Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Unconstrained optimisation

Numerical optimisation

Francesco Corona

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Numerical optimisation

Definition

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ with $n \geq 1$ be a **cost** or **objective function**

The **unconstrained optimisation** problem is

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \tag{1}$$

The **constrained optimisation** problem is

$$\min_{\mathbf{x} \in \Omega \subset \mathbb{R}^n} f(\mathbf{x}) \tag{2}$$

The closed subset Ω is determined by either equality and inequality constraints that are dictated by the nature of the problem to solve

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Numerical optimisation (cont.)

Example

Find the optimal allocation of $i = 1, \ldots, n$ bounded resources x_i, the constraints will be expressed by inequalities of type

$$0 \leq x_i \leq C_i, \quad \text{with } C_i \text{ given constants}$$

The set $\Omega = \Big\{\mathbf{x} = (x_1, \ldots, x_n) : 0 \leq x_i \leq C_i, i = 1, \ldots, n\Big\}$ is a subset of $\mathbb{R}^n$ that is determined by such constraints

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Numerical optimisation (cont.)

Examples of constrained optimisation problems are those in which Ω is characterised by conditions like:

- $\mathbf{h}(\mathbf{x}) = \mathbf{0}$: **equality constraints**
- $\mathbf{h}(\mathbf{x}) \leq \mathbf{0}$: **inequality constraints**

By $\mathbf{h} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with $m \leq n$ we denote a given function such that

- by $\mathbf{h} \leq \mathbf{0}$ we mean $h_i(\mathbf{x}) \leq 0$ for $i = 1, \dots, m$

Definition

If f is continuous and Ω is connected, the constrained optimisation problem is known also as a **non-linear programming problem**

- **Convex programming**: If f is a convex function and $\mathbf{h}$ has convex components
- **Linear programming**: If f and $\mathbf{h}$ are linear
- **Quadratic programming**: If f is quadratic and $\mathbf{h}$ is linear

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Numerical optimisation (cont.)

Remark

Computing the maximum of function f is equivalent to computing the minimum of function $g = -f$

- We only consider minimisation algorithms

Definition

More interesting of the minimum value of a given function is often the point at which such minimum is achieved

- Such point is called **minimiser**

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Numerical optimisation (cont.)

In view of numerical solutions of optimisation problems, the ideal situation would be a cost function with an unique global minimiser

- There are often several (local) minimiser, though

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Unconstrained optimisation

Numerical optimisation

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Unconstrained optimisation

When minimising an objective function, we are interested in finding either a (good) local or the global minimiser

Definition

- Point $\mathbf{x}^*$ is a **global minimiser** of f, if $f(\mathbf{x}^*) \leq f(\mathbf{x})$, $\forall \mathbf{x} \in \mathbb{R}^n$
- Point $\mathbf{x}^*$ is a **local minimiser** of f, if there is a $B_r(\mathbf{x}^*) \subset \mathbb{R}^n$, a ball centred in $\mathbf{x}^*$ and radius $r > 0$, such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$, $\forall \mathbf{x} \in B_r(\mathbf{x}^*)$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Unconstrained optimisation (cont.)

Definition

Provided that f is differentiable in $\mathbb{R}^n$ with first and second derivatives, we denote by **gradient vector** and by **Hessian matrix** of f at point $\mathbf{x} \in \mathbb{R}^n$, the following objects

$$\nabla f(\mathbf{x}) = \Big(\frac{\partial f}{\partial x_1}(\mathbf{x}), \dots, \frac{\partial f}{\partial x_n}(\mathbf{x})\Big)^T \tag{3}$$

$$\mathbf{H}(\mathbf{x}) = (h_{ij})_{i,j=1}^n, \quad \text{with } h_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_i} \tag{4}$$

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Unconstrained optimisation (cont.)

In general, it will be assumed that problem functions are smooth

- Continuous and continuously (Frétchet) differentiable, $\mathbb{C}^1$

Thus for $f(\mathbf{x})$ at any point $\mathbf{x}$ there is a **vector of first derivatives**

- **Gradient vector**

$$\begin{pmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \\ \vdots \\ \partial f/\partial x_n \end{pmatrix}_{\mathbf{x}} = \boldsymbol{\nabla} f(\mathbf{x}) \tag{5}$$

$\boldsymbol{\nabla}$ is the gradient operator $\Big(\partial/\partial x_1, \partial/\partial x_2, \cdots, \partial/\partial x_n\Big)^T$

If $f(\mathbf{x})$ is twice-differentiable, $\mathbb{C}^2$, there is a **matrix of second partial derivatives** $\boldsymbol{\nabla}^2 f(\mathbf{x})$ for whose (i,j)-th element is

$$\partial^2 f/(\partial x_i \partial x_j)$$

The **Hessian matrix** can be strictly written as $\mathbf{H}(\mathbf{x}) = \boldsymbol{\nabla}(\boldsymbol{\nabla} f^T)$

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

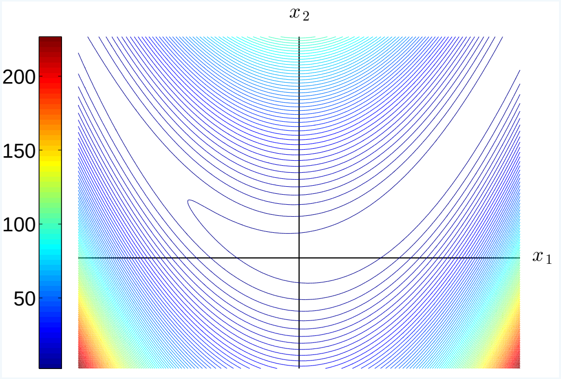
Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Unconstrained optimisation (cont.)

Example

Rosenbrock's function: A test-function for optimisation methods



$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Unconstrained optimisation (cont.)

$$\nabla f(\mathbf{x}) = \begin{pmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{pmatrix} \tag{6a}$$

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix} \tag{6b}$$

In general, ∇f and $\nabla^2 f$ will and vary from point to point

At $\mathbf{x}^T = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\nabla f(\mathbf{x}^T) = \begin{pmatrix} -2 \\ 0 \end{pmatrix}$ and $\nabla^2 f(\mathbf{x}^T) = \begin{bmatrix} 2 & 0 \\ 0 & 200 \end{bmatrix}$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Unconstrained optimisation (cont.)

Definition

If $f \in \mathbb{C}^2(\mathbb{R}^n)$, that is all first and second derivatives of f exist and are continuous, then $\mathbf{H}(\mathbf{x})$ is symmetric for every $\mathbf{x} \in \mathbb{R}^n$

Definition

A point $\mathbf{x}^*$ is called a **stationary** or **critical point** for f if $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and it is called a **regular point** if $\nabla f(\mathbf{x}^*) \neq \mathbf{0}$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Unconstrained optimisation (cont.)

Remark

A function f over $\mathbb{R}^n$ does not necessarily admit a minimiser

- Also, should this point exist it is not necessarily unique

Example

- $f(\mathbf{x}) = x_1 + 3x_2$ is unbounded in $\mathbb{R}^2$
- $f(\mathbf{x}) = \sin(x_1)\sin(x_2)\cdots\sin(x_n)$ admits an infinite number of minimisers and maximisers in $\mathbb{R}^n$, either local and global

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Unconstrained optimisation (cont.)

Definition

Function $f : \Omega \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ is **convex** in Ω if $\forall \alpha \in [0, 1]$

$$f(\alpha \mathbf{x} + (1 - \alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1 - \alpha) f(\mathbf{y}), \quad \forall \mathbf{x}, \mathbf{y} \in \Omega \tag{7}$$

Definition

f is a **Lipschitz function** in Ω if there is a constant $L > 0$

$$||f(\mathbf{x}) - f(\mathbf{y})|| \leq L||\mathbf{x} - \mathbf{y}||, \quad \forall \mathbf{x}, \mathbf{y} \in \Omega \tag{8}$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Unconstrained optimisation (cont.)

Proposition 1.1

Optimality conditions

Let $\mathbf{x}^* \in \mathbb{R}^n$ and $r > 0$ exists such that $f \in \mathbb{C}^1(B_r(\mathbf{x}^*))$

- If $\mathbf{x}^*$ is a minimiser for f (local or global), then $\nabla f(\mathbf{x}^*) = \mathbf{0}$
  - Also, if $f \in \mathbb{C}^2(B_r(\mathbf{x}^*))$, $\mathbf{H}(\mathbf{x}^*)$ is positive semidefinite

Let $\mathbf{x}^* \in \mathbb{R}^n$ and $r > 0$ exists such that $f \in \mathbb{C}^2(B_r(\mathbf{x}^*))$

- If $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\mathbf{H}(\mathbf{x}^*)$ is positive definite for all $\mathbf{x} \in B_r(\mathbf{x}^*)$, then $\mathbf{x}^*$ is a local minimiser of f
- If $f \in \mathbb{C}^1(\mathbb{R}^n)$ is convex in $\mathbb{R}^n$ and $\nabla f(\mathbf{x}^*) = \mathbf{0}$, then $\mathbf{x}^*$ is a global minimiser for f

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Unconstrained optimisation (cont.)

Definition

A symmetric real matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is **positive definite** if

$$\forall \mathbf{x} \in \mathbb{R}^n \text{ with } \mathbf{x} \neq \mathbf{0}, \quad \mathbf{x}^T \mathbf{A} \mathbf{x} > 0$$

A symmetric real matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is **positive semidefinite** if

$$\forall \mathbf{x} \in \mathbb{R}^n \text{ with } \mathbf{x} \neq \mathbf{0}, \quad \mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$$

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Unconstrained optimisation (cont.)

Most methods for numerical optimisation are of iterative type

They can be classified into two categories depending on whether they require knowledge of the derivatives of the cost function

- **Derivative-free methods** investigate the local behaviour of a cost function by direct comparison between the values it takes
- **Methods using exact derivatives** take advantage of accurate information on the local behaviour of the cost

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Unconstrained optimisation (cont.)

In general, minimisation methods based on accurate derivatives can be expected to achieve faster convergence to the minimiser

Remark

- It can be shown that given $\bar{\mathbf{x}} \in \text{dom}(f)$, if $\nabla f(\bar{\mathbf{x}})$ exists and it is not null, then the largest increase of f from $\bar{\mathbf{x}}$ is along the gradient vector whereas the largest decrease is along the opposite direction

Among them, the two most important classes of techniques are

- **Line-search methods**
- **Trust-region methods**

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Derivative-free methods
Numerical optimisation

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Derivative-free methods

We describe two simple numerical methods for

- **Minimisation of univariate real-valued functions**
- **Minimisation of multivariate real-valued functions**, along a single direction

We then describe the **Nelder and Mead method** for the **minimisation of functions of several variables**

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation

Derivative-free methods

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation

Let $f : (a, b) \rightarrow \mathbb{R}$ be a continuous function with unique minimiser

$$x^* \in (a, b)$$

Set $I_0 = (a, b)$ and for $k \geq 0$ generate a sequence of intervals I_k

$$I_k = (a^{(k)}, b^{(k)})$$

The intervals I_k are of decreasing length and each contains x^*

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)

For any given k, the next interval I_{k+1} is determined as follows:

1) Let $c^{(k)}, d^{(k)} \in I_k$ with $c^{(k)} < d^{(k)}$ be two points such that

$$\frac{b^{(k)} - a^{(k)}}{d^{(k)} - a^{(k)}} = \frac{d^{(k)} - a^{(k)}}{b^{(k)} - d^{(k)}} = \varphi \tag{9a}$$

$$\frac{b^{(k)} - a^{(k)}}{b^{(k)} - c^{(k)}} = \frac{b^{(k)} - c^{(k)}}{c^{(k)} - a^{(k)}} = \varphi \tag{9b}$$

and let φ be the **golden ratio** $\varphi = \dfrac{1 + \sqrt{5}}{2} \simeq 1.628$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)

2) Using Equation 9a and 9b, we find point $c^{(k)}$ and point $d^{(k)}$

$$c^{(k)} = a^{(k)} + \frac{1}{\varphi^2}(b^{(k)} - a^{(k)}) \tag{10a}$$

$$d^{(k)} = a^{(k)} + \frac{1}{\varphi}(b^{(k)} - a^{(k)}) \tag{10b}$$

which are symmetrically placed about the mid-point of I_k

$$\frac{a^{(k)} + b^{(k)}}{2} - c^{(k)} = d^{(k)} - \frac{a^{(k)} + b^{(k)}}{2} \tag{11}$$

Remark

By replacing $c^{(k)}$ and $d^{(k)}$ in Equation 11 and dividing by the common factor $\dfrac{b^{(k)} - a^{(k)}}{\varphi^2}$ we obtain the identity $\varphi^2 - \varphi - 1 = 0$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

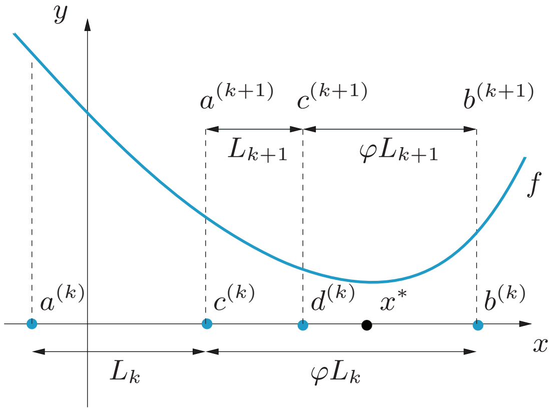
Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)



The generic iteration of the **golden-section method**

- φ is the golden ratio, while $L_k = c^{(k)} - a^{(k)}$

Golden section and quadratic interpolation (cont.)

Set $a^{(0)} = a$ and $b^{(0)} = b$, the golden section method formulates as

Pseudocode

```
For $k = 0, 1, \dots$ until convergence
Compute $c^{(k)}$ and $d^{(k)}$ through Equation 10
If $f(c^{(k)}) \geq f(d^{(k)})$
    set $I_{k+1} = (a^{(k+1)}, b^{(k+1)}) = (c^{(k)}, b^{(k)})$
else
    set $I_{k+1} = (a^{(k+1)}, b^{(k+1)}) = (a^{(k)}, d^{(k)})$
endif
```

It follows that:

- If $I_{k+1} = (c^{(k)}, b^{(k)})$, then $c^{(k+1)} = d^{(k)}$
- if $I_{k+1} = (a^{(k)}, d^{(k)})$, then $d^{(k+1)} = c^{(k)}$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)

A **stopping criterion** can be set when the normalised size of the k-th interval is smaller than a given tolerance ε

$$\frac{b^{(k+1)} - a^{(k+1)}}{|c^{(k+1)}| + |d^{(k+1)}|} < \varepsilon \qquad (12)$$

The mid-point of the last interval I_{k+1} can be taken as an **approximation of the minimiser** $\mathbf{x}^*$

By using Equation 9a and 9b, we obtain the expression

$$|b^{(k+1)} - a^{(k+1)}| = \frac{1}{\varphi}|b^{(k)} - a^{(k)}| = \cdots = \frac{1}{\varphi^{k+1}}|b^{(0)} - a^{(0)}| \qquad (13)$$

The golden-section method converges linearly with rate

$$\varphi^{-1} \simeq 0.618$$

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

```
function [xmin,fmin,iter]=gSection(fun,a,b,tol,kmax,varargin)
%GSECTION finds the minimum of a function
% XMIN=GSECTION(FUN,A,B,TOL,KMAX) approximates a min point of
%  function FUN in [A,B] by using the golden section method
% If the search fails, an error message is returned
% FUN can be i) an inline function, ii) an anonymous function
%  or iii) a function defined in a M-file
% XMIN=GSECTION(FUN,A,B,TOL,KMAX,P1,P2,...) passes parameters
%  P1, P2,... to function FUN(X,P1,P2,...)
% [XMIN,FMIN,ITER]= GSECTION(FUN,...) returns the value of FUN
%  at XMIN and number of iterations ITER done to find XMIN

phi = (1+sqrt(5))/2;
iphi(1) = inv(phi); iphi(2) = inv(1+phi);
c = iphi(2)*(b-a) + a; d = iphi(1)*(b-a) + a;
err = 1+tol; k = 0;

while err > tol & k < kmax
 if(fun(c) >= fun(d))
  a = c; c = d; d = iphi(1)*(b-a) + a;
 else
  b = d; d = c; c = iphi(2)*(b-a) + a;
 end
 k = 1 + k; err = abs(b-a)/(abs(c)+abs(d));
end

xmin = 0.5*(a+b); fmin = fun(xmin); iter = k;
if (iter == kmax & err > tol)
 fprintf('The method stopped after reaching the maximum number
          of iterations, and without meeting the tolerance');
end
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)

- `fun` is either an anonymous or an inline function for function f
- `a` and `b` are endpoints of the search interval
- `tol` is the tolerance ε
- `kmax` is the maximum allowed number of iterations

- `xmin` contains the value of the minimiser
- `fmin` is the minimum value of f in (a, b)
- `iter` is the number of iterations carried out by the algorithm

Golden section and quadratic interpolation (cont.)

Example

Evolution of an isolated culture of 250 bacteria (Verhulst model)

$$f(t) = \frac{2500}{1 + 9 \exp{-t/3}}, \quad \text{for } t > 0$$

where t denotes time (expressed in days)

Find after how many days population growth rate is maximum

- That is, when function $g(t) = -f'(t)$ has its minimum

$$g(t) = -7500 \frac{\exp\left(\frac{t}{3}\right)}{\left(\exp\left(\frac{t}{3}\right) + 9\right)^2}$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)

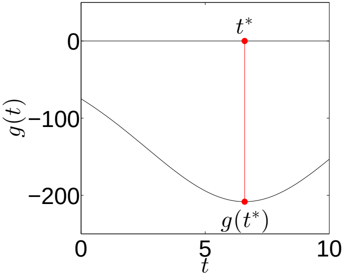
Function $g(t)$ admits a global minimiser in $[6, 7]$, see its plot

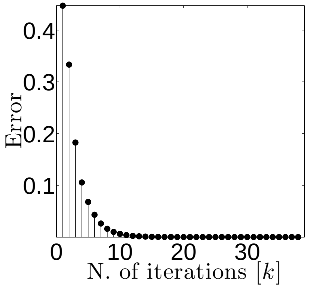
```
g = @(t) [-(7500*exp(t/3)) / (exp(t/3)+9)^2];

a = 0; b = 10;
tol = 1.0e-8; kmax = 100;

[tmin gmin,iter]= gSection(g,a,b,tol,kmax);
```

Golden section: 38 iterations, $t^* \approx 6.59$ and $g(t^*) \approx -208$





Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)

The **quadratic interpolation method** is often used as alternative

- Let f be a continuous and convex function
- Let $x^{(0)}$, $x^{(1)}$ and $x^{(2)}$ be three distinct points

We build a sequence of points $x^{(k)}$ with $k \geq 3$ such that $x^{(k+1)}$ is the vertex (and thus the minimiser) of the parabola $p_2^{(k)}$ that interpolates f at (node points) $x^{(k)}$, $x^{(k-1)}$ and $x^{(k-2)}$

Definition

For $k \geq 2$, the order-2 **Lagrange polynomial** at such nodes is

$$\begin{aligned}p_2^{(k)}(x) = f(x^{(k-2)}) &+ f[x^{(k-2)}, x^{(k-1)}](x - x^{(k-2)}) \\ &+ f[x^{(k-2)}, x^{(k-1)}, x^{(k)}](x - x^{(k-2)})(x - x^{(k-1)}))\end{aligned}$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)

$$
\begin{aligned}
p_2^{(k)}(x) = f(x^{(k-2)}) &+ f[x^{(k-2)}, x^{(k-1)}](x - x^{(k-2)}) \\
&+ f[x^{(k-2)}, x^{(k-1)}, x^{(k)}](x - x^{(k-2)})(x - x^{(k-1)})
\end{aligned}
$$

The **Newton divided differences** are the quantities

$$
\begin{aligned}
f[x_i, x_j] &= \frac{f(x_j) - f(x_i)}{x_j - x_i} \\
f[x_i, x_j, x_k] &= \frac{f[x_j, x_l] - f[x_i, x_j]}{x_l - x_i}
\end{aligned}
\tag{14}
$$

in the 2nd order Lagrange polynomial $p_2^{(k)}$ for $k \geq 2$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Theorem

For $n+1$ distinct points $\{(x_i, y_i(x_i))\}_{n=0}^{n+1}$, there exists only one polynomial $\Pi_n \in \mathbb{P}_n$ of order n or smaller that interpolates them

$$\Pi_n(x_i) = y_i, \quad \forall i = 0, \dots, n$$

If $y_i = f(x_i)$ for some continuous function f, Π_n is said to be the **interpolating polynomial** *of f and it is denoted as $\Pi_n f$*

Definition

Components of the Lagrangian basis associated to nodes $\{x_i\}_{i=0}^{n}$

$$\varphi_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}, \quad i = 0, \dots, n$$

are polynomials such that $\{\varphi_i\}$ is the only basis of $\mathbb{P}_n$ satisfying

$$\varphi_i(x) \in \mathbb{P}_n, \varphi_i(x_j) = \delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Definition

The **Lagrange polynomial** is the interpolating polynomial $\Pi_n(x)$

$$\Pi_n(x) = \sum_{i=0}^{n} y_i \varphi_i(x)$$

expressed in Lagrange form, or with respect to the Lagrange basis

$$\Pi_n(x_i) = \sum_{j=0}^{n} y_j \varphi_j(x_i) = \sum_{j=0}^{n} y_j \delta_{ij} = y_i, \quad i = 0, \dots, n$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)

By solving the first-order equation ${p'}_2^{(k)}(x^{(k+1)}) = 0$, we get

$$x^{(k+1)} = \frac{1}{2}\Big(x^{(k-2)} + x^{(k-1)} - \frac{f[x^{(k-2)}, x^{(k-1)}]}{f[x^{(k-2)}, x^{(k-1)}, x^{(k)}]}\Big) \qquad (15)$$

The next point in the sequence is obtained
by setting to zero the derivative of $p_2^{(k)}(x)$

We iterate until $|x^{(k+1)} - x^k| < \varepsilon$, for some tolerance $\varepsilon > 0$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

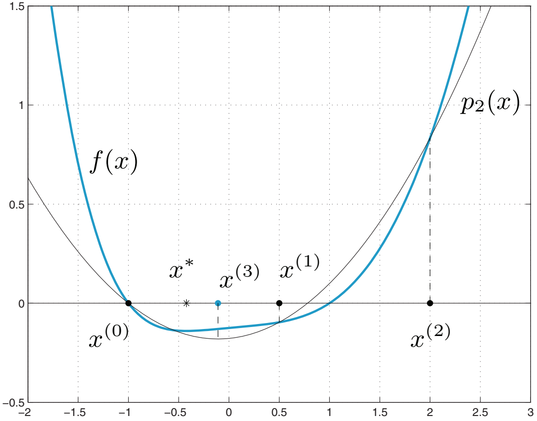
Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)



The first step of the quadratic interpolation method

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Example

$$g(t) = -7500 \frac{\exp\left(\frac{t}{3}\right)}{\left(\exp\left(\frac{t}{3}\right) + 9\right)^2}$$

`fminbnd` combines golden section and parabolic interpolation

```
g = @(t) [-(7500*exp(t/3))/(exp(t/3)+9)^2];

a = 0.0; b = 10.0;
tol = 1.0e-8; kmax = 100;

optionsQ = optimset('TolX', 1.0e-8)
[tminQ,gminQ,exitflagQ,outputQ] = fminbnd(g,a,b,optionsQ);
```

Quadratic interpolation: 8 iter, $t^* \approx 6.59$ and $f(t^*) \approx -208$

- `optimset` sets the tolerance value in structure `optionsQ`
- `qminQ` contains the evaluation of f at the minimiser `tminQ`
- `exitflagQ` indicates the termination state
- `outputQ` has number of iterations and function evaluations

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Golden section and quadratic interpolation (cont.)

The golden section and the quadratic interpolation method are genuinely one-dimensional techniques

- They can be used to solve multidimensional optimisation problems, provided they are restricted to the search of optimisers along a given one dimensional direction

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Nelder and Mead

Derivative-free methods

Nelder and Mead

Let $n > 1$ and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuous function

Definition

The **n-simplex** with $n+1$ vertices $\mathbf{x}_i \in \mathbb{R}^n$ for $i = 0, \dots, n$ is

$$S = \{\mathbf{y} \in \mathbb{R}^n : \mathbf{y} = \sum_{i=0}^{n} \lambda_i \mathbf{x}_i, \text{with } \lambda_i \geq 0 : \sum_{i=0}^{n} \lambda_i = 1\} \qquad (16)$$

Intrinsic assumption: Linearly independent vectors $\{(\mathbf{x}_i - \mathbf{x}_0)\}_{i=1}^{n}$

S is a segment in $\mathbb{R}$, it is a triangle in $\mathbb{R}^2$ and a tetrahedron in $\mathbb{R}^3$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Nelder and Mead (cont.)

The **Nelder and Mead method** is a derivative-free minimisation method that generates a sequence of simplices $\{S^{(k)}\}_{k\geq 0}$ in $\mathbb{R}^n$

- The simplices either run after or circumscribe the minimiser $\mathbf{x}^* \in \mathbb{R}^n$ of the cost function f

The method uses the evaluations of f at the simplices' vertices and geometrical transformations (reflections, expansions, contractions)

- At the k-th iteration, the 'worst' vertice of simplex $S^{(k)}$ is identified as $\mathbf{x}_M^{(k)}$ such that $f(\mathbf{x}_M^{(k)}) = \max_{0\leq i\leq n} f(\mathbf{x}_i^{(k)})$ and then substituted with a new point at which f takes a smaller value
- The new point is obtained by reflecting, expanding or contracting the simplex along the line joining $\mathbf{x}_M^{(k)}$ with the centroid of the other vertices of the simplex

$$\mathbf{x}_c^{(k)} = \frac{1}{n}\sum_{\substack{i=0\\ i\neq M}}^{n} \mathbf{x}_i^{(k)}$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Nelder and Mead (cont.)

To generate the initial simplex $S^{(0)}$, we take a point $\tilde{\mathbf{x}} \in \mathbb{R}^n$ and a positive real number η and we set $\mathbf{x}_i^{(0)} = \tilde{\mathbf{x}} + \eta \mathbf{e}_i$ with $i = 1, \dots, n$

- $\{\mathbf{e}_i\}$ are the vectors of the standard basis in $\mathbb{R}^n$

While $k \geq 0$ and until convergence, select the 'worst' vertex of $S^{(k)}$

$$\mathbf{x}_M^{(k)} = \max_{0 \leq i \leq n} f(\mathbf{x}_i^{(k)}) \tag{17}$$

and then replace it by a new point to form the new simplex $S^{(k+1)}$

Nelder and Mead (cont.)

The new point is chosen by firstly selecting

$$\begin{aligned} \mathbf{x}_m^{(k)} &= \min_{0 \leq i \leq n} f(\mathbf{x}_i^{(k)}) \\ \mathbf{x}_\mu^{(k)} &= \max_a f(\mathbf{x}_i^{(k)}) \end{aligned} \tag{18}$$

and secondly by defining the **centroid** point

$$\overline{\mathbf{x}}^{(k)} = \frac{1}{n} \sum_{\substack{i=0 \\ i \neq M}}^{n} \mathbf{x}_i^{(k)} \tag{19}$$

This is the centroid point of hyperplane $H^{(k)}$ passing through the vertices $\{\mathbf{x}_i\}_{\substack{i=0 \\ i \neq M}}^{n}$

Nelder and Mead (cont.)

Thirdly, compute reflection $\mathbf{x}_\alpha^{(k)}$ of $\mathbf{x}_M^{(k)}$ wrt hyperplane $H^{(k)}$

$$\mathbf{x}_\alpha^{(k)} = (1-\alpha)\overline{\mathbf{x}}^{(k)} + \alpha \mathbf{x}_M^{(\alpha)} \tag{20}$$

with **reflection coefficient** $\alpha < 0$ is typically set to be -1

Point $\mathbf{x}_\alpha^{(k)}$ lies on the straight line joining points $\overline{\mathbf{x}}^{(k)}$ and $\mathbf{x}_M^{(k)}$

- It is on the side of $\overline{\mathbf{x}}^{(k)}$ far from $\mathbf{x}_M^{(k)}$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

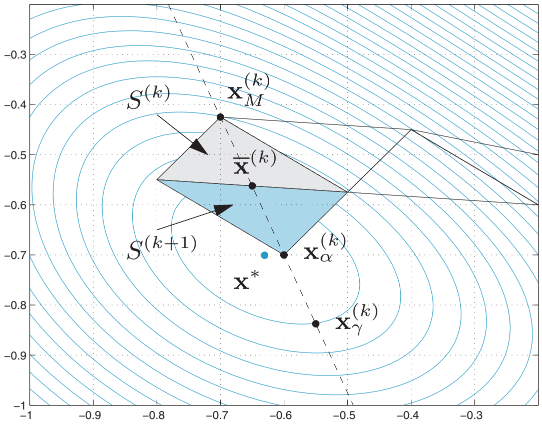
Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Nelder and Mead (cont.)



$n = 2$, the centroid is midpoint of edge of $S^{(k)}$ opposite to $\mathbf{x}_M^{(k)}$

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Nelder and Mead (cont.)

We fourthly compare $f(\mathbf{x}_\alpha^{(k)})$ with the values of f at the other vertices of the simplex before accepting $\mathbf{x}_\alpha^{(k)}$ as the new vertex

We also try to move $\mathbf{x}_\alpha^{(k)}$ on the straight line joining $\overline{\mathbf{x}}^{(k)}$ and $\mathbf{x}_M^{(k)}$ to set the new simplex $S^{(k+1)}$, as follows:

- If $f(\mathbf{x}_\alpha^{(k)}) < f(\mathbf{x}_m^{(k)})$ (reflection produced a minimum), then

$$\mathbf{x}_\gamma^{(k)} = (1-\gamma)\overline{\mathbf{x}}^{(k)} + \gamma \mathbf{x}_M^{(k)}, \quad \text{with } \gamma < -1^{1} \qquad (21)$$

  then, if $f(\mathbf{x}_\gamma^{(k)}) < f(\mathbf{x}_m^{(k)})$, replace $\mathbf{x}_M$ by $\mathbf{x}_\gamma^{(k)}$, otherwise $\mathbf{x}_M^{(k)}$ is replaced by $\mathbf{x}_\alpha^{(k)}$
  We then proceed by incrementing k by one

[1] with typically $\gamma = -2$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Nelder and Mead (cont.)

- If $f(\mathbf{x}_m^{(k)}) \leq f(\mathbf{x}_\alpha^{(k)}) < f(\mathbf{x}_\mu^{(k)})$, then $\mathbf{x}_M^{(k)}$ is replaced by $\mathbf{x}_\alpha^{(k)}$ and k is incremented by one
- If $f(\mathbf{x}_\mu^{(k)}) \leq f(\mathbf{x}_\alpha^{(k)})) < f(\mathbf{x}_M^{(k)})$, we compute

$$\mathbf{x}_\beta^{(k)} = (1-\beta)\overline{\mathbf{x}}^{(k)} + \beta \mathbf{x}_\alpha^{(k)}, \quad \text{with } \beta > 0^2 \tag{22}$$

then, if $f(\mathbf{x}_\beta^{(k)}) > f(\mathbf{x}_M^{(k)})$ define the vertices $S^{(k+1)}$ by

$$\mathbf{x}_i^{(k+1)} = \frac{1}{2}(\overline{\mathbf{x}}^{(k)} + \mathbf{x}_m^{(k)}) \tag{23}$$

otherwise $\mathbf{x}_M^{(k)}$ is replaced by $\mathbf{x}_\beta$
Then, we increment k

[2] with typically $\beta = 1/2$

Nelder and Mead (cont.)

- If $f(\mathbf{x}_\alpha^{(k)}) > f(\mathbf{x}_M^{(k)})$, we compute

$$\mathbf{x}_\beta = (1-\beta)\overline{\mathbf{x}}^{(k)} + \beta \mathbf{x}_M^{(k)}, \quad \text{with } \beta > 0 \tag{24}$$

  and if $f(\mathbf{x}_\beta^{(k)}) > f(\mathbf{x}_M^{(k)})$ define the vertices of $S^{(k+1)}$ by Equation 23, otherwise we replace $\mathbf{x}_M^{(k)}$ with $\mathbf{x}_\beta^{(k)}$
  Then we increment k

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Nelder and Mead (cont.)

When the stopping criterion $\max_{i=0,\dots,n} ||\mathbf{x}_i^{(k)} - \mathbf{x}_m^{(k)}||_\infty < \varepsilon$ is met, $\mathbf{x}_m^{(k)}$ is retained as approximation of the minimiser

Convergence is guaranteed in very special cases only

Stagnation may occur, algorithm needs to be restarted

- The algorithm is nevertheless quite robust and efficient for small dimensional problems
- Its rate of convergence is severely affected by the choice of the initials simplex

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Nelder and Mead (cont.)

Example

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

The function is the **The Rosenbrock function**, it is often used as testbench for efficiency and robustness of minimisation algos

- The global minimum is at $\mathbf{x}^* = (1, 1)$, however its variation around $\mathbf{x}^*$ is low, making algorithms' convergence problematic

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

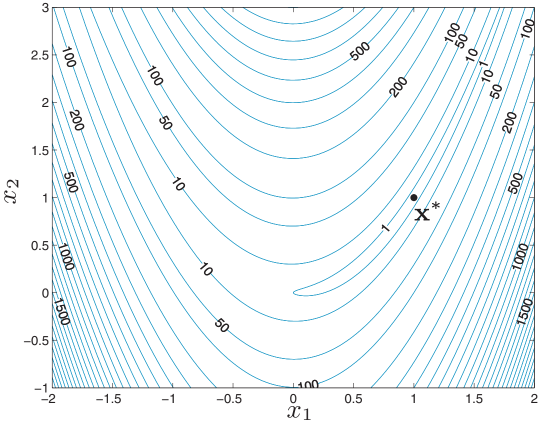
Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Nelder and Mead (cont.)



Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Nelder and Mead (cont.)

The simplex method: The M-command is `fminsearch`

```
x_0 = [-1.2,+1.0];

fun = @(x) (1-x(1))^2 + 100*(x(2)-x(1)^2)^2;

xstar = fminsearch(fun,x_0)

xstar =
1.000022021783570 1.000042219751772
```

To obtain additional information on the minimum value of f, we can replace the second instruction with the expanded one

```
[xstar,fval,exitflag,output] = fminsearch(fun,x_0)
```

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

The Newton method

Numerical optimisation

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

The Newton method

Assume $f : \mathbb{R}^n \to \mathbb{R}$ with $n \geq 1$ of class $\mathbb{C}^2(\mathbb{R}^n)$ and we know how to compute its first and second order partial derivatives

We can apply Newton's method for the solution of the system $\mathbf{F}(x) = \nabla f(\mathbf{x}) = \mathbf{0}$, whose Jacobian matrix $\mathbf{J}_\mathbf{F}(\mathbf{x}^{(k)})$ is the Hessian matrix of f computed at the generic iteration point $\mathbf{x}^{(k)}$

Pseudocode

```
Given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, for $k = 0, 1, \dots$ until convergence
    Solve  $\underbrace{\mathbf{H}(\mathbf{x}^{(k)})}_{\mathbf{J}_{\mathbf{F}(\mathbf{x}^{(k)})}} \boldsymbol{\delta}\mathbf{x}^{(k)} = -\underbrace{\nabla f(\mathbf{x}^{(k)})}_{\mathbf{F}(\mathbf{x}^{(k)})}$      (25)
    Set    $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}$
```

A suitable stopping test is $||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|| \leq \varepsilon$, $\varepsilon > 0$ the tolerance

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

The Newton method (cont.)

Remark

Consider the problem of finding the zero of $f : [a, b] \subset \mathbb{R} \rightarrow \mathbb{R}$

$$\text{Find } \alpha \in [a, b] \text{ such that } f(\alpha) = 0$$

Given the tangent to the function $(x, f(x))$ at some point $x^{(k)}$

$$y(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)})$$

and resolving for a point $x^{(k+1)}$ such that $y(x^{(k+1)}) = 0$, we get

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \quad \text{for } k \geq 0 \text{ and } f'(x^{(k)}) \neq 0$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

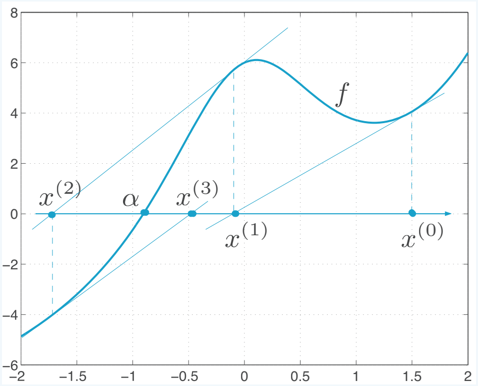
The Gauss-Newton method

Levenberg-Marquardt

The Newton method (cont.)

Remark

The sequence is the **Newton's method** for finding the zero of a function, and it reduces to locally substituting f with its tangent



Remark

Consider the following set of nonlinear equations

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ \quad \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases}$$

Let $\mathbf{f} \equiv (f_1, \dots, f_n)^T$ and $\mathbf{x} \equiv (x_1, \dots, x_n)^T$ to get $\mathbf{f}(\mathbf{x}) = \mathbf{0}$

To extend the Newton's method we replace the first derivative of scalar function f with the Jacobian matrix $\mathbf{J_f}$ of vectorial function $\mathbf{f}$

$$(\mathbf{J_f})_{ij} \equiv \frac{\partial f_i}{\partial x_j}, \quad \text{with } i, j = 1, \dots, n$$

Pseudocode

```
Solve   $\mathbf{J_f}(\mathbf{x}^{(k)})\boldsymbol{\delta}\mathbf{x}^{(k)} = -\mathbf{f}(\mathbf{x}^{(k)})$
Set     $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}$
```

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

The Newton method (cont.)

```
function [x,res,iter] = sNWT(F_fun,J_fun,x_0,tol,imx,varargin)
%SNLENEWTON Approximates a root of a nonlinear system
% [ROOT,RES,ITER]=NLSE(F_FUN,J_FUN,X_0,TOL,IMX) Calculate
%  vector ROOT, the zero of a nonlinear system defined in
%  F_FUN with Jacobian J_FUN, from initial point X_0
%
% RES is residual in ROOT and ITER is number of iterations
% F_FUN e J_FUN are external functions (as M-files)

iter = 0; err = 1 + tol; x = x_0;

while err >= tol & iter < imx
 J = J_fun(x,varargin{:});
 F = F_fun(x,varargin{:});
 delta = -J\F;
 x = x + delta;
 err = norm(delta); iter = 1 + iter;
end

res = norm(F_fun(x,varargin{:}));

if(iter==imx & err > tol)
 disp('[Out by KMAX]');
else
 disp('[Out by TOL]'));
end
return
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

The Newton method (cont.)

```
function F = F_fun(x)
 F(1,1) = F_1(x_1,x_2,...); % Add your own expression
 F(2,1) = F_2(x_1,x_2,...); % Add your own expression
 ...
 F(N,1) = F_N(x_1,x_2,...); % Add your own expression

return
```

```
function J = J_fun(x)
 J(1,1) = dF_1 / dx_1; % Add your own expression
 J(1,2) = dF_1 / dx_2; % Add your own expression
 ...

 J(2,1) = dF_2 / dx_1; % Add your own expression
 J(2,2) = dF_2 / dx_2; % Add your own expression
 ...

 J(N,1) = dF_N / dx_1; % Add your own expression
 J(N,2) = dF_N / dx_2; % Add your own expression
 ...

 return
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

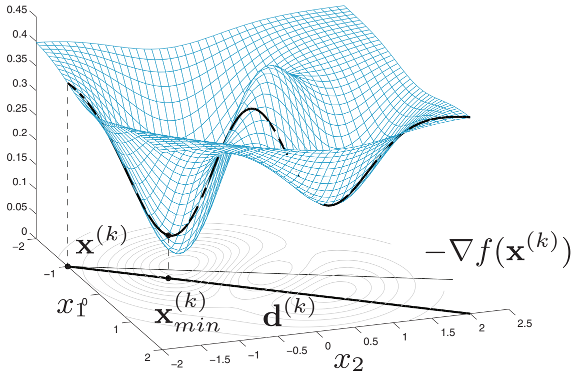
The Gauss-Newton method

Levenberg-Marquardt

Example

$$f(\mathbf{x}) = \frac{2}{5} - \frac{1}{10}(5x_1^2 + 5x_2^2 + 3x_1x_2 - x_1 - 2x_2)\exp\big(-(x_1^2 + x_2^2)\big)$$

We want to approximate the global minimum $\mathbf{x}^* \approx (-0.63, -0.70)$



Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

The Newton method (cont.)

Netwon's method with a tolerance $\varepsilon = 10^{-5}$

- If we choose $\mathbf{x}^{(0)} = (-0.9, -0.9)$, then after 5 iterations the method converges to `x=[-0.63058;-0.70074]`
- If we choose $\mathbf{x}^{(0)} = (-1.0, -1.0)$, then after 400 iterations the stop criterion still would not be fulfilled

Moreover, Newton's method may converge to any stationary point (not necessarily to a minimiser)

- With $\mathbf{x}^{(0)} = (+0.5, -0.5)$, after 5 iterations the method converges to the saddle point `x=[0.80659; -0.54010]`

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

The Newton method (cont.)

Remark

A necessary condition for convergence of Newton's method is that $\mathbf{x}^{(0)}$ should be sufficiently close to the minimiser $\mathbf{x}^*$

- Reflects the **local convergence property** of the method

Remark

General convergence criterium for the Newton's method

If $f \in \mathbb{C}^2(\mathbb{R}^n)$ with stationary point $\mathbf{x}^*$, with positive definite Hessian $\mathbf{H}(\mathbf{x}^*)$, with Lipschitz continuous components of $\mathbf{H}(\mathbf{x})$ in a neighbourhood of $\mathbf{x}^*$ and for $\mathbf{x}^{(0)}$ sufficiently close to $\mathbf{x}^*$, then the method converges quadratically to $\mathbf{x}^*$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

The Newton method (cont.)

In spite of a simple implementation, the method is demanding when n is large, as it requires the analytic expression of the derivatives and, at each iteration, the computation of both gradient and Hessian of f

- Let alone that $\mathbf{x}^{(0)}$ has to be chosen near enough $\mathbf{x}^*$

Remark

To design efficient and robust minimisation algorithms combine locally with globally convergent methods

- **Global convergence** guarantees convergence to a stationary point (not necessarily a global minimiser) for all $\mathbf{x}^{(0)} \in \mathbb{R}^n$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Line-search methods

Numerical optimisation

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Line-search methods

For simplicity, assume $f \in \mathbb{C}^2(\mathbb{R})$ and bounded from below

Line-search or **descent methods** are iterative methods

- For every step $k \geq 0$, point $\mathbf{x}^{(k+1)}$ depends on point $\mathbf{x}^k$, on a vector $\mathbf{d}^{(k)}$ which in turn depends on the gradient $\nabla f(\mathbf{x}^{(k)}))$ of f, and on a suitable step-length parameter $\alpha_k \in \mathbb{R}$

Given an initial minimiser $\mathbf{x}^{(0)} \in \mathbb{R}^n$, the method formulates as

Pseudocode

```
Find direction $\mathbf{d}^{(k)} \in \mathbb{R}^n$
Compute step $\alpha_k \in \mathbb{R}$
Set $\mathbf{x}^{(\mathbf{k+1})} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Line-search methods (cont.)

Definition

Vector $\mathbf{d}^{(k)}$ must be a **descent direction**, so it satisfies conditions

$$\begin{aligned} \mathbf{d}^{(k)}\nabla f(\mathbf{x}^{(k)}) < 0, &\quad \text{if } \nabla f(\mathbf{x}^{(k)}) \neq \mathbf{0} \\ \mathbf{d}^{(k)} = \mathbf{0}, &\quad \text{if } \nabla f(\mathbf{x}^{(k)}) = \mathbf{0} \end{aligned} \tag{26}$$

In $\mathbb{R}^n$, the gradient $\nabla f(\mathbf{x}^{(k)})$ identifies the direction with sign of maximum positive growth of f from $\mathbf{x}^{(k)}$

As $\mathbf{d}^{(k)}\nabla f(\mathbf{x}^{(k)})$ is the directional derivative of f along $\mathbf{d}^{(k)}$

- First condition ensures that we move along the opposite direction of the gradient (towards the minimiser)

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

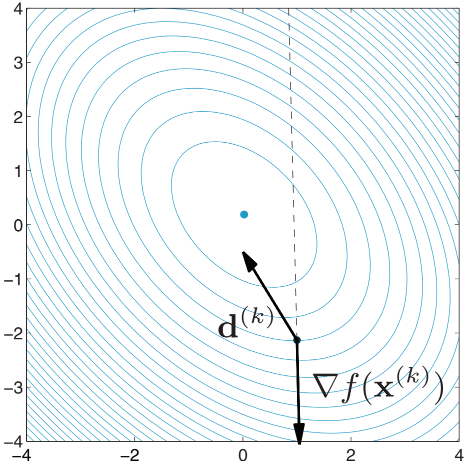
Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Contour lines of function $f(\mathbf{x})$ and its gradient evaluated at $\mathbf{x}^{(k)}$

- $\mathbf{d}^{(k)}$ is a suitable descent direction



Once $\mathbf{d}^{(k)}$ is determined, the optimal value $\alpha_k \in \mathbb{R}$ is the one that guarantees maximum variation of f along $\mathbf{d}^{(k)}$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

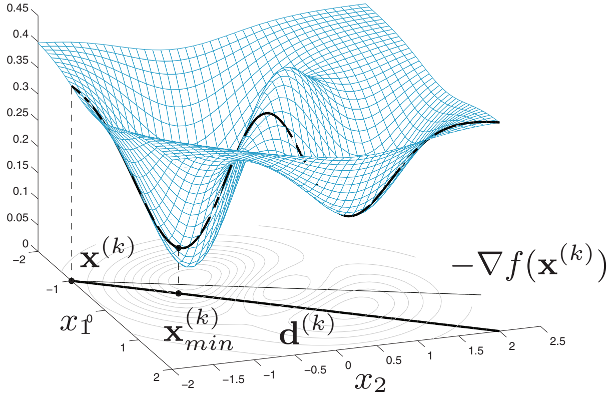
Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

α_k can be computed by solving a one-dimensional minimisation

- Minimise the restriction of $f(\mathbf{x})$ along $\mathbf{d}^{(k)}$
- $\mathbf{x}_{\text{min}}^{(k)}$ is the minimiser along $\mathbf{d}^{(k)}$



The computation of α_k is quite involved when f is not quadratic

- There are alternative techniques aimed at approximating α_k

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Descent directions

Line-search methods

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Descent directions

- **Newton's directions**

$$\mathbf{d}^{(k)} = -\mathbf{H}^{-1}(\mathbf{x}^{(k)})\nabla f(\mathbf{x}^{(k)}) \tag{27}$$

Matrix $\mathbf{H}(\mathbf{x}^{(k)})$ is the Hessian matrix at the k-th step

- **Quasi-Newton directions**

$$\mathbf{d}^{(k)} = -\mathbf{H}_k^{-1}\nabla f(\mathbf{x}^{(k)}) \tag{28}$$

Matrix $\mathbf{H}_k$ is an approximation of the true Hessian $\mathbf{H}(\mathbf{x}^{(k)})$, it is used when second derivatives are heavy to compute

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Descent directions (cont.)

- **Gradient directions**

$$\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)}) \tag{29}$$

These are quasi-Newton directions, with $\mathbf{H}_k = \mathbf{I}$, $\forall k \geq 0$

- **Conjugate-gradient directions**

$$\begin{aligned} \mathbf{d}^{(0)} &= -\nabla f(\mathbf{x}^{(0)}) \\ \mathbf{d}^{(k+1)} &= -\nabla f(\mathbf{x}^{(k+1)}) + \beta_k \mathbf{d}^{(k)}, \quad k \geq 0 \end{aligned} \tag{30}$$

Coefficients β_k can be chosen according to different criteria
When f is quadratic, the descent directions correspond to those of the conjugate-gradient method for linear systems

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Descent directions (cont.)

For all $k \geq 0$, gradient directions are valid descent directions

$$\begin{aligned} \mathbf{d}^{(k)} \nabla f(\mathbf{x}^{(k)}) < 0, &\quad \text{if } \nabla f(\mathbf{x}^{(k)}) \neq \mathbf{0} \\ \mathbf{d}^{(k)} = \mathbf{0}, &\quad \text{if } \nabla f(\mathbf{x}^{(k)}) = \mathbf{0}, \end{aligned} \tag{31}$$

Newton's and quasi Newton's directions can be valid directions

- Only if $\mathbf{H}(\mathbf{x}^{(k)})$ and $\mathbf{H}_k$ are positive definite matrices

Conjugate gradient directions are valid for suitable β_k

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

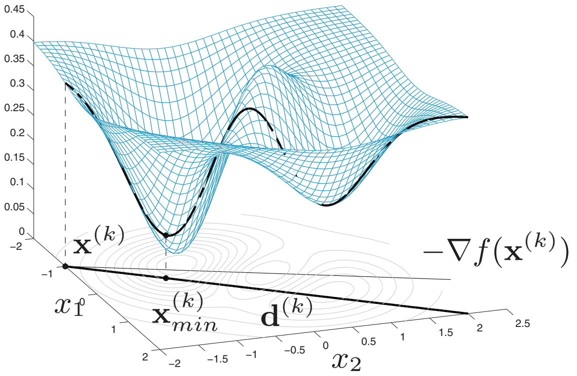
The Gauss-Newton method

Levenberg-Marquardt

Example

$$f(\mathbf{x}) = \frac{2}{5} - \frac{1}{10}(5x_1^2 + 5x_2^2 + 3x_1x_2 - x_1 - 2x_2)\exp\left(-(x_1^2 + x_2^2)\right)$$

Two local minimisers, one local maximiser and two saddle points



We compare sequences $\{\mathbf{x}^{(k)}\}$ generated by Newton's method and descent methods with various descent directions, from $\mathbf{x}_1^{(0)}$ and $\mathbf{x}_2^{(0)}$

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

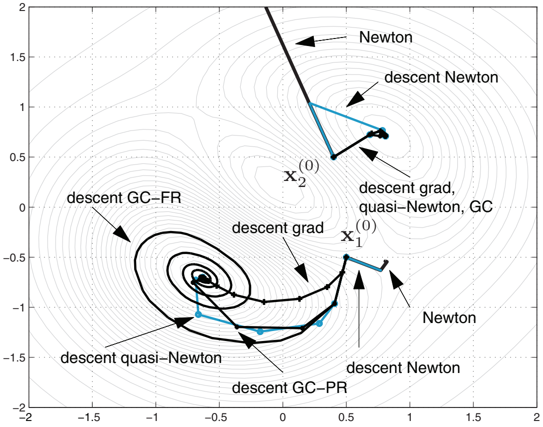
The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

From $\mathbf{x}_1^{(0)} = (0.5, 0.5)$



- Newton's method converges rapidly towards the saddle point
- Newton's directions take a first step identical to Newton's and then collapses due to a non-positive definite matrix $\mathbf{H}_k$
- The others converge with different speeds into a local minimum, fastest convergence by quasi-Newton's directions

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

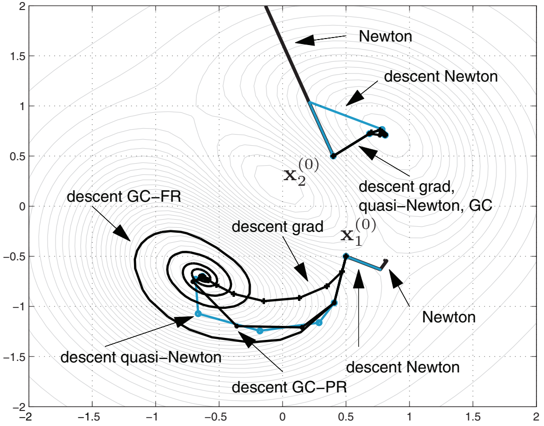
Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

From $\mathbf{x}_2^{(0)} = (0.4, 0.5)$



- Newton's method diverges, Newton's directions converge to a local minimum, though sharing the same first direction with it
- All others also converge to the same local minimiser

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Step-length α_k

Line-search methods

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Step-length α_k

Given a descent direction $\mathbf{d}^{(k)}$, step-length α_k has to be set st the new iterate $\mathbf{x}^{(k+1)}$ is (approximates) the minimiser of f along $\mathbf{d}^{(k)}$

Choose α_k such that the minimisation is exact

$$\begin{gathered}\alpha_k = \arg\min_{\alpha \in \mathbb{R}} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) \\ \text{or} \\ f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) = \min_{\alpha \in \mathbb{R}} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})\end{gathered} \tag{32}$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Step-length α_k (cont.)

A second-order Taylor expansion of f around $\mathbf{x}^{(k)}$ yields

$$\begin{aligned} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) = f(\mathbf{x}^{(k)}) &+ \alpha \mathbf{d}^{(k)} \nabla f(\mathbf{x}^{(k)}) \\ &+ \frac{\alpha^2}{2} \mathbf{d}^{(k)^T} \mathbf{H}(\mathbf{x}^{(k)}) \mathbf{d}^{(k)} \\ &+ o(||\alpha \mathbf{d}^{(k)}||^2) \end{aligned} \tag{33}$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Step-length α_k (cont.)

Remark

In the special case in which f is a quadratic function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b} + c$$

with $\mathbf{A} \in \mathbb{R}^{n \times n}$ symmetric and positive definite, $\mathbf{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$, expansion in Eq. 33 is exact and the infinitesimal residual is null

As $\mathbf{H}^{(k)} = \mathbf{A}$ for every $k \geq 0$ and $\nabla f(\mathbf{x}^{(k)}) = \mathbf{A}\mathbf{x}^{(k)} - \mathbf{b} = -\mathbf{r}^{(k)}$, by differentiating Eq. 33 wrt α and setting the derivative to zero

$$\alpha_k = \frac{\mathbf{d}^{(k)^T}\mathbf{r}^{(k)}}{\mathbf{d}^{(k)^T}\mathbf{A}\mathbf{d}^{(k)}} \qquad (34)$$

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Step-length α_k (cont.)

For gradient directions $\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$, we find $\mathbf{d}^{(k)} = \mathbf{r}^{(k)}$

- We obtain the gradient method for solving linear systems

Should direction $\mathbf{d}^k$ be chosen according to the conjugate-gradient, by setting

$$\beta_k = \frac{(\mathbf{A}\mathbf{d}^{(k)})^T \mathbf{r}^{(k+1)}}{\mathbf{d}^{(k)^T} \mathbf{A}\mathbf{d}^{(k)}} \text{ or } \beta_k = \frac{\mathbf{d}^{(k)^T} \mathbf{A}\mathbf{r}^{(k+1)}}{\mathbf{d}^{(k)^T} \mathbf{A}\mathbf{d}^{(k)}} \qquad (35)$$

we recover the conjugate-gradient for solving linear systems

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Step-length α_k (cont.)

If f is a non-quadratic function, the computation of the optimal α_k requires an iterative method to solve the minimisation along $\mathbf{d}^{(k)}$

Remark

- Demanding and not worth it, stick with an approxinmation

An approximated value of α_k can be chosen by requiring that the new iterate $\mathbf{x}^{(\mathbf{k+1})} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$ ensures that

$$f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)}) \tag{36}$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Step-length α_k (cont.)

Example

A natural strategy would be to initially assign a large α_k and reduce it iteratively until $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$ is satisfied

- Unfortunately, the strategy does not guarantee a $\{\mathbf{x}^k\}$ that converges to the desired minimiser $\mathbf{x}^*$

A better criterium for $\alpha_k > 0$ is based on **Wolfe's conditions**

Step-length α_k (cont.)

Definition

$$\begin{aligned} f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) &\leq f(\mathbf{x}^{(k)}) + \sigma \alpha_k \mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)}) \\ \mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) &\geq \delta \mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)}) \end{aligned} \tag{37}$$

The two given constants σ and δ are such that $0 < \sigma < \delta < 1$

$\mathbf{d}^{(k)} \nabla f(\mathbf{x}^{(k)})$ is the directional derivative of f along direction $\mathbf{d}^k$

- First condition (**Armijo's rule**) inhibits too small variations of f with respect to step-length and directional derivative
- This is practically obtained by requiring changes in f to be proportional to both step-length and directional derivative

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Step-length α_k (cont.)

Definition

$$f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) \leq f(\mathbf{x}^{(k)}) + \sigma \alpha_k {\mathbf{d}^{(k)}}^T \nabla f(\mathbf{x}^{(k)})$$

$${\mathbf{d}^{(k)}}^T \nabla f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) \geq \delta {\mathbf{d}^{(k)}}^T \nabla f(\mathbf{x}^{(k)})$$

The two given constants σ and δ are such that $0 < \sigma < \delta < 1$

$\mathbf{d}^{(k)} \nabla f(\mathbf{x}^{(k)})$ is the directional derivative of f along direction $\mathbf{d}^k$

- Second condition states that at new point $\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$ the value of the directional derivative of f should be δ times larger than the same derivative at previous point $\mathbf{x}^{(k)}$
- Point $\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$ is a valid candidate if f at such point decreases less than it does at $\mathbf{x}^{(k)}$ (closer to a minimiser)

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

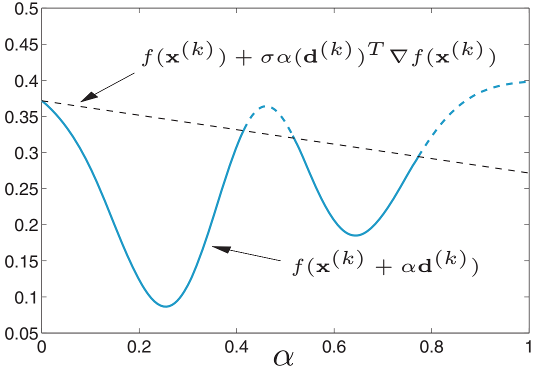
Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Step-length α_k (cont.)

The terms in the first of the two Wolfe's conditions, for $\sigma = 0.2$

$$f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) \leq f(\mathbf{x}^{(k)}) + \sigma \alpha_k \mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)})$$



Condition is satisfied for α corresponding to the continuous line

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

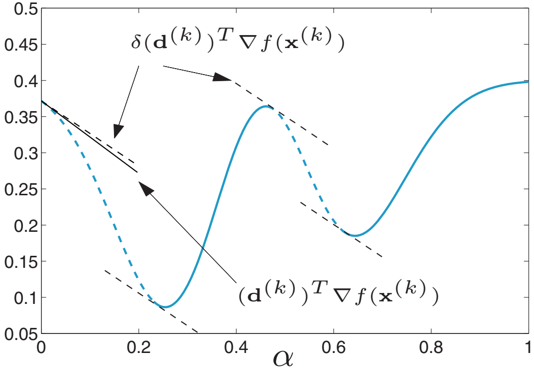
The Gauss-Newton method

Levenberg-Marquardt

Step-length α_k (cont.)

Lines with slope $\delta \mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)})$ in second condition, $\delta = 0.9$

$$\mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) \geq \delta \mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)})$$



Condition is satisfied for α corresponding to the continuous line

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Step-length α_k (cont.)

Wolfe's conditions are jointly satisfied in the interval

$$0.23 \leq \alpha \leq 0.41 \text{ or } 0.62 \leq \alpha \leq 0.77$$

That is, also far from the minimiser of f along $\mathbf{d}^{(k)}$

- Or when the directional derivative is large

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Step-length α_k (cont.)

Definition

Wolfe's strong conditions: More restrictive conditions

$$\begin{aligned} f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) &\leq f(\mathbf{x}^{(k)}) + \sigma \alpha_k {\mathbf{d}^{(k)}}^T \nabla f(\mathbf{x}^{(k)}) \\ |{\mathbf{d}^{(k)}}^T \nabla f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)})| &\leq -\delta {\mathbf{d}^{(k)}}^T \nabla f(\mathbf{x}^{(k)}) \end{aligned} \tag{38}$$

The first condition is unchanged, the second one inhibits f from large variations about $\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

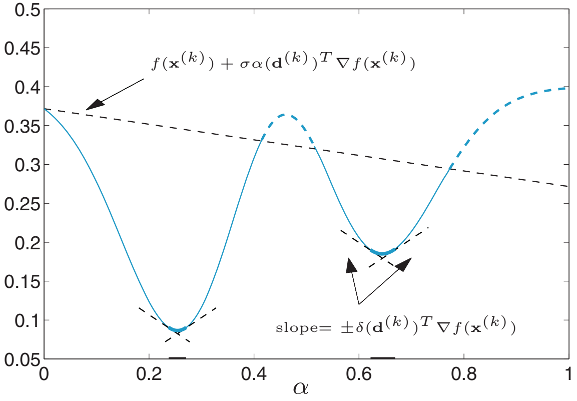
The Gauss-Newton method

Levenberg-Marquardt

Step-length α_k (cont.)

Wolfe's strong conditions are satisfied when α belongs to the small intervals around the minimisers (thick continuous arcs)

- For $\sigma = 0.2$ and $\delta = 0.9$



Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Step-length α_k (cont.)

Remark

It can be shown that if $f \in \mathbb{C}^2(\mathbb{R}^n)$ is bounded from below in $\{\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}, \alpha > 0\}$ with $\mathbf{d}^{(k)}$ a descent direction at $\mathbf{x}^{(k)}$, then for all σ and δ st $0 < \sigma < \delta < 1$ there exist non-empty intervals of α_k that satisfy Wolfe's weak and strong conditions

In practice[3], σ is usually chosen to be very small (e.g., $\sigma = 10^4$), while typical values for δ are $\delta = 0.9$ for Newton, quasi-Newton and gradient directions, and $\delta = 0.1$ for CG directions

[3] J. Nocedal and S. Wrigth (2006): *Numerical optimization*.

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Step-length α_k (cont.)

A strategy for step lengths α_k satisfying Wolfe's conditions

- **Backtracking**: Start with $\alpha = 1$ and then reduce it by a given factor ρ (tipically, $\rho \in [0.1, 0.5)$) until the first condition is satisfied

For $\mathbf{x}^{(k)}$ and a direction $\mathbf{d}^{(k)}$, for $\sigma \in (0, 1)$ and $\rho \in [0.1, 0.5)$

Pseudocode

```
Set $\alpha = 1$
    while $f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) > f(\mathbf{x}^{(k)}) + \sigma \alpha \mathbf{d}^{(k)} \nabla f(\mathbf{x}^{(k)})$
        $\alpha = \rho \alpha$
    end
Set $\alpha_k = \alpha$
```

Second condition is never checked: Step lengths are not small

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Step-length α_k (cont.)

```
function [x,alpha_k]= bTrack(fun,x_k,g_k,d_k,varargin)
%BTRACK Backtracking with line search
% [X,ALPHA_K]=BTRACK(FUN,X_K,G_K,D_K) x_{k+1}=x_k+alpha_k*d_k
%  in the descent method, alpha_k by backtracking with
%  sigma=1e-4 and rho=0.25
%
% [X,ALPHA_K]=BTRACK(FUN,X_K,G_K,D_K,SIGMA,RHO) sigma and rho
%  can be inputed - sigma in (1e-4,0.1) and rho in (0.1,0.5)
%
% FUN is the function handle of the objective function
% X_K is element x_k, G_K is the gradient, D_K is d_k

if nargin == 4
 sigma = 1.0e-4; rho = 1/4;
else
 sigma = varargin {1}; rho = varargin {2};
end

minAlpha = 1.0e-5; % Smallest steplength
alpha_k = 1.0; f_k = fun(x_k);

k = 0; x = x_k + alpha_k*d_k;
while fun(x) > f_k + sigma*alpha_k*g_k'*d_k & alpha_k >
     minAlpha
 alpha_k = alpha_k*rho;
 x = x_k + alpha_k*d_k; k = k+1;
end
```

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Step-length α_k (cont.)

The descent method with various descent directions

- α_k is determined by backtracking

```
%DSCENT Descent method of minimisation
%[X,ERR,ITER]=DSCENT(FUN,GRAD_FUN,X_0,TOL,KMAX,TYP,HESS_FUN)
% Approximates the minimiser of FUN using descent directions
%  Newton (TYP=1), BFGS (TYP=2), GRADIENT (TYP=3), and the
%  CONJUGATE-GRADIENT method with
%   beta_k by Fletcher and Reeves (TYP=41)
%   beta_k by Polak and Ribiere   (TYP=42)
%   beta_k by Hestenes and Stiefel(TYP=43)
%
% Step length is calculated using backtracking (bTrack.m)
%
% FUN, GRAD_FUN and HESS_FUN (TYP=1 only) are function handles
%  for the objective, gradient and Hessian matrix
%  With TYP=2, HESS_FUN approximates the exact Hessian at X_0
%
% TOL is the stop check tolerance
% KMAX is the maximum number of iteration
```

```
function [x,err,iter]=dScent(fun,grad_fun,x_0,tol,kmax,typ,
     varargin)
if nargin>6; if typ==1; hess=varargin{1};
   elseif typ==2; H=varargin{1}; end; end

err=tol+1; k=0; xk=x0(:); gk=grad(xk); dk=-gk; eps2=sqrt(eps);

while err>tol & k<kmax
 if typ==1; H = hess_fun(xk); dk = -H\gk;                % Newton
 elseif typ==2; dk = -H\gk;                                % BFGS
 elseif typ==3; dk = -gk;                              % Gradient
 end
 [xk1,alphak]=bTrack(fun,xk,gk,dk);
 gk1=grad_fun(xk1);
  if typ==2                                        % BFGS update
   yk = gk1-gk; sk = xk1-xk; yks = yk'*sk;
   if yks > eps2*norm(sk)*norm(yk)
    Hs=H*sk; H=H+(yk*yk')/yks-(Hs*Hs')/(sk'*Hs);
   end
  elseif typ>=40                                    % CG upgrade
   if typ==41; betak=(gk1'*gk1)/(gk'*gk);                    % FR
   elseif typ==42; betak=(gk1'*(gk1-gk))/(gk'*gk);           % PR
   elseif typ==43; betak=(gk1'*(gk1-gk))/(dk'*(gk1-gk)); % HS
   end
   dk = -gk1 + betak*dk;
  end
  xk = xk1; gk = gk1; k = 1 + k; xkt = xk1;
 for i=1:length(xk1); xkt(i) = max([abs(xk1(i)),1]); end
 err = norm((gk1.*xkt)/max([abs(fun(xk1)),1]),Inf);
end
x = xk; iter = k;
if (k==kmax & err>tol); disp('[KMAX]'); end
```

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Descent method with Newton's directions

Line-search methods

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Descent method with Newton's directions

A $f \in \mathbb{C}^2(\mathbb{R}^n)$ bounded from below and the descent method

Pseudocode

```
Find direction $\mathbf{d}^{(k)} \in \mathbb{R}^n$
Compute step $\alpha_k \in \mathbb{R}$
Set $\mathbf{x}^{(\mathbf{k+1})} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
```

- Newton directions $\mathbf{d}^{(k)} = -\mathbf{H}^{-1}(\mathbf{x}^{(k)}) \nabla f(\mathbf{x}^{(k)})$
- Wolfe step lengths α_k

$$f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) \leq f(\mathbf{x}^{(k)}) + \sigma \alpha_k \mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)})$$
$$\mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) \geq \delta \mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)})$$

Descent method with Newton's directions (cont.)

Assume that for every $k \geq 0$, the Hessian $\mathbf{H}(\mathbf{x}^{(k)})$ is symmetric (from the assumption on f) and that it is also positive definite

Let $\mathbf{B}_k = \mathbf{H}(\mathbf{x}^{(k)})$

Suppose that $\exists M > 0 : K(\mathbf{B}_k) = ||\mathbf{B}_k||||\mathbf{B}_k^{-1}|| \leq M$ with $k \geq 0$

- $K(\mathbf{B}_k)$ is the **spectral condition number** of $\mathbf{B}_k$

Under such conditions, the sequence $\{\mathbf{x}^{(k)}\}$ by Newton method converges to a stationary point $\mathbf{x}^*$ of f

- By letting $\alpha_k = 1$ for $k \geq \overline{k}$, the converge is quadratic

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Descent method with Newton's directions (cont.)

Definition

Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, consider the problem of finding a scalar λ (complex or real) and a non-null vector $\mathbf{x} \in \mathbb{C}^n$ such that

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

Any λ that satisfy the equation above is an **eigenvalue** of $\mathbf{A}$

- $\mathbf{x}$ is the corresponding **eigenvector**

Definition

The **spectral condition number** of $\mathbf{A}$ is the quantity

$$K(\mathbf{A}) = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Descent method with Newton's directions (cont.)

Remark

Since Hessians are positive definite, stationary point $\mathbf{x}^*$ cannot be a maximiser or a saddle point and must necessarily be a minimiser

- However, if $\mathbf{H}(\mathbf{x}^{(k)})$ is not positive definite for some point $\mathbf{x}^{(k)}$, then $\mathbf{d}^{(k)}$ may not be a descent direction and Wolfe's conditions might become meaningless

In such situations, the Hessian is replaced by $\mathbf{B}_k = \mathbf{H}(\mathbf{x}^{(k)}) + \mathbf{E}_k$ for some suitable matrix $\mathbf{E}_k$ (either diagonal or full) such that $\mathbf{B}_k$ is positive definite and $\mathbf{d}^{(k)} = -\mathbf{B}_k^{-1}\nabla f(\mathbf{x}^{(k)})$ is a valid direction

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Descent method with quasi-Newton's directions

Line-search methods

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Descent method with quasi-Newton

When using quasi-Newton directions $\mathbf{d}^{(k)} = -\mathbf{H}_k^{-1} \nabla f(\mathbf{x}^{(k)})$, we need to define an approximation $\mathbf{H}_k$ of the true Hessian $\mathbf{H}(\mathbf{x}^{(k)})$

Given a symmetric and positive definite matrix $\mathbf{H}_0$, the recursive **Broyden's rank-one update** for nonlinear systems is popular

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Descent method with quasi-Newton's directions (cont.)

Matrices $\mathbf{H}_k$ are required the following

- To satisfy the secant condition

$$\mathbf{H}_{k+1}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{k})$$

- To be symmetric, as $\mathbf{H}(\mathbf{x})$
- To be positive definite to guarantee that vectors $\mathbf{d}^{(k)}$ are descent directions
- To satisfy the condition

$$\lim_{k\to\infty} \frac{||(\mathbf{H}_k - \mathbf{H}(\mathbf{x}^*))\mathbf{d}^{(k)}||}{||\mathbf{d}^{(k)}||} = 0,$$

which ensures that $\mathbf{H}_k$ is a good approximation of $\mathbf{H}(\mathbf{x}^*)$ along the descent direction $\mathbf{d}^{(k)}$ and guarantees a super-linear rate of convergence

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Descent method with quasi-Newton's directions (cont.)

Definition

A strategy by Broyden, Fletcher, Goldfarb and Shanno (**BFGS**)

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{y}^{(k)}\mathbf{y}^{(k)^T}}{\mathbf{x}^{(k)^T}\mathbf{s}^{(k)}} - \frac{\mathbf{H}_k\mathbf{s}^{(k)}\mathbf{s}^{(k)^T}\mathbf{H}_k^T}{\mathbf{s}^{(k)^T}\mathbf{H}_k\mathbf{s}^{(k)}} \tag{39}$$

where $\mathbf{s}^{(k)} = (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$ and $\mathbf{y}^k = (\nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)}))$

Matrices $\mathbf{H}_{k+1}$ are symmetric and positive definite under condition

$$\mathbf{y}^{(k)^T}\mathbf{s}^{(s)} > 0$$

It is satisfied when step lengths α_k are either weak or strong Wolfe

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Descent method with quasi-Newton's directions (cont.)

BFGS is a thus a descent method, as generally implemented by

Pseudocode

```
Find direction $\mathbf{d}^{(k)} \in \mathbb{R}^n$
Compute step $\alpha_k \in \mathbb{R}$
Set $\mathbf{x}^{(\mathbf{k+1})} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Descent method with quasi-Newton's directions (cont.)

For a given $\mathbf{x}_0$ and a suitable symmetric and positive definite matrix $\mathbf{H}_0 \in \mathbb{R}^{n \times n}$ that approximates $\mathbf{H}(\mathbf{x}^{(0)})$, for $k = 0, 1, \ldots$

Pseudocode

```
Solve $\mathbf{H}_k \mathbf{d}^{(k)} = -\nabla \mathbf{f}(\mathbf{x}^{(k)})$
Compute $\alpha_k$ that satisfies Wolfe's conditions
Set
    $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
    $\mathbf{s}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$
    $\mathbf{y}^{(k)} = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})$
Compute $\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{y}^{(k)} \mathbf{y}^{(k)^T}}{\mathbf{x}^{(k)^T} \mathbf{s}^{(k)}} - \frac{\mathbf{H}_k \mathbf{s}^{(k)} \mathbf{s}^{(k)^T} \mathbf{H}_k^T}{\mathbf{s}^{(k)^T} \mathbf{H}_k \mathbf{s}^{(k)}}$
```

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Descent method with quasi-Newton's directions (cont.)

Example

Rosenbrock: $f(\mathbf{x}) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$, for a $\varepsilon = 10^{-6}$

```
x_0 = [+1.2; -1.0];

fun = @(x) (1-x(1))^2 + 100*(x(2)-x(1)^2)^2;

options = optimset ('LargeScale','off');    % Switches to BFGS
[xstar,fval,exitflag,output] = fminunc(fun,x_0,options)
```

Convergence after 24 iterations and 93 function evaluations

We did not input an expression for evaluating the gradient

- It was, silently, approximated using finite difference methods

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Descent method with quasi-Newton's directions (cont.)

We can define and input the analytical gradient expression

```
x_0 = [+1.2; -1.0];

fun = @(x) (1-x(1))^2 + 100*(x(2)-x(1)^2)^2;
grad_fun = @(x)[-400*(x(2)-x(1)^2)*x(1)-2*(1-x(1)); ...
                +200*(x(2)-x(1)^2)];

options = optimset('LargeScale','off','GradObj','on');
[xstar,fval,exitflag,output] = fminunc({fun,grad_fun},...
                                       x_0,options)
```

Convergence after 25 iterations and 32 function evaluations

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Descent method with quasi-Newton's directions (cont.)

Remark

In Octave, BFGS is implemented by the M-command `bfgsmin`

- M-command `fminunc` implements a trust-region method

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Gradient and conjugate-gradient directions

Line-search methods

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Gradient and conjugate-gradient

Let us first consider the general descent method

Pseudocode

```
Find direction $\mathbf{d}^{(k)} \in \mathbb{R}^n$
Compute step $\alpha_k \in \mathbb{R}$
Set $\mathbf{x}^{(\mathbf{k+1})} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
```

with gradient (descent) directions $\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$

If $f \in \mathbb{C}^2(\mathbb{R}^n)$ is bounded from below and step lengths α_k are Wolfe, this method converges (linearly) to a stationary point

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Gradient and conjugate-gradient directions (cont.)

Let us now consider conjugate directions,

$$\begin{aligned} \mathbf{d}^{(0)} &= -\nabla f(\mathbf{x}^{(0)}) \\ \mathbf{d}^{(k+1)} &= -\nabla f(\mathbf{x}^{(k+1)}) - \beta_k \mathbf{d}^{(k)}, \quad k \geq 0 \end{aligned}$$

several options for setting β_k are available

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Gradient and conjugate-gradient directions (cont.)

- **Fletcher-Reeves**

$$\beta_k^{FR} = -\frac{||\nabla f(\mathbf{x}^{(k)})||^2}{||\nabla f(\mathbf{x}^{(k-1)})||^2} \qquad (40)$$

- **Polak-Ribière (-Polyak)**

$$\beta_k^{PR} = -\frac{\nabla f(\mathbf{x}^{(k)})^T(\nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^{(k-1)}))}{||\nabla f(\mathbf{x}^{(k-1)})||^2} \qquad (41)$$

- **Hestenes-Stiefel**

$$\beta_k^{HS} = -\frac{\nabla f(\mathbf{x}^{(k)})^T(\nabla f(\mathbf{x}^{(k)})^T - \nabla f(\mathbf{x}^{(k-1)}))}{\mathbf{d}^{(k-1)^T}(\nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^{(k-1)}))} \qquad (42)$$

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Gradient and conjugate-gradient directions (cont.)

Remark

Under the condition that f is quadratic and strictly convex, all the aforementioned options are equivalent and reduce to

$$\beta_k = \frac{(\mathbf{A}\mathbf{d}^{(k)})^T \mathbf{r}^{(k+1)}}{\mathbf{d}^{(k)^T} \mathbf{A}\mathbf{d}^{(k)}}$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods

Numerical optimisation

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Trust-region methods

Line search methods are designed to determine first the descent direction $\mathbf{d}^{(k)}$ and then the step-length α_k, at any k-th step

Trust-region methods simultaneously choose direction and step length, by building a trust ball centred at $\mathbf{x}^{(k)}$ and of radius δ_k

- In the trust region, compute a quadratic approximation $\tilde{f}_k$ of f

The new value of $\mathbf{x}^{(k+1)}$ is the minimiser of $\tilde{f}_k$ in the trust region

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

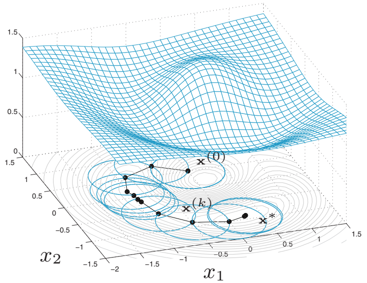
Trust-region methods

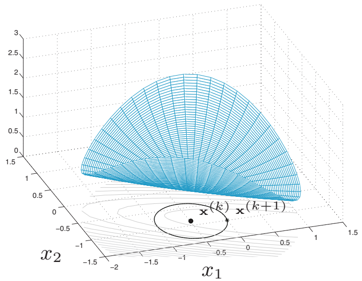
Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)





Convergence history and quadratic approximation $\tilde{f}_k$ at step $k = 8$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

To compute $\tilde{f}_k$, we start with a trust radius $\delta_k > 0$ and a second-order Taylor expansion of f about $\mathbf{x}^{(k)}$

$$\tilde{f}_k(\mathbf{s}) = f(\mathbf{x}^{(k)}) + \mathbf{s}\nabla f(\mathbf{x}^{(k)}) + \frac{1}{2}\mathbf{s}^T \mathbf{H}_k \mathbf{s}, \quad \forall \mathbf{s} \in \mathbb{R}^n \tag{43}$$

$\mathbf{H}_k$ is either the Hessian of f at $\mathbf{x}^{(k)}$ or a suitable approximation

We then compute the solution $\mathbf{s}^{(k)}$

$$\mathbf{s}^{(k)} = \underset{\mathbf{s}\in\mathbb{R}^n:||\mathbf{s}||\leq\delta_k}{\arg\min}\ \tilde{f}_k(\mathbf{s}) \tag{44}$$

At this stage, we also compute

$$\rho_k = \frac{f(\mathbf{x}^{(k)} + \mathbf{s}^{(k)}) - f(\mathbf{x}^{(k)})}{\tilde{f}_k(\mathbf{s}^{(k)}) - \tilde{f}_k(\mathbf{0})} \tag{45}$$

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Trust-region methods (cont.)

- If ρ_k is approximately one, we accept $\mathbf{s}^{(k)}$, we move on to the next iteration and set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}$ (however, if the minimiser of $\tilde{f}_k$ lie on the boundary of the trust region , we extend the latter before proceeding to next iteration)
- If ρ_k is either negative or positive and much smaller than one), we reduce the ball's size and we calculate a new $\mathbf{s}^{(k)}$

$$\mathbf{s}^{(k)} = \underset{\mathbf{s} \in \mathbf{R}^n : ||\mathbf{s}|| \leq \delta_k}{\arg\min} \tilde{f}_k(\mathbf{s})$$

- If ρ_k is much larger than one, we accept $\mathbf{s}^{(k)}$, we keep the trust region as it is and then we move to the next iteration

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

Remark

When the second derivative of f are available, we can set $\mathbf{H}_k$ to be equal to the Hessian (or a variant, if not positive definite)

- Otherwise, $\mathbf{H}_k$ can be built recursively

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Trust-region methods (cont.)

If $\mathbf{H}_k$ is symmetric positive definite and $||\mathbf{H}_k^{-1}\nabla f(\mathbf{x}^{(k)})|| \leq \delta_k$ then $\mathbf{s}^{(k)} = \mathbf{H}_k^{-1}\nabla f(\mathbf{x}^{(k)})$ is a minimiser and it is within the trust region

- Otherwise, the minimiser of $\tilde{f}_k$ lies outside the trust region

It is a minimisation of $\tilde{f}_k$ constrained to the δ_k-ball centred at $\mathbf{x}^{(k)}$

$$\min_{\mathbf{s}\in\mathbb{R}^n : ||\mathbf{s}|| = \delta_k} \tilde{f}_k(\mathbf{s}) \tag{46}$$

which can be solved using Lagrange multipliers

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

We look for the saddle point of the Lagrangian

$$\mathcal{L}(\mathbf{s}, \lambda) = \tilde{f}_k(\mathbf{s}) + \frac{1}{2}\lambda(\mathbf{s}^T\mathbf{s} - \delta_k)$$

So, a vector $\mathbf{s}^{(k)}$ and a scalar $\lambda^{(k)} > 0$ satisfying

$$\begin{aligned}(\mathbf{H}_k + \lambda^{(k)}\mathbf{I})\mathbf{s}^{(k)} &= -\nabla f(\mathbf{x}^{(k)}) \\ (\mathbf{H}_k + \lambda^{(k)}\mathbf{I}) &\text{ is PSD} \\ ||\mathbf{s}^{(k)}|| - \delta_k &= 0\end{aligned} \tag{47}$$

is what we are after in this minimisation task

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

From $(\mathbf{H}_k + \lambda^{(k)}\mathbf{I})\mathbf{s}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$, we compute $\mathbf{s}^{(k)} = \mathbf{s}^{(k)}(\lambda^{(k)})$

We substitute it in $||\mathbf{s}^{(k)}|| - \delta_k = 0$ to get

$$\varphi(\lambda^{(k)}) = \frac{1}{||\mathbf{s}^{(k)}(\lambda^{(k)})||} - \frac{1}{\delta_k} = 0$$

Alone, this non-linear equation in the unknown λ is equivalent to System 47 and can be easily solved using Newton's method

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Trust-region methods (cont.)

Given λ_0 and set $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$

Pseudocode

```
For $l = 0, 1, \dots$ (typically, less than 5 iterations are needed)
    Compute $\mathbf{s}_l^{(k)} = -(\mathbf{H}_k + \lambda_l^{(k)}\mathbf{I})^{-1}\mathbf{g}^{(k)}$
    Evaluate $\varphi(\lambda_l^{(k)}) = \dfrac{1}{||\mathbf{s}_l^{(k)}||} - \dfrac{1}{\delta_k}$
    Evaluate $\varphi'(\lambda_l^{(k)})$
    Compute $\lambda_{l+1}^{(k)} = \lambda_l^{(k)} - \dfrac{\varphi\lambda_l^{(k)}}{\varphi'(\lambda_l^{(k)})}$
```

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Trust-region methods (cont.)

Vector $\mathbf{s}_l^{(k)}$ is obtained by **Cholesky factorisation** of $(\mathbf{H}_k + \lambda_l^{(k)}\mathbf{I})$

- Provided that matrix $\mathbf{B}^{(k)} = \mathbf{H}_k + \lambda_l^{(k)}\mathbf{I}$ is positive definite
- $\mathbf{B}^{(k)}$ is symmetric (definition of $\mathbf{H}_k$)
- Its eigenvalues are all real

Remark

Usually, a regularised matrix $\mathbf{B}_l^{(k)} + \beta\mathbf{I}$ is used instead of $\mathbf{B}^{(k)}$

- β is chosen to be larger than the negative eigenvalue of $\mathbf{B}^{(k)}$ with largest modulus

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

Definition

Cholesky factorisation

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a symmetric and positive definite matrix

$$\mathbf{A} = \mathbf{R}^T \mathbf{R}$$

$\mathbf{R}$ is upper triangular with positive elements on the diagonal

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Trust region methods (cont.)

For $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$ and for a given δ_k,

Pseudocode

```
Solve $\mathbf{H}_k\mathbf{s} = -\mathbf{g}^{(k)}$ (means $\mathbf{s} = -\mathbf{H}_k^{(-1)}\mathbf{g}^{(k)}$)
If $||\mathbf{s}|| \leq \delta_k$ and $\mathbf{H}_k$ is positive definite
    Set $\mathbf{s}^{(k)} = \mathbf{s}$
else
    Let $\beta_1$ be the negative eigenvalue of $\mathbf{H}_k$ with largest modulus
    Set $\lambda_0^{(k)} = 2|\beta_1|$
    For $l = 0, 1, \dots$
        Compute $\mathbf{R} : \mathbf{R}^T\mathbf{R} = \mathbf{H}_k + \lambda_l^{(k)}\mathbf{I}$
        Solve $\mathbf{R}^T\mathbf{R}\mathbf{s} = \mathbf{g}^{(k)}$, $\mathbf{R}^T\mathbf{q} = \mathbf{s}$
        Update $\lambda_{l+1}^{(k)} = \lambda_l^{(k)} + \Big(\frac{||\mathbf{s}||}{||\mathbf{q}||}\Big)^2 \frac{||\mathbf{s}|| - \delta_k}{\delta_k}$
    Set $\mathbf{s}^{(k)} = \mathbf{s}$
endif
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

For a fast convergence, a good radius δ_k is truly fundamental

The criterion for accepting a solution $\mathbf{s}^{(k)}$ is based on a comparison between variation of f and that of its quadratic approximation $\tilde{f}_k$

- as $\mathbf{x}^{(k)}$ moves to $\mathbf{x}^{(k)} + \mathbf{s}^{(k)}$

$$\rho_k = \frac{f(\mathbf{x}^{(k)} + \mathbf{s}^{(k)}) - f(\mathbf{x}^{(k)})}{\tilde{f}_k(\mathbf{s}^{(k)}) - \tilde{f}_k(\mathbf{0})}$$

Remark

- If $\rho_k \approx 1$, $\mathbf{s}^{(k)}$ is accepted and the ball is enlarged, if the minimum is on the boundary
- If $\rho_k \approx 0$ or $\rho_k < 0$, $\mathbf{s}^{(k)}$ is not accepted and the ball is diminished

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

Given an initial solution $\mathbf{x}^{(0)}$, an initial radius of the ball $\delta_0 \in (0, \hat{\delta})$ with maximum radius $\hat{\delta} > 0$, four real parameters $\{\eta_1, \eta_2, \gamma_1, \gamma_2\}$ such that $0 < \eta_1 < \eta_2 < 1$ and $0 < \gamma_1 < 1 < \gamma_2$ for updating the ball and a real parameter $0 \leq \mu \leq \eta_1$ for accepting a solution, ...

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Trust-region methods (cont.)

... for $k = 0, 1, \dots$ until convergence

Pseudocode

```
Compute $f(\mathbf{x}^{(k)})$, $\nabla f(\mathbf{x}^{(k)})$ and $\mathbf{H}_k$
Solve $\min_{||\mathbf{s}\in\mathbf{R}^n:\mathbf{s}||_2\leq\delta_k} \tilde{f}_k(\mathbf{s})$
Compute $\rho_k$
If $\rho_k > \mu$
    Set $\mathbf{x}^{(x+1)} = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}$
else
    Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$
endif
If $\rho_k < \eta_1$
    Set $\delta_{k+1} = \gamma_1\delta_k$
elseif $\eta_1 \leq \rho_k \leq \eta_2$
    Set $\delta_{k+1} = \delta_k$
elseif $\rho_k > \eta_2$ and $||\mathbf{s}^{(k)}|| = \delta_k$
    Set $\delta_{k+1} = \min\{\gamma_2\delta_k, \hat{\delta}\}$
endif
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

Choice of parameters[4]: $\eta_1 = 1/4$, $\eta_2 = 3/4$, $\gamma_1 = 1/4$, $\gamma_2 = 8/4$

- By choosing $\mu = 0$ we accept any step yielding a decrease of f
- By choosing $\mu > 0$ we accept steps for which the variation of f is at least μ times the variation of its quadratic model $\tilde{f}_k$

[4] J. Nocedal and S. Wrigth (2006): *Numerical optimization*.

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

```
%TREGION Trust region optimisation method
%[X,ERR,ITER]=TREGION(FUN,GRAD_FUN,X_0,DELTA_0, ...
% TOL,KMAX,TYP,HESS_FUN)
% Approximates the minimiser of FUN with gradient GRAD_FUN
%
% If TYP=1 Hessian is inputed as HESS_FUN
% If TYP NE 1 Hessian is rank-one approximated
%
% FUN and GRAD_FUN (and HESS_FUN) are function handles
% X_0 is the initial  point
% TOL is stop check tolerance
% DELTA_0 is initial radius of trust ball
% KMAX are maximum number of iterations
```

```
function [x,err,iter]= tRegion(fun,grad_fun,x_0,delta_0, ...
                              tol,kmax,typ,hess_fun)

delta = delta_0; err = 1 + tol; k = 0; mu = 0.1; delta_m = 5;
eta_1 = 0.25; eta_2 = 0.75; gamma_1 = 0.25; gamma_2 = 2.00;

xk = x_0(:); gk = grad_fun(xk); eps2 = sqrt(eps);
if typ==1; Hk=hess_fun(xk); else; Hk=eye(length(xk)); end

while err > tol & k < kmax
 [s]=trust_one(Hk,gk,delta);
 rho=(fun(xk+s)-fun(xk))/(s'*gk+1/2*s'*Hk*s);
 if rho > mu; xk1 = xk + s; else; xk1 = xk; end
 if rho < eta_1; delta = gamma_1*delta;
 elseif rho > eta_2 & abs(norm(s)-delta) < sqrt(eps)
  delta=min([gamma_2*delta,delta_m]);
 end
 gk1 = grad_fun(xk1);
 err = norm((gk1.*xk1)/max([abs(fun(xk1)),1]),Inf);
 if typ == 1; xk = xk1; gk = gk1; Hk = hess_fun(xk);  % Newton
 else                                                 % quasi-Newton
  gk1 = grad(xk1); yk = gk1-gk; sk=xk1-xk; yks = yk'*sk;
  if yks > eps_2*norm(sk)*norm(yk)
   Hs = Hk*sk; Hk = Hk+(yk*yk')/yks-(Hs*Hs')/(sk'*Hs);
  end
  xk = xk1; gk = gk1;
 end
 k=k+1;
end

x = xk; iter = k;
if (k==kmax & err>tol); disp('Accuracy not met [KMAX]'); end
```

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Trust-region methods (cont.)

```
function [s] = trust_one (Hk,gk,delta)
maxiter=5;

s = -Hk\gk; d = eigs(Hk,1,'sa'); % 1st smallest algebraic
    evalue

if norm(s) > delta | d<0
 lambda = abs(2*d); I = eye(size(Hk));
 for l=1:maxiter
  R = chol(lambda*I+Hk);
  s = -R\(R'\gk); q = R'\s;
  lambda = lambda+(s'*s)/(q'*q)*(norm(s)-delta)/delta;
  if lambda < -d
   lambda = abs(2*lambda);
  end
 end
end
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

Example

Approximate the minimiser of function

$$f(x_1, x_2) = \frac{7}{5} + \frac{(x_1 + 2x_2 + 2x_1x_2 - 5x_1^2 - 5x_2^2)}{(5\exp(x_1^2 + x_2^2))}$$

using the trust-region method

A local maximum, a saddle point and two local minima at approx. $(-1.0, +0.2)$ and $(+0.3, -0.9)$, the second being the global one

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust region methods (cont.)

```
fun = @(x) (x(1)+2*x(2)+2*x(1)*x(2)-5*x(1)^2-5*x(2)^2) / ...
           (5*exp(x(1)^2+x(2)^2)) + 7.5;

grad_fun = @(x) [(1 + 2*x(2)-10*x(1)-2*x(1)*(x(1)+2*x(2) + ...
                      2*x(1)*x(2)-5*x(1)^2-5*x(2)^2)) / ...
                      (5*exp(x(1)^2+x (2)^2));
                 (2 + 2*x(1)-10*x(2)-2*x(2)*(x(1)+2*x(2) + ...
                      2*x(1)*x(2)-5*x(1)^2-5*x(2)^2)) / ...
                      (5*exp(x(1)^2+x(2)^2))];

delta_0 = 0.5; x_0 = [0.0;0.5];
tol = 1e-5; kmax = 100; imax=5;
typ = 2;

[x,er,it]=tRegion(fun,grad_fun,x_0,delta_0,tol,kmax,typ,imax)
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

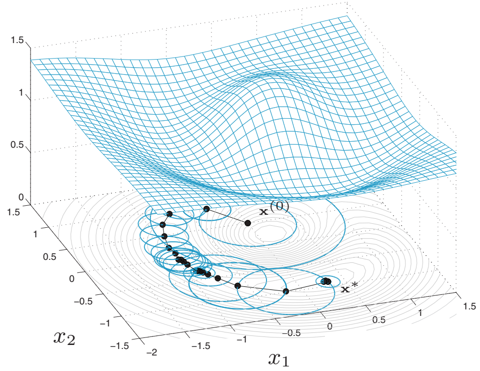
Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

Trust-region, approx. Hessian: 24 iters, $\mathbf{x}^* \approx (+0.28, -0.90)$



Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

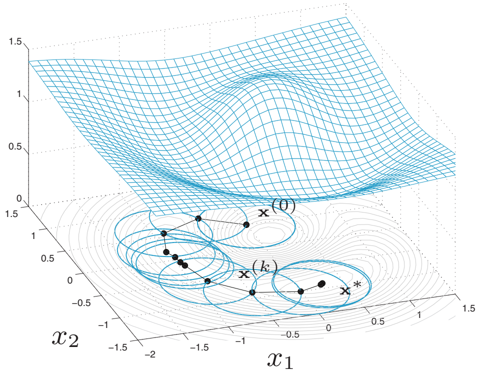
Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Trust-region methods (cont.)

Trust-region, exact Hessian: 12 iterations



Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

Trust region methods (cont.)

Example

Rosenbrock's function: $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

```
fun = @(x) (1-x(1))^2+100*(x(2)-x(1)^2)^2;
grad_fun = @(x)[-400*(x(2)-x(1)^2)*x(1)-2*(1-x(1)); ...
                     200*(x(2)-x(1)^2)];

x_0=[+1.2;-1.0];

options = optimset ('LargeScale','on'); % Trust-region
options = optimset ('GradObj','on');    % Gradient

[x,fval,exitflag,output]=fminunc({fun,grad_fun},x_0,options)
```

Trust-region (Matlab): 8 iterations, 9 function evaluations

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Trust-region methods (cont.)

Remark

The M-command `fminunc` in Octave implements the trust region method with approximated Hessians $\mathbf{H}_k$, computed with BFGS

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{y}^{(k)}\mathbf{y}^{(k)^T}}{\mathbf{x}^{(k)^T}\mathbf{s}^{(k)}} - \frac{\mathbf{H}_k\mathbf{s}^{(k)}\mathbf{s}^{(k)^T}\mathbf{H}_k^T}{\mathbf{s}^{(k)^T}\mathbf{H}_k\mathbf{s}^{(k)}}$$

The option `'LargeScale'` is not used

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Non-linear least-squares

Numerical optimisation

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Non-linear least-squares

The **least-squares method** is often used for approximating either functions $f(x)$ or sets of data $\{(x_k, y_k), k = 0, \dots, K\}$ by function $\tilde{f}$ linearly depending on a set of coefficients $\{a_j, j = 1, \dots, m\}$

Example

$$\tilde{f}(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$$

The coefficients $\{a_i\}_{i=0}^{m}$ are unknown and must be determined

$$\sum_{k=0}^{K} \Big(y_k - \tilde{f}(x_k)\Big)^2$$

- **Non-linear least-squares** refers to problems in which such a dependence is non-linear

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Non-linear least-squares(cont.)

Definition

Let $\mathbf{R}(\mathbf{x}) = (r_1(\mathbf{x}), \dots, r_n(\mathbf{x}))^T$ with $r_i : \mathbb{R}^m \to \mathbb{R}$ be some function

$$\min_{\mathbf{x} \in \mathbb{R}^m} \Phi(\mathbf{x}), \quad \text{with } \Phi(\mathbf{x}) = \frac{1}{2}||\mathbf{R}(\mathbf{x})||^2 = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\mathbf{x}) \qquad (48)$$

When functions r_i are non-linear, function Φ may not be convex

- Thus, have multiple stationary points

Newton, descent directions, trust-region methods can be used

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Non-linear least-squares(cont.)

Because of the form of Φ, gradient and Hessian can be written in terms of the Jacobian $\mathbf{J_R}(\mathbf{x}) \in \mathbb{R}^{n \times m}$ and second derivatives of $\mathbf{R}$

$$\begin{aligned} \nabla \Phi(\mathbf{x}) &= \mathbf{J_R}(\mathbf{x})^T \mathbf{R}(\mathbf{x}) \\ \mathbf{H}(\mathbf{x}) &= \mathbf{J_R}(\mathbf{x})^T \mathbf{J_R}(\mathbf{x}) + \mathbf{S}(\mathbf{x}) \end{aligned} \tag{49}$$

in which $\mathbf{S}_{lj}(\mathbf{x}) = \sum_{i=1}^{n} \dfrac{\partial^2 r_i}{\partial x_l \partial x_j}(\mathbf{x}) r_i(\mathbf{x}) r_i(\mathbf{x})$ for $l, j = 1, \dots, m$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Non-linear least-squares(cont.)

Calculation of the Hessian can be heavy when m and n are large

- This is especially due to matrix $\mathbf{S}(\mathbf{x})$

In some cases $\mathbf{S}(\mathbf{x})$ is less influent than $\mathbf{J_R}(\mathbf{x})^T \mathbf{J_R}(\mathbf{x})$ and could be approximated or neglected in the construction of the Hessian $\mathbf{H}(\mathbf{x})$

We discuss two methods devoted to handling such cases

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Gauss-Newton method

Nonlinear least-squares

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

The Gauss-Newton method

The **Gauss-Newton method** is a variant of the Newton method

Given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, for $k = 0, 1, \dots$ until convergence

Pseudocode

```
Solve $\mathbf{H}(\mathbf{x}^{(k)})\delta\mathbf{x}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$
Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}$
```

The Hessian $\mathbf{H}(\mathbf{x})$ is approximated by neglecting $\mathbf{S}(\mathbf{x})$

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

The Gauss-Newton method (cont.)

Given $\mathbf{x}^{(0)} \in \mathbb{R}^m$ and for $k = 0, 1, \dots$ until the convergence

Pseudocode

```
Solve $[\mathbf{J_R}(\mathbf{x}^k)^T \mathbf{J_R}(\mathbf{x}^{(k)})]\boldsymbol{\delta}\mathbf{x}^{(k)} = -\mathbf{J_R}(\mathbf{x}^{(k)})^T \mathbf{R}(\mathbf{x}^{(k)})$
Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}$
```

If $\mathbf{J_R}(\mathbf{x}^{(k)})$ is not full rank, the linear system in the first equation has infinitely many solutions leading either to a stagnation of the method or to convergence to a non-stationary point

If $\mathbf{J_R}(\mathbf{x}^{(k)})$ is full rank, the linear system has form $\mathbf{A}^T\mathbf{A}\mathbf{x}^* = \mathbf{A}^T\mathbf{b}$ and it can be solved by using QR or SVD factorisations of $\mathbf{J_R}(\mathbf{x})$

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

The Gauss-Newton method (cont.)

```
function [x,err,iter]=nllsGauNewtn(r,jr,x_0,tol,kmax,varargin)
%NLLSGAUNEW Nonlinear least-squares with Gauss-Newton method
% [X,ERR,ITER]=NLLSGAUNEW(R,JR,X_0,TOL,KMAX)
% R and JR: Function handles for objective R and its Jacobian
% X_0 is the initial solution
% TOL is the stop check tolerance
% KMAX is the max number of iterations

err = 1 + tol; k = 0;
xk = x_0(:);

rk = r(xk,varargin{:}); jrk = jr(xk,varargin{:});

while err > tol & k < kmax
 [Q,R] = qr(jrk,0); dk = -R\(Q'*rk);
 xk1 = xk + dk;
 rk1 = r(xk1,varargin{:});
 jrk1 = jr(xk1,varargin{:});

 k = 1 + k; err = norm(xk1 - xk);
 xk = xk1; rk = rk1; jrk = jrk1;
end

x = xk; iter = k;

if (k==kmax & err > tol)
 disp('nllsGauNewtn stopped w\o reaching accuracy [KMAX]');
end
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

The Gauss-Newton method (cont.)

Remark

It can be shown that neglecting $\mathbf{S}(\mathbf{x}^{(k)})$ at step k amounts to approximating $\mathbf{R}(\mathbf{x})$ with its first-order Taylor expansion at $\mathbf{x}^*$

$$\tilde{\mathbf{R}}_k(\mathbf{x}) = \mathbf{R}(\mathbf{x}^{(k)}) + \mathbf{J}_{\mathbf{R}}(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) \qquad (50)$$

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

The Gauss-Newton method (cont.)

Convergence of the method is not always guaranteed as it depends on both properties of Φ and initial solution

If $\mathbf{x}^*$ is stationary point for Φ and $\mathbf{J}_\mathbf{R}(\mathbf{x})$ is full rank in a suitable neighbourhood of $\mathbf{x}^*$, then

1. If $\mathbf{S}(\mathbf{x}^*) = 0$, which is the case if $\mathbf{R}(\mathbf{x})$ is linear or $\mathbf{R}(\mathbf{x}^*) = \mathbf{0}$, the Gauss-Newton method is locally quadratically convergent and it coincides with the Newton's method
2. If $||\mathbf{S}(\mathbf{x}^*)||_2$ is small compared to the smallest positive e-value of $\mathbf{J}_\mathbf{R}(\mathbf{x}^*)^T\mathbf{J}_\mathbf{R}(\mathbf{x}^*)$, then Gauss-Newton converges linearly (for instance, when $\mathbf{R}(\mathbf{x})$ is mildly non-linear or $\mathbf{R}(\mathbf{x}^*)$ is small)
3. If $||\mathbf{S}(\mathbf{x})||_2$ is large compared to the smallest positive e-value of $\mathbf{J}_\mathbf{R}(\mathbf{x}^*)^T\mathbf{J}_\mathbf{R}(\mathbf{x}^*)$, then Gauss-Newton may not converge even if $\mathbf{x}^{(0)}$ is very close to $\mathbf{x}^*$ (this happens if $\mathbf{R}(\mathbf{x})$ is strongly non-linear or if its residual $\mathbf{R}(\mathbf{x}^*)$ is large)

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

The Gauss-Newton method (cont.)

Remark

Line-search can be used in combination with Gauss-Newton by replacing $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}$ with $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \boldsymbol{\delta}\mathbf{x}^{(k)}$

- Computation of step-lengths α_k is as per usual

If $\mathbf{J}_{\mathbf{R}}(\mathbf{x}^{(k)})$ is full rank, matrix $\mathbf{J}_{\mathbf{R}}(\mathbf{x}^{(k)})^T \mathbf{J}_{\mathbf{R}}(\mathbf{x}^{(k)})$ is symmetric and positive definite and $\boldsymbol{\delta}\mathbf{x}^{(k)}$ is a descent direction for Φ

In this case, under suitable assumptions on Φ, we get the globally convergent method known as **damped Gauss-Newton method**

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

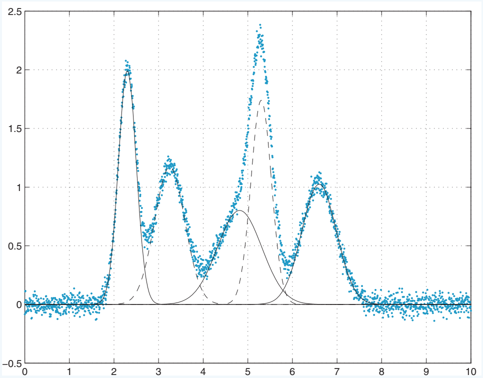
Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Example

Voice recognition: Compress an audio signal to a set of parameters



The signal intensity is modelled as a sum of m Gaussian functions

$$f_k(t|a_k,\sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\Big(-\frac{(t-a_k)^2}{2\sigma_k^2}\Big), t \in [t_0, t_F], k = 1, \dots, m$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

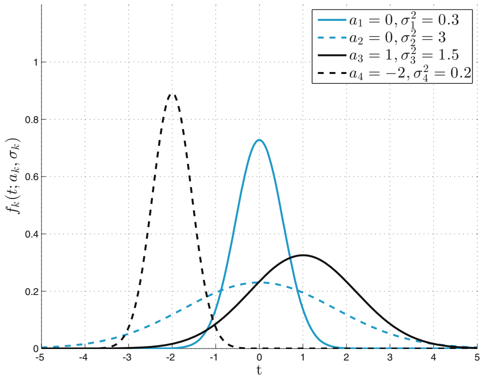
Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

The Gauss-Newton method (cont.)



Each peak or component is characterised by two coefficients

- The centre, a_k
- The (square of the) spread, σ_k^2

$$f(t|\mathbf{a}, \boldsymbol{\sigma}) = \sum_{k=1}^{m} f_k(t; a_k, \sigma_k)$$

- $\mathbf{a} = [a_1, \cdots, a_k]$
- $\boldsymbol{\sigma} = [\sigma_1, \cdots, \sigma_k]$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

The Gauss-Newton method (cont.)

Find $\mathbf{a}$ and $\boldsymbol{\sigma}$ that minimise the residual sum of squares

$$\min_{\mathbf{a},\boldsymbol{\sigma}} \sum_{i=1}^{n} \Big(f(t_i|\mathbf{a},\boldsymbol{\sigma}) - y_i \Big)^2$$

From recorded audio intensities y_i at sampling times t_i

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

The Gauss-Newton method (cont.)

Generate $n = 2000$ time-intensity pairs $(t_i, y_i)_{i=1}^{n}$ with $t_i \in (0, 10)$

- By summing 5 Gaussian components

$$f_k(t|a_k, \sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\Big(- \frac{(t-a_k)^2}{(2\sigma_k^2)}\Big)$$

- and by adding little random noise

```
a = [2.30, 3.25, 4.82, 5.30, 6.60]; m = length(a);
sigma = [0.20, 0.34, 0.50, 0.23, 0.39];

gComp = @(t,a,sigma) exp(-((t-a)/(sigma*sqrt(2))).^2)/ ...
                     (sigma*sqrt(pi*2));

n = 2000; t = linspace(0,10,n)'; y = zeros(n,1);
for k=1:m
 y = y + gComp(t,a(k),sigma(k));
end

y = y + 0.05*randn(n,1); % Additive random noise
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

The Gauss-Newton method (cont.)

We want to solve the nonlinear least-squares problem of form

$$\min_{\mathbf{x} \in \mathbb{R}^m} \Phi(\mathbf{x}), \quad \text{with } \Phi(\mathbf{x}) = \frac{1}{2}||\mathbf{R}(\mathbf{x})||^2 = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\mathbf{x})$$

in which $r_i(\mathbf{x}) = f(t_i|\mathbf{a}, \boldsymbol{\sigma}) - y_i = \sum_{k=1}^{m} f_k(t_i|a_k, \sigma_k) - y_i$ and

$$\frac{\partial r_i}{\partial a_k} = f_k(t_i|a_k, \sigma_k)\frac{t_i - a_k}{\sigma_k}$$
$$\frac{\partial r_i}{\partial \sigma_k} = f_k(t_i|a_k, \sigma_k)[\frac{(t_i - a_k)^2}{\sigma_k^3} - \frac{1}{2\sigma_k}]$$

Unconstrained optimisation

Derivative-free methods
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Line-search methods
Descent directions
Step-length α_k
Descent method with Newton's directions
Descent method with quasi-Newton's directions
Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
The Gauss-Newton method
Levenberg-Marquardt

The Gauss-Newton method (cont.)

Using the M-command `nllsGauNewtn`

```
x_0 = [2.0,3.0,4.0,5.0,6.0,0.3,0.3,0.6,0.3,0.3];

tol = 3.0e-5;
kmax = 200;

[x,err,iter]=nllsGauNew(@gmR,@gmJR,x_0,tol,kmax,t,y)

x_a = x(1:m);
x_sigma = x(m+1:end);

h = 1./(x_sigma*sqrt(2*pi));
w = 2*x_sigma*sqrt(log(4));
```

Gauss-Newton: 22 iterations

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

The Gauss-Newton method (cont.)

```
function [R]=gmR(x,t,y)

x = x(:); m = round(0.5*length(x));
a = x(1:m); sigma = x(m+1: end );

gauFun = @(t,a,sigma) [exp(-((t-a)/(sigma*sqrt(2))).^2) ...
                      /(sigma*sqrt(pi*2))];

n = length(t); R = zeros(n,1);
for k = 1:m; R = R + gauFun(t,a(k),sigma(k)); end
R = R - y;
```

```
function [Jr]=gmJR(x,t,y)
x = x(:); m = round(0.5*length(x));
a = x(1:m); sigma = x(m+1: end);

gauFun = @(t,a,sigma) [exp(-((t-a)/(sigma*sqrt(2))).^2) ...
                      /(sigma*sqrt(pi*2))];

n = length(t); JR = zeros(n,2*m); fk = zeros(n,m);
for k = 1:m; fk(:,k) = gauFun(t,a(k),sigma(k)); end
for k = 1:m; JR(:,k) = (fk(:,k).*(t-a(k))/sigma(k)^2)'; end
for k = 1:m
 JR(:,k+m) = (fk(:,k).*((t-a(k)).^2/(k)^3-1/(2*sigma(k))))';
end
```

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Levenberg-Marquardt

Nonlinear least-squares

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Levenberg-Marquardt

Levenberg-Marquardt is a trust-region method for

$$\min_{\mathbf{x} \in \mathbb{R}^m} f(\mathbf{x}), \quad \text{with } f(\mathbf{x}) = \frac{1}{2}||\mathbf{R}(\mathbf{x})||^2 = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\mathbf{x})$$

We can use the general trust-region pseudocode

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Levenberg-Marquardt (cont.)

Pseudocode

```
Compute $f(\mathbf{x}^{(k)})$, $\nabla f(\mathbf{x}^{(k)})$ and $\mathbf{H}_k$
Solve $\min_{||\mathbf{s}||_2 \leq \delta_k} \tilde{f}_k(\mathbf{s})$
Compute $\rho_k$
If $\rho_k > \mu$
    Set $\mathbf{x}^{(x+1)} = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}$
else
    Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$
endif
If $\rho_k < \eta_1$
    Set $\delta_{k+1} = \gamma_1 \delta_k$
elseif $\eta_1 \leq \rho_k \leq \eta_2$
    Set $\delta_{k+1} = \delta_k$
elseif $\rho_k > \eta_2$ and $||\mathbf{s}^{(k)}|| = \delta_k$
    Set $\delta_{k+1} = \min\{\gamma_2 \delta_k, \hat{\delta}\}$
endif
```

Unconstrained optimisation

Derivative-free methods
- Golden section and quadratic interpolation
- Nelder and Mead

The Newton method

Line-search methods
- Descent directions
- Step-length α_k
- Descent method with Newton's directions
- Descent method with quasi-Newton's directions
- Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares
- The Gauss-Newton method
- Levenberg-Marquardt

Levenberg-Marquardt (cont.)

After replacing f with Φ and $\tilde{f}$ with $\tilde{\Phi}$, at each step k we solve

$$\min_{\mathbf{s}\in\mathbb{R}^n:||\mathbf{s}||\leq\delta_k} \tilde{\Phi}_k(\mathbf{s}), \quad \text{with } \tilde{\Phi}_k(\mathbf{s}) = \frac{1}{2}||\mathbf{R}(\mathbf{x}^{(k)}) + \mathbf{J_R}(\mathbf{x}^{(k)})\mathbf{s}||^2 \quad (51)$$

Note how $\tilde{\Phi}_k(\mathbf{x})$ is a quadratic approximation of $\Phi(\mathbf{x})$ about $\mathbf{x}^{(k)}$

- It is obtained by approximating $\mathbf{R}(\mathbf{x})$ with its linear model

$$\tilde{\mathbf{R}}_k(\mathbf{x}) = \mathbf{R}(\mathbf{x}^{(k)}) + \mathbf{J_R}(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)})$$

Unconstrained optimisation

Derivative-free methods

Golden section and quadratic interpolation

Nelder and Mead

The Newton method

Line-search methods

Descent directions

Step-length α_k

Descent method with Newton's directions

Descent method with quasi-Newton's directions

Gradient and conjugate-gradient descent directions

Trust-region methods

Nonlinear least-squares

The Gauss-Newton method

Levenberg-Marquardt

Levenberg-Marquardt (cont.)

Even when $\mathbf{J_R}(\mathbf{x})$ is not full rank, the method is well suited for minimisation problems with strong non-linearities or large residuals $\Phi(\mathbf{x}^*) = \dfrac{1}{2}||\mathbf{R}(\mathbf{x}^*))||^2$ at the local minimiser $\mathbf{x}^*$

Remark

Hessian approximations are those of the Gauss-Newton method, the two methods share the same local convergence properties

- When Levenberg-Marquardt iterations converge, convergence rate is quadratic if the residual is small at a local minimiser
- Convergence rate is linear otherwise