内容简介

编写一个子程序，用来执行定义在任意区间 $[a, b]$ 上的函数 f 的 Romberg 算法。用户指定阵列中所计算的行数，并且当计算完成后要看到整个阵列，编写一个主程序并且用下列积分测试你的 Romberg 子程序：

- $\int_0^1 \frac{\sin x}{x} dx$
- $\int_{-1}^1 \frac{\cos x - e^x}{\sin x} dx$
- $\int_1^\infty \frac{1}{xe^x} dx$

编写这些积分的程序，要避免由于减法而产生有效数字的严重丢失。用等式 $f(x_0) = \lim_{x \to x_0} f(x)$ 定义任何可疑点 x_0 上的函数 f 以保证连续性。对于第三个积分，作适当的积分变换，例如 $x = \frac{1}{t}$。

计算出 Romberg 阵列中的 7 列，打印出每一种情形的阵列，并要求打印格式能反映出收敛性。

工作环境

主要程序语言：**python**
软件：**JupyterLab**
使用的包：**numpy, scipy.integrate**

输出结果

```
Integrate[sin(x) / x, {x, 0, 1}] = 0.946083070367
Error of function scipy.integrate.quad 0.000000000000
Romberg_Integrate[sin(x) / x, {x, 0, 1}] ---- M = 6
n |     R(n, 0)         R(n, 1)         R(n, 2)         R(n, 3)         R(n, 4)
0 | 0.920735492404
1 | 0.939793284806  0.946145882274
2 | 0.944513521665  0.946086933952  0.946083004064
3 | 0.945690863583  0.946083310888  0.946083069351  0.946083070387
4 | 0.945985029934  0.946083085385  0.946083070351  0.946083070367  0.946083070367
5 | 0.946058560963  0.946083071306  0.946083070367  0.946083070367  0.946083070367
6 | 0.946076943060  0.946083070426  0.946083070367  0.946083070367  0.946083070367

        R(n, 5)         R(n, 6)
5 | 0.946083070367
6 | 0.946083070367  0.946083070367
--------------------------------------------------------------------------------
```

```
Integrate[(cos(x) - e**x) / sin(x), {x, -1, 1}] = -2.246591720729
Error of function scipy.integrate.quad 0.000000000000
Romberg_Integrate[(cos(x) - e**x) / sin(x), {x, -1, 1}] ---- M = 6
n |     R(n, 0)         R(n, 1)         R(n, 2)         R(n, 3)         R(n, 4)
0 | -2.793206693662
1 | -2.396603346831 -2.264402231221
2 | -2.285217708408 -2.248089162267 -2.247001624336
3 | -2.256325874192 -2.246695262787 -2.246602336155 -2.246595998247
4 | -2.249030253178 -2.246598379506 -2.246591920621 -2.246591755295 -2.246591738656
5 | -2.247201668292 -2.246592139997 -2.246591724029 -2.246591720909 -2.246591720774
6 | -2.246744227309 -2.246591746982 -2.246591720781 -2.246591720729 -2.246591720729

         R(n, 5)         R(n, 6)
5 | -2.246591720757
6 | -2.246591720729 -2.246591720729
--------------------------------------------------------------------------------
Integrate[1 / (x * e**x), {x, 0, Infinity}]
                     = Integrate[e**(-1/t) / t, {x, 0, 1}] = 0.219383934395
Error of function scipy.integrate.quad 0.000000001573
Romberg_Integrate[1 / (x * e**x), {x, 1, Infinity}]
                     = Romberg_Integrate[e**(-1/t) / t, {x, 0, 1}] ---- M = 6
n |     R(n, 0)         R(n, 1)         R(n, 2)         R(n, 3)         R(n, 4)
0 | 0.183939720586
1 | 0.227305143529  0.241760284511
2 | 0.219833923359  0.217343516635  0.215715732110
3 | 0.219350957932  0.219189969456  0.219313066310  0.219370166853
4 | 0.219383579753  0.219394453693  0.219408085976  0.219409594224  0.219409748842
5 | 0.219383932406  0.219384049957  0.219383356374  0.219382963841  0.219382859408
6 | 0.219383934273  0.219383934896  0.219383927225  0.219383936286  0.219383940100

         R(n, 5)         R(n, 6)
5 | 0.219382833123
6 | 0.219383941156  0.219383941427

--------------------------------------------------------------------------------
```

分析

一、避免减法导致精度丢失

教材 [1] 中使用的递推公式为：

$$R(n,\, m) = R(n,\, m-1) + \frac{1}{4^m - 1}\left[R(n,\, m-1) - R(n-1,\, m-1)\right] \tag{1}$$

涉及减法 $R(n,\, m-1) - R(n-1,\, m-1)$，该过程由于相减两数实际上非常接近，将严重丢失有效数字。如果采用公式：

$$R(n,\, m) = \frac{4^m}{4^m - 1}R(n,\, m-1) - \frac{1}{4^m - 1}R(n-1,\, m-1) \tag{2}$$

精度损失将会减小。

二、关于积分换元

$$\int_0^\infty \frac{1}{xe^x}dx \overset{x=\frac{1}{t}}{=\!=\!=} \int_0^1 \frac{1}{\frac{1}{t}e^{\frac{1}{t}}}\frac{1}{t^2}dt = \int_0^1 \frac{1}{te^{\frac{1}{t}}}dt \tag{3}$$

计算此积分即可。

三、关于收敛性

不妨视使用 scipy.integrate.quad 计算出的数值积分值为准确值。列出相应的 Romberg 积分误差阵列（此处仅列出第一个积分误差阵列）：

```
Error of Romberg_Integrate[sin(x) / x, {x, 0, 1}]
n |    R(n, 0)       R(n, 1)       R(n, 2)       R(n, 3)       R(n, 4)
0 | 2.534758e-02
1 | 6.289786e-03  6.281191e-05
2 | 1.569549e-03  3.863585e-06  6.630351e-08
3 | 3.922068e-04  2.405213e-07  1.016266e-09  2.003930e-11
4 | 9.804043e-05  1.501776e-08  1.580380e-11  7.660539e-14  1.665335e-15
5 | 2.450940e-05  9.383788e-10  2.468026e-13  1.110223e-16  2.220446e-16
6 | 6.127307e-06  5.864476e-11  4.218847e-15  3.330669e-16  3.330669e-16

       R(n, 5)       R(n, 6)
5 | 2.220446e-16
6 | 3.330669e-16  3.330669e-16
```

对于第一列，$\dfrac{Error\_R(n+1,\, 0)}{Error\_R(n,\, 0)} \approx \dfrac{1}{10}$，可以认为它是收敛的。对于其它列也有类似的现象发生。但误差最终达到了 numpy.float64 的精度极限，几何级数速度收敛的现象失效。

第二、三个 Romberg 积分误差阵列通过运行程序即可得到，这里不再列出。值得注意的是第三个积分相比前两个数值性态较差，误差的数量级比前两者要高出很多。对于此积分，mathematica 给出的结果为

```
In = NumberForm[NIntegrate[1/(x E^x), {x, 1, Infinity}], 12]
Out = 0.219383934398
```

与 scipy.integrate.quad 的计算结果非常相近。但 scipy.integrate.quad 给出的估计误差却较大。

参考资料

[1] David R. Kincaid & E. Ward Cheney. *Numerical Analysis: Mathematics of Scientific of Computing Third Edition*, Brooks/Cole, 2002.