LEVENBERG-MARGUARDT TRUST REGION METHOD

ADDIS ABABA UNIVERSITY
COLLEGE OF COMPUTATIONAL AND NATURAL SCIENCE
DEPATMENT OF MATHEMATICS

In partial fulfilment of the requirements of Degree master Science in Mathematics

By: ARNEST FANGARASIO SHABAN

Stream : Optimization

Advisor: Berhanu G (Ph.D)

June 2016

June 29, 2016

Department Of Mathematics
College of Natural Sciences
Addis Ababa University

This is to certify that, the project prepared by:
ARNEST FANGARSIO SHABAN; entitled: "**LEVENBERG-MARGUARDT TRUST REGION METHOD** "submitted in partial fulfilment for the requirements of Master Degree in (OPTIMIZATION) complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee

Examiner	Signature	Date
Examiner	Signature	Date
Advisor	Signature	Date

Chair of Department or Graduate Program Coordinator

Abstract

This project addresses the solution of unconstrained optimization problems using algorithms that require only values with out using derivative (derivative free) ,the algorithms generate a sequence with an initial point x_0 and direction d_k and step length λ and look for best point (next iteration x_{k+1} for k=1,2..... in this paper we evaluate four methods in (derivative free), cyclic coordinate method , Hooke and Jeeves Mehtod and Rosenbrock Method and
. The Levenberg-Marquardt method is a standard technique used to solve non-linear least squares problems. Gradient descent method, the sum of the squared errors is reduced by updating the parameters . In the Gauss-Newton method, the sum of the squared errors is reduced by assuming the least squares function is locally quadratic, and it is acts more like a gradient-descent method when the parameters are far from their optimal value,
and trust region method is technics to find the optimal point within each trust region , the approach constricts the initial quadratic surrogate model using few of order O(n) ,where n is the number of design variables , the proposed approach adopts weighted least squares fitting for updating the surrogate model instead of interpolation which is commonly use In DF optimization , this make the approach more suitable for stochastic optimization and for functions subject to numerical error . The weights are assigned to give more emphasis to points close to the current centre point.

Key words: derivative free-optimization , levenberg-Marguardt , trust region method , Quadratic surrogate model

Dedication

I dedicated this project to my wife and parents without they are support and nurturing this work would have never been possible , I also dedicated this project to Addis Ababa University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author

THE AUTHOR RESERVES OTHER PUBLICATION RIGHTS, AND NEITHER THE THESIS NOR EXTENSIVE EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT THE AUTHOR'S WRITTEN PERMISSION. THE AUTHOR ATTESTS THAT PERMISSION HAS BEEN OBTAINED FOR THE USE OF ANY COPYRIGHTED MATERIAL APPEARING IN THIS THESIS (OTHER THAN BRIEF EXCERPTS REQUIRING ONLY PROPER ACKNOWLEDGEMENT IN SCHOLARLY WRITING) AND THAT ALL SUCH USE IS CLEARLY ACKNOWLEDGED

Acknowledgment

Above all, I would like to thank the Almighty God, for letting me accomplish this stage. A very grateful thanks to those who assist my graduate study in different aspects.My strongest thanks is addressed to Addis Ababa University , especially to Mathematics department staff for their assistance to me.

In this respect, I would like also to thank my home land S.Sudan for the opportunity and my Families for their encouragement, to cope with challenges.

Finally,I am deeply indebted to my project advisor,Dr Berhanu.Guta for his many suggestions and constant support and friendly approach during this research. His tireless follow up and his consistent support will be in my memory forever.

Contents

List of Figures

Chapter 1

BASIC CONCEPTS AND DEFINITION

1.1 PRELIMINARY

optimization also known as mathematical programming, collection of mathematical principles and method used for solving quantitative problems in many disciplines including physics ,biology , engineering , economics and business.
Optimization problems typical have three fundamental elements,the first is a single numerical quantity , or objective function , that to be maximized or minimized , the second element is collection of variable which are quantities whose value can be manipulated in order to optimize the objective. the third element of an optimization problem is a set of constraint which are restrictions on the value that the variable can take optimization can problems can be categorized into two: constrained and unconstrained problem
all optimization algorithm require the user to supply a starting point usually denote byx_0
Beginning atx_0 optimization algorithm generate a sequence
$x_1, x_2.....x_k, x_{k+1}.....$ process of iteration terminate when either.
no more progress can be made or.
it seems that solution has been approximated with sufficient occur
for moving from the current iterate x_k to new iterate x_{k+1} most of the continuous algorithm follow one of these approaches. line search method or Trust Region Method.

1.2 CONVEXITY AND OPTIMIZATION

The concepts of convexity is fundamental in optimization; it implies that the problem is benign in several respect the convex can be applied both to set and to function.
let S be a convex set in a real vector space and let $F : S \to R$ be a function

- Definition: A function f is called convex if
  for all $x, y \in S$, and $\lambda \in [0, 1]$
  $f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$

- Definition: A function f is called strictly convex if
  for all $x \neq y \in S$, and $\lambda \in [0, 1]$

$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda)f(y)$

- Definition: A function f is called concave if
  for all $x, y \in S$, and $\lambda \in [0,1]$
  $f((1-\lambda)x + \lambda y) \geq (1-\lambda)f(x) + \lambda f(y)$

- Definition: A function f is called strictly concave if
  for all $x \neq y \in S$, and $\lambda \in [0,1]$
  $f((1-\lambda)x + \lambda y) > (1-\lambda)f(x) + \lambda f(y)$

1.3 EIGENVALUE AND EIGENVECTOR

Given an $n \times n$ square matrix A of real or complex numbers .An eigenvalue λ and its associated generalized eigenvector are Pair obeying the relation
$(A - \lambda I)^k v = 0$
where v is a nonzero $n \times 1$ column vector
I is the $n \times n$ identity matrix
n is a positive integer and both λ and v are allowed to be complex even when A is real.
when n=1 the vector is called simply an eigenvector and the pair is called an eigne pairs in this case Av=λv
To solve the eigenvalue problem for an $n \times n$ matrix A , follow these steps:
• compute the determinant of $A - \lambda I$ with λ subtracted along the diagonal this determinant start with λ^n or $-\lambda^n$ it is a polynomial in λ of degree n
• find the roots of this polynomial by solving $det(A - \lambda I) = 0$ then n roots are the n eigenvalue of A they make $A - \lambda I$ singular.
• for each eigenvalue λ , solve $det(A - \lambda I) = 0$ to find an eigenvector x.
If all the eigenvalues of a real square matrix are nonnegative real numbers then it is called positive semi definite matrix and if all the eigenvalues are negative real number then A is called negative definite matrix.
If such a matrix has both positive and negative eigenvalues we shall say that is indefinite .morover we have also the following equivalent definitions G is :

- positive definite if $S^T GS > 0$ for all$S \in R^n$,$S \neq 0$
- positive semi definite if $S^T GS \geq 0$ for all$S \in R^n$
- negative definite if $S^T GS < 0$ for all$S \in R^n$, $S \neq 0$
- negative semi definite if $S^T GS \leq 0$ for all$S \in R^n$
- indefinite if there exists $S_1, S_2 \in R^n$ for which $S_1^T GS_1 > 0$ and $S_2^T GS_2 < o$

Quadratic function of n variables
$f(x_1, x_2, ...x_n)$

$f(x_{n\times1}) = x^T px$

when $x_{n\times1} = [x_1, x_2, ...x_n]^T$

suppose $p_{n\times n}$ is not symmetry matrix

f(x) = $x^T P_{n\times n} x_{n\times1}$

f(x) = $\frac{1}{2}x^T P_{n\times n} x + \frac{1}{2}x^T P x_{n\times1}$

f(x) = $\frac{1}{2}x^T (P)x + \frac{1}{2}[x^T Px]^T$

f(x) = $\frac{1}{2}x^T Px + \frac{1}{2}x^T P^T x$

$=\frac{1}{2}(x^T(\frac{p+p^T}{2})x)$
this is symmetric matrix
Test for positive definite matrix (sylvester's criterion)
if p is positive symmetric matrix
$x^T px \geq 0$
$x^T(\frac{p+p^T}{2})x$
$x^T px \geq 0$
let $p_{n\times n}$ is symmetric matrix

- all the diagonal elements matric
  (p_{ii} , for i = 1,2,3,.....n)
  must be positive and non zero elements

- all the leading principal minus (determinants) must be positive . Note: the leading principal minus of order k of an $(n \times n)$ matrix is obtained by deleting last $(n - k)$ rows and columns

Example:
consider the quadratic function in two variables
$f(x_1, x_2) = 4x_1^2 - 4x_1x_2 - x_2^2$
show that the quadratic function
$f(x_1, x_2)$ is positive semi definite

$$f(x_1, x_2) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$f(x_1, x_2) = x^T px \geq 0$ $x \neq 0$
provided $p \geq 0$

$$p = \begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix} \geq 0$$

symmetric criterion
principal minus of order -(1)
$4 > 0$, $1 > 0$
principal minus of order -(2)
$det[p] = 4 - 4 = 0$
the matrix $p_{2\times 2}$ is positive semi definite matrix
and hence the function $f(x_1, x_2)$ is a positive semi definite function

1.4 OPTIMALITY CONDITION

Definition:
consider the problem of minimizing f(x) over $\Re^n$ and let $x^* \in \Re^n$.
The point x^* is global minimizer if

$f(x^*) \leq f(x)$ for all $x \in R^n$,

a point x^* is called a local minimum if there exists $\varepsilon - nbhd$ ε
$N_\varepsilon(x^*)$ around x^* such that

$f(x^*) \leq f(x)$
for each $x \in N_\varepsilon(x^*)$,

a point x^* is a strict local minimizer (also called a strong local minimizer) if there is a neighborhood N of x^* such that

$f(x^*) < f(x)$ for all $x \in N_\varepsilon$

with $x \neq x^*$ for some $\varepsilon > 0$.

a point x^* is an isolated local minimizer if there is a $\varepsilon - nbhd$ N of x^* such that x^* is the only local minimizer in N

1.4.1 NECESSARY OPTIMALITY CONDITION

(first order necessary condition)
If x^* is a local minimizer and f is continuously differentiable in an open $\varepsilon - nbhd$of x^* then $\nabla f(x^*) = 0$
Proof
suppose not,i.e spile $\nabla f(x^*) \neq 0$ then $d = -\nabla f(x^*)$

$\nabla f(x^*)^T d = -||\nabla f(x^*)||^2 < 0$
x = $x^* - \alpha \nabla f(x^*)$

$\alpha > 0$

$f(x) = f(x^* - \alpha \nabla f(x^*))$

$= f(x^*) + \nabla f(x^*)^T (x - x^*) + 0(||x - x^*||)$

$= f(x^*) + \nabla f(x^*)^T(x^* - \alpha\nabla f(x^*) - x^*) + 0(||x - x^*||)$

$= f(x^*) - \alpha||\nabla f(x^*)||^2 + 0()$

if $\nabla(x^*) = 0$

hence $f(x) < f(x^*)$

we found a direction leading away from x^* along which f decreases
so x^* is not a local minimizer and we have a contradiction
we call x^* a stationary point if $\nabla f(x^*) = 0$
any local minimizer must be a stationary point .

1.4.2 NECESSARY OPTIMALITY CONDITION

(second order necessary conditions)
if x^* is a local minimizer of f and $\nabla^2 f$ is continuous in an open neighborhood of x^* then $\nabla f(x^*) = o$ and $\nabla^2 f(x^*)$ is positive semi definite

proof
we know from first order necessary condition that $\nabla f(x^*) = 0$,
for contradiction assume that
$\nabla^2 f(x^*)$ is not positive semi definite then choose a non zero vector d.
such that

$d^T\nabla^2 f(x^*)d < 0$

and because $\nabla^2 f$ is continuous near x^* there is scalar $\delta > 0$ such that

; $d^T\nabla^2 f(x^* + td)d < 0$

for all $t \in [0, \delta]$
by doing a Taylor series expansion around x^* we have for all $\bar{t} \in (0, 1]$
and some $t \in (0, \bar{t})$ that

$f(x^* + \bar{t}d) = f(x^*) + \bar{t}d^T\nabla f(x^*) + \frac{1}{2}\bar{t}^2 d^T\nabla^2 f(x^* + td)d$

if $\bar{t} = 0$
$f(x^* + \bar{t}d) < f(x^*)$

we have found a direction from x^* along which decreases and so again x^* is not a local minimizer.

1.4.3 NECESSARY OPTIMALITY CONDITION

(second - order sufficient conditions)
Let $f : \Re^n \to \Re$ be twice continuously differentiable in a$\varepsilon - nbhd$ of x^*.
Assume that $\nabla f(x) = 0$ and $\nabla^2 f(x^*)$ is positive semi definite, then
x^* is a local minimizer of function f.
proof

let $u \in \Re^n$ and $u \neq 0$
for sufficiently small t, we have
from the definition of the derivative
let $x = x^* - t\nabla f(x^*)$
$f(x^* + tu) = f(x^*) + t\nabla f(x^*)^T u + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + 0(t^2)$
when $\nabla f(x^*) = 0$
$= f(x^*) + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + 0(t^2)$

Hence ,
if $\lambda > 0$ is the smallest eigenvalue of $\nabla^2 f(x^*)$
$f(x^* + tu) = f(x^*) + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + 0(t^2)$
$\geq f(x^*) + \frac{\lambda}{2}||tu||^2 + 0(t^2) \geq 0$
$f(x^* + tu) - f(x^*) \geq \frac{\lambda}{2}||tu||^2 + 0(t^2)$
for t is sufficiently small
$f(x^* + tu) - f(x^*) \geq 0$
Hence , x^* is a local minimizer of f

1.4.4 Iteration

Definition:- iteration is a process of repeating by which an estimate x_k of the solution to the problem is replace by a better estimate x_{k+1}

1.5 VECTOR AND FUNCTION

we use norm to measure the size of vectors, matrices and functions
Definition:
A norm on a linear space X is a function $||.|| : X \to \Re$ with the following properties

I- $||x|| \geq 0$ and $||x|| = 0$ x = 0 for all $x \in X$ (nonnegative)

II- $||\lambda x|| = |\lambda||x||$ for all $x \in X$, $\lambda \in \Re$ (homogenous);

III- $||x + y|| \leq ||x|| + ||y||$ for all $x, y \in X$ (triangle inequality)

A normed linear space $(s, ||.||)$ is a linear space with a norm $||.||$.

A normed linear space is a metric space with the metric.

d(x,y) = $||x - y||$.

when $x \in \Re$ the most common norms are the linear programming (LP) vector norms ($p \geq 1$) defined by

$$||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$$

the most common of those are the vector L_1, L_2 ,L_∞

- $||x||_1 = \sum_{i=1}^{n} |x_i|$
- $||x||_2 = \sqrt{\langle x, y \rangle}$ and
- $||x||_\infty = max|x_i|$ where $1 \leq i \leq m$

where $\langle x, y \rangle$ denote the inner product
L_2 is call Euclidean norm .
let $||.||$ be a norm over $\Re^n$ and we consider a subset x of $\Re^n$
we define an open ball of radius $x \in X$ to be he set
$\Omega_\varepsilon(x) = \{y : ||y - x|| < \varepsilon\}$
the set S is said to be open in $\Re^n$ if for every x in S there is a scalar $\varepsilon(x) > 0$ such that.
$\Omega_\varepsilon(x) \subseteq S$

1.6 TAYLOR SERIES AND QUADRATIC MODELS

The Taylor series is a representation of a function as an infinite series calculated from the value of its derivatives at the a single point.
If the function f(x) is continuously differentiable at x = c then ,

$$\sum_{n=0}^{\infty} \frac{f^n(c)}{n!}(x-c)^n = f(c) + f'(c)(x-c) + ... + \frac{f^n(c)}{n!}(x-c)^n + ...$$

the series is centered at x = c
If the series is centered at zero the series is also called a Maclaurin series

If the function f(x) is continuously differentiable at x = c then , Maclaurin series generally.

$$\sum_{n=0}^{\infty} \frac{f^n(0)}{n!}(x)^n = f(0) + f'(0)x + ... + \frac{f^n(0)}{n!}x^n + ...$$

,

1.7 QUADRATIC APPROXIMATION

Quadratic approximation is an extension of linear approximation - we are adding one more term which is related to the second derivative ,
the formula for the quadratic approximation of a function f(x) for values of x near x_0 is

$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2$
$(x \approx x_0)$

compare this to our old formula for the linear approximation of f.

$f(x) = f(x_0) + f'(x_0)(x - x_0)$

$(x = x_0)$

we get from the linear approximation the quadratic one by adding one more term that related to the second derivative

$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2$

$(x \approx x_0)$

where $f(x_0) + f'(x_0)(x - x_0)$ is the linear part and

$\frac{f''(x_0)}{2}(x - x_0)^2$ is called quadratic part

1.8 QUADRATIC MODELS

We model $f(x_k + s)$ by either of the following

- linear model

$$m_k^L(s) = f_k + s^T g_k$$

- Quadratic model - (choose a symmetric matrix B_k)

$$m_k^Q(s) = f_k + S^T g_k + \frac{1}{2} s^T B_k s$$

- CHALLENGES:
  models may not resemble
  $f(x_k + s)$ if s is large.
  models may be unbounded from below.

1.9 THE TRUST REGION METHOD

- Newton and quasi- Newton method search a solution iteratively by choosing at each step a search direction and minimize in this direction .
- an alternative approach is to find a direction and step - length
  then if the step is successful in some sense the step is accepted otherwise another direction and step length is chosen.
- the choice of the step - length and direction is algorithm dependent but a successful approach is the one based on trust region .
  consider the problem
  min f(x)
  $x \in \Re^n$
- Newton and quasi -Newton at each step(approximately) solve the minimization problem.
- min $m(x_k + s) = f(x_k) + \nabla f(x_k)s + \frac{1}{2}s^T B_k s$

  in this case B_k is symmetric and positive definite (SPD)
- if B_k is symmetric and positive definite the minimum is

  $s = -B_k^{-1} g_k$

  where $g_k = \nabla f(x_k)^T$

  and s is the quasi Newton step.
- if $B_k = \nabla^2 f(x_k)$ and is symmetric and positive definite then
  $s = -B_k^{-1} g_k$ is Newton step.
- if B_K is not positive definite the search direction $-B_k^{-1} g_k$ may fail to be a decent direction and the previous minimization problem can have no solution .
- the problem is that the method $m(x_k + s)$ is an approximation of f(x)

  $m(x_k + s) \approx f(x_k + s)$

  and this approximation is valid only in a small neighbors of x_k so that an alternative minimization problem is the following
  min$m(x_k + s) = f(x_k) + \nabla f(x_k)s + \frac{1}{2}s^T B_k s$

  subject to $||s|| \leq \delta_k$

  δ_k is the radius of trust region of the model m(x) i.e the region where we trust the model is valid .
  how to chose the radiusδ_k at each iteration

in general if there is agreement between the model $g_k(s)$ and the objective function value $f(x_k + s)$ one should select δ_k as large as possible
this can be quantified by defining the actual reduction (Aredk) in f on the k^{th} step as
$\nabla f_k = f_k - f(x_k + s_k)$ and
the corresponding predicted reduction (Predk) as

$\Delta g_k = g_{(0)} - g_k(s_k) = f_k - g_k(s_k)$

define the ratio $r_k = \frac{Actualreduction}{predictionreduction} = \frac{Areadk}{Preadk}$

which measure the agreement between the model function g_k and the objective function f.
The ratio r_k play an important role in selecting new iterate x_{k+1} and updating the trust region radius δ_k.
If r_k is close to unity it mean there is good agreement and we can expand the trust region for the next iteration .
if r_k is close to zero or negative we shrink the trust region.

1.10 Newton's Method

Suppose we want to solve:

$$\begin{array}{c} (\mathrm{p}) = min f(x) \\ x \in \Re^2 \end{array}$$

At $x = x^*, f(x)$ can be approximated by:

$f(x) \approx g(x) := f(x^*) + \nabla f(x^*)^T (x - x^*) + \frac{1}{2}(x - x^*) H(x^*)(x - x^*)$
which is the quadratic Taylor expansion of f(x)at $x = x^*$ here $\nabla f(x)$
is the gradient of f(x)and H(x) is the Hessian of f(x).
Notice that g(x) is a quadratic function, which is minimized by solving
$\nabla g(x) = 0$. Since the gradient of g(x)is:
$\nabla g(x) = \nabla f(x^*) + H(x^*)(x - x^*)$
we therefore are motivated to solve:
$\nabla f(x^*) + H(x^*)(x - x^*) = 0$
which yields
$\nabla f(x^*) = -H(x^*)(x - x^*)$
$(x - x^*) = -H(x^*)^{-1} \nabla f(x^*)$
The direction $-H(x^*)^{-1} \nabla f(x^*)$ is called the Newton direction, or the Newton step at$x = x^*$
This leads to the following algorithm for solving (P):
Newton's Method:
step 0 given x_0 , set k=0 štep 1 $d_k = -H(x^*)^{-1} \nabla f(x^*)$
if $d_k = 0$ then stop

step 2 choose step -size $\lambda_k = 1$
step 3 set $x_{k+1} = x_k + \lambda_k d_k$,
$k = k + 1$ go to step 1
Note the following:
* The method assumes $H(x^k)$ is nonsingular at each iteration.
* There is no guarantee that $f(x_{k+1}) \leq f(x_k)$
* Step 2 could be augmented by a line-search of $f(x_k + \lambda d_k)$tofind an optimal value of the step-size parameter λ.
Recall that we call a matrix SPD if it is symmetric and positive definite.

Example:
consider minimizing

$f(x_1, x_2) = x_1^2 + x_2^2 - 4x_1 - 6x_2$

if the current iteration point is $x_k = (0, 0)$,
and step direction is $d_k = (1, 2)$,
then determine $\phi(t)$, the step length t_k
and the next iteration point x_{k+1}
solution

$x_{k+1} = x_k + t_k d_k$

$x_{k+1} = (0, 0) + t(1, 2)$

$x_{k+1} = (0, 0) + (t, 2t)$

$x_{k+1} = (t, 2t)$

$\phi(t) = t^2 + 4t^2 - 4t - 12t$

$\phi(t) = 5t^2 - 16t$

$\phi'(t) = 10t - 16 = 0$

$= 10t = 16$

$= t_1 = 1.6$ the minimum point of $\phi(t)$

$x_2 = x_1 + t_1 d_1$

$x_2 = (0, 0) + 1.6(1, 2)$

$x_2 = (1.6, 3.2)$

the next iteration point

1.11 Rate of Convergence

A sequence of numbers x_k exhibits linear convergence if

$$\lim_{k\to\infty} x_k = x^*$$

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|} = \delta < 1$$

If $\delta = 0$ in the above expression, the sequence exhibits superlinear convergence.
A sequence of numbers x_k exhibits quadratic convergence if

$$\lim_{k\to\infty} x_k = x^*$$

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|} = \delta < \infty$$

1.12 Examples of Rates of Convergence

Linear convergence: $x_k = (\frac{1}{10})^K = 0.1, 0.01, 0.001, etc\ x^* = 0$

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|} = 0.1$$

Superlinear convergence:$x_k = \frac{1}{k!} : 1, \frac{1}{2}, \frac{1}{6}, \frac{1}{24}, \frac{1}{125}, etc\ ,\ x^* = 0$

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|} = \frac{k!}{(k+1)!} = \frac{1}{k+1} \to 0$$

as $k \to 0$

Quadratic convergence: $x_k = (\frac{1}{10})^{2k} = 0.1, 0.01, 0.0001, 0.000000001etc$

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|} = \frac{(10^{2k})^2}{10^{2k+1}} = 1$$

1.12.1 CONVERGENCE OF LINE SEARCH METHOD

[ALGORITHM]
(1) initialize x_0 and ε , set k=0

(2) while $||\nabla f(x_k)|| > \varepsilon$

(a) find a descent direction d_k for f at x_k,

(b) find $t_k > 0$ along such that

(i) $f(x_k + t_k d_k) < f(x_k)$

(ii) t_k satisfies Armij - Wolfe condition

(c) $x_{k+1} = x_k + t_k d_k$

(d) $k := k + 1$

end while
output : $x^* = x_k$ a stationary point of f(x)

does this algorithm converge

consider the problem

min f(x) for all x

- let $f \in c^1$ and f be bounded below.
- An optimization algorithm to minimize f(x) generates a sequence $\{x_k\}$, $k \geq 0$
- let the corresponding sequence of function value be $\{f_k\}$, $k \geq 0$
  - $f_{k+1} < f_k$, $k \geq 0$
  - stopping condition : $||\nabla f(x_k)|| > \varepsilon$

  $||\nabla f(x_k)||$ as $k \to \infty$
  suppose at every iteration k of the optimization algorithm
  - the direction d_k is chosen such that

  $\nabla f(x_k)^T d_k \leq 0$

  - define $\Phi(t) = f(x_k + t d_k)$

  $t_k > 0$ is chosen such that

  Armijo - Wolfe conditions are satisfied

  $f_{k+1} \leq f_k + c_1 t \nabla f(x_k)^T d_k$, $c_1 \in (0, 1)$

  $\Phi'(t_k) \geq c_2 \Phi'(0)$, $c_2 \in (c_1, 1)$

  $f_{k+1} < f_k$ for all $k \geq 0$

$x_{k+1} = x_k + t_k d_k$

given $f_{k+1} < f_k$ for all $k \geq 0$

$\{f_k\}$: monotonically decreasing sequence which is also bounded below.

$\{f_k\} \to f^*$ where $f^* < \infty$

$f_0 - f_k$ for all $k \geq 0$

$$\lim_{k\to\infty} f_0 - f_k < \infty$$

using Armijo's condition $t_j's$ are chosen such that

$f_{k+1} \leq f_k + c_1 t_k \nabla f(x_k)^T d_k$

$$\leq f_0 + c_1 \sum_{j=0}^{k} t_j \nabla f(x_j)^T d_j$$

therefore

$$\infty > f_0 - f_{k+1} \geq -c_1 \sum_{j=0}^{k} t_j \nabla f(x_j) d_j$$

$$-c_1 \sum_{j=0}^{\infty} t_j \nabla f(x_j) d_j < \infty$$

$$\sum_{j=0}^{\infty} -c_1 t_j \nabla f(x_j) d_j < \infty$$

therefore sum of infinity many positive terms is finite this implies beyond certain iteration k ,

$t \nabla f(x_k)^T d_k = 0$

$\Phi'(t_k) \geq c_2 \Phi'(0), c_2 \in (c_1, 1)$

$\nabla f(x_{k+1})^T d_k \geq c_2 \nabla f(x_k)^T d_k$

$(\nabla f(x_{k+1}) - \nabla f(x_k)) d_k$

$\geq (c_2 - 1) \nabla f(x_k)^T d_k$

let $nablaf(x)$ be Lipsehitz continuous that is $\exists L\ 0 < L < \infty$ such that

$||\nabla f(x_{k+1||)-\nabla f(x)} \leq L||x_{k+1||-x_k}$

but we have

$x_{k+1} = x_k + t_k d_k$

where $x_{k+1} - x_k = t_k d_k$

t_k is parameter positive

$||g_{k+1} - g_k|| \leq L t_k$

$(\nabla f(x_{k+1}) - f(x_k))^T d_k \leq L t_k d_k^T d_k$

but using Wolfe conditions

$(\nabla f(x_{k+1}) - f(x_k))^T d_k \geq (c_2 - 1)\nabla f(x_k)^T d_k$
therefore

$t_k \geq \frac{(c_2-1)\nabla f(x_k)^T d_k}{L||d_k||^2}$

$t_k \nabla f(x_k)^T d_k \leq \frac{(c_2-1)(\nabla f(x_k)^T d_k)^2}{L||d_k||^2}$

$-c_1 t_k \nabla f(x_k)^T d_k \geq c_1 \frac{(c_2-1)(g_k^T d_k)^2}{L||d_k||^2}$

let θ_k be the angle between $\nabla f(x_k)$ and d_k therefore

$-c_1 t_k \nabla g_k^T d_k \geq c_1 \frac{(c_2-1)||g_k||^2||d_k||^2 cos^2\theta_k}{L||d_k||^2}$

$-c_1 t_k \nabla g_k^T d_k \geq c_1 \frac{(c_2-1)||g_k||^2 cos^2\theta_k}{L}$

but by using Armijo's condition's

$$-c_1 \sum_{k=0}^{\infty} t_k g_k^T d_k < \infty$$

therefore

$$c_1 \frac{(1-c_2)}{L} \sum_{k=0}^{\infty} ||g_k||^2 cos^2\theta_k < \infty$$

this implies

$||\nabla f(x_k)||^2 cos^2\theta_k \to 0$

if at every iteration ,d_k is chosen such that
$\nabla f(x_k) d_k < 0$ and $cos^2\theta_k \geq \delta > 0$

then we have

$$\lim_{k\to\infty} ||\nabla f(x_k)|| = 0$$

Chapter 2

Derivative -Free Line search

Many practical applications require the optimization of functions whose derivatives are not available. Problems of this kind can be solved, in principle, by approximating the gradient (and possibly the Hessian) using finite differences, and using these approximate gradients within the algorithms described in chapter one . Even though this finite-difference approach is effective in some applications, it cannot be regarded as general-purpose technique for derivative-free optimization because the number of function evaluations required can be excessive and the approach can be unreliable.
Because various algorithms have been developed that do not attempt to approximate the gradient. Rather, they use the function values at a set of sample points to determine a new iterate by some other means. Derivative-free optimization (DFO) algorithms differ in the way they use the sampled function values to determine the new iterate.

One class of methods constructs a linear or quadratic model of the objective function and defines the next iterate by seeking to minimize this model inside a trust region.We pay particular attention to these model-based approaches because they are related to the unconstrained minimization methods described in earlier chapters.

Other widely used DFO methods include the simplex-reflection method of Nelder and Mead, pattern-search methods, conjugate-direction methods, and simulated annealing.

In this chapter we briefly discuss these methods, with the exception of simulated annealing, which is a nondeterministic approach and has little in common with the other techniques discussed in this book.

Derivative-free optimization methods are not as well developed as gradient-based methods; current algorithms are effective only for small problems. Although most DFO methods have been adapted to handle simple types of constraints, such as bounds, the efficient treatment of general constraints is still the subject of investigation. Consequently, we limit our discussion to the unconstrained optimization problem

$$minf(x)$$

$$x \in \Re^n$$

Problems in which derivatives are not available arise often in practice. The evaluation of f(x) can, for example, be the result of an experimental measurement or a stochastic simulation, with the underlying analytic form of f unknown. Even if the objective function f is known in analytic form, coding its derivatives may be time consuming or impractical. Automatic differentiation tools may not be applicable if f(x) is provided only in the form of binary computer code. Even when the source code is available, these tools cannot be applied if the code is written in a combination of languages. Methods for derivative-free optimization are often used (with mixed success) to minimize problems with non differentiable functions or to try to locate the global minimizer of a function. Since we do not treat non smooth optimization or global optimization in this book, we will restrict our attention to smooth problems in which f has a continuous derivative.

2.1 Cyclic Coordinate Method

An approach that is frequently used in practice is to cycle through the n coordinate directions $e_1, e_2, ..., e_n$, using each in turn as a search direction. At the first iteration, we fix all except the first variable, and find a new value of this variable that minimizes (or at least reduces) the objective function. On the next iteration, we repeat the process with the second variable, and so on. After n iterations, we return to the first variable and repeat the cycle. The method is referred to as the method of alternating variables or the coordinate descent method. Though simple and somewhat intuitive, it can be quite inefficient in practice At each iterate x_k, we construct the next iterate x_{k+1} by searching along the coordinate directions
$e_1, e_2, , e_n$, in this order
where e_j is the vector of zeros except for a1 in the j^{th} position.

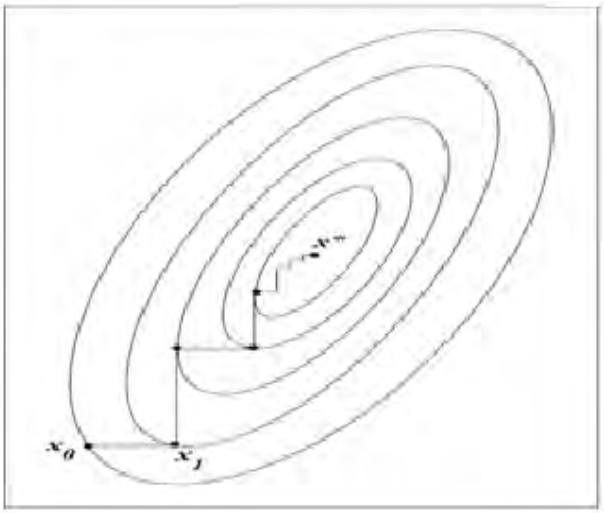


Figure 2.1: Coordinate descent

2.2 Algorithm: Cyclic Coordinate Method

step 0 Choose sufficiently small scalar $\epsilon > 0$ and initial guess x_1. Let k=1
step 1 Set $y_1 = x_k$
step 2 For j=1 to n do:
-Let t_j be a minimum point of $\phi(t) = f(y_j + te_j), t \in [a, b]$
-

$$y_{j+1} = y_j + t_j e_j$$

end - for step 3 Let $x_{k+1} = y_{n+1}$
step 4 If$||x_{k+1} - x_k|| < \varepsilon$, then Stop; otherwise let k=k+1 and repeat from 1

2.3 Hooke and Jeeves

the method of Hooke and Jeeves perform two types of search exploratory search and pattern search .Tthe first two iteration of the procedure are illustrated in figure () given x_1 ,an exploratory search along the coordinate directions produces the point x_2. Now a pattern search along the direction $x_2 - x_1$ leads to the point y_1. another exploratory search starting from y gives the point x_3 . The next parttern search is along the direction $x_3 - x_2$ yielding y_2 the proces is repeated

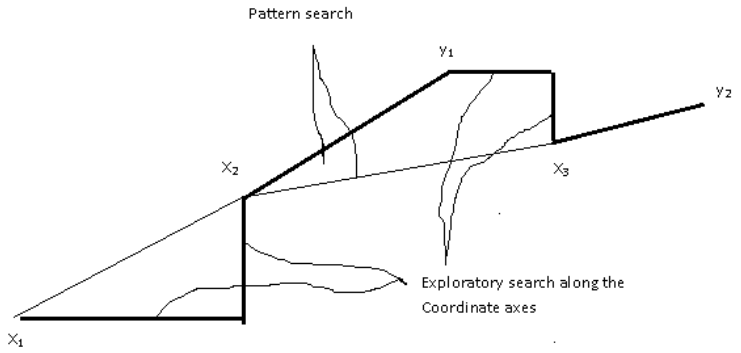


Figure 2.2: Illustration of the method of Hooke and Jeeves

2.3.1 Algorithm Hooke and Jeeves

- initial step: choose a scalar $\epsilon > 0$ to be used in terminating the algorithm
- choose a starting point x_1.let $y_1 = x_1$
  let k=j=1 and go to the main step

  1. Let λ_j be an optimal solution to the problem to
     minimize $f(y_j + \lambda d_j)$, $\lambda \in \Re$
     $y_{j+1} = y_j + {}_j d_j$

2. If $j < n$, replace j by j + 1, and repeat Step 1
   Otherwise, if j = n, let $x_{k+1} = y_{n+1}$
3. If $||x_{k+1} - x_k|| < \varepsilon$, stop;otherwise, go to Step2
4. Let d = $x_{k+1} - x_k$
5. Let λ^* be an optimal solution to the problem to
   minimize $f(x_{k+1} + \lambda d)$,$\lambda \in \Re$
6. Let $y_1 = x_{k+1} + \lambda^* d$ Let j = 1
7. replace k by k+1 ,Go to Step 1

2.4 the Method of Rosenbrock

the method of Rosenbrock dose not employ line search but rather takes direction steps along the search direction .
the method of Rosenbrock tests n orthogonal search directions which are equal to coordinate axes only at the first iteration .
in this method we search to in direction of n linearly independent and orthogonal vectors in each iteration .when a new point
is reached at the end of an iteration a new set of orthogonal vector is constructed
in figure the new dirction are denoted by $\bar{d_1}$ and $\bar{d_1}$

2.4.1 Constrction of the Search Direction

let $d_1, d_2.....d_n$ be linearly independent vectors , each with a norm equal to 1 furthermore , suppose that these vector are mulually orthogonal , that is
$d_1^t d = 0$ for $i \neq j$
starting from the current vector x_k , the objective function f is minimized along each of the direction iteratively , resulting in the point x_{k+1} .
in particular

$$x_{k+1} - x_k = \sum_{j=1}^{n} \lambda_j d_j$$

where λ_j is the distance moved along d_j .
the new collection of directions $\bar{d_1}, \bar{d_2}.....\bar{d_n}$ is formed by Gram-Schidt procedure , pronominalization procedure,

$$a_j = \begin{cases} d_j = & \text{if } \lambda_j = 0 \\ \sum_{i=j}^{n} \lambda_i d_i = & \text{if} \lambda_j \neq 0 \end{cases}$$

$$a_j = \begin{cases} b_j = & j = 0 \\ a_j - \sum_{i=1}^{j-1} (a_j^t \bar{d_1}) \bar{d_1} = & j \geq 0 \end{cases}$$

$$\bar{d_j} = \frac{d_j}{||b_j||} \tag{2.1}$$

the following theorem shows that the new directions established by Rosenbrock are indeed linearly independent and orthogonal.

2.4.2 Theorem

suppose that the vectors $d_1, ..., d_n$ are linearly independent and mutually orthogonal then the direction $d_1^-, ..., d_n^-$ defined by(2.1) are also linearly independent and mutually orthogonal for any set of $\lambda_1, ..., \lambda_n$. furthermore, if $\lambda_j = 0$ then $\bar{d_j} = d_j$. **Proof** we first show that $a_1, ..., a_n$ are linearly independent . suppose that

$$\sum_{j=1}^{n} \mu_j a_j = 0$$

let $I = \{j : \lambda_j = 0\}$,and let $J(j) = \{i : i \neq I, i \leqslant j\}$
noting we get

$$0 = \sum_{j=1}^{n} \mu_j a_j = \sum_{j \notin 1} \mu_j d_j + \sum_{j \notin 1} \mu_j (\sum_{i=j}^{n} \lambda_j d_j)$$

=

$$\sum_{j \in 1} \mu_j d_j + \sum_{j \notin 1} (\lambda_j \sum_{i \in J(j)}) d_j$$

since $d_1, ...d_n$ are linearly independent
$\mu_j = 0$ for $j \in I$ and $\lambda_j \sum_{i \in J(j)} \mu_i = 0$ for $j \notin I$. But $\lambda_j \neq 0$ for $j \notin I$ $\lambda_j \sum_{i \in J(j)} \mu_i = 0$ hence for each $j \notin I$ by the definition of J(j) we therefore have $\mu_1 = ... = \mu_n = 0$ and ,hence , $a_1, ...a_n$ are linearly independent.
To show that $b_1, ..., b_n$ are linearly independent , we use the following induction argument . since $b_1 = a_1 \neq 0$ it suffices to show that if $b_1, ...b_n$ are linearly independent , then $b_1, ...b_n$ are also linearly independent . suppose that

$$\sum_{j=1}^{k+1} = \alpha_j b_j = 0$$

using the definition of b_{k+1}

$$0 = \sum_{j=1}^{k} \alpha_j b_j + \alpha_{k+1} b_{k+1} \tag{2.2}$$

$$\sum_{j=1}^{k} [\alpha_j - \frac{\alpha_{k+1}(a_{k+1}^t \bar{d_j})}{||b_j||}] b_j + \alpha_{k+1} a_{k+1}$$

from (2.2) it follows that each vector b_j is a linearl combination for $a_1, ...a_j$ since

$a_1, ...a_{k+1}$ are linarly independent , it follows from (2.2) that $\alpha_{k+1} = 0$. since $b_1, ..., b_k$ ar assumed linearly independent by the induction hypotheses , from (2.2) we get

$$\alpha_j - \frac{\alpha_{k+1}(a_{k+1}^t \bar{d_j})}{||b_j||} = 0$$

for j= 1,.....,k
since $\alpha_{k+1} = \alpha_j = 0$ for each j ,this show thart $b_1, ...b_{k+1}$ are linearly independent . by the definition of $\bar{d_j}$, linearly independent of$\bar{d_1}, ..., \bar{d_n}$is immediate . now wwe show orthogonality of $b_1, ..., b_k$ and hence , orthogonality of$\bar{d_1}....., \bar{d_n}$
from (2.2), $b_1^t d_2 = 0$; and thus it suffices to show that $b_1, ..., b_k$ are mutually orthogonal then $b_1, ..., b_k$,b_{k+1} are also mutually orthogonal from (2.2) and n0ting that $b_j^t \bar{d_i} = 0$ for $i \neq j$, it follows that

$$b_j^t \bar{b_{k+1}} = \bar{b_k}[a_{k+1} - \sum_{i=1}^{k} (a_{k+1}^t \bar{d_{k+1}}) \bar{d_{k+1}}]$$

$$b_j^t a_{k+1} - (a_{k+1}^t \bar{d_j}) b_j^t \bar{d_j} = 0$$

Thus , $a_1, ...a_{k+1}$ are mutually orthogonal.
to complete the proof , we show that $\bar{d_j} = d_j$ if $\lambda_j = 0$ from (2.2) , if $\lambda_j = 0$ we get

$$b_j = d_j - \sum_{i=1}^{j-1} \frac{1}{||b_j||} (d_j^t b_i) \bar{d_i}$$

Note that b_i is a linear combination of $a_1, ...a_i$ so that

$$b_i = \sum_{r=1}^{i} \beta_{ir} a_r$$

$$b_i = \sum_{r \in R} \beta_{ir} d_r + = \sum_{r \in \bar{R}} \beta_{ir} (\sum_{s=r}^{n} \lambda_s d_s$$

where $R = \{r : r \leq i, \lambda_r = 0\}$ and $\bar{R} = \{r : r \leq i, \lambda_r \neq 0\}$ consider i¡j and note that $d_j^t d_v = 0$ for $r \in R$, $r \leq i < j$ and hence $d_j^t d_r = 0$ for $r \notin R$
by assumption , $\lambda_j = 0$ and
d_j^t we get $d_j^t b_j = 0$ for i¡j it follows that $b_j = d_j$ and hence $\bar{d_j} = d_j$
this completes the proof

2.4.3 Algorithm:The Method of Rosenbrock

initial step: let $\epsilon > 0$ be the termination scalar ,
choose $d_1, d_2, d_3, ...d_n$ as the coordinate directions,

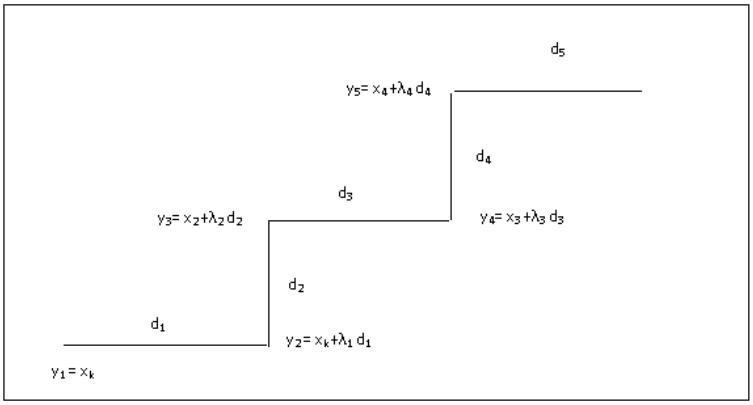


Figure 2.3: method of Rosenbrock

choose a starting point x_1 . let $y_1 = x_1$, k=j=1 and go to the main step
main step

1. let λ_i be an optimal solution to the problem to minimize
   $f(y_j + \lambda d_j)$ subject to $\lambda \in R$ and let $y_{k+1} = y_j + \lambda_j d_j$,
   if $j < n$ replace j by j+1 and replace step 1
   otherwise go to step 2

2. let $x_{k+1} = y_{k+1}$, if $||x_{k+1} - x_k|| < \epsilon$ then stop,
   otherwise let $y_1 = x_{k+1}$, replace k by k+1 . let j=1

3. generate new n orthogonal search direction $D_1, D_2, ...D_n$ as described above
   and let $d_j = D_j$ for each j= 1,2,3,.....n
   set k =k+1 and
   repeat from step1

Chapter 3

LEVENBERG- MARGUARDT TRUST REGION METHOD

Line search methods and trust-region methods both generate steps with the help of a quadratic model of the objective function, but they use this model in different ways. Line search methods use it to generate a search direction, and then focus their efforts on finding a suitable step lengthα this direction. Trust-region methods define a region around the current iterate within which they trust the model to be an adequate representation of the objective function, and then choose the step to be the approximate minimizer of the model in this region. In effect, they choose the direction and length of the step simultaneously. If a step is not acceptable, they reduce the size of the region and find a new minimizer. In general, the direction of the step changes whenever the size of the trust region is altered. The size of the trust region is critical to the effectiveness of each step. If the region is too small, the algorithm misses an opportunity to take a substantial step that will move it much closer to the minimizer of the objective function. If too large, the minimizer of the model may be far from the minimizer of the objective function in the region, so we may have to reduce the size of the region and try again. In practical algorithms, we choose the size of the region according to the performance of the algorithm during previous iterations. If the model is consistently reliable, producing good steps and accurately predicting the behavior of the objective function along these steps, the size of the trust region may be increased to allow longer, more ambitious, steps to be taken. A failed step is an indication that our model is an inadequate representation of the objective function over the current trust region. After such a step, we reduce the size of the region and try again. the trust-region approach on a function f of two variables in which the current pointx_k and the minimizerx_0 lie at opposite ends of a curved valley. The quadratic model function m_k , whose elliptical contours are shown as dashed lines, is constructed from function and derivative information at x_k and possibly also on information accumulated from previous iterations and steps. A line search method based on this model searches along the step to the minimizer ofm_k (shown), but this direction will yield at most a small reduction in f , even if the optimal step length is used. The trust-region method steps to the minimizer of m_k within the dotted circle (shown), yielding a more significant reduction in f and better progress toward the solution.

3.1 THE LEVENBERG - MARQUARDT METHOD

The algorithm was first published in 1944 by Kenneth Levenberg,[1] while working at the Frankford Army Arsenal. It was rediscovered in 1963 by Donald Marquardt[2] who worked as a statistician at DuPont and independently by Girard, is a method for solving nonlinear equations. This method is often mentioned when the history of trust region algorithms is discussed. The reason is that the technique of trust region is, in some sense, equivalent to that of the Levenberg-Marquardt method. Consider a system of nonlinear equations
Non-linear least squares is the form of least squares analysis used to fit a set of m observations with a model that is non-linear in n unknown parameters ($m > n$). It is used in some forms of non-linear regression. The basis of the method is to approximate the model by a linear one and to refine the parameters by successive iterations

3.2 Thoery:

Consider a set of m data points, $(x_1, y_1), (x_2, y_2)$,...,(x_m, y_m), and a curve (model function)$y = f(x, \beta)$, that in addition to the variable x also depends on n parameters, $\beta = (\beta_1, \beta_2, ..., \beta_n)$, with $m \geq n$. It is desired to find the vector β of parameters such that the curve fits best to the given data in the least squares sense, that is, the sum of squares

$$s = \sum_{i=1}^{m} r_i^2$$

is minimized, where the residuals (errors)r_i are given by
$r_i = y_i - f(x_i, \beta)$
for i= 1,2,...,m.
The minimum value of S occurs when the gradient is zero. Since the model contains n parameters there are n gradient equations:

$$\frac{\partial s}{\partial \beta_j} = 2 \sum_i r_i \frac{\partial r_i}{\partial \beta_j} = 0$$

(j =1,2,...,n)
In a non-linear system, the derivatives $\frac{\partial r_i}{\partial \beta_j}$ are functions of both the independent variable and the parameters, so these gradient equations do not have a closed solution. Instead, initial values must be chosen for the parameters. Then, the parameters are refined iteratively, that is, the values are obtained by successive approximation

$$\beta_j \approx \beta_j^{k+1} = B_j^k + \Delta \beta_j$$

Here, k is an iteration number and the vector of increments, $\Delta\beta$ is known as the shift vector. At each iteration the model is linearized by approximation to a first-order Taylor

series expansion about β^k

$$f(x_i, \beta) \approx f(x_i, \beta^k) + \sum_i \frac{\partial f(x_i, \beta^k)}{\partial \beta_j}(\beta_j - \beta_j^k) \approx f(x_i, \beta^k) + \sum_i J_{ij}\Delta_j$$

The Jacobian, J, is a function of constants, the independent variable and the parameters, so it changes from one iteration to the next. Thus, in terms of the linearized model, $\frac{\partial r_i}{\partial \beta_j} = -J_{ij}$and the residuals are given by

$$\Delta y_i = y_i - f(x_i, \beta^k)$$

$$r_i = y_i - f(x_i, \beta) = (y_i - f(x_i, \beta^k)) + (f(x_i, \beta^k)) - f(x_i, \beta) = \Delta y_i - \sum_{s=1}^{n} J_{ij}\Delta\beta_s$$

Substituting these expressions into the gradient equations, they become

$$-2\sum_{i=1}^{m} J_{ij}(\Delta y_i - \sum_{s=1}^{n} Jij\Delta\beta_s) = 0$$

which, on rearrangement, become n simultaneous linear equations, the normal equations

$$\sum_{i=1}^{m}\sum_{s=1}^{n} J_{ij}J_{is}\Delta\beta_s = \sum_{i=1}^{m} J_{ij}\Delta y_i$$

, (j=1,2.....n)
The normal equations are written in matrix notation as
$(J^T J)\Delta\beta = J^T\Delta y$
When the observations are not equally reliable, a weighted sum of squares may be minimized,

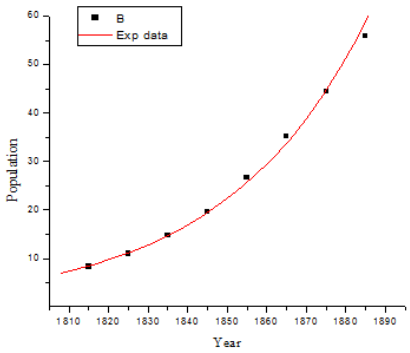
$$s = \sum_{i=1}^{m} W_{ii}r_i^2$$

Each element of the diagonal weight matrix W should, ideally, be equal to the reciprocal of the error variance of the measurement.[1] The normal equations are then
$(J^T W J)\Delta\beta = J^T W\Delta y$
These equations form the basis for the GaussNewton algorithm for a non-linear least squares problem.

3.2.1 Example: Exponential Data

Here is a set of data for the United States population (in millions)
and the corresponding year

year	1815	1825	1835	1845	1855	1865	1875	1885
population	8.3	11.0	14.7	19.7	26.7	35.2	44.4	55.9



The set of data points is $d(t_j, y_j)$ where t_j is the year and y_j is the population. The model function is $\phi(x;t) = x_1 e^{x_2} t$. Here is a graph of the data points and the model with $x_1 = 6$ and $x_2 = 3$ as our initial guesses: our residual vector is a vector of the components

$r_j = 6e^{3t} - y_j$. r(x)$= \begin{bmatrix} 6e^{3(1)} - 8.3 \\ 6e^{3(2)} - 11 \\ 6e^{3(3)} - 14.7 \\ 6e^{3(4)} - 19.7 \\ 6e^{3(5)} - 26.7 \\ 6e^{3(6)} - 35.2 \\ 6e^{3(7)} - 44.4 \\ 6e^{3(8)} - 55.9 \end{bmatrix} = \begin{bmatrix} -0.200847 \\ -0.0672872 \\ 0.0576187 \\ 0.220702 \\ 0.190134 \\ 1.09788 \\ 4.59702 \\ 10.2391 \end{bmatrix}$

The goal is to minimize the least-squares problem $f(x) = \frac{1}{2}||r(x)||_2^2$.
hence $||r(x)||^2 = 127.309$therefore
f(x) = 63.6545.

3.3 GaussNewton Method

Recall that the GaussNewton method is like Newtons method with line search, except that we use the convenient and often effective approximation for the Hessian. The LevenbergMarquardt method can be obtained by using the same Hessian approximation, but replacing the line search with a trust-region strategy. The use of a trust region avoids one of the weaknesses of GaussNewton, namely, its behavior when the Jacobian J(x) is rank-deficient, or nearly so. Since the same Hessian approximations are used in each case, the local convergence properties of the two methods are similar. The LevenbergMarquardt method can be described and analyzed using the trust region framework (In fact, the LevenbergMarquardt method is sometimes considered to be the progenitor of the trust-region approach for general unconstrained optimization For a spherical trust region,
The GaussNewton algorithm is used to solve non-linear least squares problems. It is a modification of Newton's method for finding a minimum of a function. Unlike Newton's method, the GaussNewton algorithm can only be used to minimize a sum of squared function values, but it has the advantage that second derivatives, which can be challenging to compute, are not required. Non-linear least squares problems arise for instance in non-linear regression, where parameters in a model are sought such that the model is in good agreement with available observations

3.4 GaussNewton algorithm

Given m functions $r = (r_1, ..., r_m)$ (often called residuals) of n variables $\beta = (\beta_1, ..., \beta_n)$, with $m \geq n$, the GaussNewton algorithm iteratively finds the value of the variables which minimizes the sum of squares[1]

$$s(\beta) = \sum_{i=1}^{m} r_i^2(\beta)$$

Starting with an initial guess $\beta^{(0)}$ for the minimum, the method proceeds by the iterations $\beta^{(s+1)} = \beta^{(s)} - (J_r^T J_r)^{-1} J_r^T r(\beta^{(s)})$
where, if r and β are column vectors, the entries of the Jacobian matrix are

$$(J_r)_{ij} = \frac{\partial(\beta^{(s)})}{\partial(\beta_j)}$$

and the symbol T denotes the matrix transpose
If m = n, the iteration simplifies to

$$\beta^{(s+1)} = \beta^{(s)} - (J_r)^{-1} r(\beta^{(s)})$$

which is a direct generalization of Newton's method in one dimension.
In data fitting, where the goal is to find the parameters β such that a given model function $y = f(x, \beta)$ best fits some data points(x_i, y_i), the functions r_i are the residuals

$r_i(\beta) = y_i - f(x_i, \beta)$

Then, the Gauss-Newton method can be expressed in terms of the Jacobian J_f of the function f as

$$\beta^{(s+1)} = \beta^{(s)} + (J_f^T J_f)^{-1} J_f^T r(\beta^{(s)}$$

3.4.1 Example

Guass Newton method .we apply the Gauss Newton method to an exponential model of the form $y_i = x_1 e^{x_2 t_i}$ with the data

t	1	2	4	5	8
y	3.2939	4.2699	7.1749	9.3008	20.259

using an initial guess that close to the solution $x = \begin{bmatrix} 2.50 \\ 0.25 \end{bmatrix}$ at this point

$f(x) = \begin{bmatrix} -0.0838 \\ -0.1481 \\ -0.3792 \\ -0.5749 \\ -1.7864 \end{bmatrix}$ and $\nabla f(x)^T = \begin{bmatrix} 1.2840 & 3.2101 \\ 1.6487 & 8.2436 \\ 2.7183 & 27.1828 \\ 3.4903 & 43.6293 \\ 7.3891 & 147,7811 \end{bmatrix}$

hence , $\nabla f(x) = \nabla f(x) f(x) = \begin{bmatrix} -16.5888 \\ -300.8722 \end{bmatrix}$

$\nabla f(x) \nabla f(x)^T = \begin{bmatrix} 78.5367 & 1335.8479 \\ 1335.8479 & 24559.9419 \end{bmatrix}$ The Gauss -Newton search direction is obtained by solving the linear system . $\nabla f(x) \nabla f(x)^T = -\nabla f(x) f(x)$

$\nabla^2 f(x) d = -\nabla f(x)$

$d = -\nabla^2 f(x)^{-1} \nabla f(x)$

$d = \begin{bmatrix} 0.0381 \\ 0.0102 \end{bmatrix}$

and the new estimate of the solution is $x_2 = x_1 + d = \begin{bmatrix} 2.5381 \\ 0.2602 \end{bmatrix}$

3.5 Trust Region Method

Trust-region method (TRM) is one of the most important numerical optimization methods in solving non-linear programming (NLP) problems. It works in a way that first define a region around the current best solution, in which a certain model (usually a quadratic model) can to some extent approximate the original objective function. TRM then take a step forward according to the model depicts within the region. Unlike the line search methods, TRM usually determines the step size before the improving direction (or at the same time). If a notable decrease (our following discussion will based on minimization problems) is gained after the step forward, then the model is believed to be a good representation of the original objective function. If the improvement is too subtle or even a negative improvement is gained, then the model is not to be believed as a good representation of the original objective function within that region. The convergence can be ensured that the size of the trust region (usually defined by the radius in Euclidean norm) in each iteration would depend on the improvement previously made In most cases, the trust-region is defined as a spherical area of radius Δ_k in which the trust-region subproblem lies.

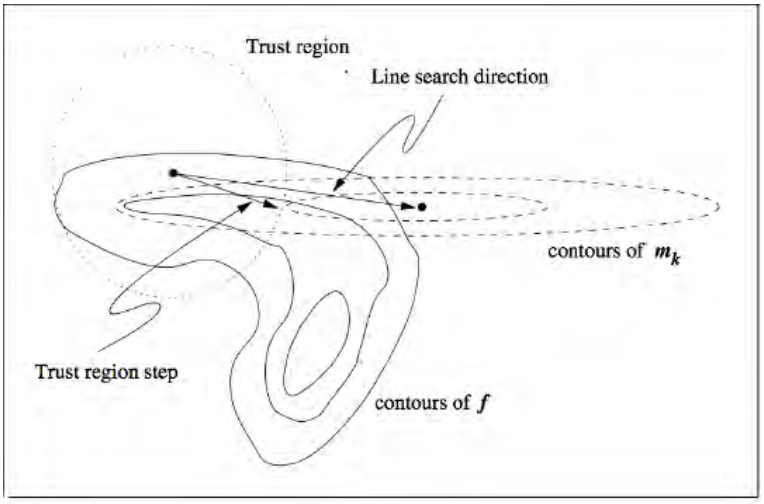


Figure 3.1: Trust Region Geometric Intuition

3.6 Trust-region subproblem

If we are using the quadratic model to approximate the original objective function, then our optimization problem is essentially reduced to solving a sequence of trust-region subporblems

$$minm_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p$$

$$\text{s.t } ||p|| \leq \Delta_k$$

Where Δ_k is the trust region radius, g_k is the gradient at current point and B_k is the hessian (or a hessian approximation). It is easy to find the solution to the trust-region subproblem if B_k is positive definite

3.6.1 Actual reduction and predicted reduction

The most critical issue underlying the trust-region method is to update the size of the trust-region at every iteration. If the current iteration makes a satisfactory reduction, we may exploits our model more in the next iteration by setting a larger Δ_k. If we only achieved a limited improvement after the current iteration, the radius of the trust-region then should not have any increase, or in the worst cases, we may decrease the size of the trust-region by adjusting the radius to a smaller value to check the models validity

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} = \frac{Actualreduction}{predictedreduction}$$

Whether to take a more ambitious step or a more conservative one is depend on the ratio between the actual reduction gained by true reduction in the original objective function and the predicted reduction expected in the model function. Empirical threshold values of the ratio ρ_k will guide us in determining the size of the trust-region.

3.6.2 Trust Region Algorithm

Before implementing the trust-region algorithm, we should first determine several parameters. Δ_M is the upper bound for the size of the trust region. η_1, η_2 and η_3, t_1, t_2 are the threshold values for evaluating the goodness of the quadratic model thus for determining the trust-regions size in the next iteration. A typical set for these values are $0 =< \eta_1 <= \eta_2, \eta_2 = 0.25$ and $\eta_3 = 0.75, t_1 = 0.25, t_2 = 2.0$

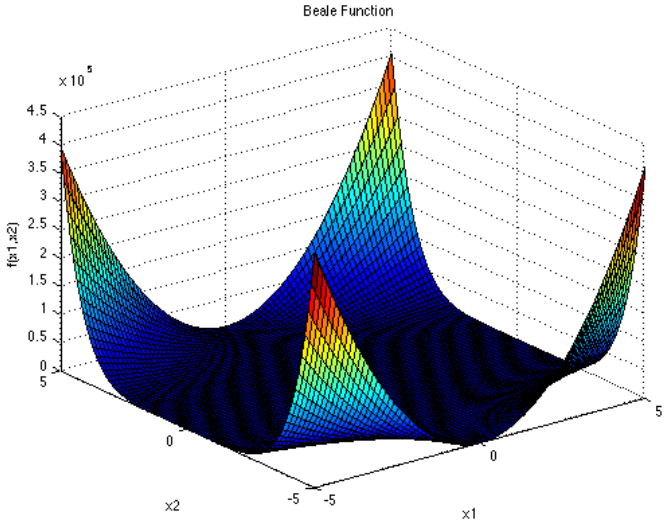
Set the initial point at x_k, set k=1
for k=1,2..., $\Delta_m > 0$, $\Delta_0 \in (0, \Delta_m)$ and constant $\eta \in (0, \frac{1}{4})$
if $r_k < \frac{3}{4}$
$\Delta_{k+1} = \frac{1}{4}\Delta_k$
else
if $r_k > \frac{3}{4}$ and$||d_k|| = \Delta_k$(full step and model is a good approximation)
$\Delta_{k+1} = min(2\Delta_k, \Delta_m)$, where Δ_m is the maximum allowed Δ_k.
else
$\Delta_{k+1} = \Delta_k$
if $r_k > \frac{1}{4}$
else
$x_{k+1} = x_k + d_k$
end if
k=k+1
end whil $x_{k+1} = x_k$(the model is not a good approximation and need to solve another trust-region subproblem within a smaller trust-region)

3.6.3 Example

the 2D Beale function [10]
The function to be minimized is given by

$$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$$

where, number of variable n =2



search domain $= -4.5 \leq x_i \leq 4.5$ for i =1,3
$a_1 = 1.5$, $a_2 = 2.25$, $a_3 = 2.625$
the initial point is $x_0 = (0.1, 0.1)^T$ and $\Delta_1 = 0.8$ this function has a valley approaching the line $x_2 = 1$
solution
$x \in \Re^2$ given

$$x_0 = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

Step 1
since n=2 we need m=2n+1 = 2(2)+1 = 5 points
the initial m points $x_1 = 1, 2, ..., 5$ can be chosen as follows

$$x_1 = x_0 = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

and

$$\begin{cases} x_{i+1} = x_0 + \Delta_1 e_i ....., i = 1, 2, ...n. \\ x_{i+2+1} = x_0 + \Delta_1 e_i ....., i = 1, 2, ...n - 1. \end{cases}$$

$$i = 1, x_{1+1} = x_2 = x_0 + \Delta_1 e_1 = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} + 0.8 \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}$$

$$i = 2, x_{2+1} = x_3 = x_0 + \Delta_1 e_2 = \begin{bmatrix}0.1\\0.1\end{bmatrix} + 0.8\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0.1\\0.9\end{bmatrix}$$

$$i = 1, x_{1+2+1} = x_4 = x_0 - \Delta_1 e_1 = \begin{bmatrix}0.1\\0.1\end{bmatrix} - 0.8\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}-0.7\\0.1\end{bmatrix}$$

$$i = 2, x_{2+2+1} = x_5 = x_0 - \Delta_1 e_2 = \begin{bmatrix}0.1\\0.1\end{bmatrix} - 0.8\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0.1\\-0.7\end{bmatrix}$$

Step 2 Construct the initial quadratic model

let $x_0 = \begin{bmatrix}0.1\\0.1\end{bmatrix}$ be the initial trust region center .
set k=1

To construct the model we use
$M^1(x_i) = f(x_i).....i = 1, 2...., 5.$
where $M^1 = a_1 + b^T(x - x_1) + \frac{1}{2}(x - x_1)^T B(x - x_1)$ assuming B is a diagonal matrix , and $b \in \Re^2$
i=1 , $x_i = x_1 = \begin{bmatrix}0.1\\0.1\end{bmatrix}$
$M^1 = f(x_1) = a_1 + b^T(0) + \frac{1}{2}(0)^T B(0) = f\begin{bmatrix}0.1\\0.1\end{bmatrix}$

$a_1 = (1.5 - 0.1 + (0.1)(0.1))^2 + (2.25 - 0.1 + (0.1)(0.1)^2)^2 + (2.625 - 0.1 + (0.1)(0.1)^3)^2$

$a_1 = 1.9881 + (2.151)^2 + 92.5251)^2 = 12.99101$

i=2 , $x_i = x_2 = \begin{bmatrix}0.9\\0.1\end{bmatrix}$
$M^1(x_2) = a_1 + (b_1 b_2) + (\begin{bmatrix}0.9\\0.1\end{bmatrix} - \begin{bmatrix}0.1\\0.1\end{bmatrix}) + \frac{1}{2}(\begin{bmatrix}0.9\\0.1\end{bmatrix} - \begin{bmatrix}0.1\\0.1\end{bmatrix})^T \begin{bmatrix}a & 0\\0 & b\end{bmatrix} (\begin{bmatrix}0.9\\0.1\end{bmatrix} - \begin{bmatrix}0.1\\0.1\end{bmatrix})$
$= f(\begin{bmatrix}0.9\\0.1\end{bmatrix})$
$M^1(x_2) = 12.999103101 + \begin{bmatrix}b_1 & b_2\end{bmatrix}\begin{bmatrix}0.8\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}0.8 & 0\end{bmatrix}\begin{bmatrix}a & 0\\0 & b\end{bmatrix}\begin{bmatrix}0.8\\0.1\end{bmatrix} = f\begin{bmatrix}0.9\\0.1\end{bmatrix}$
$\Rightarrow 12.99103101 + 0.8b_1 + 0.32a = f\begin{bmatrix}0.9\\0.1\end{bmatrix}$
$f\begin{bmatrix}0.9\\0.1\end{bmatrix} = (1.5-0.9+(0.9)(0.1))^2+(2.25-0.9+(0.9)(0.1)^2)^2+(2.625-0.9+(0.9)(0.1)^3)^2$
$= 0.4761+1.846881+2.97873081= 5.30171181$

thus ,$12.99103101 + 0.8b_1 + 0.32a = 5.30171181$

$0.8b_1 + 0.32a = -7.6893192$.....(1)

i=3 , $x_i = x_3 = \begin{bmatrix}0.1\\0.9\end{bmatrix}$

$$M^1(x_3) = 12.99103101 + \begin{bmatrix}b_1 & b_2\end{bmatrix}\begin{bmatrix}0\\0.8\end{bmatrix} + \tfrac{1}{2}\begin{bmatrix}0 & 0.8\end{bmatrix}\begin{bmatrix}a & 0\\0 & b\end{bmatrix}\begin{bmatrix}0\\0.8\end{bmatrix} = f\begin{bmatrix}0.1\\0.9\end{bmatrix}$$

$\Rightarrow 12.99103101 + 0.8b_2 + 0.32b = f\begin{bmatrix}0.1\\0.9\end{bmatrix}$

$f\begin{bmatrix}0.1\\0.9\end{bmatrix} = (1.5-0.1+(0.1)(0.9))^2+(2.25-0.1+(0.1)(0.9)^2)^2+(2.625-0.1+(0.1)(0.9)^3)^2$

thus $12.99103101 + 0.8b_2 + 0.32b = 6.3189$

$0.8b_2 + 0.32b = -6.67213101$.....(2)

i=4 , $x_i = x_4 = \begin{bmatrix}-0.7\\-0.2\end{bmatrix}$

$M^1(x_4) = 12.99103101+\begin{bmatrix}b_1 & b_2\end{bmatrix}(\begin{bmatrix}-0.7\\-0.1\end{bmatrix}-\begin{bmatrix}0.1\\0.1\end{bmatrix})+\frac{1}{2}(\begin{bmatrix}-0.7\\-0.1\end{bmatrix}-\begin{bmatrix}0.1\\0.1\end{bmatrix})^T\begin{bmatrix}a & 0\\0 & b\end{bmatrix}(\begin{bmatrix}-0.7\\-0.1\end{bmatrix}-\begin{bmatrix}0.1\\0.1\end{bmatrix})$

$= f(\begin{bmatrix}-0.7\\-0.1\end{bmatrix})$

$$\Rightarrow 12.99103101 + \begin{bmatrix}b_1 & b_2\end{bmatrix}\begin{bmatrix}-0.8\\-0.2\end{bmatrix} + \tfrac{1}{2}\begin{bmatrix}-0.8 & -0.2\end{bmatrix}B\begin{bmatrix}-0.8\\-0.2\end{bmatrix} = f\begin{bmatrix}-0.7\\-0.1\end{bmatrix}$$

$\Rightarrow 12.9991-0.8b_1-2b_2+0.32a+0.02b = f\begin{bmatrix}-0.7\\-0.1\end{bmatrix}$ $f\begin{bmatrix}-0.7\\-0.1\end{bmatrix} = (1.5+0.7-(0.7)(-0.1))^2+$
$(2.25+0.7-(0.7)(-0.1)^2)^2+(2.625-0.7-(0.7)(-0.1)^3)^2$

thus ,$12.99103101 - 0.8b_1 - 0.2b_2 + 0.32a + 0.02b = 24.87442949$

$-0.8b_1 - 0.2b_2 + 0.32a + 0.02b = 11.88339848$.....(3)

i=5 , $x_i = x_5 = \begin{bmatrix}-0.1\\-0.7\end{bmatrix}$

$M^1(x_5) = 12.99103101+\begin{bmatrix}b_1 & b_2\end{bmatrix}(\begin{bmatrix}-0.1\\-0.7\end{bmatrix}-\begin{bmatrix}0.1\\0.1\end{bmatrix})+\frac{1}{2}(\begin{bmatrix}-0.1\\-0.7\end{bmatrix}-\begin{bmatrix}0.1\\0.1\end{bmatrix})^T\begin{bmatrix}a & 0\\0 & b\end{bmatrix}(\begin{bmatrix}-0.1\\-0.7\end{bmatrix}-$

$\begin{bmatrix}0.1\\0.1\end{bmatrix}$)

$= f(\begin{bmatrix}-0.1\\-0.7\end{bmatrix}$)

$\Rightarrow 5.804781 - 0.5b_1 - 0.5b_2 + 0.125b = f\begin{bmatrix}-0.1\\-0.7\end{bmatrix}$

thus

$\Rightarrow 12.99103101 - 0.2b_1 - 0.8b_2 + 0.02a + 0.32b = f\begin{bmatrix}-0.1\\-0.7\end{bmatrix}$

$\Rightarrow 12.99103101 - 0.2b_1 - 0.8b_2 + 0.02a + 0.32b = 15.32336749$

$-0.2b_1 - 0.8b_2 + 0.02a + 0.32b = 2.33233648$.....(4)

using the above four equation we have

$$\begin{bmatrix}0.8 & 0 & 0.32 & 0\\ 0 & 0.8 & 0 & 0.32\\ -0.8 & 0.2 & 0.32 & 0.02\\ -0.2 & -0.8 & 0.02 & 0.32\end{bmatrix}\begin{bmatrix}b_1\\b_2\\a\\b\end{bmatrix} = \begin{bmatrix}-7.6893192\\-6.67213101\\11.88339848\\2.33233648\end{bmatrix}$$

using Cramer's Rule on the above , we get

a=1.972, b=-10.095 , $b_1 = -10.4$, $b_2 = -4.302$

thus, $M^1(x) = a_1 + b^T(x) + \frac{1}{2}(x)^T B(x)$

where $a_1 = 12.991, b = \begin{bmatrix}-10.4\\-4.302\end{bmatrix} \in \Re^2$,$B = \begin{bmatrix}a & 0\\0 & b\end{bmatrix} = \begin{bmatrix}1.972 & 0\\0 & -10.095\end{bmatrix}$

step 3 Minimize the model

$$minM^1(s)$$
$$s.t||s|| \leq 0.8$$

where$s = \begin{bmatrix}s_1\\s_2\end{bmatrix}$

This implies

$M^1\begin{bmatrix}s_1\\s_2\end{bmatrix} = 12.991 + \begin{bmatrix}-10.4 & -4.302\end{bmatrix}\begin{bmatrix}s_1\\s_2\end{bmatrix} + \frac{1}{2}\begin{bmatrix}s_1 & s_2\end{bmatrix}\begin{bmatrix}1.972 & 0\\0 & -10.095\end{bmatrix}\begin{bmatrix}s_1\\s_2\end{bmatrix}$

$||\begin{bmatrix}s_1\\s_2\end{bmatrix}|| \leq 0.8$

$\Rightarrow M^1 \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = 12.991 - 10.4s_1 - 4.302s_2 + 0.988(s_1)^2 - 5.048(s_2)^2$

$\sqrt{(s_1)^2 + (s_2)^2} \leq 0.8$

The lagrange function defined by

$L(S_1, s_2, \mu) = 12.991 - 10.4s_1 - 4.302s_2 + 0.988(s_1)^2 - 5.048(s_2)^2 + \mu((s_1)^2 + (s_2)^2 - 0.64)$

to solve this we use KKT condition.

$1.L(s_1) = -10.4 + 1.976s_1 + 2\mu s_1 = 0$ $2.L(s_2) = -4.302 - 10.096S_2 + 2\mu s_2 = 0$ $3.\mu((s_1)^2 + (s_2)^2 - 0.64) = 0$ $4.((s_1)^2 + (s_2)^2 - 0.64 \leq 0$ $5.\ \mu \geq 0$

Case 1: let $\mu = 0$ $10.4 + 1976s_1 = 0$ and $-4.302 - 10.096s_2 = 0$

$\Rightarrow s_1 = \frac{10.4}{1.976} = 5.279$ and $s_2 = \frac{4.303}{10.096} = 0.426$

for $s_1 = 5.279$ and $s_2 = 0.426$ condition 4 not satisfied

Case 2: let $\mu > 0$ then $(s_1)^2 + (s_2)^2 = 0.64$

$L(s_1) = -10.4 + 1.976s_1 + 2\mu s_1 = 0$, $\mu = \frac{10.4 - 1.97s_1}{2s_1}$

$L(s_2) = -4.302 - 10.096s_2 + 2\mu s_2 = 0$, $\mu = \frac{4.302 + 10.096s_2}{2s_2}$

From the above we have $\begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 0.523 \\ 0.605 \end{bmatrix}$

step 4: Compute the reduction ratio

the new point $x_n = x_k + s$ where $x_k = x_1 = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$ and $s = \begin{bmatrix} 0.523 \\ 0.605 \end{bmatrix}$

$\Rightarrow x_n = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.523 \\ 0.605 \end{bmatrix} = \begin{bmatrix} 0.623 \\ 0.705 \end{bmatrix}$

$f(x_k) = f(x_1) = f \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} = 12.991$

$f(x_n) = f \begin{bmatrix} 0.623 \\ 0.705 \end{bmatrix} = 10.121$

$M^1(x_1) = M^1 \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} = 11.48$

$M^1(x_n) = M^1 \begin{bmatrix} 0.623 \\ 0.705 \end{bmatrix} = 2.0938$

Now the reduction will be

$r_k = \frac{f(x_k) - f(x_n)}{M^1(x_k) - M^1(x_n)} = \frac{12.991 - 10.121}{11.48 - 2.0983} = 0.306$

Step 5: update the trust region center

$r_1 = 0.306$,since $0.1 \leq r_1 \leq 0.7$, $\Delta_{k+1} = \Delta_2 = \Delta_1 = 0.8$

Step 6: determine the trust region center
Since $r_1 = 0.306 > 0$,then the new center is
$x_{k=1} = x_2 = x_1 + s = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.523 \\ 0.705 \end{bmatrix} = \begin{bmatrix} 0.623 \\ 0.705 \end{bmatrix}$

Chapter 4

Test Examples

in the chapter , we choose two simple problems to illustrate some of the differences between the three multidimensional derivative- free optimization method ,Cyclic Coordinate method ,Hooke and Jeeves Method,Rosenbrock Method , Trust Region Method ,and implement the programs to find a minimizing point and minimum value of function f

4.1 Example 1

consider the following problem minimize $(x_1 - 2)^4 + (x_1 - 2x_2)^2$

4.1.1 Solve Examples 1 by using Cyclic Coordinate method

with initial point (0 ,0) using Cyclic Method Starting point $x_1 = (6.5, 0)$

Table 4.1: Cyclic Coordinate method

k	$x_k = (x_1, x_2)$	$f(x_k)$
1	(6.500000,0.000000)	28583.750000
2	(0.331742,0.663584)	-0.885174
3	(0.632244,1.264531)	-1.497000
4	(0.898185,1.796471)	-1.887819
5	(1.125331,2.250781)	-2.109623
6	(1.307311,2.614722)	-2.212871
7	(1.434397,2.868914)	-2.245696
8	(1.493430,2.986959)	-2.249957
9	(1.500010,3.000139)	-2.250000
10	(1.499991,3.000082)	-2.250000

after 10 iteration ,10 function evaluations and by using 10^{-4} tolerance (ϵ)
we get the solution ,The minimum point is: x*=(1.49999,3.00008),
The Minimum value is: f(x*)=-2.25000.

4.1.2 Solve Example 1 by using Hooke and Jeeves Method

with initial point $(0, 0)$ using Hooke and Jeevs Method at $x_1 = (6.5, 0)$

Table 4.2: Hooke and Jeeves Method

k	xk=(x_1, x_2)	$f(x_k)$
1	(6.5000,0.000000)	28583.750000
2	(-4.4298,-8.859350)	32.913106
3	(-3.4600,-6.919885)	22.351829
4	(1.5000,3.000123)	-2.250000
5	(1.5000,3.000105)	-2.250000

After 5 iteration ,5 function evaluations and by using 10^{-4} tolerance (ϵ) we get the solution ,The minimum point is: x*=(1.50001,3.00010),
The Minimum value is f(x*)= -2.25000.

4.1.3 Solve Example 1 by using Rosenbrock Method

with initial point (0,0) using Rosenbrock Method at $x_1 = (6.5, 0)$ After 16iteration 16

Table 4.3: Rosenbrock Method

k	xk=(x_1, x_2)	$f(x_k)$
1	(6.500000,0.000000)	28583.750000
2	(0.331742,0.663584)	-0.885174
3	(0.746455,1.865302)	-1.662939
4	(1.420093,2.582877)	-2.239231
5	(1.432750,2.640139)	-2.242898
6	(1.499423,2.948237)	-2.249993
7	(1.499577,2.953727)	-2.249996
8	(1.499990,2.982814)	-2.250000
9	(1.500011,2.990997)	-2.250000
10	(1.500007,2.996814)	-2.250000
11	(1.499985,3.001144)	-2.250000
12	(1.500018,2.998493)	-2.250000
13	(1.499981,2.999923)	-2.250000
14	(1.499981,2.999179)	-2.250000
15	(1.500000,2.999427)	-2.250000
16	(1.500018,2.999406)	-2.250000

function evaluations and by using 10^{-4} tolerance (ϵ) we get the solution ,The minimum point is: x*=(1.50002,2.99941)
The Minimum value is f(x*)= -2.25000.

Chapter 5

Appendices

5.1 Matlab code for Cyclic Coordinate method

```
%function [xk,fxk]=cyclicmethod(x1,tol)
% x1=[10,10];
% x1=[10,7];
% x1=[6,1];
% x1=[5,1];
x1=[6.5,0];
tol=10^(-4);
n=length(x1);
e=eye(n);
xprev=x1-1;
k=1;
xk=x1;
disp('          ')
disp('Evaluating Minimum Value of f(x,y)=(x-2)^4+(x-2y)^2')
disp('with initial point [0 0] using Cyclic Method Starting
point x1=[6.5,0]')
disp('          ')
disp('_____________________________________________')
disp('  k            xk=(x,y)               f(xk)')
disp('_____________________________________________')
disp('          ')
fprintf('%2.0f      (%8.6f,%8.6f)      %10.6f\n',
k,xk(1),xk(2),f(xk))
while norm(xk-xprev)>tol&k<50
    xprev=xk;
    for j=1:n
        ej=e(j,:);
        tj=OneDim(xk,ej,tol);
        xk=xk+tj*ej;
    end
    fk=f(xk);
    k=k+1;
 fprintf('%2.0f      (%8.6f,%8.6f)      %10.6f\n',
k,xk(1),xk(2),f(xk))
end
disp('_____________________________________________')
disp('        ')
fprintf('The minimum point is:   x*=(%6.5f,%6.5f).\n',xk)
disp('          ')
fprintf('The Minimum value is:   f(x*)=%6.5f.\n',fk)
disp('       ')
disp('       ')
disp('-----------------------------------------------')
disp('       ')
```

5.2 Matlab code for Hooke and Jeeves Mehtod

```
function [xk,fxk]=HookJeevs(x1,tol)
% x1=[10,10];
% x1=[10,7];
% x1=[6,1];
% x1=[5,1];
x1=[6.5,0];
tol=10^(-4);
n=length(x1);
e=eye(n);
xprev=x1-1;
k=1;
xk=x1;
xprev=x1+1;
disp('Evaluating Minimum Value of f(x,y)=(x_1-2)^4+(x_1-
2*x_2)^2')
disp('with initial point [0 0] using Hooke & Jeevs Method at
x1=x1=[6.5,0]')
disp('       ')
disp('_____________________________________________')
disp('k            xk=(x,y)        f(xk)')
disp('_____________________________________________')
disp('       ')
fprintf('%2.0f      (%2.4f,%2.6f)     %4.6f
\n',k,xk(1),xk(2),f(xk))
  while norm(xk-xprev)>tol & k<50
    d=xk-xprev;
    xprev=xk;
    y=xk;
    t=OneDim(y,d,tol);
    y=xk+t*d;
    for j=1:n
        ej = e(j,:);
        tj=OneDim(y,ej,tol);
        y=y+tj*ej;
     end
    xk=y;
    fk=f(xk);
    k=k+1;
 fprintf('%2.0f      (%2.4f,%2.6f)     %4.6f
\n',k,xk(1),xk(2),f(xk))
  end
disp('_____________________________________________')
disp('                                ')
fprintf('The minimum point is:  x*=(%6.5f,%6.5f)\n',xk)
disp('                                ')
fprintf('The Minimum value is f(x*)=%10.5f.\n',fk)
disp('       ')
disp('       ')
disp('-------------------------------------------------')
disp('       ')
```

5.3 Matlab code for Rosenbrock Method

```
%function [xk,fxk]=rosenbrockmethod(x1,tol)
% x1=[10,10];
% x1=[10,7];
% x1=[6,1];
% x1=[5,1];
x1=[6.5,0];
tol=10^(-4);
xprev=x1-1;
n=length(x1);
d=eye(n);
k=1;
xk=x1;
fk=f(xk);
disp('      ')
disp('Evaluating Minimum Value of f(x,y)=(x_1-2)^4+(x_1-2*x_2)^2')
disp('with initial point [0 0] using Rosenbrock Method at
x1=[6.5,0]')
disp('      ')
disp('_____________________________________________')
disp(' k            xk=(x,y)               f(xk)')
disp('_____________________________________________')
disp('        ')
fprintf('%3.0f      (%4.6f,%4.6f)      %4.6f
\n',k,xk(1),xk(2),f(xk))
while norm(xk-xprev)>tol&k<50
    xprev=xk;
    y=xk;
    for j=1:n
        dj=d(j,:);
        t(j)=OneDim(y,dj,tol);
        y=y+t(j)*dj;
    end
    xk=y;
    fk=f(xk);
    k=k+1;
fprintf('%3.0f      (%4.6f,%4.6f)      %4.6f
\n',k,xk(1),xk(2),f(xk))
dt=[d;t];
d=orthonormal(dt);
end
disp('_____________________________________________')
disp('                                  ')
fprintf('The minimum point is:  x*=(%6.5f,%6.5f)\n',xk)
disp('                                  ')
fprintf('The Minimum value is f(x*)=%10.5f.\n',fk)
disp('      ')
disp('      ')
disp('-----------------------------------------------------')
disp('      ')
```

Conclusions

line search methods and trust region methods. In this paper we give a review on derivative free optimization or method s that do not use derivatives , the line -search algorithm,s choose search direction d_k at the current iteration x_k and then , take a step to next iteration x_{k+1} with lower function value along this direction this step has the length of λ from the method that we discuses above , cyclic coordinate method , Hooke and Jeeves Mehtod and Rosenbrock Method for multidimensional, direct search method is tend to converge more slowly , but can not be tolerant Levenberg-Marquardt algorithms and Trust region algorithms are both Newton Step-based methods (they are called Restricted Newton Step methods). Thus they both exhibits quadratical speed of convergence near x^* When we are far from the solution (x_k far from x^*), we can encounter a negative curvature (H_k negative definite).
If this happens, Levenberg-Marquardt algorithms will slow down dramatically. In opposition, Trust Region Methods will perform a very long step δ_k and move quickly to a more interesting area.
the result obtaining showed the ability of the proposed method to rapidly converge to the final region containing the optimum solution when only a limited number of function evaluations is possible and when a high accuracy is not really . Thus , the proposed method is suitable for stochastic optimization or objectives that suffer from numerical inaccuracy .

Bibliography

[1] E. de Klerk, C. Roos, and T. Terlaky, Nonlinear Optimization (pdf) (ps), 1999-2004, Delft.

[2] trust region optimization approach; Adv Res (2014) pp.4-11

[3] Nocedal, J. and Wright, S. J. (1999). Numerical Optimization. Springer- Verlag.

[4] G. Yuan and X. Lu, A New Line Search Method with Trust Region for Unconstrained Optimization, Communications on Applied Nonlinear Analysis, Vol. 15, No. 1, 2008, pp. 35-49.

[5] John Wily and Son's Inc (1979-1993) ,Non-linear programming , Theory and Algorithm

[6] Fletcher, R. (1987). Practical Methods of Optimization, second edition. John Wiley.

[7] Abdel-Karims .O.Hassen ,Hany Labdel-Malek , Ahmed S.A Mohamed , Tamer M .Abdulfadel , Ahamed E RFcavity design expoiting a new derivative free Jorge Nocedal, Stephen J.Wright: Numerical Optimization, Springer Series in Operations Research (1999).5]

[8] H.H Rosenbrock , An Automatic method for finding the greatest or the least Value of a function , comput .J., Vol.3, 1960

[9] Gill, Philip E.; Murray, Walter. Algorithms for the solution of the nonlinear least-squares problem. SIAM Journal on Numerical Analysis 15 (5): 977-992.

[10] X.Want , Derivative - free otimization Algorithm,s , Report , Department of computing and Software , Mc Master University , 2003

DECLARATION

I, the undersigned, declare that this project is my original work, has not been presented for degrees in any other university and all sources of material used for the project have been duly acknowledged.

Name:ARNEST FANGARSIO SHABAN

Signature:

Place: College of Natural Sciences, Addis Ababa University

Date: