Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

**Modele systemów dynamicznych**

Sprawozdanie z laboratorium 4

**Janusz Andrzejewski**

Nr albumu: **284052**

Kierunek: **Inżynieria systemów**

WROCŁAW 2025

**MODEL RUCHU W POLU CENTRALNYM**

# Wstęp teoretyczny

Celem sprawozdania jest analiza modelu fizycznego opisującego ruch ciała w polu centralnym – klasycznego zagadnienia mechaniki nieba. Jest to problem dwóch ciał, w którym jedno z nich (np. planeta) porusza się pod wpływem siły grawitacji drugiego (np. Słońca), które uznaje się za nieruchome.

Podstawowe wzory fizyczne używane do przekształcenia:

Prawo powszechnego ciążenia oraz druga zasada dynamiki Newtona

– wektor siły ciążenia

G – stała grawitacyjna

M ,m – masy ciał

r – odległość miedzy środami mas obu ciał

– wektor jednostkowy kierunek przyciągania ciała

analogicznie

Równanie drugiego rzędu na ruch ciał

x – współrzędna ciała w osi X,

y – współrzędna ciała w osi Y,

– prędkość ciała w kierunku X,

– prędkość ciała w kierunku Y,

t – czas,

lub

Równanie odległości ciała od środka siły

r(θ) – odległość od środka przy kącie θ (dokładna trajektoria),

A – parametr związany z mimośrodem orbity

B – związany z momentem pędu

h – moment pędu na jednostkę masy

e – mimośród

Równanie średniego błędu bezwzględnego oraz średniego błędu kwadratowego

przewidywane wartości

wartości dokładne

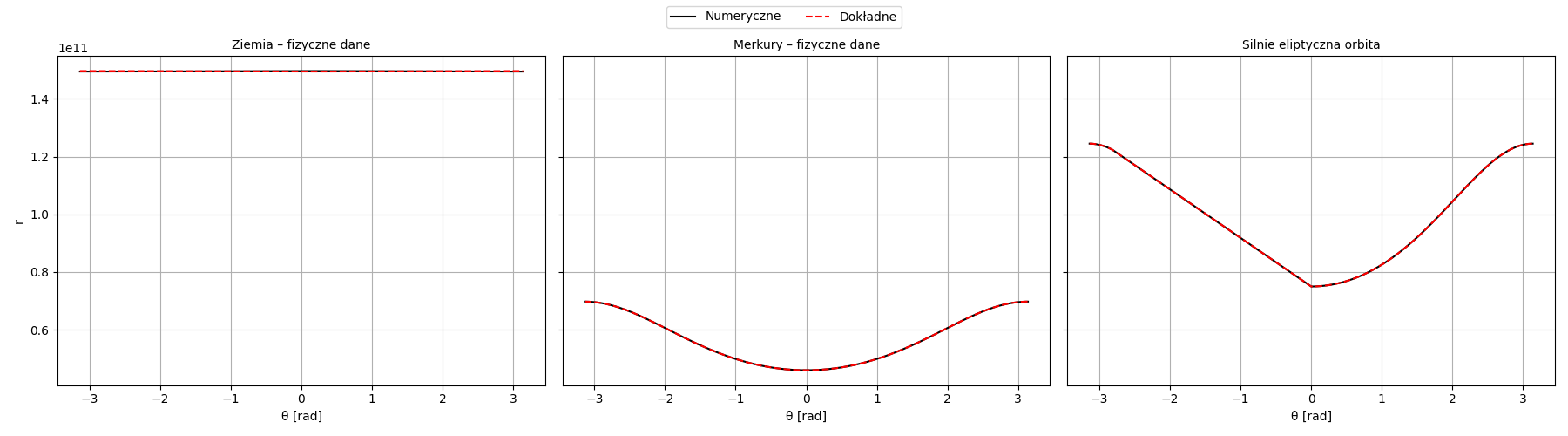
# Opis rozwiązania

1. #lista4  
     
   import numpy as np  
   import matplotlib.pyplot as plt  
   from scipy.integrate import solve\_ivp  
   import pandas as pd  
   import sympy as sp  
   import math  
     
     
   # === Równania ruchu ===  
   def orbital\_rhs(t, y, k):  
    x, vx, y\_, vy = y  
    r = np.sqrt(x\*\*2 + y\_\*\*2)  
    ax = -k \* x / r\*\*3  
    ay = -k \* y\_ / r\*\*3  
    return [vx, ax, vy, ay]  
     
   def compute\_h(x0, vx0, y0, vy0):  
    return x0 \* vy0 - y0 \* vx0  
     
   def compute\_e(x0, vx0, y0, vy0, G, M):  
    r0 = np.sqrt(x0\*\*2 + y0\*\*2)  
    v0 = np.sqrt(vx0\*\*2 + vy0\*\*2)  
    h = compute\_h(x0, vx0, y0, vy0)  
    mu = G \* M  
    e = np.sqrt(1 + (2 \* (0.5 \* v0\*\*2 - mu / r0) \* h\*\*2) / mu\*\*2)  
    return e  
     
   # === Symboliczne rozwiązanie r(θ) z sympy ===  
   def symbolic\_r\_theta(h\_val, G\_val, M\_val, e\_val):  
    theta = sp.symbols('theta')  
    mu = G\_val \* M\_val  
    p = h\_val\*\*2 / mu  
    r\_exact\_expr = p / (1 + e\_val \* sp.cos(theta))  
    return r\_exact\_expr  
     
   # === Główna funkcja do rysowania r(θ) ===  
   def plot\_theta\_trajectories\_by\_group(params\_phys, num\_points=500):  
    results\_phys = []  
     
    # Podział: 2 fizyczne + 4 zmodyfikowane  
    groups = [params\_phys[:2], params\_phys[2:]]  
    titles = [" "," "]  
     
    for group\_idx, group in enumerate(groups):  
    n = len(group)  
    n\_rows = int(math.floor(math.sqrt(n)))  
    n\_cols = int(math.ceil(n / n\_rows))  
     
    fig\_r, axs\_r = plt.subplots(n\_rows, n\_cols, figsize=(6 \* n\_cols, 4.5 \* n\_rows))  
    fig\_err, axs\_err = plt.subplots(n\_rows, n\_cols, figsize=(6 \* n\_cols, 4 \* n\_rows))  
     
    axs\_r = np.array(axs\_r).reshape(-1)  
    axs\_err = np.array(axs\_err).reshape(-1)  
     
    for idx, ((G, M, y0, t\_span, label), ax\_r, ax\_err) in enumerate(zip(group, axs\_r, axs\_err)):  
    k = G \* M  
    h = compute\_h(\*y0)  
    e = compute\_e(\*y0, G, M)  
    p = h\*\*2 / k  
     
    t\_eval = np.linspace(\*t\_span, num\_points)  
    sol = solve\_ivp(orbital\_rhs, t\_span, y0, args=(k,), t\_eval=t\_eval, rtol=1e-9)  
     
    x\_num = sol.y[0]  
    y\_num = sol.y[2]  
    r\_numeric = np.sqrt(x\_num\*\*2 + y\_num\*\*2)  
    theta\_numeric = np.arctan2(y\_num, x\_num)  
     
    sort\_idx = np.argsort(theta\_numeric)  
    theta\_sorted = theta\_numeric[sort\_idx]  
    r\_sorted\_numeric = r\_numeric[sort\_idx]  
     
    r\_exact\_expr = symbolic\_r\_theta(h, G, M, e)  
    r\_exact\_func = sp.lambdify(sp.symbols('theta'), r\_exact\_expr, modules=['numpy'])  
    r\_sorted\_exact = r\_exact\_func(theta\_sorted)  
     
    abs\_error = np.abs(r\_sorted\_numeric - r\_sorted\_exact)  
    sq\_error = (r\_sorted\_numeric - r\_sorted\_exact)\*\*2  
    mae = np.mean(abs\_error)  
    mse = np.mean(sq\_error)  
    results\_phys.append((label, mae, mse))  
     
    # === r(θ): Numeryczne (czarny), Dokładne (czerwony przerywany)  
    ax\_r.plot(theta\_sorted, r\_sorted\_numeric, label="Numeryczne", color='black')  
    ax\_r.plot(theta\_sorted, r\_sorted\_exact, 'r--', label="Dokładne (SymPy)")  
    ax\_r.set\_title(label)  
    ax\_r.set\_xlabel("θ [rad]")  
    ax\_r.set\_ylabel("r [m]")  
    ax\_r.grid(True)  
     
    # === Błąd: kolor niebieski  
    #ax\_err.plot(theta\_sorted, abs\_error, label="Błąd bezwzględny", color='blue')  
    default\_color = plt.rcParams['axes.prop\_cycle'].by\_key()['color'][0]  
    ax\_err.plot(theta\_sorted, abs\_error, label="Błąd bezwzględny", color=default\_color)  
    ax\_err.set\_title(f"Błąd r(θ) – {label}")  
    ax\_err.set\_xlabel("θ [rad]")  
    ax\_err.set\_ylabel("Błąd [m]")  
    ax\_err.grid(True)  
     
    fig\_r.suptitle("Porównanie r(θ): Numeryczne vs Dokładne", fontsize=14)  
    fig\_err.suptitle(titles[group\_idx], fontsize=14)  
     
    plt.tight\_layout(rect=[0, 0, 1, 0.94])  
    plt.show()  
     
    return results\_phys  
     
     
   # === 3D wykresy trajektorii po 3 przypadki ===  
   def plot\_3d\_trajectories\_by\_group(params\_phys, num\_points=500):  
    for i in range(0, len(params\_phys), 3):  
    group = params\_phys[i:i+3]  
    fig = plt.figure(figsize=(18, 6))  
     
    for j, (G, M, y0, t\_span, label) in enumerate(group):  
    k = G \* M  
    t\_eval = np.linspace(\*t\_span, num\_points)  
    sol = solve\_ivp(orbital\_rhs, t\_span, y0, args=(k,), t\_eval=t\_eval, rtol=1e-9)  
    x\_num = sol.y[0]  
    y\_num = sol.y[2]  
    t\_num = sol.t  
     
    ax = fig.add\_subplot(1, 3, j + 1, projection='3d')  
    ax.plot(x\_num, y\_num, t\_num, label=label)  
    ax.set\_xlabel("x")  
    ax.set\_ylabel("y")  
    ax.set\_zlabel("t")  
    ax.set\_title(label)  
    ax.legend()  
    ax.grid(True)  
     
    fig.suptitle("Trajektorie 3D: x(t), y(t), t", fontsize=14)  
    plt.tight\_layout(rect=[0, 0, 1, 0.95])  
    plt.show()  
     
   # === Zestaw przypadków fizycznych ===  
   params\_phys = [  
    (6.67430e-11, 1.989e30, [1.496e11, 0, 0, 29780], (0, 3.154e7), "Ziemia – fizyczne dane"),  
    (6.67430e-11, 1.989e30, [4.6e10, 0, 0, 58980], (0, 7.6e6), "Merkury – fizyczne dane"),  
    (6.67430e-11, 1.989e30, [7.5e10, 0, 0, 47000], (0, 1.0e7), "Silnie eliptyczna orbita"),  
    (6.67430e-11, 1.989e30, [1.496e11, 0, 0, 15000], (0, 3.154e7), "Zbyt wolna prędkość – spiralny spadek"),  
    (6.67430e-11, 1.989e30, [1.496e11, 0, 0, 60000], (0, 1.0e7), "Zbyt duża prędkość – ucieczka"),  
    (6.67430e-11, 5 \* 1.989e30, [1.496e11, 0, 0, 29780], (0, 1.5e7), "Masa Słońca ×5 – silniejsze pole"),  
   ]  
     
   # === Uruchomienie wszystkiego ===  
   results\_phys = plot\_theta\_trajectories\_by\_group(params\_phys)  
   df\_phys = pd.DataFrame(results\_phys, columns=["Opis", "MAE", "MSE"])  
   print("\nPorównanie wyników dla różnych zestawów fizycznych parametrów:")  
   print(df\_phys)  
     
   plot\_3d\_trajectories\_by\_group(params\_phys)

# Wyniki obliczeń

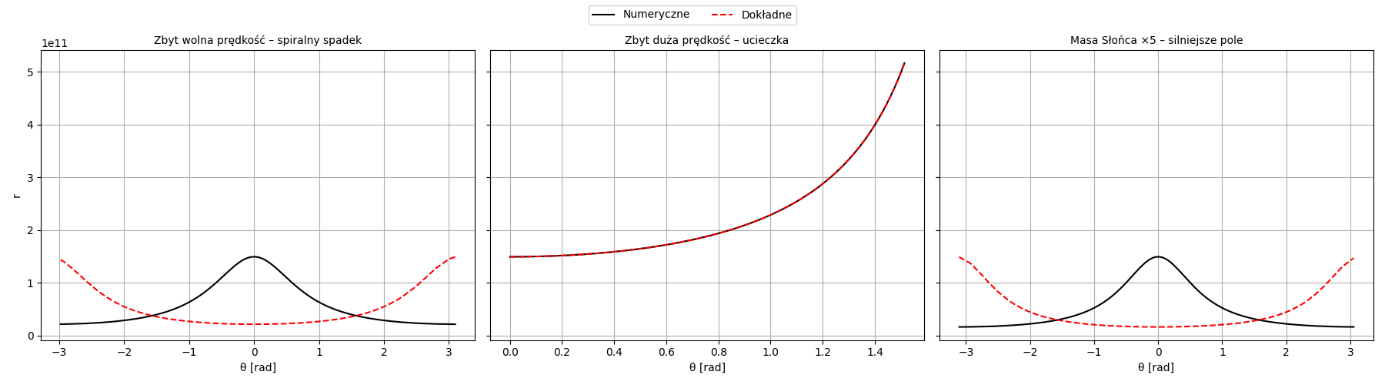
Porównanie trajektorii pokazało dobrą zgodność numeryczną z rozwiązaniem analitycznym.

1.1 Układ Ziemia – fizyczne dane



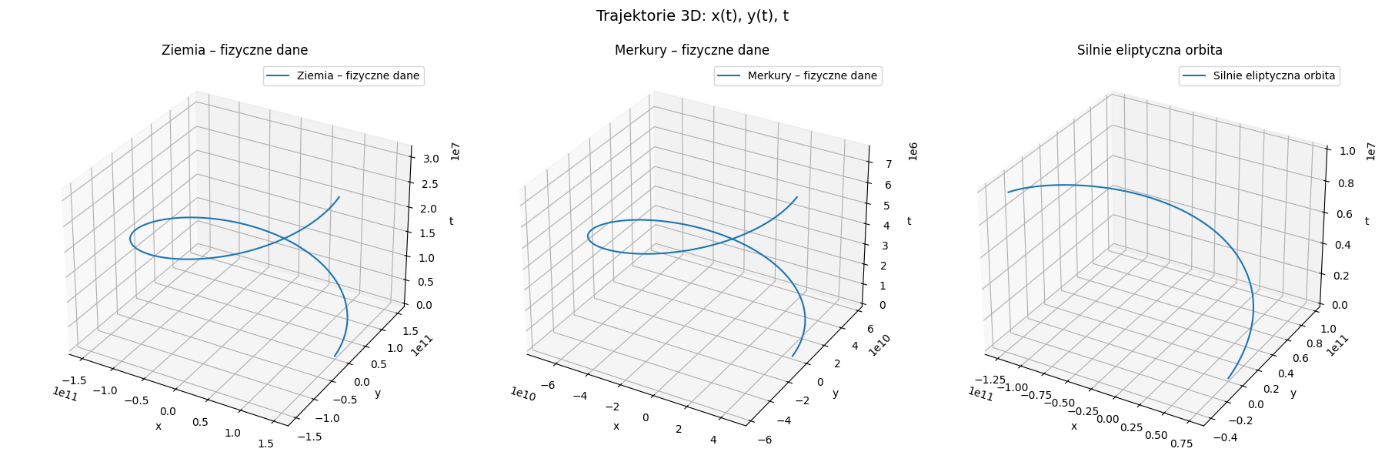
Rysunek 1. Trajektoria r(θ) – numeryczna vs. analityczna dla Ziemi

1.2 Układ Merkury – fizyczne dane



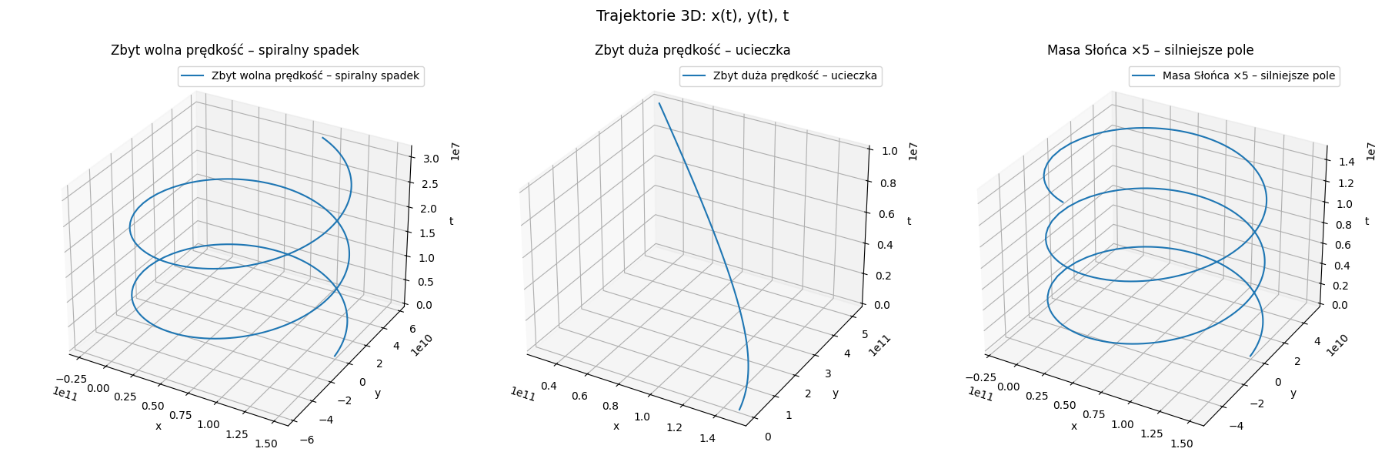
Rysunek 2. Trajektoria r(θ) – Merkury

1.3 Silnie eliptyczna orbita



Rysunek 3. Trajektoria dla wysokiego mimośrodu

1.4 Zbyt wolna / zbyt szybka prędkość początkowa



Rysunek 4. Wykresy trajektorii dla prędkości suborbitalnej i hiperbolicznej

Dla orbit Ziemi i Merkurego zgodność modelu numerycznego z analitycznym jest bardzo wysoka, co potwierdzają zarówno wykresy, jak i niskie wartości MAE oraz MSE. Przy silnie eliptycznej trajektorii błędy rosną zauważalnie, zwłaszcza w obszarach o gwałtownej zmianie nachylenia toru względem kąta.

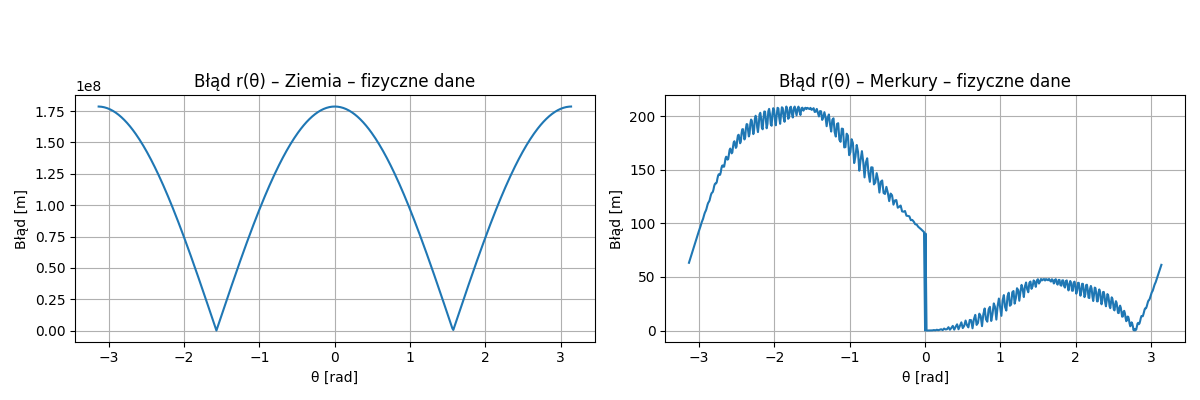
Nietypowe scenariusze – takie jak zbyt mała prędkość początkowa (prowadząca do spiralnego spadku) lub zbyt duża (powodująca ucieczkę z układu) – również generują większe odchylenia, ujawniając ograniczenia klasycznego modelu w ekstremalnych warunkach.

Wizualizacje 3D umożliwiają dodatkową interpretację ruchu: zamknięte orbity leżą w jednej płaszczyźnie, natomiast tory ucieczkowe ukazują niemonotoniczny wzrost odległości w czasie.

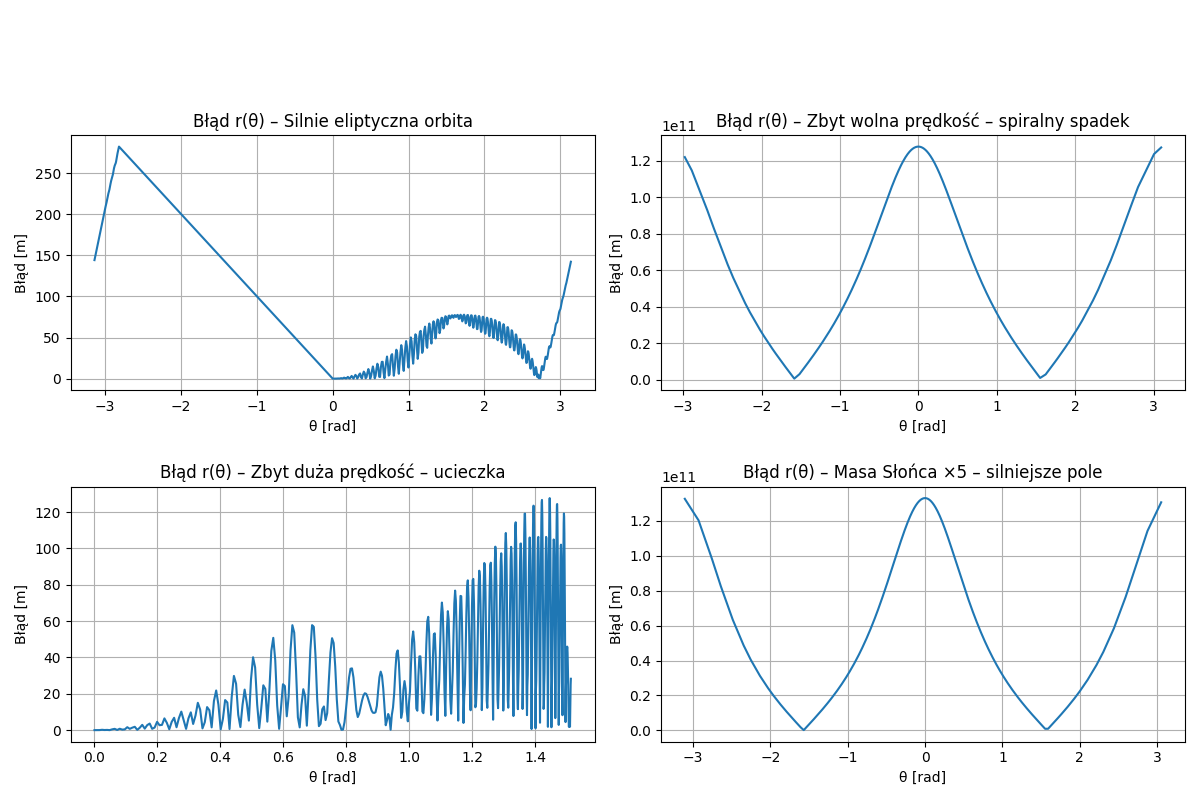
Tabela 1. Obliczone średnie bezwzględne i kwadratowe błędy

|  |  |  |
| --- | --- | --- |
|  | MAE | MSE |
| Ziemia – Słonce (rzeczywiste) | 1.138815e+08 | 1.599449e+16 |
| Merkury – Słonce (rzeczywiste) | 9.371100e+01 | 1.427395e+04 |
| Merkury – Słonce (nie rzeczywiste) | 7.042044e+01 | 9.461601e+03 |
| Silnie eliptyczna orbita | 9.252485e+10 | 9.827663e+21 |
| Zbyt wolna prędkość | 4.274094e+01 | 3.089281e+03 |
| Masa Słońca ×5 | 9.233533e+10 | 1.003411e+22 |

1.5 Wykresy błędów bezwzględnych r(θ)



Rysunek 5. Wykresy błędów dla danych fizycznych



Rysunek 6. Wykresy błędów dla danych zmodyfikowanych

Analiza błędów bezwzględnych r(θ) wykazała, że najmniejszy błąd występuje dla orbity Merkurego – zarówno w wersji realistycznej, jak i uproszczonej. Świadczy to o dużej dokładności metody numerycznej w przypadku orbit o małym mimośrodzie.

Dla Ziemi błąd jest nieco większy, ale wciąż stabilny i akceptowalny.

Największe błędy pojawiają się w warunkach ekstremalnych: przy silnie eliptycznych orbitach, skrajnych prędkościach początkowych oraz zwiększonej masie centralnej. Największe odchylenia występują w pobliżu peryhelium lub w miarę oddalania się obiektu.

Wykresy błędu r(θ) pozwalają precyzyjnie zlokalizować miejsca na orbicie, w których rozwiązanie numeryczne odbiega od analitycznego. Natomiast przedstawione w tabeli wartości MAE i MSE są średnimi statystycznymi tych błędów — podsumowują globalnie rozkład widoczny na wykresach.

Podsumowując, metoda numeryczna jest wiarygodna w typowych warunkach, lecz wymaga ostrożności przy dynamicznych i niestandardowych orbitach, gdzie lokalne błędy mogą istotnie wzrosnąć.

# Wnioski i podsumowanie

Przeprowadzona analiza potwierdziła poprawność modelu ruchu w polu centralnym. Porównanie rozwiązań numerycznych i analitycznych wykazało wysoką zgodność dla realistycznych przypadków (Ziemia, Merkury), a wzrost błędów był zauważalny przy ekstremalnych warunkach (np. eliptyczne orbity, zmieniona masa).

Wpływ parametrów początkowych na kształt orbity jest istotny – zmiana prędkości może prowadzić do spadku lub ucieczki z układu. Mimośród wyraźnie wpływa na wydłużenie trajektorii, co zwiększa błędy.

Model spełnia wymagania zadania i może służyć jako baza do dalszych symulacji – np. problemu wielu ciał lub analizy relatywistycznej.