MATLAB 第 6 次作业参考答案

几点反馈：

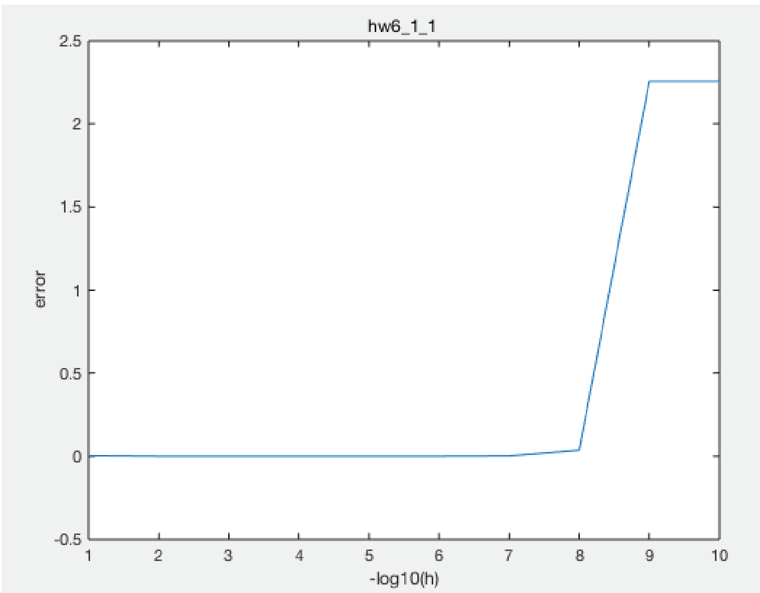
- 作业的参考答案来自同学们的作业，只是稍作调整，为保护隐私，隐去姓名。
- 若是大家发现布置的作业中有明显错误或者表达有歧义，希望可以及时反馈，我们也会及时在群中进行说明。
- 对于参考文献新方法的阅读还需要更细致一点。另外，几乎所有的算法大家都可以用课件或者参考书中的例子来验证，这也算是最简单的检查方法。

第一题

示例 1：

h	f^2(x)	r
0.1000000000	-2.2523280880	0.0033214952
0.0100000000	-2.2556163525	0.0000332307
0.0010000000	-2.2556492509	0.0000003322
0.0001000000	-2.2556495782	0.0000000049
0.0000100000	-2.2556501111	-0.0000005280
0.0000010000	-2.2556401191	0.0000094640
0.0000001000	-2.2537527400	0.0018968432
0.0000000100	-2.2204460493	0.0352035339
0.0000000010	0.0000000000	2.2556495832
0.0000000001	0.0000000000	2.2556495832

计算机误差也受差分公式中加减的先后顺序所影响（感兴趣的同学可以尝试一下），这一题体现在有的同学最后一个 h 算出来误差极大，也是判为正确的情况。




```
hw6_1_1.m
format long;
a = pi/3;
syms f x;
f = x*cos(x);
g = @(a)double(subs(f,{x},{a}));
```

```
f_2 = diff(f,x,2);
value = double(subs(f_2,{x},{a}));
h = zeros(1,10);
h(1) = 1e-1;
for i = [2:1:10]
    h(i) = h(i-1)/10;
end
f_dx2 = (g(a-h)-2*g(a)+g(a+h))./h.^2;
r = f_dx2-value;
text = [h;f_dx2;r];
fid = fopen('table.txt','w');
fprintf(fid,'h                    f^2(x)              r\n');
fprintf(fid,'%.10f %.10f %.10f\n',text);
fclose(fid);
u = -log10(h);
plot(u,r);
xlabel('-log10(h)');
ylabel('error');
title('hw6\_1\_1');
```

```
>> hw6_1_2
result is : -2.2556495830>>
```

hw6_1_2.m

```
h = 1e-3;
x = pi/3;
f = @(x)x*cos(x);
h2_x = (f(x-h)-2*f(x)+f(x+h))/h^2;
h2_x_ = (f(x-h/2)-2*f(x)+f(x+h/2))/(h/2)^2;
h2_x__ = (f(x-h/4)-2*f(x)+f(x+h/4))/(h/4)^2;
h4_x = (4*h2_x_ - h2_x)/3;
h4_x_ = (4*h2_x__ - h2_x_)/3;
h6_x = (2^4*h4_x_ - h4_x)/(2^4-1);
fprintf('result is : %.10f',h6_x);
```

两步外推，很多人只外推了一步！另外注意外推公式中的系数的变化！建议好好回顾一下，自己推导一下加深印象。示例 2 的这一部分更加清楚。

```
>> hw6_1_3
result is : -0.5931000135>>
```

hw6_1_3.m

```
f = @(x)x*cos(x);
x = pi/3;
h = 1e-3;
f_dx3 = (f(x+2*h)-f(x-2*h)-2*f(x+h)+2*f(x-h))/(2*h^3);
fprintf('result is : %.10f',f_dx3);
```

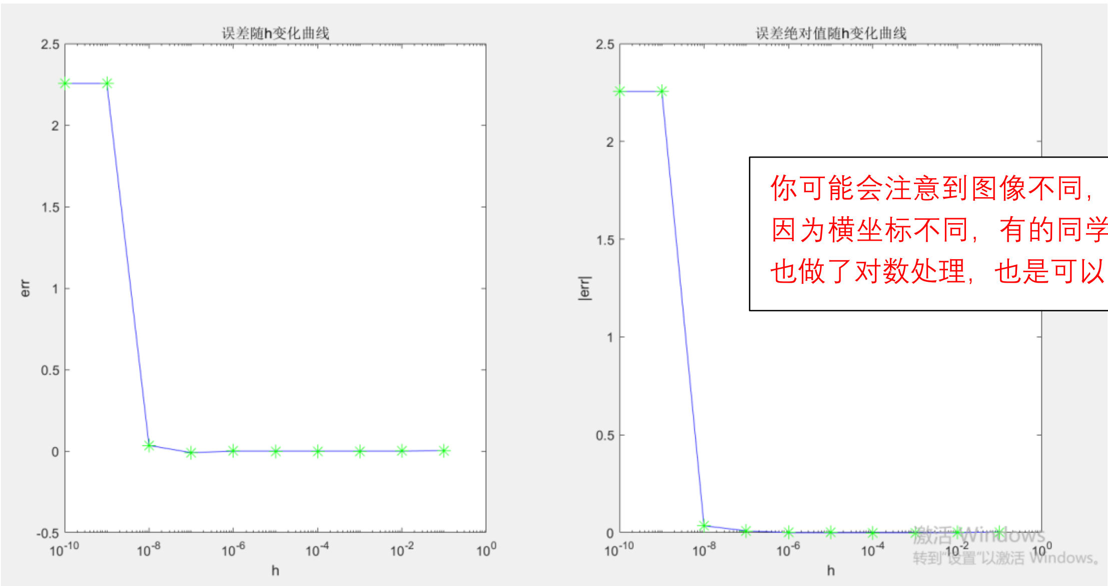
五点中心差分，注意增加的两点可为加减 2h 或者加减 h/2，公式中的系数有些许不同，两个示例刚好是两个不同的公式。

示例 2：

（1）运用 $f''(x) \approx \dfrac{f(x+h)+f(x-h)-2f(x)}{h^2}$ 公式计算 $f''(\pi/3)$，在不同的 h 取值下，输出二阶导数值和误差值如下：

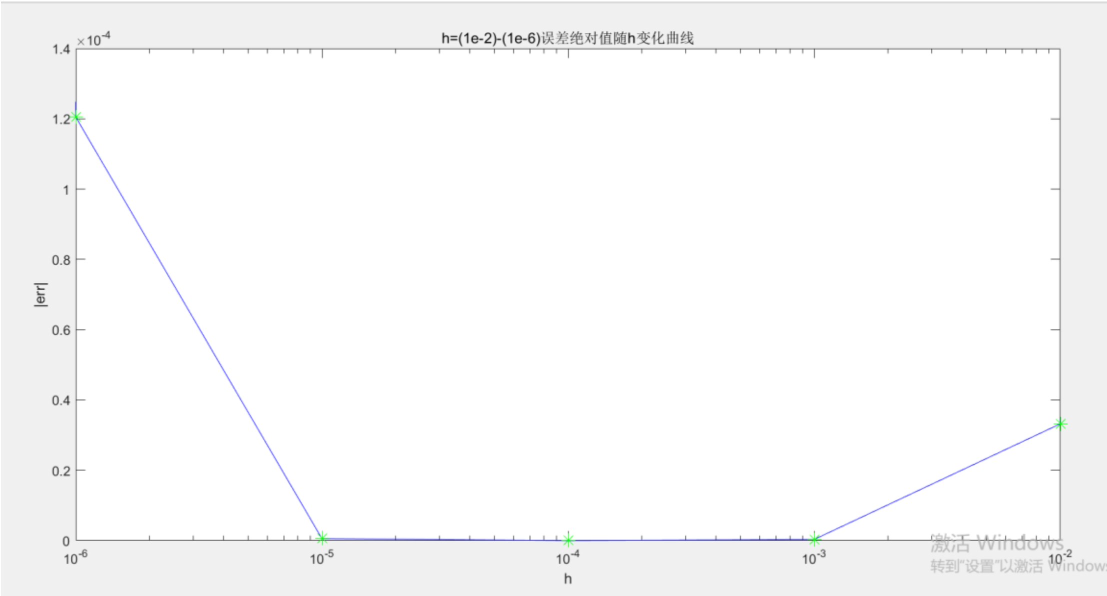
0.1000000000	-2.2523280880	0.0033214952
0.0100000000	-2.2556163525	0.0000332307
0.0010000000	-2.2556492509	0.0000003322
0.0001000000	-2.2556495782	0.0000000049
0.0000100000	-2.2556490009	0.0000005823
0.0000010000	-2.2555290968	0.0001204863
0.0000001000	-2.2648549702	-0.0092053871
0.0000000100	-2.2204460493	0.0352035339
0.0000000010	0.0000000000	2.2556495832
0.0000000001	0.0000000000	2.2556495832

由表格可以观察到，当 h = 10^{-1}-10^{-4} 时，随 h 以 1/10 的比例减小，二阶导数误差以 1/100 的比例减小，当 h>10^{-4} 时，随 h 减小，误差反而增大，当 h=10^{-9} 和 10^{-10} 时，导数值计算为 0 了，是错误的。绘出误差值和误差绝对值随 h 变化曲线如下：



你可能会注意到图像不同，其实是因为横坐标不同，有的同学纵坐标也做了对数处理，也是可以的。

因为出现 h=10^{-9} 和 10^{-10} 这两个误差极大的点，导致坐标轴刻度过大，不便于对误差随 h 变化趋势的观察，h=10^{-6}-10^{-2} 这个区段特征趋势比较明显，截取这段放大进行观察，图像如下：



由图可以看出，|err|随 h 变化曲线酷似碗口形，在 $h = 10^{-4}$ 处取得极小值。

代码：hw6_1_1.m

```
h = logspace(-1,-10,10);
h = h';
x = pi/3;
f = @(x) x.*cos(x);
d2 = tr_c_d(f,x,h);
d_prc = -2.*sin(x) - x.*cos(x);
err = d2 - d_prc;
fid = fopen('table.txt','w');
for i = 1:10
    fprintf(fid,'%1.10f    %1.10f    %1.10f\n',h(i),d2(i),err(i));
end
fclose(fid);
figure;
subplot(1,2,1);
semilogx(h,err,'b');
hold on;
semilogx(h,err,'g*','MarkerSize',10);
xlabel('h');
ylabel('err');
title('误差随 h 变化曲线');
subplot(1,2,2);
semilogx(h,abs(err),'b');
hold on;
semilogx(h,abs(err),'g*','MarkerSize',10);
xlabel('h');
ylabel('|err|');
```

```
title('误差绝对值随 h 变化曲线');
figure;
semilogx(h,abs(err),'b');
hold on;
semilogx(h,abs(err),'g*','MarkerSize',10);
xlabel('h');
ylabel('|err|');
set(gca,'XLim',[h(6) h(2)]);
title('h=(1e-2)-(1e-6)误差绝对值随 h 变化曲线');

function d2 = tr_c_d(f,x,h)
%本函数用三点中心差分计算 f 在 x 处的二阶导数值，步长为 h
d2 = (f(x+h)+f(x-h)-2*f(x))./h.^2;
end
```

（2）输出结果如下：

```
F =

  -2.255649250937353                   0                   0
  -2.255649500071399  -2.255649583116082                   0
  -2.255649562243889  -2.255649582968052  -2.255649582958184
```

对角线上元素分别为 $F_2(h)$,$F_4(h)$,$F_6(h)$。

代码：hw6_1_2.m

```
f = @(x) x.*cos(x);
x = pi/3;
h = 1e-3;
Fh = @(x,h) (f(x+h)+f(x-h)-2*f(x))/(h^2);
%构建 F 表，因为两步外推，3*3 的表就足够
F = zeros(3,3);
%F2(h)
F(1,1) = Fh(x,h);
%一步外推，F4(h)
F(2,1) = Fh(x,h/2);
F(2,2) = (2^2*F(2,1)-F(1,1))/(2^2 - 1);
%两步外推,F6(h)
F(3,1) = Fh(x,h/4);
F(3,2) = (2^2*F(3,1)-F(2,1))/(2^2-1);
F(3,3) = (2^4*F(3,2)-F(2,2))/(2^4-1);
format long;
display(F);
```

（3）进行推导

$$f(x+h)=f(x)+f^{'}(x)h+\frac{1}{2}f^{''}(x)h^2+\frac{1}{6}f^{'''}(x)h^3+\frac{1}{24}f^{''''}(c_1)h^4 \qquad (1)$$

$$f(x-h)=f(x)-f^{'}(x)h+\frac{1}{2}f^{''}(x)h^2-\frac{1}{6}f^{'''}(x)h^3+\frac{1}{24}f^{''''}(c_2)h^4 \qquad (2)$$

$$f(x+\frac{h}{2})=f(x)+\frac{1}{2}f^{'}(x)h+\frac{1}{8}f^{''}(x)h^2+\frac{1}{48}f^{'''}(x)h^3+\frac{1}{384}f^{''''}(c_3)h^4 \qquad (3)$$

$$f(x-\frac{h}{2})=f(x)-\frac{1}{2}f^{'}(x)h+\frac{1}{8}f^{''}(x)h^2-\frac{1}{48}f^{'''}(x)h^3+\frac{1}{384}f^{''''}(c_4)h^4 \qquad (4)$$

(1) (2)两式和(3)(4)两式分别相减得

$$f(x+h)-f(x-h)=2f^{'}(x)h+\frac{1}{3}f^{'''}(x)h^3+\frac{1}{24}f^{''''}(c_1)h^4+\frac{1}{24}f^{''''}(c_2)h^4 \qquad (5)$$

$$f(x+\frac{h}{2})-f(x-\frac{h}{2})=f^{'}(x)h+\frac{1}{24}f^{'''}(x)h^3+\frac{1}{384}f^{''''}(c_3)h^4+\frac{1}{384}f^{''''}(c_4)h^4 \qquad (6)$$

(6)式乘(-2)加(5)式可得

$$f(x+h)-f(x-h)-2f(x+\frac{h}{2})+2f(x-\frac{h}{2})=\frac{1}{4}f^{'''}(x)h^3+O(h^4)$$

推得 $$f^{'''}(x)=4(f(x+h)-f(x-h)-2f(x+\frac{h}{2})+2f(x-\frac{h}{2}))/h^3$$

仍取 h = 10^{-3} 编程计算，输出结果为：f’’’(π/3)= -0.5931.

代码：hw6_1_3.m

```
f = @(x) x.*cos(x);
h = 1e-3;
x = pi/3;
d_3 = 4*(f(x+h)-f(x-h)-2*f(x+h/2)+2*f(x-h/2))/h^3;
display(d_3);
```

第二题

示例 1：

```
>> f = @(x)x/(4+x.^2)

f =

  包含以下值的 function_handle:

    @(x)x/(4+x.^2)

>> hw6_2_Lagrange_int(f,0,1,10)

ans =

   0.111571775452973
```

hw6_2_Lagrange_int.m

```
function value = hw6_2_Lagrange_int(f,a,b,num)
u = linspace(a,b,num);
v = arrayfun(f,u);
value = 0;
p_check = 0;
for i = [1:1:num]
    k = v(i);
    for j = [1:1:num]
        if i~=j
            k = k/(u(i)-u(j));
            if p_check == 0
                p = poly(u(j));
                p_check = 1;
            else
                p = conv(p,poly(u(j)));
            end
        end
    end
    p_check = 0;
    p = p*k;
    for j = [1:1:num]
        temp = b.^j/j-a.^j/j;
        value = value + p(num+1-j).*temp;
    end
end
```

多项式的积分，其实动手算一下就会发现其实知道很简单，只是降次操作和系数变化而已，示例 2 这里的计算更加清楚简明。

示例 2：

w6_2.m

```
clc;clear;
interval=[0,1];
n=10;
f=@(x)x./(4+x.^2);
I = Lquadrature(f,interval,n)
fprintf('定积分值为: I = %f\n',I);
function I = Lquadrature(fun,interval,n)
a=interval(1);
b=interval(2);
x=linspace(a,b,n);
y=fun(x);
p=zeros(1,n);
for k=1:n
    den=1;
    num=[zeros(1,n-1),1];
    for j=1:n
        if j~=k
            den=den*(x(k)-x(j));
            num=conv(num,poly(x(j)),'same');
        end
    end
    p=p+y(k)*num/den;
end
% 多项式解析积分
p1=[p,0];
v=[n:-1:1,1];
p2=p1./v;
I=polyval(p2,b)-polyval(p2,a);
end
```

看这里

```
定积分值为: I = 0.111572
```

第三题

示例 1：

(1)

Hw6_3_1.m

```
clc;clear;
m1=5;m2=10;
a=1;b=2;
f=@(x)x./(log(1+x));
I1=cTrapQuad(f,a,b,m1);
fprintf('m = %d  时,复合梯形法结果为:\n I = %.15f\n\n',m1,I1);
I2=cTrapQuad(f,a,b,m2);
fprintf('m = %d  时,复合梯形法结果为:\n I = %.15f\n\n',m2,I2);
I3=cSimpsonQuad(f,a,b,m1);
fprintf('m = %d  时,复合 Simpson 法结果为:\n I = %.15f\n\n',m1,I3);
I4=cSimpsonQuad(f,a,b,m2);
fprintf('m = %d  时,复合 Simpson 法结果为:\n I = %.15f\n\n',m2,I4);
Iquad=quad(f,1,2);
fprintf('quad 的求积结果为:\n I = %.15f\n\n',Iquad);
syms x
fs=x./(log(1+x));
Iint=double(int(fs,x,1,2));
fprintf('int 的求积结果为:\n I = %.15f\n\n',Iint);
```

cTrapQuad.m

```
function I=cTrapQuad(fun,a,b,m)
%  梯形格式
h=(b-a)/m;
I=h/2*(fun(a)+fun(b)+2*sum(fun(a+h:h:b-h)));
End
```

cSimpsonQuad.m

```
function I=cSimpsonQuad(fun,a,b,m)
% Simpson 格式
h=(b-a)/(2*m);
I=h/3*(fun(a)+fun(b)+4*sum(fun(a+h:2*h:b-h))+2*sum(fun(a+2*h:2*h:b-2*h)));
End
```

```
m = 5 时,复合梯形法结果为:
 I = 1.635080810481285

m = 10 时,复合梯形法结果为:
 I = 1.635191073518018

m = 5 时,复合Simpson法结果为:
 I = 1.635227827863595

m = 10 时,复合Simpson法结果为:
 I = 1.635227842275379

quad的求积结果为:
 I = 1.635227843091316

int的求积结果为:
 I = 1.635227843238810
```

（2）

使用闭区间积分格式会包含被积函数奇异点，造成输出为NaN，可以采用复合 midpoint rule，也可对积分下限 x=0 单独定义，这里采用复合 midpoint rule 求积。

单独定义或者用复合 midpoint rule
示例 1：复合 midpoint rule
示例 2：单独定义

```
clc;clear;
a=0;
b=2;
m=10;
f=@(x)x./(log(1+x));
Imq=cMidpointQuad(f,a,b,m);
fprintf('m = %d  时, composite-midpoint rule 结果为:\n I = %.15f\n\n',m,Imq);
```

```
cMidpointQuad.m
function I=cMidpointQuad(fun,a,b,m)
% 矩形格式
h=(b-a)/m;
w=(a:h:b-h)+h/2;
I=h*sum(fun(w));
End
```

```
m = 10 时,composite-midpoint rule结果为:
 I = 2.864738389824478
```

```
 使用复合辛普森方法结果: 2.864502
使用内置函数integral结果:2.864502
```

示例 2：

（1）输出结果如下：(其中 int 做符号运算，给出结果为表达式，ans 为转为 double 的结果）

```
ctr_5 =

   1.635080810481285


ctr_10 =

   1.635191073518018


csr_5 =

   1.635227827863595


csr_10 =

   1.635227842275379
```

```
MATLAB内置函数运算结果为

int_by_int =

int(t/log(t + 1), t, 1, 2)


ans =

   1.635227843238810


int_by_quad =

   1.635227843091316
```

由输出结果可以看出，在 m 相同的情况下，Composite Simposon’s Rule 的运算结果优于 Composite Trapezoid Rule，在用相同的运算规则时，m 越大，结果越精确。观察两个 MATLAB 内置函数的输出，发现结果也不一样，原因是 int 是符号运算，而 quad 采用的是数值积分。

代码：hw6_3_1.m

```
f = @(x) (x./log(1+x));
a = 1; b = 2;
format long;
ctr_5 = CTR(f,a,b,5)
ctr_10 = CTR(f,a,b,10)
csr_5 = CSR(f,a,b,5)
csr_10 = CSR(f,a,b,10)
display('MATLAB 内置函数运算结果为');
syms t;
y = t./log(1+t);
int_by_int = int(y,a,b)
double(int_by_int)
int_by_quad = quad(f,a,b)

function I = CTR(f,a,b,m)
%本函数用 Composite Trapezoid Rule 求定积分
x = zeros(1,m+1);
y = zeros(1,m+1);
h = (b-a)/m;
for i = 1:m+1
    x(i) = a+(i-1)*h;
    y(i) = f(x(i));
end
s = y(1)+y(end);
for i = 2:m
```

```
    s = s + 2*y(i);
end
I = s*h/2;
end

function I = CSR(f,a,b,m)
%本函数用 Composite Simposon's Rule 求定积分
x = zeros(1,2*m+1);
y = zeros(1,2*m+1);
h = (b-a)/(2*m);
for i = 1:2*m+1
    x(i) = a+(i-1)*h;
    y(i) = f(x(i));
end
s = y(1)+y(end);
for i = 1:m
    s = s+4*y(2*i)+2*y(2*i+1);
end
s = s-2*y(end);
I = s*h/3;
end
```

（2）用 Composite Simposon’s Rule 计算。端点 x=0 为特殊点，用洛必达公式，x = 0 时，y=1, 特殊处理后，运行结果为：I = 2.864502.

代码：hw6_3_2.m

```
f = @(x) x./log(x+1);
a = 0; b = 2; m = 10;
x = zeros(1,2*m+1);
y = zeros(1,2*m+1);
h = (b-a)/(2*m);
x(1) = a;
y(1) = 1;
for i = 2:2*m+1
    x(i) = a+(i-1)*h;
    y(i) = f(x(i));
end
s = y(1)+y(end);
for i = 1:m
    s = s+4*y(2*i)+2*y(2*i+1);
end
s = s-2*y(end);
I = s*h/3;
fprintf('I = %1.6f\n',I);
```

第四题

示例 1：

（1）运算结果为：

```
R =

   14.230249470757707                    0                    0                    0                    0                    0
   11.171369920275840   10.151743403448551                    0                    0                    0                    0
   10.443796845270651   10.201272486935588   10.204574425834723                    0                    0                    0
   10.266367183870944   10.207223963404376   10.207620728502295   10.207669082512892                    0                    0
   10.222270190753514   10.207571193047704   10.207594341690593   10.207593922852311   10.207593628108544                    0
   10.211260730405375   10.207590910289328   10.207592224772103   10.207592191170223   10.207592184379314   10.207592182968042
```

代码： romberg.m

```
function R = romberg(f,a,b)
%本函数用于求 Romberg Integration，并打印 romberg table
eps = 1e-5;
R(1,1) = (b-a)*(f(a)+f(b))/2;
h = b-a;
j = 1;
while 1
    j = j+1;
    h = h/2
    R(j,1) = R(j-1,1)/2;
    for i = 1:2^(j-2)
        R(j,1) = R(j,1)+ h*f(a+(2*i-1)*h);
    end
    for k = 2:j
        R(j,k) = (4^(k-1)*R(j,k-1) - R(j-1,k-1))/(4^(k-1) - 1);
    end
    if abs(R(j-1,j-1)-R(j,j))<eps
        break;
    end
end
display('romberg table');
display(R);
end
```

R 表的公式建议大家还是要自己从头推一遍，明白其原理，并与上面的外推法对比一下

校验代码:hw6_4_1.m

```
f = @(x) x.*sqrt(1+x.^2);
R = romberg(f,0,3);
```

示例 2：

(1)

Hw6_4.m

```
clc;clear;
a=0;b=3;
TOL=1e-5;
f=@(x)x.*sqrt(1+x.^2);
Tableau=Romberg(f,a,b,TOL);
disp('RombergTable:');disp(Tableau);

function Tableau=Romberg(fun,a,b,TOL)
R=zeros(15); % 初始化表格，给一个较大的空间
h=b-a;
R(1,1)=h/2*(fun(a)+fun(b));
j=1;
R0=0;
R1=R(1,1);
while abs(R1-R0)>TOL
    R0=R1;
    j=j+1;
    h=h/2;
    R(j,1)=R(j-1,1)/2+h*sum(fun(a+(2*(1:1:2^(j-2))-1)*h));
    for k=2:j
        R(j,k)=(4^(k-1)*R(j,k-1)-R(j-1,k-1))/(4^(k-1)-1);
        R1=R(k,k);
    end
end
Tableau=R(1:j,1:j);
end
```

```
RombergTable:
   14.2302         0         0         0         0         0
   11.1714   10.1517         0         0         0         0
   10.4438   10.2013   10.2046         0         0         0
   10.2664   10.2072   10.2076   10.2077         0         0
   10.2223   10.2076   10.2076   10.2076   10.2076         0
   10.2113   10.2076   10.2076   10.2076   10.2076   10.2076
```

(2)

龙贝格积分使用 Richardson 外推公式对复合梯形求积法中的梯形积分格式进行外推，从而提高误差项关于 h 的阶数，降低积分误差；同时，龙贝格积分通过外推表，充分使用前期计算信息，能够加速求积过程

第五题

示例 1：

取 h_1, h_2 均为 0.1 编程计算，二重积分计算结果为:　I = 0.796599608744471。

这一题出题的时候公式打错了，这是我们的疏忽，很抱歉。但是很多同学自己改正了，希望以后还有这种情况大家可以向助教反馈一下。
这一题直接套公式，比较简单。

代码：hw6_5.m

```
f = @(x,y) exp(-x.*y);
I = B_CSR(f,0,1,0,1);
display('二重积分计算结果为:');
display(I);


function I = B_CSR(f,a,b,c,d)
%本文件用 Composite Simposon's Rule 计算二重积分
m = 10;
n = 10;
h1 = (b-a)/m;
h2 = (d-c)/n;
x = 0:m;
y = 0:n;
x = a + x*h1;
y = c + y*h2;
I = 0;
for i = 1:m
    for j = 1:n
        x_mid = (x(i) + x(i+1))/2;
        y_mid = (y(j) + y(j+1))/2;
        I = I+h1*h2/36*(f(x(i),y(j)) + f(x(i+1),y(j)) + f(x(i),y(j+1)) + f(x(i+1),y(j+1))...
            + 4*(f(x_mid,y(j)) + f(x(i),y_mid) + f(x(i+1),y_mid) + f(x_mid,y(j+1)))...
            + 16*f(x_mid,y_mid));
    end
end
end
```

示例 2：

```
>> hw6_5
result is :0.796600>>
```

hw6_5.m

```
f = @(x,y)exp(-x*y);
m = 100;
n = 100;
x = linspace(0,1,m+1);
y = linspace(0,1,n+1);
h1 = 1/m;
h2 = 1/n;
result = 0;
for j = [1:1:n]
    for i = [1:1:m]
        temp1 = f(x(i),y(j))+f(x(i+1),y(j))+f(x(i),y(j+1))+f(x(i+1),y(j+1));
        temp2 = f((x(i)+x(i+1))/2,y(j))+f(x(i),(y(j)+y(j+1))/2);
        temp3 = f((x(i)+x(i+1))/2,y(j+1))+f(x(i+1),(y(j)+y(j+1))/2);
        temp4 = f((x(i)+x(i+1))/2,(y(j)+y(j+1))/2);
        temp = h1*h2*(temp1+4*(temp2+temp3)+16*temp4)/36;
        result = result + temp;
    end
end
fprintf('result is :%.6f',result);
```

将公式拆分，增加可读性，debug 也更容易修改，何乐而不为！

第六题

示例 1：

(1)

Adap_trap.m

```
function result = adap_trap(f,startpoint,endpoint)

result = (endpoint - startpoint) / 2 * (f(startpoint) + f(endpoint));
```

hw6_6_adaptive_trap.m

```
f = @(x) exp(-x^2);

n = 1; %现存数量
m = 0; %共分段数量

tol = 1e-8;
startpoint = 0;
endpoint = 3;
a(1) = startpoint;
b(1) = endpoint;
sum = 0;

while n > 0
    add = adap_trap(f,a(n),b(n));
    c = (a(n) + b(n)) / 2;
    add_small = adap_trap(f,a(n),c) + adap_trap(f,c,b(n));
    tol_now = tol / (startpoint - endpoint) * (a(n) - b(n));
    if abs(add_small - add) > 3 * tol_now
        a(n+1) = c;
        b(n+1) = b(n);
        b(n) = c;
        n = n+1;
    else
        sum = sum + add_small;
        n = n - 1;
        m = m + 2;
    end
end
```

子区间怎么定义，怎么数？
误差限怎么变化？
请大家再回顾一下参考书，书中也有例程可以学习。细致！细致！

运行结果：

sum = 0.99997791

m =14016

（2）

Adap_simpson.m

```
function result = adap_simpson(f,startpoint,endpoint)

result = (endpoint - startpoint) / 6 * (f(startpoint) + 4 * f((startpoint+endpoint)/2) + f(endpoint));
```

Hw6_6_adap_simpson.m

```
f = @(x) exp(-x^2);

n = 1; %现存数量
m = 0; %共分段数量

tol = 1e-8;
startpoint = 0;
endpoint = 3;
a(1) = startpoint;
b(1) = endpoint;
sum = 0;

while n > 0
    add = adap_simpson(f,a(n),b(n));
    c = (a(n) + b(n)) / 2;
    add_small = adap_simpson(f,a(n),c) + adap_simpson(f,c,b(n));
    tol_now = tol / (startpoint - endpoint) * (a(n) - b(n));
    if abs(add_small - add) > 10 * tol_now
        a(n+1) = c;
        b(n+1) = b(n);
        b(n) = c;
        n = n+1;
    else
        sum = sum + add_small;
        n = n - 1;
        m = m + 2;
    end
end
sum = 2 * sum /sqrt(pi);
```

运行结果：

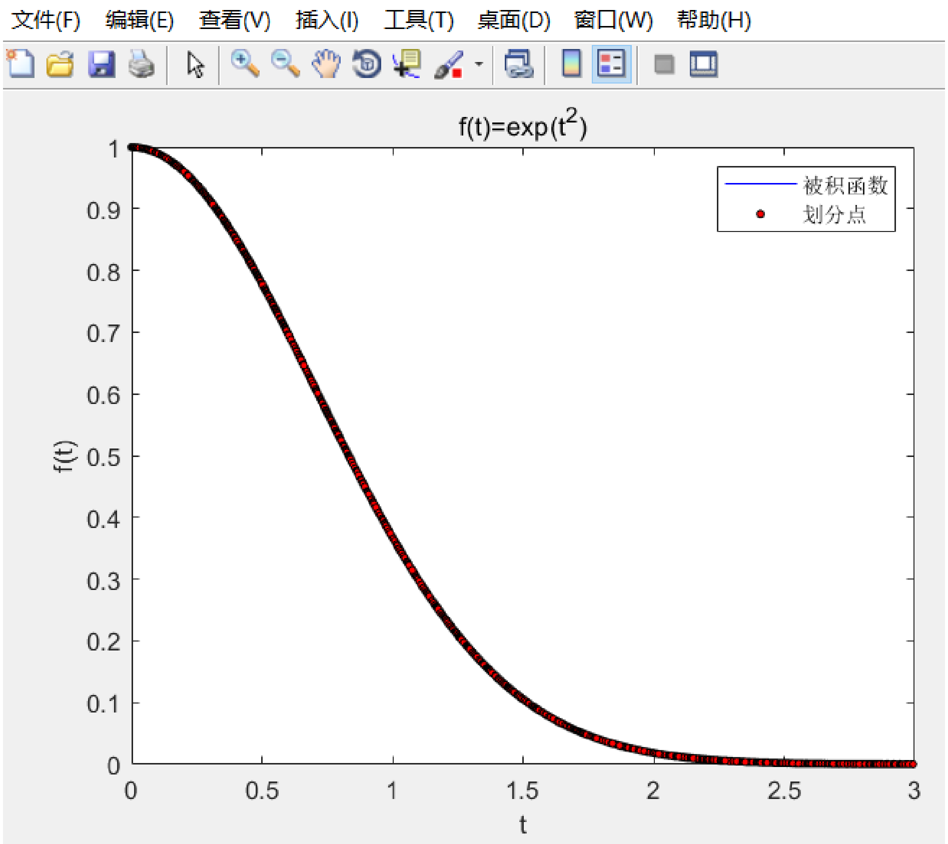
sum = 0.99997791

m =98

示例 2：

第六题

```
>> hw6_6_1
error = 0.999977913
区间数为: 14016
```



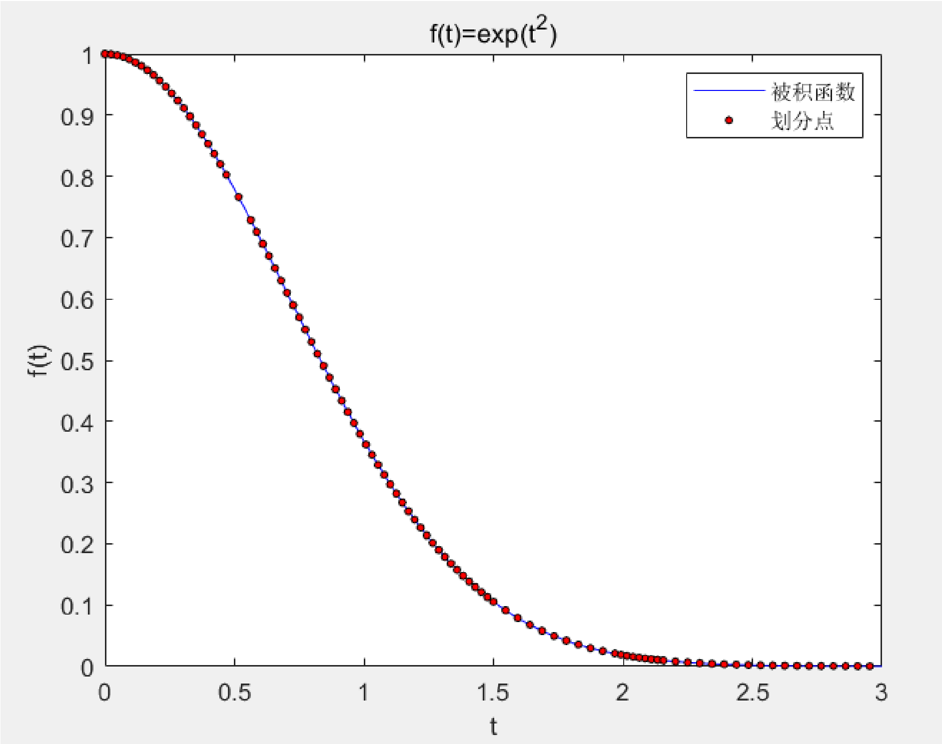
hw6_6_1.m

```
clear;
f = @(t)exp(-t^2);
[integ,A] = AdapQuad(f,0,3,1e-8);
erf=2/sqrt(pi)*integ;
fprintf('error = %10.9f\n',erf);
fprintf('区间数为: %d\n',length(A));

x=linspace(0,3);
y=exp(-x.^2);
yA=exp(-A.^2);
plot(x,y,'b',A,yA,'ko','MarkerFace','r','MarkerSize',3);
xlabel('t');
ylabel('f(t)');
title('f(t)=exp(t^2)');
legend('被积函数','划分点');
```

AdapQuad.m

```
function [integ,A] = AdapQuad(f,a,b,TOL)
% 自适应步长梯形公式求 f 在给定区间[a,b]上的积分，误差容限为 TOL
% 编程思路参考索尔的教材
integ = 0;
% 积分值初始化
lgth = abs(b-a);
% 计算区间的总长度
A=(a);
% 用来存放每个区间的左端点，特别地，除第一个元素，其他元素都是插入点的横坐标
nfitv=1;
% number of intervals,表示还有多少个区间上的近似积分待计算
S=@(x,y)(y-x)*(f(x)+f(y))/2;
% 梯形公式

while nfitv>0
    c = (a+b)/2;
    A(end+1)= c;
    if abs(S(a,b)-S(a,c)-S(c,b))<3*TOL*(b-a)/lgth
        integ = integ+S(a,c)+S(c,b);
        nfitv = nfitv-1;
        % 在容许误差内，则采用这一近似，并把待算区间数减一
        A=sort(A);
        if nfitv>0
           a=A(nfitv);
           b=A(nfitv+1);
        end
        % 可以看出每次计算的都是当前最后一个区间
        % 所以最后一个区间算完后就算倒数第二个区间
        % 又知道此时待求区间数为 nfitv
        % 给 A 排序后，元素按坐标大小排列，则对应的两个区间端点即为如上所示
    else
        a=c;
        nfitv=nfitv+1;
        % 不在容许误差内，则插入新的二分点，并把待算区间数加一
    end
end
end
```

```
>> hw6_6_2
error = 0.999977911
区间数为: 98
```



hw6_6_2.m

```
clear;
f = @(t)exp(-t^2);
[integ,A] = SimpAdapQuad(f,0,3,1e-8);
erf=2/sqrt(pi)*integ;
fprintf('error = %10.9f\n',erf);
fprintf('区间数为: %d\n',length(A));

x=linspace(0,3);
y=exp(-x.^2);
yA=exp(-A.^2);
plot(x,y,'b',A,yA,'ko','MarkerFace','r','MarkerSize',3);
xlabel('t');
ylabel('f(t)');
title('f(t)=exp(t^2)');
legend('被积函数','划分点');
```

SimpAdapQuad.m

```
function [integ,A] = SimpAdapQuad(f,a,b,TOL)
% 除了把梯形公式改成了辛普森公式，和改变容许误差条件外，其他和 AdapQuad 一样
% 自适应步长辛普森公式求 f 在给定区间[a,b]上的积分，误差容限为 TOL
% 编程思路参考索尔的教材
integ = 0;
% 积分值初始化
```

```
lgth = abs(b-a);
% 计算区间的总长度
A=(a);
% 用来存放每个区间的左端点，特别地，除第一个元素，其他元素都是插入点的横坐标
nfitv=1;
% number of intervals,表示还有多少个区间上的近似积分待计算
S=@(x,y)(y-x)*(f(x)+f(y)+4*f((x+y)/2))/6;
% 辛普森公式

while nfitv>0
    c = (a+b)/2;
    A(end+1)= c;
    if abs(S(a,b)-S(a,c)-S(c,b))<10*TOL*(b-a)/lgth
        integ = integ+S(a,c)+S(c,b);
        nfitv = nfitv-1;
        % 在容许误差内，则采用这一近似，并把待算区间数减一
        A=sort(A);
        if nfitv>0
            a=A(nfitv);
            b=A(nfitv+1);
        end
        % 可以看出每次计算的都是当前最后一个区间
        % 所以最后一个区间算完后就算倒数第二个区间
        % 又知道此时待求区间数为 nfitv
        % 给 A 排序后，元素按坐标大小排列，则对应的两个区间端点即为如上所示
    else
        a=c;
        nfitv=nfitv+1;
        % 不在容许误差内，则插入新的二分点，并把待算区间数加一
    end
end
end
```