بسم الله الرحمن الرحيم

NILE VALLEY UNIVERSITY

COLLEG OF POST GRADNATESTUDS

ALGORITHMS OF UNCONSTRAINED

OPTIMIZATION METHODS

Athese submitted patail fill fullment for

the degree of m.s.c.in mathematics

Search by: Mansor Nasr Eldeen Tawfeg

Supervised by: Dr. Ahmed Abd alfadil Mohamed

1. Jun.201

بسم الله الرحمن الرحيم

(تبرك الذي بيده الملك وهو علي كل شئ قدير (١) الذي خلق الموت والحيوة ليبلوكم أيكم

أحسن عملا وهو العزيز الغفور(٢)   )

(صدق الله العظيم)

سوره تبارك الايه(١.٢)

<u>DEDICATION</u>

I am delighted to dedicate this work to my parents (my father and to the spirit of my mother dear), my brothers, and I also dedicate this work to all those who made it possible for me to complete this study in it present form. Moreover I would like to dedicate this work to my best ever brother and friend , ALI TAG ALSER AND HUSHM OSMAN for his continuous support and kindness and being always helping me. Really words failed to express my grateful to you my brother and I find difficult to thank you in behalf of all my family for unstopping guide ness and support all of us. My right hand AHMED TAG ALSER, words don't describe you and I will be an debate to you all my life hope my god help and support me to being in the best overview, but sure that all the family will leave the reword to our God, only can reword you. I hope God keep you and support you. So I will dedicate this work to you my brother KHAILD

<u>ACKNOWLEDGEMENTS</u>

First of all, praise to the Almighty God who made it

possible for me to carryout this study

Then, I am greatly in debate to my supervisor Dr. AHMAD

ALFADE continuous follow up, and support.

I am also very grateful to my colleagues through out the whole

period of Education moreover to my parents, to my FATHER

AND, family, and friends,

<u>ABSTRACT</u>

The aim of the research is to give algorithms for unconstrained optimization. We present Conjugate Gradient, Damped Newton and Quasi Newton methods to gather with the relevant theoretical background. The reader is assumed to be familiar with algorithms for solving linear and none. Linear system of equations,

الملخص

الهدف من هذا البحث هو عمل خوارزميات لحل مسائل الامثلية الغير مقيدة لسهولة فهمها واستخدامها وقد استخدمت طرق الامثلية مثل طرق التدرج والانحدار و نيوتن والمقارنة بينهما في سرعة التقارب والدقة وكما استخدمت الخوارزميات لحل انظمه لمعادلات الخطية والغير خطية

<u>*CONTENTS*</u>

Chapter three the steepest descent method:

Chapter four conjugate gradient method

Chapter five Newton type method

Chapter one introduction

INTRODUCTION

In this search we shall discuss numerical methods for the solution of the optimization problem. For a real function of several real variables we want to find an argument vector which corresponds to a minimal function   Value
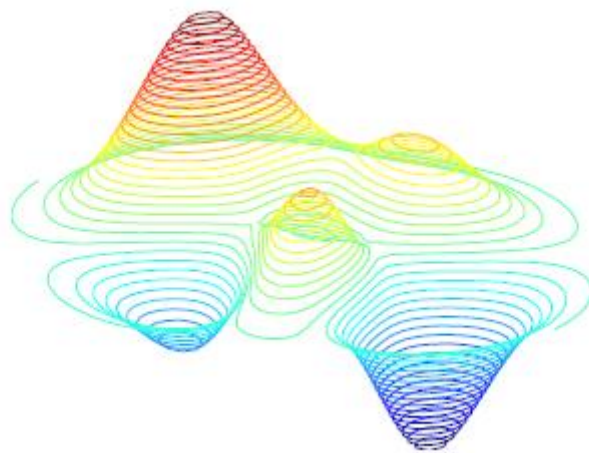
1.1Definition.   The optimization problem:

Find $x^* = \text{argmin}_x f(x)$,   where $f : \mathbf{R}^n \mapsto \mathbf{R}$. the function f is called the *objective function* or *cost function* and $\mathbf{x}^*$ is the   *minimizer.*In some cases we want a *maximizer* of a function. This is easily determined If   we   find a minimizer of the function   with   opposite   sign. Optimization plays an important role in many branches of science and applications: Economics, operations research, network analysis, optimal design of   mechanical   or   electrical   systems,   to

mention   but   a few.

Example 1.1.

consider functions of one variable. the function $f(x) = (x - x^*)^2$ has one,

unique minimizer, x^*, see Figure (1.1.) $y = (x - x^*)^2.$

Figure 1.1: *One   minimizer.*

1.2 INTRODUCTION

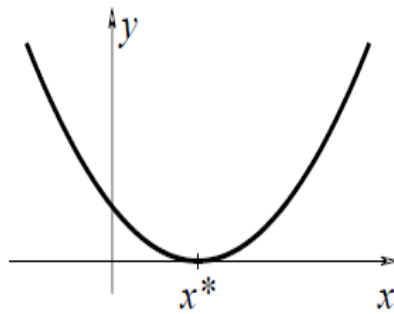The function $f(x) = 2 \cos(x- x^*)$ have infinitely many minimizers: $x = x^*+2p$
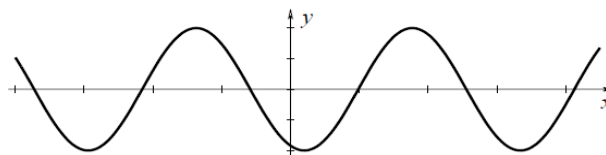
where p is an integer; see Figure 1.2.

Figure 1.2: $y = 2 \cos (x - x^*)$ *many minimizers.*

the function $f(x) = 0.015(x - x^*)2 - 2 \cos(x - x^*)$ has a unique global minimizer, x^*

besides that, it also has several so called *local minimizers,* each giving the

minimal function value inside a certain region, see Figure 1.3.
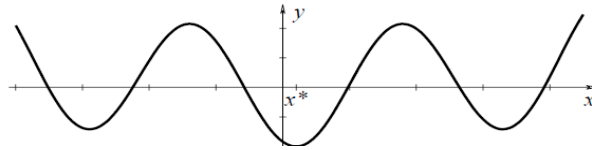
Figure 1.3: $y = 0.015(x - x^*)^2 - 2\cos(x - x^*).$

One global   minimizer   and many local minimizers.

The ideal situation for optimization computations is that the objective function has a unique minimizer. We call this the *global minimizer* .In some cases the objective function has several (or even infinitely many) minimizers. In such problems it may be sufficient to find one of these minimizers. In many objective functions from applications we have a global minimizer and several local minimizers. It is very difficult to develop methods which can find the global minimizer with certainty in this situation the methods described here can find a local minimizer for the objective function. When a local minimizer has been discovered, we do not know whether it is a global minimizer or one of the local minimizers. We cannot even be sure that our optimization method will find the local minimizer

1.3 INTRODUCTION

Closest   to the starting point. In order to explore several local minimizers we can try several runs with different starting points, or better still examine intermediate results produced by a global minimizer. we end this section with an example meant to

demonstrate that optimization methods based on too primitive ideas may be

dangerous.

Example 1.2.

We want the global minimizer of the function

$$f(\mathbf{x}) = (x_1 + x_2 - 2)^2 + 100(x_1 - x_2)^2 .$$

The idea (which we should **not** use) is the following: "Make a series of

iterations. In each iteration keep one of the variables fixed and seeks a value of

the other variables so as to minimize the *f*-value". In Figure 1.4 we show the *level*

curves or *contours* of *f*, i-e .curves consisting of positions with the same *f*-value.

we also show the first few iterations.

Figure 1.4: *The Method of Alternating Variables fails t determine the minimizer*

of quadratic
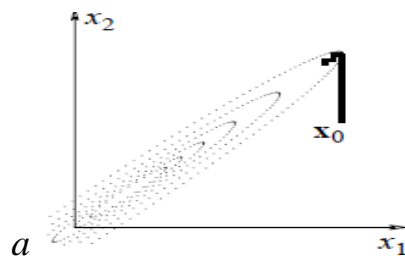
After some iteration the steps begin to decrease rapidly in size. They can become

so small that they do not influence the **x**-values, because these are represented

with finite precision in the computer, and the progress stops completely. In many

cases this happens far away from the solution. We say that the iteration is caught

in (*Stiefel's cage.*) the "method" is called the *method of alternating variables* and

it is a classical example of a dangerous method, a method we must avoid.

1.1. Conditions for a Local Minimizer

A local minimizer for *f* is an argument vector giving the smallest function

Value inside a certain region, defined by **X*** is a *local minimizer* for

$f : \mathbb{R}^n \mapsto \mathbb{R},$ if f(x*) $\leq$ f(x)   for//x*-x// $\leq$ ε

most objective functions, especially those with several local minimizers,

contain local maximizers and other points which satisfy a necessary condition for

a local minimizer. The following theorems help us find such points and

distinguish the local minimizers from the irrelevant points. We assume that *f* has

continuous partial derivatives of second order.

- First order *(Taylor expansion)* for a function of several variables gives

    usan approximation to the function value at a point **x+h** neighbouring **x**,

$$f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x}) + \mathbf{h}^\top f'(\mathbf{x}) + \tfrac{1}{2}\mathbf{h}^\top f''(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|^3) \qquad (1.1)$$

Where **f** (**x**) is the *(gradient)* of *f*, a vector containing the first partial derivatives,

$$f'(\mathbf{x}) \equiv \begin{bmatrix} \dfrac{\partial f}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \dfrac{\partial f}{\partial x_n}(\mathbf{x}) \end{bmatrix}.$$

we only consider vectors h   with   // **h** // so small that the last term in (1.3) is

negligible compared with the middle term. If the point **x** is a local minimizer it is

not possible to find an **h** so that F (**x** + **h**) < f(**x**) with // **h** // small enough. This

together with (1.3) is the

Theorem 1.5. **necessary condition for a local minimum.**

If **x*** *is* a local minimizer for $f : \mathbf{R}^n \mapsto \mathbf{R}$, then $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$.

The local minimizers are among the points with $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. they have a special

name.

Definition 1.5 Stationary point.

If $\mathbf{f}'(\mathbf{x}_s) = \mathbf{0}$, then $\mathbf{x}_s$ is said to be a *(stationary point)* for f. the stationary

points are the local maximizers, the local minimizers to distinguish between

them, we need one extra term in the Taylor expansion. Provided that f has

continuous third derivatives, then

$$f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x}) + \mathbf{h}^\top \mathbf{f}'(\mathbf{x}) + \tfrac{1}{2}\mathbf{h}^\top \mathbf{f}''(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|^3) \qquad (1.1)$$

Where the *Hessian* $\mathbf{f}(\mathbf{x})$ of the function f is a matrix containing the second Partial

derivatives of f :

$$\mathbf{f}''(\mathbf{x}) \equiv \left[\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}) \right] .$$

Note that this is a symmetric matrix. For stationary point takes the

Form $\qquad f(\mathbf{x}_s + \mathbf{h}) = f(\mathbf{x}_s) + \tfrac{1}{2}\mathbf{h}^\top \mathbf{f}''(\mathbf{x}_s)\mathbf{h} + O(\|\mathbf{h}\|^3) .$

If the second term is positive for all **h** we say that the matrix f '' (**x**s) is

(Positive definite) which also gives tools for checking definiteness). Further,

we can take //**h**// so small that the remainder term is negligible, and it follows that

١٦

x s is a local minimizer

Theorem 1.1.   **sufficient   condition for a local minimum.**

Assume that **x** s is a stationary point and that **f″** (**x** s) is positive definite. Then **x**

sis a local minimizer. The Taylor expansion is also the basis of the proof of the

following corollary   Assume that **x** s is a stationary point and that **f** ″(**x**) is

Positive semi definite when **x** is in a neighbourhood of **x**s. then **x**s is a Local

minimizer. The local maximizers and "the rest", which we call (*saddle points),*

can be characterized

Corollary 1. 2.

Assume that **x**s is a stationary point and that

$f''(x_s) \neq 0.$ Then

1) if $f''(x_s)$ is positive definite: see Theorem ⠀ (1.1)

2) if $f''(x_s)$ is positive semidefinite: x_s is a local minimizer or a saddle point,

3) if $f''(x_s)$ is negative definite: x_s is a local maximizer,

4) if $f''(x_s)$ is negative semidefinite: x_s is a local maximizer or a saddle point,

5) if $f''(x_s)$ is neither definite nor semidefinite: x_s is a saddle point.

If **f** ″ (**x** s) =**0**, then we need higher order terms in the Taylor expansion in order to

find the local minimizers among the stationary points.

Example 1.3. We consider functions of two variables. Below we show the

variation of the function value near a local minimizer (Figure 1.5(a)), a local

maximizer (Figure 1.5(b)) and a saddle point (Figure 1.5(c)). It is a characteristic

of a saddle point that there exists one line through ($\mathbf{x}_s$) , with the property that if

we follow the variation of the f-value along the line, this "looks like" a local

minimum, whereas there exists another line through($\mathbf{x}_s$), "indicating" a local

maximizer.

a) minimum   b) maximum   c) saddle point
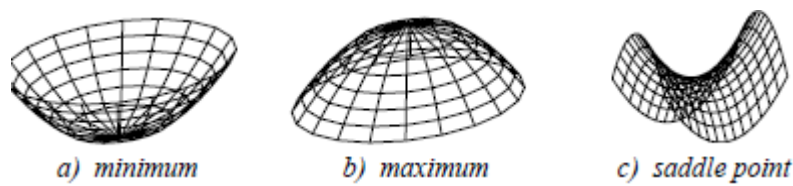
Figure 1.5: *With a 2-dimsurfacensional* $\mathbf{x}$ *we see as*

$z = f(\mathbf{x})$ *near a stationary point*

1.6. INTRODUCTION   we see curves approximately like concentric ellipses

near a local maximizer or a local minimizer (Figure1.6(a)),whereas the saddle

points exhibit the "hyperbolaes" shown in (Figure 1.6(b)).

a) maximum or minimum   b) saddle point
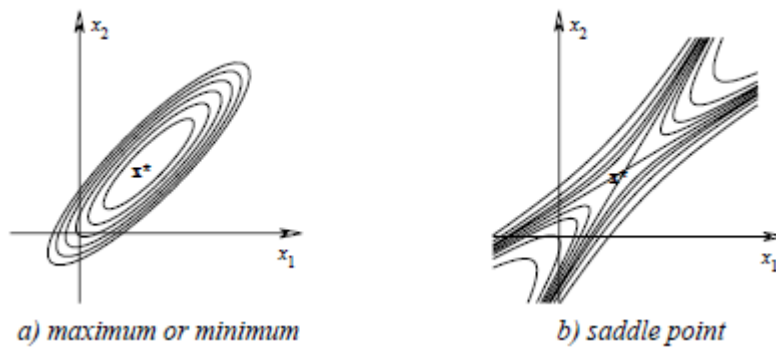
Figure 1.6: *The contours of a function near a stationary point*

Finally, the Taylor expansion (1.1) is also the basis for the following Theorem.

<u>Theorem 1. 3</u>. Second order necessary condition.

If $\mathbf{x}^*$ is a local minimizer, then $\mathbf{f}''(\mathbf{x}^*)$ is positive semidefinite.

Chapter two

2. DESCENT METHODS:

Introduction:

These are(*iterative methods).* They produce   a series of vectors

$X0.X1.X2$……(2.1(a) Which in most cases converges under certain mild

conditions? We want the series to converge towards **x*** a local minimizer for the given objective function $f : \mathbf{R}^n \mapsto \mathbf{R}$, i.e. $\mathbf{x}k \to \mathbf{x}*$ for $k \to \infty$ (2.1(b)) where **x*** is a local minimizer, the methods here are measures which enforce the descending property

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k) .$$

(2.2)

This prevents convergence to a maximizer and also makes it less probable That we get convergence to a saddle point, We talk about the (*global convergence)* properties of a method, i.e. convergence when the iteration starts in a position $\mathbf{X}_0$, which is not close to a local minimizer **x*** we want our method to produce iterates that move steadily towards a neighbourhood. of **x***.For instance, there are methods (i.e. limit of a subseries) of *($\mathbf{X}_k$)* is a stationary Point (Definition 1.6), i.e. the gradients tend to zero

$$\mathbf{f'} (\mathbf{X_k}) \to \mathbf{0} \qquad \text{for } k \to \infty \quad (2.3)$$

This does not exclude convergence to a saddle point or even a maximizer, but the descending property (2.2) prevents this in practice. In this "global part" of the iteration we are satisfied if the current errors do not increase except for the very first steps. Letting $\{\mathbf{e}_k\}$ denote the errors, $\mathbf{e}_k \equiv \mathbf{x}_k - \mathbf{x}^*$,

the requirement is $\|\mathbf{e}_{k+1}\| < \|\mathbf{e}_k\|$ for $k > K$.

In the final stages of the iteration where the $\mathbf{X}_k$ are close to **x*** we expect Faster

convergence. The local convergence results tell us how quickly we Can get a result which agrees with **x*** to a desired accuracy. Some methods have (*linear convergence*):

$$\|e_{k+1}\| \le c_1\|e_k\| \quad \text{with } 0 < c_1 < 1 \text{ and } x_k \text{ close to } x^* . \quad (2.4)$$

It is more desirable to have higher order of convergence, for instance *quadratic convergence* (convergence of order 2):

$$\|e_{k+1}\| \le c_2\|e_k\|^2 \quad \text{with } c_2 > 0 \text{ and } x_k \text{ close to } x^* . \quad (2.5)$$

Only a few of the methods used in the applications achieve quadratic final Convergence On the other hand we want better than linear final convergence. Many of the methods used in practice have   (*super linear convergence*):

$$\frac{\|e_{k+1}\|}{\|e_k\|} \to 0 \quad \text{for } k \to \infty . \quad\quad\quad (2.6)$$

This is better than linear convergence though (normally) not as good as Quadratic convergence.

Example 2.1. Consider 2 iterative methods, one with linear and one with quadratic convergence. At a given step they have both achieved the result with an accuracy of 3 decimals: $\|e_k\| \le 0.0005$. They have $c_1 = c_2 = \frac{1}{2}$ in (2.4) and (2.5)

<div align="center">Algorithm 2.1 Structure of descent methods</div>

```
begin
    k := 0;  x := x₀;  found := false                    {Starting point}
    repeat
        h_d := search_direction(x)                       {From x and downhill }
        if no such h exists
            found := true                                {x is stationary}
        else
            Find "step length" α                         { see below}
            x := x + αh_d                                {new position }
            k := k + 1
            found := update(found)
    until found or k>k_max
end                                                      {... of descent algorithm }
```

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) < \delta_2 \,.$$

Both conditions reflect the convergence **x**k--------**x***. They cannot be used in

practice, however, because **x*** and $f(\mathbf{x}^*)$ are not known. Instead we have to use

approximations to these conditions: $\quad \|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon_1 \quad$ (2.7)

$$f(\mathbf{x}_k) - f(\mathbf{x}_{k+1}) < \varepsilon_2 \,. \tag{2.8}$$

We must emphasize that even if (2.8) is fulfilled with small ε_1″ and ε_2,: we

Cannot be sure that $\|\mathbf{e}_k\|$ or $f(\mathbf{x}_k) - f(\mathbf{x}^*)$ is small. The other type of

convergence mentioned at the beginning of this chapter is $\quad$ **f** (**X**k) →-0 for k-

→ ∞ .

This can be reflected in the stopping criterion $\quad \|\mathbf{f}'(\mathbf{x}_k)\| < \varepsilon_3 \,,$ (2.9)

Which is included in many implementations of descent methods? There is a

good way of using the property of converging function values. The Taylor

expansion (1.1) of f at $\mathbf{x}^*$ is

$$f(\mathbf{x}_k) \simeq f(\mathbf{x}^*) + (\mathbf{x}_k - \mathbf{x}^*)^\top \mathbf{f}'(\mathbf{x}^*) + \tfrac{1}{2}(\mathbf{x}_k - \mathbf{x}^*)^\top \mathbf{f}''(\mathbf{x}^*)(\mathbf{x}_k - \mathbf{x}^*) .$$

Now, if $\mathbf{x}^*$ is a local minimizer, then $\mathbf{f}'(\mathbf{x}^*) = 0$ and $\mathbf{H}^* = \mathbf{f}''(\mathbf{x}^*)$ is positive

semidefinite, This gives us

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \simeq \tfrac{1}{2}(\mathbf{x}_k - \mathbf{x}^*)^\top \mathbf{H}^*(\mathbf{x}_k - \mathbf{x}^*) ,$$

so the stopping criterion could be

$$\tfrac{1}{2}(\mathbf{x}_{k+1} - \mathbf{x}_k)^\top \mathbf{H}_k(\mathbf{x}_{k+1} - \mathbf{x}_k) < \varepsilon_4 \quad \text{with } \mathbf{x}_k \simeq \mathbf{x}^* . \qquad (2.10)$$

Here $\mathbf{x}_k - \mathbf{x}^*$ is approximated by $\mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{H}^*$ is approximated by $\mathbf{H}_k = \mathbf{f}''(\mathbf{x}_k)$.

.2. Descent Directions

From the current position we wish to find a direction which brings us downhill,

to descent direction. This means that if we take a small step in that direction we

get to a position with a smaller function value. The Taylor expansion (1.1) gives

us a first order approximation to the function value in a neighbouring point to $\mathbf{x}$

in direction $\mathbf{h}$:

$$f(\mathbf{x}+\alpha\mathbf{h}) = f(\mathbf{x}) + \alpha\mathbf{h}^\top \mathbf{f}'(\mathbf{x}) + O(\alpha^2), \quad \text{with } \alpha > 0 .$$

If a is not too large, then the   First   two terms will dominate over the l

$$f(\mathbf{x} + \alpha\mathbf{h}) \simeq f(\mathbf{x}) + \alpha\mathbf{h}^\top \mathbf{f}'(\mathbf{x}) .$$

٢٤

The sign of the term $\alpha h^{\top} f'(x)$ decides whether we start of uphill or downhill.

In the space $\mathbf{R}^n$ we consider a hyper plane H through the current position and

orthogonal to $\mathbf{f}'(\mathbf{x})$, $$\mathcal{H} = \{x + h \mid h^{\top} f'(x) = 0\} .$$

This hyper plane divides the space in an "uphill" half space and a "downhill" half

space. The half space we want has the vector - $\mathbf{f}'(\mathbf{x})$ pointing into it. gives the

situation in $\mathbf{R}^3$. Figure 2.1: $\mathbf{R}^3$. *"downhill" and an* uphill *divided into*
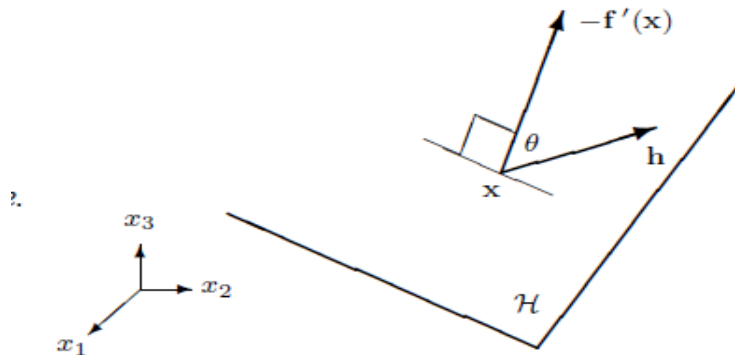
. Figure 2.1

we now define a descent direction. this is a "downhill" direction, i-e, it is

inside the "good" half space:

<u>Definition 2.1. Descent direction.</u>

$\mathbf{h}$ is a descent direction from $\mathbf{x}$ if $h^{\top} f'(x) < 0$. *:* A method based on

successive descent directions is a *descent method*. In Figure 2.1 we have a

descent direction $\mathbf{h}$. We introduce the angle between $\mathbf{h}$ and -$\mathbf{f}'(\mathbf{x})$,

$$\theta = \angle(\mathbf{h}, -\mathbf{f}'(\mathbf{x})) \quad \text{with} \quad \cos\theta = \frac{-\mathbf{h}'\,\mathbf{f}'(\mathbf{x})}{\|\mathbf{h}\| \cdot \|\mathbf{f}'(\mathbf{x})\|}\,.$$

We state a condition on this angle,

Definition 2.2 (Absolute descent method)

this is a method, where the search directions $\mathbf{h}_k$ satisfy $\theta < \dfrac{\pi}{2} - \mu$ for all

k, with $\mu > 0$ independent of k the discussion above is concerned with the

geometry in $\mathbb{R}^3$, and is easily seen to be valid also in $\mathbb{R}^2$. If the dimension n

is larger than 3, we call μ the ("*pseudo angle)* between **h** and-**f'** (**x**)". In this way

we can use this definition for all $n \geq 2.$. the restriction that *¹* must be constant

in all the steps is necessary for the Global convergence result

Search **Theorem 2.1**

If $\mathbf{f}'(\mathbf{x}) \neq 0$ ¡ And **B** is a symmetric, positive definite Matrix, then

$$\mathbf{h}_1 = -\mathbf{B}\mathbf{f}'(\mathbf{x}) \quad \text{and} \quad \mathbf{h}_2 = -\mathbf{B}^{-1}\mathbf{f}'(\mathbf{x})$$

Are descent directions.

Proof. A positive definite matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ satisfies

$$\mathbf{u}^\top \mathbf{B}\mathbf{u} > 0 \quad \text{for all } \mathbf{u} \in \mathbb{R}^n,\ \mathbf{u} \neq 0\,.$$

If we take **u**=**h**1 and exploit the symmetry of **B**, we ge

$$\mathbf{h}_1^\top \mathbf{f}'(\mathbf{x}) = -\mathbf{f}'(\mathbf{x})^\top \mathbf{B}^\top \mathbf{f}'(\mathbf{x}) = -\mathbf{f}'(\mathbf{x})^\top \mathbf{B}\mathbf{f}'(\mathbf{x}) < 0\,.$$

With **u**=**h**2 we get

$$h_2^\top f'(x) = h_2^\top(-B\,h_2) = -h_2^\top B\,h_2 < 0 \,.$$

Thus, the condition in Definition (2.1) is satisfied in both cases.

2.3. Descent Methods with Line Search

After having determined a descent direction, it must be decided how long

The step in this direction we shall introduce the idea of *line search*. and the

variation of the objective function f along the direction   **h** from the current

position **x**, $\varphi(\alpha) = f(x+\alpha h),$ With fixed **x** and **h** From the Taylor

expansion (1.1) it follows that.

$$\varphi(\alpha) = f(x) + \alpha h^\top f'(x) + \tfrac{1}{2}\alpha^2 h^\top f''(x)h + O(\alpha^3) \,,$$

Where $\varphi'(0) = h^\top f'(x) \,.$

In Figure 2.2 we show an example of the variation of $\varphi(\alpha)$ with **h** as a

descent direction. The descending condition .tells us that we have to Stop the

line search with a value *a*s so that : $\varphi(\alpha_s) < \varphi(0).$ have $\varphi'(0) < 0,$ but the

figure shows that there is a risk that, if *a* Is taken too large, then $\varphi(\alpha) > \varphi(0).$

'On the other hand, we must also guard against the step being so short that our

gain in function value diminishes. α
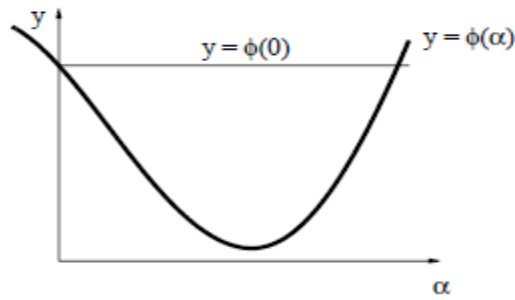
Figure 2.2: *Variation of the cost function along the search line.*

To ensure that we get a useful decrease in the *f*-value, we stop the search with a

value *a* s which gives a '-value below that of the line $: y = \lambda(\alpha),$ in Figure 2.3

below. This line goes through the starting point and has a slope which is a

fraction of the slope of the starting tangent to the φ-ı curve

$\varphi(\alpha_s) \leq \lambda(\alpha_s) ,$ where

$\lambda(\alpha) = \varphi(0) + \varrho \cdot \varphi'(0) \cdot \alpha$ with $0 < \varrho < 0.5$.   (2.10)

The parameter *e* is normally small, e.g. 0.001 Condition (2.10)) is needed in

some convergence proofs. We also want to ensure that the *a*-value is not chosen

too small. In Figure 2.3 we indicate a requirement, ensuring that the local slope is

greater than the starting slope. More specific,
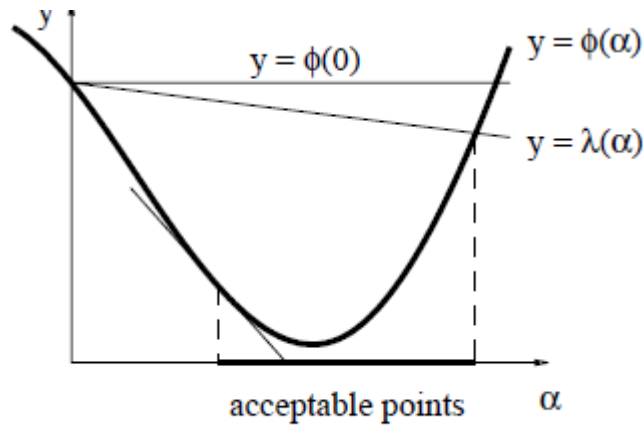
Figure 2.3: *Acceptable points according to*

Criteria (2.10) and (2.11). Descent methods with line search governed by

(2.10) and (2.11) are normally convergent. Fletcher has the proof of the following

theorem.

Theorem 2.2. Consider an absolute descent method following Algorithm 2.7

with search directions that satisfy Definition 2.13 and with line search controlled

by (2.10) and (2.11).

If $f'(x)$ exists and is uniformly continuous on the level set $\{x \mid f(x) < f(x_0)\}$, then for $k \to \infty$:

either $f'(x_k) = 0$ for some k ,

or $f(x_k) \to -\infty$,

or $f'(x_k) \to 0$.

A possible outcome is that the method finds a stationary point   (**x**k with **f**

'(Xk)=0) and then it stops. Another possibility is that $f(\mathbf{x})$ is not bounded from

below for **x** in the level set $\{x \mid f(x) < f(x_0)\}$ and the Method may converges

towards a stationary point. The method being a descent method often makes it

converge towards a point which is not only a stationary point but also a local

minimizer.   A line search as described above is often called a *soft line search*

because of its liberal stopping criteria, (2.10) and (2.11).   In contrast to this we

talk about *"Exact line search"* when we seek (an approximation to) a local

$$\alpha_e = \text{argmin}_{\alpha>0} f(x+\alpha h) \quad \text{for fixed } x \text{ and } h .$$
(2.12)

A necessary condition on α_e is $\varphi'(\alpha_e) = 0$. We have $\varphi'(\alpha) = h' f'(x+\alpha h)$ and this shows that either $f'(x+\alpha_e h) = 0$, which is a perfect result (we have found a stationary point for f), or if $f'(x+\alpha_e h) \neq 0$, then $\varphi'(\alpha_e) = 0$ is equivalent to

$$f'(x+\alpha_e h) \perp h .$$
(2.13)

this shows that the exact line search will stop at a point where the local gradient

is orthogonal to the search direction.

Example 2.4.

Orthogonality mentioned in (2.12). Figure 2.4: *An exact line search stops at*

y = **x** + *a e* **h**, *where the*

Local gradient is orthogonal to the search direction
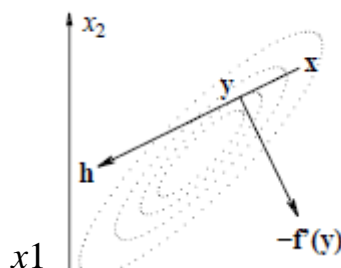
Figure 2.4

2.4. Descent Methods with Trust Region

the methods in this research series of steps leading from the starting   position to the final result, and in Newton's method of Chapter five, the directions of the steps are determined by the properties of $f(\mathbf{x})$ at the current position. Similar considerations lead us to the trust region methods, where the iteration steps are determined from the properties of a model of the objective function inside a given region. The size of the region is modified during the iteration. The Taylor expansion (1.3) provides us with a linear approximation to f near a given $\mathbf{x}$:

$$f(\mathbf{x}+\mathbf{h}) \simeq q(\mathbf{h}) \quad \text{with } q(\mathbf{h}) = f(\mathbf{x}) + \mathbf{h}^\top f'(\mathbf{x}) . \tag{2.14}$$

Likewise we can obtain a quadratic approximation to f from the Taylor expansion

$$f(\mathbf{x}+\mathbf{h}) \simeq q(\mathbf{h})$$
$$\text{with } q(\mathbf{h}) = f(\mathbf{x}) + \mathbf{h}^\top f'(\mathbf{x}) + \tfrac{1}{2}\mathbf{h}^\top f''(\mathbf{x})\mathbf{h} . \tag{2.15}$$

In both case q (**h**) is a good approximation to $f(\mathbf{x}+\mathbf{h})$ only if //**h**// is sufficiently small. These considerations lead us to determine the new iteration step as the solution to the following *model problem*:

$$\mathbf{h}_{tr} = \text{argmin}_{\mathbf{h}\in\mathcal{D}}\{q(\mathbf{h})\}$$
$$\text{where } \mathcal{D} = \{\mathbf{h} \mid \|\mathbf{h}\| \leq \Delta\}, \ \Delta > 0 .$$

(2.16)

The region ∇ is called the *trust region* and $q(\mathbf{h})$ We use $\mathbf{h} = \mathbf{h}_{tr}$ and reject **h**, if $f(\mathbf{x}+\mathbf{h}) \geq f(\mathbf{x})$. The gain in cost function value controls the size of the trust

region for the next step: The gain is compared to the gain predicted by the

approximation function, and we introduce the *gain facto*

$$r = \frac{f(x) - f(x+h)}{q(0) - q(h)} \quad .$$

(2.17)

When r is small our approximation agrees poorly with f, and when it is Large the

agreement is good. Thus, we let the gain factor regulate the size of the trust

region for the next step (or the next attempt for this step when $r <\cdot 0$ so that **h** is

rejected). These ideas are summarized in the following algorithm.

Algorithm 2.25. Descent method with trust region begin Given x0 and Δ 0

begin
    $k := 0; \quad x := x_0; \quad \Delta := \Delta_0; \quad found := $ **false**
    repeat
        $k := k+1; \quad h_{tr} := $ Solution of model problem *(2.16)*

        $r := $ gain facto   (2.17)

    if $r > 0.75$ {very good step}
        $\Delta := 2 * \Delta$ {larger trust region}
    if $r < 0.25$ {poor step}
        $\Delta := \Delta/3$ {smaller trust region}
    if $r > 0$ {reject step if $r \leq 0$}
        $x := x + h_{tr}$
    Update *found*   {stopping criteria, eg (2.8) and (2.9)}
  until *found* **or** $k > k_{max}$
end

the numbers in the algorithm, 0.75, 2, 0.25 and $P_1=3$ have been chosen from

practical experience. the method is not very sensitive to minor changes in these

values, but in the expressions $\Delta := P_1 * \Delta$ and $\Delta := \Delta /p2$ the numbers P_1 and

P_2 must be chosen so that the $\cent$-values cannot oscillate. There are versions of the

trust region method where "$r<0.25$" initiates an interpolation between **x** and **x+h**

based on known values of *f* and **f '**, and "$r>0.75$" leads to an extrapolation along

the direction **h**, a line search actually.

2.5. Soft Line Search

Many researchers in optimization have proved their inventiveness by producing

new line search methods or modifications to known methods. What we present

here are useful combinations of ideas of different origin. the description is based

on Madsen (1984){1}. In the early days of optimization exact line search was

dominant. Now, soft line search is used more and more, and we rarely see new

methods presented which require exact line search. an advantage of soft line

search over exact line search is that it is the faster of the two. If the first guess on

the step length is a rough approximation to the minimizer in the given direction,

the line search will terminate immediately if some mild criteria are satisfied. The

result of exact line search is normally a good approximation to the result, and this

can make descent methods with exact line search find the local minimizer in less

iteration than what is used by a descent method with soft line search. However, the extra function evaluations spent in each line search often makes the descent method with exact line search a loser. if we are at the start of the iteration with a descent method, where **x** is far From the solution **x***, it does not matter much that the result of the soft line search is only a rough approximation to the result; this is another point in favour of the soft line search the purpose of the algorithm is to find a s, an acceptable argument for the function

$$\varphi(\alpha_s) \leq \lambda(\alpha_s), \qquad \text{where}$$
$$\lambda(\alpha) = \varphi(0) + \varrho \cdot \varphi'(0) \cdot \alpha \quad \text{with } 0 < \varrho < 0.5 \qquad \textit{(2.18(a)}$$

$$\varphi'(\alpha_s) \geq \beta \cdot \varphi'(0) \quad \text{with } \varrho < \beta < 1. \qquad \text{(2.18(b)}$$

These two criteria express the demands that a s must be sufficiently small to give a useful decrease in the objective function, and sufficiently large To ensure that we have left the starting tangent of the curve $y = \varphi(\alpha)$ for $\alpha \geq 0$, see of Figure 2.3. The algorithm has two parts. First we find an interval $[a; b]$ that contain acceptable points, see figure 2.5. Figure 2.5: *Interval* $[a. b]$ *containing acceptable points.*
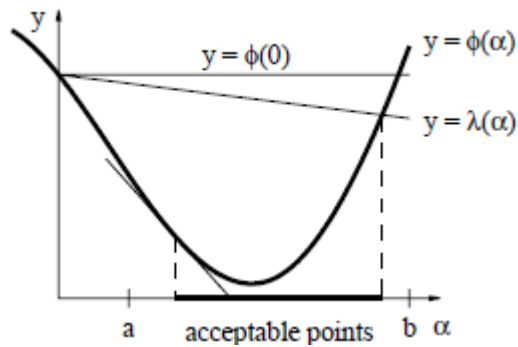
In the second part of the algorithm we successively reduce the interval: We find a point a in the strict interior of $[a , b]$. If both conditions are satisfied by this a-value, then we are finished $(as = a)$ otherwise, the reduced interval is either $[a, b]$:= $[a , a]$ or $[a, b] = [a, b]$, where the choice is made so that the reduced $[a, b]$ contains acceptable points. We have the following remarks to Algorithm 2.1 given below.

1- If **x** is a stationary point $\mathsf{f}'(\mathbf{x}) = 0 \Rightarrow \varphi'(0) = 0)$ or **h** is not downhill, then we do nothing.

2- The initial choice $b=1$ is used because in many optimization methods (e .g. Newton's method in Chapter 5) $a=1$ is a very good guess in the final steps of the iteration. The upper bound a max must be supplied by the user. It acts as a guard against an infinite loop if f is unbounded.

 3- We are to the left of a minimum and update the left hand end of the Interval Algorithm 2.3   soft line search

```
begin
    if φ'(0) ≥ 0                                                    {1°}
        α := 0
    else
        k := 0;  γ := β * φ'(0);
        a := 0;  b := min{1, α_max}                                 {2°}
        while (φ(b) ≤ λ(b)) and (φ'(b) ≤ γ)
              and (b < α_max) and (k < k_max)
            k := k + 1;  a := b                                      {3°}
            b := min{2b, α_max}                                      {4°}
        α := b                                                       {5°}
        while ((φ(α) > λ(α)) or (φ'(α) < γ)) and (k < k_max)
            k := k + 1
            Refine α and [a, b]                                      {6°}
        if φ(α) ≥ φ(0)                                               {7°}
            α := 0
end
```

4. If a max is sufficiently large, then the series of b-values is 1. 2. 4 8....

Corresponding to an " expansion factor" of 2. Other factors could be used.

5. Initialization for second part of the algorithm.

6. See Algorithm 2.4 below.

7. The algorithm may have stopped [a,b]normally, e.g. by exceeding the

permitted number k max of function evaluations. If the current value of a does

not decrease the objective function, then we return $a=0$, of 1. The refinement can

be made by the following Algorithm 2.5. the input is an interval [a, b] which we

know contains acceptable points, and the output is an a found by interpolation.

We want to be sure that the intervals have strictly decreasing widths, so we only

accept the new a if it is inside [$a + d, b\text{-}d$] , where $d = \frac{1}{10}(b - a).$ The a

splits $[a; b]$ into two subintervals, and we also return the subinterval which must

contain acceptable points.

Algorithm (2,5) Refine

```
begin
    D := b − a;   c := (φ(b) − φ(a) − D * φ'(a))/D²        {8°}
    if c > 0
        α := a − φ'(a)/(2c)
        α := min{max{α, a+0.1D}, b−0.1D}}                  {9°}
    else
        α := (a + b)/2
    if φ(α) < λ(α)                                          {10°}
        a := α
    else
        b := α
end
```

We have the following remarks to Algorithm 2.28:

8° The second order polynomial

$$\psi(t) = \varphi(a) + \varphi'(a) \cdot (t{-}a) + c \cdot (t{-}a)^2$$

satisfies $\psi(a) = \varphi(a)$, $\psi'(a) = \varphi'(a)$ and $\psi(b) = \varphi(b)$. If $c > 0$, then ψ has a minimum, and we let α be the minimizer. Otherwise we take α as the midpoint of $[a, b]$.

9° Ensure that α is in the middle 80% of the interval.

10° If $\varphi(\alpha)$ is sufficiently small, then the right hand part of $[a, b]$ contain points that satisfy both of the constraints (2.26). Otherwise, $[\alpha, b]$ is sure to contain acceptable points.

Finally, we give the following remarks about the implementation of the

algorithm. the function and slope values are computed as

$$\varphi(\alpha) = f(\mathbf{x}{+}\alpha\mathbf{h}), \qquad \varphi'(\alpha) = \mathbf{h}^\top \mathbf{f}'(\mathbf{x}{+}\alpha\mathbf{h}) .$$

the computation of f and $\mathbf{f}'$ is the "expensive" part of the line search. Therefore,

the function and slope values should be stored in auxiliary variables for use in

acceptance criteria and elsewhere, and the implementation should return the

value of the objective function and its gradient to the calling programmer, is a

descent method. they will be useful as starting function value and for the starting

slope in the next line search (the next iteration).

2.7. Exact Line Search

The older methods for line search produce a value of a s which is sufficiently

close to the true result, $\alpha_s \simeq \alpha_e$ with

$$\alpha_e \equiv \operatorname{argmin}_{\alpha \geq 0} \varphi(\alpha) .$$

The algorithm can be similar to the soft line search in Algorithm 2.27, except
that the refinement loop after remark 5° is changed to

$$\textbf{while} \left(|\varphi'(\alpha)| > \tau * |\varphi'(0)| \right)$$
$$\textbf{and} \left(b - a > \varepsilon \right) \textbf{and} \left(k < k_{\max} \right) \tag{2.29}$$

Here, E and t indicate the level of errors tolerated; both should be small,

Positive numbers

Example 2..5

  A "divine power" with a radar set follows the movements of

our wayward tourist. He has decided to continue in a given direction,

until his feet tell him that he starts to go uphill. The"divine power"

see that he stops where the given direction is tangent to a local contour.

This is equivalent to the Orthogonality formulated

Figure 2.6. An exact line search stops at $y = x+\alpha eh$, where the local gradient is

orthogonal to the search direction

Figure 2.7. An exact line search stops at $y = x+\alpha_e h$, where the local gradient is orthogonal to the search direction.

(2)

(1)

h

y

x

$-\nabla f(y)$

■

٣٩
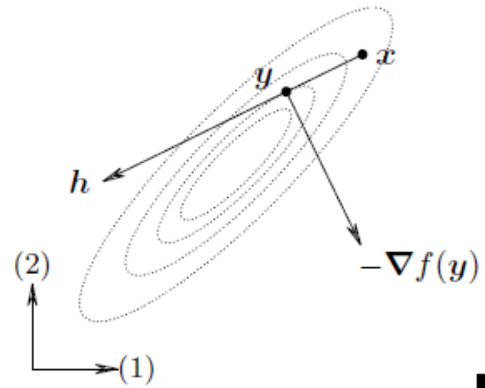
chapter three

3:THE STEEPEST DESCENT METHOD

Introduction :

Which of the possible descent directions (see Definition 2.1)

Our first considerations will be based purely on local first order information. Which descent direction gives us the greatest gain in function value relative to the step length? Using the first order Taylor expansion (1.3) we get the Following approximation   (1)

$$\frac{f(\mathbf{x}) - f(\mathbf{x} + \alpha\mathbf{h})}{\alpha\|\mathbf{h}\|} \simeq -\frac{\mathbf{h}^\top \mathbf{f}'(\mathbf{x})}{\|\mathbf{h}\|} = \|\mathbf{f}'(\mathbf{x})\| \cos\theta . \qquad (3.1)$$

In the last relation we have used the definition (2.12). We see that the relative gain is greatest when the angle $\theta = 0$, ie when $\mathbf{h} = \mathbf{h}_{sd}$, given by

$$\mathbf{h}_{sd} = -\mathbf{f}'(\mathbf{x}) . \qquad (3.2)$$

This search direction, the negative gradient direction, is called the direction of *steepest descent* is a useful gain in fun. It give sction value if the step is so short

that the 3rd term in the Taylor expansion $\left(O(\|\mathbf{h}\|^2)\right)$ is insignificant. the minimizer along the direction $\mathbf{h}_{sd}$. At the minimizer the higher order terms are large enough to have changed the slope from its negative starting value to zero. According to Theorem 2.2(a) descent method based on steepest descent and soft or exact line search is convergent. If we make a method using $\mathbf{h}_{sd}$. and a version of line search that ensures sufficiently short steps, then the global convergence will manifest itself as a very robust global performance. The disadvantage is that the method will have linear final convergence and this will often be exceedingly slow. If we use exact line search together with steepest descent,

Example 3.1.

We test a steepest descent method with exact line search on the function from

Example 1.2,

$$f(\mathbf{x}) = (x1+x2-2)2 + 100(x1 - x2)2 :$$

Figure 3.1 gives contours of this function.
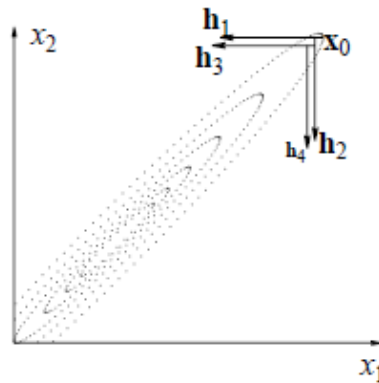
Figure 3.1: *The Steepest Descent Method fails to find the minimizer of a quadratic*

The gradient is

$$\mathbf{f}'(\mathbf{x}) = \left[\begin{array}{c} 2(x_1 + x_2 - 2) + 200(x_1 - x_2) \\ 2(x_1 + x_2 - 2) - 200(x_1 - x_2) \end{array} \right] .$$

If the starting point is taken as $\mathbf{x}_0 = [3, 598/202]^{\top}$, then the first search direction is

$$\mathbf{h}_{\mathrm{sd}} = - \left[\begin{array}{c} 3200/202 \\ 0 \end{array} \right] .$$

This is parallel to the $x1$-axis. The exact line search will stop at a point where the

gradient is orthogonal to this. Thus the next search direction will be parallel to

the X_2-axis, etc. The iteration steps will be exactly as in Example 1.2. The

iteration will stop far away from the solution because the steps become negligible

compared with the position, when represented in the computer with a finite

number of digits. this example shows how the final linear convergence of the

steepest descent method can become so slow that it makes the method completely

useless when we are near the solution. We say that the iteration is caught in(

Stiefel's cage).

The method is useful, however, when we are far from the solution. It performs a

little better if we ensure that the steps taken are small enough. In such a version

it is included in several modern hybrid methods, where there is a switch between

two methods, one with robust global performance and one with super linear (or

even quadratic) final convergence. Under these circumstances the method of

steepest descent does a very good job as the   "global part" of the hybrid. asin

Section 5.2.

Chapter four

---

4. CONJUGATE GRADIENT METHODS

Introduction:

. The conjugate gradient methods are simple and easy to implement, and

generally they are superior to the steepest descent method, but Newton's method and its relatives (asin the next chapter) are usually even better. If, however, the number n of variables is large, then the conjugate gradient methods may outperform Newton-type methods. the reason is that the latter rely on matrix operations, where as conjugate gradient methods only use vectors. Ignoring sparsity, Newton's method needs $O(n^3)$ operations per iteration step, Quasi-Newton methods need $O(n^2)$, but the conjugate gradient methods only use $O(n)$ operations per iteration step. Similarly for storage: Newton-type methods require an $n*n$ matrix to be stored, while conjugate gradient methods only need a few vectors. The basis for the methods presented in this chapter is the following definition, and the relevance for our problems is indicated in Example 4.1.

<u>Definition 4.1.</u> Conjugate directions. A set of directions corresponding to vectors $[h_1, h_2,\dots\dots\dots]$ is said to be *conjugate* with respect to a symmetric positive definite matrix $\mathbf{A}$, if $\qquad \mathbf{h}_i^\top \mathbf{A}\mathbf{h}_j = 0 \quad \text{for all } i \neq j .$ *(4.1)*

<u>Example 4.1.</u> In IR$_2$ we want to find the minimizer of a quadratic

$$q(\mathbf{x}) = a + \mathbf{b}^\top \mathbf{x} + \tfrac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} ,$$
(4.2)

where the matrix $\mathbf{H}$ is assumed to be positive definite. Figure 4.1 gives the contours of such a polynomial.
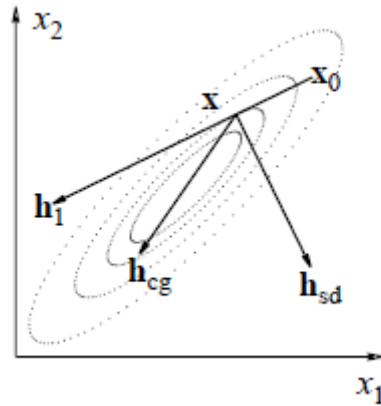
Figure 4.1: *In the 2-dimensiona lcase, the second conjugate gradient step determines the minimizer of a quadratic.*

assume that our first step was in the direction $\mathbf{h_1}$, a descent direction. Now we have reached position $\mathbf{x}$ after an exact line search. thus the direction $\mathbf{h_1}$ is tangent to the contour at $\mathbf{x}$. This means that $\mathbf{h_1}$ is orthogonal to the steepest descent direction $\mathbf{h_{sd}}$ at $\mathbf{x}$, i.e $\quad \mathbf{h_1^\top h_{sd}} = 0$:

$$\mathbf{h_1^\top}\left((-q'(\mathbf{x}))\right) = \mathbf{h_1^\top}\left(-\mathbf{b} - \mathbf{Hx}\right) = 0 \,.$$

Now, the minimizer satisfies $\mathbf{Hx^*} + \mathbf{b} = \mathbf{0}$, and inserting $\mathbf{b}$ from this we get

$$\mathbf{h_1^\top H}(\mathbf{x^*} - \mathbf{x}) = 0 \,.$$

This shows that if we are at $\mathbf{x}$ after an exact line search along a descent direction, $\mathbf{h}1$, then the direction $\mathbf{x^*}$-$\mathbf{x}$ to the minimizer is conjugate to $\mathbf{h_1}$ with respect to $\mathbf{H}$. We can further prove that the conjugate direction is a linear combination of the search direction $\mathbf{h_1}$ and the steepest descent direction, $\mathbf{h_{sd}}$, with positive coefficient i.e, it is in the angle between $\mathbf{h_1}$ and $\mathbf{h_{sd}}$. In the next sections we

discuss conjugate gradient methods which can find the minimizer of a second degree polynomial in n steps, where n is the dimension of the space.

4.1. Quadratic Models

An important tool for designing optimization methods is *quadratic modelling.the* function f is approximated locally with a quadratic function q of the form

$$q(\mathbf{x}) = a + \mathbf{b}^\top \mathbf{x} + \tfrac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x},$$

Where $\mathbf{H}$ is a symmetric matrix which is usually required to be positive definite. When the model ling is direct, we simply use the minimizer of q to approximate $\mathbf{x}*$ and then repeat the process with a new approximation. This is the basis of the Newton-type methods described in Chapter 5. For the conjugate gradient methods, the model function) will be employed indirectly. A related concept is that of *quadratic termination,* which is said to hold for methods that find the exact minimum of the quadratic in a finite number of steps. The steepest descent method does not have quadratic termination, but all the methods discussed in this chapter and the next do. Quadratic termination has proved to be an important idea and worth striving for in the design of optimization methods. Because of the importance of quadratic models we now take a closer look at the quadratic function It is not difficult to see that its gradient at $\mathbf{x}$ is given by

$$\mathbf{q}'(\mathbf{x}) = \mathbf{H}\mathbf{x} + \mathbf{b} \qquad\qquad (4.3)$$

And for all $\mathbf{x}$ the Hessian is

$$\mathbf{q}''(\mathbf{x}) = \mathbf{H}. \qquad\qquad (4.4)$$

If **H** is positive definite, then q has a unique minimizer, $\mathbf{x}^* = -\mathbf{H}\ \mathbf{b}$. If

$n=2$, then the contours of q are ellipses centered at $\mathbf{x}^*$. The shape and orientation

of the ellipses are determined by the eigenvalues and eigenvectors of **H**. For $n=3$

this generalizes to ellipsoids, and in higher dimensions we get (n-1)-dimensional

hyper-ellipsoids. It is of course possible to define quadratic functions with a non-

positive definite Hessian, but then there is no longer a unique minimizer.

Finally, a useful fact is derived in a simple way from multiplication by **H** maps

differences in **x**-values to differences in the corresponding gradients:

$$\mathbf{H}(\mathbf{x} - \mathbf{z}) = \mathbf{q}'(\mathbf{x}) - \mathbf{q}'(\mathbf{z}) \qquad\qquad (4.5)$$

4.2. Structure of a Conjugate Gradient Method

let us have another look at Figure 3.1 where the slow convergence of the

steepest descent method is demonstrated. an idea for a possible cure is to

take a linear combination of the previous search direction and the current

steepest descent direction to get a direction toward the solution. this gives

a method of the following type

Algorithm 4.1 conjugate gradient method

```
begin
    x := x₀;  k := 0;  found := false;  γ := 0;  h_cg := 0          {1°}
    repeat
        h_prev := h_cg;  h_cg := −f'(x) + γ * h_prev
        if  f'(x)ᵀ h_cg ≥ 0                                          {2°}
            h_cg := −f'(x)
        α := line_search(x, h_cg);  x := x + αh_cg                   {3°}
        γ := · · ·                                                   {4°}
        k := k+1;  found := · · ·                                    {5°}
    until found or k > k_max
end
```

We have the following remarks:

1- Initialization.

2- In most cases the vector $\mathbf{h}_{cg}$ is downhill. This is not guaranteed, e.g. if we use a soft line search, so we use this modification to ensure that each step is downhill.

3- New iterate.

4- The formula for r is characteristic for the method. This is discussed in the next sections.

5- We recommend to stop if one of the criteria

$$\|f'(x)\|_\infty \leq \varepsilon_1 \quad \text{or} \quad \|\alpha \mathbf{h}_{cg}\|_2 \leq \varepsilon_2(\varepsilon_2 + \|x\|_2) \qquad (4.7)$$

is satisfied, of (2.8) and (2.9).

in the next theorem we show that a method employing conjugate search directions and exact line searches is very good for minimizing quadratics.

In Theorem 4.12 (in Section 4.3) we show that, if f is quadratic and the line searches are exact, then a proper choice of r gives conjugate search

Directions.

Theorem 4.1. Use Algorithm 4.6 with exact line search on a quadratic

like (4.2) with $\mathbf{x}$ in $\mathbb{R}n$, and with the iteration steps $\mathbf{h}i = \mathbf{x}i - \mathbf{x}i\text{-}1$

Corresponding to conjugate directions. Then

1- The search directions $\mathbf{h}_{cg}$ are downhill.

2- The local gradient $\mathbf{f'}$ $(\mathbf{x}k)$ is orthogonal to $\mathbf{h}_1$... $\mathbf{h}_2$, ...,$\mathbf{h}_k$.

3- The algorithm terminates after at most n steps.

Proof. We examine the inner product in Definition 2.11 and insert the expression

for $\mathbf{h}_{cg}$

$$\mathbf{f}'(\mathbf{x})^\top \mathbf{h}_{cg} = -\mathbf{f}'(\mathbf{x})^\top \mathbf{f}'(\mathbf{x}) + \gamma \mathbf{f}'(\mathbf{x})^\top \mathbf{h}_{\text{prev}}$$
$$= -\|\mathbf{f}'(\mathbf{x})\|_2^2 \le 0 .$$

The second term in the first line is zero for any choice of γ since an Exact line

search terminates when the local gradient is orthogonal to the Search direction.

thus, $\mathbf{h}_{cg}$ is downhill (unless $\mathbf{x}$ is a stationary point, Where $\mathbf{f'}$ $(\mathbf{x}) = \mathbf{0}$), and we

have proved 1. Next, the exact line searches guarantee that

$$\mathbf{h}_i^\top \mathbf{f}'(\mathbf{x}_i) = 0, \quad i = 1, \ldots, k \tag{4.10}$$

and by means of (4.5) we see that for $j < k$,
$$\mathbf{h}_j^\top \mathbf{f}'(\mathbf{x}_k) = \mathbf{h}_j^\top \left(\mathbf{f}'(\mathbf{x}_j) + \mathbf{f}'(\mathbf{x}_k) - \mathbf{f}'(\mathbf{x}_j) \right)$$
$$= 0 + \mathbf{h}_j^\top \mathbf{H}(\mathbf{x}_k - \mathbf{x}_j)$$
$$= \mathbf{h}_j^\top \mathbf{H}(\mathbf{h}_k + \ldots + \mathbf{h}_{j+1}) = 0 .$$

Here, we have exploited that the directions *(h_i)* are conjugate with

respect to **H**, and we have proven 2 .Finally, **H** is non-singular, and it is easy to show that this implies that a set of conjugate vectors is linearly independent. Therefore {h_1,......., h_n} span the entire IRn, and **f'** (X_n) must be zero., we remark that if **f** '(X_k)=**0** for some $k \cdot n$, then the solution has been Found and Algorithm 4.6 stops. What remains is to find a clever way to determine $\mathbf{C}^\top \mathbf{x} = \mathbf{z}$ The approach used is to determine r in such a way that the resulting method will work well for minimizing quadratic functions. Taylor's formula shows that smooth functions are locally well approximated by quadratics, and therefore the method can be expected also to work well on more general functions.

4.3. The Fletcher–Reeves Method

The following formula for r was the first one to be suggested:

$$\gamma = \frac{\mathbf{f}'(\mathbf{x})^\top \mathbf{f}'(\mathbf{x})}{\mathbf{f}'(\mathbf{x}_{\text{prev}})^\top \mathbf{f}'(\mathbf{x}_{\text{prev}})},$$

(4.11)

Where **X** $_{prev}$ is the previous iterate. Algorithm 4.6 with this choice for r is called the(*Fletcher–Reeves method*) after the people who invented it in 1964.

Theorem 4.2. Apply the Fletcher–Reeves method with exact line Search

   To the quadratic function (4.2). If **f'** (X_k) ≠**0** for k=1............ n,

   Then the search directions h_1, ...,h $_n$ are conjugate with respect to **H**.

Proof.

We shall use induction to show that for $j = 1, \ldots \ldots , n$:

$$h_i^\top H h_j = 0 \quad \text{for all } i < j. \qquad : \text{(B.1)}$$

We use the notation

$g_i = \mathbf{f}'(\mathbf{x}i)$ and define the search directions by

$$h_i = x_i - x_{i-1}. \qquad \text{Then (4.5) leads to}$$

$$H h_r = g_r - g_{r-1}, \quad \text{(B.2)}$$

and Algorithm 4.6 and (4.11) combine to

$$h_{r+1} = \alpha_{r+1} \left(-g_r + \gamma_r \alpha_r^{-1} h_r \right) \quad \text{with} \quad \gamma_r = \frac{g_r^\top g_r}{g_{r-1}^\top g_{r-1}}, \quad ; \text{(B.3)}$$

and a_{r+1} found by exact line search. Finally, we remind the reader of (4.10) and

(4.9)

$$h_r^\top g_r = 0 \quad \text{and} \quad \alpha_{r+1}^{-1} h_{r+1}^\top g_r = -g_r^\top g_r \quad \because \text{(B.4)}$$

Now, we are ready for the induction: For $j=1$, (B.1) is trivially satisfied, there is

no $\mathbf{h}i$ vector with $i < 1$.

Next, assume that (B.1) holds for all $j = 1, \ldots \ldots , k$. Then it follows from the proof

of Theorem 4.8 that

$$g_k^\top h_i = 0 \quad \text{for } i = 1, \ldots , k \quad \because \text{(B.5)}$$

If we insert (B.3), we see that this implies

$$0 = g_k^\top \left(-g_{i-1} + \gamma_{i-1} \alpha_{i-1}^{-1} h_{i-1} \right) = -g_k^\top g_{i-1}$$

Thus, the gradients at the iterates are orthogonal,

$$g_k^\top g_i = 0 \quad \text{for } i = 1, \ldots, k-1 .$$
$$: (B.6)$$

Now, we will show that (B.1) also holds for $j = k+1$:

$$\alpha_{k+1}^{-1} h_i^\top H h_{k+1} = h_i^\top H \left(-g_k + \gamma_k \alpha_k^{-1} h_k \right)$$
$$= -g_k^\top H h_i + \gamma_k \alpha_k^{-1} h_i^\top H h_k$$
$$= -g_k^\top (g_i - g_{i-1}) + \gamma_k \alpha_k^{-1} h_i^\top H h_k .$$
For $i<k$ each term is

zero according to (B.1) for j *let than and quake* k and (B.5).

For $i < k$ each term is zero according to (B.1) for $j \le k$ and (B.5).
For $i = k$ also the term $g_k^\top g_{k-1} = 0$, and we get

$$\alpha_{k+1}^{-1} h_k^\top H h_{k+1} = -g_k^\top g_k + \gamma_k \alpha_k^{-1} h_k^\top (g_k - g_{k-1})$$
$$= -g_k^\top g_k + \gamma_k \left(0 + g_{k-1}^\top g_{k-1} \right) = 0 .$$

In the first reformulation we use both relations in (B.4), and next we use the

definition of γ_k in (B.3). Thus, we have shown that (B.1) also holds for $j = k+1$

and thereby finished the proof .B .According to Theorem 4.8 this implies that the

Fletcher–Reeves method With exact line search used on a quadratic will

terminate in at most n steps. Point 1 in Theorem 4.8 shows that a conjugate

gradient method with exact line search produces descent directions.

4.4. The Polak–Ribi`ere method

An alternative formula for r is

$$\gamma = \frac{(f'(x) - f'(x_{prev}))^\top f'(x)}{f'(x_{prev})^\top f'(x_{prev})} ,$$

(4.13)

Algorithm 4.6 with this choice of r is called the *Polak–Ribi`ere method*. It dates

from 1971 (and again it is named after the inventors). For a quadratic (4.13) is

equivalent to (4.11)   (Because then $f'(x_{prev})^\top f'(x) = 0$,

Example 4.2. (Resetting).

A possibility that has been proposed is to reset the search direction **h** to the

steepest descent direction **h**sd in every nth iteration. the rationale behind this is

the n-step quadratic termination property. If we enter a Neighbourhood of the

solution where f behaves like a quadratic, resetting will ensure quick

convergence. Another apparent advantage of resetting is that it will guarantee

global convergence (by Theorem 2.18). However, practical experience has shown

that the profit of resetting is doubtful. In connection with this we remark that the

Polak–Ribi`ere method has a kind of resetting built in. Should we encounter a

step with very little progress, so that $[[$**x**- **x**prev$]]$is small compared with$[[$**f'**(**x**prev)

$[[$**f'**(**x**prev) $]]$, then$[[$**f'**(**x**) $-$ **f'**(**x**prev)$]]$will also be small and therefore r is small,

and **h**cg $'$ **h**sd in this situation. Also, the modification before the line search in

Algorithm 4.6 may result in an occasional resetting. the method fails to

converge even with exact line search (see p 213 in Nocedal(1992)). The success

of the Polak–Ribi`ere   formula is there fore not so easily explained by theory{1}.

4.5. Convergence Properties

we saw that the search directions **hcg** of a conjugate gradient method are descent directions and thus the μ of (2. 2) satisfies $\mu < \pi / 2$. There is no guarantee, however, that the μ of Definition 2.1will stay constant, and theorem 2.1 is therefore not directly applicable .For many years it was believed that to guarantee convergence of a conjugate gradient method it would be necessary to use a complicated line search, and perhaps make some other changes to the method. But in 1985 Al- Baali managed to prove global convergence using a traditional soft line search. (2){1}

Theorem 4.3. Let the line search used in Algorithm 4.6 satisfy (2.10) and (2.11) with parameter values $Q < B < 0.5$. Then there is a $c > 0$ Such that for all k

$$\mathbf{f}'(\mathbf{x})^\top \mathbf{h_{cg}} \leq -c \, \|\mathbf{f}'(\mathbf{x})\|_2^2 \text{ and}$$
$$\lim_{k \to \infty} \|\mathbf{f}'(\mathbf{x})\|_2 = 0 .$$

:Let us finally remark on the rate of convergence. Crowder and Wolfe (1972) show that conjugate gradient methods with exact line search have a linear convergence rate, as defined in (2.4). This should be contrasted with the super linear convergence rate that holds for Quasi-Newton methods and the quadratic convergence rate that Newton's method possesses (3)

Example 4.3.   (*Rosen Brock's function,*)

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,$$ is widely used for testing optimization algorithms. Figure 4.2 shows level curves for this function (and illustrates, why it is sometimes called the *"banana function"*).
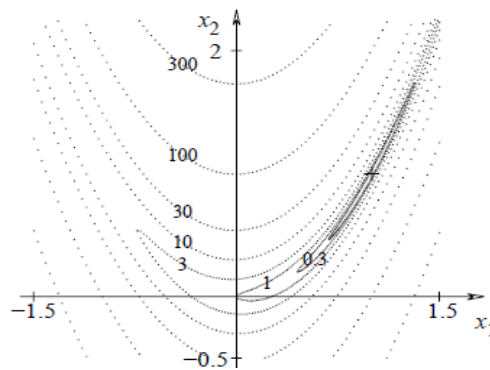
Figure 4.2: *Contours of Rosen Brock's function.*

the function has one   minimizer $\mathbf{x}_0 = [\text{-}0.\ 0\ ]\ \mathrm{T}$ $\mathbf{x}^* = [1,\ 1]^\mathsf{T}$ (marked by a + in the figure) with $f(\mathbf{x}^*)=0$, and there is a "valley" with sloping bottom following the parabola $x_2 = x_1^2$. Most optimization algorithms will try to follow this valley. Thus, a considerable amount of iteration steps is needed, if we take $\mathbf{X}_0$ in the 2nd quadrant. Below we give the number of iteration steps and evaluations of $f(\mathbf{x})$ and $\mathbf{f}'(\mathbf{x})$ when applying Algorithm 4.6 on this function. In all cases we use the starting point $\mathbf{x}_0 = [-1.2,\ 1]'$,

, and stopping criteria given by $"\varepsilon_1 = 10^{-8}, \varepsilon_2 = 10^{-12}$ (4.7). In case of exact line search we use $\tau = 10^{-6}, \varepsilon = 10^{-6}$ in (2.29), while we take $\beta = 10^{-1}, \rho = 10^{-2}$

in Algorithm 2.1 for soft line search. $\mathbf{x}_0 = [-0.\ 0\ ]$ T

Method	Line search	# it. steps	# fct. evals
Fletcher–Reeves	exact	118	1429
Fletcher–Reeves	soft	249	628
Polak–Ribière	exact	24	266
Polak–Ribière	soft	45	130

Thus, in this case the Polak–Ribi`ere method with soft line search performs best.

Below we give the iterates (of. Figure 4.2) and the values of $f(\mathbf{X}_k)$ and

$\|\mathbf{f}'(\mathbf{x}_k)\|_\infty$: note the logarithmic ordinate axis.

Figure 4.3: *Polak–Ribière method with soft line search*
applied to Rosenbrock's function.
Top: iterates $\mathbf{x}_k$. *Bottom:* $f(\mathbf{x}_k)$ *and* $\|\mathbf{f}'(\mathbf{x}_k)\|_\infty$.

4.6. Implementation

To implement a conjugate gradient algorithm in a computer program, some

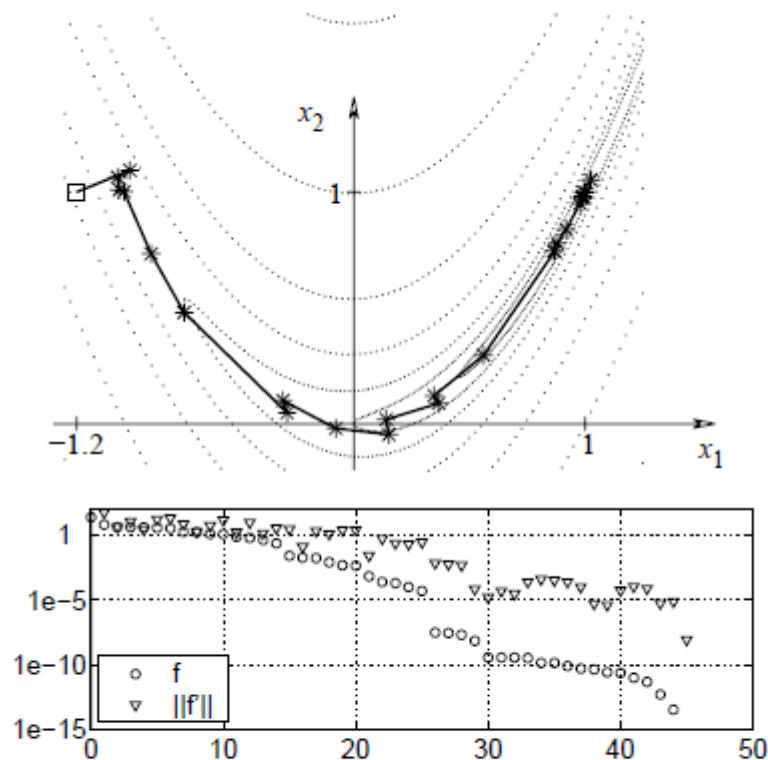decisions must be made. Of course we need to choose a formula for r; we recommend the Polak–Ribi`ere formula. We also need to specify the exactness of the line search. For Newton-type methods it is usually recommended that the line search be quite liberal, so for the line search in Algorithm 2.1 it is common to choose the parameter values $Q = 0.01$ and $B = 0.9$. For conjugate gradient methods experience dictates that a line search with stricter tolerances be used, say $Q = 0.01$ and $B = 0.1$. In addition we have to specify the stopping criterion; (2.9) is recommended .Since we do not have access to $\mathbf{f}''(\mathbf{X}_k)$, we cannot use (2.10). For methods with a fast convergence rate, (2.8) may be quite satisfactory, but its use for conjugate gradient methods must be discouraged because their final convergence rate is only linear. Finally some remarks on the storage of vectors. The Fletcher–Reeves method may be implemented using three n-vectors of storage, x, g and h. if these contain $\mathbf{x}$, $\mathbf{f}'(\mathbf{x})$ and $\mathbf{h}$ prev at the beginning of the current iteration step, we may overwrite h with $\mathbf{h}_{cg}$ and during the line search we overwrite x with $x + \alpha \mathbf{h}_{cg}$ and g with $f'(x + \alpha \mathbf{h}_{cg})$ Before overwriting the gradient, we find $f'(x)^{\top} f'(x)$ or use in the denominator in (4.11) on the next iteration. For the Polak–Ribi`ere method we need acces to $\mathbf{f}'(\mathbf{x})$ and $\mathbf{f}'(\mathbf{X}_{prev})$ simultaneously, and thus four vectors are required, say x, g, g_{new} and h

{1}{2}

4.7. The CG Method for Linear Systems

We cannot part with conjugate gradient methods without mentioning that they can of course be used to minimize the quadratic function (4.2) itself. But by (4.3) this is equivalent to solving the positive definite linear system

$\mathbf{H}_x = \mathbf{-b}$: Let $\mathbf{g}$ denote the current gradient,   (1)

$\mathbf{g} = \mathbf{q}'(\mathbf{x}) = \mathbf{H}_x + \mathbf{b}$;

and let $\mathbf{u} = \mathbf{Hh}_{cg}$. It is easily verified that the exact step length a may be

calculated directly, $\alpha \; = \; \dfrac{-\mathbf{u}^{\top}\mathbf{g}}{\mathbf{u}^{\top}\mathbf{h}_{cg}}$,   and that $\mathbf{x}$ and $\mathbf{g}$ are updated by

$\mathbf{x} := \mathbf{x} + a\,\mathbf{h}_{cg}, \qquad \mathbf{g} := \mathbf{g} + a\,\mathbf{u}$:

The Fletcher–Reeves and the Polak–Ribi`ere formulas are equivalent in this

setting, $\gamma \; = \; \dfrac{\mathbf{g}^{\top}\mathbf{g}}{\mathbf{g}_{prev}^{\top}\mathbf{g}_{prev}}$.   thus, the method can be implemented using four n-

vectors, $\mathbf{x}$, $\mathbf{g}$, $\mathbf{h}$, $\mathbf{u}$. The method is called the *conjugate gradient method for linear systems.* The method is specially useful when the matrix $\mathbf{H}$ is large and sparse.

Chapter   five

5. NEWTON-TYPE METHODS:

Introduction :

we consider a class of methods for unconstrained optimization which are based on Newton's method. This class is called Quasi- Newton methods. In order to explain these methods we first describe Newton's method for unconstrained optimization in detail. Newton's method leads to another kind of methods known as Damped Newton Methods, which will also be presented. Finally we get to the Quasi-Newton methods. This class includes some of the best methods for solving the unconstrained optimization problems.

5.1. Newton's Method {1}

Newton's method forms the basis of all Quasi-Newton methods. It is widely used for solving systems of non-linear equations, and until recently it was also widely used for solving unconstrained optimization problems. As it will appear, the two problems are closely related.

Example 5.1.

In Example 1.2 we saw the method of alternating directions fail to find the

minimizer of a simple quadratic in two dimensions and in Example 3.1

we saw the steepest descent method fail on the same quadratic. In Chapter 4 we

saw that the conjugate gradient methods finds the minimizer of a quadratic in n

steps (n being the dimension of the space), in two steps in Example 4.1.

Newton's method can find the minimizer of a quadratic in n-dimensional space in

one step. This follows from equation (5.2) below. Figure 5.1 gives the contours

of our 2-dimensional quadratic together with (an arbitrary) **X₀**. **X₁** and the

minimizer **x*** marked by *
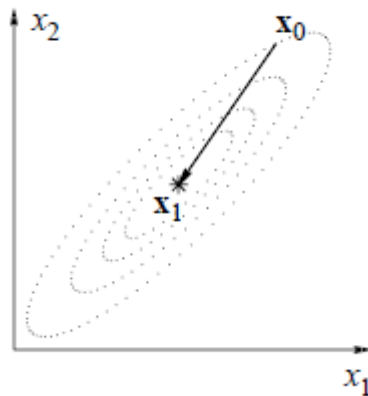
Figure 5.1: *Newton's method finds*

The minimizer of a quadratic in

The very first step

in order to derive *Newton's method* in the version used in optimization, we shall

once again consider the truncated Taylor expansion of the cost function at the current iterate **x**,

$$f(\mathbf{x} + \mathbf{h}) \simeq q(\mathbf{h}),$$

(5.1(a)

Where $q(\mathbf{h})$ is the quadratic model of f in the vicinity of **x**,

$$q(\mathbf{h}) = f(\mathbf{x}) + \mathbf{h}^\top \mathbf{f}'(\mathbf{x}) + \tfrac{1}{2}\mathbf{h}^\top \mathbf{f}''(\mathbf{x})\mathbf{h}.$$

(5.1(b)

The idea now is to minimize the model q at the current iterate. If $\mathbf{f}''(\mathbf{x})$ is Positive definite, then q has a unique minimizer at a point where the gradient of q equals zero, i.e. where

$$\mathbf{f}'(\mathbf{x}) + \mathbf{f}''(\mathbf{x})\mathbf{h} = \mathbf{0} \qquad\qquad : (5.2)$$

Hence, in Newton's method the new iteration step is obtained as the solution to the system (5.2) as shown in the following algorithm.

Algorithm 5.1 Newton's method

begin

$\mathbf{X} := \mathbf{X}_0;$ *(Initialisation)*

repeat

Solve $\mathbf{f}''(\mathbf{x})\,\mathbf{h}_n = -\mathbf{f}'(\mathbf{x})$ *(find step)*

$\quad \mathbf{x} := \mathbf{x} + \mathbf{h}_n$ *(. . . and next iterate)*

until stopping criteria satisfied

end

Newton's method is well defined as long as $\mathbf{f}''(\mathbf{x})$ remains non-singular.

Also, if the Hessian is positive definite, then it follows from Theorem 2.1 that $\mathbf{h}_n$

٦٣

is downhill. Further, if $f''(\mathbf{x})$ stays positive definite in all the steps and if the starting point is sufficiently close to a minimizer, then the method usually converges rapidly towards such a solution. more precisely the following theorem holds.

Theorem 5.1 If an iterate $\mathbf{x}$ is sufficiently close to a local minimizer $\mathbf{X}^*$ and $f''(\mathbf{x}^*)$ is positive definite, then Newton's method is well defined in all the following steps, and it converges quadratic cally towards $\mathbf{x}^*$

Example 5.2

. We shall use Newton's method to find the minimizer of the following function

$$f(\mathbf{x}) = 0.5 * x_1^2 * (x_1^2/6 + 1)$$
$$+ x_2 * \text{Arctan}(x_2) - 0.5 * \ln(x_2^2 + 1). \tag{5.5}$$

We need the derivatives of first and second order for this function:

$$f'(\mathbf{x}) = \begin{bmatrix} x_1^3/3 + x_1 \\ \text{Arctan}(x_2) \end{bmatrix}, \qquad f''(\mathbf{x}) = \begin{bmatrix} x_1^2 + 1 & 0 \\ 0 & 1/(1 + x_2^2) \end{bmatrix}.$$

We can see in Figure 5.2 that in a region around the minimizer the function looks very well-behaved and extremely simple to minimize.

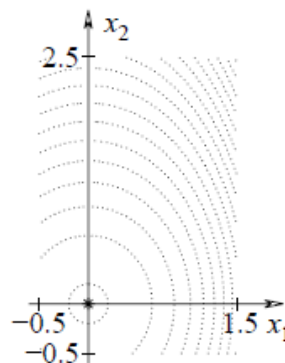Figure 5.2: *Contours of the function (5.5). The level curves are symmetric across both axes.*

Table 5.1 gives results of the iterations with the starting point $\mathbf{x}_0^T = [1, 0.7]$. $\mathbf{x}_0 = [-0. \ 0 \ ]$ T According to Theorem 5.4 we expect quadratic convergence. If the

factor C_2 in (2.5) is of the order of magnitude 1, then the column of $x_k^\top$ would

show the number of correct digits doubled in each iteration step, and the f-values

and step lengths would be squared in each iteration step. the convergence is faster

than this; actually for any starting point $x_0^\top = [u,v]$ with $[v] < 1$ we will get

cubic convergence; see the end of the next example.

k	$x_k^\top$		f	$\|f'\|$	$\|h_k\|$
0	[1.0000000000,	0.7000000000]	8.11e-01	1.47e+00	
1	[0.3333333333,	-0.2099816869]	7.85e-02	4.03e-01	1.13e+00
2	[0.0222222222,	0.0061189580]	2.66e-04	2.31e-02	3.79e-01
3	[0.0000073123,	-0.0000001527]	2.67e-11	7.31e-06	2.30e-02
4	[0.0000000000,	0.0000000000]	3.40e-32	2.61e-16	7.31e-06
5	[0.0000000000,	0.0000000000]	0.00e+00	0.00e+00	2.61e-16

Table 5.1: *Newton's method on (5.5).* $x_0^\top = [1, 0.7].$ ■

Until now, everything said about Newton's method is very promising: It

Is simple and if the conditions of Theorem 5.4 are satisfied, then the rate

of convergence is excellent. Nevertheless, due to a series of drawbacks the basic

version of the method is not suitable for a general purpose optimization

algorithm. The first and by far the most severe drawback is the method's lack of

global convergence.

Example 5.3.

With the starting point $x_0^\top = [1, 2]$ the Newton method behaves very badly:

k	$\mathbf{x}_k^\top$	f	$\|\mathbf{f}'\|$	$\|\mathbf{h}_k\|$
0	[1.0000000000, 2.0000000000]	1.99e+00	1.73e+00	
1	[0.3333333333, -3.5357435890]	3.33e+00	1.34e+00	5.58e+00
2	[0.0222222222, 13.9509590869]	1.83e+01	1.50e+00	1.75e+01
3	[0.0000073123, -2.793441e+02]	4.32e+02	1.57e+00	2.93e+02
4	[0.0000000000, 1.220170e+05]	1.92e+05	1.57e+00	1.22e+05
5	[0.0000000000, -2.338600e+10]	3.67e+10	1.57e+00	2.34e+10

Table 5.2: *Newton's method on (5.5).* $\mathbf{x}_0^\top = [1,\ 2]$.

$\mathbf{x}_0 = [-0.\ 0\ ]$ T Clearly, the sequence of iterates moves rapidly away from the solution (the firstComponent converges, but the second increases in size with alternating sign) even though $\mathbf{f}''(\mathbf{x})$ is positive definite for any $\mathbf{X}_2$ in R_2. The   for Arc tan $(0+h)$ is

$$\text{Arctan}(0+h) = \begin{cases} h - \frac{1}{3}h^3 + \frac{1}{5}h^5 - \frac{1}{7}h^7 + \cdots & \text{for } |h| < 1 \\ \text{sign}(h)\left(\frac{\pi}{2} - \frac{1}{h} + \frac{1}{3h^3} - \frac{1}{5h^5} + \cdots\right) & \text{for } |h| > 1. \end{cases}$$

The next point to discuss is that $\mathbf{f}''(\mathbf{x})$ may not be positive definite when $\mathbf{x}$ is far from the solution. In this case the sequence may be heading towards a saddle point or a maximizer since the iteration is identical to the one used for solving the non-linear system of equations $\mathbf{f}'(\mathbf{x}) = \mathbf{0}$. Any stationary point of f is a solution to this system. Also, $\mathbf{f}''(\mathbf{x})$ may be ill-conditioned or singular so that the linear system (5.2) cannot be solved without considerable errors in $\mathbf{h}_n$. Such ill-conditioning may be detected by a well designed matrix factorization (e.g. a Cholesky factorization)but it still leaves the question of what to do in case ill-conditioning occurs. The final major drawback is of a more practical nature but basically just as severe as the ones already discussed. Algorithm 5.3 requires the

analytic second order derivatives. These may be difficult to determine even though they are known to exist. Further, in case they can be obtained, users tend to make erroneous implementations of the derivatives . Also, in large scale problems the calculation of the Hessian may be costly since $\cdot\frac{1}{2}n(n+1)$ function evaluations are needed. Below, we summarize the advantages and disadvantages of Newton's method discussed above. They are the key to the development of more useful algorithms, since they point out properties to be retained and areas where improvements and modifications are required.

{Table (5.3)} **Advantages and disadvantages of Newton's method for unconstrained**

optimization problems

1: Advantages

1- Quadratic ally convergent from a good starting point if **f** ''(**x***) is

Positive definite.

2- Simple and easy to implement.

2: Disadvantages

1- Not globally convergent for many problems.

2- May converge towards a maximum or saddle point of *f*.

3- The system of linear equations to be solved in each iteration may be ill-conditioned or singular.

4- Requires analytic second order derivatives of *f.*   Table {5.3: }*Pros and Cons of Newton's Method.*

5.2. Damped Newton Methods:{1}

Although disadvantage 4 in Table 5.3 often makes it impossible to use any of the modified versions of Newton's method, Further, in case second order derivatives are obtainable, modified Newton methods may be used successfully. Hence, for the methods discussed in this subsection it is still assumed, that second order analytic derivatives of *f* are available. The more efficient modified Newton methods are constructed as either explicit or implicit hybrids between the original Newton method and the method of steepest descent. The idea is that the algorithm in some way should take advantage of the safe, global convergence properties of the steepest descent method whenever Newton's method gets into trouble. On the other hand the quadratic convergence of Newton's method should be obtained when the iterates get close enough to **x***, provided that the Hessian is positive definite. The first modification that comes to mind is a Newton method with line Search in which the Newton step $h_n = -[f''(x)]^{-1} f'(x)$ is used as a search Direction. Such a method is obtained if the step **x** := **x**+ **h**ₙ in Algorithm 5.3 is substituted by *a* := line search(**x***;* **h**ₙ); **x** := **x** + *a* **h**ₙ*:* (5.6) This will work fine as long as **f** *″*(**x**) is positive definite since in this case **h**ₙ is a descent

direction. The main difficulty thus arises when $\mathbf{f}''(\mathbf{x})$ is not positive definite. The Newton step can still be computed if $\mathbf{f}''(\mathbf{x})$ is non-singular, and one may search along $\pm\ \mathbf{h_n}$ where the sign is chosen in each iteration to ensure a descent direction. However, this rather primitive approach is questionable since the quadratic model $q(\mathbf{h})$ will not even possess a unique minimum. A number of methods have been proposed to overcome this problem. We Shall concentrate on the so-called damped Newton methods, which are considered to be the most successful in general. The framework for this class of methods is

Algorithm 5.2 Damped Newton step

Slove $\{f''(x)+\mu I)\ \mathbf{h_{dn}} = -f'(x)$ $(\mu>0)$

$X := x + x\ a\ \mathbf{h_{dn}}$ $(a>0)$

Adjust μ

Instead of finding the step as a stationary point of the quadratic (5.1b), the step $\mathbf{h_{dn}}$ is found as a stationary point of

$$
\begin{aligned}
q_\mu(\mathbf{h}) &= q(\mathbf{h}) + \tfrac{1}{2}\mu\mathbf{h}^\top\mathbf{h} \\
&= f(\mathbf{x}) + \mathbf{h}^\top\mathbf{f}'(\mathbf{x}) + \tfrac{1}{2}\mathbf{h}^\top(\mathbf{f}''(\mathbf{x})+\mu\mathbf{I})\mathbf{h}\,.
\end{aligned}
\qquad (5.8)
$$

($\mathbf{I}$ is the identity matrix). $\mathbf{f}''(\mathbf{x}) + \mu\ \mathbf{I}$ is positive definite. Thus, if μ is sufficiently large, then $\mathbf{h_{dn}}$ is not only a stationary point, but Also a minimizer for $q\ \mu$ Further, Theorem 2.14 guarantees that $\mathbf{h_{dn}}$ is a Descent direction for f at $\mathbf{x}$.

the term $\frac{1}{2}\mu h^\top h$ penalizes large steps, an from the definition in Algorithm 5.7 we

see that if μ is very large, then we get $h_{dn} \simeq (-1/\mu)f'(x)$ (5.10),.

if the current **x** is far from the minimizer **x***. On the other hand, if μ is small,

then $h_{dn} \simeq h_n$, the Newton step, which is good when we are close to **x*** (where **f**

"(**x**) is positive definite). Thus, by proper adjustment of the damping parameter μ

we   have a method that combines the good qualities of the steepest descent

method in the global part of the iteration process with the fast ultimate

convergence of Newton's method

Theorem 5.2. If the matrix **f** "(**x**)+ μ **I** is positive definite, then h_{dn} = argmin

//h//•// [[h_{dn}]] (q(h)) ,Where *q* is given by (5.1b)

Proof

In (5.8) we introduced the function $q_\mu(h) = q(h) + \frac{1}{2}\mu h^\top h \quad with \quad \mu \geq 0$,

where *q* is given by (5.1b). The gradient of

$$q'_\mu = q' + \mu h = g + (H + \mu I)h,$$ Where   **g** = **f**'(**x**), **H** = **f**''(**x**). According

to the assumption, the matrix **H**+ μ **I** is positive definite, and therefore the linear

system of equations **q**' μ = **0** has a unique solution, which is the minimizer of   *q*

μ . This solution is recognized as h_{dn}. Now, let

$$h_M = \text{argmin}_{\|h\| \le \|h_{dn}\|}\{q(h)\} .$$

Then $q(h_M) \le q(h_M)$ and $h_M^\top h_M \le h_{dn}^\top h_{dn}$, so that

$$q_\mu(h_M) = q(h_M) + \tfrac{1}{2}\mu h_M^\top h_M \le q(h_{dn}) + \tfrac{1}{2}\mu h_{dn}^\top h_{dn} = q_\mu(h_{dn}) .$$

However, h_{dn} is the unique minimizer of q_μ, so $h_M = h_{dn}$.

5.1)

In a proper trust region method we monitor the trust region radius a the theorem

shows that if we monitor the damping parameter instead, we can think of it as a

trust region method with the trust region radius given implicitly as D= // **h**dn //. In

Levenberg–Marquardt type methods μ is updated in each iteration step.Given the

present value of the parameter, the Cholesky factorization of **f** "(**x**)+ μ **I** is

employed to check for positive definiteness, and μ is increased if the matrix is

not significantly positive definite. otherwise, the solution **h**dn is easily obtained

via the factorization. the direction given by **h**dn is sure to be downhill, and we get

the "trial point" **x**+**h**dn (corresponding to *a*=1 in Algorithm 5.1). As in a trust

region Method (see Section 2.4) we can investigate the value of the cost function

at the trial point, i.e. *f*(**x**+ **h**dn). If it is sufficiently below *f*(**x**), then the trial point is

chosen as the next iterate. Otherwise, **x** is still the current iterate (corresponding

to *a*=0 in Algorithm 5.1), and *¹* is increased. It is not sufficient to check whether

f(**x** + **h**dn) < *f*(**x**). In order to prove convergence for the whole procedure one

needs to test whether the actual decrease in *f*-value is larger than some small

portion of the decrease predicted by the quadratic model (5.1b), i.e. if

$$r \equiv \frac{f(\mathbf{x}) - f(\mathbf{x}+\mathbf{h})}{q(\mathbf{0}) - q(\mathbf{h})} > \delta, \qquad\qquad (5.12)$$

where δ, is a small positive number (typically $\delta \simeq 10^{-3}$). We recognize

r as the *gain factor*, (2.24). It is also used to monitor/ μ : If r is close to one, then

one may expect the model to be a good approximation to f in a neighbourhood of

x, which is large enough to include the trial point, and the influence of Newton's

method should be increased by decreasing μ . If, on the other hand, the actual

decrease of f is much smaller than expected, then μ must be increased in order

to adjust the method more towards steepest descent. It is important to note that in

this case the length of $\mathbf{h}_{dn}$ is also reduced, cf (5.10).We could use an updating

strategy similar to the one employedin Algorithm **if** $r > 0.75$ $\mu := \mu/3$

       if $r < 0.25$ $\mu := \mu * 2$ $\qquad\qquad$ (5.13)

however, the discontinuous changes in μ when r is close to 0.25 or 0.75 can

cause a "flutter" that slows down convergence. therefore, we recommend to use

the equally simple strategy given by

$$
\begin{aligned}
&\textbf{if } r > 0 \\
&\quad \mu := \mu * \max\{\tfrac{1}{3}, 1 - (2r-1)^3\} \\
&\textbf{else} \\
&\quad \mu := \mu * 2
\end{aligned}
\qquad (5.14)
$$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ (5.15

the two strategies are illustrated in Figure 5.3
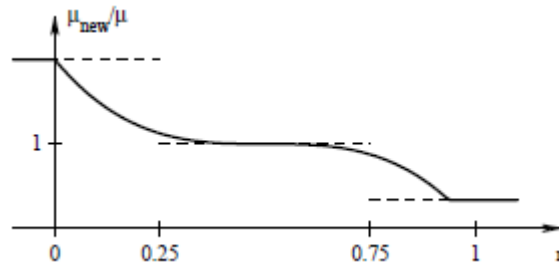
Figure 5.3:

Updating of μ by (5.13) (dasheded line)   and by (5.14) (full line).

The method is summarized in Algorithm 5.15 below. Similar to (4.7) we can use the stopping criteria

$$\|f'(x)\|_\infty \le \varepsilon_1 \quad \text{or} \quad \|h_{dn}\|_2 \le \varepsilon_2(\varepsilon_2 + \|x\|_2). \qquad (5.16)$$

The simplicity of the original Newton method has disappeared in the attempt to obtain global convergence, but this type of method does perform well in general.

Algorithm 5.3 **Levenberg–Marquardt type damped Newton method**

begin

```
x := x₀;  μ := μ₀;  found := false;  k := 0;          {Initialisation}
repeat
    while f″(x)+μI not pos. def.                       {using ...}
        μ := 2μ
    Solve (f″(x)+μI) hₐₙ = −f′(x)                      {... Cholesky}
    Compute gain factor r by (5.12)
    if r > δ                                           {f decreases}
        x := x + hₐₙ                                    {new iterate}
        μ := μ * max{⅓, 1 − (2r − 1)³}                 {...and μ}
    else
        μ := μ * 2                                     {increase μ but keep x}
    k := k+1;  Update found                            {see (5.16)}
until found or k > kₘₐₓ
: end
```

Example 5.4. Table 5.5 illustrates the performance of Algorithm 5.15 when

applied to the tricky function (5.5) with the poor starting point. We use $\mu_0 = 1$ and

$\varepsilon_1 = 10^{-8}, \varepsilon_2 = 10^{-12}$ in (5.16). $x_0 = [-0.\ 0\ ]\ T$

The solution is found without problems, and the columns with f and $[[f']]$ show

super linear convergence, as defined in

Example 5.5.

We have used Algorithm 5.15 on Rosen Brock's function from Example 4.3. We

use the same starting point, $x_0 = [-0.\ 0\ ]\ T\ X_0 = [-1,.2.\ 1\ ]T$ and with $\mu_0 = 1$,

$\varepsilon_1 = 10^{-10}, \varepsilon_2 = 10^{-12}$, we found the solution after 29 iteration steps. The

Performance is illustrated below.
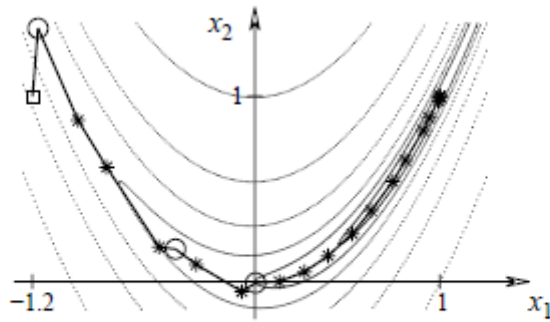
Figure 5.4a: *Damped Newton method on Rosenbrock's function. Iterates.*
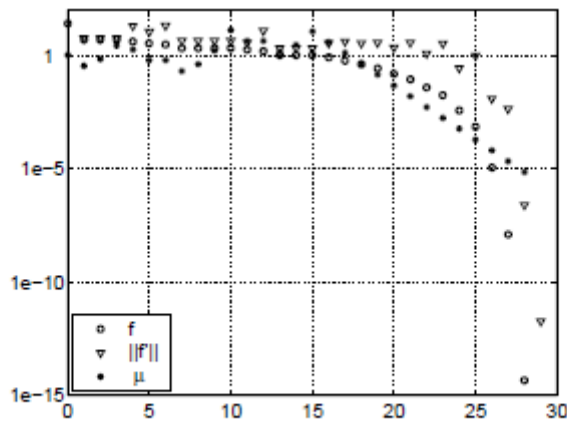
Figure 5.4b: $f(\mathbf{x}_k)$, $\|\mathbf{f}'(\mathbf{x}_k)\|_\infty$ *and* μ.

the three circles in Figure 5.4a indicates points, where the iterations stalls, i.e. the current $\mathbf{x}$ is not changed, but μ is updated. after passing the bottom of the parabola, the damping parameter μ is decreased in each step. as in the previous example we achieve super linear final convergence.

5.3. Quasi–Newton Methods: {1}

The modifications discussed in the previous section make it possible to overcome the first three of the main disadvantages of Newton's method shown in Table 5.3: the damped Newton method is globally convergent,

Ill-conditioning may be avoided, and minima are rapidly located. However, no

means of overcoming the fourth disadvantage has been considered. the user must still supply formulas and implementations of the second derivatives of the cost function. In *Quasi–Newton methods* (from Latin, quasi: nearly) the idea is to use matrices which approximate the Hessian matrix or its inverse, instead of the Hessian matrix or its inverse in Newton's equation (5.2). The matrices are normally named

$$\mathbf{B} \simeq \mathbf{f}''(\mathbf{x}) \qquad \text{and} \qquad \mathbf{D} \simeq \mathbf{f}''(\mathbf{x})^{-1} . \tag{5.17}$$

The matrices can be produced in many different ways ranging from very Simple techniques to highly advanced schemes, where the approximation is built up and adjusted dynamically on the basis of information about the first derivatives, obtained during the iteration. These advanced Quasi–Newton methods, , are some of the most powerful methods for solving unconstrained optimization problems. Possibly the simplest and most straight-forward Quasi–Newton method is Obtained if the elements of the Hessian matrix are approximated by finite differences: In each coordinate direction, $\mathbf{e}i$ ($i=1\dots n$), a small increment δ_i is added to the corresponding element of $\mathbf{x}$ and the gradient in this point is calculated. the *I* th column of a matrix $\mathbf{B}$ is calculated as the *difference approximation* $\left(\mathbf{f}'(\mathbf{x}+\delta_i\mathbf{e}_i) - \mathbf{f}'(\mathbf{x})\right)/\delta_i$ After this, the symmetric matrix $\hat{\mathbf{B}} := \frac{1}{2}(\mathbf{B}+\mathbf{B}')$ is formed. If the δ_i are chosen appropriately, this

is a good approximation to $\mathbf{f}''(\mathbf{x})$ and may be used in a damped Newton method. however, ` and n extra evaluations of the gradient guarantee that $\mathbf{B}$ is positive (semi-)definite. In the advanced Quasi–Newton methods these extra gradient evaluations are avoided. Instead we use updating formulas where the $\mathbf{B}$ or $\mathbf{D}$ matrices (see 5.17) are determined from information about the iterates, $\mathbf{X}_1$, $\mathbf{X}_2$ **……** and the g gradients of the cost function, $\mathbf{f}'(\mathbf{X}_1)$, $\mathbf{f}'(\mathbf{X}_2)$**….** gathered during the Iteration steps. thus, in each iteration step the $\mathbf{B}$ (or $\mathbf{D}$) matrix is changed So that it finally converges towards $f''(\mathbf{x}^*)$ (or respectively $f''(\mathbf{x}^*)^{-1}$), $\mathbf{x}^*$ being the minimizer.

5.4. Quasi–Newton with Updating Formulas: {1}

short discussion on why approximations to the inverse Hessian are preferred rather than approximations to the Hessian itself:

1. First, the computational lab our in the updating is the same no matter which of the matrices we update.

2. Second, if we have an approximate inverse, then the search direction is found simply by multiplying the approximation with the negative gradient of f. this is an $O(N2)$ process where is the solution of the linear system with $\mathbf{B}$ as coefficient matrix is an $O(N3)$ process.

3. A third possibility is to use approximations to the Cholesky factor of the Hessian matrix, determined at the start of the iteration and updated

in the iteration. Using these, we can find the solution of the system

(5.2) in O (N_2) operations. With updating formulas indicated that the

updating of an approximation to the inverse Hessian might become

unstable. , recent research indicates that this needs not be the case. A

classical Quasi–Newton method with updating always includes a line

search. Alternatively, updating formulas have been used in trust region

methods. Basically, these two different approaches (line search or trust

region) define two classes of methods. In this section we shall confine

ourselves to the line search approach. With these comments the

framework may be presented.

<u>Framework 5.18.</u> Iteration step in Quasi–Newton with updating

and line search. B (or **D**) is the current approximation to $\mathbf{f}\ ''(\mathbf{x})$(or $\mathbf{f}\ ''(\mathbf{x})$-1)

Solve **B** $\mathbf{h_{qn}}$= -$\mathbf{f}\ '(\mathbf{x})$ (or compute $\mathbf{h_{qn}}$:= -**Df** $'(\mathbf{x})$)line search along $\mathbf{h_{qn}}$ giving

 $\mathbf{h}$qn := a $\mathbf{h_{qn}}$; $\mathbf{x_{new}}$ = $\mathbf{x}$ + $\mathbf{h_{qn}}$

Update **B** to $\mathbf{B_{new}}$ (or **D** to $\mathbf{D_{new}}$) In the following we present the requirements to

the updating and the techniquesneeded.

5.5. The Quasi–Newton Condition: {1}

An updating formula must satisfy the so-called *Quasi–Newton condition,*

which may be derived in several ways. The condition is also referred to as

the *secant condition,* because it is closely related to the secant method for

non-linear equations with one unknown.

Let $\mathbf{x}$ and $\mathbf{B}$ be the current iterate and approximation to $\mathbf{f}''(\mathbf{x})$. Given these, the first parts of the iteration step in Framework 5.18 can be performed yielding $\mathbf{h_{qn}}$ and hence $\mathbf{x_{new}}$. The objective is to calculate $\mathbf{B}$ new by a correction of $\mathbf{B}$. The correction must contain some information about the second derivatives. Clearly, this information is only approximate. It is based on the gradients of f at the two points. Now, consider the Taylor expansion of $\mathbf{f'}$ *around* $\mathbf{X} + \mathbf{h_{qn}}$:

$$\mathbf{f}'(\mathbf{x}) = \mathbf{f}'(\mathbf{X} + \mathbf{h_{qn}}) - \mathbf{f}''(\mathbf{X} + \mathbf{h_{qn}})\,\mathbf{h_{qn}} + \ldots\ldots\ldots\ldots : (5.19)$$

We assume that we can neglect the higher order terms, and with the notation

$$y = \mathbf{f}'(\mathbf{x_{new}}) - \mathbf{f}'(\mathbf{x}) \qquad\quad ; (5.20)$$

Equation (5.19) leads to the relation, similar to (4.5),

$$y \simeq \mathbf{f}''(\mathbf{x_{new}})\,\mathbf{h_{qn}} :$$

Therefore, we require that $\mathbf{B_{new}}$ should satisfy

$$\mathbf{B_{new}}\,\mathbf{h_{qn}} = y : (5.21a)$$

This is the *Quasi–Newton condition.* The same arguments lead to the alternative formulation of the Quasi–Newton condition, $\qquad \mathbf{D_{new}}\,y = \mathbf{h_{qn}} : (5.21b)$

The Quasi–Newton condition only supplies n conditions on the matrix $\mathbf{B_{new}}$ (or $\mathbf{D_{new}}$) but the matrix has $n2$ elements. Therefore additional conditions are needed to get a well defined method. In the Quasi–Newton methods that we describe,

the **B** (or **D**) matrix is updated in each iteration step. We produce **B**_{new} (or **D**new)

by adding a correction term to the present **B** (or **D**). Important requirements to the

updating are that it must be simple and fast to perform and yet effective. This can

be obtained with a recursive relation between successive approximations,

$$\mathbf{B_{new} = B + W};$$

Where **W** is a correction matrix. In most methods used in practice, **W** is

$$B_{new} = B + ab^\top$$

or a *rank-two matrix*

$$B_{new} = B + ab^\top + uv^\top \ ,$$

Where **a***;* **b***;* **u***;* **v** *in* IR_n. Hence **W** is an *outer product* of two vectors or a sum of

two such products. Often **a**=**b** and **u**=**v**; this is a simple way of ensuring that **W** is

symmetric.

5.6. Broyden's Rank-One Formula:

Tradition calls for a presentation of the simplest of all updating formulas which

was first described by Broyden (1965) *[1]* It was not the first updating Formula

but we present it here to illustrate some of the ideas and techniques Used to

establish updating formulas.

First, consider rank-one updating of the matrix **B** : $B_{new} = B + ab^\top .$ The

vectors **a***;* **b**$_2$ *in* IR$_n$ are chosen so that they satisfy the Quasi–NewtonCondition

$$(5.21(a), \quad (B + ab^\top)\, h_{qn} = y\,, \qquad ;\ (5.22a)$$

And – in an attempt to keep information already in **B** – Broyden demands

That for all **v** orthogonal to h_{qn} we get $B_{new}\, v = Bv$, ie

$$(B + ab^\top)\, v = Bv \quad \text{for all } v \mid v^\top h_{qn} = 0\,. \qquad (5.22b)$$

These conditions are satisfied if we take **b** = h_{qn} and the vector **a** determinedby

$$(h_{qn}^\top h_{qn})\, a = y - Bh_{qn}\,. \qquad \text{thiese results in } \textit{Broyden's rank-one formula for}$$

updating the approximation to the Hessian:

$$B_{new} = B + \frac{1}{h_{qn}^\top h_{qn}}(y - Bh_{qn})\, h_{qn}^\top\,. \qquad :\ (5.23)$$

A formula for updating an approximation to the inverse Hessian may be

derived in the same way and we obtain

$$D_{new} = D + \frac{1}{y^\top y}(h_{qn} - Dy)\, y^\top\,. \qquad (5.24$$

Now, given some initial approximation D_0 (or B_0) (the choice of which shall), we

can use (5.23) or (5.24) to generate the sequence needed in the framework.

However, two important features of the Hessian (or its inverse) would then be

disregarded: We wish both matrices **B** and **D** to be symmetric and positive

definite. This is not the case for (5.23) and (5.24), and thus the use of Broyden's

formula may lead to steps which are not even downhill, and convergence towards saddle points or maxima will often occur. Therefore, these formulas are never used for unconstrained optimization. Broyden's rank-one formula was developed for solving systems of nonlinear equations. Further, the formulas have several other applications, e.g. in methods for least squares and minimax optimization.

5.7. Symmetric Updating: {1}

Since $f''(x)^{-1}$ is symmetric, it is natural to require **D** to be so. If at the Same time rank-one updating is required; the basic recursion must have the form

$$\mathbf{D}_{new} = \mathbf{D} + \mathbf{u}\mathbf{u}^{\mathsf{T}}. \qquad (5.25a)$$

The Quasi–Newton condition (5.21b) determines **u** uniquely: Substituting (5.25) into (5.21b) and letting **h** denote **h**qn yields

$$\mathbf{h} = \mathbf{D}\mathbf{y} + \mathbf{u}\mathbf{u}^{\mathsf{T}}\mathbf{y} \quad \Longleftrightarrow \quad \mathbf{h} - \mathbf{D}\mathbf{y} = (\mathbf{u}^{\mathsf{T}}\mathbf{y})\mathbf{u}. \qquad (5.25b)$$

this shows that $\mathbf{u} = \gamma(\mathbf{h} - \mathbf{D}\mathbf{y})$ where γ is the scalar $\gamma = \mathbf{u}^{\mathsf{T}}\mathbf{y}$. by rescaling **u** we get the *SR₁ formula* (symmetric rank-one updating formula)

$$\mathbf{D}_{new} = \mathbf{D} + \frac{1}{\mathbf{u}^{\mathsf{T}}\mathbf{y}}\mathbf{u}\mathbf{u}^{\mathsf{T}} \qquad \text{with } \mathbf{u} = \mathbf{h} - \mathbf{D}\mathbf{y}. \qquad : (5.26)$$

It may be shown that if $\mathbf{h} = \mathbf{D}\mathbf{y}$, then $\mathbf{D}_{new} = \mathbf{D}$ is the only solution to the Problem of finding a symmetric rank-one update which satisfies (5.21ab). If,

However $y^\top u = 0$ while at the same time $\mathbf{h} \neq \mathbf{Dy}$, then there is no solution, and the updating breaks down. Thus, in case the denominator becomes small we simply set $\mathbf{D_{new}} = \mathbf{D}$ and avoid division by zero .The SR₁ formula has some interesting properties. The most important is that a Quasi–Newton method without line search based on SR1 will minimize a quadratic function with positive definite Hessian in at most $n+1$ iteration steps, provided the search directions are linearly independent and $y^\top u$ remains positive. Further, in this case $\mathbf{D_{new}}$ equals $f''(x)^{-1}$ after $n+1$ steps. This important property is called *quadratic termination,* of Section 4.1. The SR₁ formula has only been used very little in practice. this is due to the fact that $y^\top u$ may vanish, whereby numerical instability is introduced or the updating breaks down. A similar derivation gives the SR1 formula for approximations to

$\mathbf{f''(x)}$,
$$B_{new} = B + \frac{1}{h^\top v} vv^\top \qquad \text{with } v = y - Bh ,$$

And similar comments can be made.

5.8. *Preserving Positive Definiteness{1}*

Consider Newton's equation (5.2) or a Quasi–Newton equation based on (5.17). The step is determined by $\qquad \mathbf{Gh} = -\mathbf{f}'(\mathbf{x})$;

where $\mathbf{G} = \mathbf{f}''(\mathbf{x})$ (Newton) or – in the case of Quasi–Newton, $\mathbf{G} = \mathbf{B}$ or

$G = D^{-1}$. From Theorem 2.14 on page 14 it follows that **h** is a downhill Direction if **G** is positive definite, and this is a property worth striving for. If we use **D** = **I** (the identity matrix) in all the steps in the Quasi–Newton Framework 5.18, then the method of steepest decent appears. As discussed in Chapter three this method has good global convergence properties, but the final convergence is often very slow. If, on the other hand, the iterates are near the solution **x***, a Newton method (and also a Quasi–Newton method with good Hessian approximations) will give good performance, close to quadratic convergence. Thus a good strategy for the updating would be to use **D** close to **I** in the initial iteration step and then successively let **D** approximate $f''(x)^{-1}$ better and better towards the final phase. this will make the iteration start like the steepest descent and end up some where like Newton's method. if, in addition, the updating preserves positive definiteness for all coefficient matrices, all steps will be downhill and a reasonable rate of convergence can be expected, since $f''(x)^{-1}$ is positive (semi-)definite at a minimizer.

5.9. The DFP Formula{2}

Symmetric rank-two formula can be written as $D_{new} = D + uu^{\top} + vv^{\top}$. :

We insert this in the Quasi–Newton condition (5.21b) and get

$$h = Dy + uu^\top y + vv^\top y.$$ With two updating terms there is no unique

determination of **u** and **v**, but Fletcher points out that an obvious choice is to try

$$u = \alpha h, \qquad v = \beta Dy.$$ Then the Quasi–Newton condition will be

satisfied if $u^\top y = 1$ and $v^\top y = -1$, and this yields the formula

Definition **5.27. (DFP updating.)**

$$D_{new} = D + \frac{1}{h^\top y} hh^\top - \frac{1}{y^\top v} vv^\top,$$

where

$h = x_{new} - x, \quad y = f'(x_{new}) - f'(x), \quad v = Dy$. This was the dominating

formula for more than a decade and it was found to work well in practice. In

general it is more efficient than the conjugate Gradient method (see Chapter

four). Traditionally it has been used in Quasi–Newton methods with exact line

search, but it may also be used with soft line search as we shall see in a moment.

A method like this has the following important properties:

(i)On quadratic objective functions with positive definite Hessian:

a) it terminates in at most n iterations with $\mathbf{D_{new}} = f''(x)^{-1}$,

b) it generates conjugate directions,

c) it generates conjugate gradients if $\mathbf{D_0} = \mathbf{I}$, Provided that the method uses exact

line search. On general functions:

٨٥

(ii) On general functions:

d) it preserves positive definite **D**-matrices if $\mathbf{h_{qn}}^{\top}\mathbf{y} > 0$ in all steps,

e) it gives super linear final convergence,

f) it gives global convergence for strictly convex objective functions provided that the line searches are exact Here we have a method with super linear final convergence (defined in (2.6)).Methods with this property are very useful because they finish the iteration with fast convergence. Also, in this

case $\|\mathbf{x}^* - \mathbf{x_{new}}\| \ll \|\mathbf{x}^* - \mathbf{x}\|$ for $k \to \infty$, and *[[x*-x_new]]<1*

Implying that *[[x_new –x]]* can be used to estimate the distance from **x** to **x***

Example 5.6.

Assume that **D** is positive definite. Then its *Cholesky factor* exists: $\mathbf{D} = \mathbf{CC}^{\top}$, and for any non-zero **in the** IR_n we use Definition 5.1 to find

$$\mathbf{z}^{\top}\mathbf{D_{new}z} = \mathbf{x}^{\top}\mathbf{Dz} + \frac{(\mathbf{z}^{\top}\mathbf{h})^2}{\mathbf{h}^{\top}\mathbf{y}} - \frac{(\mathbf{z}^{\top}\mathbf{Dy})^2}{\mathbf{y}^{\top}\mathbf{Dy}}.$$

We introduce $\mathbf{a} = \mathbf{C}^{\top}\mathbf{z}$, $\mathbf{b} = \mathbf{C}^{\top}\mathbf{y}$ and $\theta = \angle(\mathbf{a}, \mathbf{b})$, cf (2.12), and get

$$\mathbf{z}^{\top}\mathbf{D_{new}z} = \mathbf{a}^{\top}\mathbf{a} - \frac{(\mathbf{a}^{\top}\mathbf{b})^2}{\mathbf{b}^{\top}\mathbf{b}} + \frac{(\mathbf{z}^{\top}\mathbf{h})^2}{\mathbf{h}^{\top}\mathbf{y}}$$

$$= \|\mathbf{a}\|^2 (1 - \cos^2\theta) + \frac{(\mathbf{z}^{\top}\mathbf{h})^2}{\mathbf{h}^{\top}\mathbf{y}}.$$

If $\mathbf{h}^{\top}\mathbf{y} > 0$, then both terms on the right-hand side are non-negative. The first term vanishes only if $\theta = 0$, ie when **a** and **b** are proportional, which implies

that **z** and **y** are proportional, $\mathbf{z} = \beta\mathbf{y}$ with $\beta \neq 0$. In this case the second term

becomes $(\beta\mathbf{y}^\top\mathbf{h})^2/\mathbf{h}^\top\mathbf{y}$ which is positive due to the basic assumption.

Hence, $\mathbf{z}^\top \mathbf{D}_{new}\mathbf{z} > 0$ for any non-zero **z** and **D**new is positive definite. The

essential condition $\mathbf{h}^\top\mathbf{y} > 0$ is called the *curvature condition* because it can be

expressed as

$$\mathbf{h}^\top \mathbf{f}'_{new} > \mathbf{h}^\top \mathbf{f}' . \qquad : (5.28)$$

Notice, that if the line search slope condition (2.17) is satisfied then (5.28) is also

satisfied since $\mathbf{h}^\top\mathbf{f}' = \varphi'(0)$ and $\mathbf{h}^\top\mathbf{f}'_{new} = \varphi'(\alpha_s)$), where $\varphi(\alpha)$ is the line search

5.10. The BFGS Formulas

the final updating formulas are known as the *BFGS formulas[2]*

. They were discovered independently by Broyden, Fletcher, Goldfarb these

formulas are the most popular of all the updating formulas, as we saw with the

DFP formula, the BFGS formulas are difficult to derive directly from the

requirements. However, they arrive in a simple way through the concept of

duality, which will be discussed briefly here. Remember the Quasi–Newton

conditions (5.21): $\qquad$ $\mathbf{B}_{new}\mathbf{h} = \mathbf{y}$ $\quad$ and $\qquad\qquad$ $\mathbf{D}_{new}\,\mathbf{y} = \mathbf{h}$ $\qquad$ *(5.29)*

These two equations have the same form, except that **h** and **y** are interchanged

and **B**new is replaced by **D**new. This implies that any updating formula for **B** which

satisfies (5.21(a) can be transformed into an updating formula for **D**. Further, any

formula for **D** has a dual formula for **B** which is found by the substitution**D** $\leftrightarrow$ **B** and **h** $\leftrightarrow$ **y** .Performing this operation on the DFP formula (5.27) yields the following updating formula,

Definition (5.2). (**BFGS updating.**)

$$B_{new} = B + \frac{1}{h^\top y} yy^\top - \frac{1}{h^\top u} uu^\top ,$$

Where $h = X_{new} - x,$

           $y = f'(X_{new}) - f'(x),\ u = Bh$:

This updating formula has much better performance than the DFP formula; for an excellent explanation why this is the case. If we make the dual operation on the BFGS update we return to the DFP updating, as expected. The BFGS formula produces **B** which converges to **f** ''(**x***) and the DFP formula produces **D** which converges to $f''(x^*)^{-1}$.. Alternatively, we can find another set of matrices [**D**]which has the same convergence, although it is different from the **D**-matrices produced by DFP. The BFGS formula is a rank two update, and there are formulas which give the corresponding update for B^{-1} :

(5.3)Definition (BFGS updating for D)

$$D_{new} = D + \kappa_1 hh^\top - \kappa_2 \left(hv^\top + vh^\top \right) ,$$

Where $h = x_{new} - x, \quad y = f'(x_{new}) - f'(x);$ (5.31)

$\wedge\wedge$

$$\mathbf{v} = \mathbf{D}\mathbf{y}$$

$$\kappa_2 = \frac{1}{\mathbf{h}^\top \mathbf{y}} \, , \qquad \kappa_1 = \kappa_2\big(1 + \kappa_2(\mathbf{y}^\top \mathbf{v})\big)$$

The BFGS formulas are always used together with a soft line search and as discussed above the procedure should be initiated with the full Quasi–Newton step in each iteration step, ie the initial a in Algorithm 2.27 should be one. Experiments show that it should be implemented with a very loose line search; typical values for the parameters in (2.26) are $\varrho = 10^{-4}$ and $B = 0.9$. The properties a) – f) of the DFP formula also hold for the BFGS formulas. Moreover, Powell has proved a better convergence result for the latter formulas namely that they will also converge with a soft line search on convex problems. Unfortunately, convergence towards a stationary point has not been proved for neither the DFP nor the BFGS formulas on general nonlinear functions – no matter which type of line search. Still, BFGS with soft line search is known as the method which never fails to come out with a stationary point.

5.11. Quadratic Termination

We indicated above that there is a close relationship between the DFP update and the BFGS-updates. Still, their performances are different with the DFP update performing poorly with soft line search. Broyden suggested to combine the two sets of formulas: *[1]*

Definition Broyden's one parameter family.

$$D_{new} = D + \sigma W_{DFP} + (1-\sigma)W_{BFGS} , \quad (5.32)$$

where $0 \leq \sigma \leq 1$ and **W**DFP and **W**BFGS are the updating terms in definitions (5.3) respectively. The parameter ¾ can be adjusted during the iteration, for details. He remarks that π =0, pure BFGS updating is often the best. We want to state a result for the entire Broyden family, a result which consequently is true for both DFP and BFGS. The result is concerned with *quadratic termination*:

5.12Remark

The Broyden one parameter updating formula gives quadratic termination for all values of σ ($0 \leq \sigma \leq 1$ provided that **D**$_0$ is positive definite .this implies that a Quasi–Newton method with exact line search determines the minimizer of a positive definite quadratic after no more than *n* iteration steps (*n* being the dimension of the space). The basis of all the updating formulas in this chapter is the Quasi–Newton conditions (5.21(a)–(b). this corresponds to a linear interpolation in the gradient of the cost function. If the cost function is quadratic, then its gradient is linear in **X**, and so is its approximation. When the Quasi–Newton condition has been enforced in *n* steps, the two linear functions agree in *n*+1 positions in IR$_n$, and consequently the two functions are identical. Iterate no. *n*+1, **x**new, makes the gradient of the approximation equal to zero, and so it also makes the gradient of the cost function equal to zero; it solves the problem. the

proviso that the quadratic and $\mathbf{D}_0$ must be positive definite, ensures that $\mathbf{x_{new}}$ is not only a stationary point, but also a minimizer.

5.13. Implementation of a Quasi–Newton Method{1} discuss some details of

the implementation and end by giving the Quasi–Newton algorithm with the different parts assembled. Based on the discussion in the previous sections we have chosen a BFGS updating formula the an update of the inverse hessian , but even though $\mathbf{D}$ in theory stays positive definite, the rounding errors may cause ill conditioning and even indefiniteness. For professional codes updating of a *factorization* of the Hessian is recommended such that the effect of rounding errors can be treated properly. In the present context a less advanced remedy is described _which is to omit the updating if the curvature condition does not hold, since in this case $\mathbf{D_{new}}$ would not be positive definite. Actually, Dennis and Schnabel recommend that the updating is skipped if

$$\mathbf{h}^\top \mathbf{y} \le \varepsilon_M^{1/2} \, \|\mathbf{h}\|_2 \, \|\mathbf{y}\|_2 \, , \qquad ; (5.33)$$

where $''M$ is the *machine precision.* We shall assume the availability of a soft line search such as Algorithm 2.3. It is important to notice that all the function evaluations take place during the line search. hence, the values of f and $\mathbf{f}'$ at the new point are received from the line search subprogram. In the next iteration step these values are returned to the subprogram such that f and $\mathbf{f}'$ for $a = 0$ are ready for the next search. Sometimes the gradient needs not be calculated as often as f.

In a production code the line search should only calculate f respectively $\mathbf{f}\,'$ whenever they are needed. As regards the initial approximation to the inverse Hessian, $\mathbf{D}_0$, it is traditionally recommended to use $\mathbf{D}_0 = \mathbf{I}$, the identity matrix. this $\mathbf{D}_0$ is, of course, positive definite and the first step will be in the steepest descent direction. Finally, we outline an algorithm for a Quasi–Newton method. Actually, the curvature condition needs not be tested because it is incorporated in the soft line search as s

Algorithm 5.34. Quasi–Newton method with BFGS–updating

begin

```
x := x₀;  D := D₀;  k := 0;  nv := 0                {Initialisation
while ‖f'(x)‖ > ε and k < k_max and nv < nv_max
    h_qn := D (−f'(x))                        {Quasi–Newton equation
    [α, dv] := soft_line_search(x, h_qn)           {Algorithm 2.27
    nv := nv + dv                         {No. of function evaluations
    x_new := x + αh_qn;  k := k+1
    if h_qn^⊤ f'(x_new) > h_qn^⊤ f'(x)                  {Condition (5.28)
        Update D                                      {using 5.30
    x := x_new
end
```

$$\mathbf{x} := \mathbf{x}_0;\ \ \mathbf{D} := \mathbf{D}_0;\ \ k := 0;\ \ nv := 0 \qquad \{\text{Initialisation}$$
$$\textbf{while } \|\mathbf{f}'(\mathbf{x})\| > \varepsilon \textbf{ and } k < k_{\max} \textbf{ and } nv < nv_{\max}$$
$$\quad \mathbf{h}_{qn} := \mathbf{D}\,(-\mathbf{f}'(\mathbf{x})) \qquad \{\text{Quasi–Newton equation}$$
$$\quad [\alpha, dv] := \text{soft\_line\_search}(\mathbf{x}, \mathbf{h}_{qn}) \qquad \{\text{Algorithm 2.27}$$
$$\quad nv := nv + dv \qquad \{\text{No. of function evaluations}$$
$$\quad \mathbf{x}_{\text{new}} := \mathbf{x} + \alpha \mathbf{h}_{qn};\ \ k := k+1$$
$$\quad \textbf{if } \mathbf{h}_{qn}^{\top}\mathbf{f}'(\mathbf{x}_{\text{new}}) > \mathbf{h}_{qn}^{\top}\mathbf{f}'(\mathbf{x}) \qquad \{\text{Condition (5.28)}$$
$$\qquad \text{Update } \mathbf{D} \qquad \{\text{using 5.30}$$
$$\quad \mathbf{x} := \mathbf{x}_{\text{new}}$$
$$\textbf{end}$$

Example   5.7.

We consider Rosen Brock's function from Examples 4.3 and 5.5. We have tried

different updating formulas and line search methods. The line search parameters

were chosen as in Example 4.3. With the starting point $\mathbf{x}_0 = [\text{-1.2. 1 }]$ T,

$\mathbf{x}_0 = [0.\ 0\ ]$ T the following numbers of iteration steps and evaluations of $f(\mathbf{x})$ and

$\mathbf{f}'(\mathbf{x})$ are needed to satisfy the stopping criterion $[\ [\mathbf{f}'\ (\mathbf{x})\ ]\ ] < 10\text{-}10$. The results are

as expected: BFGS combined with soft line search needs the smallest number of

function evaluations to find the solution.

Update by	Line search	# it. steps	# fct. evals
DPF	exact	23	295
DPF	soft	31	93
BFGS	exact	23	276
BFGS	soft	29	68

Below we give the iterates (cf Figures 4.2, 4.3 and 5.4) and the values of $f(x_k)$ and $\|f'(x_k)\|_\infty$. As with the Damped Newton Method we have superlinear final convergence.
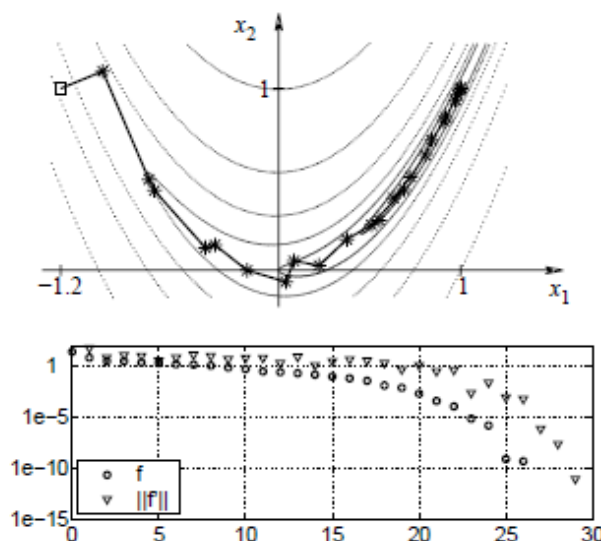

Figure 5.5: *BFGS with soft line search, applied to Rosenbrock's function.*
Top: iterates x_k.   Bottom: $f(x_k)$ and $\|f'(x_k)\|_\infty$.

X0the number of iteration steps is about the same as in Example 5.5, while the number of function evaluations is almost four times as big. Note, however, that with Algorithm 5.34 each evaluation involves $f(\mathbf{x})$ and $\mathbf{f}(\mathbf{x})$, while each evaluation in the Damped Newton Method also involves the Hessian $\mathbf{f}''(\mathbf{x})$. For many problems this is not available. If it is, it may be costly: we need to compute $1/2(n(n+1)$ elements in the symmetric matrix $\mathbf{f}''(\mathbf{x})$, while $\mathbf{f}'(\mathbf{x})$ has n elements only.

5.14 Symmetric,(Positive Definite Matrices){1}

A matrix $\mathbf{A}$ *in* $(R)n*n$ is symmetric if $\mathbf{A} = {}\_\mathbf{A}$, ie if $a_{ij} = a_{ji}$ for all $i; j$.

Definition (514).The symmetric matrix $\mathbf{A}$ *in* $(R)_{n*n}$ is

positive definite   if $\mathbf{x}^\top \mathbf{A}\, \mathbf{x} > 0$ for all $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{x} \neq 0$,

positive semidefinite   if $\mathbf{x}^\top \mathbf{A}\, \mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{x} \neq 0$.

Theorem A.2. $\mathbf{A} \in \mathbf{R}^{n \wedge n}$ be symmetric and let $\mathbf{A} = \mathbf{LU}$, where $\mathbf{L}$ is a unit lower triangular matrix and $\mathbf{U}$ is an upper triangular matrix. Then

1° (All $u_{ii} > 0$, $i=1,\ldots,n$) $\iff$ ($\mathbf{A}$ is positive definite) .

If $\mathbf{A}$ is positive definite, then

2° The LU-factorization is numerically stable.

3° $\mathbf{U} = \mathbf{DL}^\top$ with $\mathbf{D} = \mathrm{diag}(u_{ii})$.

4° $\mathbf{A} = \mathbf{CC}^\top$, the *Cholesky factorization*. $\mathbf{C} \in \mathbf{R}^{n \times n}$ is a lower triangular matrix.

Proof   *unit lower triangular matrix* $\mathbf{L}$ is characterized by $\ell_{ii} = 1$ and $\ell_{ij} = 0$

for $j > i$. Note, that the LU-factorization $\mathbf{A} = \mathbf{LU}$ is made without pivoting (which, by the way, could destroy the symmetry). Also note that points (3—4) give the following relation between the LU- and the Cholesky- factorization

$$\mathbf{A} = \mathbf{LU} = \mathbf{LDL}^\top = \mathbf{CC}^\top \quad \text{(A.3a)}$$

$$\mathbf{C} = \mathbf{LD}^{1/2}, \qquad \mathbf{D}^{1/2} = \mathrm{diag}(\sqrt{u_{ii}}) . \quad : \text{(A.3b)}$$

The Cholesky factorization with test for positive definiteness can be implemented as follows

(5.5)Algorithm . (Cholesky factorization.)

<div align="center">begin</div>

$k := 0; \quad \text{posdef} := \textbf{true}$ {Initialisation}

$\textbf{while } \text{posdef and } k < n$

$\quad k := k+1$

$\quad d := a_{kk} - \sum_{j=1}^{k-1} (c_{kj})^2$

$\quad \textbf{if } d > 0$ {test for pos. def.}

$\quad\quad c_{kk} := \sqrt{d}$ {diagonal element}

$\quad\quad \textbf{for } i := k+1, \ldots, n$ {subdiagonal elements}

$\quad\quad\quad c_{ik} := \left(a_{ik} - \sum_{j=1}^{k-1} c_{ij} c_{kj} \right) / c_{kk}$

$\quad \textbf{else}$

$\quad\quad \text{posdef} := \textbf{false}$

end

The solution to the $\quad\quad\quad$ **Ax = b**

can be computed via the Cholesky factorization: Inserting $\mathbf{A} = \mathbf{CC}^{\top}$ ،

we see that the system splits into $\mathbf{Cz} = \mathbf{b}$ and $\mathbf{C}^{\top}\mathbf{x} = \mathbf{z}$

The two triangular systems are solved by forward- and back-

substitution, respective

Algorithm.(5.6) (Cholesky solve.)

<div align="center">begin</div>

$\textbf{for } k := 1, \ldots, n-1, n$

$\quad z_k := \left(b_k - \sum_{j=1}^{k-1} c_{kj} z_j \right) / c_{kk}$

$\textbf{for } k := n, n-1, \ldots, 1$

$\quad x_k := \left(z_k - \sum_{j=k+1}^{n} c_{jk} x_j \right) / c_{kk}$

end

The "cost" of this algorithm is $O(n2)$ operations.

Example(5.4) :-

A=LDL TO illustrate this factorization consider the positive definite matrix

    A= 1.4   -0.20   .1

       ,-0.2   1.5   -0.3

       ,0.1   -0.3   1.5

Then A can be represented as LDL   where

L=   1   0   0

     -0.1429   1   0

      0.0714   -0.1942   1

D=   1.4000   0   0

      0.0714   -0.1942   1

        0   0   1.7374

The second   factorization is obtained easily from the first since D has positive

diagonal   entries we can write

D=D' D'

Where D' is a' diagonal   matrix   with   d = d   if then define

 L=LD'

Then L' is also a lower triangular matrix and

A= L' L'

EXAMPLE(5.5) :-

 A=L L

D= 1.1832   0   0

     0   1.2130   0

     0   0   1.3181]

L=LD= 1.1832   0   0

      -0.1690   1.2130   0

       0.0845   -0.2355 1.31

 So that

A=LL = 1.832   0   0   1.1832   -0.1690 0.08450,

         0.1690   1.2130   0   *   1.2130   -0.2355   0

         0.0845 -0.2355 1.3181   0   0   1.3181

Note (5.2)

Theorem 1.6. 1st and 2nd order Taylor expansions.

If $f : \mathbb{R}^n \mapsto \mathbb{R}$ has continuous partial derivatives of second order, then

$$f(x+h) = f(x) + h^T \nabla f(x) + O(\|h\|^2) .$$

If f has continuous partial derivatives of third order, then

$$f(x+h) = f(x) + h^T \nabla f(x) + \tfrac{1}{2} h^T \nabla^2 f(x) h + O(\|h\|^3) ,$$

$$\nabla f(x+h) = \nabla f(x) + \nabla^2 f(x) h + \eta , \quad \|\eta\| = O(\|h\|^2) .$$

REFERENCES:

(1).1DTU Informatics – IMM Richard Petersens Plads
DTU – building 321 DK-2800 Kgs . Lyngby   www.imm.dtu.dk

(2). M. Al- Baali (1985): Descent Property and Global

Convergence of the Fletcher- Reeves method with inexact line

search, IMA Journal on Num. Anal. 5, pp 121–124.

(3). And anther book

مصطلحات

x^*	التضغير.......................( )
e_k	أخطاء شائعة.......................( )
h_d	اتجاه.......................( )
H	حدث التناقص.......................( )
ε	زاوية.......................( )
M	زاوية مستقلة.......................( )
N	بعد الخطاء.......................( )
$\lambda(\alpha)$	منحني.......................( )
e.B	متغير.......................( )
γ	متجه المنطقة.......................( )
ε and τ	متغيرات الخطاء.......................( )
h_{sd}	أنجاة التدرج.......................( ).
h_i	البحث الخطي.......................( )
X	بعد البحث الدقيق.......................( )
$h_{cg},$	اتجاه الهبوط الأحق.......................( )
h_{prev}	اتجاه الهبوط السابق.......................( )

Q.B   ( )...................معيار التوقف

α ( )...................طول الخطوة

I   ( )...................محددة الوحدة

h_{dn} ( )...................أصل الاتجاه

$\varepsilon_1 . \varepsilon_2$ ( )...................معايير التوقف

W   (   )...................منتج خارجي لمتجهين

γ ( )...................موثر أو عددية

$\overline{\mu}$ (   )...................دقة الجهاز

D_0 (or B_0)   ( )...................معكوس هسه

DFP   Davidon Fletcher Powel...........( ).