QUASI-NEWTON METHODS FOR CONSTRAINED NONLINEAR SYSTEMS: COMPLEXITY ANALYSIS AND APPLICATIONS*

LEOPOLDO MARINI, BENEDETTA MORINI, MARGHERITA PORCELLI †

Abstract. We address the solution of constrained nonlinear systems by new linesearch Quasi-Newton methods. These methods are based on a proper use of the projection map onto the convex constraint set and on a derivative-free and nonmonotone linesearch strategy. The convergence properties of the proposed methods are presented along with a worst-case iteration complexity bound. Several implementations of the proposed scheme are discussed and validated on bound-constrained problems including gas distribution network models. The results reported show that the new methods are very efficient and competitive with an existing affine-scaling procedure.

Keywords: nonlinear systems of equations; Quasi-Newton methods; nonmonotone derivative-free linesearch; convergence theory; complexity analysis

1. Introduction.

In this work, we propose nonmonotone Quasi-Newton methods for solving the constrained nonlinear system of equations

$$F(x) = 0, \quad x \in \Omega. \tag{1.1}$$

The map $F : X \to \mathbb{R}^n$ is continuously differentiable. The set $X \subseteq \mathbb{R}^n$ is open, the set $\Omega \subset X$ is convex with nonempty interior and such that a projection onto Ω is a relatively easy task to perform. Our focus is on medium and large-scale systems.

We adopt the definition of Quasi-Newton methods given in [10, §6.1] in order to indicate a procedure where, letting x_k be the current iterate, the step is formed using the linear equation

$$B_k s = -F(x_k),$$

and B_k is either the Jacobian F' of F at x_k or an approximation to it. Our interest in Quasi-Newton procedures relies on the variety of possible implementations from the literature. Alternatively to $F'(x_k)$, the iteration matrix B_k can be formed via least-change secant update strategies and may not involve derivatives at all, see e.g., [10,19,22,25]. In general, the computational cost for building B_k is considerably smaller than the cost for computing $F'(x_k)$; moreover, the overhead for solving the linear system is reduced when building the inverse B_k^{-1} is possible [26].

A large number of numerical methods designed for (1.1) have been proposed. Several existing procedures belong to the class of the affine-scaling methods [8], see e.g, [1–3, 16, 24, 30, 33, 34]. Newton's method is the core of the affine-scaling procedures and a monotone decrease of the merit function

$$f(x) = \frac{1}{2}\|F(x)\|_2^2, \tag{1.2}$$

is typically enforced. The knowledge of the gradient of f, or an approximation of it, is required to perform the affine-scaling procedure which is specifically designed for bound constraints. On the other hand, we are aware of a few proposals [14, 18, 28] in the field of Quasi-Newton methods, and of a few methods [13, 14, 17, 18, 28] which do not belong to the affine-scaling framework. The latter mentioned papers consider a general convex set Ω, and the projection onto Ω is required in [17, 18, 28].

In the context of Quasi-Newton methods for (1.1), it is appropriate to apply a globalization strategy that does not either require the gradient of f or impose a sufficient decrease on the value of f [15, 18, 19, 21, 28]. In fact, the computation of the gradient of f is as costly as the computation of F' and a Quasi-Newton step may not be a direction of decrease for f. In the recent paper [28] we proposed a Quasi-Newton scheme embedded into a nonmonotone derivative-free linesearch; much emphasis was given to the algorithmic version where search directions are the

*Work supported by *Gruppo Nazionale per il Calcolo Scientifico* (GNCS-INdAM).

†Dipartimento di Ingegneria Industriale, Università di Firenze, viale Morgagni 40, via di S. Marta 3, 50134 Firenze, Italia, e-mail: leopoldo.marini@unifi.it, benedetta.morini@unifi.it, margherita.porcelli@unifi.it

residuals $\pm F$ at the current iterations. In this work, we design and analyze a new Quasi-Newton framework which differs from the scheme in [28] in the following two respects: the definition of the trial step, and the control on the norm of $\|F\|$ along the iterations. The new method shares basic convergence properties with the method in [28] but, due to the new linesearch strategy, is characterized by a complexity bound on how many iterations are needed, in the most adverse occurrence, either to drive $\|F\|$ below a specified norm or to stop as the linesearch steplength falls below a specified threshold. We are not aware of further complexity results in the literature concerning nonmonotone linesearch methods suited for (1.1). The projection map onto the set Ω is used in order to generate feasible iterates.

Our theoretical study is accompanied by an extensive numerical experience showing that our Quasi-Newton procedures are reliable and very competitive with the affine-scaling approach. We used both test problems from the literature and an interesting set of bound-constrained problems which model the final step of natural gas deliver to customers. Specifically, we considered the distribution from local companies to small customers such as households, small and medium-sized businesses. Local distribution companies deliver natural gas through small diameter pipes that connect wide urban areas, and the analysis and design of these networks is important for the fulfillment of strict requirements imposed by national Regulatory Authority, such as the Italian Regulatory Authority for Electricity Gas and Water (AEEGSI); e.g., the relative pressure of the delivered gas affects the flame stability. Modeling gas transport phenomena within distribution systems gives rise to sequences of large sparse nonlinear systems used here as test problems.

In the following section we introduce our Quasi-Newton method and in Section 3 we analyze its theoretical properties, including iteration complexity. In Section 4 we consider various implementations based on sparse Broyden and secant updates from the literature, introduce the problem set, and present the results of the computational experience conducted.

1.1. Notations.

The Euclidean norm is indicated as $\|\cdot\|$. The ith component of a vector x is indicated by $(x)_i$, the entry (i,j) of a matrix A is denoted as $(A)_{ij}$. Given a sequence of vectors $\{x_k\}$, we let $F_k = F(x_k)$, $F'_k = F'(x_k)$. The orthogonal projection map onto Ω is indicated with $P(\cdot)$.

2. The new algorithm.

Given an arbitrary initial guess $x_0 \in \Omega$, our procedure builds a sequence of feasible iterates $\{x_k\}$ satisfying the approximate norm descent condition [21]

$$\|F_{k+1}\| \leq (1+\eta_k)\|F_k\|. \tag{2.1}$$

In the above inequality, η_k is positive and consequently $\{\|F_k\|\}$ is not monotonically decreasing. The sequence $\{\eta_k\}$ meets the following requirement.

Assumption 2.1. *The positive sequence $\{\eta_k\}$ satisfies*

$$\sum_{k=0}^{\infty} \eta_k \leq \eta < \infty. \tag{2.2}$$

At k-th iteration, given x_k and an invertible matrix $B_k \in \mathbb{R}^{n\times n}$, our search direction is based on the *Quasi-Newton step* p_k^{QN} that solves the system

$$B_k p_k^{\mathrm{QN}} = -F_k. \tag{2.3}$$

Various choices for B_k are possible. Matrix B_k can be chosen, at each iteration, as the Jacobian F' at x_k, or as an approximation of it computed by finite differences; otherwise, B_k can be obtained by evaluating the previous matrix periodically. Alternatively, it can be computed by the Broyden's update [10], the Broyden-Schubert update [6,32], the Bogle-Perkins update [5], the limited memory inverse column update [26], or other secant procedures such as those discussed in [22].

With p_k^{QN} at hand, we apply a linesearch strategy and search for an iterate $x_{k+1} \in \Omega$, of the form

$$x_{k+1} = x_k + p_k, \tag{2.4}$$
$$p_k = \pi(p_k^{\mathrm{QN}}, \lambda_k), \quad \lambda_k \in (0, 1], \tag{2.5}$$

with $\pi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ being a function specified below. Linesearches that are monotone and require the calculation of derivatives are inappropriate for Quasi-Newton methods, see e.g., [4, 18, 21]. Thus, the new iterate must satisfy either the sufficient decrease on $\|F\|$:

$$\|F(x_k + \pi(p_k^{\mathrm{QN}}, \lambda_k))\| \le (1 - \alpha(1 + \lambda_k))\|F_k\|, \tag{2.6}$$

with $\alpha \in (0, 1)$, or the condition:

$$(1 - \alpha\gamma\epsilon_\ell)\|F_k\| \le \|F(x_k + \pi(p_k^{\mathrm{QN}}, \lambda_k))\| \le (1 + \eta_k - \alpha\lambda_k)\|F_k\|, \tag{2.7}$$

with $\gamma, \epsilon_\ell \in (0, 1)$, η_k as in Assumption 2.1. The scalars α and ϵ_ℓ are supposed to be small; a typical choice of α for sufficient decrease of $\|F\|$ is 10^{-4}, see e.g. [10]. Clearly condition (2.7) allows either a slight decrease on $\|F\|$ or an approximate norm decrease on $\|F\|$.

Both conditions (2.6) and (2.7) are derivative-free and imply inequality (2.1). Figure 2.1 illustrates the regions of acceptance of the new iterate. A nonmonotone behaviour of $\|F\|$ is allowed, and eventually damped since η_k tends to zero by Assumption 2.1.

The controls (2.6) and (2.7) differ from the acceptance criteria used in [28] in the form of (2.7). In [28] only the approximate norm decrease is imposed i.e.,

$$\|F(x_k + \pi(p_k^{\mathrm{QN}}, \lambda_k))\| \le (1 + \eta_k - \alpha\lambda_k)\|F_k\|,$$

while here we further impose the specified lower bound $(1 - \alpha\gamma\epsilon_\ell)\|F_k\|$. By testing both (2.6) and (2.7), we exclude points where $\|F\|$ belongs to the interval $((1 - \alpha(1 + \lambda_k))\|F_k\|,\ (1 - \alpha\gamma\epsilon_\ell)\|F_k\|)$; this issue provides a characterization of λ_k, derived in the next section, which is crucial for the complexity analysis.

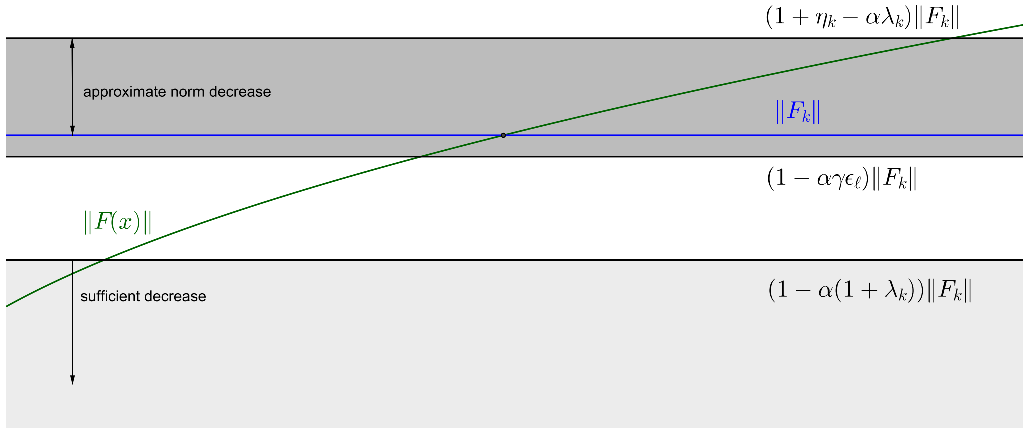
$(1 + \eta_k - \alpha\lambda_k)\|F_k\|$
approximate norm decrease
$\|F_k\|$
$(1 - \alpha\gamma\epsilon_\ell)\|F_k\|$
$\|F(x)\|$
sufficient decrease
$(1 - \alpha(1 + \lambda_k))\|F_k\|$

FIG. 2.1. *Regions of acceptance of the new iterate x_{k+1}.*

The trial steps are formed using the projection map P onto Ω. The step used in [28] lies on a projected piecewise linear path, while here we backtrack along a projected step. Specifically, for

$\lambda \in (0,1]$, we let

$$\bar{p}_k^{\rm QN} = P(x_k + p_k^{\rm QN}) - x_k, \tag{2.8}$$
$$\pi(p_k^{\rm QN}, \lambda) = \pm\lambda \bar{p}_k^{\rm QN}, \quad \text{if} \quad \|\bar{p}_k^{\rm QN}\| \neq 0, \tag{2.9}$$
$$w_k^{\rm QN} = P(x_k - p_k^{\rm QN}) - x_k, \tag{2.10}$$
$$\pi(p_k^{\rm QN}, \lambda) = \pm\lambda w_k^{\rm QN}, \quad \text{if} \quad \|\bar{p}_k^{\rm QN}\| = 0. \tag{2.11}$$

Letting ∇f be the gradient of f in (1.2), either $\lambda\bar{p}_k^{\rm QN}$ or $-\lambda\bar{p}_k^{\rm QN}$ is a descent direction for f, unless $\nabla f(x_k)^T\bar{p}_k^{\rm QN} = 0$; thus, the use of $\pm\bar{p}_k^{\rm QN}$ promotes a decrease of $\|F\|$. We observe that $x_k + \lambda\bar{p}_k^{\rm QN}$ is feasible by construction, whereas $x_k - \lambda\bar{p}_k^{\rm QN}$ may not be feasible, and a check on feasibility is necessary before attempting the use of $x_k - \lambda\bar{p}_k^{\rm QN}$. Analogous considerations hold for $\pm\lambda w_k^{\rm QN}$ in (2.10)-(2.11) and for $x_k \pm \lambda w_k^{\rm QN}$. Satisfying (2.6) promotes a decrease on $\|F\|$; hence condition (2.6) is tested on both possible steps before condition (2.7).

The complete description of our Algorithm follows.

Algorithm 1.

```
Given $x_0 \in \Omega$, $B_0 \in \mathbb{R}^{n\times n}$ invertible, $\alpha, \gamma, \epsilon_\ell, \sigma \in (0,1)$, and $\eta_k$ satisfying (2.2).
For $k = 0, 1, 2, \dots$ do
   1. Solve the linear system (2.3).
   2. Set $\bar{p}_k^{\rm QN} = P(x_k + p_k^{\rm QN}) - x_k$.
   3. If $\|\bar{p}_k^{\rm QN}\| \neq 0$, set $p_+ = \bar{p}_k^{\rm QN}$, $p_- = -\bar{p}_k^{\rm QN}$
      Else set $w_k^{\rm QN} = P(x_k - p_k^{\rm QN}) - x_k$, $p_+ = w_k^{\rm QN}$, $p_- = -w_k^{\rm QN}$.
   4. Set $\lambda = 1$.
   5. Repeat
         5.1 If $\pi(p_k^{\rm QN}, \lambda) = \lambda p_+$ satisfies (2.6), go to Step 6.
             Else if $\pi(p_k^{\rm QN}, \lambda) = \lambda p_-$ satisfies $x_k + \pi(p_k^{\rm QN}, \lambda) \in \Omega$ and (2.6),
               go to Step 6.
         5.2 If $\pi(p_k^{\rm QN}, \lambda) = \lambda p_+$ satisfies (2.7), go to Step 6.
             Else if $\pi(p_k^{\rm QN}, \lambda) = \lambda p_-$ satisfies $x_k + \pi(p_k^{\rm QN}, \lambda) \in \Omega$ and (2.7),
               go to Step 6.
         5.3 Set $\lambda = \sigma\,\lambda$.
   6. Set $\lambda_k = \lambda$, $p_k = \pi(p_k^{\rm QN}, \lambda_k)$, $x_{k+1} = x_k + p_k$.
   7. If $\|F(x_{k+1})\| = 0$ stop.
   8. Form an invertible matrix $B_{k+1} \in \mathbb{R}^{n\times n}$.
```

Acceptance of the trial steps is tested in Step 5. This repeat-loop terminates in a finite number of steps and finds the new iterate x_{k+1} satisfying either condition (2.6) or condition (2.7); the number of F-evaluations performed at each repetition of the loop in Step 5 is either 1 or 2. Condition (2.7) is ensured for λ small enough. Indeed, at kth iteration, the scalars η_k and $\alpha\gamma\epsilon_\ell$ are given and positive; consequently, $(1 - \alpha\gamma\epsilon_\ell)$ is positive and smaller than one while $(1 + \eta_k - \alpha\lambda)$ is positive and larger than one if λ is small enough. Then, from the continuity of F there exists a scalar $\bar{\lambda} > 0$ such that the components $(F(x_k))_i$ and $(F(x_k + \lambda p))_i$ of the vectors $F(x_k)$ and $F(x_k + \lambda p)$ satisfy

$$(1 - \alpha\gamma\epsilon_\ell)^2 (F(x_k))_i^2 \le (F(x_k + \lambda p))_i^2 \le (1 + \eta_k - \alpha\lambda)^2 (F(x_k))_i^2,$$

with $\lambda \in (0, \bar{\lambda})$ and $i = 1, \dots, n$.

3. Convergence analysis.

In this section we analyze the property of Algorithm 1. First, we characterize the asymptotic behaviour of the generated sequences $\{x_k\}$, $\{\|F_k\|\}$ and $\{\lambda_k\}$. Second, we show the iteration complexity. A few results are the same as in [28], and are stated for completeness.

The first two results, Lemma 3.1 and Theorem 3.2, are a consequence of the approximate norm descent condition (2.7), and do not depend on the strategy for forming the step p_k. We begin by estimating $\|F_{k+1}\|$ and λ_k on the base of the occurrences (2.6) and (2.7).

LEMMA 3.1. *Let $\{x_k\}$ and $\{\lambda_k\}$ be generated by Algorithm 1.*

i) Let $\{k_m\}$, with $m \geq 1$ and $k_1 \geq 1$, be the indices of the iterates satisfying (2.6), *i.e.,*

$$\|F_{k_m}\| \leq (1 - \alpha(1 + \lambda_{k_m-1}))\|F_{k_m-1}\|. \tag{3.1}$$

Then,

$$\|F_{k_m}\| \leq (1-\alpha)^m e^{\eta}\|F_0\|. \tag{3.2}$$

ii) Let $j \geq 1$ be the index of an iterate satisfying (2.7), i.e.,

$$(1 - \alpha\gamma\epsilon_\ell)\|F_{j-1}\| \leq \|F_j\| \leq (1 + \eta_{j-1} - \alpha\lambda_{j-1})\|F_{j-1}\|. \tag{3.3}$$

Then,

$$\lambda_{j-1} \leq \frac{\eta_{j-1}}{\alpha} + \gamma\epsilon_\ell. \tag{3.4}$$

Proof. *i)* First we observe that both conditions (2.6) and (2.7) imply inequality (2.1), and that Assumption 2.1 implies

$$\prod_{i=0}^{k}(1 + \eta_i) \leq e^{\eta}, \quad k \geq 0, \tag{3.5}$$

see [21, Lemma 2.1].

Consider an iterate, say x_{k_m}, satisfying (2.6). Inequality (3.1) gives

$$\|F_{k_m}\| \leq (1-\alpha)\|F_{k_m-1}\|,$$

whereas by (2.1) iterates x_j with $j = k_m - 1, \ldots, k_{m-1} + 1$ satisfy

$$\|F_j\| \leq (1 + \eta_{j-1})\|F_{j-1}\|.$$

Combining the previous inequalities and repeating the argument we obtain

$$\begin{aligned}
\|F_{k_m}\| &\leq (1-\alpha)\prod_{i=k_{m-1}}^{k_m-2}(1+\eta_i)\|F_{k_{m-1}}\| \\
&\leq (1-\alpha)^2\prod_{i=k_{m-1}}^{k_m-2}(1+\eta_i)\|F_{k_{m-1}-1}\| \\
&\leq \ldots \\
&\leq (1-\alpha)^m\prod_{i=k_1}^{k_m-2}(1+\eta_i)\|F_{k_1-1}\| \\
&\leq (1-\alpha)^m\prod_{i=0}^{k_m-2}(1+\eta_i)\|F_0\| \\
&\leq (1-\alpha)^m e^{\eta}\|F_0\|,
\end{aligned}$$

where the last inequality follows from (3.5).

ii) If (3.3) holds, then

$$(1 - \alpha\gamma\epsilon_\ell) \leq (1 + \eta_{j-1} - \alpha\lambda_{j-1}),$$

and this gives (3.4). □

In the previous Lemma, Item *ii*) provides an upper bound on the value of the stepsize λ when condition (2.7) is met. Such result follows from imposing both a lower and an upper bound on $\|F_k\|$ and is crucial for deriving the iteration complexity result (see Theorem 3.5). It is interesting to observe that (3.4) is trivially satisfied as long as $\eta_{j-1} \geq \alpha$ since the upper bound is larger than 1.

If we exclude the case where Algorithm 1 has a finite termination (Step 7), then the sequences $\{\|F_k\|\}$ and $\{\lambda_k\|F_k\|\}$ are convergent, as shown below. The same holds for $\{x_k\}$ if we make the following assumption.

Assumption 3.1. *Matrices B_k^{-1}, $k \geq 0$, are defined and uniformly bounded with $\|B_k^{-1}\| \leq c_B$ for some positive scalar c_B.*

Theorem 3.2. *Let Assumption 2.1 hold, and let $\{x_k\}$, $\{\|F_k\|\}$ and $\{\lambda_k\}$ be generated by Algorithm 1. Then*
i) The sequence $\{\|F_k\|\}$ is convergent.
ii) The sequence $\{\lambda_k\|F_k\|\}$ is convergent and such that

$$\lim_{k\to\infty} \lambda_k\|F_k\| = 0. \tag{3.6}$$

iii) If (2.6) is satisfied for infinitely many k, then $\lim_{k\to\infty}\|F_k\| = 0$. If (2.7) is satisfied for infinitely many k, then $\limsup_{k\to\infty}\lambda_k \leq \gamma\epsilon_\ell$. Finally, if $\|F_k\| \leq \|F_{k+1}\|$ for all k sufficiently large, $\lim_{k\to\infty}\lambda_k = 0$ and $\lim_{k\to\infty}\|F_k\| \neq 0$.
iv) If in addition Assumption 3.1 holds, then the sequence $\{x_k\}$ is convergent.

Proof. *i*) and *ii*) We refer to [28, Theorem 4.2] for the proof.

iii) If the norm decrease (2.6) holds for infinitely many k, there exists a subsequence k_m of iterates such that (3.2) holds. Consequently, $\lim_{k_m\to\infty}\|F_{k_m}\| = 0$, and the convergence of $\{\|F_k\|\}$ implies $\lim_{k\to\infty}\|F_k\| = 0$.

If (2.7) is satisfied infinitely many times, bound (3.4) and $\lim_{j\to\infty}\eta_j = 0$ give $\limsup_{k\to\infty}\lambda_k \leq \gamma\epsilon_\ell$.

If $\|F_k\| \leq \|F_{k+1}\|$ holds for all k sufficiently large, it follows that $\{\|F_k\|\}$ is not convergent to zero while (3.6) yields $\lim_{k\to\infty}\lambda_k = 0$.

iv) We refer to [28, Theorem 4.3] for the proof. □

The remaining two results detect further occurrences where $\{\|F_k\|\}$ converges to zero, namely Algorithm 1 is successful in solving (1.1). They rely on the form (2.8)–(2.11) of the step $p_k = \pi(p_k^{\mathrm{QN}}, \lambda_k)$ and require the following assumption.

Assumption 3.2. *The Jacobian $F' \in \mathbb{R}^{n\times n}$ is Lipschitz continuous on Ω and satisfies*

$$\|F'(x) - F'(y)\| \leq 2L\|x - y\|, \quad \forall x, y \in \Omega.$$

Theorem 3.3. *Let Assumptions 2.1, 3.1 and 3.2 hold, and $\{x_k\}$ be generated by Algorithm 1. If the step $\bar{p}_k^{\mathrm{QN}}$ given in (2.8) satisfies*

$$\|F_k'\bar{p}_k^{\mathrm{QN}} + F_k\| \leq \zeta_k\|F_k\|, \qquad 0 \leq \zeta_k \leq \zeta_{\max} < 1 - 4\alpha, \tag{3.7}$$

for all k sufficiently large, then $\lim_{k\to\infty}\|F_k\| = 0$.

Proof. First observe that $\|\bar{p}_k^{\mathrm{QN}}\| \leq \|p_k^{\mathrm{QN}}\|$ from the properties of the projection map, and that (3.7) implies $\|\bar{p}_k^{\mathrm{QN}}\| \neq 0$. Moreover, Assumption 3.1 gives $\|p_k^{\mathrm{QN}}\| \leq c_B\|F_k\|$. Now we show that eventually (2.6) is satisfied, and this implies our claim by Item *iii*) of Theorem 3.2.

The trial steps have the form $\pm\lambda_k \bar{p}_k^{\rm QN}$ and considering $\lambda_k \bar{p}_k^{\rm QN}$, we have

$$\begin{aligned} F(x_k + \lambda_k \bar{p}_k^{\rm QN}) &= F(x_k) + \int_0^1 F'(x_k + t\lambda_k \bar{p}_k^{\rm QN})\lambda_k \bar{p}_k^{\rm QN} dt \\ &= (1-\lambda_k)F(x_k) + \lambda_k(F'(x_k)\bar{p}_k^{\rm QN} + F(x_k)) + \\ &\int_0^1 (F'(x_k + t\lambda_k \bar{p}_k^{\rm QN}) - F'(x_k))\lambda_k \bar{p}_k^{\rm QN} dt, \end{aligned}$$

see [10, Lemma 4.1.9]. Using $\lambda_k \in (0,1]$, and Assumption 3.2, we obtain

$$\begin{aligned} \|F(x_k + \lambda_k \bar{p}_k^{\rm QN})\| &\le (1-\lambda_k)\|F_k\| + \lambda_k \zeta_k \|F_k\| + L\lambda_k^2 \|\bar{p}_k^{\rm QN}\|^2 \\ &\le (1-\lambda_k + \lambda_k \zeta_{\max})\|F_k\| + Lc_B^2 \lambda_k^2 \|F_k\|^2 \\ &\le (1-\lambda_k + \lambda_k \zeta_{\max})\|F_k\| + Lc_B^2 \lambda_k \|F_k\|^2 \\ &= (1-\lambda_k + \lambda_k \zeta_{\max} + Lc_B^2 \lambda_k \|F_k\|)\|F_k\|. \end{aligned}$$

Now, (3.6) implies $Lc_B^2 \lambda_k \|F_k\| < \alpha$ for k sufficiently large, and the acceptance condition (2.6) is satisfied if

$$1-\lambda_k + \lambda_k \zeta_{\max} + \alpha \le 1 - 2\alpha,$$

which is equivalent to

$$\lambda_k \ge \underline{\lambda} \overset{\rm def}{=} \frac{3\alpha}{1-\zeta_{\max}} \quad \text{and} \quad \underline{\lambda} \in \left[3\alpha, \frac{3}{4}\right],$$

with the form of the interval for $\underline{\lambda}$ following by (3.7). Therefore, eventually λ_k is uniformly bounded away from zero and (3.6) implies the claim. □

We conclude with a result that holds when B_k is sufficiently close to $F'(x_k)$ and the linesearch is performed along the (unprojected) Quasi-Newton step $p_k^{\rm QN}$.

Corollary 3.4. *Let Assumptions 2.1, 3.1 and 3.2 hold, and $\{x_k\}$ be generated by Algorithm 1. If the steps $p_k^{\rm QN}$ and $\bar{p}_k^{\rm QN}$ given in (2.3) and (2.8) satisfy $\bar{p}_k^{\rm QN} = p_k^{\rm QN}$ for all k sufficiently large, and*

$$\|I - F_k' B_k^{-1}\| \le \zeta_k, \quad \zeta_k \le \zeta_{\max} < 1 - 4\alpha, \tag{3.8}$$

then $\lim_{k\to\infty} \|F_k\| = 0$.

Proof. From (2.3) we get

$$\|F_k' p_k^{\rm QN} + F_k\| = \| - F_k' B_k^{-1} F_k + F_k\| = \|(I - F_k' B_k^{-1})F_k\| \le \|I - F_k' B_k^{-1}\|\|F_k\|.$$

Then, using (3.8) we can apply Theorem 3.3. □

Remark. The results stated in Theorems 3.3 and 3.4 show similarities with [4, Theorem 1] and [4, Theorem 2] which are proved for the unconstrained case.

3.1. Iteration complexity. We count how many iterations k are needed, in the worst case, either to drive $\|F_k\|$ below a specified norm or to stop as the linesearch steplength λ_k falls below a specified threshold.

In the spirit of solving (1.1), iteration k can be viewed as *successful* if x_k satisfies (2.6), and as *unsuccessful* if x_k satisfies (2.7). We carry out the iteration complexity analysis taking into account both the asymptotic behaviour of $\|F_k\|$ and λ_k shown in Theorem 3.2 and the bounds on $\|F_k\|$ and λ_k given in Lemma 3.1. Indeed, by (3.6) and Item *iii*) of Theorem 3.2, at least one of the following cases occurs

$$\lim_{k\to\infty} \|F_k\| = 0, \tag{3.9}$$

$$\limsup_{k\to\infty} \lambda_k \le \gamma\epsilon_\ell, \tag{3.10}$$

$$\lim_{k\to\infty} \lambda_k = 0. \tag{3.11}$$

Thus, it is appropriate to estimate the number of iterations performed to reach either

$$\|F_k\| \le \epsilon_F, \tag{3.12}$$

or

$$\lambda_{k-1} \le \epsilon_\ell, \tag{3.13}$$

where ϵ_ℓ is the same scalar as in (2.7), and $\epsilon_F, \epsilon_\ell \in (0,1)$ are small. Before reaching one of such conditions, the iterations of Algorithm 1 can be split into two sets:

$$\mathcal{S} = \{k : k \text{ successful and } \ \|F_k\| > \epsilon_F,\ \lambda_{k-1} > \epsilon_\ell\}, \tag{3.14}$$

$$\mathcal{U} = \{k : k \text{ unsuccessful and } \ \|F_k\| > \epsilon_F,\ \lambda_{k-1} > \epsilon_\ell\}. \tag{3.15}$$

Given $\|F_k\|$, the areas containing the values $\|F_{k+1}\|$ are illustrated in Figure 2.1. Unsuccessful iterates are such that $\|F_{k+1}\|$ belongs to the dark gray area, while at successful iterates $\|F_{k+1}\|$ lies in the light gray area.

In the following, we exclude the trivial case $\|F_0\| \le \epsilon_F$.

Theorem 3.5. *Let $\mathcal{S}$ and $\mathcal{U}$ be the sets in (3.14) and (3.15) respectively, and suppose that $\|F_0\| > \epsilon_F$. Let*

$$k_\dagger \overset{\text{def}}{=} \left\lceil \log_{1-\alpha} \ \frac{\epsilon_F}{e^\eta \|F_0\|} \right\rceil, \tag{3.16}$$

and k_* be the first index such that

$$\eta_{k-1} \le \alpha(1-\gamma)\epsilon_\ell,\ \forall k \ge k_*. \tag{3.17}$$

Then Algorithm 1 takes at most $k_\dagger + k_*$ iterations to terminate either with (3.12) or (3.13).

Proof. Letting $k_\dagger$ be as in (3.16), it holds

$$(1-\alpha)^{k_\dagger} e^\eta \|F_0\| \le \epsilon_F.$$

Using (3.2) we conclude that $|\mathcal{S}| < k_\dagger$, i.e. Algorithm 1 takes at most $k_\dagger$ successful iterations to generate a point x_k such that (3.12) is satisfied. Note that since $\|F_0\| > \epsilon_F$, then $\dfrac{\epsilon_F}{e^\eta \|F_0\|} < 1$ and $k_\dagger$ is well-defined.

If index k is unsuccessful, then λ_{k-1} satisfies (3.4). Letting k_* be as in (3.17) and $k \ge k_*$, it follows $\lambda_k \le \epsilon_\ell$; thus we can conclude that $k \notin \mathcal{U}$ and $|\mathcal{U}| < k_*$, i.e. Algorithm 1 takes at most k_* unsuccessful iterations to generate a point x_k such that (3.13) is satisfied. Note that index k_* is well-defined due to Assumption 2.2.

Combining the two bounds we deduce the worst-case iteration complexity bound $k_\dagger + k_*$.□

Example. Let $\alpha = 10^{-4}$, $\epsilon_F = 10^{-6}$, $\epsilon_\ell = 10^{-6}$, $\gamma = 1/2$, $\|F_0\| = 1$, $\eta_k = 1/(1+k)^2$, $k \ge 0$. Then $\eta = \sum_{k=1}^{\infty} \eta_k = \pi^2/6$ and it follows $k_\dagger = 154597$, $k_* = 70711$.

We conclude this section observing that (3.12) and (3.13) suggest termination criteria for the implementation of Algorithm 1 which will be used in our implementation, see Section 4.1.

4. Computational experience. In this section we use `Matlab` and apply Algorithm 1 in the solution of two sets of problems. The first set consists of sequences of large and sparse bound-constrained nonlinear systems modeling gas distribution networks; the second set consists of medium and large bound-constrained systems taken form the literature with different Jacobian structure. We first discuss some implementation issues and describe five implementations of our Quasi-Newton method which differ in the way matrices B_k's are built. Then, we present the performance of Algorithm 1 on the two problem sets and make a comparison with a monotone trust-region affine-scaling procedure. A sparse large-scale implementation of TRESNEI, described in the next subsection, is the competitor affine-scaling solver considered. TRESNEI is the Matlab implementation of a trust-region algorithm and represents the state-of-art of affine-scaling methods for systems of nonlinear equalities and inequalities [27]. TRESNEI is freely available[1] and compares favourably with the `Matlab` function `lsqnonlin` as shown in [27].

4.1. Implementation details. We conducted our numerical experience using several implementations of Algorithm 1. First, we attempted to solve the gas distribution problems by the derivative-free implementations given in [28]; in such implementations the Jacobian F' is not employed at any iterate. However these derivative-free implementations failed and computing F', at least occasionally, has appeared to be crucial for robustness. We now describe the five implementations of Algorithm 1 which yield the results presented in §4.2-§4.3.

The first implementation is based on the Finite-Difference Newton (FDN) method and the computation of F' is performed by using the groupwise differentiation [9]. Specifically, let $\mathcal{J}$ be the sparsity pattern of F'. Groupwise differentiation is based on a division of columns of F' into m groups $\mathcal{G}_1, \dots \mathcal{G}_m$ so that: each column belongs to only one group; if $(i, j_1), (i, j_2) \in \mathcal{J}$, and $j_1 \neq j_2$, then $j_1 \in \mathcal{G}_{f_1}$ and $j_2 \in \mathcal{G}_{f_2}$ with $f_1 \neq f_2$. Then, for each group $\mathcal{G}_f, 1 \le f \le m$, we let

$$\Sigma_f = F\left(x_k + \sum_{j \in \mathcal{G}_f} h_j\, e_j\right) - F(x), \tag{4.1}$$

$$(B_k)_{ij} = \frac{(\Sigma_f)_i}{h_j}, \quad \forall (i,j) \in \mathcal{J}, \quad j \in \mathcal{G}_f,$$

where $(B_k)_{ij}$ is the entry of B_k of position (i, j), e_j is the jth unit vector and h_j is the stepsize. We observe that m F-evaluations are required to approximate F' by finite difference.

The second implementation consists in the Modified Newton method (MON). Matrix B_k is the finite-difference approximation to F' described above when $k = 0$ and when $\mathrm{mod}(k-1, 5) = 0$, while such an approximation to the Jacobian matrix is frozen in the remaining iterations.

The third and fourth versions of Algorithm 1 are based on the Broyden-Schubert Update (BSU) [6,32] and the Bogle-Perkins Update (BPU) [5]. In these cases, the matrices B_k have the same sparsity pattern as the Jacobian matrix F' and take the form

$$B_{k+1} = B_k + \Delta_k. \tag{4.2}$$

Let $y_k = F_{k+1} - F_k$. The BS formula is given by

$$(\Delta_k)_{ij} = \delta_i (y_k - B_k p_k)_i (p_k)_j, \quad \forall (i,j) \in \mathcal{J}, \tag{4.3}$$

where $\delta_i = \dfrac{1}{\sum_{(i,\ell) \in \mathcal{J}} (p_k)_\ell^2}$ if $\sum_{(i,\ell) \in \mathcal{J}} (p_k)_\ell^2 \neq 0$, $\delta_i = 0$ otherwise. The BP update has the form

$$(\Delta_k)_{ij} = \phi_i (y_k - B_k p_k)_i (B_k)_{i,j}^2 (p_k)_j, \quad \forall (i,j) \in \mathcal{J}, \tag{4.4}$$

where $\phi_i = \dfrac{1}{\max\{\sum_{(i,\ell) \in \mathcal{J}} (p_k)_\ell^2 (B_k)_{i\ell}^2, 10^{-8}\}}$. We applied BSU and BPU updates, setting B_k equal to the finite-difference approximation to $F'(x_k)$ periodically as in MON implementation.

In the above implementations, Quasi-Newton equations (2.3) were solved using the factorization $P\, B_k\, Q = LU$ computed via the `Matlab` command `lu` and the `Matlab`'s backslash command. Column reordering (matrix Q) reduces fill-in in the sparse case and is commonly more time and memory efficient than just row pivoting via the permutation matrix P. Occasionally, ill-conditioning or singularity of sparse Broyden updates were detected by `Matlab` in the solution of the Quasi-Newton equations. In this case, we discarded the current matrix and recomputed it by replacing (4.2) with the formula $B_{k+1} = B_k + \tau \Delta_k$, $\tau \in [0, 1)$. Here τ is found by trial [21,25];

[1] TRESNEI is available at `http://tresnei.de.unifi.it/`.

in particular the value of τ was reduced progressively by using the rule $\tau = 10^{-t},\ t = 1, 2, ...$, until no warnings on ill-conditioning/singularity were given by `Matlab`.

Finally, we considered a version of Algorithm 1 based on the Inverse Column Update (ICU) [26]. A column of the inverse of the Jacobian approximation is updated at each iteration in such a way that the secant equation is satisfied. Letting B_k^{-1} be a given approximation to $F'(x_k)^{-1}$, the updating has the form

$$B_{k+1}^{-1} = B_k^{-1} + \frac{(s_k - B_k^{-1} y_k) e_{j_k}^T}{(y_k)_{j_k}},$$

with $|(y_k)_{j_k}| = \|y_k\|_\infty$; again B_k^{-1} is reset to the inverse of the finite difference approximation of $F'(x_k)$ periodically as in MON implementation. The computation of the Quasi-Newton step p_k^{QN} requires the multiplication of B_k^{-1} times F_k. Given the factorization of the finite difference approximations of $F'(x_k)$ (computed periodically) via Gaussian elimination with complete pivoting, in the successive iterations products of B_k^{-1} times a vector can be performed using $O(5n)$ storage positions and solving two linear systems which involve the available P, Q, L, U and current right-hand sides F_k and y_k.

All the tested algorithms were run using `Matlab R2015a` on a Intel(R) Core(TM) i5-6600K CPU @3.50 GHz x 4,16.0 GB RAM.

The parameters in Algorithm 1 were set as follows: $\alpha = 10^{-4}$, $\epsilon_\ell = 10^{-9}$, $\gamma = \sigma = 0.5$, $\eta_k = \dfrac{\|F_0\|^{\frac{1}{4}}}{(k+1)^2}$, $k \geq 0$. We considered the stopping criteria (3.12) and (3.13) with $\epsilon_F = 10^{-12}$ and imposed a maximum of F-evaluations (including those needed in (4.1)). The constraint set is the n-dimensional box $\Omega = \{x \in \mathbb{R}^n \quad \text{s.t.} \quad l \leq x \leq u\}$, $l,\ u \in \mathbb{R}^n$, and the projection map P onto Ω is given by $P(x) = \max\{l, \min\{x, u\}\}$.

We compared the five implementations of Algorithm 1 previously described with a sparse large-scale implementation of TRESNEI solver where the computation of the Jacobians and the solution of the linear systems are performed as in the FDN version of Algorithm 1. TRESNEI was run using the default parameters except for the initial trust-region radius that here was set equal to 10.

4.2. Systems modeling gas distribution network. A gas distribution network can be modeled by four descriptive elements: the pipe, the decompression valve, the citygate station and the distribution node. Assuming, for the sake of simplicity, that the temperature is constant, these elements are defined by two unknown characteristic parameters: the mass flow and the pressure; their physical behaviour is described by typical equations which represent a nonlinear system. Specifically, a set of equations represents linear continuity equations while others are pressure loss equations. The nonlinear Ferguson equation, taking into account friction losses by means of the Darcy-Weisbach formulation with Colebrook model for friction factor assessment, is used to describe gas motion in pipe elements [29]. Bound constraints are imposed on the characteristic parameters in order to find a solution of physical meaning, e.g., absolute pressures are positive, and to restrict the search space for the desired solution.

Determining the model for the network amounts to solving a sequence of nonlinear systems. In fact, the nonlinear system representing the model equations depends on some coefficients which are refined by an outer iterative procedure processing all the elements; the nonlinear system resulting from each outer refinement is solved by the inner Quasi-Newton procedure in order to update the characteristic parameters (mass flows and pressures). For details on the nested processes we refer to [7].

In our numerical experience we considered real gas distribution networks of three medium-size towns in Tuscany (later denoted as T1, T2 and T3). For each town we used the flow-network solver given in [7] and generated five sequences of nonlinear systems corresponding to different time slots; afterwords these sequences were solved offline by applying Algorithm 1. The five sequences for T1 are numbered as T1_1,..., T1_5, and analogously for T2 and T3. The features of our tests are summarized in Table 4.1 where we provide the dimension n of the systems, the number m

of groups in the groupwise differentiation (4.1), the density of the Jacobian, the average time in seconds for computing F and F'. The timings are useful for the analysis of the numerical results. We observe that the Jacobian matrices are very sparse. In our runs we imposed a maximum of 5000 F-evaluations (including those needed in (4.1)).

Results of our numerical experience are summarized in Table 4.2. They refer to the solution of the whole sequences by each variant of Algorithm 1. We report: the number ns of systems in each sequence, the execution time in seconds for solving a sequence (top table), the total number of Jacobian evaluations Je and F-evaluations Fe (excluding those needed to evaluate F') (bottom table). The symbol $\mathcal{F}_\ell$ means that the solver failed in solving ℓ systems of the sequence. In bold we highlight the best performing solver in terms of execution time. We point-out that the total number of systems solved is 216.

Let us first analyze the results in the top part of the table. BPU is the only implementation that solves all the sequences. FDN fails in solving one system from T3_1, but it is the fastest method on eleven sequences out of the fourteen solved. Among the methods where F' is not evaluated at each iteration (i.e. MON, BSU, BPU, ICU), BPU is the most effective. Moreover BPU compares very favourably with FDN in the solution of T2_1, T3_1 and its timing is similar to that of FDN on five problems (T1_1-T1_3, T1_5, T2_4). On the contrary, ICU is the slowest version of Algorithm 1. A summary of the above discussion is plotted in the total execution time performance profile of Figure 4.1, [12].

More insight into the behaviour of the method can be drawn from the bottom of Table 4.2. As expected, compared to FDN, the BSU and BPU procedures have provided a reduction in the number of Jacobian evaluation varying between 35% and 84%, while the number of F-evaluations performed is considerably higher than using FDN. The behaviour of BSU and BPU in terms of F-evaluations follows from the use of poorer trial directions than in FDN; with respect to FDN, this gives rise to slower convergence and a higher number of backtracks for BSU and BPU. These observations and the fact that two F-evaluations are (often) performed at each linesearch repetition support the values Fe reported in Table 4.2. Using BSU and BPU, a reduction in CPU time with respect to FDN has been observed only in few sequences since the Jacobian matrix of our test problems is very sparse and saving in F'-evaluations are offset by the large number of F-evaluations required. Specifically, Table 4.1 indicates that the ratio between the timing for computing F' and F is roughly 22, 23 and 30 for problems T1, T2, T3 respectively. Then, the values Je and Fe reported in Table 4.2 explain why BSU and BPU are slower than FDN in most cases.

MON and ICU versions of Algorithm 1 are not competitive as their convergence is very slow and there are not significant savings in Jacobian evaluations. For the sake of completeness, in Tables 4.3-4.4 we report the convergence history for each system of the sequences T2_3 and T3_4. Note that the number of iterations and the F'-evaluations coincide in FDN approach.

Finally, in Table 4.5 we report the time employed by the FDN variant and the derivative-based solver TRESNEI. TRESNEI fails in solving 6 systems while FDN only fails once. Overall, their reliability is comparable but FDN is faster on the largest problem T3 and seems to be a valid alternative to the affine-scaling approach implemented in TRESNEI.

We conclude pointing out that the computational time employed by all compared solvers is rather low despite the large dimension of the problems.

4.3. Systems from the literature with different Jacobian structure. The results presented in the previous section show that FDN is quite effective and compares favourably with the other implementations since computing F' is not expensive. Now we consider sparse and dense problems from the literature and focus on the evaluation cost of the Jacobian by the groupwise differentiation, and on its impact on the performance of the tested procedure. We underline that procedures alternative to groupwise differentiation may be used for forming the Jacobians of the problem considered, but our results aim to represent the solution of systems with similar Jacobian structure when the Jacobians are approximated by finite differences.

The twelve problems selected are extracted from [20,21,23] and were used in [19,20,28]. Table 4.6 displays the dimension n of these systems, the density of the Jacobian, the number m of groups

				Time	
Town	n	$dens(F')$	m	F	F'
T1	9948	$2\,10^{-4}$	6	$4.4\,10^{-3}$	$8.5\,10^{-2}$
T2	17622	$1\,10^{-4}$	6	$7.8\,10^{-3}$	$1.7\,10^{-1}$
T3	33531	$7\,10^{-5}$	7	$1.5\,10^{-2}$	$3.6\,10^{-1}$

Table 4.1

Dimension (n) of the problems T1, T2, T3, density $(dens(F'))$ of the Jacobian, number (m) of groups in the groupwise differentiation and average time in seconds for computing F and F'.

	Total execution time				
Sequence (ns)	FDN	MON	BSU	BPU	ICU
T1_1 (14)	**14.8**	34.4	29.9	22.7	35.6
T1_2 (13)	**15.6**	20.6	28.4	19.8	$\mathcal{F}_1$
T1_3 (12)	21.0	25.9	$\mathcal{F}_1$	**16.9**	$\mathcal{F}_1$
T1_4 (23)	**24.3**	56.4	49.2	46.1	$\mathcal{F}_1$
T1_5 (14)	28.7	28.3	**17.4**	28.2	35.8
T2_1 (17)	178.2	$\mathcal{F}_1$	$\mathcal{F}_1$	106.4	**67.0**
T2_2 (10)	**24.6**	$\mathcal{F}_1$	44.1	51.3	67.4
T2_3 (10)	**29.4**	74.5	53.8	53.0	76.9
T2_4 (16)	**33.8**	67.4	67.1	38.9	92.1
T2_5 (20)	**59.1**	97.6	81.1	88.5	111.7
T3_1 (13)	$\mathcal{F}_1$	$\mathcal{F}_1$	226.3	**225.6**	354.6
T3_2 (17)	**191.6**	$\mathcal{F}_2$	$\mathcal{F}_2$	428.8	$\mathcal{F}_2$
T3_3 (13)	**114.6**	247.7	209.8	188.4	351.3
T3_4 (12)	**111.2**	276.6	211.0	205.3	$\mathcal{F}_1$
T3_5 (12)	**85.4**	310.7	232.2	197.1	386.0

	Je/ Fe				
Sequence (ns)	FDN	MON	BSU	BPU	ICU
T1_1 (14)	144/364	127/5276	93/3267	87/2509	119/5136
T1_2 (13)	150/398	92/2884	89/3058	79/2132	$\mathcal{F}_1$
T1_3 (12)	161/1259	109/3877	$\mathcal{F}_1$	77/1660	$\mathcal{F}_1$
T1_4 (23)	218/551	224/7611	139/4382	164/4365	$\mathcal{F}_1$
T1_5 (14)	233/1847	120/4169	63/1554	119/3249	125/5273
T2_1 (17)	732/1970	$\mathcal{F}_1$	110/3333	216/4465	135/4214
T2_2 (10)	119/595	$\mathcal{F}_1$	68/2593	94/3477	124/5886
T2_3 (10)	138/813	131/6853	80/3290	93/3685	134/6867
T2_4 (16)	159/483	125/5314	95/2983	75/1679	153/7357
T2_5 (20)	274/735	195/7429	113/3762	149/4415	204/8421
T3_1 (13)	$\mathcal{F}_1$	$\mathcal{F}_1$	149/6029	180/7548	270/15808
T3_2 (17)	370/1972	$\mathcal{F}_2$	$\mathcal{F}_2$	263/12405	$\mathcal{F}_2$
T3_3 (13)	238/1038	225/10363	134/4917	149/5180	272/14630
T3_4 (12)	244/1170	236/12039	138/5831	158/6636	$\mathcal{F}_1$
T3_5 (12)	192/523	250/12849	143/5632	154/5555	290/16125

Table 4.2

Solution of the sequences of nonlinear systems: total execution time in seconds (top); number Je *of Jacobian evaluations and number* Fe *of F-evaluation (bottom).*

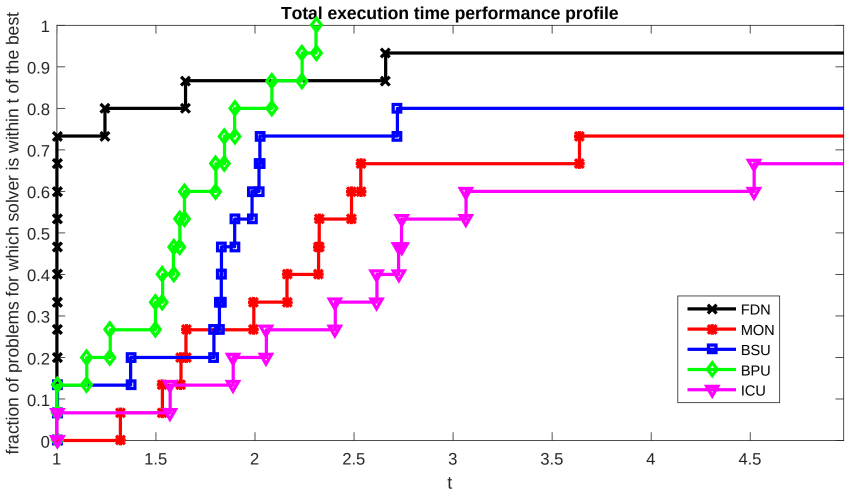

FIG. 4.1. *Total execution time performance profiles of Algorithm 1 with different implementations on problems T1÷T3.*

	It/Fe/Time	It(Je)/Fe/Time			
system	FDN	MON	BSU	BPU	ICU
1	18/71/4.5	67(14)/671/8.4	76(16)/721/14.2	73(15)/757/11.8	151(31)/2151/24.5
2	18/91/4.7	232(47)/3751/39.7	76(16)/783/14.9	71(15)/622/10.7	86(18)/899/11.7
3	18/99/4.7	41(9)/275/4.2	45(9)/298/7.2	66(14)/533/9.6	61(13)/578/7.8
4	19/120/5.1	51(11)/348/5.2	46(10)/341/8.2	35(7)/222/4.5	64(13)/756/9.2
5	16/74/4.1	27(6)/155/2.6	39(8)/431/7.7	36(8)/325/5.5	72(15)/738/9.7
6	16/146/4.6	22(5)/183/2.5	40(8)/435/7.9	33(7)/227/4.5	22(5)/160/2.5
7	10/67/2.7	56(12)/417/6.0	11(3)/64/1.8	34(7)/274/4.9	47(10)/369/5.4
8	10/74/2.7	48(10)/424/5.6	11(3)/63/1.8	26(6)/182/3.6	54(11)/507/6.7
9	9/63/2.4	48(10)/468/5.9	16(4)/94/2.7	41(9)/417/6.7	60(12)/555/7.4
10	4/8/1.0	31(7)/161/2.9	15(3)/60/2.0	24(5)/126/2.9	29(6)/154/2.7

TABLE 4.3
Statistics for the systems of the sequence T2_3: number It *of iterations, number* Je *of Jacobian evaluations, number* Fe *of F-evaluation, execution time* Time *in seconds.*

in the groupwise differentiation, the average time for computing F and F'. Six problems (P3 ÷ P6, P10, P11) have sparse Jacobian. In particular the Jacobian of the function [23, Problem 84] has variable bandwidth $b_l + b_u + 1$, b_l, $b_u > 0$ being the lower and upper bandwidth, and P3÷P6 are the variable banded problems obtained setting $b_l = 5, 15, 25, 35$ and $b_u = 1, 10, 20, 30$ respectively. The remaining six problems have dense Jacobian. The set Ω is a box with $l = (-100, \dots, -100)^T$, $u = (100, \dots, 100)^T$ for problems P1÷P8; $l = (5, \dots, 5)^T$, $u = (15, \dots, 15)^T$ for problem P9; $l = (-10, \dots, -10)^T$, $u = (0, \dots, 0)^T$ for problem P10; $l = (0, \dots, 0)^T$, $u = (10, \dots, 10)^T$ for problem P11 and $l = (0, \dots, 0)^T$, $u = (\infty, \dots, \infty)^T$ for problem P12. The initial guesses used are $x_0^{(1)} = (-1, \dots, -1)^T$, $x_0^{(2)} = (-50, \dots, -50)^T$ for problems P1÷P8; $x_0^{(\gamma)} = l + \gamma(u - l)/4$ with $\gamma = 1, 2$ for problems P9÷P11; $x_0^{(\gamma+1)} = \gamma 10^\gamma$ with $\gamma = 0, 1$ for problem P12. Due to the high value of m in most of the problems, in these tests we increased the maximum number of F-evaluations up to $10^3\, n$ (including those needed in (4.1)).

Analogously to the previous section, we report the total execution time in seconds in Table 4.7 and the number It of iterations, the number Je of Jacobian evaluations, the number Fe of F-evaluation in Tables 4.8, required by the variants of Algorithm 1 with the starting guess x_0^1 or x_0^2. In Table 4.9 we compare the results obtained with FDN and TRESNEI. Results for runs

	It/Fe/Time	It(Je)/Fe/Time			
system	FDN	MON	BSU	BPU	ICU
1	63/521/40.7	191(39)/1948/51.9	136(28)/1195/51.8	149(30)/1485/49.9	222(45)/2320/58.2
2	43/326/27.3	291(59)/3417/85.2	76(16)/572/27.6	123(25)/1164/39.9	190(38)/2057/50.4
3	18/26/9.7	161(33)/1918/47.3	181(37)/2312/80.5	101(21)/894/31.7	$\mathcal{F}$
4	14/20/7.5	87(18)/924/24.1	71(15)/625/26.7	57(12)/439/16.9	141(29)/1857/42.5
5	16/24/8.7	66(14)/622/17.1	51(11)/401/18.0	61(13)/646/20.8	77(16)/769/19.7
6	31/175/18.9	133(27)/1531/38.2	26(6)/159/8.4	61(13)/464/18.6	85(17)/1189/26.4
7	18/28/10.1	75(15)/632/17.9	28(6)/114/8.2	50(10)/303/13.3	36(8)/246/7.9
8	13/17/7.1	62(13)/552/15.5	31(7)/195/10.6	86(18)/855/28.4	41(9)/369/10.2
9	13/15/7.1	42(9)/383/10.7	21(5)/206/8.4	35(7)/271/10.2	66(14)/709/18.3
10	7/8/3.7	21(5)/93/4.1	11(3)/38/3.2	21(5)/96/5.4	19(4)/73/3.2
11	4/5/2.1	10(2)/11/1.2	6(2)/7/1.7	10(2)/11/1.9	8(2)/11/1.2
12	4/5/2.1	7(2)/8/1.1	6(2)/7/1.6	7(2)/8/1.6	6(2)/7/1.1

Table 4.4

Statistics for the systems of the sequence T3_4: number It *of iterations, number* Je *of Jacobian evaluations, number* Fe *of F-evaluation, execution time* Time *in seconds.*

	Total execution time	
Sequence (ns)	FDN	TRESNEI
T1_1 (14)	**14.8**	19.8
T1_2 (13)	**15.6**	17.8
T1_3 (12)	21.0	**13.3**
T1_4 (23)	**24.3**	32.1
T1_5 (14)	28.7	**16.6**
T2_1 (17)	178.2	$\mathcal{F}_3$
T2_2 (10)	24.6	**22.6**
T2_3 (10)	29.4	**23.9**
T2_4 (16)	**33.8**	39.4
T2_5 (20)	**59.1**	$\mathcal{F}_3$
T3_1 (13)	$\mathcal{F}_1$	**161.4**
T3_2 (17)	**191.6**	379.4
T3_3 (13)	**114.6**	155.4
T3_4 (12)	**111.2**	142.0
T3_5 (12)	**85.4**	181.4

Table 4.5

Total execution time in seconds for Algorithm 1 with FDN *and* TRESNEI.

with problem P9 are not reported in the tables since all the tested methods converge to a local minimum where the norm of F is of order 10^{-11}.

Tables 4.7-4.8 shows that our implementations solve the tests, except for BPU which fails in solving three systems and for ICU which fails in solving one system. In all the runs the MON, BSU, BPU and ICU variants yield a benefit in terms of computational time, and savings increase along with the cost associated to the Jacobian evaluations, since the number of Jacobian evaluations is larger in FDN than in the other variants of Algorithm 1. Clearly, Quasi-Newton procedures are competitive with FDN as long as the overall number of F-evaluation is not burdensome due to slow convergence rate and/or to globalization. In particular, we observe that the gain in computational time is marginal for problem P10 and P11, while grows with the bandwidth for

				Time		
Problem	n	$dens(F')$	m	F	F'	Source
P1	1000	1	1000	$3.7\,10^{-2}$	$6.0\,10^{0}$	[23, Problem 74]
P2	2000	1	2000	$5.0\,10^{-4}$	$2.8\,10^{1}$	[23, Problem 77]
P3	20000	$3\,10^{-4}$	7	$8.6\,10^{-2}$	$9.8\,10^{-1}$	[23, Problem 84]
P4	20000	$1\,10^{-3}$	26	$8.7\,10^{-2}$	$3.4\,10^{0}$	[23, Problem 84]
P5	20000	$2\,10^{-3}$	46	$8.8\,10^{-2}$	$6.1\,10^{0}$	[23, Problem 84]
P6	20000	$3\,10^{-3}$	66	$8.9\,10^{-2}$	$9.0\,10^{0}$	[23, Problem 84]
P7	1000	1	1000	$4.6\,10^{-2}$	$1.8\,10^{1}$	[23, Problem 86]
P8	1000	1	1000	$1.5\,10^{-2}$	$2.4\,10^{1}$	[23, Problem 88]
P9	2000	1	2000	$1.1\,10^{-3}$	$1.5\,10^{0}$	[20, Problem 8]
P10	20000	$3\,10^{-4}$	8	$1.0\,10^{-2}$	$5.4\,10^{-2}$	[20, Problem 15]
P11	2000	$1\,10^{-4}$	2000	$1.5\,10^{-3}$	$1.0\,10^{-1}$	[20, Problem 19]
P12	1000	1	1000	$1.8\,10^{-2}$	$5.5\,10^{0}$	[21, Problem 6]

TABLE 4.6
Dimension (n) of the problems P1÷P12, density ($dens(F')$) of the Jacobian, number (m) of groups in the groupwise differentiation, average time in seconds for computing F and F'.

problems P3÷P6, ranging from the 50% to the 70%. The CPU-time performance profile plotted in Figure 4.2 summarizes the comparison among the Quasi-Newton procedures. We remark that the ICU implementation is the fastest version of Algorithm 1 but the BSU implementation performs similarly and it is within a factor 2 of the best performing solver in the 100% of the runs.

Finally, Table 4.9 indicates that FDN is considerably faster than TRESNEI in 19 problems out of 22.

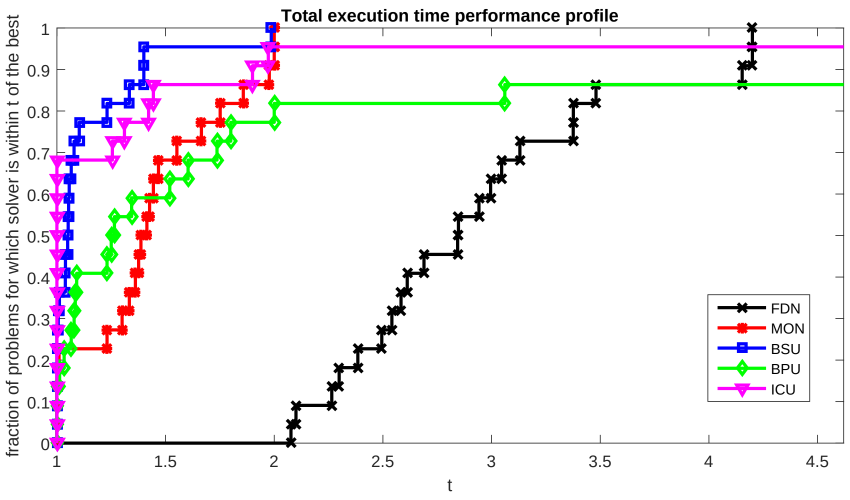

FIG. 4.2. *Total execution time performance profiles of Algorithm 1 with different implementations on problems P1÷P8 and P10÷P12.*

4.4. Final remarks. We can draw some conclusions on the features of Algorithm 1 and its implementations on the selected set of test problems.

The performance of Algorithm 1 depends on two major issues: the choice of the iteration matrices B_k, and the choice of the sequence $\{\eta_k\}$. Numerical studies on Quasi-Newton methods showed that they have the potential to improve the performance of Newton's method, but the amount of improvement depends on the Jacobian structure and the possible savings in linear algebra [25]. Note that ICU and MON are the two implementations avoiding factorizations of the

		Total execution time				
Problem	x_0	FDN	MON	BSU	BPU	ICU
P1	$x_0^{(1)}$	36.4	18.1	**12.8**	**12.8**	18.2
	$x_0^{(2)}$	60.1	29.8	**19.2**	25.8	24.1
P2	$x_0^{(1)}$	170.0	86.2	**59.7**	90.7	86.2
	$x_0^{(2)}$	410.0	206.3	127.1	188.9	**117.8**
P3	$x_0^{(1)}$	6.3	3.9	**3.0**	3.1	5.7
	$x_0^{(2)}$	15.4	9.7	7.5	8.6	**6.8**
P4	$x_0^{(1)}$	17.7	11.3	8.1	8.4	**7.7**
	$x_0^{(2)}$	49.3	26.9	20.5	$\mathfrak{F}$	**19.4**
P5	$x_0^{(1)}$	31.0	**13.0**	13.7	14.1	**13.0**
	$x_0^{(2)}$	87.5	45.6	34.8	$\mathfrak{F}$	**33.5**
P6	$x_0^{(1)}$	68.2	20.4	21.2	21.8	**20.2**
	$x_0^{(2)}$	136.6	69.9	52.8	$\mathfrak{F}$	**50.8**
P7	$x_0^{(1)}$	52.7	**17.3**	17.5	**17.3**	**17.3**
	$x_0^{(2)}$	65.9	43.5	43.7	**22.0**	43.4
P8	$x_0^{(1)}$	120.5	48.6	48.6	48.9	**48.3**
	$x_0^{(2)}$	217.7	**74.0**	**74.0**	226.5	97.0
P10	$x_0^{(1)}$	3.1	1.6	1.6	1.5	**1.2**
	$x_0^{(2)}$	2.7	1.6	1.6	1.6	**1.3**
P11	$x_0^{(1)}$	2.1	1.0	0.7	0.9	**0.5**
	$x_0^{(2)}$	2.1	1.0	0.7	1.0	**0.5**
P12	$x_0^{(1)}$	66.5	32.8	21.0	21.0	**19.7**
	$x_0^{(2)}$	116.7	52.2	**28.1**	48.8	$\mathfrak{F}$

Table 4.7
Solution of problems P1÷P8 and P10÷P12 for each starting guess x_0: total execution time in seconds.

B_k's. The numerical experience reported here confirms that our Quasi-Newton implementations, especially BSU, BPU and ICU, compares favourably with Newton's method when the evaluation and/or the factorization of the Jacobian is costly with respect to the Quasi-Newton updates and the associated linear algebra. Specifically, BSU, BPU and ICU are not competitive with Newton's method on the very sparse systems modeling gas distribution networks, while they are convenient on systems with denser Jacobians.

We remark that very cheap implementations with B_k multiple of the identity have been analyzed in [28], especially for strongly diagonal dominant matrices and matrices with positive (negative) definite symmetric part, but they are not suitable for solving the problems from gas distribution networks.

Finally, the sequence $\{\eta_k\}$ influences the performance of Algorithm 1 and some proposal for its choice were made in the field of approximate norm descent methods [18, 19, 21, 28]. The choice of such sequence may deserve further study, e.g., using sophisticated tuning methodologies as those discussed and implemented in [31].

5. Conclusions.

We have proposed a new algorithm for the solution of constrained nonlinear systems. It embeds a Quasi-Newton step into a new nonmonotone linesearch strategy that rules the step acceptance. The main properties of the method are the flexibility in the computation of the trial step and a derivative-free linesearch that allows an approximate norm decrease of the norm of the residual function. We have provided theoretical conditions for the convergence of the residual function to zero and a worst-case iteration complexity result. In order to show the practical

	x_0	It/Fe	It(Je)/Fe			
Problem	x_0	FDN	MON	BSU	BPU	ICU
P1	$x_0^{(1)}$	6/7	12(3)/13	9(2)/10	9(2)/12	11(3)/12
	$x_0^{(2)}$	10/11	21(5)/22	14(3)/15	17(4)/18	16(4)/17
P2	$x_0^{(1)}$	6/8	13(3)/17	6(2)/8	11(3)/26	11(3)/16
	$x_0^{(2)}$	14/15	31(7)/32	18(4)/19	27(6)/70	18(4)/27
P3	$x_0^{(1)}$	6/7	11(3)/12	9(2)/10	10(2)/11	11(3)/32
	$x_0^{(2)}$	15/16	35(7)/36	22(5)/23	21(5)/37	22(5)/23
P4	$x_0^{(1)}$	5/6	11(3)/12	8(2)/9	10(2)/11	8(2)/9
	$x_0^{(2)}$	14/15	32(7)/33	21(5)/22	$\mathcal{F}$	22(5)/25
P5	$x_0^{(1)}$	5/6	9(2)/10	8(2)/9	9(2)/11	8(2)/9
	$x_0^{(2)}$	14/15	31(7)/32	21(5)/22	$\mathcal{F}$	23(5)/32
P6	$x_0^{(1)}$	7/8	9(2)/10	7(2)/8	9(2)/12	8(2)/9
	$x_0^{(2)}$	14/15	31(7)/32	21(5)/22	$\mathcal{F}$	23(5)/30
P7	$x_0^{(1)}$	3/4	4(1)/5	4(1)/5	3(1)/4	4(1)/5
	$x_0^{(2)}$	3/4	6(2)/7	6(2)/7	4(1)/5	6(2)/7
P8	$x_0^{(1)}$	5/6	8(2)/9	7(2)/8	7(2)/8	8(2)/9
	$x_0^{(2)}$	9/10	17(4)/18	14(3)/15	42(9)/242	16(4)/17
P10	$x_0^{(1)}$	9/9	16(4)/17	12(3)/13	12(3)/13	13(3)/14
	$x_0^{(2)}$	7/8	16(4)/17	11(3)/13	11(3)/12	12(3)/13
P11	$x_0^{(1)}$	27/28	61(13)/69	36(8)/37	51(11)/52	29(6)/47
	$x_0^{(2)}$	28/29	64(13)/65	38(8)/39	52(11)/53	30(6)/49
P12	$x_0^{(1)}$	10/11	21(5)/22	13(3)/14	13(3)/18	11(2)/12
	$x_0^{(2)}$	18/20	36(8)/71	20(4)/26	32(7)/68	$\mathcal{F}$

TABLE 4.8

Solution of problems P1÷P8 and P10÷P12 for each starting guess x_0: number It *of iterations, number* Je *of Jacobian evaluations and number* Fe *of F-evaluation.*

behaviour of the method, we have tested it on both real-life problems (modeling of gas distribution networks) and on problems from the literature. The numerical experience showed the reliability of the method. As expected, the most efficient Quasi-Newton implementation is problem dependent and the use of secant approximations to the Jacobian matrix is beneficial when the Jacobian evaluations are computationally burdensome. Finally, the new methods compare very favourably with the monotone trust-region affine-scaling procedure implemented in TRESNEI.

Acknowledgments. We wish to thank Prof. Philippe Toint for valuable discussions on evaluation complexity and on the solution of large sparse nonlinear systems.

REFERENCES

[1] S. Bellavia, M. Macconi, B. Morini, *An affine scaling trust-region approach to bound-constrained nonlinear systems*, Appl. Numer. Math., 44 (2003), pp. 257-280.

[2] S. Bellavia, B. Morini, *Subspace trust-region methods for large bound-constrained nonlinear equations*, SIAM J. Numer. Anal., 44 (2006), pp. 1535-1555.

[3] S. Bellavia, S. Pieraccini, *On affine-scaling inexact dogleg methods for bound-constrained nonlinear systems*, Optim. Methods Softw., 30 (2015), pp. 276-300.

[4] E.G. Birgin, N. Krejić, J.M. Martínez, *Globally convergent inexact quasi-Newton methods for solving nonlinear equations*, Numer. Algor., 32 (2003), pp. 249–260.

[5] I.D.L. Bogle, J.D. Perkins, *A new sparsity preserving Quasi-Newton update for solving nonlinear equations*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 621-630.

		Total execution time	
Problem	x_0	FDN	TRESNEI
P1	$x_0^{(1)}$	**36.4**	102.1
	$x_0^{(2)}$	**60.1**	102.7
P2	$x_0^{(1)}$	**170.0**	229.8
	$x_0^{(2)}$	**410.0**	576.4
P3	$x_0^{(1)}$	**6.3**	8.3
	$x_0^{(2)}$	**15.4**	24.1
P4	$x_0^{(1)}$	**17.7**	24.4
	$x_0^{(2)}$	**49.3**	77.2
P5	$x_0^{(1)}$	**31.0**	49.5
	$x_0^{(2)}$	**87.5**	135.7
P6	$x_0^{(1)}$	68.2	**62.4**
	$x_0^{(2)}$	**136.6**	195.5
P7	$x_0^{(1)}$	**52.7**	104.5
	$x_0^{(2)}$	**65.9**	190.6
P8	$x_0^{(1)}$	**120.5**	171.3
	$x_0^{(2)}$	**217.7**	367.5
P10	$x_0^{(1)}$	**3.1**	31.9
	$x_0^{(2)}$	**2.7**	6.2
P11	$x_0^{(1)}$	2.1	**1.6**
	$x_0^{(2)}$	2.1	**1.6**
P12	$x_0^{(1)}$	**66.5**	84.9
	$x_0^{(2)}$	**116.7**	$\mathcal{F}$

TABLE 4.9
Total execution time in seconds for Algorithm 1 with FDN *and* TRESNEI *on problems P1÷P8 and P10÷P12.*

[6] C.G. Broyden, *The convergence of an algorithm for solving sparse nonlinear systems*, Math. Comput., 25 (1971), pp. 285-294.

[7] C. Carcasci, L. Marini, B. Morini, M. Porcelli, *A new modular procedure for industrial plant simulations and its reliable implementation*, Energy, 94, pp. 380-390, 2016.

[8] T.F. Coleman, Y. Li, *An interior trust region approach for nonlinear minimization subject to bounds*, SIAM J. Optim., 6 (1996), pp. 418-445.

[9] A.R. Curtis, M.J.D. Powell, J.K. Reid, *On the estimation of sparse Jacobian matrices*, IMA J. Appl. Math., 13 (1974), pp. 117-119.

[10] J.E. Dennis, R.B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice Hall, Englewood Cliffs, NJ, 1983.

[11] J.E. Dennis, J. Moré, *A characterization of superlinear convergence and its application to Quasi-Newton methods*, Math. Comput., 28 (1974), pp. 549-560.

[12] E.D. Dolan, J.J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program. 91 (2002), 201-213.

[13] M.L.N. Gonçalves, J. G. Melo, *A Newton conditional gradient method for constrained nonlinear systems*, J. Comput. Appl. Math., 311 (2017), pp. 473-483.

[14] M.L.N. Gonçalves, F.R. Oliveira, *An inexact Newton-like conditional gradient method for constrained nonlinear systems*, ARXIV: 2017arXiv170507684G, 2017.

[15] A. Griewank, *The global convergence of Broyden-like methods with a suitable line search*, Journal of the Australian Mathematical Society, Series B 28, 1 (1986), pp. 75-92.

[16] C. Kanzow, *An active set-type Newton method for constrained nonlinear systems*, M.C. Ferris, O.L. Mangasarian and J.-S. Pang (eds.): Complementarity: Applications, Algorithms and Extensions. Kluwer Academic Publishers, 2001, pp. 179-200.

[17] C. Kanzow, N. Yamashita, M. Fukushima, *Levenberg-Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints*, J. Comput. Appl. Math., 172 (2004), pp. 375-397.

[18] W. La Cruz, *A projected derivative-free algorithm for nonlinear equations with convex constraints*, Optim. Methods Softw., 29 (2014), pp. 24-41.

[19] W. La Cruz, J.M. Martínez, M. Raydan, *Spectral residual method without gradient information for solving large-scale nonlinear systems of equations*, Math. Comput., 18 (2006), pp. 1429-1448.

[20] W. La Cruz, M. Raydan, *Nonmonotone spectral methods for large-scale nonlinear systems*, Optim. Method. Softw., 18 (2003), pp. 583-599.

[21] D.H. Li, M. Fukushima, *A derivative-free line search and global convergence of Broyden-like method for nonlinear equations*, Optim. Methods Softw., 13 (2000), pp. 181-201.

[22] L. Lukšan, J. Vlček, *Computational experience with globally convergent descent methods for large sparse systems of nonlinear equations*, Optim. Methods Softw., 8 (1998), pp. 201-223.

[23] L. Lukšan, J. Vlček, *Test Problems for Unconstrained Optimization*, Tech. Rep. V-897, ICS AS CR, November 2003.

[24] M. Macconi, B. Morini, M. Porcelli, *Trust-region quadratic methods for nonlinear systems of mixed equalities and inequalities*, Appl. Numer. Math., 59 (2009), pp. 859-876.

[25] J.M. Martínez, *Practical quasi-Newton methods for solving nonlinear systems*, J. Comput. Appl. Math, 124 (2000), pp. 97-121.

[26] J.M. Martínez, M. Zambaldi, *An inverse column-updating method for solving large-scale nonlinear systems of equations*, Optim. Methods Softw., 1(2) (1992), pp. 129-140.

[27] B. Morini, M. Porcelli, *TRESNEI, a Matlab trust-region solver for systems of nonlinear equalities and inequalities*, Comput. Optim. Appl., 51 (2012), pp. 27-49.

[28] B. Morini, M. Porcelli, Ph. L. Toint, *Approximate norm descent methods for constrained nonlinear systems*, Math. Comput. (2017), published electronically, DOI: https://doi.org/10.1090/mcom/3251.

[29] B.R. Munson, D.F. Young, T.H. Okiishi, *Fundamentals of fluid mechanics*, John Wiley & Sons. Inc., USA, 2006.

[30] M. Porcelli, *On the convergence of an inexact Gauss-Newton trust-region method for nonlinear least-squares problems with simple bounds*, Optim. Lett., 7 (2013), pp. 447-465.

[31] M. Porcelli, Ph. L. Toint, *BFO, A Trainable Derivative-free Brute Force Optimizer for Nonlinear Bound-constrained Optimization and Equilibrium Computations with Continuous and Discrete Variables*, ACM Trans. Math. Softw. 44, 1, Article 6 (2017), 25 pages.

[32] L.K. Schubert, *Modifications of a Quasi-Newton for nonlinear equations with sparse Jacobian*, Math. Comput., 24 (1970), pp.27-30.

[33] L. Zhao, W. Sun, *A conic affine scaling dogleg method for nonlinear optimization with bound constraints*, Asia Pac. J. Oper. Res., 30 (2013), pp. 1340011-1–1340011-30.

[34] D. Zhu, *An affine scaling trust-region algorithm with interior backtracking technique for solving bound-constrained nonlinear systems*, J. Comput. Appl. Math., 184 (2005), pp. 343-361.