# 第五次作业

尹朝阳 物理学系

# 1 题目 1

## 1.1 题目描述

Compute the derivative of $f(x) = \sin(x)$ at $x = \frac{\pi}{3}$ using the Richardson extrapolation algorithm. Start with $h = 1$ and find the number of rows in the Richardson table required to estimate the derivative with six significant decimal digits. Output the Richardson table.

## 1.2 程序描述

本题目要求用 Richardson 外推法求 $f(x) = \sin(x)$ 在 $x = \frac{\pi}{3}$ 处的导数。

Richardson 外推法的实现思路如下：

首先构造关于 $h$ 的函数 $\phi(h) = \frac{1}{2h}[f(x+h) - f(x-h)]$，其中 $x$ 为所求的点的横坐标值，本题中 $x = \frac{\pi}{3}$。然后求解 $n \times n$ 的系数矩阵 $D$，其中 $D_{nn}$ 即为要求的导数值。

矩阵 $D$ 的初值条件为 $D_{n0} = \phi(\frac{h}{2^n})$，矩阵元满足递推关系：

$$D_{nm} = D_{n,m-1} + \frac{1}{4^m - 1} * (D_{n,m-1} - D_{n-1,m-1}) \tag{1}$$

本题中，导数值的计算精度采用 $|D_{nn} - D_{n-1,n-1}|$ 进行估计。

本题的源程序为 CalcDerivative_RichardsonExtrapolation/Compute_Derivative_with_Richardson_Extrapolation.py，导入的第三方库有 numpy。

## 1.3 伪代码

首先实现 Richardson extrapolation algorithm。

**Algorithm 1.1** Richardson extrapolation

**Input :** $x_0$, $h$ and the number of rows in the Richardson table $n$

**Output :** the Richardson Table

```
for i ← 1 to n do
    R_{i0} ← 1/(2 * h/2^i) * [f(x_0 + h/2^i) − f(x_0 − h/2^i]        ▷ R_{i0} = φ(h/2^i)
end for
for i ← 2 to n do
    for j ← 2 to i do
        R_{ij} = R_{i,j−1} + 1/(4^j − 1) * (R_{i,j−1} − R_{i−1,j−1})
    end for
end for
return the Richardson table R
```

然后实现精度估计。

**Algorithm 1.2** estimate the derivative with six significant decimal digits

**Output :** the smallest number of rows $n$ to reach the precision, and the corresponding $R$

```
n ← 1
R ← RICHARDSON(n, x_o, h)
if |R_11 − f'(x_0)| < ε then
    return n = 1, f'(x_0) = R_11
else
    n ← n + 1
    R ← RICHARDSON(n, x_o, h)
    while not |R_{n−1,n−1} − R_nn| < ε do
        n ← n + 1
        R ← RICHARDSON(n, x_o, h)
    end while
end if
return n, f'(x_0) = R_nn
```

### 1.4 测试用例

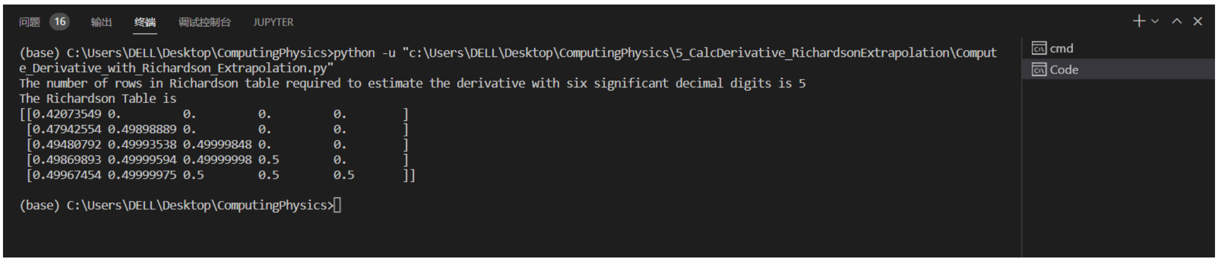



图 1: Compute Derivative with Richardson Extrapolation

求得导数值为 0.5，矩阵的行数最小为 5 能够实现 $|D_{nn} - D_{n-1,n-1}| < 1 \times 10^{-6}$，即保证 6 位小数的精度。

# 2 题目 2

### 2.1 题目描述

Radial wave function of the 3s orbital is:

$$R_{3s}(r) = \frac{1}{9\sqrt{3}} \times (6 - 6\rho + \rho^2) \times Z^{\frac{3}{2}} \times e^{-\frac{\rho}{2}}$$

- $r$ = radius expressed in atomic units (1 Bohr radius = $52.9pm$)
- $e = 2.71828$ approximately

- $Z$ = effective nuclear charge for that orbital in that atom.
- $\rho = \frac{2Zr}{n}$ where $n$ is the principal quantum number (3 for the $3s$ orbital)

Compute $\int_0^{40} |R_{3s}|^2 r^2 dr$ for Si atom ($Z = 14$) with Simpson's rule using two different radial grids:

(1) Equal spacing grids: $r[i] = (i-1)h; i = 1, ..., N$ (try different $N$)

(2) A nonuniform integration grid, more finely spaced at small r than at large $r$: $r[i] = r_0(e^{t[i]} - 1); t[i] = (i-1)h; i = 1, ..., N$ (One typically choose $r_0 = 0.0005 a.u.$, try different $N$).

(3)Find out which one is more efficient, and discuss the reason.

### 2.2 程序描述

本题要求用 Simpson's rule 计算积分。

Simpson's rule 的实现思路如下：

对于每一个小区间 $[a, b]$，取两个端点 $a$、$b$ 以及区间中点 $\frac{a+b}{2}$ 对应的函数值，认为函数 $f(x)$ 在该小区间的数值积分为 $\int_a^b f(x)dx = \frac{b-a}{2\times 3} * [f(a) + 4 * f(\frac{a+b}{2}) + f(b)]$。

当全积分限被分为偶数个区间时，记积分下限为 $x_1$，积分上限为 $x_n$，则

$$\int_{x_1}^{x_n} f(x)dx = \sum_{i=1,3,5}^{n-2} \frac{x_{i+1} - x_i}{3} * [f(x_i) + 4 * f(x_{i+1}) + f(x_{i+2})] \tag{2}$$

本题中，第 (1) 问的取点划分方式为线性等距划分，第 (2) 问的取点划分方式为 e 指数函数形式划分。对这两种取点方式，本程序采用对每个小区间进行 Simpson's rule 的方式计算积分，然后求和计算出最终结果。

本题的源程序在 CalcIntegration_SimpsonRule 中，其中 question1.py 实现第 (1) 问中的划分方式进行积分，question2.py 实现第 (2) 问中的划分方式进行积分，question3.py 对这两种方式进行评估。程序导入的第三方库有 numpy 和 matplotlib，编码方式为 utf-8。

### 2.3 伪代码

**Algorithm 2.1** question1

**Output :** the Integral $I$

$f(r) \leftarrow (R_{3s}(r) * r)^2$
**function** SimpsonRule($a$, $b$, $N$) ▷ $N$ as the number of points
  $h \leftarrow (b-a)/(N-1)$
  **for** $i \leftarrow 1$ to $N$ **do**
    $r_i \leftarrow (i-1) * h$
  **end for**
  $I \leftarrow \mathrm{sum}((r[2:N{+}1]-r[1:N])/6*(f(r[1:N])+f(r[2:N{+}1])+4*f((r[1:N]+r[2:N{+}1])/2)))$
  **return** $I$
**end function**
**for** $N \leftarrow 3$ to 2999 step 2 **do** ▷ try different $N$
  $I \leftarrow$ SIMPSONRULE($a$, $b$, $N$)
**end for**
**Plot** the function $I(N)$ as $N$ changes

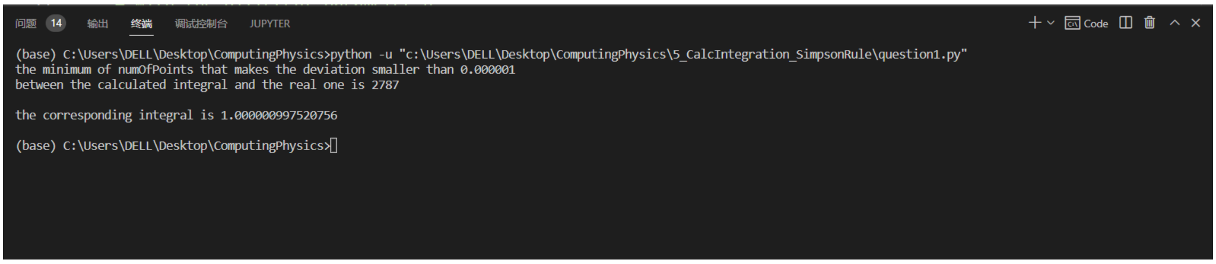
**Algorithm 2.2** question2

**Output :** the Integral $I$

$f(r) \leftarrow (R_{3s}(r) * r)^2$
**function** SimpsonRule($a$, $b$, $N$) ▷ $N$ as the number of points
  $h \leftarrow \log((b-a)/r_0+1)/(N-1)$
  **for** $i \leftarrow 1$ to $N$ **do**
    $t_i \leftarrow (i-1) * h$
    $r_i \leftarrow r_0 * (e^{t_i}-1)$
  **end for**
  $I \leftarrow \mathrm{sum}((r[2:N{+}1]-r[1:N])/6*(f(r[1:N])+f(r[2:N{+}1])+4*f((r[1:N]+r[2:N{+}1])/2)))$
  **return** $I$
**end function**
**for** $N \leftarrow 3$ to 999 step 2 **do** ▷ try different $N$
  $I \leftarrow$ SIMPSONRULE($a$, $b$, $N$)
**end for**
**Plot** the function $I(N)$ as $N$ changes

## 2.4 测试用例

第 (1) 问:



```
问题 14  输出  终端  调试控制台  JUPYTER
(base) C:\Users\DELL\Desktop\ComputingPhysics>python -u "c:\Users\DELL\Desktop\ComputingPhysics\5_CalcIntegration_SimpsonRule\question1.py"
the minimum of numOfPoints that makes the deviation smaller than 0.000001
between the calculated integral and the real one is 2787

the corresponding integral is 1.000000997520756

(base) C:\Users\DELL\Desktop\ComputingPhysics>
```

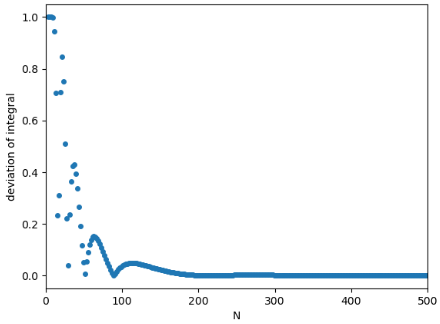
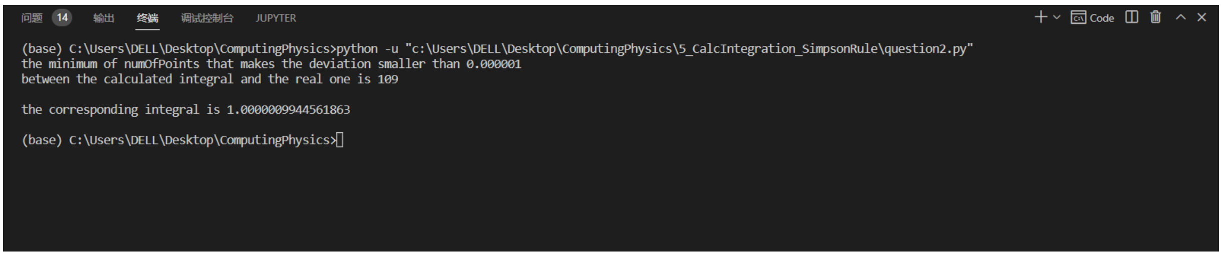



图 2: question1

计算得最终的积分值为 1。

随着取点个数 $n$ 的增大，数值积分与真实积分值的差距呈减小的趋势，当 $n$ 增大到 2787 时两者之差已经小于 $1 \times 10^{-6}$。

第 (2) 问：



```
问题 14  输出  终端  调试控制台  JUPYTER
(base) C:\Users\DELL\Desktop\ComputingPhysics>python -u "c:\Users\DELL\Desktop\ComputingPhysics\5_CalcIntegration_SimpsonRule\question2.py"
the minimum of numOfPoints that makes the deviation smaller than 0.000001
between the calculated integral and the real one is 109

the corresponding integral is 1.0000009944561863

(base) C:\Users\DELL\Desktop\ComputingPhysics>
```

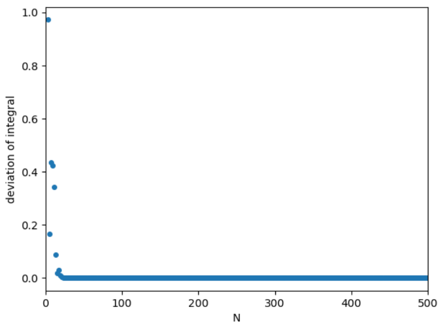



图 3: question2

计算得最终的积分值为 1。

随着取点个数 $n$ 的增大，数值积分与真实积分值的差距呈减小的趋势，当 $n$ 增大到 109 时两者之差已经小于 $1 \times 10^{-6}$。

第 (3) 问：

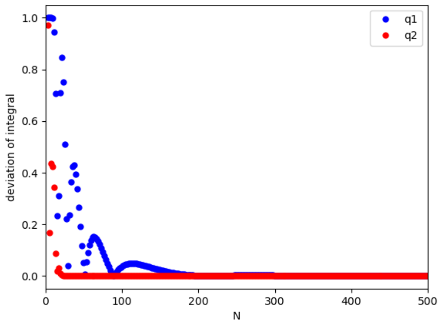



图 4: question3 – comparison

从图 4 中可以看出，第二种取点方式显然更有效，收敛得更快。这从图 2、3 中也可以看出，第一种取点方式需要取 2787 个点（即 2786 个区间）才能与第二种取点方式取 109 个点（即 108 个区间）达到同样的积分精度，这也说明了同样的结论。

分析原因如下：

首先画出被积函数 $f(r) = |R_{3s}|^2 r^2$ 如图 5 所示：

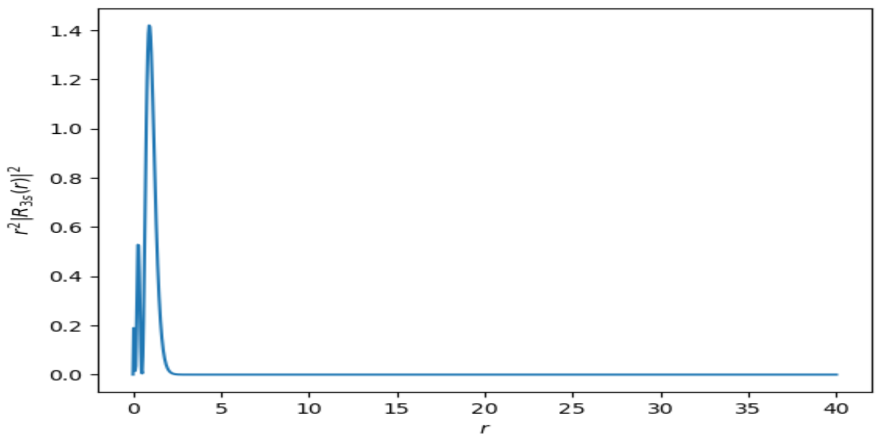



图 5: function to be integrated

可以发现，被积函数在前端呈现出很大的振荡，而 $x > 3$ 后基本为 0。

由于 Simspon's rule 在一个小区间取 3 个点，其对于三次函数不会产生积分误差，可以用拟合的三次函数在小区间上的积分值代表真实积分值。在当 $x$ 较小时，用三次函数拟合显然效果较差，为了数值积分接近真实积分值，需要点取得密一些。而当 $x$ 很大时，函数近似为 0，此时点取得较为稀疏不会影响三次插值的精确程度，则积分的精度也能保证。

综上，对于该被积函数，需要在 $x$ 较小时点取得较密，而 $x$ 较大时点取得较稀疏。

相较于第 (1) 问的线性函数取值，第 (2) 问的 e 指数函数划分更加符合上面的取值特征，故具有更快的收敛速度，即效率更高。