University of Oxford

Computing Laboratory

Prof. Nick Gould

Nonmonotone Line Search and Trust Region Methods for Optimization

Almut Eisenträger

April 21, 2007

1 Introduction

In applications such as sciences, engineering and economics, many problems that occur can be solved by minimizing a function; sometimes under constraints, sometimes without. Most of the time, the minimum of such a function cannot be found analytically. The solution is therefore to use one of the numerous algorithms which have been developed to approximately find a local minimizer numerically.

The two main classes of algorithm are line search methods and trust region methods. In line search methods, the problem is reduced to one dimension in every iteration by choosing a search direction and considering the objective function only along that line, whereas in trust region methods, the objective function is modelled in a small region by a function which is easier to approach, in the hope that by decreasing the model function one also achieves a decrease in the value of the objective function.

Throughout the world mathematicians and computer scientists try to improve existing algorithms or create new ones to achieve better and quicker results. The main criteria that define the quality of an algorithm are global convergence, speed (measured in CPU-time or number of iterations), efficiency with respect to the number of function evaluations (which can be very time consuming), numerical stability (especially when large or badly conditioned problems have to be solved) and the range of problems the algorithm can be applied to.

Global convergence is usually demanded at least for nicely behaving functions, e.g. twice continuously differentiable. To ensure this, most algorithms require monotone decrease of the objective function in every iteration. This can slow down the rate of convergence of an originally fast algorithm radically, for example when applied to a function with narrow curved valleys.

In recent years, there have been several quite successful attempts to avoid this behaviour by relaxing the request for monotony without loosing global convergence. In this paper, we consider some nonmonotone algorithms, in both classes, line search and trust region methods, and investigate their performance.

The organization of this paper is as follows. In section 2, we describe and assess several nonmonotone line search techniques. The same is done for trust region methods in section 3 and a conclusion is provided in section 4.

Notation:
Throughout this paper $g(x) = \nabla f(x)$ denotes the gradient of $f(x)$. The Hessian is denoted as $H(x) = \nabla^2 f(x)$ and at each iterate, $H_k = H(x_k)$ or an approximation to that. We write the scalar product as $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ for the Euclidean norm, but often, any norm

will do. In the line search methods, d_k is the search direction. For trust region methods, s_k denotes the trial step for the trust region radius Δ. Furthermore, Ω_0 is defined as the level set $\{x : f(x) \leq f(x_0)\}$. If applicable, $\Omega \subset \mathbb{R}^n$ denotes the feasible set of a problem and $P(z)$ a suitable projection onto Ω. In the case of equality constrained problems, $c(x)$ denotes the constraint function and $A(x) = \nabla c(x)$ the transposed Jacobian of c. For all values, a subscript k means that this is the evaluation at x_k or the value in the k-th iteration, e.g. $f_k,\ g_k,\ c_k$.

2 Nonmonotone line search methods

At first, we consider the unconstrained optimization problem

$$\min f(x) \quad \text{subject to} \quad x \in \mathbb{R}^n, \tag{1}$$

even though, some of the algorithms have been developed for constrained optimization.

For each iterate x_k, a search direction d_k is chosen, which needs to be a descent direction:

$$\langle d_k, g_k \rangle < 0. \tag{2}$$

The next iterate is then defined as $x_{k+1} = x_k + \alpha_k d_k$, where α_k is chosen to ensure sufficient decrease of the objective function. (One could also find the exact minimizer α_k, but this is generally too expensive and ineffictive, hence has been practically abandoned.)

One possibility among several to ensure sufficient decrease is the Armijo condition (see [1]):

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \alpha_k \gamma \langle d_k, g_k \rangle \tag{3}$$

for some $\gamma \in (0, 1)$. The Armijo backtracking rule chooses the integer h_k minimal, such that (3) is satisfied for $\alpha_k = a\beta^{h_k}$, where $a > 0$, $\beta \in (0, 1)$.

It has been proven in [1] that for $f \in C^1$ with gradient $g(x)$ Lipschitz continuous in $\mathbb{R}^n$, the Armijo backtracking rule is enough to ensure global convergence to a first order critical point, provided that $\langle d_k, g_k \rangle / \|d_k\|$ does not converge to 0 and f is bounded below.

Hence, if this rule is applied to Newton's or a Newton-like method, global convergence can be guaranteed, even though this is not necessarily the case when the whole Newton step is taken. On the other hand, it is very restrictive and can destroy the particularly fast convergence of these methods near a minimizer. So it is worth considering a less restrictive rule for choosing the steplength.

Probably the first nonmonotone line search algorithm was the watchdog technique in [2]. It proposed a standard and a relaxed condition for choosing the steplength, where the relaxed condition was used, when the value of the objective function at the full step $x_k + d_k$ was 'sufficiently less' than the minimal value of the objective function in the last few iterations.

2.1 Nonmonotone Armijo condition

In 1986, Grippo, Lampariello and Lucidi proposed a nonmonotone generalization of the Armijo condition (3) in [3], which was used in several subsequent papers and led to new nonmonotone algorithms. They proposed the following algorithm GLL for miminizing a twice continuously differentiable function f in problem (1), in which the steplength for $x_{k+1} = x_k + \alpha_k d_k$ is defined as $\alpha_k = a\beta^{h_k}$, for given $a > 0$ and $\beta \in (0,1)$, where h_k is the first nonnegative integer h for which the nonmonotone Armijo condition

$$f(x_k + a\beta^h d_k) \leq \max_{0 \leq j \leq m(k)} [f(x_{k-j})] + \gamma a \beta^h \langle g_k, d_k \rangle \tag{4}$$

is satisfied, where

$$m(0) = 0, \quad 0 \leq m(k) \leq \min [m(k-1) + 1, M] \quad \text{for } k \geq 1. \tag{5}$$

for a given nonnegative integer M.

Algorithm 1 (GLL)
Data: x_0, *integer* $M \geq 0$, $c_1 > 0$, $c_2 > 0$, $\gamma \in (0,1)$, $\beta \in (0,1)$.
Step 1: *Set* $k = 0$, $m(0) = 0$ *and compute* $f_0 = f(x_0)$.
Step 2: *Compute* g_k.
If $g_k = 0$ *stop, else compute* H_k.
If H_k *is singular, set* $d_k = -g_k$, $m(k) = 0$ *and go to Step 5.*
Step 3: *Compute* $d_k = -H_k^{-1} g_k$.
If $|\langle g_k, d_k \rangle| < c_1 \|g_k\|^2$ *or* $\|d_k\| > c_2 \|g_k\|$,
set $d_k = -g_k$, $m(k) = 0$ *and go to Step 5.*
Step 4: *If* $\langle g_k, d_k \rangle > 0$, *set* $d_k = -d_k$.
Step 5: *Set* $\alpha = 1$.
Step 6: *Compute* $f_\alpha = f(x_k + \alpha d_k)$.
If $f_\alpha \leq \max_{0 \leq j \leq m(k)} [f_{k-j}] + \gamma \alpha \langle g_k, d_k \rangle$,
set $f_{k+1} = f_\alpha$, $x_{k+1} = x_k + \alpha d_k$, $k = k + 1$,
$0 \leq m(k) \leq \min \{m(k-1) + 1, M\}$ *and go to Step 2.*
Step 7: *Set* $\alpha = \beta\alpha$ *and go to Step 6.*

Furthermore, they proved (for an even more general algorithm) that, if the level set $\Omega_0 = \{x : f(x) \leq f(x_0)\}$ is compact, then $\{x_k : k \in \mathbb{N}\} \subset \Omega_0$, and for every limit point $\bar{x}$ of $\{x_k\}$, $g(\bar{x}) = 0$ and $\bar{x}$ is not a local maximum. In the case where the number of stationary points in Ω_0 is finite, even global convergence has been shown.

A test of an implementation of algorithm GLL on several example problems showed that, compared to the monotone version ($M = 0$), a value of $M = 5$ or $M = 10$ is generally competitive and in most cases more efficient. Moreover, it could be observed that the nonmonotone Armijo backtracking is more valuable in the intermediate and final steps of the minimization process and that it is better to include a few monotone Armijo steps in the beginning.

2.2 Nonmonotone line search on a convex feasible set

In [5], the problem

$$\text{minimize} \quad f(x) \quad \text{subject to} \quad x \in \Omega \tag{6}$$

was considered by Birgin, Martínez and Raydan, where Ω is a closed convex set in $\mathbb{R}^n$. For this purpose, the orthogonal projection $P(z)$ onto Ω was used to ensure feasibility at every iteration. Hence, the following spectral projected gradient algorithm (SPG) is most valuable for feasible sets, where the projection can easily be computed, e.g. box- or ball-shaped regions or polygons.

The search direction is essentially the steepest descent direction $-g_k$, but projected back onto Ω. In every iteration, a trial point $x_+ = x_k + \lambda d_k$ (where $x_k + d_k \in \Omega$) is accepted if it satisfies an analogy to the nonmonotone Armijo condition (4):

$$f(x_+) \leq \max_{0 \leq j \leq \min[k,M])} [f(x_{k-j})] + \gamma \lambda \langle g_k, d_k \rangle \tag{7}$$

The parameters are $M \geq 0$, $\alpha_{max} > \alpha_{min} > 0$, $\alpha_0 \in [\alpha_{min}, \alpha_{max}]$, $\gamma \in (0, 1)$ and $0 < \sigma_1 < \sigma_2 < 1$. Given $x_k \in \Omega$ and $\alpha_k \in [\alpha_{min}, \alpha_{max}]$, the following algorithm describes how to compute x_{k+1} and α_{k+1}.

Algorithm 2 (SPG)
Step 1: *If $\|P(x_k - g(x_k)) - x_k\| = 0$, stop, declaring that x_k is stationary.*
Step 2: Backtracking:
Step 2.1: *Compute $d_k = P(x_k - \alpha_k g_k) - x_k$. Set $\lambda = 1$.*
Step 2.2: *Set $x_+ = x_k + \lambda d_k$.*

Step 2.3: *If (7) is satisfied,*
then set $\lambda_k = \lambda$, $x_{k+1} = x_+$, $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$ *and go to Step 3,*
else, choose $\lambda_{new} \in [\sigma_1\lambda, \sigma_2\lambda]$, *set* $\lambda = \lambda_{new}$ *and go to Step 2.2.*
Step 3: *Compute* $b_k = \langle s_k, y_k \rangle$.
If $b_k \leq 0$, *set* $\alpha_{k+1} = \alpha_{max}$,
else, compute $a_k = \langle s_k, s_k \rangle$ *and* $\alpha_{k+1} = \min[\alpha_{max}, \max[\alpha_{min}, a_k/b_k]]$

Any accumulation point of this algorithm is a constrained stationary point and, even though, global convergence has not been proven explicitly, the original theorem by Grippo, Lampariello and Lucidi in [3], which only needs d_k to be a descent direction, supports the assumption of global convergence of SPG in most cases. This, as well as the efficiency of the algorithm, has been corroborated in various numerical experiments for feasible sets with easy projections in [5] and - for more specific problems containing polygonial feasible sets - in [6].

More precisely, a FORTRAN 77 implementation was compared to the LANCELOT package on all the bound constrained problems from the CUTE collection which have more than 50 variables. Except for four problems, where the minimizer could not be obtained within 50000 iterations, algorithm SPG (with $M = 10$) reached the same minimal function value, and was faster in 29 out of the remaining 40 problems (72.5%). In the cases where the algorithm failed, it stopped near a local minimizer. This reveals slow local convergence. Furthermore, if the minimum of f is unconstrained and 'far' from the boundary $\partial\Omega$, this algorithm is practically the (nonmonotone) steepest descent method, thus, not particularly efficient. Hence, a Newton-like variation might be interesting as well.

2.3 The F-rule

J. Han, J. Sun and W. Sun found a generalization for several popular nonmonotone stepsize rules for unconstrained optimization (1), including the nonmonotone Armijo rule. In [7], they first defined a forcing function (F-function) to be a function $\sigma : [0, \infty) \mapsto [0, \infty)$, such that for any sequence $\{t_i\} \subset [0, \infty)$

$$\lim_{i\to\infty} \sigma(t_i) = 0 \quad \Rightarrow \quad \lim_{i\to\infty} t_i = 0 \tag{8}$$

and the nonmonotone F-rule as

$$f(x_k + \lambda_k d_k) \leq \max_{0\leq j\leq m(k)} [f(x_{k-j})] - \sigma(t_k), \tag{9}$$

for $\lambda_k \geq 0$ bounded above, $m(k)$ according to (5), σ being a forcing function and $t_k = -\langle d_k, g_k\rangle / \|d_k\|$.

The mapping $\delta : [0, \infty) \mapsto [0, \infty)$ is defined as

$$\delta(t) = \begin{cases} \inf\{\|x - y\| : \|g(x) - g(y)\| \geq t\}, & t \in [0, \eta), \\ \lim_{s \to \eta, s < \eta} \delta(s) & t \in [\eta, \infty), \end{cases} \tag{10}$$

where $\eta = \sup\{\|g(x) - g(y)\| : x, y \in \Omega_0\}$. In this context, the nonmonotone Armijo rule (4) satisfies the nonmonotone F-rule with $\sigma(t) = \gamma\beta t\delta((1-\gamma)t)$.

It has been proven that, if f is continuously differentiable, Ω_0 is compact, λ_k is defined by the nonmonotone F-rule (9) with forcing function σ, $x_{k+1} = x_k + \lambda_k d_k$,

$$\frac{|\langle d_k, g_k\rangle|}{\|d_k\|} \geq \sigma(\|g_k\|) \quad \text{and} \quad \|d_k\| \leq c_2\|g_k\| \quad \text{with} \quad c_2 > 0, \tag{11}$$

then $\{x_k\} \subset \Omega_0$ and every accumulation point of $\{x_k\}$ is a stationary point.

In particular, these requirements are satisfied if, for c_1, $c_2 > 0$,

$$\sigma(t) = \frac{c_1}{c_2} t, \quad \langle d_k, g_k\rangle \leq -c_1\|g_k\|^2 \quad \text{and} \quad \|d_k\| \leq c_2\|g_k\|. \tag{12}$$

Hence, one could also try an algorithm of the following form and examine its numerical performance.

Algorithm 3
All steps as in Algorithm 1 except for
Step 5: *Set $\alpha = 1$. Compute $t_k = -\langle g_k, d_k\rangle / \|d_k\|$.*
Step 6: *Compute $f_\alpha = f(x_k + \alpha d_k)$.*
If $f_\alpha \leq \max_{0 \leq j \leq m(k)} [f_{k-j}] + t_k c_1 / c_2$,
set $f_{k+1} = f_\alpha$, $x_{k+1} = x_k + \alpha d_k$, $k = k + 1$,
$0 \leq m(k) \leq \min [m(k-1) + 1, M]$ and go to Step 2.

2.4 A derivative-free method

Since often in applications, the derivatives of an objective function are not available or very expensive to compute, Diniz-Erhardt, Martínez and Raydan developed a derivative-free algorithm in [10]. In every iteration the gardient is approximated by the discrete

gradient $\hat{g}_k$. If then the search direction $d_k = -\hat{g}_k/\sigma_k$ is chosen, this is most likely to be a descent direction.

Furthermore, their algorithm chooses a random search direction from time to time. This need not, of course, be a descent direction. If the step $x_k + d_k$ cannot be accepted by (13), then the stepsize is computed with secured quadratic interpolation (see Step 2 of Algorithm 4)

The algorithm also uses a relaxed nonmonotone Armijo condition

$$f(x_k + \alpha d_k) \leq \bar{f}_k + \eta_k - \alpha^2 \beta_k, \tag{13}$$

$$\text{where} \quad \eta_k > 0 \ \forall k, \quad \sum_{k=0}^{\infty} \eta_k = \eta < \infty \tag{14}$$

$$\text{and} \quad \bar{f}_k = \max_{0 \leq j \leq \min[M,k]} [f(x_{k-j})]\,. \tag{15}$$

The series $\{\beta_k\}$ has been chosen, such that for all infinite subsets $K \in \mathbb{N}$,

$$\lim_{k \in K} \beta_k = 0 \quad \Rightarrow \quad \lim_{k \in K} g(x_k) = 0. \tag{16}$$

The choice $\beta_k \equiv 1$, satisfies this condition trivially.

The following algorithm is essentially the discrete gradient type algorithm proposed in [10], except for two changes: The steps have been reorganized and we left out the part, where x_{k+1} is adjusted if, during the computation of the discrete gradient, an auxiliary point with smaller function value is found. Also, for g_0, Step 4 (which can easily be put into a subroutine) has to be performed with $x_{-1} = 0$ during the initialisation. So for given x_k, g_k, σ_k, the following algorithm describes how to compute x_{k+1}, g_{k+1} and σ_{k+1}.

Algorithm 4 (DGA)

Step 1: Choose search direction:
Compute a random real number $0 < z < 1$.
If $z \leq p$ *then compute a random direction* $d_k \in \mathbb{R}^n$ *such that*
$\Delta_{min} \leq \|d_k\| \leq \Delta_{max}$,
else, compute $d_k = -\hat{g}_k/\sigma_s$.

Step 2: Backtracking:
Step 2.1: *Set* $\alpha = 1$, *compute* $\bar{f}_k$ *according to* (15).
Step 2.2: *If* (13) *holds, set* $\alpha_k = \alpha$, *else go to Step 2.5.*
Step 2.3: *If* $\alpha_k = 1$, *go to Step 3 (Extrapolation).*

Step 2.4: *Set $x_{k+1} = x_k + \alpha_k d_k$ and go to Step 4 (Discrete Gradient).*
Step 2.5: *Set $d = \alpha_k d_k$, compute $b = f(x_k + d) + f(x_k - d) - 2f(x_k)$.*
If $b \leq 0$, go to Step 2.7.
Step 2.6: *Compute $t = (f(x_k - d) - f(x_k + d))/b$*
If $t \in [\tau_{min}, \tau_{max}]$ set $\alpha_{new} = t\alpha$ and go to Step 2.8.
If $t \in [-\tau_{max}, -\tau_{min}]$ set $\alpha_{new} = -t\alpha$, $d_k = -d_k$ and go to Step 2.8.
Step 2.7: *Set $\alpha_{new} = \tau_{max}\alpha$.*
If $f(x_k - d) \leq f(x_k + d)$ set $d_k = -d_k$.
Step 2.8: *Set $\alpha = \alpha_{new}$ and go to Step 2.2.*

Step 3: Extrapolation:
Step 3.1: *Set $c = 1$.*
Step 3.2: *If $2c > c_{max}$ set $x_{k+1} = x_k + c\alpha_k d_k$ and go to Step 4 (Discrete Gradient).*
Step 3.3: *If $f(x_k + 2c\alpha_k d_k) > f(x_k + c\alpha_k d_k)$*
set $x_{k+1} = x_k + c\alpha_k d_k$ and go to Step 4 (Discrete Gradient).
Step 3.4: *Set c=2c and go to Step 3.2.*

Step 4: Discrete Gradient:
Step 4.1: *For $j = 1, \ldots, n$ execute Steps 4.2-4.4.*
Step 4.2: *Set $h = \varepsilon_k$. If $[x_{k+1}]_j < [x_k]_j$, set $h = -h$.*
Step 4.3: *Set $z = x_{k+1} + he_j$.*
Step 4.4: *Set $[\hat{g}_{k+1}]_j = [f(z) - f(x)]/h$.*

Step 5: *Compute σ_{k+1} according to* (17).

The following choice of σ_{k+1} has been proposed in [10], but other methods are possible as well:

$$\sigma_{k+1} \quad = \quad \max\left[\sigma_{min}, \min\left[\sigma_{max}, \frac{\langle \hat{g}_{k+1} - \hat{g}_k, x_{k+1} - x_k \rangle}{\|x_{k+1} - x_k\|}\right]\right]. \tag{17}$$

The following convergence statement has been proven for algorithm DGA in [10]. If the level set $\{x \in \mathbb{R}^n : f(x) \leq f(x_0) + \eta\}$ is bounded, $0 < \theta < 1$, $0 < \Delta_{min} < \Delta_{max} < \infty$ and for infinitely many d_k

$$\|d_k\| \in [\Delta_{min}, \Delta_{max}] \quad \text{and} \quad \langle d_k, g_k \rangle \leq -\theta \|g_k\| \|d_k\|, \tag{18}$$

then for all $\varepsilon > 0$ there exists $k \in \mathbb{N}$ such that $\|g_k\| \leq \varepsilon$. Since an approximate steepest descent direction is chosen, it is very likely that (18) holds infinitely often and every stopping criterion $\|g_l\| \leq tol$ will eventually be satisfied.

Another version of the algorithm uses the search direction $d_k = -H_k^{-1}\hat{g}_k$ with the inverse of the Hessian approximation H_k obtained from $y_k = \hat{g}_{k+1} - \hat{g}_k$ and $s_k = x_{k+1} - x_k$ as follows.

If

$$|\langle (s_k - H_k^{-1}y_k), y_k\rangle| \leq \rho\|y_k\|\|s_k - H_k^{-1}y_k\|, \tag{19}$$

then set $H_{k+1}^{-1} = H_k^{-1}$, else,

$$H_{k+1}^{-1} = H_k^{-1} + \frac{(s_k - H_k^{-1}y_k)(s_k - H_k^{-1}y_k)^T}{\langle (s_k - H_k^{-1}y_k), y_k\rangle} \tag{20}$$

The test (19) has been added to avoid the denominator beeing (numerically) zero which leads to instabilities. For this version, too, stationary points can be found to an arbitrary precision with probability 1.

For both alternatives, a FORTRAN implementation has been tested on several example problems. The reliability (shown e.g. in a success rate of around 70% for the steepest descent version) is satisfactory. For many problems, the Newton-type version needs less iterations than the first one, as is expected from theory. In both cases, an increase of the parameter p, which regulates the probability of random search directions, increases the stability but also the number of function evaluations, hence a value of 0.05 has been proposed as compromise. All in all, the algorithms seem to be quite promising for use in an application.

3 Nonmonotone trust region methods

In trust region methods, the objective function f is modelled in every iteration by a model function m_k, which approximates $f(x)$ around x_k. Usually, this is the second Taylor approximation of f at x_k, possibly with a bounded approximation to the Hessian. Then, a step s_k is chosen within a small region $\{s : \|s\| \leq \Delta_k\}$ (where $\|\cdot\|$ can be any norm), such that s_k (approximately) solves the subproblem

$$\min m_k(x_k + s) \quad \text{subject to} \quad \|s\| \leq \Delta_k. \tag{21}$$

If $x_k + s_k$ decreases the objective function sufficiently, the step is accepted and, according to how well the objective function has been modelled, the new trust region radius Δ_{k+1} may be increased. If the decrease of the objective function was too small, or if even $f(x_k + s_k) > f(x_k)$, then the step is rejected, x_{k+1} is set to x_k and the trust region radius is reduced. The decrease of the objctive funtion is measured with the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m(x_k + s_k)}. \tag{22}$$

The trial point is accepted if $\rho_k \geq \eta_1$, and the trust region radius is updated as follows:

$$\Delta_{k+1} = \begin{cases} \gamma_2 \Delta_k, & \rho_k > \eta_2 \\ \Delta_k, & \rho \in [\eta_1, \eta_2] \\ \gamma_1 \Delta_k, & \rho < \eta_1 \end{cases} \tag{23}$$

$$\text{where} \quad 0 < \eta_1 < \eta_2 < 1, \quad 0 < \gamma_1 < 1 < \gamma_2$$

So the standard (monotone) trust region algorithm looks as follows:

Algorithm 5

Step 0: *Initialize data*
Step 1: *Define the model function m_k.*
Step 2: *Calculate the trial step $x_k + s_k$ from* (21).
Step 3. *Compute ρ_k according to* (22).
If $\rho_k < \eta_1$ define $x_{k+1} = x_k$.
Otherwise set $x_{k+1} = x_k + s_k$.
Step 4: *Compute Δ_{k+1} by* (23) *and go to Step 1.*

Of course, this is merely a very simplified model algorithm, which does not even have a stopping criterion, but it is enough to show the general procedure of trust region methods. More sophisticated implementations also use an inner and an outer cycle for finding Δ_k and an acceptable step s_k, and reuse data such as g_k and H_k.

Several nonmonotone variations of trust region methods have been proposed, which mainly change the criteria, when to accept a trial step s_k in Step 3, but may also include precise descriptions of how to define the model function and how to obtain the trial step.

3.1 Nonmonotone trust region method on a convex feasible set

In [11], Toint considers problem (6), where x is constrained to a closed convex set $\Omega \in \mathbb{R}^n$ and $P(z)$ is the orthogonal projection of z onto Ω. Hence, this has to be taken

into account when computing the trial step, which needs to be in Ω as well. The trial step is then accepted or rejected by consideration of

$$\rho_k = \max\left[\frac{f_r - f(x_k + s_k)}{\sigma_r}, \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}\right], \tag{24}$$

where f_r is a reference value of the objective function of the last h few steps and σ_r the sum of decreases of the model functions since that reference value occured. Furthermore, f_c, the maximal value of f in the last h iterations, is a candidate reference value of f. This is taken as new reference value if l, the number of iterations since the overall best value f_{min} was attained, reaches h. The whole algorithm is as follows, where the model function and the trial step are computed as in any (constrained) monotone trust region algorithm.

Algorithm 6 (NMTR)

Step 0: *Initialize data:* $\eta_1, \eta_2, \gamma_1, \gamma_2, h, x_0, \Delta_0$.
Set $k = l = 0$, $f_{min} = f_r = f_c = f(x_0)$, $\sigma_r = \sigma_c = 0$.
Step 1: *Define the model function* m_k.
Step 2: *Calculate the trial step* $x_k + s_k$ *by* (21).
Step 3: Acceptance of trial step:
Step 3.1: *Compute* ρ_k *according to* (24).
Step 3.2: *If* $\rho_k < \eta_1$ *define* $x_{k+1} = x_k$ *and go to Step 4.*
Step 3.3: *Set* $x_{k+1} = x_k + s_k$, $l = l + 1$,
$\sigma_c = \sigma_c + m_k(x_k) - m_k(x_{k+1})$,
$\sigma_r = \sigma_r + m_k(x_k) - m_k(x_{k+1})$.
Step 3.4: *If* $f(x_{k+1}) < f_{min}$ *then set*
$f_c = f_{min} = f(x_{k+1})$, $\sigma_c = 0$, $l = 0$
and go to Step 4.
Step 3.5: *If* $f(x_{k+1}) > f_c$ *then set*
$f_c = f(x_{k+1})$ *and* $\sigma_c = 0$.
Step 3.6: *If* $l = h$, *set* $f_r = f_c$ *and* $\sigma_r = \sigma_c$.
Step 4: *Compute* Δ_{k+1} *by* (23) *and go to Step 1.*

Toint showed in several numerical experiments that his algorithm is reliable and outperforms the classical monotone trust region algorithm SBMIN in speed. In addition, he showed global convergence of the algorithm to a constrained stationary point for f twice continuously differentiable under some mild assumptions on the model function m_k and the trial step $x_k + s_k$.

The code for SBMIN of the LANCELOT package has been taken as a starting point to implement algorithm NMTR, and these two algorithms were tested on 390 problems from the CUTE collection. It could be observed that NMTR matches the excellent reliability of SBMIN. Furthermore, NMTR improves the number of iterations as well as function evaluations and the overall execution time by about 30%. The performance of NMTR is especially good for small problems and, since the average gain in computational time is achieved in only a few problems, this algorithm is most interesting for problems which are hard for SBMIN.

3.2 Box-constrained trust region method

In [13], Chen, Han and Xu considered the box constrained optimization problem:

$$\min f(x) \quad \text{subject to} \quad l \leq x \leq u, \tag{25}$$

where $l, x, u \in \mathbb{R}^n$ and $-\infty < l < u < +\infty$ pointwise. To coincide with these constraints, the indices $1, \ldots, n$ are divided into the following sets at each iteration for a given trust region radius Δ:

$$I_k^a = \{i : x_{ki} = l_i, g_{ki} \geq 0 \text{ or } x_{ki} = u_i, g_{ki} \leq 0\} \quad \text{the active set,} \tag{26}$$

$$I_k^s = \{i : x_{ki} = l_i, g_{ki} < 0 \text{ or } x_{ki} = u_i, g_{ki} > 0\} \quad \text{the semi-active set,} \tag{27}$$

$$I_k^+(\Delta) = \{i : 0 < x_{ki} - l_i < \min[\varepsilon_0, \delta_k, \Delta] \text{ or } 0 < u_i - x_{ki} < \min[\varepsilon_0, \delta_k, \Delta]\}, \tag{28}$$

$$\text{where} \quad \delta_k = \sum_{i \notin I_k^a} |g_{ki}| \quad \text{and} \quad \varepsilon_0 = \frac{1}{4} \min_{1 \leq i \leq n} [u_i - l_i],$$

$$I_k'(\Delta) = \{1, \ldots, n\} \setminus (I_k^a \cup I_k^s \cup I_k^+(\Delta)). \tag{29}$$

The components $s_{ki}(\Delta)$ of the trial step $s_k(\Delta)$, are defined by considering each coordinate of x_k. If $i \in I_k^a$, that is if x_{ki} sits on the boundary and g_{ki} points into the feasible set, i.e. we can only achieve improvement of the objective function by going beyond the boundary, then $s_{ki}(\Delta)$ is set to zero. If $i \in I_k^s$, that is if x_{ki} sits on the boundary and we can achieve improvement by going into the feasible set, i.e. g_{ki} points outward, then $s_{ki}(\Delta)$ is set to $-\text{sign}(g_{ki})$ and scaled appropriately according to the data of the problem and the trust region radius Δ. If $i \in I_k^+$, x_{ki} is inside the feasible set, but very close to the boundary. In this case, we either step onto the boundary (if g_{ki} points inward), or, with an appropriate step length, further into the feasible set (if g_{ki} points outward).

For the remaining $i \in I_k'$, $s_{ki}(\Delta)$ can be computed by considering the unconstrained trust region subproblem (21) with reduced dimension ($g_{ki}, [H_k]_{ij}$, where $i, j \in I_k'$) and the reduced trust region radius

$$\tilde{\Delta} = \sqrt{\left(\Delta^2 - \sum_{i \in I_k^s \cup I_k^+(\Delta)} s_{ki}^2(\Delta)\right)} \tag{30}$$

Finally, the ratio of the acceptance of the trial step is defined as

$$\rho_k = \frac{f(x_k + s_k(\Delta)) - \max_{0 \le j \le m(k)}[f_{k-j}]}{\langle g_k, s_k(\Delta)\rangle + \frac{1}{2}\langle s_k(\Delta), \hat{H}_k s_k(\Delta)\rangle} \tag{31}$$

with $m(k)$ as in (5) and

$$[\hat{H}_k]_{ij} = \begin{cases} [H_k]_{ij}, & i, j \notin I_k^+ \cup I_k^s \\ 0, & \text{else.} \end{cases} \tag{32}$$

Furthermore, Chen, Han and Xu showed that in the monotone version of their algorithm ($m(k) \equiv 0$), the correct active set is found after a finite number of iterations if the strict complementary slackness condition holds. In this case, the algorithm is reduced to an unconstrained trust region method with smaller dimension.

Numerical test of the FORTRAN implementation of the algorithm on several problems with two different solvers for the unconstrained subproblem and several values of M showed that the nonmonotone version is competitive with the monotone version and outperforms it for some special problems. Furthermore, it could be observed that a value of $M \in [8, 12]$ is usually better.

3.3 Equality constrained trust region method without penalty function

In [14], M. Ulbrich and S. Ulbrich described an algorithm to solve the equality constrained problem

$$\min f(x) \quad \text{subject to} \quad c(x) = 0, \tag{33}$$

where f and c are continuously differentiable, with $A = \nabla c$ the transposed Jacobian of c. Their algorithm was motivated by SQP filter methods, but instead of using a filter, they allowed nonmonotony in both, the objective function and the feasibility.

In each iteration, the trial step s_k is computed as sum of a quasi-normal step and a tangential step. The quasi-normal step s_k^n is obtained from the subproblem

$$\min \|c(x_k) + A(x_k)^T s^n\|^2 \quad \text{subject to} \quad \|s^n\| \leq \Delta_k \tag{34}$$

and is made to improve feasibility. The tangential step s_k^t, on the other hand, is made to dercrease the quadratic model

$$q_k(s) = \langle (g_k + A_k y_k), s\rangle + \frac{1}{2}\langle s, H_k s\rangle \tag{35}$$

of the Lagrange function

$$\ell(x, y) = f(x) + \langle y, c(x)\rangle, \quad y \in \mathbb{R}^m, \tag{36}$$

where H_k is a symmetric approximation of $\nabla_x^2 \ell(x_k, y_k)$ and y_k a Langrange multiplier estimate. Hence, s_k^t is obtained from the subproblem

$$\min q_k(s_k^n + s^t) \quad \text{subject to} \quad A_k^T s^t = 0, \quad \|s^t\| \leq \Delta_k. \tag{37}$$

The step is acceptable for the constraints, if the quotient of the relaxed actual reduction and the predicted reduction for the feasibility

$$\frac{\text{rared}_k^c}{\text{pred}_k^c} = \frac{\max\left[R_k, \sum_{r=0}^{\nu_k^c - 1} \lambda_{kr}^c \|c_{k-r}\|^2\right] - \|c(x_k + s_k)\|^2}{\|c_k\|^2 - \|c_k + A_k^T s_k\|^2} \geq \eta_1, \tag{38}$$

where

$$\begin{gathered} \eta_1 \in (0,1) \text{ fixed}, \quad \nu^c \in \mathbb{N}, \quad \lambda \in (0, 1/\nu^c), \\ \nu_k^c = \min[k+1, \nu^c], \quad \lambda_{kr}^c \geq \lambda > 0, \quad \sum_{r=0}^{\nu_k^c - 1} \lambda_{kr}^c = 1, \\ R_k \geq \|c_k\|^2, \qquad \text{usually } R_k = \|c_k\|^2. \end{gathered} \tag{39}$$

For the acceptability of the deacrease of the Langrange function, we need the following values

$$\text{pred}_k^t = q_k(s_k^n) - q_k(s_k), \quad \text{the predicted tangential reduction of } \ell, \tag{40}$$

$$\text{pred}_k^\ell = -q_k(s_k), \qquad \text{the predicted reduction of } \ell \text{ for the whole step}, \tag{41}$$

$$\text{rared}_k^\ell = \max\left[\ell_k, \sum_{r=0}^{\nu_k^\ell - 1} \lambda_{kr}^\ell \ell_{k-r}\right] - \ell(x_k - s_k, y_k), \quad \text{the relaxed actual reduction} \tag{42}$$

with the parameters similar to (39).

A step s_k is then acepted if (38) holds and, for $\mu \in (2/3, 1)$, $\gamma \in (0, 1)$, the following is true:

$$\begin{aligned} \text{If} \quad & \text{pred}_k^t \geq \max[\text{pred}_k^c, (\text{pred}_k^c)^\mu] \quad \text{and} \quad \text{pred}_k^\ell \geq \gamma \text{pred}_k^t \\ \text{then} \quad & \text{rared}_k^\ell \geq \eta_1 \text{pred}_k^\ell. \end{aligned} \tag{43}$$

So we only consider the decrease of the Langrange function, if the tangential step is promising and we expect a better improvement of ℓ than of the feasibility. Otherwise we are satisfied with sufficient reduction of c.

If the functions f and c, as well as their derivatives, behave nicely within an open convex subset of $\mathbb{R}^n$ and the series $\{x_k\}$ stays inside of this set, then global convergence to a feasible, constrained stationary point has been proven.

Furthermore, M. Ulbrich and S. Ulbrich proved quadratic convergence in the case that f and c are twice continuously differentiable and $\nabla^2 f$ and $\nabla^2 c$ are Lipschitz continuous in a neighbourhood of a stationary point. To achieve this, one has to apply an SQP-Newton-type step rule with the exact Hessian of the Langrange function near the stationary point and choose the Lagrange multiplier estimates y_k carefully. Then the algorithm will eventually take the whole Newton step and, hence, converge quadratically.

The numerical results of the MATLAB implementation presented in [14] are particularly satisfactory. Compared to the LANCELOT package on a set of CUTE problems, it demonstrated its robustness and efficiency. For all problems, the same minimal function value has been found - mostly with fewer evaluations of f, c, ∇f, and ∇c. The locally quadratic convergence could also be observed. The analytical and numerical results show that this whole new class of algorithms, which has been opened up, is competitive with standard techniques and is worth being examined in more detail.

3.4 Trust region method for general constraints and simple bounds

In [15], Xu, Han and Chen considered a combination of the two problems (25) and (33), namely

$$\min f(x) \quad \text{subject to} \quad c(x) = 0 \quad \text{and} \quad l \leq x \leq u \tag{44}$$

They, too, divided the step into a normal and a tangential component. The former is the solution $v_k(\Delta)$ to the subproblem

$$\begin{aligned} \min \quad & \phi_k(v) = 0.5\|c_k + A_k^T v\|^2 + 0.5\|v\|^2 \\ \text{s.t.} \quad & l \leq x_k + v \leq u \\ & \|v\|_\infty \leq 0.8\Delta, \end{aligned} \tag{45}$$

where the term $0.5\|v\|^2$ ensures that this is a strictly convex program. The whole step is, then, the solution $s_k(\Delta)$ to

$$\begin{aligned} \min \quad & \psi_k(s) = \langle g_k, s\rangle + 0.5\langle s, H_k s\rangle \\ \text{s.t.} \quad & A_k^T s = A_k^T v_k(\Delta) \\ & l \leq x_k + s \leq u \\ & \|s\|_\infty \leq \Delta. \end{aligned} \tag{46}$$

For the ratio ρ_k, the l_2 exact penalty function

$$F(x, \sigma) = f(x) + \sigma \left(\sum_{i=1}^{m} c_i^2(x) + \sum_{i=1}^{n} (\min[x_i - l_i, u_i - x_i]) \right) \tag{47}$$

is used, which, as $l \leq x_k \leq u$, is reduced to

$$F_k = F(x_k, \sigma_k) = f(x_k) + \sigma_k \|c_k\|. \tag{48}$$

So, we compute the ratio of the actual and the predicted reduction as follows,

$$\text{ared}_k(\Delta) = F(x_k + s_k(\Delta), \sigma_k) - \max_{0 \leq j \leq m(k)} [F_{k-j}] \tag{49}$$

$$\text{pred}_k(\Delta) = \langle g_k, s_k(\Delta)\rangle + 0.5\langle s_k(\Delta), H_k s_k(\Delta)\rangle + \sigma_k(\|c_k + A_k^T s_k(\Delta)\| - \|c_k\|) \tag{50}$$

$$\rho_k(\Delta) = \frac{\text{ared}_k(\Delta)}{\text{pred}_k(\Delta)}, \tag{51}$$

with $m(k)$ maximal as in (5). Even though, a further variable M_k has been introduced, this can be ignored as it was initialized with M and could only grow, hence always, $m(k) \leq M \leq M_k$.

For the algorithm from these ingredients, a detailed convergence analysis for the two cases $\liminf_k \Delta_k = 0$ and $\liminf_k \Delta_k > 0$ is presented in [15]. Provided the functions and their derivatives are nice and the iterates stay within a convex set, the existence of a φ-stationary (see (52)) or even a substationary (see (53)) limit point has been proven.

$x^\star$ is a φ-stationary point, if

$$\begin{aligned} &l \leq x^\star \leq u \text{ and } \exists \mu_l, \geq 0,\ \mu_u \geq 0 \text{ such that} \\ &\langle \mu_l, (x^\star - l)\rangle = 0, \quad \langle \mu_u, (u - x^\star)\rangle = 0, \\ &A(x^\star)c(x^\star) - \mu_l + \mu_u = 0. \end{aligned} \tag{52}$$

$x^\star$ is a substationary point, if

$$\begin{aligned} &l \leq x^\star \leq u \text{ and } \exists \mu_l, \geq 0,\ \mu_u \geq 0,\ \lambda \in \mathbb{R}^m \text{ such that} \\ &\langle \mu_l, (x^\star - l)\rangle = 0, \quad \langle \mu_u, (u - x^\star)\rangle = 0, \\ &g(x^\star) - A(x^\star)\lambda - \mu_l + \mu_u = 0. \end{aligned} \tag{53}$$

To avoid the Maratos effect, a second order correction has been proposed as s'_k which solves (55), by using the solution v'_k of (54).

$$\begin{aligned} \min \quad & \phi'_k(v) = 0.5\|c(x_k + s_k(\Delta)) + A_k^T v\|^2 + 0.5\|v\|^2 \\ \text{s.t.} \quad & l \leq x_k + s_k(\Delta) + v \leq u \\ & \|v + s_k(\Delta)\|_\infty \leq 0.8\Delta \end{aligned} \tag{54}$$

$$\begin{aligned} \min \quad & \psi'_k(s) = \langle g_k, (s + s_k(\Delta))\rangle + 0.5\langle (s + s_k(\Delta)), H_k(s + s_k(\Delta))\rangle \\ \text{s.t.} \quad & A_k^T s = A_k^T v'_k(\Delta) \\ & l \leq x_k + s_k(\Delta) + s \leq u \\ & \|s + s_k(\Delta)\|_\infty \leq \Delta \end{aligned} \tag{55}$$

The next iteration point is then set to $x_{k+1} = x_k + s_k(\Delta) + s'_k(\Delta)$.[1]

[1] In Algorithm 3.1 of [15], $x_{k+1} = x_k + s_k(\Delta) + s'_k(\Delta) + s'_k(\Delta)$ (in the suitable notation), but we assumed that this is a misprint.

The following formulas have been taken out of the algorithm for clarity:

$$\text{pred}_k(\Delta) \leq 0.5\sigma_k(\Delta)(\|c_k + A_k^T s_k(\Delta)\| - \|c_k\|) \tag{56}$$

$$\sigma_k(\Delta) = 2\max\left[\frac{\langle g_k, s_k(\Delta)\rangle + 0.5\langle s_k(\Delta), H_k s_k(\Delta)\rangle}{\|c_k\| - \|c_k + A_k^T s_k(\Delta)\|} + \alpha, \sigma_k\right] \tag{57}$$

$$\rho_k' = \frac{F(x_k + s_k(\Delta) + s_k'(\Delta), \sigma_k(\Delta)) - \max_{0\leq j\leq m(k)}[F_{k-j}]}{\text{pred}_k(\Delta)} \tag{58}$$

Algorithm 7

Step 0: *Initialize data:* $l \leq x_0 \leq u,\ H_0,\ \Delta_{start} > 0,\ \eta_1, \eta_2 \in (0,1)$, $M \geq 0,\ \sigma_o > 0,\ k = 0,\ m(0) = 0.$

Step 1: *Set* $\Delta = \Delta_{start}$.

Step 2: *Calculate* $f_k,\ g_k,\ c_k,\ A_k$.

Step 3: *Obtain* $v_k(\Delta)$ *from* (45) *and* $s_k(\Delta)$ *from* (46). *If* $v_k(\Delta) = 0$ *and* 0 *is a stationary point of* (46) *then stop.*

Step 4: *Set* $\sigma_k(\Delta) = \sigma_k$. *Compute* $\text{pred}_k(\Delta)$ *with* $\sigma_k(\Delta)$. *If* (56) *does not hold, compute* $\sigma_k(\Delta)$ *according to* (57).

Step 5: *Compute* ρ_k *from* (51) *with* $\sigma_k(\Delta)$. *If* $\rho_k \geq 0.75$ *then set* $s_k = s_k(\Delta)$ *and go to Step 7. Otherwise, compute* ρ_k' *from* (58) *with* $\sigma_k(\Delta)$.

Step 6: *If* $\rho_k' < \eta_1$, *set* $\Delta = \eta_2\Delta$ *and go to Step 3. Otherwise, set* $s_k = s_k(\Delta) + s_k'(\Delta)$.

Step 7: *Set* $x_{k+1} = x_k + s_k,\ \sigma_{k+1} = \sigma_k = \sigma_k(\Delta),\ m(k+1) = \min[m(k)+1, M]$, *Generate* H_k, *set* $k = k+1$ *and go to Step 1.*

The algorithm has been implemented in FORTRAN and tested with several example problems with small dimensions ($n, m \leq 5$). The results supplied in [15] only compared the monotone ($M = 0$) with the nonmonotone version of the algorithm. In most problems, the number of function evaluations could be improved with the nonmonotone version for at least one value of $M > 0$. Only for one problem, the number of function evaluations increased significantly compared to the monotone version for some values of M and hardly any decrease could be observed for other values of M. Hence, the algorithm is reliable due to the analytical results, but the numerical performance warrants further examination.

4 Conclusion

We have considered several nonmomotone line search and trust region algorithms for unconstrained and for different kinds of constrained optimization problems. Most of these are proven to converge globally to a (constrained) stationary point if the objective function and the constraints are nice and - in some cases - further mild assumptions hold.

All these algorithms have shown good performance in several numerical experiments. They are about as stable as the well-known monotone algorithms they have been compared to and, in most cases, more efficient, especially when applied to badly conditioned problems. Hence, the nonmonotone techniques are competitive with standard algorithms.

Other recent papers on nonmonotone optimization techniques, such as [8], where a nonmonotone algorithm is applied to solving large scale nonlinear systems of equations, [9], which uses a nonmonotone second order stepsize rule to find a second order stationary point, or [16], which considers the same problem as in 3.4, but also allows nonmonotony in the penalty parameter, show the topicality of this subject and the interest in nonmonotone optimization techniques among mathematicians and scientists throughout the world. Hence, we can expect further improvements of these methods in the future.

References

[1] N. Gould, *Contiuous Optimization (Lecture)*, University of Oxford, Hilary Term 2007, `http://web.comlab.ox.ac.uk/oucl/work/nick.gould/courses/co/paper.pdf`.

[2] R. M. Chamberlain, M. J. D. Powell, C. Lemarechal, and H. C. Pedersen, *The watchdog technique for forcing convergence in algorithms for constrained optimization*, Algorithms for constrained minimization of smooth nonlinear functions (A. G. Buckley and A. G. Goffin, eds.), Mathematical Programming Study, vol. 16, North-Holland Publishing Company, 1982, pp. 1–17.

[3] L. Grippo, F. Lampariello, and S. Lucidi, *A Nonmonotone Line Search Technique for Newton's Method*, SIAM Journal on Numerical Analysis, vol. 23, no. 4, Society for Industrial and Applied Mathematics, August 1986, pp. 707–716, available at `http://www.jstor.org/view/00361429/di976249/97p0101o/`.

[4] F. Bonnans, E. R. Panier, Tits. A. L., and J. Zhou, *Avoiding the Maratos effect by means of a nonmonotone line search II. Inequality constrained problems - feasible iterates*, SIAM Journal on Numerical Analysis, vol. 29, no. 4, Society for Industrial and Applied Mathematics, August 1992, pp. 1187–1202, available at `http://www.aemdesign.com/AlgorithmReferences/nonmonotoneII.pdf`.

[5] E. G. Birgin, J. M. Martínez, and M. Raydan, *Nonmonotone Spectral Projected Gradient Methods on Convex Sets*, SIAM Journal on Optimization, vol. 10, no. 4, Society for Industrial and Applied Mathematics, 2000, pp. 1196–1211, available at `http://www.ime.usp.br/~egbirgin/publications/bmr.pdf`.

[6] E. G. Birgin, J. M. Martínez, and M. Raydan, *SPG: Software for convex-constrained optimization*, ACM Transactions on Mathematical Software (TOMS), vol. 27, no. 3, ACM Press, 2001, pp. 340–349, available at `http://www.ime.usp.br/~egbirgin/publications/bmr2.pdf`.

[7] J. Han, J. Sun, and W. Sun, *Global Convergence of Non-Monotone Descent Methods for Unconstrained Optimization Problems*, Journal of Computational and Applied Mathematics, vol. 146, no. 1, North-Holland Publishing Company, 2002, pp. 89–98, available at `http://www.sciencedirect.com/science/journal/03770427`.

[8] W. La Cruz, J. M. Martínez, and Raydan M., *Spectral residual method without gradient information for solving large-scale nonlinear systems of equations*, Mathematics of Computation, vol. 75, no. 255, American Mathematical Society, July 2006, pp. 1429–1448, available at `http://www.ams.org/mcom/2006-75-255/S0025-5718-06-01840-0/home.html`.

[9] Q. Zhou and W. Sun, *A nonmonotone second-order steplength method for unconstrained minimization*, Journal of Computational Mathematics, vol. 25, no. 1, ICM-SEC, 2007, pp. 104–112, available at `http://www.global-sci.org/jcm/volumes/v25n1/pdf/251-104.pdf`.

[10] M. A. Diniz-Erhardt, J. M. Martínez, and Raydan M., *A derivative-free nonmonotone line search technique for unconstrained optimization* (2007), available at `http://kuainasi.ciens.ucv.ve/ccct/download_papers/RT-2006-10.pdf`.

[11] P. L. Toint, *A non-monotone trust-region algorithm for nonlinear optimization subject to convex constraints*, Mathematical Programming, vol. 77, no. 1, Springer Berlin/Heidelberg, 1 April 1997, pp. 69–94, available at `http://perso.fundp.ac.be/~phtoint/pubs/TR94-26.ps`.

[12] F. A. M. Gomes, M. C. Maciel, and J. M. Martínez, *Nonlinear programming algorithms using trust regions and augmented Lagrangians with nonmonotone penalty parameters*, Mathematical Programming, vol. 84, no. 1, Springer Berlin/Heidelberg, January 1999, pp. 161–200, available at `http://www.springerlink.com/content/9g0mj64au5vwcnfm/?p=1852bf5af03d4e4998b0f3aef039b71f&pi=8`.

[13] Z. W. Chen, J. Y. Han, and D. C. Xu, *A Nonmonotone Trust Region Method for Nonlinear Programming with Simple Bound Constraints*, Applied Mathematics and Optimization, vol. 43, no. 1, Springer New York, Januar 2001, pp. 63–85, available at `http://www.springerlink.com/content/35t62g89dr17cnuw/?p=1852bf5af03d4e4998b0f3aef039b71f&pi=7`.

[14] M. Ulbrich and S. Ulbrich, *Non-monotone trust region methods for nonlinear equality constrained optimization without a penalty function*, Mathematical Programming, vol. 95, no. 1, Springer Berlin/Heidelberg, January 2003, available at `http://www.springerlink.com/content/37qupmt0fbu17en0/?p=1852bf5af03d4e4998b0f3aef039b71f&pi=6`.

[15] D. C. Xu, J. Y. Han, and Z. W. Chen, *Nonmonotone Trust-Region Method for Nonlinear Programming with General Constraints and Simple Bounds*, Journal of Optimization Theory and Applications, vol. 122, no. 1, Springer Netherlands, July 2004, pp. 185–206, available at `http://www.springerlink.com/content/n9p33w30370341u1/?p=1852bf5af03d4e4998b0f3aef039b71f&pi=4`.

[16] Z. Chen and X. Zhang, *A nonmonotone trust-region algorithm with nonmonotone penalty parameters for constrained optimization*, Journal of Computational and Applied Mathematics, vol. 172, no. 1, North-Holland Publishing Company, 1 November 2004, pp. 7-39, available at `http://www.sciencedirect.com`.