Applied Mathematical Sciences, Vol. 4, 2010, no. 69, 3403 - 3412

An Efficient Solver for Systems of Nonlinear Equations with Singular Jacobian via Diagonal Updating

M. Y. Waziri, W. J. Leong, M. A. Hassan and M. Monsi

Department of Mathematics, Faculty of Science
Universiti Putra, Malaysia 43400 Serdang, Malaysia
waziri@math.upm.edu.my

Abstract

It is well known that the quadratic rate of convergence of Newton method for solving nonlinear equations is depends on when the Jacobian is nonsingular in the neighborhood of the solution. Disobeying this condition, i.e. the Jacobian to be singular the convergence is too slow and may even lost. In this paper, we report on design and implementation of an efficient solver for systems of nonlinear equations with singular Jacobian at a solution. Our approach is based on approximation of the Jacobian inverse into a nonsingular diagonal matrix without computing the Jacobian. The proposed algorithm is simple and straightforward to implement. We report on several numerical experiments which shows that, the proposed method is very efficient.

Mathematics Subject Classification: 65H11, 65K05

Keywords: Nonlinear equations, diagonally updating, approximation, non singular Jacobian, Inverse Jacobian

1 Introduction

Let us consider the problem of finding the solution of nonlinear equations

$$F(x) = 0, \tag{1}$$

where $F = (f_1, f_2...f_n) : R^n \to R^n$.

Assuming the mapping F has the following classical assumptions:

- F is continuously differentiable in on open neighborhood $E \subset R^n$:
- there exists $x^* \in E$ with $F(x^*) = 0$ and $F'(x^*) \neq 0$:
- F' is Lipschitz continuous at x^*.

The Prominent method for finding the solution to (1), is the Newton method. The method generates an iterative sequence $\{x_k\}$ according to the following steps:

Algorithm CNM (Newton's method)

Step 1: solve $F'(x_k)s_k = -F(x_k)$

Step 2: Update $x_{k+1} = x_k + s_k$

Step 3: Repeat 1-2 until converges

for $k = 0, 1, 2, ...$ where $F'(x_k)$ is the Jacobian matrix of F and s_k is the Newton correction. When the Jacobian matrix $F'(x^*)$ is nonsingular at a solution of (1), the method converges quadratically from any given initial guess x_0 in the neighborhood of x^* [3, 6], i.e.

$$\|x_{k+1} - x^*\| \leq h\|x_k - x^*\|^2 \tag{2}$$

for some h. This in fact makes the method to be more standard in comparison with rapidly convergent methods for solving nonlinear equations. Violating this condition, i.e. when the Jacobian is singular the convergence rate may be unsatisfactory , may even be lost [10]. Furthermore Newton method slows down when approaching a singular root, this may vanishes the possibility of convergence to x^* [4]. The nonsingular requirement of Jacobian at a solution ($F'(x^*) \neq 0$) limits to some extent the application of Newton method for solving systems of nonlinear equations. There are a lot of modifications to avoids the point in which the Jacobian is singular, however there converges are also slow, resulting from less Jacobian information at each iteration. The simple modification is fixed Newton method [5]. Fixed Newton method is given by the following stages:

Algorithm FN (Fixed Newton method)

Step 1: solve $F'(x_0)s_k = -F(x_k)$

Step 2: Update $x_{k+1} = x_k + s_k$. for $k = 0, 1, 2...$

This method overcomes both the disadvantages of Newton method(computing and storing the Jacobian in each iteration), moreover it needs to computes the Jacobian at x_0. However the method is significantly slower due to insuffi-

cient information of the Jacobian in each iterations. From the computational complexity point of view, the fixed Newton is cheaper than Newton method [5, 2]. In this paper we presents a modification of Newton method for solving nonlinear systems with singular Jacobian at the solution x^*, by approximating the Jacobian inverse into a diagonal matrix. The basic idea on our approach (inverse approximation) is to avoids the point in which the Jacobian is singular and to reduce the cost of computing and storing the Jacobian, as well as solving n linear equations in each iterations. The method proposed in this paper it has a very simple form, which will be favorable to making code and is significantly cheaper than Newton method as well as faster than both Newton and fixed Newton methods in term of CPU time and number of iterations. We organize the paper as follows: we present our propose method in Section 2. Some numerical results are reported in section 3, and finally conclusion is given in section 4.

2 An efficient solver for systems of nonlinear equations with singular Jacobian (ASSJ)

Consider Taylor expansion of $F(x)$ about x_k

$$F(x) = F(x_k) + F'(x_k)(x - x_k) + \bigcirc(\|x - x_k\|^2). \tag{3}$$

Then the incomplete Taylor series expansion of $F(x)$ is given by:

$$\hat{F}(x) = F(x_k) + F'(x_k)(x - x_k) \tag{4}$$

where $F'(x_k)$ is the Jacobian of F at x_k .
In order to incorporate more information on the Jacobian to the updating matrix ,from (4) we impose the following condition: (see [11] for details)

$$\hat{F}(x_{k+1}) = F(x_{k+1}) \tag{5}$$

where $\hat{F}(x_{k+1})$ is an approximated F evaluates at x_{k+1} .
Then (4) turns into

$$F(x_{k+1}) \approx F(x_k) + F'(x_k)(x_{k+1} - x_k). \tag{6}$$

Hence we have

$$F'(x_k)(x_{k+1} - x_k) \approx F(x_{k+1}) - F(x_k). \tag{7}$$

Multiplying (7) by $F'(x_k)^{-1}$, we have

$$(x_{k+1} - x_k) \approx F'(x_k)^{-1}(F(x_{k+1}) - F(x_k)). \tag{8}$$

We proposed the approximation of the Jacobian inverse $F'(x_k)^{-1}$ by a diagonal matrix says D_k. i.e.

$$F'(x_k)^{-1} \approx D_k \tag{9}$$

Where D_k is a given diagonal matrix , updated at each iteration. Then (8) turns to

$$x_{k+1} - x_k \approx D_k(F(x_{k+1}) - F(x_k)). \tag{10}$$

Since we require D_k to be a diagonal matrix , says $D = diag(d^1, d^2, \dots, d^n)$, we consider to let components of the vector $\frac{x_{k+1}-x_k}{F(x_{k+1})-F(x_k)}$ as the diagonal elements of D_k. From (10) it follows that Using (9),it yields

$$d_{k+1}^{(i)} = \frac{x_{k+1}^{(i)} - x_k^{(i)}}{F_i(x_{k+1}) - F_i(x_k)}, \tag{11}$$

hence,

$$D_{k+1} = diag(d_{k+1}^{(i)}), \tag{12}$$

for $i = 1, 2, \dots, n$ and $k = 0, 1, 2, \dots, n$,
where $F_i(x_{k+1})$ is the i^{th} component of the vector $F(x_{k+1})$, $F_i(x_k)$ is the i^{th} component of the vector $F(x_k)$, $x_{k+1}^{(i)}$ is the i^{th} component of the vector x_{k+1}, $x_k^{(i)}$ is the i^{th} component of the vector x_k and $d_{k+1}^{(i)}$ is the i^{th} diagonal element of D_{k+1} respectively .
we use (12) (to safeguard very small $F_i(x_{k+1}) - F_i(x_k)$) if only denominator is not equal to zero or $|F_i(x_{k+1}) - F_i(x_k)| > 10^{-8}$ for $i = 1, 2, \dots$, else set

$d_k^{(i)} = d_{k-1}^{(i)}$. Finally we presents the update for our proposed method (ASSJ) as below :

$$x_{k+1} = x_k - D_k F(x_k), \tag{13}$$

where D_k is defined by (12) for $k = 1, 2, \ldots$.

Algorithm ASSJ
Consider $F(x) : \Re^n \to \Re^n$ with the same property as (1)
step 1 : Given x_0 and $D_0 = I_n$, set $k = 0$
step 2 : Compute $F(x_k)$ and $x_{k+1} = x_k - D_k F(x_k)$ where D_k is defined by (12)
step 3 : If $\|x_{k+1} - x_k\| + \|F(x_k)\| \leq 10^{-8}$ stop. Else goto 4 for $k = 1, 2, \ldots$.
step 4 : If $\|F(x_{k+1}) - F(x_k)\| > 10^{-8}$, compute D_{k+1}, else $D_{k+1} = D_k$. Set $k := k + 1$ and go to step 2.

3 Numerical results

In this section, we present several numerical tests to illustrate the performance of our new method(ASSJ) for solving nonlinear systems with singular Jacobian at a solution. We compared ASSJ method with two prominent methods and the comparison was based on Number of iterations and CPU time in seconds. The methods are namely:
(1) ASSJ method, the method proposed in this paper.
(2) The Newton method (CNM).
(3) The fixed Newton method (FN).
The numerical experiments were accomplished using MATLAB 7.0. All the calculations were carried out in double precision computer. The stopping criterion used is:

$$\|x_{k+1} - x_k\| + \|F(x_k)\| \leq 10^{-8}.$$

We introduced the following notations: *iter*: number of iterations and Cpu: Cpu time in seconds . We present some details of the the used test problems as follows:
Problem 1 [1]. $f : R^2 \longrightarrow R^2$ is defined by

$$f(x) = \begin{cases} (x_1 - 1)^2(x_1 - x_2) \\ (x_1 - 2)^5 cos(\frac{2x_1}{x_2}) \end{cases} .$$

$x_0 = (1.5, 2.5), (2, 3)$ and $(0.5, 1.5)$ are chosen, the solution is $x^* = (1, 2)$.

Problem 2 [1]. $f : R^3 \longrightarrow R^3$ is defined by

$$f(x) = \begin{cases} (x_1 - 1)^4 e^{x_2} \\ (x_2 - 2)^5 (x_1 x_2 - 1) \\ (x_3 + 4)^6 \end{cases}.$$

$x^* = (1, 2, -4)$, $x_0 = (2, 1, -2)$ and $(1.5, 1.5, -3.0)$ are chosen .

Problem 3 [1]. $f : R^2 \longrightarrow R^2$ is defined by

$$f(x) = \begin{cases} (6x_1 - x_2)^4 \\ cos(x_1) - 1 + x_2 \end{cases}.$$

$x_0 = (-0.5, 0.5), (0.5, 0.5)$ and $(-0.5, -0.5)$ are chosen and $x^* = (0, 0)$.

Problem 4 [9]. $f : R^2 \longrightarrow R^2$ is defined by

$$f(x) = \begin{cases} x_1^2 + x_2^2 \\ x_1^2 + 3x_2 \end{cases}.$$

$x_0 = (0.5, -0.3)$ and $(0, -0.3)$ are chosen and $x^* = (0, 0)$.

Problem 5. $f : R^2 \longrightarrow R^2$ is defined by

$$f(x) = \begin{cases} e^{x_1} - 1 \\ e^{x_2} - 1 \end{cases}.$$

$x_0 = (0.5, 0.5)$ and $(-1.5, -1.5)$ are chosen and $x^* = (0, 0)$.

Problem 6. $f : R^2 \longrightarrow R^2$ is defined by

$$f(x) = \begin{cases} 5x_1^2 + cos(x_1) x_2^2 \\ x_1^2 cos(x_1 e^{x_2}) + 3x_2 \end{cases}.$$

$x_0 = (0.2, -0.1)$ and $x^* = (0, 0)$

Problem 7 [8]. $f : R^3 \longrightarrow R^3$ is defined by

$$f(x) = \begin{cases} x_1^3 - x_1 x_2 x_3 \\ x_2^2 - x_1 x_3 \\ 10 x_1 x_2 x_3 - x_1 - 0.1 \end{cases}.$$

$x_0 = (0.1, 0.5, 0.2)$ *and* $(-1, -2, 0.6)$ are chosen and $x^* = (0.1; 0.1; 0.1)$

Problem 8 [8]. $f : R^3 \longrightarrow R^3$ is defined by

$$f(x) = \begin{cases} x_1 x_3 - x_3 e^{x_1^2} + 10^{-4} \\ x_1 (x_1^2 + x_2^2) + x_2^2 (x_3 - x_2) \\ x_1 + x_3^3 \end{cases}.$$

Choose $x_0 = (3, 3, 3)$ and $x^* = (-0.99990001 \times 10^{-4}; -0.99990001 \times 10^{-4}; 0.99990001 \times 10^{-4})$.

Problem 9 [7]. $f : R^2 \longrightarrow R^2$ is defined by

$$f(x) = \begin{cases} x_1^2 + x_2^2 - x_1^3 x_2 \\ x_1^2 - 2x_2^2 + 3x_1 x_2^2 \end{cases} .$$

$x_0 = (0.02, 0.02)$ and $x^* = (0, 0)$

Problem 10 [7]. $f : R^2 \longrightarrow R^2$ is defined by

$$f(x) = \begin{cases} x_1^2 - x_2^2 \\ 3x_1^2 - 3x_2^2 \end{cases} .$$

$x_0 = (0.05, 0.04), (0.5, 0.4)$ and (-0.5,-0.4) are chosen and $x^* = (0, 0)$.

Table 1
The Numerical Results of Problems 1-10 :
number of iterations(CPU time in seconds)

problems	x_0	CNM	FN	ASSJ
1	$(1.5, 2.5)$	108(90.4503)	133(68.1724)	10(5.2416)
	$(0.5, 1.5)$	33(28.6261)	71(47.0925)	9(4.6956)
	$(0, -0.5)$	34(30.0614)	69(39.4629)	25(13.3068)
2	$(2, 1, -2)$	143(267.8698)	–	35(41.0463)
	$(1.5, 1.5, -3)$	42(84.9581)	–	9(15.5209)
3	$(-0.5, 0.5)$	27(21.3877)	–	25(13.0573)
	$(0.5, 0.5)$	26(20.4517)	–	23(14.2585)
	$(-0.5, -0.5)$	26(20.4989)	–	21(10.7796)
4	$(0.5, -.3)$	13(10.0152)	114(59.1244)	7(3.6504)
	$(0.0, -0.3)$	–	–	8(4.1808)
5	(0.5,0.5)	4(3.8064)	9(5.1012)	4(1.9812)
	(-1.5,-1.5)	7(5.5224)	–	6(3.6831)
6	(0.2,-0.1)	12(19.0298)	85(46.0203)	7(4.900)
7	(0.1,0.5,0.2)	38(58.3652)	–	22(29.4902)
	(-1,-2,0.6)	51(79.7207)	–	30(33.6210)
8	(3,3,3)	122(105.9120)	–	28(31.0832)
9	(-0.5,-0.5)	24(15.6120)	19(16.0926)	15(8.31485)
10	(2,1)	–	–	11(5.7252)
	(0.5,0.4)	–	–	7(3.6660)
	(-0.5,-0.4)	–	–	5(2.6052)

The numerical results of the three (3) methods are reported in Table 1, including their number of iterations and the CPU time in seconds. The symbol "$-$" indicates a failure due to memory requirement or/and number of iteration more than 250. From Table 1, we can easily perceive that all these methods attempted to solve the system of nonlinear equations. In particular, the ASSJ method considerably outperforms the Newton and fixed Newton method for all the tested problems, as it has the least number of iteration and CPU time, which are much less than CNM and FN methods. This is due to the low computational cost associated with the building the approximation of the inverse Jacobian as well as eliminating the needs of solving Newtonian equation in each iteration.

Moreover the numerical results are very encouraging for our method. Indeed we observe that ASSJ method is the best with 100% of success (convergence to the solution) compares with CNM method having 80% and FN method with 60% respectively.

Another advantage of ASSJ method over CNM and FN methods is the storage requirement, this is more noticeable as the dimension increases.

4 Conclusion

In this paper, we have present a modification of Newton method described in section 2 to solve nonlinear equations when the Jacobian is singular at a solution. Because the convergence of Newton method in solving such systems is unsatisfactory and may even fail. We have designed a new algorithm by approximating the Jacobian inverse directly into a diagonal matrix without computing the Jacobian at all. Resulting in avoiding the point of singularity of the Jacobian. Numerical results clearly shows that it reduces efficiently the expenses of the possible necessary storage. Among its advantageous features are that it requires low CPU time to converge due to low computational cost associated with the building the updating scheme. To this end we pronounce that ASSJ method is faster than Newton method and Fixed Newton method and significantly cheaper than Newton method respectively. An additional fact that makes the ASSJ method more attractive is that throughout the tested problems it has never failed to converge, therefore, we can declare that ASSJ method is a good alternative to Newton method and fixed Newton, particularly when the Jacobian is singular at a solution x^*.

References

[1] L.H Josee , M. Eulalia and R.M. Juan *Modified Newton's method for systems of nonlinear equations with singular Jacobian*, Compt. Appl. Math, **224** (2009), 77-83.

[2] D.W.Decker and C.T. Kelly *Broyden's for a class of problems having singular Jacobian at root*, SIAM J. of Num. Anal., **23** (1985), 566-574.

[3] J. E. Dennis "Numerical methods for unconstrained optimization and nonlinear equations," Prince-Hall, Englewood Cliffs, NJ, 1983.

[4] A. Griewank and M.R. Osborne *Analysis of Newton's method at irregular singularities* , SIAM J. of Num. Anal., **20** (1983), 747-773.

[5] K. Natasa and L. Zorna *Newton-like method with modification of the right-hand vector*,J. Math. Compt., **71** (2001), 237-250.

[6] J.M. Ortega and W.C. Rheinboldt *Iterative Solution of Nonlinear equations in several variables*, Academic press, New York, 1970.

[7] Y.Q Shen and T. J.Ypma *Newton's method for singular nonlinear equations using approximate left and right nullspaces of the Jacobian*, App. Num. Math, **54** (2005), 256-265.

[8] T.N. Grapsay and E.N. Malihoutsakit *Newton's method without direct function evaluation*, In: Proceedings of 8th Hellenic European Conference on Computer Mathematics and its Applications (HERCMA 2007), Athens, Hellas, 2007.

[9] K. Ishihara, *Iterative methods for solving Nonlinear equations with singular Jacobian matrix at the solution.*, in the Proceedings of the International Conference on Recent Advances in Computational Mathematics (ICRACM 2001), Japan, 2001.

[10] C.T. Kelley "Iterative Methods for Linear and Nonlinear Equations", SIAM, Philadelphia, PA, 1995.

[11] A. Albert and J.E.Snyman *Incomplete series expansion for function approximation*, J. struct. Multdisc. Optim., **34** (2007), 21-40

Received: June, 2010