计算物理作业 7

谢昀城 22307110070

2024 年 11 月 22 日

1 题目 1

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

在本程序中我们将使用 Euler’ s method, midpoint method, RK4, Euler-trapezoidal method 计算单摆的运动，并通过绘制 $\theta - t$,$\dot{\theta} - t$,$E - t$ 图来测试结果的稳定性。

本程序源文件为 simplePendulum.py，在终端进入当前目录，使用命令 python -u simplePendulum.py 运行本程序。运行时请保证 Python 第三方库 Numpy,Matplotlib 已安装。程序开发环境为 Python3.12.3，可在 Python3.8 以上版本中运行。

1.3 伪代码

Pendulum Class:

Algorithm 1 PendulumClass

Class: Pendulum($m, g, \theta, \dot{\theta}$)
method:
function GetA
  OUTPUT: acceleration
  acceleration $\leftarrow -g \times m \times \sin(\theta)$
  return acceleration
end function
function GetZ
  OUTPUT: stateVector
  stateVector $\leftarrow$ array($[\theta, \dot{\theta}]$)
  return stateVector
end function
function GetHZ

OUTPUT: differentialStateVector

differentialStateVector $\leftarrow$ array($[\dot{\theta}, \text{GetA}()]$)

return differentialStateVector

end function

function UPDATE($newtheta, newdtheta$)

INPUT: *newtheta* (new angle), *newdtheta* (new angular velocity)

$\theta \leftarrow \text{mod}(newtheta + \pi, 2 \times \pi) - \pi$

$\dot{\theta} \leftarrow dtheta$

end function

end Class

EulerStep:

Algorithm 2 EulerStep

function EULERSTEP($pendulum, dt$)

INPUT: *pendulum* (Pendulum object), dt (time step)

OUTPUT: None

$z \leftarrow pendulum$.GetZ()

$Hz \leftarrow pendulum$.GetHZ()

$z \leftarrow z + Hz \times dt$

$pendulum$.Update($z[0], z[1]$)

end function

MidpointStep:

Algorithm 3 MidpointStep

function MIDPOINTSTEP($pendulum, dt$)

INPUT: *pendulum* (Pendulum object), dt (time step)

OUTPUT: None

$z \leftarrow pendulum$.GetZ()

$Hz \leftarrow pendulum$.GetHZ()

$z \leftarrow z + Hz \times \frac{dt}{2}$

$pendulum$.Update($z[0], z[1]$)

$Hz \leftarrow pendulum$.GetHZ()

$z \leftarrow z + Hz \times \frac{dt}{2}$

$pendulum$.Update($z[0], z[1]$)

end function

RK4Step:

Algorithm 4 RK4Step

function RK4STEP($pendulum, dt$)

INPUT: *pendulum* (Pendulum object), dt (time step)

OUTPUT: None

$z \leftarrow pendulum$.GetZ()
$Hz \leftarrow pendulum$.GetHZ()
$k1 \leftarrow Hz \times dt$
$pendulum$.Update($z[0] + \frac{k1[0]}{2}, z[1] + \frac{k1[1]}{2}$)
$Hz \leftarrow pendulum$.GetHZ()
$k2 \leftarrow Hz \times dt$
$pendulum$.Update($z[0] + \frac{k2[0]}{2}, z[1] + \frac{k2[1]}{2}$)
$Hz \leftarrow pendulum$.GetHZ()
$k3 \leftarrow Hz \times dt$
$pendulum$.Update($z[0] + k3[0], z[1] + k3[1]$)
$Hz \leftarrow pendulum$.GetHZ()
$k4 \leftarrow Hz \times dt$
$pendulum$.Update($z[0] + \frac{k1[0]+2\times k2[0]+2\times k3[0]+k4[0]}{6}, z[1] + \frac{k1[1]+2\times k2[1]+2\times k3[1]+k4[1]}{6}$)
end function

EulerTrapezoidalStep:

Algorithm 5 EulerTrapezoidalStep

function EulerTrapezoidalStep($pendulum, dt, tol = 1e-6, maxiter = 1000$)
INPUT: *pendulum* (Pendulum object), *dt* (time step), *tol* (tolerance), *maxiter* (maximum iterations)
OUTPUT: None
$z0 \leftarrow pendulum$.GetZ()
$Hz0 \leftarrow pendulum$.GetHZ()
$z \leftarrow z0$.Copy()
for $i \leftarrow 0$ **to** $maxiter - 1$ **do**
$Hzi \leftarrow pendulum$.GetHZ()
$pendulum$.Update($z0[0] + \frac{Hz0[0]\times dt}{2} + \frac{Hzi[0]\times dt}{2}, z0[1] + \frac{Hz0[1]\times dt}{2} + \frac{Hzi[1]\times dt}{2}$)
$zhis \leftarrow z$.Copy()
$z \leftarrow pendulum$.GetZ()
if np.linalg.norm($z - zhis$) $< tol$ **then**
break
end if
end for
end function

1.4 输入输出实例

对于本程序，运行后会生成图1至6为"pendulum_dt=$dt_method=$method.png" 至当前目录 output 路径下, 分别为不同求解方法得到的 $\theta - t$,$\dot{\theta} - t$,$E - t$ 图像。程序运行截图为图7。

从图1-4可以看到在 $dt = 0.1s$ 时, 在 0-100s 范围内,Euler 方法和 Midpoint 方法的结果随着时间增长，角度振幅和能量快速发散，且 Euler 方法增长更大。而 RK4 和 EulerTrapezoidal 方法的结果稳定，能量和角度振幅基本保持不变，RK4 方法观察到能量随时间轻微减小。

为放大 RK4 和 EulerTrapezoidal 方法的结果的区别，我们将时间范围扩大至 1000s，$dt = 1.0s$，从5-6可以看到 EulerTrapezoidal 方法的结果虽然能量出现了一定程度的振荡，但仍然稳定不会发散。而 RK4 方法的能量则随着时间逐渐衰减到 0。这说明通过精度自适应控制迭代次数的 EulerTrapezoidal 方法能够更好的保持体系的稳定性。

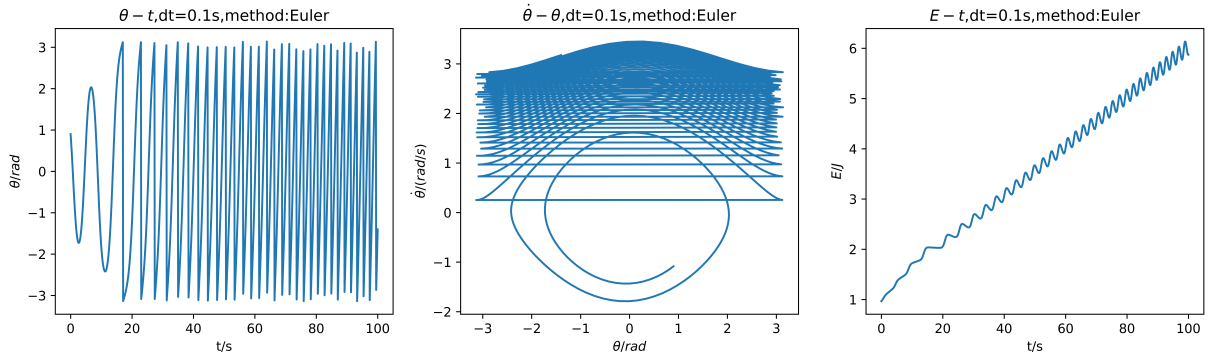

图 1: Euler method of 0 to 100s,dt=0.1s

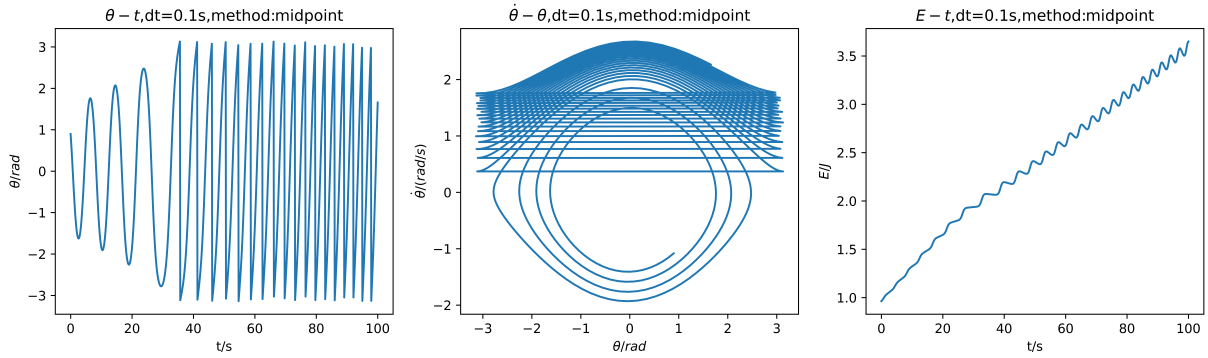

图 2: Midpoint method of 0 to 100s,dt=0.1s

2 题目 2

Write a code to numerically solve the radial Schrödinger equation for

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

For the hydrogen atom:

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

For the local potential:

$$V_{\text{loc}}(r) = -\frac{Z_{\text{ion}}}{r}\,\text{erf}\left(\frac{r}{\sqrt{2}r_{\text{loc}}}\right) + \exp\left[-\frac{1}{2}\left(\frac{r}{r_{\text{loc}}}\right)^2\right]$$

$$\times\left[C_1 + C_2\left(\frac{r}{r_{\text{loc}}}\right)^2 + C_3\left(\frac{r}{r_{\text{loc}}}\right)^4 + C_4\left(\frac{r}{r_{\text{loc}}}\right)^6\right].$$

For the lithium (Li) atom, the constants are:

$$r_{\text{loc}} = 0.4000000, \quad C_1 = -14.0093922, \quad C_2 = 9.5099073,$$

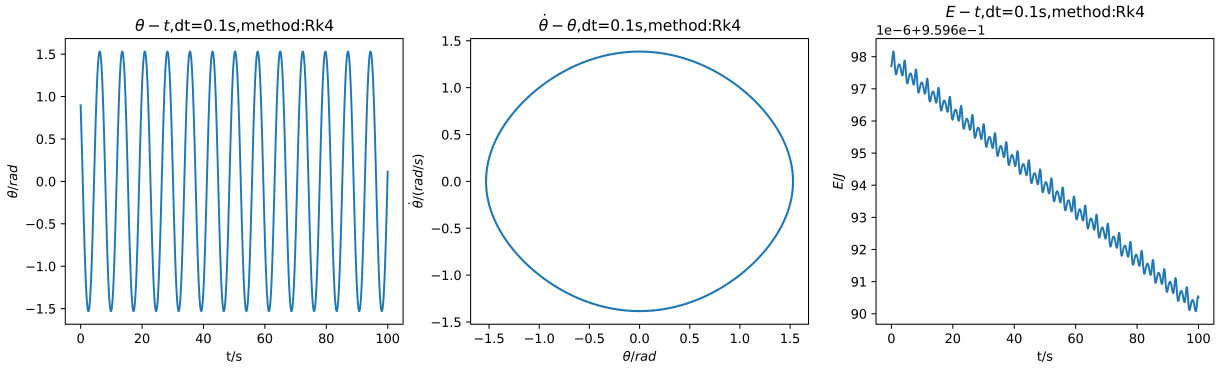

图 3: RK4 method of 0 to 100s,dt=0.1s

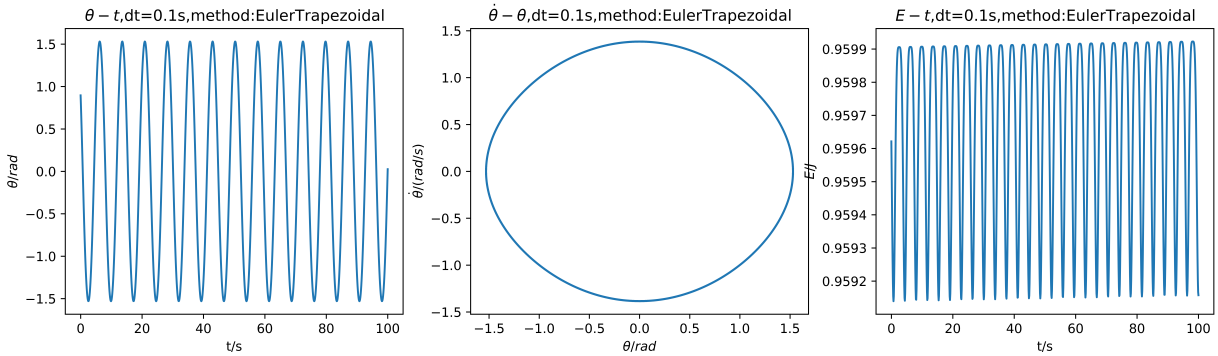

图 4: EulerTrapezoidal method of 0 to 100s,dt=0.1s

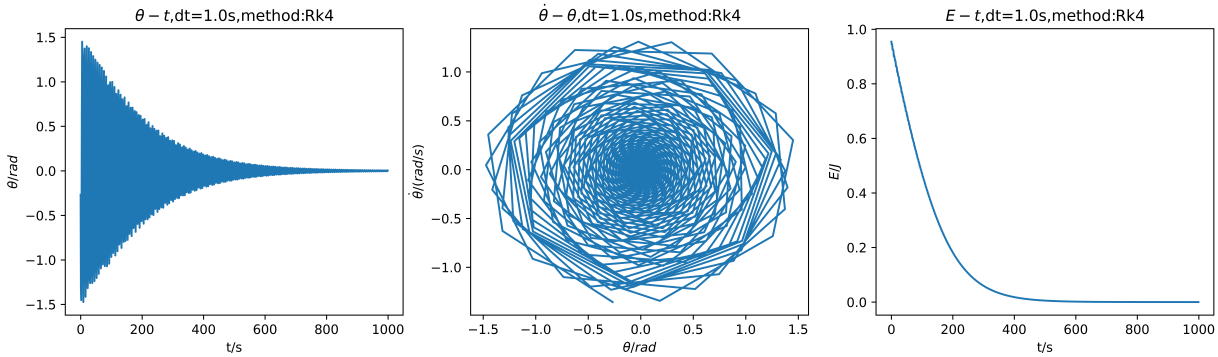

图 5: RK4 method of 0 to 1000s,dt=1.0s

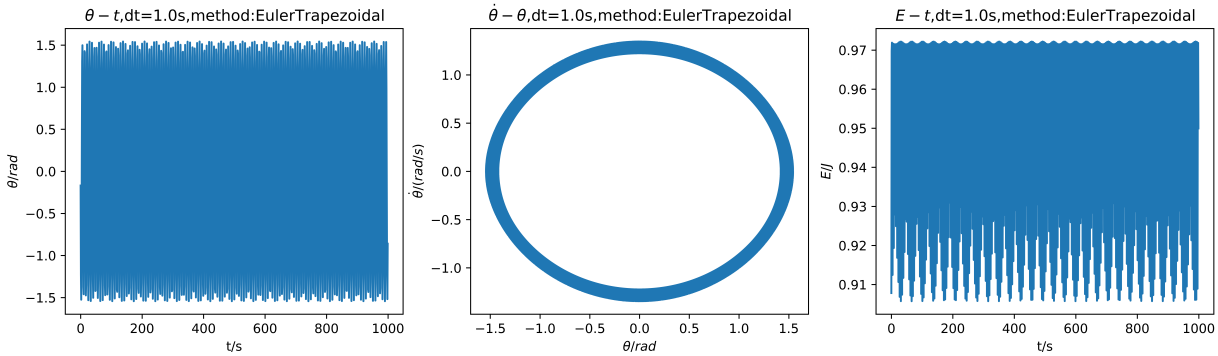

图 6: EulerTrapezoidal method of 0 to 1000s,dt=0.1s

```
(base) PS C:\Users\ASUS\Desktop\计算物理基础\hw7> python -u .\simplePendulum.py
save to output\pendulum_dt=0.1_method=Euler.png
save to output\pendulum_dt=0.1_method=midpoint.png
save to output\pendulum_dt=0.1_method=Rk4.png
save to output\pendulum_dt=0.1_method=EulerTrapezoidal.png
save to output\pendulum_dt=1.0_method=Rk4.png
save to output\pendulum_dt=1.0_method=EulerTrapezoidal.png
```

图 7: 题目 1 程序运行截图

$$C_3 = -1.7532723, \quad C_4 = 0.0834586, \quad Z_{\text{ion}} = 3.$$

2.1 程序描述

本程序通过有限差分法在均匀和非均匀网格上分别求解 H 和 Li 原子的径向薛定谔方程，并比较均匀和非均匀网格的求解效果。

薛定谔方程解可分离变量为 $\Psi(r) = R(r)Y_l^m$，Y_l^m 为球谐函数。径向部分 $R(r)$ 满足方程:

$$\frac{d}{dr}\left(r^2\frac{dR}{dr}\right) - 2r^2\left[V(r) - E\right]R = l(l+1)R.$$

令 $u(r) = R(r)/r$，则方程可化为：

$$-\frac{1}{2}\frac{d^2u}{dr^2} + \left[V + \frac{l(l+1)}{2r^2}\right]u = Eu.$$

只需求解 u 即可。

使用有限差分法，考虑 $\frac{d^2u}{dr^2} = \frac{u_{i+1}-u_i-u_{i-1}}{\Delta r^2}$，和 $u_0 = u_n = 0$ 边界条件，可将其转换为 $Hu = Eu$ 的矩阵本征值问题。其本征值从小到大依次对应角量子数 l, 主量子数 $l+1, l+2, ...$ 的能级。

当采用非均匀网格

$$r_j = r_p\left[\exp(j\delta) - 1\right], \quad j = 0, 1, \ldots, j_{\max}$$

$$r_p = \frac{r_{\max}}{\exp(\delta j_{\max}) - 1}$$

时，做变换 $u_j = u(r_j) = v(j)e^(\delta j/2)$ 方程将变为 (注意:PPT 上的 $\frac{1}{2r_p^2\delta^2e^{2j\delta}}$ 系数有误)

$$\frac{1}{2r_p^2\delta^2e^{2j\delta}}(-\frac{d^2}{dj^2}v(j) + \delta^2v(j)) + V_eff(j) = -Ev(j)$$

$$V_eff(j) = V(r_j) + \frac{l(l+1)}{2r_j^2}$$

同样可以化为矩阵本征值问题求解。本程序中使用 numpy.linalg.eig 求解矩阵本征值问题。

程序源文件为 solveSchrodinger.py，在终端进入当前目录，使用命令 python -u solveSchrodinger.py 运行本程序。运行时请保证 Python 第三方库 Numpy,Matplotlib 已安装。程序开发环境为 Python3.12.3，可在 Python3.8 以上版本中运行。

3 伪代码

均匀网格求解

Algorithm 6 SolveSchrodinger

function SolveSchrodinger($l, n = 1000, rmin = 0, rmax = 100, type = "Li"$)

INPUT: l (angular momentum quantum number), n (number of points), $rmin$ (minimum radius), $rmax$ (maximum radius), $type$ (type of atom)

OUTPUT: El (eigenvalues), r (radius array), vl (eigenvectors)

$H \leftarrow$ zero matrix of size (n, n)

$dr \leftarrow \frac{rmax - rmin}{n}$

$r \leftarrow \text{linspace}(dr, rmax, n, \text{endpoint} = \text{False})$

for $i \leftarrow 0$ **To** $n - 1$ **do**

$H[i, i] \leftarrow 1 + \text{Veff}(r[i], l, type) \times dr^2$

if $i < n - 1$ **then**

$H[i, i + 1] \leftarrow -\frac{1}{2}$

$H[i + 1, i] \leftarrow -\frac{1}{2}$

end if

end for

$re \leftarrow \text{linalg.eigh}(H)$

$El \leftarrow re[0]/dr^2$

$sortedIndices \leftarrow \text{argsort}(El)$

$El \leftarrow El[sortedIndices]$

$vl \leftarrow (re[1])[:, sortedIndices]$

return El, r, vl

end function

非均匀网格求解

Algorithm 7 SolveSchrodingerNonuniform

function SolveSchrodingerNonuniform($l, jmax = 1000, rmin = 0, rmax = 100, delta = 0.01, type = "Li"$)

INPUT: l (angular momentum quantum number), $jmax$ (number of points), $rmax$ (maximum radius), $delta$ (scaling factor), $type$ (type of atom)

OUTPUT: El (eigenvalues), r (radius array), vl (eigenvectors)

$H \leftarrow$ zero matrix of size $(jmax - 1, jmax - 1)$

$rp \leftarrow \frac{rmax}{\exp(delta \times jmax) - 1}$

function A(j)

return $2rp^2 delta^2 \exp(2jdelta)$

end function

for $j \leftarrow 1$ **To** $jmax - 1$ **do**

$rj \leftarrow rp(\exp(deltaj) - 1)$

$H[j - 1, j - 1] \leftarrow \frac{2 + delta^2/4}{a(j)} + \text{Veff}(rj, l, type)$

if $j < jmax - 1$ **then**

$H[j-1,j] \leftarrow -\frac{1}{a(j)}$
$H[j,j-1] \leftarrow -\frac{1}{a(j+1)}$
end if
end for
$re \leftarrow \text{linalg.eig}(H)$
$El \leftarrow re[0]$
$sortedIndices \leftarrow \text{argsort}(El)$
$El \leftarrow El[sortedIndices]$
$s \leftarrow (\exp(delta \times \text{arange}(1, jmax)/2)).\text{reshape}(-1, 1)$
$vl \leftarrow (re[1])[:, sortedIndices] \times s$
$r \leftarrow \text{array}([rp(\exp(deltaj) - 1) \text{ for } j \text{ in range}(1, jmax)])$
return El, r, vl
end function

4 输入输出实例

运行本程序后，将在控制台输出由 1000 个格点采用均匀和非均匀网格求解的 H 和 Li 的最低的三个能级能量（其中 H 存在简并）。并生成由非均匀网格求解的 H 和 Li 的 $n = 1, 2, 3, 4$ 不同 l 的 $u(r) = rR(r)$ 的图像如图10和11，为"wavefunctionsH.png" 和"wavefunctionsLi.png" 于当前路径下。此外还将输出 Li 的势能曲线和网格间距变化图(图9,12) 为"potentialLi.png" 和"ununiformgrid.png" 于当前路径下。最后，我们比较了 $n = 2, l = 1$ 能级非均匀网格和均匀网格的收敛速率，如图13，输出为"compare.png" 于当前路径下。为保持格点间距变化速度一致，采用 $\delta = 5/jmax$ 可以看到相同格点数量下，非均匀网格求解的收敛速率和精度都要高于均匀网格，但是非均匀网格的 H 矩阵是非对称的，也将有更高的计算成本。

最终得到 H 前三个能级为:

$$E_1 = 13.6057eV, n = 1, l = 0$$

$$E_2 = -3.4014eV, n = 2, l = 0; n = 2, l = 1$$

$$E_3 = -1.5117eV, n = 3, l = 0; n = 3, l = 1; n = 3, l = 2$$

Li 前三个能级为:

$$E_1 = -121.3165eV, n = 1, l = 0$$

$$E_2 = -30.5389eV, n = 2, l = 1$$

$$E_3 = -30.3529eV, n = 2, l = 0$$

程序运行截图如图8。

```
(base) PS C:\Users\ASUS\Desktop\计算物理基础\hw7> python -u .\solveSchrodinger.py

############
solve schrodinger of Li and H by uniform grid:n=1000,rmax=50

first 3 energy of H:
n=1, l=0, E=-13.621823981257936eV
n=2, l=0, E=-3.4073781651670654eV
n=2, l=1, E=-3.4038685792235626eV
n=3, l=0, E=-1.5145698718004603eV
n=3, l=1, E=-1.5135062055288826eV
n=3, l=2, E=-1.512355453146946eV

first 3 energy of Li:
n=1, l=0, E=-121.94165994797527eV
n=2, l=1, E=-30.57438714656927eV
n=2, l=0, E=-30.487958158651438eV

############
solve schrodinger of Li and H by ununiform grid:jmax=1000,delta=0.005,rmax=50

first 3 energy of H:
n=1, l=0, E=-13.605727237378463eV
n=2, l=0, E=-3.401444164471444eV
n=2, l=1, E=-3.401429491387497eV
n=3, l=0, E=-1.5117607572275873eV
n=3, l=1, E=-1.5117529661170506eV
n=3, l=2, E=-1.5117460852821827eV

first 3 energy of Li:
n=1, l=0, E=-121.31649263639444eV
n=2, l=1, E=-30.538852636490606eV
n=2, l=0, E=-30.352871397894855eV
```

图 8: 题目 2 程序运行截图

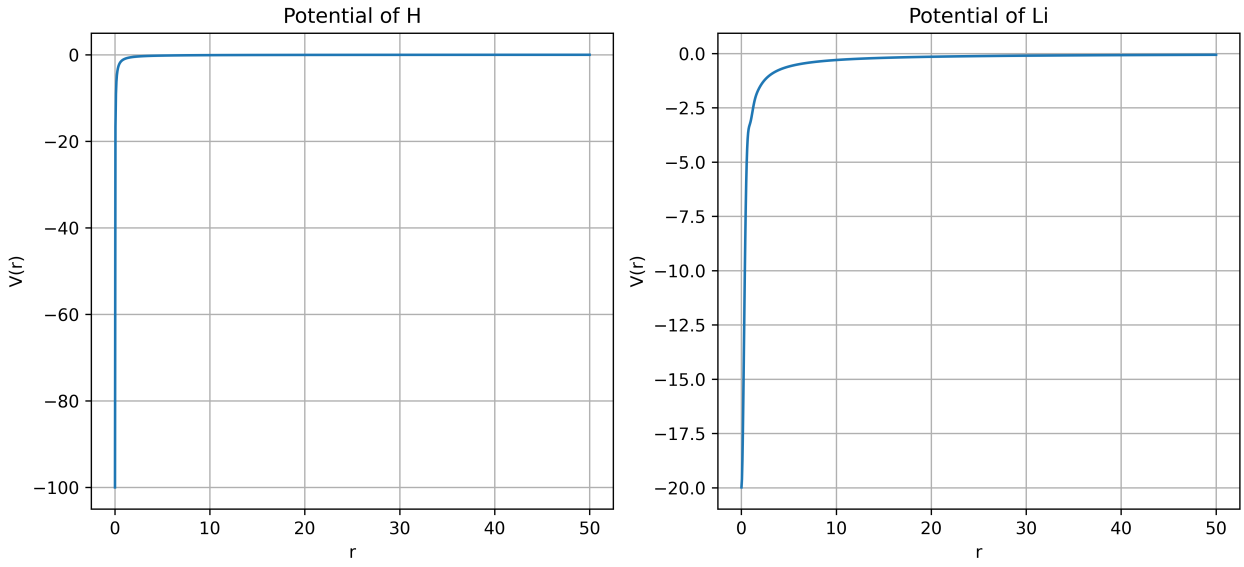

图 9: H 和 Li 原子势能曲线

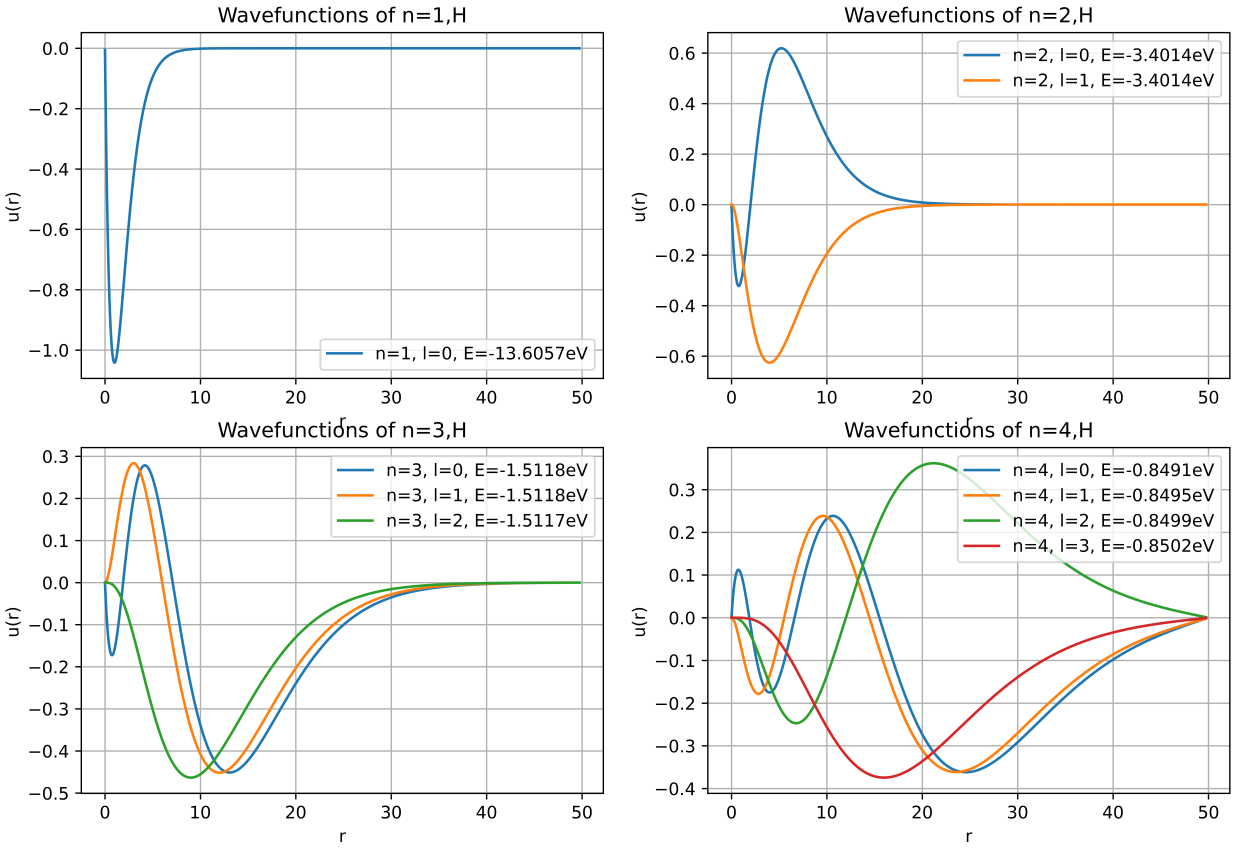

图 10: H 原子不同能级波函数 (已归一化)

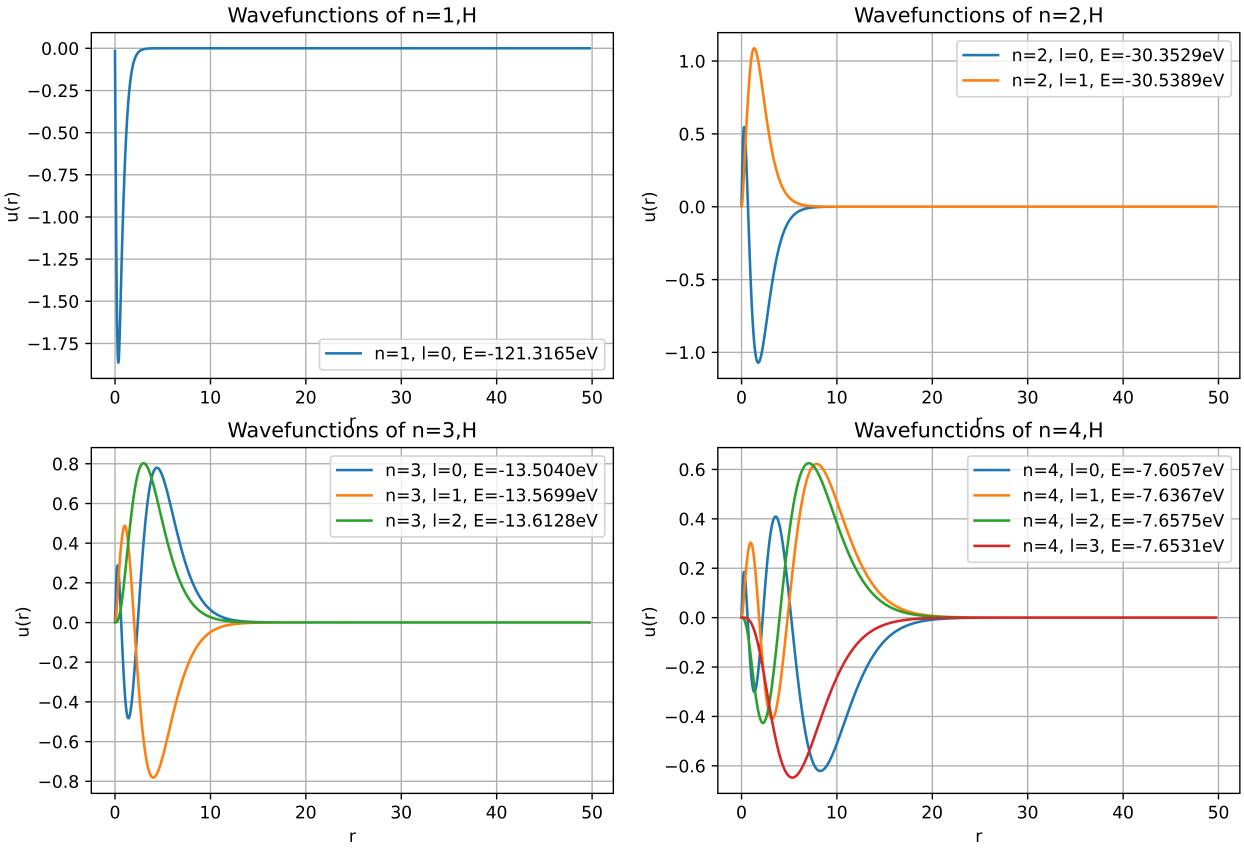

图 11: Li 原子不同能级波函数 (已归一化)

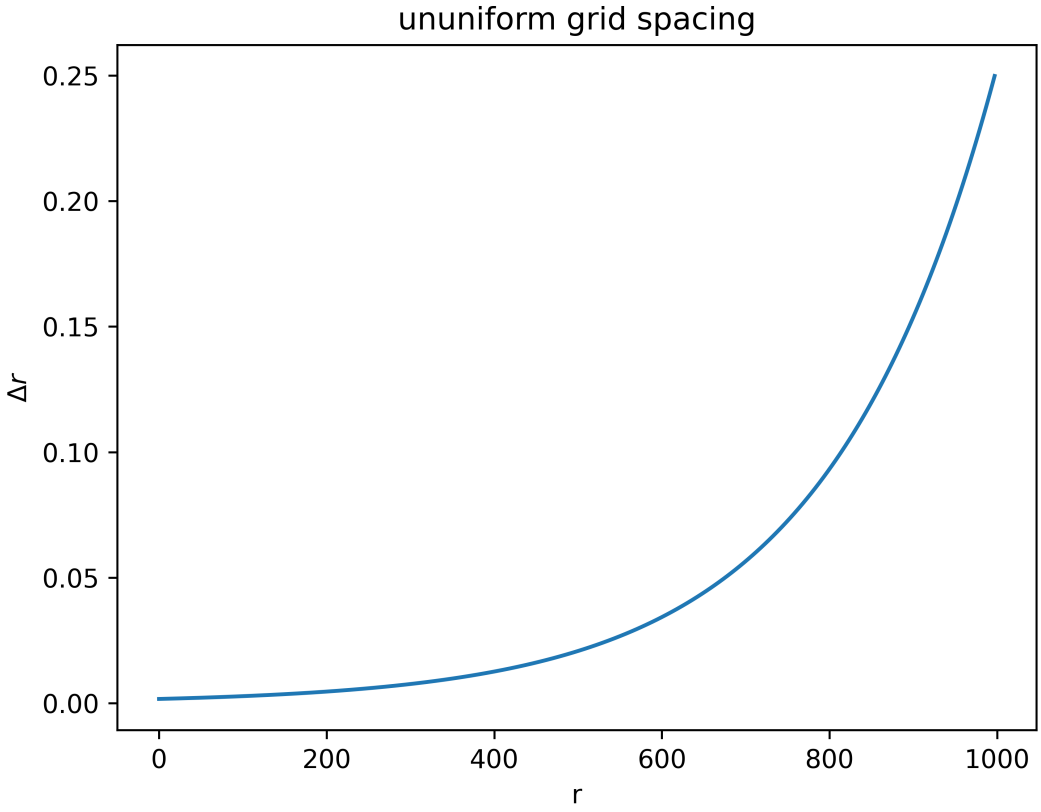

图 12: 非均匀网格间距变化

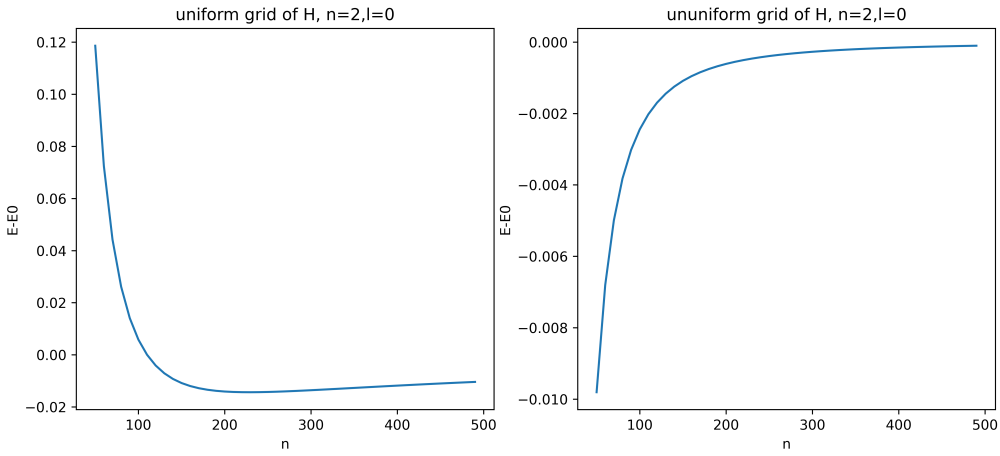

图 13: 均匀网格和非均匀网格收敛速率比较