Applied Mathematics and Computation 205 (2008) 202–211

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

# An autoadaptative limited memory Broyden's method to solve systems of nonlinear equations

M. Ziani [a,b], F. Guyomarc'h [b,*]

[a] LERMA, Mohammadia School Engineering, BP 765, Rabat-Agdal, Morocco
[b] IRISA, INRIA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France

ARTICLE INFO

*Keywords:*
Limited memory Broyden method
Rank reduction
Superlinear convergence
Autoadaptativity

ABSTRACT

We propose a new Broyden-like method that we call autoadaptative limited memory method. Unlike classical limited memory method, we do not need to set any parameters such as the maximal size, that solver can use. In fact, the autoadaptative algorithm automatically increases the approximate subspace when the convergence rate decreases. The convergence of this algorithm is superlinear under classical hypothesis. A few numerical results with well-known benchmarks functions are also provided and show the efficiency of the method.



## 1. Introduction

Consider the problem of finding a solution of the system of nonlinear equations

$$F(x) = 0, \quad F : \mathbb{R}^n \to \mathbb{R}^n. \tag{1}$$

The mapping $F$ is assumed to have the following classical assumptions:

- the mapping $F$ is continuously differentiable in an open convex set $\mathscr{D}$;
- there is an $x_*$ in $\mathscr{D}$ such that $F(x_*) = 0$ and $F'(x_*)$ is nonsingular; (CA)
- the Jacobian $F'$ is Lipschitz continuous at $x_*$.

The well-known method for solving this problem is Newton's method. For an initial guess $x_0$ near $x_*$, this method converges quadratically. However, an iteration of the algorithm turns out to be expensive, because it requires one $F$-evaluation, one $F'$-evaluation and solving a linear system implying the Jacobian matrix. For more details see [10,12,16].

A direct solution of the above linear system is often expensive. Inexact Newton methods [4,5] allow to find approximately its solution using an iterative solver like Newton–Krylov. The convergence of the linear solver is stopped as soon as its residual is lower than the current Newton iteration's residual. Quasi-Newton methods are used to reduce the evaluation cost of the Jacobian matrix. They use only approximations of this matrix [7,13]. Also, if the function evaluations are very expensive, the cost of a solution by quasi-Newton methods could be much smaller than with inexact Newton methods. In particular, Broyden's method [2] that uses successive approximations by carrying out rank-one updates, requires only one $F$-evaluation per iteration. Given an initial guess $x_0$ and an initial Jacobian approximation $B_0$, and denoting $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$, the Broyden algorithm can be written as follows:

* Corresponding author.
*E-mail addresses:* Mohammed.Ziani@irisa.fr (M. Ziani), Frederic.Guyomarch@irisa.fr (F. Guyomarc'h).

0096-3003/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2008.06.047

Broyden's algorithm

For $k = 0, 1, 2, \ldots$ until convergence,
Solve $B_k s_k = -F(x_k)$ for $s_k$,
$x_{k+1} = x_k + s_k$,
$y_k = F(x_{k+1}) - F(x_k)$,

$$B_{k+1} = B_k + (y_k - B_k s_k)\frac{s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} s_k}, \tag{2}$$

Under the classical hypothesis (CA), this algorithm converges locally and superlinearly [3]. However, a drawback of this method is the storage of the Broyden's matrix. In fact, this matrix contains one vector per iteration. Thus, the storage costs $\mathcal{O}(kn)$ where $k$ is the total number of iterations. And this cost becomes prohibiting in case of poor convergence. Thus, a restarted version of this method was introduced in [9]. Unfortunately, convergence becomes slow if there are too many restarts. In fact, all information gathered during previous iterations is lost when restarting. To overcome this drawback, the limited memory Broyden methods do not discard the approximate subspace [17,18] but refine it. Among these methods, the rank reduction method, detailed below, gives good results.

To describe the algorithms in this paper, we need the concept of an update function. Update functions are only a mean to denote the various Jacobian approximations which might be used in the iterative process [6]. Let $\mathscr{L}(\mathbb{R}^n)$ denote the space of all linear maps from $\mathbb{R}^n$ to $\mathbb{R}^n$, and $\mathscr{P}(\mathscr{L}(\mathbb{R}^n))$ denotes the collection of all subsets of $\mathscr{L}(\mathbb{R}^n)$. If the scheme for a quasi-Newton method is written

$$x_{k+1} = x_k - B_k^{-1} F(x_k), \tag{3}$$

the method of generating the matrices $\{B_k\}$ can then be described by specifying for each $(x_k, B_k)$ a nonempty set $\Phi(x_k, B_k)$ of possible candidates for $B_{k+1}$, where

$$\Phi : \mathbb{R}^n \times \mathscr{L}_n(\mathbb{R}) \to \mathscr{P}(\mathscr{L}_n(\mathbb{R}))$$

is a well defined update function. For example, if

$$\overline{B} = B + \frac{(y - Bs)s^{\mathrm{T}}}{s^{\mathrm{T}} s}, \tag{4}$$

then the Broyden algorithm can be written as

$$x_{k+1} = x_k - B_k^{-1} F(x_k),$$

where $B_{k+1} \in \Phi(x_k, B_k)$ and

$$\Phi(x, B) = \{\overline{B} : s \neq 0\}.$$

In this case $s = \bar{x} - x$ and $y = F(\bar{x}) - F(x)$, where $\bar{x} = x - B^{-1}F(x)$.

The organization of the paper is as follows. Section 2 introduces a description of the rank reduction method. In Section 3, we introduce an autoadaptative limited memory and show its local and superlinear convergence. Section 4 introduces a modified version of the autoadaptative algorithm. Finally, in Section 5 we present some numerical results. In the following, $\|.\|$ and $\|.\|_F$ stand, respectively, for the Euclidean and the Frobenius norm.

## 2. Rank reduction method

The Broyden rank reduction method consists in approaching the update matrix by a low rank matrix [18]. Eq. (2) implies that if an initial matrix $B_0$ is updated $l$ times, the resulting matrix $B_l$ can be written as follows:

$$B_l = B_0 + \sum_{k=0}^{l-1} (y_k - B_k s_k)\frac{s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} s_k} = B_0 + CD^{\mathrm{T}} = B_0 + Q, \tag{5}$$

with $C = [c_1, \ldots, c_l]$, $D = [d_1, \ldots, d_l]$, defined by

$$c_{k+1} = \frac{(y_k - B_k s_k)}{\|s_k\|}, \quad d_{k+1} = \frac{s_k}{\|s_k\|}, \quad k = 0, \ldots, l-1.$$

The matrix

$$Q = CD^{\mathrm{T}} = \sum_{k=1}^{l} c_k d_k^{\mathrm{T}}$$