Combined Method for Nonlinear Systems of Equations*

Peng Jiang¹, Geng Yang², and Chunming Rong³

^{1,2} School of Computer Science and Technology, P.O.Box 43,
Nanjing University of Posts and Telecommunications, 210003, Nanjing, China
alice20006@hotmail.com,
angg@njupt.edu.cn

³ Department of Electrical and Computer Engineering, University of Stavanger, Norway
chunming.rong@uis.no

Abstract. This paper proposes Block Broyden Method combined with SSOR preconditioning technique for solving nonlinear systems of equations. The implementation process is described in detail and the time complexity is analyzed. This method has faster solving speed and better performance than the unpreconditioned one, which is shown by several numerical tests arising from the Bratu problem. Therefore, it can be used in the large-scale problems arising from scientific and engineering computing.

1 Introduction

The Block Broyden Method^[1] is an effective iterative algorithm for solving nonlinear equations. It parallelizes very well and gives good CPU time savings. However, the iterative matrix in the Block Broyden Algorithm is a block diagonal matrix so that partial relevant information among the nodes is lost, therefore affecting the convergence speed of the algorithm to some extent. Hence, seeking for proper preconditioning methods^[2, 3] is one of the effective ways to solve this problem. Some preconditioners are proposed and analyzed in paper [4, 5].

In this article, we combine SSOR preconditioner with the Block Broyden Method. We organize our article as follows. Section 2 introduces SSOR preconditioner. Section 3 gives general remarks on SSOR Preconditioner based on Block Broyden Algorithm, and it also shows the implementation details and analyzes the time complexity. Some numerical results and interpretation of these results are included in Section 4. Section 5 contains the summary remarks.

2 SSOR Preconditioner

We suppose there are large sparse linear systems of the form as (1)

$$Ax = b. \quad (1)$$

* This work was supported in part by the Natural Science Foundation of Jiangsu Province under Grant No 05KJD520144 and the Foundation of the QingLan Project (KZ0040704006).

where $A = [a_{i,j}]$ is an $n \times n$ matrix and b a given right-hand-side vector. If the original, symmetric, matrix is decomposed as $A = D + L + L^T$ in its diagonal, lower and upper triangular part, the SSOR matrix is defined as

$$M = (D + L)D^{-1}(D + L)^T. \quad (2)$$

Then we can calculate the inverse of the preconditioner and get the transformed linear system $M^{-1}Ax = M^{-1}b$ which has the same solution as (1).

3 SSOR Preconditioner Based on Broyden Algorithm

3.1 General Remarks

In the following discussion, we are concerned with the problem of solving the large system of nonlinear equations as (3):

$$F(x) = 0. \quad (3)$$

where $F(x) = (f_1, \dots, f_n)^T$ is a nonlinear operator from R^n to R^n , and $x^* \in R^n$ is an exact solution. Suppose that the components of x and F are divided into q blocks:

$$F = \begin{pmatrix} F_1 \\ \vdots \\ F_q \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_q \end{pmatrix}$$

We consider generalization of the Block Broyden algorithm with Preconditioning method (BBP) as follows:

Algorithm 3.1.1. BBP Method

1. Let x^0 be an initial guess of x^* , and B^0 an initial block diagonal approximation of $J(x^0)$. Calculate $r^0 = F(x^0)$.
2. For $k = 0, 1, \dots$ until convergence:
 - 2.1 Solve $B^k s^k = -r^k$:
 - 2.1.1 Calculate SSOR preconditioner according to (2) and get the preconditioned equation.
 - 2.1.2 Solve the preconditioned equation by Jacobi method.
 - 2.2 Update the solution $x^{k+1} = x^k + s^k$.
 - 2.3 Calculate $r^{k+1} = F(x^{k+1})$. If r^{k+1} is small enough, stop.
 - 2.4 Calculate $(s^k)^T s^k$ and update B^{k+1} by

$$B_i^{k+1} = B_i^k + \frac{r_i^{k+1} (s_i^k)^T}{(s^k)^T s^k} . \quad (4)$$

Then set $k = k + 1$, and go to step 2.

3.2 Implementation

The first step in Algorithm 3.1.1 is the preparation stage. The second step is an iterative process. We update the solution x^{k+1} according to step 2.2, and calculate the residual according to step 2.3. If the residual is smaller than a given value or the number of iterations is larger than an upper bound value, the iteration process is stopped. Otherwise, the inner product of s^k is calculated and B^{k+1} is updated to repeat the iteration. Fig.1 describes the computing flow.

Fig. 1. The computing flow of BBP method

3.3 Time Complexity

In Algorithm 3.1.1, let $L(n_i)$ be the complexity of solving each block linear equations

$B_i^k s_i^k = -r_i^k$, then the complexity of step 2.1 is $L(n) = \sum_{i=1}^q L(n_i)$. Step 2.2 needs

n additions and the complexity is $n = \sum_{i=1}^q n_i$. The residual is calculated in step 2.3,

and let $R(n)$ be the corresponding calculative cost. The complexity of step 2.4 is

$\sum_{i=1}^q (3n_i - 1 + 2n_i^2)$. Therefore the total complexity of Algorithm 3.1.1 is:

$$U = \sum_{i=1}^q (L(n_i) + n_i + 3n_i - 1 + 2n_i^2) + R(n) = \sum_{i=1}^q (4n_i - 1 + 2n_i^2) + L(n) + R(n). \quad (5)$$

We can know that from (5) that the total complexity U differs in $L(n)$, which means the complexity of solving q block linear equations $B_i^k s_i^k = -r_i^k$. We will calculate $L(n)$ for unpreconditioned method and SSOR preconditioning method respectively.

For unpreconditioned method, we simply use Jacobi method to solve each block linear system. So $L(n_i)$ is the complexity of jacobi, which is (number of steps) $\times$ (cost per step) $= k_i \times J_n$. It can be easily known that

$$J_n = (2\bar{n}^{-3} + \bar{n}^{-2} + \bar{n}). \quad (6)$$

So we can calculate $L(n)$ as follows:

$$L_n(n) = \sum_{i=1}^q k_i \times (2\bar{n}^{-3} + \bar{n}^{-2} + \bar{n}) = k_n \times J_n. \quad (7)$$

where $k_n = \sum_{i=1}^q k_i$ refers to the addition of number of iterations in each block using unpreconditioned method.

For SSOR preconditioner combined with Jacobi method, $L(n_i)$ is the computational cost of construct the preconditioner plus the complexity of Jacobi, then we can get $L(n)$ as follows:

$$L_p(n) = \sum_{i=1}^q L(n_i) = \sum_{i=1}^q (M_i + k_i \times J_n) = q \times \bar{M} + k_p \times J_n. \quad (8)$$

where $k_p = \sum_{i=1}^q k_i$ refers to the addition of number of iterations in each block for SSOR method, and $\bar{M}$ means the complexity of constructing SSOR Preconditioner. $\bar{M}$ can be easily known as follows:

$$\bar{M} = \frac{\bar{n}^{-4}}{3} + 5\bar{n}^{-3} + \frac{\bar{n}^{-2}}{6} + \frac{\bar{n}}{2}. \quad (9)$$

Now the performance of each method can be compared exactly according to (7) and (8).

4 Numerical Experiments

In the numerical experiments, we compare the convergence speed and efficiency of SSOR preconditioning method with the unpreconditioned one for solving a Bratu problem in computational physics. The programming language is C++, using double float variables to calculate the problem.

The nonlinear partial differential equation can be written as

$$\begin{cases} -\Delta u + u_x + \lambda e^u = f, & (x, y) \in \Omega = [0,1] \times [0,1] \\ u|_{\partial\Omega} = 0 \end{cases} \quad (10)$$

on the unit square Ω of R^2 with zero Dirichlet boundary conditions. It is known as the Bratu problem and has been used as a test problem by Yang in [1] and Jiang in [4]. The function f is chosen such that the solution of the discretized problem is unity. Using a linear element P , a standard finite element approximation gives a large system of nonlinear equations of size N^2 . For $\lambda \geq 0$, the system has a unique solution. In the following tests, we suppose $f=e$, $\lambda=1$ and $N=50, 80$ and 110 , giving three grids, M1, M2 and M3, with 2500, 6400 and 12100 unknowns, respectively.

Let r be the nonlinear discretized function of dimension $n = N^2$ obtained from (10), so the residual vector norm is $\|r^k\|_2$, where k is the number of nonlinear iterations. We also set an upper bound value $m=5000$. The stopping criterion used in the following tests is either $\|r^k\|_2 < 10^{-5}$ or $m > 5000$. From boundary condition we can get the initial solution $x^0 = 0$ and the initial block jacobi matrix B^0 .

When the grid is M1, we set the block number as $q_1=800$. When the grid is M2, we set $q_2=1600$. When the grid is M3, we set $q_3=1000$ and $q_4=2000$ respectively. Table 1, 2, 3, and 4 show the number of nonlinear iterations, which is denoted by “ k ”, and the sum of numbers of iterations for solving each linear block during the i -th nonlinear iteration, which is denoted by “ $k[i]$ ”.

Table 1. Comparison of the total number of iterations in M1, q_1

	SSOR	No Preconditioner
k	983	1289
$k[145]$	1467	5618
$k[440]$	880	3173

Table 2. Comparison of the total number of iterations in M2, q_2

	SSOR	No Preconditioner
k	3526	4003
$k[675]$	2941	7597
$k[1100]$	2356	7032

Table 3. Comparison of the total number of iterations in M3, q3

	SSOR	No Preconditioner
k	3987	4479
k[790]	1369	5839
k[2420]	528	5010

Table 4. Comparison of the total number of iterations in M3, q4

	SSOR	No Preconditioner
k	4013	4499
k[1200]	2417	8919
k[2400]	1100	7661

From Table 1 to Table 4, the following observations can be made: The number of nonlinear iterations is almost the same, but the number of iterations in each block is much smaller for SSOR method. This is because we do not apply any preconditioning technique to solve the nonlinear system (3), but combine it with Block Broyden Algorithm to solve each linear block, as described in step 2.1 of Algorithm 3.1.1.

However, the performance of each method should not be determined by the numbers of iterations, but be judged according to formulas given in Section 3.3. Data shown in Table 4 is used as an example. During the 2400-th iteration, the following can be known:

$$k_p = 1100, k_n = 7661, q_4 = 2000, \bar{n} = M_3 / q_4 = 12100 / 2000 = 6$$

Thus we can calculate $\bar{M}$ according to (9):

$$\bar{M} = \frac{\bar{n}^{-4}}{3} + 5\bar{n}^{-3} + \frac{\bar{n}^{-2}}{6} + \frac{\bar{n}}{2} = \frac{6^4}{3} + 5 \times 6^3 + \frac{6^2}{6} + \frac{6}{2} = 1521$$

J_n is calculated according to (6):

$$J_n = 2\bar{n}^{-3} + \bar{n}^{-2} + \bar{n} = 2 \times 6^3 + 6^2 + 6 = 474$$

The value of $L_n(n)$ for the unpreconditioned method can be calculated according to (7):

$$L_n(n) = k_n \times J_n = 7661 \times 474 = 3631314$$

The value of $L_p(n)$ for SSOR method can be calculated according to (8):

$$L_p(n) = q \times \bar{M} + k_p \times J_n = 2000 \times 1521 + 1100 \times 474 = 3563400$$

We find that $L_p(n) < L_n(n)$, which can be verified by other experimental data, so the performance of SSOR method is much better than the unpreconditioned one.

5 Conclusions

We have proposed Block Broyden Method combined with SSOR preconditioner for solving nonlinear systems arising from scientific and engineering computing. This method has faster solving speed and better performance than the unpreconditioned one. One of the reasons is that a proper preconditioner is used to transform the system, so the spectral properties of the broyden matrix is improved and quick convergence speed is gained. On the other hand, as the iterative matrix is block diagonal, the algorithm only needs to store the diagonal matrix, thus largely reduces memory space.

In future work, we expect to develop more effective preconditioning methods combined with Block Broyden Algorithm so that the preconditioned system can converge quickly and meanwhile can be constructed as easily as possible.

References

1. Yang G, Dutto L, Fortin M: Inexact block Jacobi Broyden methods for solving nonlinear systems of equations. *SIAM J on Scientific Computing* (1997) 1367-1392
2. Y. Saad, H. A. van der Vorst: Iterative solution of linear systems in the 20th century. *J Comput Appl Math* (2000) 1-33
3. Von Hagen. J, Wiesbeck.W: Physics-based preconditioner for iterative algorithms in MoM-problems. *IEEE Trans on Antennas and Propagation* (2002) 1315-1316
4. Peng Jiang, Geng Yang: Performance Analysis of Preconditioners based on Broyden Method. *Applied Mathematics and Computation* (Accepted for publication)
5. Peng Jiang, Geng Yang: A Preconditioning Method based on Broyden Algorithm. *Journal of Nanjing University of Posts and Telecommunications* (in Chinese) (Accepted for Publication)