Optimization Methods to Solve *Structured Total Least Squares* (STLS)

Josefin Häggström

June 4, 2005
Master's Thesis in Computing Science, 20 credits
Supervisor at CS-UmU: Jerry Eriksson
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

This Master's thesis describes the invention and development of a new method to solve Structured Total Least Squares; STLS problems without making use of the Hessian. The new method is a Gauß-Newton method with the Lagrange function and it is called *The center point method*. The center point method is simple to comprehend and implement, its convergence rate is fast and it has a reliable probability of convergence.

Several test (such as the the probability of convergence, the correctness of the minimum, the required number of iterations and CPU-time and the reduction rate of the constraints etc.) have been done where the results of the center point method are compared to the results of a regular Gauß-Newton method with the Lagrange function and Matlab 7's built-in function `fmincon`.

The results obtained clearly show that the center point method is faster (in iterations, CPU-time and reduction rate of the constraints) and more reliable (in probability of convergence) than the other methods. The results also deduce that a minimum found by the center point method differs from the minimum found by Matlab's `fmincon` on the 8th to the 2nd decimal. The sign of $(x_{min\,center\ point} - x_{min\texttt{fmincon}})$ is random.

In addition to the subjects above, this Master's thesis also contains the theory behind the Tikhonov method and comparisons between the center point method and the Tikhonov method with the addendum of a center point. The minima found by the two methods are not always the same and they differ on the 10th to the 2nd decimal. The results regarding reliability and performance of the two methods are similar, but the probability of convergence for center point method is a bit higher and its amount of CPU-time is a bit shorter.

Optimization Methods to Solve *Structured Total Least Squares* (STLS)

Contents

Chapter 1

Introduction

This master's thesis is developed at Umeå University, at the Department of Computing Science. The original idea of this research area and of this particular project are both by Jerry Eriksson.

The first goal of this project is to study the existing methods which can solve STLS (Structured Total Least Squares) problems and from the results of the studies discover their weakness as well as their strength. The second goal is to develop a new method which solves STLS problems in an efficient, intuitive and also reliable way, as such a method does not exist today. To achieve efficiency, the method must not make use of the Hessian, to make the method intuitive its procedures must be simple and to attain reliability, the method must always find a minimum. A method which is both fast, simple and reliable is highly desired in all sectors where physical real-world results have to be refined into a nonlinear minimization problem which eventually has to solved.
From these desires and constraints a new method, called *The center point method* is born.

Chapter 2 and 3 cover the basic definitions and elementary optimization theory needed by an unexperienced reader to comprehend the overall meaning, goal and in the end results of this research project.
The idea of the center point method is thoroughly described in chapter 4 and 5. Both the mathematical theory and the implementation of the method are discussed.
After the theoretical descriptions, follows chapter 6 which consists of results. Matlab 7's built-in function `fmincon`, the general Gauß-Newton method and the Tikhonov method with the addendum of a center point are used as a comparison and reference. The results are in the form of graphs showing different properties of the methods, such as the probability of convergence, the CPU-time, the number of iterations required to find a minimum etc. The chapter finishes with reflections and conclusions regarding the results.
Finally, the thesis is ended with chapter 7 which contains a summary of conclusions. The reliability, performance and correctness of the general Gauß-Newton method, Matlab's `fmincon`, the Tikhonov method with the addendum of a center point and the center point method are discussed respectively.

Chapter 2

Basic Definitions for Least Squares

2.1 Least Squares; LS

A Least Squares (LS) problem can be formulated as a problem of the form:

$$\min_x \ \|b - Ax\|$$

where $Ax \approx b$ and $A \in \Re^{m \times n}$ with $m > n$, $x \in \Re^{n \times 1}$ and $b \in \Re^{m \times 1}$. A reformulation of $Ax \approx b$ is $Ax + r = b$, where $r \perp R(A)$ is the residual.
That is;

$$\min_x \ \|b - Ax\| = \min_x \ \|r\|$$

and hence r is to be minimized for Ax to be as close to b as possible.

A LS problem can be solved in one step with SVD, QR or cholevsky decomposition.

2.2 Total Least Squares; TLS

A Total Least Squares (TLS) problem is a LS problem where there are errors in b as well as in A. Hence, the problem $Ax \approx b$ has the form:

$$(A + \Delta A)x = b + \Delta b$$

where ΔA and Δb are the errors in the matrices A and B respectively.

This reduces the minimization of $\|Ax - b\|_2$ to the following TLS problem:

$$\begin{aligned} &\min_{\Delta A, \Delta b} \ \|\Delta A, \Delta b\|_F \\ &s.t. \ (A + \Delta A)x = b + \Delta b \end{aligned}$$

where $\|A\|_F$ is the Frobenius matrix norm of A.

A TLS problem can be solved in one step with SVD (see [3]); that is compute $[A; b] = U\Sigma V^T$. A TLS problem has a unique solution if the singular values; $\sigma_i \in \Sigma$ have the property that $\sigma_n > \sigma_{n+1}$ and $\nu_{n+1,n+1} \neq 0$ where $\nu_{j,j} \in V$.

2.3 Structured Total Least Squares; STLS

A Structured Total Least Squares (STLS) problem is a TLS problem of the form:

$$(A + \Delta A)x = b + \Delta b$$

where ΔA and Δb are the errors in the matrices A and B respectively.

A STLS problem has in addition to a regular TLS problem the property that ΔA has the same structure as A (for example the Toeplitz structure, see section 3.8 on page 12).

A vector $\alpha \in \Re^{k\times 1}$ represents the errors in ΔA, where k is the number of *distinct* elements in ΔA (that is; elements which do not have the same value). This means that the position of the elements in ΔA is specified by the position of those of A. Δb is also a function of α and x and hence $\Delta b = \Delta b(\alpha, x)$. Let $D \in \Re^{k\times k}$ be a diagonal matrix that contains the repetition of α in ΔA.

Then the minimization of $\|Ax - b\|$ becomes the following STLS problem:

$$\begin{aligned} &\min_{\alpha,x} \left\| \begin{array}{c} \Delta b(\alpha, x) \\ D\alpha \end{array} \right\|_p \\ &s.t.\ (A + \Delta A)x = b + \Delta b \end{aligned}$$

where $\|A\|_p$ is the vector p-norm of A, for $p = 1, 2$ or ∞.

Two examples of matrices with the properties of a STLS problem are Toeplitz matrices and the Hankel matrices. (see section 3.8 on page 12 and section 3.9 on page 13).

There is no way to solve a STLS problem in one step. A minimum has to be found iteratively.

Chapter 3

Elementary Optimization Theory

3.1 Newton' s Method

Newton's method is commonly used for general nonlinear problems without constraints. It is simple to comprehend and implement. Though when it comes to solving huge problems where efficiency is required it does not suffice.

To calculate the nonlinear least squares (NLS) problem:

$$\min_x \ \tfrac{1}{2} \|f(x)\|^2$$

is equivalent to calculating:

$$\min_p \ \tfrac{1}{2} \|f(x+p)\|^2$$

where p is the direction to the next point x_{k+1}, and this new point $x_{k+1} = x_k + p$. p is such that $f(x_{k+1}) < f(x_k)$.

Newton's method uses the Taylor approximation of $f(x+p)$. The Taylor approximation is:

$$f(x+p) = \lim_{n\to\infty} f(x) + Jp + \tfrac{1}{2}p^T Hp + ... + \mathcal{O}\|p\|^n)$$

where J is the Jacobian of $f(x)$ and H is the Hessian of $f(x)$.

Hence the least squares problem

$$\min_p \ \tfrac{1}{2} \|f(x+p)\|^2$$

can be approximated by:

$$\min_p \ \tfrac{1}{2} \left\| f(x) + Jp + ... + \mathcal{O}(\|p\|^k) \right\|^2$$

where the error in the approximation depends on k. That is; if $k \to \infty$ the approximation will be exact. J is the Jacobian of $f(x)$ and p is the direction to the next point x_{k+1} where $x_{k+1} = x_k + p$ such that $f(x_{k+1}) < f(x_k)$.

There are several versions of Newton's method, with different values of k and hence also different approximations of the k:th term in the Taylor approximation. Usually in

the different versions of Newton's method, terms up to the power of 2 are used. If terms of a higher were to be added, the approximation would be more precise, but the high cost to calculate them would make the method too slow.

3.2 The Gauß-Newton Method

The Gauß-Newton method is a solver for NLS problems and it is mainly used when there are no constraints to respect. To be able to solve constrained minimization problems, another course of action must be added, for example the Lagrange function (See section 3.4 on page 7).
The Gauß-Newton method only uses an approximation of the Hessian and hence it does not always converge towards a solution point.

The Gauß-Newton method is a Newton method with $k = 2$, that is:

$$f(x+p) \approx f(x) + Jp + \mathcal{O}(\|p\|^2)$$

where J is the Jacobian of $f(x)$, $x_{k+1} = x_k + p$ with p such that $f(x_{k+1}) < f(x_k)$ and the approximation of $H = J^T J + \sum f_i f_i''$ is $H = J^T J$. This approximation works very well close to a minimum x_{min} where the residual $r = \sum f_i f_i''$ is small, but far from x_{min} this method acts in an unpredictable manner.
From the approximation of H, the direction p is obtained as the following:

$$p = -J^+ f$$

Hence, the least squares problem to solve is:

$$\min_p \ \tfrac{1}{2} \|f(x) + Jp\|^2$$

with $x_{k+1} = x_k + p$ where $p = -J^+ f$.

3.3 The Tikhonov Method

The Tikhonov method solves nonlinear LS problems with constraints reformulating the LS problem such that it will not have constraints anymore.

A NLS problem of the form:

$$\begin{aligned} &\min_p \ \tfrac{1}{2} \|f(x+p)\|^2 \\ &s.t.\ g(x+p) = 0 \end{aligned}$$

is reformulated as the following with the Tikhonov method:

$$\min_p \ \tfrac{1}{2} \|g(x+p)\|^2 + \mu^2 \|f(x+p)\|^2$$

where p is the search direction and μ is a "punishment-scalar".

Since the constraints are "skipped" and inserted into the object function with the Tikhonov method, it is very efficient when the NLS problem is big. This because a lot of calculations with big matrices are rejected.

The next point $x_{k+1} = x_k + \alpha p_k$, where p_k is the search direction and α is a scalar which scales p_k (see section 3.6.2 on page 10 for further information on the choice of α).

The formula for the search direction p_k in step k is:

$$A_{aug} p_k = -g_{aug}$$

where $A_{aug} = \begin{bmatrix} & A & \\ \mu & & \\ & \ddots & \\ & & \mu \end{bmatrix}$, A is the Jacobian of $g(x)$ and $g_{aug} = \begin{bmatrix} g(x) \\ \mu f(x) \end{bmatrix}$

As for the punishment-scalar μ, it has empirically been discovered [1] that a good start value; μ_{start} of μ is 0.1. To pick the punishment-scalar μ_k in each step the following algorithm is used:

1. if $\alpha = 1$:
   $\mu_{k+1} = \mu_k / 10$
2. if $\alpha = 0.5$:
   $\mu_{k+1} = \mu_k / 5$
3. for all other values of α:
   $\mu_{k+1} = \mu_k$

Thus, $\mu_k = \mu_k(\alpha)$ depends on how good the search direction p_k is and consequently it depends on the scaling factor α.

3.4 The Lagrange Function

The Lagrange function is used when in addition to the minimization of $\frac{1}{2}\|f(x)\|^2$, there is a constraint $g(x) = 0$ to respect.

For a least squares problem of the form:

$$\begin{aligned} &\min_x \ \tfrac{1}{2}\|f(x)\|^2 \\ &s.t.\ g(x) = 0 \end{aligned}$$

The definition of the Lagrange function is:

$$\mathfrak{L}(x, \lambda) = \frac{1}{2}\|f(x)\|^2 - \lambda^T g(x)$$

where $\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix}$ and the λ_i, $i = \{1, 2, \ldots, m\}$ are called *Lagrange multipliers.*

From the definition of $\mathfrak{L}(x, \lambda)$, the gradients can be calculated:

$$\begin{aligned} \nabla \mathfrak{L}_x(x, \lambda) &= J^T f(x) - A^T \lambda \\ \nabla \mathfrak{L}_\lambda(x, \lambda) &= -g(x) \end{aligned}$$

Lagrange discovered that when $x = x_{min}$ is a minimizer, the following property holds:

$$\nabla \mathfrak{L}_x(x_{min}, \lambda) = 0$$

3.5 Convergence Rate

3.5.1 Gauß-Newton with the Lagrange Function

A STLS problem of the form:

$$\begin{aligned} &\min_x \ \tfrac{1}{2}\|f(x)\|^2 \\ &s.t.\ g(x) = 0 \end{aligned}$$

where $f(x) = x$, can be rewritten as:

$$\begin{aligned} &\min_x \ \tfrac{1}{2}\|x - c\|^2 \\ &s.t.\ g(x) = 0 \end{aligned}$$

Here c is the center point around which we try to find a minimum. In the Gauß-Newton method $c = 0$ and consequently $f(x) = x - 0 = x$.

The Gauß-Newton method approximates the Hessian $H = J^T J + \sum f_i f_i''$ with $H = J^T J$ and therefore its convergence rate is fast when the residual $r = \sum_{i=1}^{m} f_i f_i''$ is small, that is; when we are close to x_{min}.

In [1], **Theorem** 3.1 for the convergence rate can be found and it says the following:

$$x_{k+1} - x_{min} = K_{matGN}(x_k - x_{min}) + J^+ err_r$$

where $x_{k+1} - x_{min}$ is the convergence rate, K_{matGN} is the convergence constant matrix, x_{min} is a minimum for the minimization problem and err_r is the error caused by the disturbance r.

The equation for the matrix K_{matGN} is ([1]):

$$\begin{aligned} K_{matGN} &= (A^T A)^+ \sum \big(g_i'' \underbrace{g_i}_{=0 \ at \ x_{min}}\big) + P_N \sum \big(g_i'' A^{+T}(x_{min} - c)\big) \\ &= (A^T A)^+ \sum g_i'' g_i + P_N \sum g_i'' l_i \end{aligned}$$

where A is the Jacobian of $g(x)$, $P_N = N(A)$ and $l = A^{+T}(x_{min} - c)$.

The formula that calculates the convergence constant K_{GN} is:

$$K_{GN} = \max_{eig} K_{matGN}$$

which means that K_{GN} is the maximum eigen value of K_{matGN}.

The Gauß-Newton method converges in step $k+1$ if the following property is satisfied:

$$\|x_{k+1} - x_{min}\| \leq K_{GN}\|x_k - x_{min}\| + err_r$$

and thus the property that $K_{GN} \leq 1$ is a prerequisite.

The size of K_{GN} depends on the size of the elements of K_{matGN} since $K_{GN} = \max_{eig} K_{matGN}$. If K_{matGN} is composed of small elements then its eigenvalues are small and hence K_{GN} is small as well. By the same reason, K_{GN} is big if the elements of K_{matGN} are big.

K_{matGN} consists of two parts:

$$K_{matGN} = \underbrace{(A^T A)^+ \sum g_i'' g_i}_{Part\ 1} + \underbrace{P_N \sum g_i'' l_i}_{Part\ 2}$$

The first part is small when the norm $\|g\| \to 0$, that is; when we approach x_{min}. The second part is small when l_i is small. $l = A^{+T}(x_{min} - c)$ and hence l_i decreases when $(x_{min} - c) \to 0$. Since $c = 0$ in the Gauß-Newton method, the second part of K_{matGN} diminishes when $x \to 0$.

the Gauß-Newton method (with the Lagrange function) does not always converge for minimization problems with nonlinear constraints because the second part of K_{matGN} can get very big even if the first part approaches zero (which leads to $K_{GN} > 1$) . In the case above, for example a STLS problem, the convergence of the Gauß-Newton method depends only on the initial search direction and thus on the start point x_{start}.

3.5.2 Matlab 7's Built-in Function `fmincon`

This method uses a Quasi-Newton[1] method to approximate the Hessian and thus its convergence rate is super linear.

3.6 Global Convergence

3.6.1 Gauß-Newton with Constraints

To obtain global convergence a *step length* α is introduced. This step length is such that $x_{k+1} = x_k + \alpha p$. Thus, α scales the direction p.

To calculate α a merit function is used. The definition of the merit function $\phi(\alpha)$ is:

$$\phi(\alpha) = \frac{1}{2}\|f(x+\alpha p)\|^2 + \nu\frac{1}{2}\|g(x+\alpha p)\|^2$$

ν is a "punishment-scalar" in the sense that when $\|g(x+\alpha p)\|^2$ gets small, ν gets big and vice versa. The minimum value that ν can get is 1 as the part of $\frac{1}{2}\|g(x+\alpha p)\|^2$ must contribute to $\phi(\alpha)$ with at least its proper value.

From the definition of $\phi(\alpha)$, the definition of ν can be deduced:

$$\nu = max\left(\frac{\frac{|g(x)^T\lambda|}{\delta} - \|Jp\|^2}{\|g(x)\|^2}, 1\right)$$

for some appropriate value for δ. Empirically it has been discovered that $\delta = 0.2$ is a good value [2].

When ν is chosen according to the criterion above, α_{start} is found in the interval $1 - \delta \leq \alpha_{start} \leq 1 + \delta$. α is such that:

$$\phi(\alpha) < \phi(1)$$

Originally $x_{k+1} = x_k + p$, where the scaling factor $\alpha = 1$ and sequentially p is trusted to be a "good" direction. α is used only in the purpose to improve the value of f and g when p is not a perfect direction on its own. $\alpha = 1$ (that is, no scaling of p at all) is always achievable and consequently $\phi(1)$ is the maximum value of $\phi(\alpha)$ that is accepted.

[1] Quasi-Newton is a method which approximates the Hessian H with $H = B$ where B is calculated with a secant equation

This leads to comparing $\phi(\alpha_0), \phi(\alpha_1), ..., \phi(\alpha_n)$ to $\phi(1)$, where $0 < \alpha_i \leq \alpha_{start}$ and

$$\begin{aligned}
\alpha_0 &= \alpha_{start} \\
\alpha_1 &= \frac{\alpha_0}{2} \\
&\vdots \\
\alpha_{n-1} &= \frac{\alpha_{n-2}}{2} \\
\alpha_n &= \frac{\alpha_{n-1}}{2} \\
&= \frac{\alpha_{n-2}}{2^2} \\
&\vdots \\
&= \frac{\alpha_0}{2^n}
\end{aligned}$$

and n is such that the property $\alpha_n > \epsilon$ for some small $\epsilon > 0$, is fulfilled.

Thus, α_i must fulfill these two criteria:

$$\begin{aligned}
0 < \alpha_i \leq \alpha_{start} \qquad (1) \\
\phi(\alpha_i) < \phi(1) \qquad (2)
\end{aligned}$$

Consequently, after picking the required α_i the new point x_{k+1} is:

$$x_{k+1} = x_k + \alpha_i p$$

3.6.2 The Tikhonov Method

To obtain global convergence for a nonlinear LS problem with constraints, a *step length* α is utilized. This step length is such that $x_{k+1} = x_k + \alpha p$. Thus, α scales the direction p.

To calculate α a merit function is used. The merit function $\phi(\alpha)$ is the very function that is to be minimized; that is:

$$\phi(\alpha) = \frac{1}{2} \left\| g(x + \alpha p) \right\|^2 + \mu^2 \left\| f(x + \alpha p) \right\|^2$$

where μ is the punishment-scalar used in the Tikhonov method (see section 3.3 on page 6).

Since we are dealing with minimization problems, $\phi(\alpha)$ is to be as small as possible. Originally $x_{k+1} = x_k + p$, where the scaling factor $\alpha = 1$ and sequentially p is trusted to be a "good" direction. α is used only in the purpose to improve the value of f and g when p in itself is not a perfect direction. $\alpha = 1$ (that is, no scaling of p at all) is always achievable and consequently $\phi(1)$ is the maximum value of $\phi(\alpha)$ that is accepted.

The initial value of α; α_{start} is always 1.

This leads to comparing $\phi(\alpha_0), \phi(\alpha_1), ..., \phi(\alpha_n)$ to $\phi(1)$, where $0 < \alpha_i \leq \alpha_{start}$ and

$$\begin{aligned}
\alpha_0 &= \alpha_{start} \\
\alpha_1 &= \frac{\alpha_0}{2} \\
&\vdots \\
\alpha_{n-1} &= \frac{\alpha_{n-2}}{2} \\
\alpha_n &= \frac{\alpha_{n-1}}{2} \\
&= \frac{\alpha_{n-2}}{2^2} \\
&\vdots \\
&= \frac{\alpha_0}{2^n}
\end{aligned}$$

and n is such that the property $\alpha_n > \epsilon$ for some small $\epsilon > 0$, is fulfilled.

Thus, α_i must fulfill these two criteria:

$$\begin{aligned}
0 < \alpha_i \leq \alpha_{start} \quad (1) \\
\phi(\alpha_i) < \phi(1) \quad (2)
\end{aligned}$$

Sequentially, after picking α_i the new point x_{k+1} is:

$$x_{k+1} = x_k + \alpha_i p$$

3.7 The Karusch-Kuhn-Tucker; KKT Conditions

The KKT conditions are applied to least squares problems which use the Lagrange function to find a solution.

A LS problem of the form:

$$\begin{aligned}
&\min_x \ \tfrac{1}{2} \|f(x)\|^2 \\
&s.t.\ g(x) = 0
\end{aligned}$$

is equivalent to:

$$\begin{aligned}
&\min_p \ \tfrac{1}{2} \|f(x+p)\|^2 \\
&s.t.\ g(x+p) = 0
\end{aligned}$$

which can be approximated by (see section 3.1 on page 5 for more details concerning Newton's method):

$$\begin{aligned}
&\min_p \ \tfrac{1}{2} \|f(x) + Jp\|^2 \\
&s.t.\ g(x) + Ap = 0
\end{aligned}$$

The definition of the Lagrange function is the following:

$$\mathfrak{L}(p, \lambda) = \frac{1}{2} \|f(x) + Jp\|^2 - \lambda^T (g(x) + Ap)$$

Whereas the gradient of the Lagrange function is:

$$\nabla \mathfrak{L}_p(p, \lambda) \quad = \quad J^T (f(x) + Jp) - A^T \lambda$$

where $r = -f(x) - Jp$ is the residual.

When $p = p_{min}$ is a minimum direction then the following property holds:

$$\nabla \mathfrak{L}_p(p_{min}, \lambda) = 0$$

The combination of the properties of the Lagrange function for a minimum direction $p = p_{min}$ deduces the KKT conditions:

$$\begin{aligned} Ap &= -g(x) \\ Jp + r &= -f(x) \\ J^T r + A^T \lambda &= 0 \end{aligned}$$

which can be written in matrix form as:

$$\begin{bmatrix} 0 & 0 & A \\ 0 & I & J \\ A^T & J^T & 0 \end{bmatrix} \begin{bmatrix} -g(x) \\ -f(x) \\ 0 \end{bmatrix} = \begin{bmatrix} \lambda \\ r \\ p \end{bmatrix}$$

3.8 The Toeplitz Matrix

A Toeplitz matrix, named after *Otto Toeplitz*, is a special kind of square matrix where each descending diagonal from left to right is constant.

Thus, a square matrix T of size $n \times n$, where

$$T = \begin{bmatrix} t_{1,1} & t_{1,2} & \dots & t_{1,n} \\ t_{2,1} & t_{2,2} & \dots & t_{2,n} \\ \vdots & \vdots & & \vdots \\ t_{n,1} & t_{n,2} & \dots & t_{n,n} \end{bmatrix}$$

is a Toeplitz matrix if and only if the elements of T have the property:

$$t_{i,j} = t_{i+1,j+1}$$

Or equivalently; T of size $n \times n$ is a Toeplitz matrix if and only if it has the form:

$$T = \begin{bmatrix} t_{1,1} & t_{1,2} & \dots & t_{1,n-1} & t_{1,n} \\ t_{2,1} & t_{1,1} & \dots & t_{1,n-2} & t_{1,n-1} \\ t_{3,1} & t_{2,1} & \dots & t_{1,n-3} & t_{1,n-2} \\ \vdots & \vdots & & \vdots & \vdots \\ t_{n,1} & t_{n-1,1} & \dots & t_{2,1} & t_{1,1} \end{bmatrix}$$

An example of a Toeplitz matrix A:

$$A = \begin{bmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ j & h & g & f & a \end{bmatrix}$$

3.9 The Hankel Matrix

A Hankel matrix, named after *Hermann Hankel*, is a square matrix with constant (positive sloping) skew-diagonals.

Thus, a square matrix H of size $n \times n$, where

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,n} \\ h_{2,1} & h_{2,2} & \dots & h_{2,n} \\ \vdots & \vdots & & \vdots \\ h_{n,1} & h_{n,2} & \dots & h_{n,n} \end{bmatrix}$$

is a Hankel matrix if and only if the elements of H have the property:

$$h_{i,j} = h_{i-1,j+1}$$

Or equivalently; H of size $n \times n$ is a Hankel matrix if and only if it has the form:

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,n} \\ h_{1,2} & h_{2,2} & \dots & h_{2,n} \\ \vdots & \vdots & & \vdots \\ h_{1,n} & h_{2,n} & \dots & h_{n,n} \end{bmatrix}$$

An example of a Hankel matrix A:

$$A = \begin{bmatrix} a & b & c & d & e \\ b & c & d & e & f \\ c & d & e & f & g \\ d & e & f & g & h \\ e & f & g & h & i \end{bmatrix}$$

The Hankel matrix is closely related to the Toeplitz matrix (a Hankel matrix is an upside-down Toeplitz matrix).

Chapter 4

Research Area

4.1 Present Algorithms and their Major Weakness

The methods to solve a STLS optimization problem of today cannot guarantee that a solution will always be found. Two examples of algorithms solving STLS problems are: the STLN (Structured Total Least Norm) algorithm [4] and the SNTLN (Structured Nonlinear Total Least Norm) algorithm [5].

Most of the existing methods demand the use of the second derivative; the Hessian. The calculations needed to compute the Hessian are numerous and terribly expensive when the STLS problem to solve is big. Even when the Hessian is actually calculated it only guarantees the approach of an extreme point, not a minimum. To circumvent the problem where the convergence is towards a maxima or terrace point, several methods (all quite costly) do exist. Even the methods that not use the Hessian, for example the Tikhonov method; either have other procedures which are expensive and require complicated reformulations of the original problem or they cannot guarantee that a minimum will indeed be found. Hence, they are too inefficient/unreliable as well.

The tasks of today become larger by the minute and to be able to reliably solve these tasks in a time which is "acceptable", some revolutionary algorithm has to be invented. In other words, we conclude that a method which is both reliable and fast does not exist but is highly desired.

4.1.1 Gauß-Newton with the Lagrange Function

This is an example of a method which solves general nonlinear least squares problems. The method is very fast and easy to implement but does not guarantee that an extreme point will be found. If an extreme point is found, it is not guaranteed that it is a minimum.

4.1.2 The Tikhonov Method

This method solves NLS problems with or without constraints. If there are constraints, the problem is reformulated such that the constraints need not be considered. At present, there is no optimal algorithm for choosing the punishment-scalar in an optimal way and this makes the method slow in some situations. This method is easy to implement since

no particular course of action must be taken to include the constraints. The Tikhonov method cannot guarantee that an extreme point will be found.

4.2 Proposed Algorithm: Reject the Hessian and Introduce a Center Point

The ideal optimization algorithm is the one that always finds a minimum fast. To make an algorithm fast, one condition is that it manages without the Hessian. To make an algorithm reliable, it must be made general so that it works even for the most difficult problems. So far it has not been proved whether this can be done or not. The assumption is therefore that a method that finds a minimum without making use of the Hessian can be found and implemented.

This assumption leads to the search of a method that minimizes a STLS problem of the form:

$$\begin{aligned} &\min_{\alpha,x} \ \tfrac{1}{2} \left\| \begin{array}{c} \Delta b(\alpha, x) \\ D\alpha \end{array} \right\|^2 \\ &s.t. \ (A + \Delta A)x = b + \Delta b \end{aligned}$$

which can be rewritten as:

$$\begin{aligned} &\min_{\alpha,x} \ \tfrac{1}{2} \left\| f(x) \right\|^2 \\ &s.t. \ g(x) = 0 \end{aligned}$$

where $x = \left[\begin{array}{c} \Delta b(\alpha, x) \\ D\alpha \end{array} \right]$, $f(x) = x$, $g(x) = (A + \Delta A)x - (b + \Delta b)$, D is a weight matrix and the vector α represents the errors in ΔA.

Another view of the minimization problem $\min_{\alpha,x} \ \frac{1}{2} \left\| f(x) \right\|^2$ is the perspective in a more geometrical way. The function $\|f(x) = x\|$ is to be minimized, which will deduce a result where $\|x\| \rightarrow \epsilon$ for some $\epsilon \geq 0$ as small as possible. Geometrically this is equivalent to saying that we search the minimum x_{min} *around the point zero.* Therefore $f(x) = x$ can be rewritten as $f(x) = f_{center}(x)$ where $f_{center}(x) = x - c$ and c is the center point around which we search the minimum. In the original equation $c = 0$ and sequentially $f(x) = \left[\begin{array}{c} \Delta b(\alpha, x) \\ D\alpha \end{array} \right]$ is the function that is minimized.

The assumption is that this center point c can be changed during the procedure of finding a minimum without changing the object function $f(x) = x - c$ too much for the minimum x_{min} to be completely wrong. This should be the right assumption since $f(x)$ is a linear function. The general idea behind this method is that once we have found an x such that $\|g(x)\| \approx 0$ (here a regular method may start taking the wrong direction, depending on the value of $\|f(x)\|$), changing c so that it is in the neighborhood of the x which is such that $\|g(x)\| \approx 0$ will rapidly lead to finding a minimum x_{min} such that $\|g(x_{min})\| = 0$.

This new method is a plain Gauß-Newton method with the Lagrange function, which is fast and does not use the Hessian and the addition of the center point, which will make it reliable.

The method will be referred to as *The center point method* from now on.

Chapter 5

Implementations of the Center Point Method

Two methods have been implemented; Gauß-Newton with the Lagrange function and the center point method. Matlab's built-in function `fmincon`[1] that minimizes linear functions with nonlinear constraints has been used as comparison and reference.

5.1 The Center Point Method

This method sees the object function $f(x) = x$ from another perspective, namely as $f(x) = f_{center}(x)$, where $f_{center}(x) = x - c$ and c is the center point. The goal is to solve a minimization problem efficiently and reliably without making use of the Hessian.

One huge obstacle with STLS problems is the case where we approach a minimum x_{min} (where $g(x_{min}) = 0$) and $\|f(x)\|$ gets too big before reaching x_{min} and consequently the search direction p changes and x_{min} will never be found. The idea of the center point method is to change the center point $c = c_{new}$ ($f(x) = x - c$) when we start diverging from x_{min} and thus search for x_{min} around the new center point c_{new}. The desired outcome of the change of the center point is to search for x_{min} (not considering the value of $\|f(x)\|$) more thoroughly in the area around c_{new} in which we know that $\|g(x)\| \approx 0$.

Two approaches of the center point method have been implemented.

5.1.1 The Center Point in Step k is $c_k = x_{k+1}$

This idea is basically to check the $\|f(x)\|$-values; $\|f(x_1)\|, \|f(x_2)\|, ..., \|f(x_k)\|$ continuously. If $\|f(x_1)\|, \|f(x_2)\|, ..., \|f(x_k)\|$ increase monotonically though p does not, this should imply that we're approaching a minimum x_{min} where $g(x_{min}) = 0$, since finding a point x_0 such that $g(x_0) = 0$ is a higher priority than that of x_0 minimizing $f(x_0)$.

In this case, we have to change c where $f(x) = x - c$. Changing $c = c_{new}$ means that we start looking for x_{min} around a new center point $c = c_{new}$.

c_k is chosen such that $c_k = x_{k+1} = x_k + \alpha_k p_k$. That is; choose c_k such that it will be a point one step closer in the direction p_k of the minimum x_{min}.

[1] Henceforth when Matlab is mentioned, it refers to Mathworks Matlab version 7

Then we have for $f_k(x_k)$:

$$\begin{aligned} f(x_k) &= x_k - c_k \\ &= x_k - x_{k+1} \\ &= x_k - (x_k + \alpha_k p_k) \\ &= -\alpha_k p_k \end{aligned}$$

which geometrically means that we search for x_{min} around the point x_{k+1}. In other words, around the x-point in step $k+1$.

5.1.2 The Center Point in Step k is $c_k = x_{k-1}$

Here the idea is basically to check the $\|g(x)\|$-values; $\|g(x_1)\|, \|g(x_2)\|, ..., \|g(x_k)\|$ continuously. Control if $\|g(x_k)\| > \|g(x_{k-1})\| > \|g(x_{k-2})\|$, which means that $\|g(x)\|$ has been increasing two iterations one after the other. It is not sufficient that $\|g(x_k)\| > \|g(x_{k-1})\|$, that is; $\|g(x)\|$ has increased compared to the last iteration only, because sometimes the value of $\|g(x)\|$ is zigzagging when $x \to x_{min}$.

Figure 5.1 shows an example of the zigzagging behavior that the norm $\|g(x)\|$ may exhibit. The norm of g is plotted as a function of the iterations. In this case, $m = 30$ and $n = 20$.

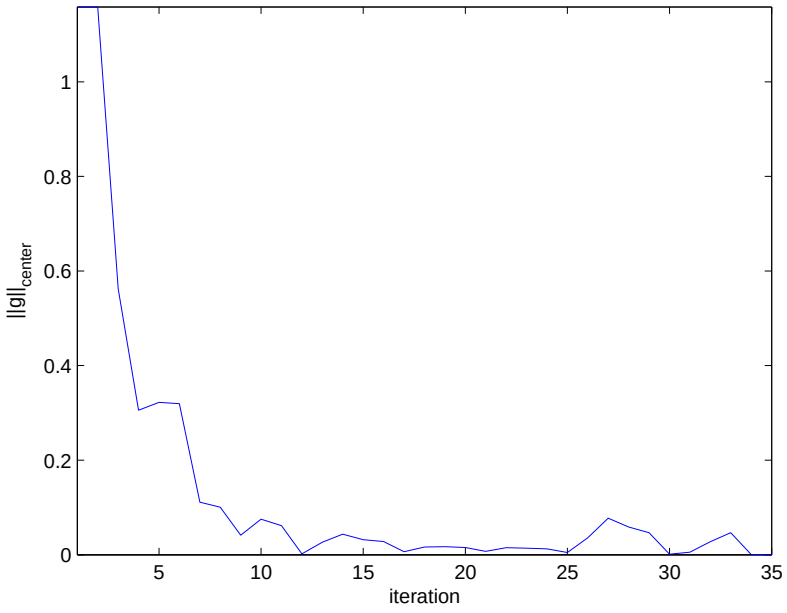

Figure 5.1: $\|g\|_{center\ point}$ in each iteration. $m = 30$ and $n = 20$

The value of $\|f(x)\|$ is irrelevant compared to the value of $\|g(x)\|$ when x is in the vicinity of x_{min}. This property exists since $\|f(x)\|$ *only* has to be minimized while it is

a constraint that $g(x) = 0$. Hence, $\|f(x)\|$ is not taken into consideration in the choice of the new center point.

In the case where $\|g(x_k)\| > \|g(x_{k-1})\| > \|g(x_{k-2})\|$, we have to change c where $f(x) = x - c$. Changing $c = c_{new}$ means that we start looking for x_{min} around a new center point $c = c_{new}$.

The new center point is picked such that $c_k = x_{k-1}$. That is; $c_k = x_{k-1}$ is chosen such that it is the previous point that we have passed such that $\|g(x_k)\| > \|g(x_{k-1})\| > \|g(x_{k-2})\|$ and $\|g(x_{k-2})\| < \|g(x_{k-3})\|$.

Then, $f(x_k)$ becomes:

$$\begin{aligned} f(x_k) &= x_k - c_k \\ &= x_k - x_{k-1} \\ &= x_{k-1} + \alpha_{k-1}p_{k-1} - x_{k-1} \\ &= \alpha_{k-1}p_{k-1} \end{aligned}$$

which geometrically means that we search x_{min} around x_{k-1}. Hence, we go back to the point in which we were one iterations ago.

5.1.3 The Center Point in Step k is $c_k = x_{k-2}$

The idea of this algorithm resembles a lot the idea of the center point $c = x_{k-1}$, that is; to check the $\|g(x)\|$-values; $\|g(x_1)\|, \|g(x_2)\|, ..., \|g(x_k)\|$ continuously. Control if $\|g(x_k)\| > \|g(x_{k-1})\| > \|g(x_{k-2})\|$, which means that $\|g(x)\|$ has been increasing two iterations one after the other. It is not sufficient that $\|g(x_k)\| > \|g(x_{k-1})\|$, that is; $\|g(x)\|$ has increased compared only to the previous iteration, because sometimes the value of $\|g(x)\|$ oscillates when we approach x_{min}.

The value of $\|f(x)\|$ is not important compared to the value of $\|g(x)\|$ when x is close to x_{min}. The minimization problem has this property since $\|f(x)\|$ being small is a *desire* while $g(x) = 0$ is a constraint. Therefore, no attention is paid to the value of $\|f(x)\|$ in the choice of the new center point.

In the case where $\|g(x_k)\| > \|g(x_{k-1})\| > \|g(x_{k-2})\|$, we have to change c where $f(x) = x - c$. Changing $c = c_{new}$ means that we start looking for x_{min} around a new center point $c = c_{new}$.

The new center point is picked such that $c_k = x_{k-2}$. That is; $c_k = x_{k-2}$ is chosen such that it is the *last* point that we have passed for which $g(x)$ was still decreasing, that is; for which $\|g(x_{k-2})\| < \|g(x_{k-3})\|$. In other words, $c_k = x_{k-2}$ is the last good point that we have found. Since $\|g(x)\|$ was decreasing up until $x = x_{k-2}$, a minimum x_{min} is likely to be found in the neighborhood of x_{k-2} and thus $c_k = x_{k-2}$ is a good choice of center point.

Then, $f(x_k)$ becomes:

$$\begin{aligned} f(x_k) &= x_k - c_k \\ &= x_k - x_{k-2} \\ &= x_{k-1} + \alpha_{k-1}p_{k-1} - x_{k-2} \\ &= x_{k-2} + \alpha_{k-2}p_{k-2} + \alpha_{k-1}p_{k-1} - x_{k-2} \\ &= \alpha_{k-2}p_{k-2} + \alpha_{k-1}p_{k-1} \end{aligned}$$

which geometrically means that we search x_{min} around x_{k-2}. Hence, we go back to the point in which we were two iterations ago. Two iterations ago we were still in a good point, in the sense that the point x_{k-2} has the property $\|g(x_{k-2})\| < \|g(x_{k-3})\|$.

5.1.4 Comments

The first algorithm in which $c_k = x_k + \alpha_k p_k$ did not work.
The values of $\|f(x_1)\|, \|f(x_2)\|, ..., \|f(x_k)\|$ have proven themselves irrelevant regarding the convergence. The point $x_{k+1} = x_k + \alpha_k p_k$ is not a good choice as a new center point, because it is already too far away in the wrong direction with respect to x_{min}.

The second algorithm works better than the first one, but it is still too unreliable. The new center point $c = x_{k-1}$ is a good choice in most cases, but in a situation where the step between x_{k-2} and x_{k-1} is very big (when $\|\alpha_{k-2} p_{k-2}\|$ is big), the point x_{k-1} has already taken a turn which leads away from x_{min}.

The third algorithm in which $c_k = x_{k-2}$ is functioning well.
The new center point $c_k = x_{k-2}$ is chosen such that it is the *last* point for which $\|g(x)\|$ was still decreasing. This change of c leads to a fast and reliable convergence since the convergence constant K_{GN} gets very small. (See section 3.5.1 on page 8 for a more thorough description of the convergence rate for the Gauß-Newton method).

5.2 The Tikhonov Method

The Tikhonov method with the addition of the change of the center point has been implemented. The Tikhonov method with the addendum of the center point will be referred to as *the Tikhonov-center method* from now on.

This is done in the same way as in the center point method; that is, the object function $f(x) = x$ is rewritten as $f(x) = x - c$ where c is the center point. Hence, a nonlinear LS problem to solve becomes:

$$\min_x \ \tfrac{1}{2} \|g(x)\|^2 + \mu^2 \|x - c\|^2$$

where c is changed according to the center point method (see section 4.2 on page 16).

Only the functional version of the change of the center point when in Step k, $c_k = x_{k-2}$ (see section 5.1.3 on page 20 for more details) is used.

Hence, once the property $\|g(x_k)\| > \|g(x_{k-1})\| > \|g(x_{k-2})\|$ is fulfilled, the center point c_k is changed to $c_k = x_{k-2}$. In the Tikhonov-center method the punishment-scalar μ_k is set to $\mu_k = \mu_{start}$ when the center point is changed.

Chapter 6

Results

In this section follow the results from the implementations of the Gauß-Newton method with the Lagrange function, the center point method and the Tikhonov-center method. Matlab's `fmincon` is used as a reference. All the methods try to solve STLS problems of the form:

$$\begin{aligned} &\min_{x} \ \tfrac{1}{2} \left\| \ \Delta A, \Delta b \ \right\|_F \\ &s.t. \ (A + \Delta A)x = b + \Delta b \end{aligned}$$

where A has size $m \times n$, b has size $m \times 1$ and $m > n$.
Termination will occur once $\|g\| < \epsilon$ where ϵ is chosen to be 10^{-9}. The maximum number of iterations that each method has at its disposal is *MAXITER* and *MAXITER* is always 100.

The values of the matrices A and b are random and such that A is a Hankel matrix. A and b are set randomly each test run and all the methods use the same A, b and start point x_{start}.

6.1 Comparison of Gauß-Newton, Matlab's `fmincon`, the Tikhonov-center method and the Center Point Method

6.1.1 The Ratio between Convergence and Divergence

Results from ten test runs with the different methods. They show the ratio between **Convergence:Divergence** within the range of *MAXITER* iterations, respectively.

A has size $m \times n$ and b has size $m \times 1$.
The domain of m is $\{5, 10, 15, 20, \ldots, 100\}$ and that of n is from $\{1, 2, \ldots, 5, 10, 15, 20, \ldots, 50\}$ and A and b vary from one test run to another. A method converges within the permissible number of iterations if $\|g\| < \epsilon$ where $\epsilon = 10^{-9}$.

Table 6.1: Gauß-Newton with the Lagrange Function

m \ n	1	2	3	4	5	10	20	30	40	50
5	1:0	1:0.67	1:0.25	1:0.67						
10	1:0	1:0.25	1:0.25	1:0.25	1:0.11					
20	1:0	1:1.5	1:0	1:0.43	1:0.25	1:0.43				
30	1:0	1:1	1:0	1:0.67	1:0	1:1	1:0.25			
40	1:0	1:2:33	1:0	1:2.33	1:0.11	1:1.5	1:0.67	1:0.11		
50	1:0	1:1.5	1:0	1:0.43	1:1.25	1:1.5	1:0.43	1:0.11	1:0.25	
60	1:0	1:1.5	1:0.11	1:0.67	1:0.11	1:2.33	1:0.67	1:2.33	1:1.5	1:0.43
70	1:0	1:2.33	1:0.11	1:1	1:0.25	1:1.5	1:2.33	1:4	1:0.25	1:1.5
80	1:0	1:1.5	1:0	1:4	1:0.11	1:1.5	1:1	1:1.5	1:1	1:1.5
90	1:0	1:1.5	1:0	1:1	1:0	1:0.67	1:1	1:1	1:0.67	1:2.33
100	1:0	1:1.5	1:0	1:1	1:0.11	1:0.25	1:2.33	1:0.67	1:1	1:1

Table 6.2: Matlab's `fmincon`

m \ n	1	2	3	4	5	10	20	30	40	50
5	1:0	1:0	1:0	1:0						
10	1:0	1:0	1:0	1:0	1:0					
20	1:0	1:0	1:0.11	1:0	1:0.25	1:0.43				
30	1:0	1:0	1:0.11	1:0.11	1:1	1:2.33	1:4			
40	1:0	1:0	1:0.11	1:0.43	1:0.43	1:1	1:1.5	1:1		
50	1:0	1:0.11	1:0.11	1:0.25	1:1	1:9	1:2.33	1:2.33	1:9	
60	1:0	1:0.43	1:0.43	1:0.25	1:0.67	1:1.5	1:4	1:∞	1:9	1:∞
70	1:0	1:0.11	1:0.25	1:0.43	1:0.43	1:2.33	1:9	1:∞	1:∞	1:∞
80	1:0	1:0.25	1:0.43	1:0.67	1:0.43	1:4	1:9	1:9	1:∞	1:∞
90	1:0	1:0.25	1:0.25	1:1	1:1.5	1:2.33	1:∞	1:∞	1:∞	1:∞
100	1:0	1:0	1:0.43	1:0.43	1:0.67	1:4	1:9	1:0.67	1:∞	1:∞

Table 6.3: The Tikhonov-center Method

m \ n	1	2	3	4	5	10	20	30	40	50
5	1:0	1:0	1:0	1:0						
10	1:0	1:0	1:0	1:0	1:0					
20	1:0	1:0.11	1:0	1:0.11	1:0.11	1:0.11				
30	1:0	1:0	1:0	1:0	1:0.25	1:0.11	1:0			
40	1:0	1:0.11	1:0	1:0	1:0.11	1:0.11	1:0.25	1:0		
50	1:0	1:0	1:0.11	1:0	1:0.11	1:0	1:0	1:0.25	1:0.11	
60	1:0	1:0	1:0.11	1:0	1:0.11	1:0.11	1:0	1:0	1:0	1:0
70	1:0	1:0	1:0	1:0.11	1:0.11	1:0	1:0	1:0.43	1:0	1:0.11
80	1:0.11	1:0	1:0	1:0	1:0.11	1:0.25	1:0.11	1:0	1:0.11	1:0
90	1:0	1:0	1:0	1:0.11	1:0.25	1:0.11	1:0	1:0.11	1:0	1:0.43
100	1:0	1:0	1:0	1:0.25	1:0	1:0	1:0	1:0.11	1:0	1:0.25

Table 6.4: The Center Point Method

m \ n	1	2	3	4	5	10	20	30	40	50
5	1:0	1:0	1:0	1:0.11						
10	1:0	1:0	1:0	1:0	1:0					
20	1:0	1:0	1:0	1:0	1:0.11	1:0				
30	1:0	1:0	1:0	1:0	1:0	1:0	1:0			
40	1:0	1:0	1:0	1:0	1:0	1:0	1:0	1:0		
50	1:0	1:0	1:0	1:0	1:0	1:0	1:0.11	1:0	1:0	
60	1:0	1:0	1:0	1:0.11	1:0.11	1:0	1:0	1:0.11	1:0	1:0
70	1:0	1:0	1:0	1:0	1:0	1:0	1:0	1:0.11	1:0	1:0
80	1:0	1:0	1:0	1:0	1:0	1:0	1:0.11	1:0	1:0	1:0.11
90	1:0	1:0.11	1:0	1:0.11	1:0	1:0	1:0	1:0	1:0.11	1:0
100	1:0	1:0	1:0	1:0.11	1:0	1:0	1:0	1:0.11	1:0	1:0.11

6.1.2 Conclusions

The probability of convergence for the Gauß-Newton method appears to be random for $n > 1$. Whether it converges or diverges depends only on the start point x_{start}.
Matlab's `fmincon` has a reliable convergence when $n < 10$. If $m > 40$ and $n > 20$ the probability of convergence approaches zero.
The probability of convergence of the Tikhonov-center method is higher than both the regular Gauß-Newton method and Matlab's `fmincon`. Though it is less reliable than the center point method and overall not sufficiently reliable when n exceeds 3.
The probability of convergence of the center point method is stable and high in the whole interval of m and n.

6.2 Comparison of Gauß-Newton, Matlab's `fmincon` and the Center Point Method

6.2.1 The Size of the Norm $\|g\|$

This section contains graphs showing the norm $\|g\|$ once a method has found a minimum or reached *MAXITER*. To get the 10:th power of the norms; log_{10} for $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ are plotted in each graph. $log_{10}(\epsilon)$ where $\epsilon = 10^{-9}$ is such that a minimum is found once $\|g\| < \epsilon$, is plotted as a reference. That is, if a curve is above the dotted $log_{10}(\epsilon)$ line, then that method diverges in that interval of m.

n is constant each test run and its range and m is the variable. The initial value of m is dependent on n since $m > n$.

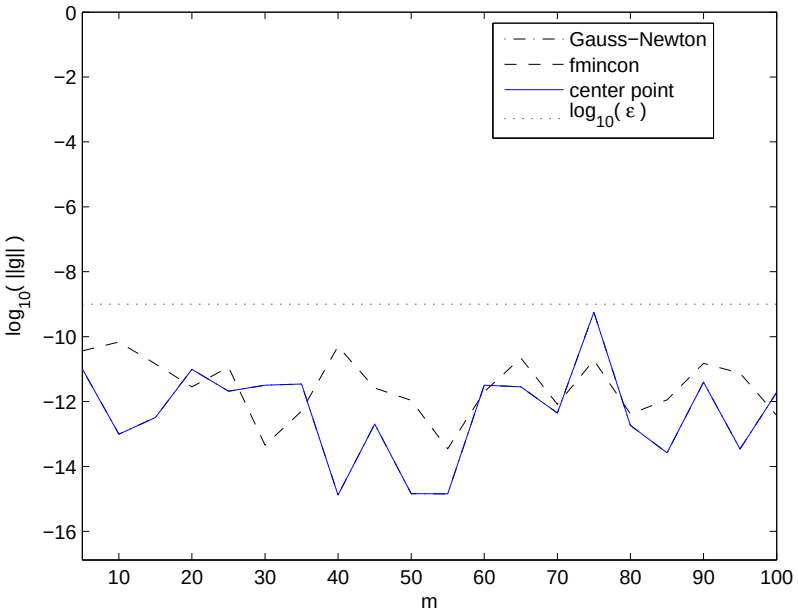

Figure 6.1: $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ for $n = 1$. The curve of Gauß-Newton has the exact same values as the curve of the center point method, therefore it cannot be seen

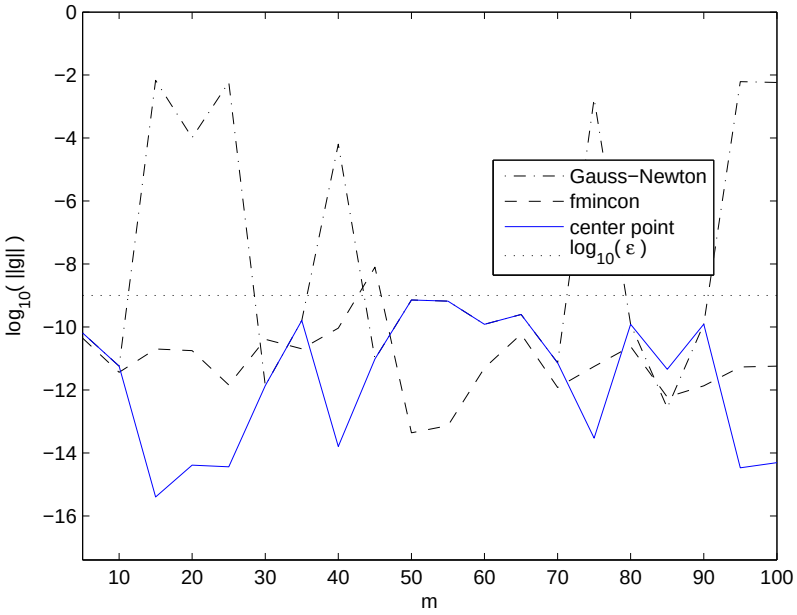

Figure 6.2: $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ for $n = 2$

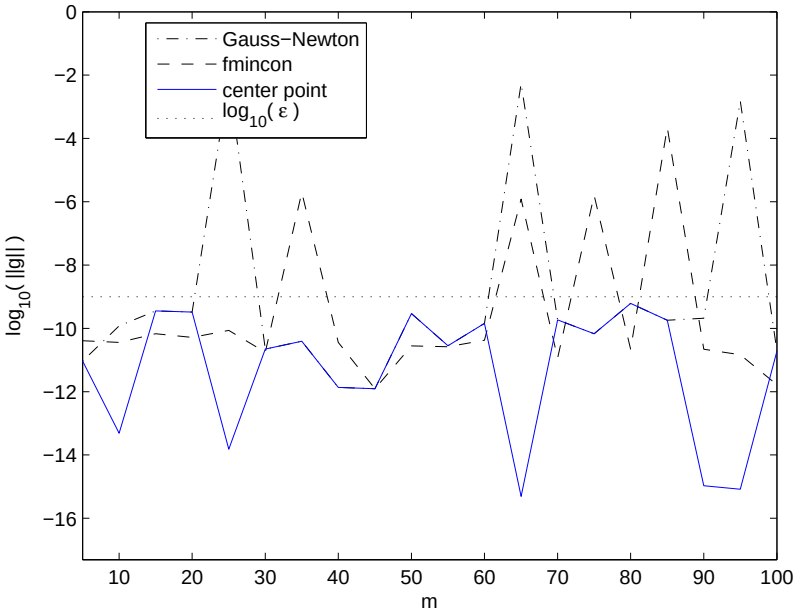

Figure 6.3: $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ for $n = 3$

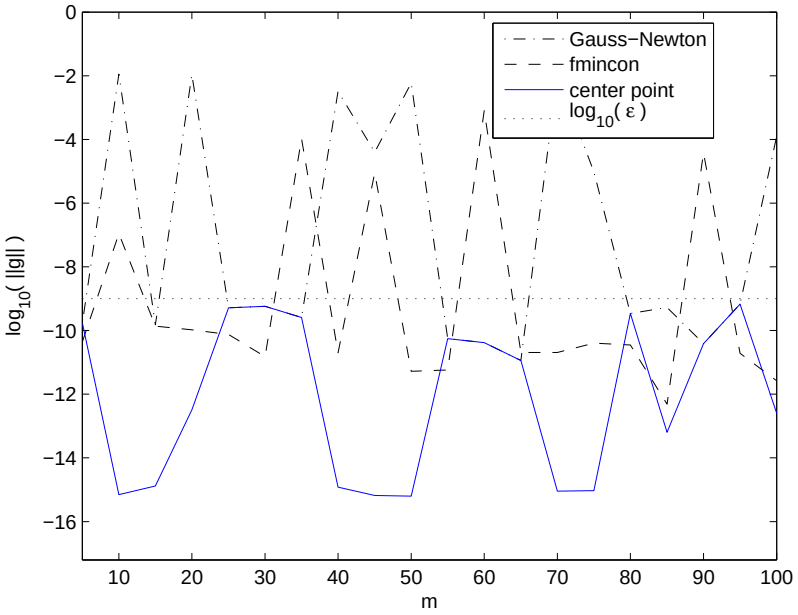

Figure 6.4: $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ for $n = 4$

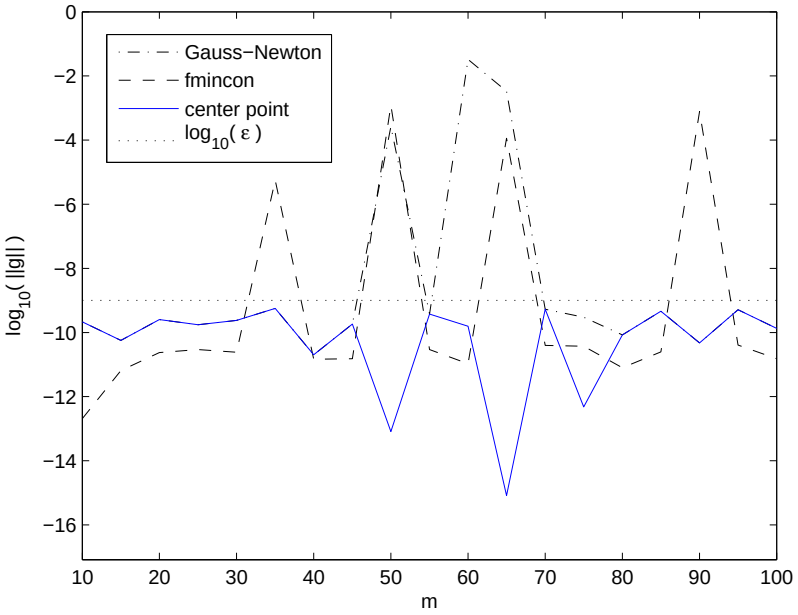

Figure 6.5: $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ for $n = 5$

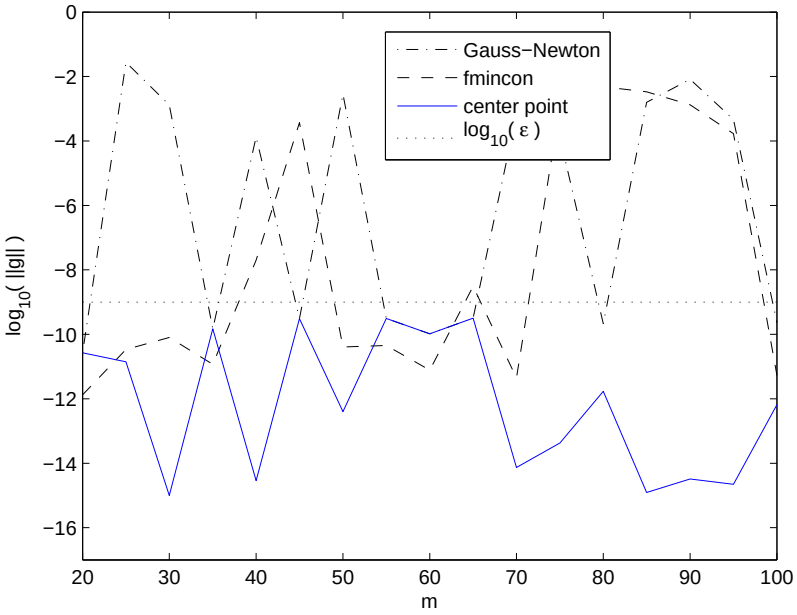

Figure 6.6: $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ for $n = 10$

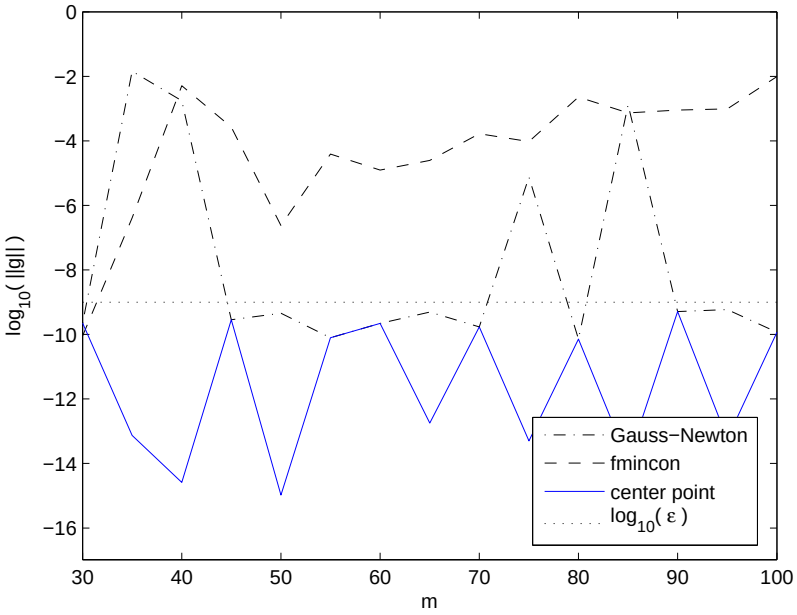

Figure 6.7: $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ for $n = 20$

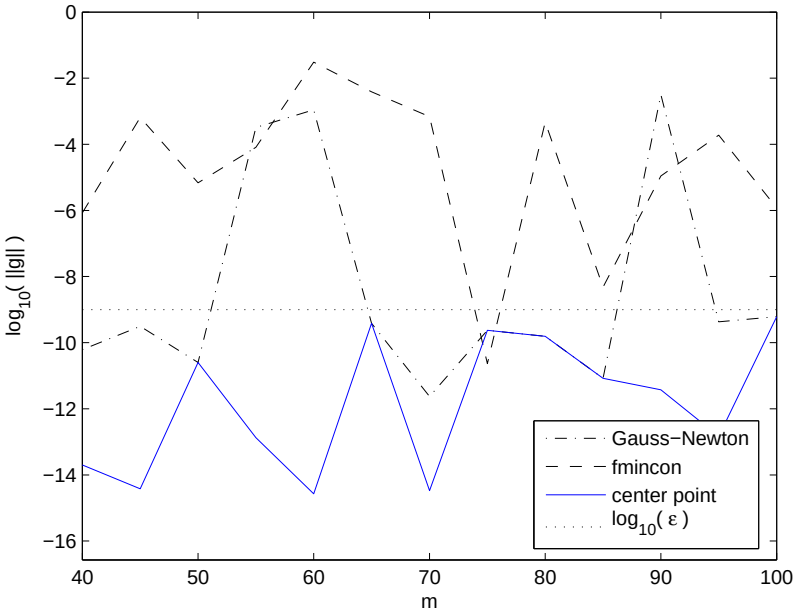

Figure 6.8: $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ for $n = 30$

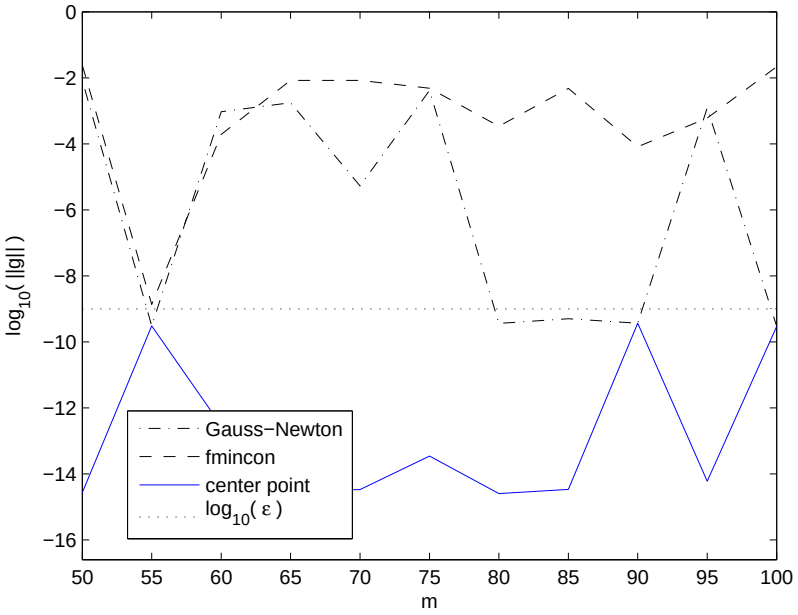

Figure 6.9: $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ for $n = 40$

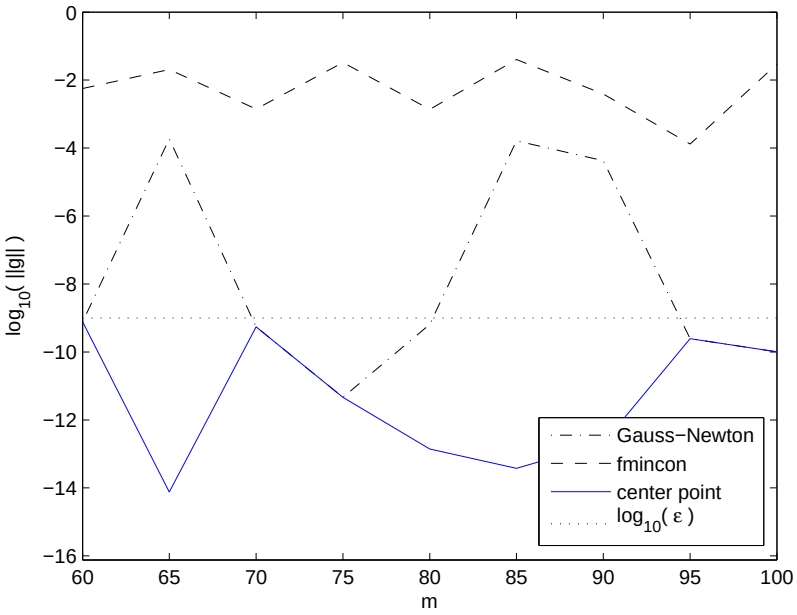

Figure 6.10: $\|g\|_{Gauss-Newton}$, $\|g\|_{\texttt{fmincon}}$ and $\|g\|_{center\ point}$ for $n = 50$

6.2.2 Conclusions

The value of the norm $\|g\|$ is in the interval 10^{-12} to 10^{-9} for the Gauß-Newton method and Matlab's `fmincon` when they converge. The norm $\|g\|$ of the center point method varies between 10^{-15} and 10^{-9} when a minimum is found. From the figures above it can be deduced that the center point method reduces the value of $\|g\|$ with in average two decimal places more in each step, compared to the other two methods.
The value of $\|g\|$ lies between 10^{-6} and 10^{-2} when a method diverges. This interval of $\|g\|$ does not depend on m and n and applies to all the three methods.

6.3 Comparison of Matlab's `fmincon` and the Center Point Method

This section contains comparisons of the results obtained by the different methods in trying to find a minimum. Comparisons of values such as the norm $\|\Delta A, \Delta b\|_F$, the CPU-time and the required number of iterations are done.

Only the values of m and n for which both `fmincon` and the center point method actually converge, are included in the graphs.

In each figure, n is constant and m is the variable. The range of n is $\{1, 2, \ldots, 5, 10\}$, with one value of n per figure. The range of m is $\{5, 10, 15, 20, \ldots, 100\}$, with its initial value depending on n ($m > n$).

6.3.1 The Size of the Norm $\|\Delta A, \Delta b\|_F$

Here follow graphs showing the size of the difference between $\|\Delta A, \Delta b\|_F$ at a minimum found by Matlab's `fmincon` and $\|\Delta A, \Delta b\|_F$ at a minimum found by the center point method. When the center point c is changed, the minimization problem becomes another and hence the localization of the minimum; $x_{min\texttt{fmincon}} \neq x_{min\,center\ point}$. The localization of the minimization point x_{min} does not actually matter for the STLS problem, what matters is the size of $\|\Delta A, \Delta b\|_F$ since $g(x) = (A + \Delta A)x - (b + \Delta b)$ must be zero.

Two different types of graphs are shown.

1. The first graph illustrates this difference by using the formula:

$$\Psi log_{10}(\|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point})$$

The formula reveals the decimal place that differs the two norms.

Ψ is the sign of $\|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point}$,
and its definition is:

$$\Psi = \begin{cases} 1, & \text{if } \|\Delta A, \Delta b\|_{\texttt{fmincon}} > \|\Delta A, \Delta b\|_{center\ point} \\ 0, & \text{if } \|\Delta A, \Delta b\|_{\texttt{fmincon}} = \|\Delta A, \Delta b\|_{center\ point} \\ -1, & \text{if } \|\Delta A, \Delta b\|_{\texttt{fmincon}} < \|\Delta A, \Delta b\|_{center\ point} \end{cases}$$

Example: If the value of $\|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point}$ differs in the 7th decimal when $m = 10$ and the value of $\|\Delta A, \Delta b\|_{\texttt{fmincon}} < \|\Delta A, \Delta b\|_{center\ point}$ then $\Psi = -1$ and consequently the value -7 will appear for $m = 10$.

2. The second graph illustrates this difference using the formula:

$$\left| \|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point} \right|$$

and the values on the x-axis are logarithmic to underline the difference in the 10th power. This figure only shows the absolute difference and hence it is not possible to see which norm is smaller.

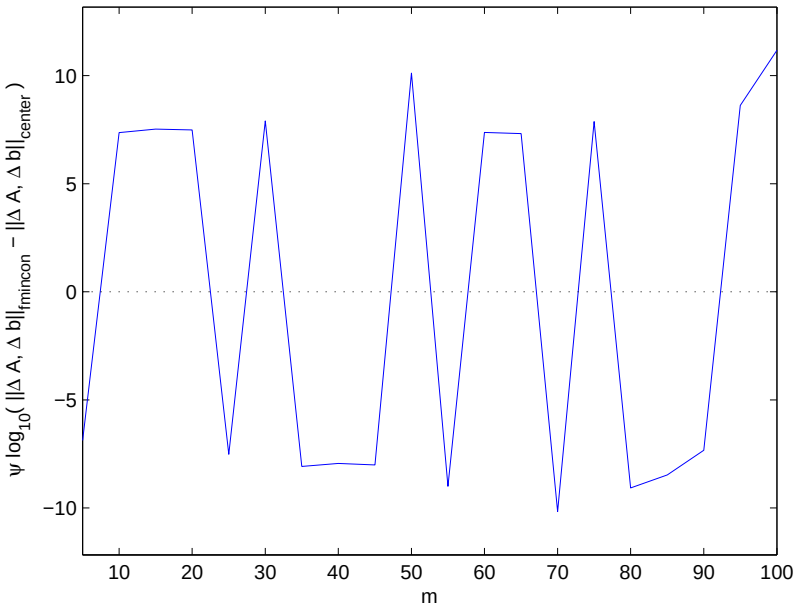

Figure 6.11: Decimal difference in $\|\Delta A, \Delta b\|_{\texttt{fmincon}}$ - $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 1$

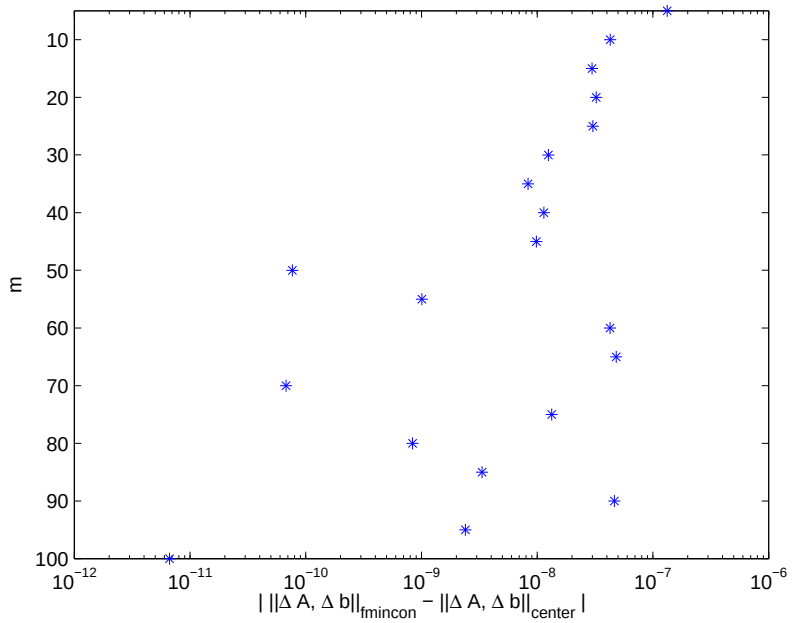

Figure 6.12: Difference in $\left| \|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point} \right|$ for $n = 1$

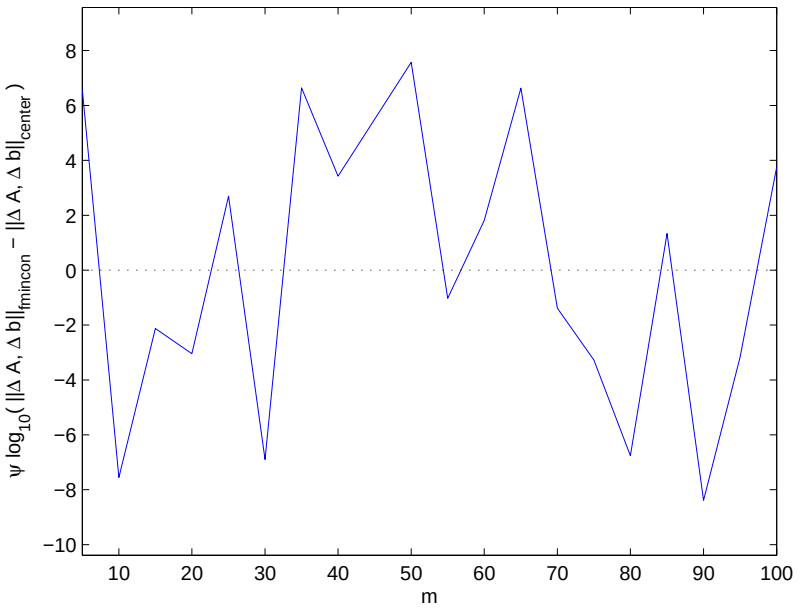

Figure 6.13: Decimal difference in $\|\Delta A, \Delta b\|_{\texttt{fmincon}}$ - $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 2$

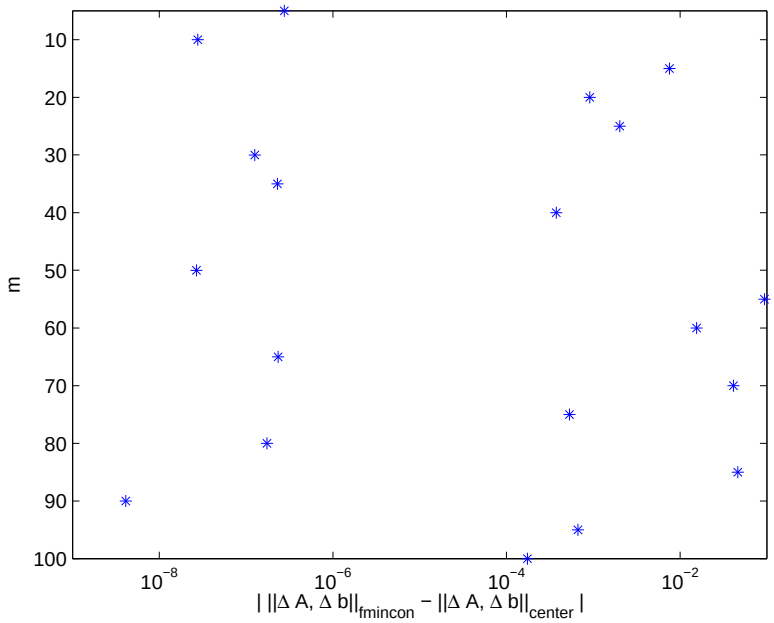

Figure 6.14: Difference in $\left|\|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point}\right|$ for $n = 2$

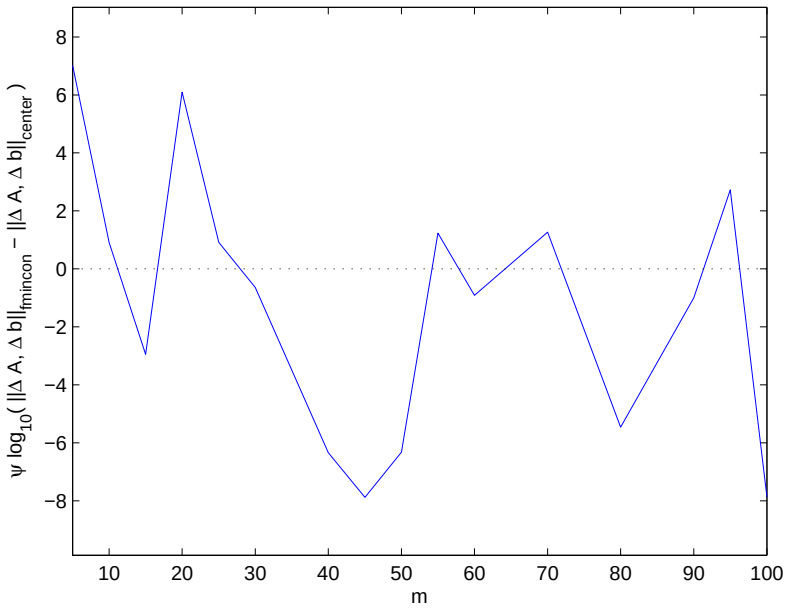

Figure 6.15: Decimal difference in $\|\Delta A, \Delta b\|_{\texttt{fmincon}}$ - $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 3$

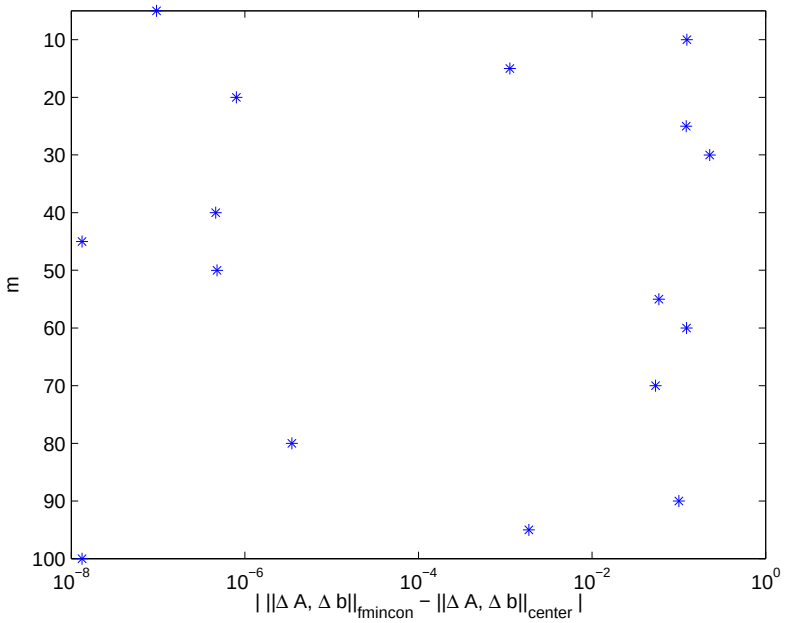

Figure 6.16: Difference in $\left| \|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point} \right|$ for $n = 3$

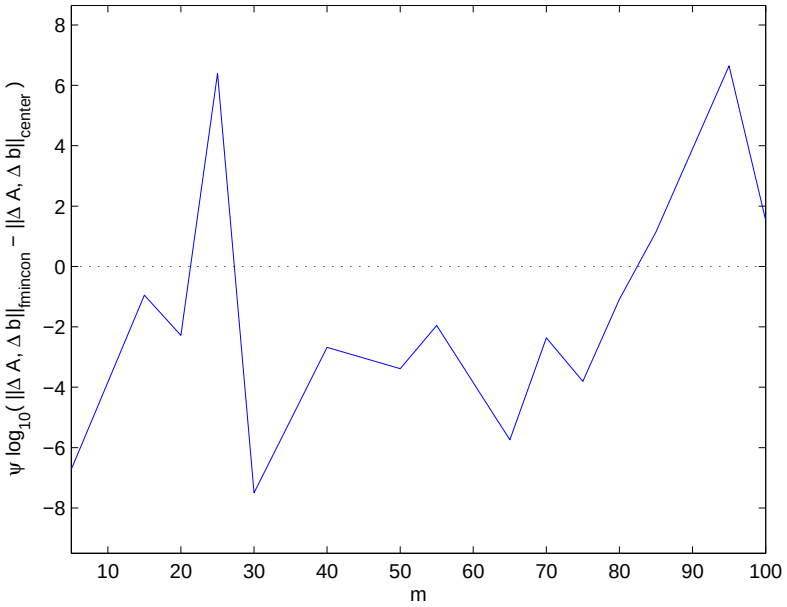

Figure 6.17: Decimal difference in $\|\Delta A, \Delta b\|_{\texttt{fmincon}}$ - $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 4$

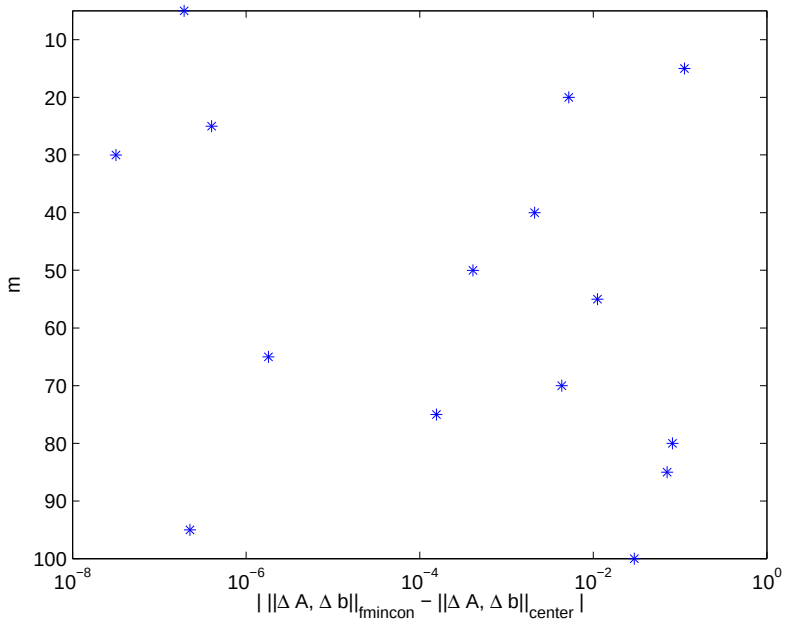

Figure 6.18: Difference in $\left| \|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point} \right|$ for $n = 4$

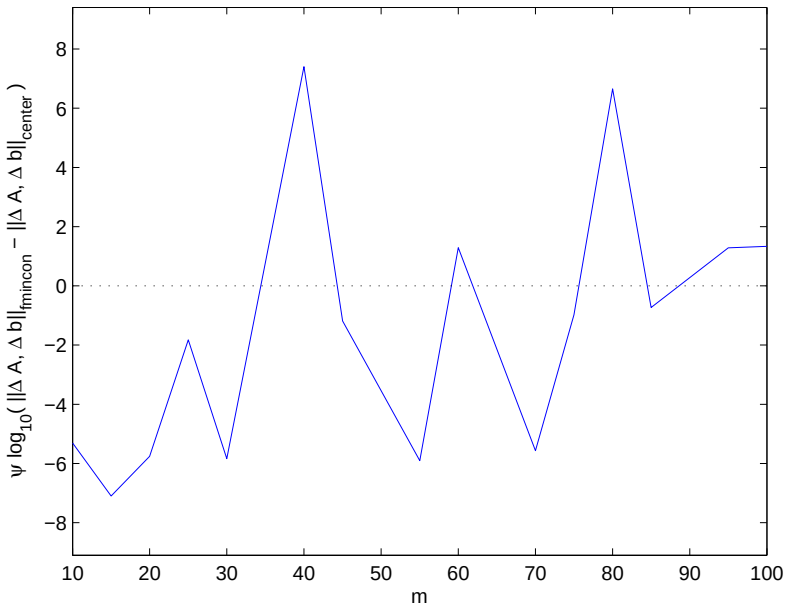

Figure 6.19: Decimal difference in $\|\Delta A, \Delta b\|_{\texttt{fmincon}}$ - $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 5$

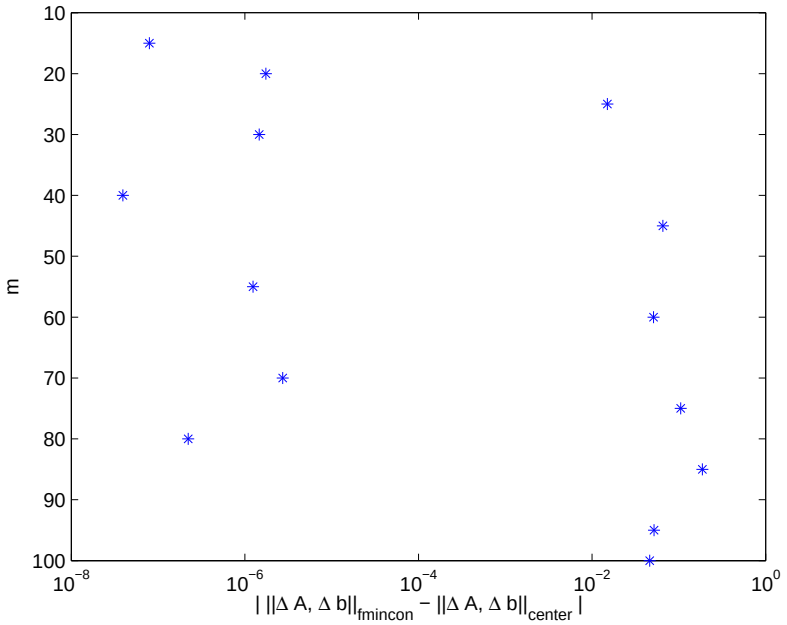

Figure 6.20: Difference in $\left| \|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point} \right|$ for $n = 5$

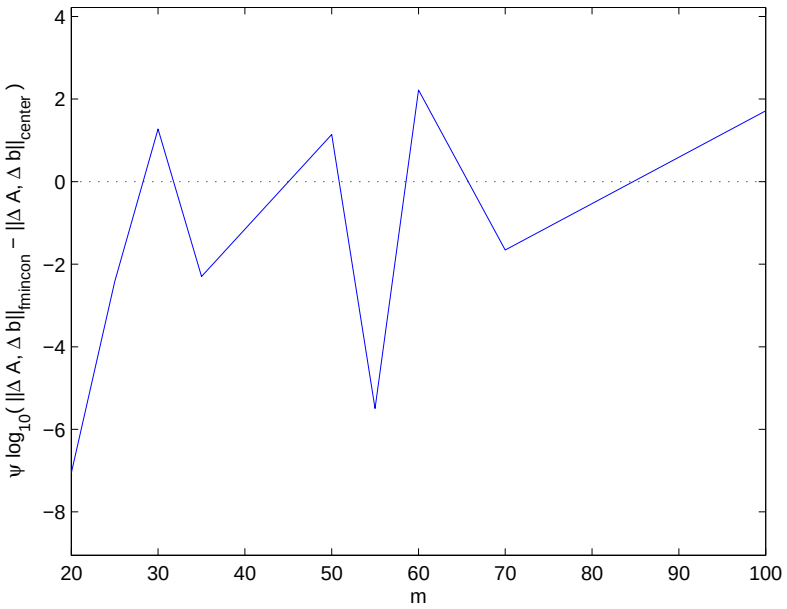

Figure 6.21: Decimal difference in $\|\Delta A, \Delta b\|_{\texttt{fmincon}}$ - $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 10$

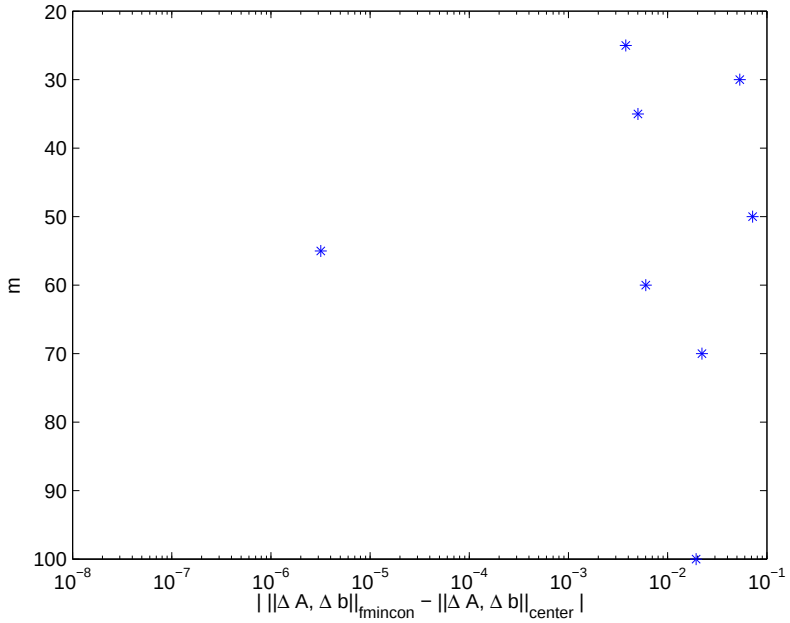

Figure 6.22: Difference in $\left| \|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point} \right|$ for $n = 10$

6.3.2 The CPU-time

This section contains graphs showing the CPU-time needed to find a minimum by Matlab's `fmincon` versus the CPU-time required by the center point method.

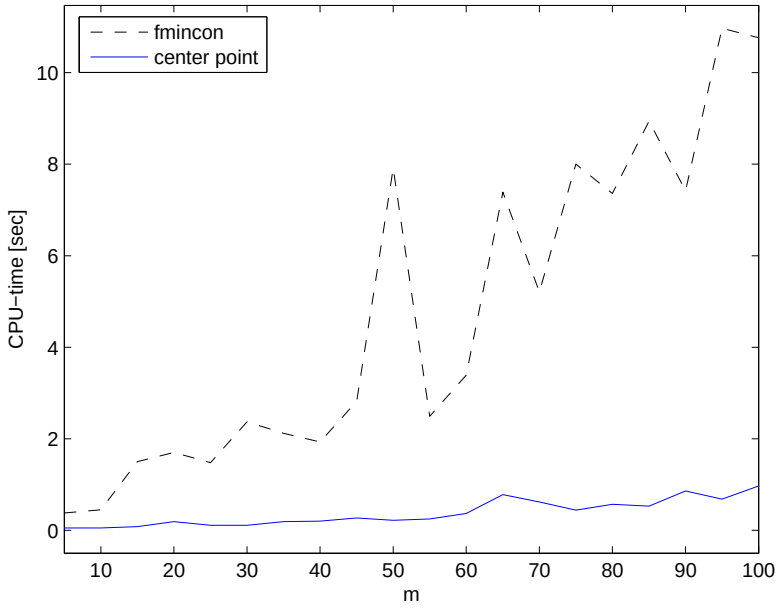

Figure 6.23: CPU-time$_{\texttt{fmincon}}$ and CPU-time$_{center\ point}$ for $n = 1$

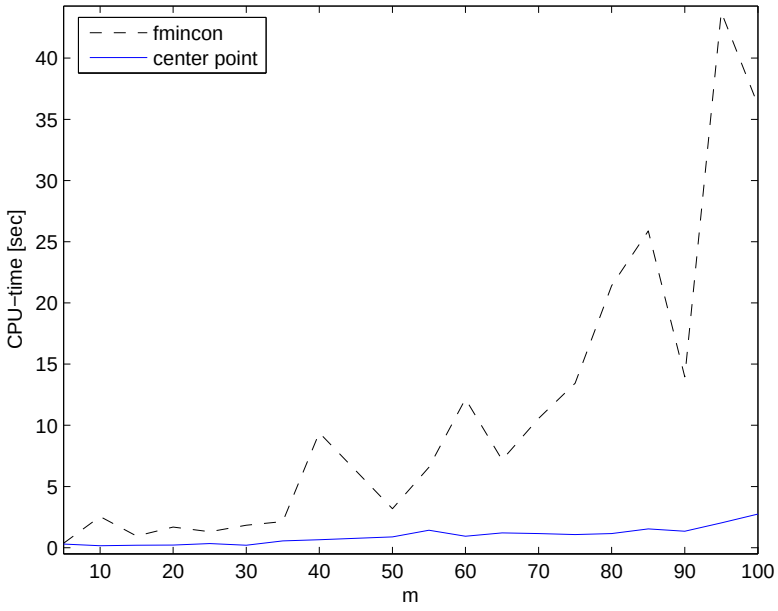

Figure 6.24: CPU-time$_{\texttt{fmincon}}$ and CPU-time$_{center\ point}$ for $n = 2$

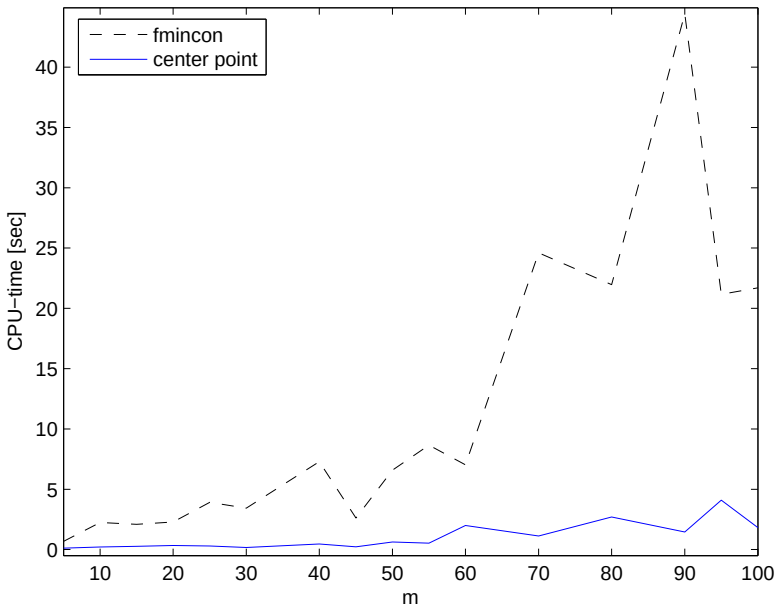

Figure 6.25: CPU-time$_{\texttt{fmincon}}$ and CPU-time$_{center\ point}$ for $n = 3$

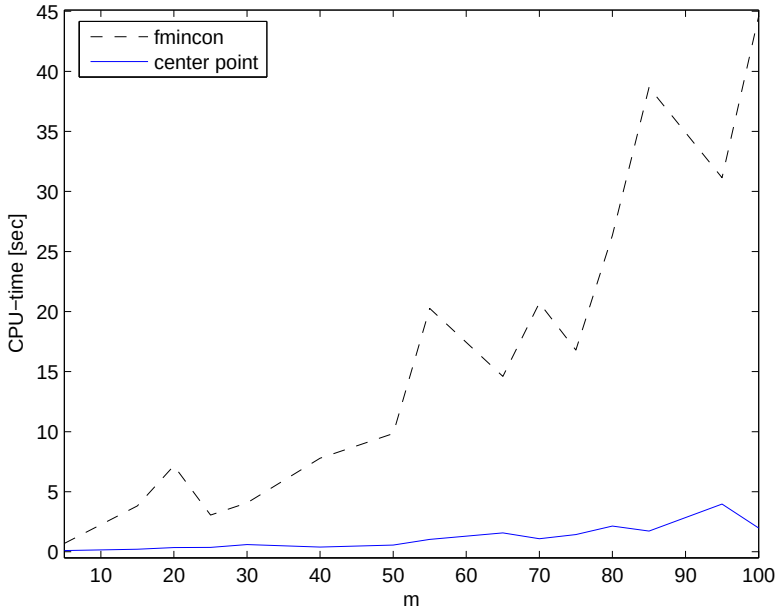

Figure 6.26: CPU-time$_{\texttt{fmincon}}$ and CPU-time$_{center\ point}$ for $n = 4$

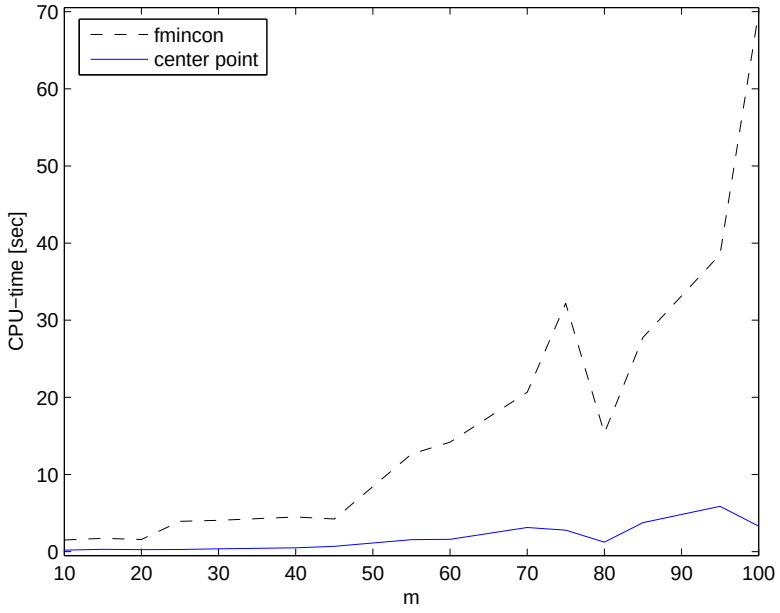

Figure 6.27: CPU-time$_{\texttt{fmincon}}$ and CPU-time$_{center\ point}$ for $n = 5$

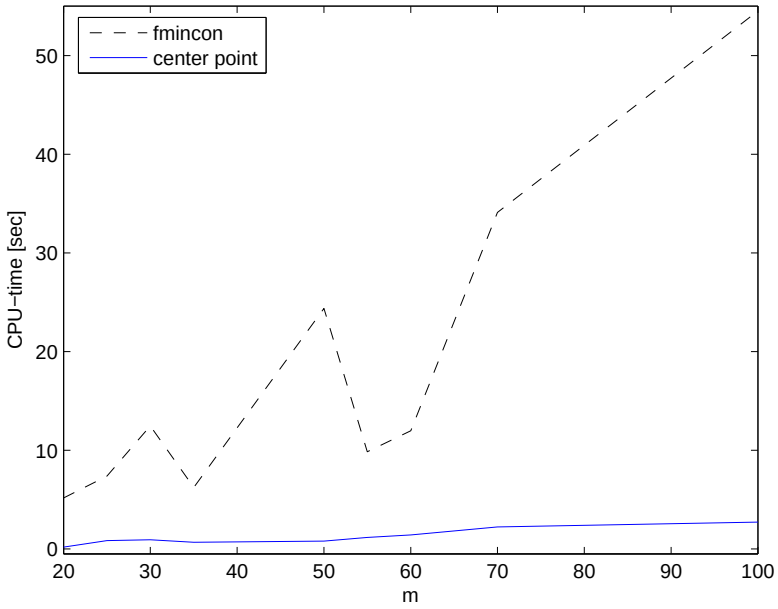

Figure 6.28: CPU-time$_{\texttt{fmincon}}$ and CPU-time$_{center\ point}$ for $n = 10$

6.3.3 The Required Number of Iterations

The number of iterations needed for Matlab's `fmincon` and for the center point method respectively are shown in the graphs below.

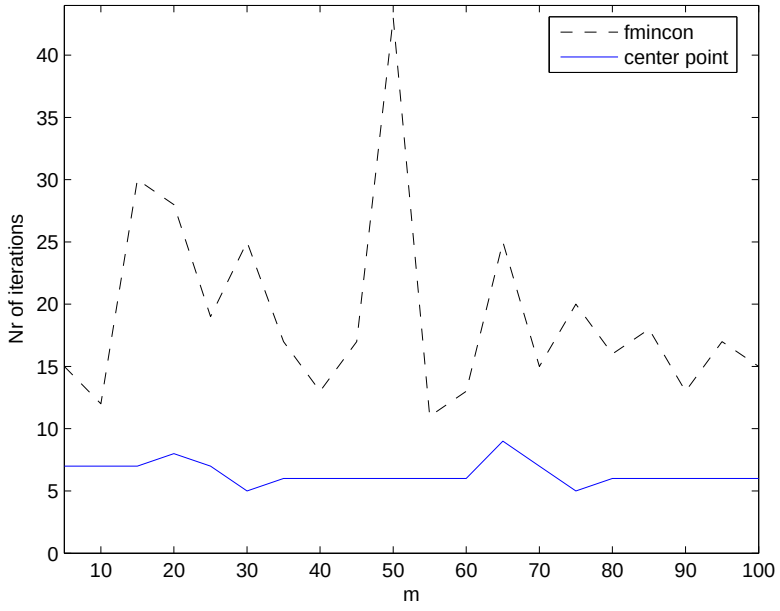

Figure 6.29: $\text{iter}_{\texttt{fmincon}}$ and $\text{iter}_{center\ point}$ for $n = 1$

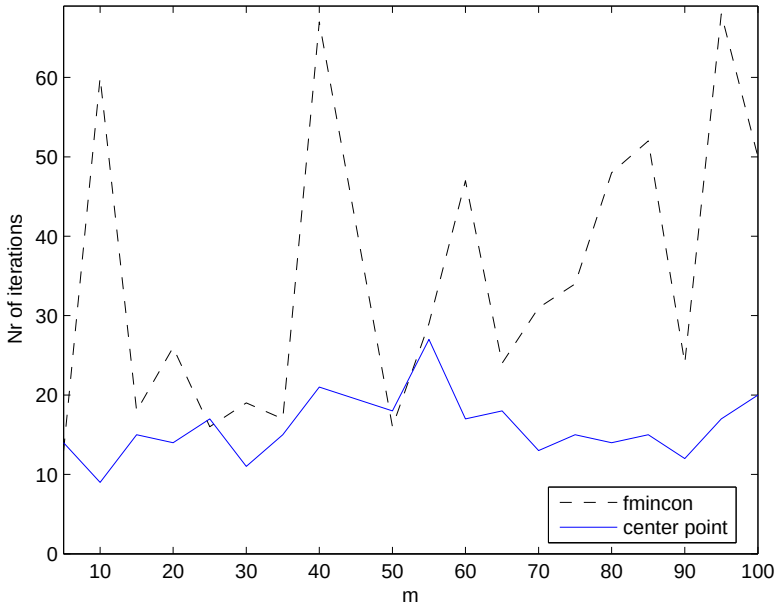

Figure 6.30: $\text{iter}_{\texttt{fmincon}}$ and $\text{iter}_{center\ point}$ for $n = 2$

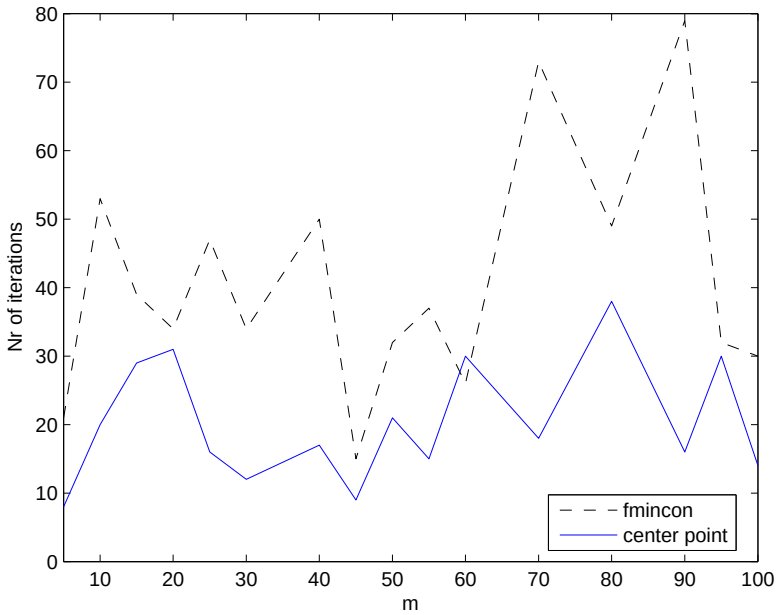

Figure 6.31: $\text{iter}_{\texttt{fmincon}}$ and $\text{iter}_{center\ point}$ for $n = 3$

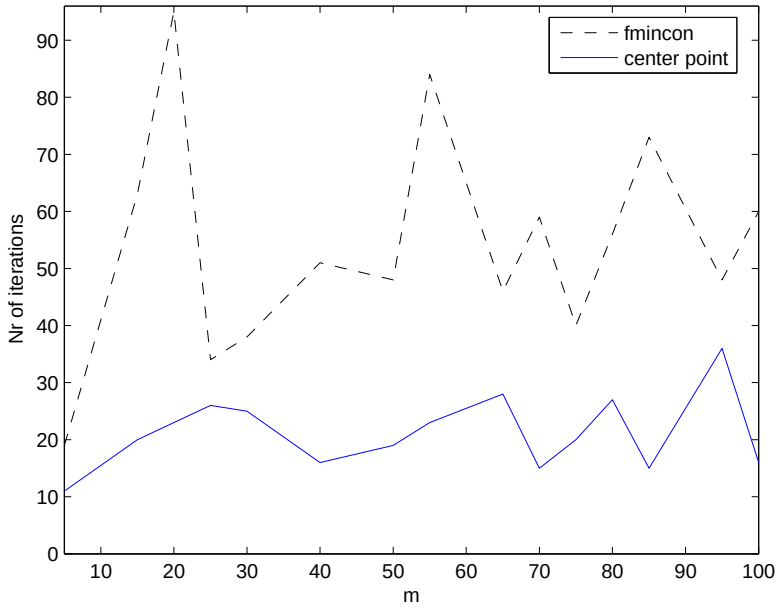

Figure 6.32: $\text{iter}_{\texttt{fmincon}}$ and $\text{iter}_{center\ point}$ for $n = 4$

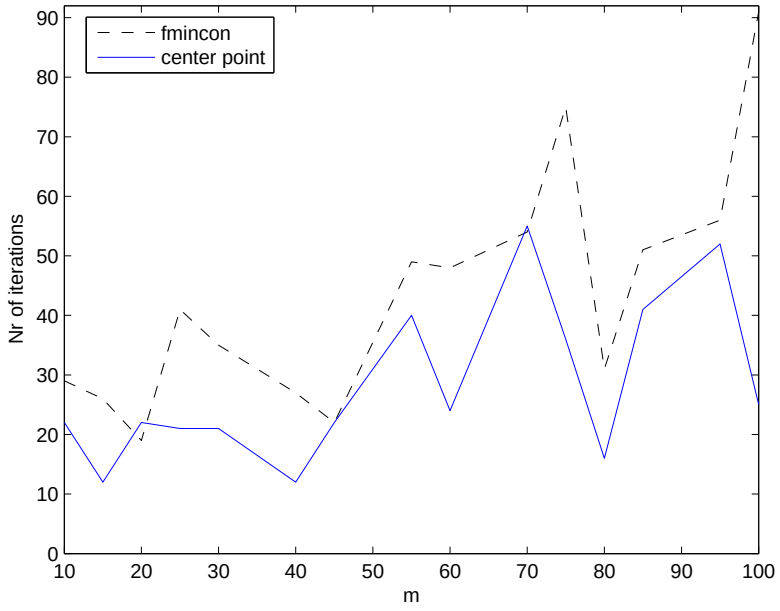

Figure 6.33: $\text{iter}_{\texttt{fmincon}}$ and $\text{iter}_{center\ point}$ for $n = 5$

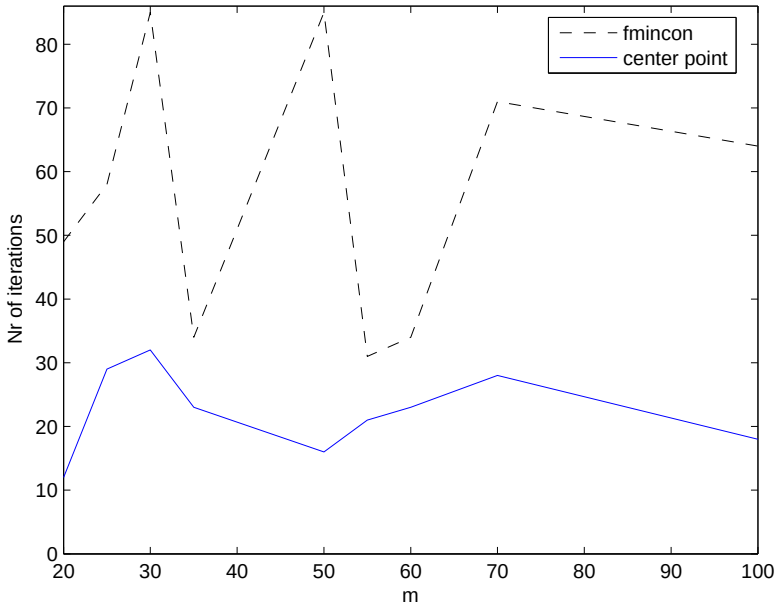

Figure 6.34: $\text{iter}_{\texttt{fmincon}}$ and $\text{iter}_{center\ point}$ for $n = 10$

6.3.4 Conclusions

The difference between $\left| \|\Delta A, \Delta b\|_{\texttt{fmincon}} - \|\Delta A, \Delta b\|_{center\ point} \right|$ grows with the growth of n. The sign Ψ of norm difference is changing randomly and independently of m and n.

The CPU-time required to find a minimum is exponential for Matlab's `fmincon` and linear for the center point method. This property is independent of the value of n.

The number of iterations for Matlab's `fmincon` increase with the value of m and n while the number of iterations required by the center point method remain between 10 and 40 for all values of m and n.

6.4 Comparisons of the Tikhonov-center Method and the Center Point Method

This section contains comparisons of the results obtained by the Tikhonov-center method and the center point method. Comparisons of values such as the norm $\|g\|$, $\|\Delta A, \Delta b\|_F$ and the CPU-time are done.

Only the values of m and n for which both the Tikhonov-center method and the center point method actually converge, are included in the graphs which show the norm $\|\Delta A, \Delta b\|_F$ and the CPU-time. The graphs showing the norm $\|g\|$ do not depend on the convergence of the methods.

In each figure, n is constant and m is the variable. The range of n is $\{1, 2, \ldots, 5, 10\}$, with one value of n per figure. The range of m is $\{5, 10, 15, 20, \ldots, 100\}$, with its initial value depending on n ($m > n$).

6.4.1 The Size of the Norm $\|g\|$

This section contains graphs showing the norm $\|g\|$ once a method has found a minimum or reached *MAXITER*. To get the 10:th power of the norms; log_{10} for $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ are plotted in each graph. $log_{10}(\epsilon)$ where $\epsilon = 10^{-9}$ is such that a minimum is found once $\|g\| < \epsilon$, is plotted as a reference.

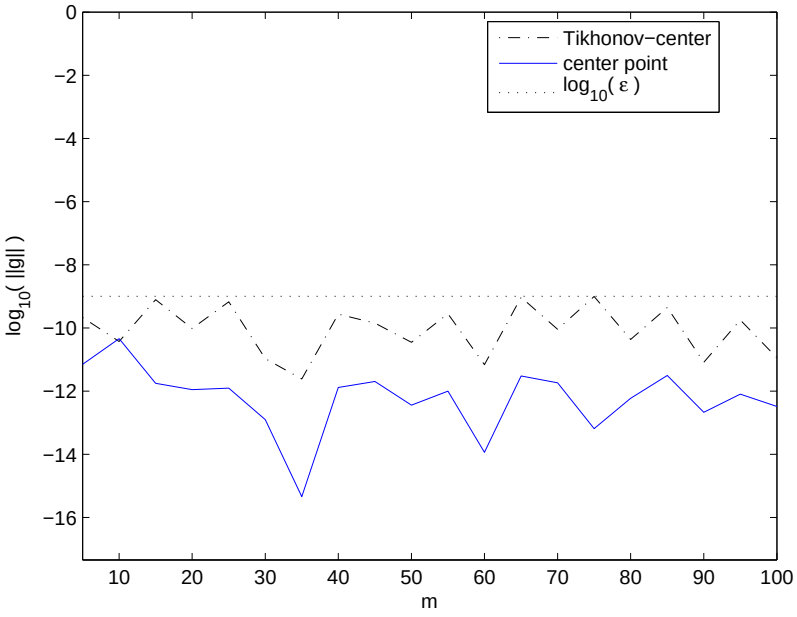

Figure 6.35: $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ for $n = 1$

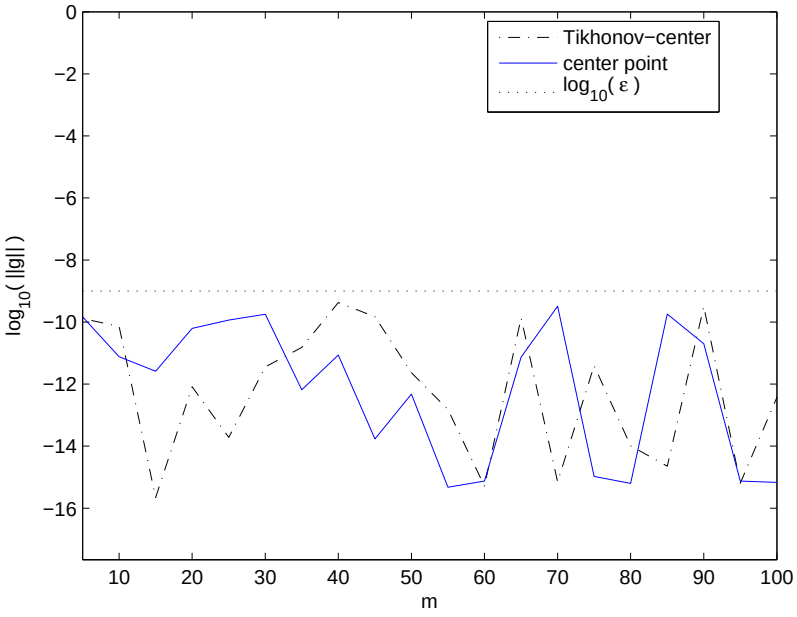

Figure 6.36: $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ for $n = 2$

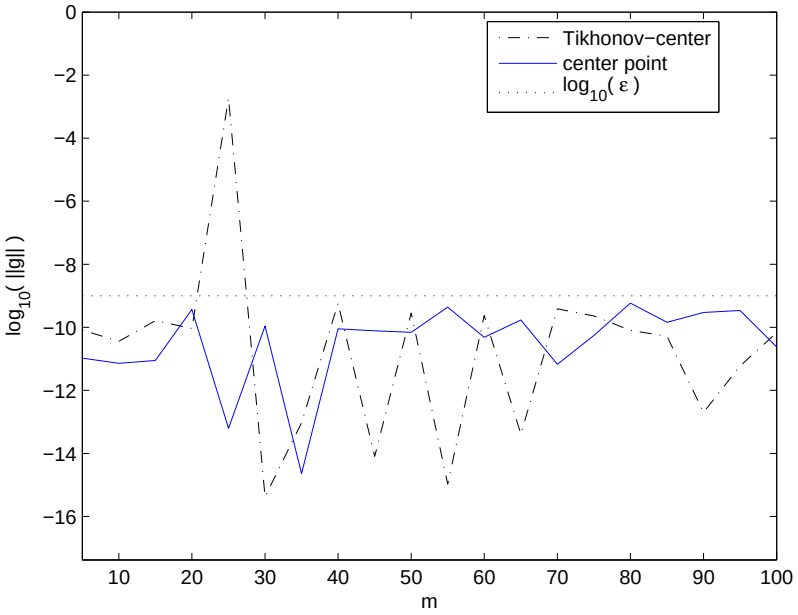

Figure 6.37: $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ for $n = 3$

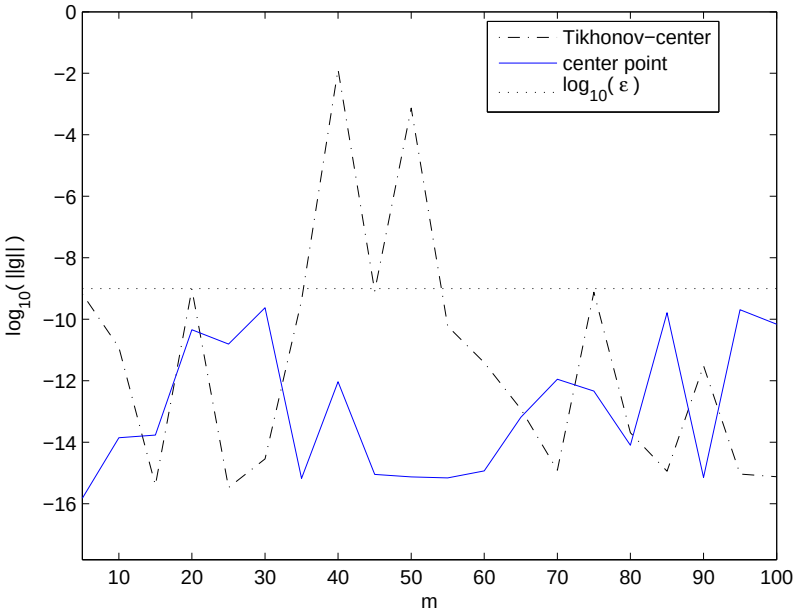

Figure 6.38: $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ for $n = 4$

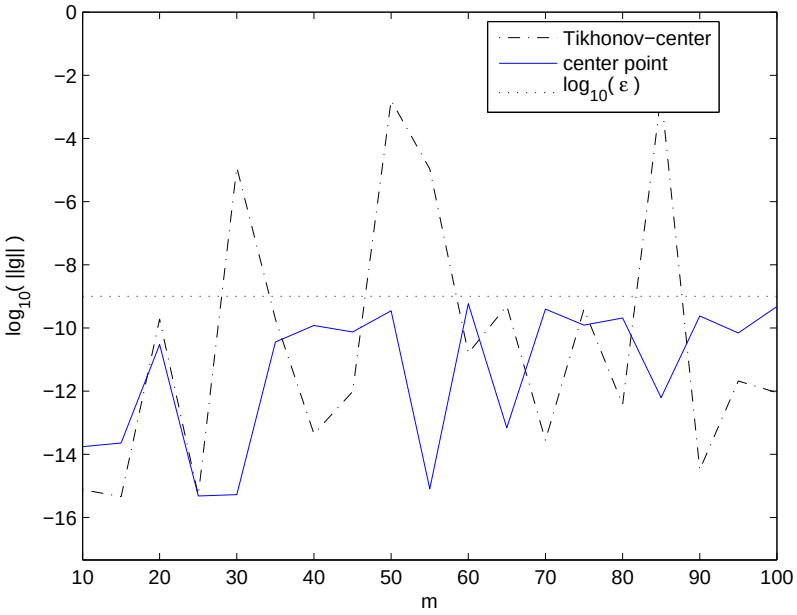

Figure 6.39: $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ for $n = 5$

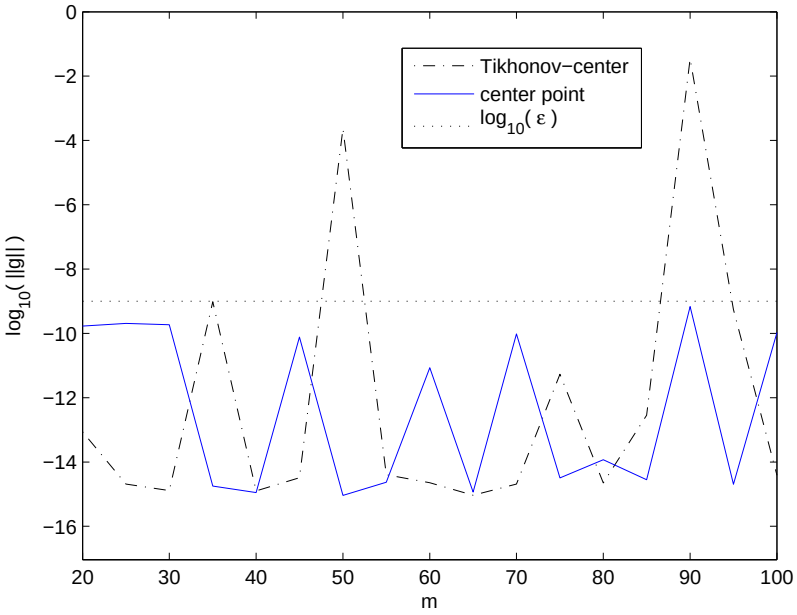

Figure 6.40: $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ for $n = 10$

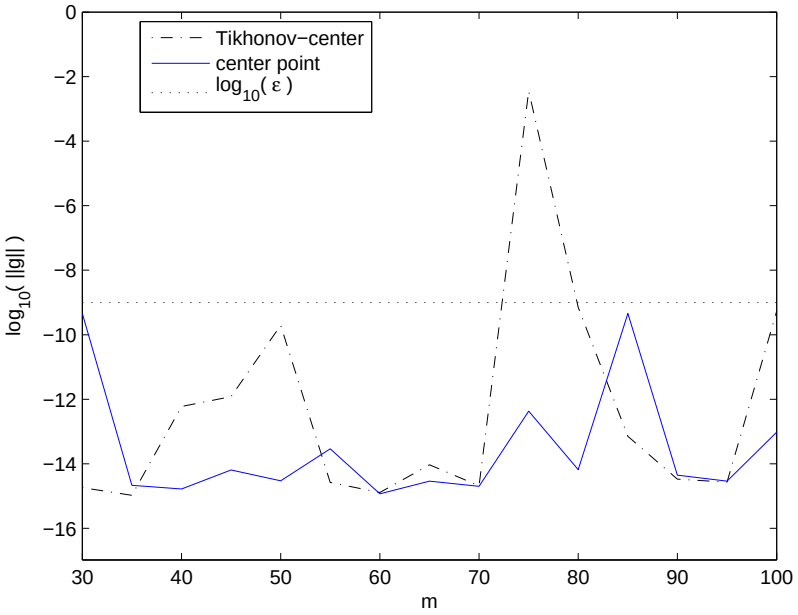

Figure 6.41: $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ for $n = 20$

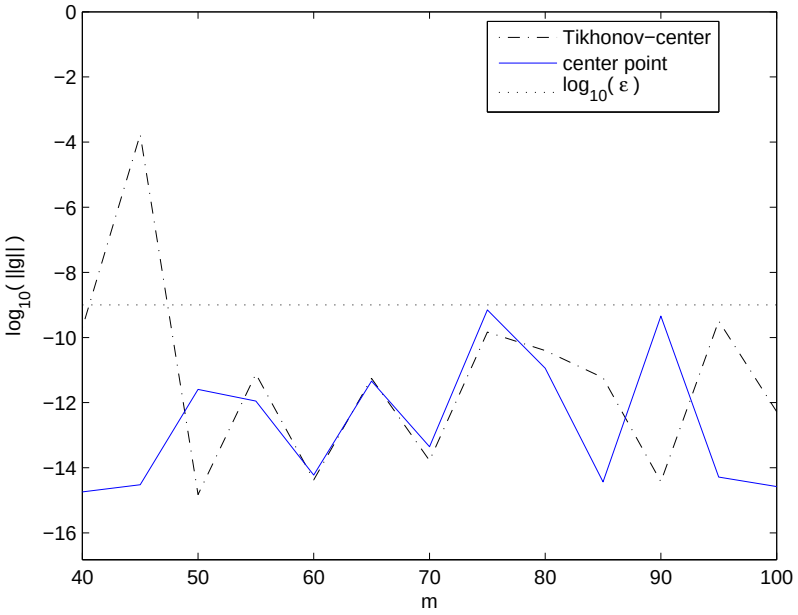

Figure 6.42: $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ for $n = 30$

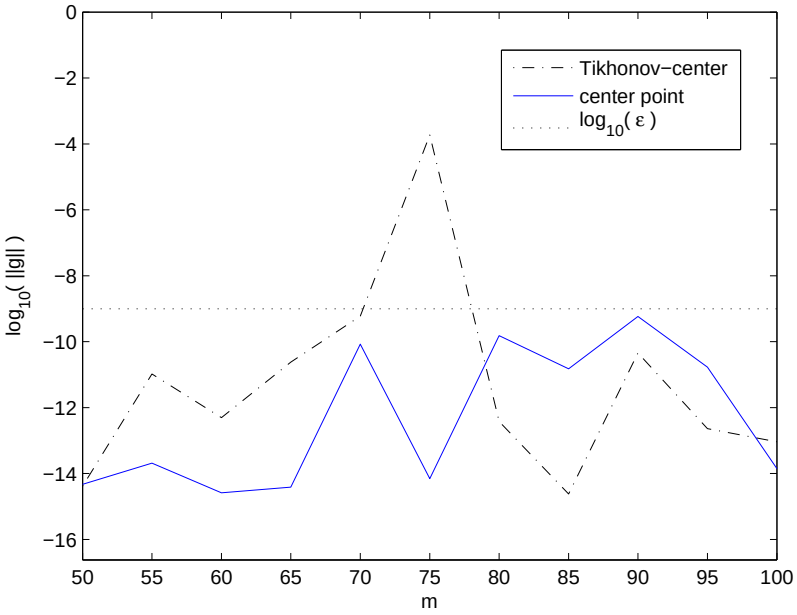

Figure 6.43: $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ for $n = 40$

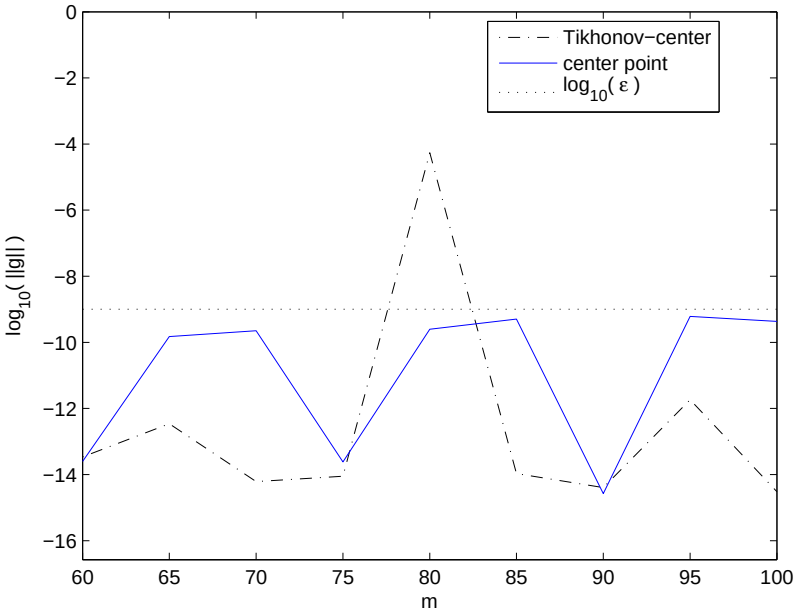

Figure 6.44: $\|g\|_{Tikhonov-center}$ and $\|g\|_{center\ point}$ for $n = 50$

6.4.2 The Size of the Norm $\|\Delta A, \Delta b\|_F$

Here follow graphs showing the size of the difference between $\|\Delta A, \Delta b\|_F$ at a minimum found by the Tikhonov-center method and $\|\Delta A, \Delta b\|_F$ at a minimum found by the center point method. (See section 6.3.1 at page 30 for a more thorough explanation of the formulas used in the graphs.)

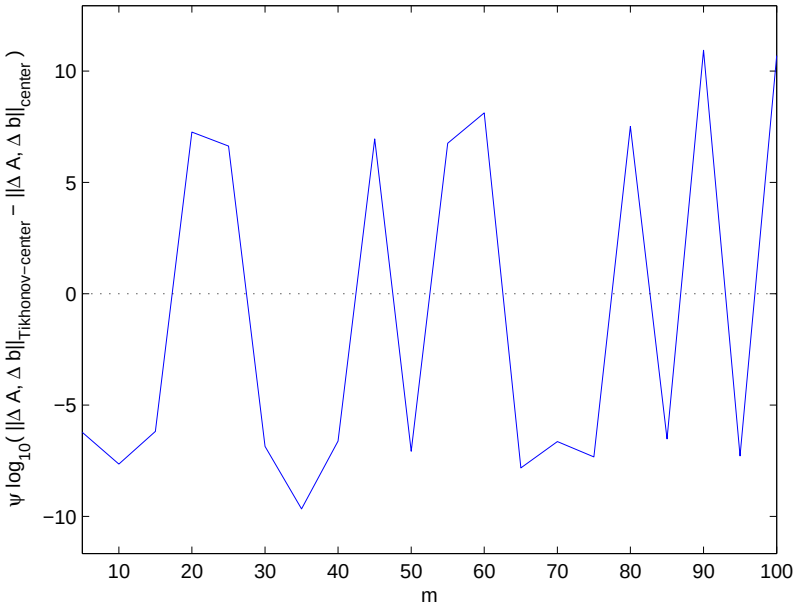

Figure 6.45: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 1$

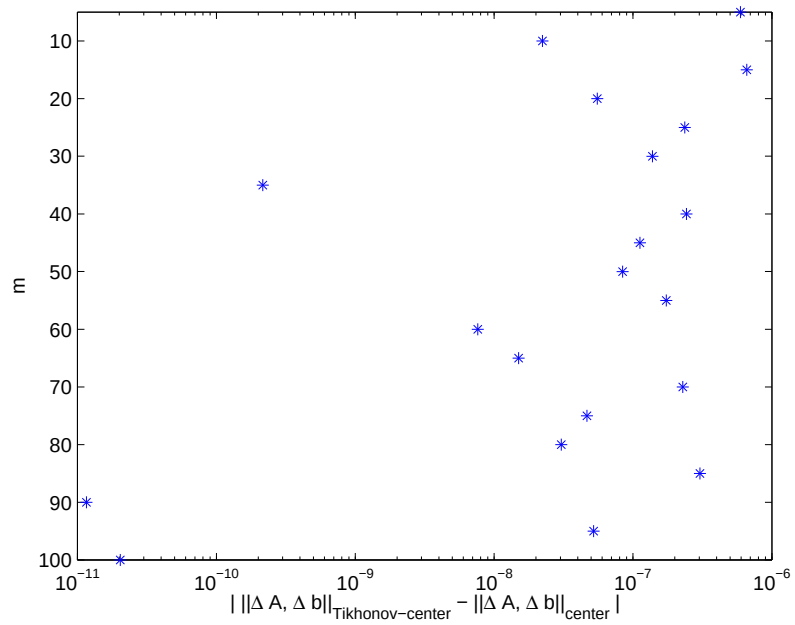

Figure 6.46: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 1$

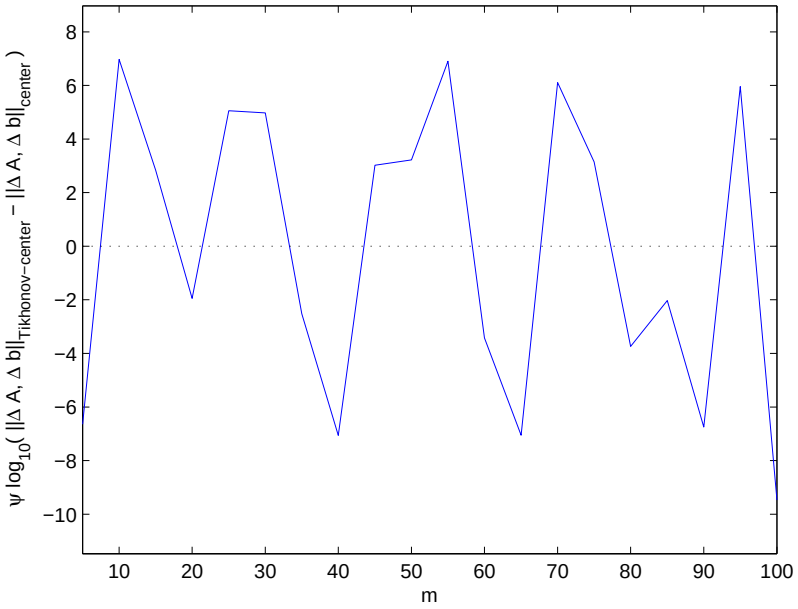

Figure 6.47: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 2$

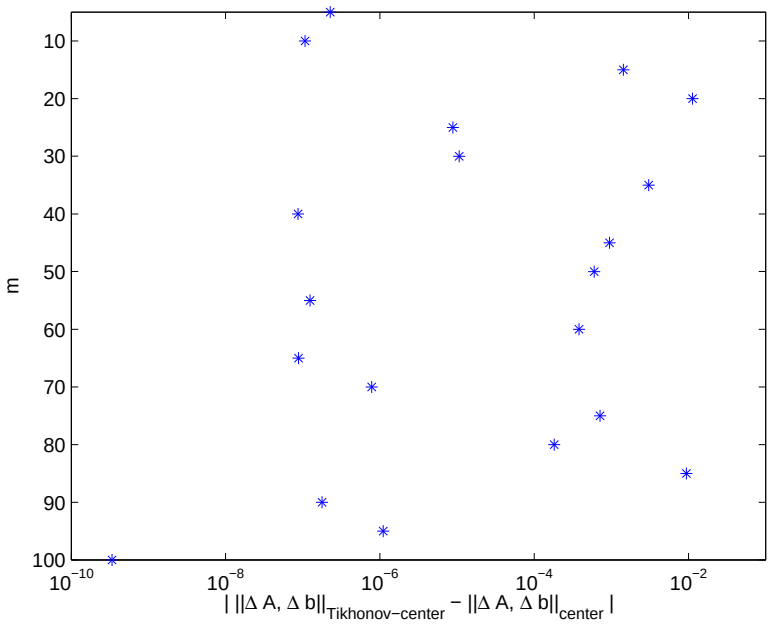

Figure 6.48: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 2$

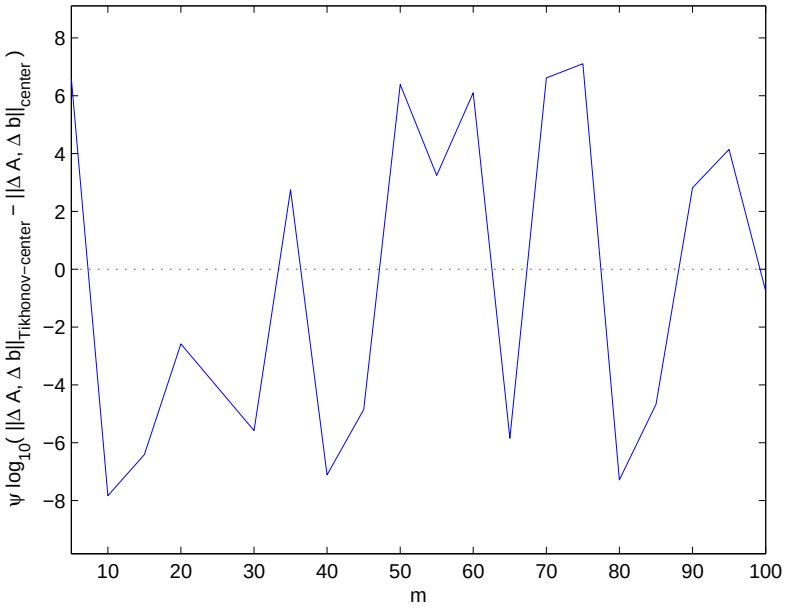

Figure 6.49: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 3$

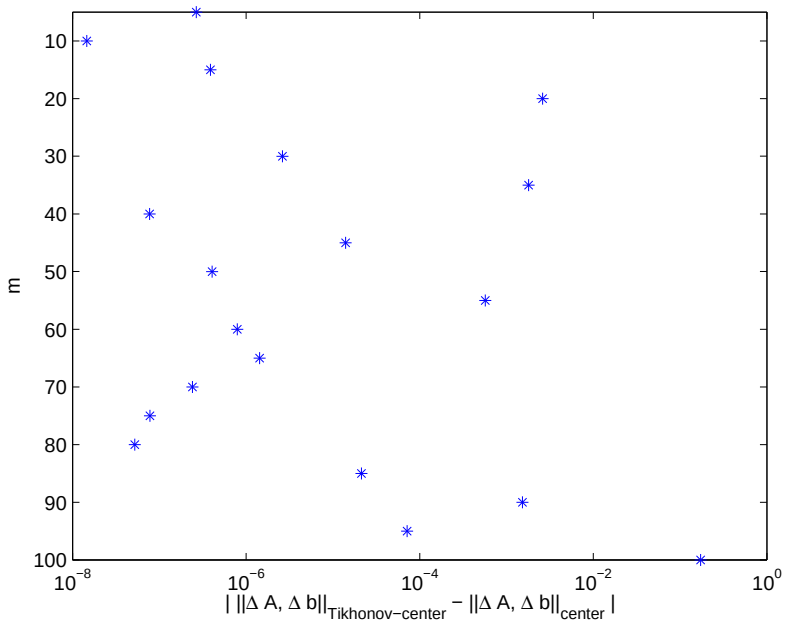

Figure 6.50: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 3$

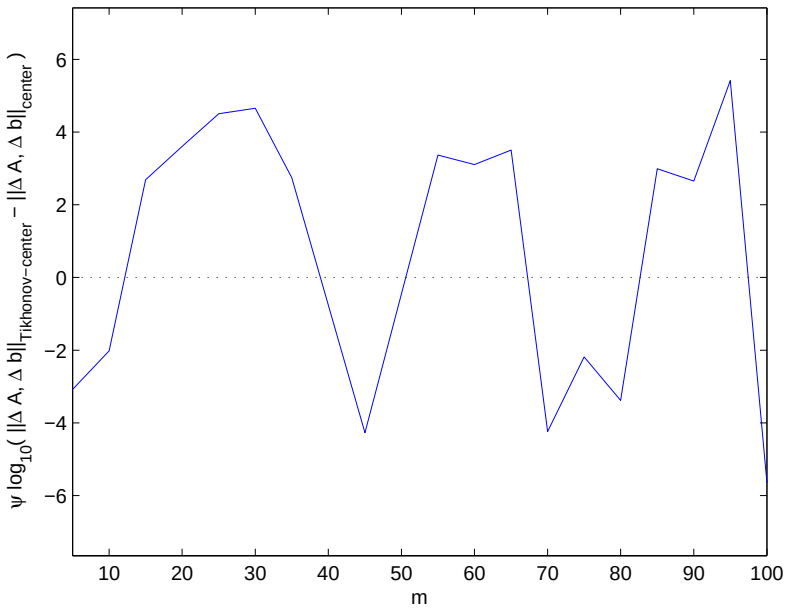

Figure 6.51: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 4$

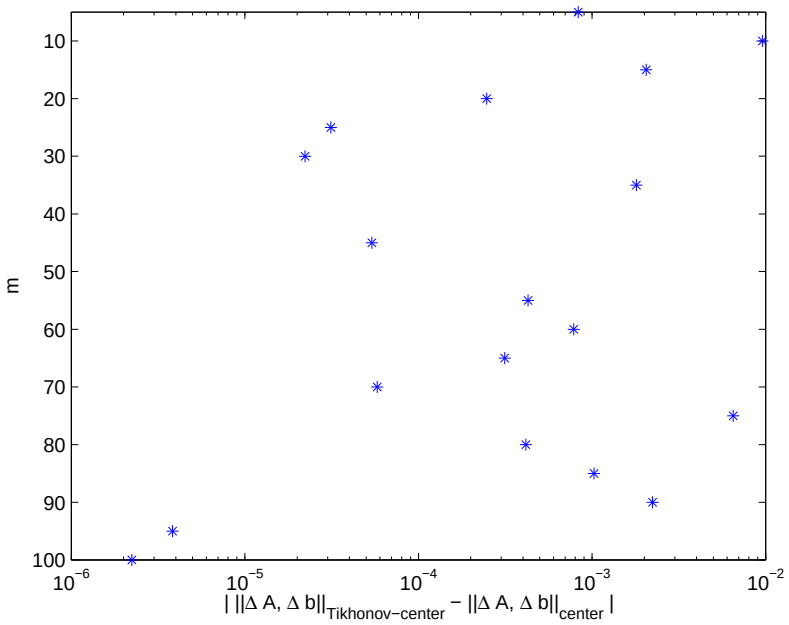

Figure 6.52: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 4$

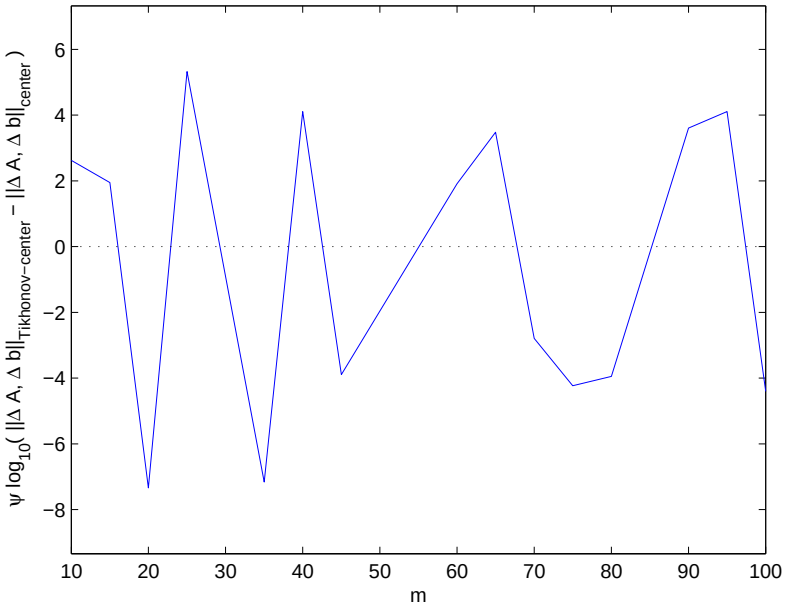

Figure 6.53: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 5$

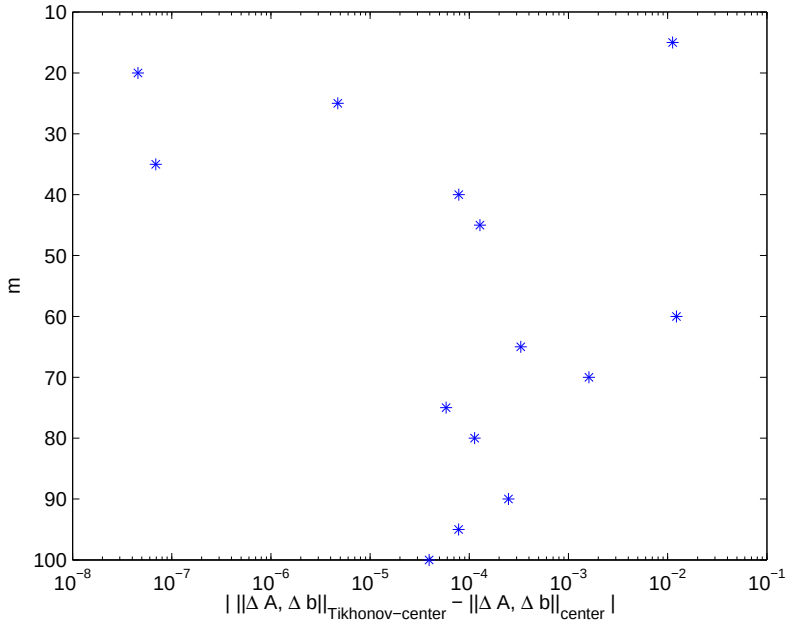

Figure 6.54: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 5$

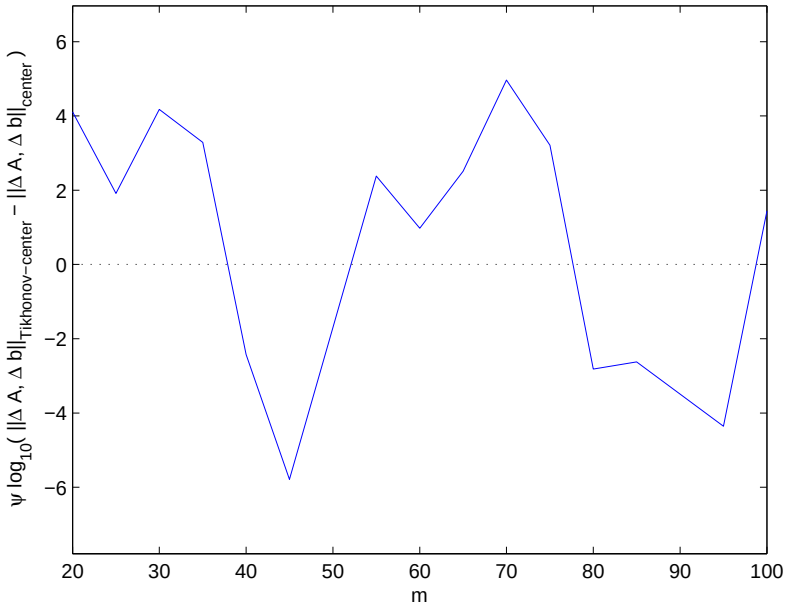

Figure 6.55: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 10$

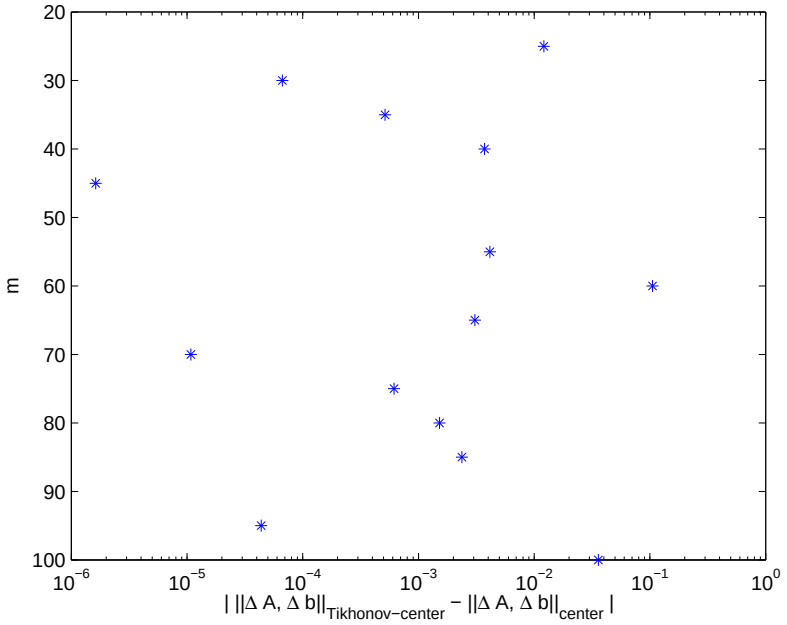

Figure 6.56: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 10$

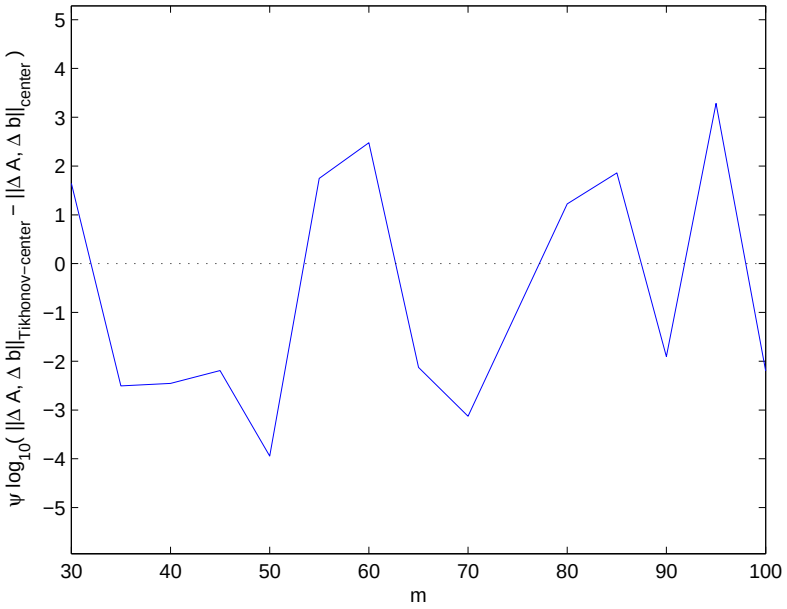

Figure 6.57: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 20$

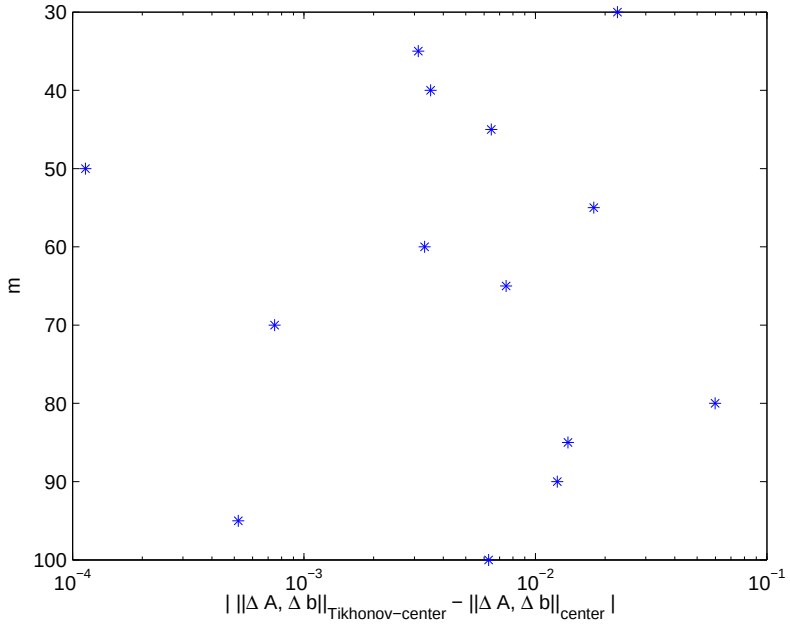

Figure 6.58: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 20$

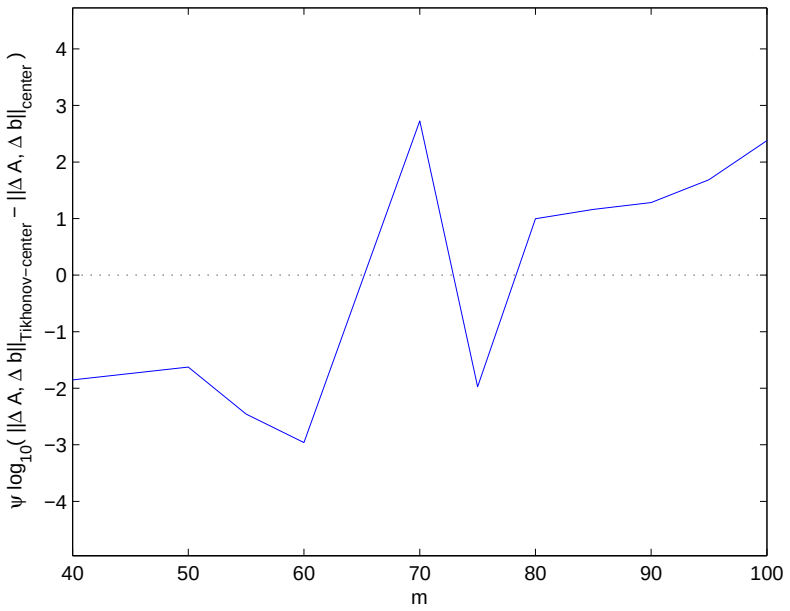

Figure 6.59: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 30$

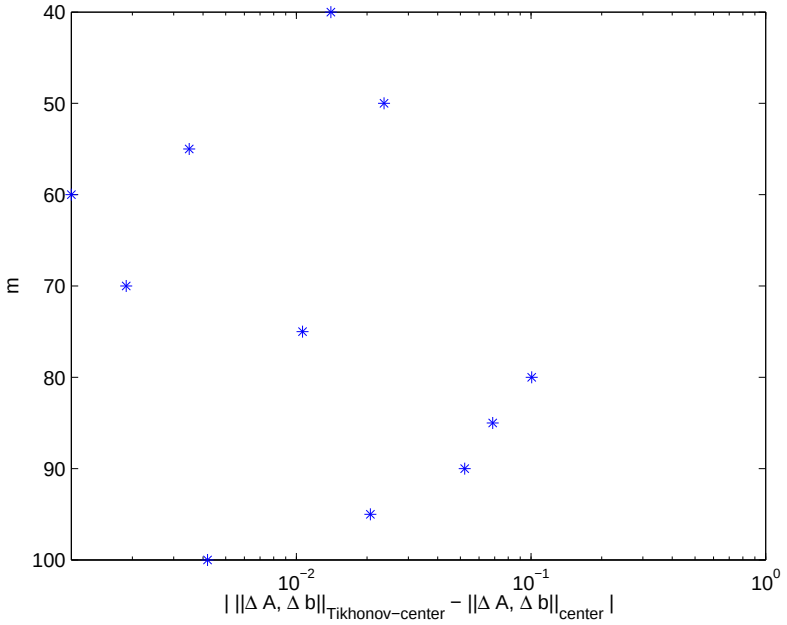

Figure 6.60: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 30$

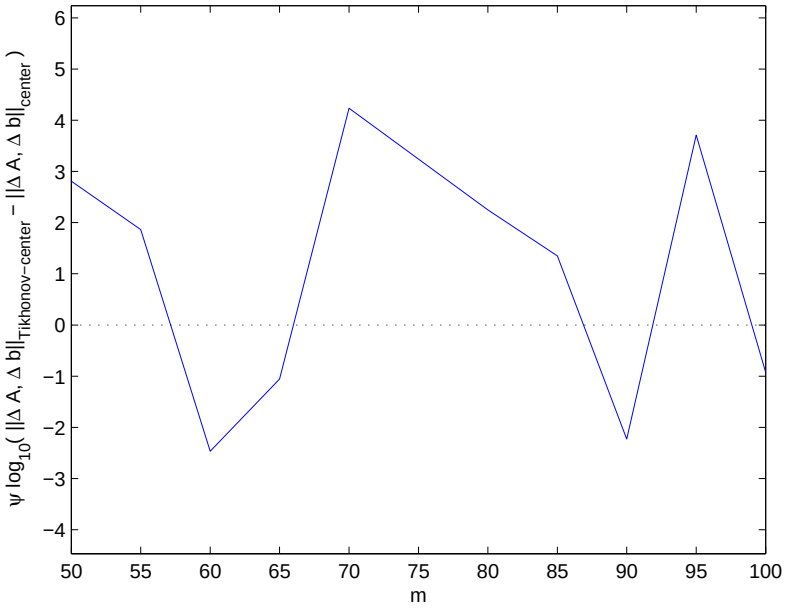

Figure 6.61: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 40$

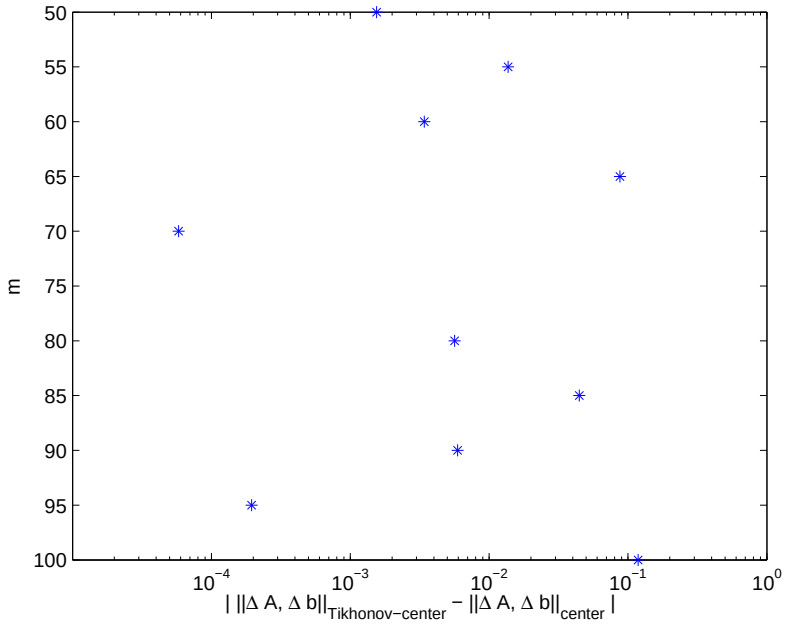

Figure 6.62: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 40$

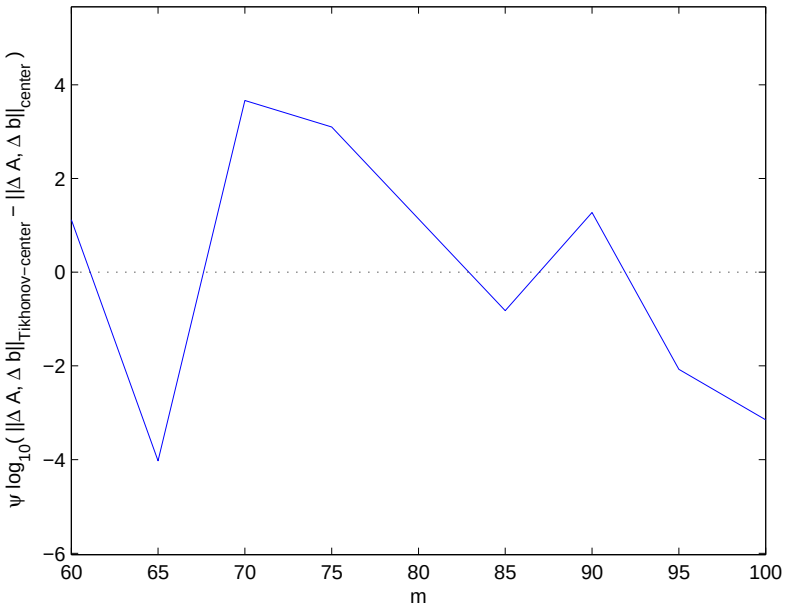

Figure 6.63: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 50$

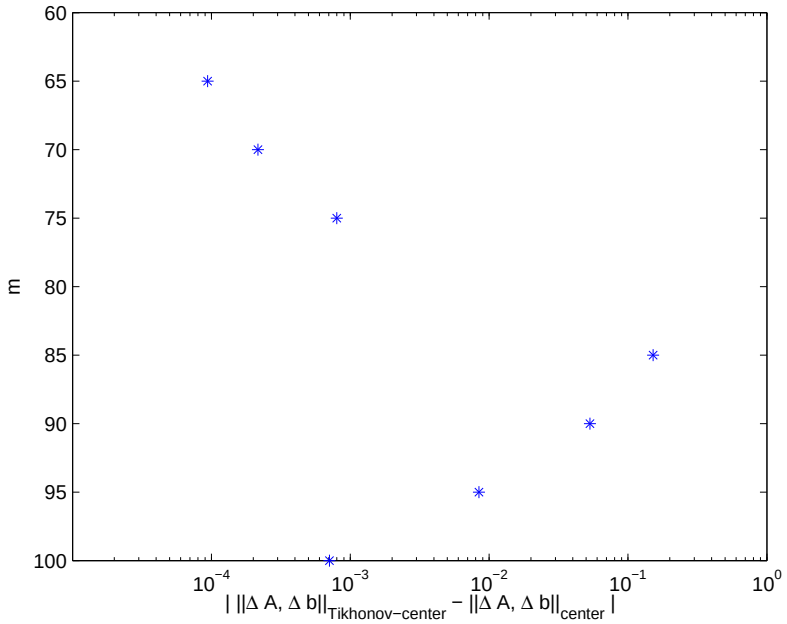

Figure 6.64: $\|\Delta A, \Delta b\|_{Tikhonov-center}$ and $\|\Delta A, \Delta b\|_{center\ point}$ for $n = 50$

6.4.3 The CPU-time

This section contains graphs showing the CPU-time needed to find a minimum by the Tikhonov-center method versus the CPU-time required by the center point method.

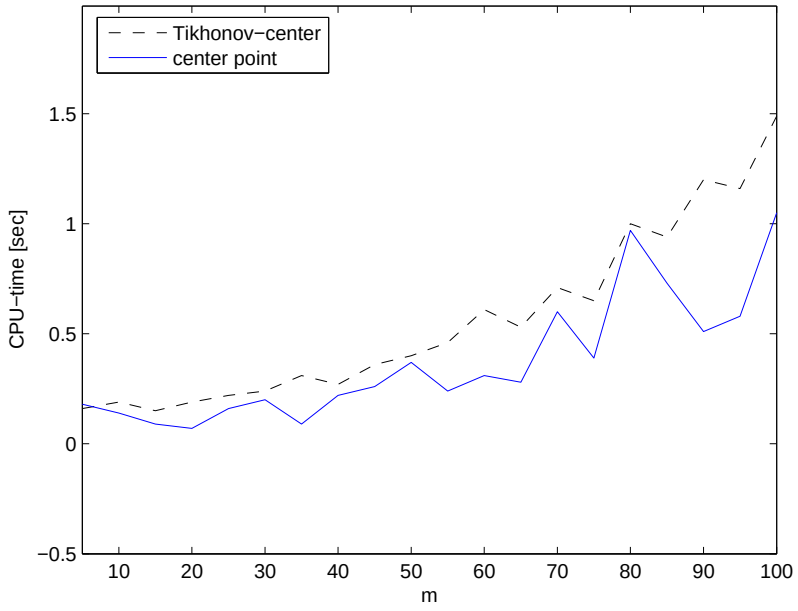

Figure 6.65: CPU-time$_{Tikhonov-center}$ and CPU-time$_{center\ point}$ for $n = 1$

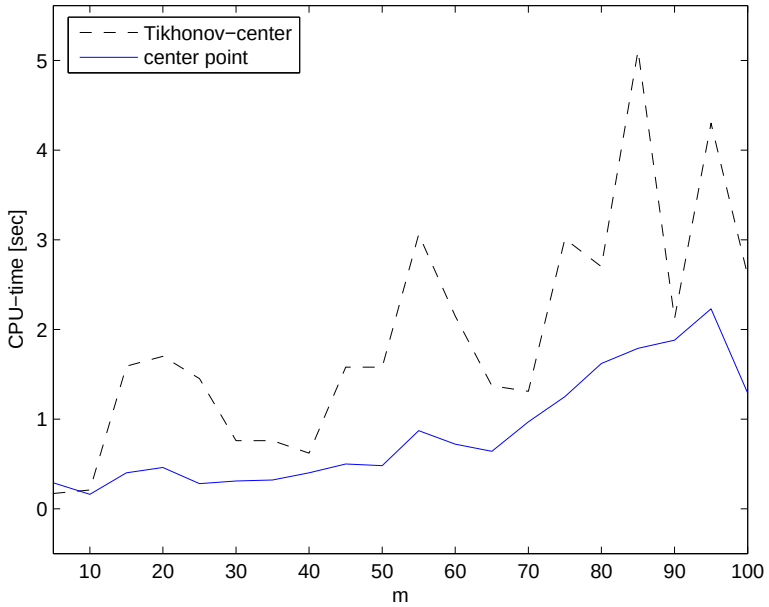

Figure 6.66: CPU-time$_{Tikhonov-center}$ and CPU-time$_{center\ point}$ for $n = 2$

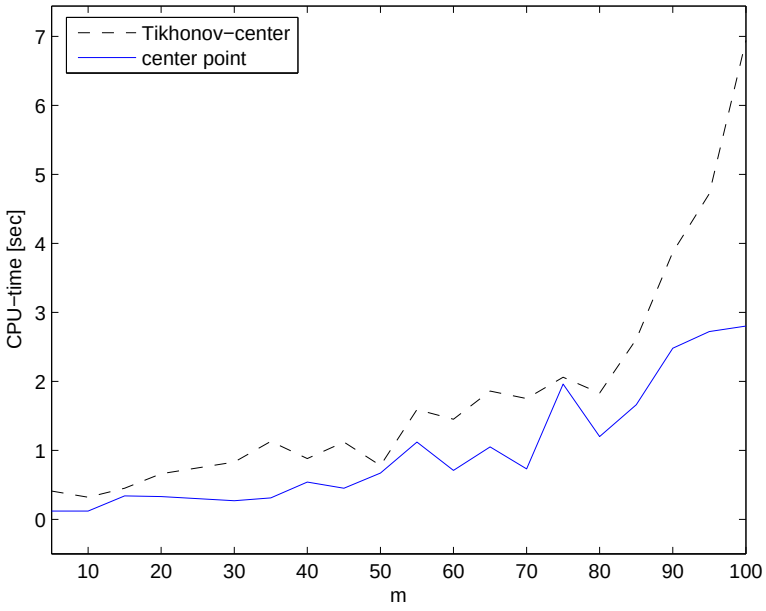

Figure 6.67: CPU-time$_{Tikhonov-center}$ and CPU-time$_{center\ point}$ for $n = 3$

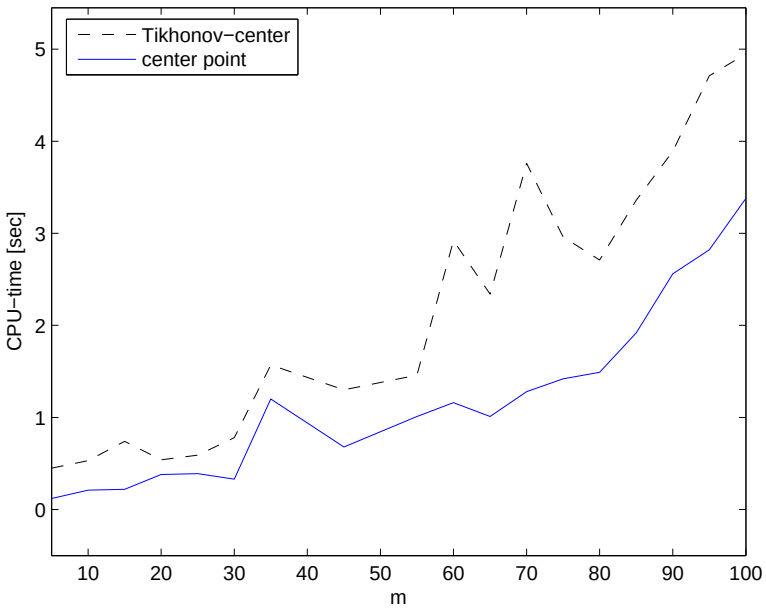

Figure 6.68: CPU-time$_{Tikhonov-center}$ and CPU-time$_{center\ point}$ for $n = 4$

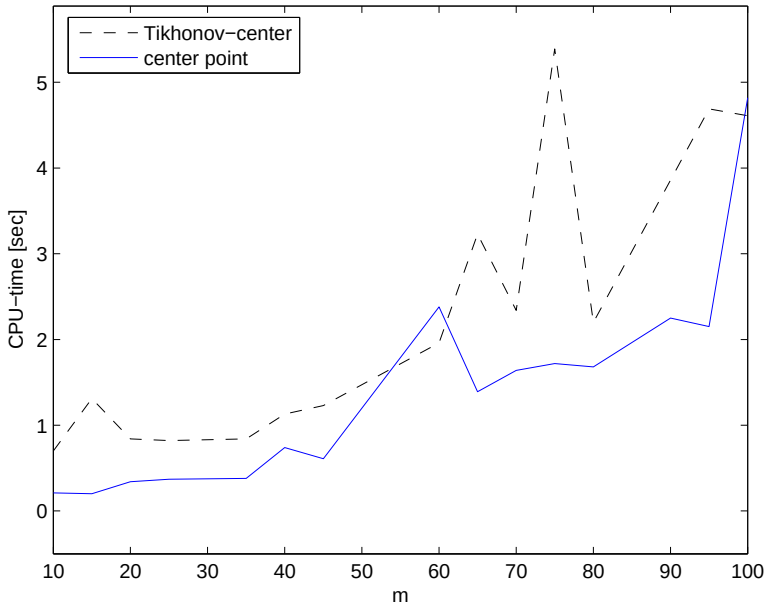

Figure 6.69: CPU-time$_{Tikhonov-center}$ and CPU-time$_{center\ point}$ for $n = 5$

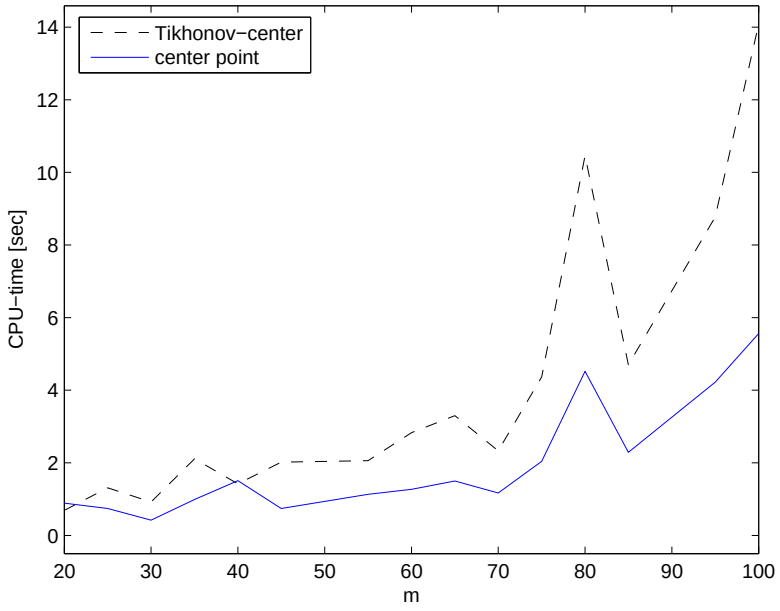

Figure 6.70: CPU-time$_{Tikhonov-center}$ and CPU-time$_{center\ point}$ for $n = 10$

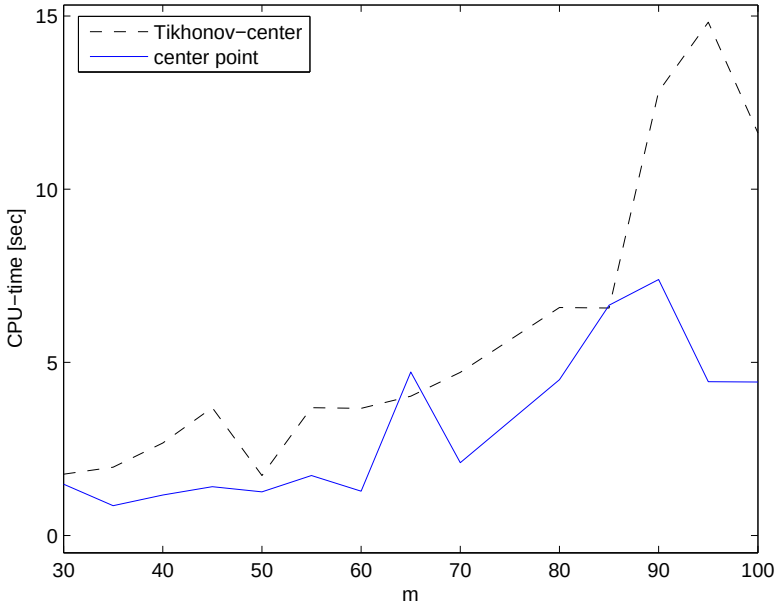

Figure 6.71: CPU-time$_{Tikhonov-center}$ and CPU-time$_{center\ point}$ for $n = 20$

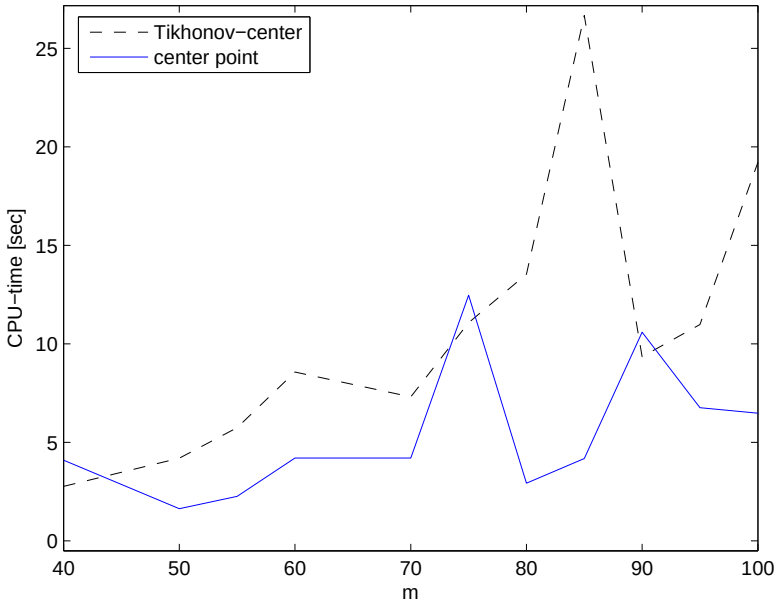

Figure 6.72: CPU-time$_{Tikhonov-center}$ and CPU-time$_{center\ point}$ for $n = 30$

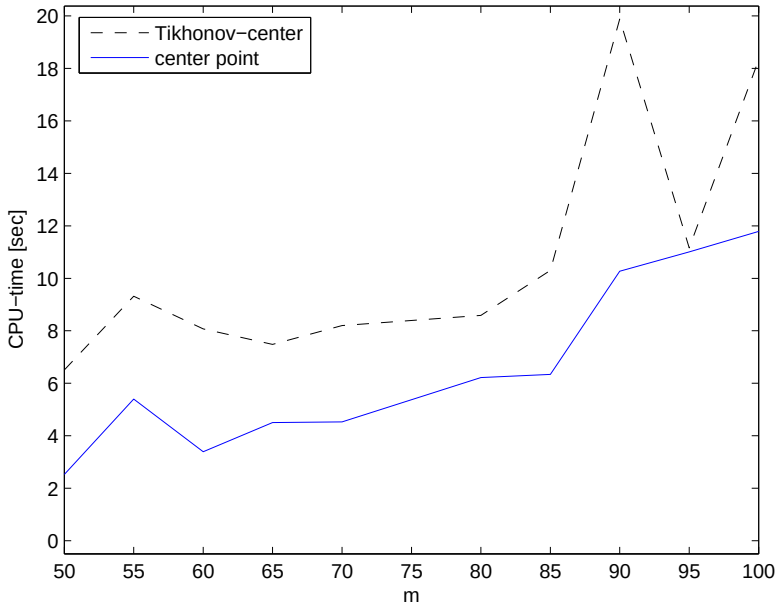

Figure 6.73: CPU-time$_{Tikhonov-center}$ and CPU-time$_{center\ point}$ for $n = 40$

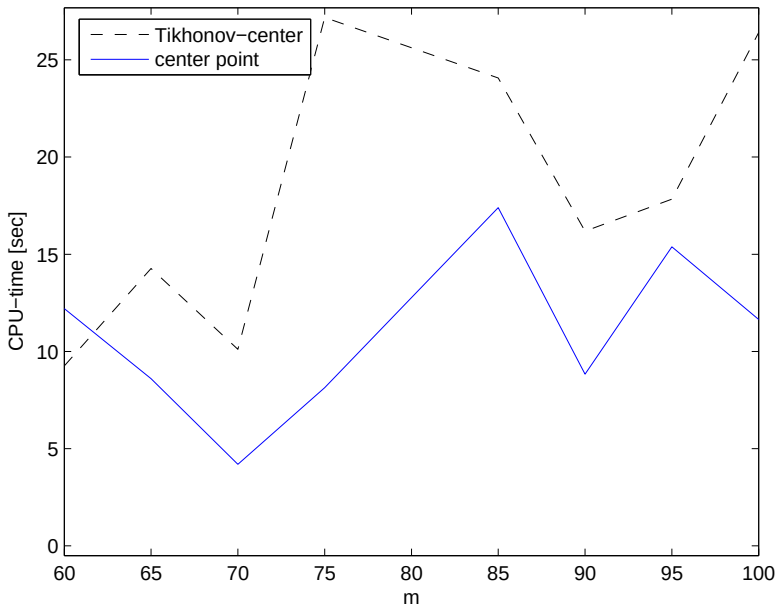

Figure 6.74: CPU-time$_{Tikhonov-center}$ and CPU-time$_{center\ point}$ for $n = 50$

6.4.4 Conclusions

The norm $\|g\|$ is in the interval 10^{-15} to 10^{-9} for both the methods when they converge. When the methods diverge, it applies to both of them that $10^{-4} \leq \|g\| \leq 10^{-2}$.

The difference between $\left| \|\Delta A, \Delta b\|_{Tikhonov-center} - \|\Delta A, \Delta b\|_{center\ point} \right|$ increases when n grows. The sign Ψ of the difference between the norms is changing randomly and independently of the values of m and n.

The amount of CPU-time required by the Tikhonov-center method is longer than that of the center point method. The difference is at most 20 seconds in the tested intervals of m and n.

Chapter 7

Summary of Conclusions

A summary of what can be concluded from the results obtained from numerous tests and study of optimization theory in general. The reliability, performance and correctness of the Gauß-Newton method, Matlab's `fmincon`, the Tikhonov-center method and of the center point method are discussed.

7.1 Reliability

The outcome of the Gauß-Newton method depends only on the start point, since this method does not make any use at all of the Hessian or any other property which may change the direction versus a minimum. Thus, for $n > 1$ this method's convergence is not reliable.

The reliability of Matlab's `fmincon` substantially decreases when $n > 5$ and it diminishes even more if m increases as well.

The Tikhonov-center method has a high level of reliability with respect to both the Gauß-Newton method and Matlab's `fmincon`. Its reliability is lower than that of the center point though. The probability of convergence for the Tikhonov-center method decreases with the value of m, but mostly with the value of n.

The center point method has a high reliability due to the change of the center point. Its reliability is independent of the values of m and n.

7.2 Performance

The Gauß-Newton method is fast for all values of m and n in the tested intervals.

Matlab's `fmincon` is already slow for $n > 1$ and becomes terribly slow when m increases. It is slow in number of iterations as well as the CPU-time. The performance of this method depends a lot on the start point x_{start} as well, this can be seen in the graphs showing the CPU-time and the number of iterations required to find a minimum. The CPU-time and the number of iterations are both zigzagging in each graph, which means that the performance depends on x_{start}, which is random since it is set by the randomly picked A and B.

The Tikhonov-center method is faster than the Gauß-Newton method and Matlab's `fmincon`, but a bit slower than the center point method in CPU-time.

The center point method is the fastest of the three methods for all the tested values of m and n. Its CPU-time is linear and the number of iterations vary between 10 and 40 independently of m and n. The value of $\|g\|$ is in average reduced with two more decimal places each step, compared to the other methods.
The convergence rate of the center point method is faster that that of a the general Gauß-Newton method because the first part as well as the second part of $Kmat_{GN}$ decrease when we approach x_{min} and therefore K_{GN} is always small. This happens because not only the norm $\|g\| \to 0$, but c continuously changes so that the value of $(x_{min} - c) \to 0$ as well. (See section 3.5.1 on page 8 for details about the convergence rate of the Gauß-Newton method).

7.3 Correctness

The Gauß-Newton method and Matlab's `fmincon` both guarantee that if a minimum is found, it is correct.

The Tikhonov-center method cannot guarantee the correctness of a minimum, since changing the center point also changes the problem that is minimized.
The minima found by the Tikhonov-center method differ from the ones found by the center point method on the 10th to the 1st decimal. And it appears random which method finds a better minimum.

The center point method cannot guarantee the correctness of a minimum, since the change of the center point also changes the object function and thus the problem. A minimum found by the center point method differs from a correct minimum on the 8th to the 1st decimal. The decimal difference aggravates when n grows.

There are a number of possible explanations to the different minima found by the Gauß-Newton method, Matlab's `fmincon` and the center point method:

1. The minimum found by the center point method is a completely different minimum than the one found by the Gauß-Newton method and Matlab's `fmincon`. This is possible and also likely, since all these optimization methods only find *local* minima.

2. The minimum found by the center point method really is the same minimum that is found by the Gauß-Newton method and Matlab's `fmincon`, but in the procedure of changing the center point, the center point method "ruined" the real value of the minimum.

In any case, the sign of $(x_{min\texttt{fmincon}} - x_{min\,center\ point})$ is random in the whole interval of m and n, which means that the center point method finds a smaller (better) minimum in approximately 50% of the cases.

Chapter 8

Future Works

The algorithm for the change of the center point can surely be more efficient. A better algorithm for this should be found.

The choice of the punishment-scalar μ in the Tikhonov-center method is not optimal. A better algorithm should be invented and developed.

Comparisons between the center point method and a method which solves STLS problems more efficiently and reliably than Matlab's `fmincon` should be performed.

It would be interesting to do a more profound study of the minima found by the center point method and other methods known to find correct minima, to see which properties the minima found by the center point method really possess.

Chapter 9

Related Works

Here follows a list with interesting works related to STLS problems. The publications are from the year 2000 and onwards.

1. N. Mastronardi, P. Lemmerling and S. Van Huffel. Fast Structured Total Least Squares Algorithm for Solving the Basic Deconvolution Problem. *SIAM J. Matrix Analysis and Applications*, 22(2):533−553, 2000.

2. P. Lemmerling, N. Mastronardi and S. Van Huffel. Fast Algorithm for Solving the Hankel/Toeplitz Structured Total Least Squares Problem. *Numerical Algorithms*, 23(4):371−392, 2000.

3. P. Lemmerling and S. Van Huffel. Structured Total Least Squares : Analysis, Algorithms and Applications. Internal report, *ESAT-SISTA*, K.U.Leuven (Leuven, Belgium), 2001.

4. N. Mastronardi, P. Lemmerling, and S. Van Huffel. The structured total least squares problem. *Contemporary Mathematics, AMS*, 280(1):157−176, 2001.

5. N. Mastronardi. *Fast and Reliable Algorithms for Structured Total Least Squares and Related Matrix Problems.* PhD thesis, K.U.Leuven (Leuven, Belgium), 2001.

6. P. Lemmerling and S. Van Huffel. Analysis of the Structured Total Least Squares Problem for Hankel/Toeplitz Matrices. *Numerical Algorithms*, 27(1):89−114, 2001.

7. P. Lemmerling, S. Van Huffel and B. De Moor. The structured total least-squares approach for non-linearly structured matrices. *Numerical Linear Algebra with Applications*, 9(4):321−332, 2002.

8. A. Kukush, I. Markovsky and S. Van Huffel. Consistency of the Structured Total Least Squares Estimator in a Multivariate Errors-in-variables Model. Internal report 02-192, *ESAT-SISTA*, K.U.Leuven (Leuven, Belgium), 2002.

9. P. Yalamov, Y. Plamen and J.Y. Yuan. A Successive Least Squares Method For Structured Total Least Squares. *Journal of Computational Mathematics*, 21(4):463−472, 2003

10. A. Pruessner and D. P. O'Leary. Blind Deconvolution Using a Regularized Structured Total Least Norm Algorithm. *SIAM J. Matrix Analysis and Applications*, 24(4):1018−1037, 2003.

11. I. Markovsky, S. Van Huffel and R. Pintelon. Block-Toeplitz/Hankel Structured Total Least Squares. Internal report 03-135, *ESAT-SISTA*, K.U.Leuven (Leuven, Belgium), 2003.

12. I. Markovsky, J.C. Willems, S. Van Huffel, B. De Moor and R. Pintelon. Application of Structured Total Least Squares for System Identification and Model Reduction. Internal report 04-51, *ESAT-SISTA*, K.U.Leuven (Leuven, Belgium), 2004.

13. I. Markovsky, S. Van Huffel, and A. Kukush. On the Computation of the Structured Total Least Squares Estimator. *Numerical Linear Algebra with Applications*, 11:591−608, 2004.

14. I. Markovsky and S. Van Huffel. High-performance Numerical Algorithms and Software for Structured Total Least Squares. Internal Report 04-102, *ESAT-SISTA*, K.U.Leuven (Leuven, Belgium), 2004.

15. I. Markovsky and S. Van Huffel. Weighted Structured Total Least Squares. Internal Report 05-41, *ESAT-SISTA*, K.U.Leuven (Leuven, Belgium), 2005.

References

[1] Jerry Eriksson. Optimization and regularization of nonlinear least squares problems. Master's thesis, Umeå University (Umeå, Sweden), 1996.

[2] Umeå Optimization Group. Private discussions. 1993.

[3] S. Van Huffel and J. Vanderwalle. *The Total Least Squares Problem; Computational Aspects and Analysis*. Society of Industrial and Applied Mathematics (SIAM), 1991.

[4] J. B. Rosen, Hausen Park, and John Glick. Total least norm formulation and solution for structured problems. *SIAM J. Matrix Analysis and Application*, 17(1):16–19, 1996.

[5] J. B. Rosen, Hausen Park, and John Glick. Structured total least norm for nonlinear problems. *SIAM J. Matrix Analysis and Application*, 20(1):112–119, 1998.