単振り子の運動シミュレーション：
オイラー法とルンゲ・クッタ法(RK4)の比較

学籍番号：1423107
氏名：山北倫太郎
日付：July 22, 2025

目次

1　目的

常微分方程式（Ordinary Differential Equation, ODE）は、物理学、工学、生物学など、自然科学の多岐にわたる分野で時間発展するシステムを記述するための基本的な数学的ツールである。これらの多く、特に非線形性を含むものは解析的に解を求めることが困難であり、その挙動を理解するためには数値計算によるアプローチが不可欠となる。

本レポートでは、常微分方程式の数値解法の中でも最も基本的で代表的な手法であるオイラー法と、高次精度解法として広く実用に供されている 4 次のルンゲ・クッタ法（以下、RK4）を対象とする。これらの手法を Python 言語を用いて実装し、非線形振動系の典型例である単振り子の運動シミュレーションを行う。

本研究の主たる目的は、シミュレーション結果に基づき、両手法の解の精度、安定性、および物理量の保存性（特に力学的エネルギー）を定量的かつ定性的に比較・評価することにある。異なる時間刻み幅におけるシミュレーションを通じて、各手法の数値的特性、収束性、そして物理法則の再現能力を検証し、それぞれの長所と短所、さらには適用限界を明らかにすることを目指す。これにより、特定の物理問題に対して適切な数値解法を選択する際の基準に関する深い洞察を得る。

2 理論

本セクションでは、シミュレーションの対象である単振り子の物理モデルと、その運動を記述する数学的定式化、そして本研究で用いる数値解法のアルゴリズムと理論的背景について詳述する。

2.1 単振り子の運動方程式

単振り子は、力学系における最も基本的なモデルの一つである。本研究で扱う単振り子モデルは、質量が無視できる長さ L の伸縮しない剛体棒の先端に、質量 m の質点が取り付けられている系と定義する。この質点は、棒のもう一方の端点である固定点を中心として、鉛直面内を円弧に沿って運動する。

質点の運動を記述するため、鉛直下向きを基準（角度 0）とし、そこからの振り子の振れ角を θ とする。質点に作用する力は、鉛直下向きに働く重力 mg と、棒の張力 S のみである。空気抵抗や摩擦などの散逸力は無視する。接線方向の運動方程式を考えることにより、単振り子の運動を支配する以下の 2 階非線形常微分方程式が得られる。

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0$$

この方程式に含まれる $\sin\theta$ の項が非線形性の源である。本レポートではこの非線形方程式を直接扱う。

2.2 1 階連立常微分方程式への変換

本研究で用いるオイラー法や RK4 を含む多くの標準的な数値解法は、1 階の常微分方程式（またはその連立系）を解くために設計されている。したがって、2 階の運動方程式を 1 階の連立方程式系に変換する必要がある。状態変数として角度 θ と角速度 $\omega = \frac{d\theta}{dt}$ を導入し、状態ベクトル $\mathbf{y}$ を以下のように定義する。

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \theta \\ \omega \end{pmatrix}$$

これにより、単振り子の運動方程式は以下の連立1階常微分方程式系に等価的に変換される。

$$\begin{cases} \frac{dy_1}{dt} = y_2 \\ \frac{dy_2}{dt} = -\frac{g}{L}\sin(y_1) \end{cases}$$

この形式は、$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y})$ の形をしており、数値解法を直接適用できる。

2.3 数値解法アルゴリズム

2.3.1 オイラー法

オイラー法は、ある点での傾きを用いて微小時間後の値を線形に外挿する、最も単純な数値解法である。時刻 t_n での解の近似値を $\mathbf{y}_n$ とすると、次の時刻 $t_{n+1} = t_n + h$ での解 $\mathbf{y}_{n+1}$ は以下の漸化式で計算される。

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \cdot \mathbf{f}(t_n, \mathbf{y}_n)$$

オイラー法は1次精度の手法であり、局所離散化誤差は $O(h^2)$、大域離散化誤差は $O(h)$ である。

2.3.2 4 次ルンゲ・クッタ法（RK4）

RK4 は、1ステップの区間内で複数回（4回）傾きを計算し、それらを加重平均することで高精度を実現する手法である。具体的なアルゴリズムは以下の通りである。

$$\begin{aligned} \mathbf{k}_1 &= h \cdot \mathbf{f}(t_n, \mathbf{y}_n) \\ \mathbf{k}_2 &= h \cdot \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{\mathbf{k}_1}{2}\right) \\ \mathbf{k}_3 &= h \cdot \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{\mathbf{k}_2}{2}\right) \\ \mathbf{k}_4 &= h \cdot \mathbf{f}(t_n + h, \mathbf{y}_n + \mathbf{k}_3) \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \end{aligned}$$

RK4 は 4 次精度の手法であり、局所離散化誤差は $O(h^5)$、大域離散化誤差は $O(h^4)$ となる。

2.4 物理系のエネルギー保存則

空気抵抗や摩擦を無視した理想的な単振り子では、全力学的エネルギー E は保存される。E は運動エネルギー K と位置エネルギー U の和で定義される。

$$E(\theta, \omega) = K + U = \frac{1}{2}mL^2\omega^2 + mgL(1 - \cos\theta)$$

理想系では $\frac{dE}{dt} = 0$ が厳密に成り立つ。数値シミュレーションにおいて、このエネルギー保存則がどの程度満たされるかは、解法の物理的な妥当性を評価する重要な指標となる。

3　問題設定

本レポートで実施する数値実験の条件を以下に定義する。

- **物理定数**:
  - 重力加速度 g: $9.8\,\mathrm{m/s^2}$
  - 振り子の長さ L: $1.0\,\mathrm{m}$
  - 質点の質量 m: $1.0\,\mathrm{kg}$
- **初期条件**:
  - 初期角度 θ_0: $30°$ ($\approx 0.5236\,\mathrm{rad}$)
  - 初期角速度 ω_0: $0\,\mathrm{rad/s}$
- **シミュレーション時間**:
  - 開始時刻 t_{start}: $0\,\mathrm{s}$
  - 終了時刻 t_{end}: $10\,\mathrm{s}$
- **時間刻み幅** h:
  - $h = 0.1\,\mathrm{s}$ （粗い刻み幅）
  - $h = 0.01\,\mathrm{s}$ （中間の刻み幅）
  - $h = 0.001\,\mathrm{s}$ （細かい刻み幅）

4　実験結果

前節で定義した問題設定に基づき、シミュレーションを実行した。系の初期エネルギーは、$E_0 = mgL(1-\cos\theta_0) \approx 1.312951\,\mathrm{J}$ である。

Table 1: 各手法と刻み幅における最終時点（t=10s）の角度とエネルギー

手法 (Method)	刻み幅 h (s)	最終角度 $\theta(10)$ (rad)	最終エネルギー $E(10)$ (J)
オイラー法	0.1	37.614072	46.166622
オイラー法	0.01	0.449571	3.270946
オイラー法	0.001	0.430550	1.441952
RK4	0.1	0.417948	1.311415
RK4	0.01	0.418772	1.312951
RK4	0.001	0.418772	1.312951

図1: 異なる刻み幅における角度の時間変化

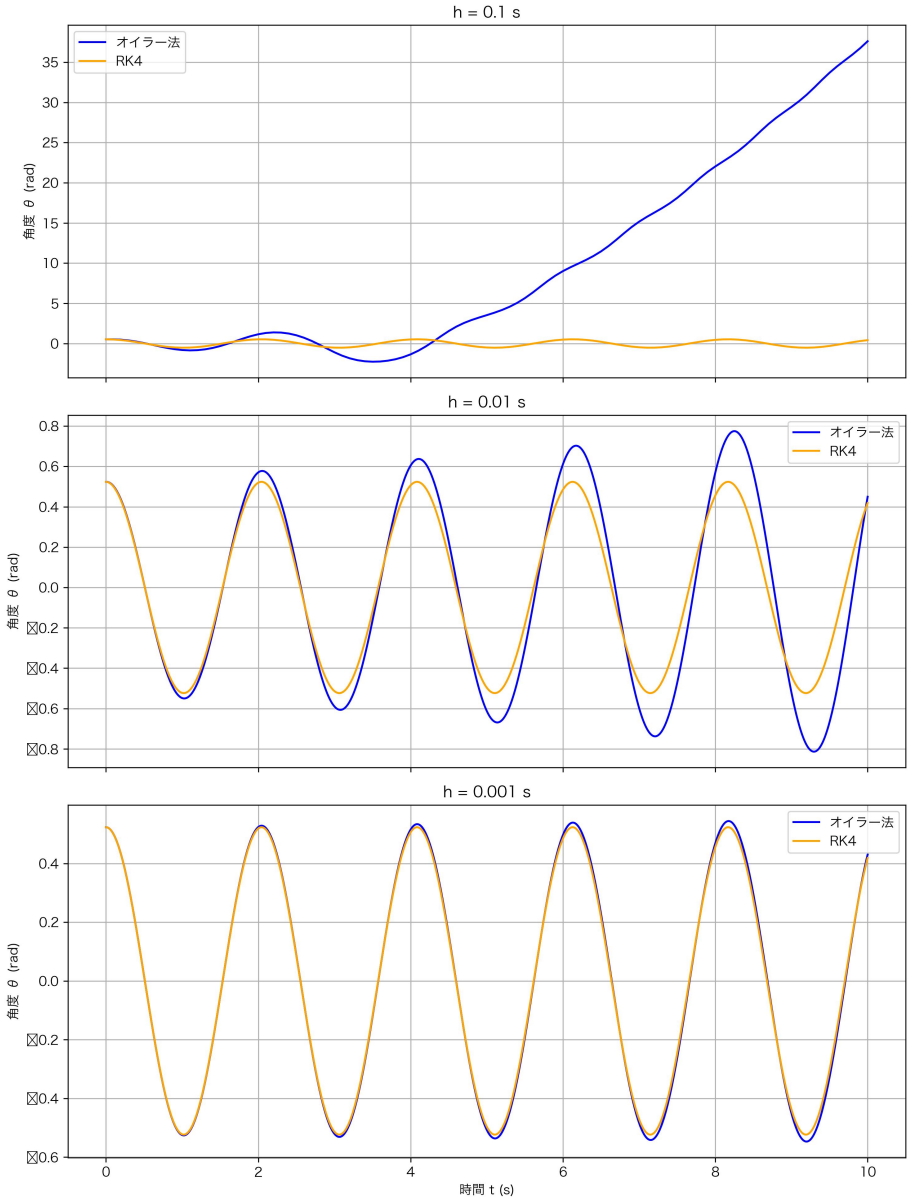

Figure 1: 3 つの異なる刻み幅（上から $h = 0.1, h = 0.01, h = 0.001$）における、オイラー法と RK4 による角度 θ の時間変化。横軸は時間 t (s)、縦軸は角度 θ (rad) を示す。

図2: 異なる刻み幅における全エネルギーの時間変化

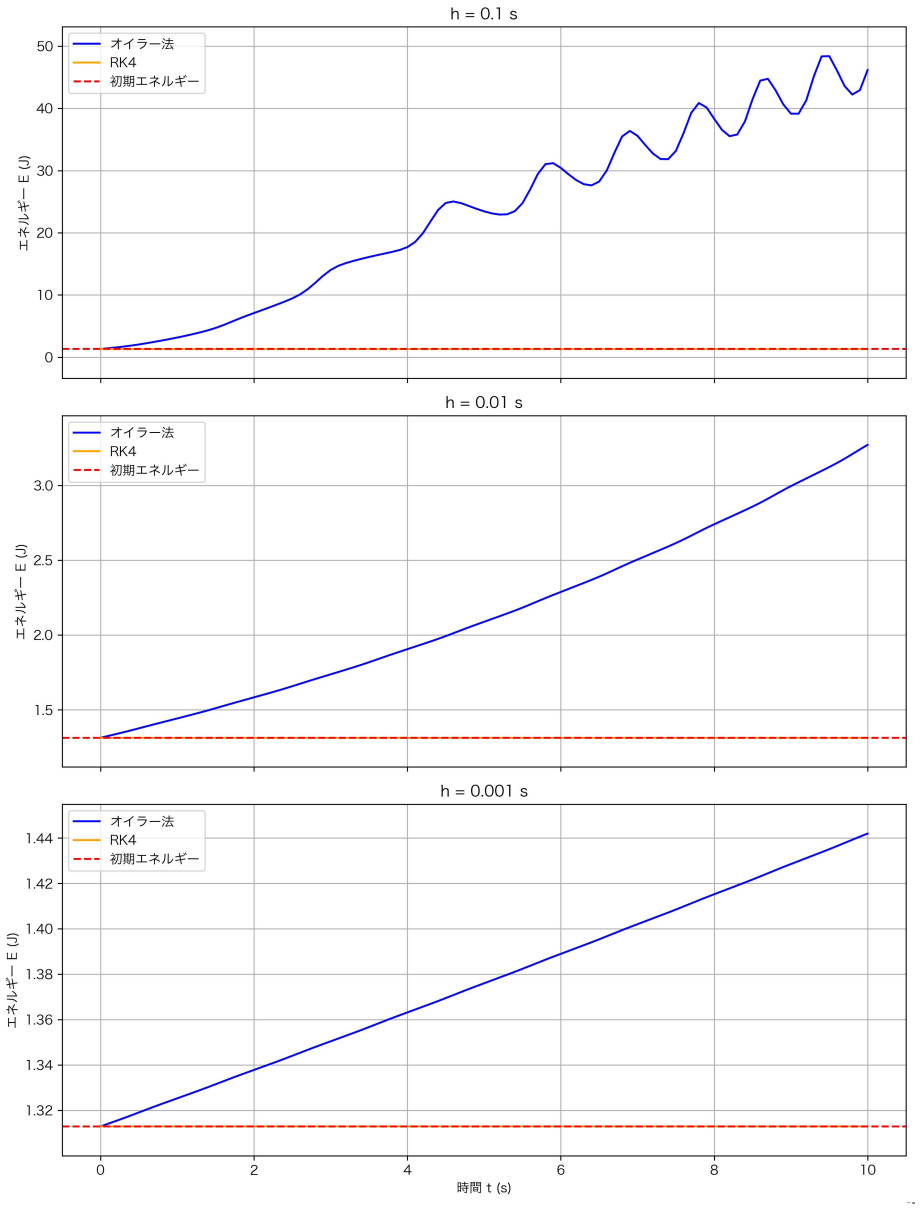

Figure 2: 3 つの異なる刻み幅（上から $h = 0.1, h = 0.01, h = 0.001$）における、オイラー法と RK4 で計算された全力学的エネルギー E の時間変化。横軸は時間 t (s)、縦軸はエネルギー E (J) を示す。破線は初期エネルギーの値を示す。

5　考察

5.1　解の精度と安定性の比較

図 1は、両手法の解の精度と安定性の違いを明確に示している。$h = 0.1$ の場合、オイラー法による解は振幅が際限なく増大し、物理的にありえない挙動を示す。これは、オイラー法が容易に破綻することを示している。一方、RK4 は安定した周期的振動を維持している。刻み幅を $h = 0.01$ に狭めると、オイラー法の解は改善されるが、RK4 の解と比較すると位相が徐々にずれていく。$h = 0.001$ にすると、オイラー法の解は RK4 の解に視覚的に近くなるが、表 1の最終角度を見ると依然として誤差の蓄積が見られる。対照的に、RK4 は $h = 0.1$ ですら安定した解を生成し、$h = 0.01$ と $h = 0.001$ の結果はほぼ完全に一致しており、非常に速く真の解に収束することを示している。

5.2　物理法則の再現性：エネルギー保存

図 2は、シミュレーションの物理的な妥当性を評価するための決定的な証拠を提供する。オイラー法では、すべての刻み幅において、計算された全力学的エネルギーが時間とともに単調に増加している。$h = 0.1$ の場合、10 秒後には初期エネルギーの約 35 倍にまで増加しており、シミュレーションが物理的現実から完全に乖離していることを示している。これは、オイラー法が系の軌道を常に外側へ更新してしまうアルゴリズムの構造的欠陥に起因する。対照的に、RK4 ではエネルギーは初期値の周りで非常に小さく振動するのみで、系統的な増加や減少は見られない。特に $h = 0.01$ と $h = 0.001$ では、エネルギーは極めて高い精度で保存されており、RK4 が系の物理法則を忠実に再現する能力を持つことがわかる。

5.3　数値誤差の理論的考察

実験で観測された性能差は、両手法の理論的な精度の次数の違いによって説明できる。オイラー法の大域離散化誤差は $O(h)$、RK4 のそれは $O(h^4)$ である。これは、刻み幅 h を 1/10 にすると、オイラー法の誤差は約 1/10 に減少するのに対し、RK4 の誤差は約 1/10000 に減少することを意味する。この圧倒的な収束率の違いが、RK4 がはるかに少ない計算ステップで高精度な解を得られる理由である。この精度の違いは、ステップ内で複数回傾きを評価し加重平均する RK4 の洗練されたアルゴリズム構造に起因する。

5.4　計算効率と精度のトレードオフ

1 ステップあたりの計算量は RK4 がオイラー法の 4 倍であるが、「目標とする精度を達成するための総コスト」で比較すると、多くの場合 RK4 が優れる。本実験が示すように、実用的な精度を求める場合、RK4 はより大きな刻み幅を使えるため、総計算量においてオイラー法よりも効率的になる。低精度でよい一部のケースを除き、RK4 の方が計算効率と精度のバランスに優れた手法であると言える。

6　結論

本研究では、オイラー法と４次ルンゲ・クッタ法（RK4）を実装し、単振り子の運動シミュレーションを通じて両者の性能を比較・評価した。

1. **オイラー法**は、実装が容易だが精度が低く、特に力学的エネルギーを保存できないという致命的な欠陥を持つ。そのため、保存則が重要な物理系の長期シミュレーションには不適切である。

2. **４次ルンゲ・クッタ法（RK4）**は、アルゴリズムは複雑だが４次という高い精度を持ち、粗い刻み幅でも安定かつ高精度な解を生成する。特に、力学的エネルギーを極めて高い精度で保存する能力があり、物理法則に忠実なシミュレーションが可能である。

3. 計算効率の観点からは、目標とする精度を達成するために必要な総計算量では、多くの場合 RK4 がオイラー法を上回る。

総じて、非線形な物理系を正確にシミュレーションするためには、問題の特性に適した高次精度で安定した数値解法を選択することが極めて重要である。オイラー法は教育的な導入としては有用だが、実際の科学技術計算においては、RK4 のようなより洗練された手法を用いることが、信頼性の高い結果を得るための必須要件であると言えよう。

8. 感想

本演習に取り組むまで、常微分方程式の数値解法は、単に真の解を近似計算するための数学的な道具という程度の認識でした。オイラー法もルンゲ・クッタ法も、その精度が異なるだけで、本質的な挙動は同じようなものになるだろうと漠然と考えていました。

しかし、実際に単振り子のシミュレーションを実装し、その結果を可視化してみて、その考えは完全に覆されました。特に衝撃的だったのは、オイラー法によるシミュレーションで力学的エネルギーが保存されず、時間とともに増大していく結果（図 2）を目の当たりにしたことです。物理法則を明らかに満たさない解が、アルゴリズムから導出されるという事実は、数値計算の「正しさ」とは何かを深く考えさせられるきっかけとなりました。単に数式を離散化するだけでなく、その手法が元の系の物理的性質をどれだけ忠実に再現できるかが、解法の選択において極めて重要であるということを実感しました。

また、理論で学んだ誤差のオーダー（オイラー法が $O(h)$、RK4 が $O(h^4)$）が、これほどまでに結果の安定性や精度に劇的な違いを生むことにも驚きました。RK4 が、より粗い刻み幅でも安定した振動とエネルギー保存を達成できる効率の良さは、計算コストと精度のトレードオフを考える上で非常に示唆に富むものでした。

今回のレポート作成を通して、数値シミュレーションとは、単なる計算作業ではなく、物理現象への深い洞察と、それを実現するための数学的・アルゴリズム的な知識が不可欠な、奥深い分野であることを学びました。今後は、

本レポートで扱わなかった他の数値解法（シンプレクティック法など）が、なぜエネルギー保存のような物理量の保存に優れているのか、その理論的背景についても探求していきたいと考えています。

References

[1] 三井斌友(2003). 『常微分方程式の数値解法』. 岩波書店.

[2] 山本哲朗(2003). 『数値解析入門[増訂版]』. サイエンス社.

A　プログラムコード

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

# 日本語フォントの設定
plt.rcParams['font.family'] = 'sans-serif'
plt.rcParams['font.sans-serif'] = ['Hiragino Sans GB', 'Arial', '
    Helvetica', 'Yu Gothic', 'Meiryo', 'MS Gothic']


# 数式フォントの設定
mpl.rcParams['mathtext.default'] = 'regular'

# --- グローバル設定 ---
# 物理定数
G = 9.8  # 重力加速度 (m/s^2)
L = 1.0  # 振り子の長さ (m)
M = 1.0  # 質点の質量 (kg)


# 初期条件
THETA_0 = np.pi / 6  # 初期角度 30度 (rad)
OMEGA_0 = 0.0        # 初期角速度 (rad/s)

# シミュレーション時間
T_START = 0.0
T_END = 10.0

# --- 連立1階常微分方程式の定義 ---
# y = [theta, omega]
# dy/dt = f(t, y)
def f(t, y):
    """
    単振り子の運動方程式を記述するベクトル関数。
    Args:
        t (float): 時刻 (未使用だが、一般的なソルバーのインターフェース
            に合わせる)
        y (np.ndarray): 状態ベクトル [theta, omega]
    Returns:
```

```
        np.ndarray: 状態ベクトルの時間微分 [d(theta)/dt, d(omega)/dt]
    """
    theta, omega = y
    return np.array([omega, -(G / L) * np.sin(theta)])

# --- エネルギー計算関数 ---
def calculate_energy(y):
    """
    与えられた状態ベクトルから全力学的エネルギーを計算する。
    Args:
        y (np.ndarray): 状態ベクトル [theta, omega]
    Returns:
        float: 全力学的エネルギー (J)
    """
    theta, omega = y
    kinetic_energy = 0.5 * M * (L * omega)**2
    potential_energy = M * G * L * (1 - np.cos(theta))
    return kinetic_energy + potential_energy

# --- 数値解法ソルバー ---
def euler_step(f_func, t, y, h):
    """オイラー法による1ステップ計算"""
    return y + h * f_func(t, y)

def rk4_step(f_func, t, y, h):
    """4次ルンゲ・クッタ法による1ステップ計算"""
    k1 = h * f_func(t, y)
    k2 = h * f_func(t + h / 2, y + k1 / 2)
    k3 = h * f_func(t + h / 2, y + k2 / 2)
    k4 = h * f_func(t + h, y + k3)
    return y + (k1 + 2 * k2 + 2 * k3 + k4) / 6

# --- シミュレーション実行関数 ---
def run_simulation(h, solver_func):
    """
    指定された刻み幅とソルバーでシミュレーションを実行する。
    Args:
        h (float): 時間刻み幅
        solver_func (function): 使用するソルバー関数 (euler_step or
            rk4_step)
    Returns:
        tuple: (時間配列, 角度配列, エネルギー配列)
    """
    time_points = np.arange(T_START, T_END + h, h)
    num_steps = len(time_points)

    y_history = np.zeros((num_steps, 2))
    y_history[0] = np.array([THETA_0, OMEGA_0])  # 初期条件を設定

    energy_history = np.zeros(num_steps)
    energy_history[0] = calculate_energy(y_history[0])

    for i in range(num_steps - 1):
        y_history[i+1] = solver_func(f, time_points[i], y_history[i], h
            )
        energy_history[i+1] = calculate_energy(y_history[i+1])
```

```
92
93      theta_history = y_history[:, 0]
94      return time_points, theta_history, energy_history
95
96  # --- メイン処理 ---
97  if __name__ == '__main__':
98      H_LIST = [0.1, 0.01, 0.001]
99
100     # --- 表1のデータ生成 ---
101     initial_energy = calculate_energy(np.array([THETA_0, OMEGA_0]))
102     print("表1: 各手法と刻み幅における最終時点（t=10s）の角度とエネルギー")
103     print(f"(初期エネルギー E_0 = {initial_energy:.6f} J)")
104     print("| 手法 (Method) | 刻み幅 h (s) | 最終角度 θ(10) (rad) | 最終エネルギー E(10) (J) |")
105     print("|:---|:---|:---|:---|")
106
107     solvers = {'オイラー法': euler_step, 'RK4': rk4_step}
108     results_for_table = {}
109
110     for name, solver in solvers.items():
111         for h in H_LIST:
112             t, theta, energy = run_simulation(h, solver)
113             print(f"| {name} | {h:<12} | {theta[-1]:.6f} | {energy[-1]:.6f} |")
114             results_for_table[(name, h)] = (t, theta, energy)
115
116     # --- 図1と図2のプロット生成 ---
117     fig_angle, axes_angle = plt.subplots(3, 1, figsize=(10, 15), sharex=True)
118     fig_energy, axes_energy = plt.subplots(3, 1, figsize=(10, 15), sharex=True)
119
120     fig_angle.suptitle('図1: 異なる刻み幅における角度の時間変化', fontsize=16, y=0.92)
121     fig_energy.suptitle('図2: 異なる刻み幅における全エネルギーの時間変化', fontsize=16, y=0.92)
122
123     for i, h in enumerate(H_LIST):
124         # オイラー法の結果を取得
125         t_e, theta_e, energy_e = results_for_table[('オイラー法', h)]
126         # RK4の結果を取得
127         t_r, theta_r, energy_r = results_for_table[('RK4', h)]
128
129         # 図1: 角度のプロット
130         axes_angle[i].plot(t_e, theta_e, label='オイラー法', color='blue')
131         axes_angle[i].plot(t_r, theta_r, label='RK4', color='orange')
132         axes_angle[i].set_title(f'h = {h} s')
133         axes_angle[i].set_ylabel('角度 $\\theta$ (rad)')
134         axes_angle[i].grid(True)
135         axes_angle[i].legend()
136
137         # 図2: エネルギーのプロット
138         axes_energy[i].plot(t_e, energy_e, label='オイラー法', color='blue')
```