A NEW ITERATIVE METHOD FOR THE SOLUTION OF SYSTEMS OF NONLINEAR EQUATIONS

Beny Neta

Department of Mathematical Sciences
Northern Illinois University
DeKalb, Illinois

A new quasi-Newton method for the solution of systems of nonlinear algebraic equations is introduced. This method is a generalization of a sixth-order one developed by the author for approximating the solution of one nonlinear equation. The R-order of the method is four. Numerical experiments comparing the method to Newton's show that one can save over 20% of the cost of solving a system of algebraic equations. The saving is greater when the dimension is higher or the number of iterations needed is larger.

I. INTRODUCTION

Let $F : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$ be a nonlinear mapping with both its domain and its range in the n-dimensional real linear space $\mathbb{R}^n$. In this paper we consider the numerical solution of the system of n equations in n variables

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

Two special cases of (1), in particular, are much better understood than most others - namely, the n-dimensional linear

ISBN 0-12-780650-4

systems, and the one-dimensional nonlinear equations. See Varga [10], Householder [2] and Young [14] for the first case, and Ostrowski [7], Traub [9] and Householder [3] for the second one. In recent years there has been much interest in n-dimensional variations of Newton's method, the secant method, and other classical one-dimensional iterative methods. See for example Rheinboldt [8], Ortega and Rheinboldt [6], Werner [13], Voight [11],[12], Dennis and Moré [1], and references there.

The algorithm in most common usage for problem (1) can be written as follows:

Given $\underline{x}_0$

$$\underline{x}_{k+1} = \underline{x}_k - J^{-1}(\underline{x}_k)\underline{F}(\underline{x}_k), \quad k = 0,1,2,\ldots , \tag{2}$$

where

$$J_{i\ell}(\underline{x}) = \left(\frac{\partial F_i(\underline{x})}{\partial x^\ell}\right) \text{ is the Jacobian matrix.}$$

The iteration formula (2) is certainly not the way Newton's method should be implemented on a computer. The following form is much more like an actual implementation.

(i) Given $\underline{x}_k$, $\underline{F}(\underline{x}_k)$ and $J(\underline{x}_k)$

(ii) Solve the n x n linear system

$$J(\underline{x}_k)\underline{\sigma}_k = - \underline{F}(\underline{x}_k) \tag{3}$$

for the Newton step $\underline{\sigma}_k$.

(iii) Using $\underline{\sigma}_k$ and perhaps some other values of $\underline{F}(\underline{x})$, choose $\underline{x}_{k+1}$.

(iv) Evaluate $\underline{F}(\underline{x}_{k+1})$ and test for convergence. Either terminate the computation or proceed to (v).

(v) Evaluate (or approximate) $J(\underline{x}_{k+1})$, set the counter to k+1 and return to (ii).

The traditional area of research on quasi-Newton methods are steps (iii) and (v). The reason is that for real problems, evaluations of $\underline{F}$ and J dominate the cost of solution and so it is in these steps that the potential saving is greatest.

In the next section the new method will be introduced. In section 3 the order of convergence will be established. In the last section we present some of the numerical experiments performed and compare the performance of the method to Newton's.

II. DESCRIPTION OF ALGORITHM

Let us consider the one-dimensional case for a moment. In [5] the author developed a sixth-order method to approximate the solution x* of f(x) = 0. An iteration consists of a Newton substep followed by two substeps, of "modified" Newton (i.e., using the derivative of f at the first substep instead of the current one). Given x_k solve the three equations:

$$\left\{ \begin{aligned} w_k &= x_k - \frac{f(x_k)}{f'(x_k)} \\ z_k &= w_k - \frac{f(w_k)}{f'(x_k)} \cdot \frac{f(x_k) + Af(w_k)}{f(x_k) + (A-2)f(w_k)} \\ x_{k+1} &= z_k - \frac{f(z_k)}{f'(x_k)} \cdot \frac{f(x_k) - f(w_k)}{f(x_k) - 3f(w_k)} \end{aligned} \right. \tag{1}$$

where A is a parameter. If we choose A = -1 then the correcting term in the last two substeps is the same.

For a system of n equations the algorithm will be as follows:

(i) Given $\underline{x}_k$, $\underline{F}(\underline{x}_k)$ and $J(\underline{x}_k)$

(ii) Solve

$$J(\underline{x}_k)(\underline{w}_k - \underline{x}_k) = -\underline{F}(\underline{x}_k) \tag{2}$$

for $\underline{w}_k$.

(iii) Evaluate $\underline{F}(\underline{w}_k)$ and test for convergence. Either terminate the computation or proceed to (iv).

(iv) Evaluate the entries of the diagonal matrix D

$$D_{ii}(\underline{x}_k, \underline{w}_k) = \begin{cases} \dfrac{F_i(\underline{x}_k) - F_i(\underline{w}_k)}{F_i(\underline{x}_k) - 3F_i(\underline{w}_k)}, & \text{if denominator} \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

(v) Solve

$$J(\underline{x}_k)(\underline{z}_k - \underline{w}_k) = -D(\underline{x}_k, \underline{w}_k)F(\underline{w}_k) \tag{4}$$

for $\underline{z}_k$.

(vi) Evaluate $\underline{F}(\underline{z}_k)$ and test for convergence. Either terminate the process or proceed to (vii).

(vii) Solve

$$J(\underline{x}_k)(\underline{x}_{k+1} - \underline{z}_k) = -D(\underline{x}_k, \underline{w}_k)F(\underline{z}_k) \tag{5}$$

for $\underline{x}_{k+1}$.

(viii) Evaluate $\underline{F}(\underline{x}_{k+1})$ and test for convergence. Either terminate the computation or proceed to (ix).

(ix) Evaluate $J(\underline{x}_{k+1})$, set the computer to k+1 and return to (ii).

Remark: Since the evaluation and factorization of the Jacobian J is costly, one can save by keeping the Jacobian fixed. This idea is not new. Our claim is that if one modifies the righthand side as described in steps (v) and (vii) there will be no serious reduction in the rate of convergence.

Let us now compare the number of multiplications and divisions needed in one step of this algorithm with two consecutive steps of Newton's method (both are of order four). The number N_C of multiplications needed to calculate the entries of the Jacobian is given by:

$$N_C \sim nb \tag{6}$$

where b is half the bandwidth. The number N_F of multiplications needed to factor the Jacobian is

$$N_F \sim \frac{1}{2}nb^2 \tag{7}$$

and the number N_S of multiplications needed to back solve the two systems is given by:

$$N_S \sim 2nb \ . \tag{8}$$

If J is a full matrix instead of banded, one has to replace b by n in (6) - (8). The number N_D of multiplications needed to evaluate the entries of D and multiply by F in both steps is:

$$N_D \sim 4n \ . \tag{9}$$

Note that if J is not symmetric N_C and N_F should be doubled. Thus, the total number of multiplications needed for one step of our algorithm is:

$$T_0 = nb + \frac{1}{2}nb^2 + 6nb + 4n = n(7b + \frac{1}{2}b^2 + 4). \tag{10}$$

The total number of multiplications needed for two steps of Newton's method is:

$$T_N = 2(nb + \frac{1}{2}nb^2 + 2nb) = n(8b + b^2) \ . \tag{11}$$

Clearly, T_0 is smaller than T_N if $b > 2$.

Remark: It is known that the natural extension of a procedure to higher dimensions does not preserve the order of convergence exhibited by the one-dimensional procedure (see Voigt [12]). Therefore, one cannot expect the method to have a sixth-order in either O_Q or O_R measures.

In the next section we recall some definitions of measures and prove that our algorithm has at least fourth order.

III. ORDER OF CONVERGENCE

In this section we are interested in measuring how fast the sequence $\{\underline{x}_k\}$ converges to $\underline{x}^*$ (the solution of (1.1)). We shall use two measures of the order of convergence denoted by O_Q and O_R which depend on the asymptotic convergence factors Q_p and R_p, respectively. A complete discussion of these measures may be found in Ortega and Rheinboldt [6].

<u>Definition</u>: Let F be the iterative procedure

$$F: \underline{x}_{k+1} = G(\underline{x}_k, \ldots, \underline{x}_{k-m+1}), \quad k = m-1, m, \ldots . \tag{1}$$

Let $S(F, x^*)$ denote the set of all sequences generated by an iterative procedure F with limit point $\underline{x}^*$. Then

$$Q_p(F, \underline{x}^*) = \sup \left\{ Q_p\{\underline{x}_k\} \,\middle|\, \{\underline{x}_k\} \in S(F, \underline{x}^*) \right\}, \tag{2}$$

where

$$Q_p\{\underline{x}_k\} = \begin{cases} 0 & \text{if } \underline{x}_k = \underline{x}^* \text{ for all } k \geqslant k_0 \\ \limsup\limits_{k\to\infty} \dfrac{\|\underline{x}_{k+1} - \underline{x}^*\|}{\|\underline{x}_k - \underline{x}^*\|^p}, & 1 \leqslant p < \infty, \text{if } \underline{x}_k \neq \underline{x}^* \text{ for all but finitely many } k \\ \infty & \text{otherwise,} \end{cases} \tag{3}$$

are the <u>Q-convergence factors</u> of F at $\underline{x}^*$.

<u>Definition</u>: Let $Q_p(F, \underline{x}^*)$ be the Q-convergence factors of an iterative procedure F at $\underline{x}^*$. Then

$$O_Q = \inf \left\{ p \in [1, \infty) \,\middle|\, Q_p(F, \underline{x}^*) = \infty \right\} \tag{4}$$

is the <u>Q-order</u> of F at $\underline{x}^*$.

Definition: Let $S(F,\underline{x}^*)$ denote the set of all sequences $\{\underline{x}_k\}$ generated by an iterative procedure F with limit point $\underline{x}^*$. Then

$$R_p(F,\underline{x}^*) = \operatorname{Sup}\left\{R_p\{\underline{x}_k\} \mid \{\underline{x}_k\} \in S(F,\underline{x}^*)\right\}, \tag{5}$$

where

$$R_p\{\underline{x}_k\} = \begin{cases} \limsup\limits_{k\to\infty} \|\underline{x}_k - \underline{x}^*\|^{1/k} & \text{if } p = 1 \\ \limsup\limits_{k\to\infty} \|\underline{x}_k - \underline{x}^*\|^{1/p^k} & \text{if } p \in (1,\infty), \end{cases} \tag{6}$$

are the R-convergence factors of F at $\underline{x}^*$.

Definition: Let $R_p(F,\underline{x}^*)$ be the R-convergence factors of an iterative procedure F at $\underline{x}^*$. Then

$$O_R(F,\underline{x}^*) = \inf\left\{p \in [1,\infty) \mid R_p(F,\underline{x}^*) = 1\right\} \tag{7}$$

is the R-order of F at $\underline{x}^*$.

Definition: The mapping $F: D \subset \mathbb{R}^n \to \mathbb{R}^n$ is Frechet- (or F-) differentiable at $x \in \operatorname{int}(D)$ if there is an $A \in L(\mathbb{R}^n, \mathbb{R}^n)$ such that

$$\lim_{h\to 0} \frac{\|F(x+h) - Fx - Ah\|}{\|h\|} = 0 . \tag{8}$$

The linear operator A is denoted by $F'(x)$, and is called the F-derivative of F at x.

Theorem: Let $F: D \subset \mathbb{R}^n \to \mathbb{R}^n$ be F-differentiable in an open ball $S = S(x^*,\delta) \subset D$ and satisfy

$$\|J(\underline{x}) - J(\underline{x}^*)\| \leq \gamma\|\underline{x} - \underline{x}^*\| , \quad , \text{ for all } \underline{x} \in S.$$

Assume, further, that $F(\underline{x}^*) = 0$ and $J(\underline{x}^*)$ is nonsingular. Then $\underline{x}^*$ is a point of attraction of the iteration F defined by

$$\underline{x}_{k+1} = H\underline{x}_k \tag{9}$$

where

$$\begin{cases} N\underline{x} = \underline{x} - J^{-1}(\underline{x})\underline{F}(\underline{x}) \\ G\underline{x} = N\underline{x} - J^{-1}(\underline{x})D(\underline{x},N\underline{x})\underline{F}(N\underline{x}) \ , \\ H\underline{x} = G\underline{x} - J^{-1}(\underline{x})D(\underline{x},N\underline{x})\underline{F}(G\underline{x}) \end{cases} \tag{10}$$

and

$$O_R(F,\underline{x}^*) \geq O_Q(F,\underline{x}^*) \geq 4 \tag{11}$$

__Proof__: Since $N\underline{x} = \underline{x} - J^{-1}(\underline{x})F(\underline{x})$ it is clear that

$$\|N\underline{x} - \underline{x}^*\| \leq \eta\|\underline{x} - \underline{x}^*\|^2 \text{ on } S_1 \subset S \ . \tag{12}$$

$G\underline{x} = N\underline{x} - J^{-1}(\underline{x})D(\underline{x},N\underline{x})F(N\underline{x})$ is well defined on $S_2 \subset S_1$.
Therefore, if $\|J^{-1}(\underline{x})\| \leq \beta$ for all $x \in S_2$, then

$$\begin{aligned} G\underline{x} - \underline{x}^* &= N\underline{x} - \underline{x}^* - J^{-1}(\underline{x})D(\underline{x}, N\underline{x})F(N\underline{x}) = \\ &= J^{-1}(\underline{x})\{J(\underline{x})(N\underline{x} - \underline{x}^*) - D(\underline{x}, N\underline{x})F(N\underline{x})\} = \\ &= J^{-1}(\underline{x})\{-D(\underline{x}, N\underline{x})F(N\underline{x}) + F(\underline{x}^*) + J(\underline{x}^*)(N\underline{x}-\underline{x}^*) - \\ &\qquad -[J(\underline{x}^*) - J(\underline{x})](N\underline{x} - \underline{x}^*)\} \end{aligned} \tag{13}$$

$$\begin{aligned} \|G\underline{x}-\underline{x}^*\| \leq{} & \beta\|D(\underline{x},N\underline{x})F(N\underline{x}) - F(\underline{x}^*) - J(\underline{x}^*)(N\underline{x}-\underline{x}^*)\| + \\ & + \beta\gamma\|\underline{x}-\underline{x}^*\| \ \|Nx - \underline{x}^*\| \end{aligned} \tag{14}$$

In order to bound the first term on the right, let us examine the i-th component of the vector

$$A_i = D_{ii}(\underline{x}, N\underline{x})F_i(N\underline{x}) - F_i(\underline{x}^*) - \sum_{\ell=1}^{n} J_{i\ell}(\underline{x}^*)(N\underline{x} - \underline{x}^*)_\ell$$

$$\begin{aligned} D_{ii}(\underline{x},N\underline{x})F_i(N\underline{x}) &= \frac{F_i(\underline{x}) - F_i(N\underline{x})}{F_i(\underline{x}) - 3F_i(N\underline{x})} F_i(N\underline{x}) = \\ &= \left(1 + \frac{2F_i(N\underline{x})}{F_i(\underline{x}) - 3F_i(N\underline{x})}\right)F_i(N\underline{x}) = \\ &= F_i(N\underline{x}) + \frac{2F_i^2(N\underline{x})}{F_i(\underline{x}) - 3F_i(N\underline{x})} \end{aligned} \tag{15}$$

Expanding $F_i(\underline{x})$ and $F_i(N\underline{x})$ in Taylor series one obtains

$$A_i = \sum_{j=1}^{n} \sum_{\ell=1}^{n} F''_{ij\ell} \frac{(N\underline{x} - \underline{x}^*)_j (N\underline{x} - x^*)_\ell}{2} + \tag{16}$$

$$+ \frac{\sum_{j=1}^{n} J_{ij}(\underline{x}^*)(N\underline{x} - \underline{x}^*)_j + \text{H.O.T.}^2}{\sum_{j=1}^{n} J_{ij}(\underline{x}^*)(\underline{x} - \underline{x}^*)_j + \text{H.O.T.}}$$

Thus

$$\|G\underline{x} - \underline{x}^*\| \leqslant \beta\rho\|\underline{x} - \underline{x}^*\|^3 + \beta\gamma\eta\|\underline{x} - \underline{x}^*\|^3 =$$

$$= \beta(\rho + \gamma\eta)\|\underline{x} - \underline{x}^*\|^3 . \tag{17}$$

$H\underline{x} = G\underline{x} - J^{-1}(\underline{x})D(\underline{x},N\underline{x})F(G\underline{x})$ is well defined on $S_3 \subset S_2$.

$$H\underline{x} - \underline{x}^* = G\underline{x} - \underline{x}^* - J^{-1}(\underline{x})D(\underline{x},N\underline{x})F(G\underline{x}) =$$

$$= J^{-1}(\underline{x})\{J(\underline{x})[G\underline{x} - \underline{x}^*] - D(\underline{x},N\underline{x})F(G\underline{x})\} =$$

$$= J^{-1}(\underline{x})\{[-D(\underline{x},N\underline{x})F(G\underline{x}) + F(\underline{x}^*) + \tag{18}$$

$$+ J(\underline{x}^*)(G\underline{x} - \underline{x}^*)] + [J(\underline{x}^*) - J(\underline{x})](G\underline{x} - \underline{x}^*)\}$$

$$\|H\underline{x} - \underline{x}^*\| \leqslant \beta\|D(\underline{x},N\underline{x})F(G\underline{x}) - F(\underline{x}^*) - J(\underline{x}^*)(G\underline{x}-\underline{x}^*\| +$$

$$+ \beta\gamma\|G\underline{x} - \underline{x}^*\| \; \|\underline{x} - \underline{x}^*\| \tag{19}$$

The first term on the right can be bounded in a similar way to yield

$$\|D(\underline{x},N\underline{x})F(G\underline{x}) - F(\underline{x}^*) - J(\underline{x}^*)(G\underline{x} - \underline{x}^*)\| \leqslant \lambda\|\underline{x}-\underline{x}^*\|^4 \tag{20}$$

Combining (19) - (20) with (17) one obtains

$$\|H\underline{x} - \underline{x}^*\| \leqslant \beta\lambda\|\underline{x} - \underline{x}^*\|^4 + \beta^2\gamma(\rho + \gamma\eta)\|\underline{x} - \underline{x}^*\|^4 =$$
$$= \beta[\lambda + \beta\gamma(\rho + \gamma\eta)]\|\underline{x} - \underline{x}^*\|^4 \ . \qquad (21)$$

This implies that

$$O_R \geqslant O_Q \geqslant 4 \ .$$

IV. NUMERICAL EXPERIMENTS

In this section we present some of the numerical experiments and compare the performance of our algorithm to Newton's. It is clear that the saving is in calculating the entries of the Jacobian and in factoring it. Thus one cannot expect to see any difference in the performance of the two algorithms when solving systems of 2×2 or 3×3. We have compared the CPU time needed to solve 5 different systems of 2×2 and 3 different systems of 3×3. The results are summarized in Table 1.

TABLE 1

Problem No.	CPU time in seconds NETA	NEWTON
1	.68	.71
2.1	.84	.84
2.2	.76	.79
3	.68	.76
4	.82	.74
5	.81	.75
6.1	1.16	1.22
6.2	Divergence	1.05
7	1.04	Divergence
8	.73	.93

TABLE 2

Problem No.	System of Equations	Initial Value
1	$x + 3\log x - y^2 = 0$ $2x^2 - xy - 5x + 1 = 0$	(1, -2)
2	$x^2 + xy^3 - 9 = 0$ $3x^2y - y^3 - 4 = 0$	1. (1.2, 2.5) 2. (-1.2, -2.5)
3	$x + 2y - 3 = 0$ $2x^2 + y^2 - 5 = 0$	(1.5, 1)
4	$3x^2 + 4y^2 - 1 = 0$ $y^3 - 8x^3 - 1 = 0$	(-.5, .25)
5	$4x^2 + y^2 - 4 = 0$ $x + y - \sin(x-y) = 0$	(1, 0)
6	$x^5 + y^3z^4 + 1 = 0$ $x^2yz = 0$ $z^4 - 1 = 0$	1. (-1000, -1000, -1000) 2. (-100, 0, 100)
7	$x^2 + y = 37$ $x - y^2 = 5$ $x + y + z = 3$	(5, 0, -2)
8	$12x - 3y^2 - 4z = 7.17$ $x^2 + 10y - z = 11.54$ $y^3 + 7z = 7.631$	(3, 0, 1)

In Table 2 we list the systems solved and initial values used. We found one example of each where only one of the methods converged.

Note that the CPU time is for the execution step only. All numerical results were obtained on IBM 370/148 computer.

In our last experiment we consider a system of algebraic equations arising in the finite element approximation to the solution of:

$$-\nabla\left(|\nabla u(\underline{x})|^{p-2}\nabla u(\underline{x})\right) = f(\underline{x}) \quad \underline{x} \in \Omega \,, \; p \geq 2 \qquad (1)$$
$$u(\underline{x}) = 0 \qquad \underline{x} \in \partial\Omega.$$

Fix and Neta [4] showed that the finite element approximation u^h to the solution u can be written as follows

$$u^h(\underline{x}) = \sum_{i=1}^{n} u_i \, \phi_i(\underline{x}) \qquad (2)$$

where $\phi_i(\underline{x})$ are the basis functions of the finite dimensional subspace S (dim S = n). The weights u_i can be computed by solving the algebraic system of equations

$$K(\underline{u})\underline{u} = \underline{g} \,, \qquad (3)$$

where

$$K_{ij} = \int_\Omega \sum_{\ell=1}^{n} u_\ell \; \nabla\phi_\ell|^{p-2}\nabla\phi_j \cdot \nabla\phi_i \, dx \,, \qquad (4)$$

$$g_i = \int_\Omega f(\underline{x})\phi_i(\underline{x})d\underline{x}. \qquad (5)$$

The results are summarized in Table 3.

Note that the system (3) is linear when p = 2 and the saving is a result of only one less computation and factorization of the Jacobian.

It is clear from Table 3 that the saving is larger when either the dimension is higher or the number of iterations is larger.

TABLE 3

Exact Solution	P	Dimension of Matrix K	No. of Iterations NETA	No. of Iterations NEWTON	CPU time (sec) NETA	CPU time (sec) NEWTON	% Change
sin(x+y)	2	25x25, 49x49, 81x81	1	2	28.89	35.33	22.3
sin(x+y)	2	121 x 121	1	2	32.74	40.23	23
sin(x+y)	2	169 x 169	1	2	46.75	60.43	29.3
xy(1-x)(1-y)	2	25x25, 49x49, 81x81	1	2	25.80	30.97	20
xy(1-x)(1-y)	2	121 x 121	1	2	32.48	39.47	21.5
xy(1-x)(1-y)	2	169 x 169	1	2	46.10	57.15	24
x(1-x)	2	25x25. 49x49, 81x81	1	2	26.23	32.26	23
xy(1-x)(1-y)	3	25x25, 49x49, 81x81	2	4	54.40	67.60	24
xy(1-x)(1-y)	3	121 x 121	2	4	70.30	90.70	29
xy(1-x)(1-y)	3	169 x 169	2	4	91.10	119.30	31
xy(1-x)(1-y)	4	121 x 121	3	6	82.50	107.20	30
xy(1-x)(1-y)	4	169 x 169	3	7	112.40	149.50	33
sin(x+y)	3	121 x 121	2	4	70.50	93.00	32
sin(x+y)	3	169 x 169	2	5	90.60	122.30	35
sin(x+y)	4	121 x 121	3	6	80.90	108.40	34
sin(x+y)	4	169 x 169	3	7	110.70	152.80	38

REFERENCES

1. Dennis, J.E. and More, J.J., Quasi-Newton Methods, Motivation and Theory, *SIAM Rev., Vol. 19* (1977), 46-89.
2. Householder, A., *The Theory of Matrices in Numerical Analysis,* Ginn, Boston, 1964.
3. Householder, A., *The Numerical Treatment of a Single Nonlinear Equation,* McGraw-Hill, New York, 1970.
4. Fix, G.J. and Neta, B., Finite Element Approximation of a Nonlinear Diffusion Problem, *Computers and Math. with Applic. Vol. 3* (1977), 287-298.
5. Neta, B., A Sixth-Order Family of Methods for Nonlinear Equations, *Int. J. Computer Math., Vol. 7* (1979), 157-161.
6. Ortega, J. and Rheinboldt, W., *Iterative Solution of Nonlinear Equations in Several Variables,* Academic Press, New York, 1970.
7. Ostrowski, A., *Solution of Equations and Systems of Equations,* Academic Press, New York, 1966.
8. Rheinboldt, W., *Methods for Solving Systems of Nonlinear Equations,* SIAM, Phila. 1974.
9. Traub, J., *Iterative Methods for the Solution of Equations,* Prentice-Hall, Englewood Cliffs, N.J., 1964.
10. Varga, R., *Matrix Iterative Analysis,* Prentice-Hall, Englewood Cliffs, N.J., 1962.
11. Voigt, R., Rates of Convergence for a Class of Iterative Procedures, *SIAM J. Numer. Anal. Vol. 8* (1971), 127-134.
12. Voigt, R., Orders of Convergence for Iterative Procedures, *SIAM J. Numer. Anal. Vol. 8* (1971), 222-243.

13. Werner, W., Über ein Verfahren der Ordnung $1+\sqrt{2}$ zur Nullstellenbestimmung, *Numer. Math. Vol. 32* (1979), 333-342.

14. Young, D., *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.