigma_be (M_sun/kpc^2): 5.99e+08 ± 7.72e+07
a_b (kpc): 1.59 ± 0.23
Sigma_d0 (M_sun/kpc^2): 9.30e+08 ± 1.47e+08
a_d (kpc): 4.73 ± 1.05
rho_0 (M_sun/kpc^3): 4.06e+06 ± 3.36e+06
h (kpc): 22.00 ± 8.64
```



==

PERBANDINGAN M31: ΛCDM NFW vs MOND Std

==

Model	^{2}	AIC	BIC	R^2
ΛCDM NFW	38.71	50.71	62.75	-1.4759
MOND Std	21.55	29.55	37.58	0.0974

Perbedaan:
Δ ^{2} (ΛCDM NFW - MOND Std): 17.16
ΔAIC (ΛCDM NFW - MOND Std): 21.16
ΔBIC (ΛCDM NFW - MOND Std): 25.17

Interpretasi Berdasarkan AIC:
Bukti sangat kuat MOND Std lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 0.0002
Perbedaan signifikan ($p < 0.05$). Model terbaik: MOND Std

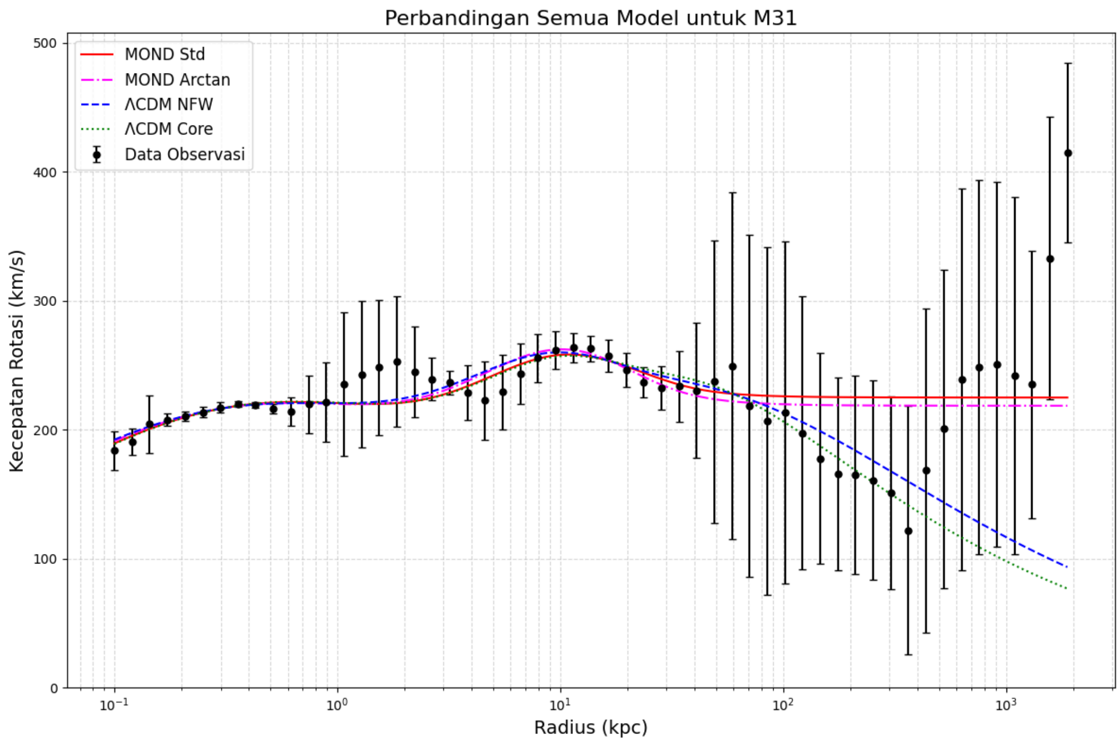
==

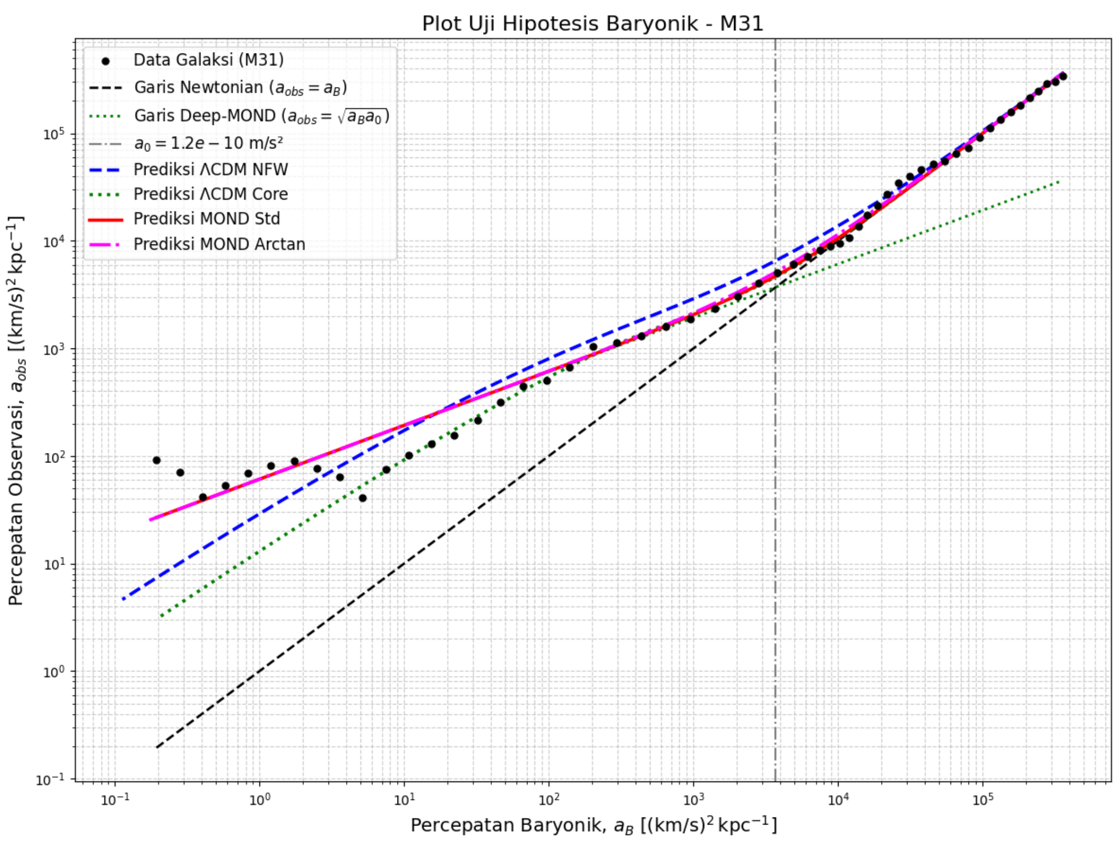
==

```
PERBANDINGAN M31: ΛCDM Core vs MOND Std
================================================================================
Model                |          ² |        AIC |        BIC |         R²
---------------------|------------|------------|------------|------------
ΛCDM Core            |      44.84 |      56.84 |      68.88 |    -1.9273
MOND Std             |      21.55 |      29.55 |      37.58 |     0.0974

Perbedaan:
Δ ² (ΛCDM Core - MOND Std): 23.29
ΔAIC (ΛCDM Core - MOND Std): 27.29
ΔBIC (ΛCDM Core - MOND Std): 31.31

Interpretasi Berdasarkan AIC:
  Bukti sangat kuat MOND Std lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 0.0000
  Perbedaan signifikan (p < 0.05). Model terbaik: MOND Std
================================================================================


================================================================================
PERBANDINGAN M31: ΛCDM NFW vs ΛCDM Core
================================================================================
Model                |          ² |        AIC |        BIC |         R²
---------------------|------------|------------|------------|------------
ΛCDM NFW             |      38.71 |      50.71 |      62.75 |    -1.4759
ΛCDM Core            |      44.84 |      56.84 |      68.88 |    -1.9273

Perbedaan:
Δ ² (ΛCDM NFW - ΛCDM Core): -6.13
ΔAIC (ΛCDM NFW - ΛCDM Core): -6.13
ΔBIC (ΛCDM NFW - ΛCDM Core): -6.13

Interpretasi Berdasarkan AIC:
  Bukti kuat ΛCDM NFW lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 1.0000
  Tidak ada perbedaan signifikan (p  0.05)
================================================================================


================================================================================
PERBANDINGAN M31: MOND Std vs MOND Arctan
================================================================================
Model                |          ² |        AIC |        BIC |         R²
---------------------|------------|------------|------------|------------
```

```
MOND Std             |      21.55 |      29.55 |      37.58 |     0.0974
MOND Arctan          |      20.03 |      28.03 |      36.06 |     0.1048

Perbedaan:
Δ ² (MOND Std - MOND Arctan): 1.52
ΔAIC (MOND Std - MOND Arctan): 1.52
ΔBIC (MOND Std - MOND Arctan): 1.52

Interpretasi Berdasarkan AIC:
  Bukti lemah MOND Arctan lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 1.0000
  Tidak ada perbedaan signifikan (p  0.05)
================================================================================


================================================================================
PERBANDINGAN M31: ΛCDM NFW vs MOND Arctan
================================================================================
Model                |          ² |        AIC |        BIC |         R²
---------------------|------------|------------|------------|------------
ΛCDM NFW             |      38.71 |      50.71 |      62.75 |    -1.4759
MOND Arctan          |      20.03 |      28.03 |      36.06 |     0.1048

Perbedaan:
Δ ² (ΛCDM NFW - MOND Arctan): 18.68
ΔAIC (ΛCDM NFW - MOND Arctan): 22.68
ΔBIC (ΛCDM NFW - MOND Arctan): 26.69

Interpretasi Berdasarkan AIC:
  Bukti sangat kuat MOND Arctan lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 0.0001
  Perbedaan signifikan (p < 0.05). Model terbaik: MOND Arctan
================================================================================


================================================================================
PERBANDINGAN M31: ΛCDM Core vs MOND Arctan
================================================================================
Model                |          ² |        AIC |        BIC |         R²
---------------------|------------|------------|------------|------------
ΛCDM Core            |      44.84 |      56.84 |      68.88 |    -1.9273
MOND Arctan          |      20.03 |      28.03 |      36.06 |     0.1048

Perbedaan:
```

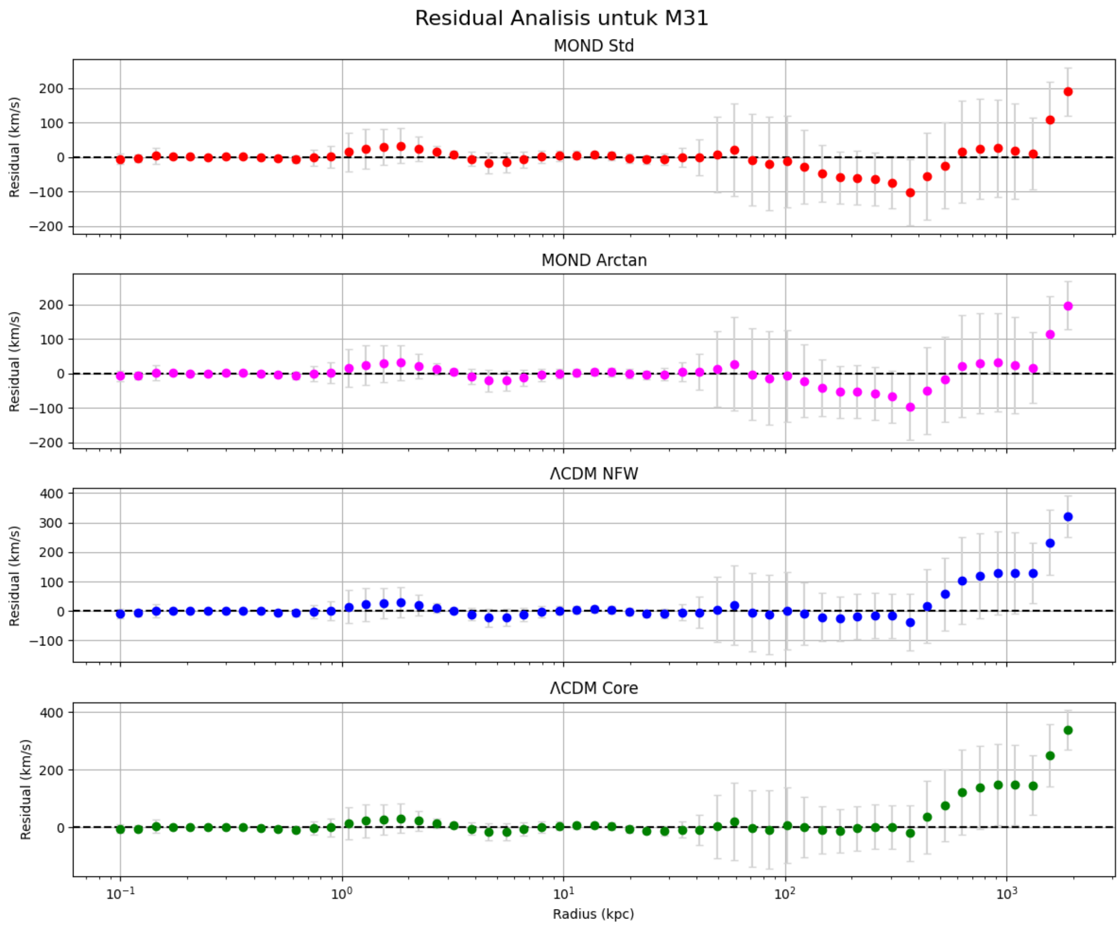
```
Δ ² (ΛCDM Core - MOND Arctan): 24.81
ΔAIC (ΛCDM Core - MOND Arctan): 28.81
ΔBIC (ΛCDM Core - MOND Arctan): 32.83

Interpretasi Berdasarkan AIC:
  Bukti sangat kuat MOND Arctan lebih baik.

Uji Signifikansi Chi-kuadrat:
P-value: 0.0000
  Perbedaan signifikan (p < 0.05). Model terbaik: MOND Arctan
================================================================================


--- VISUALISASI DAN STATISTIK UNTUK M31 ---

================================================================================
EVALUASI STATISTIK UNTUK M31
================================================================================
Model                |          ² |     ²/dof |        AIC |        BIC |
R²
---------------------|------------|------------|------------|------------|------
------
MOND Std             |      21.55 |       0.42 |      29.55 |      37.58 |
0.0974
MOND Arctan          |      20.03 |       0.39 |      28.03 |      36.06 |
0.1048
ΛCDM NFW             |      38.71 |       0.79 |      50.71 |      62.75 |
-1.4759
ΛCDM Core            |      44.84 |       0.92 |      56.84 |      68.88 |
-1.9273
--------------------------------------------------------------------------------

Perbandingan AIC (Arctan - Std): -1.52
-> Model setara.
Perbandingan AIC (Core - NFW): 6.13
-> Bukti positif mendukung ΛCDM NFW.
```


Perbandingan Semua Model untuk M31
MOND Std
MOND Arctan
ΛCDM NFW
ΛCDM Core
Data Observasi
Kecepatan Rotasi (km/s)
Radius (kpc)


Plot Uji Hipotesis Baryonik - M31
Data Galaksi (M31)
Garis Newtonian ($a_{obs} = a_B$)
Garis Deep-MOND ($a_{obs} = \sqrt{a_B a_0}$)
$a_0 = 1.2e-10$ m/s²
Prediksi ΛCDM NFW
Prediksi ΛCDM Core
Prediksi MOND Std
Prediksi MOND Arctan
Percepatan Observasi, a_{obs} [(km/s)^2 kpc^{-1}]
Percepatan Baryonik, a_B [(km/s)^2 kpc^{-1}]

```
============================================================
ANALISIS STATISTIK DEVIASI - UJI HIPOTESIS BARYONIK - M31
============================================================

--- ΛCDM NFW ---
Jumlah data points: 55
Rata-rata deviasi absolut: 2.148e+03
Std dev deviasi absolut: 5.680e+03
Deviasi maksimum absolut: 3.109e+04
Rata-rata deviasi relatif: 20.07%
Deviasi relatif maksimum: 94.93%
Chi-square: 20.90
Reduced chi-square: 0.43
Korelasi |a_B| vs |deviasi|: -0.321
Data within 5%: 40.0%
Data within 10%: 54.5%
Data within 20%: 70.9%
Deviasi rata-rata (a_B > a0): 7.22%
Deviasi rata-rata (a_B   a0): 33.40%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- ΛCDM Core ---
Jumlah data points: 55
Rata-rata deviasi absolut: 2.031e+03
Std dev deviasi absolut: 4.271e+03
Deviasi maksimum absolut: 1.848e+04
Rata-rata deviasi relatif: 18.97%
Deviasi relatif maksimum: 96.58%
Chi-square: 23.17
Reduced chi-square: 0.47
Korelasi |a_B| vs |deviasi|: -0.290
Data within 5%: 41.8%
Data within 10%: 60.0%
Data within 20%: 78.2%
Deviasi rata-rata (a_B > a0): 7.06%
Deviasi rata-rata (a_B   a0): 31.33%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- MOND Std ---
Jumlah data points: 55
Rata-rata deviasi absolut: 2.049e+03
Std dev deviasi absolut: 4.432e+03
```

```
Deviasi maksimum absolut: 2.017e+04
Rata-rata deviasi relatif: 23.65%
Deviasi relatif maksimum: 239.80%
Chi-square: 20.80
Reduced chi-square: 0.41
Korelasi |a_B| vs |deviasi|: -0.261
Data within 5%: 36.4%
Data within 10%: 52.7%
Data within 20%: 76.4%
Deviasi rata-rata (a_B > a0): 7.07%
Deviasi rata-rata (a_B   a0): 40.85%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- MOND Arctan ---
Jumlah data points: 55
Rata-rata deviasi absolut: 2.116e+03
Std dev deviasi absolut: 5.228e+03
Deviasi maksimum absolut: 2.730e+04
Rata-rata deviasi relatif: 22.33%
Deviasi relatif maksimum: 220.95%
Chi-square: 18.40
Reduced chi-square: 0.36
Korelasi |a_B| vs |deviasi|: -0.264
Data within 5%: 40.0%
Data within 10%: 50.9%
Data within 20%: 72.7%
Deviasi rata-rata (a_B > a0): 7.32%
Deviasi rata-rata (a_B   a0): 37.90%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

============================================================
```

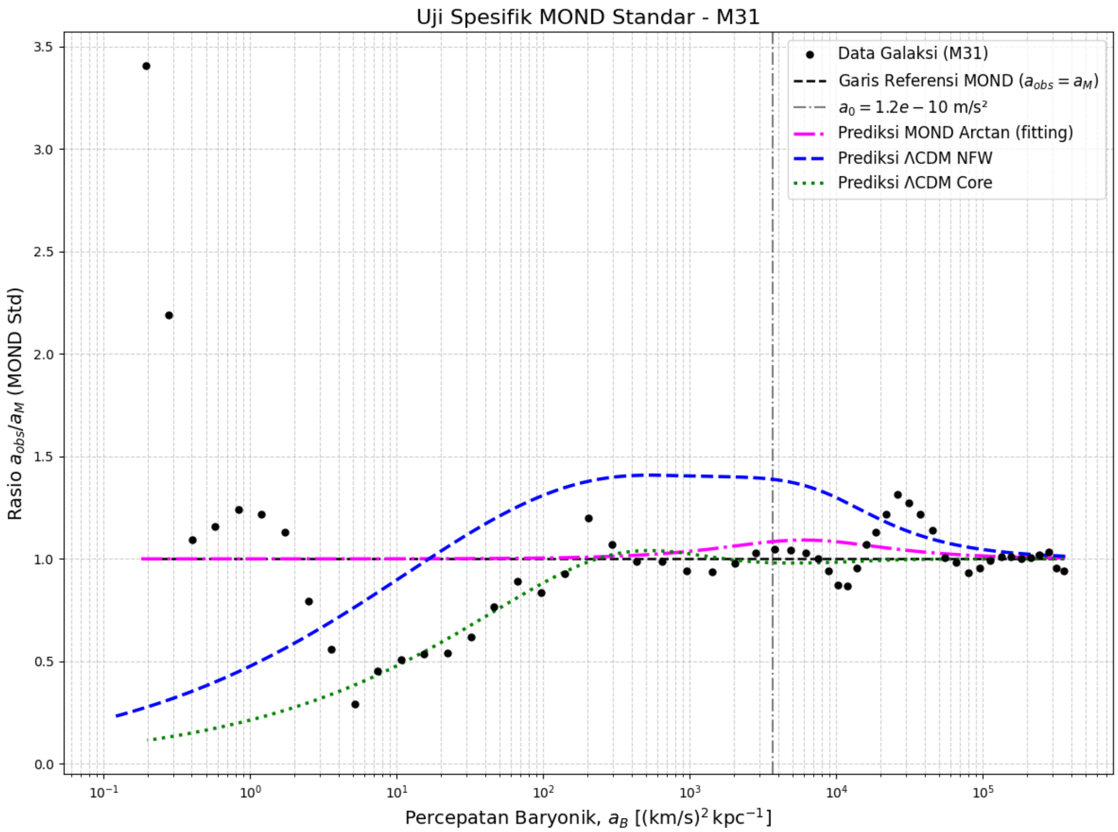



```
==============================================================
ANALISIS STATISTIK RESIDUAL - M31
==============================================================

--- ΛCDM NFW ---
Jumlah data points: 55
Rata-rata residual: 20.46 km/s
Std dev residual: 63.51 km/s
Rata-rata residual absolut: 30.91 km/s
Residual maksimum: 321.50 km/s
Rata-rata residual relatif: 12.13%
Residual relatif maksimum: 77.49%
Chi-square: 38.71
Reduced chi-square: 0.79
Data within 5%: 56.4%
Data within 10%: 69.1%
Data within 20%: 83.6%
Shapiro-Wilk test (normalitas): p-value = 0.0000
```

```
  → Residual TIDAK normal (p   0.05)

--- ΛCDM Core ---
Jumlah data points: 55
Rata-rata residual: 26.45 km/s
Std dev residual: 67.56 km/s
Rata-rata residual absolut: 32.37 km/s
Residual maksimum: 338.17 km/s
Rata-rata residual relatif: 12.32%
Residual relatif maksimum: 81.51%
Chi-square: 44.84
Reduced chi-square: 0.92
Data within 5%: 60.0%
Data within 10%: 78.2%
Data within 20%: 83.6%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- MOND Std ---
Jumlah data points: 55
Rata-rata residual: -1.04 km/s
Std dev residual: 40.27 km/s
Rata-rata residual absolut: 22.51 km/s
Residual maksimum: 190.08 km/s
Rata-rata residual relatif: 10.72%
Residual relatif maksimum: 84.34%
Chi-square: 21.55
Reduced chi-square: 0.42
Data within 5%: 52.7%
Data within 10%: 74.5%
Data within 20%: 83.6%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- MOND Arctan ---
Jumlah data points: 55
Rata-rata residual: 0.89 km/s
Std dev residual: 40.11 km/s
Rata-rata residual absolut: 22.19 km/s
Residual maksimum: 196.42 km/s
Rata-rata residual relatif: 10.35%
Residual relatif maksimum: 79.15%
Chi-square: 20.03
Reduced chi-square: 0.39
Data within 5%: 50.9%
Data within 10%: 72.7%
Data within 20%: 83.6%
Shapiro-Wilk test (normalitas): p-value = 0.0000
```

→ Residual TIDAK normal (p 0.05)

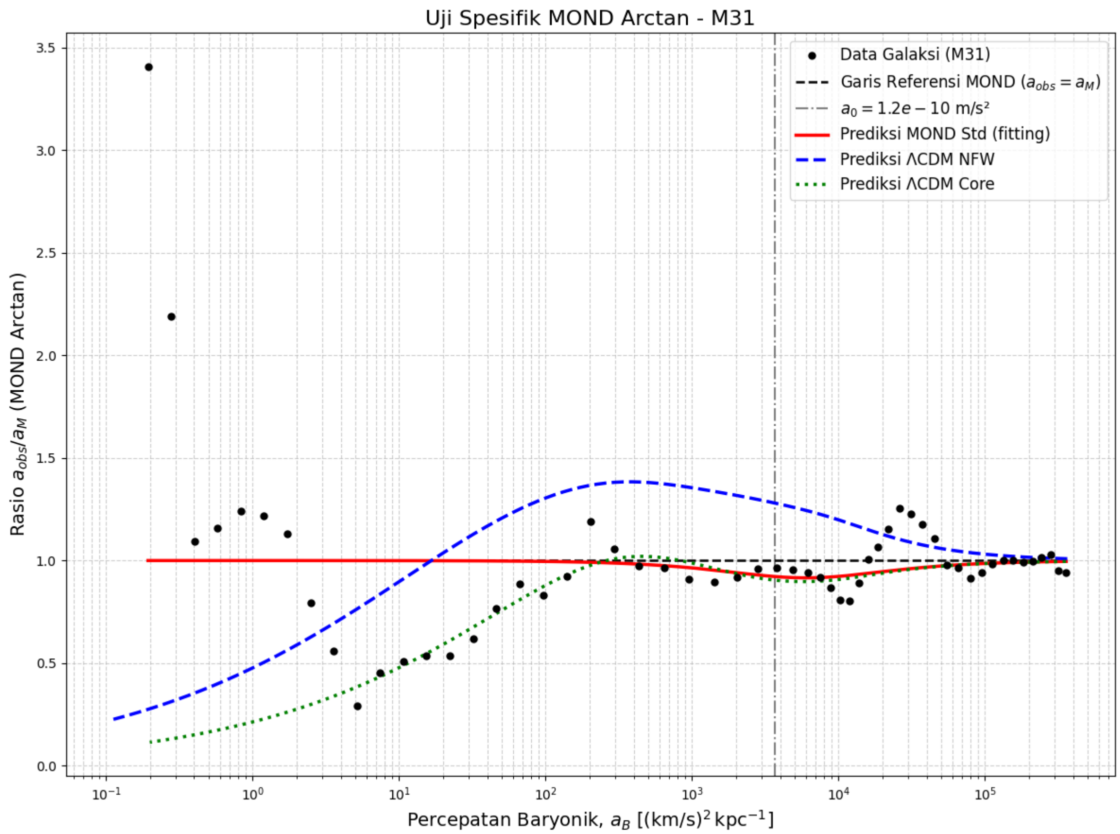
==



==
ANALISIS STATISTIK DEVIASI - UJI SPESIFIK MOND STD - M31
==

--- MOND Std (vs MOND Std) ---
Jumlah data points: 55
Rata-rata deviasi absolut dari 1: 0.0000
Std dev deviasi: 0.0000
Deviasi maksimum: 0.0000
Rata-rata deviasi relatif: 0.00%
Deviasi relatif maksimum: 0.00%
Data within 5% dari y=1: 100.0%
Data within 10% dari y=1: 100.0%
Data within 20% dari y=1: 100.0%
Shapiro-Wilk test (normalitas): p-value = 0.0001
→ Residual TIDAK normal (p 0.05)

```
--- MOND Arctan (vs MOND Std) ---
Jumlah data points: 55
Rata-rata deviasi absolut dari 1: 0.0265
Std dev deviasi: 0.0309
Deviasi maksimum: 0.0919
Rata-rata deviasi relatif: 2.65%
Deviasi relatif maksimum: 9.19%
Data within 5% dari y=1: 76.4%
Data within 10% dari y=1: 100.0%
Data within 20% dari y=1: 100.0%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- ΛCDM NFW (vs MOND Std) ---
Jumlah data points: 55
Rata-rata deviasi absolut dari 1: 0.2712
Std dev deviasi: 0.3382
Deviasi maksimum: 0.7735
Rata-rata deviasi relatif: 27.12%
Deviasi relatif maksimum: 77.35%
Data within 5% dari y=1: 18.2%
Data within 10% dari y=1: 29.1%
Data within 20% dari y=1: 41.8%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- ΛCDM Core (vs MOND Std) ---
Jumlah data points: 55
Rata-rata deviasi absolut dari 1: 0.1977
Std dev deviasi: 0.3044
Deviasi maksimum: 0.8851
Rata-rata deviasi relatif: 19.77%
Deviasi relatif maksimum: 88.51%
Data within 5% dari y=1: 65.5%
Data within 10% dari y=1: 67.3%
Data within 20% dari y=1: 70.9%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

============================================================
```




```
============================================================
ANALISIS STATISTIK DEVIASI - UJI SPESIFIK MOND ARCTAN - M31
============================================================

--- MOND Std (vs MOND Arctan) ---
Jumlah data points: 55
Rata-rata deviasi absolut dari 1: 0.0251
Std dev deviasi: 0.0286
Deviasi maksimum: 0.0842
Rata-rata deviasi relatif: 2.51%
Deviasi relatif maksimum: 8.42%
Data within 5% dari y=1: 76.4%
Data within 10% dari y=1: 100.0%
Data within 20% dari y=1: 100.0%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- MOND Arctan (vs MOND Arctan) ---
Jumlah data points: 55
Rata-rata deviasi absolut dari 1: 0.0000
```

```
Std dev deviasi: 0.0000
Deviasi maksimum: 0.0000
Rata-rata deviasi relatif: 0.00%
Deviasi relatif maksimum: 0.00%
Data within 5% dari y=1: 100.0%
Data within 10% dari y=1: 100.0%
Data within 20% dari y=1: 100.0%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- ΛCDM NFW (vs MOND Arctan) ---
Jumlah data points: 55
Rata-rata deviasi absolut dari 1: 0.2407
Std dev deviasi: 0.3167
Deviasi maksimum: 0.7735
Rata-rata deviasi relatif: 24.07%
Deviasi relatif maksimum: 77.35%
Data within 5% dari y=1: 23.6%
Data within 10% dari y=1: 34.5%
Data within 20% dari y=1: 49.1%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

--- ΛCDM Core (vs MOND Arctan) ---
Jumlah data points: 55
Rata-rata deviasi absolut dari 1: 0.2193
Std dev deviasi: 0.2905
Deviasi maksimum: 0.8851
Rata-rata deviasi relatif: 21.93%
Deviasi relatif maksimum: 88.51%
Data within 5% dari y=1: 41.8%
Data within 10% dari y=1: 63.6%
Data within 20% dari y=1: 70.9%
Shapiro-Wilk test (normalitas): p-value = 0.0000
  → Residual TIDAK normal (p   0.05)

============================================================

Analisis selesai!
```