作业三

姓名 郑龙韬 学号 **PB18061352** 日期 **2021/6/9**

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

(a) 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 $10^0, 10^{-1}, 10^{-2}, 10^{-3}, \ldots, 10^{-15}$。

取误差最小的 h，使用向前差分计算出的 $f(x) = \sin(x)$ 在 $x = 1.2$ 处的导数为 0.36235775491277877336，如以下代码输出：

```
Diff
0.36235775491277877336
```

精度随离散区间大小 h 的变化的 loglog 图如图 1。

本题 Matlab 代码如下：

```
clear, clc
format long
h_exp = linspace(0,-15,16);
h = 10.^h_exp;
val = 1.2;
f = @(x) sin(x);
f_prime = @(x) cos(x);
% forward difference
difference = (f(val+h)-f(val))./h;
err = abs(difference-f_prime(val));
% draw loglog
```

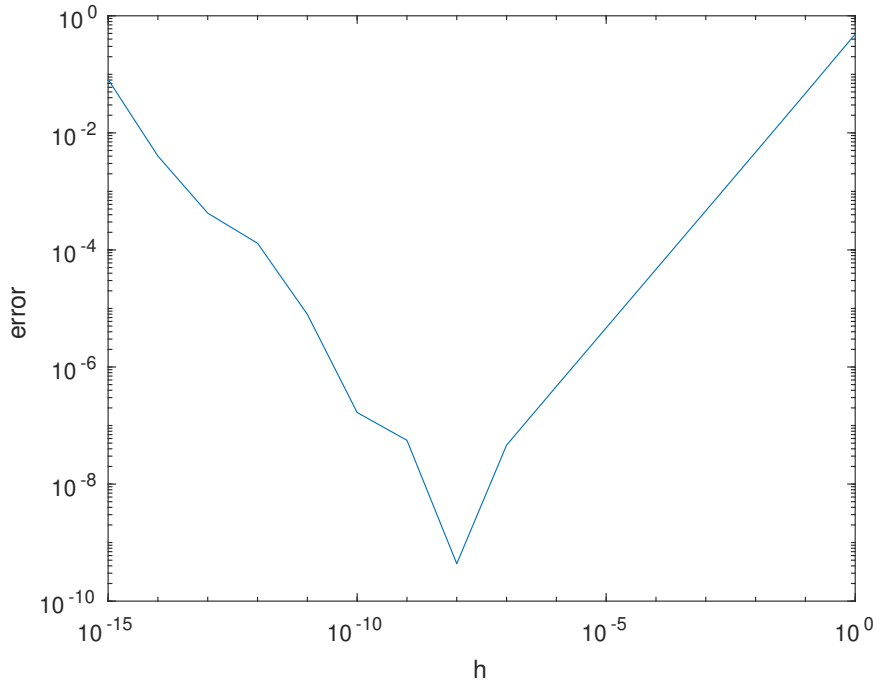

图 1: 精度随离散区间大小 h 的变化

```
loglog(h, err)
xlabel('h')
ylabel('error')
[min_err, ind] = min(err);
grad = difference(ind);
% output result
fprintf(['Diff\n']);
fprintf('%.20f\n', grad);
```

(b) 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。

$$f(x_0+h)-f(x_0)=hf'(x_0)+\frac{h^2}{2!}f''(x_0)+O(h^3)$$

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

记

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

$$f'(x_0)=N_1(h)-\frac{h}{2}f''(x_0)+O(h^2)=N_1(h)+c_1h+O(h^2) \tag{1}$$

$$f'(x_0) = N_1(\frac{h}{2}) - \frac{h}{4}f''(x_0) + O(h^2) = N_1(\frac{h}{2}) + c_1\frac{h}{2} + O(h^2) \tag{2}$$

2·(2)-(1)

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

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

继续外推下去

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

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

伪代码如下，使用递归写法，其中 j, h, x_0, f 符号同上。

Algorithm 1 Forward(j, h, x_0, f)

if $j == 1$ **then** ▷ condition to end recursion
  return $\frac{f(x_0+h)-f(x_0)}{h}$ ▷ $N_1(h)$
else
  return Forward$(j-1, \frac{h}{2}, x_0, f) + \frac{\text{Forward}(j-1,\frac{h}{2},x_0,f)-\text{Forward}(j-1,h,x_0,f)}{2^{j-1}-1}$ ▷ $N_j(h)$
end if

(c) 精度随外推次数变化的情况的 semilogy 图如图 2。

使用外推方法计算出的 $f(x) = \sin(x)$ 在 $x = 1.2$ 处的导数值为 0.36235775447667323279，误差为 3.885781×10^{-16}，h 的初始值为 1，外推了 8 次，使得算出的最精确的导数尽量精确。如以下代码输出：

Diff	error	Initial h	iter.
0.36235775447667323279	3.885781e-16	1	8

本题 Matlab 代码如下：

```
clear, clc
format long
h = 10^(0);
val = 1.2;
max_iter = 20;
iter = linspace(1, max_iter, max_iter);
f = @(x) sin(x);
f_prime = @(x) cos(x);
```

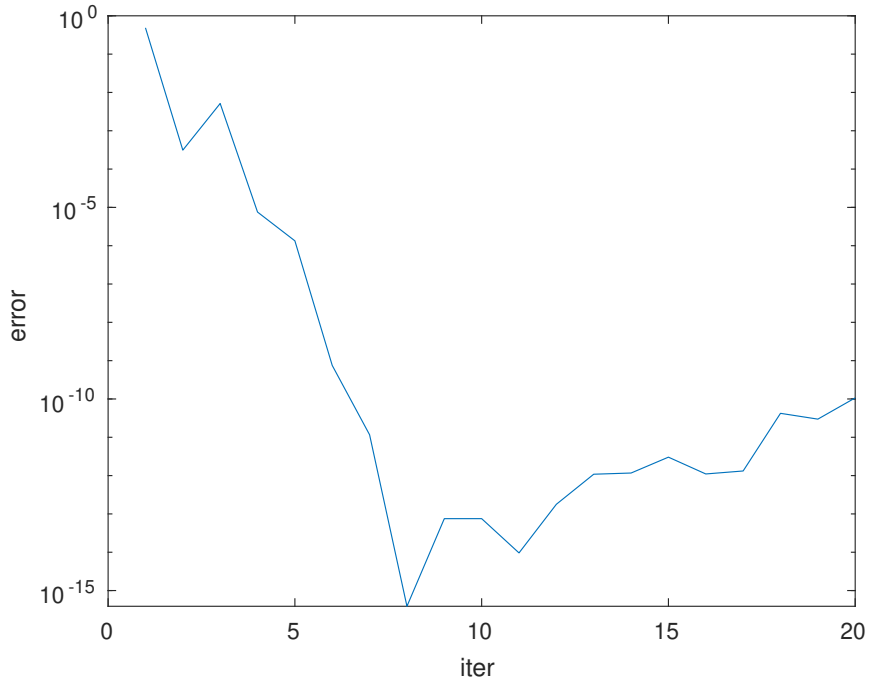

图 2: 精度随外推次数变化的情况

```
% iteration
for j = 1:max_iter
    Richardson(j) = N(j, h);
    err(j) = abs(Richardson(j)-f_prime(val));
end
% draw and output result
semilogy(iter, err)
xlabel('iter')
ylabel('error')
[min_err, ind] = min(err);
grad = Richardson(ind);
fprintf(['Diff\t\terror\t\tInitial h\titer.\n']);
fprintf('%.20f\t%d\t%d\t%d\t\n', grad, min_err, h, ind);
% Richardson forward
function N_j = N(j, h)
    val = 1.2;
    f = @(x) sin(x);
    if j == 1
```

```
        N_j = (f(val+h)-f(val))/h;
    else
        item1 = N(j-1,h/2);
        item2 = N(j-1,h);
        N_j = item1+(item1-item2)/(2^(j-1)-1);
    end
end
```

第二题 使用复化梯形公式求周期函数的积分。

(a) 证明：

当 r 不是 m 的整倍数的情况下，

(i) 对 $\int_{-\pi}^{\pi}\cos(rx)dx$，

$$
\begin{aligned}
T(h) &= h[\frac{1}{2}\cos(-r\pi)+\sum_{i=1}^{m-1}\cos(r(-\pi+ih))+\frac{1}{2}\cos(r\pi)] \\
&= h(\cos(r\pi)+\sum_{i=1}^{m-1}\cos(-r\pi+rih)) \\
&= h(\cos(r\pi)+\sum_{i=1}^{m-1}(\cos(-r\pi)\cos(rih)-0)) \\
&= h\cos(r\pi)(1+\sum_{i=1}^{m-1}\cos(rih)) \\
&= h\cos(r\pi)\frac{\sin(mhr)\cos(hr/2)-\cos(mhr)\sin(hr/2)+\sin(hr/2)}{2\sin(hr/2)} \\
&= h\cos(r\pi)\frac{\sin(2\pi r)\cos(hr/2)-\cos(2\pi r)\sin(hr/2)+\sin(hr/2)}{2\sin(hr/2)} \\
&= h\cos(r\pi)\frac{(1-\cos(2\pi r))\sin(hr/2)}{2\sin(hr/2)} \\
&= 0 \\
&= \int_{-\pi}^{\pi}\cos(rx)dx
\end{aligned}
$$

(ii) 对 $\int_{-\pi}^{\pi}\sin(rx)dx$,

$$\begin{aligned}
T(h) &= h[\frac{1}{2}\sin(-r\pi)+\sum_{i=1}^{m-1}\sin(r(-\pi+ih))+\frac{1}{2}\sin(r\pi)] \\
&= h\sum_{i=1}^{m-1}\sin(-r\pi+rih) \\
&= h\cos(r\pi)\sum_{i=1}^{m-1}\sin(rih) \\
&= h\cos(r\pi)\frac{(1-\cos(mhr))\cos(hr/2)-\sin(mhr)\sin(hr/2)}{2\sin(hr/2)} \\
&= h\cos(r\pi)\frac{(1-\cos(2\pi r))\cos(hr/2)-\sin(2\pi r)\sin(hr/2)}{2\sin(hr/2)} \\
&= 0 \\
&= \int_{-\pi}^{\pi}\sin(rx)dx
\end{aligned}$$

所以，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以上二式。 □

当 r 为 m 的整倍数时,上述证明过程中对求和项的展开不继续成立,对于 $\int_{-\pi}^{\pi}\cos(rx)dx$,就无法精确积分;而对于 $\int_{-\pi}^{\pi}\sin(rx)dx$,由于 r 为 m 的整倍数,代入 $h\cos(r\pi)\sum_{i=1}^{m-1}\sin(rih)$,结果仍为 0，故可以精确积分。

(b) 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 Wolfram alpha 求出这个函数的真实积分值，画出积分精度随着子区间数量 m 变化的 semilogy 图，如图 3。

使用复化梯形公式求出的最精确的积分为 7.95492652101284569710，对应误差为 0，子区间数量为 16，代码输出如下：

```
result                    error   m
7.95492652101284569710    0       16
```

本题 Matlab 代码如下：

```
clear, clc
format long
max_range_num = 20;
m_s = linspace(1, max_range_num, max_range_num);
f = @(x) exp(cos(x));
% calculated by Wolfram alpha
```

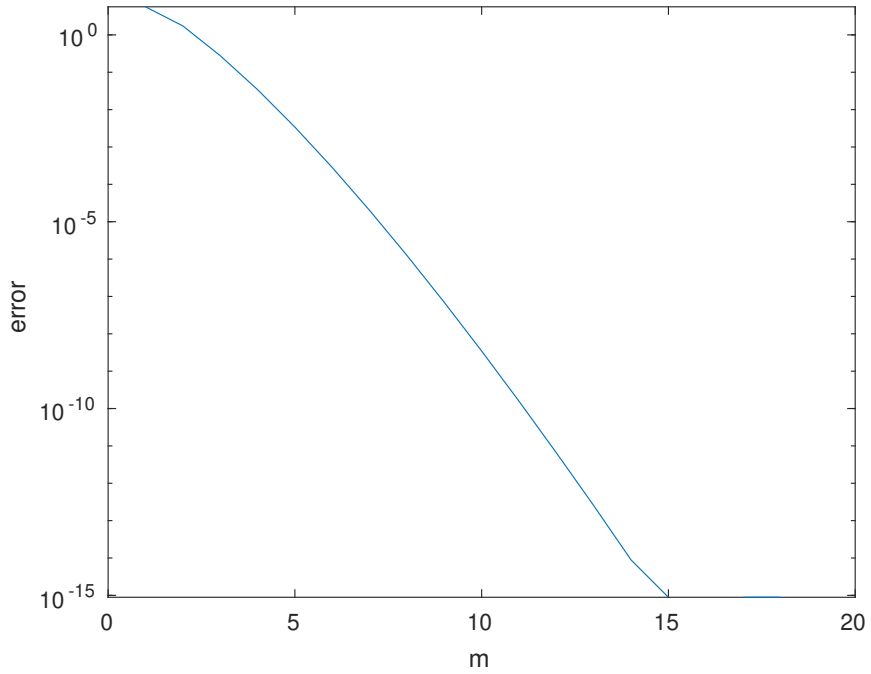

图 3: 精度随子区间数量 m 变化的情况

```
true_integral = 7.954926521012845274513219665329394...
    328161342771816638573400595955383360608164694 66...
    6995137357228568774;
left = -pi;
right = pi;
for m = 1:max_range_num
    h = (right-left)/m;
    item = 1/2*f(left)+1/2*f(right);
    for i = 1:m-1
        item = item+f(left+i*h);
    end
    trapezoid(m) = h*item;
    err(m) = abs(trapezoid(m)-true_integral);
end
semilogy(m_s, err)
xlabel('m')
ylabel('error')
[min_err, ind] = min(err);
```

```
result = trapezoid(ind);
fprintf(['result\t\terror\tm\n']);
fprintf('%.20f\t%2d\t%d\n', result, min_err, ind);
```

第三题 **(a)** 推导出如下格式的多步法公式:

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

解: 构造格式如下,

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

其中的积分系数为

$$\begin{aligned}
a_0 h &= \int_{x_{n-1}}^{x_{n+1}} \frac{(x-x_n)(x-x_{n-1})}{(x_{n+1}-x_n)(x_{n+1}-x_{n-1})} dx = \frac{1}{h^2}(\frac{4h^3}{3} - h^3) = \frac{1}{3}h \\
a_1 h &= \int_{x_{n-1}}^{x_{n+1}} \frac{(x-x_{n+1})(x-x_{n-1})}{(x_n-x_{n+1})(x_n-x_{n-1})} dx = \frac{1}{h^2}(4h^3 - \frac{8h^3}{3}) = \frac{4}{3}h \\
a_2 h &= \int_{x_{n-1}}^{x_{n+1}} \frac{(x-x_n)(x-x_{n+1})}{(x_{n-1}-x_n)(x_{n-1}-x_{n+1})} dx = \frac{1}{h^2}(\frac{4h^3}{3} - h^3) = \frac{1}{3}h
\end{aligned}$$

令 $\alpha = \frac{1}{3}h, \beta = \frac{4}{3}h, \gamma = \frac{1}{3}h$, 得到计算格式

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

□

(b) 推导此格式的局部截断误差, 并由此指明此格式的阶数。

解: 对上面的格式, 使用 Taylor 展开来估计线性多步格式的局部截断误差, 若 $y_{n+1} = y(x_{n+1}), y_n = y(x_n), y_{n-1} = y(x_{n-1})$, 则有

$$\begin{aligned}
y_{n+1} &= y_{n-1} + \alpha f_{n+1} + \beta f_n + \gamma f_{n-1} \\
&= y(x_{n-1}) + \alpha y'(x_{n+1}) + \beta y'(x_n) + \gamma y'(x_{n-1})
\end{aligned}$$

在 x_n 处作 Taylor 展开,

$$\begin{aligned}
y(x_{n-1}) &= y(x_n) - hy'(x_n) + \frac{h^2}{2!}y^{(2)}(x_n) - \frac{h^3}{3!}y^{(3)}(x_n) + \frac{h^4}{4!}y^{(4)}(x_n) - \frac{h^5}{5!}y^{(5)}(x_n) + o(h^5) \\
y'(x_{n+1}) &= y'(x_n) + hy^{(2)}(x_n) + \frac{h^2}{2!}y^{(3)}(x_n) + \frac{h^3}{3!}y^{(4)}(x_n) + \frac{h^4}{4!}y^{(5)}(x_n) + o(h^4) \\
y'(x_{n-1}) &= y'(x_n) - hy^{(2)}(x_n) + \frac{h^2}{2!}y^{(3)}(x_n) - \frac{h^3}{3!}y^{(4)}(x_n) + \frac{h^4}{4!}y^{(5)}(x_n) + o(h^4)
\end{aligned}$$

代入得，

$$y_{n+1} = y(x_n) + hy'(x_n) + \frac{h^2}{2!}y^{(2)}(x_n) + \frac{h^3}{3!}y^{(3)}(x_n) + \frac{h^4}{4!}y^{(4)}(x_n) + \frac{7}{360}y^{(5)}(x_n) + o(h^5)$$

与 y_{n+1} 在 x_n 处的 Taylor 展开式作比较，

$$y_{n+1} = y(x_n) + hy'(x_n) + \frac{h^2}{2!}y^{(2)}(x_n) + \frac{h^3}{3!}y^{(3)}(x_n) + \frac{h^4}{4!}y^{(4)}(x_n) + \frac{h^5}{5!}y^{(5)}(x_n) + o(h^5)$$

得到局部截断误差表达式，

$$T_{n+1} = (\frac{7}{360} - \frac{1}{5!})h^5y^{(5)}(\eta) = \frac{1}{90}h^5y^{(5)}(\eta)$$

故此格式的阶数为四阶。 □

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

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

解：该格式为

$$y_{n+1} = y_{n-1} + \frac{1}{3}hf_{n+1} + \frac{4}{3}hf_n + \frac{1}{3}hf_{n-1}$$

化为显式，

$$\begin{aligned}
y_{n+1} - \frac{1}{3}hf_{n+1} &= y_{n-1} + \frac{4}{3}hf_n + \frac{1}{3}hf_{n-1} \\
y_{n+1} - \frac{1}{3}h(xe^{-4x} - 4y_{n+1}) &= y_{n-1} + \frac{4}{3}hf_n + \frac{1}{3}hf_{n-1} \\
(1 + \frac{4}{3}h)y_{n+1} - \frac{1}{3}hxe^{-4x} &= y_{n-1} + \frac{4}{3}hf_n + \frac{1}{3}hf_{n-1} \\
y_{n+1} &= \frac{1}{(1 + \frac{4}{3}h)}(\frac{1}{3}hxe^{-4x} + y_{n-1} + \frac{4}{3}hf_n + \frac{1}{3}hf_{n-1})
\end{aligned}$$

利用二阶 Runge-Kutta 方法起步，画出的解函数如图 4。

本题 Matlab 代码如下：

```
clear, clc
format long
f = @(x, y) x*exp(-4*x)-4*y;
g = @(x) x*exp(-4*x);
m = 500;
left = 0;
right = 2;
h = (right-left)/m;
```

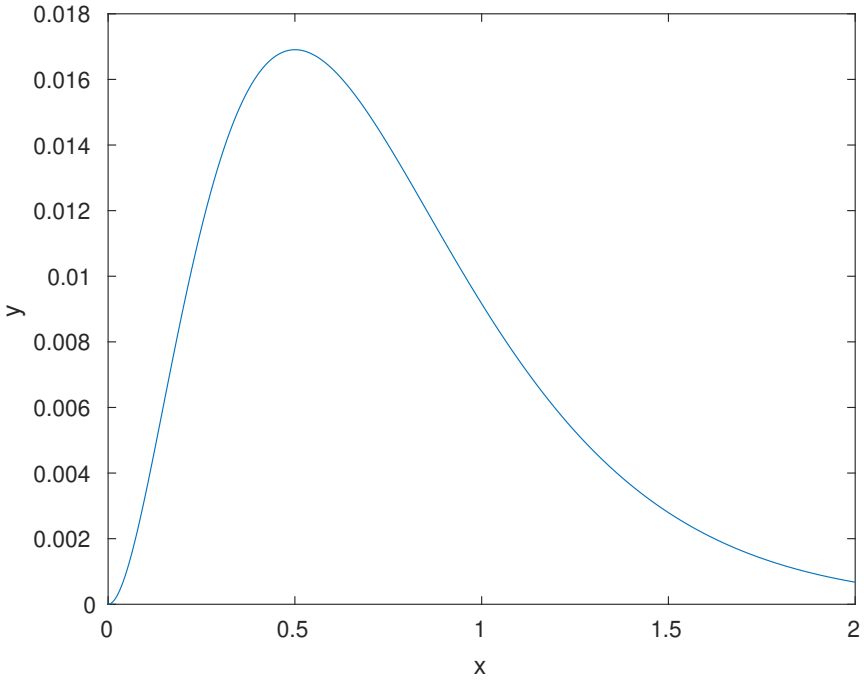

图 4: 解函数

```
x = linspace(left, right, m);
% 2-order Runge-Kutta
y(1) = 0;
k1 = f(x(1), y(1));
k2 = f(x(1)+h/2, y(1)+h/2*k1);
y(2) = y(1)+h*k2;
% linear multistep
for n = 2:m-1
    y(n+1) = (1/(1+4/3*h))* ...
        (1/3*h*g(x(n+1))+y(n-1) ...
        +4/3*h*f(x(n),y(n))+ ...
        1/3*h*f(x(n-1),y(n-1)));
end
plot(x, y)
xlabel('x')
ylabel('y')
```

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

解：推导精确解如下：

$$\begin{aligned} y' &= xe^{-4x} - 4y \\ y'e^{4x} + 4ye^{4x} &= x \\ (ye^{4x})' &= x \\ y &= \frac{1}{2}x^2e^{-4x} \end{aligned}$$

loglog 图如图 5。

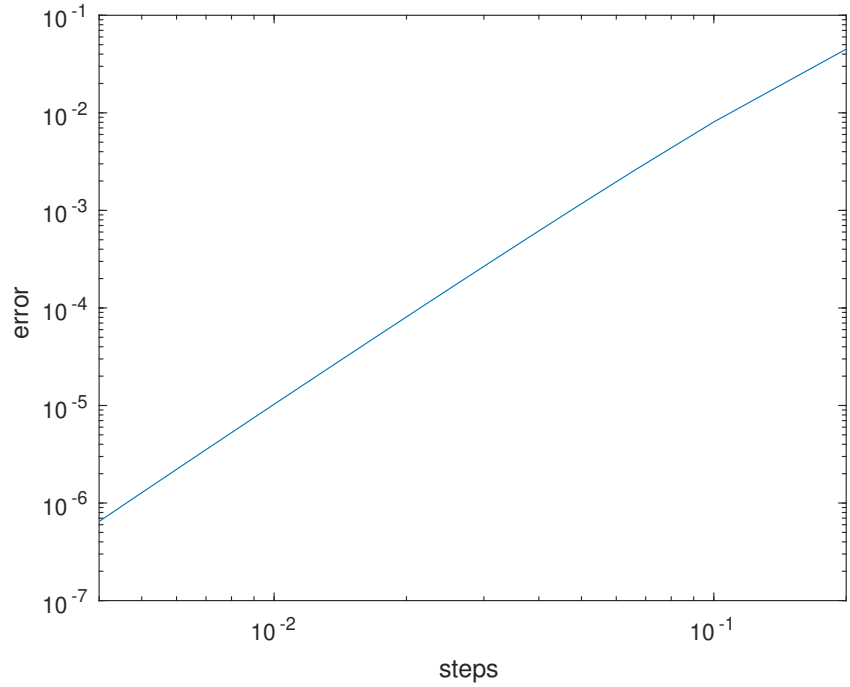

图 5: 所用方法的阶数

由图可观察到，使用的方法阶数为三阶。而在上面我们已推导出该方法的阶数为四阶，由于使用的起步方法为二阶 Runge-Kutta，而在多步格式的计算中，起步计算的精度至多只能比该格式的精度低一阶，所以影响了该格式的整体截断误差。对于如上的四阶格式，我们需要用至少三阶的格式来做起步计算。

本题 Matlab 代码如下：

```
clear, clc
format long
f = @(x, y) x*exp(-4*x)-4*y;
g = @(x) x*exp(-4*x);
acc = @(x) 1/2*x^2*exp(-4*x);
```

```
left = 0;
right = 2;
val = 2;
steps = [];
err = [];
for m = 10:10:500
    x = linspace(left, right, m);
    h = (right-left)/m;
    steps = [steps, h];
    % 2-order Runge-Kutta
    y(1) = 0;
    k1 = f(x(1), y(1));
    k2 = f(x(1)+h/2, y(1)+h/2*k1);
    y(2) = y(1)+h*k2;
    % linear multistep
    for n = 2:m-1
        y(n+1) = (1/(1+4/3*h))* ...
            (1/3*h*g(x(n+1))+y(n-1) ...
            +4/3*h*f(x(n),y(n))+ ...
            1/3*h*f(x(n-1),y(n-1)));
    end
    err = [err, abs(acc(val)-y(m)-(acc(val-h)-y(m-1)))];
end
loglog(steps, err)
xlabel('steps')
ylabel('error')
```