作业三

姓名　徐家恒　　学号 PB18000334　　日期 2021. 6.16

第一题 本题考虑使用Richardson外推技术提高向前差商求给定函数导数的精度。

(a) （5分）使用向前差分计算$f(x) = \sin(x)$在$x = 1.2$处的导数并使用loglog图展示其精度随离散区间大小h的变化。h取$10^0, 10^{-1}, 10^{-2}, 10^{-3}, \ldots, 10^{-15}$。

代码如下：

```
clc,close;

f = @(x) sin(x);
x0 = 1.2;

x1 = zeros(1, 16);
x2 = zeros(1, 16);
for i = 1 : 16
    x1(i) = 10^(1 - i);
    x2(i) = abs((f(x0 + x1(i)) - f(x0)) / x1(i) - cos(x0));
end

loglog(x1, x2)
```

得到的精度如下图1

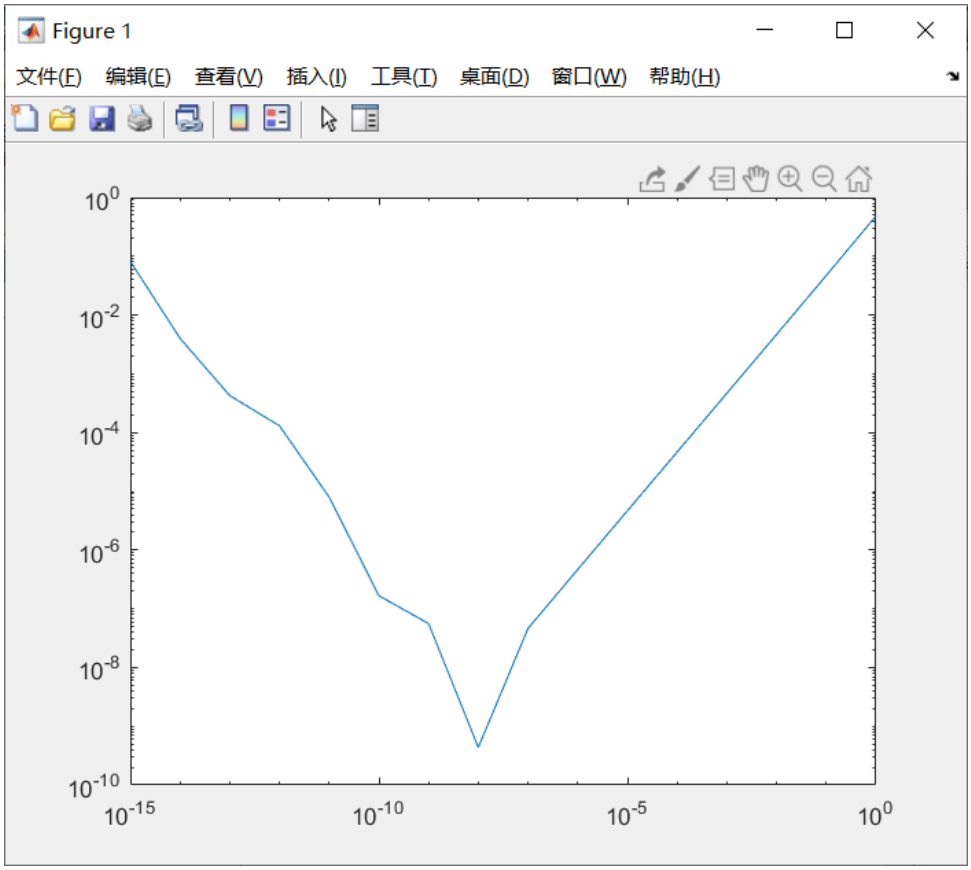

图 1: 用向前差分计算导数的精度图像

(b) （10分）推导出使用Richardson外推技术的向前差商的计算公式并用伪代码给出算法。

由泰勒展开

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + O(h^2)$$

得

$$f'(x_0) = \frac{1}{h}(f(x_0+h) - f(x_0)) - \frac{h}{2}f''(x_0) + O(h)$$

记

$$N_1(h) = \frac{f(x_0+h) - f(x_0)}{h}$$

$$f'(x_0) = N_1(h) + c_1 h + O(h) \tag{1}$$

$$f'(x_0) = N_1(\frac{h}{2}) + c_1(\frac{h}{2}) + O(h) \tag{2}$$

2·(2)-(1)

$$f'(x_0) = \frac{1}{2-1}(2N_1(\frac{h}{2}) - N_1(h)) + O(h) \approx N_2(h)$$

$$N_2(h) = N_1(\frac{h}{2}) + \frac{N_1(\frac{h}{2}) - N_1(h)}{2-1}$$

继续外推下去，得到

$$N_j(h) = N_{j-1}(\frac{h}{2}) + \frac{N_{j-1}(\frac{h}{2}) - N_{j-1}(h)}{2^{j-1}-1}, \quad j = 2, 3, \cdots$$

$$f'(x_0) = N_j(\frac{h}{2}) + O(h^j)$$

伪代码：

$let \quad A[1 \ldots maxrept] \quad a \quad new \quad array$

$A[1] = N_1(h)$

$for \quad i = 1 \quad to \quad maxrept$

$\quad tmp = A[1]$

$\quad for \quad j = i \quad to \quad 1 \quad descend$

$\quad\quad A[j] = A[j+1] + (A[j+1] - A[j])/(2^{i-j+1} - 1)$

$\quad end$

$\quad if \quad |tmp - A[1]| < \epsilon$

$\quad\quad return \quad tmp$

$\quad endif$

end

(c) （5分）使用你的算法计算$f(x) = \sin(x)$在$x = 1.2$处的导数并使用semilogy图展示其精度随外推次数变化的情况。请自己选取h初始值，使得用外推方法能够算出的最精确的导数尽量精确。你的回答需要说明你使用外推方法算出的导数值是多少、误差又是多少、h的初始值是多少、外推了多少次。

代码如下:

```
1 clear,clc
2
3 syms x;
4 f = @(x) sin(x);
5 x0 = 1.2;
```

```
6  maxrept = 200;
7  tmp0 = cos(1.2);
8
9  B = ones(1, 11);
10 for k = 5 : 15
11     epsilon = 10^(-k);
12     maxp = 0;
13     maxi = 0;
14     for p = 0 : 15
15         h = 10 ^ (-p);
16         A = zeros(1, maxrept);
17         A(1) = N1(f, x0, h);
18         tmp1 = 0;
19         for i = 1 : maxrept
20             tmp1 = A(1);
21             A(i + 1) = N1(f, x0, h / 2 ^ i);
22             for j = i : -1 : 1
23                 A(j) = A(j + 1) + (A(j + 1) - A(j)) / (2 ^ (i - j +
24             end
25             if abs(tmp1 - A(1)) < epsilon
26                 if abs(A(1) - tmp0) < abs(B(k - 4) - tmp0)
27                     B(k - 4) = A(1);
28                     maxp = p;
29                     maxi = i;
30                 end
31             end
32         end
33         if i == maxrept
34             %fprintf('h : %e\texceed\n', h);
35         end
36     end
37     fprintf('epsilon : %e\th : %e\ttimes : %d\tres : %.15f\terror :
38 end
39 semilogy(abs(B - tmp0))
40
```

```
41 function [res] = N1(f, x0, h)
42     res = (f(x0 + h) - f(x0)) / h;
43 end
```

结果:

```
1  epsilon : 1.000000e-05  h : 1.000000e+00    times : 7   res : 0.362
2  epsilon : 1.000000e-06  h : 1.000000e+00    times : 7   res : 0.362
3  epsilon : 1.000000e-07  h : 1.000000e+00    times : 7   res : 0.362
4  epsilon : 1.000000e-08  h : 1.000000e+00    times : 7   res : 0.362
5  epsilon : 1.000000e-09  h : 1.000000e+00    times : 7   res : 0.362
6  epsilon : 1.000000e-10  h : 1.000000e+00    times : 7   res : 0.362
7  epsilon : 1.000000e-11  h : 1.000000e-01    times : 6   res : 0.362
8  epsilon : 1.000000e-12  h : 1.000000e-01    times : 6   res : 0.362
9  epsilon : 1.000000e-13  h : 1.000000e-01    times : 6   res : 0.362
10 epsilon : 1.000000e-14  h : 1.000000e+00    times : 9   res : 0.362
11 epsilon : 1.000000e-15  h : 1.000000e+00    times : 9   res : 0.362
```

得到的不同精度在最优h下得到的导数如下图2

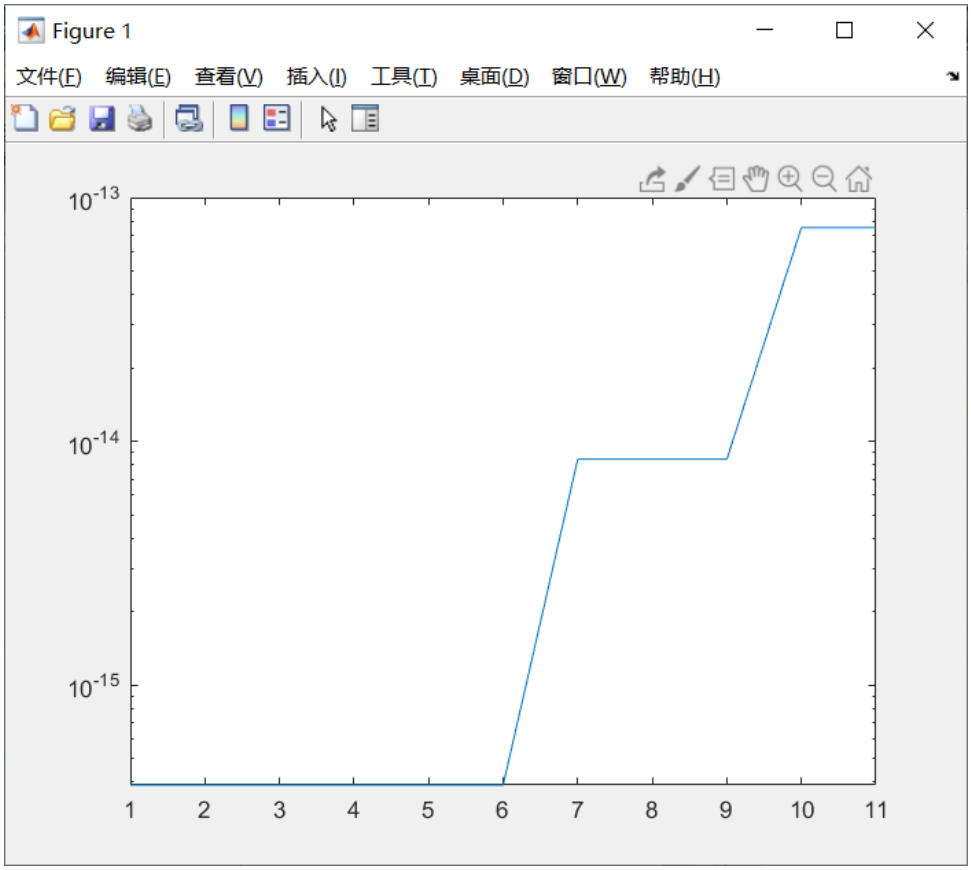

图 2: 不同精度在最优h下得到的导数图像

第二题 本题讨论使用复化梯形公式求周期函数的积分。

(a)(10分）证明当 r 不是 m 的整倍数的情况下，使用 m 个子区间的复化梯形公式可以精确积分

$$\int_{-\pi}^{\pi} \cos(rx)dx \text{ 和 } \int_{-\pi}^{\pi} \sin(rx)dx$$

注意：这一结论类似于课堂上我们定义的代数精度。同时说明如果 r 为 m 的整倍数时复化梯形公式对于上面两个积分会给出怎样的结果。

记 $M_2 = \max_{a\leqslant x\leqslant b} |f''(x)|$, 则

$$|E_n(f)| \leqslant \frac{(b-a)^3}{12n^2} M_2 = O\left(\frac{1}{n^2}\right)$$

对于任给的误差控制小量 $\varepsilon > 0$,

$$\frac{(b-a)^3}{12n^2} M_2 < \varepsilon \quad \text{或} \quad n \geqslant \left[\sqrt{\frac{(b-a)^3 M_2}{12\varepsilon}}\right] + 1$$

就有 $|E_n(f)| < \varepsilon$，由

$$cos''(rx) = -r^2 cos(rx), sin''(rx) = -r^2 sin(rx)$$

得$M_2 = r^2$，则r是m的整数倍时，有

$$\frac{(2\pi)^3}{12m^2}k^2m^2 = \frac{(2\pi)^3}{12}k^2 > \varepsilon$$

无论如何都无法精确积分，r不是m的整数倍时，只需满足

$$m \geqslant \left[\sqrt{\frac{(2\pi)^3 r^2}{12\varepsilon}}\right] + 1$$

便可使用m个子区间精确积分。

（b)（10分）由于任意的以 2π 为周期的周期函数都可以表示为由正弦与余弦函数的线性组合，所以上一问中所证明的定理实际上告诉我们求解一个周期函数的积分的有效方法正是复化梯形公式。现使用复化梯形公式和不同数量的子区间个数来求 $f(x) = e^{\cos(x)}$ 在 $[-\pi, \pi]$ 的积分，并用这个函数的真实积分值作为参照使用semilogy图画出随着子区间数量 m 变化所得到的积分精度的变化。

题后语: 复化梯形公式之于周期函数的积分如同Gauss积分公式之于非周期函数的基于多项式的积分。因此复化梯形公式是最有效的求解周期函数积分的方法。

代码如下:

```
clc,clear

f = @(x) exp(cos(x));
exact = 7.954926521012845274513219665 3;

for i = 10 : 100
    step = 2 * pi / i;
    tmp = f(-pi)/2 + f(pi)/2;
    para = -pi + step;
    for j = 1 : i - 1
        tmp = tmp + f(para);
        para = para + step;
    end
    tmp = tmp * step;
    T(i - 9) = abs(tmp - exact);
end

semilogy(T)
```

得到的不同子区间数量下计算面积的误差如下图3

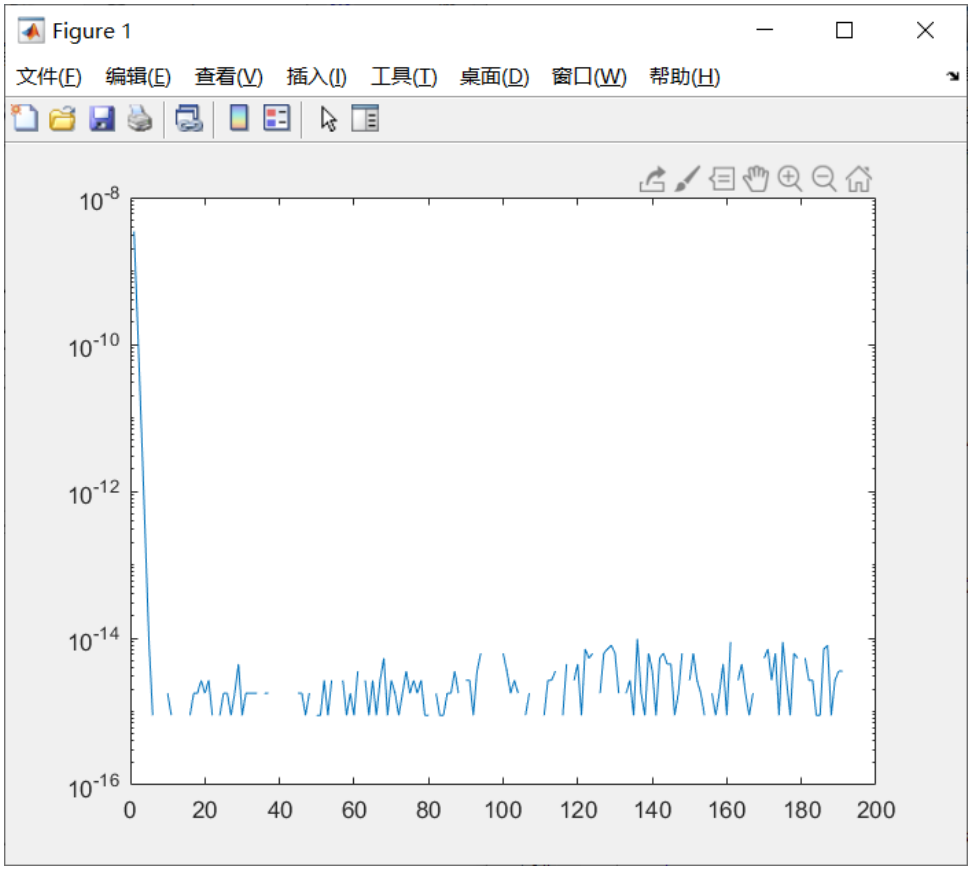

图 3: 不同子区间数量下计算面积的误差图像

第三题 (a)（10分）推导出如下格式的多步法公式：

$$y_{n+1} = y_{n-1} + \alpha f_{n+1} + \beta f_n + \gamma f_{n-1}$$

则积分区间为$[x_{n-1}, x_{n+1}]$，积分节点为$\{x_{n+1}, x_n, x_{n-1}\}$

$$y_{n+1} = y_{n-1} + h(a_0 f(x_{n+1}, y_{n+1}), a_1 f(x_n, y_n), a_2 f(x_{n-1}, y_{n-1}))$$

其中的积分系数为

$$\alpha = \int_{x_{n-1}}^{x_{n+1}} \frac{(x - x_n)(x - x_{n-1})}{(x_{n+1} - x_n)(x_{n+1} - x_{n-1})} dx = \frac{1}{3}h$$

$$\beta = \int_{x_{n-1}}^{x_{n+1}} \frac{(x - x_{n+1})(x - x_{n-1})}{(x_n - x_{n+1})(x_n - x_{n-1})} dx = \frac{4}{3}h$$

$$\gamma = \int_{x_{n-1}}^{x_{n+1}} \frac{(x - x_{n+1})(x - x_n)}{(x_{n-1} - x_{n+1})(x_{n-1} - x_n)} dx = \frac{1}{3}h$$

即

$$y_{n+1} = y_{n-1} + \frac{h}{3}(f_{n+1} + 4f_n + f_{n-1})$$

(b) （10分）推导此格式的局部截断误差，并由此指明此格式的阶数。

将

$$y_{n+1} = y_{n-1} + \frac{h}{3}(f_{n+1} + 4f_n + f_{n-1})$$

在x_n处作 Taylor 展开，与$y(x_{n+1})$在x_n处的 Taylor 展开式做比较，即可得到局部截断误差表达式

$$T_{n+1} = \frac{1}{3}h^3 y^{(3)}(\eta)$$

为2阶

(c) （10分）选取合适的步长值，用此格式在$[0, 2]$上解如下的初值问题：

$$y' = xe^{-4x} - 4y, y(0) = 0$$

使用你刚刚推导出的格式，并利用二阶Runge-Kutta方法(即课本公式(7.15))起步。画出解函数。

预估校正公式

$$y_{n+1} = y_{n-1} + \frac{h}{3}(7f_{n+1} - 2f_n + f_{n-1})$$

代码如下：

```
clc,close

f = @(x, y) x * exp(-4 * x) - 4 * y;

n = 2000;
h = 2 / n;

x = zeros(1, n);
y = zeros(1, n);
k1 = f(0, 0);
k2 = f(h / 2, h / 2 * k1);
x(2) = h;
y(2) = y(1) + h * k2;
```

```
k1 = f(h, y(2));
k2 = f(x(2) + h / 2, y(2) + h / 2 * k1);
x(3) = x(2) + h;
y(3) = y(2) + h * k2;
f1 = f(x(1), y(1));
f2 = f(x(2), y(2));
f3 = f(x(3), y(3));
for i = 3 : n
    x(i + 1) = x(i) + h;
    tmp = (y(i - 1) + h * (7/3 * f3 - 2/3 * f2 + 1/3 * f1));
    y(i + 1) = (y(i - 1) + h * (1/3 * f(x(i + 1), tmp) + 4/3 * f3 +
    f1 = f2;
    f3 = f(x(i + 1), y(i + 1));
end

plot(x, y)
```

得到的解函数如下图4

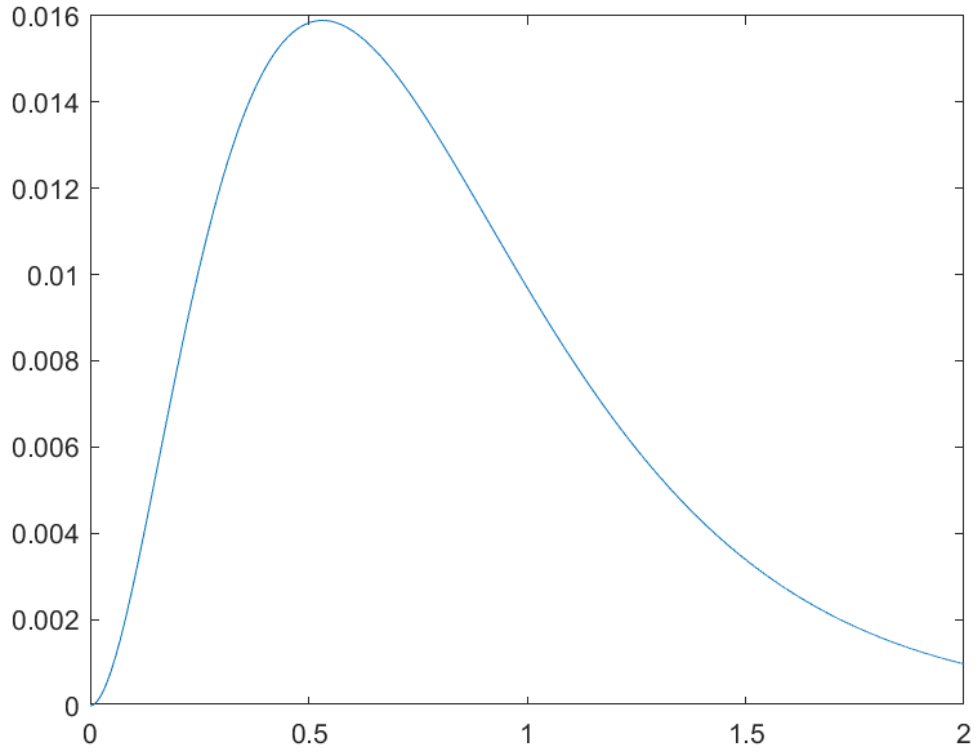

图 4: [0,2]的解函数图像

(d) （10分）推导出(2)的精确解。对比该精确解在x = 2这一点的值，用log-log图展示所用方法的阶数，并给出解释。

观察得，可设$y = (ax^2 + bx + c)e^{-4x}$，代入得

$$(-4ax^2 + (2a - 4b)x + b - 4c)e^{-4x} = xe^{-4x} - 4(ax^2 + bx + c)e^{-4x}$$

对比系数可得到

$$\begin{cases} -4a = -4a \\ 2a - 4b = 1 - 4b \\ b - 4c = -4c \end{cases}$$

则$a = \frac{1}{2}, b = 0$，又由$y(0) = 0$得c=0,代入得

$$y = \frac{1}{2}x^2e^{-4x}$$

代码如下：

```
clc,close

f = @(x, y) x * exp(-4 * x) - 4 * y;
exc = 1/2 * 2^2 * exp(-4 * 2);
x0 = 10:10000;
z = zeros(1, length(x0));
for n = 10 : 10000
    h = 2 / n;
    x = zeros(1, n);
    y = zeros(1, n);
    k1 = f(0, 0);
    k2 = f(h / 2, h / 2 * k1);
    x(2) = h;
    y(2) = y(1) + h * k2;
    k1 = f(h, y(2));
    k2 = f(x(2) + h / 2, y(2) + h / 2 * k1);
    x(3) = x(2) + h;
    y(3) = y(2) + h * k2;
    f1 = f(x(1), y(1));
```

```
20      f2 = f(x(2), y(2));
21      f3 = f(x(3), y(3));
22      for i = 3 : n
23          x(i + 1) = x(i) + h;
24          tmp = (y(i - 1) + h * (7/3 * f3 - 2/3 * f2 + 1/3 * f1));
25          y(i + 1) = (y(i - 1) + h * (1/3 * f(x(i + 1), tmp) + 4/3 *
26          f1 = f2;
27          f2 = f3;
28          f3 = f(x(i + 1), y(i + 1));
29      end
30      z(n - 9) = abs(y(n + 1) - exc);
31  end
32
33  loglog(x0,z)
```

得到的loglog图像如下图5

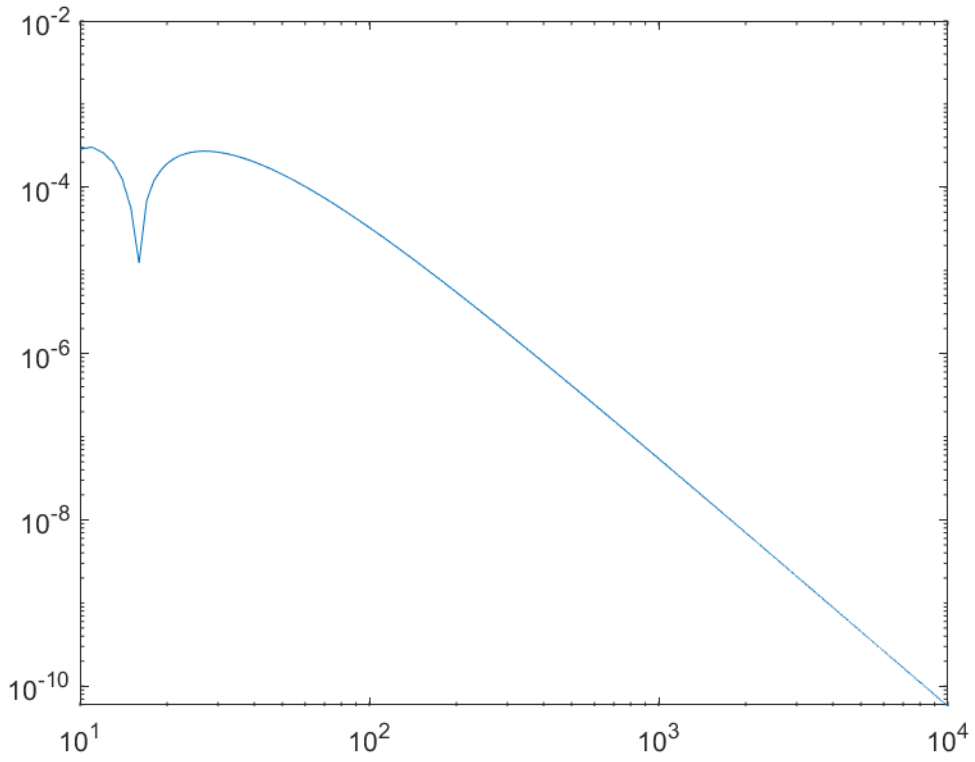

图 5: loglog图像

由图像知斜率约为3，阶数为3