# CVX Implementation Notes

Michael J. Meyer

December 14, 2017

## Contents

**1 Basic terminology and setup** **2**

**2 Handling the equality constraints** **3**

**3 Hessian** **5**

**4 Solving systems of equations** **8**
4.1 Null space . . . . . 9
4.1.1 Null space with QR-decomposition of $A$ . . . . . 10
4.1.2 Null space via SVD decomposition of $A$ . . . . . 12
4.2 Solving a system of equations with the SVD . . . . . 13
4.3 Preconditioning . . . . . 16
4.4 Solving the KKT system . . . . . 16
4.4.1 KKT system without inequality constraints . . . . . 16
4.4.2 Singular Hessians and KKT systems . . . . . 17

**5 Regularization of systems of equations** **18**
5.1 Using the SVD of $A$ . . . . . 22
5.2 Stochastic formulation . . . . . 22
5.3 Parameter estimation . . . . . 24
5.3.1 Estimation of $\tau$ and $\Delta$ . . . . . 26
5.4 Tichonov regularization and trust regions . . . . . 28

**6 Kullback-Leibler distance** **29**
6.1 Minimization of $d_{KL}$ under probability constraints . . . . . 30

**7 Problem transformations** **31**

**8 Code Structure** **32**

**9 Pitfalls** **33**
9.1 Tolerance in the BarrierSolver too small . . . . . 33

# 1 Basic terminology and setup

An *OptimizationProblem* consists of a convex *ObjectiveFunction* $f(x) \in \mathbb{R}$, $x \in \mathbb{R}^n$, a *ConstraintSet* (list of inequality constraints $g_j(x) \leq u_j$, $j = 1, \ldots, m$) and possibly equality constraints in the form $Ax = b$, where $A$ is an $n \times p$ matrix and $b \in \mathbb{R}^p$ a vector. This means that we have $p$ equality constraints

$$row_k(A) \cdot x = b_k, \quad k = 1, \ldots, p.$$

It is assumed that $p < n$ and that the matrix $A$ is of full rank $p$ (i.e. no equalities can be eliminated because they are merely a linear combination of other equalities). Typically $p$ is much smaller than $n$.

The equality constraints are treated separately and are not included in the ConstraintSet of the problem. In other words: the ConstraintSet consists only of the inequality constraints.
The optimization problem now has the following form:

$$\text{Minimize } f(x) \quad \text{subject to } g_j(x) \leq u_j \text{ and } Ax = b. \tag{1}$$

The problem is called *feasible* if there exists a point $x \in \mathbb{R}^n$ satisfying all the constraints $g_j(x) \leq u_j$ and $Ax = b$ (a so called *feasible point*) and is called *infeasible* otherwise.

Clearly, if the problem is infeasible nothing further has to be done and some of the solution algorithms need a feasible point as a starting point.

Thus the first step in the solution must be to decide whether the problem is feasible at all and find a feasible point. In general this is a nontrivial problem requiring a *feasibility analysis* (think of thousands of variables $x_j$ and hundreds of constraints).

Interestingly such an analysis can be carried out as a modified convex minimization problem

$$\text{Minimize } \tilde{f}(x, s) = s \quad \text{subject to } g_j(x) - s \leq u_j \text{ and } Ax = b. \tag{2}$$

for which a feasible point $(x_0, s_0)$ is easily found (any solution $Ax_0 = b$ combined with sufficiently large $s_0$). If the minimization comes up with a point $(x^*, s^*)$ satisfying $s^* \leq 0$, then $x^*$ is a feasible point for the original problem. This is called *basic Phase I feasibility analysis*.

If the problem only admits feasible points $x$ such that $g_j(x) = u_j$ for some $j$, then in general, in finite precision arithmetic, we cannot definitively decide if the problem is in fact feasible. Such a decision may still be possible in special cases (e.g. if the functions $g_j(x)$ are integer valued) but this will not be pursued here.

Our analysis will only be able to find feasible points $x$ satisfying $g_j(x) < u_j - \epsilon$ or conclude that the problem is infeasible if a *certificate of infeasibility* can be found. To see what such a certificate might be let us assume that we have no equality constraints (the case with equality constraints is only slightly more involved) and consider the function

$$L(x, s, \nu) := \tilde{f}(x, s) + \sum\nolimits_{j=1}^{m} \nu_j (g_j(x) - s - u_j), \quad \nu \in \mathbb{R}^m.$$

If the vector $\nu$ satisfies $\nu_j \geq 0$, for all $j = 1, \ldots, m$, then we have

$$L(x, s, \nu) \leq \tilde{f}(x, s) = s,$$

for all points $(x, s)$ satisfying the constraints in (2). Taking the infimum over feasible points $(x, s)$ we have

$$L_*(\nu) = \inf_{x,s} L(x, s, \nu) \leq \inf\{\, s \,:\, g_j(x) - s \leq u_j \,\}.$$

If it is known that $L_*(\nu) > 0$, then the constraints $g_j(x) \leq u_j$ are infeasible and the vector $\nu$ is a certificate of infeasibility. Some solution algorithms yield such $\nu$ as a byproduct of the computation.

Indeed, for any such $\nu$ the function $L(x, s, \nu)$ is convex in the variables $(x, s)$ so that the equation

$$\nabla_{x,s} L(x_0, s_0, \nu) = 0$$

implies that the function $l(x, s) = L(x, s, \nu)$ assumes a global minimum at the point $(x_0, s_0)$. In other words, we have $L_*(\nu) = L(x_0, s_0, \nu)$.

Some algorithms actually solve this equation for certain $\nu \in \mathbb{R}^m_+$ obtaining a solution $(x_0, s_0)$ so that the value $L(x_0, s_0, \nu)$ itself provides the lower bound on the variable $s$ in the problem (2).

Again it is clear that $L_*(\nu) = L(x_0, s_0, \nu) > 0$ can only be detected in finite precision arithmetic if in fact $L_*(\nu) \geq \epsilon > 0$, for some sufficiently large $\epsilon > 0$. Thus a gray zone remains in which the problem is feasible but this cannot be detected by our algorithms.

We will tackle the optimization problem in a naive fashion in the general context of twice differentiable convex functions. In this generality no performance guarantees exist. To get such guarantees one needs to restrict oneself to standard problems (which are however quite general). See [1] and [3].

Once this is done one can work with all manner of tricks to transform a wide variety of problems (even non differentiable ones) to standard form (see the introduction of [4]). This is the purview of *disciplined convex programming*. Such an approach is far more complex than what is attempted here. The reader is referred to [5] for more information.

## 2 Handling the equality constraints

The equality constraints $Ax = b$ always form an under-determined system, that is $A$ is an $n \times p$ matrix where $p < n$ and we assume that the matrix $A$ has rank $p$.

The solution set is then an affine subspace of dimension $n - p$ in the space $\mathbb{R}^n$ in which the decision variable $x$ lives. We will deal with these constraints in one of two ways:

- Handle them in the KKT conditions, or

- Eliminate them by solving the system $Ax = b$ as $x = z + Fu$, where $z$ is the minimum norm solution of $Ax = b$ and $Im(F) = ker(A)$, that is, $w = Fu$ is the general solution of $Ax = 0$. This approach will be called *reduction*.

Which of the two approaches is more efficient depends on the precise circumstances of the problem. For example in the Barrier method the inequality constraints are absorbed into the objective function leading to a new problem without inequality constraints. In this case the equality constraints increase the dimension of the resulting KKT system from $n \times n$ to $(n + p) \times (n + p)$.

On the other hand reduction is achieved by a change of variable $x \in \mathbb{R}^n \to u \in \mathbb{R}^{n-p}$ leading to a reduction in dimension. However this change of variables affects the computation of gradients and Hessian matrices:

$$\nabla_u h(z + Fu) = F' \nabla_x h(z + Fu) \quad \text{and} \tag{3}$$

$$\nabla_u^2 h(z + Fu) = F' \nabla_x^2 h(z + Fu) F \tag{4}$$

Note that $dim(Im(F)) = dim(ker(A)) = n-p$ and so $F$ is an $(n-p)\times n$-matrix, where usually $p$ is much smaller than n. Consider the extreme case $p = 1$: in this case $F$ is a large matrix and the matrix multiplications in the computation of $\nabla_u^2 h(z + Fu)$ are expensive.

If nothing is known about the structure of the function $h$ this multiplication has to be carried out whenever the Hessian $\nabla_u^2 h(z + Fu)$ is evaluated at a new point $u$. In the course of the our minimization algorithms this has to done for the objective function $h = f$ as well as all the functions $h = g_j$ defining the inequality side conditions (which could number in the hundreds).

Clearly this has the potential to introduce catastrophic overhead. On the other hand, if $h$ is known to be a quadratic function:

$$h(x) = r + a'x + \tfrac{1}{2} x'Px := h_{r,a,P}(x)$$

with a symmetric matrix $P$ then, using that $h$ agrees with its second order Taylor expansion and that $\nabla h(x) = a + Px$ and $\nabla^2 h(x) = P$, we get

$$\begin{aligned} h(z + Fu) &= h(z) + (Fu)' \nabla_x h(z) + \tfrac{1}{2} (Fu)' \nabla_x^2 h(z) Fu \\ &= h(z) + (Fu)'(a + Pz) + \tfrac{1}{2} u'(F'PF)u \\ &= h_{q,b,Q}(u), \end{aligned} \tag{5}$$

where

$$q = h(z),\ b = F'(a + Pz),\ \text{and}\ Q = F'PF.$$

In other words $h(z + Fu)$ is another quadratic function of $u$ with known coefficients which can be precomputed. The change of variables $x \to u$ is then implemented simply by passing to the new coefficients which costs only one expensive matrix multiplication.

The gradient and Hessian with respect to the variable $u$ can be read off from the new coefficients (at each point $u$) and incur no expensive matrix multiplications.

# 3 Hessian

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function. The second order Taylor expansion of $f$ centered at $x$ has the form

$$f(x+h) = f(x) + L(h) + \tfrac{1}{2}B(h,h) + R(h),$$

where $L$ is a linear function of $h \in \mathbb{R}^n$, $B(u,v)$ is a bilinear function of $(u,v) \in \mathbb{R}^n \times \mathbb{R}^n$ and the remainder $R(h)$ satisfies $R(h) = o(\|h\|^2)$. This condition on the remainder ensures that $L$ and $B$ are uniquely determined as

$$L(h) = \nabla f(x)'h \quad \text{and} \quad B(u,v) = u'Hv,$$

where $H := \nabla^2 f(x) \in Mat_{n\times n}(\mathbb{R})$ is the matrix with entries

$$H_{ij} = B(e_i, e_j) = \frac{\partial^2 f}{\partial x_i \partial x_j}(x),$$

i.e. the Hessian matrix of $f$ at $x$.

To compute the gradient and Hessian of a $C^2$-function $f$ we make use of the fact that the remainder condition $R(h) = o(\|h\|^2)$ in a quadratic expansion

$$f(x+h) = f(x) + \Delta'h + h'Hh + R(h)$$

uniquely determines the "coefficients" $\Delta$ and $H$ as $\Delta = \nabla f(x)$ and $H = \nabla^2 f(x)$. We only need to find such an expansion for $f(x+h)$ and check that the remainder satisfies $R(h) = o(\|h\|^2)$. This is how we will derive our formulas below.

**Hessian of affine transformation.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a $C^2$-function and $\overline{f}(u) = f(x_0 + Fu)$, where $x_0 \in \mathbb{R}^n$ and $F : \mathbb{R}^m \to \mathbb{R}^n$ is a linear map, i.e. $F \in Mat_{n\times m}$.

We want to compute the gradient and Hessian of $h$ at any point $u$ from those of $f$. To get these do a second order Taylor expansion of $f$ about $x$:

$$f(x+h) = f(x) + h^T g + h^T H h + o(\|h\|^2),$$

where $g = \nabla f(x)$ and $H = \nabla^2 f(x)$ are uniquely determined by the fact that the residual is $o(\|h\|^2)$. Applying this to the point $x = x_0 + Fu$ this implies that

$$\begin{aligned}\overline{f}(u+h) &= f(x_0 + Fu + Fh) \\ &= f(x_0 + Fu) + (Fh)^T g + (Fh)^T H F h + o(\|Fh\|^2).\end{aligned}$$

Since $o(\|Fh\|^2)$ is $o(\|h\|^2)$ we conclude from this that

$$\nabla\overline{f}(u) = F^T g \quad \text{and} \quad \nabla^2\overline{f}(u) = F^T H F,$$

or, more explicitly

$$\nabla\overline{f}(u) = F^T \nabla f(x_0 + Fu) \quad \text{and} \quad \nabla^2\overline{f}(u) = F^T \nabla^2 f(x_0 + Fu) F. \tag{6}$$

**Hessian of composition.** With a similar approach we can compute the Hessian of a composition $g(f(x))$, where here $g : \mathbb{R} \to \mathbb{R}$ is a scalar function of one variable (more general $g$ are much harder to handle and we do not need them). Indeed, set

$$y = f(x), \quad \nabla = \nabla f(x) \quad H = \nabla^2 f(x) \quad \text{and} \quad k = h^T \nabla + \tfrac{1}{2} h^T H h$$

and use second order Taylor approximations on $f$ at the point $x$ and $g$ at the point $y = f(x)$ to obtain:

$$\begin{aligned} g(f(x+h))) &= g\left(f(x) + h^T \nabla + \tfrac{1}{2} h^T H h\right) \\ &= g(y+k) = g(y) + g'(y) + \tfrac{1}{2} g''(y) k^2 + o(k^2) \\ &= g(y) + g'(y)\left(h^T \nabla + \tfrac{1}{2} h^T H h\right) + \frac{1}{2} g''(y)\left(h^T \nabla + \tfrac{1}{2} h^T H h\right)^2 + o(\|h\|^2). \end{aligned}$$

Here we have used that $o(k^2) = o(\|h\|^2)$. Collect terms of first and second order in $h$ together and sticking all terms of higher order into the residual $o(\|h\|^2)$. Note that the squared term contributes no first order terms and only one second order term, this being the term

$$(h^T \nabla)^2 = (h^T \nabla)(h^T \nabla) = (h^T \nabla)(h^T \nabla)^T = h^T (\nabla \nabla^T) h.$$

We obtain

$$g(f(x+h))) = g(y) + h^T [g'(y) \nabla] + \frac{1}{2} h^T \left[g'(y) H + g''(y) \nabla \nabla^T\right] h + o(\|h\|^2).$$

and from this we can read off that

$$\nabla (g \circ f)(x) = g'(f(x)) \nabla f(x), \quad \text{and} \tag{7}$$

$$H(g \circ f)(x) = g'(f(x)) H + g''(f(x)) \nabla f(x) \nabla f(x)^T \tag{8}$$

Note that here $d = \nabla f(x)$ is viewed as a column vector and so $\nabla f(x) \nabla f(x)^T$ is the *outer product*

$$\nabla f(x) \nabla f(x)^T = d d^T = (d_i d_j)_{ij}.$$

We will need the following example to construct test functions for unconstrained minimization:

**Example 3.1.** Let $\phi : G \subseteq \mathbb{R} \to \mathbb{R}$ be a function of one variable defined on an open subset $G \subseteq \mathbb{R}$, fix $a \in \mathbb{R}^n$ and set $g(x) = \phi(a \cdot x)$, for all $x \in \mathbb{R}^n$ such that $a \cdot x \in G$.

Since $f(x) = a \cdot x$ satisfies $\nabla f(x) = a$ and $H = \nabla^2 f(x) = 0$, for all $x \in \mathbb{R}^n$, the formulas (7) and (8) yield

$$\nabla g(x) = \phi'(a \cdot x) a \quad \text{and} \quad \nabla^2 g(x) = \phi''(a \cdot x) a a'.$$

The idea is to construct test functions of the form

$$objF(x) = \sum\nolimits_j \phi_j(a_j \cdot x)$$

where all the $\phi_j = \phi_j(u)$ have a unique minimum at $u = 0$. Then $objF(x)$ assumes its global minimum at all points $x$ satisfying $a_j \cdot x = 0$, for all $j$, equivalently $Ax = 0$, where $A$ is the matrix with rows $a_j$, provided such a solution exists.

If the functions $\phi_j$ are all convex, the same is true of our objective function $objF$ (sums and compositions of convex functions are again convex) With this we can construct examples with well conditioned, poorly conditioned and even singular Hessians ($ker(A) \neq \{0\}$) with known minimizers. The conditioning of $\nabla^2 f(x)$ is closely related to that of the matrix $A$.

# 4 Solving systems of equations

Consider a general system $Ax = b$ of equations, where $A$ is an $p \times n$ matrix, not necessarily square (i.e. $A : \mathbb{R}^n \to \mathbb{R}^p$ as a linear operator). This this system could be over-determined ($p > n$, with possibly no solutions) or under-determined ($p < n$, with infinitely many solution).

To treat all cases at once we reframe the problem of solving the system $Ax = b$ as minimizing the norm $\|Ax - b\|$ which makes sense in all cases. The system has a solution if and only if this minimum is zero. Now the minimizers $x$ of the quadratic form

$$g(x) := \tfrac{1}{2}\|Ax - b\|^2 = \tfrac{1}{2}(Ax - b, Ax - b) = \tfrac{1}{2}x'A'Ax - (A * b)'x + \tfrac{1}{2}\|b\|^2$$

are exactly the points $x$ satisfying $0 = \nabla g(x) = A'Ax - A'b$, that is, the so called normal equations

$$A'Ax = A'b.$$

In particular any two minimizers $x, w$ satisfy $A'Ax = A'Aw$, that is, the difference $x - w$ must be in the kernel $ker(A'A)$.

Clearly $ker(A'A) \supseteq ker(A)$ and $Im(A'A) \subseteq Im(A')$ and because of $(A'Ax, x) = \|Ax\|^2$ we have $ker(A'A) \subseteq ker(A)$ so that in fact

$$ker(A'A) = ker(A) := N.$$

Thus any two minimizers $x, w$ of $\|Ax - b\|$ differ by an element $u \in ker(A)$ so that if both are in $ker(A)^\perp$, then $x = w$. Moreover if $x$ is any minimizer, we obtain a minimizer $x_0$ in $ker(A)^\perp$ simply by subtracting the projection of $x$ onto $ker(A)$ from $x$.

In other words there is a unique minimizer $x_0$ of $\|Ax - b\|$ in $ker(A)^\perp$. It is now easily seen that all minimizers $x$ of $\|Ax - b\|$ are of the form $x = x_0 + u$, where $u \in ker(A)$. In particular it follows that $x_0$ is also the unique minimizer of $\|Ax - b\|$ which has itself minimal norm. All this is in complete analogy to the case where the system $Ax = b$ has an exact solution.

From dimension arguments ($dim(ker(A)) + dim(Im(A)) = n$) it now follows that also $Im(A'A) = Im(A')$. Clearly then the restriction

$$A'A|_{N^\perp} : ker(A)^\perp \to Im(A')$$

is bijective, that is, $A'A$ is invertible on the range $Im(A')$, and the unique minimizer of $\|Ax - b\|$ in $ker(A)^\perp$ is given by

$$x_0 = (A'A)^{-1}A'b.$$

If the matrix $A$ is ill conditioned any least squares solution $x$ computed from the normal equations

$$A'Ax = A'b$$

is mostly useless. The problem can be seen more explicitly as follows: let $A = UDV'$ be the SVD of $A$ where $D = diag(\sigma_j)$ is a diagonal matrix with the singular values of $A$ on the main diagonal.

We will see in (23) the least squares solution $x_0$ of minimal norm, i.e. the minimizer of $\|Ax - b\|$ which has itself minimal norm is given by the expression

$$x_0 = \sum\nolimits_{\sigma_j > 0} \frac{u_j' b}{\sigma_j} v_j.$$

If now $b$ has a significant component $u_j' b$ in the direction of some $u_j$ where the singular value $\sigma_j$ is very small then we are dividing a number of significant size by a very small number which can lead to error blowup.

Regularization methods deal with this problem. We will discuss this in detail in the sections below. Next we deal with under-determined systems $Ax = b$ where the general solution has the form $x = x_0 + Fu$ with $F$ any matrix such that $Im(F) = ker(A)$ and $x_0$ any particular solution.

## 4.1 Null space

Here we discuss methods to solve an under determined set of linear equations $Ax = b$ where the number $n$ of variables is larger than the number $p$ of equations. In other words $A$ is an $p \times n$ matrix with $p < n$.

The set of solutions is then the hyperplane $x_0 + ker(A)$ where $ker(A)$ denotes the null space of $A$ and $x_0$ is any particular solution of $Ax = b$.

We will represent the null space of $A$ as the column space $colspace(F)$ of a matrix $F$ such that

$$AF = 0.$$

If $F$ satisfies this equation, then the column space of $F$ is a subspace of the null space $ker(A)$. Recall also that the column space of $F$ is the range $Im(F)$ of the linear map defined by the matrix $F$.

We will generally assume that $A$ *has full rank*, i.e. $rank(A) = p$. The null space $ker(A)$ then has dimension $n - p$ and the columns of $F$ span the entire null space of $A$ if and only if $rank(F) = n - p$.

If we have found such a matrix $F$ and $x_0$ with $Ax_0 = b$, then the hyperplane of all solutions to $Ax = b$ can be represented as

$$x_0 + ker(A) = x_0 + Im(F).$$

We want to apply this to the minimization problem

$$\text{minimize } f : \mathbb{R}^n \to \mathbb{R} \text{ under the constraint } Ax = b. \tag{9}$$

Since the solutions $x$ to the constraint are exactly the vectors $x$ of the form $x = x_0 + Fu$, $u \in \mathbb{R}^{n-p}$, we can turn this into the unconstrained problem

$$\text{minimize } g(u) = f(x_0 + Fu) \text{ on all of } \mathbb{R}^{n-p}. \tag{10}$$

Thereby we reduce the dimension of the problem from $n$ to $n - p$. In large scale problems one would not do this since this operation can destroy the sparseness of the Hessian $H(f)$ which is critical for computation in very large dimensional problems.

In dense small problem (up to say $n = 1000$ variables) the above elimination of the constraint $Ax = b$ is a reasonable approach. We will discuss two algorithms to finding $F$ and $x_0$. In both cases the columns of $F$ will be orthonormal and hence an orthonormal basis for the null space $ker(A)$.

### 4.1.1 Null space with QR-decomposition of $A$

This approach relies on the relation $ker(A) = Im(A')^{\perp}$, where the prime denotes the matrix transpose and $V^{\perp}$ is the orthogonal complement of a subspace $V$. Recall that

$$\begin{aligned} n &= dim(ker(A)) + dim(ker(A)^{\perp}) = dim(ker(A)) + dim(Im(A')) \\ &= (n-p) + dim(Im(A')). \end{aligned}$$

Thus $dim(Im(A')) = p$. We will find the range $Im(A')$ and its orthogonal complement using the QR-decomposition of the transpose $A'$ (which has dimension $n \times p$, i.e. maps from $\mathbb{R}^p$ to $\mathbb{R}^n$):

$$A' = QR, \tag{11}$$

where $Q$ is orthogonal and $R$ upper triangular with nonzero diagonal, where $R$ is $r \times p$ (i.e. maps from $\mathbb{R}^p \to \mathbb{R}^r$ and $Q$ is $p \times r$, i.e maps from $\mathbb{R}^r \to \mathbb{R}^p$.

In such a factorization $Q$ and $R$ must be $n \times r$ and $r \times p$ respectively, since $A'$ is $n \times p$. Moreover $rank(Q) \geq rank(A') = rank(A) = p$ and so we must have $r \geq p$. Because the columns of $Q$ are linearly independent and of dimension $n$ we also must have $r \leq n$.

Indeed such factorizations exist for all $p \leq r \leq n$ but in practice (i.e. available in libraries) there are only two flavors:

(A) The reduced (minimal) form gives us $R$ as an $p \times p$ matrix and $Q = [q_1, \ldots, q_p]$ as an $n \times p$ matrix with $p$-columns $q_j \in \mathbb{R}^n$. It follows that $R$ maps onto $\mathbb{R}^p = dom(Q)$ and hence

$$Im(A') = Im(Q) = span(q_1, \ldots, q_p).$$

From this we see that $ker(A) = Im(A')^{\perp} = span(q_1, \ldots, q_p)^{\perp}$ but if we use the reduced form we have to compute this orthogonal complement ourselves. In other words we have to extend $\{ q_1, \ldots, q_p \}$ to an orthonormal basis $\{ q_1, \ldots, q_p, q_{p+1}, \ldots q_n \}$ of $\mathbb{R}^n$ ourselves.

In `R` the minimal form is obtained as follows

```
qrA <- qr(t(A)); Q <- qr.Q(qrA); R <- qr.R(qrA)
```

Here we first compute a QR-decomposition object `qrA` and from this object extract the matrices $Q$ and $R$ using the helper functions `qr.Q` and `qr.R`.

The intermediate forms of the decomposition (11) with $p \leq r \leq n$ extend this matrix $Q$ by adding (arbitrarily) orthonormal columns on the right

$$Q \to Q_+ = [q_1, \ldots, q_p, q_{p+1}, \ldots q_r]$$

and adjusting the matrix $R$ by adding zero rows at the bottom (so that $R$ becomes $r \times p$. This of course does not change the matrix product $QR$ as we can see from block multiplication

$$Q_+ = [Q, Q_1],\ R_+ = \begin{pmatrix} R \\ 0 \end{pmatrix} \quad \Longrightarrow \quad Q_+R_+ = QR + Q1 * 0 = QR.$$

(B) The full (maximal) form of the decomposition has $r = n$, that is, the columns of $Q$ have been extended to a full orthonormal basis of $\mathbb{R}^n$ and so clearly

$$ker(A) = Im(A')^{\perp} = span(q_1, \ldots, q_p)^{\perp} = span(q_{p+1}, \ldots, q_n). \tag{12}$$

In other words the matrix $F$ can be chosen to be the matrix with columns $q_{p+1}, \ldots, q_n$:

$$F = [q_{p+1}, \ldots, q_n] \in Mat_{n \times (n-p)}(\mathbb{R}). \tag{13}$$

To get a special solution $x_0$ of $Ax = b$ note that $A = R'Q'$ and solve the factored form $R'Q'x = b$. Set $y = Q'x$. Then *forward solve* the lower triangular system $R'y = b$ and get $x$ from $Q'x = y$ as $x = Qy$.

To get the complete factorization $A' = Q_+R_+$ in $R$ we do

```
qrA <- qr(t(A)); Qp <- qr.Q(qrA,complete=TRUE); R <- qr.R(qrA)
```

and then extract the matrix $Q$ via `Q <- Qp[,1:p]`. $R$ is already in incomplete form (no zero rows at bottom) since we have not specified `complete=TRUE` in the function `qr.R` (the default is `complete=FALSE`).

With this the factorization of $A'$ becomes $A' = QR$ (and not $A' = QpR$) and this is what we need in the computation of the special solution $x_0$ above.

The same holds true in the Scala `breeze` library where we get the `QR` decomposition as

```
val qr.QR(q,r)=qr(A).
```

This yields $Q = q$ as $n \times n$-matrix but yields $R = r$ as $p \times p$-matrix, that is, without the $n-p$ bottom zero rows. With this the decomposition of $A'$ becomes $A' = Q[\cdot, 1:p] * R$ and so $A = R'P'$ where the matrix

$$P = Q[\cdot, 1:p] = [col_1(Q), \ldots, col_p(Q)]$$

has orthogonal columns and we must solve for $x_0$ as $R'y = b$, $P'x = y$, that is, $x = Py$. Let us note that this $x_0$ is the minimal norm solution of the system $Ax = b$. Indeed $x_0 = Py \in span(col_1(Q), \ldots, col_p(Q))$ while $Fu \in span(col_{p+1}(Q), \ldots, col_n(Q))$ and so $Fu \perp x_0$, for all $u \in \mathbb{R}^{n-p}$, from which the claim follows.

The QR factorization is computed by repeatedly applying (orthogonal) Householder updates and is thus a very stable algorithm.

### 4.1.2 Null space via SVD decomposition of $A$

We can also compute the null space $ker(A)$ and special solution $x_0$ of $Ax = b$ using the more involved *SVD-decomposition* of $A$:

$$A = U\Sigma V'$$

where $U \in Mat_{p\times p}(\mathbb{R})$ and $V \in Mat_{n\times p}(\mathbb{R})$ are orthonormal matrices and $\Sigma$ is an $p \times n$ diagonal matrix with entries

$$\sigma_1 \geq \sigma_2 \cdots \geq \sigma_p \geq 0$$

(the singular values of $A$). Since $rank(A) = p$ by assumption we must have $\sigma_i > 0$ and so the matrix $\Sigma$ is invertible. Write $V = [v_1, \ldots, v_p]$, where the $v_j \in \mathbb{R}^n$ are the columns of $V$. Thus, for all $x \in \mathbb{R}^n$,

$$\begin{aligned} Ax = 0 \quad &\Longleftrightarrow \quad U\Sigma V'x = 0 \\ &\Longleftrightarrow \quad \Sigma V'x = 0 \\ &\Longleftrightarrow \quad 0 = V'x = (v_1 \cdot x, v_2 \cdot x, \ldots, v_p \cdot x)' = 0 \\ &\Longleftrightarrow \quad x \perp \{\, v_1, \ldots, v_p \,\} \\ &\Longleftrightarrow \quad x \in span(v_{p+1}, \ldots, v_n) \end{aligned}$$

In other words

$$ker(A) = span(v_{p+1}, \ldots, v_n). \tag{14}$$

where $v_{p+1}, \ldots, v_n \in \mathbb{R}^n$ are vectors which extend $v_1, \ldots, v_p$ to an ON-basis of $\mathbb{R}^n$. To get the full $n \times n$ matrix

$$V_+ = [V, v_{p+1}, \ldots, v_n]$$

we must compute the SVD in `R` via

```
svdA <- svd(A,nv=n); Vp <- svdA$v; V <- V[,1:p]
```

Now a particular solution of $U\Sigma V'x = Ax = b$ can be found by solving $\Sigma V'x = U'b := c$ which is equivalent to

$$V'x = (c_1/\sigma_1, \ldots, c_p/\sigma_p)',$$

A particular solution of this is given by

$$x_0 = Vw, \quad \text{where} \quad w = (c_1/\sigma_1, \ldots, c_p/\sigma_p)' \in \mathbb{R}^p. \tag{15}$$

since the columns of $V$ are orthonormal.

## 4.2 Solving a system of equations with the SVD

The SVD decomposition is a convenient way to deal with systems of linear equations even if they have no exact solution (e.g. singular or overdetermined). Consider the system $Ax = b$, where $A$ is any given $p \times n$ matrix and $x \in \mathbb{R}^n$, $b \in \mathbb{R}^p$. Let

$$A = UDV' \tag{16}$$

be the SVD of $A$. Here $D$ is an $r \times r$ diagonal matrix, where $r$ is the rank of $A$ and the diagonal elements $\sigma_j$ are the *nonzero* singular values of $A$, in particular $\sigma_j > 0$.

$U \in Mat_{p\times r}$ and $V \in Mat_{n\times r}$ are matrices with orthonormal columns: $U'U = I_{r\times r} = V'V$. Note that this *does not imply* that $UU' = I_{n\times n}$ or $VV' = I_{n\times n}$ since $r \leq n$ (for reasons of rank) and possibly $r < n$. Indeed these equalities hold exactly if $r = n$.

Write $U = (u_1, v_2 \ldots, u_r)$ and $V = (v_1, v_2, \ldots, v_r)$, where the $u_j$ and $v_j$ are the columns of $U$ and $V$ respectively. Then the decomposition 16 has the form

$$A = (u_1, v_2 \ldots, u_r) \begin{pmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & \sigma_r \end{pmatrix} \begin{pmatrix} v_1' \\ v_2' \\ \vdots \\ v_r' \end{pmatrix} = \sum\nolimits_{j=1}^{r} \sigma_j u_j v_j' \tag{17}$$

with strictly positive $\sigma_j$. It follows that $A$ is the operator

$$Ax = \sum\nolimits_j \sigma_j (v_j' x) u_j = \sum\nolimits_j \sigma_j (x, v_j) u_j \tag{18}$$

with linearly independent coefficient functionals $\sigma_j(\cdot, v_j)$. It follows immediately that

$$Im(A) = Im(U) = span(\{u_1, u_2, \ldots, u_r\}).$$

especially $rank(A) = r$ so that $A$ is nonsingular if and only if $r = n$. Moreover

$$ker(A) = \{v_1, \ldots, v_r\}^{\perp} = ker(V'). \tag{19}$$

As an aside let us note that $P = UU'$ is the orthogonal projection onto the range of $U$: indeed $P^2 = (UU')(UU') = U(U'U)U' = UU' = P$ and $P$ is self adjoint and hence is the orthogonal projection onto its range. It remains to be seen that $Im(P) = Im(U)$. Clearly $Im(P) = Im(UU') \subseteq Im(U)$. Conversely let $y \in Im(U)$, say $y = Ux$. Then $y = U(U'U)x = (UU')Ux \in Im(UU')$. Likewise $VV'$ is the orthogonal projection onto the range of $V$.

Note that $Im(U) \subseteq \mathbb{R}^p$ has dimension $r = rank(U)$ which may be less than $p$. If $p = n$ (the square case) this is the case if and only if the matrix $A$ is singular.

From $A = UDV'$ it follows directly that $Im(A) \subseteq Im(U)$. On the other hand $AV = UD$ so that $Im(A) \supseteq Im(UD) = Im(U)$.

Since $Im(A) = Im(U)$ in general only the system $Ax = Pb$ is solvable and any solution is a minimizer of the norm $\|Ax - b\|$. Now we will solve this system using the representation (18).

Rewrite $Ax = Pb$ as

$$\sum\nolimits_{j=1}^{r} \sigma_j (v_j' x) u_j = \sum\nolimits_{j=1}^{r} (u_j' x) u_j \tag{20}$$

We get the solution by comparison of coefficients in the expansion in terms of the orthonormal $u_j$. Enlarge $\{v_1, \ldots, v_r\}$ to an orthonormal basis $\{v_1, \ldots, v_n\}$ of $\mathbb{R}^n$. Then $x$ has the form

$$x = \sum\nolimits_{j=1}^{n} (v_j' x) v_j \tag{21}$$

but the left hand side of (20) does not depend on the coefficients $v_j' x$, where $j > r$, so that these can be arbitrary, while for $j < r$ we want $\sigma_j (v_j' x) = u_j' b$, that is,

$$v_j' x = \frac{u_j' b}{\sigma_j}.$$

Putting this into (21) we see that the general solution of $Ax = Pb$ is given by

$$x = \sum\nolimits_{j \le r} \frac{u_j' b}{\sigma_j} v_j + \sum\nolimits_{j=r+1}^{n} \lambda_j v_j \tag{22}$$

with arbitrary scalars $\lambda_j$. Obviously then the solution of minimal norm to the equation $Ax = Pb$ is given by

$$x = \sum\nolimits_{j \le r} \frac{u_j' b}{\sigma_j} u_j. \tag{23}$$

This $x$ is then the minimizer of $\|Ax - b\|$ which has itself minimal norm. Note that we can directly compute the error $\|Ax - b\|$ as follows

$$\|Ax - b\| = \left\| b - \sum\nolimits_{\sigma_j > 0} (u_j' b) u_j \right\|. \tag{24}$$

Now minimization of the norm $\|Ax - b\|$ leads to a solution of the system $Ax = b$ if and only if this system has a solution and in this case (23) is the solution of minimal norm. Let us verify that (23) is indeed a solution of

$$Ax = Pb = \sum\nolimits_{j \le r} (u_j' b) u_j.$$

Because of orthonormality of the $v_j$ we have $V' v_j = 0$, for $j > r$, and $V' v_j = e_j$ (the standard vector), for $j \le r$. Thus

$$V' x = \sum\nolimits_{j \le r} \frac{u_j' b}{\sigma_j} e_j.$$

The claim now follows if we observe that $D e_j = \sigma_j e_j$ and $U e_j = u_j$.

Software packages will often provide the *full* SVD (16) where $U, V, D$ are all $n \times n$ and $D$ has zeros on the main diagonal (the zero singular values of $A$). This

can be obtained from the above reduced SVD as follows: extend the columns of $U$ and $V$ to full orthonormal bases

$$\{u_1, \ldots, u_r, u_{r+1}, \ldots, u_n\} \quad \text{and} \quad \{v_1, \ldots, v_r, v_{r+1}, \ldots, v_n\},$$

write $\sigma_j = 0$, for $j = r+1, \ldots, n$, and let $\overline{U}$, $\overline{V}$ and $\overline{D}$ be the matrices

$$\overline{U} = (u_1, \ldots, u_n), \quad \overline{V} = (v_1, \ldots, v_n), \quad \text{and} \quad \overline{D} = diag(\sigma_1, \ldots, \sigma_n)$$

i.e. in block matrix form

$$\overline{U} = (U, U_c), \quad \overline{V} = (V, V_c) \quad \text{and} \quad \overline{D} = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix},$$

where $U_c = (u_{r+1}, \ldots, u_n)$, $V_c = (v_{r+1}, \ldots, v_n)$ and "0" denotes an $(n-r) \times (n-r)$ bock of zeros. With this we have an analogous decomposition

$$\overline{U}\overline{D}\overline{V}' = (U, U_c) \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V' \\ V_c' \end{pmatrix} = (U, U_c) \begin{pmatrix} DV' \\ 0 \end{pmatrix} = UDV' = A.$$

From now on we will write $U, V, D$ also for the full matrices $\overline{U}, \overline{V}, \overline{D}$ since it will be clear from the context whether the full or reduced matrices are considered.

Then we have $U'U = UU' = I_{n\times n} = V'V = VV'$ and we will now find necessary and sufficient conditions for the system $Ax = b$ to have a solution. Rewrite this system as $UDV'x = b$, equivalently $Uz = b$, where $z = DV'x$.

Here $U$ is $n \times n$ orthogonal so that the first equation always has a solution $z = U'b$. The second equation $DV'x = z$ has a solution exactly if the diagonal elements $\sigma_j$ of $D$ satisfy

$$\sigma_j = 0 \quad \Longrightarrow \quad z_j = u_j'b = 0. \tag{25}$$

In this case the general solution is given by

$$x = \sum\nolimits_j \lambda_j v_j, \quad \text{where} \quad \lambda_j = \begin{cases} \frac{u_j'b}{\sigma_j} & \text{if } \sigma_j > 0 \\ \text{arbitrary} & \text{if } \sigma_j = 0. \end{cases} \tag{26}$$

**Numerical issues.** Exact comparison to zero does not make sense in floating point arithmetic. In other words we must declare a number to be zero if its absolute value is small enough and how small that is, depends on the context (in its most trivial form: the scale of the problem, that is, the size of the numbers involved).

This makes the criterion (25) problematic (and is related to the problem of determining the rank of a matrix). Worse still the small singular values are not computed accurately (if the condition number of $A$ is bad, see [9]). Problems with (26) can thus be expected, if the inner products $u_j'b$ are not small for very small singular values $\sigma_j$.

In other words: the solution will be difficult for right hand sides $b$ with significant components in the direction of left singular vectors $u_j$ corresponding to very small singular values $\sigma_j > 0$. This makes some sort of regularization mandatory whereby small singular values are suppressed in (26). This will be treated in detail in section 5.

## 4.3 Preconditioning

The accuracy of every algorithm suffers if the condition number of the matrix $A$ (the largest singular value divided by the smallest one) is large. If the matrix $A$ is symmetric and positive definite we can try to alleviate this problem by replacing $A$ with $A + \delta I$, where $\delta > 0$ is *small*. This adds $\delta$ to each singular value (here identical with the eigenvalues) and this improves the condition number.

But what is "small"? We cannot use an absolute constant, say $\delta = 1e - 10$ since the elements of the matrix $A$ might in fact be much smaller still. To deal with this problem it is a good idea to bring the matrix closer to unit size and we do this by *diagonal scaling*, i.e. we replace the matrix $A$ with a product

$$A \quad \to DAD,$$

where $D$ is a diagonal matrix with positive values on the diagonal. This modifies the system $Ax = b$ in trivial ways that is easily handled by any solution algorithm. Note that such scaling preserves the symmetry and positive definiteness of $A$.

If the diagonal matrix $D$ is computed cleverly the scaling even improves the condition number in most cases. We use algorithm of Ruiz [8] in a manner which brings the $L^1$ and $L^\infty$ norms of the rows of $A$ close to 1.

## 4.4 Solving the KKT system

The BarrierSolver absorbs the inequality constraints into the objective function so that only the equality constraints are left in the KKT conditions. Thus we deal with this case first.

### 4.4.1 KKT system without inequality constraints

If there are no inequality constraints, the KKT system has the form

$$Hx + A'\nu = -q \tag{27}$$

$$Ax = b \tag{28}$$

Here $H$ is symmetric and positive semidefinite (convexity!) and $A$ satisfies $ncol(A) = ncol(H)$. First we assume that $H$ is balanced and positive definite. Then we proceed as follows: do a Cholesky factorization $H = LL'$. Solve

$$HX = LL'X = [A', q] \quad \text{via} \quad LY = [A', q] \text{ then } L'X = Y.$$

Obtain $H^{-1}A'$ and $H^{-1}q$. From the first equation get $x = -H^{-1}(q + A'\nu)$. Stick this into the second equation to obtain

$$AH^{-1}A'\nu = -(b + AH^{-1}q).$$

Here the matrix $S = AH^{-1}A'$ is symmetric and positive semidefinite. It is also invertible since $A$ has full rank. Thus it is positive definite and can be solved with a Cholesky factorization

$$S = R'R.$$

Moreover this matrix is small $(p \times p)$ and so this effort is negligible. We set $\mu = R\nu$ and solve for $\nu$ in 2 steps via

$$R'\mu = -(b + AH^{-1}q) \text{ then } R\nu = \mu.$$

Next we treat the general case. First we equilibrate the matrix $H$ by passing to $H \to DHD$, for a suitable diagonal matrix $D = diag(d)$ with $d_i \neq 0$, for all $i$. Setting $x = Dy$ we can rewrite (27) as

$$\begin{aligned} HDy + A'\nu &= -q \\ ADy &= b \end{aligned}$$

Multiply the first equation with $D$ on the left to get

$$My + B'\nu = -Dq \tag{29}$$
$$By = b \tag{30}$$

with $M = DHD$ and $B = AD$. This system is of the same form but with a balanced matrix $M$ instead of $H$. I.e. we apply the transformation

$$x \to y = D^{-1}x, \quad H \to DHD, \quad A \to AD, \quad q \to Dq. \tag{31}$$

After solving this new system as outlined above we have to undo the transformation $x \to D^{-1}x$ by replacing $x \to Dx$.

If now $DHD$ is not positive definite (this happens exactly if $H$ is not positive definite since the $d_i$ are all nonzero), we pass to the equivalent system

$$DHD \to DHD + rA'A, \quad -q \to -q + rA'b. \tag{32}$$

Here $DHD + rA'A$ is positive definite if and only if the KKT system (29) itself is nonsingular. If it is singular, we stop. Assume now it is nonsingular. Then the new matrix $H$ is positive definite.

### 4.4.2 Singular Hessians and KKT systems

Consider an equality constrained problem $x_0 = argminf(x)$ subject to $Ax = b$, where $x \in \mathbb{R}^n$ and fix $1 \leq j \leq n$. If the objective function does not depend on $x_j$ then the gradient $\nabla f(x)$ has a zero at the $j$th component and the Hessian $H = \nabla^2 f(x)$ has a zero row and column in position $j$, that is $row_j(H) = col_j(H) = 0$ (recall that this matrix is symmetric).

In particular $H$ is singular and any algorithm which relies on a Cholesky factorization of $H$ will fail (in particular block elimination is not possible). If the equality constraints $Ax = b$ also do not depend on the variable $x_j$, then $A$ has a zero column at position j, i.e. $col_j(A) = 0$. It follows that the KKT-matrix

$$K = \begin{pmatrix} H & A' \\ A & 0 \end{pmatrix}$$

(which is also symmetric) satisfies $row_j(K) = col_j(K) = 0$. Clearly, if neither the objective function nor the constraints depend on $x_j$, then this variable is superfluous and should be removed, that is, the $j$th row and column deleted from the KKT matrix.

If we do this we have to keep track at which positions $j$ superfluous variables have been removed and reinsert them (with an arbitrary value) at these positions after we have solved the system.

Now you will say that nobody will set up such a problem in the first place. However such problems can occur naturally during phase I analysis. Consider the inequality and equality constrained problem

$$x_0 = argmin f(x) \quad \text{subject to } g_j(x) \leq u_j \text{ and } Ax = b.$$

Here it is possible that $f(x)$ depends on a variable $x_j$ which is unconstrained, that is, none of the $g_j(x)$ depends on $x_j$ and $col_j(A) = 0$ (all coefficients of $x_j$ in $Ax = b$ are zero).
Now in a simple phase I analysis we switch to the new objective function $\tilde{f}(x, s) = s$ which depends only on the new variable $s$ leading to barrier functions

$$\tilde{f}_t(x, s) = \tilde{f}(x, s) + t \sum\nolimits_j \log(u_j + s - g_j(x)) = s + t \sum\nolimits_j \log(u_j + s - g_j(x))$$

which do not depend on the variable $x_j$ and will be minimized under the equality constraints $Ax = b$, that is exactly the case described above.

This singular Hessians $H$ can occur naturally in the barrier function of a phase I feasibility analysis if the inequality constraints do not depend on some of the variables, i.e. if some variables are unconstrained or constrained only by equality constraints (the corresponding row and column of the Hessian are then zero).

This case is dealt with by first deleting zero rows and corresponding columns from the Hessian H, the corresponding columns from $A$ and the corresponding component from $\nabla f(x)$ on the right hand side of the KKT system. If $H$ then becomes nonsingular all approaches apply. If it is singular, the solution via SVD remains viable.

# 5 Regularization of systems of equations

This section is an incomplete summary of [7]. If the matrix $A$ is ill conditioned any least squares solution $x$ computed from the normal equations

$$A^*Ax = A^*b$$

(see the discussion at the beginning of section 4) is mostly useless. If $A$ is ill conditioned, then the matrix $A^*A$ is even more ill conditioned making the solution of this system very difficult.

The same is true if we compute the minimal norm least squares solution $x_0$ as (23) using the SVD $A = UDV'$ of $A$, where $D = diag(\sigma_j)$ is a diagonal matrix with the singular values of $A$ on the main diagonal.

The problem can be seen explicitly as follows: from (23) we know that the least squares solution $x_0$ of minimal norm, i.e. the minimizer of $\|Ax - b\|$ which has itself minimal norm is given by the expression

$$x_0 = \sum\nolimits_{\sigma_j > 0} \frac{u_j' b}{\sigma_j} v_j.$$

If now $b$ has a significant component $u_j' b$ in the direction of some $u_j$ where the singular value $\sigma_j$ is very small then we are dividing a number of significant size by a very small number which can lead to error blowup.

For this reason we have to modify the matrix $A$ to avoid small singular values. The basic ill conditioned problem is numeric differentiation and this can be used to motivate the following approach: the system $Ax = b$ is reformulated as

$$Ax = b \quad \text{subject to } x = Sw, \tag{33}$$

where $S$ is a "smoothing operator". I.e. we enforce some smoothness on the solution $x$ by requiring it to be in the range of $S$. The basic example, numeric differentiation ([7], p5, eq(1)), suggests the operator

$$S = \begin{cases} (A^*A)^{p/2} & \text{if } p \text{ is odd,} \\ (A^*A)^{(p-1)/2} A^* & \text{if } p \text{ is even,} \end{cases} \tag{34}$$

which in this example is $p$-fold iterated integration. In general the integer $p$ is a parameter that controls the smoothness of the solution $x$.

Note that the range of $S$ shrinks as the parameter $p$ increases. Since (33) stipulates that the solution $x$ must come from the range of $S$ this condition becomes more and more strict as $p$ increases.

In the basic example of numeric differentiation the parameter $p$ is literally the degree of differentiability. This parameter has to be estimated. Let us note that

$$SS^* = (A^*A)^p, \quad \forall p. \tag{35}$$

Regularization of the system $Ax = b$ replaces the inverse $A^{-1}$ by a family $C_h$, $h > 0$, of matrices such that

$$C_h A \to I, \quad \text{as } h \downarrow 0,$$

where $I$ denotes the identity matrix as usual. The $C_h$ are chosen such that the constants

$$\gamma_1 = \sup\nolimits_{h>0} h \, \|C_h\| \quad \text{and} \quad \gamma_2 = \sup\nolimits_{h>0} h^{-p} \, \|(I - C_h A)S\| \,, \tag{36}$$

where $p$ is as above, are finite and not too large. Then, by definition,

$$\|C_h\| \le \gamma_1 / h \quad \text{and} \quad \|(I - C_h A)S\| \le \gamma_2 h^p. \tag{37}$$

This implies the following fundamental error bound:

**Proposition 5.1.** *If $\Delta > 0$, $x = Sw$ and $\|Ax - b\| \leq \Delta \|w\|$, then we have*

$$\|x - C_h b\| \leq ((\gamma_1 \Delta)/h + \gamma_2 h^p) \|w\| . \tag{38}$$

In other words: if the problem $Ax = b$ with $x = Sw$ is approximately solvable at all to within error $\Delta \|w\|$, then $x_0 := C_h y$ will differ from any such approximate solution $x$ by at most the right hand side of (38), meaning of course that we will use $x_0$ as a practical solution of (33).

**Proof.**

$$\begin{aligned}
\|x - C_h b\| &= \|(x - C_h Ax) + (C_h Ax - C_h b)\| \\
&= \|(I - C_h A)Sw + C_h(Ax - b)\| \\
&\leq \|(I - C_h A)S\| \|w\| + \|C_h\| \|Ax - b\| \\
&\leq \|(I - C_h A)S\| \|w\| + \Delta \|C_h\| \|w\|
\end{aligned}$$

and the claim follows by substituting the upper bounds from (37) for $\|(I - C_h A)S\|$ and $\|C_h\|$. ∎

Rewriting the system $Ax = b$ as in (33) is an *assumption* that we can solve the system with a smooth solution $x = Sw$ while the bounds (37) can be satisfied by suitable choice of the approximate inverses $C_h$ of $A$.

The optimal parameter $h > 0$ for the (approximate) solution $x_0 = C_h b$ is then computed by minimizing the right hand side

$$RHS(h) := (\gamma_1 \Delta)/h + \gamma_2 h^p$$

of (38). Setting the derivative $RHS'(h)$ equal to zero immediately yields the following equality for the optimal $h$

$$p\gamma_2 h^{p+1} = \gamma_1 \Delta \tag{39}$$

that is, explicitly,

$$h = \left( \frac{\gamma_1 \Delta}{p\gamma_2} \right)^{1/(p+1)} .$$

Using (39) to simplify $RHS(h)$ for the optimal $h$ we obtain

$$RHS(h) = \gamma_2 (p+1) h^p$$

so that (38) simplifies to

$$\|x - C_h b\| \leq \gamma_2 (p+1) h^p \|w\| = \kappa \Delta^{p/(p+1)} = O(\Delta^{p/(p+1)}) \tag{40}$$

with a constant $\kappa$ given by

$$\kappa = \|w\| \gamma_2 (p+1) \left( \frac{\gamma_1}{p\gamma_2} \right)^{p/(p+1)} .$$

Here we are interested in the behavior as $\Delta \to 0$. Note that an exact solution $x$ of the system $Ax = b$ with $x = Sw$ satisfies the assumptions of the above

proposition for all $\Delta > 0$ but the discussion applies also to the case where this system does not have an exact solution.

To construct suitable approximate inverses $C_h$ we use the following proposition:

**Proposition 5.2.** *Set $Q = A^*A$, let $S$ be any (smoothing) matrix satisfying $SS^* = (A^*A)^p = Q^p$ and $\phi : [0, +\infty) \to \mathbb{R}$ any continuous function satisfying*

$$\gamma_1 := \sup_{t>0} |\phi(t)| t^{1/2} < +\infty \quad \textit{and} \quad \gamma_2 := \sup_{t>0} |1 - t\phi(t)| t^{p/2} < +\infty \tag{41}$$

*Then the family*

$$C_h := h^{-2}\phi(h^{-2}Q)A^*, \quad h > 0, \tag{42}$$

*satisfies the conditions (36) with the very same constants $\gamma_1$ and $\gamma_2$ (for the operator norm with respect to the $L^2$-norm).*

**Proof.** The square of the operator norm of a matrix $B$ with respect to the $L^2$-norm is the largest eigenvalue of $BB^*$. Note that $\phi(h^{-2}Q)$ is self-adjoint and that

$$C_h A = h^{-2}\phi(h^{-2}A^*A)A^*A = h^{-2}\phi(h^{-2}Q)Q.$$

Set $R = I - C_h A = I - h^{-2}\phi(h^{-2}Q)Q$. Note that $R$ is self-adjoint. With this we obtain $\|C_h\|^2$ and $\|(I - C_h A)S\| = \|RS\|$ as the largest eigenvalue of

$$\begin{aligned} C_h C_h^* &= h^{-4}\phi(h^{-2}Q)A^*A\phi(h^{-2}Q) = h^{-4}Q\phi^2(h^{-2}Q) \\ &= \psi_h(Q), \end{aligned}$$

where $\psi_h(u) = h^{-4}u\phi^2(h^{-2}u)$, respectively

$$\begin{aligned} (RS)(RS)^* &= R(SS^*)R = RQR \\ &= [I - h^{-2}\phi(h^{-2}Q)Q]Q[I - h^{-2}\phi(h^{-2}Q)Q] \\ &= \chi_h(Q), \end{aligned}$$

where $\chi_h(u) = [1 - h^{-2}u\phi(h^{-2}u)]^2 u$. Let $\lambda_i$ denote the eigenvalues of $Q = A^*A$. Then the eigenvalues of $C_h C_h^*$ are the numbers $\psi_h(\lambda_i)$ and the eigenvalues of $(RS)^*(RS)$ are the numbers $\chi_h(\lambda_i)$. The claim follows if we can show that

$$\sup_h \sup_i \psi_h(\lambda_i) < \infty \quad \text{and} \quad \sup_h \sup_i \chi_h(\lambda_i) < \infty.$$

Since there are only finitely many $\lambda_i$ it will suffice to show that

$$\sup_h \psi_h(u) < \infty \quad \text{and} \quad \sup_h \chi_h(u) < \infty$$

for each $u > 0$ and this follows from the assumptions on the function $\phi$. ∎

**Example 5.1.** Tichonov regularization Here $\phi(u) = 1/(u+1)$ and

$$C_h = h^{-2}(I + h^{-2}A^*A)^{-1}A^* = (A^*A + h^{-2}I)^{-1}A^*.$$

For this $\phi$ one computes the constants $\gamma_i$ from proposition 5.2 as $\gamma_1 = 1/2$ and

$$\gamma_2 = \begin{cases} 1/2 & \text{if } p = 1, \\ 1 & \text{if } p = 2 \quad \text{and} \\ +\infty & \text{if } p > 2. \end{cases}$$

In particular Tichonov regularization cannot take advantage of smoothness of degree $p > 2$.

**Example 5.2.** Iterated Tichonov regularization Let $l > 0$ be an integer. Then the function

$$\phi_l(u) = t^{-1}(1 - (1+t)^{-l}$$

Corresponds to $l$-times iterated Tichonov regularization ([7], p10, proposition 5.2). It has finite $\gamma$-constants up to smoothness of degree $p = 2l$.

## 5.1 Using the SVD of $A$

Using the SVD decomposition $A = UDV'$, where $D = diag(\sigma_j)$ is the diagonal matrix containing the singular values of $A$, we can compute the matrix function $\phi(h^{-2}Q)$ in explicit form.

Note first that $Q = A^*A = VDU'UDV' = VD^2V'$ since the orthogonal matrix $U$ satisfies $U'U = I$. From this it follows that $Q^k = V(D^2)^kV'$ for each integer power $k \geq 0$, whence $p(Q) = Vp(D^2)V'$ for each polynomial $p = p(u)$ where $p$ acts on the diagonal elements of the diagonal matrix $D^2$, that is,

$$p(D^2) = diag(p(\sigma_j^2)).$$

Approximation by polynomials then yields

$$\phi(Q) = V\phi(D^2)V' = Vdiag(\phi(\sigma_j^2))V' \tag{43}$$

for each continuous function $\phi : [0, +\infty) \to \mathbb{R}$. Because of the compactness of the spectrum we need to approximate only on compact sets. In particular

$$\phi(h^{-2}Q) = V\phi(D^2)V' = Vdiag(\phi(\sigma_j^2/h^2))V', \quad h > 0 \tag{44}$$

(replace $\phi(u)$ with $\phi(h^{-2}u)$ for fixed $h > 0$).

## 5.2 Stochastic formulation

The constants $\gamma_1$ and $\gamma_2$ can be computed directly from our choice of the function $\phi$ in the construction of the approximate inverses $C_h$ of $A$. However we also need

the smoothness parameter $p$ and given $p$, the optimal parameter value $h$ from (39) and for this we need the parameter $\Delta$.

$\Delta$ turns out to be hard to estimate in general so we turn to a reformulation of the problem in stochastic terms. We write

$$Ax = b + \epsilon \quad \text{and} \quad x = Sw, \tag{45}$$

where now $\epsilon$ and $w$ are *mean zero* random vectors with pairwise uncorrelated components

$$E(\epsilon w') = 0, \quad E(\epsilon\epsilon') = \sigma^2 I, \quad \text{and} \quad E(ww') = \tau^2 I.$$

We then set

$$\Delta = \sigma/\tau$$

for the level or relative noise (input / output) of the system. Here the assumption of zero mean for the variable $w$ is a bit jarring but needed to support the following proceedings. This quantity $\Delta$ is not related to the quantity $\Delta$ in the preceding discussion.

With these assumptions one can show ([7], theorem 8.1, p14) that the expected error $E\,\|x - Cy\|$ is minimized for the choice of approximate inverse $C$ of $A$ given by

$$C := (SS^*A^*A + \Delta^2 I)^{-1} SS^*A^* \tag{46}$$

with no assumptions on the smoothing matrix $S$. For this $C$ we can compute the optimal estimator $x = Cb$ as follows: setting $P = AS$ we have

$$x = Sw, \quad \text{with} \quad (P * P + \Delta^2 I)w = P^*b. \tag{47}$$

This still leaves open the question how the new parameter $\Delta$ should be estimated form the right hand side $y$ but this is now a simpler problem.

Typically the noise level $\sigma$ in the right hand side $b$ will be small while the noise level $\tau$ for $w$ will be large (recall $w$ centered at zero) so that the quotient $\Delta = \sigma/\tau$ is small. This is desirable so that the regularization term $\Delta^2 I$ does not dominate the computation.

Here the degree of smoothness is implicit in the smoothness matrix $S$. In the particular case of $S$ as in (34) it is given by the parameter $p$ and should be estimated rather than chosen arbitrarily.

For this particular smoothing matrix $S$, using the SVD decomposition $A = UDV'$ with $D = diag(\sigma_j)$ the computation of the optimal estimator $x = Cb$ in (47) can be simplified as follows:

$$x = Vz \quad \text{where } z_j = \frac{\sigma_j^{2p+1} c_j}{\sigma_j^{2p+2} + \Delta^2} \tag{48}$$

where the vector $c = (c_j)$ is given by $c = U'b$. In other words we have

$$x = \sum\nolimits_j \frac{\sigma_j^{2p+1} u_j' b}{\sigma_j^{2p+2} + \Delta^2} v_j \tag{49}$$

where $u_j = col_j(U)$ and $v_j = col_j(V)$. Clearly we need to sum only over the nonzero singular values $\sigma_j > 0$.

**Estimation of $\Delta$.**

Let $c$ be the vector $c = U'b$ above. From $b = Ax + \epsilon = ASw + \epsilon$ we see that $b$ has mean zero and from the assumptions on the correlations between the $\epsilon_i$ and $w_j$ we get

$$Cov(b) = E(bb') = E[(ASw + \epsilon)(w'S'A' + \epsilon')] = \tau^2 ASS'A' + \sigma^2 I. \quad (50)$$

Thus $c = U'b$ also has mean zero and covariance matrix

$$Cov(c) = E(cc') = U'Cov(b)U.$$

Observing that $U'U = I$, $V'V = I$ and recalling that $SS' = (A'A)^p$ and $AA' = UDV'VDU' = UD^2U'$ we have

$$ASS'A' = (AA')^{p+1} = (UD^2U')^{p+1} = UD^{2p+2}U'$$

so that $U'(ASS'A')U$ is the diagonal matrix $D^{2p+2}$. Now it follows from (50) that

$$Cov(c) = U'Cov(b)U = \tau^2 D^{2p+2} + \sigma^2 I \quad (51)$$

and in particular

$$E(c_j^2) = \tau^2 \sigma_j^{2p+2} + \sigma^2 \quad (52)$$

and we have one realization of the (many) variables $c_j$ but only two unknown parameters $\tau$ and $\sigma$ to be estimated.

## 5.3 Parameter estimation

Now we turn to the following problem ([7], p18, section 10): We have *a single* realization of a high dimensional random vector $X = (X_1, \ldots, X_n)$ known to satisfy

$$EX = v(\theta), \quad \text{that is, } EX_j = v_j(\theta), \quad (53)$$

where $v(\theta) = (v_1(\theta), \ldots, v_n(\theta))$ is a function of a low dimensional parameter $\theta$. We need to estimate the parameter $\theta$ from the one realization of $X$ which we have.

In this situation we have only a few unknown values $\theta_k$ and many equations $EX_j = v_j(\theta)$ to work with but have only one realization of $X$ to provide information about the mean $EX$. For this problem Neumaier derives the following approach ([7], p18, section 10):

**Theorem 5.3.** *Let $\theta^*$ be any value of the parameter satisfying $EX = v(\theta^*)$, $\psi : I \to \mathbb{R}$ a strictly concave, differentiable function defined on an interval $I$ containing the range of all the functions $v_j(\theta)$ and set*

$$f(x,\theta) := \sum\nolimits_j \alpha_j[\psi(v_j(\theta) + \psi'(v_j(\theta)(x_j - v_j(\theta)]$$

*with positive weights $\alpha_j$. Then the expectation $g(\theta) := Ef(X,\theta)$ is bounded below and any global minimizer $\hat{\theta}$ of $g$ satisfies*

$$v_j(\hat{\theta}) = v_j(\theta^*), \quad \forall j = 1, \ldots, n.$$

**Remark.** The idea here is that we will replace the unknown function $g(\theta)$ with $f(x,\theta)$, where $x$ is the single realization of $X$ which we have and then minimize $f(x,\theta)$ instead. Since $f$ is linear in the variable $x$ we have

$$g(\theta) = f(EX,\theta) = \sum\nolimits_j \alpha_j[\psi(v_j(\theta) + \psi'(v_j(\theta)(EX_j - v_j(\theta)] \tag{54}$$

and this approach has some merit if many of the realizations $x_j$ are close to the mean $EX_j$.

If the values $v_j(\theta)$, $j = 1, \ldots, n$, determine the parameter $\theta$ uniquely, the conclusion of the theorem can be strengthened to $\hat{\theta} = \theta^*$, that is, the minimization of $g(\theta)$ identifies the true parameter $\theta^*$.

**Proof.** The strict concavity of $\psi$ implies that at any point $z \in I$ the graph of $\psi$ is below its tangent at the point $z$ and is strictly below this tangent at all points $w \neq z$. More formally we have

$$f(w) \leq f(z) + f'(z)(w - z)), \quad z, w \in I, \tag{55}$$

with equality only in case of $w = z$. Now, for any $\theta$ we have

$$\begin{aligned} g(\theta) &= \sum\nolimits_j \alpha_j[\psi(v_j(\theta) + \psi'(v_j(\theta)(EX_j - v_j(\theta)] \\ &= \sum\nolimits_j \alpha_j[\psi(v_j(\theta) + \psi'(v_j(\theta)(v_j(\theta^*) - v_j(\theta)] \end{aligned}$$

and in particular $g(\theta^*) = \sum_j \alpha_j \psi(v_j(\theta^*))$. Subtraction yields

$$g(\theta) - g(\theta^*) = \sum\nolimits_j \alpha_j[\psi(v_j(\theta) + \psi'(v_j(\theta)(v_j(\theta^*) - v_j(\theta)) - \psi(v_j(\theta^*))],$$

where each expression in square brackets is nonnegative and is strictly positive if $v_j(\theta) \neq v_j(\theta^*)$. I.e.

$$g(\theta) \geq g(\theta^*)$$

with equality if and only if $v_j(\theta) = v_j(\theta^*)$, for all $j = 1, \ldots, n$. ∎

### 5.3.1 Estimation of $\tau$ and $\Delta$

Now we apply this to our problem. Here $X = (c_1^2, \ldots, c_r^2)$, where $c$ is the vector $c = U'b$ in the problem $Ax = b$ and $A = UDV'$ is the SVD of the matrix $A$. Recall that we had

$$EX_j = Ec_j^2 = \sigma^2 + \tau^2\sigma_j^{2(p+1)}$$

and we will treat the slightly more general case

$$EX_j = Ec_j^2 = v_j(\sigma, \tau) := \sigma^2\mu_j + \tau^2\lambda_j \tag{56}$$

with given constants $\mu_j$ and $\lambda_j$, $j = 1, \ldots, r = rank(A)$. The special case we are interested in is the case

$$\mu_j = 1 \quad \text{and} \quad \lambda_j = \sigma_j^{2(p+1)}. \tag{57}$$

Recall that we need only the parameter

$$t := \Delta^2 = \sigma^2/\tau^2 \tag{58}$$

which is positive and we can thus work with the function $\psi(u) = \log(u)$ in theorem 5.3. Fix weights $\alpha_j > 0$ and normalize them so that

$$\alpha := \sum\nolimits_j \alpha_j = 1. \tag{59}$$

We now apply theorem 5.3 with

$$v_j(\theta) = v_j(\sigma^2, \tau^2) = \sigma^2\mu_j + \tau^2\lambda_j = \tau^2[\lambda_j + t\mu_j].$$

Write $f(c, \theta)$ instead of $f(X, \theta)$. Then

$$\begin{aligned} g(\theta) = Ef(c, \theta) &= \sum\nolimits_j \alpha_j[\log(v_j(\theta)) + v_j(\theta)^{-1}(Ec_j^2 - v_j(\theta)] \\ &= \alpha + \alpha\log(\tau^2) + \sum\nolimits_j \alpha_j \log(\lambda_j + t\mu_j) + \tau^{-2}\sum\nolimits_j \alpha_j \frac{Ec_j^2}{\lambda_j + t\mu_j} \\ &:= g(\tau, t). \end{aligned}$$

Here we have reparameterized our problem in terms of the variables $\tau$ and $t = \sigma^2/\tau^2$ and need to minimize the function $g(\tau, t)$. To do this we will set the partial derivative $\partial g/\partial\tau$ equal to zero and solve for $\tau$ in terms of $t$ yielding a path $\tau = \tau(t)$. The minimum of $g$ is assumed along that path and we have thus reduced the problem to the minimization of a function

$$g(t) = g(t, \tau(t))$$

of one variable $t$ only. Indeed, using $\alpha = 1$, we have

$$0 = \partial g/\partial\tau = \frac{2}{\tau} - \frac{2}{\tau^3}\sum\nolimits_j \alpha_j \frac{Ec_j^2}{\lambda_j + t\mu_j}$$

yielding

$$\tau^2 = \sum\nolimits_j \alpha_j \frac{Ec_j^2}{\lambda_j + t\mu_j}. \tag{60}$$

Putting this into the function $g(\tau, t)$ yields

$$g(t) = 2 + \sum\nolimits_j \alpha_j \log(\lambda_j + t\mu_j) + \log\left[\sum\nolimits_j \alpha_j \frac{Ec_j^2}{\lambda_j + t\mu_j}\right]. \tag{61}$$

From scaling arguments Neumaier makes the case that the weights $\alpha_j$ should all be equal (hence $\alpha_j = 1/r$, $j = 1, \ldots, r$), so that (neglecting constant summands) the objective function assumes the form

$$\hat{g}(t) = \log Q(t) + R(t) \quad \text{where} \tag{62}$$

$$Q(t) = \sum\nolimits_j \frac{Ec_j^2}{\lambda_j + t\mu_j} \quad \text{and} \tag{63}$$

$$R(t) = \frac{1}{r} \sum\nolimits_{j=1}^{r} \log(\lambda_j + t\mu_j). \tag{64}$$

This function is called the *generalized maximum likelihood* (GML) score function. Here $r = rank(A)$ is the number of nonzero singular values of $A$.

In practice the expectations $Ec_j^2$ are replaced with the realizations $c_j^2$ and the function $g(t)$ above minimized numerically. We will do this by setting the derivative $g'(t) = 0$ and solve this equation using Newton's method. As a suitable starting point for the minimization Neumaier gives

$$t := median(\lambda_j/\mu_j)$$

which is $t = median(\sigma_j^{2(p+1)})$ in our case. To solve $\hat{g}'(t) = 0$ with Newtons method we need both derivatives $\hat{g}'(t)$ and $\hat{g}''(t)$. Note first that

$$\hat{g}'(t) = \frac{Q'(t)}{Q(t)} + R'(t) \quad \text{where} \tag{65}$$

$$Q'(t) = -\sum\nolimits_j \frac{\mu_j Ec_j^2}{(\lambda_j + \mu_j t)^2} \quad \text{and} \tag{66}$$

$$R'(t) = \frac{1}{r} \sum\nolimits_j \frac{\mu_j}{\lambda_j + t\mu_j}. \tag{67}$$

From this it follows that

$$\hat{g}''(t) = \frac{Q''(t)Q(t) - Q'(t)^2}{Q(t)^2} + R''(t) \quad \text{where} \tag{68}$$

$$Q''(t) = 2\sum\nolimits_j \frac{\mu_j^2 Ec_j^2}{(\lambda_j + \mu_j t)^3} \quad \text{and} \tag{69}$$

$$R''(t) = -\frac{1}{r} \sum\nolimits_j \frac{\mu_j^2}{(\lambda_j + t\mu_j)^2}. \tag{70}$$

It is also recommended that the smoothness parameter $p$ should be estimated by minimizing $\hat{g}(t)$ as a function of $p$ also, although that is not supported by developments in the paper.

**Concluding remarks.** The motivation of the choice of smoothing matrix $S$ from the problem of numeric differentiation is reasonable. This leads to the suppression of the singular values in the standard solution (49) by means of powers $\sigma_j^{2(p+1)}$.

On the other hand the stochastic heuristic for choosing the regularization parameter $\Delta$ is rather weak and nonexistent for choosing the smoothing parameter $p$. Here we could simply try solutions for various values of these parameters and check which ones produce the best results in terms of $\|Ax - b\|$. These are $O(n^2)$ operations whereas the SVD is much more expensive than that.

In that case the above development could be used to locate an interval for $t$ in which a grid search for the optimal solution of $Ax = b$ is conducted.

## 5.4 Tichonov regularization and trust regions

The quadratic approximation of the objective function $f(x)$ at iterate $x_k$ is

$$\tilde{f}(x_k + \Delta x) = f(x_k) + \nabla f(x_k)'\Delta x + \tfrac{1}{2}\Delta x' H_k \Delta x,$$

where $H_k$ is the Hessian $\nabla^2 f(x_k)$ or an approximation thereof. The search direction $\Delta x$ from iterate $x_k$ is computed by minimizing this function over the variable $\Delta x$ resulting in the equation

$$H_k \Delta x = -\nabla f(x_k) := -y_k \tag{71}$$

for the search direction $\Delta x$. Clearly we can assume that $y_k = \nabla f(x_k) \neq 0$ since the search terminates at a zero gradient .

In our algorithms we can also assume that $H_k$ is positive semidefinite, but not necessarily nonsingular. If $H_k$ is nonsingular (i.e. positive definite), then the solution $\Delta x$ is a *descent direction*:

$$\Delta x' \nabla f(x_k) = -\Delta x' H_k \Delta x < 0$$

and this is all we care about. We will solve (71) by Cholesky factorization $H_k = LL'$ with lower triangular $L$, by solving the triangular systems

$$Lv = -y_k, \quad L'\Delta x = v. \tag{72}$$

The Cholesky factorization fails if $H_k$ is singular. In that case we replace $H_k$ with $H_k(\delta) := H_k + \delta I$, for some positive constant $\delta$. This matrix is now positive definite, in fact

$$(H_k(\delta)u, u) = u' H_k(\delta) u = u' H_k u + \delta u' I u \geq \delta \|u\|^2$$

so that (71) now yields a descent direction. Moreover it improves the conditioning of (71): if $H_k + \delta I = L(\delta)L(\delta)'$ is the Cholesky factorization of $H_k(\delta)$, then we have

$$|L(\delta)_{ii}| \geq \delta.$$

Indeed, the diagonal element $L(\delta)_{ii}$ is an eigenvalue of the triangular matrix $L(\delta)'$. Let $u$ be a corresponding eigenvector. Then we have

$$|L(\delta)_{ii}|^2 \|u\|^2 = \|L(\delta)'u\|^2 = (L(\delta)L(\delta)'u, u) = ((H_k(\delta)u, u) \geq \delta \|u\|^2$$

from which it follows that

$$|L(\delta)_{ii}| \geq \sqrt{\delta}$$

with obvious implications for the numerical stability of the triangular systems (72). Moreover this suggests that we should replace $H_k$ with $H_k + \delta I$ not only if the Cholesky factorization fails but rahter as soon as the minimal diagonal element (in absolute value) of the Cholesky factor $L$ is below the threshold $\sqrt{\delta}$.

**Trust region interpretation.** The passage from the matrix $H_k$ to the regularization $H_k + \delta I$ has an interpretation in terms of *trust regions*: the solution $\Delta x^*$ of

$$H_k(\delta)\Delta x = -y_k, \quad \text{where } y_k = \nabla f(x_k),$$

is the minimizer of the quadratic function

$$\begin{aligned}\phi(\Delta x) &= f(x_k) + y_k'\Delta x + \Delta x' H_k(\delta)\Delta x \\ &= f(x_k) + y_k'\Delta x + \Delta x' H_k \Delta x + \delta \|\Delta x\|^2 \\ &= \tilde{f}(x + \Delta x) + \delta \|\Delta x\|^2 .\end{aligned}$$

Now note that this minimizer $\Delta x^*$ is automatically also the minimizer of the quadratic approximation $\tilde{f}(x_k + \Delta x)$ on the ball $B(x_k, r_k)$ with radius $r_k = \|\Delta x^*\|$. Indeed, if this ball contained a point $u$ with $\tilde{f}(x_k + u) < \tilde{f}(x_k + \Delta x^*)$, then, since also $\|u\| \leq \|\Delta x^*\|$ it follows that

$$\phi(u) = \tilde{f}(x_k + u) + \delta \|u\|^2 < \tilde{f}(x_k + \Delta x^*) + \delta \|\Delta x^*\|^2 = \phi(\Delta x^*).$$

In other words: passing from $H_k$ to $H_k + \delta I$ we compute the search direction $\Delta x$ by minimizing the quadratic approximation $\tilde{f}(x_k + \Delta x)$ not globally but instead on the ball $B(x_k, r_k)$ (the region in which we trust the approximation) where the trust radius $r_k$ is defined implicitly as $r_k = \|\Delta x^*\|$.

This indicates that the regularization $H_k \to H_k(\delta)$ is not unreasonable and in any case it solves the problem of non singularity of $H_k$ for us, improves the conditioning and results in a descent direction $\Delta x$ at iterate $x_k$.

# 6 Kullback-Leibler distance

Consider the uniform discrete probability distribution $p = (p_j)_{1 \leq j \leq n}$ on the set $\Omega = \{\, \omega_1, \omega_2, \ldots, \omega_n \,\}$:

$$p_j = P(\{\omega_j\}) = 1/n.$$

If $x = (x_j)$ with $x_j > 0$ and $\sum x_j = 1$ is another probability distribution on $\Omega$, then the *Kullback-Leibler* distance $d_{KL}(x, p)$ of $x$ from $p$ is defined as

$$d_{KL}(x, p) = \sum\nolimits_j p_j \log(p_j/x_j) = -log(n) - \frac{1}{n}\sum\nolimits_j \log(x_j). \tag{73}$$

This function is convex in the variable $x$ and also symmetric in $x$. The symmetry uses the fact that the $p_j$ are all equal and will be used for the analytic solution of the minimization problems below. Note that we have

$$\nabla d_{KL}(x,p) = -\tfrac{1}{n}(1/x_1, 1/x_2, \ldots, 1/x_n)' \quad \text{and} \tag{74}$$

$$\nabla^2 d_{KL}(x,p) = \tfrac{1}{n} diag(1/x_1^2, 1/x_2^2, \ldots, 1/x_n^2), \tag{75}$$

where $diag(\lambda)$ denotes the diagonal matrix with the vector $\lambda$ on the diagonal as usual.

## 6.1 Minimization of $d_{KL}$ under probability constraints

Now let $A_k \subseteq \Omega$, $k = 1, \ldots, m$ be *disjoint* events (subsets) and consider the convex minimization problem

$$x^* = argmin\{\, d_{KL}(x,p) \,:\, P^x(A_k) = q_k \,\}. \tag{76}$$

Here $P^x(A) = E^x[1_A]$ denotes the probability of the event $A$ under the discrete probability distribution $x = (x_j)$ on the set $\Omega$. Note that a constraint on the probabilities $x$ of the form $P^x(A) = r$ has the form

$$r = P^x(A) = \sum\nolimits_j x_j 1_A(\omega_j)$$

and is therefore a linear constraint in the variable $x$. Moreover the right hand side is a symmetric function of the variables $x_j$. Consequently the solution $x^*$ of (76) must be symmetric under all permutations of coordinates which leave the sets $A_k$ invariant, in other words the probability function

$$x^* : \omega_j \mapsto x_j^* = P^*(\omega_j)$$

is constant on all the sets $A_k$ as well as the complement $D = [\cup A_k]^c$. This uses the fact that the $A_k$ are disjoint since this implies that points $\omega \in \Omega$ which are in the same set $A_k$ or are in $D$ cannot be distinguished by the conditions $\omega \in A_k$ (i.e. if it is only determined in which of the sets $A_k$ they are).

More formally the system of constraints

$$r_k = P^x(A_k) = \sum\nolimits_j x_j 1_{A_k}(\omega_j) \tag{77}$$

is invariant under all permutations of the variables $x_j$ which (when applied to the points $\omega_j$) leave the sets $A_k$ invariant. Thus the solution $x^*$ has the form

$$x_j^* = \begin{cases} q_k & \text{if } j \in A_k \\ q_* & \text{if } j \in D \end{cases}$$

and the variables $q_k, q_*$ can be computed from the following system of equations

$$r_k = P^{x^*}(A_k) = q_k |A_k|$$

$$1 - \sum\nolimits_k r_k = P^{x^*}(D) = q_* |D|$$

or explicitly

$$x_j^* = \begin{cases} r_k/|A_k| & \text{if } \omega_j \in A_k \\ \frac{1}{|D|}\left(1 - \sum_k r_k\right) & \text{if } \omega_j \in D. \end{cases} \tag{78}$$

Here $|D|$ denotes the cardinality of the set $D$ as usual.

# 7 Problem transformations

A convex problem can often be transformed into an equivalent one which is more amenable to solution (e.g from non differentiable to differentiable, even linear).

A device which is often used is the introduction of new variables (so called "slack variables"). The number of Newton steps of interior point methods does not increase much if the problem dimension is increased and the complexity of the equation solving does not increase much either if the new matrices are sparse and the sparsity is used by the algorithm. Below are two examples that show what is possible:

**Example 7.1.** $L^1$-norm. The $L^1$-norm $f(x) = \|x\|_1 = \sum_j |x_j|$ is plainly non differentiable. Suppose we want to minimize $f(x)$ given some constraints $Cts$. Then we can introduce additional variables $t_j$ and minimize

$$\tilde{f}(x,t) = \sum\nolimits_j t_j$$

subject to the constraints $Cts$ and additional constraints

$$t \geq 0 \quad \text{and} \quad -t \leq x \leq t. \tag{79}$$

In similar fashion a constraint of the form $\|x\|_1 \leq C$ can be replaced with $\sum_j t_j \leq C$ and (79).

**Example 7.2.** $|\cdot|_L$-norm. A more interesting example is the following: fix a positive integer $L \leq n$. For $x \in \mathbb{R}^n$ define the *L-largest norm* $|x|_L$ as follows: order the absolute values $|x_j|$ in increasing order and let $|x|_L$ be the sum of the $L$-largest of these.

This is easily seen to define a norm on $\mathbb{R}^n$. This norm is relevant for example if we want to minimize the sum of the $L$ largest residuals $|(Ax)_i - b_i|$ in a general linear equation $Ax = b$ without exact solution.

Note that if we minimize $|Ax - b|_L$ instead of $\|Ax - b\|_1$, we will be able to push the sum of the $L$ largest residuals down more at the expense of an increase in the remaining residuals which however, by definition are smaller!

We claim that the problem

$$\text{minimize } |x|_L \quad \text{subject to constraints } Cts \tag{80}$$

is equivalent to the following problem: introduce new variables $t$ of the same dimension as $x$ (one $t_j$ for each $x_j$) and one new scalar variable $\rho$ and minimize the new objective function

$$\hat{f}(x,t,\rho) = L\rho + \sum\nolimits_j t_j \tag{81}$$

subject to the constraints $Cts$ augmented with the constraints $t, \rho \geq 0$ and $-t_j - \rho \leq x_j \leq t_j + \rho$, that is, $|x_j| \leq t_j + \rho$.

Note first that for all $x, t, \rho$ satisfying these constraints we trivially have $\hat{f}(x,t,\rho) \geq |x|_L$.

Conversely, for any $x \in \mathbb{R}^n$ let $\rho$ be the $(L+1)$-largest of the $|x_j|$ and set

$$t_j := \begin{cases} 0 & \text{if } |x_j| \leq \rho \\ |x_j| - \rho & \text{if } |x_j| > \rho \end{cases}$$

Plainly then $\rho$ and $t$ satisfy the new constraints and $t_j = 0$ if and only if $|x_j|$ is not among the $L$ largest absolute values $|x_k|$. It follows that

$$\hat{f}(x,t,\rho) = L\rho + \sum\nolimits_j t_j = |x|_L$$

The claim follows easily from this. In similar fashion a constraint of the form $|x|_L \leq C$ is replaced with a linear constraint $L\rho + \sum_j t_j \leq C$.

For the systematic elaboration of such ideas see [6].

# 8 Code Structure

Currently only the barrier method is implemented. A *BarrierSolver* expects a feasible point as a starting point for the optimization. Consequently an *OptimizationProblem* needs to be allocated with a feasible starting point. If none is supplied the *ConstraintSet* will conduct a simple *Phase I analysis* to find such a point (or determine that the constraints are infeasible or fail on both counts).

For this the ConstraintSet needs to be provided with a point $x = a$ at which all the constraints $g_j(x) <= u_j$ are defined. The search for a feasible point of the original constraints is another optimization problem with new objective function and constraints for which $a$ is a feasible point.

The ConstraintSet collects only the inequality constraints. Wether or not an *EqualityConstraint* of the form $Ax = b$ is also present is optional. This constraint is then handed to the various functions as a parameter.

Once a feasible starting point for the original problem is at hand the OptimizationProblem allocates a BarrierSolver with this starting point. The BarrierSolver then tries to move along the central path to a solution.

To do this it allocates a barrier function $b(t,x)$ depending on a parameter $t$ and the decision variables $x$ of the problem. This function is a new objective

function which absorbs the inequality constraints of the original problem. The points $x(t)$ along the central path

$$t \mapsto x(t)$$

are computed by minimizing the barrier function $b(t, x)$ in the variable $x$. The central path moves toward the solution as $t \uparrow \infty$ and the BarrierSolver controls the loop over increasing values of the parameter $t$.

For each fixed value of $t$ the minimization of the barrier function $b(t, x)$ is constrained only by the equality constraints $Ax = b$ of the problem. If there are no such constraints, the BarrierSolver hands off the minimization to an *UnconstrainedSolver*, otherwise it is handed off to an *EqualityConstrainedSolver*.

# 9 Pitfalls

## 9.1 Tolerance in the BarrierSolver too small

Recall that the BarrierSolver increases the parameter $t$ along the central path until the upper bound on the duality gap

$$\text{dualityGap} \ \leq \#(\text{InequalityConstraints})/t < tol$$

is below the tolerance *tol* which we set. The significance of this tolerance lies in the following inequality

$$f(x) < \min(f) + tol, \tag{82}$$

where $f$ is the objective function and $x = x(t)$ the point on the central path corresponding to the parameter $t$. Now as this parameter $t$ increases the KKT system becomes increasingly ill conditioned up to a point where it is classified as singular and can no longer be solved.

The solution to this problem is to increase the tolerance and live with a larger duality gap and weaker estimate (82).

# References


[1] Arkadi Nemirovski, *Interior Point Polynomial Time Methods In Convex Programming*, Lecture Notes, Georgia Institute of Technology, 2004,
`http://www2.isye.gatech.edu/~nemirovs/Lect_IPM.pdf`

[2] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004, ISBN: 9780521833783,
`https://web.stanford.edu/~boyd/cvxbook/`

[3] L. Vandenberghe and S. Boyd, *Semidefinite Programming*, SIAM Review, 38(1): 49-95, March 1996,
`http://web.stanford.edu/~boyd/papers/sdp.html`

[4] Grant, Michael C., *Disciplined Convex Programming*, PhD. thesis, Stanford University, 2004.
`https://web.stanford.edu/~boyd/papers/disc_cvx_prog.html`

[5] Disciplined Convex Programming, Stanford website,
`http://dcp.stanford.edu/`

[6] M.C. Grant, S.P. Boyd, *Graph Implementations for Nonsmooth Convex Programs*,
`https://www.stanford.edu/~boyd/papers/pdf/graph_dcp.pdf`

[7] Arnold Neumaier, *Solving ill conditioned and singular systems, a tutorial on regularization*, Department of Mathematics, University of Vienna,
`https://www.mat.univie.ac.at/~neum/regul.html`,

[8] Philip A. Knight, Daniel Ruiz, Bora Ucar, *A Symmetry Preserving Algorithm for Matrix Scaling*, SIAM Journal on Matrix Analysis and Applications, Society for Industrial and Applied Mathematics, 2014, 35 (3), pp.25.
`https://hal.inria.fr/inria-00569250/file/symmetryPreservingScaling-82575.pdf`

[9] James Demmel,Ming Guy, Stanley Eisenstatz, Ivan Slapnicarx, Kresimir Veselic, Zlatko Drmack,
*Computing the Singular Value Decomposition with High Relative Accuracy*,
LAPACK Working Note 119, CS-97-348, February 11, 1997
`http://www.netlib.org/lapack/lawnspdf/lawn119.pdf`