CS450 XiaowenLin xlin11

5.4 p248

$g(x) = x - \frac{f(x)}{f(x)'} = x - \frac{1/x-y}{-x^{-2}} = 2x - yx^2$

So this is the iterative scheme for approximating the reciprocal of y, $x_{k+1} = g(x) = 2x_k - yx_k^2$

5.12 p249

$g(x) = x - f(x)/d$

(a) $g(x)' = 1 - f(x)'/d$

When $|g(x)'| < 1$, it will convergent. That is, $1 > f(x)'/d > 0 or 2 > f(x)'/d > 1$

When $f(x)' > 0$, $d > f(x)' or f(x)' > d > f(x)'/2$

When$f(x)' < 0$, $d < f(x)' or f(x)' < d < f(x)'/2$

(b) Asymptotic convergence rate of fixed-point iteration is usually linear, with constant$C = |g(x)'| = |1 - f(x)'/d|$.

(c) $g(x*)' = 0 \Rightarrow d = g(x*)'$

5.13 p249

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1 - 1 \\ x_1x_2 - 1 \end{bmatrix}$$

$$\therefore \boldsymbol{J}(\boldsymbol{x}) = \frac{\partial f_i(\boldsymbol{x})}{\partial x_j} = \begin{bmatrix} 1 & 0 \\ x_2 & x_1 \end{bmatrix}$$

$$|\boldsymbol{J}(\boldsymbol{x})| = | \begin{matrix} 1 & 0 \\ x_2 & x_1 \end{matrix} | = x_1 - 0 = 0$$

When $x_1 = 0$and$x_2$is arbitary value, Jacobian is singular.

When Jacobian matrix is singular, while the presumption of Newton's method is that Jacobian matrix is nonsigular. The Newton step will be infinite. Thus, Newton's method fails.

```
function cp05_12

syms t f;
a=0;
b=1000;
g=9.8065;
k=0.00341;


f(t)=log10(cosh(t*sqrt(g*k)))/k-1000;

while ((b-a)>1e-6)
    m=a+(b-a)/2;
    if f(a)/abs(f(a))==f(m)/abs(f(m))
        a=m;
    else
        b=m;
    end
end

disp('5.12');
disp('Biseciton method is used. ');
fprintf('It takes about %7.5f s.\n', b);
```

```
>> cp05_12
5.12
Biseciton method is used.
It takes about 46.72787 s.
>>
```

```
function cp05_18


syms x1 x2 f1 f2 F;
X=[x1; x2];
%f function
f1=(x1+3)*(x2^3-7)+18;
f2=sin(x2*exp(x1)-1);
F(x1,x2)=[f1; f2];
%Jacobian function
J=jacobian(F(x1,x2),X);

%(a)
X=[-0.5;1.4];
S=[1;1];
while (norm(S)>1e-6)
    evalJ=subs(J, [x1 x2], [X(1) X(2)]);%update Jacobian
    evalF=eval(F(X(1), X(2)));%compute f(x)
    S=linsolve(evalJ,-evalF);
    X=X+S;
end

X=round(X*10^4)/10^4;
disp('(a)');
disp('So based on Newton iteration, the solution is ');
disp(X);

%(b)
X=[-0.5;1.4];
S=[1;1];
B=subs(J, [x1 x2], [X(1) X(2)]);
while(norm(S)>1e-6)
    evalF=eval(F(X(1), X(2)));%compute f(x)
    S=linsolve(B,-evalF);%compute Newton-llike step
    X=X+S;
    Y=eval(F(X(1), X(2)))-evalF;
    B=B+((Y-B*S)*S')/(S'*S);
end

X=round(X*10^4)/10^4;
disp('(b)');
disp('So based on Broyden iteration, the solution is ');
disp(X);

%(c)
disp('(c)');
X=[-0.5;1.4];
S=[1;1];
countA=0;
ErrA=[0 0 0];
while (norm(X-[0; 1],Inf)>1e-16)
```

```
    evalJ=subs(J, [x1 x2], [X(1) X(2)]);%update Jacobian
    evalF=eval(F(X(1), X(2)));%compute f(x)
    S=linsolve(evalJ,-evalF);
    X=X+S;
    countA=countA+1;
    ErrA(countA,:)=[round(countA) X'-[0;1]'];
end
disp('Newton method');
disp(' k  x1-0                 x2-1');
fprintf('%2d %18.16f %18.16f \n', ErrA');

[n,m]=size(ErrA);
TimesA=zeros(n-1,1);
for i=2:n
    TimesA(i-1,1)=ErrA(i,2)/ErrA(i-1,2);
end
disp('    ek/e(k-1)');
disp(TimesA);


X=[-0.5;1.4];
S=[1;1];
countB=0;
ErrB=[0 0 0];
B=subs(J, [x1 x2], [X(1) X(2)]);
disp('Broyden method');
while(norm(X-[0; 1],Inf)>1e-16)
    evalF=eval(F(X(1), X(2)));%compute f(x)
    S=linsolve(B,-evalF);%compute Newton-like step
    X=X+S;
    Y=eval(F(X(1), X(2)))-evalF;
    if(norm(S-[0;0],Inf)==0)
        fprintf('Newton-like step beomces zero at k=%d.\n',round(countB));
        break
    end
    B=B+((Y-B*S)*S')/(S'*S);%when Step closes to zero, this value is NaN
    countB=countB+1;
    ErrB(countB,:)=[round(countB) X'-[0;1]'];
end
disp(' k  x1-0                 x2-1');
fprintf('%2d %18.16f %18.16f \n', ErrB');

[n,m]=size(ErrB);
TimesB=zeros(n-1,1);
for i=2:n
    TimesB(i-1,1)=ErrB(i,2)/ErrB(i-1,2);
end
disp('    ek/e(k-1)');
disp(TimesB);

disp('Compare the convergence rate of Newton method and Broyden method: newton method is↙
```

```
quadratic and Broyden method is much slower.');
```

```
>> cp05_18
(a)
So based on Newton iteration, the solution is
     0
     1

(b)
So based on Broyden iteration, the solution is
     0
     1

(c)
Newton method
 k  x1-0               x2-1
 1 -0.0553151357177094 0.0280665838357432
 2 -0.0001403508964316 0.0001574043269821
 3 -0.0000000177907769 0.0000000055513851
 4 0.0000000000000000 0.0000000000000000
    ek/e(k-1)
    0.0025
    0.0001
   -0.0000

Broyden method
Newton-like step beomces zero at k=13.
 k  x1-0               x2-1
 1 -0.0553151357177094 0.0280665838357432
 2 0.0005099530701464 0.0001236434918626
 3 -0.0002338478636409 0.0000765608693296
 4 -0.0000408262210375 0.0000135978907723
 5 -0.0000001327508960 0.0000000453383613
 6 -0.0000000005391206 0.0000000001806633
 7 0.0000000000016679 -0.0000000000005573
 8 -0.0000000000000008 0.0000000000000002
 9 0.0000000000000001 0.0000000000000002
10 0.0000000000000001 0.0000000000000002
11 0.0000000000000000 0.0000000000000002
12 0.0000000000000000 0.0000000000000002
13 0.0000000000000000 0.0000000000000002
    ek/e(k-1)
   -0.0092
   -0.4586
    0.1746
    0.0033
    0.0041
   -0.0031
   -0.0005
   -0.1019
    0.7349
    0.0000
    0.0000
```

```
        0

Compare the convergence rate of Newton method and Broyden method: newton method is↙
quadratic and Broyden method is much slower.
>>
```