实验报告（第四章）

第1题

(1)

代码

```
% 定义被积函数
f = @(x) sqrt(x).*log(x);
exact = -4/9;
h_values = [0.1, 0.05, 0.025];
a = eps; % 修改积分下限为 eps
b = 1;

% 复合梯形公式
T_results = zeros(1, length(h_values));
error_T = zeros(1, length(h_values));
for i = 1:length(h_values)
    h = h_values(i);
    n = floor((b - a)/h);
    x = linspace(a, b, n+1);
    y = f(x);
    T = (h/2)*(y(1) + 2*sum(y(2:end-1)) + y(end));
    T_results(i) = T;
    error_T(i) = abs(T - exact);
end

% 复合辛普森公式
S_results = zeros(1, length(h_values));
error_S = zeros(1, length(h_values));
for i = 1:length(h_values)
    h = h_values(i);
    n = floor((b - a)/h);
    if mod(n, 2) ~= 0, n = n + 1; end % 确保 n 为偶数
    h = (b - a)/n;
    x = linspace(a, b, n+1);
    y = f(x);
    S = (h/6)*(y(1) + 4*sum(y(2:2:end-1)) + 2*sum(y(3:2:end-2)) + y(end));
    S_results(i) = S;
    error_S(i) = abs(S - exact);
end

% 输出结果
disp('复合梯形公式结果：'); disp(T_results);
disp('复合梯形误差：'); disp(error_T);
disp('复合辛普森公式结果：'); disp(S_results);
disp('复合辛普森误差：'); disp(error_S);
```

思路

在不同步长下套用复合梯形公式和复合辛普森公式，并带入相应数据点，计算结果和误差

输出结果

复合梯形公式结果： -0.3718 -0.4108 -0.4288

复合梯形误差： 0.0726 0.0336 0.0157

复合辛普森公式结果： -0.2149 -0.2193 -0.2211

复合辛普森误差： 0.2296 0.2251 0.2234

(2)

代码

```
function R = romberg_integration(f, a, b, tol)
    % 最大迭代次数
    max_iter = 10;
    % 初始化 R 矩阵
    R = zeros(max_iter, max_iter);
    % 第一次梯形积分计算
    R(1, 1) = (b - a) / 2 * (f(a) + f(b));
    for iter = 1:max_iter - 1
        % 区间细分
        n = 2^iter;
        h = (b - a) / n;
        x = a + h/2:h:b - h/2;
        y = f(x);
        % 更新梯形积分值
        R(iter + 1, 1) = R(iter, 1) / 2 + h/2 * sum(y);
        for k = 2:iter + 1
            % 龙贝格外推公式
            R(iter + 1, k) = R(iter + 1, k - 1) + (R(iter + 1, k - 1) - R(iter, k
- 1)) / (4^(k - 1) - 1);
        end
        % 判断是否满足收敛条件
        if abs(R(iter + 1, iter + 1) - R(iter, iter)) < tol
            break;
        end
    end
    % 截取有效部分
    R = R(1:iter + 1, 1:iter + 1);
end

% 定义被积函数
f = @(x) sqrt(x).*log(x);
% 积分下限，避免 log(0)
a = eps;
```

```
% 积分上限
b = 1;
% 误差容限
tol = 1e-6;
% 调用龙贝格积分函数
R = romberg_integration(f, a, b, tol);
disp('龙贝格结果：');
disp(R);
```

思路

运用龙贝格求积算法，不断外推得到结果

输出

龙贝格结果： -0.0000 0 0 0 0 0 0 0 0 0 -0.2356 -0.3141 0 0 0 0 0 0 0 0 -0.3468 -0.3839 -0.3886 0 0 0 0 0 0 0 -0.3988 -0.4162 -0.4183 -0.4188 0 0 0 0 0 0 -0.4231 -0.4312 -0.4321 -0.4324 -0.4324 0 0 0 0 0 -0.4344 -0.4381 -0.4386 -0.4387 -0.4387 -0.4387 0 0 0 0 -0.4397 -0.4414 -0.4416 -0.4417 -0.4417 -0.4417 -0.4417 0 0 0 -0.4422 -0.4430 -0.4431 -0.4431 -0.4431 -0.4431 -0.4431 -0.4431 0 0 -0.4433 -0.4437 -0.4438 -0.4438 -0.4438 -0.4438 -0.4438 -0.4438 -0.4438 0 -0.4439 -0.4441 -0.4441 -0.4441 -0.4441 -0.4441 -0.4441 -0.4441 -0.4441 -0.4441 误差约为0.000776

(3)

代码

```
function Q = adaptive_simpson(f, a, b, tol, depth, max_depth)
 % 如果没有传入 depth 和 max_depth，则进行初始化
 if nargin < 5
     depth = 0;
 end
 if nargin < 6
     max_depth = 20; % 最大递归深度
 end

 c = (a + b) / 2;
 h = b - a;
 S1 = h / 6 * (f(a) + 4 * f(c) + f(b));
 S2 = h / 12 * (f(a) + 4 * f((a + c) / 2) + 2 * f(c) + 4 * f((c + b) / 2) + f(b));

 % 判断是否达到最大递归深度或满足误差条件
 if depth >= max_depth || abs(S2 - S1) < 15 * tol
     Q = S2 + (S2 - S1) / 15;
 else
     % 递归调用
     Q = adaptive_simpson(f, a, c, tol / 2, depth + 1, max_depth) + ...
         adaptive_simpson(f, c, b, tol / 2, depth + 1, max_depth);
 end
end
```

```
% 调用
f = @(x) sqrt(x).*log(x);
a = eps; % 修改积分下限为 eps 避免 log(0)
b = 1;
tol = 1e-4;
Q = adaptive_simpson(f, a, b, tol);
disp(['自适应辛普森结果：', num2str(Q)]);
```

思路

利用自适应积分方法，不断细分每个区间，直到精度满足要求

输出

自适应辛普森结果：-0.44444 误差约为0

第二题

(1)

代码

```
% 复合辛普森公式
% 定义被积函数
f = @(x,y) exp(-x.*y);

n = 4;
h_x = 1/n;
h_y = 1/n;
x = linspace(0, 1, n+1);
y = linspace(0, 1, n+1);

sum_y = 0;
for j = 1:n+1
    sum_x = 0;
    for i = 1:n+1
        if i == 1 || i == n+1
            c_x = 1;
        elseif mod(i, 2) == 1
            c_x = 4;
        else
            c_x = 2;
        end
        sum_x = sum_x + c_x * f(x(i), y(j));
    end
    integral_x = sum_x * h_x / 3;
    if j == 1 || j == n+1
        c_y = 1;
    elseif mod(j, 2) == 1
```

```
        c_y = 4;
    else
        c_y = 2;
    end
    sum_y = sum_y + c_y * integral_x;
end
result_simpson = sum_y * h_y / 3;
disp(['复合辛普森公式结果: ', num2str(result_simpson)]);

% 高斯求积公式
function [x, w] = leggauss(n)
    % 初始化
    x = zeros(n, 1);
    w = zeros(n, 1);
    m = floor((n + 1) / 2);
    eps = 1e-15;
    for i = 1:m
        % 初始猜测
        z = cos(pi * (i - 0.25) / (n + 0.5));
        z1 = z + 1;
        while abs(z - z1) > eps
            p1 = 1;
            p2 = 0;
            for j = 1:n
                p3 = p2;
                p2 = p1;
                p1 = ((2 * j - 1) * z * p2 - (j - 1) * p3) / j;
            end
            pp = n * (z * p1 - p2) / (z^2 - 1);
            z1 = z;
            z = z1 - p1 / pp;
        end
        x(i) = -z;
        x(n + 1 - i) = z;
        w(i) = 2 / ((1 - z^2) * pp^2);
        w(n + 1 - i) = w(i);
    end
end

% 主程序
f = @(x,y) exp(-x.*y);
[xi, wi] = leggauss(4);
xi_x = (xi + 1) / 2;
xi_y = (xi + 1) / 2;
wi_x = wi / 2;
wi_y = wi / 2;

result_gauss = 0;
for i = 1:4
    for j = 1:4
        result_gauss = result_gauss + wi_x(i) * wi_y(j) * f(xi_x(i), xi_y(j));
    end
end
disp(['高斯求积公式结果: ', num2str(result_gauss)]);
```

思路

利用复合辛普森求积公式(n=4)和高斯求积公式(n=4)计算结果

输出

复合辛普森公式结果：0.55188 高斯求积公式结果：0.7966

(2)

代码

```
f = @(x,y) exp(-x.*y);
n = 4;
h_x = 1/n;
h_y = 1/n;
x = linspace(0, 1, n+1);
y = linspace(0, 1, n+1);
sum_y = 0;
for j = 1:n+1
    integral_x = 0;
    for i = 1:n+1
        if x(i)^2 + y(j)^2 <= 1  % 判断点是否在区域内
            if i == 1 || i == n+1
                c_x = 1;
            elseif mod(i, 2) == 1
                c_x = 4;
            else
                c_x = 2;
            end
            integral_x = integral_x + c_x * f(x(i), y(j));
        end
    end
    integral_x = integral_x * h_x / 3;
    if j == 1 || j == n+1
        c_y = 1;
    elseif mod(j, 2) == 1
        c_y = 4;
    else
        c_y = 2;
    end
    sum_y = sum_y + c_y * integral_x;
end
result = sum_y * h_y / 3;
disp(['复合辛普森公式结果：', num2str(result)]);
```

思路

利用复合辛普森公式(n=4)计算结果

输出

复合辛普森公式结果：0.46505