Newton-type Methods

Walter Murray
Department of Management Science and Engineering,
Stanford University, Stanford, CA

July 5, 2010

Abstract

Newton's method is one of the most famous numerical methods. Its origins, as the name suggests, lies in part with Newton, but the form familiar to us today is due to Simpson of "Simpson's Rule" fame. Originally proposed to find the roots of polynomials it was Simpson who proposed using it for general nonlinear equations and for its use in optimization by finding a root of the gradient. Here we discuss Newton's method for both of these problems with the focus on the somewhat harder optimization problem. A disadvantage of Newton's method, in its original form, is that it has only local convergence unless the class of functions is severely restricted. Much of the discussion here will be about the many ways Newton's method may be modified to achieve global convergence. A key aim of all these methods is that once the iterates become sufficiently close to a solution the method takes Newton steps.

Keywords: nonlinear equations, optimization methods, modified Newton.

1 Introduction

As noted Newton's method is famous. Type it into the search in Youtube and you will get 144 videos with one having had over 30,000 hits. Despite its historic origins it would not be so well known today were it not for its widespread utility. It lies at the heart not just of methods to find roots, but also many other methods such as those for solving partial differential equations. Even Karmakar's geometric based algorithm [9] for linear programming that had such acclaim when it was discovered was subsequently shown in [8] to be equivalent to Newton's method applied to a sequence of barrier functions. Newton's method also serves as a model for other algorithms. It has a theoretical purpose enabling rates of convergence to be determined easily by showing that the algorithm of interest behaves asymptotically similarly to Newton's method. Naturally a lot has been written about the method and a classic book well worth reading is that by Ortega and Rheinboldt [11]. Other books that cover the material here and much more are [7], [2], and [10].

There would not be so much to read were it not for the fact that Newton's method is only locally convergent. Moreover, as in most cases of algorithms that are only locally convergent it is usually not known apriori whether an initial estimate

is indeed close enough. Ideally algorithms should be globally convergent (converge from an arbitrary initial point). However, it should be remembered that proofs of global convergence do need assumptions about the character of the problem that may not be verifiable for specific problems.

There are many approaches to incorporating Newton's method into a more complex algorithm to ensure global convergence and that is the issue we focus on here. It is somewhat more difficult to deal with the case for Newton's method to minimize a function rather than solving a system of nonlinear equations, so it is that class of problems that we give our main attention. Note that solving $f(x) = 0$ can be transformed into the minimization problem $\min_x \frac{1}{2} f(x)^T f(x)$, but minimizing a function cannot be transformed into solving a system of equations. Simply finding a stationary point by equating the gradient to zero does not do the job. Consequently, Newton's method for optimization, in addition to the deficiencies faced when solving systems of equations, needs to be augmented to enable iterates to move off saddle points. This is the key augmentation that is needed for minimization problems.

Note that there may not be a solution to $f(x) = 0$. Any algorithm will fail in some sense on such problems. It is useful to know how it fails and, should it converge, what are the properties of the point of convergence. Identifying a solution of a minimization is more problematic. We may converge to a point for which we can neither confirm the point is or is not a minimizer. That is a another instance of why minimization is a more difficult problem.

Much is made, and rightly so, of Newton's method having a rapid rate of convergence. However, while little can be proved, observations show that when Newton method converges it usually behaves well away from the solution. It is the fact that it is a good local model of the behavior of a function that its popularity and success can be attributed. It is also the reason why alternative methods endeavor to emulate Newton and why it is worthwhile trying to preserve the elements of Newton's method when modifying it to be robust.

We start by addressing the problem in one variable. While it does not highlight all the issues it does illustrate the critical issue of local convergence being the best that can be achieved. In some of the text we drop the indices that denote the iteration when doing so does not cause confusion.

2 Functions of one variable

Newton's method for finding the root of a function of one variable is very simple to appreciate. Given some point, say, x_k, we may estimate the root of a function, say $f(x)$, by constructing the tangent to the curve of $f(x)$ at x_k and noting where that linear function is zero. Clearly for Newton's method to be defined we need $f(x)$ to be differentiable, otherwise the tangent may not exist. Figure 1 shows a single step of Newton's method. Figure 2 illustrates that Newton's method may not give an improved estimate.

Algebraically the method is that of approximating the nonlinear function at the current iterate by a linear one and using the location of the zero of the linear approximation as the next iterate. Consider

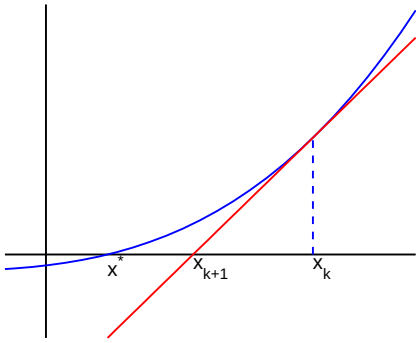

Figure 1: The tangent to $f(x)$ at x_k

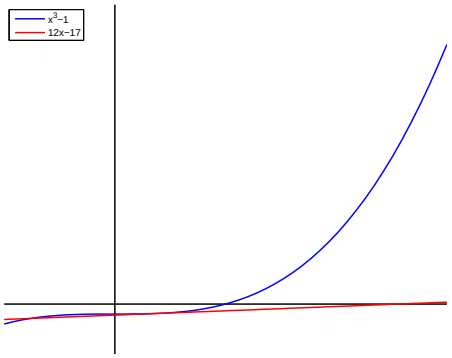

Figure 2: When f'_k is small x_{k+1} may be poor.

$$f(x) \approx ax + b,$$

where a and b are scalars. Since the two function values and their gradients are equal at x_k we have $a = f'_k \equiv f'(x_k)$ and $b = f_k - f'_k x_k$ giving

$$f(x) \approx f'_k x + f_k - f'_k x_k.$$

It follows that the zero of the linear function, say x_{k+1}, is given by

$$x_{k+1} = x_k - f_k/f'_k. \tag{1}$$

We can now repeat the process replacing x_k by x_{k+1}, which is Newton's method for finding the zero or root of a function of one variable. Note that the iteration given by (1) is only defined if $f'_k \neq 0$ and it is this issue that is the flaw in Newton's method. It is not only when $f'_k = 0$ that is a cause of concern. If $|f'_k|$ is small compared to $|f_k|$ then the step may be so large as to make the new iterate a much worse approximation. It is not hard to construct a sequence of iterates that cycle.

Theorem 1.1 (Local convergence of Newton's method). *Let $f(x)$ be a univariate continuously differentiable real-valued function on an open convex set $D \subseteq \mathbb{R}^1$. Assume that $f(x^*) = 0$ for some $x^* \subset D$ and that $f'(x^*) \neq 0$. Then there exists an open interval $S \subset D$ containing x^* such that, for any x_0 in S, the Newton iterates (1) are well defined, remain in S and converge to x^*.*

Proof. Let α be a fixed constant in $(0, 1)$. Since f' is continuous at x^* and $f'(x^*)$ is non-zero, there is an open interval $S = (x^* - \epsilon, x^* + \epsilon)$ and a positive constant μ such that

$$\frac{1}{|f'(x)|} \leq \mu \quad \text{and} \quad |f'(y) - f'(x)| \leq \alpha/\mu \tag{2}$$

for every x and y in S. Suppose that $x_k \in S$. Since $f(x^*) = 0$, it follows from (1) that

$$x^* - x_{k+1} = (f_k - f(x^*) - f'_k(x_k - x^*))/f'_k.$$

Hence the first bound in (2) implies

$$|x_{k+1} - x^*| \le \mu |f_k - f(x^*) - f_k'(x_k - x^*)|.$$

From the integral mean-value theorem we have

$$f(x_k) - f(x^*) - f_k'(x_k - x^*) = \int_0^1 (f'(x^* + \xi(x_k - x^*)) - f_k'\big(x_k - x^*\big))\, d\xi.$$

Hence,

$$|x_{k+1} - x^*| \le \mu \left\{ \max_{0 \le \xi \le 1} \big| f'\big(x^* + \xi(x_k - x^*)\big) - f_k' \big) \big| \right\} |x_k - x^*|. \tag{3}$$

Thus, the second bound in (2) gives

$$|x_{k+1} - x^*| \le \alpha |x_k - x^*|$$

provided $x_k \in S$. Since $\alpha < 1$ and $x_0 \in S$, this last inequality implies that $x_k \in S$ for $k = 1, 2, \ldots$, and that $\{x_k\}$ converges to x^*. ∎

Theorem 1.2. (Rate of convergence of Newton's method). *If the conditions of Theorem 1.1 hold, then the sequence $\{x_k\}$ generated by Newton's method converges q-superlinearly to x^*. Moreover, if $f'(x)$ is Lipschitz continuous in S, where S is defined by Theorem 1.1, with*

$$|f'(x) - f'(x^*)| \le \gamma |x - x^*|, \qquad x \in S, \tag{4}$$

for some constant $\gamma > 0$, then the sequence converges q-quadratically to x^*.

Proof. The linear convergence of the sequence $\{x_k\}$ to x^* is established in Theorem 1.1. Define

$$\beta_k = \mu \left\{ \max_{0 \le \xi \le 1} |f'\big(x^* + \xi(x_k - x^*)\big) - f_k'| \right\}, \tag{5}$$

and assume that $x_0 \in S$, with μ defined as in Theorem 1.1. The continuity of f' and the convergence of $\{x_k\}$ to x^* imply that β_k converges to zero. Since (3) can be written as

$$|x_{k+1} - x^*| \le \beta_k |x_k - x^*|,$$

the definition of q-superlinear convergence to x^* is satisfied by $\{x_k\}$. Let ξ^* denote the value of ξ for which the maximum in (5) is achieved, and let y_k^* denote $x^* + \xi_k^*(x_k - x^*)$. Then

$$|f'(y_k^*) - f_k'| \le \big|f'(y_k^*) - f'(x^*)\big| + \big|f'(x^*) - f_k'\big|. \tag{6}$$

Applying (4) in (6) and (5) gives

$$\beta_k \le 2\mu\gamma |x_k - x^*|.$$

Hence, $\{x_k\}$ satisfies the definition that the rate of convergence to x^* is at least quadratic. ∎

It should be stressed that Newton's method can converge even if $f'(x^*) = 0$. For example, suppose $f(x) = x^3$. Clearly $x^* = 0$. The Newton iteration is:

$$x_{k+1} = x_k - x_k^3/(3x_k^2) = 2x_k/3,$$

which obviously converges to 0, but at a linear rate. We can see from Figure 2 that simply shifting the function to be $x^3 - 1$ does not result in such a smooth set of iterates, although in that case when it converges it does so at a quadratic rate. What this example also illustrates is that the interval of fast convergence may well be small when $f'(x^*)$ is small. In the example above x^* is a point of inflection. It could well be that x^* is a local minimizer or a local maximizer of $f(x)$ in which case there exists a neighboring function that does not have a zero in the neighborhood of x^*. Typically when problems are solved numerically the best that can be achieved is to solve exactly a neighboring problem. Even if the iterates converge numerically to a point for which $|f(x)|$ is small there will remain some uncertainty as to whether a zero exists when $f'(x^*)$ is also very small. In the case where the function changes sign at x^* this may be checked. More importantly knowing an interval for which the sign of the function at the end points are different provides a means of modifying Newton's method to enable it to find a zero. Specifically when the Newton step lies outside the interval it needs to be replaced by a point in the interval. A more sophisticated approach is to replace the Newton step even when it lies in the interval. Since we are taking the Newton step from the better to the two end points we "expect" the next iterate to be close to that point.

2.1 A hybrid algorithm

If an interval is known, say $[a, b]$, at which sign$(f(a))$ = -sign$(f(b))$ then a zero may be found by the method of bisection. The function is evaluated at the midpoint of the interval and the end point of the same sign then discarded to give a new interval half the previous size. The bisection point is optimal in the sense it gives the best worse case performance. This approach may be combined with Newton's method by discarding the Newton iterate whenever it lies outside of the interval in which it is known a root lies. Some care needs to be exercised to ensure the algorithm converges and to ensure that the convergence is not very slow. For example, we need to reject the Newton iterate even if it lies within the interval if it is very close to an end point. When performing bisection alone the optimal placement of the new iterate is the midpoint. When combining the method with Newton we are assuming that eventually Newton's method will make bisection steps unnecessary. If the bisection step is such that the *best* iterate remains the same then at the next iteration we will perform yet one more bisection step. Consequently, we may wish to choose instead a point that looks more likely to replace the best iterate my making it closer to the that iterate than the bisection point.

2.2 Modifying Newton's method to ensure global convergence

In one variable the hybrid algorithm described above is likely the best way to go. However, interval reduction does not generalize to n dimensions so it is worthwhile to discuss other means of making the method more robust. A necessary feature we need to ensure is the existence of the iterates. One way of achieving that is to replace the Newton iteration whenever $|f_k'| \leq \delta$, where $\delta > 0$. For example,

$$x_{k+1} = x_k - f_k/\beta, \tag{7}$$

where $\beta = \text{sign}(f_k')\delta$ if $|f_k'| \leq \delta$, otherwise $\beta = f_k'$. This modification alone does not ensure convergence. What is needed is something that ensures the iterates are improving. One means of defining improvement is a reduction in $f(x)^2$. We could have chosen $|f(x)|$, but $f(x)^2$ has the advantage of being differentiable. The basic idea is to define the iterate as

$$x_{k+1} = x_k - \alpha_k f_k/\beta, \tag{8}$$

where α_k is chosen such that $f_{k+1}^2 < f_k^2$. We need something slightly stronger to prove convergence, but it is enough to choose $\alpha_k = \frac{1}{2}^j$, where j is the smallest index ≥ 1 such that $f(x_k + \frac{1}{2}^j f_k/\beta)^2 < f_k^2$. Determining α_k is a common procedure in $n-$dimensional problems and more elaborate and more efficient methods are known than the simple backtracking just described. However, they also require more elaborate termination conditions, which we discuss later.

2.3 Minimizing a function of one variable

Consider now the problem of

$$\min_x f(x).$$

As noted earlier we could apply Newton's method to $g(x)$, where $g(x) = f'(x)$. The Newton iteration is then given by

$$x_{k+1} = x_k - g_k/g_k'. \tag{9}$$

Clearly, we can only expect this iteration to converge to x^*, where $g(x^*) = 0$. However, if $g'(x^*) < 0$ we would know that x^* is a maximizer and not a minimizer. Another interpretation is that we are approximating $f(x)$ by a quadratic function, for example

$$f(x) \approx \tfrac{1}{2}ax^2 + bx + c.$$

The parameters a, b and c may be determined by the three pieces of information f_k, g_k and g_k'. The next iterate is defined to be the minimizer of this quadratic function. However, when $a = g_k' < 0$, the quadratic function does not have a minimizer. Even when $a = 0$, unless $b = 0$, this function does not have a minimizer. An essential first step then is to define the Newton iteration for minimizing a function as a modification to (9), namely

$$x_{k+1} = x_k - g_k/|g_k'|. \tag{10}$$

In essence we reverse the step when g'_k is negative and go in the direction away from the maximizer. To amend the algorithm when $|g'_k| = 0$ or is very small we can make the same changes as in (7) except $\beta = \delta$ if $|g'_k| \leq \delta$, otherwise $\beta = |g'_k|$. Also α_k is chosen to reduce $f(x)$ rather than $g(x)^T g(x)$. However, there is one more case to consider and that is if $g_k = 0$. In this case, if $g'_k = 0$, we may not be able to make progress. If $g'_k < 0$ then we are at a maximum and a tiny step in either direction from x_k will produce a new iterate x_{k+1} such that $f_{k+1} < f_k$.

As in the case of finding a zero we could also combine Newton with an interval reduction algorithm such as bisection. We know a minimizer exists in an interval $[a, b]$ if $g(a) > 0$ and $g(b) < 0$. Consequently, we can always maintain an interval containing a minimizer by discarding the appropriate end point. There is a minor complication if $g(x) = 0$ at the new point. Also if there is more than one minimizer in the interval there may be a choice.

3 Functions with n variables

The extension to n variables is straightforward. The Newton iteration for $f(x) = 0$ becomes

$$x_{k+1} = x_k - J_k^{-1} f_k, \tag{11}$$

where $J_k \equiv J(x_k)$, and $J(x)$ is the Jacobian matrix of $f(x)$. Likewise, the iteration for $\min_x f(x)$ is

$$x_{k+1} = x_k - H_k^{-1} g_k, \tag{12}$$

where $H_k \equiv H(x_k)$, and $H(x)$ is the Hessian matrix of $f(x)$. Obviously the iterations are only defined if the relevant inverses exist. The conditions for convergence and for the rate of convergence to be at least quadratic in the n-dimensional case is similar to that for the one-dimensional case. Rather then an interval we now require $J(x)$ (and $H(x)$) to be nonsingular in a ball about x^*. Despite being in n-dimensions the Taylor expansions used in the proofs are still in one dimension since we are relating x_{k+1} to x_k along the vector connecting them.

Rather than define the iterations as in (11) and (12) it is more typical to define them as

$$x_{k+1} = x_k + p_k,$$

where p_k is thought of as the "Newton step". When modified as

$$x_{k+1} = x_k + \alpha_k p_k,$$

then the Newton step p_k is viewed as a search direction and the scalar $\alpha_k > 0$ is the steplength. We are considering the Newton step as a trial to be rejected if it proves worse (in some sense) than the current approximation.

Rather than using inverses we define p_k as the solution of a system of linear equation, which for solving $f(x) = 0$ becomes

$$J_k p_k = -f_k,$$

and for minimizing a function

$$H_k p_k = -g_k.$$

In the case of minimizing, the iteration only makes sense when H_k is positive definite. Unlike spotting whether g_k' is positive, determining whether a matrix is positive definite is nontrivial. Since H_k is symmetric if it was know to be positive definite we would usually determine p_k using Cholesky's algorithm. The algorithm can fail when H_k is indefinite and it is that failure that is the simplest way of determining whether or not H_k is positive definite.

4 Linesearch methods

Cast in the manner above Newton's method may be viewed to be in the class of linesearch methods for minimizing a function (see [7]). From now on we shall treat the problem of $f(x) = 0$ as $\min_x \frac{1}{2} f(x)^T f(x)$. A key requirement for linesearch methods is that p_k is a direction of *sufficient descent*, which is defined as

$$\frac{g_k^T p_k}{\|g_k\| \|p_k\|} \leq -\epsilon, \tag{13}$$

where $\epsilon > 0$, and $\|a\|^2 \equiv a^T a$. This condition bounds the elements of the sequence $\{p_k\}$ from being arbitrarily close to orthogonality to the gradient. Typically methods are such that p_k is defined in a manner that satisfies (13) even though an explicit value of ϵ is not known.

The other key requirement is that the steplength α_k is chosen not to be too large or too small. This may be achieved by a suitable termination condition in the search for α_k. The termination criteria used in the optimization routine SNOPT (see [6]) is:

$$|g(x_k + \alpha p_k)^T p_k| \leq -\eta g_k^T p_k, \tag{14}$$

where $0 < \eta \leq 1$. Typically η is chosen in the range $[.1, .9]$. This termination criterion ensures the step α_k is not too small. Usually it will also ensure it is not too large. However, to be absolutely sure we need an additional condition and

$$f(x_k + \alpha p_k) \leq f_k + \mu \alpha g(x_k + \alpha p_k)^T p_k, \tag{15}$$

where $0 < \mu < \frac{1}{2}$ will suffice. Typically μ is chosen quite small and as a consequence this condition is almost always satisfied once (14) is satisfied. To find a step that satisfies (14) one can search for a minimizer along p_k and terminate at the first point (14) is satisfied. Since we are generating iterates that decrease $f(x)$ it follows that (15) is satisfied with μ replaced by $\varepsilon_k > 0$.

It can be shown that under modest assumptions linesearch methods as described generate a sequence such that $\lim_{k\to} g_k = 0$. For nonlinear equations, since $g_k = J_k^T f_k$ it implies $f_k \to 0$ provided J^* is nonsingular. In the minimization case we are only sure we are at a minimizer only if $\lim H_k = H^*$ and H^* is positive definite. When H^* is positive semidefinite with at least one zero eigenvalue and when higher derivatives of $f(x)$ are unknown there is usually nothing further that can be done

to confirm the iterates have converged to a minimizer. Indeed it is not known just how many orders of higher derivatives will be needed to resolve the issue.

A more general issue is what do in any iteration when either J_k is singular or H_k is indefinite since under such circumstances the Newton step is not necessarily a sufficient descent direction. If p_k is defined to be the solution of a system

$$B_k p_k = -g_k,$$

where $\{B_k\}$ is a sequence of bounded positive definite symmetric matrices whose condition number is also bounded (smallest eigenvalue bounded away from 0) then $\{p_k\}$ is a sequence of sufficient descent directions. It is easy to see why. Let A be a symmetric matrix with eigenvalues

$$0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

Let p and g denote vectors in $\mathbb{R}^n$ such that $Ap = -g$. From the definition of the smallest eigenvalue λ_1 we have

$$u^T A u \geq \lambda_1 \|u\|^2.$$

From and the relationship $Ap = -g$ and the above inequality we have

$$-g^T p/(\|p\|\|g\|) = p^T A p/(\|p\|\|g\|) \geq \lambda_1 \|p\|/\|g\|.$$

Moreover, from norm inequalities we may assert that

$$\|g\| = \|Ap\| \leq \|A\|\|p\| = \lambda_n \|p\|.$$

Using the bound gives

$$-g^T p/(\|p\|\|g\|) \geq \lambda_1/\lambda_n.$$

Applying this result to each matrix in the sequence $\{B_k\}$ gives have

$$\cos\theta_k = -g_k^T p_k/(\|g_k\|\|p_k\|) \geq \lambda_1^{(k)}/\lambda_n^{(k)} \geq 1/M.$$

Consequently $\{p_k\}$ is a sequence of sufficient descent directions as required. The main idea on how to modify the Newton system defining p_k is to ensure that it is the solution of a system that has the same properties as B_k.

Consider first the nonlinear equation case. When J_k is nonsingular the search direction p_k is the solution of

$$J_k^T J_k p_k = -J_k^T f_k.$$

Obviously if J_k is singular then so is $J_k^T J_k$. However, we may modify the symmetric system above by adding a multiple of the identity giving

$$(J_k^T J_k + \lambda_k I) p_k = -J_k^T f_k.$$

The above system will be nonsingular provided $\lambda_k > 0$. Hence p_k is always well defined and, provided λ_k is chosen appropriately, is a sufficient descent direction for

$\frac{1}{2}f(x)^T f(x)$. There are details that need explaining such as how best to solve this system and how to determine a good value for λ_k, but the issue of convergence is resolved. When $J(x^*)$ is nonsingular then there exists an index, say K, such that for $k \geq K$ we may set $\lambda_k = 0$. By doing so the terminal search directions are pure Newton's steps.

We have left one detail out and that is the assumption that $\alpha_k = 1$ for $k \geq K$. Since we do not in general know K, when we compute the pure Newton direction in the neighborhood of the solution, we need the linesearch procedure to automatically take the unit step. It is easy to show this will happen provided the initial step taken is 1. Setting $\alpha = 1$ in the LHS of (14) and noting there exists a constant $M < \infty$ such that $\|g(x_k + p_k)\| \leq M\|g_k\|^2$ when $J(x^*)$ is nonsingular, we get

$$|g(x_k + p_k)^T p_k| \leq \|g(x_k + p_k)\|\|p_k\| \leq M\|g_k\|^2\|p_k\|. \tag{16}$$

Since p_k is a sufficient descent direction then the RHS of (14) satisfies

$$-\eta g_k^T p_k \geq \eta\epsilon\|g_k\|\|p_k\|. \tag{17}$$

Since $\lim \|g_k\| \to 0$ it follows there exists K for which the unit step satisfies (14) for $k \geq K$. Note that for the problem $f(x) = 0$ we have $g_k = J_k^T f_k$.

By a similar argument we can show that (15) is also satisfied, which gives us the desired result.

4.1 Obtaining a direction of sufficient descent from the Newton equations

It may be thought that a simple way would be to solve the Newton equations and then reverse the direction if $g^T p > 0$. While this works in the one variable case it is flawed in the n-dimensional case. It is not hard to see why although doubtless this approach is still used. Suppose we have two variables and $f(x_1, x_2)$ is a separable function, that is $f(x) = f_1(x_1) + f_2(x_2)$. It follows that the Hessian is a diagonal matrix. Suppose $H_{1,1} > 0$ and $H_{2,2} < 0$ and the Newton step is p. In the x_1 variable the step p_1 is correctly stepping towards the minimizer, while the step p_2 is stepping away from a minimizer. It could happen that $-p_1 g_1 < p_2 g_2$ in which case p is not a descent direction. What we would do if we knew this was a separable function is reverse the sign of p_2 only. More generally what we wish to do is to preserve the Newton step in the subspace for which the H is positive definite and reverse it in the subspace it is not. Quite how to do that we now explain. Note that reversing the sign of p does not give an assured descent direction since $p^T g$ could be zero even when g is not zero.

Consider the spectral decomposition: $H = V\Lambda V^T$, where V is the matrix of eigenvectors, Λ =Diag(λ), and λ is the vector of eigenvalues. Define

$$\bar{H} \equiv V\bar{\Lambda}V^T,$$

where $\bar{\lambda}_i = |\lambda_i|$, if $|\lambda_i| \geq \delta$, otherwise $\bar{\lambda}_i = \delta \geq 0$, and $\bar{\Lambda}$ =Diag($\bar{\lambda}$). A means of achieving our objective is to define p as

$$p \equiv -\bar{H}^{-1} g.$$

Provided the minimum eigenvalue of H^* is bigger than δ then Newton steps will be taken in the neighborhood of the solution. It may be thought that δ could be allowed to converge to zero as $g_k \to 0$, which would assure Newton's steps will always be taken in the neighborhood of the solution provided H^* is positive definite. For example, we could define $\delta \equiv \min\{|g_k|, \epsilon\}$, where $\epsilon \geq 0$. However, in practice p is computed numerically and the error in the computed solution will overwhelm the Newton step when H^* is ill conditioned. Indeed the error analysis establishing that Cholesky is a stable algorithm assumes the matrix is sufficiently positive definite.

The drawback of the scheme above is that when n is large it is expensive both in computational effort and memory to compute the spectral decomposition. When H is sufficiently positive definite it is also unnecessary. Since Cholesky is so cheap it is always worth trying first. An alternative is to compute a direction based on modifying the Cholesky factorization. Cholesky fails on indefinite matrices due to the need to compute a square root of a negative number when computing the diagonal elements of the factor. An obvious change is to define the diagonal elements in the same vein as the modification made to the eigenvalues. If no zero pivots occur, the factorization LDL^T exists, where L is a unit lower triangular matrix and D is diagonal. We can now define $\bar{D}$ in relationship to D in the identical manner to the relationship of $\bar{\Lambda}$ to Λ. It can be shown that $\bar{H} \equiv L\bar{D}L^T = H + E$, where E is also diagonal. In practice the occurrence of a zero diagonal is of no consequence since should one arise it is replaced by δ. Unfortunately, it can be shown that the bound on $\|E\|$ is $O(1/\delta)$ (see [7]).

In order to obtain a Cholesky factor that is not close to being unbounded it is necessary to perform the column form of the algorithm. The basic idea is not simply to alter pivots that are tiny or negative but to alter any pivot for which it is necessary to ensure that the magnitude of the elements of the Cholesky factor are *suitably* bounded. If on first computing an element it exceeds a prespecified bound, say β, then it is recomputed using an increase to the pivot that makes it match the bound. Care needs to be exercised in defining β to ensure that matrices that are sufficiently positive definite are not modified. Let $\bar{R}$ denote the Cholesky factor generated by this algorithm, then β is chosen such that $|\bar{R}_{i,j}| \leq \beta$, where

$$\beta^2 = \max\{\gamma, \xi/n, \delta\},$$

γ and ξ are the largest in modulus of the diagonal and off-diagonal elements of H respectively, and as before δ is some small positive scalar. It can be shown (see [7]) that $\bar{H} \equiv H + E$, E is a diagonal matrix, $\|E\|$ is bounded and $\bar{H}$ has a bounded condition number. The algorithm requires almost the same computational effort as the regular Cholesky algorithm. Let $\bar{R}_k$ denote the Cholesky factor of $\bar{H}_k$. It follows a suitable sequence of sufficient descent directions, $\{p_k\}$, are obtained by solving:

$$\bar{R}_k^T \bar{R}_k p_k = -g_k.$$

When H^* is sufficiently positive definite then in the neighborhood of the solution $H_k = \bar{H}_k$ and Newton steps are taken.

Generally using a modified direction as defined avoids saddle points. We are deliberately defining the direction not to be a step to such points. However, we

could start at one or end up at one by chance. There are also problems (see [4]) that have a form of symmetry that results in the gradient being in the space of the eigenvectors corresponding to the positive eigenvalues of H. It is a consequence of variables whose values, when they are switched, give the same objective. For such problems the only way of breaking the tie is to use a direction of negative curvature.

4.2 Directions of negative curvature

If $g = 0$ then there is no direction of descent and if H is indefinite then x is a saddle point. The key to moving off a saddle point (or avoiding converging to one) is to use directions of negative curvature. A vector d is termed a direction of negative curvature if $d^THd < 0$. Obviously the Hessian needs to be indefinite for such a direction to exist. It may be thought that it is relatively easy to find such a direction and in some cases it is. If H has half its eigenvalues that are negative and equal in value to the positive set then a random direction has a 50% chance of being a direction of negative curvature (DNC). However, it can be shown (see [3]) that the probability is extremely small when the number of positive eigenvalues dominate (approximately 10^{-14} when there are 99 eigenvalues of 1 and one of -1). Typically in optimization problems there are only a few negative eigenvalues (the iterates are converging to a point the Hessian is positive semidefinite) and even these are small. A single large negative eigenvalue would signify that there is a DNC along which the objective will decrease rapidly and we are rarely that lucky.

Unlike directions of descent there is a"best" DNC, namely, the eigenvector corresponding to the smallest eigenvalue. If the smallest eigenvalue is not unique then any vector spanned by the set of corresponding eigenvalues is equally good. It is best in the sense that this direction has the smallest value of d^THd/d^Td and it is independent of a linear transformation of variables. A DNC predicts a potential infinite reduction in the objective. Of course in practice the step will be finite but there is no good initial estimate of what the finite step should be. This contrasts with a direction of descent which is typically derived from a positive definite quadratic model for which the unit step is the predicted to be optimal.

In order to show the iterates will converge to a point for which the Hessian is at least positive semidefinite it is necessary for the DNC to be a direction of sufficient negative curvature. A sufficient condition for $\{d_k\}$ to be a sequence of directions of sufficient negative curvature is that

$$d_k^T H_k d_k \leq \theta \lambda_{min_k} d_k^T d_k,$$

where λ_{min_k} is the smallest eigenvalue of H_k, and $\theta > 0$. Obviously $\theta \leq 1$. What the requirement does is to prevent d_k from becoming arbitrary close to being a direction of zero curvature.

If d_k is a DNC then so is $-d_k$. The sign is chosen so that $d_k^T g_k \leq 0$. If $H(x^*)$ is positive definite then there exists a index K such that for $k \geq K$ then $d_k = 0$.

A DNC may be computed from the modified Cholesky factorization. It can be shown an index, j, exists (it can be identified during the factorization) such that d, where

$$\bar{R}d = e_j, \tag{18}$$

is a DNC. However, it is not a direction of sufficient negative curvature. Fortunately, this is of little consequence. Rather than applying the modified Cholesky algorithm to H it is better to apply it to $H + \delta I$, where $\delta > 0$ is suitably small. This modification serves a number of purposes. If this matrix is indefinite then (18) does give a sufficient DNC. When H^* is singular and positive semidefinite it avoids tripping the modifications, and from a getting a DNC that is not needed in the neighborhood of x^*. Given a DNC it can be improved by using it as a starting point to minimize $d^T H d / d^T d$. For example, by just using a single pass though all the variables of a univariate search (see [3]).

An issue is how best to use a DNC. An obvious choice is to define the search direction as:

$$\bar{p} = p + d,$$

where p is a direction of sufficient descent. It is prudent to have scaled d. There are various schemes that can be used but the general principle is that if $d^T H d / d^T d$ is small then $\|d\|$ should be small.

It can be shown under suitable assumptions that this algorithm converges to a point satisfying the second-order necessary conditions for optimality.

5 Trust-region methods

An alternative to modifying the Hessian is to minimize the quadratic approximation subject to a limit on the size of the step. This is similar to using the interval to limit the step in problems with one variable. However, unlike the one variable case the solution does not usually lie within the limit. If at this new point the objective is not an improvement then the limit is reduced. We wish to solve:

$$\min_s \{\psi(s) \equiv \tfrac{1}{2} s^T H s + g^T s | s^T s \leq \Delta\}, \tag{19}$$

where $\Delta > 0$. Clearly a minimizer exists. It can be shown (see [10]) that there exists a scalar $\lambda \geq 0$ such that the global minimizer, s^*, satisfies

$$(H + \lambda I) s^* = -g, \tag{20}$$

where $H + \lambda I$ is positive semidefinite and $\lambda(s^T s - \Delta) = 0$. It is rare for $H + \lambda I$ to be singular. It is immediately apparent that the definition of s^* is similar to that of p, since the matrix is symmetric, at least positive semidefinite and is a diagonal modification of H. If H is positive definite and the solution of $Hs = -g$ is such that $s^T s \leq \Delta$ then s^* is the Newton step. When $f(x + s^*) \geq f(x)$ the step is rejected and the system (20) resolved with a smaller value of Δ, typically $\Delta \to \Delta/2$. It can be shown only a finite number of reductions need be taken. If a satisfactory step is obtained without the need to reduce Δ and $\psi(s^*)$ is a reasonable prediction of $f(x) - f(x + s^*)$ then Δ at the next iteration is increased, typically $\Delta \to 2\Delta$. It can be shown under typical assumptions that the sequence generated by the trust-region algorithm converges to a point satisfying second-order necessary conditions for optimality. It can be shown also that when H^* is positive definite that Δ does

not converge to zero. Consequently, in the neighborhood of the solution Newton steps are taken.

In order to solve (20) it necessary to determine λ. To do so requires solving (20)with estimates of λ. Also if Δ is unsatisfactory the process has to be repeated, which requires more solves of (20). It is not necessary to determine λ accurately and often Δ is satisfactory, Nonetheless it is clear a trust-region algorithm requires more solves of similar linear systems than a linesearch method.

Note that when $g = 0$ and H is indefinite then the solution of (20) is the eigenvector corresponding to the minimum eigenvalue (assuming it is unique).

6 Gradient-flow methods

Unlike the other two classes of methods these are not in widespread use. The methods were originally proposed by Courant to solve PDEs. Their simplest form is to compute the arc $x(t)$, where $x(t)$ satisfies

$$x'(t) = -g(x(t)) \quad \text{and} \quad x(0) = x_0. \tag{21}$$

The minimization problem has been replaced by the problem of solving a system of nonlinear differential equations. This approach is grossly inefficient. The method is made reasonable when $g(x(t))$ is approximated at x_k by $g_k + H_k(x(t) - x_k)$. The idea now is compute x_{x+1} by searching along the arc $p(t)$, where $p(t)$ is the solution of a linear system of differential equations:

$$p'(t) = -g_k - H_k p(t) \quad \text{and} \quad p(0) = 0. \tag{22}$$

The connection to Newton's method is now apparent. When H_k is positive definite the arc is finite and the end point is the Newton step from x_k. If $H(x^*)$ is positive definite then when the iterates are in a small neighborhood of the solution the Newton step is taken. An interesting property of this approach is that the arcs and iterates generated are invariant under an affine transformation, a property that is typically not true for either linesearch or trust-region methods when H_k is not positive definite for all k.

7 Estimating derivatives

For minimization problems Newton-type methods assume the availability of second derivatives. When these derivatives are not available there are alternatives such as quasi-Newton methods. Usually these alternatives are to be preferred although there are problems where using the Newton-type methods described, except with the Hessian approximated by finite differences of the gradients, are better. A key difference with quasi-Newton methods is the Hessian approximation in those methods is usually positive definite, hence directions of negative curvature are not available. All the methods discussed may be applied to the case in which $H(x)$ is approximated by finite differences with little difficulty except for the need to compute the differences.

The performance of such algorithms is almost indistinguishable, in most cases, from the case when an exact Hessian is used. Of course the "exact" derivatives do themselves contain errors since they are computed in finite-precision arithmetic. There are not the same numerical issues in approximating second derivatives as there are to approximating first derivatives. It can be shown that when finite differences are used and the finite-difference interval does not converge to zero, which for numerical purposes it can not, the quadratic rate of convergence is lost. However, this result is misleading. The iterates, except in pathological cases, mimic those of the exact case. Indeed the quadratic convergence of the exact cases also breaks down, but typically at a point the algorithm terminates.

When n is large the number of differences could grow with n. When $H(x)$ (or $J(x)$ in the nonlinear equation case) is a sparse matrix there are clever algorithms that can choose the direction of the differences in a manner that significantly reduces the number of differences required (see, [10]). For example, if $H(x)$ is tridiagonal only two differences are required.

8 Constrained problems

The general nonlinearly constrained problem may be stated as:

$$\min_{x \in \mathbb{R}^n} \; f(x) \;\; \text{such that} \;\; c(x) \geq 0, \tag{23}$$

where $c(x) \in \mathbb{R}^m$ is a vector of smooth functions. It is usual to distinguish linear from nonlinear constraints, and further, to distinguish simple bounds on the variables from linear constraints. Generally equality constraints are easier to handle than inequality and, for simplicity, are often omitted. For problems with only linear constraints the iterates sequence generated is usually feasible and that has great benefits. Constraints complicate the adaptation of Newton's method as does size and not surprising large constrained problems are the most difficult of all. It is beyond the scope of this paper to consider all the issues in detail but we shall cover some of the salient issues.

Consider first a problem whose only constraints are simple equalities. By that we mean some subset of the variables are fixed at some value. Clearly this creates no issues since the problem is unconstrained in the remaining variables. If there are simple inequalities then at each iteration an active-set methods fixes some subset of the variables. Consequently, Newton-type methods using an active-set strategy are relatively easy to adapt to such problems. However, consider a trust-region method whose only constraints are $x \geq 0$. Adding the constraints $x_k + s_k \geq 0$ to the subproblem (20) makes the subproblem intractable. Typically the quadratic constraint is replaced by bounds on the step, which then gives a pure bounds constrained subproblem. However, finding the global minimizer of even this subproblem is intractable. It is still possible to find a step that reduces the objective. What is lost is that the modified algorithm is no longer assured of generating a sequence that converges to a point that satisfies the *second-order* necessary conditions.

As noted, for an active-set linesearch method, there are no complications for problems with simple bounds. There are also no complications with general linear constraints provided the problem is not too large. Issues do arise when the problem is large. These are not unsurmountable but requires some ingenuity. When we have linear constraints that are active we need to find a direction of negative curvature that lies on the constraints or to put it another way a DNC that is orthogonal to the constraint normals. We also need to modify the *reduced* Hessian rather than the Hessian. Suppose at the kth iteration we wish to remain on a set of constraints $A_k = b_k$. The subproblem we need to solve to obtain the Newton step is

$$\min_p \{\tfrac{1}{2} p^T H_k p + g_k^T p \mid A_k p = 0\}.$$

Let Z_k be a $n \times (n - m_k)$ full rank matrix such that $A_k Z_k = 0$, where m_k is the number of rows of A_k. We are assuming A_k is full rank. The solution to this problem is given by

$$Z_k^T H_k Z_k p = -Z^T g_k. \tag{24}$$

However, if $Z_k^T H Z_k$ is not positive definite it needs to be modified in a similar manner to how H_k is modified in the unconstrained case. There is little difficulty in doing this when n is of modest size. There are not a lot of issues if $n - m_k$ is of modest size even if n is large, although forming the product may be expensive. Fortunately, H_k is typically sparse and an implicit form of Z_k may be found that is also sparse. In some cases advantage can be made of structure both in A_k and H_k. However, there are cases where both n and $n - m_k$ are large and $Z_k^T H_k Z_k$ is not sparse. Under such circumstances there are two approaches. One is to consider, instead of using the reduced system (24), solving the equivalent KKT system

$$\begin{pmatrix} H_k & A_k \\ A_k^T & 0 \end{pmatrix} \begin{pmatrix} p_k \\ -\lambda_k \end{pmatrix} = \begin{pmatrix} -g_k \\ 0 \end{pmatrix}, \tag{25}$$

where λ_k is the Lagrange multiplier estimate. This matrix is indefinite regardless of the character of H_k. However, it may be deduced from the inertia of this matrix whether or not the reduced Hessian is positive definite. Factorizing this matrix in the form $PLBL^T P^T$, where P is a permutation matrix, L is a unit lower triangular matrix, and B is a 2×2 block-diagonal matrix reveals the inertia. Unfortunately, if the reduced Hessian is not positive definite this factorization does not reveal how to obtain a suitable sufficient descent direction. How to obtain suitable descent directions by modifying this factorization is described in [5]. They also suggest how to modify the original matrix prior to using an unmodified factorization.

An alternative to factorizations is to attempt to solve (24) using the conjugate gradient (CG) method. Since the CG algorithm requires only matrix-vector products the reduced Hessian need not be formed explicitly. The CG method may fail on indefinite matrices, but it is precisely because it may fail that makes it useful. It can also be shown that when it fails a direction of sufficient descent may be constructed from the partial solution. While it may not give a direction of sufficient negative curvature in general it will usually give a direction of at least zero curvature, which if

need be may improved by using this as an initial vector in minimizing $d^T Z^T H Z d / d^T d$ either by a few iterations of the Lanczos algorithm or a few iterations of univariate search.

When there are nonlinear inequality constraints then adapting Newton's method is considerably more problematic except if the nonlinear constrained problem is transformed into solving a sequence of linearly constrained problems. Assuming that is not the case most methods generate infeasible iterates. As a consequence the iterate is typically composed of two components, one is a Newton step to get feasible and the other to "reduce" (it may be increasing) the objective. If there are only equality constraints it is somewhat simpler so we shall describe that first. We are now concerned not with the Hessian of the objective, but with the Hessian of the Lagrangian, where the Lagrangian $L(x, \lambda)$ is defined as $L(x, \lambda) \equiv f(x) - \lambda^T c(x)$. Assuming there are no issues with the reduced Hessian of the Lagrangian the system of equations we need to solve are:

$$g(x) - J(x)^T \lambda = 0 \quad \text{and} \quad c(x) = 0, \tag{26}$$

which is a system of $n + m$ equations in $n + m$ unknowns. In other words we are now applying Newton's method in both the primal and dual variables. The Newton step in these variables is given by:

$$\begin{pmatrix} H_k^L & J_k \\ J_k^T & 0 \end{pmatrix} \begin{pmatrix} p_k \\ -\Delta\lambda_k \end{pmatrix} = \begin{pmatrix} -g_k + J_k \lambda_k \\ -c_k \end{pmatrix}, \tag{27}$$

where H_k^L is the Hessian of the Lagrangian at x_k, λ_k, and $\Delta\lambda_k$ is the Newton step in the dual variables. The issues in addressing the case when the reduced Hessian is not positive definite in the space of the x-variables is similar to the linearly constrained case. We have no concern about the character of the Hessian in the space of the dual variables. However, since the search is in the space of both variables there needs to be a search direction defined for those variables. How to do that is described in [12]. Note that we are leaving out the fact we are minimizing a merit function so care needs to be taken to ensure directions of negative curvature for the Hessian of the Lagrangian are also directions of negative curvature for the merit function. Again how this may be done is described in [12].

The issue with an active-set method for nonlinear constraints is that the most efficient approach is a sequential quadratic programming (SQP) method. Such methods construct a quadratic objective whose Hessian is an approximation to the Hessian of the Lagrangian. When second derivatives are known of both the original objective and the of the nonlinear constraint functions only the Lagrange multipliers need be estimated. If the QP is not convex the solution of the QP is not necessarily a direction of sufficient descent. Indeed the QP may not have a bounded solution and even if it does, and the global minimizer is found, that may not yield a direction of descent. In solving the QP the iterates move on to different sets of active constraints. It is possible for the reduced Hessian to be positive definite on every subspace and yet the solution is not a descent direction. Simply observing the character of the reduced Hessians may not reveal nonconvexity. Unlike the equality constrained case

we do not know initially the null space of the active set of the constraints since we do not know which constraints are active at the solution of the QP. Moreover, altering the Hessian on that subspace will alter the set of constraints active at the solution. This conundrum may be resolved by monitoring the total step being taken in the QP from the initial feasible point. Assuming the initial reduced Hessian is positive definite (if it is not the the Hessian may be modified by applying the modified Cholesky factorization to the reduced Hessian) there are no problems provided only constraints are added. It is when a constraint is deleted that indefiniteness may arise. While it may not arise in the resulting reduced Hessian the indefiniteness in the direction from the initial point must be checked. While the Hessian still has positive curvature along that direction a descent direction is assured. When it does not the situation is similar to that when applying the CG algorithm and it breaks down, the prior step is a descent direction and the new step yields a direction of negative curvature.

Approximating second derivatives is sometimes cheaper for constrained problems especially in the linearly constrained case. Since we only need to approximate Z^THZ then we can approximate HZ directly by differencing along the column vectors of Z. When $n - m$ is small this is quite efficient. Also if the system (24) is solved using the conjugate gradient algorithm we need only difference (this holds for the unconstrained case also) along the CG directions, which may be only a few before a sufficiently good search direction is obtained.

8.1 A homotopy approach

Rather than change the algorithm another way to extend the range of convergence of Newton's method is to change the problem or rather replace the problem by a sequence of problems. For example, we can use a homotopy approach. The basic idea is to replace the original problem with a parameterized problem. At one end of the parameter range the problem is easy to solve and at the other end it is the original problem. Such an approach may be used for globally convergent algorithms also since it can be more efficient than simply solving the original problem directly. Here our use is not because the problem is necessarily hard to solve but because Newton's method may not work at all. Note that when you do start close to the solution with Newton's method the problem is usually easy to solve since the iterative sequence converges quickly.

Constructing a homotopy for the case of solving $f(x) = 0$ is particularly easy. Instead of solving $f(x) = 0$ we could solve $f(x) = \gamma f(x_0)$, where x_0 is our initial estimate, and $0 \leq \gamma < 1$. Obviously if γ is very close to 1 then x_0 is very close to $x^*(\gamma)$, where $x^*(\gamma)$ is a zero of $f(x) = \gamma f(x_0)$. Since we can make γ as sufficiently close to 1 as we wish we can be within the interval of convergence provided the gradient of $f'(x^*(\gamma)) \neq 0$. It is not necessary to find the solution to the intermediate problems precisely. The main difficulty is assessing how conservative to make the initial choice of γ and how quickly it can be reduced. In one variable this approach makes the Newton step smaller and as such plays a similar role to α_k in (8). However, note that it is not identical to a fixed scaling of the Newton step.

Another way of modifying functions is to make them more like the ideal case for Newton's method. When solving $f(x) = 0$ that can be done by adding a linear function, for example, solve $(1-\gamma)f(x)+\gamma(x-x_0) = 0$. Here we change the gradient of the function as well as move the zero closer to x_0. A similar approach may be taken for the problem $\min_x f(x)$ by adding a quadratic function.

If at the initial estimate $f_0' = 0$ then a different initial estimate is needed regardless of the approach being used to solve the problem.

It can be particularly useful to use a homotopy approach when using an SQP algorithm As noted when the Hessian is indefinite difficulties arise. Adding a term $\frac{1}{2}\rho(x-\bar{x})^T(x-\bar{x})$ to the objective, where $\bar{x}$ is the initial approximation, adds γI to the Hessian. Not only will this reduce the need to modify the Hessian it also almost always reduces the number of iterations required to solve the QP subproblem. If γ was very large then we would be taking a very small optimization step from the initial feasible point of the QP subproblem. It is advisable to change $\bar{x}$ periodically (perhaps even every iteration) and also to reduce γ to zero. When the iterates are close to the solution and the reduced Hessian of the Lagrangian is positive definite then indefiniteness will not arise when solving the QP. Indeed close to the solution only one iteration of the QP is required since the correct active set has been identified.

9 Acknowledgement and comments

There is much more that could be written in particular on extensions of Newton's method to both preserve and improve upon the rate of convergence. it would also be remiss not to cite the classic work of Kantorovich [1]. Thanks are due to Nick Henderson for reading the manuscript and to the referees for their prompt reviews and and insightful comments. Support for this work was provided by the Office of Naval Research (Grant: N00014-02-1-0076) and the Army (Grant: W911NF-07-2-0027-1).

References

[1] Kantorovich, L.V., Functional Analysis and Applied Mathematics, Uspekhi Mathematicheskikh Nauk, Vol. 3, pp. 89-185, 1948.

[2] D. Bertsekas (1999), Nonlinear Programming, *Athena Scientific.*

[3] E. Boman (1999). Infeasibility and Negative Curvature in Optimization. PhD thesis, Stanford University,
http://www.stanford.edu/group/SOL/dissertations.html.

[4] V. Ejov, J.A. Filar, W. Murray, G.T. Nguyen (2009), Determinants and longest cycles of graphs (2009). *SIAM Journal on Discrete Mathematics,* 22 (3), 1215–1225.

[5] A. Forsgren and W. Murray (1993), Newton methods for large-scale linear equality-constrained minimization, *SIAM J. Matrix Anal. Appl.*, 14, 560–587.

[6] P. E. Gill, W. Murray, and M. A. Saunders (2005). SNOPT: An SQP algorithm for large-scale constrained optimization, *SIAM Review* 47, 99–131.

[7] P. E. Gill, W. Murray, and M. H. Wright (1981). Practical Optimization, *Academic Press, London and New York*.

[8] P. E. Gill, W. Murray, and M. A. Saunders , J. A. Tomlin, and M. H. Wright(1986). On projected Newton barrier methods for linear programming and an equivalence to Karmarkars projective method, *Mathematical Programming* 36 (2), 183-209.

[9] N. Karmarkar (1984). A New Polynomial Time Algorithm for Linear Programming, *Combinatorica*, Vol 4, nr. 4, 373-395.

[10] J. Nocedal and S. J. Wright (2006), Numerical Optimization, *Springer*.

[11] J.M. Ortega and W.C.Rheinboldt (2000). Iterative Solutions of Nonlinear Equations in Several Variables, *SIAM Calssics in Applied Mathematics*.

[12] W. Murray and F. Prieto (1995), A second-derivative method for nonlinearly constrained optimization, Report SOL 95–3, 48 pages.