```
>> %%% Edited transcript of matlab analysis done in class on 10/25/12

>> clear all

>> %%%  This is a comparison of quasi-Newton methods for the function
%%%       f(x) = x1^4/4 + x2^2/2 - x1*x2 + x1 - x2
%%%  of Example 11.2 pp. 195-196.

>> %%%  We are given x0 and H0:

>> x0 = .59607*[1 1]'

x0 =

    0.5961
    0.5961


>> H0 = [.94913 .14318
     .14318 .59702]

H0 =

    0.9491    0.1432
    0.1432    0.5970

>> %%%  First we find x1 for the Newton method.

%%%  We calculate the gradient and Hessian at x0:

>> g0 = [x0(1)^3-x0(2)+1; x0(2)-x0(1)-1]

g0 =

    0.6157
   -1.0000

>> F0 = [3*x0(1)^2-1 -1; -1 1]

F0 =

    0.0659   -1.0000
   -1.0000    1.0000

>> x1N = x0 - eye(2)/F0 *g0

x1N =

    0.1847
    1.1847

>> %%%  For the rank one correction formula, the DFP algorithm,
%%%  and the BFGS algorithm, the initial direction vector is
```

```
>> d0 = -H0*g0

d0 =

   -0.4412
    0.5089

>> %%%  We find alpha = arg min f(x0 + alpha g0)

>> syms x01 x02 d01 d02 alpha

>> %%%  We take the derivative with respect to alpha

>> collect(diff((x01+alpha*d01)^4/4+(x02+alpha*d02)^2/2-(x01+alpha*d01)*(x02+alpha*d02)+(x01+alpha*d01)-
(x02+alpha*d02),alpha),alpha)

ans =

d01^4*alpha^3 + (3*d01^3*x01)*alpha^2 + (3*d01^2*x01^2 - 2*d01*d02 + d02^2)*alpha + d01*x01^3 - d02*x01 +
d01 - d02 - d01*x02 + d02*x02

>> %%%  We take the second derivative so we can check that we found a minimum.

>> collect(diff(diff((x01+alpha*d01)^4/4+(x02+alpha*d02)^2/2-
(x01+alpha*d01)*(x02+alpha*d02)+(x01+alpha*d01)-(x02+alpha*d02),alpha),alpha),alpha)

ans =

(3*d01^4)*alpha^2 + (6*d01^3*x01)*alpha + 3*d01^2*x01^2 - 2*d01*d02 + d02^2

>> d01 = d0(1); d02 = d0(2);

>> x01 = x0(1); x02 = x0(2);

>> coefs = [d01^4 3*d01^3*x01 (3*d01^2*x01^2-2*d01*d02-d01+d02^2-d02) d01*x01^3-d02-x02+d02*x02]

coefs =

    0.0379   -0.1536    0.8478   -0.8951

>> roots(coefs)

 =

1.4005 + 4.1115i
1.4005 - 4.1115i
1.2519

>> %%% Only one root is real, so that is the minimizing alpha (if one exists).

>> alpha0 = 1.2519;

>> (3*d01^4)*alpha0^2 + (6*d01^3*x01)*alpha0 + 3*d01^2*x01^2 - 2*d01*d02 + d02^2
```

```
ans =

    0.7091

>>  %%% ... so the second derivative being positive, our alpha is minimizing.

>> x1r1 =[x01+alpha0*d01;x02+alpha0*d02]

x1r1 =

    0.0437
    1.2331

>> g1 = [x1r1(1)^3-x1r1(2)+1; x1r1(2)-x1r1(1)-1]

g1 =

   -0.2330
    0.1894

>> %%% We calculate the increments in x and the gradient:

>> del0x = alpha0*d0

del0x =

   -0.5524
    0.6370

>> del0g = g1-g0

del0g =

   -0.8487
    1.1894

>> %%% Rank One Correction Formula H1

>> H1 = H0 + (del0x-H0*del0g)*(del0x-H0*del0g)'/(del0g'*(del0x-H0*del0g))

H1 =

    0.4087   -0.1728
   -0.1728    0.4123

>> d1 = -H1*g1

d1 =

    0.1280
   -0.1184

>> %%% DFP Algorithm H1
```

```
>> H1DFP = H0 + del0x*del0x'/(del0x'*del0g) - (H0*del0g)*(H0*del0g)'/(del0g'*H0*del0g)

H1DFP =

    0.8722    0.1580
    0.1580    0.6484

>> [evec,eval] = jordan(H1)

evec =

    1.0106   -0.9895
    1.0000    1.0000


eval =

    0.2377         0
         0    0.5833

>> [evec,eval] = jordan(H1DFP)


evec =

   -0.5171    1.9339
    1.0000    1.0000


eval =

    0.5666         0
         0    0.9539

>> H1BFGS = H0 + (1 + del0g'*H0*del0g/(del0g'*del0x))*del0x*del0x'/(del0x'*del0g) -
(H0*del0g*del0x'+(H0*del0g*del0x')')/(del0g'*del0x)

H1BFGS =

    0.8770    0.1614
    0.1614    0.6508

>> [evec,eval]=jordan(H1BFGS)

evec =

   -0.5204    1.9216
    1.0000    1.0000


eval =

    0.5668         0
```

```
        0    0.9610

>> %%% Comparison of eigenvalues

>> evalr1 = jordan(H1)

evalr1 =

    0.2377         0
         0    0.5833

>> evalDFP = jordan(H1DFP)

evalDFP =

    0.5666         0
         0    0.9539

>> evalBFGS = jordan(H1BFGS)

evalBFGS =

    0.5668         0
         0    0.9610

>> %%% We see that H1BFGS is more positive definite than H1DFP which
%%% is more positive definite than H1.
```