第一部分 非线性方程组求解

第一部分 实验代码

```
%%
% 哈工大数值分析 2020 年秋研究生，上机实验
% 第一部分 | 非线性方程组求解
% 时间: 2020/10/15
% 学生:
% ----------------------------------------------------------
% 1、【二分法】
% 2、【牛顿法】
% 3、【割线法】
% 4、【改进的牛顿法】
% 5、【拟牛顿法】
%%
% define the algorithm for which be used to solve the nolinear equation
% where the variable can be 0、1、2、3、4 , each one corresponds to the algorithm above

algorithm_index = 0;

%%
% 方法一：二分法
% 题目：用二分法计算方程 sin(x)-pow(x,2)/2 = 0 在(1,2)内的根的近似值，要求 ε=0.5*10^(-5)

if algorithm_index == 0
    % 易知函数 equation(x)在区间(1,2)内连续可导，且方程的根在区间(1,2)内存在且唯一
    a = 1;
    b = 2;
    count = 0;
    % 定义允许的误差
    delta = 0.5*10^(-5);
    % 定义函数原型，方便用于不同的方程，提高程序应用的普遍性
    syms f x;
    f = sin(x)-x^2/2;
    % 求解迭代解以及迭代次数
    [answer,count] = dichotomy(a,b,count,delta,f);
    fprintf("二分法求非线性方程，解为 %f 迭代了 %d 次\n",answer,count);
end
%%
% 方法二：牛顿法
% 题目：用牛顿法求解下列非线性方程的根，题目见实验报告册

if algorithm_index == 1
    % 定义 3 个方程的初始值
    x1_initial = 0.5;
    x2_initial = 1;
    x3_initial_1 = 0.45;
    x3_initial_2 = 0.65;
```

```
    % 定义允许的误差以及最大的迭代次数
    delta = 0.5*10^(-5);
    N = 100;
    % 定义函数原型，方便用于不同的方程，提高程序应用的普遍性
    syms f x;
    % 求解方程 1
    count = 0;
    f = x*exp(x)-1;
    [answer,count] = newton(x1_initial,count,delta,f,N);
    fprintf("牛顿法求非线性方程 1，初始值为 %f 解为 %f 迭代了 %d 次\n",x1_initial,answer,count);
    % 求解方程 2
    count = 0;
    f = x^3-x-1;
    [answer,count] = newton(x2_initial,count,delta,f,N);
    fprintf("牛顿法求非线性方程 2，初始值为 %f 解为 %f 迭代了 %d 次\n",x2_initial,answer,count);
    % 求解方程 3
    count = 0;
    f = (x-1)^2*(2*x-1);
    [answer,count] = newton(x3_initial_1,count,delta,f,N);
    fprintf("牛顿法求非线性方程 3，初始值为 %f 解为 %f 迭代了 %d 次\n",x3_initial_1,answer,count);
    count = 0;
    [answer,count] = newton(x3_initial_2,count,delta,f,N);
    fprintf("牛顿法求非线性方程 3，初始值为 %f 解为 %f 迭代了 %d 次\n",x3_initial_2,answer,count);
end
%%
% 方法三：割线法/多点迭代法
% 题目：用割线法求解下列非线性方程的根，题目见实验报告册

if algorithm_index == 2
    % 定义迭代初始点
    x0_initial = 0.4;
    x1_initial = 0.6;
    % 定义允许的误差以及最大的迭代次数
    delta = 0.5*10^(-5);
    N = 100;
    count = 0;
    % 定义函数原型，方便用于不同的方程，提高程序应用的普遍性
    syms f x;
    f = x*exp(x)-1;
    % 求解方程
    [answer,count] = secant(x0_initial,x1_initial,count,delta,f,N);
    fprintf("割线法求非线性方程，初始值 x0 为 %f 初始值 x1 为 %f 解为 %f 迭代了 %d 次\n",x0_initial,x1_initial,answer,count);
end
%%
% 方法四：改进的牛顿法
% 题目：用改进的牛顿法求解下列非线性方程的根，题目见实验报告册

if algorithm_index == 3
```

```
    % 定义迭代初始点
    x_initial = 0.55;
    % 定义允许的误差以及最大的迭代次数
    delta = 0.5*10^(-5);
    N = 100;
    count = 0;
    % 定义函数原型，方便用于不同的方程，提高程序应用的普遍性
    syms f x;
    f = (x-1)^2*(2*x-1);
    % 求解方程
    [answer,count] = advance_newton(x_initial,count,delta,f,N);
    fprintf("改进的牛顿法求非线性方程，初始值为 %f 解为 %f 迭代了 %d 次\n",x_initial,answer,count);
end
%%
% 方法五：拟牛顿法-秩 1 的拟牛顿法-逆 Broyden 法
% 题目：用拟牛顿法-逆 Broyden 求解下列非线性方程组的根，题目见实验报告册
% 注意，以下 Fcn 仅适用于 3x3 阶非线性方程组求解
% TODO 改成 NxN 阶非线性方程组求解

if algorithm_index == 4
    % 定义迭代初始解向量
    X_initial = [1.0 1.0 1.0]';
    % 定义允许的误差以及最大的迭代次数
    delta = 0.5*10^(-5);
    N = 100;
    count = 0;
    % 定义函数原型，方便用于不同的方程，提高程序应用的普遍性
    syms f_1 f_2 f_3 x y z;
    f_1 = x*y-z^2-1;
    f_2 = x*y*z+y^2-x^2-2;
    f_3 = exp(x)+z-exp(y)-3;
    F = [f_1 f_2 f_3]';
    % 求系数矩阵 A0
    A_initial = [diff(f_1,x) diff(f_1,y) diff(f_1,z); diff(f_2,x) diff(f_2,y) diff(f_2,z); diff(f_3,x) diff(f_3,y) diff(f_3,z)];
    x = X_initial(1);
    y = X_initial(2);
    z = X_initial(3);
    % 求系数矩阵 H0
    H_initial = inv(eval(A_initial));
    % 求解方程
    [answer,count] = quasi_newton(X_initial,H_initial,count,delta,F,N);
    fprintf("拟牛顿法求非线性方程组，初始值为 [%f %f %f]' 解为 [%f %f %f]' 迭代了 %d 次\n",X_initial(1),X_initial(2),X_initial(3),answer(1),answer(2),answer(3),count);
end
%%
% 定义迭代函数实现二分法
function [answer,count] = dichotomy(next_x,next_y,count,delta,f)
answer = (next_x+next_y)/2;
```

```

x = next_y;
if eval(f) ==0
    answer = next_y;
elseif next_y - next_x > 2*delta
    count = count+1;
    x = answer;
    f1 = eval(f);
    x = next_x;
    f2 = eval(f);
    if f1*f2 > 0
        next_x = answer;
    else % 包含了端点值为解的情况
        next_y = answer;
    end
    [answer,count] = dichotomy(next_x,next_y,count,delta,f);
end
end
%%
% 定义迭代函数实现牛顿法
function [answer,count] = newton(x_initial,count,delta,f,N)
answer = x_initial;
x = answer;
answer = answer - eval(f)/eval(diff(f));
count = count +1;
%if (abs(eval(f)) > delta)||(abs(eval(f)/eval(diff(f))) > delta)
if abs(eval(f)/eval(diff(f))) > delta
    if count < N
        [answer,count] = newton(answer,count,delta,f,N);
    else
        fprinf("Error, can not solve this equation in a limited count of %d",N);
    end
end
end
%%
% 定义迭代函数实现割线法
function [answer,count] = secant(x0_initial,x1_initial,count,delta,f,N)
answer = x1_initial;
answer_k = x1_initial;
answer_k_1 = x0_initial;
x = answer_k_1;
f0 = eval(f);
x = answer_k;
f1 = eval(f);
answer = answer_k - f1/(f1-f0)*(answer_k-answer_k_1);
count = count +1;
if abs(f1/(f1-f0)*(answer_k-answer_k_1)) > delta
    if count < N
        [answer,count] = secant(answer_k,answer,count,delta,f,N);
```

```
    else
        fprinf("Error, can not solve this equation in a limited count of %d",N);
    end
end
end
%%
% 定义迭代函数实现改进的牛顿法
function [answer,count] = advance_newton(x_initial,count,delta,f,N)
answer = x_initial;
x = answer;
answer = answer - 2*eval(f)/eval(diff(f));
count = count +1;
fprintf("第%d 次迭代，迭代解为%f 两次解的差值为%f delta 为%f\n",count,answer,abs(2*eval(f)/eval(diff(f))),delta);
%if (abs(eval(f)) > delta)||(abs(eval(f)/eval(diff(f))) > delta)
if abs(2*eval(f)/eval(diff(f))) > delta
    if count < N
        [answer,count] = advance_newton(answer,count,delta,f,N);
    else
        fprinf("Error, can not solve this equation in a limited count of %d",N);
    end
end
end
%%
% 定义迭代函数实现拟牛顿法-逆 Broyden 法
function [answer,count] = quasi_newton(X_initial,H_initial,count,delta,F,N)
answer = X_initial;
x = X_initial(1);
y = X_initial(2);
z = X_initial(3);
F_i = eval(F);
answer = answer - H_initial*F_i;
count = count + 1;
% 比较差值向量的 X(i+1) - X(i)的无穷范数与 delta
if norm(H_initial*F_i,inf) > delta
    R =  - H_initial*eval(F);
    x = answer(1);
    y = answer(2);
    z = answer(3);
    F_i_1 = eval(F);
    Y = F_i_1 - F_i;
    H_initial = H_initial + (R-H_initial*Y)*(R'*H_initial)/(R'*H_initial*Y);
    if count < N
        [answer,count] = quasi_newton(answer,H_initial,count,delta,F,N);
    else
        fprinf("Error, can not solve this equation set in a limited count of %d",N);
    end
end
end
```

```

%% ------------------END OF THE FILE------------------
```

第一部分 实验结果

```
二分法求非线性方程，解为 1.404415 迭代了 17 次

牛顿法求非线性方程 1，初始值为 0.500000 解为 0.567143 迭代了 4 次
牛顿法求非线性方程 2，初始值为 1.000000 解为 1.324718 迭代了 5 次
牛顿法求非线性方程 3，初始值为 0.450000 解为 0.500000 迭代了 4 次
牛顿法求非线性方程 3，初始值为 0.650000 解为 0.500000 迭代了 9 次

割线法求非线性方程，初始值 x0 为 0.400000 初始值 x1 为 0.600000 解为 0.567143 迭代了 4 次

牛顿下山法
下山因子为 2，初始值为 0.55 时，在 100 次迭代次数内不能达到精度要求
Error, can not solve this equation in a limited count of 100

拟牛顿法求非线性方程组，初始值为 [1.000000 1.000000 1.000000]' 解为 [1.777672 1.423961 1.237471]' 迭代了 10 次
```

第二部分 高斯（列）主元消去法

第二部分 实验代码

```
%%
% 哈工大数值分析 2020 年秋研究生，上机实验
% 第二部分 | 线性方程组求解/高斯列主元消去法
% 时间: 2020/10/22
% 学生:
% ----------------------------------------------------------
% 1、【高斯消去法】
% 2、【高斯列主元消去法】

% 若高斯消去法解线性方程组时，如果主元元素等于 0，则消去法无法继续，或者主元元素接近于 0，继续使用消去法将导致不稳定现象，
% 此时需要使用高斯列主元消去法
%%
% 定义要求解的方程组一
A = [10^(-8) 2 3; -1 3.712 4.623; -2 1.072 5.643];
b = [1 2 3];
% 定义要求解的方程组二
C = [4 -2 4; -2 17 10; -4 10 9];
d = [10 3 7];
%%
% 分别用高斯列主元消去法和高斯消去法求解方程组
[answer] = gauss_elimination(A,b);
fprintf("高斯法 方程一 answer = [ %f %f %f ]\n",answer(1),answer(2),answer(3));
```

```
[answer] = gauss_principal_element_elimination(A,b);
fprintf("高斯列主元法 方程一 answer = [ %f %f %f ]\n",answer(1),answer(2),answer(3));
[answer] = gauss_elimination(C,d);
fprintf("高斯法 方程二 answer = [ %f %f %f ]\n",answer(1),answer(2),answer(3));
[answer] = gauss_principal_element_elimination(C,d);
fprintf("高斯列主元法 方程二 answer = [ %f %f %f ]\n",answer(1),answer(2),answer(3));
%%
%定义函数实现高斯消去法
function [answer] = gauss_elimination(A,b)
% 获取系数矩阵的阶数
[count] = size(A,1);
% 消元过程
for i = 1:count-1
    for j = i:count-1
        for n = 1:count-i
            A(i+n,j+1) =  A(i+n,j+1)-A(i,j+1)*A(i+n,i)/A(i,i);
        end
    end
    for n = 1:count-i
        b(i+n) = b(i+n)-b(i)*A(i+n,i)/A(i,i);
    end
end
% 回代求解过程
answer = zeros(count,1);
for i = 1:count
    answer(count-i+1) = b(count-i+1);
    if i>1
        for j = 1:i-1
            answer(count-i+1) = answer(count-i+1) - answer(count-i+1+j)*A(count-i+1,count-i+1+j);
        end
    end
    answer(count-i+1) = answer(count-i+1)/A(count-i+1,count-i+1);
end
end
%%
%定义函数实现高斯列主元消去法
function [answer] = gauss_principal_element_elimination(A,b)
% 获取系数矩阵的阶数
[count] = size(A,1);
% 消元过程
for i = 1:count-1
    % 每一次消元前进行列选主元
    cursor = i; max = abs(A(i,i));
    for m = i:count
       if abs(A(m,i))>max
           max = abs(A(m,i));
           cursor = m;
       end
```

```
    end
    % 列主元行交换
    if cursor ~= i
        row_temp = A(i,:);
        A(i,:) = A(cursor,:);
        A(cursor,:) = row_temp;
        b_temp = b(i);
        b(i) = b(cursor);
        b(cursor) = b_temp;
    end
    % 消元
    for j = i:count-1
        for n = 1:count-i
            A(i+n,j+1) =  A(i+n,j+1)-A(i,j+1)*A(i+n,i)/A(i,i);
        end
    end
    for n = 1:count-i
        b(i+n) = b(i+n)-b(i)*A(i+n,i)/A(i,i);
    end
end
% 回代求解过程
answer = zeros(count,1);
for i = 1:count
    answer(count-i+1) = b(count-i+1);
    if i>1
        for j = 1:i-1
            answer(count-i+1) = answer(count-i+1) - answer(count-i+1+j)*A(count-i+1,count-i+1+j);
        end
    end
    answer(count-i+1) = answer(count-i+1)/A(count-i+1,count-i+1);
end
end

%% -----------------END OF THE FILE-----------------
```

第二部分 实验结果

```
高斯法 方程一 answer = [ -0.491058 -0.050886 0.367257 ]
高斯列主元法 方程一 answer = [ -0.491058 -0.050886 0.367257 ]
高斯法 方程二 answer = [ 0.196429 -0.892857 1.857143 ]
高斯列主元法 方程二 answer = [ 0.196429 -0.892857 1.857143 ]
```

第三部分 多项式最小二乘拟合

第三部分 实验代码

```
%%
% 哈工大数值分析 2020 年秋研究生，上机实验
```

```
% 第三部分 | 最小二乘拟合/Least squares fitting
% 时间: 2020/10/29
% 学生:
% ----------------------------------------------------------
% 1、【利用最小二乘法处理实验数据】
%%
% 定义拟合的函数模型及多项式的最高的幂数和数据点
syms f x;
r = 1;
X = [3 4 5 6 7 8 9];
Y = [2.01 2.98 3.50 5.02 5.47 6.02 7.05];
% 调用函数进行拟合
[answer] = Lsf(r,X,Y);
fprintf("多项式最小二乘拟合  系数分别为\n");
for i = 0:r
    fprintf("  %f  ",answer(i+1));
end
fprintf("\n");
% 绘制图像比较拟合的结果
scatter(X,Y,15,'filled');
hold on;
grid on;
num = 0;
fit_x = zeros(1,ceil((max(X)-min(X))/0.01)+1);
fit_y = zeros(1,ceil((max(X)-min(X))/0.01)+1);
for m = min(X):0.01:max(X)
    num = num + 1;
    fit_x(num) = m;
    for j = 0:r
        fit_y(num) = fit_y(num) + answer(j+1)*fit_x(num)^j;
    end
end
plot(fit_x,fit_y);
%%
% 定义函数求解多项式最小二乘拟合系数
function [answer] = Lsf(r,X,Y)
% 定义法方程系数矩阵及方程右侧向量
A = zeros(r+1);
b = zeros(r+1,1);
% 由最小二乘原则/最佳平方逼近求出法方程各个系数/多元函数求极值
for i = 0:r
    for j = 0:r
        for n = 0:length(X)-1
            A(i+1,j+1) = A(i+1,j+1) + 1*X(n+1)^(i+j);
        end
    end
end
% 算出法方程右侧向量
for i = 0:r
```

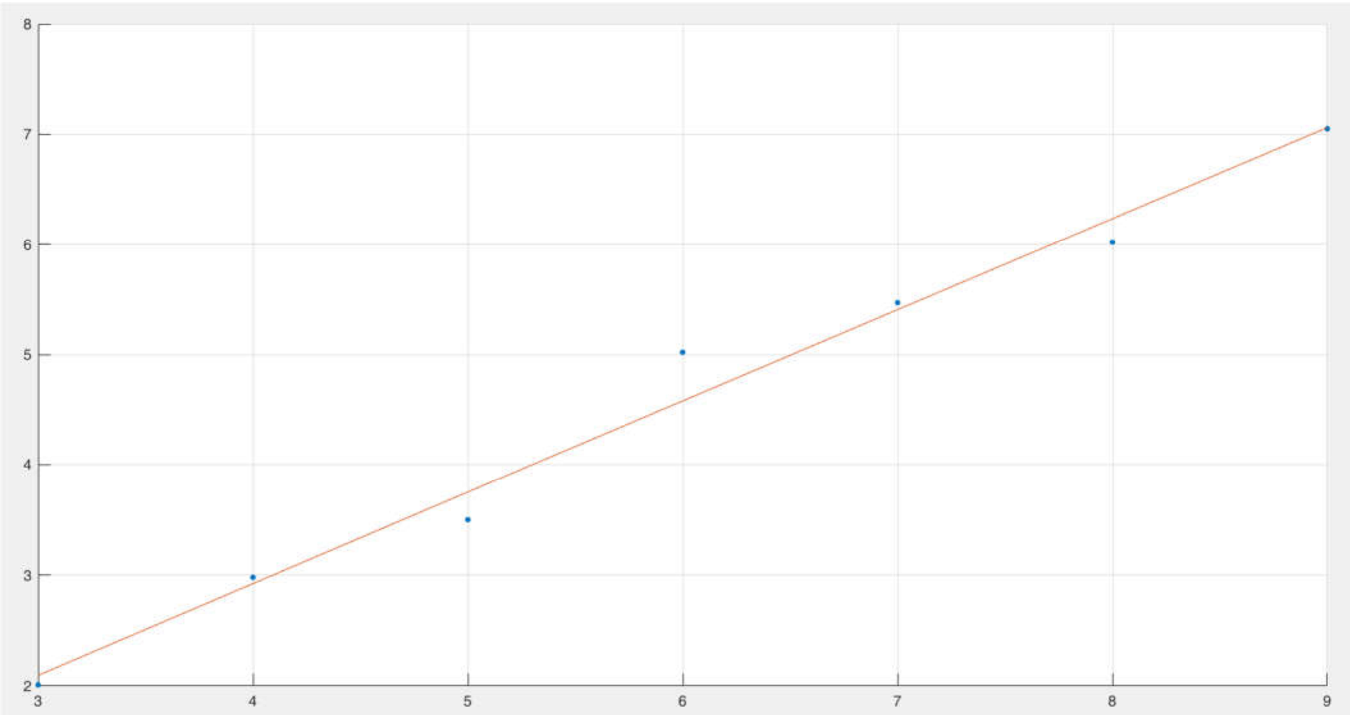
```
    for n = 0:length(X)-1
        b(i+1) = b(i+1) + 1*X(n+1)^i*Y(n+1);
    end
end
% 求出系数矩阵后，利用上个实验中定义的高斯列主元消去法求解非齐次线性方程组
[answer] = gauss_principal_element_elimination(A,b);
end

%%
%定义函数实现高斯列主元消去法
function [answer] = gauss_principal_element_elimination(A,b)
% 获取系数矩阵的阶数
[count] = size(A,1);
% 消元过程
for i = 1:count-1
    % 每一次消元前进行列选主元
    cursor = i; max = abs(A(i,i));
    for m = i:count
       if abs(A(m,i))>max
           max = abs(A(m,i));
           cursor = m;
       end
    end
    % 列主元行交换
    if cursor ~= i
        row_temp = A(i,:);
        A(i,:) = A(cursor,:);
        A(cursor,:) = row_temp;
        b_temp = b(i);
        b(i) = b(cursor);
        b(cursor) = b_temp;
    end
    % 消元
    for j = i:count-1
        for n = 1:count-i
            A(i+n,j+1) =  A(i+n,j+1)-A(i,j+1)*A(i+n,i)/A(i,i);
        end
    end
    for n = 1:count-i
        b(i+n) = b(i+n)-b(i)*A(i+n,i)/A(i,i);
    end
end
% 回代求解过程
answer = zeros(count,1);
for i = 1:count
    answer(count-i+1) = b(count-i+1);
    if i>1
        for j = 1:i-1
```

```
            answer(count-i+1) = answer(count-i+1) - answer(count-i+1+j)*A(count-i+1,count-i+1+j);
        end
    end
    answer(count-i+1) = answer(count-i+1)/A(count-i+1,count-i+1);
end
end
%% ------------------END OF THE FILE------------------
```

第三部分 实验结果

```
多项式最小二乘拟合 系数分别为
0.386429    0.827500
```



第四部分 龙贝格积分法

第四部分 实验代码

```
%%
% 哈工大数值分析 2020 年秋研究生，上机实验
% 第四部分 | 龙贝格积分法/Romberg Integral
% 时间: 2020/10/29
% 学生:
% ---------------------------------------------------------
% 1、【龙贝格积分法计算以下积分的近似值】

%%
% 定义被积函数原型及精度要求
syms f x;
delta = 7*10^(-6);
T = zeros(8);
```

```
% 求解定积分 1
f = x^3;
a = 6; b = 100;m = 1;
% f = 4/(1+x^2);
% a = 0; b = 1;m = 1;
[answer,m] = Romberg(f,a,b,m,delta,T);
fprintf("\n 求解定积分 1，解为 %6f    m =  %d  验证解为%f\n-------------------------------------\n",answer, m, int(eval(f),x,a,b));
% 求解定积分 2
f = sin(x)/x;
a = 0; b = 1;m = 1;
[answer,m] = Romberg(f,a,b,m,delta,T);
fprintf("\n 求解定积分 2，解为 %f    m =  %d  验证解为%f\n-------------------------------------\n",answer, m, int(eval(f),x,a,b));
% 求解定积分 3
f = sin(x^2);
a = 0; b = 1;m = 1;
[answer,m] = Romberg(f,a,b,m,delta,T);
fprintf("\n 求解定积分 3，解为 %f    m =  %d  验证解为%f\n-------------------------------------\n",answer, m, int(eval(f),x,a,b));
%%
% 定义 Romberg 积分迭代函数
function [answer,m] = Romberg(f,a,b,m,delta,T)
if m == 1
    x = a;
    if x == 0
        syms x;
        fa = limit(eval(f),x,0);
    else
        fa = eval(f);
    end
    x = b;
    fb = eval(f);
    T(1,1) = 0.5*(b-a)*(fa+fb);
end
for i = 0:m
    % 计算龙贝格 T-数表
    if  i == 0 % T 型求积公式
        sum = 0;
        for j = 0:2^(m-1)-1
            x = a + (b-a)/2^(m-1)*(j+0.5);
            if x == 0
                syms x;
                sum = sum + limit(eval(f),x,0);
            else
                sum = sum + eval(f);
            end
        end
        T(m+1,i+1) = 0.5*T(m,i+1)+0.5*(b-a)/2^(m-1)*sum;
```

```
    else % 高阶求积公式
        T(m+1,i+1) = (4^i*T(m+1,i)-T(m,i))/(4^i-1);
    end
end
% 计算龙贝格数表上的对角线最后两元素之差，与给定的精度进行比较
if abs(T(m+1,m+1)-T(m,m)) <= delta
    answer = T(m+1,m+1);
    fprintf("求解完成!  T-数表为\n\n");
    for i = 0:m
        for j = 0:i
            fprintf("%f  ",T(i+1,j+1));
        end
        fprintf("\n");
    end
else
    % 未达到精度要求，继续迭代求解
    [answer,m] = Romberg(f,a,b,m+1,delta,T);
end
end
%% ------------------END OF THE FILE------------------
```

第四部分 实验结果

```
求解完成!  T-数表为

47010152.000000
30502295.000000  24999676.000000
26375330.750000  24999676.000000  24999676.000000

求解定积分 1，解为 24999676.000000    m =  2   验证解为 24999676.000000
-------------------------------------
求解完成!  T-数表为

0.920735
0.939793  0.946146
0.944514  0.946087  0.946083
0.945691  0.946083  0.946083  0.946083

求解定积分 2，解为 0.946083    m =  3   验证解为 0.946083
-------------------------------------
求解完成!  T-数表为

0.420735
0.334070  0.305181
0.315975  0.309944  0.310261
0.311680  0.310249  0.310269  0.310269
0.310620  0.310267  0.310268  0.310268  0.310268

求解定积分 3，解为 0.310268    m =  4   验证解为 0.310268
```