计算物理作业 7

李明钰 22307110156

2024 年 11 月 23 日

1 题目 1：单摆运动方程

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, 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

题目要求解一个简谐摆1的运动，其运动所满足的偏微分方程为

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0 \qquad \theta|_{t=0} = \theta_0 \qquad \frac{d\theta}{dt}|_{t=0} = 0 \tag{1}$$

令 $\omega = \frac{d\theta}{dt}$，上述二阶微分方程可以被转化为两个一阶微分方程所组成的方程组。

$$\begin{aligned} \frac{d\omega}{dt} &= -\frac{g}{L}\sin\theta \\ \frac{d\theta}{dt} &= \omega \\ \omega|_{t=0} = \omega_0 &\qquad \theta|_{t=0} = \theta_0 \end{aligned} \tag{2}$$

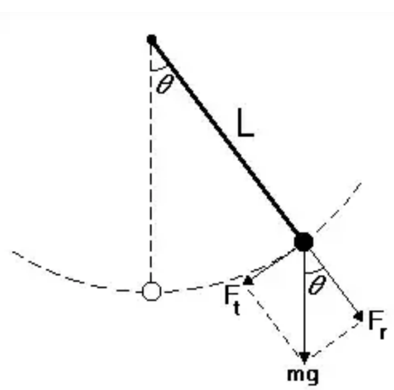

图 1: 简谐摆受力示意图

1.3 伪代码

PS: 这里因为 Latex 的 algorithm 包和 algpseudocode 包之间的冲突末尾多了一个 0，但是不影响阅读

Algorithm 1 Euler Method

Require: θ_0, ω_0 $t \in \mathrm{R}^n$

Ensure: $\theta(t)$,$E(t)$

$\Delta t \leftarrow \mathrm{diff}(t)$

$\theta[0] \leftarrow \theta_0$

$\omega[0] \leftarrow \omega_0$

for $i = 1$ to $n-1$ **do**

  $h \leftarrow \Delta t[i-1]$

  $\omega[i] \leftarrow \omega[i-1] - h \cdot \frac{g}{L} \cdot \sin(\theta[i-1])$

  $\theta[i] \leftarrow \theta[i-1] + h \cdot \omega[i-1]$

end for

$E \leftarrow -m \cdot g \cdot L \cdot \cos(\theta) + 0.5 \cdot m \cdot (L \cdot \omega)^2$

return $\theta(t), E(t)$ =0

Algorithm 2 Midpoint Method

Require: Initial conditions θ_0, ω_0, time array t, constants g, L, m

$n \leftarrow$ length of t

$\Delta t \leftarrow$ array of time steps (computed as $t[i+1] - t[i]$)

Initialize arrays: $\theta, \omega, \Delta_\theta, \Delta_\omega \leftarrow$ zeros of size n

$\theta[0] \leftarrow \theta_0$, $\omega[0] \leftarrow \omega_0$

for $i \leftarrow 1$ to $n-1$ **do**

  $h \leftarrow \Delta t[i-1]$

  $\Delta_\omega[i-1] \leftarrow h \cdot \left(-\frac{g}{L}\sin(\theta[i-1])\right)$

  $\Delta_\theta[i-1] \leftarrow h \cdot \omega[i-1]$

  $\omega[i] \leftarrow \omega[i-1] + h \cdot \left(-\frac{g}{L}\sin\left(\theta[i-1] + 0.5 \cdot \Delta_\theta[i-1]\right)\right)$

  $\theta[i] \leftarrow \theta[i-1] + h \cdot (\omega[i-1] + 0.5 \cdot \Delta_\omega[i-1])$

end for

$E \leftarrow -m \cdot g \cdot L \cdot \cos(\theta) + 0.5 \cdot m \cdot (L \cdot \omega)^2$

return θ, E =0

Algorithm 3 RK4

Require: Initial conditions θ_0, ω_0, time array t, constants g, L, m

$n \leftarrow$ length of t

$\Delta t \leftarrow$ array of time steps (computed as $t[i+1] - t[i]$)

Initialize arrays: $\theta, \omega \leftarrow$ zeros of size n

$\theta[0] \leftarrow \theta_0$, $\omega[0] \leftarrow \omega_0$

for $i \leftarrow 1$ to $n-1$ **do**

  $h \leftarrow \Delta t[i-1]$

  $K_1^\theta \leftarrow \omega[i-1]$

$K_1^\omega \leftarrow -\frac{g}{L}\sin(\theta[i-1])$

$K_2^\theta \leftarrow \omega[i-1] + 0.5 \cdot h \cdot K_1^\omega$

$K_2^\omega \leftarrow -\frac{g}{L}\sin\left(\theta[i-1] + 0.5 \cdot h \cdot K_1^\theta\right)$

$K_3^\theta \leftarrow \omega[i-1] + 0.5 \cdot h \cdot K_2^\omega$

$K_3^\omega \leftarrow -\frac{g}{L}\sin\left(\theta[i-1] + 0.5 \cdot h \cdot K_2^\theta\right)$

$K_4^\theta \leftarrow \omega[i-1] + h \cdot K_3^\omega$

$K_4^\omega \leftarrow -\frac{g}{L}\sin\left(\theta[i-1] + h \cdot K_3^\theta\right)$

$\theta[i] \leftarrow \theta[i-1] + \frac{h}{6} \cdot \left(K_1^\theta + 2 \cdot K_2^\theta + 2 \cdot K_3^\theta + K_4^\theta\right)$

$\omega[i] \leftarrow \omega[i-1] + \frac{h}{6} \cdot (K_1^\omega + 2 \cdot K_2^\omega + 2 \cdot K_3^\omega + K_4^\omega)$

end for

$E \leftarrow -m \cdot g \cdot L \cdot \cos(\theta) + 0.5 \cdot m \cdot (L \cdot \omega)^2$

return θ, E =0

Algorithm 4 EulerTrapezoidalMethod

Require: Initial conditions θ_0, ω_0, time array t, constants g, L, m, tolerance ϵ

$n \leftarrow$ length of t

$\Delta t \leftarrow$ array of time steps (computed as $t[i+1] - t[i]$)

Initialize arrays: $\theta, \omega \leftarrow$ zeros of size n

$\theta[0] \leftarrow \theta_0$, $\omega[0] \leftarrow \omega_0$

for $i \leftarrow 1$ to $n-1$ **do**

$h \leftarrow \Delta t[i-1]$

Euler Prediction:

$\omega_{\text{Euler}} \leftarrow \omega[i-1] + h \cdot \left(-\frac{g}{L}\sin(\theta[i-1])\right)$

$\theta_{\text{Euler}} \leftarrow \theta[i-1] + h \cdot \omega[i-1]$

First Correction:

$\omega_0^{\text{correct}} \leftarrow \omega[i-1] + \frac{h}{2} \cdot \left(\left(-\frac{g}{L}\sin(\theta[i-1])\right) + \left(-\frac{g}{L}\sin(\theta_{\text{Euler}})\right)\right)$

$\theta_0^{\text{correct}} \leftarrow \theta[i-1] + \frac{h}{2} \cdot (\omega[i-1] + \omega_{\text{Euler}})$

$\omega_1^{\text{correct}} \leftarrow \omega[i-1] + \frac{h}{2} \cdot \left(\left(-\frac{g}{L}\sin(\theta_0^{\text{correct}})\right) + \left(-\frac{g}{L}\sin(\theta[i-1])\right)\right)$

$\theta_1^{\text{correct}} \leftarrow \theta[i-1] + \frac{h}{2} \cdot \left(\omega[i-1] + \omega_0^{\text{correct}}\right)$

Iterative Refinement:

while $\left|\omega_0^{\text{correct}} - \omega_1^{\text{correct}}\right| > \epsilon$ **or** $\left|\theta_0^{\text{correct}} - \theta_1^{\text{correct}}\right| > \epsilon$ **do**

$\omega_0^{\text{correct}} \leftarrow \omega_1^{\text{correct}}$

$\theta_0^{\text{correct}} \leftarrow \theta_1^{\text{correct}}$

$\omega_1^{\text{correct}} \leftarrow \omega[i-1] + \frac{h}{2} \cdot \left(\left(-\frac{g}{L}\sin(\theta_0^{\text{correct}})\right) + \left(-\frac{g}{L}\sin(\theta[i-1])\right)\right)$

$\theta_1^{\text{correct}} \leftarrow \theta[i-1] + \frac{h}{2} \cdot \left(\omega[i-1] + \omega_0^{\text{correct}}\right)$

end while

$\theta[i] \leftarrow \theta_0^{\text{correct}}$

$\omega[i] \leftarrow \omega_0^{\text{correct}}$

end for

$E \leftarrow -m \cdot g \cdot L \cdot \cos(\theta) + 0.5 \cdot m \cdot (L \cdot \omega)^2$

return θ, E =0

1.4 输入输出实例

初始条件设为最常见的

$$\theta|_{t=0} = \frac{\pi}{2} \qquad \omega|_{t=0} = 0 \tag{3}$$

用不同算法解得的 $\theta(t)$(图2) 和 $E(t)$(图3), 可以发现欧拉法的计算得到的能量较快发散，而其他算法发散的很慢。

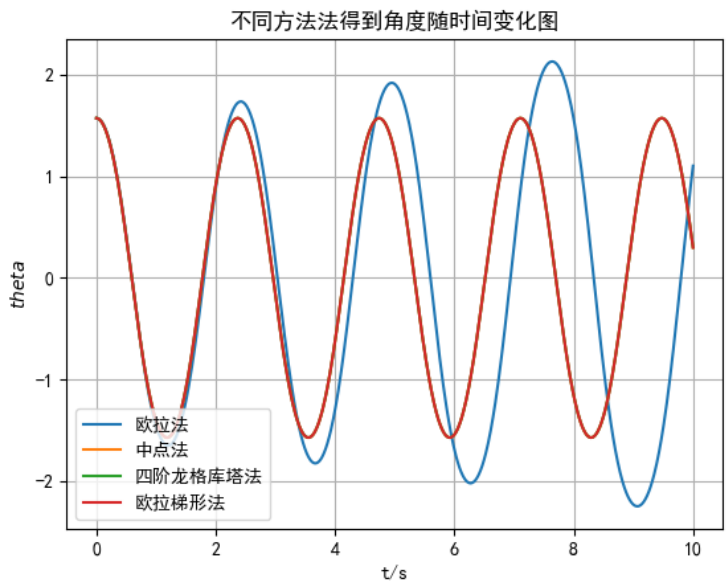

图 2: 不同算法解得的 $\theta(t)$

1.5 对现象的解释

所有的算法都会有一个误差最终导致发散，这一发散速度与算法的误差量级有关，其中欧拉法 (算法1) 的全局误差 $\propto O(h)$，而中点法 (算法2) 全局误差 $\propto O(h^2)$，四阶龙格-库塔方法 (算法3) 全局误差 $\propto O(h^4)$，而欧拉梯形法则 (算法4) 的误差则只与我们设置的可允许误差 ϵ 有关，其全局误差 $\propto O(\epsilon)$，但是此误差不是单向误差不容易堆积，因此其结果发散速度也很慢。

2 题目 2：氢原子和锂原子的薛定谔方程

2.1 题目描述

Write a code to numerically solves radial Schrödinger equation for

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

(1)V(r) = -1/r(hydrogen atom) (2)V(r) = -Z_{ion}

2.2 程序描述

适用有限差分法 (算法5) 分别解氢原子和锂原子的薛定谔方程，并输出能量最低的三个本征能量。

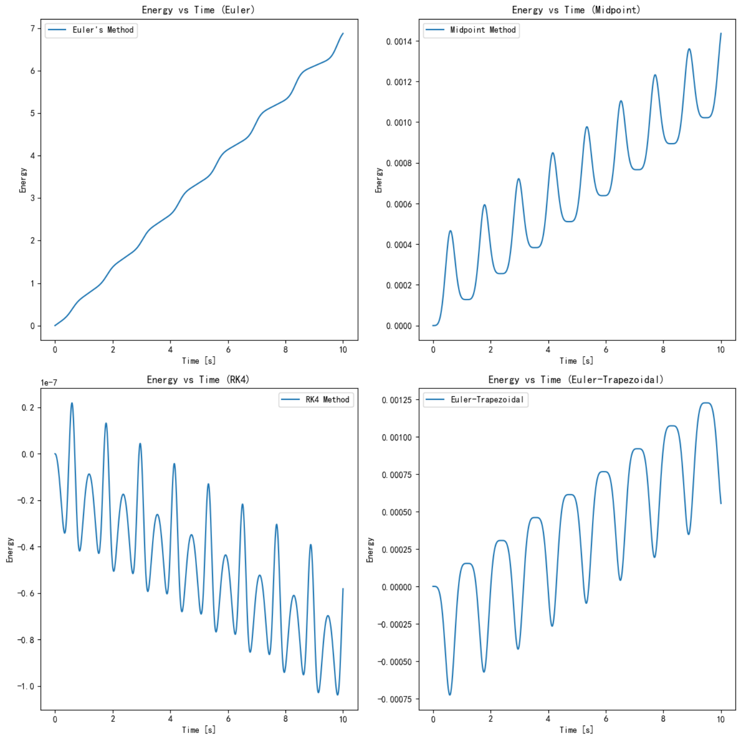

图 3: 不同算法解得的 $E(t)$

2.3 伪代码

PS: 这里因为 Latex 的 algorithm 包和 algpseudocode 包之间的冲突行号全部变为 0 且末尾多了一个 0，但是不影响阅读

Algorithm 5 Finite Difference Method for Solving Radial Schrödinger Equation

Input: Potential function `potential_function`
Output: Eigenvalues and Eigenvectors
$r_{\min} \leftarrow 1$ {Initial radius (avoid singularity at r=0)}
$num\_points \leftarrow 2000$ {Number of grid points}
$t_{\max} \leftarrow \ln(40)$ {Transformed time range}
$dt \leftarrow \frac{t_{\max}}{num\_points}$ {Time step size}
$t\_values \leftarrow \text{linspace}(dt, t_{\max}, num\_points)$
$r\_values \leftarrow r_{\min} \cdot (\exp(t\_values) - 1)$ {Generate radial grid points via transformation}
$eigenvalues \leftarrow []$ {List to store eigenvalues}
$eigenvectors \leftarrow []$ {List to store eigenvectors}
for angular_momentum = 0, 1, 2 **do** {For each angular momentum quantum number l = 0, 1, 2}
  $hamiltonian \leftarrow \text{zeros}(num\_points, num\_points)$ {Construct Hamiltonian matrix}
  $potential\_values \leftarrow \text{potential\_function}(r\_values)$
  for $i = 0$ to $num\_points - 1$ **do**
    $scaling\_factor \leftarrow \frac{1}{(r_{\min} \cdot \exp(t\_values[i]))^2} \div 2$
    $hamiltonian[i,i] \leftarrow potential\_values[i] + \frac{angular\_momentum \cdot (angular\_momentum+1)}{r\_values[i]^2} + \frac{2}{dt^2} \cdot scaling\_factor$
    if $i < num\_points - 1$ **then**

$hamiltonian[i, i+1] \leftarrow -\frac{1}{dt^2} \cdot scaling\_factor + \frac{1}{2dt} \cdot scaling\_factor$

end if

if $i > 0$ **then**

$hamiltonian[i, i-1] \leftarrow -\frac{1}{dt^2} \cdot scaling\_factor - \frac{1}{2dt} \cdot scaling\_factor$

end if

end for

$eigvals, eigvecs \leftarrow \mathrm{eig}(hamiltonian)$ {Solve for eigenvalues and eigenvectors}

$sorted\_indices \leftarrow \mathrm{argsort}(eigvals)$ {Sort eigenvalues}

for idx in $sorted\_indices[0:3]$ **do** {Take the first 3 smallest eigenvalues}

$eigenvalues.append([eigvals[idx], angular\_momentum, angular\_momentum + idx + 1])$

$eigenvectors.append(eigvecs[:, idx])$

end for

end for

$eigenvalues \leftarrow \mathrm{array}(eigenvalues)$

$lowest\_indices \leftarrow \mathrm{argsort}(eigenvalues[:, 0])[0:3]$ {Get indices of the three lowest energy eigenvalues} =0

2.4 输入输出实例

计算得到结果如下：

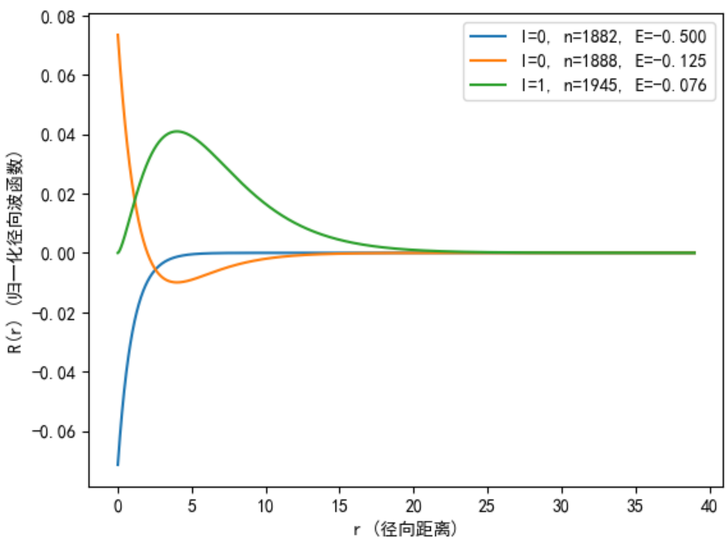

图 4: 氢原子能量最低的三个本征态 R(r)

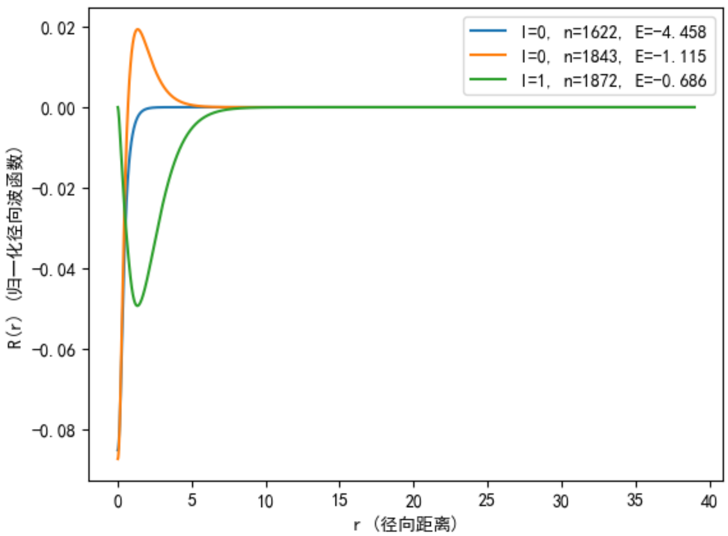

图 5: 锂原子能量最低的三个本征态 R(r)