1. Richardson extrapolation 方法求解$f(x) = \sin(x)$在$x = \frac{\pi}{3}$时的导数，最终要输出一个 Richardson table，如果迭代n次，Richardson table 就是一个$n \times n$的矩阵。所以定义$Richardson(h, n, x)$函数，在函数内部，先初始化一个$n \times n$矩阵mat，然后用 for 循环将mat矩阵的第一列填上$D(0,0), D(1,0), \dots D(n-1,0)$。其中$D(i,0) = \varphi\left(\frac{h}{2^i}\right) = \frac{1}{2 \times \frac{h}{2^i}}\left[f\left(x + \frac{h}{2^i}\right) - f\left(x - \frac{h}{2^i}\right)\right]$。

现在mat矩阵即为$\begin{pmatrix} D(0,0) & 0 & \dots & 0 \\ D(1,0) & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ D(n-1,0) & 0 & \dots & 0 \end{pmatrix}$。接下去用 for 循环处理余下的$(n-1)$列，在每一个 for 循环内部再用一个 for 循环处理每一列中的元素，对于第 j 循环，要处理$(n-j)$个元素。$D(k,j) = D(k,j-1) + \frac{1}{4^j - 1}[D(k,j-1) - D(k-1,j-1)]$。

返回mat矩阵，即$\begin{pmatrix} D(0,0) & 0 & 0 & \dots & 0 \\ D(1,0) & D(1,1) & 0 & \dots & 0 \\ D(2,0) & D(2,1) & D(2,2) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ D(n-1,0) & D(n-1,1) & D(n-1,2) & \dots & D(n-1,n-1) \end{pmatrix}$。

由于在$Richardson(h, n, x)$函数中调用的是函数$f(x)$，所以在函数外部修改$f(x)$的定义，就可以实现对不同的函数求导；若增加迭代次数n，就可以提高结果的精确度。

Input: a function $f(x)$, stepsize h, iteration times n, a particular x

Output: derivative of the function $f'(x)$

1. **Function** $Richardson(h, n, x)$
2. $mat \leftarrow \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$ /* $n \times n$ matrix */
3. **For** $i \leftarrow 0$ to $(\mathrm{n} - 1)$ **Do**
4. $\quad mat[i, 0] \leftarrow \frac{1}{2h}[f(x+h) - f(x-h)]$
5. $\quad h \leftarrow \mathrm{h}/2$
6. **For** $j \leftarrow 1$ to $(\mathrm{n} - 1)$ **Do**

7. **For** $k \leftarrow$ j to (n − 1) **Do**

8. $mat[k,j] \leftarrow mat[k,j-1] + \frac{1}{4^{j}-1}(mat[k,j-1] - mat[k-1,j-1])$

9. **Return** mat

10. **End Function**

源代码：

```python
import numpy as np

def Richardson(h, n, x):
# Richardson extrapolation to approximate f'(x)
# h: initial stepsize
# n: max iteration value
# x: the value to find derivative at
    mat = np.zeros([n, n])

    for i in range(n):
        mat[i,0] = (f(x+h) - f(x-h)) / (2*h)
        h = h/2

    for j in range(1,n):
        for k in range(j,n):
            mat[k,j] = mat[k,j-1] + (1 / (4**j - 1)) * (mat[k,j-1] - mat[k-1,j-1])

    return mat

def f(x):
    return np.sin(x)

print(Richardson(1, 4, (np.pi/3)))
```

运行结果：

```
PS C:\Users\11765\Desktop\学习\物理\计算物理\作业\homework5>
gpy\launcher' '61592' '--' 'c:\Users\11765\Desktop\学习\物理\
[[0.42073549 0.         0.         0.        ]
 [0.47942554 0.49898889 0.         0.        ]
 [0.49480792 0.49993538 0.49999848 0.        ]
 [0.49869893 0.49999594 0.49999998 0.5       ]]
```

2. 使用 Simpson's Rule 计算单摆的周期。由于周期$T(\varphi_0)$可由下式计算：

$$T(\varphi_0)=4\sqrt{\frac{l}{2g}}\int_0^{\varphi_0}\frac{d\varphi}{\sqrt{cos\varphi-cos\varphi_0}}$$

作变量替换：$\sin\left(\frac{\varphi}{2}\right)=\sin\left(\frac{\varphi_0}{2}\right)sin\theta$，则：$d\varphi=\frac{2cos\theta\sin\left(\frac{\varphi_0}{2}\right)}{\sqrt{1-(sin\theta)^2\sin\left(\frac{\varphi_0}{2}\right)^2}}d\theta$

$$T(\varphi_0)=4\sqrt{\frac{l}{2g}}\int_0^{\frac{\pi}{2}}\frac{1}{\sqrt{cos\varphi-cos\varphi_0}}\frac{2cos\theta\sin\left(\frac{\varphi_0}{2}\right)}{\sqrt{1-(sin\theta)^2\sin\left(\frac{\varphi_0}{2}\right)^2}}d\theta$$

$$=4\sqrt{\frac{l}{g}}\int_0^{\frac{\pi}{2}}\frac{1}{\sqrt{1-(sin\theta)^2\sin\left(\frac{\varphi_0}{2}\right)^2}}d\theta$$

这样就可以通过 Simpson's Rule 来数值计算积分。

定义$Simpson(x_1,x_2,n)$函数来计算积分，x_1,x_2分别为积分上下限，n为 Simpson's Rule 计算时的区间数，n必须为偶数，且n越大积分结果越精确。

用 NumPy 的 linspace 函数在区间$[x_1,x_2]$上取$(n+1)$个点，即n段等间隔的区间，将这$(n+1)$个点的值存入列表$interval$，每段区间长为$step=\frac{x_2-x_1}{n}$，积分结果$integrate$是累加出来的，初始设为 0。用 for 循环每次取三个点即两段区间，计算出这两段区间上积分的近似结果$result=\frac{step}{3}[f(x_i)+4f(x_{i+1})+f(x_{i+2})]$，将结果$result$累加至$integrate$即可得到结果。最终将$integrate$返回。

$f(x)$定义了被积函数，即：

$$f(x)=\frac{1}{\sqrt{1-(sinx)^2\sin\left(\frac{\phi}{2}\right)^2}}$$

注意，其中ϕ是单摆的振幅φ_0，是一个未知的常数。在最后计算周期的$T(x)$函数中，我们要实现根据不同的ϕ，计算不同的$T(\phi)$。所以我们把ϕ设为一个全局变量，它在$T(x)$函数中输入，在$f(x)$函数中被调用。

最终的$T(x)$函数可以计算不同振幅下的单摆周期，结果$result=4\times Simpson\left(0,\frac{\pi}{2},100\right)\approx 4\times\int_0^{\frac{\pi}{2}}\frac{1}{\sqrt{1-(sin\theta)^2\sin\left(\frac{\phi}{2}\right)^2}}d\theta$，系数$\sqrt{\frac{l}{g}}$并未考虑在内，所以周期应该等于计算得到的常数再乘以$\sqrt{\frac{l}{g}}$。

Input: lower bound x_1, upper bound x_2, number of intervals n

Output: integration of the function $\int_{x_1}^{x_2} f(x)dx$

1. **Function** $Simpson(x_1, x_2, n)$
2. $step \leftarrow \frac{x_2 - x_1}{n}$, $integrate \leftarrow 0$
3. $interval \leftarrow [x_1, (x_1 + step), (x_1 + 2step), \ldots, x_2]$
4. **For** $i \leftarrow 0,2,4, \ldots, (\mathrm{n} - 2)$ **Do**
5. $\quad result \leftarrow \frac{step}{3}[f(interval[i]) + 4f(interval[i+1]) + f(interval[i+2])]$
6. $\quad integrate \leftarrow integrate + result$
7. **Return** $integrate$
8. **End Function**

源代码：

```python
import numpy as np
import sympy as sp
from sympy.abc import t

def Simpson(x1, x2, n):
# Simpson's rule to calculate integration of f(x)
# x1: lower bound of integration
# x2: upper bound of integration
# n: number of intervals, which must be even
    interval = np.linspace(x1, x2, num=n+1)
    step = (x2-x1) / n
    integrate = 0

    for i in range(0,n-1,2):
        result = (step/3) * (f(interval[i]) + 4*f(interval[i+1]) + f(interval[i+2]))
        integrate += result

    return integrate

def f(x):
# Define the function which needs to be integrated.
    global Φ
    temp1 = (sp.sin(x)) ** 2
    temp2 = (sp.sin(Φ/2)) ** 2
    result = 1 / sp.sqrt(1 - temp1*temp2)
    return result

def T(x):
# Calculate the oscillation period T of a pendulum
# Initial amplitude is x, and initial angular velocity is 0.
    global Φ
    Φ = x
    result = 4 * Simpson(0, np.pi/2, 100)
    return result

print('The period T is: %f*sqrt(l/g)' % T(np.pi/3))
```

源代码测试了初始振幅$\varphi_0 = \frac{\pi}{3}$ 时的周期T，得到的结果：$T = 6.743 \times \sqrt{\frac{l}{g}}$

```
PS C:\Users\11765\Desktop\学习\物理\计算物理\作业\homework5>
gpy\launcher' '61822' '--' 'c:\Users\11765\Desktop\学习\物理
The period T is: 6.743001*sqrt(l/g)
PS C:\Users\11765\Desktop\学习\物理\计算物理\作业\homework5>
```

用 Mathematica 验证：

```
In[4]:= ϕ0 = Pi / 3
              圆周率
        Integrate[4 / (Sqrt[2] * Sqrt[Cos[ϕ] - Cos[ϕ0]]), {ϕ, 0, ϕ0}]
        积分              平方根      …     余弦        余弦
```

Out[4]= $\frac{\pi}{3}$

Out[5]= 4 EllipticK$\left[\frac{1}{4}\right]$

In[6]:= N$\left[4\ \text{EllipticK}\left[\frac{1}{4}\right]\right]$

(… 第一类完全椭圆积分)

Out[6]= 6.743