计算物理作业 7

杨远青　22300190015　CompPhys 24

2024 年 11 月 22 日

在尝试抵御 GPT 的诱惑!

1　题目 1：单摆运动积分

1.1　题目描述

Write a code to numerically solves the motion of a simple pendulum using **Euler's method, midpoint method, RK4 method** and **Euler-trapezoidal method** (implement these methods by yourself). Plot the angle and total energy as a function of time. Explain the results.

1.2　程序描述

本程序内置了一个 Pendulum 类，具有绳长，质量（小球视作质点），初始角度，初始角速度，重力加速度等属性。通过调用 Pendulum 类的方法，可以使用 Euler's method, midpoint method, RK4 method 和 Euler-trapezoidal method 来求解简单摆的运动，会返回角度与角速度的 numpy 数组。类的方法还包括辅助的导数计算，即演化方程

$$\frac{d\theta}{dt} = \omega,$$
$$\frac{d\omega}{dt} = -\frac{g}{L}\sin(\theta),$$

与总能量采集方法

$$E = T + V = \frac{1}{2}m(\omega L)^2 + mgL(1-\cos\theta)$$

主程序还有内置的解析解、误差计算与用户输入采集函数，其中解析解借助了 `scipy.special` 的雅可比椭圆积分 `sn,cn`，模数 $k=\sin(\theta_0/2)$，固有频率 $\omega_0=\sqrt{\frac{g}{L}}$，所以对大角度的摆动也是精确的。

$$\theta(t) = 2\arcsin\left(k\,\mathrm{sn}(\omega_0 t + \frac{\pi}{2}, k^2)\right)$$

$$\omega(t) = \frac{2k\omega_0\,\mathrm{cn}(\omega_0 t + \frac{\pi}{2}, k^2)}{\sqrt{1-k^2\,\mathrm{sn}^2(\omega_0 t + \frac{\pi}{2}, k^2)}}$$

1.2.1　欧拉法 (Euler’s Method)

$$\theta_{i+1} = \theta_i + h\cdot\left.\frac{d\theta}{dt}\right|_{t_i} \qquad \omega_{i+1} = \omega_i + h\cdot\left.\frac{d\omega}{dt}\right|_{t_i}$$

1.2.2 中点法 (Midpoint Method)

$$\begin{aligned}
\text{计算中点值:}\quad & \theta_{\text{mid}} = \theta_i + \frac{h}{2} \cdot \left.\frac{d\theta}{dt}\right|_{t_i}, \quad \omega_{\text{mid}} = \omega_i + \frac{h}{2} \cdot \left.\frac{d\omega}{dt}\right|_{t_i} \\
\text{使用中点斜率更新:}\quad & \theta_{i+1} = \theta_i + h \cdot \left.\frac{d\theta}{dt}\right|_{\text{mid}}, \quad \omega_{i+1} = \omega_i + h \cdot \left.\frac{d\omega}{dt}\right|_{\text{mid}}
\end{aligned}$$

1.2.3 四阶龙格-库塔法 (RK4 Method)

$$\begin{aligned}
\text{第一步 } (k_1):\quad & k_1^\theta = \left.\frac{d\theta}{dt}\right|_{t_i,\theta_i,\omega_i}, \quad k_1^\omega = \left.\frac{d\omega}{dt}\right|_{t_i,\theta_i,\omega_i}; \\
\text{第二步 } (k_2):\quad & k_2^\theta = \left.\frac{d\theta}{dt}\right|_{t_i+\frac{h}{2},\theta_i+\frac{h}{2}k_1^\theta,\omega_i+\frac{h}{2}k_1^\omega}, \quad k_2^\omega = \left.\frac{d\omega}{dt}\right|_{t_i+\frac{h}{2},\theta_i+\frac{h}{2}k_1^\theta,\omega_i+\frac{h}{2}k_1^\omega}; \\
\text{第三步 } (k_3):\quad & k_3^\theta = \left.\frac{d\theta}{dt}\right|_{t_i+\frac{h}{2},\theta_i+\frac{h}{2}k_2^\theta,\omega_i+\frac{h}{2}k_2^\omega}, \quad k_3^\omega = \left.\frac{d\omega}{dt}\right|_{t_i+\frac{h}{2},\theta_i+\frac{h}{2}k_2^\theta,\omega_i+\frac{h}{2}k_2^\omega}; \\
\text{第四步 } (k_4):\quad & k_4^\theta = \left.\frac{d\theta}{dt}\right|_{t_i+h,\theta_i+hk_3^\theta,\omega_i+hk_3^\omega}, \quad k_4^\omega = \left.\frac{d\omega}{dt}\right|_{t_i+h,\theta_i+hk_3^\theta,\omega_i+hk_3^\omega},; \\
\text{更新公式:}\quad & \theta_{i+1} = \theta_i + \frac{h}{6}\left(k_1^\theta + 2k_2^\theta + 2k_3^\theta + k_4^\theta\right), \quad \omega_{i+1} = \omega_i + \frac{h}{6}\left(k_1^\omega + 2k_2^\omega + 2k_3^\omega + k_4^\omega\right).
\end{aligned}$$

1.2.4 欧拉-梯形法 (Euler-Trapezoidal Method)

$$\begin{aligned}
\text{预测:}\quad & \theta_{\text{pred}} = \theta_i + h \cdot \left.\frac{d\theta}{dt}\right|_{t_i}, \quad \omega_{\text{pred}} = \omega_i + h \cdot \left.\frac{d\omega}{dt}\right|_{t_i} \\
\text{校正:}\quad & \theta_{i+1} = \theta_i + \frac{h}{2}\left(\left.\frac{d\theta}{dt}\right|_{t_i} + \left.\frac{d\theta}{dt}\right|_{\text{pred}}\right) \quad \omega_{i+1} = \omega_i + \frac{h}{2}\left(\left.\frac{d\omega}{dt}\right|_{t_i} + \left.\frac{d\omega}{dt}\right|_{\text{pred}}\right)
\end{aligned}$$

1.3 伪代码

Powered by LaTeX pseudocode generator

Algorithm 1: Euler Method for Simple Harmonic Oscillator

Input: h: Time step size (float), N: Total number of steps (int)

Output: θ: Angle array (rad), ω: Angular velocity array (rad/s)

Initialize $\theta[0] \leftarrow \theta_0$, $\omega[0] \leftarrow \omega_0$; `// Set initial conditions`

for $i \leftarrow 0$ **to** $N-1$ **do**

  Compute $(\dot\theta, \dot\omega) \leftarrow$ `Derivatives`$(\theta[i], \omega[i])$;

  Update $\theta[i+1] \leftarrow \theta[i] + h \cdot \dot\theta$, $\omega[i+1] \leftarrow \omega[i] + h \cdot \dot\omega$; `// Update values`

end

return θ, ω ; `// Return results as arrays`

Algorithm 2: Midpoint Method for Simple Harmonic Oscillator

Input: h: Time step size (float), N: Total number of steps (int)

Output: θ: Angle array (rad), ω: Angular velocity array (rad/s)

Initialize $\theta[0] \leftarrow \theta_0$, $\omega[0] \leftarrow \omega_0$; `// Set initial conditions`
for $i \leftarrow 0$ **to** $N-1$ **do**
  Compute $(\dot{\theta}, \dot{\omega}) \leftarrow$ `Derivatives(`$\theta[i], \omega[i]$`)` ; `// Slope at initial point`
  Compute $\theta_{\text{mid}} \leftarrow \theta[i] + 0.5 \cdot h \cdot \dot{\theta}$, $\omega_{\text{mid}} \leftarrow \omega[i] + 0.5 \cdot h \cdot \dot{\omega}$; `// Midpoint values`
  Compute $(\dot{\theta}_{\text{mid}}, \dot{\omega}_{\text{mid}}) \leftarrow$ `Derivatives(`$\theta_{mid}, \omega_{mid}$`)` ; `// Slope at midpoint`
  Update $\theta[i+1] \leftarrow \theta[i] + h \cdot \dot{\theta}_{\text{mid}}$, $\omega[i+1] \leftarrow \omega[i] + h \cdot \dot{\omega}_{\text{mid}}$; `// Update values`
end
return θ, ω ; `// Return results as arrays`

Algorithm 3: RK4 Method for Simple Harmonic Oscillator

Input: h: Time step size (float), N: Total number of steps (int)

Output: θ: Angle array (rad), ω: Angular velocity array (rad/s)

Initialize $\theta[0] \leftarrow \theta_0$, $\omega[0] \leftarrow \omega_0$; `// Set initial conditions`
for $i \leftarrow 0$ **to** $N-1$ **do**
  Compute $(k_1^\theta, k_1^\omega) \leftarrow$ `Derivatives(`$\theta[i], \omega[i]$`)` ; `// Stage 1`
  Compute $(k_2^\theta, k_2^\omega) \leftarrow$ `Derivatives(`$\theta[i] + 0.5 \cdot h \cdot k_1^\theta, \omega[i] + 0.5 \cdot h \cdot k_1^\omega$`)` ; `// Stage 2`
  Compute $(k_3^\theta, k_3^\omega) \leftarrow$ `Derivatives(`$\theta[i] + 0.5 \cdot h \cdot k_2^\theta, \omega[i] + 0.5 \cdot h \cdot k_2^\omega$`)` ; `// Stage 3`
  Compute $(k_4^\theta, k_4^\omega) \leftarrow$ `Derivatives(`$\theta[i] + h \cdot k_3^\theta, \omega[i] + h \cdot k_3^\omega$`)` ; `// Stage 4`
  Update $\theta[i+1] \leftarrow \theta[i] + \frac{h}{6} \cdot (k_1^\theta + 2 \cdot k_2^\theta + 2 \cdot k_3^\theta + k_4^\theta)$;
  Update $\omega[i+1] \leftarrow \omega[i] + \frac{h}{6} \cdot (k_1^\omega + 2 \cdot k_2^\omega + 2 \cdot k_3^\omega + k_4^\omega)$;
end
return θ, ω ; `// Return results as arrays`

Algorithm 4: Euler-Trapezoidal Method for Simple Harmonic Oscillator

Input: h: Time step size (float), N: Total number of steps (int)

Output: θ: Angle array (rad), ω: Angular velocity array (rad/s)

Initialize $\theta[0] \leftarrow \theta_0$, $\omega[0] \leftarrow \omega_0$; `// Set initial conditions`
for $i \leftarrow 0$ **to** $N-1$ **do**
  Compute $(\dot{\theta}, \dot{\omega}) \leftarrow$ `Derivatives(`$\theta[i], \omega[i]$`)` ; `// Predictor step slopes`
  Compute $\theta_{\text{pred}} \leftarrow \theta[i] + h \cdot \dot{\theta}$, $\omega_{\text{pred}} \leftarrow \omega[i] + h \cdot \dot{\omega}$; `// Euler predictor values`
  Compute $(\dot{\theta}_{\text{pred}}, \dot{\omega}_{\text{pred}}) \leftarrow$ `Derivatives(`$\theta_{pred}, \omega_{pred}$`)` ; `// Corrector step slopes`
  Update $\theta[i+1] \leftarrow \theta[i] + \frac{h}{2} \cdot (\dot{\theta} + \dot{\theta}_{\text{pred}})$, $\omega[i+1] \leftarrow \omega[i] + \frac{h}{2} \cdot (\dot{\omega} + \dot{\omega}_{\text{pred}})$; `// Trapezoidal corrector`
end
return θ, ω ; `// Return results as arrays`

1.4 结果示例

```
(base) gilbert@Gilbert-YoungMacBook src % python -u pendulum.py
请输入摆的参数(直接回车使用默认值):
摆长 L (m) (默认: 1.0):
质量 m (kg) (默认: 1.0):
重力加速度 g (m/s²) (默认: 9.81):
初始角度 θ₀ (rad) (默认: 1.0):
初始角速度 ω₀ (rad/s) (默认: 0.0):
时间步长 h (s) (默认: 0.05):
总模拟时间 T (s) (默认: 50.0):
```

图 1: 终端处理用户输入，此处均采用默认值

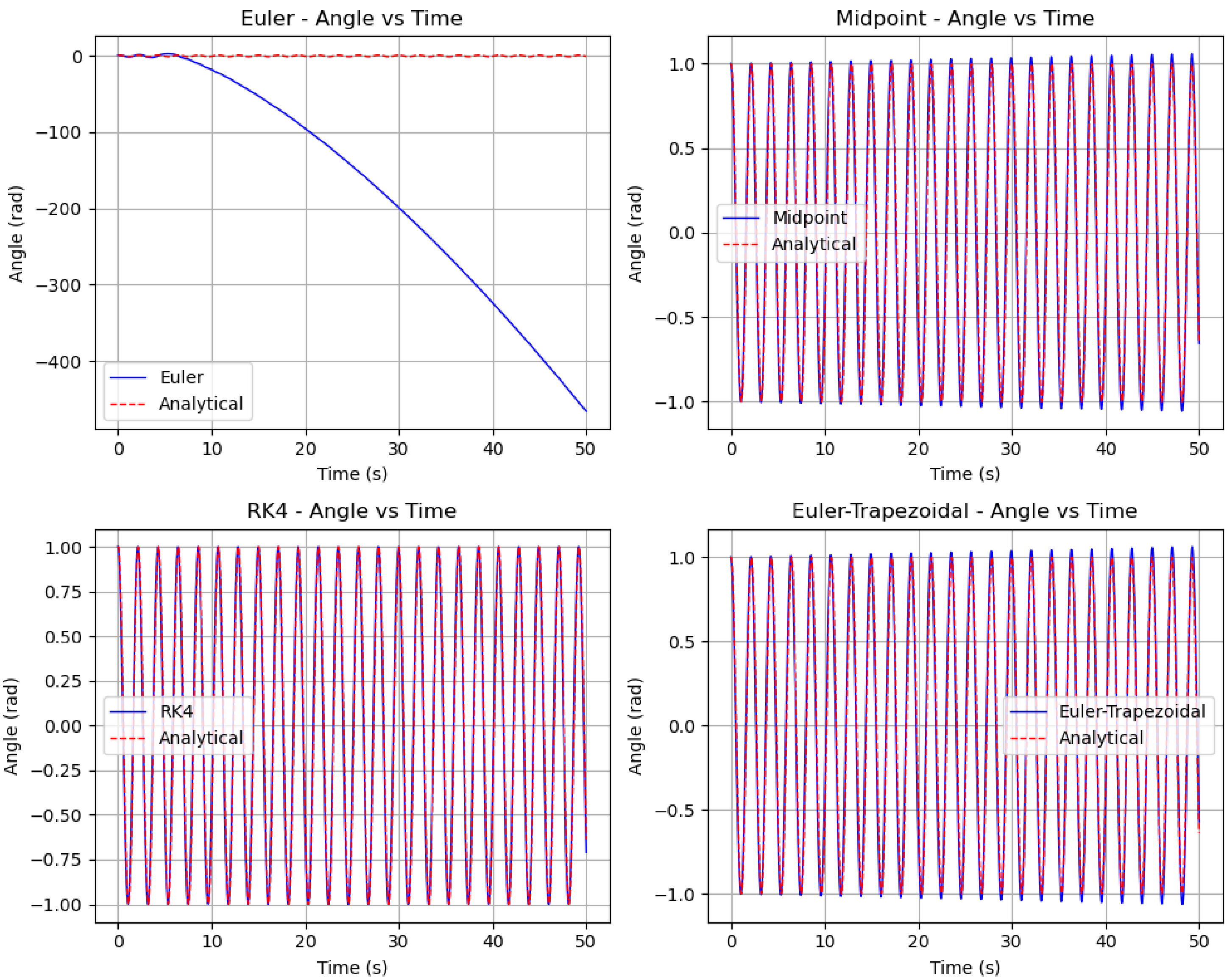

图 2: 角度随时间变化

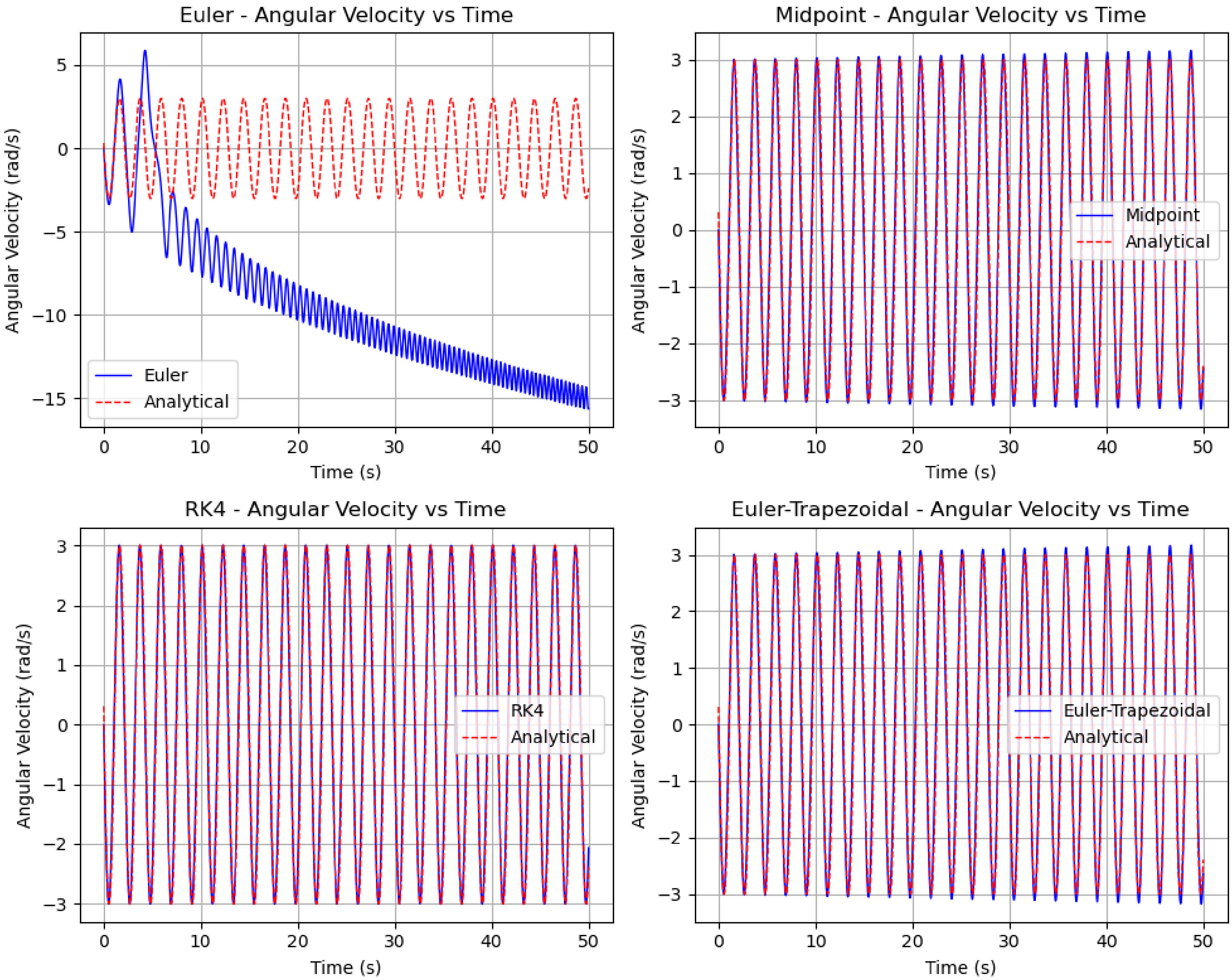
Euler - Angular Velocity vs Time
Midpoint - Angular Velocity vs Time
RK4 - Angular Velocity vs Time
Euler-Trapezoidal - Angular Velocity vs Time
Angular Velocity (rad/s)
Time (s)
Euler
Midpoint
RK4
Euler-Trapezoidal
Analytical

图 3: 角速度随时间变化

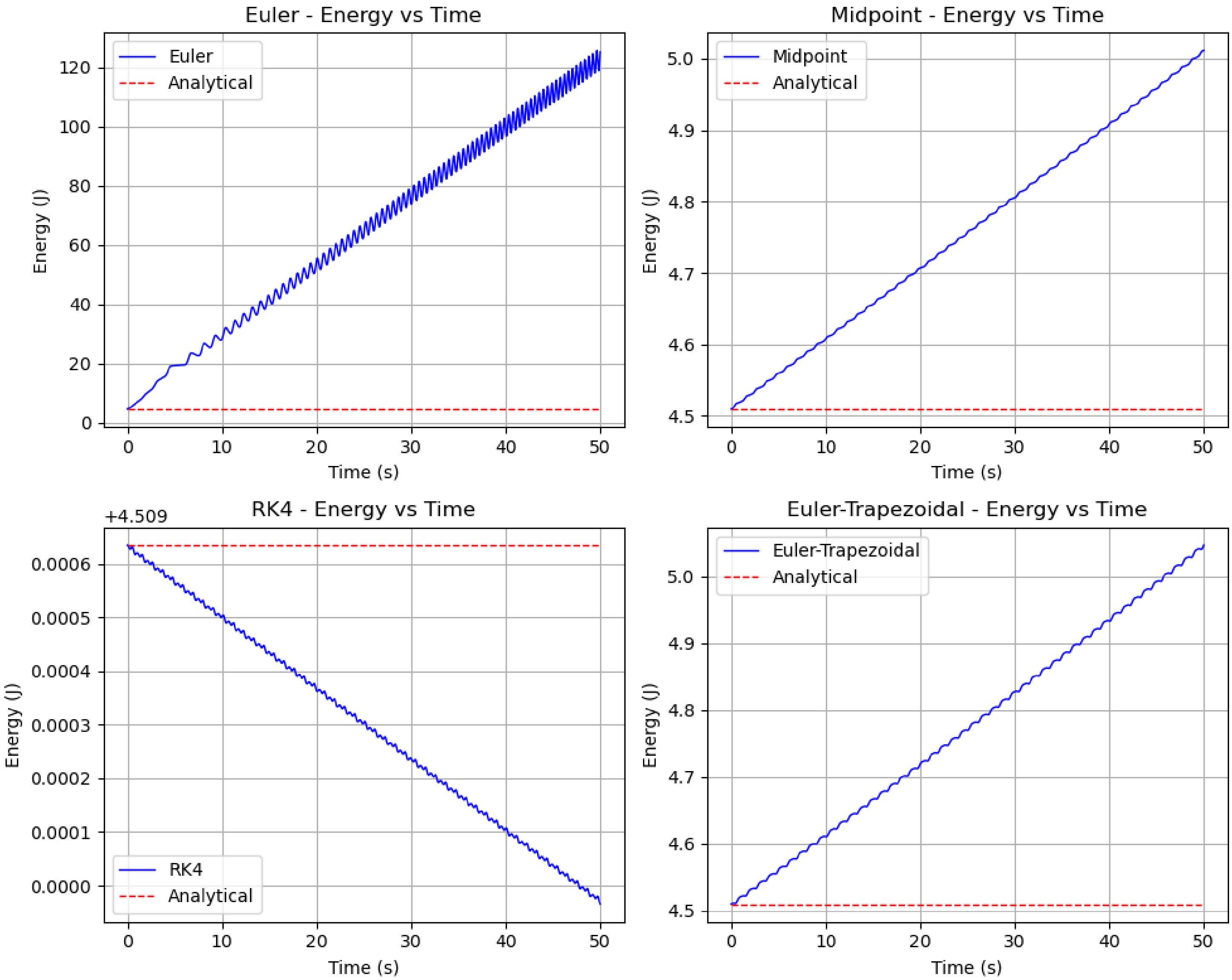
Euler - Energy vs Time
Euler
Analytical
Energy (J)
Time (s)
Midpoint - Energy vs Time
Midpoint
Analytical
RK4 - Energy vs Time
+4.509
RK4
Euler-Trapezoidal - Energy vs Time
Euler-Trapezoidal

图 4: 能量随时间漂移

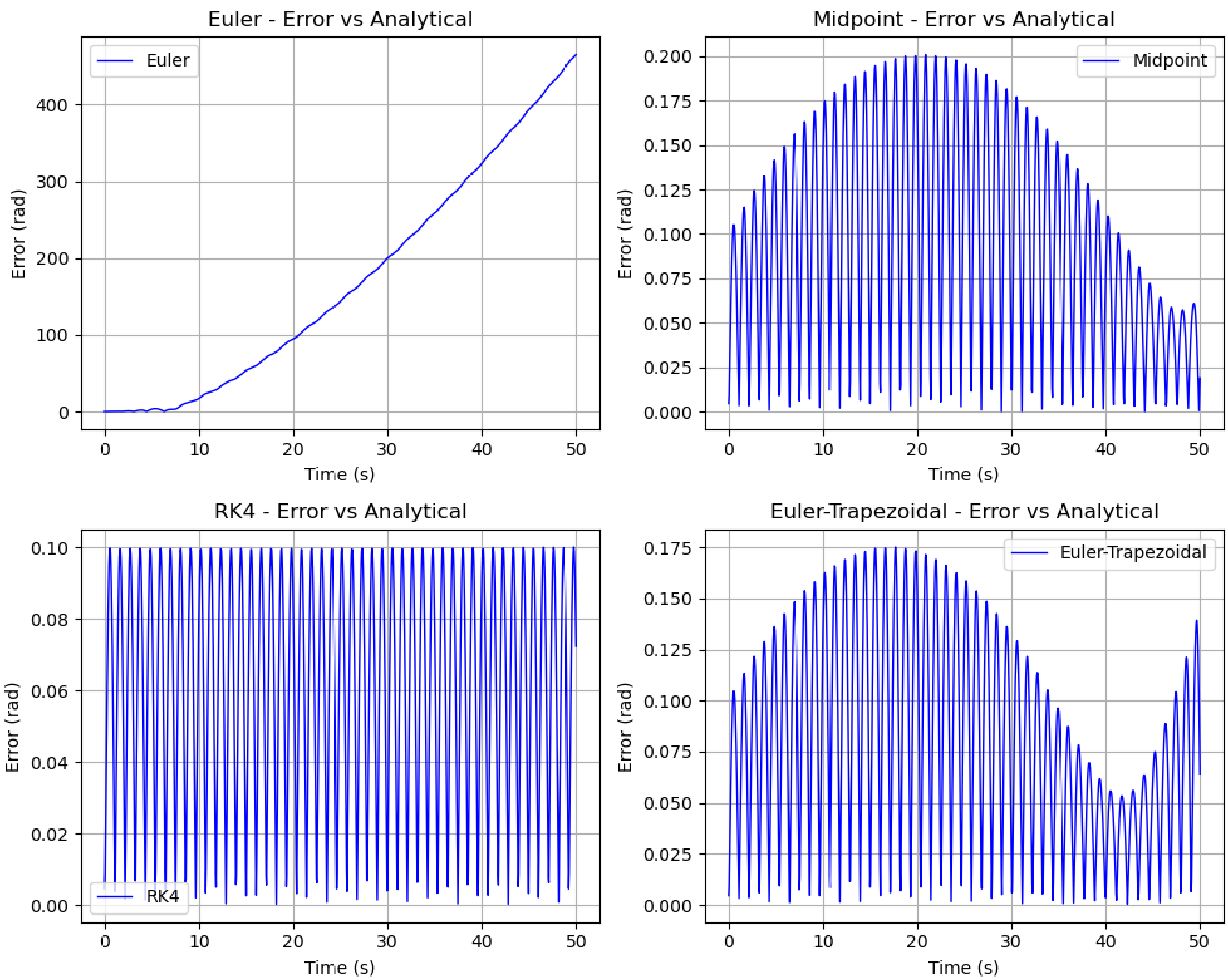

图 5: 角度与解析解误差随时间变化

可以看出，四种方法对比中，RK4 方法的能量漂移最小，角度与角速度误差也最小。中点法与欧拉-梯形法次之，虽然角度、角速度误差在长时间后才逐渐显现，但能量漂移较大。欧拉法误差最大，一段时间后就崩溃。综上，RK4 方法最为精确且稳定。

2 题目 2：径向薛定谔方程求解

2.1 题目描述

Write a code to numerically solve the radial Schrödinger equation for

$$\left[-\frac{1}{2}\nabla^2 + V(\mathbf{r})\right]\psi(\mathbf{r}) = E\psi(\mathbf{r}), \quad V(\mathbf{r}) = V(r)$$

1. $V(r) = \frac{1}{r}$ (hydrogen atom)

2. Considering the following potential:

$$V(r) = -\frac{Z_{\text{ion}}}{r}\text{erf}\left(\frac{r}{\sqrt{2}r_{\text{loc}}}\right) + \exp\left[-\frac{1}{2}\left(\frac{r}{r_{\text{loc}}}\right)^2\right] \times \left[C_1 + C_2\left(\frac{r}{r_{\text{loc}}}\right)^2 + C_3\left(\frac{r}{r_{\text{loc}}}\right)^4 + C_4\left(\frac{r}{r_{\text{loc}}}\right)^6\right]$$

where erf is the error function. And for Li, you could set:

- $Z_{\text{ion}} = 3$
- $r_{\text{loc}} = 0.4$
- $C_1 = -14.0093922$
- $C_2 = 9.5099073$
- $C_3 = -1.7532723$
- $C_4 = 0.0834586$

Compute and plot the first three eigenstates. You could find more information about 'how to solve radial Schrödinger equation' and 'use of non-uniform grid (optional)' in the PPT.

Special Note: You may call any library functions for diagonalization.

2.2 程序描述

2.3 伪代码

Powered by LaTeX pseudocode generator

2.4 结果示例