Numerical Optimization

Numerical Introductory Course

Petra Burdejova

Franziska Schulz

Ladislaus von Bortkiewicz Chair of Statistics

C.A.S.E. – Center for Applied Statistics

and Economics

Humboldt–Universität zu Berlin

by

Michael Lebacher (564377)

Ken Schröder (568399)

Johannes Stoiber (564392)

Berlin, February 28, 2016

Contents

1 Introduction

Optimization is a topic that is of great importance in almost all aspects of life. People optimize the way they use their time and money and businesses optimize their profit functions through the maximization of revenues or the minimization of costs. But also in most branches of science optimization plays a central role. From this it can clearly be seen that the development of mathematical tools for optimization is very important and vital in practice as well as in academics.

Central to optimization in a mathematical sense is the so called objective function, which is the mathematical expression one wishes to minimize or maximize. The character of objective functions varies widely and in some cases the problem of maximization requires special methods. Especially in statistical applications the problem of maximization comes up very often. This is of major importance in computer based methods where no closed form solution is available. Here numerical optimization tools have improved the quality of statistical tools dramatically.

But also outside of the field of statistics numerical optimization algorithms are very useful. For example when the objective function is highly complex and dealing with it manually would require too much calculation time. In other cases, numerical optimization is strictly necessary, e.g. when the first derivative of the objective function does not exist in closed form and the solution cannot be found analytically.

This paper is structured as follows. In Section 2 the basics of numerical optimization will be explained. Section 3 discusses three different types of numerical optimization methods. In Section 4 the data and the used empirical method is described. Finally, the performance of the self-written codes will be compared with the pre-programmed algorithms in the `optim()`-package in `R`.

2 Methods of Numerical Optimization

This section presents the fundamentals of numerical optimization. All explanations and further theoretical aspects can be seen in detail for example in Nocedal and Wright (2006).

2.1 General Classifications of Optimization Problems

Many types of optimization problems require special procedures to accommodate for specific characteristics of the problem. There are three major distinctions which are suitable to cate-

gorize optimization problems. The three distinctions are: *discrete vs. continuous*, *constrained vs. non-constrained* and *local vs. global*.

The first distinction deals with the question whether the input (or the output) of the objective function is discrete or continuous. Depending on this question different optimization strategies must be applied. An example would be optimization in the field of logistics or decision theory. One often tries to optimize non-differentiable functions with discrete inputs in these fields. Therefore, one requires integer programming. If the number of possible outcomes or input values is small, simply *brute force* may be used. Another challenge arises if the maximization problem is constrained. This is very common e.g. in economic applications where households maximize their utility given a budget constraint. Such constraints on the inputs of a given problem will rule out a large set of the possible outcomes. A possible solution would be simplex algorithm. A further central issue is finding out whether or not some stationary point is a global or a local optimum. This is a very important problem in almost all mathematical theories where corner solutions are possible. However, most numerical methods are limited to finding local optima.

One could wonder whether it is important to distinguish between maximization and optimization. Here the convention in literature and practice is very convenient. All methods and theories are built for the goal of minimization. The maximum is found by minimizing the negative objective function.

In this paper only unconstrained maximization problems with continuous inputs and outputs will be considered. The problem of global optima will be ignored in the application of the algorithms to the logit model since its log-likelihood is globally concave which ensures a global and unique maximum.

2.2 The Role of the Gradient

Gradient based methods include the gradient of the objective function. The gradient will determine the direction in which the algorithm should move. In some methods additionally the hessian is needed. While the Gradient-Descent method only needs the gradient to work, the Newton-Raphson algorithm relies on the gradient as well as on the hessian. A middle way between this approaches can be seen in the so called quasi newton methods as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (see Broyden (1970), Fletcher (1970), Goldfarb (1970) and Shanno (1970)), that relies on the gradient but approximates the hessian matrix.

A major drawback of gradient based methods is the necessity of knowing the gradient.[1] In some cases the gradient might not exist or may be very costly to find. In such situations non-gradient based methods such as the Nelder-Mead algorithm (see Nelder and Mead (1965)) have proven to be valuable. The fact that no gradient is required makes this algorithm well-suited for general optimization problems. A drawback is that the low level of required information on the objective function generally results in much higher computational costs. Therefore the trade-off is very clear. The more information one has about the objective function and its properties (i.e. gradient, hessian) the less computational power is needed in order to find maxima or minima. However obtaining the derivatives may also be very problematic and costly.

3 Algorithms

The theoretical foundations of the selected algorithms will be discussed in this section. Each subsection starts with a short formal description of the problem that is solved by the algorithm of interest, followed by a description of specific features. In general, every algorithm is based on the following simplified structure:

1. $updater \leftarrow$ pre-defined rule
2. $x_{new} \leftarrow x_{old} + updater$
3. Convergence criterion satisfied?
    - No: start over at step 1.
    - Yes: terminate procedure.

In the beginning a rule is defined which defines how to move from an old value x_{old} to the new value x_{new} in the search for an optimum. This rule is called *updater*. The *updater* represents essentially "the next step" in the algorithm, which consists of a step size and a step direction. Since it is very unlikely that the algorithm hits the exact optima within a finite number of iterations, a termination criterion must be defined. This criterion often refers to the difference of x_{old} and x_{new}. If this difference becomes very small the procedure terminates and returns the last value of x_{new} which is the solution of the numerical maximization procedure.

[1] A problem that becomes even worse if additional to the gradient the hessian is needed.

3.1 Gradient-Descent

The Gradient-Descent is a standard procedure and provides a convenient method for unconstrained optimization problems in n-dimensional spaces. It is easy to implement but requires the first derivative of the objective function. This algorithm gives a solution to a minimization problem that is structured as follows:

$$\text{Minimize a function } f(x),\ x \in \mathbb{R}^n \text{ where } f \in C^1(\mathbb{R}^n, \mathbb{R}), n \geq 1.$$

The general idea of the Gradient-Descent is based on the fact that a differentiable function decreases fastest if one goes in the direction of its negative gradient. From this, a simple updating rule for the $(n+1)$th step can be derived:

$$x_{n+1} = x_n - \alpha_n \nabla f(x_n).$$

The objective function is denoted by $f(x)$, $\nabla f(x)$ gives the gradient and α_n is the so called learning parameter that determines the step size. The learning parameter has an important role to play since the speed of convergence depends crucially on it. As a naive but useful starting point $\alpha_n = \alpha \ \forall\ n$ can be used. This has the benefit of being simple to implement and does not demand any additional computational power. On the one hand the step size should be big if the algorithm is far from the solution in order to reach the goal very quickly. On the other hand the step size should be small if it becomes closer to the optimum in order to avoid overshooting. Such behavior is possible if the learning parameter is defined as $\alpha_n = h(n)$, where $h(n)$ gradually decreases with each iteration. This can be even further refined with the following approach:

$$\alpha_n = \begin{cases} \beta\alpha_{n-1}, & L(x_n) < 0 \\ \alpha_{n-1}, & L(x_n) \geq 0 \end{cases}$$

where $\beta \in (0, 1)$ and therefore reduces the step size. $L(x_n)$ specifies the change in a predefined loss function. A simple example for a loss function could be: $L(x_n) = f(x_n) - f(x_{n-1})$.

The choice for the learning parameter is crucial for the speed of convergence and in some cases even the functionality of the algorithm. In the self-written code, several options for the learning parameter have been implemented.

The iterations stop as soon as the Euclidean distance between x_n and x_{n-1} falls below a predefined threshold $\mathbf{tol} > 0$.

3.2 Newton-Raphson

The Newton-Raphson procedure is named after Sir Isaac Newton and Joseph Raphson and is a very popular algorithm for the numerical solution of nonlinear optimization problems. It requires the first and the second derivatives of the objective function. Therefore it has high demand for information but is extremely fast. The method is applied to problems that are structured as follows:

$$\text{Minimize a function } f(x),\ x \in \mathbb{R}^n \text{ where } f \in C^2(\mathbb{R}^n, \mathbb{R}), n \geq 1.$$

From a theoretical perspective, the Newton-Raphson algorithm solves the minimization problem by applying a local linearization. In order to see this, consider a Taylor expansion of second order near the optimum x^*, applied to $f(x)$:

$$f(x) \approx f(x^*) + \nabla f(x^*)'(x - x^*) + 0.5(x - x^*)'\nabla^2 f(x^*)(x - x^*).$$

Obviously $\nabla^2 f(x^*)$ denotes the hessian. This linearization can be differentiated w.r.t. x:

$$\nabla f(x) \approx \nabla f(x^*) + \nabla^2 f(x^*)(x - x^*).$$

A minimizer must satisfy $\nabla f(x) = 0$. Therefore we get:

$$\nabla f(x^*) + \nabla^2 f(x^*)(x - x^*) = 0.$$

This results after rearrangement and given that the hessian is nonsingular in the following expression:

$$x = x^* - [\nabla^2 f(x^*)]^{-1} \nabla f(x^*).$$

The formula above easily leads to an updater function:

$$x_{n+1} \leftarrow x_n - [\nabla^2 f(x_n)]^{-1} \nabla f(x_n).$$

Note that this expression can be interpreted as a Gradient-Descent algorithm, that uses the inverted hessian as learning parameter: $\alpha_n = [\nabla^2 f(x_n)]^{-1}$. Even though the Newton-Raphson procedure requires a lot of information input, this procedure is very popular, due to its exceptional convergence speed if $\nabla^2 f(x)$ is positive definite.

The iterations stop as soon as the Euclidean distance between x_n and x_{n-1} falls below a predefined threshold $\mathbf{tol} > 0$.

3.3 Nelder-Mead

The Nelder-Mead method is also known as downhill simplex method[2] and was proposed by Nelder and Mead (1965) to solve unrestricted optimization problems in n-dimensional spaces. It is a non-gradient based approach and therefore very popular if it is impossible or costly to differentiate the objective function. Formally expressed, the Nelder-Mead method solves the following types of problems:

$$\text{Minimize a function } f(x),\ x \in \mathbb{R}^n \text{ with continuous } f \in C(\mathbb{R}^n, \mathbb{R}), n \geq 2.$$

There is no general rule which initial values should be taken to start the algorithm. For the implementation of the Nelder-Mead algorithm the same procedure as in the `fminsearch`-function in `R` is used (Sebastien Bihorel, 2015). The first vertex $x_0 \in \mathbb{R}^n$ is a random guess. The remaining n vertices are computed as follows:

$$(x_j)_i = \begin{cases} (x_0)_i + 0.05 & \text{if} (x_0)_i \neq 0 \\ (x_0)_i + 0.00025 & \text{if} (x_0)_i = 0 \end{cases}$$

for i $\in 1...n, j \in 1...n$.

This initialization process results in a simplex which represents the the convex hull around its $n+1$ vertices:

$$\Delta = \{x_0, \cdots, x_n\} \subset \mathbb{R}^n.$$

After having constructed the initial simplex, the iteration process starts. A graphical overview is given in Figure 7 in the Appendix. As a first step, the components of Δ will be assessed and ordered in the following way:

x_w	:=	$\text{argmax}\{f(x) : x \in \Delta\}$	worst point
x_{sw}	:=	$\text{argmax}\{f(x) : x \in \Delta, sw \neq w\}$	second worst point
x_b	:=	$\text{argmin}\{f(x) : x \in \Delta, b \neq sw, w\}$	best point

In the second step, the corresponding centroid point $z := \frac{1}{n} \sum_{j \neq w} x_j$ is computed by excluding the worst performing vertex x_w. Since one wants to move away from the worst point x_w, one reflects this point in the opposite direction and obtains $r = \alpha(z - x_w)$, with $\alpha = 1$. Graphically this can be seen in Figure 1.

[2]Note that downhill simplex is not connected to the simplex algorithm mentioned in section 2.1.

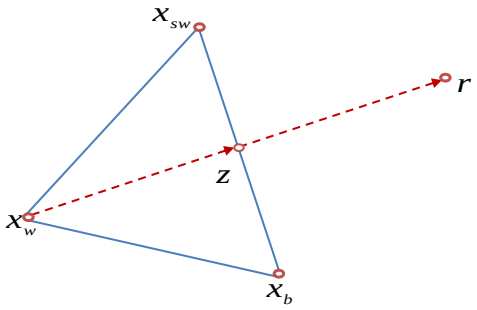

Figure 1: Reflection of worst point.

As a third step the objective function is evaluated at the reflection point as well as at the best and second best one. If this results in $f(x_b) \leq f(r) \leq f(x_{sw})$, then x_w is replaced by r as depicted in Figure 2.

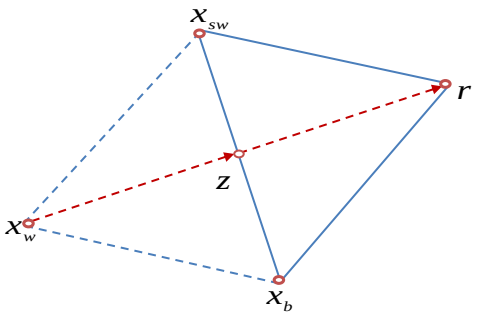

Figure 2: Simplex after replacing x_w by r.

At step four $f(r) < f(x_b)$ is checked. This case means r performs even better than the best vertex in the origin simplex. In this case one knows, that one moves in the right direction and wants to go even further. Therefore a expansion point e defined as $\beta(z - x_w)$ is computed, with $\beta = 2$ (in general, $\beta > \alpha$). If $f(e)$ ¡ $f(r)$, then x_w is replaced by e, which is illustrated in in Figure 3.

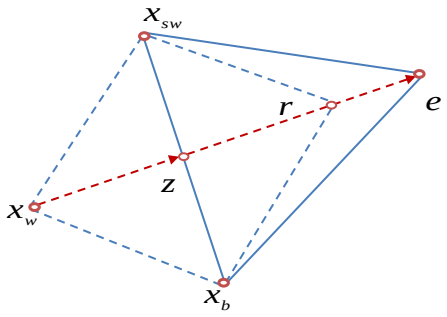

Figure 3: Simplex after replacing x_w by e.

As a fifth step, in the case $f(r) > f(x_{sw})$ a contraction point k is computed. Obviously one hasn't identified a point that would help to move the simplex in the right direction. If $f(r) > f(x_w)$, then k is computed by $z + \gamma(x_w - z)$ otherwise it is computed via $z + \gamma(z - x_w)$ with $0 < \gamma < \alpha$ (common practice: $\gamma = 0.5$). If this contraction point improves the vertices of the simplex, formally $f(k) < f(x_w)$, then x_w is replaced by k and is shown in Figure 4.

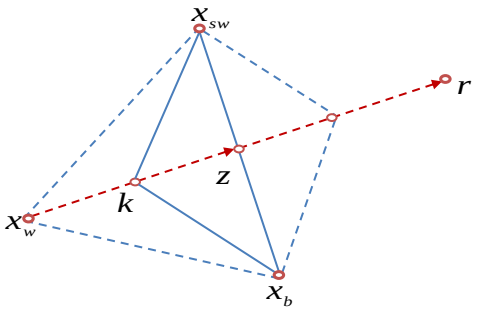

Figure 4: Simplex after replacing x_w by k.

In the case the contraction point also does not help to find the minimum, the whole simplex is contracted by the worst performing point. This means that the new simplex is then stretched by $x_j := \frac{(x_j + x_b)}{2}$. This five steps are repeated until the termination criterion is satisfied. Nelder and Mead proposed the following termination criterion $\frac{1}{n+1} \sum_{j=0}^{n} \left(f(x_j) - \bar{f} \right)^2 \leqslant \mathbf{tol^2}$ with $\bar{f} = \frac{1}{n+1} \sum_{j=0}^{n} f(x_j)$ and $\mathbf{tol} > 0$ is some minimum required update size between iterations.

4 Data and Empirical Model

The algorithms presented in the previous section where written and applied in `R` in order to optimize a likelihood function, for which a data set from Mroz (1987) was used. This analysis cannot be seen as an approach of careful statistical modelling and interpretation but only as a demonstration of the working of the algorithms. The relationship of interest is kept as simply as possible and is just the functional dependence between female labor force participation (inlf), education (educ) and the income of the husband (huswage).[3] Table 3 in the Appendix gives the summary statistics of the used variables.

In order to model the relationship between the probability of being in the labor force and the explanatory variables a logit model was employed:[4]

$$P(infl_i = 1) = \Lambda(\beta_0 + \beta_1 educ_i + \beta_2 huswage_i) = \Lambda(x_i'\boldsymbol{\beta}) \; \forall \; i \in \{1, ..., 753\}$$

where $\Lambda(x)$ gives the logistic link function and x_i is the vector of covariates. The vector $\boldsymbol{\beta} = (\beta_0 \;\; \beta_1 \;\; \beta_2)'$ gives the parameters that should be estimated by Maximum Likelihood. Assuming independent and identically distributed observations the likelihood function is given by:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{N} \Lambda(x_i'\boldsymbol{\beta})^{y_i} [1 - \Lambda(x_i'\boldsymbol{\beta})]^{(1-y_i)}.$$

Since taking the natural logarithm is a monotone transformation, this is equivalent to the maximization of the log-likelihood:

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{N} \Big[y_i \ln \big[\Lambda(x_i'\boldsymbol{\beta})\big] + (1 - y_i) \ln \big[1 - \Lambda(x_i'\boldsymbol{\beta})\big] \Big].$$

Normally, the maximum would be found by setting the gradient $g(\boldsymbol{\beta})$ equal to zero:

$$g(\boldsymbol{\beta}) = \sum_{i=1}^{N} (y_i - \Lambda(x_i'\boldsymbol{\beta}))x_i \stackrel{!}{=} 0.$$

However, no closed form solution is available and therefore numerical methods must be employed. In order to employ the Newton-Raphson procedure for the logit model estimation, the gradient from above will be taken and also the hessian is needed:

$$h(\boldsymbol{\beta}) = -\sum_{i=1}^{N} \Lambda(x_i'\boldsymbol{\beta})[1 - \Lambda(x_i'\boldsymbol{\beta})]x_i x_i'.$$

[3] For the sake of a more stable performance of the numerical methods the data has been scaled according to the following formula: $x^* = \frac{x - \min(x)}{\max(x) - \min(x)}$.

[4] The following definitions and derivations are standard in statistical and econometric analysis and can be found in textbooks like Cameron and Trivedi (2005) or Wooldridge (2010).

Now the objective function is formulated and its gradient and hessian are available. In order to initiate iterations starting values are needed. Choosing starting values close to the optimum would decrease computational costs significantly. However, it is reasonable to include as little prior knowledge in the algorithms as possible. Therefore random starting values from a uniform distribution are used.

The log-likelihood function is a globally concave function, which implies that every local maximum is always a global maximum as well. It immediately follows that the negative log-likelihood leads to an unique minimum. Therefore no problems related to multiple maxima arise. Figure 4 shows the log-likelihood function in the three dimensional space. The coloring indicates that there exists only one unique maximum.

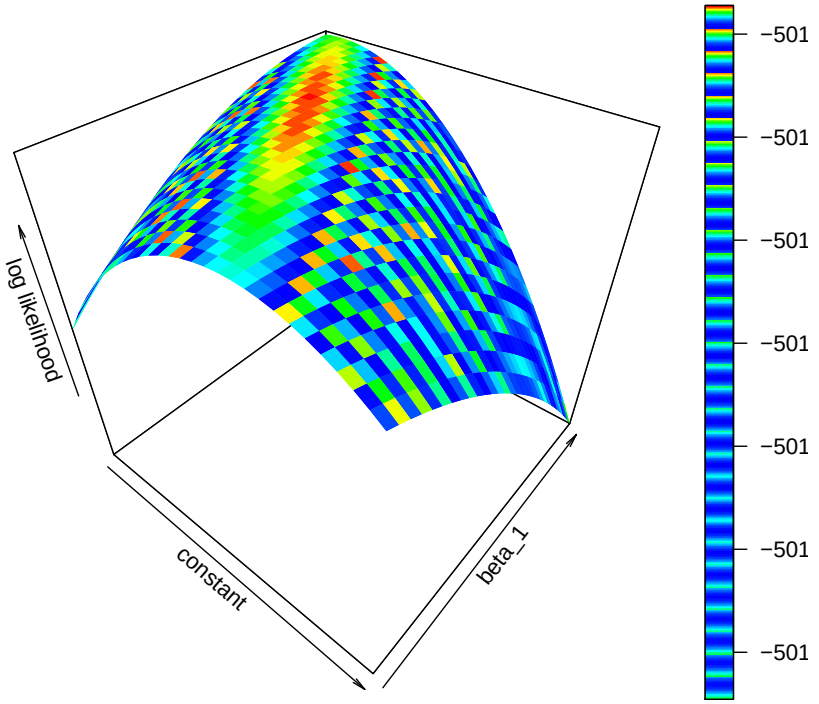

Figure 5: Visualization of the log-likelihood function.

5 Results and Conclusions

All algorithms were able to compute a solution vector. The logit model reveals a strong relationship between the female labour force participation rate and hourly wage of the husband and years of education. The concrete results are shown in table 1. This results are intuitive and have the expected signs. Furthermore they are very close to the solution of `optim` package. But there are huge differences regarding the performance of the different algorithms as

can be seen in table 1. The results clearly confirm the trade-off between prior information and computational costs. Additionally, it becomes obvious that the self-written Nelder-Mead algorithm needs 10 times more iterations than the pre-programmed one.

Table 1: Model Comparison

	β_0	β_1	β_2	Iterations	Log-Likelihood
Nelder-Mead, `optim` package	-0.783	2.571	-2.778	156	494.798
Nelder-Mead	-0.783	2.569	-2.770	1,575	494.798
Newton-Raphson	-0.782	2.570	-2.774	6	494.798
Gradien Descent, fix α	-0.783	2.568	-2.769	3,227	494.798
Gradient Descent, flex α	-0.782	2.570	-2.775	1,191	494.798

Notes: Application of the algorithms in a logit regression model. With tolerance level **tol** = 0.0001.

As can be seen in figure 6, the convergence behavior of the algorithms show different patterns. While Newton-Raphson converges extremely fast, Gradient-Descent without a flexible learning parameter converges very slow and gradual. However, with a flexible learning parameter the Gradient-Descent becomes faster but also more jumpy indicating frequent overshooting until stabilization. Nealder-Mead exhibits a very slow and ragged convergence behavior.

Besides the number of iterations it is also of interest how the calculation costs measuring calculation time develop differently among algorithms when the level of accuracy increases. In table 2 this development is shown. Especially the costs of the Nelder-Mead algorithm clearly increase with the level of accuracy, whereas the other codes do not seem to require a significant extra calculation time.

Finally, the tables 1 and 2 indicate that the `optim` package manages to optimize the log-likelihood function much more efficient and in a fraction of a second, whereas the self-written algorithms require up to 1000 times more time.

It is worth to mention that all algorithms also were applied to optimize $f(x_1, x_2) = 2x_1^2 - 4x_1x_2 + x_2^4 + 2$, which is neither a globally convex nor a globally concave function with multiple optima. The convergence behavoir of the self-written algorithms depends heavily on the starting values chosen. In some cases the algorithms were unable to find any local optima because of overshooting. All in all, none of the used algorithms can savely be applied for functions with multiple optima.

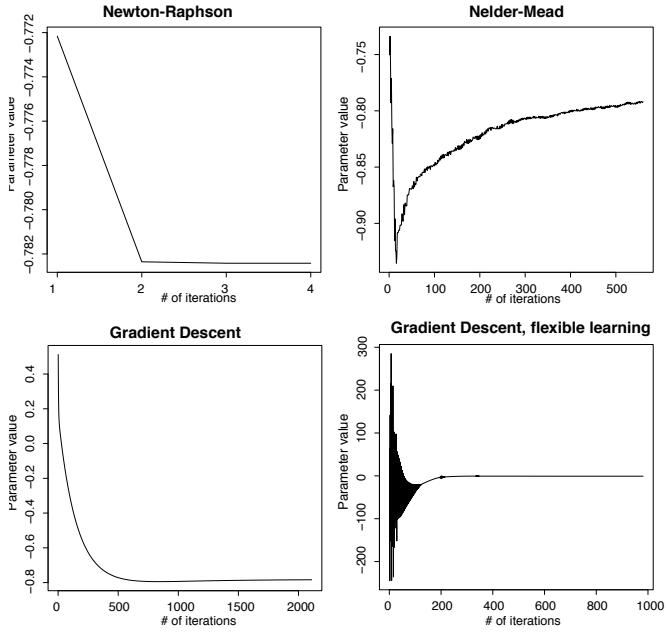

Figure 6: Paths of convergence of β_0, with **tol** $= 0.0001$.

Table 2: Accuracy Sensitivity, Calculation Time in Seconds

tol	Newton-Raphson	Nelder-Mead	Gradient Desc., fix	Gradient Desc., flex	`optim`
0.1	0.110	0.380	0.336	0.228	0.0003
0.01	0.127	0.728	0.355	0.233	0.0003
0.001	0.168	1.372	0.339	0.219	0.003
1e-04	0.176	1.957	0.357	0.226	0.003
1e-05	0.163	2.472	0.361	0.216	0.007

Notes: Application of the algorithms in a logit regression model. The numbers are averaged across the performance of 10 estimates.

To sum up, four major results can be identified. First, there exists a general trade-off between the information one puts in an algorithm and its performance. As very robust algorithm, Nelder-Mead can work out an existing global optima, but has proven to be very inefficient. From this it becomes clear that whenever possible, gradient based methods should be employed. Second, if one decides for a gradient based method it is always preferable also to include the hessian if possible, since the inclusion of the hessian dramatically improves the performance. If one opts for the Gradient-Descent without the hessian it is very important

to think about an intelligent learning parameter since results change extremely depending on this parameter. Third all these algorithms considered here cannot detect global optima and should be used with care therefore. As the fourth and last finding it must be highlighted that the usage of the pre-programmed `optim` package is highly recommended since it performs astonishing well.

References

Broyden, C. G. (1970): "The convergence of a class of double-rank minimization algorithms 1. general considerations," *IMA Journal of Applied Mathematics*, 6, 76–90.

Cameron, A. C. and P. K. Trivedi (2005): *Microeconometrics: methods and applications*, Cambridge university press.

Fletcher, R. (1970): "A new approach to variable metric algorithms," *The computer journal*, 13, 317–322.

Goldfarb, D. (1970): "A family of variable-metric methods derived by variational means," *Mathematics of computation*, 24, 23–26.

Mroz, T. A. (1987): "The sensitivity of an empirical model of married women's hours of work to economic and statistical assumptions," *Econometrica: Journal of the Econometric Society*, 765–799.

Nelder, J. A. and R. Mead (1965): "A Simplex Method for Function Minimization," *The Computer Journal*, 7, 308–313.

Nocedal, J. and S. Wright (2006): *Numerical optimization*, Springer Science & Business Media.

Sebastien Bihorel, M. B. (2015): *R port of the Scilab neldermead module*, r package version 1.0 - 10.

Shanno, D. F. (1970): "Conditioning of quasi-Newton methods for function minimization," *Mathematics of computation*, 24, 647–656.

Wooldridge, J. M. (2010): *Econometric analysis of cross section and panel data*, MIT press.

Appendix

Table 3: Summary Statistics

	inlf	educ	huswage
Description	labor force dummy	years of schooling	wage/h husband
Min.	0	5	0.412
1st Qu.	0	12	4.788
Median	1	12	6.976
Mean	0.568	12.290	7.482
3rd Qu.	1	13	9.167
Max.	1	17	40.510
N	753	753	753

Source: Mroz (1987)

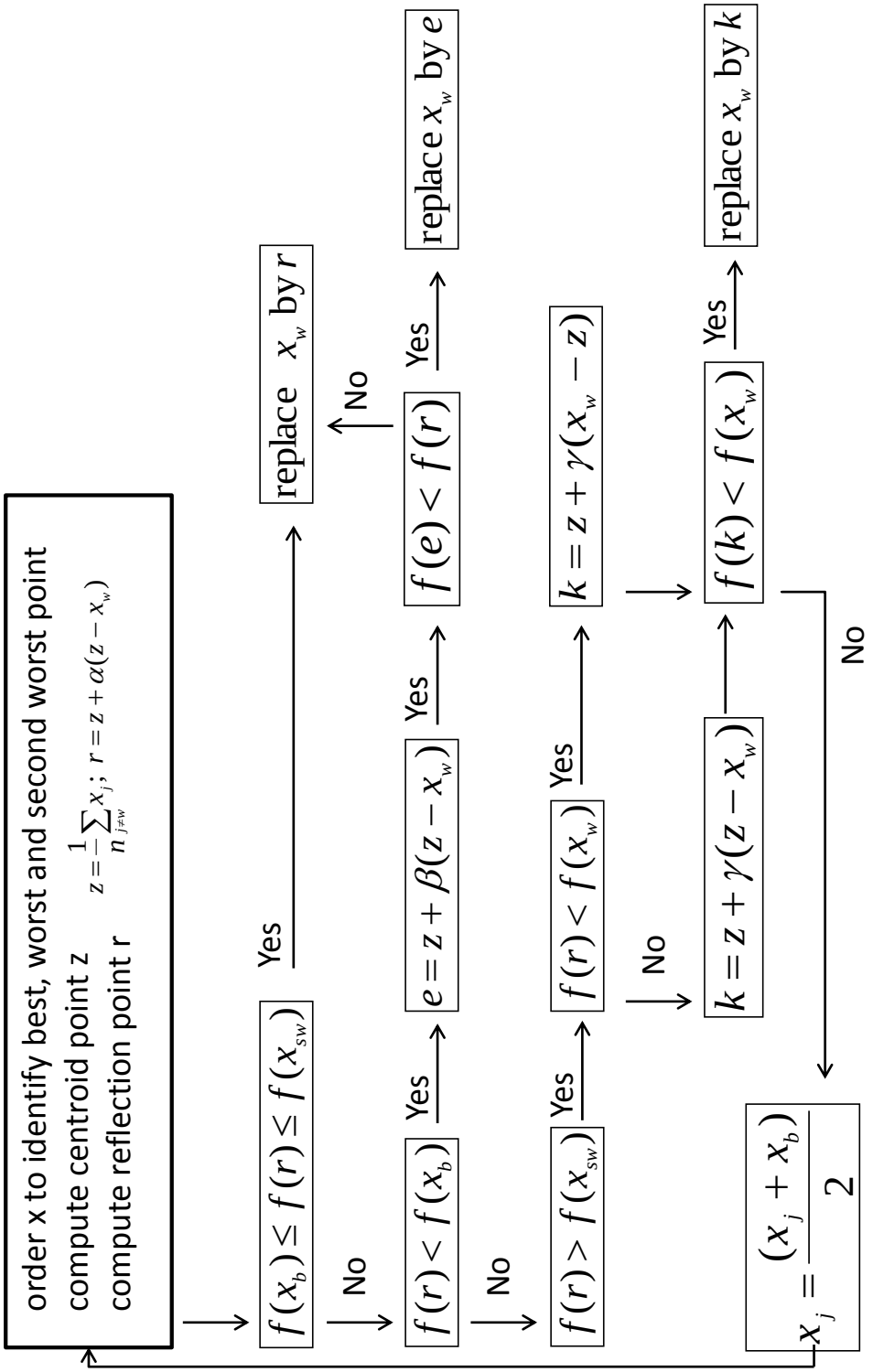

Figure 7: Nelder-Mead algorithm depicted as a flow chart.