作业二

林成渊　　PB18051113　　2021 年 8 月 21 日

第一题 使用多种方法计算积分

(a) 根据题意，编写代码使用自动控制误差的复化梯形公式计算积分

```
clear, clc, clf
MS = 'MarkerSize'; ms = 8;
MC = 'MarkerFaceColor';

% Def of the problem
F = @(x) sin(cos(sin(cos(x))));
a = -1; b = 1;
tol = 1e-15;



%%
% Init the Loop
n = 1; % Initial of the intervals

h = (b - a) / n;
x = linspace(a, b, n + 1);
% Save each Int value, Dynamic Programming
I = zeros(1, 100);
I(2) = h * (F(a) / 2 + sum(F(x(2:end - 1))) + F(b) / 2);
I(1) = I(2)+1;
i = 2;
% Loop begin
while abs(I(i) - I(i-1)) > 3*tol
    % If this loop begin, the former loop failed
    % Switch to a new loop
    i = i + 1;
    h = h / 2;
    % Calculate for this loop
    H = h * sum(F(a + (2 * (1:n) - 1) * h));
    I(i) = I(i-1)/2 + H;

    n = n * 2;
end
%%
% Output
format long
display(I(i));
display(n);
figure(1);
semilogy((1:i-1), abs(I(2:i) - I(i))/3, 'ro-', MC, 'b', MS, ms);
xlabel('迭代次数');
ylabel('误差');
```

输出结果，积分值为 1.339880713117285，使用插值点数为 16777216。得到图 1

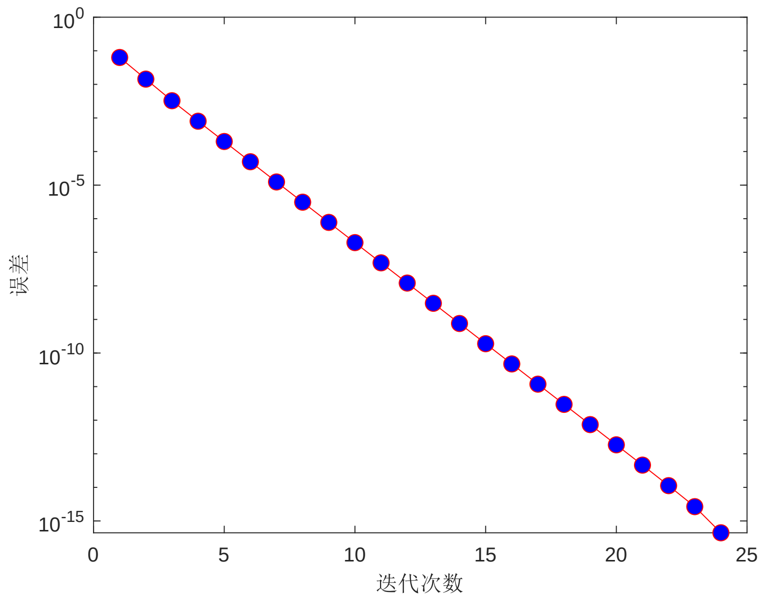

图 1: 第一题 (a) 小题作图结果

(b) 根据题意，编写代码使用 Richardson 外推方法计算积分，代码如下

```
clear , clc , clf
MS = 'MarkerSize'; ms = 8;
MC = 'MarkerFaceColor';
```

```
% Def of the problem
F = @(x) sin(cos(sin(cos(x))));
a = -1; b = 1;
tol = 1e-15;

% Some other options
M = 15;

%%
% Init the Loop
R = zeros(M,M);
n = 1; % Initial of the intervals
h = b - a;
R(1,1) = (F(a) + F(b)) * h / 2;
errors = zeros(M-1);
for k = 2:M
    R(k,1) = (R(k-1,1) +
            h*sum(F(a + (2.*(1:n)-1) .* (h/2)))) / 2;
    h = h / 2;
    n = n * 2;
    for j = 2:k
        R(k,j) = R(k,j-1) +
                (R(k,j-1) - R(k-1,j-1))/(4^(j-1) - 1);
    end
    errors(k-1) = abs(R(k,k)-R(k-1,k-1));
    if errors(k-1) < tol
        break;
    else
    end
end

%%
% Output
format long
```

```
disp(R(k,k));
disp(k-1);
figure(1);
semilogy((1:k-1), errors(1:k-1), 'ro-', MC, 'b', MS, ms);
xlabel('迭代次数');
ylabel('误差');
```

输出结果，积分值为 1.339880713117284，使用插值点数为 $2^8 = 256$。得到图 2

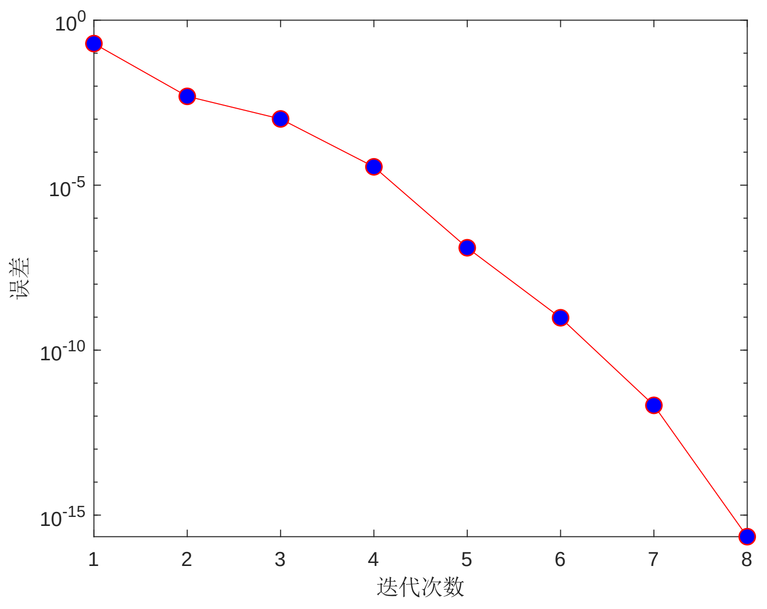

图 2: 第一题 (b) 小题作图结果

(c) 根据题意，编写代码使用 Gauß 积分方法计算积分，代码如下

```
clear,clc
LW = 'LineWidth'; lw = 2;
MS = 'MarkerSize'; ms = 8;
MC = 'MarkerFaceColor';

% Def of the problem
F = @(x) sin(cos(sin(cos(x))));
a = -1; b = 1;
tol = 1e-15;

% Other Options
```

```
M = 20;
I = zeros(1,M);
[x, w] = gauss(1);
I(1) = w*F(x);
errors = zeros(1,M-1);
for n = 2:M
    [x,w] = gauss(n);
    I(n) = w*F(x);
    errors(n-1) = abs(I(n) - I(n-1));
    display(errors(n-1));
    if abs(I(n)-I(n-1)) < tol
        break;
    else
    end
end
%%
% Output
format long
disp(I(n));
disp(n);
figure(1);
semilogy((2:n), errors(1:n-1), 'ro-', MC, 'b', MS, ms);
xlabel('n的取值');
ylabel('误差');

% GAUSS  nodes x (Legendre points) and weights w
%        for Gauss quadrature

function [x,w] = gauss(N)
    beta = .5./sqrt(1-(2*(1:N-1)).^(-2));
    T = diag(beta,1) + diag(beta,-1);
    [V,D] = eig(T);
    x = diag(D); [x,i] = sort(x);
    w = 2*V(1,i).^2;
end
```

输出结果，积分值为 1.339880713117284，得出这个积分本身使用了 15 个插值点，每次迭代使用的插值点数及其误差如图 3

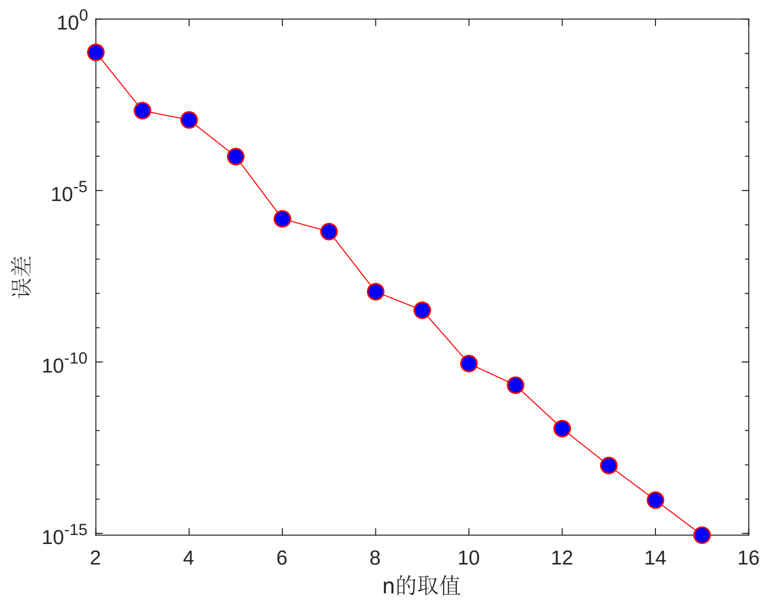

图 3: 第一题 (c) 小题作图结果

(d) 以上三种方法计算积分均以接近计算机的机器精度为其迭代的目标误差上界。最后，使用自动控制误差的复化梯形公式总共采样 16777216 个点，使用 Richardson 外推方法计算积分总共采样 256 个点，使用 Gauß 积分方法计算积分总共采样 15 个点，由此可见，就效率而言，三种方法从高到低依次排序为：Gauß 积分方法、Richardson 外推方法、使用自动控制误差的复化梯形公式方法。

第二题 函数求导与微分矩阵 (differentiation matrix)

(a) 证明：根据题设条件得

$$\begin{aligned}
\ell_j'(x) &= \left(\frac{1}{\pi_j}\prod_{k=0,k\neq j}^{n}(x-x_k)\right)' \\
&= \frac{1}{\pi_j}\left(\prod_{k=0,k\neq j}^{n}(x-x_k)\right)' \\
&= \frac{1}{\pi_j}((x-x_1)(x-x_2)\dots(x-x_n)+(x-x_0)(x-x_3)\dots(x-x_n)+\dots) \\
&= \frac{1}{\pi_j}\left(\frac{\prod_{k=0,k\neq j}^{n}(x-x_k)}{x-x_0}+\frac{\prod_{k=0,k\neq j}^{n}(x-x_k)}{x-x_1}+\dots+\frac{\prod_{k=0,k\neq j}^{n}(x-x_k)}{x-x_n}\right) \\
&= \frac{1}{\pi_j}\prod_{k=0,k\neq j}^{n}(x-x_k)*\left(\frac{1}{x-x_0}+\frac{1}{x-x_1}+\dots\right) \\
&= \frac{1}{\pi_j}\prod_{k=0,k\neq j}^{n}(x-x_k)*\sum_{k=0,k\neq j}^{n}\frac{1}{x-x_k} \\
&= \ell_j(x)\sum_{k=0,k\neq j}^{n}(x-x_k)^{-1}
\end{aligned}$$

所以有

$$\begin{aligned}
p'(x) &= \sum_{j=0}^{n} f_j\ell_j'(x) \\
&= \sum_{j=0}^{n}\left(f_j\ell_j(x)\sum_{k=0,k\neq j}^{n}(x-x_k)^{-1}\right)
\end{aligned}$$

(b) 根据题设编写 matlab 代码进行实现

```
clear, clc, clf
LW = 'linewidth'; lw = 2;
%%
%Calculate all the 16 fj
n = 15;
x = linspace(-1, 1, n+1)';
F = @(x)sin(x);
f = F(x);
%%
%Calculate all the 1001 exact value
m = 1000;
xx = linspace(-1, 1, m+1)';
F_divide = @(x)cos(x);
exact = F_divide(xx);
```

```
%%
%Calculate all the 1001 approximate value
approx = zeros(m+1, 1);
for j = 1:n+1
    l = ones(m+1, 1);
    div_factor = zeros(m+1,1);
    for k = 1:j-1
        l = l.*(xx - x(k))/(x(j) - x(k));
        div_factor = div_factor + (1./(xx-x(k)));
    end
    for k = j+1:n+1
        l = l.*(xx - x(k))/(x(j) - x(k));
        div_factor = div_factor + (1./(xx-x(k)));
    end
    approx = approx + f(j)*l.*div_factor;
end
%%
%Calculate all the 1001 error
error = abs(exact - approx);
%%
figure(1);
subplot(2,1,1);
p1=plot(xx,exact,'k',LW,lw); hold on
p2=plot(xx,approx,'b--',LW,lw); hold on
legend([p1,p2], 'exact', 'approx');
subplot(2,1,2);
p3=semilogy(xx,error,'b',LW,lw); hold on
legend(p3, 'error','location','se');
```

运行得到图 4 结果

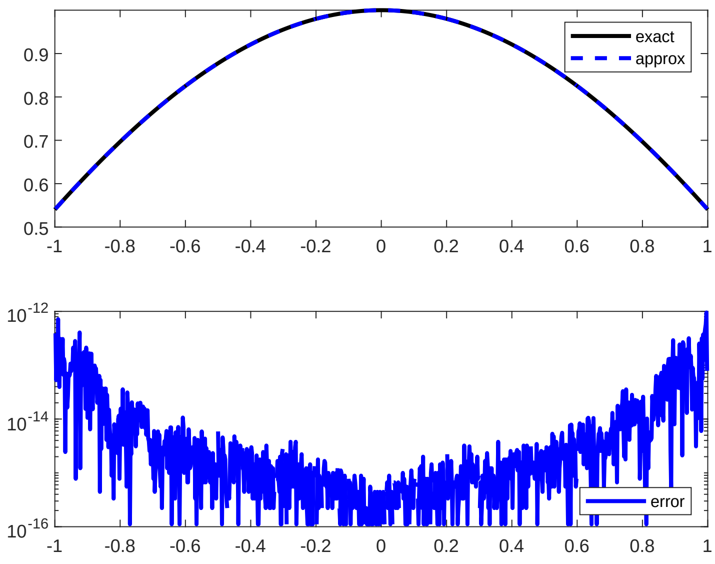

图 4: 第二题 (b) 小题作图结果

(c) 由题设分析得

$$P' = \begin{bmatrix} P'(x_0) \\ P'(x_1) \\ \vdots \\ P'(x_n) \end{bmatrix} = \begin{bmatrix} D_{00} & D_{01} & \dots & D_{0n} \\ D_{10} & D_{11} & \dots & D_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ D_{n0} & D_{n1} & \dots & D_{nn} \end{bmatrix} \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} = \begin{bmatrix} \sum_{j=0}^n f_j D_{0j} \\ \sum_{j=0}^n f_j D_{1j} \\ \vdots \\ \sum_{j=0}^n f_j D_{nj} \end{bmatrix}$$

对照式子

$$p'(x) = \sum_{j=0}^{n} \left(f_j \ell_j(x) \sum_{k=0,k\neq j}^{n} (x - x_k)^{-1} \right)$$

得到

$$D_{ij} = \ell_j(x_i) \sum_{k=0,k\neq j}^{n} (x - x_k)^{-1}$$

又由

$$\ell_j(x_k) = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases}$$

有

$$\begin{aligned} D_{ij} &= \ell_j(x_i) \sum_{k=0,k\neq j}^{n} (x - x_k)^{-1} \\ &= \begin{cases} 0 + 0 + \cdots + \ell_j(x_i)\dfrac{1}{x - x_i} + \cdots + 0 = \dfrac{1}{\pi_j} \displaystyle\prod_{k=0,k\neq i,j}^{n} (x_i - x_k) = \dfrac{\pi_i}{\pi_j(x_i - x_j)} & i \neq j \\ \displaystyle\sum_{k=0,k\neq j}^{n} (x_j - x_k)^{-1} & i = j \end{cases} \end{aligned}$$

(d) 由题意编写 matlab 代码如下

```
clear, clc, clf
LW = 'linewidth'; lw = 2;
F = @(x)sin(3.*x.^2);
F_diff = @(x)6.*x.*cos(3.*x.^2);
n = linspace(1,59,30);
max_error_U = zeros(1,30);
for i = 1:30
    max_error_U(i) = Diff_Matrix_U(n(i),F,F_diff);
end
max_error_C = zeros(1,30);
for i = 1:30
    max_error_C(i) = Diff_Matrix_C(n(i),F,F_diff);
end
%%
figure(1);
p1=semilogy(n,max_error_U,'k',LW,lw); hold on
p2=semilogy(n,max_error_C,'b--',LW,lw); hold on
legend([p1,p2], 'max error of Equidistant Points',
                'max error of Chebyshev Points',
                'location','se');
%%
function S = Diff_Matrix_U(n,F,F_diff)
LW = 'linewidth'; lw = 2;
```

```
%%
%Calculate all the n+1 fj and exact value
x = linspace(-1, 1, n+1)';
f = F(x);
exact = F_diff(x);
%%
%Calculate the D_(n+1)*(n+1)
D = zeros(n+1,n+1);
%Calculate the Pi(i)
Pi = ones(n+1);
for i = 1:n+1
    for k = 1:i-1
        Pi(i) = Pi(i) * (x(i) - x(k));
    end
    for k = i+1:n+1
        Pi(i) = Pi(i) * (x(i) - x(k));
    end
end
for i = 1:n+1
    %Calculate the D_ii
    D(i,i) = 0;
    for k = 1:i-1
        D(i,i) = D(i,i) + 1/(x(i) - x(k));
    end
    for k = i+1:n+1
        D(i,i) = D(i,i) + 1/(x(i) - x(k));
    end
    %Calculate the D_ij
    for j = 1:i-1
        D(i,j) = Pi(i) / (Pi(j) * (x(i) - x(j)));
    end
    for j = i+1:n+1
        D(i,j) = Pi(i) / (Pi(j) * (x(i) - x(j)));
    end
end
```

```
approx = D * f;
%%
%Calculate all the n+1 error
error = abs(exact - approx);
S = max(error);
end
%%
function S = Diff_Matrix_C(n,F,F_diff)
%%
%Calculate all the n+1 fj and exact value
n_set = linspace(0, n, n+1)';
Chebyshev = @(x)cos(x.* pi./ n);
x = Chebyshev(n_set);
f = F(x);
exact = F_diff(x);
%%
%Calculate the D_(n+1)*(n+1)
D = zeros(n+1,n+1);
%Calculate the Pi(i)
Pi = ones(n+1);
for i = 1:n+1
    for k = 1:i-1
        Pi(i) = Pi(i) * (x(i) - x(k));
    end
    for k = i+1:n+1
        Pi(i) = Pi(i) * (x(i) - x(k));
    end
end
for i = 1:n+1
    %Calculate the D_ii
    D(i,i) = 0;
    for k = 1:i-1
        D(i,i) = D(i,i) + 1/(x(i) - x(k));
    end
    for k = i+1:n+1
```

```
            D(i,i) = D(i,i) + 1/(x(i) - x(k));
        end
        %Calculate the D_ij
        for j = 1:i-1
            D(i,j) = Pi(i) / (Pi(j) * (x(i) - x(j)));
        end
        for j = i+1:n+1
            D(i,j) = Pi(i) / (Pi(j) * (x(i) - x(j)));
        end
    end
    approx = D * f;
    %%
    %Calculate all the n+1 error
    error = abs(exact - approx);
    S = max(error);
    end
```

运行得到图 5 结果 可以看到按照等距点插值的做法在点数增多到一定程度之后

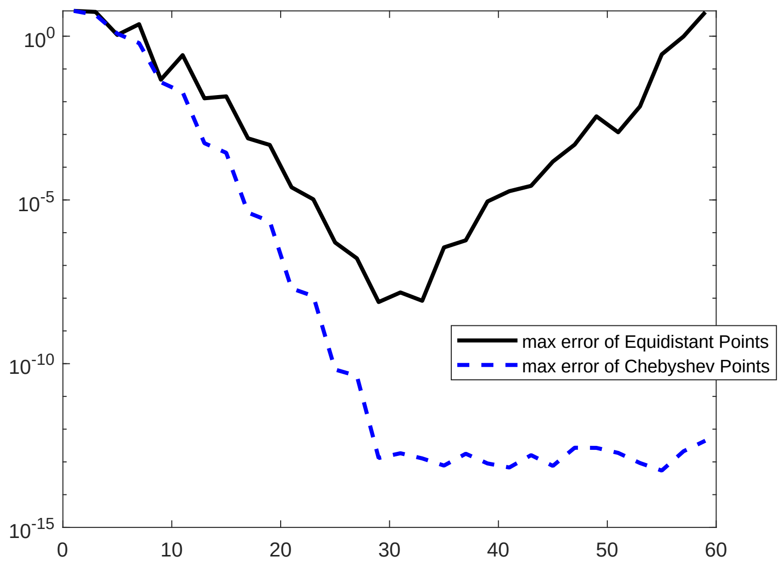

图 5: 第二题 (d) 小题作图结果

反倒其最大误差会急剧增大，而 Chebyshev 点不仅在等插值点数情况下其最大误差普遍更小，而且随着点数增多，即使接近 50 之前，其最大误差也尚未像等距插

值一样大幅回升。

其原因在于等距点插值随着插值点数的增多，Runge 现象在高阶情形下影响逐渐凸显，使得拟合的数值在拟合区间边缘出现较大的偏差，后者采用的是高阶而非等距的 Chebyshev 点，通过精心选取一系列在区间中心稀疏而在边缘密集的插值点，减少了 Runge 现象的影响，于是可以看到图中即使在 n 到达 30 以上的值后，依然尚为出现明显的 Runge 现象，即如等距插值那般出现由于区间边缘误差突然增长而最大误差值陡然回升的情况。

第三题

(a) 积分区间 $[x_{n-1}, x_{n+1}]$，积分结点 $\{x_{n+1}, x_n, x_{n-1}, x_{n-2}\}$
记 $\alpha = ha_0, \beta = ha_1, \gamma = ha_2, \mu = ha_3$，于是有

$$a_0 h = \int_{x_{n-1}}^{x_{n+1}} \frac{(x-x_n)(x-x_{n-1})(x-x_{n-2})}{(x_{n+1}-x_n)(x_{n+1}-x_{n-1})(x_{n+1}-x_{n-2})}\,dx = \frac{1}{3}h$$

同理得,$a_1 h = \frac{4}{3}h$，$a_2 h = \frac{1}{3}h$，$a_3 h = 0$。
故得到公式

$$y_{n+1} = y_{n-1} + \frac{h}{3}(f_{n+1} + 4f_n + f_{n-1} + 0f_{n-2})$$

(b) 展开得

$$\begin{aligned} y_{n+1} &= y(x_{n-1}) + \frac{h}{3}(f(x_{n+1}, y(x_{n+1})) + 4f(x_n, y(x_n)) + f(x_{n-1}, y(x_{n-1}))) \\ &= y(x_{n-1}) + \frac{h}{3}(y'(x_{n+1}) + 4y'(x_n) + y'(x_{n-1})) \end{aligned}$$

比较得局部截断误差，

$$T_{n+1} = \int_{x_{n-1}}^{x_{n+1}} \frac{y^{(5)}(\xi)}{5!}(x-x_{n+1})(x-x_n)(x-x_{n-1})(x-x_{n-2})dx = O(h^5)$$

故公式为 4 阶

(c) 由前面所得公式结合题目函数，直接得到其递推式

$$y_{n+1} = (y_{n-1} + \frac{h}{3}(x_{n-1}e^{-5x_{n-1}} + 4x_n e^{-5x_n} + x_{n+1}e^{-5x_{n+1}} - 5y_{n-1} - 20y_n))\Big/(1 + \frac{5h}{3})$$

显然，我们所使用的多步法本身就是四阶的，为了不影响格式的整体截断误差，起步计算的格式精度至多只能比该格式的精度低一阶，即至少需要用三阶以上精度的格式来计算，而三阶的 RungeKutta 方法作为三阶格式恰满足其要求。
籍此可书写代码进行计算

```matlab
clear ,clc , clf
LW = 'linewidth';lw = 2;

% Def of the problem
n = 100;
F = @(x)(x.^2).*exp(-5.*x)/2;
a = 0;b = 2;

% Init Case
x = linspace(a,b,n+1);
y = zeros(1,n+1);
h = (b - a) / n;

% for error
f = F(x);

%%
% RK-3 Calculate
y(1) = 0;
y(2) = RK_3(h,x(1),y(1));
y(3) = RK_3(h,x(2),y(2));
%%
% Multi-Step Calculate
for i = 3:n
    y(i+1) = (y(i-1) + h*(x(i-1)*exp(-5*x(i-1)) +
            4*x(i)*exp(-5*x(i)) + x(i+1)*exp(-5*x(i+1)) -
            5*y(i-1) - 20*y(i)) / 3) /(1+5*h/3);
end
figure(1);
subplot(2,1,1);
p1 = plot(x,y,'b--',LW,lw);hold on
p2 = plot(x,f,'k',LW,lw);
legend([p1,p2],'approx','exact');
subplot(2,1,2);
p3 = plot(x,abs(f-y),'k',LW,lw);
```

```
legend(p3,'error','location','se');
%%
function y_next = RK_3(h,x_curr,y_curr)
    K1 = x_curr*exp(-5*x_curr)-5*y_curr;
    K2 = (x_curr + h/2)*exp(-5*(x_curr + h/2))-
            5*(y_curr + h*K1/2);
    K3 = (x_curr + h)*exp(-5*(x_curr + h))-
            5*(y_curr - h*K1 + 2*h*K2);
    y_next = y_curr + h*(K1+4*K2+K3)/6;
end
```

运行得到图 6 结果

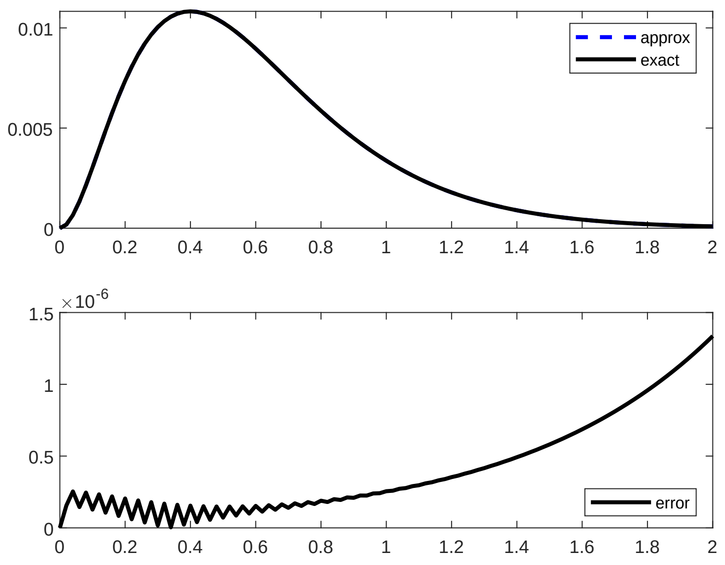

图 6: 第三题 (c) 小题作图结果

(d) 解微分方程得其解为

$$y = x^2 \frac{e^{-5x}}{2}$$

据此编写代码计算得

```
clear ,clc , clf
LW = 'linewidth';lw = 2;

% Def of the problem
```

```
F = @(x)(x.^2).*exp(-5.*x)/2;
a = 0;b = 2;

error_inf = 1e2; error_sup = 1e4;
error = zeros(1,error_sup)
for n = error_inf:error_sup
    error(n) = Multi_Step(F,a,b,n);
end
figure(1);
p1 = loglog((error_inf:error_sup),
        error(error_inf:error_sup),'k');
legend(p1,'error','location','se');
xlabel('n');
ylabel('误差');
%%
function S = Multi_Step(F,a,b,n)
% Init Case
    x = linspace(a,b,n+1);
    y = zeros(1,n+1);
    h = (b - a) / n;

% for error
    f = F(x);

%%
% RK-3 Calculate
    y(1) = 0;
    y(2) = RK_3(h,x(1),y(1));
    y(3) = RK_3(h,x(2),y(2));
%%
% Multi-Step Calculate
    for i = 3:n
        y(i+1) = (y(i-1) + h*(x(i-1)*exp(-5*x(i-1)) +
                4*x(i)*exp(-5*x(i)) + x(i+1)*exp(-5*x(i+1)) -
                5*y(i-1) - 20*y(i)) / 3) /(1+5*h/3);
    end
%%
% Output
    S = max(abs(f-y));
end
%%
function y_next = RK_3(h,x_curr,y_curr)
    K1 = x_curr*exp(-5*x_curr)-5*y_curr;
    K2 = (x_curr + h/2)*exp(-5*(x_curr + h/2))-
            5*(y_curr + h*K1/2);
    K3 = (x_curr + h)*exp(-5*(x_curr + h))-
            5*(y_curr - h*K1 + 2*h*K2);
    y_next = y_curr + h*(K1+4*K2+K3)/6;
end
```

运行得到图 7 结果 从图中不难看出其斜率为 4，验证了先前对其公式为 4 阶的推

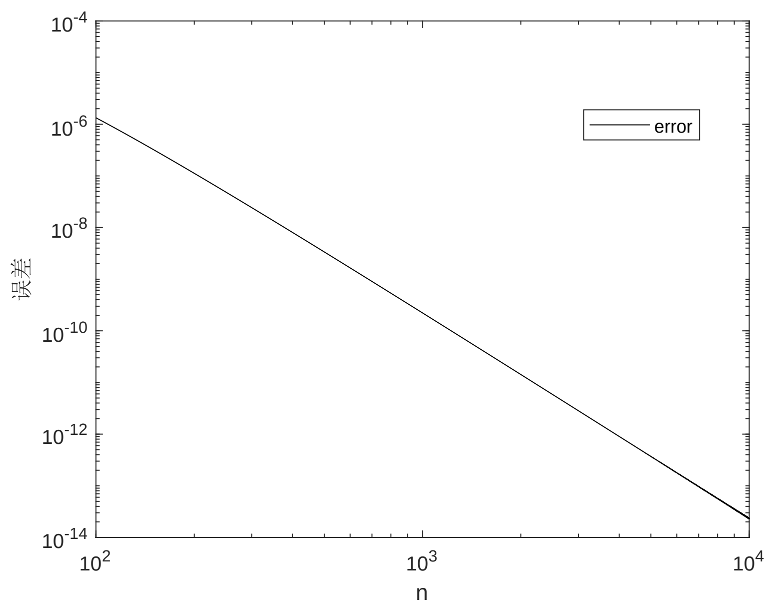

图 7: 第三题 (d) 小题作图结果

论。