MÄLARDALEN UNIVERSITY
SWEDEN

School of Education, Culture and Communication
Division of Applied Mathematics

MASTER THESIS IN MATHEMATICS / APPLIED MATHEMATICS

Non-linear Curve Fitting

by

Farhad Morad

masterarbete i matematik / tillämpad matematik

DIVISION OF APPLIED MATHEMATICS
MÄLARDALEN UNIVERSITY
SE-721 23 VÄSTERÅS, SWEDEN

MÄLARDALEN UNIVERSITY
SWEDEN

School of Education, Culture and Communication
Division of Applied Mathematics

Master thesis in mathematics / applied mathematics

Date:
2019-05-22

Project name:
Non-linear Curve Fitting

Author:
Farhad Morad

Supervisor:
Milica Rančić, Karl Lundengård

Reviewer:
Thomas Westerbäck

Examiner:
Christopher Engström

Comprising:
15 ECTS credits

Abstract

The work done in this thesis is to examine various methods for curve fitting. Linear least squares and non-linear least squares will be described and compared, and the Newton method, Gauss–Newton method and Levenberg–Marquardt method will be applied to example problems.

Keywords: Curve Fitting, Power-exponential function, Gauss–Newton algorithm, Levenberg–Marquardt algorithm, Linear and non-linear curve fitting

Sammanfattning

Syftet med denna uppsats är att beskriva och använda olika metoder för kurvanpassning, det vill säga att passa matematiska funktioner till data. De metoder som undersöks är Newtons metod, Gauss–Newton metoden och Levenberg–Marquardt metoden. Även skillnaden mellan linjär minsta kvadrat anpassning och olinjär minsta kvadrat anpassning. Till sist tillämpas Newton, Gauss Newton och Levenberg–Marquardt metoderna på olika exempel.

Nyckelord: Kurvanpassning, Potens-exponential funktion, Gauss–Newton-algoritm, Levenberg–Marquardt-algoritm, Linjär och ickelinjär kurvanpassning

Acknowledgements

First and Foremost I wish to express my sincere and deepest thanks to my supervisor Karl Lundengård for his guidance and limitless patience. He walked me through all the stages in the progress of writing thesis. Special thanks to the reviewer Thomas Westerbäck and to examiner Christopher Engström at Mälardalens University.

Contents

List of Tables

List of Figures

1 - Introduction

Maths or mathematics is the study of quantities and structures, and space and change. Another viewpoint of mathematics is knowledge, in which we arrive at precise and new results with logical reasoning of the principles (there are other views expressed in the philosophy of mathematics). Although mathematics is not a natural science, the particular structures that mathematicians tend to look at often belong appear in natural science, especially physics, so mathematics are often very useful for solving problems and describing phenomena in the natural sciences. Natural science, engineering, economics, and medicine rely heavily on mathematics, but mathematicians often define methods and structures independently of the sciences that then apply them [1].

1.1 What is curve fitting?

What is modelling with a mathematical function? It means predicting and expressing changes of a quantity based on information about another quantity. Example: Consider the relationship height and weight of humans. We all know that this relationship is not a direct mathematical relation and not a hundred percent understood, a taller person is not necessarily heavier than a shorter person, but it can be said that with a reasonable likelihood, people with a taller height also have more weight. Here, the prediction of weight from height is described using a mathematical function, for example a linear relationship. But how can we choose the model?

We can talk about fitting a curve to a set of points, $\{(x_i, y_i)\}$. For example, x is the height and y is the weight and a model could be that $y = kx + m$, a linear relationship. Curve fitting would then mean choosing k and m such that the models predicts the values of y from x as accurately as possible.

1.2 Methods of curve fitting

The study of relations between quantities is important in many areas. If we have data we might want to describe the relation using a mathematical formula. Finding the right parameter values for such a formula is the basis of curve fitting work. Usually, two variables have different behaviours relative to each other. The pattern of the relationship between two variables may be linear, logarithmic, exponential, partial, and the like. Curve fitting process has two general categories for these behaviours. One curve fitting group is linear and the other group is known as non-linear curve fitting. This type of curve fitting is a method for finding a non-linear model to find the relation between a dependent variable and a set of independent variables.

It is worth noting that this method will not exactly match the data, but it will give an approximation

When we study curve fitting methods, we will divide them into two categories

1. Linear curve fitting
2. Non-linear curve fitting

The most common form of curve-fitting is linear curve fitting that is limited to the fitting of linear models. Contrary to that method, non-linear curve fitting can fit arbitrarily relationships between independent and dependent variables. Non-linear curve fitting is based on numerical algorithms and it is more difficult to compute than linear curve fitting. For a more detailed discussion see Section 2.4.

1.2.1 Linear curve fitting

Suppose we have a set of points (t_1, y_1), (t_2, y_2), ..., (t_n, y_n) and a mathematical function $m(\vec{x}, t)$ whose shape is defined by the values in $\vec{x}$. A linear least squares problem is an unconstrained minimization problem of the form

$$\min_{\vec{x}} f(\vec{x}) = \min_{\vec{x}} \sum_{i=1}^{n} r_i(\vec{x})^2 \tag{1.1}$$

where $r_i(\vec{x}) = y_i - m(\vec{x}, t_i)$ and the function $m(\vec{x}, t)$ depends linearly on x. If the number of independent variables in this linear relationship is more

than one, the curve fitting model is called multiple linear. For example, $m(x,t)$ could be a straight line $m(\vec{x},t) = x_1 t + x_2$. We can also have a linear combination of several functions of t, for example $m(\vec{x},t) = x_1 f_1(t) + x_2 f_2(t) + ... + x_k f_k(t)$.

For a linear least-squares problem (1.2) can be rewritten using vectors as follows

$$\begin{aligned} \min_{\vec{x}} f(\vec{x}) &= \min_{\vec{x}} \sum_{i=1}^{n} r_i(\vec{x})^2 \\ &= F(\vec{x})^T F(\vec{x}) \\ F(\vec{x}) &= \begin{pmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ \vdots \\ f_k(\vec{x}) \end{pmatrix}^T \end{aligned}$$

1.3 Non-linear curve fitting

Linear and non-linear curve-fitting have a lot on common. The definition of a non-linear least square problem is very similar to the definition of a linear lest square problem. We will write it out in full for clarity.

Suppose we have a set of points (t_1, y_1), (t_2, y_2), ..., (t_n, y_n) and a mathematical function $m(\vec{x},t)$ whose shape is defined by the values in $\vec{x}$. A linear least squares problem is an unconstrained minimization problem of the form

$$\min_{\vec{x}} f(\vec{x}) = \min_{\vec{x}} \sum_{i=1}^{n} r_i(\vec{x})^2 \tag{1.2}$$

where $r_i(\vec{x}) = y_i - m(\vec{x}, t_i)$ and the function $m(\vec{x},t)$ depends non-linearly on $\vec{x}$.

This means that there are many more types of functions that could be used in non-linear curve fitting. That is why we often need to use numerical methods to find $\vec{x}$.

1.3.1 Gradient, Jacobian and Hessian

In the coming sections we will discuss optimization problems and algorithms for curve fitting and in all of these subjects we will use the gradient, the Jacobian and the Hessian so we will define these for convenience. We will use the notation $\vec{x} = (x_1, x_2, ..., x_n)$ to denote a row vector.

Definition 1.3.1. *The gradient ∇ of a function f is a vector consisting of the function's partial derivatives:*

$$\nabla f(x) = \nabla f(x_1, x_2, ..., x_n) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \right)^T$$

Definition 1.3.2. *The Jacobian $J(x)$ is a matrix that contains the gradients of the functions $r_1, r_2, ..., r_m$.*

$$J(x) = \left[\frac{\partial r_j}{\partial x_i} \right] = \begin{pmatrix} \nabla r_1(x) \\ \nabla r_2(x) \\ \vdots \\ \nabla r_m(x) \end{pmatrix}, \ j = 1, ..., m, i = 1, ..., n$$

Definition 1.3.3. *The Hessian matrix ∇^2 of a function f is the square matrix of the second partial derivatives.*

$$G(x) = \nabla^2 f(x_1, x_2, ..., x_n) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Note: We only consider function that has continuous second order partial derivatives so that the order of the second partial derivative is not important by Schwarz's theorem, which means that this matrix is symmetrical.

2 - Linear and non-linear optimization

2.1 Linear equation

A linear equation in n unknowns $x_1, x_2, ..., x_n$ is an equation of the form

$$a_1x_1 + a_2x_2 + ... + a_nx_n = b,$$

where $a_1, a_2, ..., a_n$ and b are given real numbers.

A system of m linear equations in n unknowns $x_1, x_2, ..., x_n$ is a system of linear equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n &= b_m \end{aligned}$$

We want determine if such a system has a solution, that is to find out if there exist numbers $x_1, x_2, ..., x_n$ which satisfy all of the equations simultaneously. The system is *consistent* if it has a solution, otherwise the system is called *inconsistent*.

Note that the above system can be written concisely as

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, i = 1, 2, ..., m. \tag{2.1}$$

We can also write the system on matrix form

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}. \tag{2.2}$$

The matrix in (2.2) is called the *coefficient matrix* of the system.

Solving a linear equation system in two or three geometric dimensions, is equivalent to determining whether or not a family of lines or planes, has a common point of intersection [2].

Systems of linear equations are central to almost all optimization algorithms and is a part of many optimization models. Linear equation systems can be used to represent constraints in a model. Finally solving systems of linear equation is an important step in the simplex method for linear programming and Newtons method for non-linear optimization, and is a technique used to determine dual variables (Lagrange multipliers) [2].

2.2 Linear Least Squares Data Fitting

Linear least squares data can be used to fit linear models. We start with an example where we fit a quadratic function.

Example: If we have the same number of points as unknown parameters we solve the problem exactly without using curve-fitting. We can fit a quadratic function to the data in the table below in the following way:[2]

t	2	3	5
b	1	6	4

Quadratic function: $b(t) = x_1 + x_2 t + x_3 t^2$. x_1, x_2, x_3, are unknown parameters

solution:

$$b(t_i) = b_i \Rightarrow \tag{2.3}$$

$$x_1 + x_2 \cdot 2 + x_3 \cdot 2^2 = 1 \Rightarrow x_1 + 2x_2 + 4x_3 = 1 \tag{2.4}$$
$$x_1 + x_2 \cdot 3 + x_3 \cdot 3^2 = 6 \Rightarrow x_1 + 3x_2 + 9x_3 = 6 \tag{2.5}$$
$$x_1 + x_2 \cdot 5 + x_3 \cdot 5^2 = 4 \Rightarrow x_1 + 5x_2 + 25x_3 = 4 \tag{2.6}$$

In this Example, we can write the system of equation in matrix form as:

$$\begin{pmatrix} 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 5 & 25 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 6 \\ 4 \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -21 \\ 14.97 \\ -1.98 \end{pmatrix}$$

$$\big(x_1, x_2, x_3\big) = \big(-21, 14.97, -1.98\big)$$

$$b_i = -21 + 14.97t - 1.98t^2$$

We can also describe fitting of polynomials in a more general way.

Form:

$$y = a + bx + cx^2 \tag{2.7}$$

Data:

$$(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$$

Here we still have the same number of points as unknown parameters so do not need to use least-square fitting.

Least square fitting of a polynomial

If we have more data points than unknown parameters we need to use least squares fitting.

If there were n data points and $k < n$ parameters the model would be of the form

$$y(x) = a_1 + a_2 x + ... + a_n x^{k-1}$$

Then the system would have the form:

$$\begin{pmatrix} 1 & x_1 & \cdots & x_1^{k-1} \\ 1 & x_2 & \cdots & x_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{k-1} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

As an example we can consider the case of $k = 3$.

By principle of least squares, see Section 1.2.1, we want to minimize S (the sum of the squares of the residuals) given by

$$S = \sum_{i=1}^{n} e_i^2 = e_1^2 + e_2^2 + e_3^2 + ... + e_i^2 + ... + e_n^2 \tag{2.8}$$

where

$$e_i = y_i - (a + bx_i + cx_i^2)$$

$$S = \sum_{i=1}^{n} (y_i - a - bx_i - cx_i^2)^2$$

To find the minimum we look values of our unknown parameters that gives zero partial derivatives

$$\frac{\partial S}{\partial a} = 0$$
$$\frac{\partial S}{\partial b} = 0$$
$$\frac{\partial S}{\partial c} = 0$$

We can take the partial derivatives of S given by (2.8), set them to zero and then obtain the following formulas:

$$\sum_{i=1}^{n} y_i = na + b\sum_{i=1}^{n} x_i + c\sum_{i=1}^{n} x_i^2 \tag{2.9}$$

$$\sum_{i=1}^{n} x_i y_i = a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} x_i^2 + c\sum_{i=1}^{n} x_i^3 \tag{2.10}$$

$$\sum_{i=1}^{n} x_i^2 y_i = \sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i^3 + c\sum_{i=1}^{n} x_i^4 \tag{2.11}$$

Equation (2.9), (2.10) and (2.11) are called the normal equations.
Note that (2.9), (2.10) and (2.11) together form a linear equation system with unknowns a, b and c. Solving this equations system we obtain the values a, b, c, the is the solution to the least squares problem. Below is an explicit example.

Example: Find best fit in the least-square sense for values a, b for $y = a+bx$ with given data [3].

i	x	y	xy	x^2
1	0	1	0	0
2	1	1.8	1.8	1
3	2	3.3	6.6	4
4	3	4.5	13.5	9
5	4	6.3	25.2	16
	$\sum x = 10$	$\sum y = 16.9$	$\sum xy = 47.1$	$\sum x^2 = 30$

Solution:

$$y = a + bx \tag{2.12}$$

$$\sum y = na + b\sum x \tag{2.13}$$

$$\sum xy = a\sum x + b\sum x^2 \tag{2.14}$$

$$n = 5,\ \sum x = 10,\ \sum y = 16.9,\ \sum x^2 = 30,\ \sum xy = 47.1,$$

Expanding equation (2.13) gives $16.9 = 5a + 10b$
Expanding equation (2.13) and (2.14)and gives $47.1 = 10a + 30b$

$$47.1 = 10a + 30b \Rightarrow 10a = 47.1 - 30b \Rightarrow a = \frac{47.1 - 30b}{10}$$

By changing the value of a we obtain from equation (2.12) we calculate the value of b in Equation (2.13).

Expanding Equation (2.14) below gives

$$a = \frac{47.1 - 30b}{10}$$

$16.9 = 5a + 10b$
$16.9 = 5\frac{47.1-30b}{10} + 10b$
$16.9 = \frac{47.1-30b}{2} + 10b$
$-10b = 1/2(47.1 - 30b) - 16.9$
$-10b = \frac{47.1}{2} - \frac{30b}{2} - 16.9$
$-10b = 23.55 - 15b - 16.9$
$15b - 10b = 23.55 - 16.9$
$5b = 6.65$
$b = \frac{6.65}{5} = 1.33$

We put in one of the equations (2.11) or (2.12) or (2.13).

$a = \frac{47.1-30\cdot 1.33}{10} = 0.72$

$y = a + bx \Rightarrow y = 0.72 + 1.33x$

The expression we found is not an exact solution to the original linear equation system. We will have errors, but they are minimized in the least square sense [2].

2.3 Non-linear Least Squares curve fitting

We described a non-linear curve fitting problem in Section 1.3. Here we will give an example.

We want the function $m(x, t)$ to be non-linear with respect to at least one of its paramters. An example of this is the exponential decay model:

$$m(x_1, x_2, t) = x_1 e^{-x_2 t} \tag{2.15}$$
$$\frac{\partial m}{\partial x_1} = e^{-x_2 t} = 0$$
$$\frac{\partial m}{\partial x_2} = -t x_1 e^{-x_2 t} = 0$$

From the partial derivatives we can see that $m(x_1, x_,t)$ depends linearly on x_1 but non-linearly on x_2. The partial derivative with respect to x_1, is

independent of x_1, but the partial derivative with respect to x_2 depends on variable x_2 and therefore $m(x_1, x_2, t)$ depends non-linearly on x_2.

First, let's have a definition of the differences of the linear equation with the non-linear equation.

Linear equation

1. The diagram is a straight line.
2. The graph gradient is constant.
3. Solving linear equations is simple and easy.

Non-linear equation

1. All parameters of the linear least squares are linear, but in non-linear terms, at least one parameter is a non-linear equation.
2. Parameters that exist in linear equations are limited, but the parameters that exist in non-linear equations are extensive.
3. Many processes in science and engineering can be described with linear models or other relatively easy models, but there are many other processes that cannot be described with linear models, because intrinsically these processes are non-linear. For example in construction we have concrete strengths that initially increase rapidly, but after a short time turns into a curve the linear model can not describe well. We need to use a non-linear model instead.

Convergence rate

When we use an iterative numerical method to try to solve an equation and we come closer and close to the solution in every iteration we say that the method is convergent and the speed at which the sequence comes closer to its true value is called the convergence rate. If we have a sequence $x_0, x_1, x_2, ..., x_n, ...$ they go to the main root x or, in mathematical language, they say they are converging.

$$x_n \to x \text{ as } n \to \infty$$

If it is true that

$$\lim_{n\to\infty} \frac{|x_{n-1} - x|}{|x_n - x|^b} = \lambda < \infty$$

with

$$b \geq 1$$

then we call b the rate of convergence.

1. if $b = 1$ the sequence converges linearly.
2. if $b = 2$ the sequence converges quadratically.
3. if $1 < b < 2$ the sequence converges super-linearly

Definition 2.3.1. *Let $x^* \in \mathbb{R}$, and consider a sequence x_k for $k = 0, 1, 2, ...$ The sequence x_k is said to converge to x^* if*

$$\lim_{n\to\infty} |x_k - x^*| = 0$$

Type of convergence

1. Linear if $\exists\ c \in [0, 1)$ and there is an integer $K > 0$ such that for $k \geq K$

$$|x_{k+1} - x^*| \leq c|x_k - x^*|;$$

2. Super linear if $c_k \to 0$ and there is an integer $K > 0$ such that for $k \geq K$

$$|x_{k+1} - x^*| \leq c_k|x_k - x^*|;$$

3. Quadratic if $\exists\ c \in [0, 1)$ and there is an integer $K > 0$ such that for $k \geq K$

$$|x_{k+1} - x^*| \leq c|x_k - x^*|^2;$$

Definition 2.3.2 (Global and local convergence)**.** *A locally convergent iterative algorithm converges to the correct answer if the iteration starts close enough. A globally convergent iterative algorithm converges when starting from almost any point.*

Definition 2.3.3 (Global and local minimizer)**.** *x^* is a global minimizer of a function $f : \mathbb{R}^n \to \mathbb{R}$ on a compact domain D if $f(x^*) \leq f(x)$, $\forall x \in \mathbb{R}^n$. x^* is a local minimizer inside a certain region, usually defined as an open (ball) of size δ around x^*, if $(x^*) \leq f(x)$ for $\| x - x^* \|_2 < \delta$.*

2.4 Numerical solutions of Non-linear least squares (NLSQ) problems

A non-linear least squares problem can often be solved numerically using one the following three methods [7]:

1. Newton method
2. Gauss–Newton method
3. Levenberg–Marquardet method

2.4.1 Newton method

The Newton method is based on taking a second-order Taylor approximation of a function $f(\vec{x})$ and then apply the Newton method to solve a non-linear system of equations of the form [9].

$$\nabla f(\vec{x}) = 0$$

We can summarize Newton's method this way: To find an approximation of the root of a function $f(\vec{x})$ we choose a point and draw the tangent line to $f(\vec{x})$ at that point. The point where this tangent line crosses the x-axis gives a new approximation and if this procedure is repeated many times you usually get a good approximation of the root.

Algorithm for the Newton method[6]:

Some notation:

$$\begin{aligned} \nabla f(\vec{x}) &= J(\vec{x})^T r(\vec{x}) \\ \nabla^2 f(\vec{x}) &= (J(\vec{x}_k)^T) J(\vec{x}_k) + S(\vec{x}_k) \\ S(\vec{x}_k) &= \sum_{i=1}^{m} r_i(\vec{x}_k) \nabla^2 r_i(\vec{x}_k) \end{aligned} \tag{2.16}$$

1. Starting from an initial guess $\vec{x}_0$ Note: (if we do not have $\vec{x}_0$, we choose one manually)
2. Compute new approximation

$$\vec{x}_{k+1} = \vec{x}_k - \nabla^2 f(\vec{x}_k)^{-1} \nabla f(\vec{x}_k)$$

3. Compute $\Delta\vec{x}_k = \vec{x}_{k+1} - \vec{x}_k$. If the method is converging then $|\Delta\vec{x}_k|$ is expected to get smaller and smaller. If $|\Delta\vec{x}_k|$ is much smaller that the required accuracy the algorithm have found a good approximation. If $|\Delta\vec{x}_k|$ is still large, repeat step 2 again.

The Newton method is locally quadratically convergent [8]. This means that when we get close to the solution the method will quickly find the solution.

2.4.2 Damped Gauss–Newton method

If $S(\vec{x}_k)$ in (2.16) is small it can be interpreted as the problem being only slightly non-linear near $\vec{x}_k$. Then $S(\vec{x}_k)$ can be ignored and still have quadratic convergence. If $S(\vec{x}_k)$ is large we can get very poor convergence and instead of using the Newton method described before a better alternative is the damped Gauss–Newton method [8].

1. Start initial guess $\vec{x}_0$ for $k = 0, 1, 2, ...$
2. $\triangle\vec{x}_k = -\frac{J(\vec{x}_k^T)r(\vec{x})}{J(\vec{x}_k^T)J(\vec{x}_k)}$
3. Choose a step length c_k so that there is enough descent
   $c_k \in (0, 1)$ or $0 < c_k < 1$. There are several ways to determine appropriate c_k, a common method is to choose a c_k such that

   $$f(\vec{x}_k + c_k\triangle\vec{x}_k) < f(\vec{x}_k) + c_k\nabla f(\vec{x}_k)^T\triangle\vec{x}_k = f(\vec{x}_k) + c_k r(\vec{x}_k)^T J(\vec{x}_k)^T\triangle\vec{x}_k.$$

4. New $\vec{x}_{k+1} = \vec{x}_k + c_k\triangle\vec{x}_k$.
5. Check for convergence similarly to the Newton method described in the previous section.

The Gauss–Newton method can vary a lot in terms of convergence rate. It is more reliable than Newton's method but can be quite a bit slower [8].

2.4.3 Levenberg–Marquardt method

If the Newton method does not converge, or converges too slowly, we use the damped Gauss–Newton approach, and if we do get the right convergence we go to Newton's method. Although we use Gauss–Newton's method for ill-conditioned problem mode, it might not give us satisfactory results since it tends to be slow [10].

One way to keep the faster convergence of the Newton method and the stability of the Gauss–Newton method is to use the Levenberg–Marquardt method [8].

The Levenberg–Marquardt method is based on solving the following equation

$$\begin{aligned}(J(\vec{x}_k)^T J(\vec{x}_k) + \lambda_k I)\triangle\vec{x}_k &= -J(\vec{x}_k)^T r(\vec{x}_k) \\ \Rightarrow \triangle\vec{x}_k &= (J(\vec{x}_k)^T J(\vec{x}_k) + \lambda_k I)^{-1}(-J(\vec{x})_k^T r(\vec{x}_k))\end{aligned}$$

Solving method:

1. Initial guess $\vec{x}_0$ for iterate $k = 0, 1, 2, 3, ...$
2. At each step k choose the Lagrange multiplier λ_k
3. Compute $\triangle\vec{x}_k$ using the formula above.
4. $\vec{x}_{k+1} = \vec{x}_k + \triangle\vec{x}_k$
5. Check for convergence similarly to the Newton method.

There are different methods for choosing λ_k, below a commonly used one is described:

1. The initial value λ_0 as $\|J(\vec{x}_0)^T J(\vec{x}_0)\|_2$
   $\lambda_0 \cong \|J/(\vec{x}_0)^T J(\vec{x}_0)\|_2$
2. Subsequent steps
   $p_k = \frac{\text{actual reduction}}{\text{predicted reduction}} = \frac{f(\vec{x}_{k+1}) - f(\vec{x}_k)}{1/2\triangle\vec{x}_k^T(J(\vec{x}_k)^T r(\vec{x}_k) + \lambda_k\vec{x}_k)}$

If p_k is large then λ_k can be decreased and if p_k is small then λ_k should be increased.

Gauss–Newton search direction is obtained by solving the linear system

$$\nabla^2 f(\vec{x}^*)P = -\nabla f(\vec{x})\nabla F(\vec{x})\nabla F(\vec{x})^T P = -\nabla f(\vec{x})f(\vec{x})$$

If P_k is a large value then Gauss–Newton is good

If $P_k \leq 0$ the λ_{k+1} is increased, therefore used Levenberg–Marquardt's updating.

Levenberg–Marquardt's updating works by checking the following cases:

If $P_k > 0.75$ then $\lambda_{k+1} = \frac{\lambda_k}{3}$.

If $P_k < 0.25$ then $\lambda_k = 2\lambda_k$.

Otherwise $\lambda_{k+1} = \lambda_k$.

The update step should only be performed if $P_k > 0$.

3 - Examples

Example 1

In this example we will illustrate how to apply the Gauss–Newton method on a simple problem with five data points. In the next example we will use the same model on real data with more data points. We had collected data, $\{(t_i, y_i)\}_{i=1}^{m}$ consisting of the size of a population of antelope at various times. Here t_i corresponds to the time at which the population y_i was counted. The model we want to use has the following form:

$$y = x_1 e^{x_2 t}$$

Use initial guess:

$$x = \begin{pmatrix} 2.50 \\ 0.25 \end{pmatrix}$$

t_i	y_i
1	3.2939
2	4.2699
4	7.1799
5	9.3005
8	20.259

We get the least-squares problem :

$$\min_{\vec{x}} f(\vec{x}) = \min_{\vec{x}} \frac{1}{2} \sum_{i=1}^{m} r_i(\vec{x})^2 = \min_{\vec{x}} \frac{1}{2} F(\vec{x})^T F(\vec{x})$$

where F is the vector-valued function

$$F(x) = (f_1(\vec{x})), (f_2(\vec{x}), (f_3(\vec{x}), ..., (f_m(\vec{x}))^T$$

$$F(\vec{x}) = \begin{pmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ \vdots \\ f_m(\vec{x}) \end{pmatrix} = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \\ f_3(x_1, x_2) \\ f_4(x_1, x_2) \\ f_5(x_1, x_2) \end{pmatrix}$$

$$= \begin{pmatrix} x_1 e^{x_2 t_1} - y_1 \\ x_1 e^{x_2 t_2} - y_2 \\ x_1 e^{x_2 t_3} - y_3 \\ x_1 e^{x_2 t_4} - y_4 \\ x_1 e^{x_2 t_5} - y_5 \end{pmatrix} = \begin{pmatrix} x_1 e^{1x_2} - 3 \\ x_1 e^{2x_2} - 4 \\ x_1 e^{4x_2} - 6 \\ x_1 e^{5x_2} - 11 \\ x_1 e^{8x_2} - 20 \end{pmatrix}$$

$\nabla f(\vec{x})$ can be derived using the chain rule:

$$\nabla f(\vec{x}) = \nabla F(\vec{x}) F(\vec{x})$$

$\nabla^2 f(\vec{x})$ can be derived by differentiating this formula:

$$\nabla^2 f(\vec{x}) = \nabla F(\vec{x}) F(\vec{x})^T + \sum_{i=1}^{m} f_i(\vec{x}) \nabla^2 f_i(\vec{x})$$

$$\Rightarrow \min_{x_1, x_2} f(x_1, x_2) = \frac{1}{2} \sum_{i=1}^{5} f(x_1, x_2)^2$$

This formula the least squares objective function is:

$$f(x_1, x_2) = \frac{1}{2} \sum_{i=1}^{5} (x_1 e^{x_2} t - y_i)^2 = \frac{1}{2} F(\vec{x})^T F(\vec{x})$$

$$\nabla f(x_1, x_2) = \begin{pmatrix} \sum_{i=1}^{5} (x_1 e^{x_2 t_i} - y_i) e^{x_2 t_i} \\ \\ \sum_{i=1}^{5} (x_1 e^{x_2 t_i} - y_i) x_1 t_i e^{x_2 t_i} \end{pmatrix}$$

$$(x_1 e^{x_2 t_i} - y_i) \Rightarrow \frac{\partial f(x_1, x_2)}{\partial x_2} = e^{x_2 t_i}$$

$$x_1 t_i e^{x_2 t_i} \Rightarrow \frac{\partial f(x_1, x_2)}{\partial x_2} = x_1 t e^{x_2 t}$$

$$\nabla f(x_1, x_2) = \nabla F(\vec{x}) F(\vec{x})$$

$$\nabla F(\vec{x}) = \begin{pmatrix} e^{x_2 t} & e^{x_2 t_2} & e^{x_2 t_3} & e^{x_2 t_4} & e^{x_2 t_5} \\ x_1 t_1 e^{x_2 t_1} & x_1 t_2 e^{x_2 t_2} & x_1 t_3 e^{x_2 t_3} & x_1 t_4 e^{x_2 t_4} & x_1 t_5 e^{x_2 t_5} \end{pmatrix}$$

So that:

$$\nabla F(x_1, x_2) =$$

$$\begin{pmatrix} e^{x_2 t} & e^{x_2 t_2} & e^{x_2 t_3} & e^{x_2 t_4} & e^{x_2 t_5} \\ x_1 t_1 e^{x_2 t_1} & x_1 t_2 e^{x_2 t_2} & x_1 t_3 e^{x_2 t_3} & x_1 t_4 e^{x_2 t_4} & x_1 t_5 e^{x_2 t_5} \end{pmatrix} \begin{pmatrix} x_1 e^{x_2 t_1} - y_1 \\ x_1 e^{x_2 t_2} - y_2 \\ x_1 e^{x_2 t_3} - y_3 \\ x_1 e^{x_2 t_4} - y_4 \\ x_1 e^{x_2 t_5} - y_5 \end{pmatrix}$$

The Hessian matrix is:

$$\nabla^2 F(x_1, x_2) = \nabla F(\vec{x}) \nabla F(\vec{x})^T + \sum_{i=1}^{m} f_i(\vec{x}) \nabla^2 f_i(\vec{x})$$

$$= \begin{pmatrix} e^{x_2 t} & e^{x_2 t_2} & e^{x_2 t_3} & e^{x_2 t_4} & e^{x_2 t_5} \\ x_1 t_1 e^{x_2 t_1} & x_1 t_2 e^{x_2 t_2} & x_1 t_3 e^{x_2 t_3} & x_1 t_4 e^{x_2 t_4} & x_1 t_5 e^{x_2 t_5} \end{pmatrix} \begin{pmatrix} x_1 e^{x_2 t_1} & x_1 t_1 e^{x_2 t_1} \\ x_1 e^{x_2 t_2} & x_1 t_2 e^{x_2 t_2} \\ x_1 e^{x_2 t_3} & x_1 t_3 e^{x_2 t_3} \\ x_1 e^{x_2 t_4} & x_1 t_4 e^{x_2 t_4} \\ x_1 e^{x_2 t_5} & x_1 t_5 e^{x_2 t_5} \end{pmatrix}$$

$$+ \left(\sum_{i=1}^{5} (x_1 e^{x_2 t_i} - y_i) \right) \begin{pmatrix} 0 & t_i e^{x_2 t_i} \\ t_i e^{x_2 t_i} & x_1 t_i^2 e^{x_2 t_5} \end{pmatrix}$$

Now we assign the values of t, x_1, x_2, y in the obtained formulas.

$$\therefore F(\vec{x}) = \begin{pmatrix} -0.0838 \\ -0.1481 \\ -0.3792 \\ -0.5749 \\ -1.7864 \end{pmatrix}$$

$$\nabla F(\vec{x})^T = \begin{pmatrix} 1.2840 & 3.2101 \\ 1.6487 & 8.2436 \\ 2.7183 & 27.1828 \\ 3.4903 & 43.6293 \\ 7.3891 & 147.7811 \end{pmatrix}$$

$$\nabla f(\vec{x}) = \nabla F(\vec{x}) F(\vec{x}) = \begin{pmatrix} -16,5888 \\ -300,8722 \end{pmatrix}$$

$$\nabla^2 f(\vec{x}) = \nabla F(\vec{x}) \nabla F(\vec{x}) = \begin{pmatrix} 78.5367 & 1335.8479 \\ 1335.8479 & 24559.9419 \end{pmatrix}$$

The Gauss–Newton search direction is obtained by solving the linear system

$$\nabla F(\vec{x}) \nabla F(\vec{x})^T p = -\nabla F(\vec{x}) F(\vec{x})$$

$$p = \begin{pmatrix} 0.0381 \\ 0.0102 \end{pmatrix}$$

New estimate of the solution is

$$x_{new} = x_{old} + p = \begin{pmatrix} 2.50 \\ 0.25 \end{pmatrix} + \begin{pmatrix} 0.0381 \\ 0.0102 \end{pmatrix} = \begin{pmatrix} 2.5381 \\ 0.2602 \end{pmatrix}$$

We will start again with all the steps, but this time we use the new x.

$$\nabla^2 f(\vec{x}) p = -\nabla f(\vec{x})$$

$$\nabla F(\vec{x}) \nabla F(\vec{x})^T p = -\nabla F(\vec{x}) F(\vec{x})$$

Performing computations gives the following system

$$\underbrace{\begin{pmatrix} 78.5367 & 1335.8479 \\ 1335.8479 & 24559.9419 \end{pmatrix}}_{A} p = -\underbrace{\begin{pmatrix} -16.588 \\ -300.8722 \end{pmatrix}}_{B}$$

$$A^{-1} = \frac{1}{\det A} \begin{pmatrix} A_{2,2} & -A_{1,2} \\ -A_{2,1} & A_{1,1} \end{pmatrix}$$

$$\det A = (78.5367 \cdot 24559.9419) - (1335.8479 \cdot 1335.8479) = 144367.1771$$

$$\Rightarrow A^{-1} = \frac{1}{144367.1771} \begin{pmatrix} 24559.9419 & -1335.8479 \\ -1335.8479 & 78.5367 \end{pmatrix}$$

$$\Rightarrow A^{-1} = \begin{pmatrix} 0.17012 & 0.00925 \\ 0.00925 & 0.000544 \end{pmatrix}$$

$$A^{-1}Ap = -A^{-1}B$$

$$p = \begin{pmatrix} 0.17012 & -0.00925 \\ -0.00925 & 0.000544 \end{pmatrix} \begin{pmatrix} 16.588 \\ 300.8722 \end{pmatrix}$$

$$= \begin{pmatrix} 16.588 \cdot 0.17012 + 300.8722 \cdot (-0.00925) \\ 16.588 \cdot (-0.00925) + 300.8722 \cdot 0.000544 \end{pmatrix}$$

Thus we get the new search direction

$$p = \begin{pmatrix} 0.038 \\ 0.010 \end{pmatrix}$$

k	$f(\vec{x}_k)$	$\|\nabla f(\vec{x}_k)\|$
0	$2 \cdot 10^{0}$	$3 \cdot 10^{2}$
1	$4 \cdot 10^{-3}$	$2 \cdot 10^{1}$
2	$2 \cdot 10^{-8}$	$3 \cdot 10^{-2}$
3	$3 \cdot 10^{-9}$	$4 \cdot 10^{-8}$
4	$3 \cdot 10^{-9}$	$3 \cdot 10^{-13}$

Table 3.1: Values of $f(\vec{x})$ and $\|\nabla f(\vec{x}_k)\|$ given by the Gauss–Newton method.

Repeating the above procedure several times the sum of least squares converges towards the minimum. We can see this by looking at the values of $f(\vec{x})$ that we got, see Table 3.1.

We draw the model with the new parameters in Figure 3.1. As we can see the new parameters give a much better fit that the initial guess.

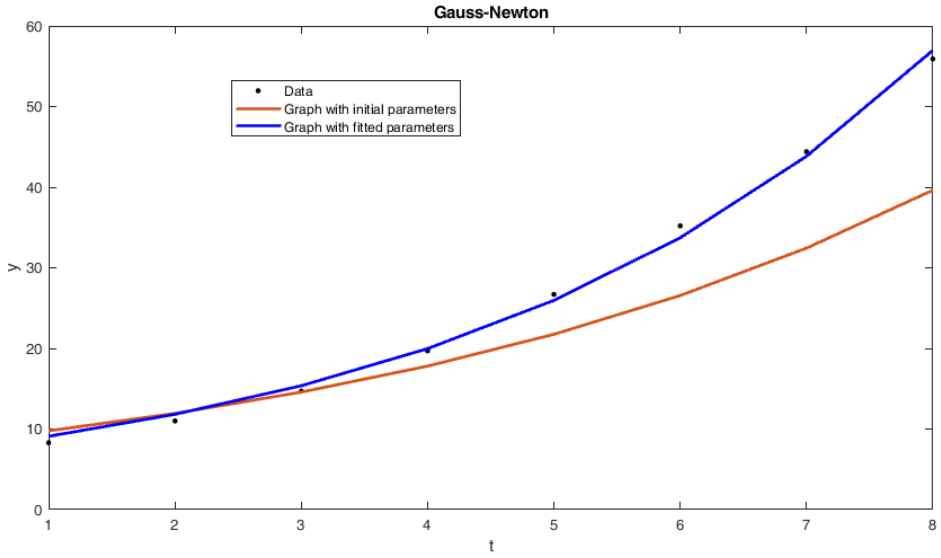

Figure 3.1: Comparison of function graph with initial value of parameters (red) and the new parameters given by the Gauss-Newton algorithm (blue).

Remark on Example 1

We saw that $m(x_1, x_2, t) = x_1 e^{x_2 t}$ was linear with respect to x_1 but non-linear with respect to x_2. In this case we can rewrite this as a linear least square problem. We can do it this way: Begin by setting $g(z_1, z_2, t) = \ln(m(e^{z_1}, z_2, t))$. This gives the function

$$g(z_1, z_2, t) = \ln(e^{z_1} e^{z_2 t}) = z_1 + z_2 t$$

which is linear in both z_1 and z_2. Note that the two least square problems $\min_{\vec{x}} \sum_{i=1}^{n} (y_i - m(x_1, x_2, t))^2$ does not necessarily give an equivalent answer as the problem $\min_{\vec{z}} \sum_{i=1}^{n} (\ln(y_i) - g(z_1, z_2, t))^2$. So while linearizing problems with this kind of model is easy to do, it depends on what problem you actually want to solve. In a way a method like Newton's method can be considered a small, local linearization.

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

Table 3.2: Population data (in millions) for the United States every ten years from 1815 to 1885.

Example 2 (Exponential Data)

Here we will look at some real data and fit an exponential model both with the Gauss–Newton method and with the Levenberg–Marquardt method.

In Table 3.2 is a set of data for the United States population (in millions) and the corresponding year. We will fit an exponential model to this data [4].

The model function is

$$M(x;t) = x_1 e^{x_2 t}$$

graph of the data points and the model with
$x_1 = 6$ and $x_2 = 0.3$ as our initial guesses:

$$r_i(x) = M(x;t_i) - y_i$$

$$r(x) = \begin{pmatrix} 6e^{0.3_1} - 8.3 \\ 6e^{0.3_2} - 11 \\ 6e^{0.3_3} - 14.7 \\ 6e^{0.3_4} - 19.7 \\ 6e^{0.3_5} - 26.7 \\ 6e^{0.3_6} - 35.2 \\ 6e^{0.3_7} - 44.4 \\ 6e^{0.3_8} - 55.9 \end{pmatrix} = \begin{pmatrix} -0.200847 \\ -0.0672872 \\ 0.0576187 \\ 0.220702 \\ 0.190134 \\ 1.09788 \\ 4.59702 \\ 10.2391 \end{pmatrix}$$

the goal is to minimize the least squares problem,

$$f(x) = \frac{1}{2}\|r(x)\|_2^2$$

We find $\|r(x)\|^2$ by adding the squares of all the entries in $r(x)$.

$$\|r(x)\|_2 = \sqrt{x_1^2 + x_2^2 + ... + x_8^2}$$

$$\|r(x)\|_2^2 = 127.309$$

$$f(x) = \frac{1}{2}\|r(x)\|_2^2 = \frac{1}{2} \cdot 127.309 = 63.6545$$

$$J(x) = \frac{\partial r(j)}{\partial x(i)} = \begin{pmatrix} \nabla r_1(x)^T \\ \nabla r_2(x)^T \\ \nabla r_3(x)^T \end{pmatrix}$$

$$j = 1, ..., m$$

$$i = 1, ..., n$$

Because here we have two variables we must take two different partial derivatives

$$M(x;t) = x_1 e^{x_2 t}$$

$$\therefore \frac{\partial M}{\partial(x_1)} = e^{x_2 t_i}$$

$$\therefore \frac{\partial M}{\partial(x_2)} = x_1 \cdot x_2 e^{x_2 t_i}$$

$$J(x) = \begin{pmatrix} e^{x_2} & e^{x_2}x_1 \\ e^{2x_2} & 2e^{2x_2}x_1 \\ e^{3x_2} & 3e^{3x_2}x_1 \\ e^{4x_2} & 4e^{4x_2}x_1 \\ e^{5x_2} & 5e^{5x_2}x_1 \\ e^{6x_2} & 6e^{6x_2}x_1 \\ e^{7x_2} & 7e^{7x_2}x_1 \\ e^{8x_2} & 8e^{8x_2}x_1 \end{pmatrix} = \begin{pmatrix} 1.34986 & 8.09915 \\ 1.82212 & 21.8654 \\ 2.4596 & 44.2729 \\ 3.32012 & 79.6828 \\ 4.48169 & 6010.1 \\ 6.04965 & 217.787 \\ 8.16617 & 342.979 \\ 11.0232 & 529.112 \end{pmatrix}$$

The rest of the steps are just the multiplication of the transpose matrix and finally, to find the value of p, just like the previous example we use the inverse matrix A^{-1} and the value of p, obtained.

$$J^T Jp = -J^T r(x)$$
$$D \cdot CP = -D \cdot B \Rightarrow Ap = +E$$
$$D \cdot C = A$$
$$-D \cdot B = E$$
$$\Rightarrow A^{-1}Ap = E \cdot A^{-1} \Rightarrow p = ok$$
$$A^{-1} \cdot A = 1$$

For this first iteration p_1, and the approximations of x_1 and x_2 can now be updated:

$$p_1 = (0.923529, -0.0368979)$$
$$x_{1_1} = 6 + 0.923529 = 6.92353$$
$$x_{2_1} = 3 - 0.0368979 = 0.263103$$

$$r(x) = \begin{pmatrix} -0.200847 \\ -0.0672872 \\ 0.0576187 \\ 0.220702 \\ 0.190134 \\ 1.09788 \\ 4.59702 \\ 10.2391 \end{pmatrix} = B$$

$$J(x) = \begin{pmatrix} 1.34986 & 8.09915 \\ 1.82212 & 21.8654 \\ 2.4596 & 44.2729 \\ 3.32012 & 79.6828 \\ 4.48169 & 6010.1 \\ 6.04965 & 217.787 \\ 8.16617 & 342.979 \\ 11.0232 & 529.112 \end{pmatrix} = C$$

$$J(x)^T = \begin{pmatrix} 1.34986 & 1.82212 & 2.4596 & 3.32012 & 4.48169 & 6.04965 & 8.16617 & 11.0232 \\ 8.09915 & 21.8654 & 44.2729 & 79.6828 & 6010.1 & 217.787 & 342.979 & 529.112 \end{pmatrix} = D$$

We perform the steps of the Gauss-Newton method like above two more times and after the third time we have these values

$$x_{1_3} = 7.00009$$
$$x_{2_3} = 0.262078$$

Comparing the values we get in each iteration with each other like we did in the previous example we can see that the method has converged.

We use Levenberg–Marquardt to solve the same problem but get better precision and better convergence. First we assume a small value of $\triangle$:

$$\triangle = 0.05$$

Next we check the value of p:

$$p = 0.924266$$

The information we got before is larger than $\triangle$, that's mean $p > \triangle$, as a result, we use the formula:

$$\frac{1}{2}\|Jp + r\|^2$$

and also for p, we use formula:

$$\frac{1}{2}\|J_k p_k + r_k\|^2$$

to obtain $f(x)$ we use formula:

$$f(x) = \frac{1}{2}\sum_{1}^{m} r^2(x)$$

If the $f(x)$ is large, then we go to Equation:

$$(J_k^T J_k + \lambda I)p_k^{LM} = -J_k^T r_k$$

and we obtain the equation for λ. We assume the value of 1 in the first, and then the values x_1 and x_2, we get we see that the resulting value of p, is smaller than the old p. This means the error rate is lower. So let's consider

$$\lambda = 0.1$$

We repeat the steps above and find new values and we continue to do so until

$$p < \triangle$$

Then our answer is correct. Our x, which was obtained in the above example for the third time is $p_3 = 0.000401997$, and this is smaller than $\triangle$, so we do not continue the solution, and we obtain from this new p and x, which is

$$x_{1_3} = 7.00005, x_{2_3} = 0.262079, f(x) = 3.00654$$

In this case both methods give a good result. We can see that Levenberg–Marquardt is more stable by choosing initial values $x_1 = 6$, $x_2 = 1.5$. After six iterations with Gauss–Newton we get

$$x_{1_6} = -2607.9, x_{2_6} = 437.35$$

which is far from the results and if we do more iterations the parameters keep getting worse and worse.

Using Levenberg–Marquardt we get a better result, after six iterations we have

$$x_{1_6} = 0.87651, x_{2_6} = 0.49621$$

which is much better. After fourteen iterations we have

$$x_{1_{14}} = 6.99995, x_{2_{14}} = 0.262051$$

which is close to the good answer we had before.

Example 3

In this example, we examine the central mortality rate, $\mu(t)$ where t is the age of death, of men in Sweden. We want to fit a mathematical model to estimated central mortality rate for men in Sweden 2010. In this example, the data will be modelled using the following function

$$\mu(t) = \frac{c_1 e^{c_2 t}}{t} + e^{a_1}(a_2 t e^{-a_2 t})^{a_3}.$$

This function was fitted using the Levenberg–Marquardt method. The central mortality rate was estimated by dividing the deaths at a certain age divided by the total population at that age.

To use the Levenberg–Marquardt method we need the function and its partial derivatives:

$$\mu(t) = \frac{c_1 e^{c_2 t}}{t} + e^{a_1}(a_2 t e^{-a_2 t}).$$

$$\frac{d\mu}{dc_1} = \frac{e^{c_2 t}}{t}.$$

$$\frac{d\mu}{dc_2} = t\frac{c_1 e^{c_2 t}}{t} = c_1 e^{c_2 t}.$$

$$\frac{d\mu}{da_1} = e^{a_1 t} \cdot (a_2 t \cdot e^{-a_2 t}))^{a_3}$$

$$\frac{d\mu}{da_2} = -e^{a_1} \cdot ((a_2 t \cdot e^{-a_2 t})^{a_3})) \cdot a_3 \cdot (a_2 t - 1)/a_2$$

$$\frac{d\mu}{da_3} = e^{a_1} \cdot ((a_2 t \cdot e^{-a_2 t})^{a_3} \cdot \ln(a_2 t \cdot e^{-a_2 t}))$$

We start with parameters $c_1 = 0.0004, c_2 = 0.12, a_1 = 13, a_2 = 1/25, a_3 = 20$ and use the Levenberg–Marquardt method to fit to the data from 2010. We get new parameter values $c_1 = 0.0002, c_2 = 0.1275, a_1 = 12.4929, a_2 = 0.0384, a_3 = 19.9733$. As the mathematical steps are clearly shown in the example by the Levenberg–Marquardt method, see section 2.5.3. I show the result of this example in graphic form in Figure 3.2. Here we can see that the initial parameter values overestimated the mortality rate but the parameters found by the Levenberg–Marquardt method fits quite well.

Choosing parameters that are far from the appropriate values makes it much harder for the method to find a good fit, even for a relatively reliable method like Levenberg–Marquardt. An example when we start with initial values $c_1 = 0.4, c_2 = 1.2, a_1 = 0.13, a_2 = 0.3, a_3 = 2$ is shown in Figure 3.3 and here the method improves the fit but is still far from optimal.

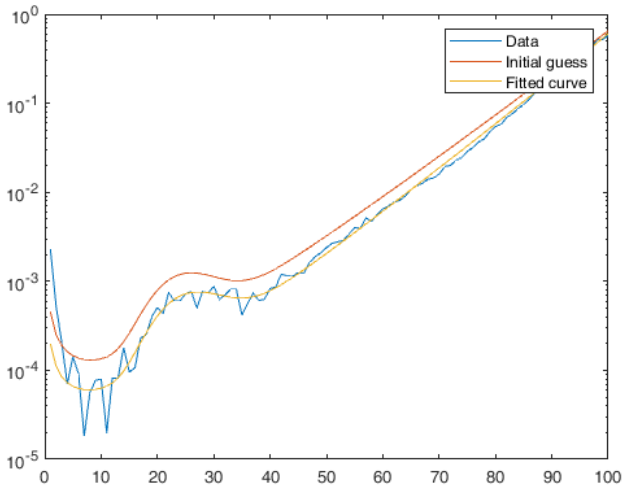

Figure 3.2: Comparison of function graph with initial value of parameters (red) and the parameters when fit with the Levenberg–Marquardt algorithm (yellow) and the estimated mortality rate for men in Sweden 2010 (blue). The initial parameters were chosen to be $c_1 = 0.0004, c_2 = 0.12, a_1 = 13, a_2 = 1/25, a_3 = 20$ and the method gives new parameter values $c_1 = 0.0002, c_2 = 0.1275, a_1 = 12.4929, a_2 = 0.0384, a_3 = 19.9733$ that fits the data well.

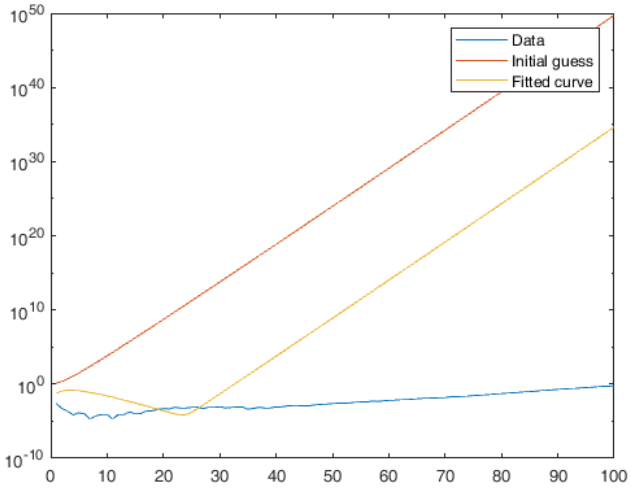

Figure 3.3: Comparison of function graph with initial value of parameters (red) and the parameters when fit with the Levenberg–Marquardt algorithm (yellow) and the estimated mortality rate for men in Sweden 2010 (blue). The initial parameters were chosen to be $c_1 = 0.4, c_2 = 1.2, a_1 = 0.13, a_2 = 0.3, a_3 = 2$ and the method gives new parameter values $c_1 = 0, c_2 = 1.2, a_1 = 0.13, a_2 = 0.3, a_3 = 2$ so even though the method is relatively reliable it cannot manage when the initial values of the parameters are too far from a good fit.

4 - Conclusions

In this thesis, we came described some of the most commonly used method for non-linear curve-fitting. In engineering sciences and other sciences, most of its processes are non-linear or curve linear. We have demonstrated that, using the Newton method, the Gauss–Newton or the Levenberg–Marquardt method, we can get answers that are correct and accurate.

Newton's method is not guaranteed to give good solution. The method can give different solutions depending on the initial guess which is chosen manually. When the Newton method does not give an acceptable root we can use the Gauss–Newton method instead. But when we use the Gaussian–Newton method to solve problem it can be slow. To avoid these weaknesses, we can use the Levenberg–Marquardt method, in Example 2 we saw that sometimes the Levenberg–Marquardt is reliable but still gives a good answer after not too many iterations.

4.1 Future Work

The various methods presented in this thesis all discuss non-linear curve fitting. This topic can be considered important for future research in analysis of non-linear phenomena. These methods are suitable for the study and analysis of many different mathematical data in various sciences. There are also many other methods for non-linear curve fitting, for example the Nelder–Mead method [13] or Powell's method [14]. Comparing the properties of the methods discussed in this degree project and the properties of other methods discussed could also be very valuable.

5 - Summary of reflection of objectives in the thesis

5.1 Objective 1: Knowledge and Understanding

In this thesis I have learned a lot about non linear curve fitting from articles in scientific journals and mathematical textbooks. I described various methods for non linear curve fitting from the literature, including the Newton method, Gauss–Newton method and the Levenberg–Marquardt method. There are other methods for non linear curve fitting, I focused on analysing these three methods and examples of how they can be used to get a precise understanding. I also saw that there are many applications that build on these techniques in regression and modelling but they are outside the scope of this thesis.

5.2 Objective 2: Methodological knowledge

I can more clearly say that I learned very useful applied mathematics. I used many of the skills I learned from the master's programme in Engineering mathematics. I am now much more capable and interested in learning more and deeper mathematics and their applications. I now see applied mathematics as the trunk of a tree where the branches are different engineering applications. I have also learned more about using the MATLAB programming tool.

5.3 Objective 3: Critically and Systematically Integrate Knowledge

I tried to use the resources in this thesis mainly from journal articles and books available at the Mälardalen University Library Portal and online databases. I have also worked with my supervisors to organise and choose appropriate material. A lot of the work was condensing and improving the notes I had taken while studying the different methods on my own.

5.4 Objective 4: Independently and Creatively Identify and Carry out Advanced Tasks

Working on this degree project I learned a lot about how to choose the right content and improve the my writings. I examined what methods were commonly used in research and decided what to write about after discussion with my supervisors, primarily with Karl Lundengård. One important thing I have learned is to focus and specify my work.

5.5 Objective 5: Present and Discuss Conclusions and Knowledge

With regard to this thesis I presented many results from various sources. I have worked hard to make it easy for people with a basic knowledge of applied mathematics and linear algebra to understand this thesis. I believe the thesis is interesting for mathematical enthusiasts and people interested in applications of mathematics. I have tried to show both the simplicity and the complexity of these topics in using the methods discussed in the thesis.

5.6 Objective 6: Scientific, Social and Ethical Aspects

In this dissertation, we use algorithms and three general approaches to non-linear curve fitting. All methods have been described before but I have chosen my own way to describe them, based on what I found in the literature and discussion with my supervisor. The significant and important subject in this thesis is to understand the importance of non-linear and linear issues and how they appear in applications. In modern applied mathematics it is very important to understand the properties of different methods so that you can advise those who work in other areas so that they choose appropriate tools.

A - MATLAB code Example 3

Here is the MATLAB code used to fit the function using the Levenberg–Marquardt method. It uses an implementation of the Levenberg–Marquardt method included in the Optimization Toolbox.

Code written in cooperation with Karl Lundengård.

A.1 fit_to_mortality_rate_Sweden_levenberg.m

```
% These are the ages (in years) that we consider
ages = (1:100)';

% Here we import data from SCB (statistiska centralbyrån)
data_pop_men = ...
  xlsread('scb_befolkning_man_1968-2016',1,'D4:AZ103','basic');
data_deaths_men = ...
  xlsread('scb_dodsfall_man_1968-2016',1,'D4:AZ103','basic');

% To avoid problems with 0 deaths when taking the logaritm
% we set any point with 0 deaths to 0.1
data_deaths_men_adjusted = data_deaths_men;
data_deaths_men_adjusted(data_deaths_men_adjusted == 0) = 0.1;

% Here we estimate the mortality rate from data
mortality_rate_men = data_deaths_men_adjusted./data_pop_men;

% Here you choose year, we have data from 1968 to 2016
year = 2010;
mu_measured = mortality_rate_men(ages,year-1967);
```

```
% Here we define our model for the mortality rate
mu = @(c1,c2,a1,a2,a3) c1*exp(c2*ages)./ages + ...
                       exp(a1)*(a2*ages.*exp(-a2*ages)).^a3;

% This is the initial guess. Here we use parameters
% hand-fitted to be in the ball-park for Sweden 1985
% The parameters are written on the format
% p0 = [c1,c2,a1,a2,a3]
p0 = [0.0004,0.12,13,1/25,20];

% This calls a method that fits the parameters using the
% Levenberg-Marquardt method, see lsm_fit_mortality_rate_Sweden.m
% for more details
pfit = lsm_fit_mortality_rate_Sweden(ages,mu_measured,p0);

% Here we use the model to compute the mortality rate from
% initial guess and from the final result
initial = mu(p0(1),p0(2),p0(3),p0(4),p0(5));
result = mu(pfit(1),pfit(2),pfit(3),pfit(4),pfit(5));

% Plot the results
figure(1)
plot(ages,mortality_rate_men(ages,year-1967),ages,initial,ages,result)
legend('Data','Initial guess','Fitted curve')

% Plot the results using logarithmic axis so that it is easier to see
% what happens for young ages
figure(2)
semilogy(ages,mortality_rate_men(ages,year-1967),ages,initial,ages,result)
legend('Data','Initial guess','Fitted curve')
```

A.2 lsm_fit_mortality_rate_Sweden.m

```
% ------------------------------------------- %
% Function that fits a function described in the %
% f_res_and_jacobian_mortality_rate_Sweden.m to  %
% the data given by x,y.                      %
% ------------------------------------------- %
function pfit = lsm_fit_mortality_rate_Sweden(x,y,p0)
```

```
size(x)
size(y)
% Input:
% x - n x 1 vector with data on x-axis.
% y - n x 1 vector with data on x-axis.

% Output:
% pfit - 1 x m vector with b-parameters of fitted function.

% Here we set MATLAB to use the Levenburg-Marquardt
% method with a Jacobian we calculate ourselves.
% Here is two ways of setting the options if one does
% not work with your MATLAB function try the other one
% options = optimoptions('lsqnonlin','Algorithm','levenberg-marquardt',...
%                        'Jacobian','on');
options = optimset('Algorithm','levenberg-marquardt','Jacobian','on',...
                   'MaxFunEvals',20000,'MaxIter',100000,'TolX',1e-12,...
                   'Display','off');

% Here we use the built-in 'lsqnonlin' to
% find the non-linear least-square fitting
pfit = lsqnonlin(@(p)f_res_and_jacobian_mortality_rate_Sweden...
                     (p(1),p(2),p(3),p(4),p(5),x,y),...
                 p0,[],[],options);

end
```

A.3 f_res_and_jacobian_mortality_rate_Sweden

```
function [res,J] = ...
  f_res_and_jacobian_mortality_rate_Sweden(c1,c2,a1,a2,a3,x,y)
% ------------------------------------------ %
% Function that calculates the residuals and  %
% the Jacobian for a function in a format that %
% fits lsqnonlin                              %
% ------------------------------------------ %

% Input:
% b - 1 x m vector of parameters that define model
```

```
% x - n x 1 vector with data on x-axis.
% y - n x 1 vector with data on x-axis.

% Output:
% res - n x 1 vector of residuals.
% J - n x m matrix partial derivatives with respect
%     to the b-parameters at corresponding times.

%************************************************%
% Here we use the function                       %
% f(x) = c1*exp(c2*x)./x                         %
%        + exp(a1)*(a2*x.*exp(-a2*x)).^a3;       %
%************************************************%

% Function and partial derivatives
f = @(x) c1*exp(c2*x)./x + exp(a1)*(a2*x.*exp(-a2*x)).^a3;
dfdc1 = @(x) exp(c2*x)./x;
dfdc2 = @(x) c1*exp(c2*x);
dfda1 = @(x) exp(a1)*(a2*x.*exp(-a2*x)).^a3;
dfda2 = @(x) -exp(a1)*(a2*x.*exp(-a2*x)).^a3.*a3.*(a2*x-1)/a2;
dfda3 = @(x) exp(a1)*(a2*x.*exp(-a2*x)).^a3.*log(a2*x.*exp(-a2*x));

% Calculates Jacobian
J = [dfdc1(x) dfdc2(x) dfda1(x) dfda2(x) dfda3(x)];

% Calculates residuals
res = (f(x)-y);

end
```

Bibliography

[1] Gh. S. Hamedani. *Encyclopedia Academic Mathematics.* Mathematical publishing, Tehran, 2002.

[2] Igor Griva, Stephen G. Nash, Ariela Sofer. *Linear and non–linear optimization.* SIAM, 2008.

[3] N.P. Bali. K.L. Sai Prasad. *A textbook of engineering mathematics-III*, University Science Press, 2017.

[4] E. Malczewski. *The Population of the United States* University of Illinois. (mste.illinois.edu/malcz/ExpFit/data.html), accessed on 2019-05-01.

[5] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Non-linear Equations.* SIAM, Philadelphia, 1996.

[6] H. B. Keller and V. Pereyra. *Finite Difference Solution of Two-Point BVP. Preprint 69.* Department of Mathematics, University of Southern California, 1976.

[7] A. Griewank and A. Walther. *Principles and Techniques of Algorithmic Differentiation.* Evaluating Derivatives: Other Titles in Applied Mathematics 105. Second edition, SIAM, Philadelphia, 2008.

[8] P. C. Hansen, V. Pereyra and G. Scherer. *Least Squares Data Fitting with applications.* The John Hopkins University Press, Baltimore, 2013.

[9] V. Pereyra. *Iterative methods for solving non-linear least squares problems.* SIAM J. Numer. Anal. 4:27–36, 1967.

[10] *Netlib Repository at UTK and ORNL [Online]* http://www.netlib.org.

[11] P. C. Hansen. *Inverse Problems – Insight and Algorithms.* Discrete SIAM, Philadelphia, 2010.

[12] S. J. Wright, J. N. Holt. *Algorithms for non-linear least squares with linear inequality constraints.* SIAM J. Sci. and Stat. Comput. 6:1033–1048, 1985.

[13] J. A. Nelder, R. Mead. *A simplex method for function optimization.* Computer Journal 7(4): 308-313, 1965.

[14] M. J. D. Powell. *An efficient method for finding the minimum of a function of several variables without calculating derivatives* Computer Journal 7(2): 155-162, 1964.