Lecture 6

First and second order unconstrained optimization

The Steepest Descent Method

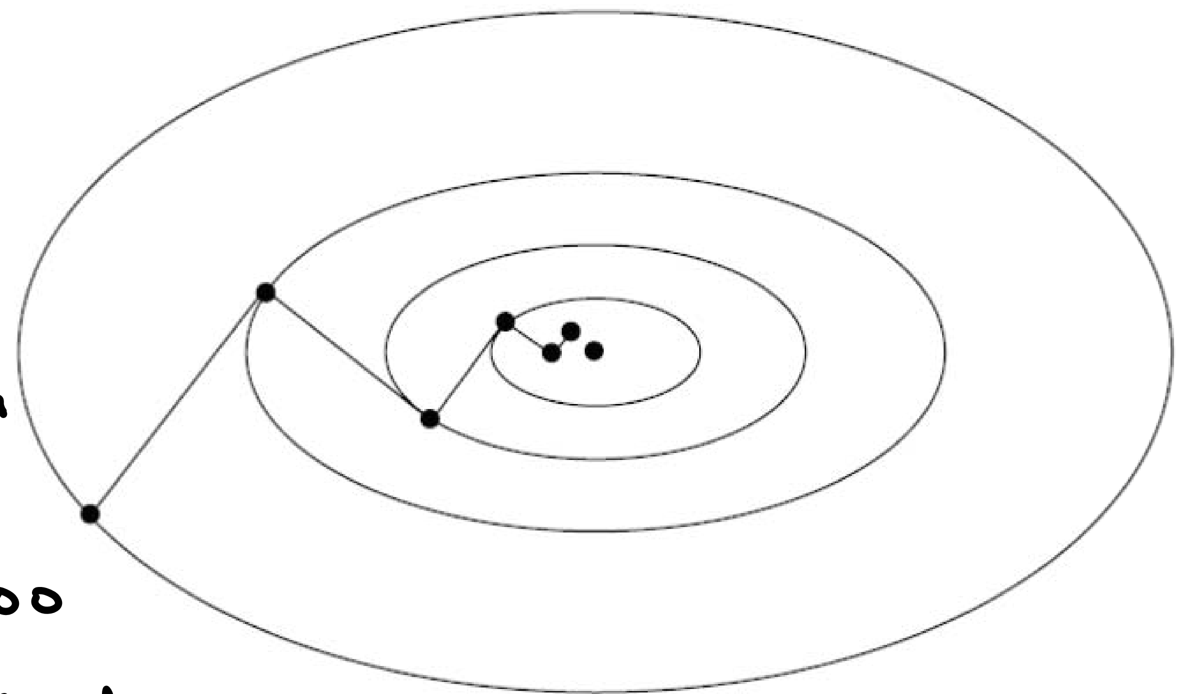
* The direction of ∇f at any point in the parameter space gives the direction of maximum function ascent $\Rightarrow$ $-\nabla f$ is direction of steepest function descent.

* At the kth time step $\underline{x}_k$, the new point $\underline{x}_{k+1}$ is given by $\underline{x}_{k+1} = \underline{x}_k + \lambda^* (-\nabla f)$.
λ^* is obtained through line search
$\lambda^* = \arg\min_{\lambda} f(\underline{x}_k - \lambda \nabla f)$

Illustration

* Fast reduction in objective function far from the solution

* Convergence is too slow near solution!



Example

Use the method of steepest descent to find the minimizer of

$$f(x_1, x_2, x_3) = (x_1-4)^4 + (x_2-3)^2 + 4(x_3+5)^4.$$

Start from the point $\underline{x}^{(0)} = [4, 2, -1]^T$ and carry out only 2 iterations.

Solution: $\nabla f(\underline{x}) = [4(x_1-4) \quad 2(x_2-3) \quad 16(x_3+5)^3]^T$

$$\nabla f(\underline{x}^{(0)}) = [0 \quad -2 \quad 1024]^T$$

Example (Cont'd)

$$\lambda_0^* = \arg\min_{\lambda > 0} f(\underline{x}^{(0)} - \lambda \underline{\nabla} f(\underline{x}^{0}))$$

$$f(\lambda) = f(\underline{x}^{(0)} - \lambda \underline{\nabla} f(\underline{x}^{(0)}))$$

$$f(\lambda) = (0)^4 + (2 + 2\lambda - 3)^2 + 4(-1 - 1024\lambda + 5)^4$$

Using the secant method, we obtain

$$\lambda_0^* = 3.967 \times 10^{-3} \Longrightarrow \underline{x}^{(1)} = [4.000 \quad 2.008 \quad -5.062]^T$$

$$\nabla f(\underline{x}^{(1)}) = [0 \quad -1.984 \quad -0.003875]^T$$

$$\lambda_1^* = \arg\min_{\lambda > 0} f(\underline{x}^{(1)} - \lambda \nabla f(\underline{x}^{(1)}))$$

Example (Cont'd)

$$f(\lambda) = (2.088 + 1.984\lambda - 3)^2 + 4(-5.062 + 0.003875\lambda + 5)^4$$

Using a secant method, we obtain $\lambda_1^* = 0.500$

$$\underline{x}^{(2)} = \underline{x}^{(1)} - \lambda_1^* \nabla f(\underline{x}^{(1)}) = [4.000 \quad 3.000 \quad -5.060]^T$$

$$\nabla f(\underline{x}^{(2)}) = [0.000 \quad 0.000 \quad -0.003525]^T$$

MATLAB CODE

```
%This program carries out the steepest descent direction
NumberOfParameters=3; %This is n for this problem
OldPoint=[3  -3  5]' %This is the starting point
OldValue=getObjective(OldPoint) %Get the objective function at the old point
Tolerance=0.001; %terminating tolerance for line search
Epsilon=0.001; %exploration step
LambdaMax=4.0; %maximum value of lambda for line search
StepNorm=1000; %initialize stepNorm
MinimumDistance=1.0e-4; %This is the terminating distance
while(StepNorm>MinimumDistance) %repeat until maximum number of iteration is achieved
    Gradient=getGradient('getObjecive',OldPoint, Epsilon); %get the gradient at the old point
    NormalizedNegativeGradient=-1.0*Gradient/norm(Gradient); %normalize the gradient
    LambdaOptimal = GoldenSection('getObjective',Tolerance,OldPoint,
                                  NormalizedNegativeGradient,LambdaMax); %get the optimal value
    NewPoint=OldPoint+LambdaOptimal*NormalizedNegativeGradient; %Get new point
    NewValue=feval('getObjective',NewPoint); %Get the New Value
    StepNorm=norm(NewPoint-OldPoint); %get the norm of the step
    OldPoint=NewPoint %update the current point
    OldValue=NewValue %update the current value
end
```

Code Output

minimize the objective function

$$f(x_1, x_2, x_3) = (x_1 - 1)^2 + (x_2 - 5)^2 + (x_3 - 4)^2$$

```
OldPoint =[3   -3    5]       OldValue =69

OldPoint =[   2.0368   0.8517   4.5183]     OldValue = 18.5518

OldPoint = [1.0735    4.7034   4.0365]      OldValue =  0.0947

OldPoint =[ 0.9994    5.0000   3.9994]       OldValue = 6.9495e-007
```

why only 3 iterations?

What Are Conjugate Directions?

* A general quadratic function of n variables is given by $f(\underset{\sim}{x}) = \frac{1}{2}\underset{\sim}{x}^T \underset{\sim}{H} \underset{\sim}{x} + \underset{\sim}{b}^T \underset{\sim}{x} + c$

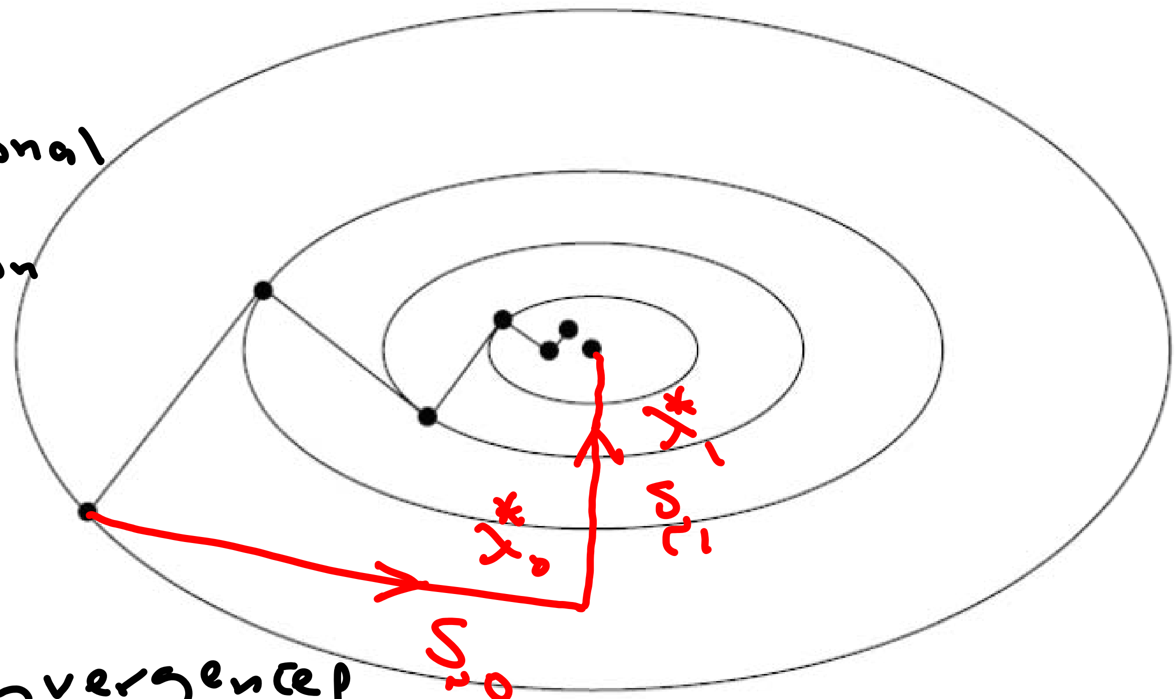
$\underset{\sim}{H} \in R^{n\times n}$, $\underset{\sim}{b} \in R^n$, $c \in R$.

* Two directions $\underset{\sim}{s}_i$ and $\underset{\sim}{s}_j$ are $\underset{\sim}{H}$-conjugate if they satisfy $\underset{\sim}{s}_i^T \underset{\sim}{H} \underset{\sim}{s}_j = 0$, $i \neq j$
* Orthogonal directions are conjugate relative to the identity matrix.

Optimization Using Conjugate Directions

The minimizer of an n-dimensional quadratic function is obtained in at most n iterations.

$\Rightarrow$ quadratic convergence

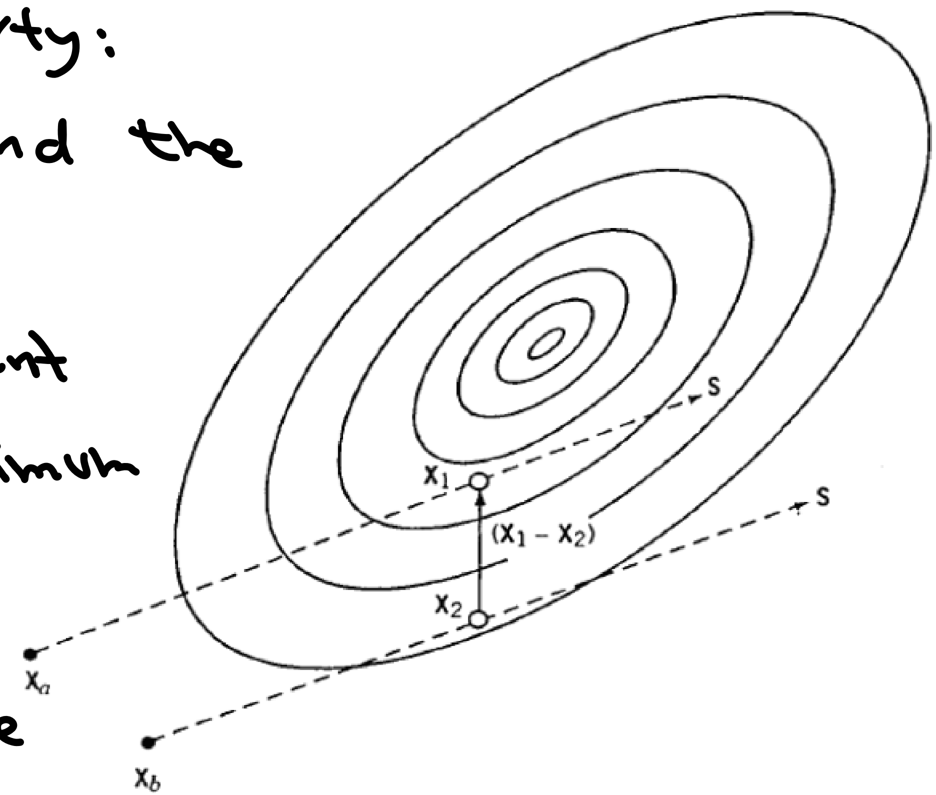


Derivative Free Conjugate Directions

parallel subspace property:

* Starting from $\underline{x}_a$, find the minimum $\underset{\sim}{x}_1$ along $\underset{\sim}{s}$
* Starting from a different point $\underline{x}_b$, find the minimum $\underset{\sim}{x}_2$ along $\underset{\sim}{s}$

$\Rightarrow$ $(\underset{\sim}{x}_1 - \underline{x}_2)$ is Conjugate to $\underset{\sim}{s}$!



__Proof__: Assume $f(\underset{\sim}{x}) = \frac{1}{2}\underset{\sim}{x}^T \underset{\sim}{H}\,\underset{\sim}{x} + \underset{\sim}{b}^T \underset{\sim}{x} + c$

$\Longrightarrow \underset{\sim}{\nabla} f(\underset{\sim}{x}) = \underset{\sim}{H}\,\underset{\sim}{x} + \underset{\sim}{b}$

$\underset{\sim}{\nabla} f(\underset{\sim}{x}_1) = \underset{\sim}{H}\,\underset{\sim}{x}_1 + \underset{\sim}{b}, \quad \underset{\sim}{\nabla} f(\underset{\sim}{x}_2) = \underset{\sim}{H}\,\underset{\sim}{x}_2 + \underset{\sim}{b}$

$\underset{\sim}{\nabla} f(\underset{\sim}{x}_1) - \underset{\sim}{\nabla} f(\underset{\sim}{x}_2) = \underset{\sim}{H}(\underset{\sim}{x}_1 - \underset{\sim}{x}_2) \longleftarrow 1$

Because $\underset{\sim}{x}_1$ and $\underset{\sim}{x}_2$ are minimizers in the direction of $\underset{\sim}{s}$, we have $\underset{\sim}{s}^T \underset{\sim}{\nabla} f(\underset{\sim}{x}_1) = \underset{\sim}{s}^T \underset{\sim}{\nabla} f(\underset{\sim}{x}_2) = 0$

$\underset{\sim}{s}^T \underset{\sim}{H}\,\underset{\sim}{x}_1 + \underset{\sim}{s}^T \underset{\sim}{b} = 0, \quad \underset{\sim}{s}^T \underset{\sim}{H}\,\underset{\sim}{x}_2 + \underset{\sim}{s}^T \underset{\sim}{b} = 0$

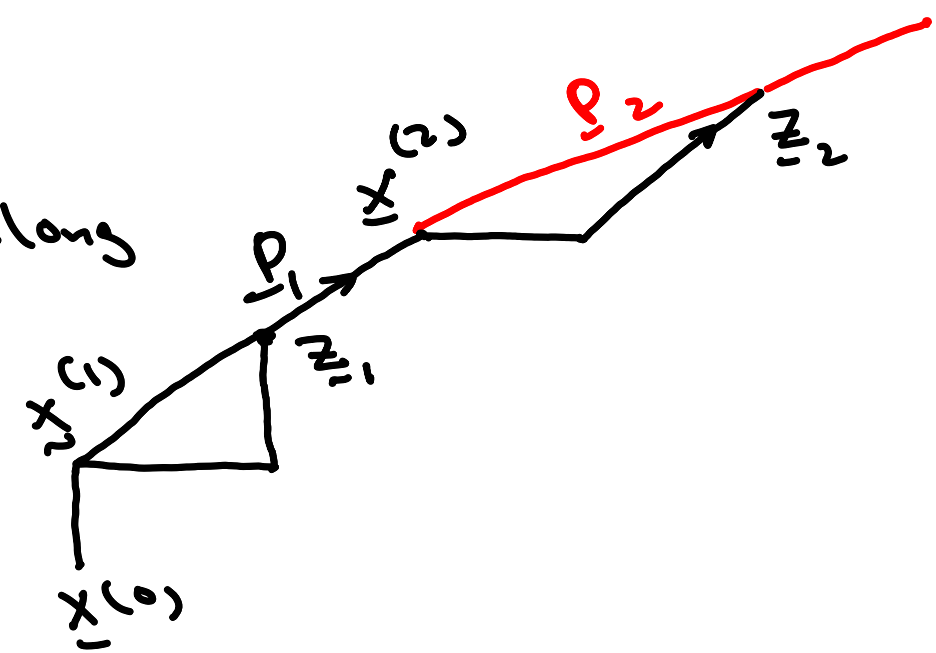
$\Longrightarrow \underset{\sim}{s}^T \underset{\sim}{H}(\underset{\sim}{x}_1 - \underset{\sim}{x}_2) = 0 \Longrightarrow (\underset{\sim}{x}_1 - \underset{\sim}{x}_2)$ is conjugate to $\underset{\sim}{s}$!

A Method For Conjugate Directions

For $n = 3$

* Intialize all 3 directions to $\underset{\sim}{e}_i$, $i = 1, 2, 3$
* Starting from $\underline{x}_0$ find minimum along $\underset{\sim}{e}_3$ to get $\underset{\sim}{x}_1$
* From $\underset{\sim}{x}_1$, sequentially minimize along $\underset{\sim}{e}_1$, $\underline{e}_2$, $\underline{e}_3$ to get $\underset{\sim}{z}$
* $\underset{\sim}{p}_1 = (\underset{\sim}{z} - \underset{\sim}{x}_1)$ is conjugate to $\underline{e}_3$ (why?)
* Search from $\underset{\sim}{z}$ along $\underset{\sim}{p}_1$ to get $\underline{x}_2$.
* repeat next step using $\underline{e}_2$, $\underline{e}_3$, $\underset{\sim}{p}_1$

Illustration



* from $\underline{x}_0$ minimize along $\underline{e}_2$ to get $\underline{x}_1$
* from $\underline{x}_1$ minimize along both $\underline{e}_1$ and $\underline{e}_2$ to get $\underline{z}_1$
* $\underline{p}_1 = \underline{z}_1 - \underline{x}_1$ is conjugate to $\underline{e}_2$. Search along $\underline{p}_1$ to get $\underline{x}_2$. Repeat using $\underline{e}_1$ and $\underline{p}_1$

The Algorithm

Initialization: Set $\underset{\sim}{x}_0$. Set $\underset{\sim}{p}_i = \underset{\sim}{e}_i$, $i = 1, 2, \ldots, n$.

Compute $\underset{\sim}{x}_1$ as the minimizer of f along the line $\underset{\sim}{x}_0 + \lambda \underset{\sim}{p}_n$

Repeat

Set $\underset{\sim}{z}_1 = \underset{\sim}{x}_k$

Get $\underset{\sim}{z}_{n+1} = \underset{\sim}{z}_1 + \lambda_1^* \underset{\sim}{p}_1 + \lambda_2^* \underset{\sim}{p}_2 + \cdots + \lambda_n^* \underset{\sim}{p}_n$

Set $\underset{\sim}{p}_j = \underset{\sim}{p}_{j+1}$ for $j = 1, 2, \ldots, n-1$. Set $\underset{\sim}{p}_n = \underset{\sim}{z}_{n+1} - \underset{\sim}{z}_1$.

Get $\underset{\sim}{x}_{k+1} = \underset{\sim}{z}_{n+1} + \lambda^* \underset{\sim}{p}_n$.

$k = k+1$

end

MATLAB CODE

```
up=Directions(:,NumberOfParameters); %get the vector ei
un=-1.0*up; %get the vector -ei
fp=feval('getObjective',OldPoint+Epsilon*up); %get +ve value
if(fp<OldValue)  u=up; else  u=un; end
LambdaOptimal = GoldenSection
('getObjective',Tolerance,OldPoint,u,LambdaMax);
NewPoint=OldPoint+LambdaOptimal*u; %get new point
NewValue=getObjective(NewPoint);
OldPoint=NewPoint;
OldValue=NewValue;
while(StepNorm>1.0e-4) %repeat
    YOld=OldPoint; %start exploring
    YOldValue=OldValue; %store also the old value
    for i=1:NumberOfParameters %repeat for all coordinates
      up=Directions(:,i); %get the vector ei
      un=-1.0*up; %get the vector -ei
      fp=feval('getObjective',YOld+Epsilon*up); %get value
      if(fp<OldValue)  u=up;  else  u=un; end
      LambdaOptimal = GoldenSection
('getObjective',Tolerance,YOld,u,LambdaMax);%get optimal
      YNew=YOld+LambdaOptimal*u; %Get new  exploration point
      YNewValue=feval('getObjective',YNew); %get new value
      YOld=YNew;
      YOldValue=YNewValue;
    end
```

```
ConjugateDirection=YOld-OldPoint; %determine new
direction
    LambdaOptimal = GoldenSection
('getObjective',Tolerance,YOld,ConjugateDirection,Lambda
Max);
    NewPoint=YOld+LambdaOptimal*ConjugateDirection; %Get
new point
    NewValue=feval('getObjective',NewPoint); %Get the
New Value
    StepNorm=norm(NewPoint-OldPoint); %get the norm of
the step
    for j=1:(NumberOfParameters-1)
      Directions(:,j)=Directions(:,(j+1));
    end
    Directions(:,NumberOfParameters)=ConjugateDirection;
%store the new conjugate direction
    OldPoint=NewPoint %update the current point
    OldValue=NewValue %update the current value
    pause
end
OldPoint
OldValue
```

Example:

Utilize the Conjugate Directions method to minimize the function

$$f(\underline{x}) = 4(x_1 - 1)^2 + 3(x_2 - 5)^2 + (x_3 - 4)^2$$

starting from $\underline{x}^{(0)} = [0 \quad 0 \quad 0]^T$.

OldPoint =[0   0   0]'   OldValue =95

OldPoint = [1.0001 5.0003   4.0001]'   OldValue = 2.1657e-007

Solution is obtained in one iteration!

Example 2

Find the minimum of the function

$$f(\underline{x}) = 1.5x_1^2 + 2x_2^2 + 1.5x_3^2 + x_1x_3 + 2x_2x_3 - 3x_1 - x_3$$

starting from the point $\underline{x}^{(0)} = [0 \quad 0 \quad 0]^T$.

OldPoint =[0 0 0] OldValue = 0

OldPoint =[0.8889 0.0000 0.3333] OldValue =-1.3518

OldPoint =[0.8889 -0.1667 0.3333] OldValue = -1.4074

OldPoint = [0.9583 -0.1667 0.1250] OldValue =-1.4653

OldPoint =[1.0000 0.0000 -0.0000] OldValue = -1.5000

Analytical Solution

$$\nabla f(\underline{x}) = \begin{bmatrix} 3x_1 + x_3 - 3 \\ 4x_2 + 2x_3 \\ x_1 + 2x_2 + 2x_3 - 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Solving these 3 eqns, we get

$$\underline{x}^* = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad f(\underline{x}^*) = -1.5 \checkmark$$

Conjugate Gradient Methods

* These methods exploit the gradient at each point to construct the conjugate direction at the current step.

* The new iteration is obtained by carrying out a line search along the current conjugate direction.

Fletcher-Reeves Method

* Start with an initial point $\underset{\sim}{x}_1$. The first conjugate direction is $\underset{\sim}{S}_1 = -\underset{\sim}{\nabla} f(\underset{\sim}{x}_1)$
* At the kth iteration, the new solution is obtained through a line search $\underset{\sim}{x}_{k+1} = \underset{\sim}{x}_k + \lambda_k^* \underset{\sim}{S}_k$.
* The new conjugate direction $\underset{\sim}{S}_{k+1}$ is given by

$$\underset{\sim}{S}_{k+1} = -\underset{\sim}{\nabla} f(\underset{\sim}{x}_{k+1}) + \frac{\|\underset{\sim}{\nabla} f(\underset{\sim}{x}_{k+1})\|^2}{\|\underset{\sim}{\nabla} f(\underset{\sim}{x}_k)\|^2} \underset{\sim}{S}_k$$

* Set $k = k+1$ and repeat!

MATLAB CODE

```
NumberOfParameters=3; %This is n for this problem
OldPoint=[3  -7   0]' %This is the starting point
OldValue=getObjective(OldPoint) %Get the objective function at the old point
Epsilon=1.0e-3; %this is the perturbation
LambdaMax=5; %maximum lambda for line search
Tolerance=1.0e-4; %termination condition for line search
Gradient=getGradient('getObjective', OldPoint, Epsilon); %get the gradient at the first point
NormGradientOld=norm(Gradient); %get the norm of the gradient at the current point
ConjugateDirection=-1.0*Gradient; %initialize the first direction
IterationCounter=0;
while(IterationCounter<NumberOfParameters) %do only n iterations
   LambdaOptimal = GoldenSection('getObjective',Tolerance,OldPoint,ConjugateDirection,LambdaMax);%do line search
   NewPoint=OldPoint+LambdaOptimal*ConjugateDirection; %get new point
   NewValue=feval('getObjective',NewPoint); %get new objective function value
   NewGradient=getGradient('getObjective', NewPoint, Epsilon); %get new gradient
   NormGradientNew=norm(NewGradient); %get norm of the new gradient
   %now we determine the new conjugate direction
   NewConjugateDirection=-1.0*NewGradient+((NormGradientNew*NormGradientNew)/(NormGradientOld*NormGradientOld))
*ConjugateDirection;
   %now we make the new point the current point
   OldPoint=NewPoint
   OldValue=NewValue
   Gradient=NewGradient;
   NormGradientOld=NormGradientNew;
   ConjugateDirection=NewConjugateDirection;
   IterationCounter=IterationCounter+1;
end
```

Code Output

Utilize the Fletcher-Reeves method to minimize the function

$$f(\underline{x}) = 4(x_1-1)^2 + 3(x_2-5)^2 + (x_3-4)^2$$

starting from $\underline{x}^{(0)} = [3 \quad -7 \quad 0]^T$.

OldPoint =[3 -7 0]' OldValue = 464

OldPoint =[0.3529 4.9085 1.3231] OldValue = 8.8661

OldPoint = [1.3885 5.1720 2.4461] OldValue = 3.1071

OldPoint =[0.9996 4.9990 3.9992] OldValue = 4.6080e-006

Newton's Method

* This method exploits the 2nd order information of the function under consideration

$$\nabla f(\underset{\sim}{x}) = \nabla f(\underline{x}_k) + \underset{\sim}{H}(\underset{\sim}{x} - \underline{x}_k)$$

by setting $\nabla f(\underset{\sim}{x}) = \underset{\sim}{0}$, we obtain

$$\underset{\sim}{H}\,\Delta \underset{\sim}{x} = -\nabla f(\underline{x}_k) \Longrightarrow \underset{\sim}{x}_{k+1} = \underset{\sim}{x}_k - \underline{H}^{-1}\nabla f(\underline{x}_k)$$

* The method may diverge if the starting point is far!

Newton's Method (Cont'd)

* For a quadratic function, this method converges in only ONE iteration.
* This method may converge to a local maximum, a local minimum, or a saddle point
* Notice that we move along the direction $\underset{\sim}{S} = - \underset{\sim}{H}^{-1} \underline{\nabla} f(\underline{x}_k)$ a distance of $\lambda = 1$.
* For non quadratic function, we do a line search

<u>Example</u> Use Newton's method to minimize the function

$$f(\underset{\sim}{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

starting from the point $\underset{\sim}{x}^{(0)} = [3 \quad -1.0 \quad 0 \quad 1]^T$

Solution: $$\nabla f(\underset{\sim}{x}) = \begin{bmatrix} 2(x_1 + 10x_2) + 40(x_1 - x_4)^3 \\ 20(x_1 + 10x_2) + 4(x_2 - 2x_3)^3 \\ 10(x_3 - x_4) - 8(x_2 - 2x_3) \\ -10(x_3 - x_4) - 40(x_1 - x_4)^3 \end{bmatrix}$$

Example (Cont'd)

$$\underset{\sim}{H} = \begin{bmatrix} 2+120(x_1-x_4)^3 & 20 & 0 & -120(x_1-x_4)^2 \\ 20 & 200+12(x_2-2x_3)^2 & -24(x_2-2x_3)^2 & 0 \\ 0 & -24(x_2-2x_3)^2 & 10+48(x_2-2x_3)^2 & -10 \\ -120(x_1-x_4)^2 & 0 & -10 & 10+120(x_1-x_4)^2 \end{bmatrix}$$

First iteration $\qquad f(\underline{x}^{(0)}) = 215$

$$\underline{\nabla} f(\underline{x}^{(0)}) = \begin{bmatrix} 306 \\ -144 \\ -2 \\ -310 \end{bmatrix}$$

Example (Cont'd)

$$\underline{H}(\underline{x}^{(0)}) = \begin{bmatrix} 482 & 20 & 0 & -480 \\ 20 & 212 & -24 & 0 \\ 0 & -24 & 58 & -10 \\ -480 & 0 & -10 & 490 \end{bmatrix}$$

$$\left(\underline{H}(\underline{x}^{(0)})\right)^{-1} = \begin{bmatrix} 0.1126 & -.0089 & 0.0154 & 0.1106 \\ -0.0089 & 0.0057 & 0.0008 & -0.0087 \\ 0.0154 & 0.0008 & 0.0203 & 0.0155 \\ 0.1106 & -0.0087 & 0.0155 & 0.1107 \end{bmatrix}$$

Example (Cont'd)

$$\Delta \underline{x}^{(0)} = -\underline{H}^{(0)^{-1}} \underline{\nabla} f(\underline{x}^{(0)}) = \begin{bmatrix} 1.4127 \\ -0.8413 \\ -0.2540 \\ 0.7460 \end{bmatrix}$$

$$\underline{x}^{(1)} = \underline{x}^{(0)} + \Delta \underline{x}^{(0)} = \begin{bmatrix} 3 \\ -1.0 \\ 0 \\ 1.0 \end{bmatrix} - \begin{bmatrix} 1.4127 \\ -0.8413 \\ -0.2540 \\ 0.7460 \end{bmatrix} = \begin{bmatrix} 1.5873 \\ -0.1587 \\ 0.2540 \\ 0.2540 \end{bmatrix}$$

$$f(\underline{x}^{(1)}) = 31.8$$

Example (Cont'd)

second iteration

$$\nabla f(\underline{x}^{(1)}) = \begin{bmatrix} 94.81 \\ -1.179 \\ 2.371 \\ -94.81 \end{bmatrix}, \quad \underline{H}^{(1)} = \begin{bmatrix} 215.3 & 20 & 0 & -213.3 \\ 20 & 205.3 & -10.67 & 0 \\ 0 & -10.67 & 31.34 & -10 \\ -213.3 & 0 & -10 & 223.3 \end{bmatrix}$$

$$\underline{H}^{(1)^{-1}} \nabla f(\underline{x}^{(1)}) = \begin{bmatrix} 0.5291 \\ -0.0529 \\ 0.0846 \\ 0.0846 \end{bmatrix}$$

$$\underline{x}^{(2)} = \underline{x}^{(1)} + \Delta \underline{x}^{(1)} = \begin{bmatrix} 1.5873 \\ -0.1587 \\ 0.2540 \\ 0.2540 \end{bmatrix} - \begin{bmatrix} 0.5291 \\ -0.0529 \\ 0.0846 \\ 0.0846 \end{bmatrix} = \begin{bmatrix} 1.0582 \\ -0.1058 \\ 0.1694 \\ 0.1694 \end{bmatrix}$$

$$f(\underline{x}^{(2)}) = 6.28$$

MAIN ISSUES WITH Newton's Method

* The direction $\underline{s}_k = -H^{-1}\underline{\nabla} f$ may not be a descent direction if the Hessian is not positive definite.
* The Hessian matrix of a non quadratic function is +ve definite near local minima.
* Starting from a far point, the method may diverge.

The Levenberg-Marquardt Method

* This Method addresses the drawbacks of Newton Method.
* The kth iteration step is given by

$$\Delta \underline{x}_k = -\left(\underset{\sim}{H}^{(k)} + \mu \underline{I}\right)^{-1} \underline{\nabla} f(\underline{x}^{k})$$

μ is selected such that $\underline{H}^{(k)} + \mu \underline{I}$ is +ve definite.

* What are the eigenvalues of $\underline{H}^{(k)} + \mu \underline{I}$?

The Levenberg-Marquardt Method (Cont'd)

* Notice that if $\mu = 0$, $\Delta \underline{x} = -\underline{H}^{-1} \nabla f \Rightarrow$ regular Newton's step
* If μ is large enough $\mu \underline{I} \gg \underline{H} \Rightarrow$

  $\Delta \underline{x} \simeq -\frac{1}{\mu} \underline{\nabla} f \Rightarrow$ Steepest descent direction
* Usually we start each iteration with a small μ and increase μ until a descent step is achieved.

MATLAB CODE

```
NumberOfParameters=3; %This is n for this problem
Epsilon=1.0e-3; %perturbation used in finite differences
OldPoint=[3  -7   0]' %This is the starting point
OldValue=getObjective(OldPoint)
OldGradient=getGradient('getObjective', OldPoint, Epsilon)
OldGradientNorm=norm(OldGradient); %get the old gradient norm
OldHessian=getHessian('getObjective', OldPoint, Epsilon)
Identity=eye(NumberOfParameters); %This is the identity matrix
while (OldGradientNorm>1.0e-5)
  TrustRegionParameter=0.001; %initialize trust region parameter
  DescentFlag=0;
  while(DescentFlag==0)
      MarquardtMatrix=OldHessian+TrustRegionParameter*Identity;
      NewStep=-1.0*inv(MarquardtMatrix)*OldGradient
      NewPoint=OldPoint+NewStep %get the new trial point
      NewValue=feval('getObjective',NewPoint); %calculate new value
      if(NewValue<OldValue) %a descent step
          DescentFlag=1.0;
      else
          TrustRegionParameter=TrustRegionParameter*4;
      end
  end
Step=NewPoint-OldPoint; %get the new step
  StepNorm=norm(Step); %get the step norm
  NewGradient=getGradient('getObjective',
NewPoint, Epsilon); %get new gradient
  NewHessian=getHessian('getObjective', NewPoint,
Epsilon); %get new Hessian
  %now we swap paremeters
  OldPoint=NewPoint;
  OldGradient=NewGradient;
  OldHessian=NewHessian;
  OldGradientNorm=norm(OldGradient);
end
```

Example

Find the minimum of the function

$$f(\underline{x}) = 1.5x_1^2 + 2x_2^2 + 1.5x_3^2 + x_1x_3 + 2x_2x_3 - 3x_1 - x_3$$

starting from $\underline{x}^{(0)} = [3 \quad -7 \quad 0]^T$.

OldPoint =[3 -7 0]' OldValue = 102.5000

OldGradient =[6.0015 -27.9980 -11.9985]'

OldHessian = [3.0000 0 1.0000
0 4.0000 2.0000
1.0000 2.0000 3.0000]

NewStep =[-2.0004 6.9969 0.0017]'

NewPoint =[0.9996 -0.0031 0.0017]'

NewValue = -1.5000

exact solution

$\underline{x}^* = [1.0 \quad 0 \quad 0]^T$