CVX Implementation Notes

Michael J. Meyer

February 4, 2017

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

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

If the vector ν 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 ν is a certificate of infeasibility. Some solution algorithms yield such ν as a byproduct of the computation.

Indeed, for any such ν 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 grey zone remains in which the problem is feasible but this cannot be detected by our algorithms.

2 Handling the equality constraints

The equality constraints $Ax = b$ always form an underdetermined system. The solution set is an affine subspace of the space 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 $x \in \mathbb{R}^n \rightarrow u \in \mathbb{R}^{n-p}$ leading to a reduction in dimension. However this change of variables

affects the computation of gradients and Hessian matrices:

$$\nabla_u h(z+Fu) = F'\nabla_x h(z+Fu) \quad \text{and} \tag{3}$$
$$\nabla_u^2 h(z+Fu) = F'\nabla_x^2 h(z+Fu)F \tag{4}$$

Here F is an $n \times p$-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 Regularization

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 Δx from iterate x_k is computed by minimizing this function over the variable Δx resulting in the equation

$$H_k \Delta x = -\nabla f(x_k) := -y_k \tag{6}$$

for the search direction Δx. 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 Δx 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 (6) by Cholesky factorization $H_k = LL'$ with lower triangular L, by solving the triangular systems

$$Lv = -y_k, \quad L'\Delta x = v. \tag{7}$$

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 δ. This matrix is now positiv definite, in fact

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

so that (6) now yields a descent direction. Moreover it improves the conditioning of (6): if $H_k + \delta I = L(\delta)L(\delta)'$ is the Cholesky factoriztion 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

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

from which it follows that

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

with obvious implications for the numerical stability of the triangular systems (7). 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 Δx^* of

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

is the minimizer of the quadratic function

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

Now note that this minimizer Δx^* 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 Δx 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 nonsingularity of H_k for us, improves the conditioning and results in a descent direction Δx at iterate x_k.

4 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 remaider 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" Δ 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

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

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

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

or, more explicitly

$$\nabla\overline{f}(u) = F^T \nabla f(x_0 + Fu) \quad \text{and} \quad \nabla^2\overline{f}(u) = F^T \nabla^2 f(x_0 + Fu)F. \tag{8}$$

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

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

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

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

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

$$\nabla(g\circ f)(x) = g'(f(x))\nabla f(x), \quad \text{and} \tag{9}$$

$$H(g\circ f)(x) = g'(f(x))H + g''(f(x))\nabla f(x)\nabla f(x)^T \tag{10}$$

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*

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

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

Example 4.1. 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 (9) and (10) yield

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

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 ϕ_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.

5 Nullspace

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

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

We will represent the nullspace of A as the columnspace $colspace(F)$ 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. $rank(A) = m$. The nullspace $ker(A)$ then has dimension $n - m$ and the columns of F span the entire nullspace of A if and only if $rank(F) = n - m$.

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

$$\text{minimize } f : \mathbb{R}^n \to \mathbb{R} \text{ under the constraint } Ax = b. \tag{11}$$

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

$$\text{minimize } g(u) = f(x_0 + Fu) \text{ on all of } \mathbb{R}^{n-m}. \tag{12}$$

Thereby we reduce the dimension of the problem from n to $n-m$. 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 $n = 3000$ 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 orthornomal basis for the null space $ker(A)$.

5.1 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

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

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

$$A' = QR, \tag{13}$$

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

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

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

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

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

From this we see that $ker(A) = Im(A')^{\perp} = span(q_1, \ldots, q_m)^{\perp}$ but if we use the reduced form we have to compute this orthogonal complement ourselves. In other words we have to extend $\{ q_1, \ldots, q_m \}$ to an orthonormal basis $\{ q_1, \ldots, q_m, q_{m+1}, \ldots q_n \}$ 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 (13) with $m \leq p \leq n$ extend this matrix Q by adding (arbitrarily) orthonormal columns on the right

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

and adjusting the matrix R by adding zero rows at the bottom (so that R becomes $p \times m$. 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 $p = n$, that is, the columns of Q have been extended to a full orthonormal basis of $\mathbb{R}^n$ and so clearly

$$ker(A) = Im(A')^{\perp} = span(q_1, \ldots, q_m)^{\perp} = span(q_{m+1}, \ldots, q_n). \tag{14}$$

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

$$F = [q_{m+1}, \ldots, q_n] \in Mat_{n \times (n-m)}(\mathbb{R}). \tag{15}$$

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:m]`. 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 $m \times m$-matrix, that is, without the $n - m$ bottom zero rows. With this the decomposition of A' becomes $A' = Q[\cdot, 1:m] * R$ and so $A = R'P'$ where the matrix

$$P = Q[\cdot, 1:m] = [col_1(Q), \ldots, col_m(Q)]$$

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 $x_0 = Py \in span(col_1(Q), \ldots, col_m(Q))$ while $Fu \in span(col_{m+1}(Q), \ldots, col_n(Q))$ and so $Fu \perp x_0$, for all $u \in \mathbb{R}^{n-m}$, from which the claim follows.

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

5.2 Nullspace via SVD decomposition of A

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

$$A = U\Sigma V'$$

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

$$\sigma_1 \geq \sigma_2 \cdots \geq \sigma_m \geq 0$$

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

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

In other words

$$ker(A) = span(v_{m+1}, \ldots, v_n). \tag{16}$$

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

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

we must compute the SVD in `R` via

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

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

$$V'x = (c_1/\sigma_1, \ldots, c_m/\sigma_m)',$$

A particular solution of this is given by

$$x_0 = Vw, \quad \text{where} \quad w = (c_1/\sigma_1, \ldots, c_m/\sigma_m)' \in \mathbb{R}^m. \tag{17}$$

since the columns of V are orthonormal.