IMM
DEPARTMENT OF MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Lyngby – Denmark

J. No. PQSOLV
28.9.1998
HBN/ms

COMPUTING A MINIMIZER OF A PIECEWISE QUADRATIC

IMPLEMENTATION

Hans Bruun Nielsen

TECHNICAL REPORT
IMM-REP-1998-14

IMM

COMPUTING A MINIMIZER OF A PIECEWISE QUADRATIC

IMPLEMENTATION

Hans Bruun Nielsen

Contents

1. Introduction

Given the scalars γ and μ, the vectors $\boldsymbol{b} \in I\!R^m$ and $\boldsymbol{c} \in I\!R^n$, and the matrices $\boldsymbol{A} \in I\!R^{m\times n}$, $\boldsymbol{L} \in I\!R^{q\times n}$. We consider the problem of finding

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\,\in\, I\!R^n}{\operatorname{argmin}} \left\{ F(\boldsymbol{x}) \equiv \sum_{i=1}^{m} \rho(r_i(\boldsymbol{x})) + \tfrac{1}{2}\mu\|\boldsymbol{L}\boldsymbol{x}\|_2^2 + \boldsymbol{c}^T\boldsymbol{x} \right\} , \tag{1.1a}$$

where $r_i(\boldsymbol{x}))$ is the ith component of the residual

$$\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} , \tag{1.1b}$$

and

$$\rho(r) = \begin{cases} \dfrac{1}{2\gamma} r^2 & \text{for } |r| \le \gamma , \\ |r| - \frac{1}{2}\gamma & \text{for } |r| > \gamma . \end{cases} \tag{1.1c}$$

In Section 4 we present alternative choices of ρ and discuss the modifications in the algorithm for computing $\boldsymbol{x}^*$.

We introduce a "generalized sign vector" $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{x}) \in I\!R^m$ and "activity matrix" $\boldsymbol{W} = \boldsymbol{W}(\boldsymbol{x}) = \mathrm{diag}(w_1(\boldsymbol{x}), \ldots, w_m(\boldsymbol{x}))$ defined by

$$\begin{array}{lll} \textbf{if } |r_i(\boldsymbol{x})| \le \gamma & \textbf{then} & s_i(\boldsymbol{x}) = 0, \quad w_i(\boldsymbol{x}) = 1 \\ & \textbf{else} & s_i(\boldsymbol{x}) = \mathrm{sign}(r_i(\boldsymbol{x})), \quad w_i(\boldsymbol{x}) = 0 \ . \end{array} \tag{1.2}$$

Now, (1.1) can be expressed in the form

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\,\in\, I\!R^n}{\operatorname{argmin}} \{F(\boldsymbol{x})\} \tag{1.3a}$$

with

$$F(\boldsymbol{x}) = \frac{1}{2\gamma}\boldsymbol{r}^T\boldsymbol{W}\boldsymbol{r} + \boldsymbol{s}^T\left(\boldsymbol{r} - \tfrac{1}{2}\gamma\boldsymbol{s}\right) + \tfrac{1}{2}\mu\boldsymbol{x}^T\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} . \tag{1.3b}$$

The hyperplanes $|r_i(\boldsymbol{x})| = \gamma$ divide $I\!R^n$ into subregions $\{\Omega_k\}$, in each of which $\boldsymbol{s}(\boldsymbol{x})$ and $\boldsymbol{W}(\boldsymbol{x})$ is constant, and $F(\boldsymbol{x})$ is a second order polynomial. Further, F and its gradient

$$F'(\boldsymbol{x}) = -\boldsymbol{A}^T\left(\frac{1}{\gamma}\boldsymbol{W}(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x}) + \boldsymbol{s}(\boldsymbol{x})\right) + \mu\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{x} + \boldsymbol{c} \tag{1.4a}$$

are continuous, so F is a second order spline. Also, under certain assumptions about $\boldsymbol{c}$, F is convex, implying that a stationary point of F is a minimizer, i.e. a solution to

$$F'(\boldsymbol{x}^*) = \boldsymbol{0} \tag{1.4b}$$

is a minimizer of F.

The simplest case occurs when $\mu = 0$ and $\boldsymbol{c} = \boldsymbol{0}$. Then the function F is a modified least-squares model, where a large residual ($|r_i| > \gamma$) essentially contributes with its absolute value instead of its square, and (1.4) shows that the contribution to the modified normal equations is the threshold γ times the *sign* of r_i. This method for reducing the effect of "wild points" was introduced by *Huber* [4], cf. Figure 1.1, where the dotted line is the simple function $\frac{1}{2\gamma} r^2$.

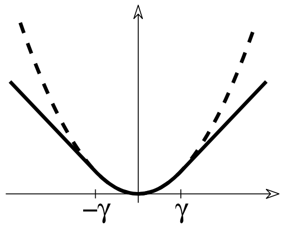

Figure 1.1. *Huber function*

In [10] we discussed an algorithm for finding $\boldsymbol{x}^*$, and in [11] we extended the algorithm to find the ℓ_1-solution of an overdetermined system of equations. The basic idea is to solve the dual problem via a series of problems of the form (1.1) with $\mu = 0$, $\boldsymbol{c} = \boldsymbol{0}$ and $\gamma \searrow 0$.

In [13] we extended the dual approach to linear programming problems, giving rise to series of problems of the form (1.1) with $\mu = 0$, $\boldsymbol{c} \neq \boldsymbol{0}$ and $\gamma \searrow 0$, and in [12] and [14] quadratic programming problems were solved via (1.1) with $\mu \neq 0$, $\boldsymbol{L} = \boldsymbol{I}$ and $\boldsymbol{c} = \boldsymbol{0}$. Another instance of this case occurs with robust nonlinear parameter estimation, where a Gauss-Newton iteration with Marquardt modification gives rise to series of problems of the form (1.1), where $\boldsymbol{x}$ and $\frac{1}{2}\mu$ is the increment vector and the Marquardt parameter, respectively.

Robust fitting of data in inverse problems can be modelled by (1.1) with $\mu \neq 0$, $\boldsymbol{L} \neq \boldsymbol{I}$; this is obtained by straight-forward modification of the models in [3], [8], [9]. Finally, the smoothing techniques applied in [6] and [7] are closely related to (1.1).

The workhorse in all these methods is the algorithm for solving (1.1). It has developed over the last 10 years, and the purpose of this report is to describe the present version of the algorithm and its implementation.

2. The Algorithm

As discussed above, the problem is to solve (1.4). We do this by iteration: Let $\boldsymbol{x}$ denote the current iterate, located in subregion $\Omega(\boldsymbol{x})$, and let $\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{x})$, $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{x})$ and $\boldsymbol{W} = \boldsymbol{W}(\boldsymbol{x})$. Then, for $\boldsymbol{h}$ small enough to let $\boldsymbol{x}+\boldsymbol{h} \in \Omega(\boldsymbol{x})$,

$$\begin{aligned} F'(\boldsymbol{x}+\boldsymbol{h}) &= -\boldsymbol{A}^T \left(\tfrac{1}{\gamma} \boldsymbol{W}(\boldsymbol{r} - \boldsymbol{A}\boldsymbol{h}) + \boldsymbol{s} \right) + \mu \boldsymbol{L}^T \boldsymbol{L}(\boldsymbol{x}+\boldsymbol{h}) + \boldsymbol{c} \\ &= F'(\boldsymbol{x}) + \left(\tfrac{1}{\gamma} \boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A} + \mu \boldsymbol{L}^T \boldsymbol{L} \right) \boldsymbol{h} \quad . \end{aligned}$$

The Newton step $\boldsymbol{h}$ is found by equating this expession with the zero vector, i.e.

$$\boldsymbol{H}\boldsymbol{h} = \boldsymbol{g} \ , \tag{2.1a}$$

where

$$\boldsymbol{H} = \boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A} + \gamma\mu \boldsymbol{L}^T \boldsymbol{L} \ , \tag{2.1b}$$

$$\boldsymbol{g} = -\gamma F'(\boldsymbol{x}) = \boldsymbol{A}^T(\boldsymbol{W}\boldsymbol{r} + \gamma \boldsymbol{s}) - \gamma\mu \boldsymbol{L}^T \boldsymbol{L}\boldsymbol{x} - \gamma \boldsymbol{c} \ . \tag{2.1c}$$

In earlier implementations, [10] – [14], [16], [17], [19], the aim was to find $\Omega(\boldsymbol{x}^*)$, since $\boldsymbol{x}+\boldsymbol{h} \in \Omega(\boldsymbol{x})$ implies that $\boldsymbol{x}^* = \boldsymbol{x}+\boldsymbol{h}$, and we are through. This, however, implied costly extra computation in the case of a rank deficient $\boldsymbol{H} = \boldsymbol{H}(\boldsymbol{x})$, where we have to decide on the *consistency* of the problem, see below.

The present implementation is based on the factorization of a matrix $\widetilde{\boldsymbol{H}}$, which is strictly full rank and is obtained by a Marquardt-like perturbation of $\boldsymbol{H}$, see Section 3.1. This is easy to compute, but the crucial decision about consistency is difficult. Therefore, the aim is to find $\boldsymbol{x}^*$ so that $F'(\boldsymbol{x}^*) = \boldsymbol{0}$. The algorithm is outlined below.

As a "safety valve" we allow at most $\max\{2m, 20\}$ iterations, and in Section 3 we give additional stopping criteria to handle exceptional cases. A more detailed presentation of the algorithm is presented in Section 3.5.

We refer to [10], [11], [13] or [12] for proof of finite convergence. The emphasis in this report is on computational aspects.

Basic Algorithm (2.2)

Get starting point $\boldsymbol{x}$
$\boldsymbol{r} := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
repeat
  $\boldsymbol{s} := \boldsymbol{s}(\boldsymbol{x})$; $\boldsymbol{W} := \boldsymbol{W}(\boldsymbol{x})$
  $\boldsymbol{g} := \boldsymbol{A}^T(\boldsymbol{W}\boldsymbol{r} + \gamma\boldsymbol{s}) - \gamma\mu\boldsymbol{L}^T(\boldsymbol{L}\boldsymbol{x}) - \gamma\boldsymbol{c}$
  if $\boldsymbol{g} \neq \boldsymbol{0}$ **then**
    Solve (2.1a) for $\boldsymbol{h}$;
    Find t^* by line search
    $\boldsymbol{x} := \boldsymbol{x} + t^*\boldsymbol{h}$; $\boldsymbol{r} := \boldsymbol{r} - t^*\boldsymbol{A}\boldsymbol{h}$
  end
until $\boldsymbol{g} = \boldsymbol{0}$

We can write the matrix $\boldsymbol{H}$ as

$$\boldsymbol{H} = \boldsymbol{B}^T\boldsymbol{B}, \quad \text{where} \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{W}\boldsymbol{A} \\ \sqrt{\gamma\mu}\boldsymbol{L} \end{bmatrix} . \tag{2.3}$$

This shows that $\boldsymbol{H}$ is symmetric and positive semidefinite. Let $\{(\lambda_j, \boldsymbol{v}_j)\}$ denote the eigensolutions,

$$\boldsymbol{H}\boldsymbol{v}_j = \lambda_j\boldsymbol{v}_j , \quad j = 1, \ldots, n , \tag{2.4a}$$

with the eigenvalues ordered so that

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p > 0, \quad \lambda_{p+1} = \cdots = \lambda_n = 0 . \tag{2.4b}$$

The number p is the *rank* of $\boldsymbol{H}$. If $p < n$, then we have to distinguish between two cases

1° The system (2.1) is consistent, i.e. $\boldsymbol{g} \in \text{range}(\boldsymbol{H})$. The solution is not unique, but we want $\boldsymbol{h} = \boldsymbol{h}_{\min}$, the minimum norm solution.

2° The system (2.1) is not consistent. We want $\boldsymbol{h} = \boldsymbol{h}_{\text{ort}}$, a vector in the direction of $\boldsymbol{g}_{\mathcal{N}}$, the projection of $\boldsymbol{g}$ onto the null space of $\boldsymbol{W}\boldsymbol{A}$.

The eigenvectors $\{\boldsymbol{v}_j\}$ form an orthonormal basis of $\mathbb{R}^n$, and we can write

$$\boldsymbol{g} = \textstyle\sum_{j=1}^n \alpha_j\boldsymbol{v}_j . \tag{2.5}$$

The Newton equations (2.1) are consistent if $\alpha_j = 0$ for $j = p{+}1, \ldots, n$ (especially, this is satisfied if $p = n$), and the minimum norm solution is

$$\boldsymbol{h}_{\min} = \sum_{j=1}^{p} \frac{\alpha_j}{\lambda_j}\boldsymbol{v}_j . \tag{2.6a}$$

For a non consistent problem we want

$$\boldsymbol{h}_{\text{ort}} = \eta\boldsymbol{g}_{\mathcal{N}} \quad \text{with} \quad \eta > 0, \quad \boldsymbol{g}_{\mathcal{N}} = \textstyle\sum_{j=p+1}^n \alpha_j\boldsymbol{v}_j . \tag{2.6b}$$

Note that $\boldsymbol{W}\boldsymbol{A}\boldsymbol{h}_{\text{ort}} = \boldsymbol{0}$ since the $\{\boldsymbol{v}_j\}$ are right singular vectors in the *singular value decomposition* (SVD) of the matrix $\boldsymbol{B}$, (2.3), and $\boldsymbol{B}\boldsymbol{v}_j = \sqrt{\lambda_j}\boldsymbol{u}_j = \boldsymbol{0}$ for $j > p$. Here, $\boldsymbol{u}_j$ is the jth left singular vector.

In Section 3.1 we discuss, how to compute (an approximation to the desired) $\boldsymbol{h}$ without actual knowledge of the eigensolutions.

3. Implementation

In the implementation we stress efficiency and accuracy. Detailed discussion of previous versions of the algorithm was given in [17] and [19]. For the sake of completeness we repeat some of it here together with new features.

As indicated in Sections 1–2, the size of $|\boldsymbol{r}(\boldsymbol{x})|$ plays a crucial role. The computed value, however, is afflicted with rounding errors,

$$\widetilde{\boldsymbol{r}}(\boldsymbol{x}) \equiv \mathsf{fl}\big(\boldsymbol{r}(\boldsymbol{x})\big) = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} .$$

Letting ε_M denote the machine accuracy (unit round-off), we use

$$\tau = 10\varepsilon_M\|\boldsymbol{A}\|_\infty\|\boldsymbol{x}\|_\infty \tag{3.1}$$

as a slightly optimistic upper bound on $\|\boldsymbol{e}\|_\infty$. This is initialized with the starting vector found as described in Section 3.4, and the value is updated whenever $10\varepsilon_M\|\boldsymbol{A}\|_\infty\|\boldsymbol{x}\|_\infty$ is outside the range $[\frac{1}{2}\tau, 2\tau]$. In the implementation we replace $\boldsymbol{s}$ and $\boldsymbol{W}$ by $\widetilde{\boldsymbol{s}}$ and $\widetilde{\boldsymbol{W}}$ defined by

$\lvert\widetilde{r}_i(\boldsymbol{x})\rvert \leq \gamma{-}\tau$	$\widetilde{s}_i(\boldsymbol{x}) = 0$	$\widetilde{w}_i(\boldsymbol{x}) = 1$
$\gamma{-}\tau < \lvert\widetilde{r}_i(\boldsymbol{x})\rvert \leq \gamma{+}\tau$	$\widetilde{s}_i(\boldsymbol{x}) = \frac{1}{4}\text{sign}(\widetilde{r}_i(\boldsymbol{x}))$	$\widetilde{w}_i(\boldsymbol{x}) = 1$
$\lvert\widetilde{r}_i(\boldsymbol{x})\rvert > \gamma{+}\tau$	$\widetilde{s}_i(\boldsymbol{x}) = \text{sign}(\widetilde{r}_i(\boldsymbol{x}))$	$\widetilde{w}_i(\boldsymbol{x}) = 0$

(3.2)

The vectors $\boldsymbol{x}$ and $\boldsymbol{x}{+}\boldsymbol{h}$ are considered to belong to the same subregion, $\Omega(\boldsymbol{x}{+}\boldsymbol{h}) = \Omega(\boldsymbol{x})$, if

$$\|\widetilde{\boldsymbol{s}}(\boldsymbol{x}{+}\boldsymbol{h}) - \widetilde{\boldsymbol{s}}(\boldsymbol{x})\|_\infty < 1 . \tag{3.3}$$

We say that $\boldsymbol{A}_{i,:}$ is *active* at $\boldsymbol{x}$ if $\widetilde{w}_i(\boldsymbol{x}) = 1$. The *active set* $\mathcal{A}(\boldsymbol{x})$ consists of these indices and includes the *maybe actives*, characterized by $|\widetilde{s}_i(\boldsymbol{x})| = \frac{1}{4}$. In the remainder of the report we use $\boldsymbol{r}$, $\boldsymbol{s}$, $\boldsymbol{W}$ for $\widetilde{\boldsymbol{r}}$, $\widetilde{\boldsymbol{s}}$, $\widetilde{\boldsymbol{W}}$.

The check for zero gradient is implemented as

$$\|\boldsymbol{g}\|_\infty \leq 10\varepsilon_M\gamma\left(\|\boldsymbol{A}\|_1 + \mu\|\boldsymbol{L}\|_1\|\boldsymbol{L}\|_\infty\|\boldsymbol{x}\|_\infty + \|\boldsymbol{c}\|_\infty\right) \ . \tag{3.4}$$

The right hand side is an estimate of accumulated rounding errors during the evaluation of $\boldsymbol{g}$, (2.1c).

3.1. Solution of the Newton Equations

The system (2.1),

$$\boldsymbol{H}\boldsymbol{h} = \boldsymbol{g} \ ,$$

is solved via a factorization (see Section 3.2 for its computation),

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{H} + \sigma^2\boldsymbol{I} \equiv \widetilde{\boldsymbol{H}} \ , \tag{3.5a}$$

where $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ is upper triangular and $\sigma \geq 0$ is chosen so that $\widetilde{\boldsymbol{H}}$ is significantly positive definite, taking into account the unavoidable rounding errors. These are of the order of magnitude $\lambda_1\varepsilon_M$, and the choice of σ depends on $\boldsymbol{A}$, γ, μ and $\boldsymbol{L}$.

If $\mu = 0$, then we let

$$\sigma^2 = \sigma_0^2 \equiv 100\varepsilon_M\|\boldsymbol{A}\|_1\|\boldsymbol{A}\|_\infty \ . \tag{3.5b}$$

The product $\|\boldsymbol{A}\|_1\|\boldsymbol{A}\|_\infty$ is easily seen to be an upper bound on λ_1, the largest eigenvalue of $\boldsymbol{H} = \boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}$.

If $\mu > 0$ and $\boldsymbol{L} = \boldsymbol{I}$, then the eigenvalues of $\boldsymbol{H}$ are $\lambda_j(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}) + \gamma\mu$, and if $\gamma\mu$ is sufficiently large, then we do not need to modify $\boldsymbol{H}$,

$$\textbf{if}\ \ \gamma\mu \geq \sigma_0^2\ \ \textbf{then}\ \ \sigma := 0 \ . \tag{3.5c}$$

Finally, if $\mu > 0$ and $\boldsymbol{L} \neq \boldsymbol{I}$, then we can similarly take

$$\sigma^2 = 100\varepsilon_M\|\boldsymbol{B}\|_1\|\boldsymbol{B}\|_\infty \ ,$$

where $\boldsymbol{B}$ is defined in (2.3). It follows that

$$\|\boldsymbol{B}\|_1 \leq \beta \equiv \|\boldsymbol{A}\|_1 + \sqrt{\gamma\mu}\|\boldsymbol{L}\|_1 \ , \quad \|\boldsymbol{B}\|_\infty = \max\{\|\boldsymbol{A}\|_\infty, \sqrt{\gamma\mu}\|\boldsymbol{L}\|_\infty\} \ ,$$

and we use

$$\begin{array}{ll}\textbf{if}\ \ \gamma\mu\|\boldsymbol{L}\|_1\|\boldsymbol{L}\|_\infty \geq \sigma_0^2\ \textbf{then} & \sigma := 10\sqrt{\varepsilon_M\beta\|\boldsymbol{B}\|_\infty} \\ \qquad\qquad\qquad\qquad\quad\ \textbf{else} & \sigma := \sigma_0\end{array} \tag{3.5d}$$

The logical variable

$$\mathcal{Z} := (\min\{|r_{ii}|\} < 10\sigma_0) \tag{3.6}$$

is used to distinguish between the cases $p < n$ ($\mathcal{Z} = \textbf{true}$) and full rank ($\mathcal{Z} = \textbf{false}$).

Given $\boldsymbol{R}$ and the right hand side $\boldsymbol{g}$, the solution to $\widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{h}} = \boldsymbol{g}$ is found by solving the two triangular systems

$$\boldsymbol{R}^T\boldsymbol{y} = \boldsymbol{g}, \qquad \boldsymbol{R}\widetilde{\boldsymbol{h}} = \boldsymbol{y} \tag{3.7}$$

(by forward and back substitution, respectively). With $\boldsymbol{g}$ given by (2.5) we see that

$$\widetilde{\boldsymbol{h}} = \sum_{j=1}^{n} \frac{\alpha_j}{\lambda_j + \sigma^2}\boldsymbol{v}_j \ . \tag{3.8}$$

In the $\boxed{\text{case } p = n}$ the true solution to the Newton equations and the error are

$$\boldsymbol{h} = \sum_{j=1}^{n} \frac{\alpha_j}{\lambda_j}\boldsymbol{v}_j \ , \qquad \boldsymbol{h} - \widetilde{\boldsymbol{h}} = \sigma^2\sum_{j=1}^{n} \frac{\alpha_j}{\lambda_j(\lambda_j + \sigma^2)}\boldsymbol{v}_j \ . \tag{3.9}$$

For comparison, if we solve the problem $\boldsymbol{H}\boldsymbol{h} = \boldsymbol{g}$ without the diagonal modification in (3.5), then we can expect the *computed solution* $\overline{\boldsymbol{h}}$ to have an error of the order of magnitude

$$\|\boldsymbol{h} - \overline{\boldsymbol{h}}\|_2 \sim \|\boldsymbol{H}^{-1}\|_2\,\varepsilon_M\,\|\boldsymbol{H}\|_2\,\|\overline{\boldsymbol{h}}\|_2 = \varepsilon_M\,\frac{\lambda_1}{\lambda_n}\,\|\overline{\boldsymbol{h}}\|_2 \ .$$

With σ given by (3.5b) we see from (3.9) that $\|\boldsymbol{h} - \widetilde{\boldsymbol{h}}\|_2$ is of the same order of magnitude.

We improve the accuracy by one step of *iterative refinement* as in the "corrected seminormal equations", [2, Section 6.6.5],

$$\begin{aligned}\boldsymbol{u} &:= \boldsymbol{g} - (\boldsymbol{W}\boldsymbol{A})^T(\boldsymbol{W}\boldsymbol{A}\widetilde{\boldsymbol{h}}) - \gamma\mu\boldsymbol{L}^T(\boldsymbol{L}\widetilde{\boldsymbol{h}}) \ , \\ \widehat{\boldsymbol{h}} &:= \widetilde{\boldsymbol{h}} + \boldsymbol{R}^{-1}(\boldsymbol{R}^{-T}\boldsymbol{u}) \ .\end{aligned} \tag{3.10a}$$

In the absence of rounding errors we find

$$\boldsymbol{u} = \boldsymbol{g} - \boldsymbol{H}\widetilde{\boldsymbol{h}} = \sum_{j=1}^{n} \left(1 - \frac{\lambda_j}{\lambda_j + \sigma^2}\right) \alpha_j \boldsymbol{v}_j = \sigma^2 \widetilde{\boldsymbol{h}} \ , \tag{3.10b}$$

$$\widehat{\boldsymbol{h}} = \sum_{j=1}^{n} \frac{(\lambda_j + 2\sigma^2)\alpha_j}{(\lambda_j+\sigma^2)^2}\, \boldsymbol{v}_j, \quad \boldsymbol{h} - \widehat{\boldsymbol{h}} = \sum_{j=1}^{n} \frac{\sigma^4 \alpha_j}{\lambda_j (\lambda_j+\sigma^2)^2}\, \boldsymbol{v}_j \ . \tag{3.10c}$$

The relative error is of the order of magnitude $(\varepsilon_M \lambda_1/\lambda_n)^2$.

In the $\boxed{\text{case } p<n}$ the algorithm returns $\widehat{\boldsymbol{h}}$ computed to match the requirements stated at the end of Section 2.

$$\begin{aligned} &\boldsymbol{v} := \boldsymbol{R}^{-1}(\boldsymbol{R}^{-T}\widetilde{\boldsymbol{h}}); \quad \boldsymbol{z} := \widetilde{\boldsymbol{h}} + \sigma^2 \boldsymbol{v} \\ &\boldsymbol{w} := (\boldsymbol{WA})^T(\boldsymbol{WAz}) + \gamma\mu \boldsymbol{L}^T(\boldsymbol{Lz}) \\ &\textbf{if } \|\boldsymbol{g} - \boldsymbol{w}\|_2 \le \sqrt{\varepsilon_M}\|\boldsymbol{g}\|_2 \textbf{ then } \widehat{\boldsymbol{h}} := \boldsymbol{z} \textbf{ else } \widehat{\boldsymbol{h}} := \frac{1}{\|\boldsymbol{v}\|_2}\boldsymbol{v} \end{aligned} \tag{3.11}$$

By comparison with (3.10) we see that the computation of $\boldsymbol{v}$ and $\boldsymbol{z}$ correspond to a step of iterative refinement (but we shall see that it improves only that component of $\boldsymbol{h}$, which is in the range of $\boldsymbol{H}$). The vector $\boldsymbol{w}$ is the mapping $\boldsymbol{w} = \boldsymbol{Hz}$, and $\boldsymbol{g} - \boldsymbol{w} \simeq \boldsymbol{g}_{\mathcal{N}}$. In terms of the eigensolutions we write (2.5) in the form

$$\boldsymbol{g} = \textstyle\sum_{j=1}^{p} \alpha_j \boldsymbol{v}_j + \boldsymbol{g}_{\mathcal{N}} \ , \tag{3.12a}$$

cf. (2.6b), and we see from (3.8) that

$$\widetilde{\boldsymbol{h}} = \sum_{j=1}^{p} \frac{\alpha_j}{\lambda_j + \sigma^2}\, \boldsymbol{v}_j + \frac{1}{\sigma^2}\, \boldsymbol{g}_{\mathcal{N}} \ . \tag{3.12b}$$

For a consistent problem $\boldsymbol{g}_{\mathcal{N}} = \boldsymbol{0}$, but rounding errors are almost sure to give a significant component in the null space. This is enhanced in $\boldsymbol{v}$,

$$\boldsymbol{v} = \sum_{j=1}^{p} \frac{\alpha_j}{(\lambda_j + \sigma^2)^2}\, \boldsymbol{v}_j + \frac{1}{\sigma^4} \boldsymbol{g}_{\mathcal{N}}. \tag{3.12c}$$

Further, we find

$$\boldsymbol{z} = \sum_{j=1}^{p} \frac{(\lambda_j + 2\sigma^2)\alpha_j}{(\lambda_j+\sigma^2)^2}\, \boldsymbol{v}_j + \frac{2}{\sigma^2}\, \boldsymbol{g}_{\mathcal{N}}, \quad \boldsymbol{g} - \boldsymbol{w} = \sum_{j=1}^{p} \frac{\sigma^4 \alpha_j}{(\lambda_j+\sigma^2)^2}\, \boldsymbol{v}_j + \boldsymbol{g}_{\mathcal{N}} \ . \tag{3.12d}$$

The norm of the first constribution to $\boldsymbol{g} - \boldsymbol{w}$ has an upper bound of the order of magnitude $(\varepsilon_M \lambda_1/\lambda_n)^2 \|\boldsymbol{g}\|_2$, implying that we get a good estimate of $\boldsymbol{g}_{\mathcal{N}}$. If this is considerable, then (3.12c) shows that $\boldsymbol{v}$ is dominated by its component in the null space of $\boldsymbol{H}$, and $\widehat{\boldsymbol{h}}$ is close to $\boldsymbol{h}_{\mathrm{ort}}$ defined in (2.6b).

3.2. Factorization

According to (2.3) and (3.5a) we can write $\widetilde{\boldsymbol{H}}$, in the form

$$\widetilde{\boldsymbol{H}} = \boldsymbol{C}^T\boldsymbol{C}, \quad \text{where} \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{A}_{\mathcal{A}} \\ \sqrt{\gamma\mu}\boldsymbol{L} \\ \sigma\boldsymbol{I} \end{bmatrix} . \tag{3.13}$$

Here, $\boldsymbol{A}_{\mathcal{A}}$ denotes the "active part of $\boldsymbol{A}$, consisting of the rows $\boldsymbol{A}_{i,:}$ with $i \in \mathcal{A}(\boldsymbol{x})$. This is empty if $\#(\mathcal{A}(\boldsymbol{x})) = 0$. Further, if $\mu = 0$, then the contribution with $\boldsymbol{L}$ disappears, and if $\boldsymbol{L} = \boldsymbol{I}$, then the two contributions $\sqrt{\gamma\mu}\boldsymbol{L}$ and $\sigma\boldsymbol{I}$ mingle into $\eta\boldsymbol{I}$ with $\eta = \max\{\sqrt{\gamma\mu}, \sigma\}$, cf. (3.5b–c).

From one iteration step to the next the active set changes. Let

$$\mathcal{A}(\boldsymbol{x}_{\mathrm{new}}) = \mathcal{A}(\boldsymbol{x}_{\mathrm{old}}) + \mathcal{E} - \mathcal{L} \ ,$$

where $\mathcal{E}$ and $\mathcal{L}$ hold the indices that respectively enter and leave the active set, and let $\boldsymbol{A}_{\mathcal{E}}$ and $\boldsymbol{A}_{\mathcal{L}}$ denote the corresponding submatrices of $\boldsymbol{A}$. Then

$$\begin{aligned} \widetilde{\boldsymbol{H}}_{\mathrm{new}} &= \widetilde{\boldsymbol{H}}_{\mathrm{old}} + \boldsymbol{A}_{\mathcal{E}}^T \boldsymbol{A}_{\mathcal{E}} - \boldsymbol{A}_{\mathcal{L}}^T \boldsymbol{A}_{\mathcal{L}} \\ &= \boldsymbol{R}^T\boldsymbol{R} + \boldsymbol{A}_{\mathcal{E}}^T \boldsymbol{A}_{\mathcal{E}} - \boldsymbol{A}_{\mathcal{L}}^T \boldsymbol{A}_{\mathcal{L}} \ , \end{aligned} \tag{3.14}$$

where $\boldsymbol{R}$ is the old factor. Let $\widehat{\boldsymbol{R}}$ denote the new factor in the case where $\mathcal{L} = \emptyset$. We find it by *orthogonal transformation*

$$\begin{bmatrix} \widehat{\boldsymbol{R}} \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{A}_{\mathcal{E}} \end{bmatrix} \ , \tag{3.15}$$

performed as a series of *Householder transformations.*[1)] The triangular form of $\boldsymbol{R}$ is exploited so that the kth transformation (giving $\widehat{\boldsymbol{R}}_{k,:}$) involves only the kth row in $\boldsymbol{R}$ and columns $k:n$ in the partly transformed $\boldsymbol{A}_{\mathcal{E}}$, which we denote $\boldsymbol{B}$. Further, let

[1)] See e.g. [2, Section 2.3.1].

$$\boldsymbol{y} = \begin{bmatrix} r_{kk} \\ \boldsymbol{B}_{:,k} \end{bmatrix}, \quad \widehat{\boldsymbol{y}} = \begin{bmatrix} \widehat{r}_{kk} \\ \boldsymbol{0} \end{bmatrix} \quad \text{with} \quad \widehat{r}_{kk} = -\mathrm{sign}(r_{kk})\|\boldsymbol{y}\|_2 \ . \tag{3.16a}$$

Then

$$\widehat{\boldsymbol{y}} = \boldsymbol{Q}_k \boldsymbol{y} \ , \tag{3.16b}$$

where $\boldsymbol{Q}_k$ is the orthogonal matrix

$$\boldsymbol{Q}_k = \boldsymbol{I} + \beta \boldsymbol{w}\boldsymbol{w}^T \quad \text{with} \quad \boldsymbol{w} = \boldsymbol{y} - \widehat{\boldsymbol{y}}, \quad \beta = \frac{1}{\widehat{r}_{kk} w_1} \ . \tag{3.16c}$$

Columns $k+1, \ldots, n$ in $\boldsymbol{R}_{k,:}$ and $\boldsymbol{B}$ are changed to

$$(\boldsymbol{I} + \beta \boldsymbol{w}\boldsymbol{w}^T) \begin{bmatrix} \boldsymbol{R}_{k,k+1:n} \\ \boldsymbol{B}_{:,k+1:n} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{k,k+1:n} \\ \boldsymbol{B}_{:,k+1:n} \end{bmatrix} + \boldsymbol{w}\boldsymbol{u}^T \ , \tag{3.16d}$$

where $\boldsymbol{u}^T$ is the row vector

$$\boldsymbol{u}^T = \beta \boldsymbol{w}^T \begin{bmatrix} \boldsymbol{R}_{k,k+1:n} \\ \boldsymbol{B}_{:,k+1:n} \end{bmatrix} = \beta \left(w_1 \boldsymbol{R}_{k,k+1:n} + \boldsymbol{B}^T_{:,k} \boldsymbol{B}_{:,k+1:n} \right) \ . \tag{3.16e}$$

The full transformation can be expressed in the following form, where $\widehat{\boldsymbol{R}}$ overwrites $\boldsymbol{R}$ and $\boldsymbol{B}$ denotes $\boldsymbol{A}_{\mathcal{E}}$. This algorithm involves about $2\#(\mathcal{E})n^2$ flops.

Algorithm Update (3.17)
  for $k := 1, \ldots, n$
    $v := r_{k,k}; \quad r_{k,k} := -\mathrm{sign}(v)\sqrt{v^2 + \|\boldsymbol{B}_{:,k}\|_2^2}$
    $w_1 := v - r_{k,k}; \quad \beta := 1/(r_{k,k} w_1)$
    $\boldsymbol{u} := \beta(w_1 \boldsymbol{R}_{k,k+1:n} + \boldsymbol{B}^T_{:,k} \boldsymbol{B}_{:,k+1:n})$
    $\boldsymbol{R}_{k,k+1:n} := \boldsymbol{R}_{k,k+1:n} + w_1 \boldsymbol{u}$
    $\boldsymbol{B}_{:,k+1:n} := \boldsymbol{B}_{:,k+1:n} + \boldsymbol{B}_{:,k} \boldsymbol{u}$
  end

The initial factorization or a *refactorization* is done by means of (3.17):

Algorithm Factorize (3.18)
  $\boldsymbol{R} := \eta \boldsymbol{I}$
  if $\mu > 0$ **and** $\boldsymbol{L} \neq \boldsymbol{I}$ **then** $\boldsymbol{R} := \mathit{Update}(\boldsymbol{R}, \sqrt{\gamma\mu}\boldsymbol{L})$
  $\boldsymbol{R} := \mathit{Update}(\boldsymbol{R}, \boldsymbol{A}_{\mathcal{A}})$

The cost of a (re)factorization is about $2(\#(\mathcal{A}) + q_{\mathrm{L}})n^2$ flops, where $q_{\mathrm{L}} = 0$ if $\mu = 0$ or $\boldsymbol{L} = \boldsymbol{I}$. Otherwise, q_{L} is the number of rows in $\boldsymbol{L}$.

Now, consider the *downdating* problem

$$\widehat{\boldsymbol{R}}^T \widehat{\boldsymbol{R}} = \boldsymbol{R}^T \boldsymbol{R} - \boldsymbol{A}_{\mathcal{L}}^T \boldsymbol{A}_{\mathcal{L}} \ .$$

This is equivalent with $\boldsymbol{R}^T \boldsymbol{R} = \widehat{\boldsymbol{R}}^T \widehat{\boldsymbol{R}} + \boldsymbol{A}_{\mathcal{L}}^T \boldsymbol{A}_{\mathcal{L}}$, and can be found by orthogonal transformation

$$\begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q} \begin{bmatrix} \widehat{\boldsymbol{R}} \\ \boldsymbol{A}_{\mathcal{L}} \end{bmatrix} \ .$$

Note that the unknown matrix appears on the wrong side of the equation. This is handled by using (3.16) "backwards": Let $\boldsymbol{B}$ denote the partly transformed $\boldsymbol{A}_{\mathcal{L}}$, then (3.16a) is replaced by

$$\boldsymbol{y} = \begin{bmatrix} r_{kk} \\ \boldsymbol{0} \end{bmatrix}, \quad \widehat{\boldsymbol{y}} = \begin{bmatrix} \widehat{r}_{kk} \\ \boldsymbol{B}_{:,k} \end{bmatrix}, \quad \text{with} \quad r_{kk}^2 = \widehat{r}_{kk}^2 + \|\boldsymbol{B}_{:,k}\|_2^2 \ . \tag{3.19a}$$

To reduce cancellation error we compute $a = \widehat{r}_{kk}^2$ as

$$a = (r_{kk} + \|\boldsymbol{B}_{:,k}\|_2)(r_{kk} - \|\boldsymbol{B}_{:,k}\|_2) \ , \tag{3.19b}$$

and with σ_0 given by (3.5b) and η discussed above, we modify a if it is too small,

$$\textbf{if} \ \ a \leq 0.01\sigma_0^2 \ \ \textbf{then} \ \ a := \eta^2 \ . \tag{3.19c}$$

Next,

$$\widehat{r}_{kk} = -\mathrm{sign}(r_{kk})\sqrt{a} \ , \quad \boldsymbol{w} = \widehat{\boldsymbol{y}} - \boldsymbol{y} \ , \quad \beta = \frac{1}{r_{kk} w_1} \ , \tag{3.19d}$$

and the remaining elements in the kth row of $\widehat{\boldsymbol{R}}$ are found from the relation

$$\boldsymbol{R}_{k,k+1:n} = \widehat{\boldsymbol{R}}_{k,k+1:n} + w_1 \boldsymbol{u}^T \ , \tag{3.19e}$$

where

$$\boldsymbol{u}^T = \beta \left(w_1 \widehat{\boldsymbol{R}}_{k,k+1:n} + \boldsymbol{B}^T_{:,k} \boldsymbol{B}_{:,k+1:n} \right) \ . \tag{3.19f}$$

This implies

$$\widehat{\boldsymbol{R}}_{k,k+1:n} = \frac{1}{w_1 + r_{kk}} (r_{kk} \boldsymbol{R}_{k,k+1:n} - \boldsymbol{B}^T_{:,k} \boldsymbol{B}_{:,k+1:n}) \ , \tag{3.19g}$$

where we have used (3.19d): $\beta w_1 = 1/r_{kk}$.

Now we can formulate the dowdating algorithm, where $\widehat{\boldsymbol{R}}$ overwrites $\boldsymbol{R}$ and $\boldsymbol{B}$ denotes $\boldsymbol{A}_{\mathcal{L}}$. The full transformation involves about $2\#(\mathcal{L})n^2$ flops.

Algorithm Downdate (3.20)

$$
\begin{array}{l}
\textbf{for}\ \ k := 1,\ldots,n \\
\quad v := r_{k,k};\quad N := \|\boldsymbol{B}_{:,k}\|_2;\quad a := (v+N)(v-N) \\
\quad \textbf{if}\ \ a \le 0.01\sigma_0^2\ \ \textbf{then}\ a := \eta^2; \\
\quad r_{k,k} := -\mathrm{sign}(v)\sqrt{a} \\
\quad w_1 := r_{k,k} - v;\quad \boldsymbol{u} := \boldsymbol{B}_{:,k}^T \boldsymbol{B}_{:,k+1:n} \\
\quad \boldsymbol{R}_{k,k+1:n} := (v\boldsymbol{R}_{k,k+1:n} - \boldsymbol{u})/(w_1+v) \\
\quad \boldsymbol{u} := (\boldsymbol{u} + w_1\boldsymbol{R}_{k,k+1:n})/(r_{k,k}w_1) \\
\quad \boldsymbol{B}_{:,k+1:n} := \boldsymbol{B}_{:,k+1:n} + \boldsymbol{B}_{:,k}\boldsymbol{u} \\
\textbf{end}
\end{array}
$$

Typically, the change in active set involves both $\mathcal{L}$ and $\mathcal{E}$ being non empty, and the new $\boldsymbol{R}$ is found by

$$
\begin{array}{l}
\textbf{if}\ \ (\#(\mathcal{L}) + \#(\mathcal{E}) > \#(\mathcal{A}) + q_{\mathrm{L}})\ \ \textbf{or}\ \ (\nu > n)\ \ \textbf{then} \\
\quad \boldsymbol{R} := \mathit{Factorize}(\boldsymbol{A}_{\mathcal{A}}) \\
\textbf{else} \\
\quad \boldsymbol{R} := \mathit{Update}(\boldsymbol{R}, \boldsymbol{A}_{\mathcal{E}}), \qquad \boldsymbol{R} := \mathit{Downdate}(\boldsymbol{R}, \boldsymbol{A}_{\mathcal{L}})\ .
\end{array} \tag{3.21}
$$

The number ν is the accumulated values of $\#(\mathcal{L})$ since the last refactorization, and when this becomes large we do a "clean up" (a refactorization) similar to the common practice in Simplex codes.

3.3. Line Search

Given the current iterate $\boldsymbol{x}$ and the Newton step $\boldsymbol{h}$. Let

$$
\boldsymbol{y} = \boldsymbol{L}\boldsymbol{x}, \quad \boldsymbol{u} = \boldsymbol{A}\boldsymbol{h}, \quad \boldsymbol{v} = \boldsymbol{L}\boldsymbol{h}\ , \tag{3.22}
$$

and consider the function

$$
\varphi(t) \equiv \gamma F(\boldsymbol{x}+t\boldsymbol{h}) = c_0(t) - c_1(t)\cdot t + \tfrac{1}{2}c_2(t)\cdot t^2\ , \tag{3.23a}
$$

where

$$
\begin{aligned}
c_0(t) &= \tfrac{1}{2}\boldsymbol{r}^T\boldsymbol{W}\,\boldsymbol{r} + \gamma\left(\boldsymbol{s}^T(\boldsymbol{r} - \tfrac{1}{2}\gamma\boldsymbol{s}) + \tfrac{1}{2}\mu\boldsymbol{y}^T\boldsymbol{y} + \boldsymbol{c}^T\boldsymbol{x}\right)\ , \\
c_1(t) &= \boldsymbol{u}^T(\boldsymbol{W}\boldsymbol{r} + \gamma\boldsymbol{s}) - \gamma\mu\boldsymbol{v}^T\boldsymbol{y} - \gamma\boldsymbol{c}^T\boldsymbol{h}\ , \\
c_2(t) &= \boldsymbol{u}^T\boldsymbol{W}\boldsymbol{u} + \gamma\mu\boldsymbol{v}^T\boldsymbol{v}\ .
\end{aligned} \tag{3.23b}
$$

The coefficients $c_{0:2}(t)$ are piecewise constant. They change at the *kink values* $\{\kappa_j\}$, where one or more components of $\boldsymbol{r}(\boldsymbol{x}+t\boldsymbol{h}) = \boldsymbol{r}(\boldsymbol{x}) - t\boldsymbol{u}$ pass the value of γ or $-\gamma$.

First, the computed $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{h}$ is adjusted to keep only the significantly nonzero components,

$$
\textbf{if}\ \ |u_k| \le \varepsilon_M\|\boldsymbol{A}\|_\infty\|\boldsymbol{h}\|_\infty\ \ \textbf{then}\ \ u_k := 0 \quad (k=1,\ldots,m)\ . \tag{3.24}
$$

Next, if $w_k(\boldsymbol{x}) = 0$ and $s_k(\boldsymbol{x})u_k > 0$, this gives rise to two kink values, where k respectively enters and leaves the active set,

$$
\kappa_{\mathcal{E}}^{(k)} = \frac{r_k - \gamma s_k}{u_k}\ , \qquad \kappa_{\mathcal{L}}^{(k)} = \frac{r_k + \gamma s_k}{u_k}\ . \tag{3.25a}
$$

Finally, for $w_k(\boldsymbol{x}) = 1$ we have to distinguish between "maybe active" and "truly active". If, e.g. $|r_k - \gamma| < \tau$ and $u_k < 0$, then this index is considered as allready non active in connection with the line search, and we only have to consider

$$
\begin{array}{l}
\textbf{if}\ \ (|s_k| = \frac{1}{4}\ \textbf{and}\ s_k u_k > 0)\ \textbf{or}\ (s_k = 0\ \textbf{and}\ u_k \ne 0)\ \ \textbf{then} \\
\quad \kappa_{\mathcal{L}}^{(k)} = \dfrac{r_k + \gamma\,\mathrm{sign}(u_k)}{u_k}\ .
\end{array} \tag{3.25b}
$$

The kink values are ordered so that

$$
0 < \kappa_1 \le \kappa_2 \le \cdots \le \kappa_K\ . \tag{3.26}
$$

Note that the number of kink values is bounded: $K \le 2m$.

We seek t^*, the minimizer of φ. It satisfies $\varphi'(t^*) = 0$. The function φ' is a continuous, piecewise first order polynomial, which we shall write in the form

$$
\varphi'(t) = -c_{1,j} + c_{2,j}\cdot(t - \kappa_{j-1}) \quad \text{for} \quad \kappa_{j-1} < t \le \kappa_j \tag{3.27a}
$$

(with $\kappa_0 = 0$). The initial values are[2)]

$$
\begin{aligned}
c_{1,1} &= c_1(0) = \boldsymbol{g}^T\boldsymbol{h}\ , \\
c_{2,1} &= c_2(0) = \|\boldsymbol{W}\boldsymbol{u}\|_2^2 + \gamma\mu\|\boldsymbol{L}\boldsymbol{h}\|_2^2\ .
\end{aligned} \tag{3.27b}
$$

The computation of $\boldsymbol{h}$ ensures that $\boldsymbol{g}^T\boldsymbol{h} > 0$, so that $\varphi'(0) = -c_{1,1} < 0$. We increase j as long as $j < K$ and

$$
\varphi'(\kappa_j) = -c_{1,j} + c_{2,j}\cdot(\kappa_j - \kappa_{j-1}) \tag{3.27c}
$$

is negative, and use the updatings based on the continuity of φ' and on (3.23b)

[2)] It is easily veryfied that $c_{1,1} = c_{2,1}$ if the system (2.1) is consistent.

$$c_{1,j+1} = -\varphi'(\kappa_j)\ , \qquad c_{2,j+1} = c_{2,j} \pm u_k^2\ , \tag{3.27d}$$

where $+$ $(-)$ is used if k enters (leaves) the active set at κ_j.

If $\varphi'(\kappa_j) \geq 0$, then we have passed the minimum and return

$$t^* = \kappa_{j-1} + c_{1,j}/c_{2,j}\ . \tag{3.28a}$$

For a non convex problem we may run through all the kinks without passing a minimum. Then we return the t^*-value determined by

$$\begin{array}{llll} \textbf{if} & K > 1 & \textbf{then} & t^* = \frac{1}{2}(\kappa_{K-1} + \kappa_K) \\ \textbf{elseif} & K = 1 & \textbf{then} & t^* = 1.5\kappa_1 \\ \textbf{elseif} & c_{2,1} > 0 & \textbf{then} & t^* = c_{1,1}/c_{2,1} \\ & & \textbf{else} & t^* = -1 \end{array} \tag{3.28b}$$

The value $t^* = -1$ is obtained only if $\mu \boldsymbol{L}\boldsymbol{h} = \boldsymbol{0}$, $(\boldsymbol{A}\boldsymbol{h})_i = 0$ for all $i \in \mathcal{A}(\boldsymbol{x})$ and $K = 0$. The last condition is equivalent with

$$s_k u_k \leq 0 \quad \text{for all } k = 1, \ldots, m\ .$$

This implies, that for all $\alpha > 0$ we have $\boldsymbol{s}(\boldsymbol{x} + \alpha\boldsymbol{h}) = \boldsymbol{s}(\boldsymbol{x})$ and

$$F(\boldsymbol{x} + \alpha\boldsymbol{h}) = F(\boldsymbol{x}) - c_{1,1}\alpha \to -\infty \quad \text{for} \quad \alpha \to +\infty\ .$$

Thus, $t^* = -1$ flags an unbounded problem.

3.4. Starting Point

The algorithm is well suited for warm start. This is important in many applications, e.g. in nonlinear parameter estimation, where $\boldsymbol{x}$ is an increment vector with $\boldsymbol{x} = \boldsymbol{0}$ as the natural starting point.

In the algorithms of [11] and [13] we solve (1.1) for a series of successively reduced γ-values. Here it is worthwhile to use the sign vector and active set to get the initial $\boldsymbol{x}$ for algorithm (2.2). This is obtained from (2.1) by setting $\boldsymbol{x} = \boldsymbol{h}$, $\boldsymbol{r} = \boldsymbol{b}$, leading to

$$(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A} + \gamma\mu\boldsymbol{L}^T\boldsymbol{L})\,\boldsymbol{x} = \boldsymbol{A}^T(\boldsymbol{W}\boldsymbol{b} + \gamma\boldsymbol{s}) - \gamma\boldsymbol{c}\ . \tag{3.29}$$

In these cases $\mu = 0$, implying that $\widetilde{\boldsymbol{H}}$ in (3.5) is independent of γ, thus making it possible to use a "hot start": The system (3.29) can be solved by means of the factor $\boldsymbol{R}$ found in connection with solving the previous problem (with a larger γ-value).

The implementation allows for these variants of warm start. If no such information is available, then the starting point is found by assuming that the active set is full, $\mathcal{A} = \{1, \ldots, m\}$. Thus, the initial $\boldsymbol{x}$ is found by (3.29) with $\boldsymbol{W} = \boldsymbol{I}$, $\boldsymbol{s} = \boldsymbol{0}$. If $\mu = 0$ and $\boldsymbol{c} = \boldsymbol{0}$, then this starting point $\widehat{\boldsymbol{x}}$ is the least squares solution to the overdetermined system $\boldsymbol{A}\boldsymbol{x} \simeq \boldsymbol{b}$, and if $\gamma \geq \|\boldsymbol{r}(\widehat{\boldsymbol{x}})\|_\infty$, then $\widehat{\boldsymbol{x}} = \boldsymbol{x}^*$, the minimizer of (1.1).

3.5. The Algorithm. II

In this section we give the algorithm in more details than shown in (2.2). The vector $\boldsymbol{x}$ is the current iterate, $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$, $\boldsymbol{s}$ denotes $\widetilde{\boldsymbol{s}}(\boldsymbol{x})$ as defined in (3.2) and computed for the current $\boldsymbol{r}$, and $\boldsymbol{R}$ is the factorization corresponding to the sign vector $\boldsymbol{s}_F$. The iteration counter k is augmented by 1 with each solution of $\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{h} = \boldsymbol{g}$, and to avoid ill effects of the inherent instability of downdating we allow a refactorization if the iteration seems to be stuck close to the solution; the number ν is defined in connection with (3.21).

Algorithm PQmin (3.30)

```
k := 0;   more := true;   fail := false
Initialize x, r, s and R; cf. Section 3.4
while more and (k < max{20, 2m}) do
  if ||s_F − s||_∞ < 1 then
    r := b − Ax;   Recompute s
    refac := (ν > 0);   try := (||s_F − s||_∞ < 1)
  else refac := try := false
  Compute g
  if g = 0 then more := false
  else
    if refac then R := Factorize(A_A) else update R
    Find h (Section 3.1) and u = Ah, modified by (3.24)
    Find t* by line search, Section 3.3
    if t* > 0 then
      x := x + t*h;   r := r − t*u
      if try and (||s_F − s||_∞ < 1) then more := false
    else
      fail := true;   more := false
    end
  end
end
```

4. One Sided Huber

In some applications, see e.g. [1], [5], [18] and [20, Chapter 5], we want to reduce the contribution from positive components of the residual vector, and replace (1.1c) by

$$\widehat{\rho}(r) = \begin{cases} |r| - \frac{1}{2}\gamma & \text{for} \quad r < -\gamma \ , \\ \dfrac{1}{2\gamma} r^2 & \text{for} \quad -\gamma \le r \le 0 \ , \\ 0 & \text{for} \quad r > 0 \ . \end{cases} \tag{4.1a}$$

or

$$\widetilde{\rho}(r) = \begin{cases} \dfrac{1}{2\gamma} r^2 & \text{for} \quad r \le \gamma \ , \\ r - \frac{1}{2}\gamma & \text{for} \quad r > \gamma \ . \end{cases} \tag{4.1b}$$

These two choices are illustrated below, cf. Figure 1.1.

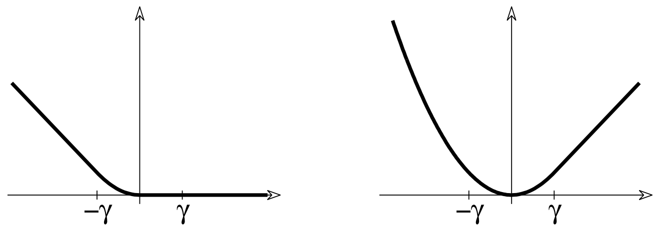

Figure 4.1. *One sided Huber functions.* $\widehat{\rho}(r)$ *(left) and* $\widetilde{\rho}(r)$ *(right)*

To ease the implementation in the case (4.1a) we introduce

$$\widehat{\boldsymbol{r}}(\boldsymbol{x}) = \boldsymbol{r}(\boldsymbol{x}) + \tfrac{1}{2}\gamma \boldsymbol{e} \ , \tag{4.2}$$

where $\boldsymbol{e} \in \mathbb{R}^m$ is the vector of all ones. Further, let $\boldsymbol{s}(\boldsymbol{x})$ and $\boldsymbol{W}(\boldsymbol{x})$ be defined as is (1.2),

$$\begin{array}{ll} \textbf{if} \ \ |\widehat{r}_i(\boldsymbol{x})| \le \frac{1}{2}\gamma \ \ \textbf{then} & s_i(\boldsymbol{x}) = 0, \quad w_i(\boldsymbol{x}) = 1 \\ \qquad\qquad\qquad\qquad \ \ \textbf{else} & s_i(\boldsymbol{x}) = \mathrm{sign}(\widehat{r}_i(\boldsymbol{x})), \quad w_i(\boldsymbol{x}) = 0 \ , \end{array} \tag{4.3a}$$

and let

$$\widehat{\boldsymbol{s}}(\boldsymbol{x}) = \tfrac{1}{2}(\boldsymbol{s}(\boldsymbol{x}) - \boldsymbol{e}) \ . \tag{4.3b}$$

Then the objective function is

$$\begin{aligned} \widehat{F}(\boldsymbol{x}) &= \sum_{i=1}^{m} \widehat{\rho}(r_i(\boldsymbol{x})) + \tfrac{1}{2}\mu \|\boldsymbol{L}\boldsymbol{x}\|_2^2 + \boldsymbol{c}^T\boldsymbol{x} \\ &= \tfrac{1}{2\gamma} \boldsymbol{r}^T \boldsymbol{W} \boldsymbol{r} + \widehat{\boldsymbol{s}}^T (\boldsymbol{r} - \tfrac{1}{2}\gamma\widehat{\boldsymbol{s}}) + \tfrac{1}{2}\mu \boldsymbol{x}^T \boldsymbol{L}^T \boldsymbol{L}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \ , \end{aligned} \tag{4.4a}$$

with the gradient

$$\widehat{F}'(\boldsymbol{x}) = -\boldsymbol{A}^T \left(\frac{1}{\gamma} \boldsymbol{W}(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x}) + \widehat{\boldsymbol{s}}(\boldsymbol{x}) \right) + \mu \boldsymbol{L}^T\boldsymbol{L}\boldsymbol{x} + \boldsymbol{c} \ . \tag{4.4b}$$

This is similar with (1.3–4), and the algorithm is as described in Sections 2 – 3, except that $\boldsymbol{s}$ is replaced by $\widehat{\boldsymbol{s}}$ in the computation (2.1c) of $\boldsymbol{g}$ and that (γ, r_k) is replaced by $(\frac{\gamma}{2}, \widehat{r}_k)$ in the computation (3.24) of the kink points.

In the case (4.1b) the sign vector $\boldsymbol{s}$ and activity matrix $\boldsymbol{W}$ is computed by

$$\begin{array}{ll} \textbf{if} \ \ r_i(\boldsymbol{x}) \le \gamma \ \ \textbf{then} & s_i(\boldsymbol{x}) = 0, \quad w_i(\boldsymbol{x}) = 1 \\ \qquad\qquad\quad \ \ \textbf{else} & s_i(\boldsymbol{x}) = 1, \quad w_i(\boldsymbol{x}) = 0 \ , \end{array} \tag{4.5}$$

and the only other modification in the algorithm is in the computation of kink values, where (3.25a–b) simplify to

$$\textbf{if} \ \ (w_k = 0) \ \textbf{and} \ (u_k > 0) \ \ \textbf{then} \ \ \kappa_{\mathcal{E}}^{(k)} = \frac{r_k - \gamma}{u_k} \ , \tag{4.6a}$$

$$\textbf{if} \ \ (s_k = 0) \ \textbf{and} \ (u_k < 0) \ \ \textbf{then} \ \ \kappa_{\mathcal{L}}^{(k)} = \frac{r_k - \gamma}{u_k} \ . \tag{4.6b}$$

5. Testing

The algorithm has been tested on a large number of problems generated as described in Section 5.1. In Section 5.2 we give results for the overall performance and in Section 5.3 we look at a special application.

5.1. Test Problems

The generator for test problems is based on ideas from [10] and [15]. The input is m, n and a parameter vector $\boldsymbol{p}$ with 8 elements

p_1 : γ. ($p_1 > 0$).

p_2 : μ.

p_3 : Choice of Huber function.
$p_3 = 1$: (4.1a), $p_3 \; := 2$: (4.1b), otherwise: (1.1c).

p_4 : Choice of $\boldsymbol{L}$ (ignored if $\mu \le 0$).
$p_4 = 1$: $\boldsymbol{L} = \boldsymbol{I}$,
$p_4 = 2$: $\boldsymbol{L} \in \mathbb{R}^{(n-1)\times n}$ with $\ell_{i,i} = 1$ and $\ell_{i,i+1} = -1$ as the only nonzero elements,
$p_4 = 3$: $\boldsymbol{L} \in \mathbb{R}^{(n-2)\times n}$ with $\ell_{i,i} = \ell_{i,i+2} = 1$ and $\ell_{i,i+1} = -2$ as the only nonzero elements.

p_5 : Type of $\boldsymbol{c}$. $\boldsymbol{c} = \boldsymbol{0}$ if $p_5 = 0$.

p_6 : The desired condition number of $\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}$ is 10^{p_6} $(p_6 \ge 0)$.

p_7 : "Wildness" of non active residuals.

p_8 : The active set is $\mathcal{A}(\boldsymbol{x}^*) = \{1, 2, \ldots, q\}$, where $q = \min\{m-1, \lceil p_8 n \rceil\}$.

In all cases the solution is given by

$$\boldsymbol{x}^* = \begin{bmatrix} 1 & -2 & 3 & \ldots & (-1)^{n-1}n \end{bmatrix}^T . \tag{5.1}$$

In the description we let $\mho$ and $\aleph$ denote random numbers, which are respectively from a uniform $[0,1]$ and a normal $(0,1)$ distribution, and define

$$q = \min\{m-1, \lceil p_8 n \rceil\}, \qquad k = \min\{q, n\} . \tag{5.2}$$

The active part of $\boldsymbol{A}$ is computed as

$$\boldsymbol{A}_{1:q,:} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T , \tag{5.3a}$$

where $\boldsymbol{U} \in \mathbb{R}^{q\times k}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times k}$ have orthonormal columns and

$$\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_k) \quad \text{with} \quad \log_{10}\sigma_j = -p_6 \cdot \frac{j-1}{2(k-1)} . \tag{5.3b}$$

This means that $\|\boldsymbol{W}\boldsymbol{A}\|_2 = 1$, $\kappa_2(\boldsymbol{W}\boldsymbol{A}) = 10^{p_6/2}$, $\kappa_2(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}) = 10^{p_6}$.

The generator is implemented with Householder matrices for $\boldsymbol{U}$ and $\boldsymbol{V}$: Let $\boldsymbol{u} \in \mathbb{R}^q$ and $\boldsymbol{v} \in \mathbb{R}^n$ be generated with $\aleph$-random elements, followed by normalization,

$$\boldsymbol{u} := \frac{\sqrt{2}}{\|\boldsymbol{u}\|_2}\,\boldsymbol{u}, \qquad \boldsymbol{v} := \frac{\sqrt{2}}{\|\boldsymbol{v}\|_2}\,\boldsymbol{v} . \tag{5.3c}$$

Then the matrices $\boldsymbol{I} - \boldsymbol{u}\boldsymbol{u}^T$ and $\boldsymbol{I} - \boldsymbol{v}\boldsymbol{v}^T$ are orthogonal, and it is easily verified that the nonzero rows in $\boldsymbol{W}\boldsymbol{A}$ can be computed by

$$\begin{array}{l}
\textbf{if}\;\; q \ge n \;\; \textbf{then} \\
\quad \boldsymbol{w} := (\boldsymbol{v}^T\boldsymbol{\Sigma}u_{1:n})\boldsymbol{v} - \boldsymbol{\Sigma}u_{1:n} , \\
\quad \boldsymbol{A}_{1:n,:} := \boldsymbol{\Sigma} - (\boldsymbol{\Sigma}\boldsymbol{v})\boldsymbol{v}^T + \boldsymbol{u}_{1:n}\boldsymbol{w}^T , \\
\quad \boldsymbol{A}_{n+1:q,:} := \boldsymbol{u}_{n+1:q}\boldsymbol{w}^T , \\
\textbf{else} \\
\quad \boldsymbol{w} := (\boldsymbol{u}^T\boldsymbol{\Sigma}v_{1:q})\boldsymbol{u} - \boldsymbol{\Sigma}v_{1:q} , \\
\quad \boldsymbol{A}_{1:q,1:q} := \boldsymbol{\Sigma} - \boldsymbol{u}(\boldsymbol{\Sigma}\boldsymbol{u})^T + \boldsymbol{w}\,\boldsymbol{v}_{1:q}^T , \\
\quad \boldsymbol{A}_{1:q,q+1:n} := \boldsymbol{w}\,\boldsymbol{v}_{q+1:n}^T . \\
\textbf{end}
\end{array} \tag{5.3d}$$

The remaining rows of $\boldsymbol{A}$ are generated with $\aleph$-random elements, but may be modified, see (5.7).

Next, we ensure that (1.4b) is satisfied:

$$\boldsymbol{A}^T\boldsymbol{f} - \mu\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{x}^* - \boldsymbol{c} = \boldsymbol{0} \quad \text{where} \quad \boldsymbol{f} = \frac{1}{\gamma}\boldsymbol{W}\boldsymbol{r} + \boldsymbol{s} . \tag{5.4}$$

For a *two sided Huber problem* the vector $\boldsymbol{f}$ is generated by

$$\boldsymbol{f} := \boldsymbol{e} - 2\boldsymbol{u}; \qquad \boldsymbol{f}_{q+1:m} := \operatorname{sign}(\boldsymbol{f}_{q+1:m}) , \tag{5.5}$$

where $\boldsymbol{e}$ is the vector of all ones and $\boldsymbol{u}$ has $\mho$-random elements, i.e. $f_i \in [-1, 1]$. If $p_5 \ne 0$, then we compute $\boldsymbol{c}$ by

$$\boldsymbol{c} := \boldsymbol{A}^T\boldsymbol{f} - \mu\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{x}^* . \tag{5.6}$$

Otherwise, $\boldsymbol{c} = \boldsymbol{0}$ and the last row in $\boldsymbol{A}$ is modified,

$$\boldsymbol{A}_{m,:}^T := f_m\left(\mu\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{x}^* - \boldsymbol{A}_{1:m-1,:}^T\boldsymbol{f}_{1:m-1}\right) . \tag{5.7}$$

Finally,

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}^* + \boldsymbol{r}(\boldsymbol{x}^*) , \tag{5.8a}$$

where

$$r_i(\boldsymbol{x}^*) = \begin{cases} \gamma f_i & \text{for} \;\; i = 1, \ldots, q \\ \gamma f_i(1 + \eta + p_7\mho_i) & \text{for} \;\; i = q+1, \ldots, m . \end{cases} \tag{5.8b}$$

The condition $|r_i(\boldsymbol{x}^*)| > \gamma+\tau$ for $i > q$ (cf. (3.1)) is satisfied if we take

$$\eta = \frac{11}{\gamma}\varepsilon_M \|\boldsymbol{A}\|_\infty \|\boldsymbol{x}^*\|_\infty \ . \tag{5.8c}$$

To get a *one sided Huber problem* defined by (4.1a) we replace (5.5) and (5.8b) by

$$\boldsymbol{f} := -\boldsymbol{u}; \qquad \boldsymbol{f}_{q+1:m} := \mathrm{round}(\boldsymbol{f}_{q+1:m}) \ , \tag{5.9a}$$

and

$$r_i(\boldsymbol{x}^*) = \begin{cases} \gamma f_i & \text{for} \ \ i = 1,\ldots,q \\ -\gamma(1+\eta+p_7\mho_i) & \text{for} \ \ i > q, \ f_i = -1 \\ \gamma(\eta+p_7\mho_i) & \text{for} \ \ i > q, \ f_i = 0 \end{cases} \tag{5.9b}$$

respectively. Also, if $p_5 = 0$, then $f_m := -1$ befores (5.7) is used.

Finally, if $p_3 = 2$, then we use (5.2) – (5.8) except that (5.5) is replaced by

$$f_i = \begin{cases} -2\boldsymbol{u} & \text{for} \ \ i = 1,\ldots,\lceil q/2\rceil \ , \\ \boldsymbol{u} & \text{for} \ \ i = \lceil q/2\rceil+1,\ldots,q \ , \\ 1 & \text{for} \ \ i = q+1,\ldots,m \ . \end{cases} \tag{5.10}$$

5.2. Performance

We have run a series of tests, where all parameters except one are fixed at their default value,

$$\begin{aligned} &n = 50, \quad m = 100 \ , \\ &p_1 = 10^{-2}, \quad p_2 = 0, \quad p_3 = 3, \quad p_4 = p_5 = 0 \ , \\ &p_6 = 4, \quad p_7 = 1, \quad p_8 \in \{0.8, 1, 1.6\} \ . \end{aligned} \tag{5.11}$$

In most of the tests we use all three choices of the number of active indices at the solution: $p_8 = 0.8$, 1, 1.6 give an underdetermined, a square and an overdetermined $\boldsymbol{WA}$, respectively. In the figures this is marked by $\triangledown$, $\bigcirc$, and $\triangle$, respectively.

First, consider the influence of size. Figure 5.1 shows the number of iterations needed to solve a problem of size $(n, m) = (5, 10), (10, 20), \ldots, (100, 200)$.

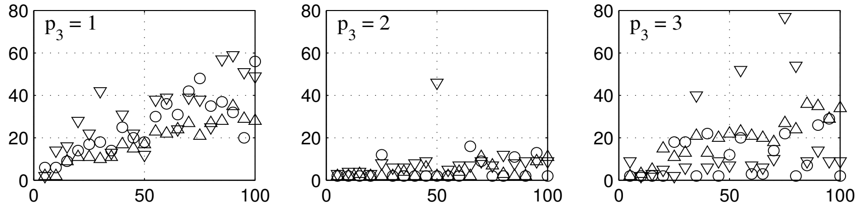

Figure 5.1. *Varying size.* $m = 2n$

The figure shows a strong influence of the randomness, and surprisingly often the choice of p_8 has random effect. As regards the choices of Huber function, there seems to be no significant difference between $p_3 = 1$ and $p_3 = 1$, while $p_3 = 2$ yields easier problems; this may be a peculiarity of the problem generator. In all cases the number of iterations seems to grow with n. A least squares fit with the model αn^β applied to all the results in Figure 5.1 gives $\alpha = 1.01$, $\beta = 0.55$, i.e. *#its* $\sim \sqrt{n}$. The number of factorizations varies between 1 and 4, and in all cases the computed $F(\boldsymbol{x}^*)$ has a relative error in the range [`-8.13e-12`, `6.31e-12`]. The negative lower bound indicates that the true solution to the computed problem may be different from what we thought. The relative error in the computed $\boldsymbol{x}^*$ is less than $100\varepsilon_M$ in all cases.

Next, we again vary the size, but now we include a vector $\boldsymbol{c}$.

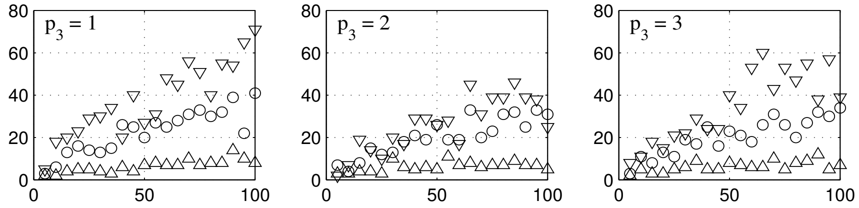

Figure 5.2. $\boldsymbol{c} \neq \boldsymbol{0}$, *varying size.* $m = 2n$

Here, there is no significant difference between the three choices of Huber function, but, the rank deficient problems are harder to solve, while the solution to an overdetermined problem is found very fast. We find *#factorizations* in the range $[1, 6]$. The relative errors in the computed $F(\boldsymbol{x}^*)$ are in the range [`-4.73e-14`, `1.41e-14`], and again the worst relative error in a computed $\boldsymbol{x}^*$ is less than $100\varepsilon_M$.

Figure 5.3 shows the effect of the condition of $\boldsymbol{WA}$.

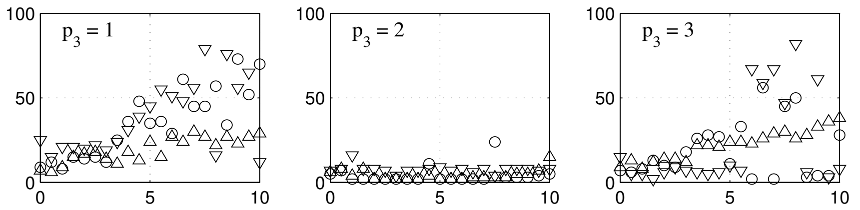

Figure 5.3. *Varying condition. #its as function of* p_5

As in Figure 5.1 the problems generated with $p_3 = 2$ are solved considerably faster than the problems corresponding to the other choices of Huber function. The number of factorizations is in the range $[1, 5]$, and we get the following results for relative errors in the computed $F(\boldsymbol{x}^*)$ and $\boldsymbol{x}^*$. The latter are given only for $p_8 \geq 1$ since $\boldsymbol{x}^*$ is not unique for an underdetermined $\boldsymbol{WA}$. Negative errors in F are indicated by the value 10^{-18}. In all cases the relative error in $F(\boldsymbol{x}^*)$ is almost constant, while the relative error in $\boldsymbol{x}^*$ grows with the condition. This is in accordance with expectations.

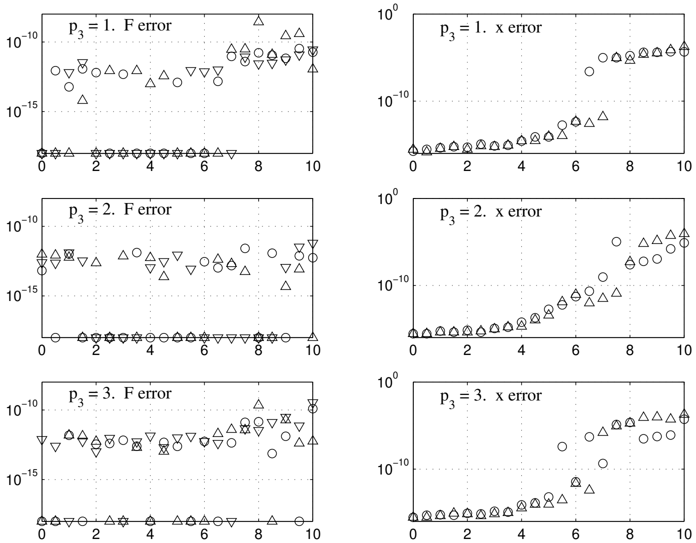

Figure 5.4. *Errors for varying condition*

In the remaining tests we give results for $p_3 = 3$ only.

Essentially, the residuals of the generated problems satisfy

$$|r_i| = \gamma\omega_i \quad \text{with} \quad \begin{cases} 0 \leq \omega_i \leq 1 & \text{for} \quad i \in \mathcal{A}(\boldsymbol{x}^*) \\ 1 < \omega_i \leq 1 + p_6 & \text{for} \quad i \notin \mathcal{A}(\boldsymbol{x}^*) \ . \end{cases}$$

Therefore, we can expect the behaviour of the algorithm to be independent of γ, and this is confirmed by Figure 5.5. Conversely, Figure 5.6 shows a strong influence of the "wildness" parameter p_6.

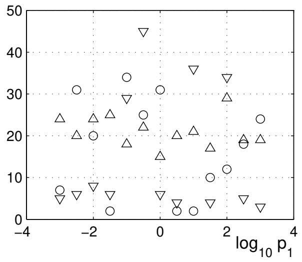

Figure 5.5. *Varying* γ

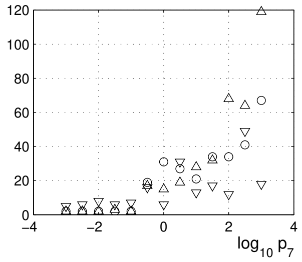

Figure 5.6. *Varying "wildness"*

Figure 5.7 is another illustration of varying wildness: It shows the number of iterations needed to solve a problem with given number of elements in $\mathcal{A}(\boldsymbol{x}^*)$. With $n = 50$ the default values of p_8 correspond to $\#\mathcal{A}(\boldsymbol{x}^*) \in \{40, 50, 80\}$. It seems that problems with $\#\mathcal{A}$ slightly larger than n are the hardest to solve.

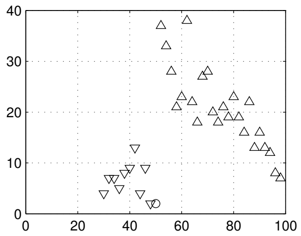

Figure 5.7. *Varying* $\#\mathcal{A}(\boldsymbol{x}^*)$

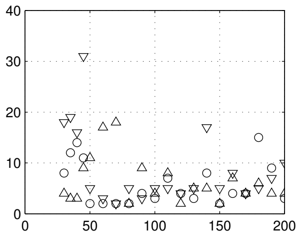

Figure 5.8. *Varying* m. $n = 25$

Figure 5.8 illustrates that the ratio m/n seems to have no strong influence on the behaviour of the algorithm.

Figure 5.9 gives results for $\mu > 0$, $\boldsymbol{L} = \boldsymbol{I}$. Almost identical results are found with the other two choices of $\boldsymbol{L}$. On the average the number of iterations is a decreasing function of μ. This is most clearly observed with the rank deficient $\boldsymbol{WA}$, where we avoid the problems connected with a rank deficient $\boldsymbol{H}$, (2.3).

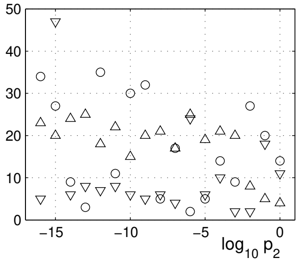

Figure 5.9. *Varying* μ, $\boldsymbol{L} = \boldsymbol{I}$

Finally, we have tested the warm start capabilities of the algorithm, cf. Section 3.4. The "worst" problem of Figure 5.6 was generated with

$$n = 50, \quad m = 100, \quad \#\mathcal{A}(\boldsymbol{x}^*) = 80,$$
$$\gamma = 10^{-2}, \quad \kappa_2(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}) = 10^4, \quad \mu = 0, \quad \boldsymbol{c} = \boldsymbol{0} \ ,$$

and corresponding to the Huber function defined by (1.1c). The algorithm needs 119 iterations to find the solution.

In Table 5.1 we give results for "cold start" on this problem with three successively reduced γ-values. "`mflops`" is the flop count from MATLAB, counted in millions.

γ	*its*	*facts*	$\#\mathcal{A}$	`mflops`
1	24	3	96	3.57
10^{-1}	55	4	86	5.54
10^{-2}	119	6	80	9.42

Table 5.1. *Cold start*

For the same problem we have used warm start, where the solution from $\gamma = 1$ (10^{-1}) is used to start the iteration for $\gamma = 10^{-1}$ (10^{-2}). First, consider warm start from $\boldsymbol{x}$. The numbers without and with parenthesis are found when the algorithm is used respectively with warm start and "hot start", i.e. when we also transfer the current $\boldsymbol{R}$.

γ	*its*	*facts*	$\#\mathcal{A}$	`mflops`
1	24	3	96	3.57
10^{-1}	40 (40)	2 (2)	86	3.52 (3.52)
10^{-2}	33 (33)	2 (2)	80	2.89 (2.89)

Table 5.2. *Warm (hot) start on* $\boldsymbol{x}$

Finally, we similarly use warm (and hot) start from the current active set.

γ	*its*	*facts*	$\#\mathcal{A}$	`mflops`
1	24	3	96	3.57
10^{-1}	18 (18)	3 (2)	86	3.15 (2.53)
10^{-2}	19 (19)	3 (2)	80	2.93 (2.38)

Table 5.3. *Warm (hot) start on* $\boldsymbol{s}$

These results confirm the experience from [10]: for a series of decreasing γ-values it is better to warm start on $\boldsymbol{s}$ than on $\boldsymbol{x}$. Also note the disappointing `mflops`-results in Table 5.2 when switching from warm to hot start.

5.3. Application

Figure 5.10a shows data points $(t_1, y_1), \ldots, (t_m, y_m)$, illustrating a typical behaviour of results from electrophoretic analysis, in casu of wheat, [5].

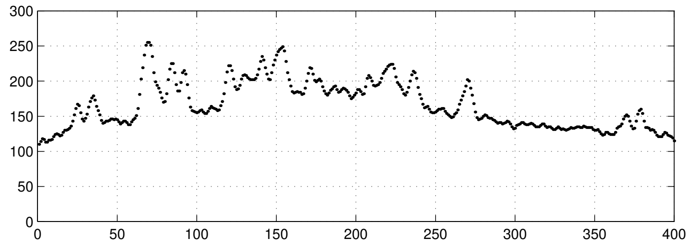

Figure 5.10a. *Contaminated electrophoresis spectrum*

The spectrum is contaminated by a smoothly varying background disturbance with values of the same order of magnitude as the spectrum itself. We can model the background by a natural cubic spline,

$$p(t) = \sum_{j=1}^{n} x_j N_j(t) \ ,$$

where the $\{N_j\}$ are B-splines defined with $n{-}1$ equal knot intervals covering the range, and modified to satisfy the natural end conditions $N_j''(t_1) = N_j''(t_n) = 0$.

In Figure 5.10b we give $\{(t_1, z_1)\}$ obtained by $\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^*$, where $a_{ij} = N_j(t_i)$ and $\boldsymbol{x}^*$ is defined by (1.1), except that we use the one sided

Huber function (4.1b). The computation was done with $n=5$, $\gamma=0.01$, and it needed 7 iterations from cold start. The minimal z-value is -1.48 and the standard deviation of the data is about 2, [18].

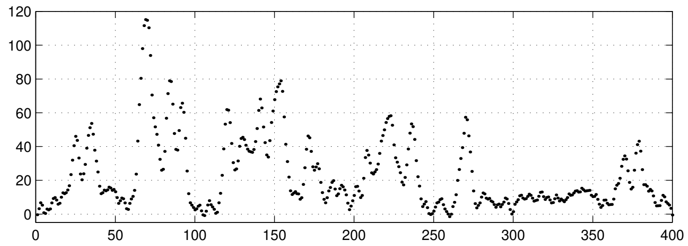

Figure 5.10b. *Purified electrophoresis spectrum*

6. Conclusion

The new implementation of the algorithm for computing a minimizer of a piecewise quadratic compares well with previous, specialized implementations. It was developed and tested in MATLAB, but will soon be available in Fortran77.

References

[1] J.W. Bandler, R. Biernacki, S. Chen, R. Hemmers and K. Madsen. 1994. *The Huber Concept in Device Modeling, Circuit Diagnosis and Design Centering,* IEEE Int. Symp. Circuits and Systems, **1**, 129–132.

[2] Å. Björck. 1996. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia.

[3] P.C. Hansen and K. Mosegaard. 1996. *Piecewise polynomial solutions without a priori break points,* Num. Lin. Alg. Appl. **3**, 513–524.

[4] P. Huber. 1981. *Robust Statistics.* John Wiley, New York.

[5] K. Jensen, I. Søndergaard, I.M. Skovgaard and H.B. Nielsen. 1995. *From Image Processing to Classification: 1. Modelling Disturbances of Isoelectric Focusing Patterns.* Electrophoresis **16**, 921–926.

[6] W. Li. 1995. *Conjugate gradient algorithms for convex quadratic splines,* Math. Programming, to appear.

[7] W. Li and J. Swetits. 1993. *A Newton method for convex regression, data smoothing, and quadratic programming with bounded constraints,* SIAM J. on Optimization **3**, 466–468.

[8] M.H. Loke and R.D. Barker. 1995. *Least-squares Deconvolution of Apparent Resistivity Pseudosections,* Geophysics **60**, 1682–1690.

[9] M.H. Loke and R.D. Barker. 1996. *Rapid Least-squares Inversion of Apparent Resistivity Pseudosections by a Quasi-Newton Method,* Geophysical Prospecting **44**, 131–152.

[10] K. Madsen and H.B. Nielsen. 1990. *Finite Algorithms for Robust Linear Regression,* BIT **30**, 682–699.

[11] K. Madsen, H.B. Nielsen. 1993. *A Finite Smoothing Algorithm for Linear L_1 Estimation,* SIAM J. on Optimization **3**, 223–235.

[12] K. Madsen, H. B. Nielsen and M.Ç. Pınar. 1995. *A New Finite Continuation Algorithm for Bound Constrained Quadratic Programming.* Report IMM-REP-95-22, Department of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark. Accepted for publication in SIAM J. on Optimization.

[13] K. Madsen, H.B. Nielsen and M.Ç. Pınar. 1996. *A New Finite Continuation Algorithm for Linear Programming,* SIAM J. on Optimization **6**, 600–616.

[14] K. Madsen, H.B. Nielsen and M.Ç. Pınar. 1996. *Bound Constrained Quadratic Programming via Piecewise Quadratic Functions.* Report IMM-REP-97-05, Department of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark. To appear in MATH. PROGRAMMING 1999.

[15] J. Moré and G. Toraldo. 1989. *Algorithms for Bound Constrained Quadratic Programming Problems,* Numer. Math. **55**, 377–400.

[16] H.B. Nielsen. 1990. *AAFAC: A Package of Fortran 77 subprograms for solving $A^TAx = c$,* Report NI-90-11, Institute for Numerical Analysis, Technical University of Denmark, DK-2800 Lyngby, Denmark.

[17] H.B. Nielsen. 1991. *Implementation of a Finite Algorithm for linear ℓ_1 Estimation,* Report NI-91-01, Institute for Numerical Analysis, Technical University of Denmark, DK-2800 Lyngby, Denmark.

[18] H.B. Nielsen. 1994. *Fitting Data from Electrophoretic Analysis of Wheat.* Report IMM-1994-08, Department of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark.

[19] H.B. Nielsen. 1996. *Bound Constrained Quadratic Programming Solved via Piecewise Quadratic Functions: Implementation.* Report IMM–1996–21, Department of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark. Available as http://www.imm.dtu.dk/~hbn/publ/TR9621.ps

[20] H.B. Nielsen. 1998. *Algorithms for Linear Optimization - an Introduction.* Department of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark.