Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

The choices of weights in the iterative convex minorant algorithm

(De gewichtskeuzes in het iteratieve convexe minoranten algoritme)

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

door

Jos Smalbil

Delft, Nederland
January 2015

BSc verslag TECHNISCHE WISKUNDE

The choices of weights in the iterative convex minorant algorithm

(De gewichtskeuzes in het iteratieve convexe minoranten algoritme)

Jos Smalbil

Technische Universiteit Delft

Begeleider

Prof.dr.ir. G. Jongbloed

Overige commissieleden

Dr.ir. M. Keijzer Prof.dr.ir. A.W. Heemink

January, 2015 Delft

Contents

Abstract

In statistics one often encounters the problem of estimating a function based on a given dataset. Sometimes shape properties such as monotonicity of the function are known. This property can be used in a non-parametric regression model. The iterative convex minorant(ICM) algorithm can be used to compute an estimate of a convex regression function. In the ICM algorithm positive weights can be chosen arbitrarily. In this thesis we describe the (solution of the) isotonic regression problem, explain the ICM algorithm, describe the convex regression problem and present a simulation study to assess the effect of the choice of weights.

Acknowledgments

I would like to acknowledge my supervisor Prof.dr.ir. G. Jongbloed who helped me throughout the making of this thesis. I appreciate his expert knowledge of the subject of this thesis. His motivational and encouraging skills have helped me a lot during the development of this thesis. Most of all I would like to thank him because of the time he took to guide me and his patience during our conversations.

I would also like to thank my family. I want to thank my parents for their moral and financial support during my study. I appreciate the mental support from my brother and I recognize the importance of the support of my sister who helped me during study hours in the library.

1 Introduction

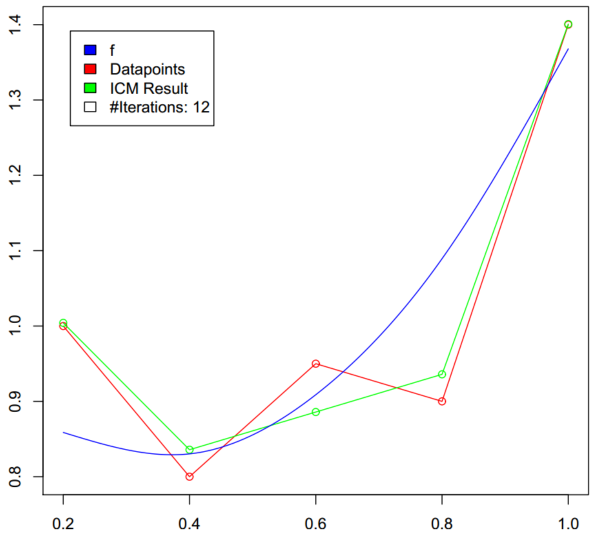

Figure 1.1: ICM example

In classical statistics one often has the problem of estimating the underlying distribution that generated a given dataset. Such estimation problems can be approached parametrically, where the distribution of interest is assumed to be contained in a class of distributions that can be parametrized by a low-dimensional parameter vector, and non-parametrically. Non-parametric problems do not rely on assumptions that the probability distribution belongs to a 'small' parametric class of distributions. However there can be made some assumptions about the shape of the underlying probability density function (or regression function). For example it is sometimes known that a regression function is differentiable or concave/convex. Other possible shape constraints are uni- or bimodality.

The iterative convex minorant (ICM) algorithm is an algorithm that can be used to compute shape constrained estimates in various regression and inverse problems. The algorithm is very useful for the non-parametric estimation of density functions in statistical inverse problems. The algorithm is used when a convex function must be 'minimized' over a specific convex cone in the n-dimensional Euclidean space. It concerns the set

$$C = \{\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n : x_1 \leq x_2 \leq \cdots \leq x_n\}. \tag{1.1}$$

The basic principle originates from the theory of isotonic regression. Isotonic regression is a statistical analysis technique for estimating an isotonic (increasing) function that is observed with random errors. Computing the least squares isotonic regression estimate leads to a specific convex optimization problem. A well known solving algorithm for the isotonic regression problem is the pool adjacent violators algorithm (PAVA).

For more general convex optimization problems over C, that is, for minimizing convex functions more general than the isotonic regression criterion function over the 'monotone cone' C, the ICM algorithm can be used. The core of the ICM algorithm is to sequentially approximate a general convex function by a function of the 'isotonic regression type' as described in the next chapter Def. 2.2.

The aim of this thesis is to investigate the effect of the choices in weights one can make in the ICM algorithm. First we describe the isotonic regression problem and the characterization of the solution of this problem in Chapter 2. Then we describe the ICM algorithm in Chapter 3. In Chapter 4 we show that the ICM algorithm can be used for computing the least squares estimator in the convex regression problem. Subsequently we investigate the choices in weights one has to choose within the ICM algorithm in Chapter 5.

2 Isotonic regression

2.1 Introduction

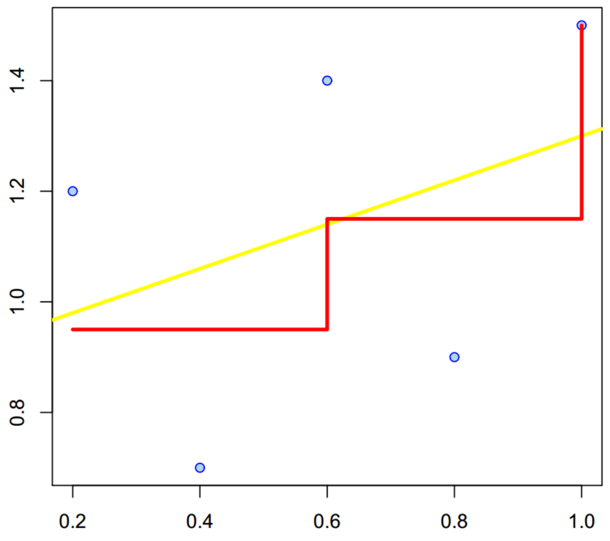

Figure 2.1: Isotonic regression example: Datapoints (blue), isotonic regression solution (red), linear regression solution (yellow).

The theory of isotonic regression concerns the observation of isotonic functions with random errors. Isotonic regression is part of the general regression theory. The regression function of Y on X is the conditional expectation of the random variable Y, given an explanatory variable X. The regression line of Y on X is a fit on Y by a linear function of X in the sense of least squares. The regression solution is then called the linear regression of Y on X. In this chapter we will focus on isotonic or monotonic regression because we will show in the following chapters that the ICM algorithm is based on the method of solving this kind of problem. Note that the assumption of monotonicity is an assumption which is useful in a wide range of applications. For example when evaluating the life-time of factory machines, the failure rate can be assumed to be a non-decreasing function due to aging. Before we give a solution of the isotonic regression problem we will define an isotonic function and then give two slightly different definitions of the isotonic regression problem.

Definition 2.1 (Isotonic function) $f : \mathbb{R}^n \to \mathbb{R}$ is an isotonic function if $\forall\, x, y \in \mathbb{R}^n$ we have $f(x) \leq f(y)$ whenever x is majorized by y, that is $x_i \leq y_i$ for $1 \leq i \leq n$. Note that these kind of functions are more widely known as Schur-convex functions or order-preserving functions. [12]

Consider an unknown isotonic function f with $f : [0, 1] \to \mathbb{R}$ and a dataset $\{(x_i, z_i) \in [0, 1] \times \mathbb{R} : x_1 \leq x_2 \leq \ldots \leq x_n, 1 \leq i \leq n\}$ where each z_i is a realization of the dependent random variable $Z_i = f(x_i) + \epsilon_i$ with $\epsilon_i \sim N(0, \sigma^2)$. The problem of weighted least squares isotonic regression is then defined as

Definition 2.2 (Isotonic regression) minimize the quadratic function

$$Q(f) = \frac{1}{2} \sum_{i=1}^{n} (f(x_i) - y_i)^2 w_i \tag{2.1}$$

over all isotonic functions f (Def. 2.1) where the weights $w_i \in (0, \infty)$ are arbitrary.

Note that in this definition the objective function Q only depends on the function f via its values at the datapoints x_i. Therefore, only those values, collected in the vector $(f(x_1), \ldots, f(x_n))$, are of interest in the minimization and minimizing Q over all isotonic functions boils down to minimizing a function over the monotone cone C (Eq. 1.1) in the n-dimensional Euclidean space. In the next section we will give a graphical interpretation of the solution of the isotonic regression problem. It will be seen that the solution can be characterized as the derivative of the greatest convex minorant of a diagram of points.

2.2 Greatest convex minorant

First we will give some basic definitions about the meaning of convex.

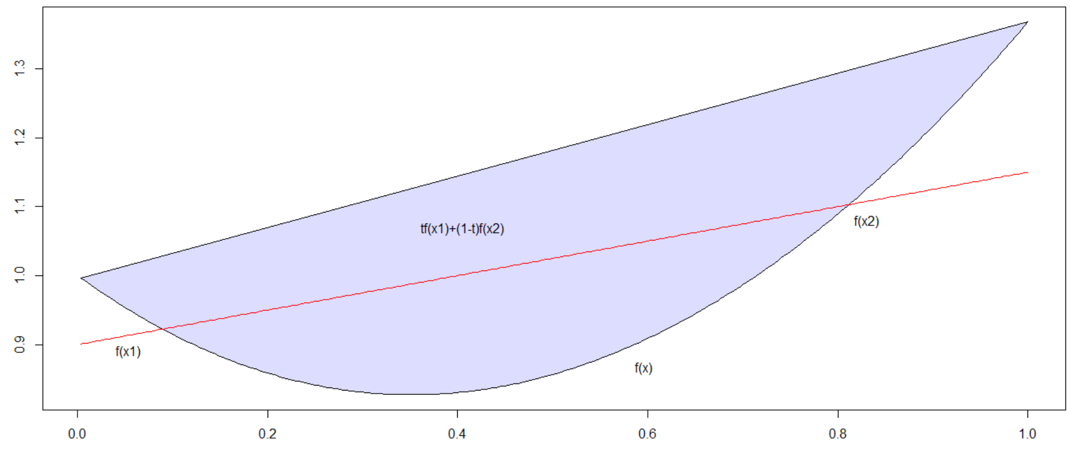

Figure 2.2: Convex function

Definition 2.3 (Convex set) $K \in \mathbb{R}^n$ is a convex set if $(1-t)x + ty \in K \; \forall \; t \in [0,1]$ and for any $x, y \in K$.

Definition 2.4 (Convex cone) $K \in \mathbb{R}^n$ is a convex cone if $ax + by \in K \; \forall \; a, b \in [0, \infty)$ and for any $x, y \in K$.

Definition 2.5 (Convex function)

- $f\colon \mathbb{R} \to \mathbb{R}$ is a convex function if $f(t_1 x + t_2 y) \leq t_1 f(x) + t_2 f(y)$ when $t_1 + t_2 = 1 \; \forall \; x, y \in \mathbb{R}$ and $t_1 \in [0,1]$.
- f is a convex function if and only if the epigraph, the set of points on or above the graph, of f is a convex set.
- Graphically speaking, f is convex if the line segment between any two points of the function f lies on or above the graph.

In order to solve the isotonic regression problem as described in the last paragraph it is useful to define the greatest convex minorant of a set of points in $\mathbb{R}^2$.

Definition 2.6 Let $P \subset \mathbb{R}^2$ be a set of points and define the set of convex minorants by

$$M = \{f\colon \mathbb{R} \to \mathbb{R} : f \text{ is convex and } f(x) \leq y \forall (x,y) \in P\}.$$

Then the greatest convex minorant (GCM) of P is defined by

$$GCM(x) := \sup_{f \in M} f(x) \text{ for all } x \in \mathbb{R}.$$

The resulting graph of the GCM is also known as the Newton diagram or the Newton-Puiseux polygon. [10]

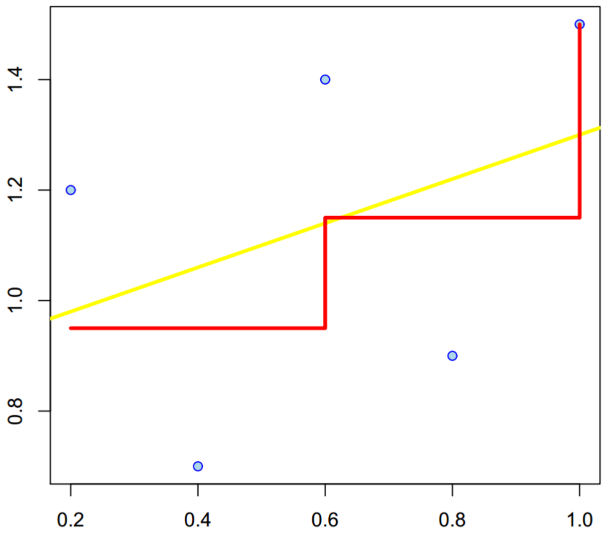

Figure 2.3: Isotonic regression example: Datapoints (blue), isotonic regression solution (red), linear regression solution (yellow).

Now we can characterize the solution of the isotonic regression problem in terms of the derivative of the greatest convex minorant of a set of points. (Section 3.5)

Theorem 2.7 *Consider the isotonic regression problem as described in Def. 2.2. Define the cumulative sum diagram (CSD) P of $\mathbf{y}$ with weights $\mathbf{w}$ as $P_0 = (0,0)$ and $P_j = \left(\sum_{i=1}^{j} w_i, \sum_{i=1}^{j} y_i w_i\right) = (W_j, Y_j)$, for $j = 1, 2, \ldots, n$. Then, $\hat{\mathbf{f}}$ is the optimal solution of the isotonic regression problem if and only if $\hat{f}_i = \left.\frac{\partial_- GCM(P)}{\partial x}\right|_{x=W_i}$ for $i = 1, \ldots, n$, that is, the left derivate of the greatest convex minorant of the cumulative sum diagram of $\mathbf{y}$ with weights $\mathbf{w}$, evaluated at the point $W_i = \sum_{j=1}^{i} w_j$.*

Thus, for example, the solution of the isotonic regression problem in Figure 2.3 is the left derivative of the blue line in Figure 2.4.

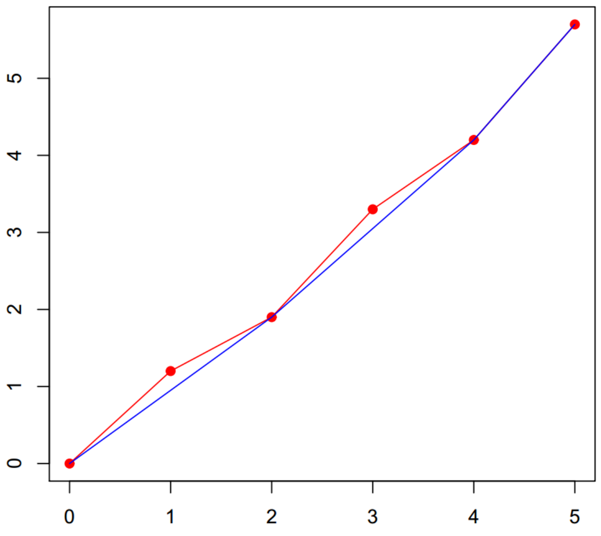

Figure 2.4: Isotonic regression: The cumulative sum diagram of the datapoints in figure 2.2 (red), the greatest convex minorant of the CSD (blue).

2.3 Pool adjacent violators algorithm

To find the greatest convex minorant of a set of points a simple iterative algorithm, the pool adjacent violators algorithm (PAVA) can be used. The idea of the algorithm is to start with the first point and then to iteratively draw a line to the next point. When the convexity of the resulting graph is violated, the last points are 'pooled' until the non-convexity of the graph is removed. The result is thus a convex graph which is the greatest minorant of the given set of points.

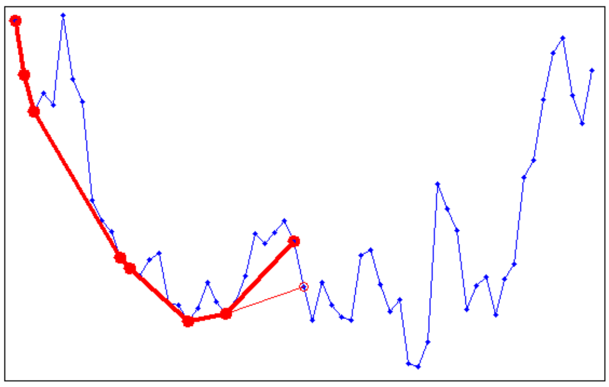
Figure 2.5: Pool adjacent violators algorithm

In Figure 2.5 we see that the convexity of the red graph is violated if the next point of the diagram is added to the graph. To solve this problem the algorithm omits the last point and draws a line to the next point of the diagram immediately. Now the graph is convex again and the algorithm continues.

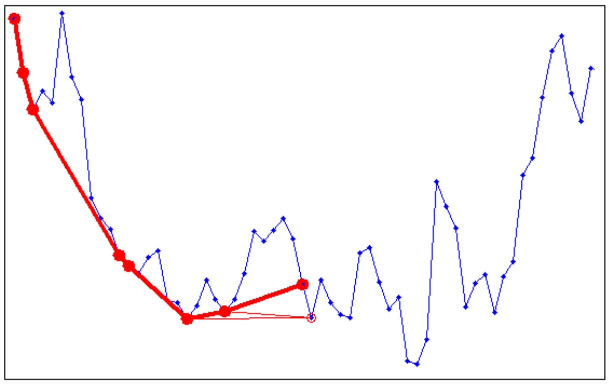

Figure 2.6: Pool adjacent violators algorithm

In Figure 2.6 the algorithm again finds that the next point of the diagram results in a non-convex graph. Thus the last point is again omitted and a line is drawn to the next point of the graph. However now it turns out that the graph is still non-convex. Now a second line is drawn from an earlier point of the convex graph. We see that this again results in a convex graph and the algorithm continues.

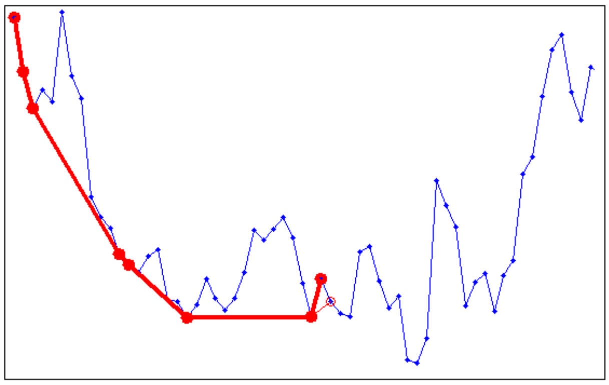

Figure 2.7: Pool adjacent violators algorithm

In Figure 2.7 we see that the algorithm has found a point which maintains the convexity of the graph, so it has added this point to the convex graph. However the next point immediately turns out to make the graph non-convex again and so this point is omitted and a line is drawn to the next point.

3 The ICM algorithm

3.1 Introduction

The iterative convex minorant (ICM) algorithm is an algorithm that uses the solution of the isotonic regression problem as described in Chapter 2 to sequentially estimate the solution of a more general convex optimization problem. More precisely, the ICM algorithm can be used for convex optimization problems over the convex cone C defined in Eq. 1.1. A specific application of the algorithm is the convex regression problem as described in Chapter 4. The ICM algorithm was first described by P. Groeneboom and J. A. Wellner in 1992 [5]. Convergence properties of the ICM algorithm with line search were derived by G. Jongbloed in 1998 [3].

The basic idea of the ICM algorithm is to define a starting value $\mathbf{x}_0 \in C$ (Eq. 1.1). Then, the ICM algorithm consists of three main steps. The first step is to define an approximate optimization problem (Section 3.2). The second step is to solve this problem (Section 3.3). The last step is to execute a line search (Section 3.4). These three steps are repeated until a suitable stop condition is met.

3.2 Step 1 - Define approximating isotonic regression problem

The first step of the ICM algorithm is to define an approximate optimization problem. Given a convex optimization problem over the convex cone C, one can construct an approximate optimization problem over the monotone cone C. In Chapter 4 we will see how a least squares problem over another convex cone can be rewritten as a quadratic optimization problem over C using a reparametrization.

The ICM algorithm starts with choosing a starting value $\mathbf{q}^0 \in C$ (Eq. 1.1). Then the approximation function is defined as:

$$\begin{aligned}\phi(\mathbf{q}|\mathbf{q}^0) &= \phi(\mathbf{q}^0) + (\mathbf{q}-\mathbf{q}^0)^T\nabla\phi(\mathbf{q}^0) + 1/2(\mathbf{q}-\mathbf{q}^0)^T W(\mathbf{q}-\mathbf{q}^0)\\ &= \mathit{constant} + 1/2(\mathbf{q}-\mathbf{q}^0 - W^{-1}\nabla\phi(\mathbf{q}^0))^T W(\mathbf{q}-\mathbf{q}^0 - W^{-1}\nabla\phi(\mathbf{q}^0))\\ &= \mathit{constant} - (\mathbf{q}-\mathbf{q}^0)^T\nabla\phi(\mathbf{q}^0) + 1/2(\mathbf{q}-\mathbf{q}^0)^T W(\mathbf{q}-\mathbf{q}^0)\end{aligned}$$

Minimizing this function is a quadratic optimization problem. If W is chosen to be a positive definite diagonal matrix it is an isotonic regression problem. Note that at $\mathbf{q}^0$ the value of $\phi(.|\mathbf{q}^0)$ as well as it's partial derivatives coincide with those of ϕ; $\phi(.|\mathbf{q}^0)$ is a first order approximate of ϕ at $\mathbf{q}^0$. Although it is not made explicit in the notation, the diagonal matrix W may be taken dependent on $\mathbf{q}^0$ as well.

3.3 Step 2 - Solve problem

The second step of the ICM algorithm is to solve the defined approximate optimization problem. Note that this problem is an isotonic regression problem as described in Def. 2.2. Thus, by Theorem 2.7, the solution is the left derivative of the greatest convex minorant of the cumulative sum diagram of $\mathbf{y}$ with weights $\mathbf{w}$. The cumulative sum diagram can be found by projecting $\mathbf{q}^{(0)} + W^{-1}\nabla\phi(\mathbf{q}^{(0)})$ on C with weights W. In this case the CSD will be:

$$\left(\sum_{i=1}^{j} w_i, \sum_{i=1}^{j}\left[q_i^{(0)} + \frac{1}{w_i}\frac{\partial\phi}{\partial q_i}(q^{(0)})\right] w_i\right) = \left(\sum_{i=1}^{j} w_i, \sum_{i=1}^{j} q_i^{(0)} w_i + \frac{\partial\phi}{\partial q_i}(q^{(0)})\right)$$

The greatest convex minorant can then be found by using the PAV algorithm as described in Section 2.3. The solution $q^{(1)}$ is then the left derivative of the greatest convex minorant.

3.4 Step 3 - Line search

In Jongbloed (1998), [3], it is shown that by adding an appropriate line search in every iteration of the ICM algorithm will yield an algorithm that will always converge. The line search boils down to finding $\lambda \in (0,1]$ and corresponding $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda(\mathbf{x}_i + \mathbf{x}_{new})$ such that $Q(\mathbf{x}_i + \lambda(\mathbf{x}_i + \mathbf{x}_{new}))$ is minimal. Note that this line search is only performed when there is no sufficient monotonicity in the resulting value of ϕ, i.e. if $\phi(\mathbf{x}_{new})$ is not sufficiently smaller than $\phi(\mathbf{x}_i)$. The resulting algorithm is also known as the modified iterative convex minorant algorithm (MICM) or the damped convex minorant algorithm (DICM). This algorithm can also be seen as a special case of the generalized gradient projection (GGP) optimization scheme.

3.5 Optimality conditions

Consider the convex optimization problem over a convex cone and assume that the following conditions hold:

(i) ϕ is continuous and attains its minimum over the monotonic cone C (Eq. 1.1) at a unique point $\hat{x}$;
(ii) ϕ is continuously differentiable on the set $\{\mathbf{x} \in \mathbb{R}^n : \phi(x) < \infty\}$.

Write $\nabla\phi$ for the gradient of ϕ and $(\cdot,\cdot)$ for the usual inner product in $\mathbb{R}^n$. It is known from Robertson, Wright and Dykstra (1988, sec 6.2) [4] , that $\hat{x}$ is the solution of this optimization problem if and only if $\hat{x} \in C$ satisfies the Fenchel optimality conditions

$$(\hat{x}, \nabla\phi(\hat{x})) = 0 \text{ and } (x, \nabla\phi(\hat{x})) \geq 0 \ \forall\ x \in C. \tag{3.1}$$

A useful example for ϕ is when it is of the quadratic form as in Eq. 2.1.

$$\phi(x) = 1/2 \sum_{i=1}^{n} (x_i - y_i)^2 w_i.$$

If we now take for x $x = 1_K = (0,0,0,\ldots,1,1,1)$ for $K = 1 \ldots n$ we can write

$$\nabla\phi(x) = \begin{pmatrix} (x_1 - y_1)w_1 \\ \cdots \\ (x_n - y_n)w_n \end{pmatrix}$$

and $\langle 1_K, \nabla\phi(\hat{x})\rangle = \sum_{i=K}^{n} \nabla\phi(\hat{x}) = \sum_{i=K}^{n} (\hat{x}_i - y_i)w_i = \sum_{i=K}^{n} \hat{x}_i w_i - \sum_{i=K}^{n} y_i w_i$.

We see that the last element in this equation is the cumulative sum of y_i with weights w_i which is also being used in Theorem 2.7 to construct the solution of the isotonic regression problem.

3.6 Constant multiple of weights

In Section 3.3 we have seen the cumulative sum diagram of the isotonic regression problem with weights $(w_1, \ldots, w_n)$. In this section we will do the same but with a constant multiple of these weights. This in order to get an idea of the effects of the constant multiple on the number of iterations and the number of line searches in the ICM algorithm.

We project $\mathbf{q}_0 + cW^{-1}\nabla\phi(\mathbf{q}_0)$ on C with weights cW. The CSD is:

$$\left(\sum_{i=1}^{j} cw_i, \sum_{i=1}^{j} \left[q_i^{(0)} + \frac{1}{cw_i}\frac{\partial\phi}{\partial q_i}(q^{(0)})\right] cw_i\right) = \left(\sum_{i=1}^{j} cw_i, \sum_{i=1}^{j} q_i^{(0)} cw_i + \frac{\partial\phi}{\partial q_i}(q^{(0)})\right)$$

Note that we can divide both coordinates by the constant c without changing the derivative of the GCM of the CSD at the points of which the CSD consists. Then the CSD is:

$$\left(\sum_{i=1}^{j} w_i, \sum_{i=1}^{j} q_i^{(0)} w_i + \frac{1}{c}\frac{\partial\phi}{\partial q_i}(q^{(0)})\right)$$

In our simulation study Section 5.4 we will study the effect of different values for c.

4 Convex regression

4.1 Introduction

Consider an unknown convex function f on $[0,1]$ and a simply ordered dataset

$$\{(x_i, z_i) \in [0,1] \times \mathbb{R} : x_1 \leq x_2 \leq \ldots \leq x_n, 1 \leq i \leq n\}$$

where each z_i is a realization of the dependent random variable $Z_i = f(x_i) + \epsilon_i$ with $(\epsilon_i)_{i=1}^n \sim N(0, \sigma^2)$ i.i.d.. The problem of non-parametric convex regression is then to estimate the function f with the weighted least squares method. This is defined as:

Definition 4.1 (Convex regression) Minimize the quadratic function

$$Q(f) = \sum_{i=1}^{n} (f(x_i) - y_i)^2 w_i$$

over all convex functions f Def. 2.5 and with $w_i \in (0, \infty)$.

4.2 Reparametrization

To solve this specific optimization problem we will first reparametrize this problem as an optimization problem over the monotone cone C (Eq. 1.1).

Write $f_i = f(x_i)$, $\mathbf{f} = (f_1, \ldots, f_n)^T$ and $\mathbf{y} = (y_1, \ldots, y_n)^T$. Then we can rewrite Eq. 4.1 in vector notation as

$$Q(\mathbf{f}) = (\mathbf{f} - \mathbf{y})^T W (\mathbf{f} - \mathbf{y})$$

where W is a symmetric positive definite $n \times n$ matrix with diagonal entries $W(i,i) = w_i$ and off-diagonal entries zero.

To reparametrize this equation so that the feasible region becomes the monotone cone C, we write $f_i = f_1 + \sum_{j=1}^{i-1} \Delta_j q_j$, with $f_1 \in \mathbb{R}$, $\Delta_j = x_{j+1} - x_j$ and $q_j = \frac{f_{j+1} - f_j}{xj+1 - x_j}$, the slope from f_j to f_{j+1}. This is a parametrization of f in terms of $f_1, q_1, q_2, \ldots, q_{n-1}$. In vector notation this is equivalent to $\mathbf{f} = f_1\mathbf{1} + Kq$ where

$$K = \begin{pmatrix} 0 & 0 & \ldots & 0 \\ \Delta_1 & 0 & \ldots & 0 \\ \Delta_1 & \Delta_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & 0 \\ \Delta 1 & \Delta_2 & \ldots & \Delta_{n-1} \end{pmatrix}$$

is a $n \times (n-1)$ matrix and $\mathbf{q} = (q_1, \ldots, q_{n-1})^T$. Thus, Def. 4.1 is equivalent to the following definition:

Definition 4.2 minimize the quadratic function

$$\widetilde{Q}(\mathbf{q}, f_1) = (f_1\mathbf{1} + K\mathbf{q} - \mathbf{y})^T W (f_1\mathbf{1} + K\mathbf{q} - \mathbf{y})$$

over the monotone cone C (Eq. 1.1).

We can simplify this problem by eliminating the argument f_1 from the minimization. To this end, we fix $\mathbf{q}$ in Eq. 4.3 and then minimize for f_1. Because the terms of Eq. 4.3 without f_1 are negligible when differentiating to f_1 we consider the following function of f_1, with $\mathbf{q}$ fixed:

$$Q_{\mathbf{q}}(f_1) = 2(f_1\mathbf{1})^T WK\mathbf{q} + f_1^2 \mathbf{1}^T W\mathbf{1} - 2f_1\mathbf{1}^T W\mathbf{y} = f_1^2(\mathbf{1}^T W\mathbf{1}) + 2f_1(\mathbf{1}WK\mathbf{q} - \mathbf{1}^T W\mathbf{y})$$

Then it is clear that $\frac{\partial \widetilde{Q}(\mathbf{q},f_1)}{\partial f_1} = \frac{dQ_{\mathbf{q}}(f_1)}{df_1}$, thus it is sufficient to solve

$$\frac{dQ_{\mathbf{q}}(f_1)}{df_1} = 0 \Leftrightarrow f_1(\mathbf{1}^T W\mathbf{1}) = \mathbf{1}^T W(\mathbf{y} - K\mathbf{q}) \Leftrightarrow f_1 = \frac{\mathbf{1}^T W(\mathbf{y} - K\mathbf{q})}{\mathbf{1}^T W\mathbf{1}}$$

So we have found f_1 as an explicit function of $\mathbf{q}$. Now define $\hat{f}_1$ as the f_1 where the function $\widetilde{Q}(\mathbf{q}, f_1)$ (Eq. 4.3) attains its minimum for a fixed $\mathbf{q}$ as follows:

$$\hat{f}_1 = \hat{f}_1(\mathbf{q}) = \operatorname*{argmin}_{f_1} Q_q(f_1) = \operatorname*{argmin}_{f_1} \widetilde{Q}(\mathbf{q}, f_1)$$

Now we can write the optimization function as a function of one variable.

$$Q : C \longmapsto \mathbb{R}$$

defined by

$$\begin{aligned}
Q(\mathbf{q}) &= (\hat{f}_1\mathbf{1} + K\mathbf{q} - \mathbf{y})^T W(\hat{f}_1\mathbf{1} + K\mathbf{q} - \mathbf{y}) - \mathbf{y}^T W\mathbf{y} \\
&= (\hat{f}_1\mathbf{1} + K\mathbf{q} - \mathbf{y})^T(\hat{f}_1 W\mathbf{1} + WK\mathbf{q} - W\mathbf{y}) - \mathbf{y}^T W\mathbf{y} \\
&= (\hat{f}_1^2\mathbf{1}^T W\mathbf{1} + 2\hat{f}_1(\mathbf{1}^T WK\mathbf{q} - \mathbf{1}^T W\mathbf{y}) + \mathbf{q}^T K^T WK\mathbf{q} - \mathbf{q}^T K^T W\mathbf{y} - \mathbf{y}^T WK\mathbf{q} \\
&= \frac{(\mathbf{1}^T W(\mathbf{y} - K\mathbf{q}))^2}{\mathbf{1}^T W\mathbf{1}} - 2\frac{(\mathbf{1}^T W\mathbf{y} - \mathbf{1}^T WK\mathbf{q})^2}{\mathbf{1}^T W\mathbf{1}} + \mathbf{q}^T K^T WK\mathbf{q} - 2\mathbf{y}^T WK\mathbf{q} \\
&= \mathbf{q}^T K^T WK\mathbf{q} - \frac{(\mathbf{1}^T W(\mathbf{y} - K\mathbf{q}))^2}{\mathbf{1}^T W\mathbf{1}} - 2\mathbf{y}^T WK\mathbf{q} \\
&= \mathbf{q}^T K^T WK\mathbf{q} - \frac{1}{\sum w_i}(\mathbf{y}^T W\mathbf{1}\mathbf{1}^T W\mathbf{y} - 2\mathbf{y}^T W\mathbf{1}\mathbf{1}^T WK\mathbf{q} + \mathbf{q}^T K^T W\mathbf{1}\mathbf{1}^T WK\mathbf{q}) \\
&= \mathbf{q}^T(K^T WK - \frac{1}{\sum w_i}K^T W\mathbf{1}\mathbf{1}^T WK)\mathbf{q} - (2\mathbf{y}^T WK - \frac{1}{\sum w_i}2\mathbf{y}^T W\mathbf{1}\mathbf{1}^T WK)\mathbf{q} \\
&= \mathbf{q}^T(K^T(W - \frac{1}{\sum w_i}W\mathbf{1}\mathbf{1}^T W)K)\mathbf{q} - 2\mathbf{y}^T(W - \frac{1}{\sum w_i}W\mathbf{1}\mathbf{1}^T W)K\mathbf{q} \\
&= \mathbf{q}^T A\mathbf{q} - \mathbf{b}^T\mathbf{q} \\
&= (\mathbf{q} - {}^1\!/\!_2 A^{-1}\mathbf{b})^T A(\mathbf{q} - {}^1\!/\!_2 A^{-1}\mathbf{b})
\end{aligned}$$

where $A = K^T(W - \frac{1}{\sum w_i}W\mathbf{1}\mathbf{1}^T W)K$ and $\mathbf{b} = 2K^T(W - \frac{1}{\sum w_i}W\mathbf{1}\mathbf{1}^T W)\mathbf{y}$.

This leads us to the following definition:

Definition 4.3 minimize the quadratic function

$$Q(\mathbf{q}) = (\mathbf{q} - {}^1\!/\!_2 A^{-1}\mathbf{b})^T A(\mathbf{q} - {}^1\!/\!_2 A^{-1}\mathbf{b})$$

over the monotone cone C (Eq. 1.1).

Remark that this definition is equivalent to the isotonic regression problem as described in Def. 2.2 if and only if A is a (symmetric positive definite) $n \times n$ matrix with diagonal entries $A(i,i) = w_i$ and off-diagonal entries zero, where $w_i \in (0, \infty)$.

Lemma 4.4 (Positive semidefiniteness) *Define* $A = K^T(W - \frac{1}{\sum w_i}W\mathbf{1}\mathbf{1}^T W)K$. *Then* $\mathbf{z}^T A\mathbf{z} \geq 0\ \forall\ \mathbf{z} \in \mathbb{R}^n$.

Proof First note that if $A = U^T U$, then $z^T A z = z^T U^T U z = ||Uz||^2 > 0$ if U is of full rank.

Write $W - \frac{1}{\sum w_i} W \mathbf{1}\mathbf{1}^T W = \begin{pmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \dots & \dots & w_i & \dots \\ 0 & 0 & \dots & w_n \end{pmatrix} - \frac{1}{\sum w_i} \begin{pmatrix} w_1 \\ \dots \\ w_i \\ \dots \\ w_n \end{pmatrix} (w_1, \dots, w_i, \dots, w_n)$

$= \begin{pmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \dots & \dots & w_i & \dots \\ 0 & 0 & \dots & w_n \end{pmatrix} - \frac{1}{\sum w_i} \begin{pmatrix} w_1^2 & w_1 w_2 & \dots & w_1 w_n \\ w_2 w_1 & w_2^2 & \dots & w_2 w_n \\ \dots & \dots & w_i^2 & \dots \\ w_n w_1 & w_n w_2 & \dots & w_n^2 \end{pmatrix}$

$= \begin{pmatrix} w_1(1-\alpha w_1) & -\alpha w_1 w_2 & \dots & -\alpha w_1 w_n \\ -\alpha w_1 w_2 & w_2(1-\alpha w_2) & \dots & -\alpha w_2 w_n \\ \dots & \dots & w_i(1-\alpha w_i) & \dots \\ -\alpha w_1 w_n & -\alpha w_2 w_n & \dots & w_n(1-\alpha w_n) \end{pmatrix}$

$= X$ where $\alpha = \frac{1}{\sum w_i}$

We see that the matrix X is symmetric so it has real eigenvalues. If we now apply the Gersghorin circle theorem [7] we can see in the illustration below that all the eigenvalues are nonnegative, because the Gersghorin circle discs lies completely on the positive real axis. For example the first Gersghorin circle disc is $D\left(w_1(1-\alpha w_1), \sum_{i=2\dots n} |-\alpha w_1 w_i|\right)$ where $w_1(1-\alpha w_1)$ is the center of the disc and $\sum_{i=2\dots n} |-\alpha w_1 w_i|$ the radius. Also note that $\sum_{i=2\dots n} |-\alpha w_1 w_i| = \alpha w_1 \sum_{i=2\dots n} w_i = w_1(1-\alpha w_1)$. Thus, we can write $X = V^T V$ and $A = K^T V^T V K = (VK)^T V K = B^T B$. ∎

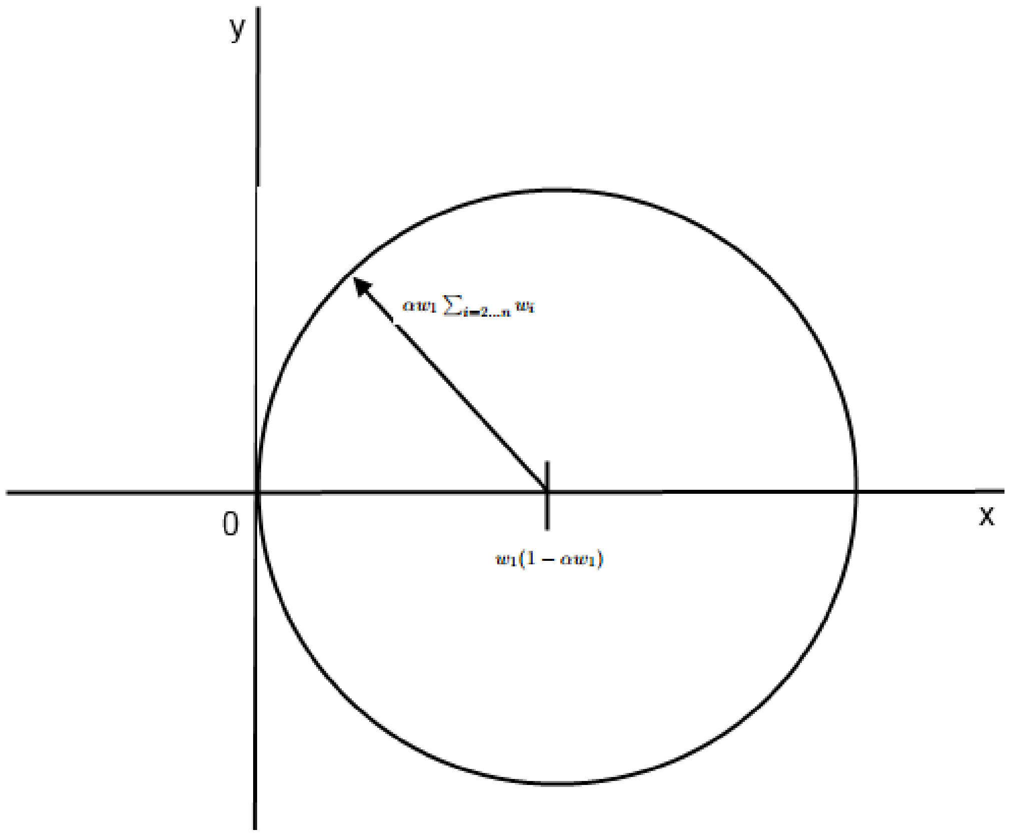

Figure 4.1: Gersghorin circle disc

Corollary 4.5 *The function Q defined in Def. 4.3 is convex on C.*

5 Choices in weights

5.1 Introduction

In this chapter we will discuss the choices in weights within the ICM algorithm. The simplest choice is where all the weights are taken equal to one. Intuitively the most natural choice is to choose the weights equal to the diagonal of the Hessian matrix of the convex optimization problem, because in case the Hessian is diagonal it corresponds to Newton's algorithm. The Hessian matrix is the matrix of second-order partial derivatives of the convex optimization function. We will compare the effectiveness of the ICM algorithm for these choices of weights and will also empirically try to determine if there are better choices for the weights. First we will do two simulation studies and then we will apply them in real-world data examples.

5.2 Simulation example

First we will give an example of a simulation. We choose $f_3(x) = e^{-x} + x^2$, $n = 100$ and Hessian diagonal weights. As stopping criterion we choose $abs(sum(\nabla\phi)) < 10^{-5}$ and $all(cumsum(\nabla\phi) < 10^{-5}$ which corresponds to the Fenchel optimality conditions (Eq. 3.1).

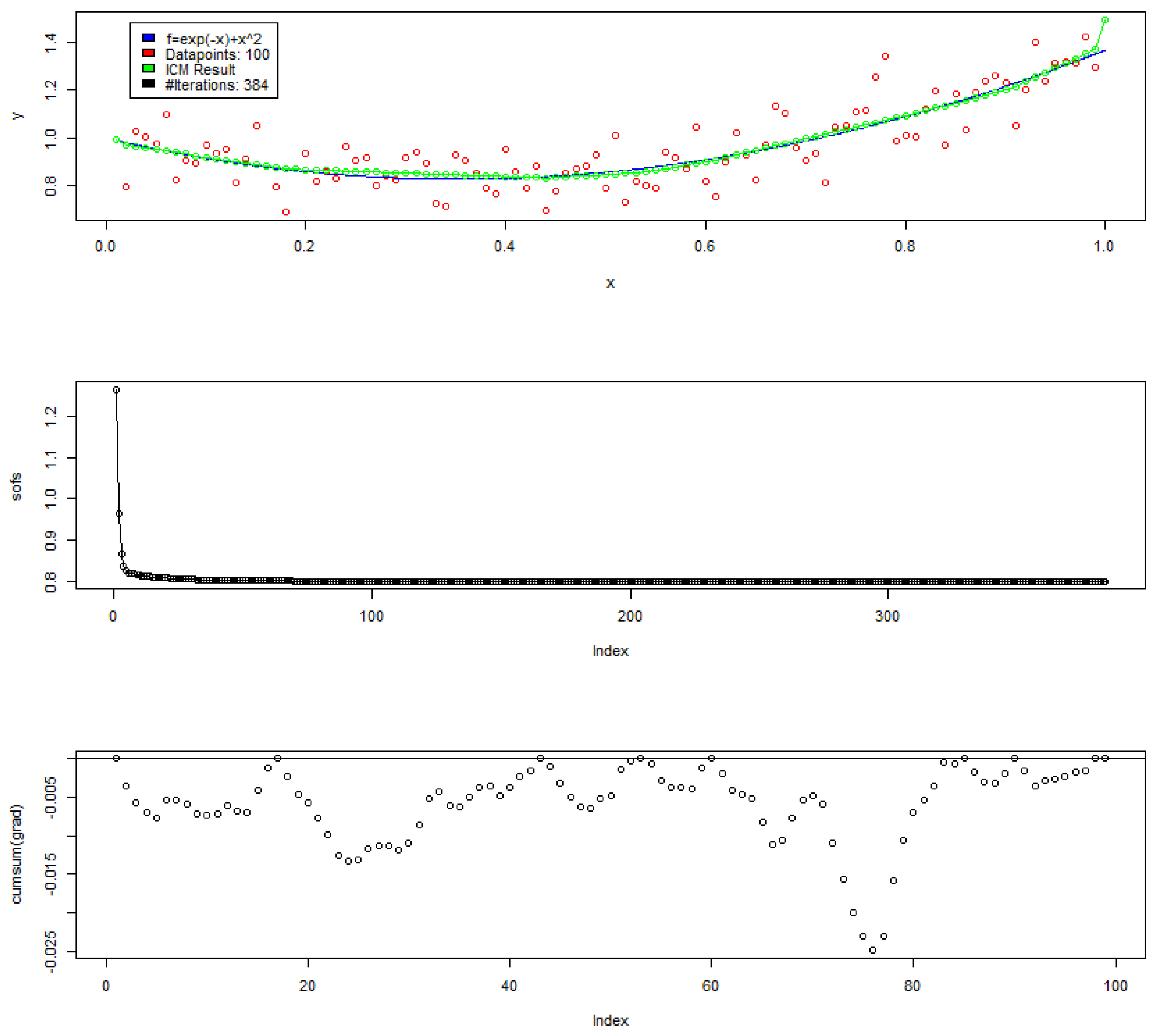

Figure 5.1: ICM Algorithm Example: A plot of the ICM result (green), the given datapoints (red) and the original function $f_3 = e^{-x} + x^2$ (blue) (above). A plot of the sum of squares against the number of iterations (mid). A plot of cumsum(grad), the cumulative sum of the gradient, to check whether the solution satisfies the Fenchel optimality conditions (below).

We see that this example required 384 iterations to satisfy the stopping criteria. The total running time was 1.56 seconds.

5.3 Simulation study 1

For this simulation study we will choose three different convex regression functions, namely $f_1(x) = x$, $f_2(x) = x^2$ and $f_3(x) = e^{-x} + x^2$. To determine the efficiency of the ICM algorithm depending on the sample size we will use three different sample sizes, $n = 10$, $n = 100$ and $n = 1000$. For the weights we will first use the weights equal to one and the weights equal to the Hessian matrix diagonal and then try to empirically find better choices for the weights. Thus, for each function we will run the ICM algorithm with the three different sample sizes and with at least two different choices of weights. This will result in at least 18 simulation results. For each simulation we will take the same stopping criterion and then determine the simulation running time and the number of iterations. Our main goal is to determine which choices of weights give the fastest results.

We will now use the same method as in the example to run the simulations and the results are shown in the following tables.

Function:	$f_1 = x$	$f_1 = x$	$f_1 = x$	$f_1 = x$	$f_1 = x$	$f_1 = x$
# of datapoints:	$n = 10$	$n = 10$	$n = 100$	$n = 100$	$n = 1000$	$n = 1000$
Weights:	$ones$	$diag(H)$	$ones$	$diag(H)$	$ones$	$diag(H)$
# of iterations:	841	24	23812	405	-	5024
Total time:	0.39s	0.00s	16.77s	1.815s	-	1120s

Function:	$f_2 = x^2$	$f_2 = x^2$	$f_2 = x^2$	$f_2 = x^2$	$f_2 = x^2$	$f_2 = x^2$
# of datapoints:	$n = 10$	$n = 10$	$n = 100$	$n = 100$	$n = 1000$	$n = 1000$
Weights:	$ones$	$diag(H)$	$ones$	$diag(H)$	$ones$	$diag(H)$
# of iterations:	389	26	30310	502	-	4791
Total time:	0.17s	0.00s	21.79s	1.765s	-	1069s

Function:	$e^{-x} + x^2$	$e^{-x} + x^2$	$e^{-x} + x^2$	$e^{-x} + x^2$	$e^{-x} + x^2$	$e^{-x} + x^2$
# of datapoints:	$n = 10$	$n = 10$	$n = 100$	$n = 100$	$n = 1000$	$n = 1000$
Weights:	$ones$	$diag(H)$	$ones$	$diag(H)$	$ones$	$diag(H)$
# of iterations:	902	49	28654	384	-	4621
Total time:	0.39s	0.03s	20.46s	2.255s	-	1032s

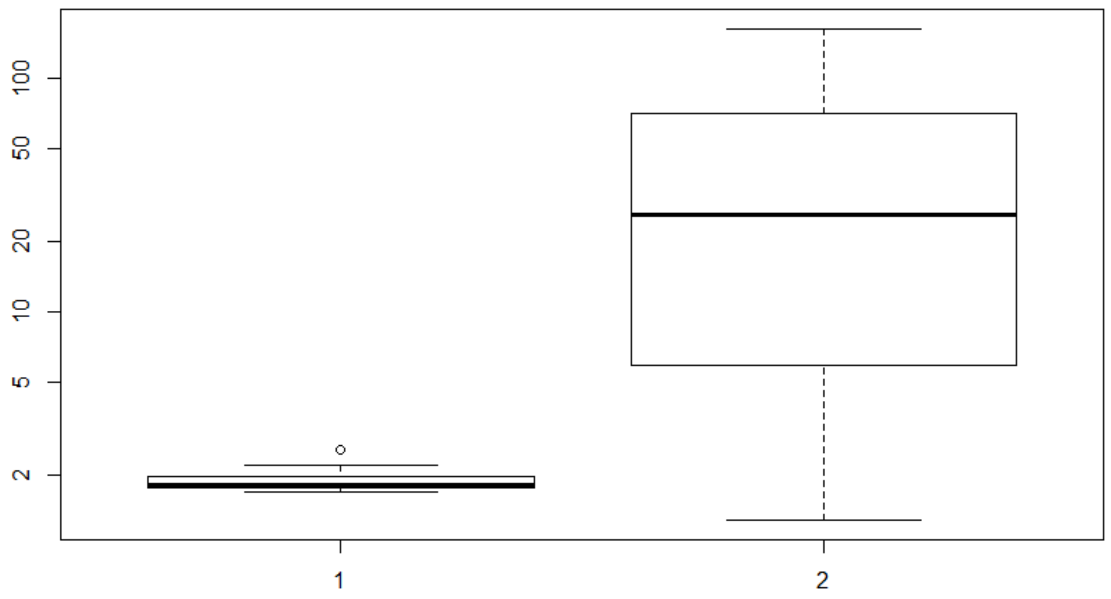

Figure 5.2: Boxplots: The left boxplot is of 20 calculcation times of the ICM algorithm with function $f_1 = x$, 100 datapoints and weights equal to the Hessian diagonal. The right boxplot is with weights equal to one.

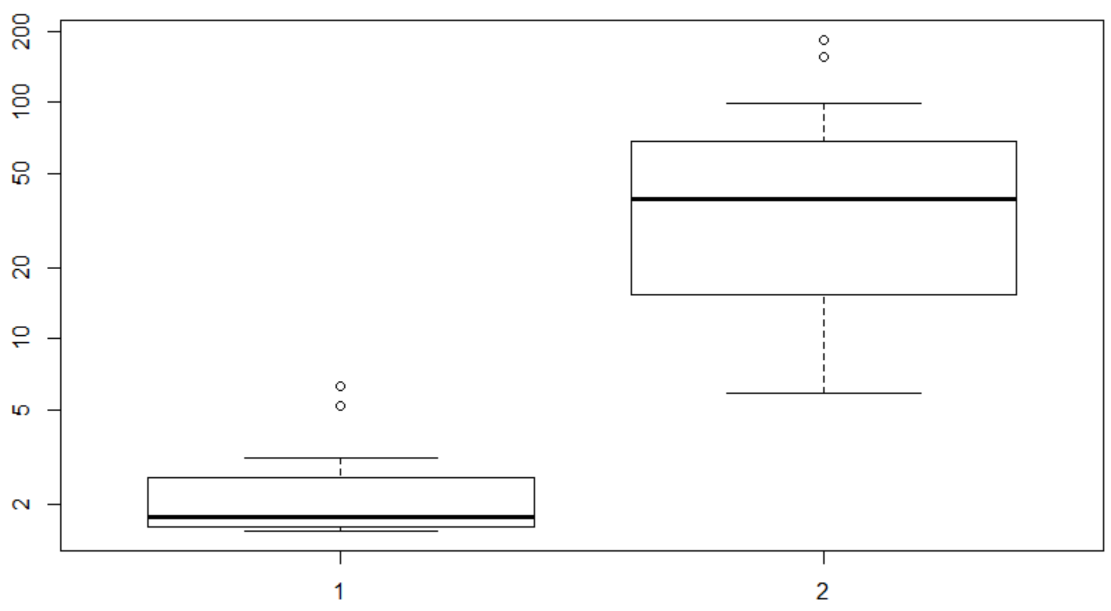

Figure 5.3: Boxplots: The left boxplot is of 20 calculcation times of the ICM algorithm with function $f_2 = x^2$, 100 datapoints and weights equal to the Hessian diagonal. The right boxplot is with weights equal to one.

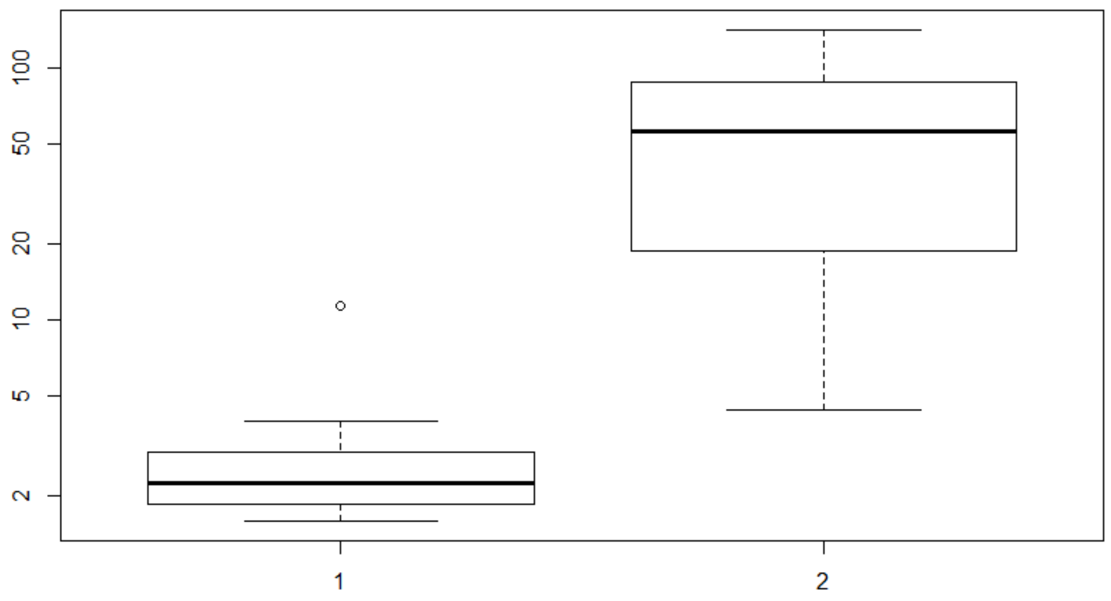

Figure 5.4: Boxplots: The left boxplot is of 20 calculcation times of the ICM algorithm with function $f_3 = exp(-x) + x^2$, 100 datapoints and weights equal to the Hessian diagonal. The right boxplot is with weights equal to one.

5.4 Simulation study 2

In the first simulation study we saw that the weights with the diagonal of the Hessian matrix are a better choice than just taking them equal to one. In this simulation study we want to find the best value for the constant c when multiplying it with the Hessian diagonal weights. For the simulations we take the convex function $f_2(x) = x^2$ on the interval $[0, 1]$. Each simulation result will exist of 100 separate simulations. We will calculate the 0.1, 0.5 and 0.9 quantile of the number iterations for 20 values of c from $c = 0.25$ to $c = 5$. For the number of datapoints we have chosen the values $n = 10$, $n = 50$ and $n = 100$. For the size of the random error we have chosen the values $\sigma = 0.05$, $\sigma = 0.1$ and $\sigma = 0.5$. This leads to 9 simulation results.

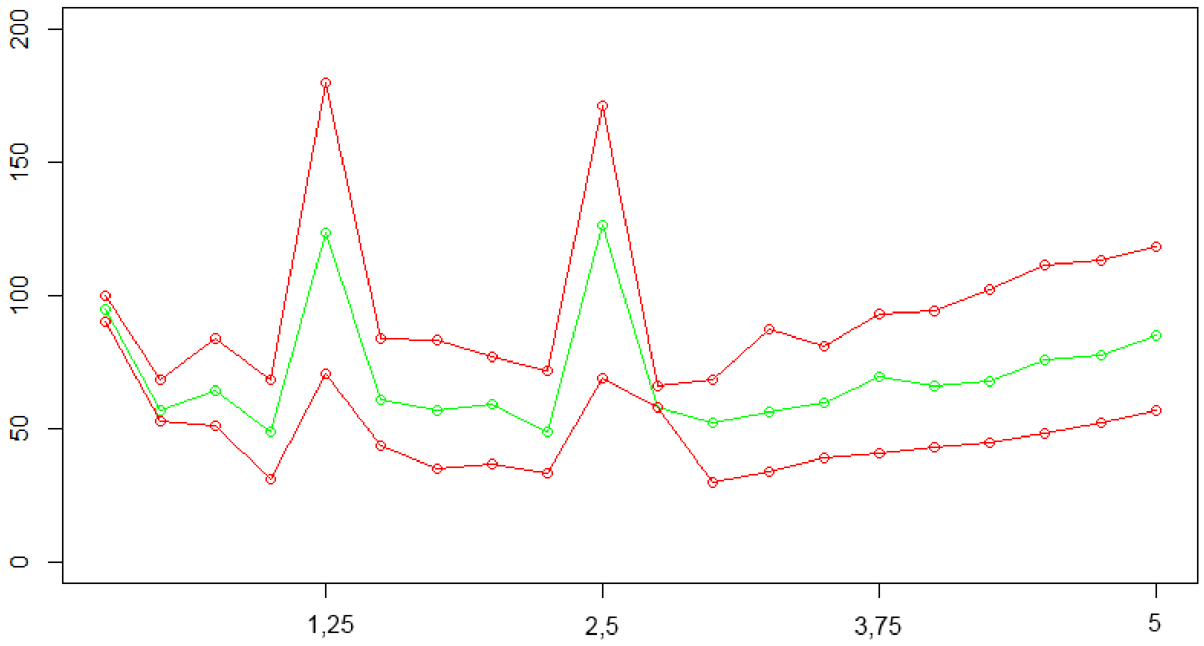

Figure 5.5: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 10$ and $\sigma = 0.05$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

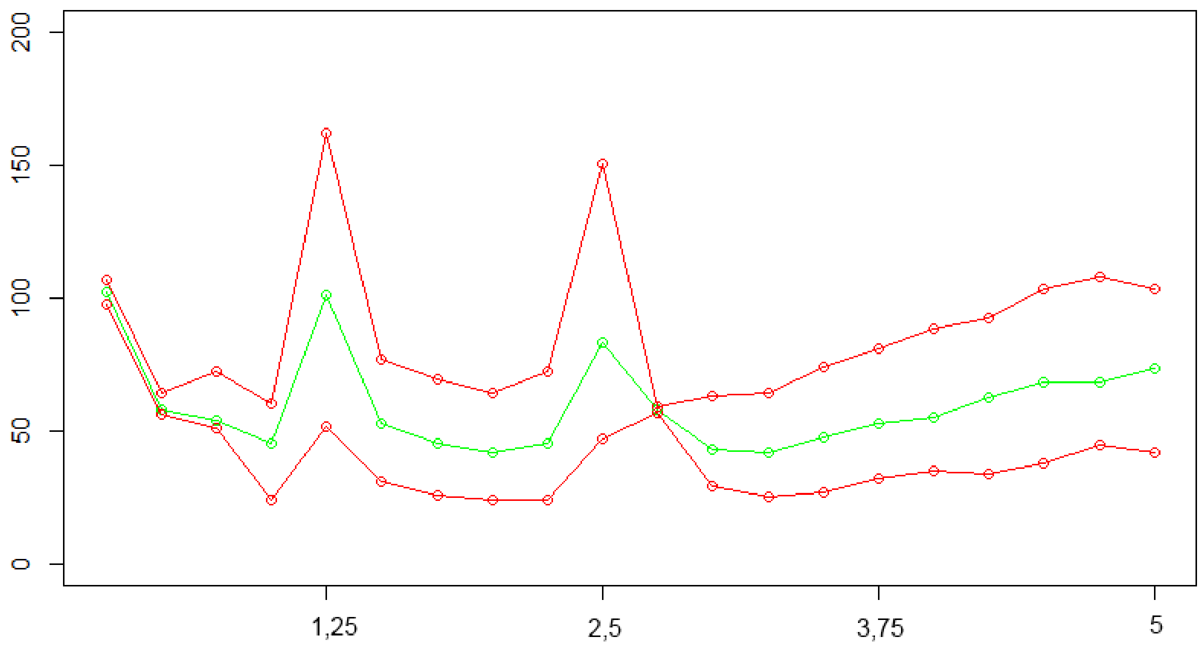

Figure 5.6: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 10$ and $\sigma = 0.1$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

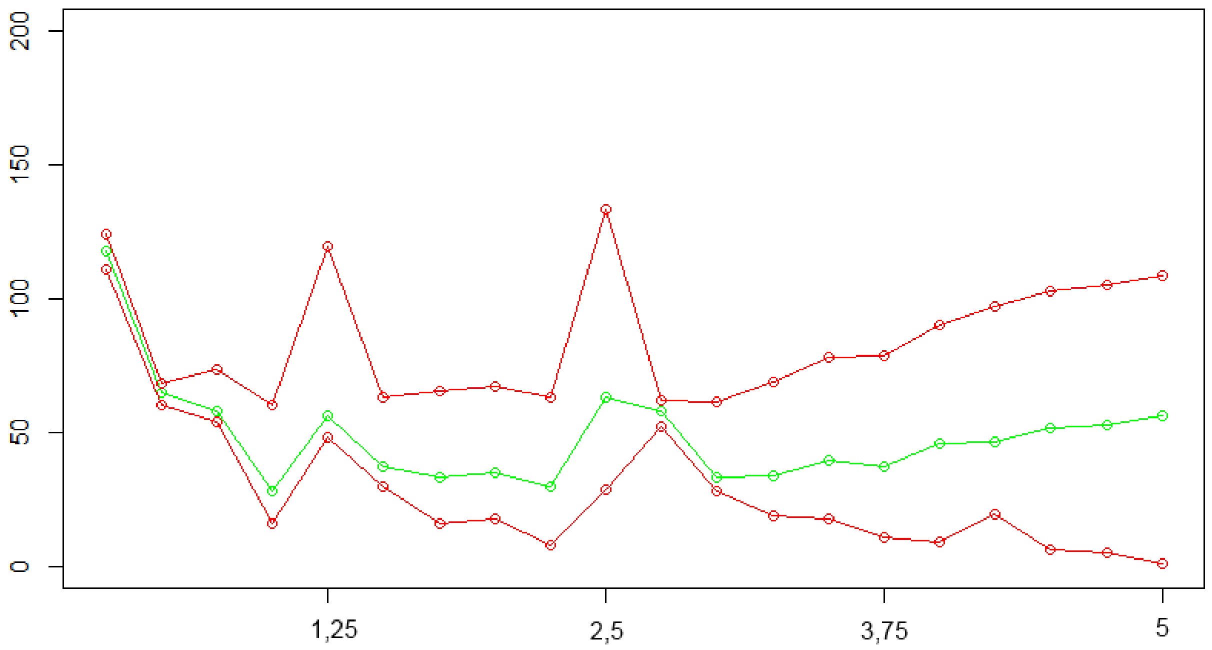

Figure 5.7: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 10$ and $\sigma = 0.5$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

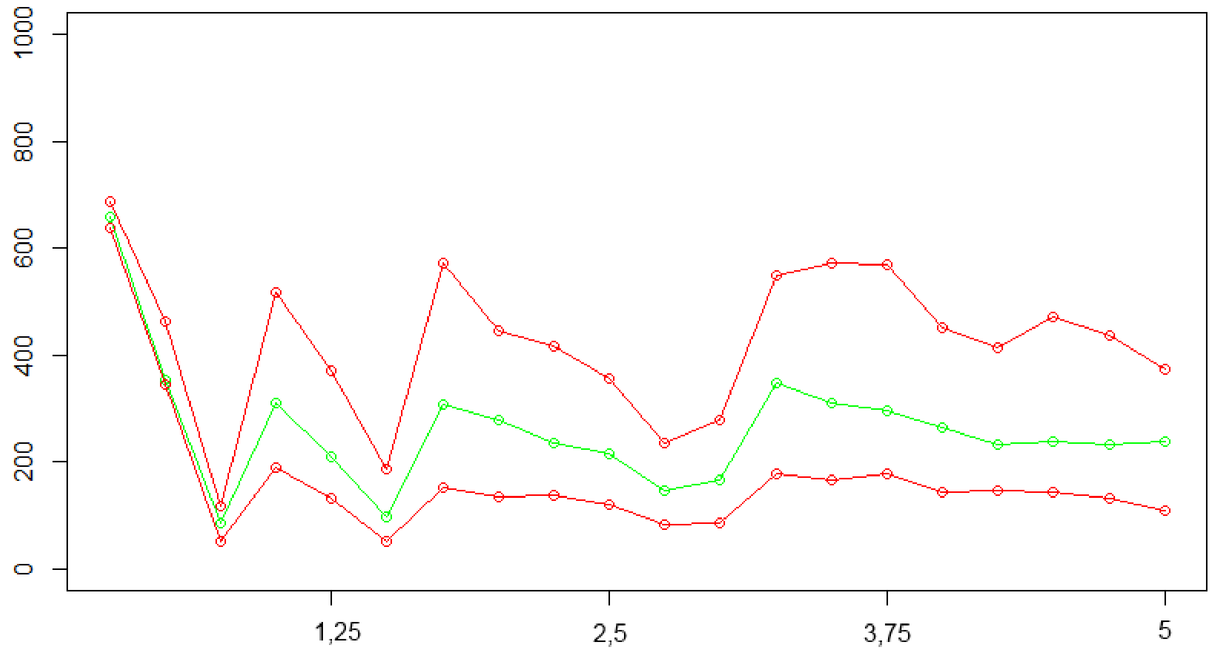

Figure 5.8: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 50$ and $\sigma = 0.05$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

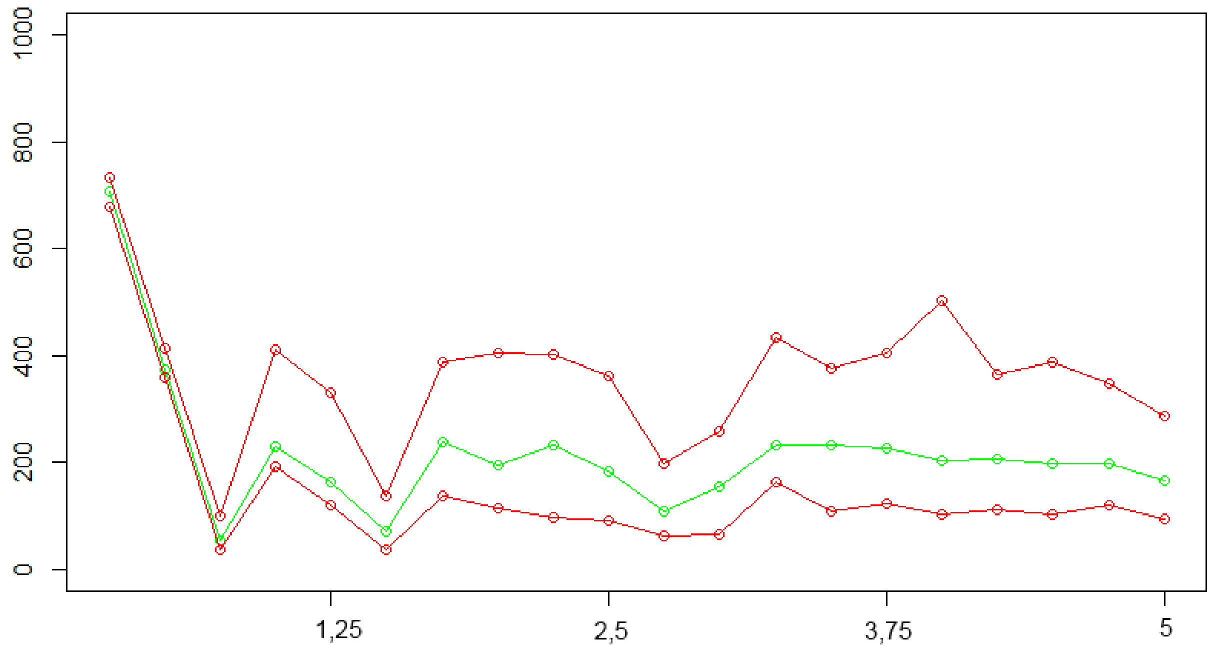

Figure 5.9: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 50$ and $\sigma = 0.1$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

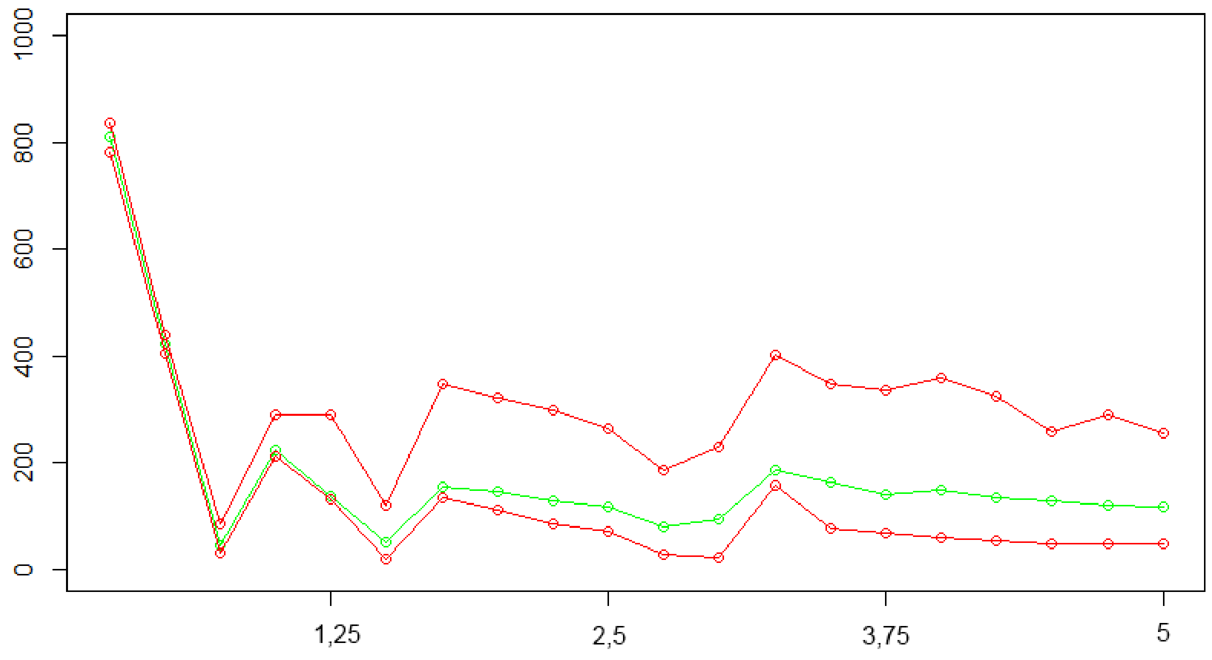

Figure 5.10: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 50$ and $\sigma = 0.5$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

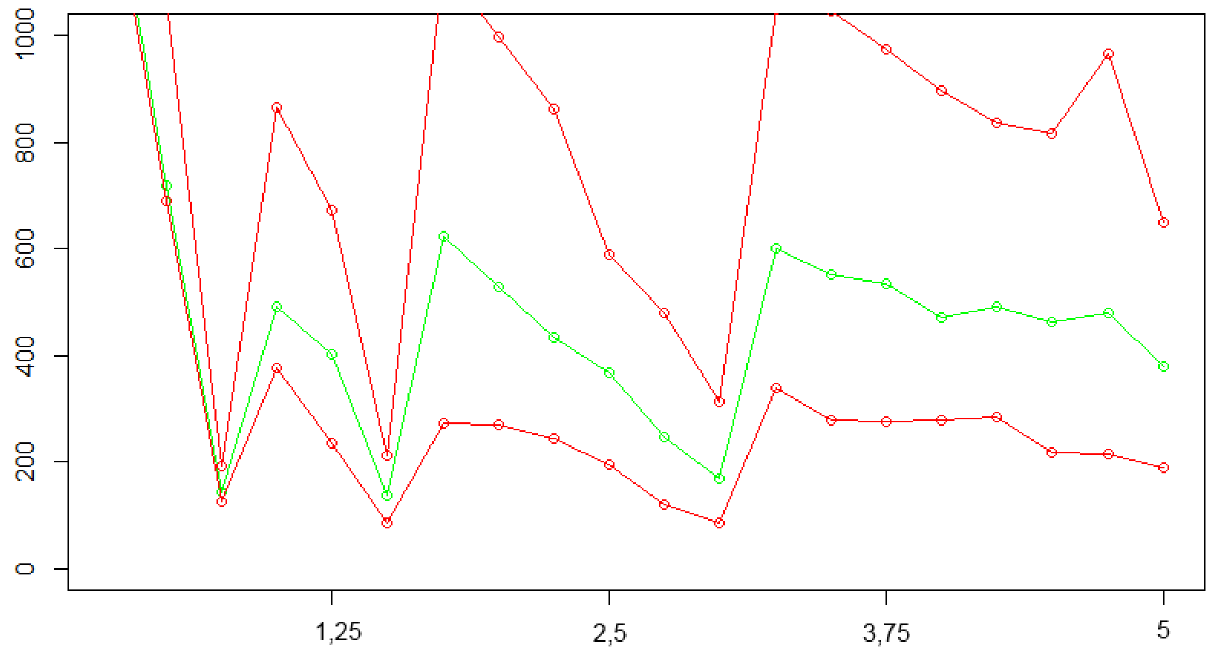

Figure 5.11: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 100$ and $\sigma = 0.05$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

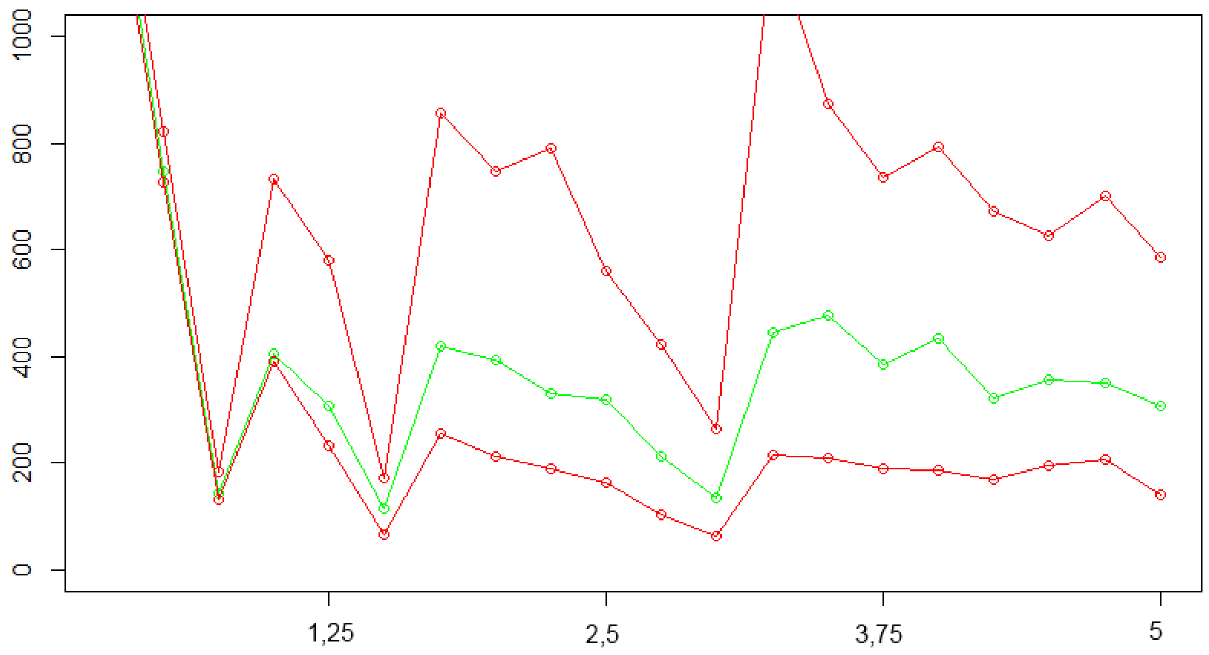

Figure 5.12: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 100$ and $\sigma = 0.1$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

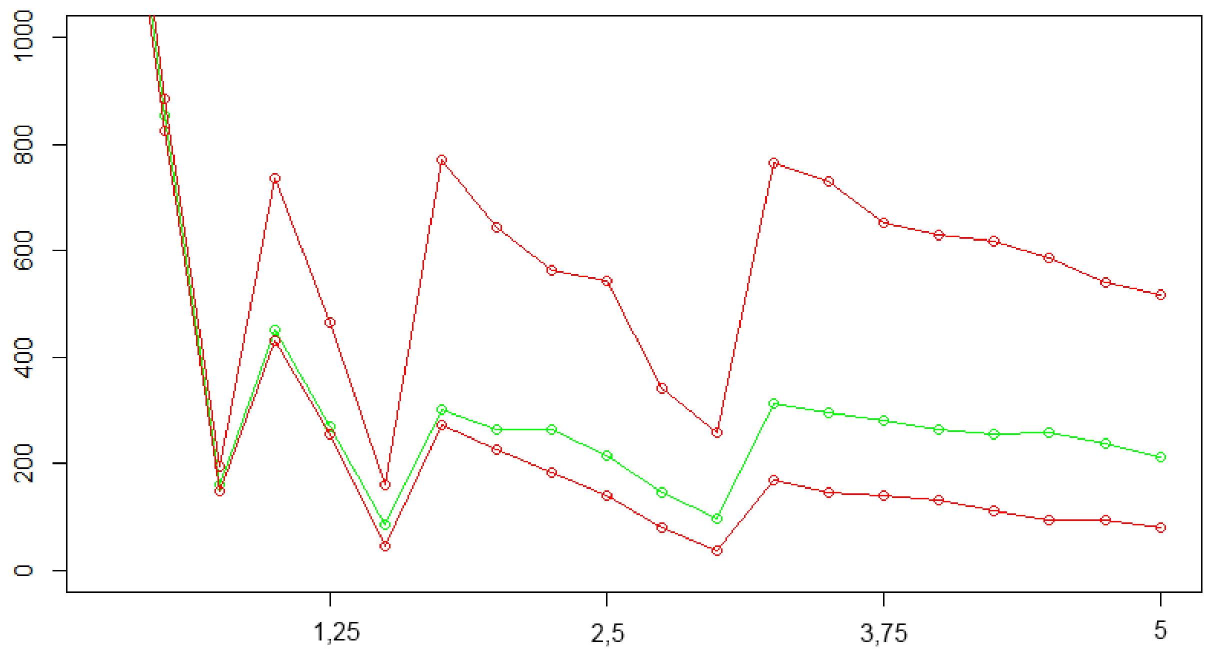

Figure 5.13: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 100$ and $\sigma = 0.5$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

In addition to these 9 simulations with the number of iterations as result we have also run 2 simulations where we also check the total time and the number of linesearches of the ICM algorithm. For this simulations we have used the convex function $f_3(x) = exp(-x) + x^2$ on the interval $[0, 1]$. The other values remain the same and we choose for the 2 simulations as standard error $\sigma = 0.1$ and for the number of datapoints $n = 100$ and $n = 500$.

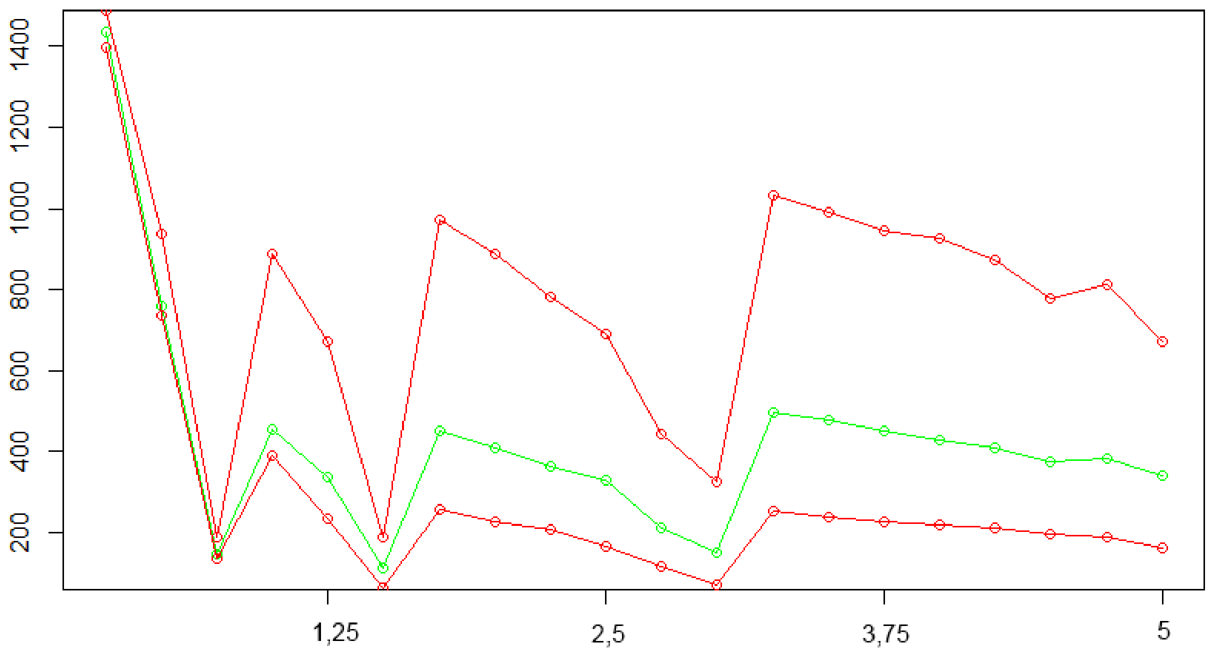

Figure 5.14: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 100$ and $\sigma = 0.1$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

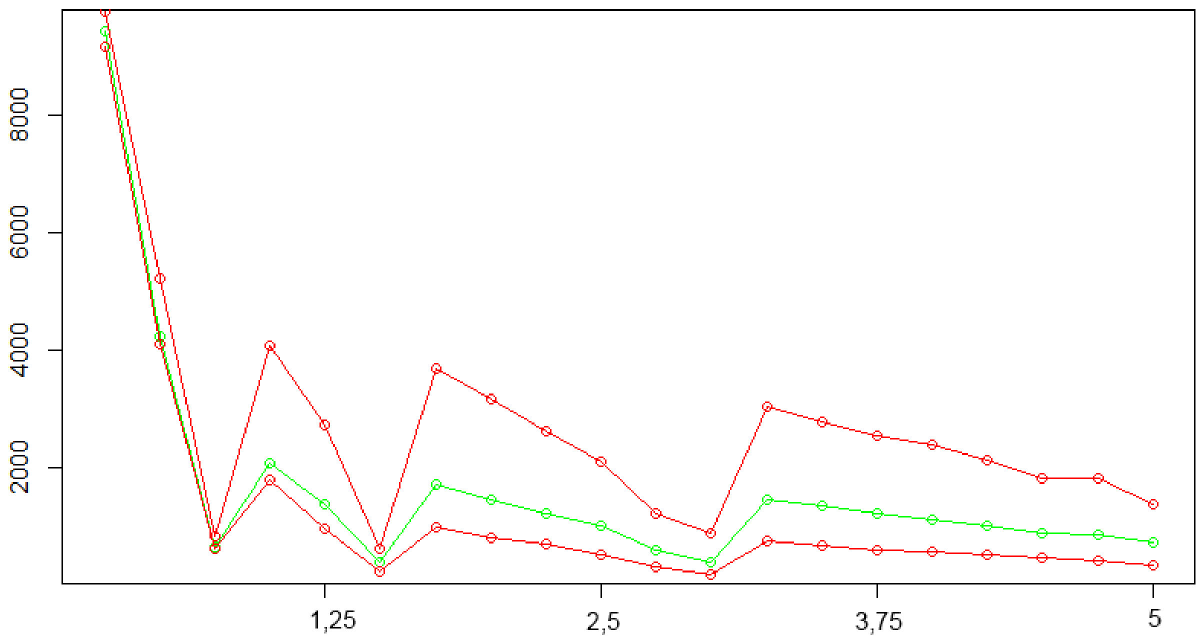

Figure 5.15: A plot of the number of linesearches of 100 simulations of the ICM algorithm with $n = 100$ and $\sigma = 0.1$. On the x-axis are the values for the constant c. The green line is the median of the number of linesearches (mid). The first red line is the 0.9 quantile of the total number of linesearches (above). The second red line is the 0.1 quantile of the total number of linesearches (below).

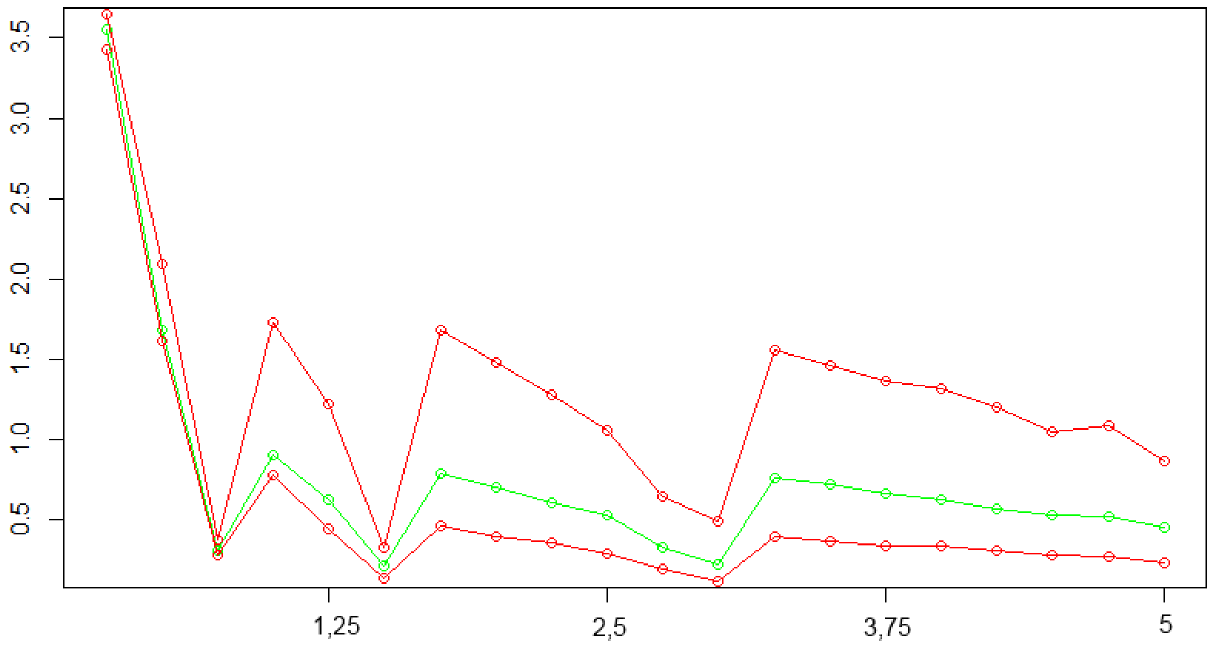

Figure 5.16: A plot of the elapsed time of 100 simulations of the ICM algorithm with $n = 100$ and $\sigma = 0.1$. On the x-axis are the values for the constant c. The green line is the median of the elapsed time (mid). The first red line is the 0.9 quantile of the elapsed time (above). The second red line is the 0.1 quantile of the elapsed time (below).

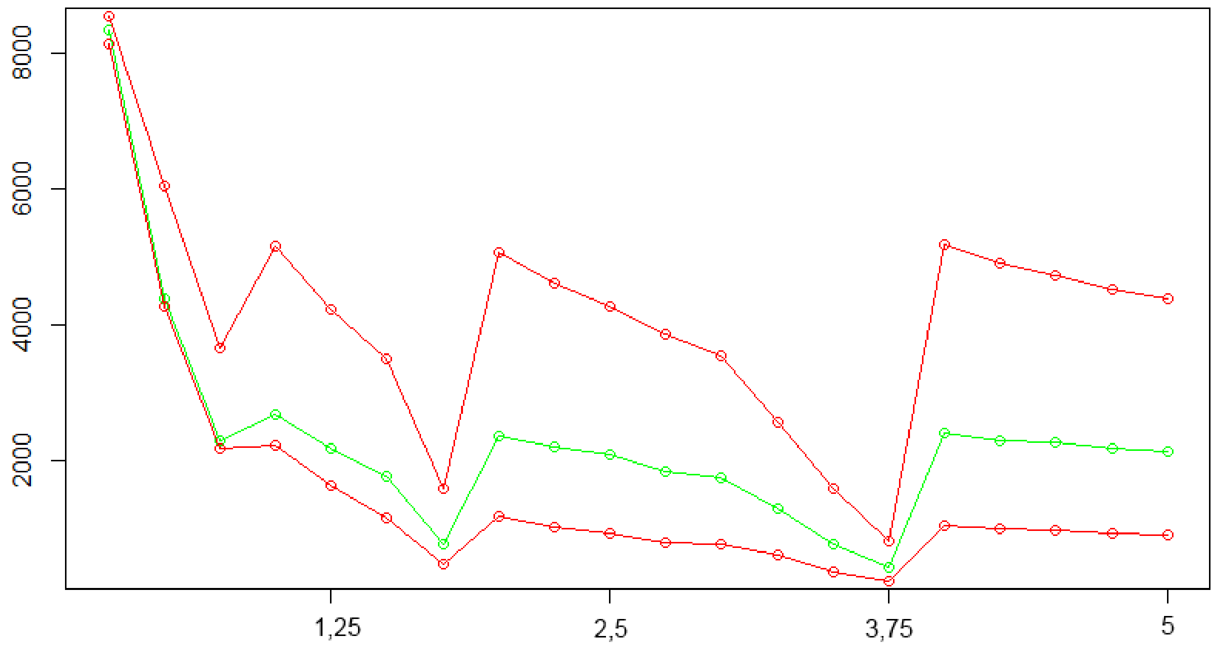

Figure 5.17: A plot of the number of iterations of 100 simulations of the ICM algorithm with $n = 500$ and $\sigma = 0.1$. On the x-axis are the values for the constant c. The green line is the median of the number of iterations (mid). The first red line is the 0.9 quantile of the total number of iterations (above). The second red line is the 0.1 quantile of the total number of iterations (below).

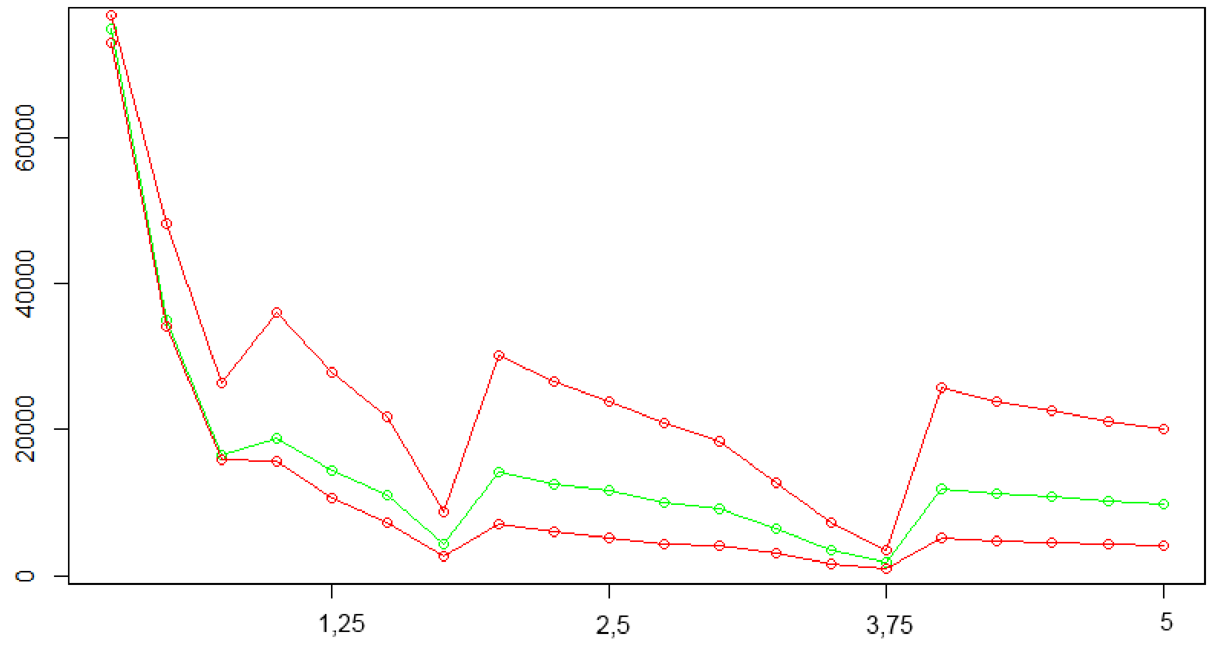

Figure 5.18: A plot of the number of linesearches of 100 simulations of the ICM algorithm with $n = 500$ and $\sigma = 0.1$. On the x-axis are the values for the constant c. The green line is the median of the number of linesearches (mid). The first red line is the 0.9 quantile of the total number of linesearches (above). The second red line is the 0.1 quantile of the total number of linesearches (below).

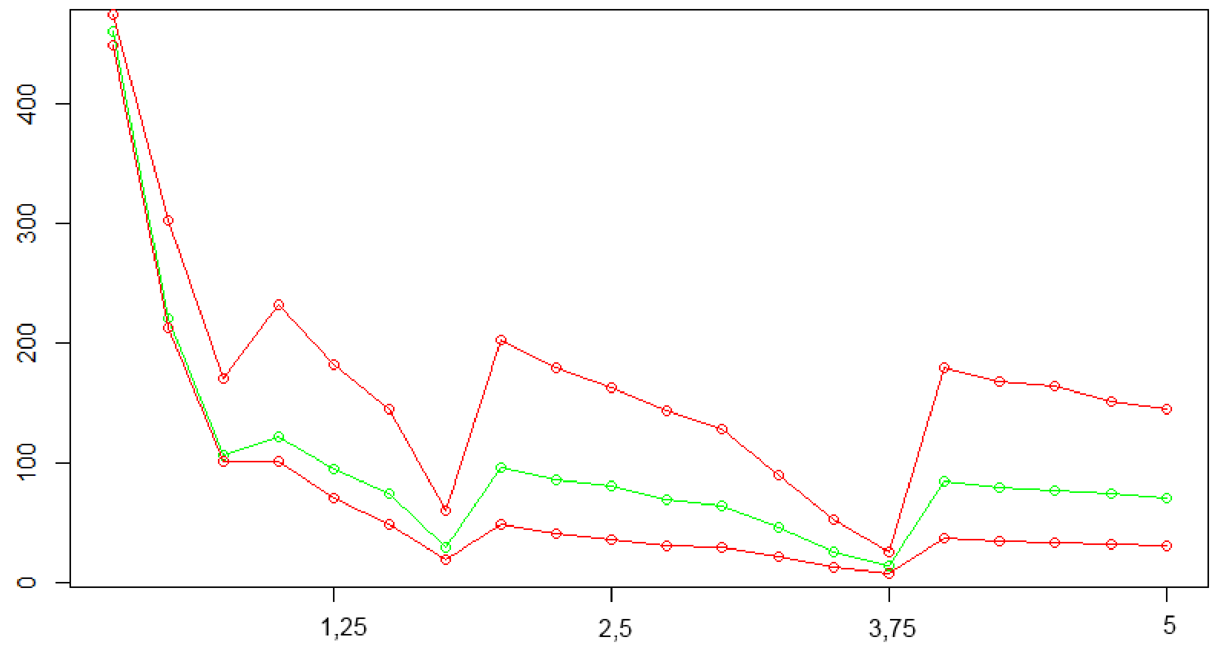

Figure 5.19: A plot of the elapsed time of 100 simulations of the ICM algorithm with $n = 500$ and $\sigma = 0.1$. On the x-axis are the values for the constant c. The green line is the median of the elapsed time (mid). The first red line is the 0.9 quantile of the elapsed time (above). The second red line is the 0.1 quantile of the elapsed time (below).

5.5 Real-world problems

Now we can apply the best choices of weights to some real-world datasets. The first dataset is about the relationship between the age of rabbits and their eye lens weight [9]. The second dataset is about the relationship between the income of Canadian workers and their age [8]. Both relationships can be expected to be concave and we can estimate these relationships nonparametrically using only the concavity restriction with the ICM algorithm.

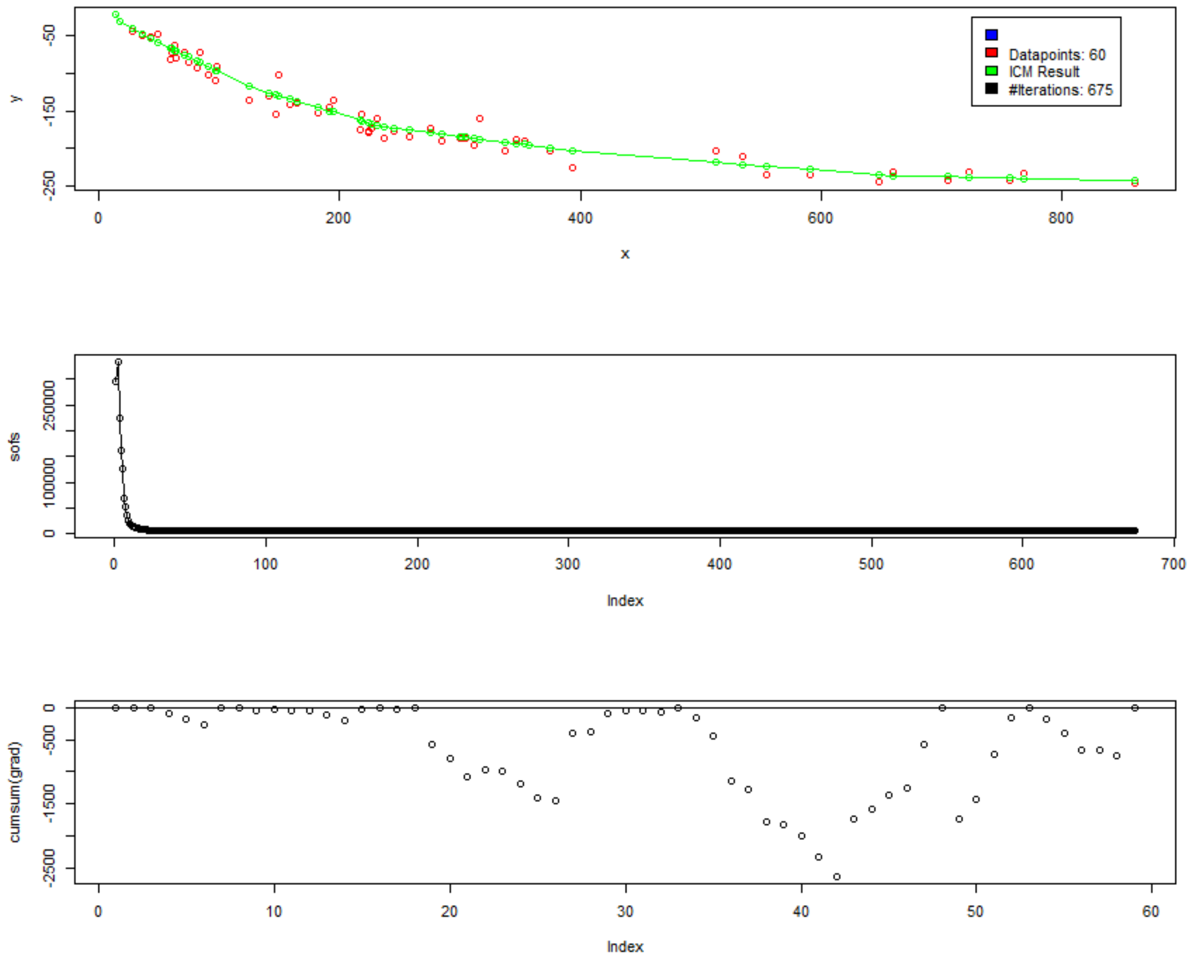

Figure 5.20: A plot of the ICM result of the rabbits data. The green line is the ICM regression result (above). The sum of squares (mid). A plot of cumsum(grad) to check the optimality conditions. (below)

For the rabbits data the ICM algorithm is performed with Hessian diagonal weights. Because the data contained some duplicates we first had to remove these duplicates and changed the value of the weights accordingly. The algorithm stopped after 675 iterations because the solution didn't change anymore. However we have found the optimal solution because we can see in the plot of $cumsum(grad)$ that all the values lie at or below zero, thus the Fenchel optimality conditions have been met. The total calculation time for this computation was 0.57s. The algorithm has performed the line search or damping a total of 2203 times. To compare this with weights equal to one: after 100000 iterations the stop condition was still not met and the $cumsum(grad)$ plot clearly showed some values above zero so the optimal solution was still not found. The algorithm also performed the line search or damping a total of more than 1500000 times.

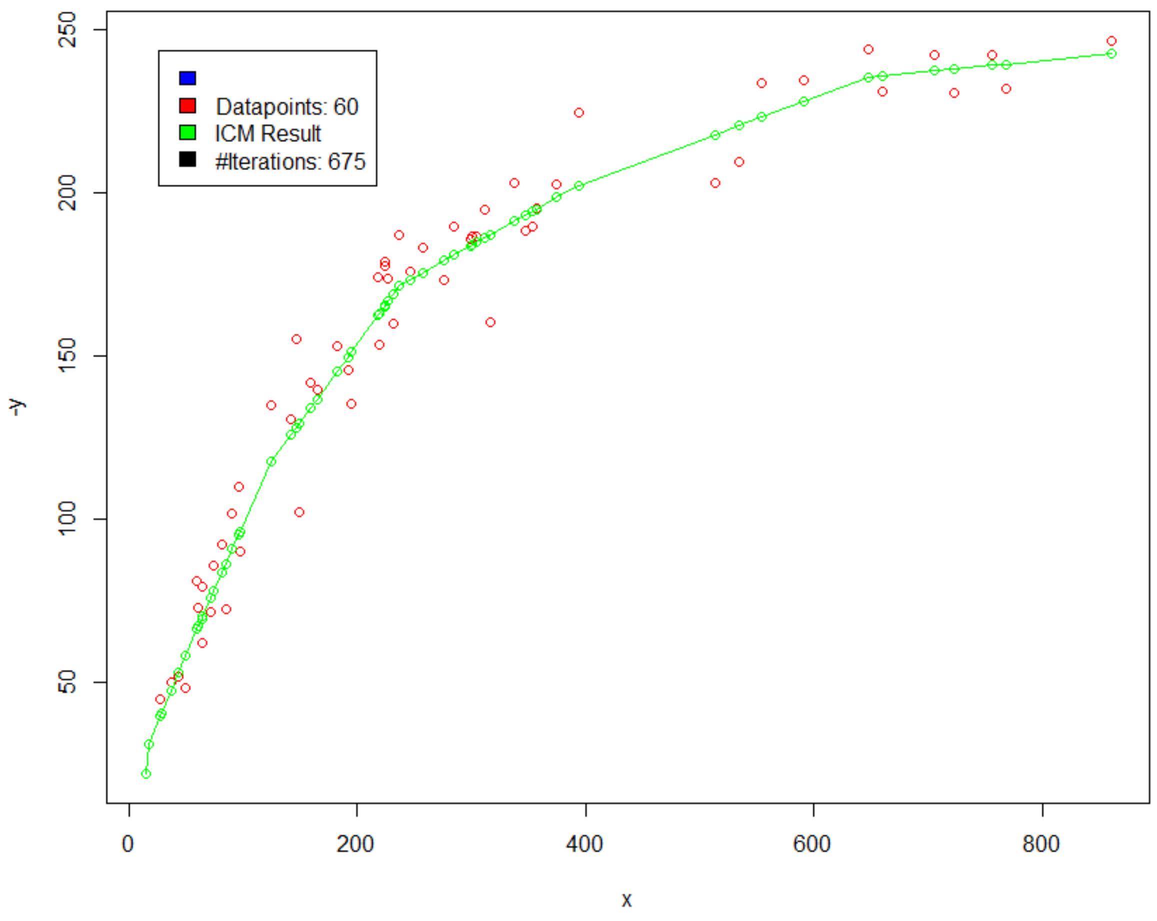

Figure 5.21: A plot of the rabbits data (red) together with the ICM solution (green). The x-axis represents the age of the rabbit in days and the y-axis the dry weight of the eye lens in milligrams.

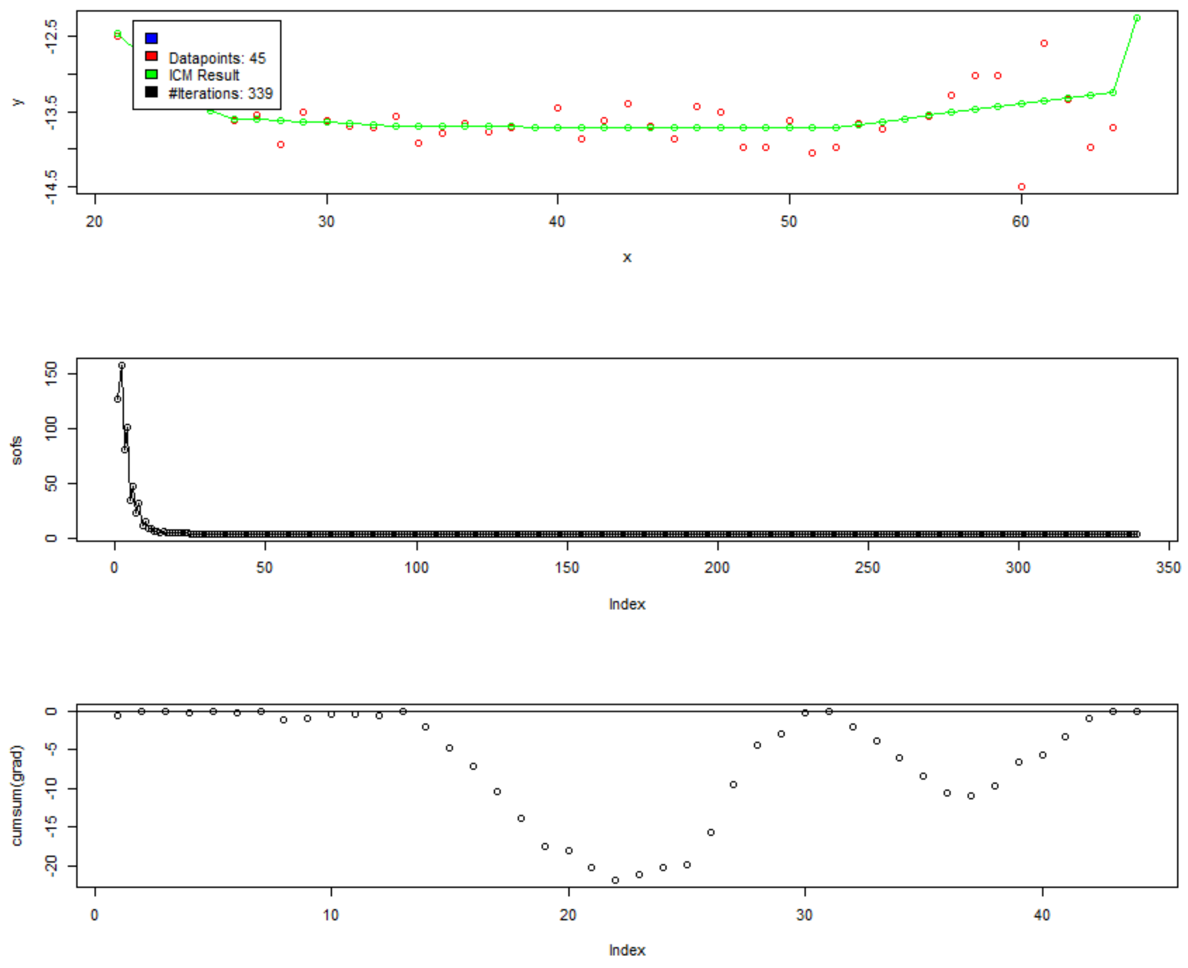

Figure 5.22: A plot of the ICM result of the Canadian income data. The green line is the ICM regression result (above). The sum of squares (mid). A plot of cumsum(grad) to check the optimality conditions. (below)

For the Canadian Income Data the ICM algorithm is again performed with Hessian diagonal weights. Again the duplicates are removed. The algorithm stopped after a total of 339 iterations because the stop conditions were met. The total calculation time was 0.22s. The damping was performed 1108 times. With weights equal to one the calculation took again more time. The stop conditions were met after 13469 iterations and the total calculation time was 17.49s. The damping was performed 129379 times.

5.6 Conclusions

The first simulation study has shown that weights equal to the Hessian diagonal of the optimization problem almost always outperform weights equal to one. In the second simulation study we tried to find out where it can be advantageous to vary the value of the constant c, multiplied with the Hessian diagonal weights. The results were less clarifying then in the first simulation study. In general we can say that the optimal value of c can not be determined in advance but in most cases the value $c = 1$ is the optimal or close to the optimal value of c. Another result we can see from the study is that when c gets larger the variance in the total number of iterations increases. Also, when we look at the number of linesearches, we see that for larger values of the constant c the number of linesearches decreases. We can explain this by our work in Section 3.6. For small values of c the next iterate of the algorithm makes a big step outside the cone C, for large values of c the point that has to be projected lies only a small step outside the cone C. Therefore, in the latter case less linesearches seems more likely.

For this thesis we only studied the choice of weights of the ICM algorithm without considering other improvement options. We think there are some improvement possibilities for the ICM algorithm. For example piecewise chained regression could be tried. Also the conjugate gradient method can be considered. Another option is to change the value of the weights on certain points based on the absolute change of the solution in the last iteration.

A Appendix

File ICM_INITIALIZATION.R

```
#### INITIALIZE FUNCTIONS FOR THE ICM ALGORITHM
ptm <- proc.time()
system.time({
library(fdrtool)
isomean<-function (y, w)
{
    n = length(y)
    if (n == 1) {
        return(y)
    }
    else {
        ghat = .C("C_isomean", as.double(y), as.double(w), as.integer(n),
            ghat = double(n), PACKAGE = "fdrtool", DUP = FALSE)$ghat
        return(ghat)
    }
}

#gcmlcmJos returns the GCM or LCM of the given CSD

gcmlcmJos<-function (x, y, type = c("gcm", "lcm"))
{
    type = match.arg(type)
    if (is.unsorted(x))
        stop("The x values must be arranged in sorted order!")
    if (any(duplicated(x)))
        stop("No duplicated x values allowed!")
    dx = diff(x)
    dy = diff(y)
    rawslope = dy/dx
    rawslope[rawslope == Inf] <- .Machine$double.xmax
    rawslope[rawslope == -Inf] <- -.Machine$double.xmax
    if (type == "gcm")
        slope <- isomean(rawslope, dx)
    if (type == "lcm")
        slope <- -isomean(-rawslope, dx)
    keep = !duplicated(slope)
    x.knots = x[c(keep, TRUE)]
    dx.knots = diff(x.knots)
    slope.knots = slope[keep]
    y.knots = y[1] + c(0, cumsum(dx.knots * slope.knots))
    list(x.knots = x.knots, y.knots = y.knots, slope.knots = slope.knots, slope=slope)
}

#weightmin2 returns the CSD, the GCM and the left derivative of the GCM
#weightmin2()[[1]]$slope
weightmin2 <- function(grad, weight, current){
y=current+1/weight*grad
gg=gcmlcmJos(c(0,cumsum(weight)),c(0,cumsum(weight*y)),type="gcm")
#plot(c(0,cumsum(weight)), c(0,cumsum(weight*y)), type="l", lty=3, main="GCM (red)")
GCM=matrix(0, nrow=length(gg$x.knots), ncol=2)
GCM[,1]=gg$x.knots
GCM[,2]=gg$y.knots
#lines(gg$x.knots, gg$y.knots, col=2, lwd=2)
CSD=matrix(0, nrow=length(current)+1, ncol=2)
CSD[,1]=c(0,cumsum(weight))
CSD[,2]=c(0,cumsum(weight*y))
return(list(gg,CSD,GCM))
}

})
time=proc.time() - ptm

cat("Initalization finished in ",time[3]," seconds \n")

#### END INITIALIZE
```

File ICM_Algorithm.R

```
rm(list=ls(all=TRUE))

#ICM performs the damped ICM algorithm

ICM<-function(x,y,weight)
{
  ptm <- proc.time()
  system.time({
  q=1:(n-1)/n #First estimate
  current=q
  grad=gradf(current)
  qfun=numeric(s)
  #Define function for qf, the weighted least squares estimator
  qf <- function(q){f1h<-f1hat(q)
                    t(K%*%q+f1h*ones-y)%*%W%*%(K%*%q+f1h*ones-y)
  }
  qfun[1]=qf(q)
  sofs=c(1)
  sos <- function(q) sum((y-(f1hat(q)*ones+K%*%q))^2)
  damped=0

  #START DICM ITERATIONS
    for(i in 1:s){
      #Get left derivative of the GCM
      q=weightmin2(-grad, weight, current)[[1]]$slope
      qff<-qf(q)
      while(qff>qfun[i]){ #Damping
        if(all(abs(q-current)<10^-15)){
          cat("Stopped at i=",i,"because q~current \n")
          time=proc.time() - ptm
          return(list(grad,q,i,sofs,damped,time[[1]]))
          stop("Stopped")
        }
        q<-(current+q)/2          #stopt als q=current
        qff<-qf(q)
        damped=damped+1
      }
      sofs[i]=sos(q)
      grad=gradf(q)
      if((abs(sum(grad))<10^-5)&&all(cumsum(grad)<10^-5)){
        cat("Stop conditions met at i=",i,"\n")
        cat("Damped:",damped,"\n")
        time=proc.time() - ptm
        return(list(grad,q,i,sofs,damped,time[[1]]))
        stop("Stopped")
      }
      qfun[i+1]=qff
      current<-q
    }
    if(i==s){
      cat("Stop conditions not met. Stopped at i=",i,"\n")
    }
  time=proc.time() - ptm
  return(list(grad,q,i,sofs,damped,time[[1]]))
  })
  proc.time() - ptm
}

source("ICM_INITIALIZATION.R")

#Give a function to generate random data from
f<-function(x) x
funcf="x"
n=100 #Number of datapoints
datapoints=paste("Datapoints:",n)
x=1:n/n
y=f(x)+rnorm(n,sd=0.1) #Generate random data
s=100000 #Maximum number of iterations for the ICM algorithm
### ICM Initialization
#Construct K matrix
data=diff(x)
K=matrix(0,n,(n-1))
for (i in 1:(n-1)){
  K[(i+1):n,i]=data[i]
}
#Construct W matrix
ones=rep(1,n)
W=matrix(0,n,n)
diag(W)<-ones
#diag(W)<-w #These weights are used when data has duplicates.
#Define function for f1hat
f1hat <- function(q) (t(ones)%*%W%*%(y-K%*%q))/(t(ones)%*%W%*%ones)
A=t(K)%*%(W-1/sum(diag(W))*W%*%ones%*%t(ones)%*%W)%*%K
b=2*t(K)%*%(W-1/sum(diag(W))*W%*%ones%*%t(ones)%*%W)%*%y
gradf<- function(q) 2*A%*%q-b #gradient in q
H=2*A
weight=diag(H)
#weight=rep(1,n-1)

#Perform the ICM algorithm and save it in ICMresult
ICMresult=ICM(x,y,weight)

grad=ICMresult[[1]]
q=ICMresult[[2]]
i=ICMresult[[3]]
sofs=ICMresult[[4]]
damped=ICMresult[[5]]
time=ICMresult[[6]]

par(mfrow=c(3,1))

plot(x,y,type="p",col="red")
xx=1:n/n
points(spline(xx,f(xx),method="n",n=1000),type="l",col="blue")
points(x,f1hat(q)*ones+K%*%q,type="o",col="green")
iterations=paste("#Iterations:",i)
```

```
legend("topleft", inset=.05,c(funcf,datapoints,"ICM Result",iterations), fill=c("blue","r

plot(sofs,type="o")

plot(cumsum(grad))
abline(h=0)
```

File Duplicated_Data.R

```
##### ICM Initialization for duplicate data
ytemp=ydata*!duplicated(xdata)
y=ytemp[ytemp!=0]
## Convert data if not one-to-one.
if (anyDuplicated(xdata) != 0) {
  x=unique(xdata)
  n=length(x)
  w=vector("double",n)+1
  #x=x[!duplicated(x)]
  duplicates=which(duplicated(xdata))
  duplicates=duplicates[which(diff(c(0,duplicates))!=1)]
  l=length(duplicates)
  for (i in 1:l) {
    for (j in (duplicates[i]):n) {
      k=duplicates[i]-1
      y[k]=sum(ydata[k:j])/(j-k+1)
      w[k]=j-k+1
      if (!duplicated(xdata)[j+1]) {
        break
      }
    }
  }
}
```

References

[1] Barlow, R. E., Bartholomew, D. J., Bremner, J. M., and Brunk, H. D. (1972) *Statistical Inference Under Ordered Restrictions*, John Wiley & Sons, New York.

[2] Groeneboom, P. and Jongloed, G. (2014) *Nonparametric estimation under shape constraints*, Cambridge University Press

[3] Jongbloed, G. (1998) *The iterative convex minorant algorithm for nonparametric estimation*, Journal of Computational and Graphical Statistics 7, p. 310-321 [6, 7]

[4] Robertson, T., Wright, F. T., and Dykstra, R. L. (1988) *Order Restricted Statistical Inference*, Wiley & Sons, New York. [7]

[5] Groeneboom, P. and Wellner, J. A. (1992) *Information Bounds and Nonparametric Maximum Likelihood Estimation* [6]

[6] Wei Pan (1999) *Extending the Iterative Convex Minorant Algorithm to the Cox Model for Interval-Censored Data*, Journal of Computational and Graphical Statistics, 8:1, 109-120

[7] Weisstein, Eric W. *Gershgorin Circle Theorem*, From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/GershgorinCircleTheorem.html [11]

[8] Ruppert, D., Wand, M. P., Carroll, R. J. (2003) *Semiparametric regression*, Cambridge University Press [25]

[9] *Age and Eye Lens Weight for Rabbits in Australia*, http://www.statsci.org/data/oz/rabbit.html [25]

[10] V.A. Trenogin (originator). *Newton diagram*, Encyclopedia of Mathematics. http://www.encyclopediaofmath.org/index.php?title=Newton_diagram [3]

[11] *Convex function (of a real variable)*, Encyclopedia of Mathematics. http://www.encyclopediaofmath.org/index.php?title=Convex_function_(of_a_real_variable)

[12] *Schur-convex function*, https://en.wikipedia.org/wiki/Schur-convex_function [2]