(1)Minimize $f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$ with initial value $= (0,3)$.

1. Newton method:

   Step 1: choose $x_0 = (0,3)$

   Step 2: $x_n = x_{n-1} - H^{-1}(x_{n-1})\nabla f(x_{n-1})$

   where ∇f *is gradient*, H *is Hessian matrix of* f.

   Step 3: repeat step1 until stopping rule satisfied, $||x_n - x_{n-1}|| < 10^{-5}$.

```
The  1  step  f(x) is  3.163488     ,and x is ( 0.6663594 0.3287011 )
The  2  step  f(x) is  0.6248001    ,and x is ( 1.110931 0.5554527 )
The  3  step  f(x) is  0.1234075    ,and x is ( 1.407299 0.7036451 )
The  4  step  f(x) is  0.02437439   ,and x is ( 1.604876 0.8024378 )
The  5  step  f(x) is  0.004814528   ,and x is ( 1.736586 0.8682926 )
The  6  step  f(x) is  0.0009510564  ,and x is ( 1.824389 0.912194 )
The  7  step  f(x) is  0.0001879092  ,and x is ( 1.882919 0.9414589 )
The  8  step  f(x) is  3.715161e-05  ,and x is ( 1.921928 0.9609636 )
The  9  step  f(x) is  7.361574e-06  ,and x is ( 1.947911 0.9739552 )
The  10  step  f(x) is  1.469595e-06  ,and x is ( 1.965182 0.9825907)
The  11  step  f(x) is  3.007365e-07  ,and x is ( 1.976582 0.9882906)
The  12  step  f(x) is  6.665171e-08  ,and x is ( 1.983932 0.9919657)
The  13  step  f(x) is  1.861239e-08  ,and x is ( 1.98832 0.9941595 )
The  14  step  f(x) is  8.563238e-09  ,and x is ( 1.990381 0.9951898)
The  15  step  f(x) is  6.904031e-09  ,and x is ( 1.990885 0.995442 )
The  16  step  f(x) is  6.816969e-09  ,and x is ( 1.990914 0.9954564)
The  17  step  f(x) is  6.816702e-09  ,and x is ( 1.990914 0.9954564)
```

So, The minimum of $f(x_1, x_2) = 6.816702 \times 10^{-9}$, at $(1.990914\,, 0.9954564\,)$.

And, I find that $f(x_n) \leq f(x_{n-1}), \forall\, n$ in each iteration.

2. Steepest Descent method:

   Step 1: choose $x_0 = (0,3)$

   Step 2: $x_n = x_{n-1} - \hat{\alpha}\,\nabla f(x_{n-1})$, $\quad \hat{\alpha} = \text{argmin}_{\alpha \geq 0}\, g(\alpha)$

   $$where\ g(\alpha) = f\big(x_{n-1} - \alpha\,\nabla f(x_{n-1})\big).$$

   Step 3: repeat step2 until stopping rule satisfied,

   $$||x_n - x_{n-1}|| < 10^{-5}\ or\ the\ iteration\ times < 40.$$

   Note that:

   In Step2, I use Newton method to estimate $\hat{\alpha} = \text{argmin}_{\alpha \geq 0}\, g(\alpha)$.

   Choose initial value $\alpha_0 = 0$, *and* $\alpha_n = \alpha_{n-1} - \frac{g'(\alpha_{n-1})}{g''(\alpha_{n-1})}$,

   until $|\alpha_n - \alpha_{n-1}| < 10^{-5}$.

```
The 1 step f(x) is 0.3653851 ,and x is ( 2.707512 1.523175 ),alpha is 0.06153437
The 2 step f(x) is 0.09664148 ,and x is ( 2.537016 1.210467 ),alpha is 0.2307209
The 3 step f(x) is 0.04498044 ,and x is ( 2.441973 1.262286 ),alpha is 0.1116007
The 4 step f(x) is 0.02615951 ,and x is ( 2.393854 1.174031 ),alpha is 0.2671172
The 5 step f(x) is 0.0170911 ,and x is ( 2.351988 1.196856 ),alpha is 0.1246142
The 6 step f(x) is 0.01205521 ,and x is ( 2.326571 1.150235 ),alpha is 0.2793399
The 7 step f(x) is 0.008955163 ,and x is ( 2.301534 1.163884 ),alpha is 0.1307298
The 8 step f(x) is 0.00691754 ,and x is ( 2.285194 1.133907 ),alpha is 0.2856728
The 9 step f(x) is 0.005503061 ,and x is ( 2.26806 1.143246 ),alpha is 0.1343392
The 10 step f(x) is 0.004483106 ,and x is ( 2.256426 1.121899 ),alpha is 0.2895291
The 11 step f(x) is 0.003722197 ,and x is ( 2.24375 1.128806 ),alpha is 0.1367373
The 12 step f(x) is 0.003140246 ,and x is ( 2.234928 1.112611 ),alpha is 0.2920878
The 13 step f(x) is 0.002684686 ,and x is ( 2.225059 1.117986 ),alpha is 0.1384572
The 14 step f(x) is 0.002321756 ,and x is ( 2.218073 1.105158 ),alpha is 0.2938689
The 15 step f(x) is 0.002027679 ,and x is ( 2.210107 1.109495 ),alpha is 0.1397609
The 16 step f(x) is 0.001786267 ,and x is ( 2.2044 1.099009 ),alpha is 0.2951376
The 17 step f(x) is 0.001585512 ,and x is ( 2.197794 1.102603 ),alpha is 0.1407923
The 18 step f(x) is 0.001416881 ,and x is ( 2.19302 1.093825 ),alpha is 0.2960429
The 19 step f(x) is 0.001273783 ,and x is ( 2.187424 1.096867 ),alpha is 0.1416378
The 20 step f(x) is 0.00115138 ,and x is ( 2.183355 1.089379 ),alpha is 0.2966747
The 21 step f(x) is 0.001045813 ,and x is ( 2.178537 1.091997 ),alpha is 0.1423522
The 22 step f(x) is 0.0009541725 ,and x is ( 2.175016 1.085512 ),alpha is 0.2970903
The 23 step f(x) is 0.0008740782 ,and x is ( 2.170809 1.087795 ),alpha is 0.1429721
The 24 step f(x) is 0.0008036991 ,and x is ( 2.167725 1.082109 ),alpha is 0.2973275
The 25 step f(x) is 0.0007415001 ,and x is ( 2.164009 1.084122 ),alpha is 0.1435231
The 26 step f(x) is 0.0006862833 ,and x is ( 2.16128 1.079084 ),alpha is 0.2974123
The 27 step f(x) is 0.0006370227 ,and x is ( 2.157967 1.080877 ),alpha is 0.1440235
The 28 step f(x) is 0.0005929086 ,and x is ( 2.155531 1.076372 ),alpha is 0.2973631
The 29 step f(x) is 0.0005532338 ,and x is ( 2.152551 1.077983 ),alpha is 0.144487
The 30 step f(x) is 0.0005174355 ,and x is ( 2.15036 1.073924 ),alpha is 0.2971929
The 31 step f(x) is 0.0004850131 ,and x is ( 2.147661 1.07538 ),alpha is 0.1449238
The 32 step f(x) is 0.0004555659 ,and x is ( 2.145678 1.0717 ),alpha is 0.2969115
The 33 step f(x) is 0.0004287315 ,and x is ( 2.143218 1.073024 ),alpha is 0.1453421
The 34 step f(x) is 0.0004042185 ,and x is ( 2.141413 1.069667 ),alpha is 0.2965262
The 35 step f(x) is 0.0003817589 ,and x is ( 2.139158 1.070879 ),alpha is 0.1457483
The 36 step f(x) is 0.0003611371 ,and x is ( 2.137506 1.067801 ),alpha is 0.2960424
The 37 step f(x) is 0.0003421514 ,and x is ( 2.135429 1.068914 ),alpha is 0.1461475
The 38 step f(x) is 0.0003246395 ,and x is ( 2.133911 1.066079 ),alpha is 0.2954645
```

```
The 39 step f(x) is 0.000308447 ,and x is ( 2.13199 1.067107 ),alpha is 0.146544
The 40 step f(x) is 0.0002934503 ,and x is ( 2.130589 1.064485 ),alpha is 0.294796
```

So, The minimum of $f(x_1, x_2) = 2.934503 \times 10^{-4}$, at $(2.130589\,,\ 1.064485\,)$.
In this case, I find that the first step of f(x) is 0.3653851 (c.f. Newton's is 3.163488),
The steepest descent work faster than newton method.
But after the first step, steepest descent performs poorly. In fact, the steepest descent stops due to the iteration times reaching 40, not $||x_n - x_{n-1}|| < 10^{-5}$.

(2) Minimize $f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$ with initial value $= (0,3)$.

1. Newton method:

   by using the same stopping rule , $||x_n - x_{n-1}|| < 10^{-5}$.

```
The 1 step f(x) is 0.0002503457 ,and x is ( 1.985624 0.9895074 )
The 2 step f(x) is 4.944115e-12 ,and x is ( 1.999998 0.9999985 )
The 3 step f(x) is 5e-12 ,and x is ( 1.999998 0.9999985 )
```

So, The minimum of $f(x_1, x_2) = 5 \times 10^{-12}$, at $(1.999998\,,\ 0.9999985\,)$.

2. Steepest Descent method:

   by using the same stopping rule , $||x_n - x_{n-1}|| < 10^{-5}$.

```
The 1 step f(x) is 0.2352944 ,and x is ( 1.529403 0.7058942 ),alpha is 0.09558773
The 2 step f(x) is 0.001369673 ,and x is ( 1.988313 1.011714 ),alpha is 0.6500493
The 3 step f(x) is 7.976838e-06 ,and x is ( 1.99726 0.9982876 ),alpha is 0.09558536
The 4 step f(x) is 4.450311e-08 ,and x is ( 1.999934 1.000067 ),alpha is 0.6507236
The 5 step f(x) is 2.598115e-10 ,and x is ( 1.999984 0.9999902 ),alpha is 0.0953407
The 6 step f(x) is 7.730237e-12 ,and x is ( 2 0.9999987 ),alpha is 0.7356083
The 7 step f(x) is 5.686266e-13 ,and x is ( 1.999999 0.9999995 ),alpha is 0.1138628
```

So, The minimum of $f(x_1, x_2) = 5.686266 \times 10^{-13}$, at $(1.999999\,,\ 0.9999995\,)$.

3. Conjugate Gradient method:

$$f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2 = \frac{1}{2}X^TQX + C^TX$$

$$\text{where } Q = \begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix} \ \ and\ C = \begin{pmatrix} -2 \\ 0 \end{pmatrix}.$$

Step 1: choose $x_1 = (0\,,3)$, $d_1 = -\nabla f(x_1)$.

Step 2: $x_{k+1} = x_k + \hat{\alpha}_k d_k$ with $\hat{\alpha}_k = \frac{-\nabla f(x_k)^T d_k}{d_k^T Q d_k}$

$$d_{k+1} = -\nabla f(x_{k+1}) + \lambda_k d_k \quad \text{where } \lambda_k = \frac{-\nabla f(x_{k+1})^T Q d_k}{d_k^T Q d_k}$$

Step 3: repeat step2 one time, since $Q_{2\times 2}$ $and\ d_1, d_2\ are\ Q - conjugate$,

by theorem, $\{x_k\}$ $converges\ to\ solution\ x^*\ after\ 2\ step$.

```
The 1 step f(x) is 0.2352944 ,and x is ( 1.529412 0.705882 )
The 2 step f(x) is 0.005440224 ,and x is ( 1.971437 1.01972 )
```

So, the minimum of $f(x_1, x_2) = 5.440224 \times 10^{-3}$, $at\ (1.971437\ ,\ 1.01972\ )$.

In this case, d_1, d_2 are

```
d1 ( 15.999998 -24.0000040 )
d2 ( 0.692039   0.4913474 )
```

However, there is some strange, $d_1, d_2\ are\ Q - conjugate\ \ implies\ d_1^T Q d_2 = 0$ but $d_1^T Q d_2 = -15.05871 \neq 0$.

(1)

```
#(1)    f(x1,x2)=(x1-2)^4+(x1-2*x2)^2
rm(list = ls())
f<-function(x){ (x[1]-2)^4+(x[1]-2*x[2])^2 }
#ff<-function(x1,x2){ (x1-2)^4+(x1-2*x2)^2 }
#library(rgl);plot3d(ff)
#gradient & Hessian
h<-10^-6
gradient.f<-function(x){
   y<-c( ( f(c(x[1]+h,x[2]))-f(x) ) / h ,    ( f(c(x[1],x[2]+h))-f(x) ) / h )
   return(y)
}
Hessian.f<-function(x){
   y<-matrix(c( ( f(c(x[1]+2*h,x[2])) - 2*f(c(x[1]+h,x[2])) + f(x) ) / h^2,
                          ( f(c(x[1]+h,x[2]+h)) - f(c(x[1]+h,x[2])) - f(c(x[1],x[2]+h)) + f(x) ) /
h^2,
                          ( f(c(x[1]+h,x[2]+h)) - f(c(x[1]+h,x[2])) - f(c(x[1],x[2]+h)) + f(x) ) /
h^2,
                          ( f(c(x[1],x[2]+2*h)) - 2*f(c(x[1],x[2]+h)) + f(x) ) / h^2) ,2,2 )
   return(y)
}

#Newton method
#initial value
```

```
x0<-c(0,3)
epslon<-10^-5
loop<-0
#step
x2<-x0 ;x1<-c(1,1)
while( sqrt(sum((x2-x1)^2)) >= epslon){
   loop<-loop+1
   x1<-x2
   x2<-x1 - solve(Hessian.f(x1)) %*% gradient.f(x1)
   cat("The ",loop," step ","f(x) is ",f(x2)," ,and x is (",x2,") \n")
};cat("The minimum of f(x) is ",f(x2),", at (",x2,")")

#Steepest Descent method
#initial value
x0<-c(0,3)
epslon<-10^-5
loop1<-0
#step
x2<-x0
x1<-c(1,1)
A<-c()
g<-function(alpha){ f(x1-alpha*gradient.f(x1)) }
while(sqrt(sum((x2-x1)^2)) >= epslon &&loop1<40){
   loop1<-loop1+1
   x1<-x2

   #alpha.hat = argmin g(alpha)
   alpha0<-0
   loop2<-0
   h<-10^-6

   alpha2<-alpha0 ;alpha1<-1
   while( abs(alpha2-alpha1) >= epslon ){
      loop2<-loop2+1
      alpha1<-alpha2
      firdif<-(g(alpha1+h)-g(alpha1))/h
      secdif<-(g(alpha1+2*h)-2*g(alpha1+h)+g(alpha1) ) / h^2
      alpha2<-alpha1 - firdif/secdif
```

```
        #cat("The ",loop2," step ","g(alpha) is ",g(alpha2),"\n")
    };#cat("The minimum of g(alpha) is ",g(alpha2),", the alpha.hat is ",alpha2,"\n")
    alpha.hat<-alpha2
    #step
    x2<-x1 - alpha.hat*grad(f,x1)

    cat("The",loop1,"step f(x) is",f(x2),",and x is (",x2,"),alpha is",alpha.hat,"\n")
};cat("The minimum of f(x) is ",f(x2),", at (",x2,")")
```

(2)

```
#(2)    f(x1,x2)=(x1-2)^2+(x1-2*x2)^2
rm(list = ls())
ff<-function(x){ (x[1]-2)^2+(x[1]-2*x[2])^2 }

#gradient & Hessian
h<-10^-6
gradient.ff<-function(x){
    y<-c( ( ff(c(x[1]+h,x[2]))-ff(x) ) / h ,    ( ff(c(x[1],x[2]+h))-ff(x) ) / h )
    return(y)
}

Hessian.ff<-function(x){
    y<-matrix(c( ( ff(c(x[1]+2*h,x[2])) - 2*ff(c(x[1]+h,x[2])) + ff(x) ) / h^2,
                          ( ff(c(x[1]+h,x[2]+h)) - ff(c(x[1]+h,x[2])) - ff(c(x[1],x[2]+h)) + ff(x) )
/ h^2,
                          ( ff(c(x[1]+h,x[2]+h)) - ff(c(x[1]+h,x[2])) - ff(c(x[1],x[2]+h)) + ff(x) )
/ h^2,
                          ( ff(c(x[1],x[2]+2*h)) - 2*ff(c(x[1],x[2]+h)) + ff(x) ) / h^2) ,2,2 )
    return(y)
}
#Newton method
#initial value
x0<-c(0,3)
epslon<-10^-5
loop<-0
#step
x2<-x0 ;x1<-c(1,1)
while( sqrt(sum((x2-x1)^2)) >= epslon){
    loop<-loop+1
```

```
    x1<-x2
    x2<-x1 - solve(Hessian.ff(x1)) %*% gradient.ff(x1)
    cat("The ",loop," step ","f(x) is ",ff(x2)," ,and x is (",x2,") \n")
};cat("The minimum of f(x) is ",ff(x2),", at (",x2,")")

#Steepest Descent method
#initial value
x0<-c(0,3)
epslon<-10^-5
loop1<-0
#step
x2<-x0
x1<-c(1,1)
g<-function(alpha){ ff(x1-alpha*gradient.ff(x1)) }
while(sqrt(sum((x2-x1)^2)) >= epslon &&loop1<40){
    loop1<-loop1+1
    x1<-x2

    #alpha.hat = argmin g(alpha)
    alpha0<-0
    loop2<-0
    h<-10^-6

    alpha2<-alpha0 ;alpha1<-1
    while( abs(alpha2-alpha1) >= epslon ){
        loop2<-loop2+1
        alpha1<-alpha2
        firdiff<-(g(alpha1+h)-g(alpha1))/h
        secdiff<-(g(alpha1+2*h)-2*g(alpha1+h)+g(alpha1) ) / h^2
        alpha2<-alpha1 - firdiff/secdiff
        #cat("The ",loop2," step ","g(alpha) is ",g(alpha2),"\n")
    };#cat("The minimum of g(alpha) is ",g(alpha2),", the alpha.hat is ",alpha2,"\n")
    alpha.hat<-alpha2
    #step
    x2<-x1 - alpha.hat*gradient.ff(x1)

    cat("The",loop1,"step f(x) is",ff(x2),",and x is (",x2,"),alpha is",alpha.hat,"\n")
};cat("The minimum of f(x) is ",ff(x2),", at (",x2,")")
```

```
#Conjugate gradient method
Q<-matrix(c(4,-4,-4,8),2,2)
y1<-c(0,3)
d1<--gradient.ff(y1) ;d3<-d1
D<-c()

loop<-0
y3<-y1
while( loop<dim(Q)[1] ){
   loop<-loop+1
   d2<-d3
   y2<-y3              #"2" is now ,"3" is next

   alpha.hat<-- gradient.ff(y2) %*% d2 / d2 %*% Q %*% d2
   y3<-y2 + alpha.hat*d2
   D<-rbind(D,d2)
   #if(loop<dim(Q)[1]){
      lambda2<- - gradient.ff(y3) %*% Q %*% d2 / d2 %*% Q %*% d2
      d3<- - gradient.ff(y3) + lambda2*d2
   #}
   cat("The",loop,"step f(x) is",ff(y3),",and x is (",y3,") \n")
}
cat("The minimum of f(x) is ",ff(y3),", at (",y3,")")
```