**写之前记得加上clear;clc; !!!!!!!!!!!!!!!!!!**

二分法：

function res = division\_fun(f, a, b, e)

% f:函数指针

% a:区间左端点

% b:区间右端点

% e:精度

while abs(b - a) > e

mid = f((a + b) / 2);

if mid == 0

res = (a + b) / 2;

break

elseif mid \* f(a) < 0

b = (a + b) / 2;

elseif mid \* f(a) > 0

a = (a + b) / 2;

end

end

res = (a + b) / 2;

end

不动点迭代法：

function res = fix\_point\_fun(f, x0, e)

% f:函数指针

% x0:迭代初值

% e:精度

x1 = f(x0);

while abs(x1 - x0) >= e

x0 = x1;

x1 = f(x0);

end

res = x1;

end

牛顿迭代法：

function res = newton\_fun(f, x0, e)

% f:函数

% x0:初始迭代值

% e:精度

x1 = x0 - f(x0) / dif\_fun(f, x0);

while abs(x1 - x0) >= e

x0 = x1;

x1 = x0 - f(x0) / dif\_fun(f, x0);

end

res = x1;

end

牛顿下山法：

function res = newton\_downhill\_fun(f, x0, e)

% f:函数

% x0:初始迭代值

% e:精度

x1 = x0 - f(x0) / dif\_fun(f, x0);

while abs(x1 - x0) >= e

lambda = 1;

while abs(f(x1)) >= abs(f(x0))

lambda = lambda / 2;

x1 = x0 - lambda \* f(x0) / dif\_fun(f, x0);

end

lambda = 1;

x0 = x1;

x1 = x0 - lambda \* f(x0) / dif\_fun(f, x0);

end

res = x1;

end

牛顿插值：

function res = newton\_interpolation\_fun(X, Y)

% X:横坐标向量

% Y:纵坐标向量

ps = [];

n = length(X);

for i = 1:n

ps = [ps; poly(X(i))];

end

T = Y;

for i = 1:n-1

ls = zeros(1, n);

for j = i:n-1

ls(j+1) = (T(i, j+1) - T(i, j)) / (X(j+1) - X(j+1-i));

end

T = [T; ls];

end

res = zeros(1, n);

for i = 2:n

xp = 1;

for j = 1:i-1

xp = conv(xp, ps(j, :));

end

xpf = zeros(1, n);

for j = 1:length(xp)

xpf(n-length(xp)+j) = xp(j);

end

res = res + T(i, i) \* xpf;

disp(res)

end

res(n) = res(n) + T(1, 1);

res = res';

end

新牛顿插值：

function f = new\_newton\_interpolation\_fun(X, Y)

n=length(X);

T=zeros(n,n);

% 对差商表第一列赋值

for k=1:n

T(k)=Y(k);

end

% 求差商表

for i=2:n

for k=i:n

T(k,i)=(T(k,i-1)-T(k-1,i-1))/(X(k)-X(k+1-i));

end

end

f = @(x)(T(1,1)+0\*x);

for i = 2:length(X)

w = @(x)(1+0\*x);

for j = 1:i-1

w = @(x)(w(x).\*(x-X(j)));

end

f = @(x)(f(x)+T(i, i).\*w(x));

end

end

拉格朗日插值：

function res = lagrange\_interpolation\_fun(X, Y)

% X:横坐标向量

% Y:纵坐标向量

ps = [];

for i = 1:length(X)

ps = [ps; poly(X(i))];

end

ls = [];

for i = 1:length(ps)

ll = 1;

for j = 1:length(ps)

if j == i

continue

end

ll = conv(ll, ps(j, :));

end

div\_ll = 1;

for j = 1:length(ps)

if j == i

continue

end

div\_ll = div\_ll \* (X(i) - X(j));

end

ll = ll ./ div\_ll;

ls = [ls; ll];

end

res = 0;

for i = 1:length(ls)

res = res + ls(i, :) \* Y(i);

end

end

Jacobi迭代：

function x = jacobi\_fun(a, b, x0, e)

% a:系数矩阵

% b:右边向量

% x0:初始向量(列向量)

% e:精度

% 最大迭代次数M

n = length(b);

M = 100000;

m = 0;

x = zeros(n, 1);

% 求系数矩阵的对角矩阵

cm\_diag = diag(diag(a));

B = cm\_diag \ (cm\_diag - a);

% 计算谱半径

R = max(abs(eig(B)));

if R >= 1

x = zeros(n, 1);

disp('谱半径不小于1，无法收敛')

return

end

while m <= M

m = m + 1;

for i = 1:n

sum\_ax = 0;

for j = 1:n

if j == i

continue

end

sum\_ax = sum\_ax + a(i, j) \* x0(j);

end

x(i) = -(sum\_ax - b(i)) / a(i, i);

end

if norm(x - x0, 1) < e

break

end

x0 = x;

end

if m > M

disp('达到最大循环次数')

end

end

G-S迭代：

function x = gauss\_seidel\_fun(a, b, x0, e)

% a:系数矩阵

% b:右边向量

% x0:初始向量(列向量)

% e:精度

% 最大迭代次数M

n = length(b);

% 求系数矩阵的对角矩阵

cm\_diag = diag(diag(a));

B = cm\_diag \ (cm\_diag - a);

% 计算谱半径

R = max(abs(eig(B)));

if R >= 1

x = zeros(n, 1);

disp('谱半径不小于1，无法收敛')

return

end

M = 100000;

m = 0;

x = zeros(n, 1);

while m <= M

m = m + 1;

for i = 1:n

sum\_ax = 0;

for j = 1:i-1

sum\_ax = sum\_ax + a(i, j) \* x(j);

end

for j = i+1:n

sum\_ax = sum\_ax + a(i, j) \* x0(j);

end

x(i) = -(sum\_ax - b(i)) / a(i, i);

end

if norm(x - x0, 1) < e

break

end

x0 = x;

end

if m > M

disp('达到最大循环次数')

end

end

高斯消去法：

function x = gauss\_elimi\_fun(a, b)

% a:系数矩阵

% b：右边向量

% n：方程组的阶数

% x: 求解结果列向量

n = length(b);

m = zeros(n, n);

x = zeros(n, 1);

for k = 1:n-1

if det(a) == 0

disp('第'+str(k)+'次迭代的矩阵a的顺序主子式为0，无法继续运算');

return

end

for i = k+1:n

m(i, k) = a(i, k) / a(k, k);

for j = k+1:n

a(i, j) = a(i, j) - m(i, k) \* a(k, j);

end

b(i) = b(i) - m(i, k) \* b(k);

end

end

x(n) = b(n) / a(n, n);

for i = n-1:-1:1

sum\_of\_ax = 0;

for j = i+1:n

sum\_of\_ax = sum\_of\_ax + a(i, j) \* x(j);

end

disp(sum\_of\_ax)

x(i) = (b(i) - sum\_of\_ax) / a(i, i);

end

end

列主元高斯消去法：

function x = col\_pivot\_gauss\_elimi\_fun(a, b)

% a:系数矩阵

% b：右边向量

% n：方程组的阶数

% x: 求解结果列向量

n = length(b);

m = zeros(n, n);

x = zeros(n, 1);

for k = 1:n-1

if det(a) == 0

disp('第'+str(k)+'次迭代的矩阵a的顺序主子式为0，无法继续运算');

return

end

max\_ai = k;

max\_a = a(k, k);

for i = k:n

if a(i, k) > max\_a

max\_a = a(i, k);

max\_ai = i;

end

end

if max\_ai ~= k

tmp = a(k, :);

a(k, :) = a(max\_ai, :);

a(max\_ai, :) = tmp;

end

for i = k+1:n

m(i, k) = a(i, k) / a(k, k);

for j = k+1:n

a(i, j) = a(i, j) - m(i, k) \* a(k, j);

end

b(i) = b(i) - m(i, k) \* b(k);

end

end

x(n) = b(n) / a(n, n);

for i = n-1:-1:1

sum\_of\_ax = 0;

for j = i+1:n

sum\_of\_ax = sum\_of\_ax + a(i, j) \* x(j);

end

x(i) = (b(i) - sum\_of\_ax) / a(i, i);

end

end

三次样条插值：

function res = cubic\_spline\_interpolation\_fun(X, Y, condi)

% 目前只能设定自然条件

% X:横坐标向量

% Y:纵坐标向量

% condi:边界条件值

n = length(X);

form = zeros(n,n);

form(:,1)=Y;

M0 = condi(1);

Mn = condi(2);

for i=2:n

for j=i:n

form(j,i) = (form(j,i-1)-form(j-1,i-1))/(X(j)-X(j-i+1));

end

end

h = zeros(n-1,1);

for i=1:n-1

h(i)=X(i+1)-X(i);

end

b = zeros(n-2,1);

c = zeros(n-2,n-2);

for i=1:n-2

c(i,i)=2;

if (i==1)

b(i,1)=6 \* form(i+2,3)-h(i)/(h(i)+h(i+1))\* M0;

elseif (i==(n-2))

b(i,1)=6 \* form(i+2,3)-(h(i+1)/(h(i)+h(i+1)))\* Mn;

else

b(i,1)=6 \* form(i+2,3);

end

end

for i=2:n-2

c(i,i-1)= h(i)/(h(i)+h(i+1));

c(i-1,i)= h(i)/(h(i-1)+h(i));

end

c(1,n-2) = h(1)/(h(1)+h(2));

c(n-2,1) = h(n-1)/(h(n-2)+h(n-1));

M = c\b;

M = [M0; M; Mn];

res = [];

for i = 1:n-1

s1 = conv(conv([-1, X(i+1)], [-1, X(i+1)]), [-1, X(i+1)]);

s1 = s1 \* M(i) / (6 \* h(i));

s2 = conv(conv([1, -X(i)], [1, -X(i)]), [1, -X(i)]);

s2 = s2 \* M(i+1) / (6 \* h(i));

s3 = [-1, X(i+1)] \* (Y(i) - M(i) \* h(i) \* h(i) / 6) / h(i);

s4 = [1, -X(i)] \* (Y(i+1) - M(i+1) \* h(i) \* h(i) / 6) / h(i);

ss = zeros(1, 4);

for j = 1:length(s1)

ss(4-length(s1)+j) = ss(4-length(s1)+j) + s1(j);

end

for j = 1:length(s2)

ss(4-length(s2)+j) = ss(4-length(s2)+j) + s2(j);

end

for j = 1:length(s3)

ss(4-length(s3)+j) = ss(4-length(s3)+j) + s3(j);

end

for j = 1:length(s4)

ss(4-length(s4)+j) = ss(4-length(s4)+j) + s4(j);

end

res = [res; ss];

end

end

二阶导：

function res = ddif\_fun(f, x)

% 求函数数值二阶导

e = 0.0001;

left = diff([f(x-e), f(x)])/e;

right = diff([f(x), f(x+e)])/e;

res = diff([left, right])/e;

end

一阶导：

function res = dif\_fun(f, x)

% 求函数数值一阶导

% f:需要求导的函数

% x:求x处的导数值

e = 0.00000001;

res = diff([f(x), f(x+e)])/e;

end

展示多项式：

function disp\_fun(X)

% X:系数矩阵

% 系数个数n

n = length(X);

if n == 1

fprintf('%.2f', X(1))

elseif n > 1

res = '';

for i = 1:n

if i == 1

if X(i) == -1

res = [res, '-'];

elseif X(i) == 1.0

res = res;

else

res = [res, num2str(X(i))];

end

res = [[res, 'x^'], num2str(n-i)];

elseif i < n - 1

if X(i) > 0

res = [[[[res, '+'], num2str(X(i))], 'x^'], num2str(n-i)];

elseif X(i) < 0

res = [[[res, num2str(X(i))], 'x^'], num2str(n-i)];

end

elseif i == n - 1

if X(i) > 0

res = [[[res, '+'], num2str(X(i))], 'x'];

elseif X(i) < 0

res = [[res, num2str(X(i))], 'x'];

end

else

if X(i) > 0

res = [[res, '+'], num2str(X(i))];

elseif X(i) < 0

res = [res, num2str(X(i))];

end

end

end

end

disp(res)

end

句柄函数绘图：

function f\_plot\_fun(f, X, Y, e, x0, y0)

% f:函数句柄

% X:插值节点横坐标

% Y:插值节点纵坐标

% left:左边界

% right:右边界

% e:绘制图像中的两点间隔

% x:（可选）要预测的点横坐标

% y:（可选）要预测的点纵坐标

if nargin > 4

left = min([x0 - 2 \* e, X(1) - 2 \* e]);

right = max([x0 + 2 \* e, X(length(X)) + 2 \* e]);

else

left = X(1) - 2 \* e;

right = X(length(X)) + 2 \* e;

end

x = left:e:right;

y = zeros(1, length(x));

for i = 1:length(x)

y(i) = f(x(i));

end

plot(x, y, 'b');

hold on

plot(X, Y, 'ro');

if nargin > 4

hold on

plot(x0, y0, '\*');

end

end

三次埃尔米特插值：

function res = hermite\_fun(X,Y,y0,yn)

% y0:左边界导数值

% yn:右边界导数值

x\_input = X;

y\_input = Y;

n = length(X);

y\_0 = y0;

y\_n = yn;

[~,number] = size(x\_input);

delta\_h = zeros(1,number-1);

delta\_f = zeros(1,number-1);

lambda\_ = zeros(1,number-2);

miu = zeros(1,number-2);

e = zeros(1,number-2);

for i = 1:(number-1)

delta\_h(i) = x\_input(i+1) - x\_input(i);

delta\_f(i) = (y\_input(i+1) - y\_input(i))/ delta\_h(i);

end

for i=1:number-2

lambda\_(1,i) = delta\_h(1,i+1) / (delta\_h(1,i+1) + delta\_h(1,i));

miu(1,i) = 1 - lambda\_(1,i);

e(1,i) = 3\*(lambda\_(1,i)\*delta\_f(1,i) + miu(1,i)\*delta\_f(1,i+1));

end

A = zeros(number-2,number-2);

B = zeros(number-2,1);

A(1,1) = 2;

A(1,2) = miu(1,1);

B(1,1) = e(1,1) - lambda\_(1,1) \* y\_0;

for i = 2:number-3

B(i,1) = e(1,i);

A(i,i-1) = lambda\_(1,i);

A(i,i) = 2;

A(i,i+1) = miu(1,i);

end

A(number-2,number-3) = lambda\_(1,number-2);

A(number-2,number-2) = 2;

B(number-2,1) = e(1,number-2) - miu(1,number-2)\*y\_n;

m\_matrix = A\B;

m = zeros(1,number);

m(1) = y\_0;

m(number) = y\_n;

for i = 2:number-1

m(i) = m\_matrix(i-1,1);

end

res = [];

for i = 1:n-1

s1 = conv([1/(X(i)-X(i+1)), -X(i+1)/(X(i)-X(i+1))], [1/(X(i)-X(i+1)), -X(i+1)/(X(i)-X(i+1))]);

s1 = conv(s1, [2/(X(i+1)-X(i)), 1-2\*X(i)/(X(i+1)-X(i))]);

s1 = s1 \* Y(i);

s2 = conv([1/(X(i+1)-X(i)), -X(i)/(X(i+1)-X(i))], [1/(X(i+1)-X(i)), -X(i)/(X(i+1)-X(i))]);

s2 = conv(s2, [2/(X(i)-X(i+1)), 1-2\*X(i+1)/(X(i)-X(i+1))]);

s2 = Y(i+1) \* s2;

s3 = conv([1/(X(i)-X(i+1)), -X(i+1)/(X(i)-X(i+1))], [1/(X(i)-X(i+1)), -X(i+1)/(X(i)-X(i+1))]);

s3 = m(i) \* conv(s3, [1, -X(i)]);

s4 = conv([1/(X(i+1)-X(i)), -X(i)/(X(i+1)-X(i))], [1/(X(i+1)-X(i)), -X(i)/(X(i+1)-X(i))]);

s4 = m(i+1) \* conv(s4, [1, -X(i+1)]);

res = [res; s1+s2+s3+s4];

end

end

分段插值函数绘图：

function multi\_poly\_plot\_fun(param, X, Y, e, x0, y0)

% param:多项式系数向量

% X:插值节点横坐标

% Y:插值节点纵坐标

% left:左边界

% right:右边界

% e:绘制图像中的两点间隔

% x:（可选）要预测的点横坐标

% y:（可选）要预测的点纵坐标

[n, m] = size(param);

for i = 1:n

if i == 1

if nargin > 4

left = min([x0 - 3 \* e, X(i) - 3 \* e]);

x = left:e:X(i+1);

else

x = (X(i)-3 \* e):e:X(i+1);

end

elseif i == n

if nargin > 4

right = max([x0 + 3 \* e, X(i+1) + 3 \* e]);

x = X(i):e:right;

else

x = X(i):e:X(i+1) + 3 \* e;

end

else

x = X(i):e:X(i+1);

end

y = zeros(1, length(x));

for j = 1:m

y = y + param(i, j) \* x .^ (m-j);

end

plot(x, y, 'b');

hold on

end

plot(X, Y, 'ro');

if nargin > 6

hold on

plot(x0, y0, '\*');

end

end

牛顿前向插值

function [res, param] = newton\_forward\_interpolation\_fun(X, Y, x)

% X:横坐标向量

% Y:纵坐标向量

% x:插值节点

n = length(X);

inter = X(2) - X(1);

dt = Y;

for i = 2:n

dl = zeros(1, n);

for j = i:n

dl(j) = dt(i-1, j) - dt(i-1, j-1);

end

dt = [dt; dl];

end

t = (x - X(1)) / inter;

param = zeros(1, n);

pse = zeros(1, n);

pse(n) = dt(1, 1);

param = param + pse;

ps = 1;

for i = 2:n

ps = conv(ps, [1, 2-i]);

for j = 1:length(ps)

pse(n-length(ps)+j) = ps(j);

end

pse = pse \* dt(i, i);

pse = pse ./ factorial(i-1);

param = param + pse;

end

res = 0;

for i = 1:n

res = res + param(i) \* t^(n-i);

end

end

已知重根数的牛顿迭代法：

function res = newton\_know\_root\_fun(f, x0, m, e)

% f:函数

% x0:初始迭代值

% m:m重根

% e:精度

x1 = x0 - f(x0) / dif\_fun(f, x0);

while abs(x1 - x0) >= e

x0 = x1;

x1 = x0 - m \* f(x0) / dif\_fun(f, x0);

end

res = x1;

end

未知重根数的牛顿迭代法：

function res = newton\_unknow\_root\_fun(f, x0, e)

% f:函数

% x0:初始迭代值

% e:精度

x1 = x0 - f(x0) / dif\_fun(f, x0);

while abs(x1 - x0) >= e

x0 = x1;

x1 = x0 - dif\_fun(f, x0) \* f(x0) / (dif\_fun(f, x0) \* dif\_fun(f, x0) - f(x0) \* ddif\_fun(f, x0));

end

res = x1;

end

线性插值：

function res = piecewise\_linear\_interpolation\_fun(X, Y)

% X:横坐标向量

% Y:纵坐标向量

n = length(X);

res = zeros(n-1, 2);

for i = 1:n-1

res(i, 1) = Y(i) / (X(i) - X(i+1)) + Y(i+1) / (X(i+1) - X(i));

res(i, 2) = -X(i+1) \* Y(i) / (X(i) - X(i+1)) - X(i) \* Y(i+1) / (X(i+1) - X(i));

end

多项式求值：

function y = poly\_value\_fun(param, x)

y = 0;

for j = 1:length(param)

y = y + param(j) \* x ^ (length(param)-j);

end

end

SOR迭代法：

function x = sor\_fun(a, b, n, x0, e, w)

% a:系数矩阵

% b:右边向量

% x0:初始向量(列向量)

% e:精度

% w:松弛因子

% 最大迭代次数M

M = 100000;

m = 0;

x = zeros(n, 1);

while m <= M

m = m + 1;

for i = 1:n

sum\_ax = 0;

for j = 1:i-1

sum\_ax = sum\_ax + a(i, j) \* x(j);

end

for j = i+1:n

sum\_ax = sum\_ax + a(i, j) \* x0(j);

end

if i == 0

x(i) = -(sum\_ax - b(i)) / a(i, i);

elseif i > 0

x(i) = -w \* (sum\_ax - b(i)) / a(i, i) + (1 - w) \* x0(i);

end

end

if norm(x - x0, 1) < e

break

end

x0 = x;

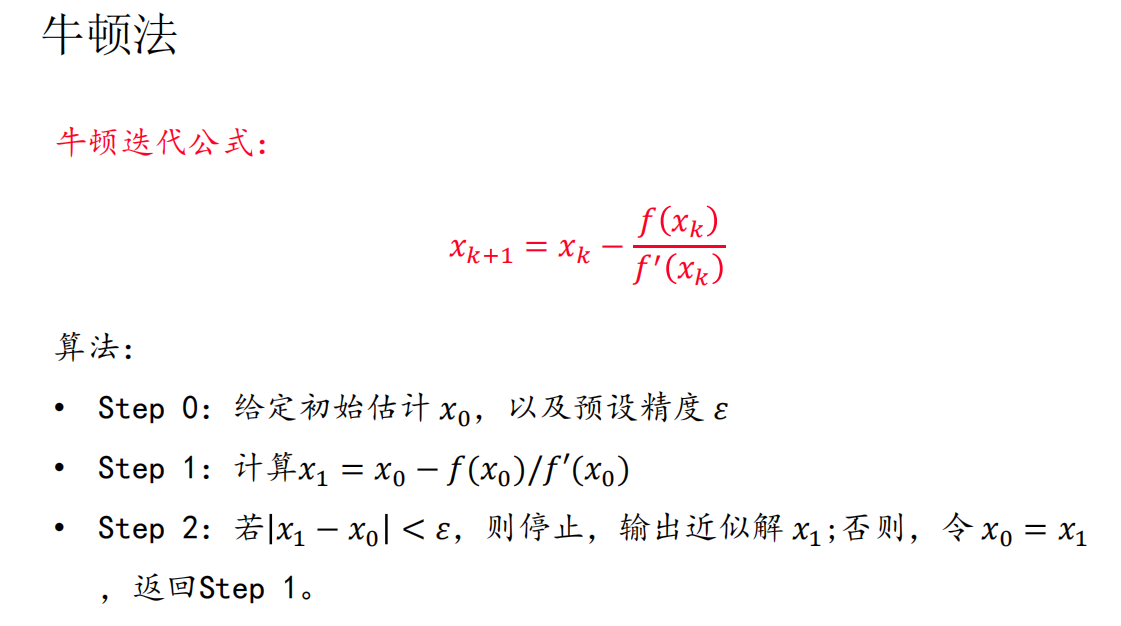
end

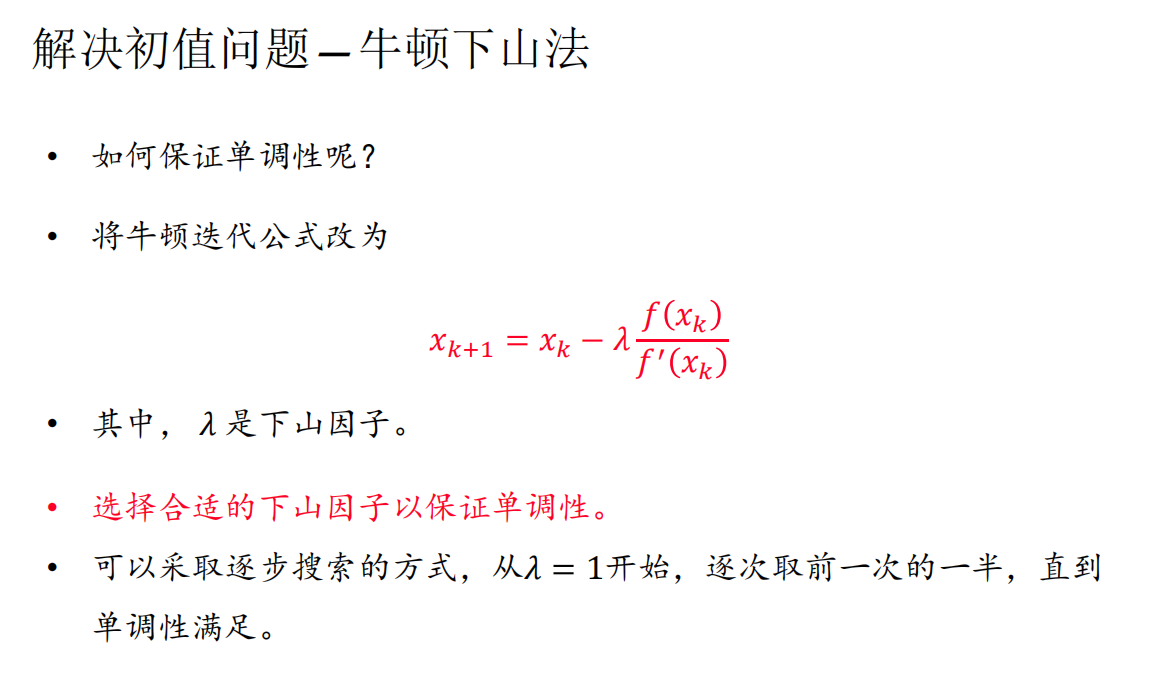
if m > M

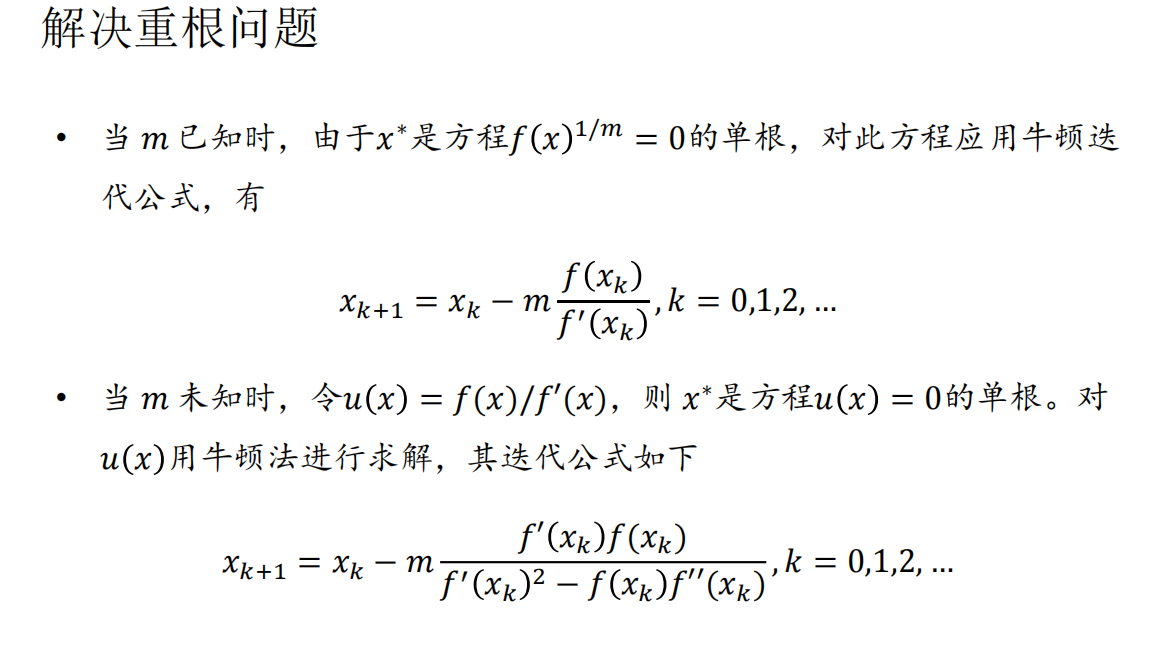
disp('达到最大循环次数')

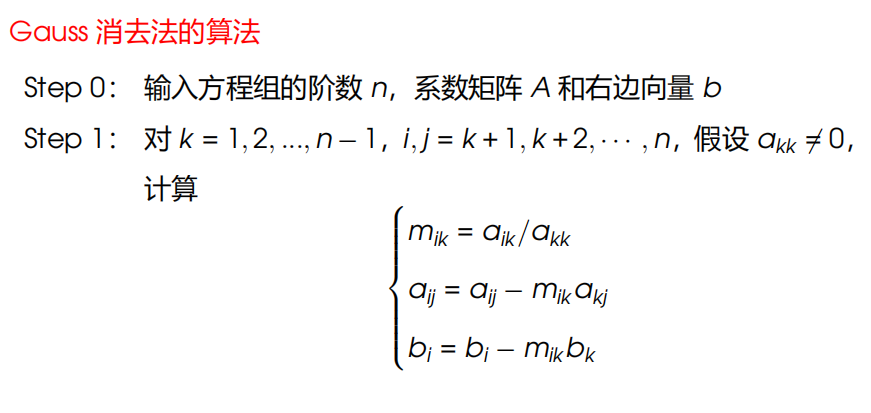
end

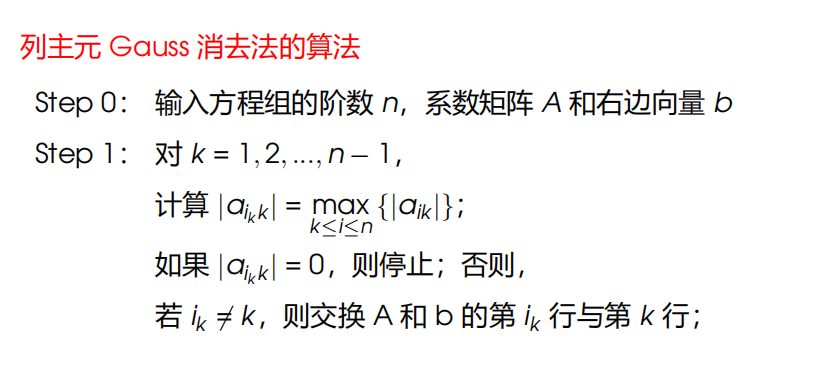
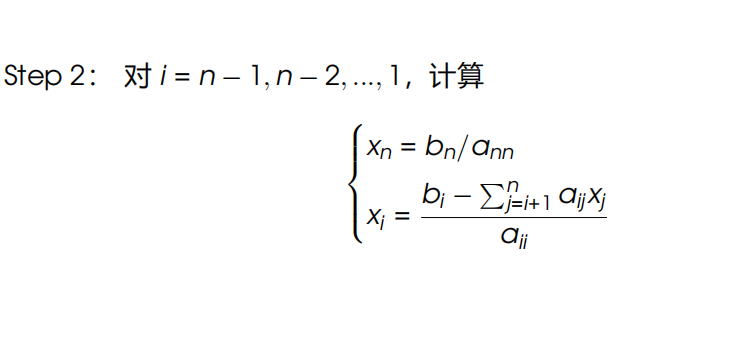
end

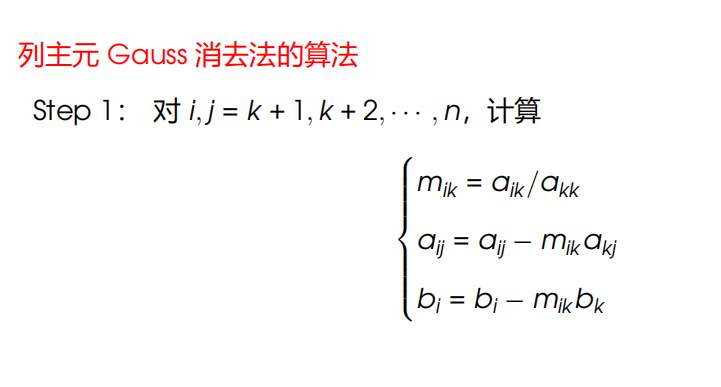


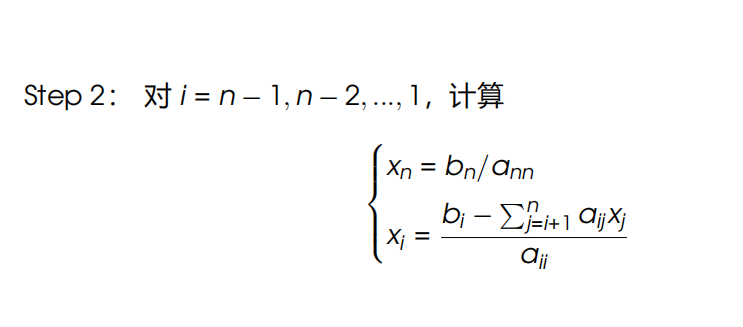


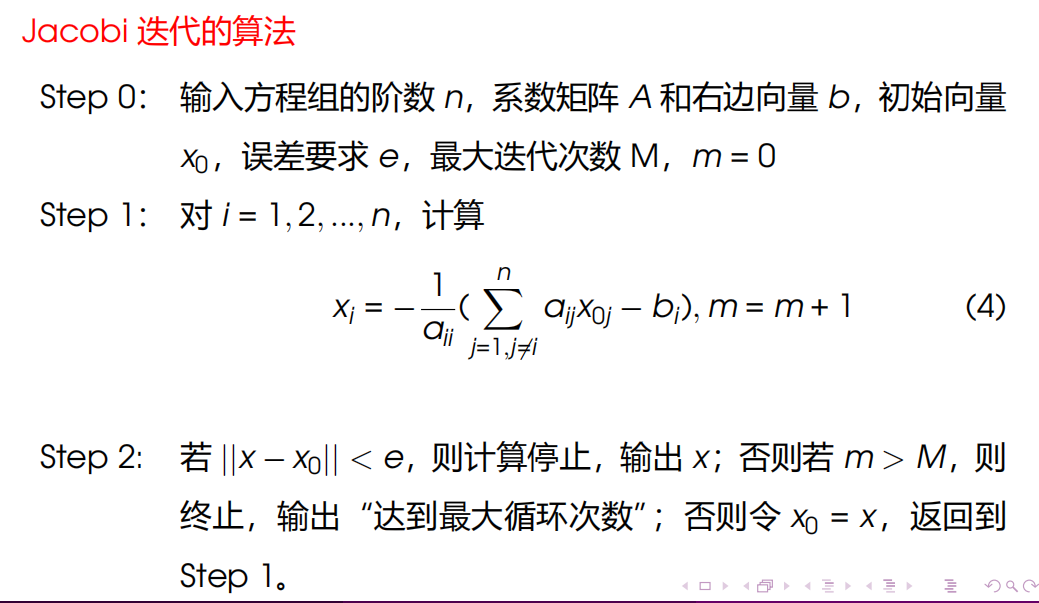


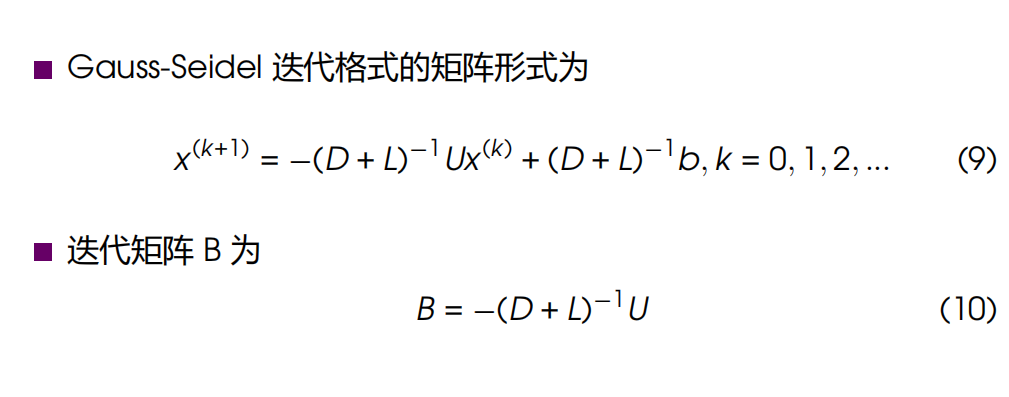
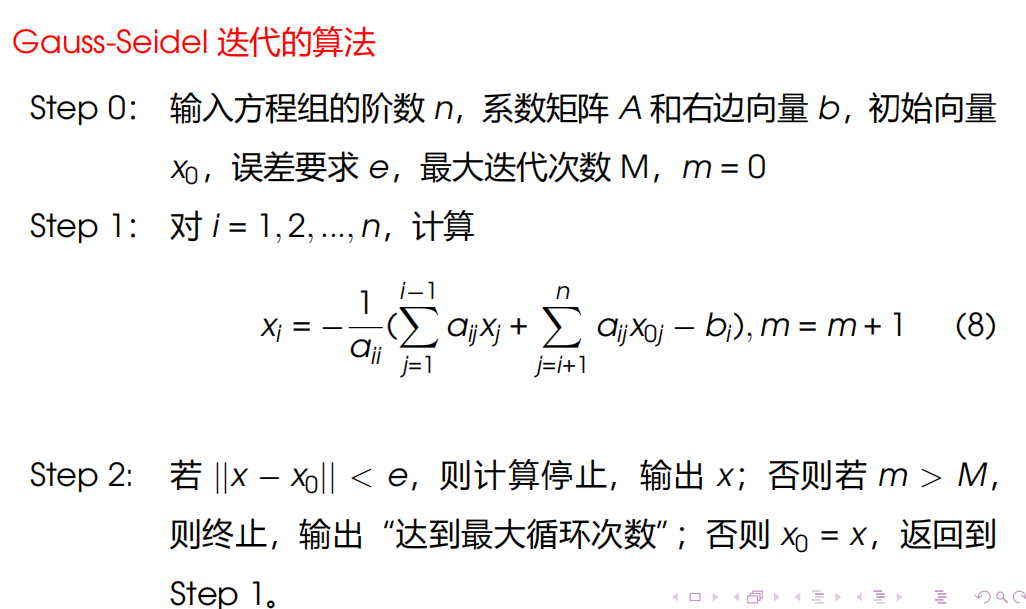
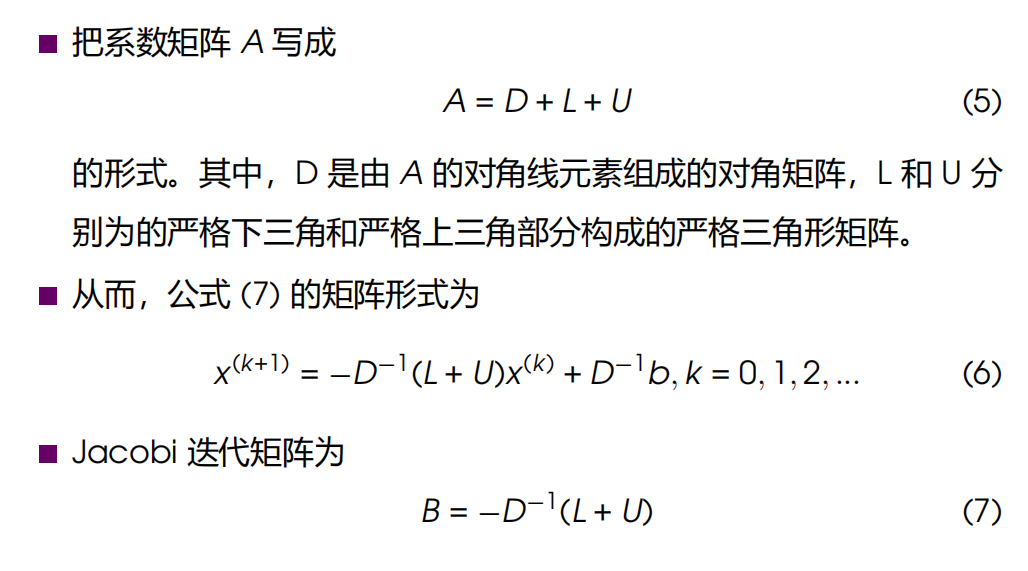




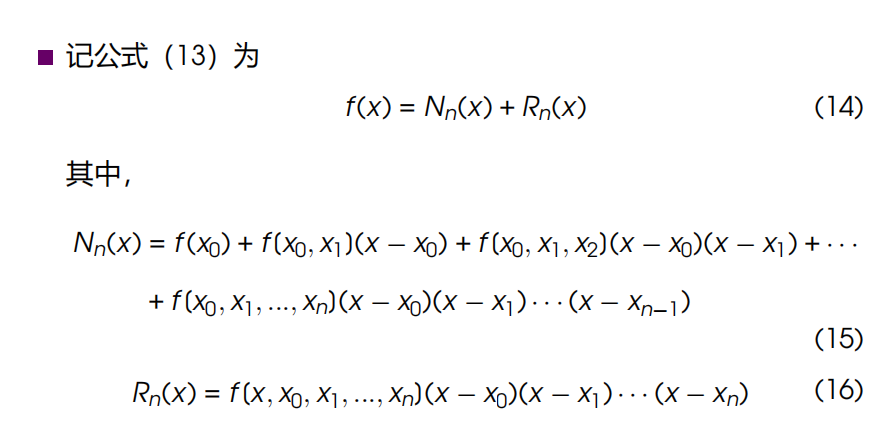
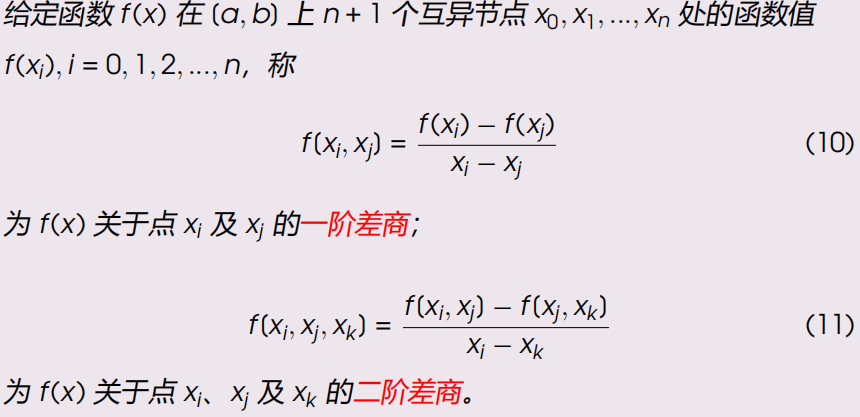
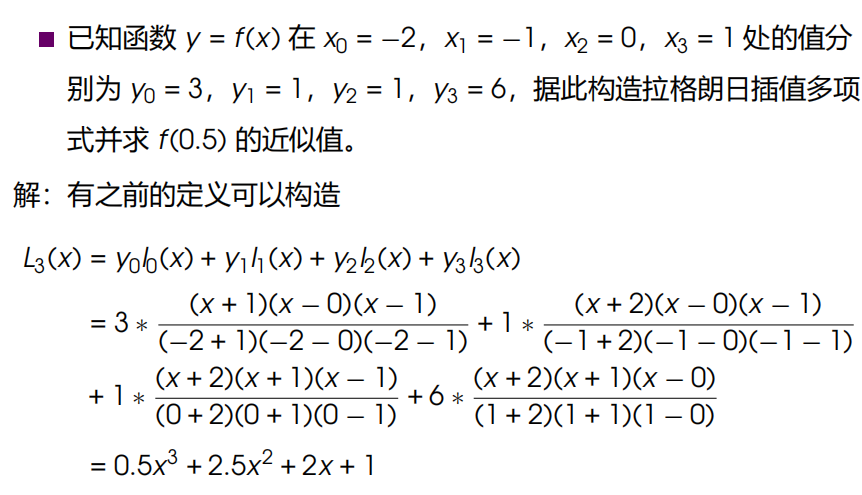
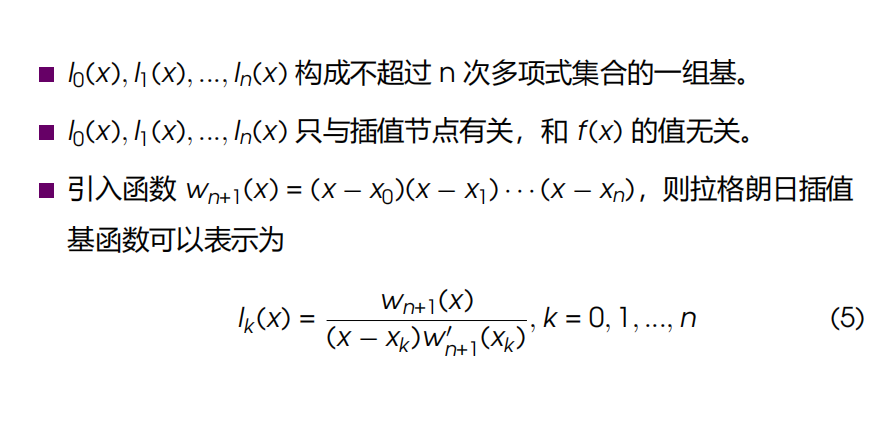




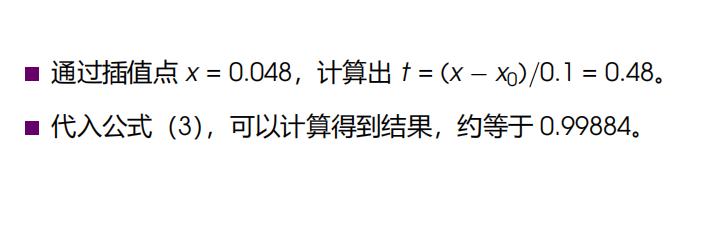
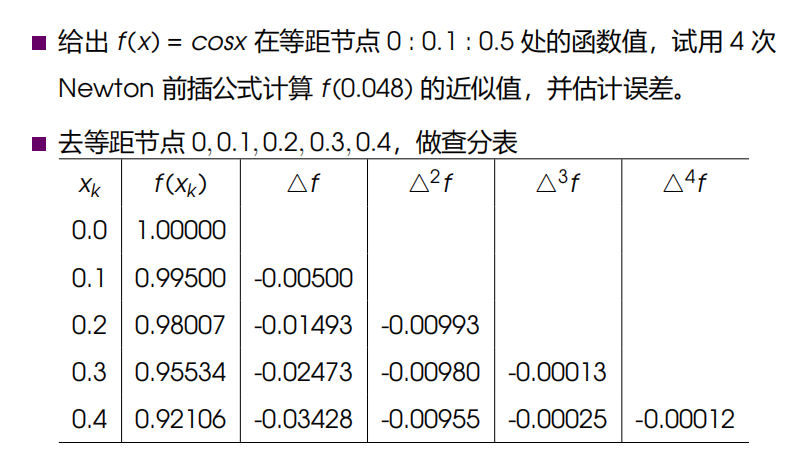
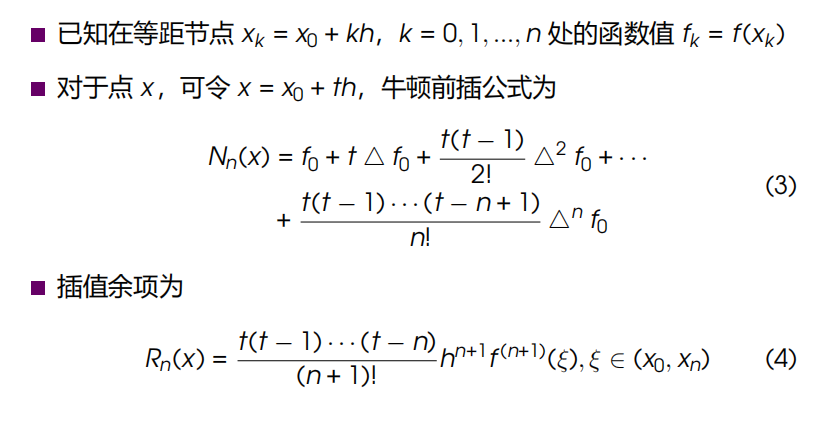




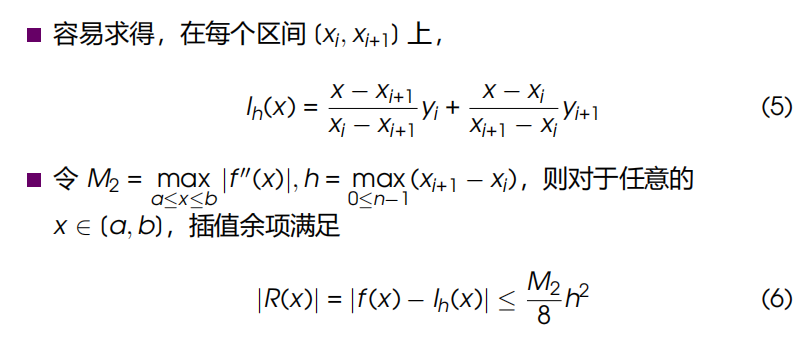
拉格朗日插值：



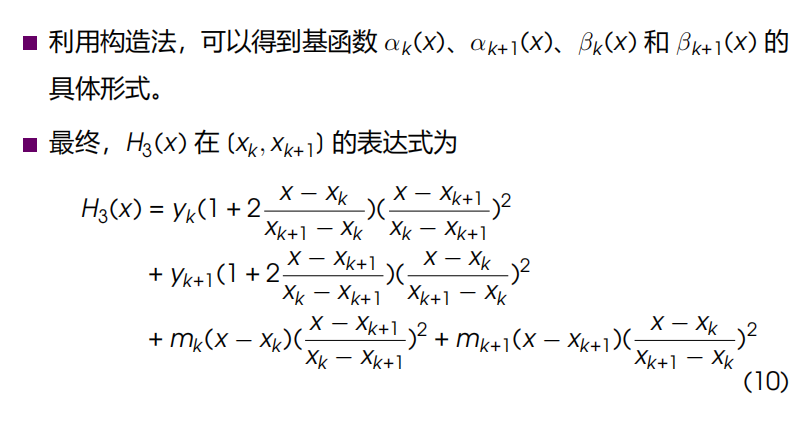
牛顿前插：



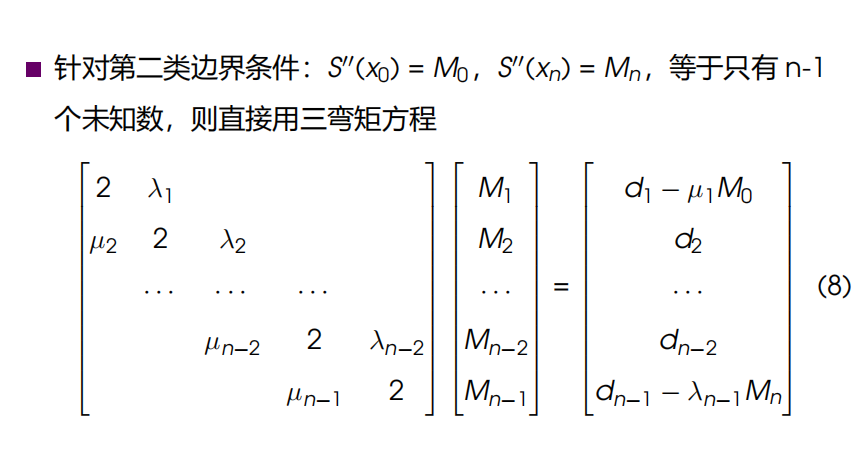
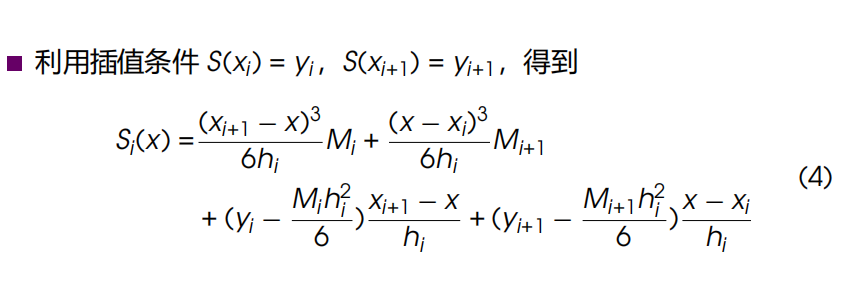
分段线性插值：



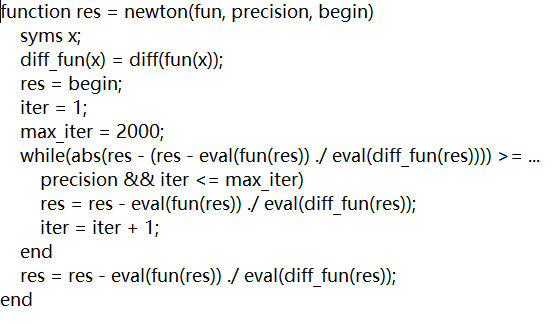
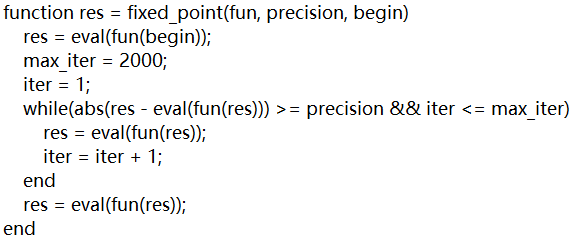
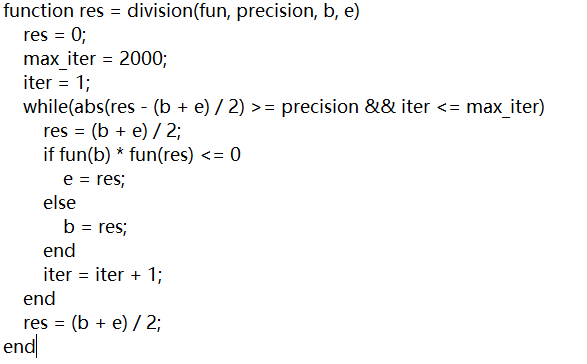
埃尔米特插值：



三次样条插值：



过去题目代码：



%% 用二分法、不动点迭代（与牛顿法不一样）、牛顿法求解以下非线性方程。

% （1）sin x = 6x + 5

% （2）lnx + x^2 = 3

% （3）e^x + x = 7

clear;clc;

% 终止条件为前后两次近似解之差小于10^?3

precision = 0.001;

% 声明自变量x

syms x;

%% sin x = 6x + 5

disp('方程一：sin x = 6x + 5');

% 二分法

fun(x) = sin(x) - 6\*x - 5;

division\_res = division(fun, precision, -1, 0);

disp('二分法：');

disp('观察可知该方程在-1和0间有解');

disp('求解结果：');

disp(division\_res);

% 不动点法

fun(x) = (sin(x) - 5) / 6;

fixed\_point\_res = fixed\_point(fun, precision, -1);

disp('不动点迭代法：');

disp('将方程变形为x = (sin(x) - 5) / 6，取x = -1作为初始迭代解');

disp('求解结果：');

disp(fixed\_point\_res);

% 牛顿法

fun(x) = sin(x) - 6\*x - 5;

newton\_res = newton(fun, precision, -1);

disp('牛顿法：');

disp('取x = -1作为初始迭代解');

disp('求解结果：');

disp(newton\_res);

%% ln x + x^2 = 3

disp('方程二：lnx + x^2 = 3');

% 二分法

fun(x) = log(x) + x^2 - 3;

division\_res = division(fun, precision, 1, 2);

disp('二分法：');

disp('观察可知该方程在1和2间有解');

disp('求解结果：');

disp(division\_res);

% 不动点法

fun(x) = sqrt(3 - log(x));

fixed\_point\_res = fixed\_point(fun, precision, 2);

disp('不动点迭代法：');

disp('将方程变形为x = (3 - ln(x))^0.5，取x = 2作为初始迭代解');

disp('求解结果：');

disp(fixed\_point\_res);

% 牛顿法

fun(x) = log(x) + x^2 - 3;

newton\_res = newton(fun, precision, 2);

disp('牛顿法：');

disp('取x = 2作为初始迭代解');

disp('求解结果：');

disp(newton\_res);

%% e^x + x = 7

disp('方程三：e^x + x = 7');

% 二分法

fun(x) = exp(x) + x - 7;

division\_res = division(fun, precision, 1, 2);

disp('二分法：');

disp('观察可知该方程在1和2间有解');

disp('求解结果：');

disp(division\_res);

% 不动点法

fun(x) = log(7 - x);

fixed\_point\_res = fixed\_point(fun, precision, 2);

disp('不动点迭代法：');

disp('将方程变形为x = ln(7 - x)，取x = 2作为初始迭代解');

disp('求解结果：');

disp(fixed\_point\_res);

% 牛顿法

fun(x) = exp(x) + x - 7;

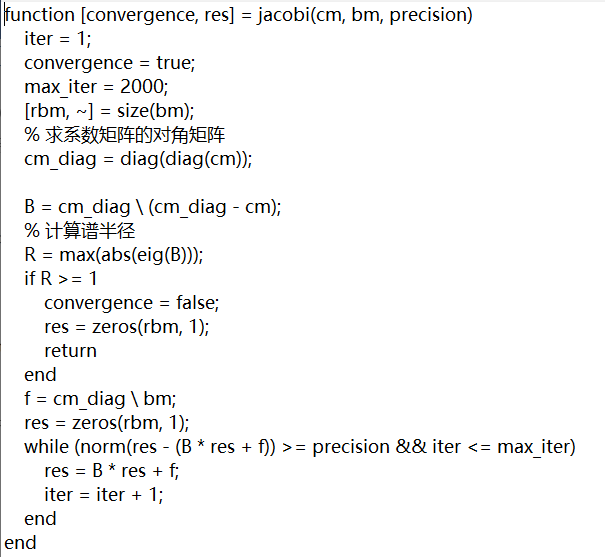
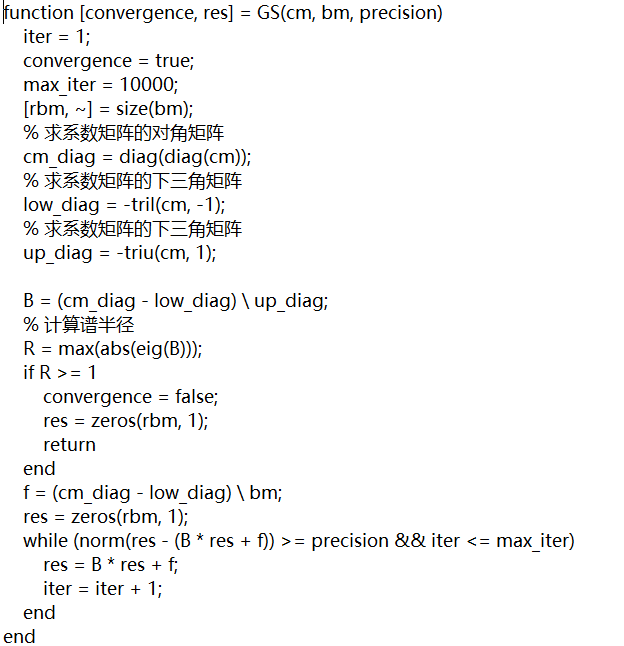
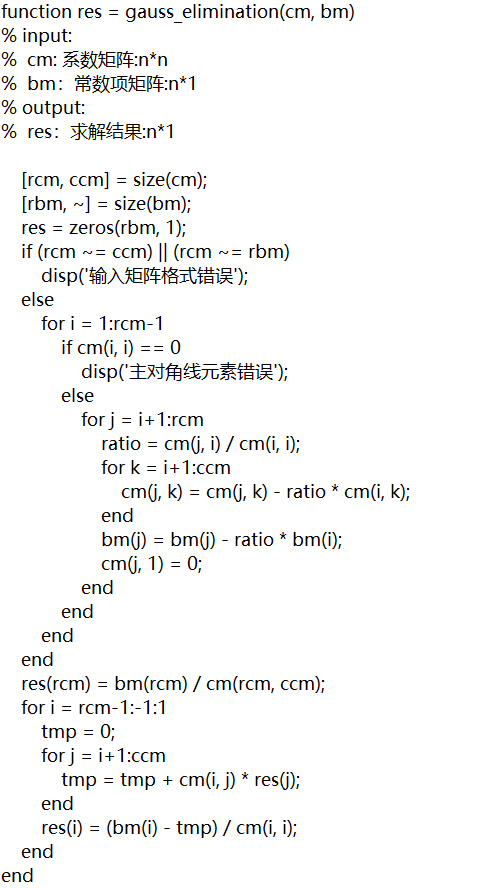
newton\_res = newton(fun, precision, 2);

disp('牛顿法：');

disp('取x = 2作为初始迭代解');

disp('求解结果：');

disp(newton\_res);

%% 用高斯消去法、Jacobi 迭代、G-S 迭代求解以下线性方程组。

clear;clc;

precision = 0.001;

%% 第一问

% 2x - 2y - z = ?2

% 4x + y - 2z = 1

% -2x + y - z = ?3

disp('第一问：');

% 系数矩阵cm

cm = [2, -2, -1;4, 1, -2;-2, 1, -1];

% 常数项矩阵

bm = [-2;1;-3];

% 高斯消去法

gauss\_res = gauss\_elimination(cm, bm);

disp('高斯消去法结果：');

disp(gauss\_res);

% Jacobi迭代

[convergence, jacobi\_res] = jacobi(cm, bm, precision);

disp('Jacobi迭代结果：');

if convergence

disp(jacobi\_res);

else

disp('谱半径不小于1，迭代不收敛');

end

% G-S迭代

[convergence, GS\_res] = GS(cm, bm, precision);

disp('G-S迭代结果：');

if convergence

disp(GS\_res);

else

disp('谱半径不小于1，迭代不收敛');

end

%% 第二问

disp('第二问：');

% 系数矩阵cm

cm = zeros(100, 100);

for i = 1:100

cm(i, i) = 3;

if i < 100

cm(i+1, i) = -1;

cm(i, i+1) = -1;

end

end

% 常数项矩阵

bm = ones(100, 1);

bm(1) = 2;

bm(100) = 2;

% 高斯消去法

gauss\_res = gauss\_elimination(cm, bm);

disp('高斯消去法结果：');

disp(gauss\_res);

% Jacobi迭代

[convergence, jacobi\_res] = jacobi(cm, bm, precision);

disp('Jacobi迭代结果：');

if convergence

disp(jacobi\_res);

else

disp('谱半径不小于1，迭代不收敛');

end

% G-S迭代

[convergence, GS\_res] = GS(cm, bm, precision);

disp('G-S迭代结果：');

if convergence

disp(GS\_res);

else

disp('谱半径不小于1，迭代不收敛');

end