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

Mohammed Ziani, Frédéric Guyomarc'H

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

M. Ziani [1,2] F. Guyomarc'h [2]

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.

Key words: Limited memory Broyden method; rank reduction; superlinear convergence; autoadaptativity.

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 $\mathcal{D}$;

Email addresses: Mohammed.Ziani@irisa.fr (M. Ziani), Frederic.Guyomarch@irisa.fr (F. Guyomarc'h).
[1] LERMA, Mohammadia School Engineering, BP 765, Rabat-Agdal, Morocco
[2] IRISA, INRIA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France

- there is an x_* in $\mathcal{D}$ such that $F(x_*) = 0$ and $F'(x_*)$ is nonsingular;
- the Jacobian F' is Lipschitz continuous at x_*. (CA)

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, 16, 12].

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.

Broyden's algorithm

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

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

Under the classical hypothesis (CA), this algorithm converges locally and superlinearly (Broyden, Dennis and Moré [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 [18, 17] 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 $\mathcal{L}(\mathbb{R}^n)$ denote the space of all linear maps from $\mathbb{R}^n$ to $\mathbb{R}^n$, and $\mathcal{P}(\mathcal{L}(\mathbb{R}^n))$ denotes the collection of all subsets of $\mathcal{L}(\mathbb{R}^n)$. 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

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

is a well defined update function. For example, if

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

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

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

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]. Equation (2) implies that if an initial matrix B_0 is updated l times, the resulting matrix B_l can be written as follows:

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

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

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

The matrix

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

is the update matrix. Its rank is no more than l and its storage requires at most $2l$ vectors. In many cases, we can choose $B_0 = I$, then B_l can be stored implicitly using also $2l$ vectors.

The rank reduction idea uses the fact that the best approximation of rank p is given by the truncated SVD [8]. When the rank of the update matrix Q is higher than p, a given parameter limiting the available memory, we compute the singular values decomposition of the matrix $Q = U\Sigma V^T$ and then remove the smallest singular value and its corresponding left and right singular vectors from the decomposition of Q.
The matrix B_l in (5) is replaced by

$$\tilde{B} = B_l - \sigma_p u_p v_p^T = B_0 + \sum_{k=1}^{p-1} \sigma_k u_k v_k^T.$$

Then, the memory liberated is used to store a new update. Formally, the rank reduction method is given in definition 2.1.

Definition 2.1 *[18] Let $F: \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}^n$ be given. Choose x_0 in a neighborhood $\mathcal{V}(x_*)$ of the solution x_* ($F(x_*) = 0$), and $B_0 \in \mathcal{L}(\mathbb{R}^n)$. Define the update function $\Phi : \mathbb{R}^n \times \mathcal{L}(\mathbb{R}^n) \to \mathcal{P}(\mathcal{L}(\mathbb{R}^n))$ by $\Phi(x, B) = \{\bar{B} : s \neq 0\}$, where*

$$\bar{B} = B + (y - Bs)\frac{s^T}{s^T s} - \sigma_p u_p v_p^T \left(I - \frac{ss^T}{s^T s}\right),$$

with $s = \bar{x} - x$ and $y = F(\bar{x}) - F(x)$ for $\bar{x} = x - B^{-1}F(x)$. Then an iteration of the rank reduction method is defined by

$$\begin{cases} x_{k+1} &= x_k - B_k^{-1}F(x_k), \\ \tilde{B_k} &= B_k - \sigma_p u_p v_p^T, \\ B_{k+1} &= \tilde{B_k} + (F(x_{k+1}) + \sigma_p u_p v_p^T s_k)\frac{s_k^T}{s_k^T s_k} \\ &= B_k + (y_k - B_k s_k)\frac{s_k^T}{s_k^T s_k} - \sigma_p u_p v_p^T \left(I - \frac{s_k s_k^T}{s_k^T s_k}\right) \end{cases} \tag{6}$$

where σ_p, u_p and v_p are respectively the minimal singular value and its left and right vectors.

In implementations, the matrices C and D are used instead of the matrix B_k. After a reduction, the pth columns of the matrices C and D are replaced by

$$\begin{aligned} c_p &= \frac{1}{\|s_k\|}(F(x_{k+1}) + \sigma_p u_p v_p^T s_k) \\ d_p &= \frac{s_k}{\|s_k\|}. \end{aligned}$$

Rotten and Verduyn Lunel ([18]) showed that the rank reduction method converges locally and superlinearly if the removed singular value σ_p is smaller than current step size, i.e.,

$$\sigma_p \leq \|s_{k-1}\|.$$

Simulations show, however, that after several iterations of the rank reduction method the pth singular value of the update matrix remains more or less of the same size (see Fig. 1), whereas the step size $\|s_{k-1}\|$ decreases to zero in case of convergence. This implies that the local and superlinear convergence of the rank reduction method cannot be exhibited since the assumptions in theorem 3.3 given in [18] are not all satisfied. In fact, the choice of the value of p has an impact on the convergence. Table 1 presents the execution time of the Broyden rank reduction method, for different values of p, in the case of the Martinez function (see appendicies). The size of the problem is $n = 100000$ and $(x_0)_j = 0.1,\ j = 1, ..., n$. The value of p should not be too large, where the SVD computation will be costly, nor too small because the algorithm does not have good Broyden directions and the globalization line search strategy needs more evaluations of the function F. However, it is difficult to choose *a priori* the value of p. To eliminate this drawback, we introduce an autoadaptative limited memory Broyden's method and we study its local and superlinear convergence in the following paragraph.

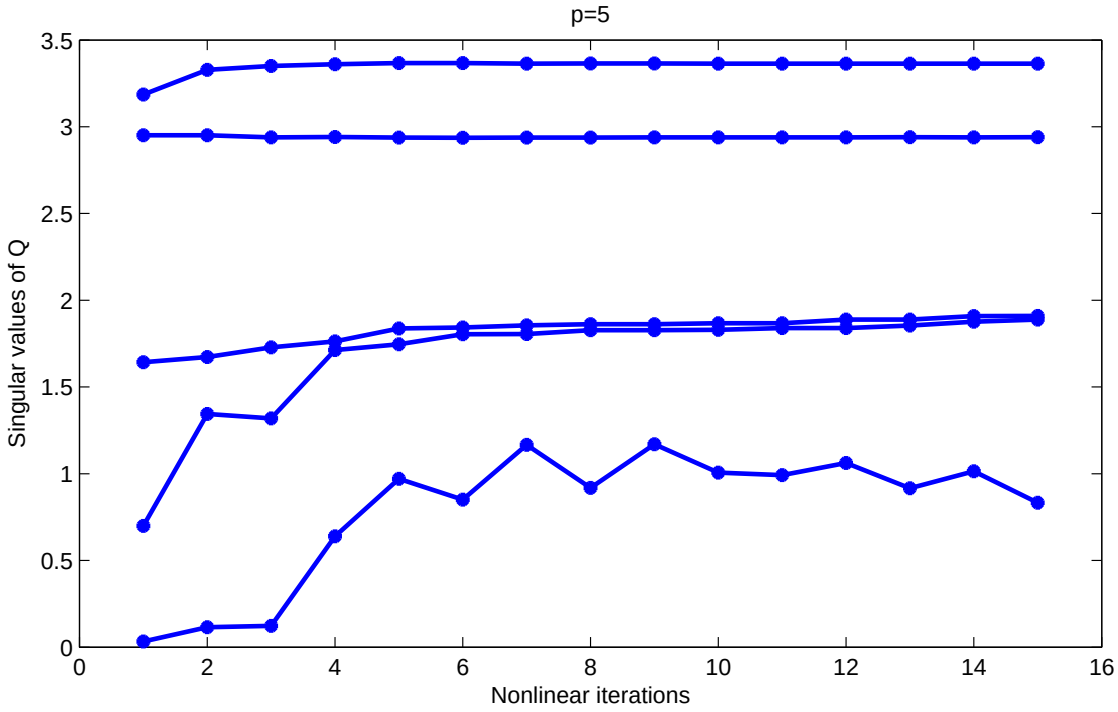

Fig. 1. The distribution of the singular values of the update matrix, Q, for $p = 5$, in case of the convection-diffusion equation ([9])

Table 1
Impact of the choice of p on the execution time, in case of the Martinez function.

p	1	3	4	5	6	10
CPU time	68.955	65.827	63.438	70.405	100.070	132.046
Iterations	340	134	114	104	117	112
F evaluataions	1017	386	317	287	330	311

3 An autoadaptative limited memory method

The idea of this approach is to apply the rank reduction method as long as the quotient $\sigma_p/\|s_{k-1}\|$ remains controlled (lower than a threshold), i.e.,

$$\sigma_p \leq \eta \|s_{k-1}\|, \qquad k \in \mathbb{N}, \qquad \eta > 0, \tag{7}$$

where p stands for the parameter limiting the memory and k denotes the current nonlinear iteration. If this condition is not satisfied, we increase the rank of the approximation, *i.e.* we perform a classical Broyden's iteration. For a given threshold parameter $\eta > 0$, the algorithm is written as follows.

Autoadaptative algorithm

1. Set $p = 1$.
2. For $k = 0, 1, ...$ until convergence
   2.1 Apply a Broyden step
   Solve $B_k s_k = -F(x_k)$ for s_k,
   $x_{k+1} = x_k + s_k$,
   $y_k = F(x_{k+1}) - F(x_k)$,
   $B_{k+1} = B_k + (y_k - B_k s_k)\frac{s_k^T}{s_k^T s_k}$,
   2.2 If $\| F(x_{k+1}) \| < \varepsilon$, convergence is satisfied
   2.3 If $\sigma_p \leq \eta \| s_k \|$, reduce the approximate space

$$B_{k+1} \leftarrow B_{k+1} - \sigma_p u_p v_p^T.$$

   Else $p \leftarrow p + 1$.

Note that in this algorithm the matrices C and D are used in implementations instead of the matrix B_k. For more details of the update of these matrices, see [17]. The autoadaptative limited memory algorithm has a local and superlinear convergence. The details of the proof are given in the next paragraph.

The first step is to generalize the theorem 3.3 given in [18]. Indeed we need a modified version of this theorem which allow us to ensure convergence for any fixed threshold for the ratio $\frac{\|R\|}{\|s\|}$. This corresponds to the update function

$$\Phi(x, B) = \{\bar{B} + R : \| R \|_F \leq \eta \| s \|, \quad \eta > 0, \quad s \neq 0\}. \tag{8}$$

Then we can prove the local and superlinearly convergence.

Theorem 3.1 *Let $\eta > 0$ and $F : \mathbb{R}^n \to \mathbb{R}^n$ be differentiable in the open, convex set $\mathcal{D}$, and assume that for some x_* in $\mathcal{D}$ and $F'(x_*)$ is K-Lipschitz at the point x_*, where $F(x_*) = 0$ and $F'(x_*)$ is nonsingular. Then the update function*

$$\Phi(x, B) = \left\{ \bar{B} - R\left(I - \frac{ss^T}{s^T s}\right) : \|R\|_F < \eta\|s\|, \eta > 0, s \neq 0 \right\},$$

where $\bar{B}$ is given by the equation (4), is well defined in a neighborhood $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$

of $(x_*, F'(x_*))$*, and the corresponding iteration*

$$x_{k+1} = x_k - B_k^{-1} F(x_k) \tag{9}$$

with $B_{k+1} \in \Phi(x_k, B_k), k \geq 0$*, is locally and superlinearly convergent at* x_**.*

The sufficient condition [3] for the sequence $\{x_k\}$ to converge superlinearly to x_* is:

$$\lim_{k\to\infty} \frac{\|E_k s_k\|}{\|s_k\|} = 0,$$

where $E_k = B_k - F'(x_*)$ and $s_k = x_{k+1} - x_k$. The proof follows the proof of the theorem 3.3 given in [18]. Notice just that with the new hypothesis we have

$$\|\bar{E}\|_F \leq \|E\|_F + (K + 2\eta) \max\{\|\bar{e}\|, \|e\|\},$$

when proving Lemma 3.5 in [18].

■

In case of the autoadaptative limited memory method, the round-off matrix in theorem 3.1 is given by the pth term in the singular value decomposition of the update matrix

$$Q = \sum_{i=1}^{p} \sigma_i u_i v_i^T.$$

The matrix $B_{k+1} = B_0 + Q$, in case of the autoadaptative limited memory method is in the set $\Phi_{aa}(x_k, B_k)$, where

$$\Phi_{aa}(x, B) = \begin{cases} \left\{ \bar{B} - \sigma_p u_p v_p^T \left(I - \frac{ss^T}{s^T s} \right) : \sigma_p < \eta \|s\|, s \neq 0, \eta > 0 \right\}, & \text{if } \sigma_p \leq \eta \|s\| \\ \\ \left\{ \bar{B} : s \neq 0 \right\}, & \text{otherwise .} \end{cases} \tag{10}$$

And we have the following theorem

Theorem 3.2 *Under classical hypothesis, the autoadaptative limited memory method converges locally and superlinearly.*

Table 2
Performance of the autoadaptative limited memory method, for the Broyden banded function

η	$1e-6$	$1e-2$	$1e2$	$1e10$	$1e12$	Broyden
CPU time	315.283	226.770	136.017	**106.532**	111.901	124.763
Iterations	72	66	58	**40**	41	73
F evaluations	132	126	**113**	116	121	132
p final value	66	51	31	2	1	–

The proof comes directly from theorem 3.1 because

$$\Phi_{aa}(x,B) \subseteq \left\{ \bar{B} - R\left(I - \frac{ss^T}{s^Ts}\right) : \|R\|_F < \eta\|s\|, \eta > 0, s \neq 0 \right\}.$$

In fact if $\sigma_p > \eta\|s\|$ then there is no rank reduction and then $R = 0$ in this case.

■

4 Preliminary results and modified version of the algorithm

We apply first the autoadaptative limited memory algorithm to the Broyden banded function [15]. The performance of this algorithm is compared to the Broyden algorithm described in [9]. For this example, the size of the problem is $n = 100000$, and the initial guess is given by $x_0 = 0$. We search a solution of the inequality

$$\| F(x) \| < 10^{-10}.$$

The performance of the autoadaptative limited memory method, for different values of η, is given in table 2.
For too small values of η, for example $\eta \leq 10^{-2}$, the autoadaptative method does not converge as well as the Broyden's method. Indeed, in this case, the value of p tends to increase and thus the singular values decomposition of the update matrix takes more computational time.
On the other hand, when the value is too large, the convergence is slow because the value of p remains too small and then the approximation of the Jacobian matrix is not good enough during the first iterations. In fact, if we look closer at the convergence, we can say that the ideal behavior for p value is to increase in the first phase of convergence and then to remain constant while the approximation size is good enough to ensure a good convergence. This didn't occur with the fixed

threshold. In fact, since σ_p remains almost constant, when $\| s_{k-1} \|$ becomes too small, the p value may increase almost at each iteration. Increasing the η value whenever the ratio $\sigma_p / \| s_{k-1} \|$ increases too much will allow us to keep the p value constant for many iterations. That is why the algorithm is modified to adjust the threshold value during the iterations. We start with a low η and increase it each time that the p value is also increased. Then the p value becomes harder to increase and the rank of the update matrix Q remains more or less constant while the convergence occurs. Finally, we use the following modified autoadaptative limited memory algorithm.

Modified autoadaptative algorithm

---

1. Set $p = 1$ and an initial value η. Let $\alpha > 1$.
2. For $k = 0, 1, ...$ until convergence
   2.1 Apply a Broyden step
   Solve $B_k s_k = -F(x_k)$ for s_k,
   $x_{k+1} = x_k + s_k$,
   $y_k = F(x_{k+1}) - F(x_k)$,
   $B_{k+1} = B_k + (y_k - B_k s_k) \frac{s_k^T}{s_k^T s_k}$,
   2.2 If $\| F(x_{k+1}) \| < \varepsilon$, convergence is satisfied
   2.3 If $\sigma_p \leq \eta \| s_k \|$, reduce the approximate space

$$B_{k+1} \leftarrow B_{k+1} - \sigma_p u_p v_p^T.$$

   Else $p \leftarrow p + 1$ and $\eta \leftarrow \alpha \eta$.

For the remaining numerical tests we use the modified version of the algorithm and take arbitrarily $\alpha = 10$. We recommend to choose an initial value η_{init} for η in $[1e-2, 1e2]$. In implementations we should also set an upper limit of the η value. In this case, η remains bounded and consequently the convergence theorem can be applied.

Table 3
Performance of the modified autoadaptative limited memory Broyden method, in case of the Martinez function

η_{init}	$1e-6$	$1e-4$	$1e-2$	1	$1e3$	$1e5$	Broyden
CPU time	226.055	129.760	119.133	68.886	48.963	47.218	422.279
Iterations	105	95	95	87	132	184	196
F evaluations	285	258	249	221	369	526	582
p final value	18	16	14	12	9	7	-

5 Numerical results

In this section we present numerical tests by applying the modified autoadaptative limited memory method to a few classical test functions from the literature and we present a comparison of this method with the Broyden's method described in [9]. Details of the test functions are given in the appendices.

5.1 Martinez function

The size of this problem is $n = 100000$, and the initial guess is given by $x_i = 0.1,\ \ i = 1, ..., n$. We search a solution of the inequality

$$\| F(x) \| < 10^{-10}.$$

For the Martinez function, the autoadaptative limited memory Broyden's method efficiently reduces the computational time (see table 3). Indeed, it requires less evaluations of the function F, and the size of the update matrix does not increase too much. In figure 2, we plot the nonlinear residual against the number of nonlinear iterations, for different values of η. The black circles represent the increase of p during the iterations. We first notice that a too big η_{init} slows down the convergence too much during the first phase and then it is important to choose it not too big. If we choose a very small η_{init} then we may consume memory which is not compulsory for a good convergence.

For the Broyden's method, the computational time is spent in the evaluation of the function F, and the computation of products B_k with a vector, see table 4. For the modified autoadaptative limited memory Broyden's method the computational time is spent in evaluations of the function F and the computation of the singular

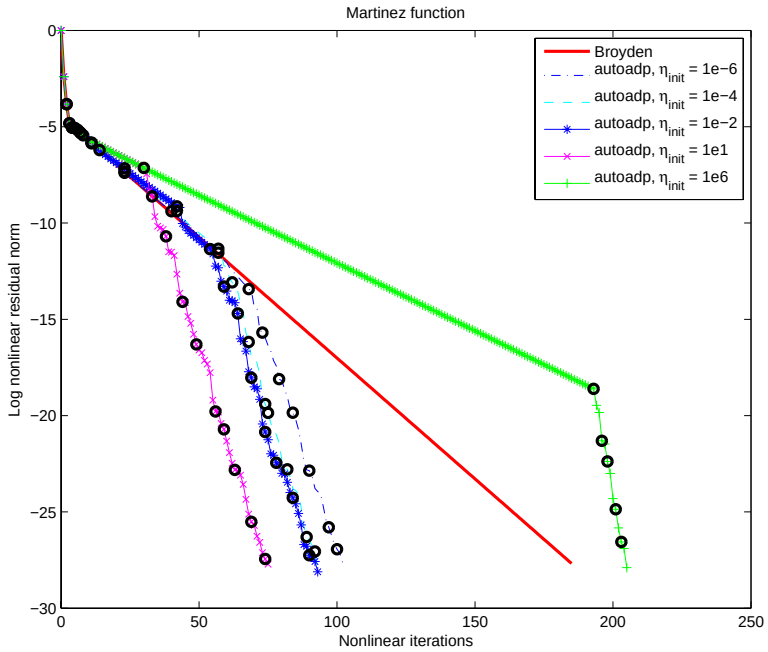

Fig. 2. The convergence of the modified autoadaptative limited memory method, in case of the Martinez function

Table 4
Profiling results for the Broyden method, in case of the Martinez function

	F evaluations	products with B_k
Calls	582	18721
%time	50.6	45.8

value decomposition of the update matrix (if the value of p increases too much), see table 5.

5.2 Broyden tridiagonal function[15]

The size of the problem is $n = 100000$, and the initial guess is given by $x_0 = 0$. We search a solution of the inequality

$$\| F(x) \| < 10^{-10}.$$

For different initial values of η, the performance of the autoadaptative limited memory Broyden's method is given in table 6. Again, for this example the modified

Table 5
Profiling results for the modified autoadaptative limited memory method, in the case of the Martinez function

η_{init}	$1e-6$	$1e-4$	$1e-2$	$1e1$	$1e6$
p final value	18	16	14	11	4
F calls	285	258	249	200	587
% F time	48.8	56.0	60.0	72.6	92.6
SVD calls	103	93	93	77	199
% SVD time	42.5	31.8	27.2	17.2	2.2

Table 6
Performance of the modified autoadaptative limited memory Broyden method, in case of the Broyden tridiagonal function

η_{init}	$1e-4$	$1e-2$	$1e1$	$1e3$	$1e8$	Broyden
CPU time	44.671	32.589	30.022	**27.203**	31.508	52.396
Iterations	72	**68**	70	70	88	109
F evaluations	177	**161**	173	179	249	320
p final value	16	14	11	9	5	–

autoadaptative method converges better than the Broyden's method. In figure 3, we plot the nonlinear residual during the nonlinear iterations. The black circles represent an increase of p.

In the modified autoadaptative limited memory method, to reduce the necessary computational time of the SVD of the update matrix and the function evaluations, it is necessary to avoid use of too small or too large initial values for η. The table 7 shows the number of function evaluations and singular value decompositions for different values of η_{init}. It also shows the portion of the total time spent for these two tasks. The goal is then to find a balance between these two costs.

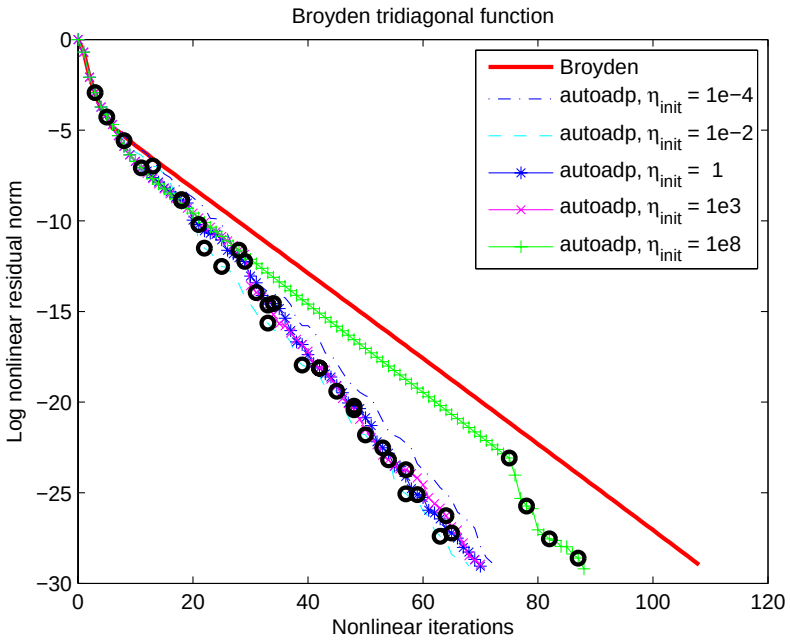

Fig. 3. The convergence of the modified autoadaptative limited memory Broyden method, in case of the Broyden tridiagonal function

Table 7
Profiling results for the modified autoadaptative limited memory method, in case of the Broyden tridiagonal function

η_{init}	$1e-4$	$1e-2$	$1e1$	$1e3$	$1e8$
p final value	16	14	11	9	5
F calls	177	161	173	179	249
% F time	48.5	54.0	67.1	75.9	91.3
SVD calls	70	66	68	68	86
% SVD time	38.2	33.3	21.7	14.2	2.2

5.3 Spedicato4 function [15], function 4

The size of the problem is $n = 100000$, and the initial guess is given by $x_0 = (-1.2, ..., -1.2, 1)^T$. We search a solution of the inequality

$$\| F(x) \| < 10^{-12}.$$

Table 8
Performance of the modified autoadaptative limited memory Broyden method, in case of the Spedicato4 function

η_{init}	$1e-6$	$1e-4$	$1e-2$	1	Broyden
CPU time	25.675	25.194	203.114	301.995	65.110
Iterations	33	38	200	200	55
F evaluations	180	199	2311	3600	734
p final value	7	7	2	1	–

Table 9
Performance of the modified autoadaptative limited memory Broyden method, in case of the discrete integral equation function

η_{init}	$1e-2$	$1e-1$	1	$1e1$	$1e2$	$1e4$	Broyden
CPU time	1065.208	1109.916	1045.173	1225.619	1150.727	1295.578	1058.041
Iterations	7	7	7	7	7	8	8
F evaluations	8	8	8	8	8	9	8
p final value	4	3	3	3	3	3	–

For initial values of η that are more than 10^{-4} the modified autoadaptative algorithm does not converge before 200 iterations. Otherwise, the method converges as well as the Broyden method, see table 8. In fact, for these cases, the value of p does not increase. Thus the approximate subspace does not allow to determine more precisely the Broyden directions. Starting with more than one Broyden direction ($p > 1$) can lead to the convergence of the algorithm.

5.4 Discrete integral equation function[14]

The size of the problem is $n = 10000$, and the initial guess is given by $x_0 = 0$. We search a solution of the inequality

$$\| F(x) \| < 10^{-10}.$$

For different initial values of η, the performance of the autoadaptative limited memory is nearly the same as that of the Broyden method, see table 9. In this example also the p value does not increase too much. The computational time is especially spent in the evaluation of the function F although that, for each initial value of η, the convergence of the algorithm requires only a few F-evaluations.

Table 10
Choice of α, in case of the Martinez function

α	2	10	100	500	10^3	10^4	10^5	10^6
CPU time	482.797	116.748	53.790	45.031	42.406	37.788	39.941	38.139
Iterations	97	95	85	94	96	96	104	110
F evaluations	250	249	218	253	258	254	283	302
p final value	42	14	8	6	6	5	4	4

Hence, a solution by the autoadaptative limited memory method would cost much less than that of one by an inexact Newton method, which requires many F-evaluations for each nonlinear iteration.

5.5 Influence of α on the convergence

To see how the algorithm depends on the choice of α, we apply the modified scheme to the function of Martinez for different values of α. We take the case where $\eta_{init} = 10^{-2}$. The table 10 introduces the obtained results. We have seen in the section 2 that it is not trivial to identify an optimal value of p. For the modified scheme, the convergence depends on the choice of the value of α, but a sufficiently large value of α may identify a good memory to ensure convergence. In our tests we take arbitrarily $\alpha = 10$. We can take also a sequence that increases in the iteration.

6 Conclusion

We have presented an autoadaptative limited memory Broyden's method, and shown its locally superlinear convergence. It automatically adapts the memory to store the Broyden's directions. Numerical tests show that it reduces efficiently the cost of the necessary storage and the time to obtain the convergence. Increasing the threshold η with the quotient $\sigma_p/\|s_k\|$ gives the satisfactory results, but the presented strategy can be certainly refined. However, the procedure of choosing the parameter α to increase η in modified algorithm is not studied yet. We set $\alpha = 10$ arbitrarily. We show numerically that this choice is sufficient to prevent p to increase in each iteration when $\| s_{k-1} \|$ becomes too small. The threshold η can be updated by a technique as that in the choice of forcing terms in inexact

Newton methods [1]. The idea is to change η value depending on the increase of the ratio $\sigma_p / \parallel s_{k-1} \parallel$. The main drawback of this algorithm is that there is no result yet about the global convergence, and thus the initial guess must be not too far from the solution. But we can use the theoretical results given in [11] for the Globalization of this algorithm.

Appendix

(1) **Broyden banded function**

- Origin : [15]
- Dimension : $n \geq 7$
- Initial guess : not specified

$$
\begin{aligned}
F_1(x) &= x_1(2+5x_1^2)+1-x_2(1+x_2)\\
F_2(x) &= x_2(2+5x_2^2)+1-x_1(1+x_1)-x_3(1+x_3)\\
F_3(x) &= x_3(2+5x_3^2)+1-\sum_{j=1}^{2} x_j(1+x_j)-x_4(1+x_4)\\
F_4(x) &= x_4(2+5x_4^2)+1-\sum_{j=1}^{3} x_j(1+x_j)-x_5(1+x_5)\\
F_5(x) &= x_5(2+5x_5^2)+1-\sum_{j=1}^{4} x_j(1+x_j)-x_6(1+x_6)\\
F_n(x) &= x_n(2+5x_n^2)+1-\sum_{j=n-5}^{n-1} x_j(1+x_j)\\
F_i(x) &= x_i(2+5x_i^2)+1-\sum_{j=i-5}^{i-1} x_j(1+x_j)-x_{i+1}(1+x_{i+1}), \quad i=6,...,n-1
\end{aligned}
$$

(2) **Martinez function**

- Origin: 59th Martinez paper , 13th problem
- Dimension: n
- Initial guess: not specified

$$
\begin{aligned}
F_1(x) &= (3 - 0.1x_1)x_1 + 1 - 2x_2 + x_1 \\
F_i(x) &= (3 - 0.1x_i)x_i + 1 - x_{i-1} - 2x_{i+1} + x_i, \quad i = 2, ..., n-1 \\
F_n(x) &= (3 - 0.1x_n)x_n + 1 - 2x_{n-1} + x_n
\end{aligned}
$$

(3) **Broyden Tridiagonal function**

- Origin: [15]
- Dimension: n
- Initial guess: not specified

$$
\begin{aligned}
F_1(x) &= (3 - 2x_1)x_1 - 2x_2 + 1 \\
F_n(x) &= (3 - 2x_n)x_n - x_{n-1} + 1 \\
F_i(x) &= (3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1, \quad i = 2, ..., n-1
\end{aligned}
$$

(4) **Spedicato4 function**

- Origin: [15], function 4
- Dimension: n
- Initial guess: $x_0 = (-1.2, ..., -1.2, 1)^T$

$$
F_i(x) = \begin{cases} 1 - x_i & \text{if } i \text{ odd} \\ 10(x_i - x_{i-1}^2) & \text{if } i \text{ even} \end{cases}
$$

(5) **Discrete integral equation function**

- Origin: [14]
- Dimension: n
- Initial guess: $(x_0)_j = t_j(t_j - 1)$

$$
F_i(x) = x_i + \frac{h}{2}\left[(1 - t_i)\sum_{j=1}^{i} t_j(x_j + t_j + 1)^3 + t_i \sum_{j=i+1}^{n} (1 - t_j)(x_j + t_j + 1)^3\right],
$$

where $h = \frac{1}{n+1}$, $t_i = ih$ and $x_0 = x_{n+1} = 0$.

References

[1] Heng-Bin An, Ze-Yao Mo, and Xing-Ping Liu. A choice of forcing terms in inexact Newton method. *Journal of Comp. and Appl. Math.*, 200:47–60, 2007.

[2] C.G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Math. Comput.*, 19:577–593, 1965.

[3] C.G. Broyden, J.E.Dennis, and J.J. Moré. On the local and superlinear convergence of quasi-Newton methods. *J. Ins. Maths. Applics.*, 12:223–245, 1973.

[4] R. S. Dembo, C. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Num. Anal.*, 19:400–408, 1982.

[5] R. S. Dembo and T. Steihaug. Truncated Newton algorithms for large scale optimization. *Math. Prog.*, 26:190–212, 1983.

[6] J.E. Dennis and J.J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Rev.*, 19(1):46–89, 1977.

[7] J.E. Dennis and R.B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, 1983.

[8] G. Golub and C.F. Van Loan. *Matrix computations*. 3rd edition. John Hokins Press, 1996.

[9] C.T. Kelly. *Iterative methods for linear and nonlinear equations*. SIAM, 1995.

[10] C.T. Kelly. *Solving nonlinear equations with Newton's method*. SIAM, 2003.

[11] D.H. Li and M. Fukushima. Derivative-free line search and global convergence of Broyden-like method for nonlinear equations. *Optimization methods and software*, 13:181–201, 2001.

[12] J. M. Martinez. *Continuous Optimization. The state of art*, chapter Algorithms for solving nonlinear systems of equations, pages 81–108. Kluwer Academic Publishers, 1994.

[13] J. M. Martinez. Practical quasi-Newton methods for solving nonlinear systems. *Journal of Computational and Applied Mathematics*, 124:143–167, 2000.

[14] J.J Moré and M.Y. Cosnard. Numerical solution of nonlinear equations. *ACM Trans. Math. Soft.*, 5(1):64–85, 1979.

[15] J.J Moré, B.S. Garbow, and K.E. Hillstrom. Testing unconstrained optimization software. *ACM Trans. Math. Soft.*, 7(1):17–41, 1981.

[16] J.M. Ortega and W.C. Reinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[17] B. Van De Rotten. *A limited memory Broyden method to solve high dimensional systems of nonlinear equations*. PhD thesis, Mathematical Institute, University of Leiden, The Netherlands, 2003.

[18] B. Van De Rotten and S.M. Verduyn Lunel. A limited memory Broyden method to solve high dimensional systems of nonlinear equations. Technical Report MI 2003-06, Mathematical Institute, University of Leiden, The Netherlands, 2003.