系统优化期中大作业

17363011 陈政培 作业1

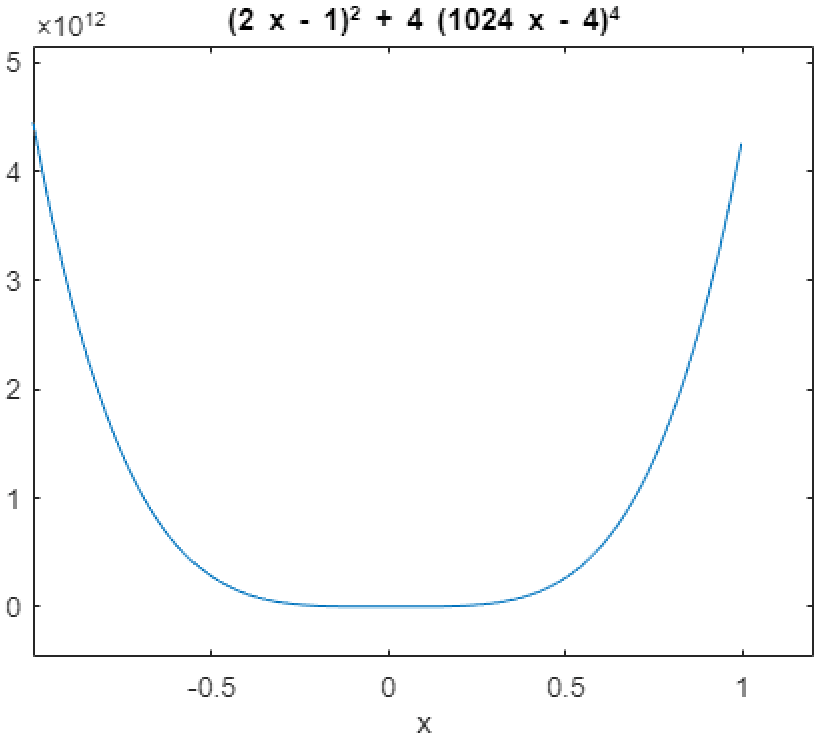
第一题——ffib.m

利用MATLAB编程实现：Newton切线法求解 $g(x)=0$ 的根，初始值为 $x(0)=1$， $\varepsilon=10^{-5}$。

$$g(x)=(2x-1)^2+4(4-1024x)^4$$

题目分析

　本题就是将牛顿法付诸代码实现，但是经过手动模拟，可能实际上有根，但是在计算机上只能得到一个近似解



上图为生成的 `g(x)` 函数大致图像，对于此类牛顿切线法，就是一维搜索法，实现课本上的计算流程即可

matlab程序

```
function ffib
x0=1;
eps=1e-5;
syms x;
fx=(2*x-1)^2+4*(4-1024*x)^4;
format long;
dfx=diff(fx);
ddfx=diff(dfx);

tol=1;
k=0;

while tol>eps
    %fprintf('第%d次迭代：',k);
    g=subs(dfx,symvar(dfx),x0);
    h=subs(ddfx,symvar(ddfx),x0);
    x1=x0-g/h;
    x1=vpa(x1,8);
    k=k+1;
```

```
    tol=abs(x1-x0);%tol=abs(dfx);
    x0=x1;
end
disp('结果如下：best_x =')
disp(x1)
disp(subs(fx,symvar(fx),x1))
format short;
end
```

运行结果

```
>> ffib
迭代次数：
    24

结果如下：best_x =
0.0039686168

0.98425507108153664238653021644796
```

迭代24次，得到最优点 0.0039686168，最优值 0.984255071081

第二题MinValue_Gold.m

利用MATLAB编程实现：黄金分割法将如下函数的最佳步长所在的区间压缩在0.01之内。初始值为 $\mathbf{x}(0)=\begin{bmatrix}0.8 & -0.25\end{bmatrix}^T$。

$$f(x)=\frac{1}{2}\mathbf{x}^T\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}\mathbf{x}$$

题目分析

若我们已经知道了一个下降方向 `dk`，就只需要求参数 α 使其满足一维优化问题 `minf(xk+αdk)` 的解，令 `φ(α)=f(xk+αdk)` ，则 `φ'(α)=∇f(xk+αdk)Tdk=0` 求解该线性方程组即可。此时问题转换为了求 `φ(α)=f(x0+αdk)` 的极值问题，用黄金分割法求解即可。

`φ(α)=f(x0+αdk)=0.5 * (x0+αdk) ' * H * (x0+αdk)` ，`dk= -Hx`

所以 `φ(α)=0.5*(0.8-27/20α) *((0.8-27/20α) * 2+(-0.25-0.3*α))+0.5 *(-0.25-0.3 *α) * ((0.8-27/20α)+(-0.25-0.3*α) * 2)`

直接将上述方程带入代码中计算即可得到压缩后的步长

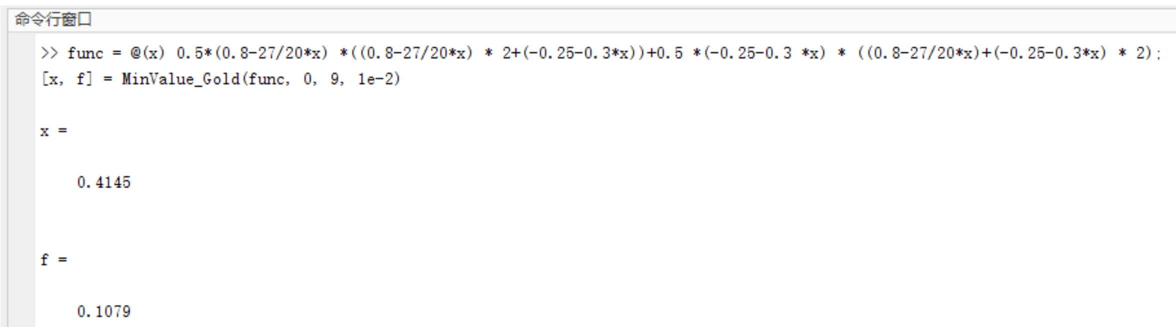
matlab程序

```
function [xo, fo] = MinValue_Gold(func, a, b, eps)
if nargin < 3
   error('输入参数不足! ');
end
if nargin == 3
   eps = 1e-6;
end
% 初始情况
a1 = a + 0.382*(b - a);
a2 = a + 0.618*(b - a);
f1 = func(a1);
f2 = func(a2);
ite = 0;
while abs(b - a) >= eps
   if f1 < f2
      b = a2;
      a2 = a1;
      f2 = f1;
      a1 = a + 0.382*(b - a);
      f1 = func(a1);
   else
      a = a1;
      a1 = a2;
      f1 = f2;
      a2 = a + 0.618*(b - a);
      f2 = func(a2);
   end
   ite = ite + 1;
end
xo = 0.5*(a + b);
fo = func(xo);
```

运行结果

人为给定一个起始范围 [0,9]， 按照0.01的精度，在命令行窗口输入以下代码开始计算

```
func = @(x) 0.5*(0.8-27/20*x) *((0.8-27/20*x) * 2+(-0.25-0.3*x))+0.5 *(-0.25-0.3*x) * ((0.8-27/20*x)+(-0.25-0.3*x) * 2);
[x, f] = MinValue_Gold(func, 0, 9, 1e-2)
```


命令行窗口

```
>> func = @(x) 0.5*(0.8-27/20*x) *((0.8-27/20*x) * 2+(-0.25-0.3*x))+0.5 *(-0.25-0.3 *x) * ((0.8-27/20*x)+(-0.25-0.3*x) * 2);
[x, f] = MinValue_Gold(func, 0, 9, 1e-2)

x =

    0.4145


f =

    0.1079
```

结果显示最佳步长 `α` 为0.4145

第三题——GDMin.m

利用MATLAB编程实现：最速下降法求解如下函数的极小点，初始值为$\mathbf{x}(0)=\begin{bmatrix}-2 & 2\end{bmatrix}^T$。当函数的梯度范数小于$10^{-4}$时，停迭代。

$$f(x)=100(x_2-x_1^2)^2+(1-x_1)^2$$

题目分析

最速下降法解决问题的核心难点，在于找到每一个迭代步骤中的，当前搜索点 `xk` ，搜索方向 `dk` ，以及搜索步长 `ak`

如果最速下降法的步长使用 line-search，和 `Matlab` 符号运算，计算非常耗时。所以改用 `argmin` 来求步长

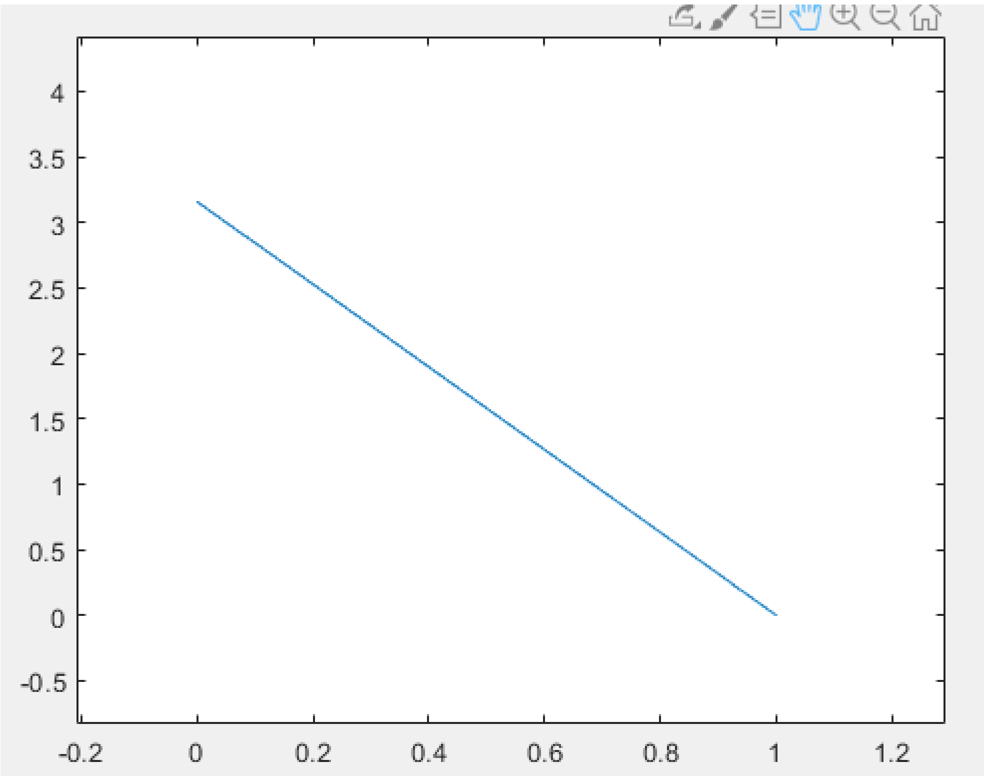
matlab程序

```
clc; clear;

f = @(x) 100*(x(2)-x(1).^2)^2+(1-x(1))^2; %待求函数，x1,x2,x3...
% f = @(x) x(1).^2+2*x(2).^2;
paraNum = 2; %函数参数的个数，x1,x2,x3...的个数
x0 = [-2,2]; %初始值
tol = 1e-4; %迭代容忍度
flag = inf; %结束条件
error = []; %函数变化

while flag > tol
    p = g(f,x0,paraNum); %列向量
    if norm(p) < tol
            buChang = 0;
    else
        buChang = argmin(f,x0,p,paraNum); %求步长，line search: argmin function
    end
    x1 = x0-buChang.*p';
    flag = norm(x1-x0);
    error = [error,flag];
    x0 = x1;
end
plot(0:length(error)-1,error)
disp('结果如下：best_x =')
disp(x0)

function [f_grad] = g(f,x0,paraNum) %求搜索方向
temp = sym('x',[1,paraNum]);
f1=f(temp);
z = gradient(f1);
f_grad = double(subs(z,temp,x0));
end

function [x] = argmin(f,x0,p,num) %求步长
temp = sym('x',[1,num]);
f1=f(x0 - temp.*p');
for i = 1:num
```

```
    temp(i) = diff(f1,temp(i));
end
%转换为double类型
jieGuo = solve(temp);
jieGuo = struct2cell(jieGuo);
x = zeros(1,num);
for i = 1:num
    x(i) = double(jieGuo{i});
end
end
```

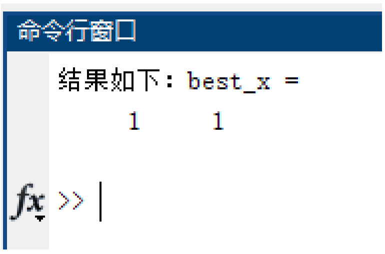
运行结果

误差收敛曲线



使用 `argmin` 后收敛速度明显更快，更快的迭代出结果

运算结果



最优点 `[1 1]` ，最优值 0

第四题——ddd.m

利用MATLAB编程实现：共轭梯度法求解如下方程组的根，初始值为 $\mathbf{x}(0)=\begin{bmatrix}0 & 0\end{bmatrix}^T$.

$$\begin{cases} x_1 - x_2 = 3 \\ x_1 + x_2 = 6 \end{cases}$$

题目分析

在解线性方程组Ax=b时，因为系数矩阵A是正定的稀疏矩阵，所以采用了共轭梯度法解。也计入了 `x=inv(A)*b` 的基础解题方法，用以检验答案。

matlab程序

```
A=[1 -1;1 1];
b=[3 6]';
N=length(b);     %解向量的维数
fprintf('库函数计算结果: ');
x=inv(A)*b       %库函数计算结果
x=zeros(N,1);%迭代近似向量
eps=0.0000001;%精度
r=b-A*x;d=r;
for k=0:N-1
    fprintf('第%d次迭代: ',k+1);
    a=(norm(r)^2)/(d'*A*d)
    x=x+a*d
    rr=b-A*x;    %rr=r(k+1)
    disp(norm(rr))
    if (norm(rr)<=eps)||(k==N-1)
        break;
    end
    B=(norm(rr)^2)/(norm(r)^2);
    d=rr-B*d;
    r=rr;
end
```

运行结果

```
命令行窗口

>> ddd
库函数计算结果:
x =

    4.5000
    1.5000

第1次迭代:
a =

    1.0000



x =

    3.0000
    6.0000

第2次迭代:
a =

    0.5000



x =

    4.5000
    1.5000
```

迭代结果与答案一致。`x = [ 4.5 1.5 ] '`

第五题——DFP.m

利用MATLAB编程实现：DFP算法求如下函数的极小点。令起始点分别为 $\mathbf{x}(0)=[0 \quad 0]^T$ 和 $\mathbf{x}(0)=[1.5 \quad 1]^T$，$H_0=I_2$。分析在这两个起始点下，算法是否收敛到同一点，如果不是，请给出原因。

$$f(x)=\frac{x_1^2}{4}+\frac{x_2^2}{2}-x_1x_2+x_1-x_2$$

题目分析

DFP法的解答过程中需要满足一个拟牛顿条件，于是通过DFP修正公式可以得到下一个搜索方向

而函数的Hessian矩阵是负定矩阵，函数特征值为-1/7，所以 `f` 不是凸函数，`x*` 为局部极小点

matlab程序

```
function [best_x,best_fx,count]=DFP(x0,ess)
colormap Jet

syms x1 x2 t;
f=x1*x1/4+x2*x2/2-x1*x2+x1-x2;
fx=diff(f,x1);%求表达式f对x1的一阶求导
fy=diff(f,x2);%求表达式f对x2的一阶求导
fi=[fx fy];%构造函数f的梯度函数
%初始点的梯度和函数值
g0=subs(fi,[x1 x2],x0);
f0=subs(f,[x1 x2],x0);
H0=eye(2); %输出x0,f0,g0
x0
f0
g0
xk=x0;
fk=f0;
gk=g0;
Hk=H0;
k=1;
while(norm(gk)>ess)%迭代终止条件||gk||<=ess
    disp('************************************************************')
        disp(['第' num2str(k) '次寻优'])
%确定搜索方向
        pk=-Hk*gk';
%由步长找到下一点x(k+1)
        xk=xk+t*pk';
        f_t=subs(f,[x1 x2],xk); %构造一元搜索的一元函数φ(t) %由一维搜索找到最优步长
        df_t=diff(f_t,t);
        tk=solve(df_t);
if tk~=0
    tk=double(tk);
else
    break;
end
%计算下一点的函数值和梯度
        xk = subs(xk,t,tk)
        fk=subs(f,[x1 x2],xk)
```

```
        gk0=gk;
        gk=subs(fi,[x1 x2],xk)
%DPF校正公式，找到修正矩阵
        yk=gk-gk0;
        sk=tk*pk';
        Hk=Hk-(Hk*yk'*yk*Hk)/(yk*Hk*yk')+sk'*sk/(yk*sk')%修正公式
        k=k+1;
end

disp('结果如下：')
best_x=xk;%最优点
best_fx=fk;%最优值
count=k-1;
end
```

运行结果

在命令行窗口分别键入两个不同的起始点

```
[best_x,best_fx,count]=DFP([0 0],1e-6)
```

```
结果如下：

best_x =

[ 0, 1]


best_fx =

-1/2


count =

     2
```

```
[best_x,best_fx,count]=DFP([1.5 1],1e-6)
```

> 结果如下：
>
> best_x =
>
> [0, 1]
>
> best_fx =
>
> -1/2
>
> count =
>
> 2

两者都收敛到 `[0 1]` 点，综合题目分析，得到两个初始点都收敛到同一点，但是是局部极小点

参考文献

1. `一维搜索方法与MATLAB实现` https://wenku.baidu.com/view/5b278f3ace17552707221 92e453610661fd95a46.html#
2. `深入浅出最优化(2) 步长的计算方法` https://blog.csdn.net/weixin_43441742/article/details/105963383
3. `最优化方法之修正牛顿法matlab源码(含黄金分割法寻找步长)` https://wenku.baidu.com/view/3e399d36a8114431b90dd864.html
4. `最速梯度下降法及matlab实践` https://blog.csdn.net/wangdingqiaoit/article/details/23454769
5. `个人代码博客` https://www.cnblogs.com/kexve/p/11737898.html
6. `并行解线性方程组Ax=b` https://www.ilovematlab.cn/thread-479911-1-1.html
7. `拟牛顿法-DFP算法举例与matlab代码实现(转载+整理)` https://blog.csdn.net/appleyuchi/article/details/97395358