University of Tartu
Faculty of Mathematics and Computer Science

Miguel Cuevas Urosa

Optimization of Model Points

Master's Thesis in Financial Mathematics (15 ECTS)

Supervisor: Prof. Kalev Pärna

Tartu
2014

Mudelpunktide optimiseerimine

Magistritöö finantsmatemaatika erialal

Miguel Cuevas Urosa

Lühikokkuvõte

Antud magistritöö eesmärk on uurida, kas mittenegatiivne vähimruutude meetod on rakendatav mudelpunktide leidmiseks elukindlustuses arvutusaja kokkuhoidmise eesmärgil. Me lähtume väga suurest reaalsest andmestikust ja püüame selle jaoks leida mudelpunktid mittenegatiivse vähimruutude meetodi abil. Töös on tutvustatud mittenegatiivse vähimruutude meetodit, selle rakendust elukindlustuspoliiside info kokkusurumisel ja tulemusi.

Võtmesõnad: Mittenegatiivne vähimruutude meetod, mudelpunnktid, elukindlustus

Optimization of Model Points

Master Thesis in Financial Mathematics

Miguel Cuevas Urosa

Abstract

The aim of this work is to study the Nonnegative Least Squares Optimization, to investigate if it is possible to reduce the number of model points in a dataset to save time. We will start with a huge dataset from an insurance company, we are going to optimize this dataset and reduce the number of model point without losing significant accuracy. We do this with the Nonnegative Least Squares (NNLS) method. In this thesis, NNLS will be described briefly, results and conclusions from the NNLS optimization are shown and discussed.

Keywords: Non-negative least squares, model points, life insurance.

Contents

1 Introduction **4**

2 Life insurances **6**
2.1 Solvency II 6
2.2 Model points 6

3 Method **7**
3.1 Method of least squares 7
3.2 Perron-Frobenius theorem 8
3.3 Non-negative method of least squares 15
3.4 History of NNLS 17
3.5 NNLS algorithm 17

4 Data **19**
4.1 Description of the data 19
4.2 Current model point method 20
4.3 NNLS model point method 21

5 Results **24**
5.1 Model points 24
5.2 Residuals 27
5.3 Weights of model points 28

6 Conclusion **31**

A Non-linear least squares **32**

References **33**

1 Introduction

Insurance companies use Asset Liability models to calculate their future cash flows as input for evaluating their solvency position. Insurers have a portfolio of policies with different attributes and different scenarios that can happen. To make a calculation of their expected cash flows, they need to calculate the present value for each scenario or policy combination. These calculations can spend a big amount of time given computing power. So that they needed to create more sophisticated techniques to manage this problem. Multiple options exist to deal with the problem:

- Replicating portfolios
- Software optimization
- Scenario optimization
- Grid technology
- Model Points

We will focus on the model points which are already being used by life insurers and have been based on actuarial knowledge. Calculations of big datasets can consume a lot of time. We are going to look at a case where the objective is to go from a deterministic to a random method for evaluate insurances policies. Consequently, many of the calculations that are being performed in the computer program Algo Financial Modeler, (AFM), will take longer. AFM is a modeling system that is being used by actuaries to calculate risks and value information. In January 2014 a new EU directive came into force. This directive is called Solvency II and force development of a lot of new calculations. Solvency II has replaced the old EU directive that existed to make sure that all the insurance companies have access to replace their policyholders. Since big datasets are being used as input in AFM, it will take a long time to perform these random calculations. If the number of model points as input data is reduced, the datasets size will be short and also the computation in AFM. It is important that the number of model points stay as few as possible with big accuracy. The input set is called policy by policy (P-by-P), and is created by the insurance company, it contains the different expected values of cash flows for different insurance policies. We will proceed from P-by-P for creating model points. We want to check if we can reduce the number of model points from the input dataset. The way that we have chose to reduce the number of model points in a dataset is through the help

of a data algorithm, namely Nonnegative Least Squares, NNLS.

The questions we will answer in this project is:

- Is it possible to reduce the number of model points through nonnegative least squares optimization?
- How many model points do we get through the help of NNLS?
- How does the new model points accuracy look?

The theoretical part of the thesis is based on [1] and [2], while the practical part of the thesis is based in R program [6].

2 Life insurances

Insurers protect the insured for some unwished events that can happen in the future. Of course, this work is not for free. The insured pays a premium which can be monthly, yearly or some other frequency. Whenever a certain event happens, the insurer has a liability to fulfil. To calculate the present value of the expected liabilities which they are exposed in a certain time moment, they use an Asset Liability Model (ALM). ALM models are used to calculate the present value of all future assets and liabilities.
By using these ALM models, companies can show that they are able to pay out all policies under normal conditions. There exists a minimum amount of reserves that insurers must have in order to cover the risks. The rules that determine those amounts are included under the name Solvency.

2.1 Solvency II

Solvency II has replaced old requirements and established more levels for protect the customer. So, it has important implications for runing times and there is a huge interest in methods that can predict the liabilities which they are exposed quickly and accurately as well as; and model points are a way to do this.

2.2 Model points

A model point is an aggregate of policies that is a good representation of a group of policies. An additional criterion, is that we do not want too many of those, otherwise we do not see any reduction in the computing time. A typical model can be grouped in three components: scenarios, policies and attributes. We assume that scenarios are already optimized, so they are assumed to be different. The attributes are characteristics of the policyholder. Policies can be grouped together. This is done, in a way that, if some policies are similar to another one, we can group them in a new fictional policy, which represents the individual policies. This is called the model point.

3 Method

We will proceed from input data and with the help of NNLS we will obtain weights that we can use to reduce the dataset. A weight, is a finite real number greater than or equal to zero which indicates how many times we are going to count every policy. We will obtain a weight for each policy. When we are using NNLS we will get a number of insurance policies which weight is zero. These policies will not have any significance for the dataset and thus we can create a new dataset without the policies that have gotten the weight of zero. In this way, we will obtain a dataset with less model points. NNLS, is an algorithm that can be used in many different data programs, we will choose to use the program R. We will describe how the dataset is built and also how we can reduce the data set with NNLS.

3.1 Method of least squares

The method of least squares is an approximate method that is used to estimate solutions to system of equations. The systems can be an over-determined system which contains more equations than variables or under-determined system which contains more variables than equations. Least Squares Problems are classified into two groups, linear least squares problems and nonlinear least squares problems. Linear least squares problems occur most often in closed form, which means that they can be solved with a finite number of standard operations. Here is an example of curve fitting using least squares optimization.

Example 3.1 Let t be an independent variable and let $b(t)$ an unknown function of t that we want to estimate. Let suppose that we have m observations.

$$b_i = b(t_i), \quad i = 1, \ldots, m.$$

The idea is to build $b(t)$ by a linear combination of base functions, $\phi_j(t)$, which are non-linear functions of t:

$$b(t) \approx A_1\phi_1(t) + \cdots A_n\phi_n(t). \tag{3.1}$$

The design matrix, which we define as x is a rectangular matrix of order $m \times n$. The design matrix is defined as a matrix consisting of values of variables.

$$x_{i,j} = \phi_j(t_i), \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

We can write the equation (3.1) in matrix form:

$$b \approx Ax. \tag{3.2}$$

The unknown variables, A_j, are linear in the model. The system of linear equations (3.2) is an over-determined system. So, it has more equations than the number of variables that we want to estimate. The residuals are the differences between the actual observed value and the value as the model suggests. Mathematically, we can write the residual as

$$r_i = b_i - \sum_j A_j \phi_j(t), \quad i = 1, \ldots, m.$$

So, we want to find A_j that give the least possible sum of least squares residual. In summary, the least squares method minimizes the sum of squared residuals

$$\min ||r^2|| = \min \sum_{i=1}^{m} r_i^2.$$

Nonlinear least squares problems can be formulated as follows

$$\text{minimize} \quad F(X) = \frac{1}{2} \sum_{i=1}^{m} (f_i(x))^2,$$

where $f_1, \ldots, f_m$ are given functions of $\mathbb{R}^m$ to $\mathbb{R}$. Nonlinear least squares problems have no solution in closed form and must be solved numerically.

In order to make use of NNLS, we need to make certain assumptions about the matrix that represents the dataset. The matrix must be non-negative, irreducible and also satisfy the conditions of the Perron-Frobenius Theorem. All matrices discussed are over the real numbers.

3.2 Perron-Frobenius theorem

Firstly, we need some definitions that we will use in the theorem and his proof.

Definition 3.1 *Let A be a matrix. We can say that the matrix A is* **non-negative** *if all elements $a_{ij} \geq 0$ for each i and j. We write it as $A \geq 0$.*

Note that it follows that the transpose matrix A^T is also a non-negative matrix and the linear systems of equations $Ax = b$ have non-negative solutions whenever $b \geq 0$.

Definition 3.2 *A non-negative matrix square T is called* **primitive** *if there is a k such that all the entries of T^k are positive.*

Definition 3.3 *A $n \times n$ matrix A is said to be* **reducible** *if $n \geq 2$ and there exists a permutation matrix P such that*

$$\mathbf{PAP^T} = \begin{pmatrix} \mathbf{B} & 0 \\ \mathbf{C} & \mathbf{D} \end{pmatrix},$$

where $\mathbf{B}$ and $\mathbf{D}$ are square matrices and 0 is the zero matrix. The matrix A is **irreducible** *if it is not reducible.*

Definition 3.4 *We let*

$$Q := \{x \in \mathbb{R}^n : x \geq 0,\ x \neq 0\},$$

so Q is the non-negative orthant which we will call the **positive orthant**. *Also let*

$$C := \{x \geq 0 : ||x|| = 1\},$$

so C is the intersection of the positive orthant with the unit sphere.

Remark 3.1 *If T is irreducible then $I+T$ is primitive, where I is the identity matrix. Indeed, the binomial expansion*

$$(I+T)^k = I + kT + \frac{k(k-1)}{2}T^2 + \cdots$$

will eventually have positive entries in all positions if k large enough.

Lemma 3.1 *Let T be a square matrix and let Λ be a diagonal matrix of the same size, with entries $\lambda_1, \ldots, \lambda_n$ along the diagonal. Expanding $\det(\Lambda - T)$ along the i-th row shows that*

$$\frac{\partial}{\partial \lambda_i} \det(\Lambda - T) = \det(\Lambda_i - T_i),$$

where the subscript i means the matrix obtained by eliminating the i-th row and the i-th column from each matrix.

Setting $\lambda_i = \lambda$ and applying the chain rule, we get

$$\frac{d}{d\lambda} \det(\lambda I - T) = \sum_i \det(\lambda I - T_{(i)}).$$

With the above, we are ready to study the following theorem.

Theorem 3.1 (Perron-Frobenius) *Let T be a $n \times n$ non-negative irreducible matrix.*

1. *T has a positive (real) eigenvalue $\lambda_{\max}$ such that other eigenvalues of T satisfy*
$$|\lambda| \leq \lambda_{\max}.$$
2. *Furthermore $\lambda_{\max}$ has algebraic and geometric multiplicity one, and has an eigenvector x with $x > 0$.*
3. *Any non-negative eigenvector is a multiple of x.*
4. *More generally, if $y \geq 0$, $y \neq 0$ is a vector and μ is a number such that*
$$Ty \leq \mu y,$$
then
$$y > 0 \quad \textit{and} \quad \mu \geq \lambda_{\max}.$$
5. *If $0 \leq S \leq T$, $S \neq T$ then every eigenvalue σ of S satisfies*
$$|\sigma| < \lambda_{\max}.$$
6. *In particular, all the diagonal minors $T_{(i)}$ obtained from T by deleting the i-th row and column have eigenvalues all of which have absolute value $< \lambda_{\max}$.*
7. *If T is primitive, then all other eigenvalues of T satisfy*
$$|\lambda| < \lambda_{\max}.$$

Proof: (See [1])

Let define
$$P := (I + T)^k,$$
where k is chosen so large that P is a positive matrix. Then $v \leq w$, $v \neq w \Rightarrow Pv < Pw$. Recall that Q denotes the positive orthant and that C denotes the intersection of the unit sphere with the positive orthant. For any $z \in Q$ let
$$L(z) := \max\{s : sz \leq Tz\} = \min_{\substack{1 \leq i \leq n \\ z_i \neq 0}} \frac{(Tz)_i}{z_i}. \tag{3.3}$$
By definition $L(rz) = L(z)$ for any $r > 0$, so $L(z)$ depends only on the way though z. If $z \leq y$, $z \neq y$ we have $Pz < Py$. Also $PT = TP$. So, if $sz \leq Tz$ then
$$sPz \leq PTz = TPz,$$

so

$$L(Pz) \geq L(z).$$

Furthermore, if $L(z)z \neq Tz$ then $L(z)Pz < TPz$. So $L(Pz) \geq L(z)$ unless z is an eigenvector of T with eigenvalue $L(z)$.

Thus, we have to look for a positive vector which maximizes L, show that it is the eigenvector we want in the theorem and establish the properties stated in the theorem.

Finding a positive eigenvector.

Consider the image of C under P, $P(C)$. It is compact (being the image of a compact set under a continuous function) and all of the elements of $P(C)$ have all their components strictly positive (since P is positive). Hence the function L is continuous on $P(C)$. Thus L achieves a maximum value, $L_{\max}$, on $P(C)$. Since $L(z) \leq L(Pz)$ this is in fact the maximum value of L on all of Q, and since $L(Pz) > L(z)$ unless z is an eigenvector of T, we conclude that

$L_{\max}$ is achieved at an eigenvector, call it x of T and $x > 0$ with $L_{\max}$ the eigenvalue.

Since $Tx > 0$ and $Tx = L_{\max}x$ we have $L_{\max} > 0$.

Showing that $L_{\max}$ is the maximum eigenvalue.

Let y be any eigenvector with eigenvalue λ, and let $|y|$ denote the vector whose components are $|y_j|$, the absolute values of the components of y. We have $|y| \in Q$ and from

$$Ty = \lambda y,$$

which says that

$$\lambda y_i = \sum_j T_{ij} y_j$$

and the fact that the $T_{ij} \geq 0$ we conclude that

$$|\lambda||y_i| \leq \sum_i T_{ij}|y_j|$$

which we write for short as

$$|\lambda||y| \leq T|y|.$$

Recalling the definition (3.3) of L, this says $|\lambda| \leq L(|y|) \leq L_{\max}$. So we may use the notation

$$\lambda := L_{\max}$$

since we have proved that

$$|\lambda| \leq \lambda_{\max}.$$

We have proved item 1 in the theorem.

Notice that we can not have $\lambda_{\max} = 0$ since then T would have all eigenvalues zero, and hence be nilpotent, contrary to the assumption that T is irreducible.
So

$$\lambda_{\max} > 0.$$

Showing that $0 \leq S \leq T$, $S \neq T \Rightarrow \lambda_{\max}(S) \leq \lambda_{\max}(T)$.

Suppose that $0 \leq S \leq T$. If $z \in Q$ is a vector such that $sz \leq Sz$ then since $Sz \leq Tz$, we get $sz \leq Tz$ so $L_S(z) \leq L_T(z)$ for all z and hence

$$0 \leq S \leq T \Rightarrow L_{\max}(S) \leq L_{\max}(T).$$

So

$$0 \leq S \leq T, S \neq T \Rightarrow \lambda_{\max}(S) \leq \lambda_{\max}(T).$$

Showing that $\lambda_{\max}(T^t) = \lambda_{\max}(T)$.

We can apply the previous results to T^t, the transpose of T, to conclude that it also has a positive maximum eigenvalue. Let us call it η (we shall soon show that $\eta = \lambda_{\max}$). This means that there is a row vector $w > 0$ such that

$$w^t T = \eta w^t.$$

Recall that $x > 0$ denotes the eigenvector with maximum eigenvalue $\lambda_{\max}$ of T. We have

$$w^t T x = \eta w^t x = \lambda_{\max} w^t x$$

implying that $\eta = \lambda_{\max}$ since $w^t x > 0$.

Proving the first two assertions in item 4 of the theorem.

Suppose that $y \in Q$ and $Ty \leq \mu y$. Then

$$\lambda_{\max} w^t y = w^t T y \leq \mu w^t y$$

implying that $\lambda_{\max} \leq \mu$, again using the fact that all the components of w are positive and some component of y is positive so $w^t y > 0$. In particular,

if $Ty = \mu y$ then $\mu = \lambda_{\max}$. Futhermore, if $y \in Q$ and $Ty \leq \mu y$ then $\mu \geq 0$ and

$$0 < Py = (I + T)^{n-1} y \leq (1 + \mu)^{n-1} y$$

so

$$y > 0.$$

This proves the first two assertions in item 4.

If $\mu = \lambda_{\max}$ then $w^t(Ty - \lambda_{\max}) = 0$ but $Ty - \lambda_{\max} y \leq 0$ and therefore $w^t(Ty - \lambda_{\max} y) = 0$ implies that $Ty = \lambda_{\max} y$. Then the last assertion of item 4 (that y is a scalar multilpe of x) will then follow from item 2 (that $\lambda_{\max}$ has a multiplicity one) once we prove item 2, since we have shown that y must be an eigenvector with eigenvalue λ_{max}.

Proof that if $0 \leq S \leq T$, $S \neq T$ then every eigenvalue σ of S satisfies $|\sigma| < \lambda_{\max}$.

Suppose that $0 \leq S \leq T$ and $Sz = \sigma z$, $z \neq 0$. Then

$$T|z| \geq S|z| \geq |\sigma||z|$$

so

$$|\sigma| \leq L_{\max}(T) = \lambda_{\max},$$

as we have already seen. But if $|\sigma| = \lambda_{\max}(T)$ then $L_T(|z|) = L_{\max}(T)$ so $|z| > 0$ and $|z|$ is also an eigenvector of T with the same eigenvalue. But then $(T - S)|z| = 0$ and this is impossible unless $S = T$ since $|z| > 0$. Replacing the i-th row and column of T by zeros give a $S \geq 0$ with $S < T$ since the irreducibility of T precludes all the entries in a row being. This proves the assertion that the eigenvalues of T_i are all less in absolute value that $\lambda_{\max}$.

Showing that $\lambda_{\max}$ has algebraic (and hence geometric) multiplicity one.

Each of the matrices $\lambda_{\max} I - T_{(i)}$ has strictly positive determinant by lemma 3.1. This shows that the derivative of the characteristic polynomial of T is not zero at $\lambda_{\max}$, and therefore the algebraic multiplicity and hence the geometric multiplicity of $\lambda_{\max}$ is one. This proves item 2 and hence all but the assertion of the theorem, which says that if T is primitive, then all the other eigenvalues of T satisfy

$$|\lambda| < \lambda_{\max}.$$

Proof of the last assertion of the theorem.

The eigenvalues of T^k are the k-th powers of the eigenvalues of T. So if we want to show that there are not other eigenvalues of a primitive matrix with absolute value equal to $\lambda_{\max}$, it is enough to prove this for a positive matrix. Dividing the positive matrix by $\lambda_{\max}$, we are reduced to proving the following

Lemma 3.2 *Let $A > 0$ be a positive matrix with $\lambda_{max} = 1$. Then all other eigenvalues of A satisfy $|\lambda| < 1$.*

Proof: Suppose that z is an eigenvector of A with eigenvalue λ with $|\lambda = 1|$. Then $|z| = |Az| \leq |A||z| = A|z| \Rightarrow |z| \leq A|z|$. Let $y := A|z| - |z|$ so $y \geq 0$. Suppose (contrary to fact) that $y \neq 0$. Then $Ay > 0$ and $A|z| > 0$ so there is an $\epsilon > 0$ so that $Ay > \epsilon A|z|$ and hence $A(A|z| - |z|) > \epsilon A|z|$ or

$$B(A|z|) > A|z|, \quad \text{where} B := \frac{1}{1+\epsilon}A.$$

This implies that $B^k A|z| > A|z|$ for all k. But the eigenvalues of B are all < 1 in absolute value, so $B^k \to 0$. Thus all the entries of $A|z|$ are ≤ 0 contradicting the fact that $A|z| > 0$. So $|z|$ is an eigenvector of A with eigenvalue 1. But $|Az| = |z|$ so $|Az| = A|z|$ which can only happen if all the entries of z are of the same sign. So z must be a multiple of our eigenvector x since there are not eigenvectors with all entries of the same sign other than multiples of x, so $\lambda = 1$.

■

This completes the proof of the theorem. ■

Another concept that will be useful is the classical Karush-Kuhn-Tucker conditions. The set of conditions is a generalization of the method of Lagrange multipliers.

The Karush-Kuhn-Tucker (KKT) conditions must be satisfied for a solution to be optimal (which in our case means the solution with the lowest sum of least square residual). This condition is necessary for the solution to be optimal but not enough to show that the solution is optimal.

Theorem 3.2 (Karush-Kuhn-Tucker conditions) *(See [2]).*
Let f be objective function. Let h and g be the restriction functions. We will find the value of x that gives the least value of the function. Let x_{min} be a local minimum of

$$\min_x f(x) \quad subject \quad to \quad \begin{cases} h(x) = 0 \\ g(x) \leq 0 \end{cases}$$

Assume x_{min} lies within these limits, then the Jacobian of the binding constraints at that point is of full rank. Then $\exists$ λ and μ such that

$$\nabla f(x_{min}) + \lambda^T \nabla h(x_{min}) + \mu^T \nabla g(x_{min}) = 0 \tag{3.4}$$

$$\mu^T g(x_{min}) = 0 \tag{3.5}$$

$$h(x_{min}) = 0$$

$$\mu \geq 0. \tag{3.6}$$

3.3 Non-negative method of least squares

A fundamental problem in data modeling is the estimation of a parameterized model for describing the data. We assume that several observations that are linear functions of the parameters have been made. Given a sufficiently large number of such observations, we can reliably estimate the true parameters. Let the unknown model parameters be denoted by the vector $x = (x_1, \ldots, x_n)^T$. The different experiments relating x are represented by the matrix $A \in \mathbb{R}^{m \times n}$, and the set of observed values are given by $b \in \mathbb{R}^m$. The aim is to build a vector $x \in \mathbb{R}^n$ that explains the observed values as well as possible. This requirement can be written as

$$Ax = b,$$

where the system may be either under-determined ($m < n$) or over-determined ($m \geq n$), (see [3]). In the case of over-determined systems, the method of least squares proposes to compute x so that the sum of least squares residual is minimized:

$$f(x) = \frac{1}{2}||Ax - b||^2. \tag{3.7}$$

However, in many real world problems the parameters represent quantities that can take only non-negative values. In such case, problem (3.7) must be modified to include non-negativity constraints on the model parameters x. The resulting problem is called Nonnegative Least Squares (NNLS), and is formulated as follows:

NNLS Problem:

Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and the set of observed values given by $\mathbf{b} \in \mathbb{R}^m$, find a nonnegative vector $x \in \mathbb{R}^n$ to minimize the function $f(x) = \frac{1}{2}||\mathbf{A}x - \mathbf{b}||^2$, i.e.

$$\min_x f(x) = \frac{1}{2}||\mathbf{A}x - \mathbf{b}||^2,$$
$$\text{subject to} \quad x \geq 0. \tag{3.8}$$

The gradient of $f(x)$ is $\nabla f(x) = \mathbf{A}^T(\mathbf{A}x - \mathbf{b})$ and the KKT optimality conditions for NNLS problem (3.8) are

$$\begin{aligned} x &\geq 0, \\ \nabla f(x) &\geq 0, \\ \nabla f(x)^T x &= 0. \end{aligned}$$

Some of the iterative methods for solving (3.8) are based on the solution of the corresponding linear complementarity problem (LCP).

Let's see that the constraints above are KKT conditions:
We have from theorem 3.2 that

$$\begin{aligned} f(x) &= \frac{1}{2}||Ax - b||^2, \\ g(x) &= -x, \\ h(x) &= 0 \end{aligned}$$

so

$$\begin{aligned} \nabla f(x) &= A^T(Ax - b), \\ \nabla g(x) &= -1 \end{aligned}$$

The first constraint is obvious from definition of the problem (3.8). The second and third constraints, we can see that they are equivalence if we substitute in (3.4), (3.5) and (3.6) and we get

$$\begin{aligned} \nabla f(x) - \mu^T &= 0, \\ -\mu^T x &= 0, \\ \mu &\geq 0. \end{aligned}$$

so we can see that

$$\begin{aligned} \nabla f(x) = \mu^T \geq 0 \quad &\Rightarrow \quad \nabla f(x) \geq 0, \\ -\nabla f(x)^T x = 0 \quad &\Rightarrow \quad \nabla f(x)^T x = 0. \end{aligned}$$

Linear Complementarity Problem:

Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and the set of observed values be given by $\mathbf{b} \in \mathbb{R}^m$, find a vector $x \in \mathbb{R}^n$ to minimize the function

$$\begin{aligned} \lambda = \nabla f(x) = \mathbf{A}^T\mathbf{A}x - \mathbf{A}^T\mathbf{b} \geq 0, \\ x \geq 0, \\ \lambda^T x = 0. \end{aligned} \tag{3.9}$$

Problem (3.9) is essentially the set of KKT optimality conditions for quadratic programming. The problem reduces to find a nonnegative x which satisfies $(\mathbf{A}x - \mathbf{b})^T \mathbf{A}x = 0$.

Dealing with nonnegative constraints when we are working with large amounts of non-linear equations is difficult. Sometimes is more manageable a different formulation of NNLS using the residual vector variable $\mathbf{p} = \mathbf{b} - \mathbf{A}x$ as follows:

$$\min_{x,p} \frac{1}{2}\mathbf{p}^T\mathbf{p},$$
$$\text{subject to} \quad \mathbf{A}x + \mathbf{p} = \mathbf{b}, \quad x \geq 0.$$

The advantage of this formulation is that we have a simple and separable objective function with linear and nonnegativity constraints.

3.4 History of NNLS

Since the 1990's, NNLS calculations have been generalizated to approximate non-negative matrices and tensor factorizations, in order to obtain low-dimensional representations of nonnegative data. A tensor is a multidimensional vector or a generalization of vectors, matrices and scalars. Albert Einstein was one of the first to use tensors and he used them to describe the laws of physics. After this, several tensor analysis techniques have become. Tensor analysis have multiple uses but is especially suitable for large data sets. NNLS is used on large data sets non negative. Using low rank constraints on high dimensional data set, this method can reduce the number of elements in the data set . There are several different methods for reducing the dimensions of the matrices and data sets. Some examples are:

- Principal Component Analysis (PCA)
- Independent Component Analysis (ICA)
- Approximate Nonnegative Matrix Factorization (NMF)
- Nonnegative Tensor Factorization (NTF)

3.5 NNLS algorithm

NNLS algorithm computes the solution to a vector x. This vector solves the least squares problem, where $||\mathbf{A}x - \mathbf{b}||$ is minimized with constraints that $\mathbf{G}x \geq \mathbf{h}$. $\mathbf{A}$ is a $m \times n$ matrix and $\mathbf{b}$ is a vector with m elements. What

distinguishes NNLS from other least squares optimization is that **A** in other least squares optimizations tend to be an arbitrary $m \times n$ matrix, but in NNLS, **G** is the identity matrix and **h** is the zero vector. This problem has always at least one solution, and this solution usually will contain at least $n - m$ of elements equal to zero.

4 Data

In this section, we are going to show a numerical example with real data. We will get the model points for two different data sets and we will compare graphically the difference if we use the full data or the model points, for checking the accuracy of the result.

4.1 Description of the data

We start from a data set for creating the model points P-by-P (Policy by Policy). We have two differents data sets built P-by-P with insurance policies. These data sets contain the same variables and we will treat them independently.

We can consider the data sets as matrices, where each column of the matrix corresponds to policies and each row corresponds to a variable. In the first data set we have 7045 policies and in the second one, we have 13924 policies.

Each policy is equivalent to a model point, so when calculations are performed with data sets, is important to have the fewest model points as possible of our data set because calculations will perform faster.

The number of variables is the same for both data sets. In total, we have 4 different variables. We know the value of the variables for each year until the next 50 years, such that, we have 50 values for the first variable, 50 values for the second one and so on. However, there are some variables which include also year 0 (present value).

In our data set, for the first variable Best Estimate Reserve, we have 51 values and for the rest of variables Cash Flow Benefit Maturity, Cash Flow Benefit Surrender and Cash Flow Premium 50 values, so the total number of rows is $m = 201$.

Each year of each variable corresponds to a row in the matrix $\mathbf{A}$. For using the algorithm "nnls", we need an agregate data set $\mathbf{b}$. This data set consists of the sum of the insurance policies per year, i.e. the agregate data set $\mathbf{b}$ is equal to the sum of the columns of the matrix $\mathbf{A}$.

The following table show us the variables which represent our data set and how many years we have values for the respective variables.

	Variable	Number of years
1	Best Estimate Reserve	51
2	Cash Flow Benefit Maturity	50
3	Cash Flow Benefit Surrender	50
4	Cash Flow Premium	50

So we have $m = 201$ rows and $n = 7045$ columns in the matrix A. Let V_j be the variable j for each $j = 1, \ldots, m$ and F_i the policy i for each $i = 1, \ldots, n$, so we have that the elements $a_{ij} \in A$ are $a_{ij} = V_j F_i$, hence we can write

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \ldots & a_{m,n} \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} \sum_i^n a_{1,i} \\ \sum_i^n a_{2,i} \\ \vdots \\ \sum_i^n a_{m,i} \end{pmatrix}$$

So we will treat the data set as a huge matrix with dimension 201×7045. From this data set, we can create the agregate data set for each variable denoted by $\mathbf{b}$, which is a matrix with dimension $m \times 1$ including all the agregate values for each variable. We take this aggregate vector $\mathbf{b}$, since we want that the weights of the model points produce the same aggregated values of the variables as original policies. After that, we will estimate the vector $\mathbf{x}$ with the algorithm NNLS and we will get a vector with dimension $n \times 1$ for this data set.

4.2 Current model point method

Before starting to compute the NNLS algorithm, we are going to show an example of the current model points method to understand why it is important to introduce the new method, Non-Negative Least Square.

The previous method for creating model points, (the current method), is carried out in two steps. The first step is to create model points P-by-P in different policy groups and the second one is to reduce the number of model points.

Example of current method:

We have a portfolio with 100 policyholders and we are interesting in dividing people by two categories, gender and premium. The gender can take 2 values, women or men, and the premiums can take 3 values, A, B or C. So, for example, we can have 20 men belonging to type A, 10 women who belong to type A, 25 men belonging to type B, 15 women who belong to type B, 15 men belonging to type C and 15 women who belong to type C.
Then we get the following table:

	Men	Women
A	20	10
B	25	15
C	15	15

This example contains six model points with weights 20, 10, 25, 15, 15 and 15. In the second step, we have to reduce the number of model points. To do this, we have to analyze each category separately.

For instance, we can reduce the number of variables in the range of premiums from (A, B, C) to (A, B), if B and C have the same properties.
Then we get the following table

	Men	Women
A	20	10
B	40	30

Hence, we have gotten to reduce the number of model points from six to four and the new model points have weights 20, 10, 40 and 30.

In the case of a set with real data, there are too many categories and we get a result with a lot of model points. Another disadvantange of this method is that when we reduce a varible, the model has to be checked with the AFM program to evaluate his efficiency and also we need a large knowledge of the variables to group them as efficient as possible, so this process is really slow.

Therefore, this method become difficult, ineffective and takes too much time.

4.3 NNLS model point method

Now, we are going to explain how to use the algorithm NNLS for getting the model points.

We have used the program R. Before starting to read the data, we need to install and load some additional packages called "XLConnect", "MASS" and "nnls". Then, we can see the code in R where we will explain the algorithm.

Code in R: (See [6])

```
#First, we have to import the data set from excel to R.
excel.file=file.path("data_all.xls")
x=readWorksheetFromFile(excel.file,sheet=1)
#where "data.file.xls" is the name of the data set.

#We create an empty matrix where we can write the values
```

```
#of the data set and the last column will fill with
#the policy number.
M=matrix(rep(0),nrow=dim(x)[1],ncol=dim(x)[2])
for (i in 1:dim(x)[2]){
        M[,i]=x[,i]
}

#We create an empty matrix A^T and a vector policy and
#we will fill them with the data and the policy number.
A_T=matrix(rep(0),nrow=dim(x)[1],ncol=dim(x)[2]-1)
policy=matrix(rep(0),nrow=dim(x)[1],ncol=1)

for (i in 1:dim(M)[1]){
        policy[i,1]=M[i,dim(M)[2]]
}
for (i in 1:dim(M)[2]-1){
        A_T[,i]=M[,i]
}

#Finally, we get the matrix A.
A=t(A_T)

#Now, we create the agregate vector b,
#which contains the sum of the rows of the matrix A.
b=matrix(rep(0),nrow=dim(A)[1],ncol=1)
for (i in 1:dim(A)[1]){
        b[i,1]=sum(A_T[,i])
}

#Then, we can compute the algorithm NNLS.
result=nnls(A,b)

#After that, we write a vector with the weights
#of the model points.
weights=matrix(rep(0),nrow=result$nsetp,ncol=2)
model_points=matrix(rep(0),nrow=result$nsetp,ncol=1)
a=0
for (i in 1:length(result$x)){
        if(result$x[i]!=0){
                a=a+1
                weights[a,1]=result$x[i]
```

```
                weights[a,2]=policy[i]
                model_points[a]=i
        }
}

#Finally, we can export the weights
#with the respective policy number from R to Excel.
base=data.frame(weights)
colnames(base)=c("weights","policy_number")
wb=loadWorkbook("Result.xls", create=TRUE)
createSheet(wb,name="outputs")
createName(wb,name="outputs",formula="outputs!$A$1")
writeNamedRegion(wb, base ,name="outputs")
saveWorkbook(wb)
```

5 Results

In this section, we will show the results that we obtained in the previous one.

5.1 Model points

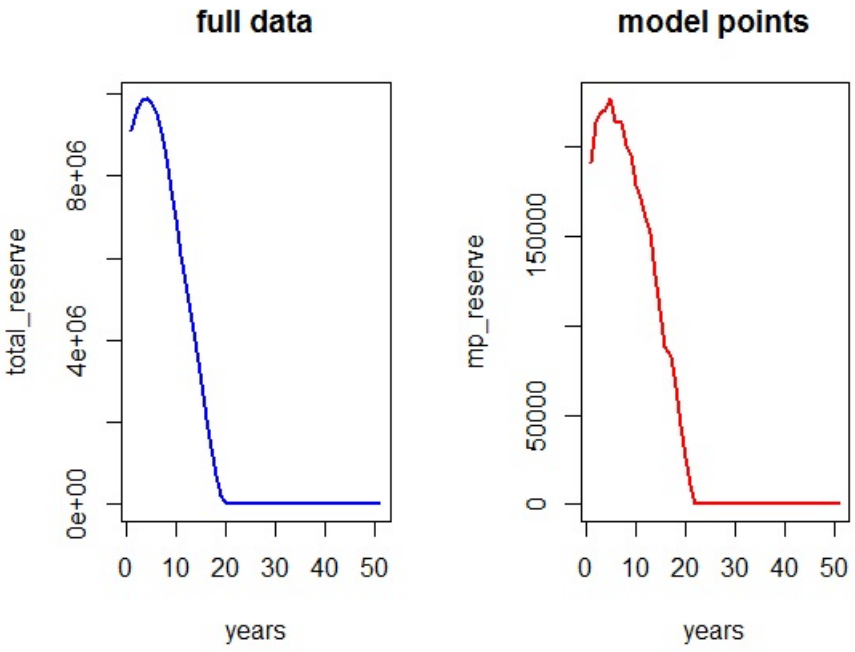
Here we show the code in R for plotting the graphs.

```
#Graphs (Full data vs model points)

#We write the vector with the full data for each variable
total_reserve=matrix(rep(0),nrow=51,ncol=1)
for (i in 1:51){
        total_reserve[i,1]=sum(A[i,])
}

#We fill the vector below with the values
#of the model points for each variable
mp_reserve=matrix(rep(0),nrow=51,ncol=1)
for (i in 1:51){
        for (j in 1:dim(model_points)[1]){
                mp_reserve[i,1]=mp_reserve[i,1]
                +A[i,model_points[j]]
        }
}

#We plot the corresponding graphs
years=seq(1,51)
par(mfrow = c(1,2))

plot(years,total_reserve,type="l",col="blue",lwd=2,
main="full data")
plot(years,mp_reserve,type="l",col="red",lwd=2,
main="model points")
```

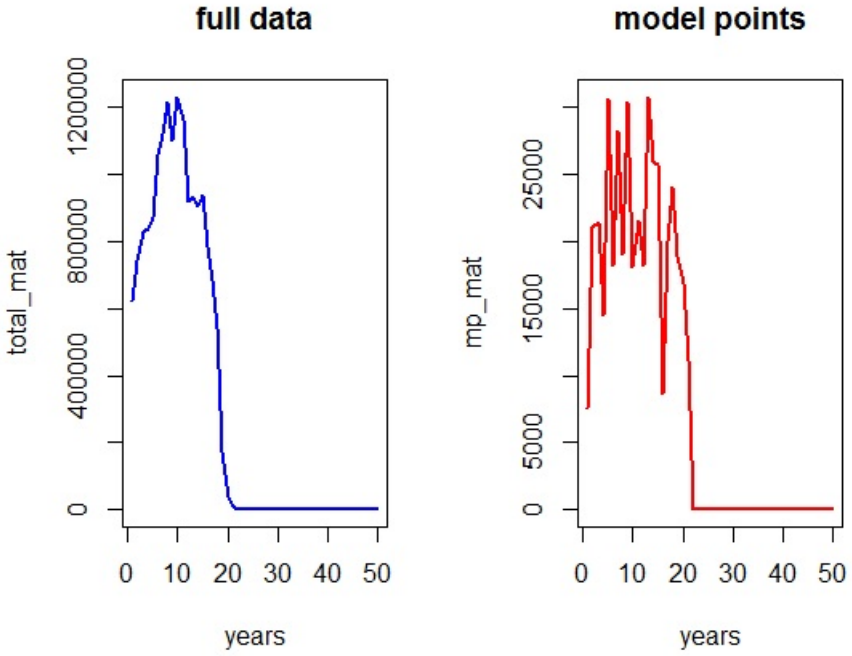
If we analyze the first variable Best Estimate Reserve and we plot the values using the original data in one graph and then the model points in a different one, we can see that the graphs below are similar so that it is a good optimization.

For the rest of the variables, Cash Flow Benefit Maturity, Cash Flow Benefit Surrender and Cash Flow Premium, the result is also good as we can see in the following graphs.

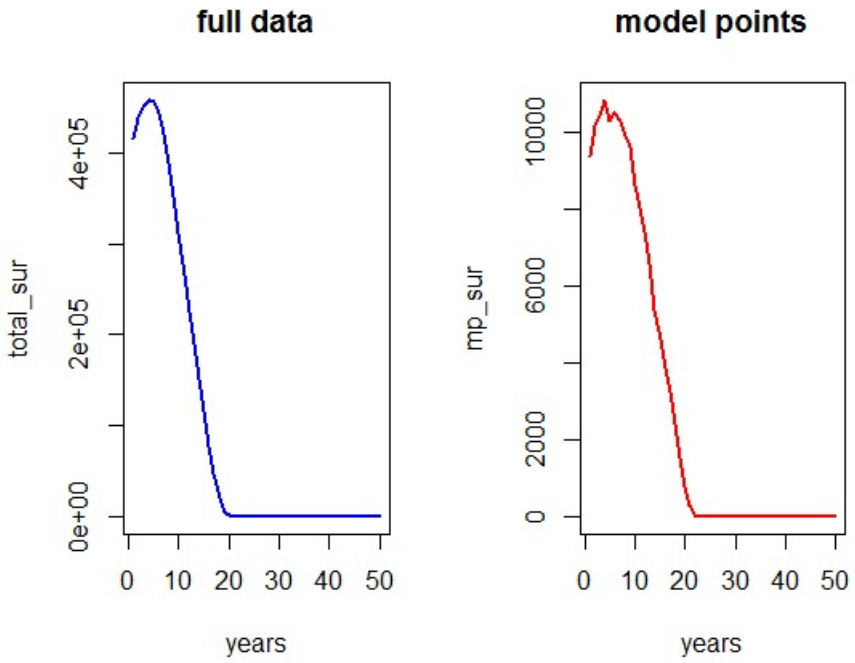
Best Estimate Reserve:



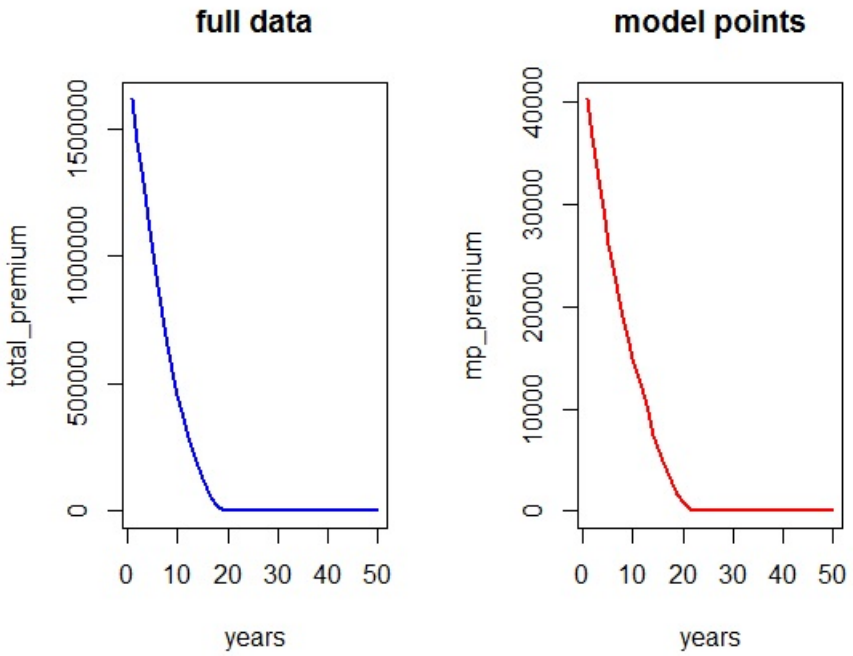
Cash Flow Benefit Maturity:



Cash Flow Benefit Surrender:



Cash Flow Premium:

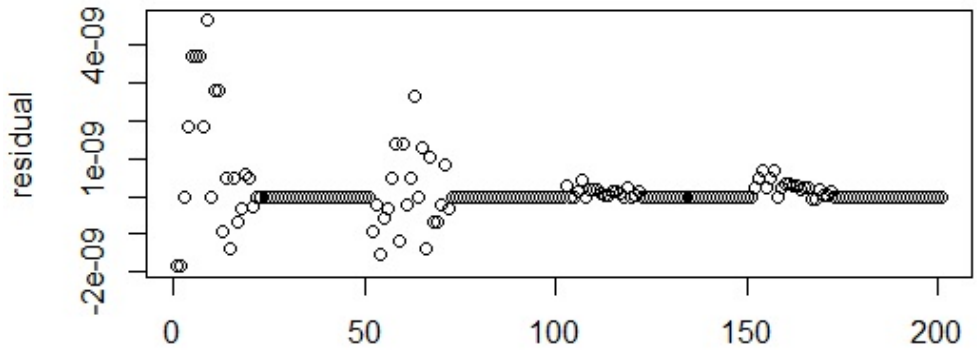


5.2 Residuals

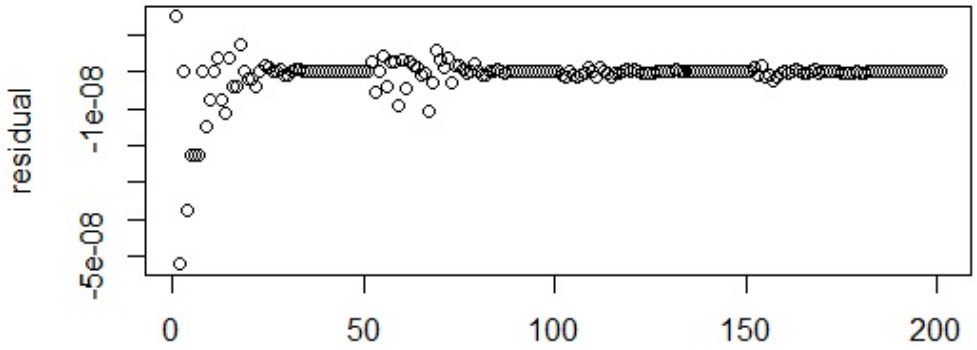
The result obtained when we optimize the first data set using the algorithm NNLS is a vector x of dimension $n \times 1$ with $n = 7045$. This vector contains 6934 elements equal to zero, hence we get 111 model points. Recall that the residuals are the differences between the actual observed value and the value as the model suggests. So, we have that

$$r_i = b_i - \sum_j A_{ij} x_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n,$$

and the objective of the least squares method is minimizes the sum of the squared residuals ($\sum_i r_i^2$). The residual sum of squares is 2.916×10^{-25}. We can see the graph which represents the residuals for each variable. The notation 4e-09 in the graph means 4×10^{-9}.



For the second data set, the result obtained with NNLS algorithm is a vector x which has a dimension $n \times 1$ with $n = 13924$. This vector contains 13748 elements equal to zero, therefore we get 176 model points. The residual sum of squares is 6.215×10^{-25}. We can show the same graph which represents the residuals for each variable.



5.3 Weights of model points

Now, we will show the main result, i.e. the vector x with the weights and the corresponding policy number taking the first data set as a reference.

weights	policy_number
23,2793965	10553589
54,7548323	10260885
58,9540202	10734007
55,5245756	10667860
121,01455	10705816
28,0590006	10577808
64,7385097	10700170
1	10731699
43,4821265	10161146
41,7088221	10636972
53,1121764	9973552
38,6636149	10199370
64,8888636	10704846
90,1889564	10671131
228,826671	10218507
89,1627496	10662357
33,7718938	10596384
9,98094281	10306066
55,4043659	10653269
32,6609099	10694419
30,4529152	10669525
20,6338355	10075452
2,40587591	10675399
87,8066256	10553602
23,2373751	10690565
96,6561426	10208533
13,4146818	9918533
18,2281978	10744518
11,6051549	9926266
15,521083	10551578
331,284659	10536072
12,0514817	10511039
38,9014655	10662580
4,68786302	10282423
41,9593828	10223576
74,0641325	10284751
69,051323	10223275
29,22451	10582026
75,28625	10697209
29,50164	10404696
36,68936	10091342
5,778776	9972045
80,0384	10231128
48,34975	10644016
176,7975	10570711
0,106851	10671571
31,07199	10706912
227,7385	10125807
44,18671	10690934
82,77315	10211850
47,36869	10664067
14,2846	10739578
262,4433	10493267
3,106747	10698567
68,29745	9803884
233,8324	10268049
1,097637	10635009
73,75743	10157367
90,51517	10663385
34,96842	10502938
6,134181	10564950
2,80307	10728398
128,9748	10660142
12,40796	10297955
5,35089	10713918
13,59736	10701894
133,2117	10172070
119,3993	10164004
20,03279	10675878
62,94311	10221471
3,844309	10734625
153,7205	10214271
47,34282	10612598
23,68216	10711208
26,26716	10697018
81,64236	10697843
40,24195	10673498
48,88901	10031454
96,5132	10727917
48,24618	10414822
6,668401	10660029
0,082449	10726992
90,97831	10125810
128,7463	10631142
167,8213	10639445
32,46399	10617713
115,4018	10659690
24,05216	10607370
31,52682	10595194
58,77992	10397790
13,34227	10620454
1	10739358
5,485393	10729151
6,611657	10670404
205,2639	10486821
98,01829	10568862
45,64564	10239230
2,116134	10612611
126,2407	10281136
99,90529	10296422
65,99279	10740554
74,05919	10231474
137,4378	10416875
133,0865	10682890
1	10709922
58,95371	10308941
97,43195	10392407
2,994094	10727467
43,6467	10611269
25,07895	10642827
204,9679	10167849

After this analysis, we deleted the first variable, Best Estimate Reserve, and we repeated the same process for the first data set with the other three variables. After that, we did the analysis with the first, third and fourth variables, i.e. removing the second one. Then, we repeated the procedure without the third variable and then without the fourth one.

The results were similar when we removed some variables. We get 89 model points in the first case, 81 model points in the second one, 90 and 80 model points in the other cases, respectively. So, we can reduce the model points since we got in the first calculation 111. Now, if we analyze the residual sum of the squares, we get a low sum only when we remove the second variable, since if we remove the first, third or fourth variables we do not get a good accuracy, as we can see in the graphs below.

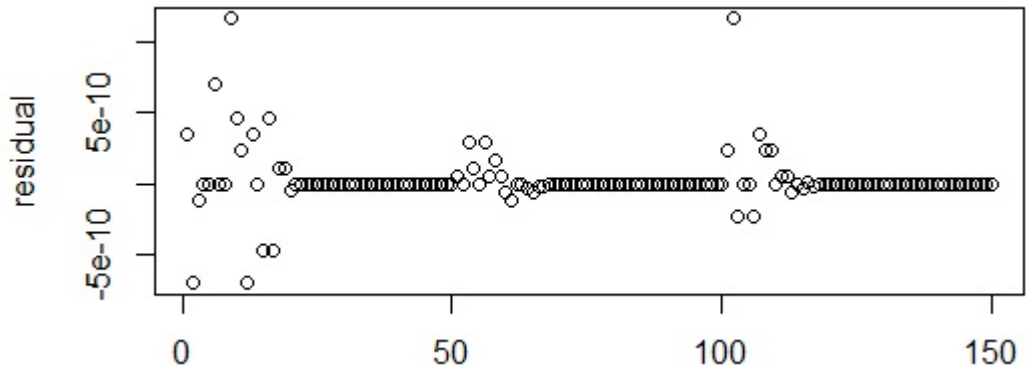


Figure 1: Residual values for each variable without the first one.

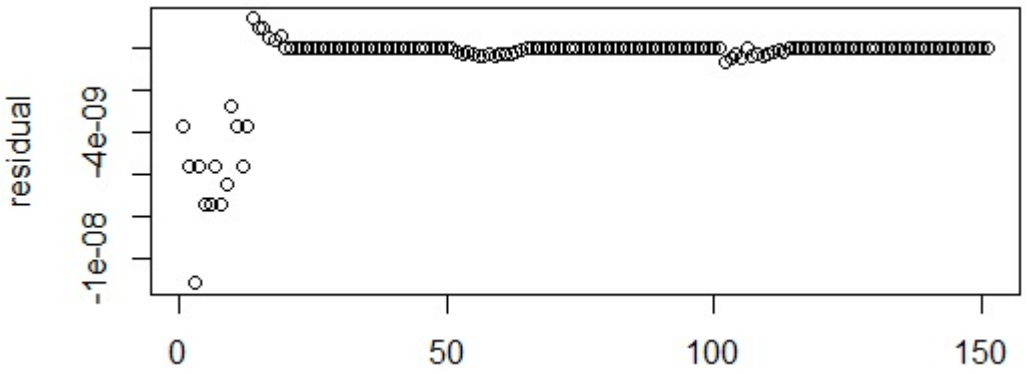


Figure 2: Residual values for each variable without the second one.

For the second data set, we did the same procedure with similar results, we can reduce the model points from 176 to around 135, but we have the same problem, we get only a good accuracy if we remove the second variable, Cash Flow Benefit Maturity.

Finally, we can conclude that we could remove just the second variable for computing the results, but since we have only four variables, it will take the same computing time more or less, so is better do the calculations with all the variables to get a better accuracy. Anyway, when NNLS optimization is carried out in a data set, we have to pay attention with the input data.

The number of variables that we consider when we implement the optimization affect the accuracy of the results, i.e. the more variables you have into account, the more exact the results are.

The accuracy of the data set has a negative correlation with the number of model points; when we reduce the number of variables, then we get less model points, but in proportion the residual sum of the squares increases.

6 Conclusion

In this work, we start from two data sets of insurance policies and we use NNLS optimization in program R to answer the following questions:

1. **Is it possible to reduce the number of model points with non-negative least squares optimization?**

   We can say that it is possible to reduce the number of model points with NNLS, since the result obtained using the algorithm NNLS gave good results fulfilling the constraints that we have established.

2. **How many model points do we obtain using NNLS?**

   We obtained 111 model points from 7045 policies in the first data set and 176 model points from 13924 policies en the second data set. We can even get less model points if we remove some variables but since we have just four variables, it does not worth it.

3. **What is the accuracy of the new model points?**

   The accuracy was good in all the variables as we can see in the graphs where we compare the use of the model points with the use of the full data.

   On the other hand, the sum of the residuals is 2.916×10^{-25} for the first data set and 6.215×10^{-25} for the second data set, but both residuals are higher in the variables which have high values, i.e. Best Estimate Reserve and Cash Flow Benefit Maturity.

The overall conclusion from this work is that the NNLS algorithm is a reasonable tool for findind model points in life insurance.

A Non-linear least squares

Below we will show an example of a non-linear least squares problem which can be solved using the Gauss-Newton method, (See [5]).

$$\text{minimize} \quad F(x) = \frac{1}{2}\sum_{i=0}^{n} f_i(x)^2 = \frac{1}{2}||f(x)||^2,$$

where $x \in \mathbb{R}^n$, $f_i : \mathbb{R}^n \to \mathbb{R}$ and $f(x) = (f_1(x), \ldots, f_m(x))^T$ is an even function of x. A necessary condition for any x, $\bar{x}$ must be a critical point of $g(x)$, which we define as $\nabla f(\bar{x})$ is equal to zero, $\nabla f(x)$ is equal to $J(x)^T f(x)$, where we define as Jacobian

$$J(x) := \frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Note that $g : \mathbb{R}^n \to \mathbb{R}^n$. We can know when $g(x)$ is equal to zero using Newton Raphson method, let x be the current estimator of $\bar{x}$, and calculates a Newton step p_k by solving the equation

$$g'(x^{(k)})p_k = -g(x^{(x)}),$$

and then update x to

$$x^{(k+1)} = x^{(k)} + \tau_k p_k,$$

where the variable τ_k is chosen in such a way that $F(x)^{(k+1)}$ strictly monotonically increasing. The matrix $g'(x)$ is the Hessian matrix of F and calculated

$$g'(x) = J_f(x)^T J_f(x) + \sum_{i}^{n} H_i f_i(x), \tag{A.1}$$

where $H_i(x) := \frac{\partial^2 f_i}{\partial x_s \partial x_t}$ is the Hessian matrix of $f_i(x)$.
Usually, if $||f_i||$ approaches zero when x^{k+1} approaching the solution, the second matrix of (A.1) also go to zero

$$J(x^{(k)})^T J(x^{(k)})P_k = -J(x^{(k)})^T f(x^{(k)}). \tag{A.2}$$

The solution to (A.2) is the solution to the least squares problem

$$\text{minimize} \quad \frac{1}{2}||J(X)^{(K)}P + F(x^k)||^2,$$

and the solution is unique if $J(x^{(k)})$ is of full rank.

References

[1] Sternberg Shlomo. *Dinamical systems, 2011.* *http://www.math.harvard.edu/library/sternberg/text/book.pdf*

[2] Donghui Chen and Robert J. Plemmons. *Non-negativity Constraints in numerical analysis.* *http://users.wfu.edu/plemmons/papers/nonneg.pdf*

[3] C.L.Lawson and R.J.Hanson. *Solving least squares problems.*

[4] Takeaki Kariya and Hiroshi Kurata. *Generalized least squares.*

[5] Ake Björck. *Numerical Methods for Least Squares Problems.*

[6] Soetaert K., Van den Meersche K., Van Oelven. *N.R-project User's Guide for the limSolve Package.* *http://cran.r-project.org/web/packages/limSolve/limSolve.pdf*

Non-exclusive licence to reproduce thesis and make thesis public

I, Miguel Cuevas Urosa
(*author's name*)
(date of birth: 10/01/1991),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Optimization of Model Points,
(title of thesis)

supervised by Kalev Pärna,
(supervisor's name)

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu/Tallinn/Narva/Pärnu/Viljandi, **19.05.2014**