**Chapter 7 Optimization**

**optimset //显示控制参数**

optimset fminunc //显示fminunc的控制参数

opt = optimset;

opt.Display = "Iter";

options = optimoptions('fminunc','GradObj','on')

**主要控制参数(对大规模/中等规模算法均有效)**

**Diagnostics ‘on’ | {‘off’} //是否显示诊断信息**

**Display 'off' | 'iter' |**

**‘final’ | ‘notify‘ //显示信息的级别**

**GradObj ‘on’ | {‘off’}//是否采用分析梯度？？**

**Jacobian ‘on’ | {‘off’}**

**//采用分析Jacob阵（用于约束优化中）**

**LargeScale {‘on’} | ‘off’ //是否采用大规模算法**

**MaxFunEvals 最大函数调用次数**

**MaxIter 最大迭代次数**

**TolCon 约束的控制精度（用于约束优化中）**

**TolFun 函数值的控制精度**

**TolX 解的控制精度**

**最一般的输出形式**

**[x,f,exitflag,out,grad,hess]=fminunc(...)**

**f 目标函数值**

**exitflag >0收敛,0达到函数或迭代次数, <0不收敛**

**Output**

**iterations 实际迭代次数**

**funcCount 实际函数调用次数**

**algorithm 实际采用的算法**

**cgiterations 实际PCG迭代次数（大规模算法用）**

**stepsize 最后迭代步长（中等规模算法用）**

**firstorderopt 一阶最优条件（梯度的范数）**

**message 收敛信息等**

**grad 目标函数的梯度**

**hess 目标函数的Hessian矩阵**

例1： min (3sinx +ｘ)，　ｘ＼ｉｎ［１，８］

echo on

clear all;clc;

x1=1;x2=8;

f=inline('3\*sin(x)+x');

[x,fv]=fminbnd(f,x1,x2);

echo off

[x1,fv1]=fminbnd('3\*sin(x)+x',1,8);

[x2,fv2]=fminbnd(inline('3\*sin(x)+x'),1,8); %初值

[x,x1,x2;fv,fv1,fv2]

ezplot('3\*sin(x)+x',[1,8])

ginput(1)



function y=exam0702fun(x,a,b)

y=x(1)^2/a+x(2)^2/b;

x0=[1,1];a=2;b=2;

[x1,f1,e1,out1]=fminunc(@exam0702fun,x0,[],a,b)

pause

[x2,f2,e2,out2]=fminsearch(@exam0702fun,x0,[],a,b)

function [f,g]=exam0702fun2(x,a,b)

f=x(1)^2/a+x(2)^2/b;

g=[2\*x(1)/a;2\*x(2)/b];

options = optimoptions('fminunc','GradObj','on')

[x1,f1,e1,out1]=fminunc(@exam0702fun2,x0,options,a,b)

例:4：**用optimset控制参数选择, 求解例2**

clear all

a=10;b=1;x0=[1,1]; %初始值

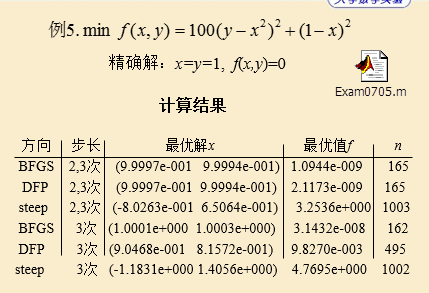
opt=optimset('fminunc'); %程序fminunc缺省的控制参数

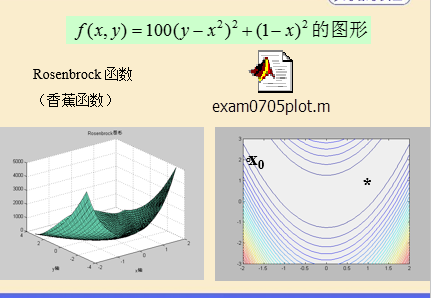
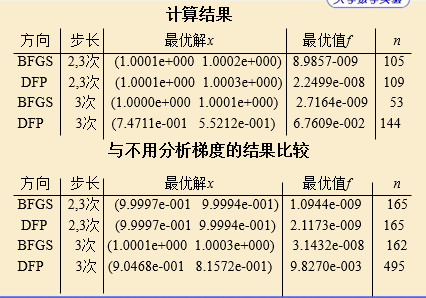
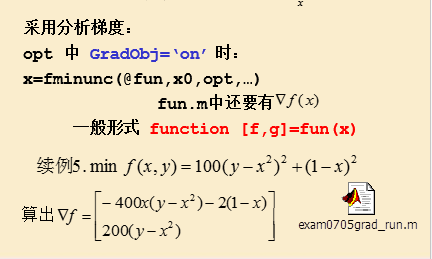
%opt=optimset(opt,'Disp','iter'); %设定输出中间结果

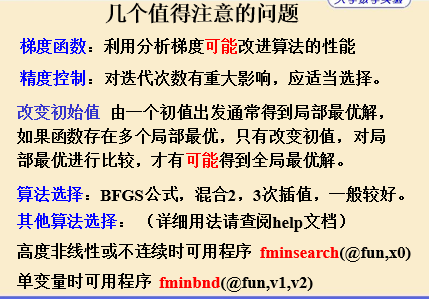
%也可直接写成opt.Display='iter' （此时Display不能简写，且区分大小写字母）

opt=optimset(opt,'tolx',1e-10,'tolf',1e-10); %设定控制精度

[x,f,ef,out]=fminunc(@exam0702fun,x0,opt,a,b)







约束优化问题的求解

