SIAM J. OPTIMIZATION
Vol. 4, No. 2, pp. 358–382, May 1994

TRIANGULAR DECOMPOSITION METHODS FOR SOLVING REDUCIBLE NONLINEAR SYSTEMS OF EQUATIONS*

J. E. DENNIS, JR.[†], JOSÉ MARIO MARTÍNEZ[‡], AND XIAODONG ZHANG[§]

Abstract. This paper generalizes to the nonlinear case a standard way to solve general sparse systems of linear equations. In particular, Duff [*J. Inst. Math. Appl.*, 19 (1977), pp. 339–342] has suggested that row and column interchanges be applied to permute the coefficient matrix of a linear system into block lower triangular form. The linear system then can be solved by using the associated block Gauss–Seidel or forward block substitution scheme. This is the approach taken in the Harwell MA28 routine. If more than one matrix with the same sparsity pattern is to be factored, then the reordering need not be redone. In extending this approach to nonlinear problems, it is necessary to assume as in the linear case that component equations can be evaluated separately from equations in other blocks. The algorithm for doing the reordering is very fast, and if the equations can be put into block lower triangular form with each block size much less than the dimension of the system, then a large savings is in order because only the diagonal blocks need to be factored. In the nonlinear variants considered here, there are additional advantages. Not only are just the diagonal blocks of the Jacobian matrix computed and factored, but the off-diagonal partial derivatives need not even exist. Numerical tests and analytic results affirm the intuition that these variants are superior locally to Newton's method. Current work is concerned with globalizing these methods and with variants especially suited to parallel implementations.

Key words. block triangular decomposition, Gauss–Seidel–Newton method, sparse nonlinear systems of equations, parallel processing

AMS subject classifications. 65H10, 65W05

1. Introduction.

This paper will consider the problem of solving the large-scale nonlinear system of equations

$$F(x) = 0, \tag{1.1}$$

where $F : R^n \to R^n$ and the Jacobian $J(x)$ is sparse. We will assume we know the structural sparsity pattern of the Jacobian matrix $J(x^k)$. By this we mean that we will assume we have encoded which components of x affect the value of each component f_i of F. We admit that there may be some incidental additional sparsity in $J(x)$ at some particular x caused by some of the partial derivatives happening to be zero at that x, but we do not attempt to exploit incidental sparsity. One way to determine structural sparsity would be to use finite differences to compute the Jacobian at some point, and assume that any exact zero in the result is a structural zero. Of course, it is possible to foil this scheme, but it is highly unlikely to fail in practice.

* Received by the editors March 16, 1992; accepted for publication (in revised form) March 16, 1993. This research was supported in part by National Science Foundation Cooperative Agreement CCR-8809615, National Science Foundation grants CCR-9008991, CCR-9047481, CCR-9102954, and DMS-8903751, Air Force Office of Scientific Research grant 89-0363, Department of Energy grant ER/25017-3, and Army Research Office grant DAAL grant 03-90-0093.

[†] Department of Computational and Applied Mathematics and Center for Research on Parallel Computation, Rice University, Houston, Texas 77251-1892.

[‡] Department of Applied Mathematics, University of Campinas, 13081 Campinas, SP, Brazil. This work was partially conducted while the author visited the Center for Research on Parallel Computation at Rice University. The work of this author was supported by FAPESP grant 90/3724/6 and by FAEP-UNICAMP.

[§] High Performance Computing and Software Laboratory, University of Texas at San Antonio, San Antonio, Texas 78249. This work was partially conducted while the author visited the Center for Research on Parallel Computation at Rice University.

We will analyze a rather large class of methods. They are decomposition methods, and they share the congenital advantages and disadvantages of this class. Among them, if a problem can be decomposed in the requisite way, it will be rather obvious that the correct choice of a method from this class would be a sensible way to attack the problem. On the other hand, the choice of a particular member of this class for a particular problem is always going to depend on some properties of the function, but we think there are many hints here as to how to make this decision. However, there are some basic assumptions we make in order to be reasonably confident that one should choose one of the methods discussed here rather than a straightforward Newton's method, which is in fact one of the methods considered here when the sparsity is sufficiently random.

The main assumption we make in order to use effectively one of the methods considered here, is that the components of F should be able to be partitioned into blocks of components that can be evaluated each at different points in a total time roughly equivalent to the evaluation of F at a single point. This is a strong assumption which is by no means always correct, but it is true for many large problems, where it is not at all unusual to have difficulty in applying library versions of Newton's method because they usually assume that the user at least will furnish a routine that accepts x and returns $F(x)$. In many cases, the user assembles and encodes the residual calculations in blocks and never thinks in terms of computing the entire vector $F(x)$ at one time. To make things simple, we make our presentation here as if each component could constitute such a block.

The second assumption is that Tarjan's algorithm [16] applied in the usual way Duff [6] to the sparsity structure of the Jacobian $F'(x)$ would result in a set of row and column permutations P and Q that would give $PF'(x)Q$ a nontrivial block lower triangular structure. This actually goes somewhat with the first assumption in practice, but when the decomposition gives the entire Jacobian as a single block, then all the methods given here reduce to Newton's method with various strategies for when to reevaluate the Jacobian.

In large engineering systems, these properties are very common. To see this, consider the simulation of a large system of roughly sequential processes like a chemical plant. It is standard in chemical engineering to have a library of equation models of component processes (distillation columns, catalytic crackers, etc.) and, for a specific design problem, to pull these off the shelf and connect them by additional equations that set the outputs from a certain process to the inputs to its daughter processes. Of course, there can be feedback loops that complicate the Jacobian structure, but that is dealt with cheaply by Tarjan's algorithm, and the resulting block lower triangular decomposition of such structures generally is found to be very useful in solving for the Newton step.

The last difficulty we mention is again common to decomposition methods. These methods generally are more difficult to implement in such a way as to ensure convergence from poor initial guesses. For example, no one has yet published a convincing way to globalize any of the Brown–Brent methods. And yet, Brown–Brent methods and Gauss–Seidel methods, which include the ones proposed here, are used regularly in practice. There are at least two reasons for this. The first is that these decomposition methods seem to be able to converge unmodified from worse initial guesses than methods that make more superficial use of structure. (See Moré and Cosnard [12].) The second is that for many applications there are very good initial guesses available as a matter of course. Still, we are investigating ways to extend the region of convergence for these methods.

2. Newton's method. The standard Newton's method for (1.1) generates successive estimates $x^k \in R^n$ of a solution $x^* \in R^n$ of (1.1) as:

$$\text{(2.1)} \quad \text{Solve} \quad J(x^k)s^N = -F(x^k) \quad \text{and set} \quad x^{k+1} = x^k + s^N, \qquad k = 0, 1, \ldots.$$

In the case of large sparse nonlinear problems, a single standard method, such as the Newton's method, may not handle all the instances of (1.1) efficiently, but rather the algorithm must take into account the sparsity structure and other special characteristics of the problem. Our purpose here is to suggest one way to use existing technology automatically to tailor methods for particular sparsity patterns.

As motivation, we will suppose we decided to use Newton's method on a general instance of (1.1). Then we probably would use a graph coloring approach to compute $J(x^k)$ efficiently (see Curtis, Powell, and Reid [4], Coleman and Moré [3], and Dennis and Schnabel [5]). After that, we might apply the Harwell code MA28 to solve the linear system (2.1). If we did so, then permutation matrices P and Q would be found such that

$$\text{(2.2)} \qquad PJ(x^k)Q(Q^Ts^N) = -PF(x^k)$$

is a block lower triangular system. Of course, the system may have only one block, but the search for the permutations is cheap enough (see Harary [10], Duff [6], and Duff, Erisman, and Reid [8]) that nothing significant is lost in trying. On the other hand, if a nontrivial block triangular structure is revealed, then the Newton step can be found by forward block substitution, which is the same as a linear block Gauss–Seidel sweep down the permuted Newton system. This would require only that the diagonal blocks be factored. Of course, the diagonal blocks of $PJ(x)Q$ are nonsingular if and only if $J(x)$ is nonsingular. For many problems, the structural sparsity of $J(x)$ stays constant across all iterations and so it is not necessary to reorder at every iteration.

The foregoing is exactly the algorithm we refer to when we compare our suggested methods to Newton's method.

In a nutshell, what we propose here is to reorder the equations and variables still only once, but *before*, rather than after, starting the nonlinear solution process. If Newton's method is implemented as outlined above, then reordering information is being computed and used in the linear inner loop, but is not being used by the nonlinear outer loop. Our purpose is to argue that this is wasteful.

In the next section, we will make precise the comparison between the Newton method defined by (2.2) and the algorithms we suggest, which are variants of block Gauss–Seidel–Newton, or block Jacobi–Newton in the terminology introduced in Ortega and Rheinboldt [14]. In §3, we outline some variants, which we analyze in §4. Section 5 contains the results of some numerical experiments that show the advantage of our approach. Section 6 is devoted to a discussion and summary of these results and where they seem to point. The proofs of the convergence results are included in the Appendix.

3. Newton–Gauss–Seidel vs. Gauss–Seidel–Newton. It will simplify notation greatly if we assume for the remainder of this paper that the permutations have been absorbed into the problem. Thus, when we write x, we really mean Q^Tx; when we write $F(x)$, we really mean $PF(Q^Tx)$; and $J(x)$ is really the block lower triangular matrix $PF'(x)Q$. Thus, if we apply Newton's method to solving this incarnation of

(1.1) given by

$$F(x) = \begin{cases} F_1(x_1), \\ F_2(x_1,\ x_2), \\ \vdots \\ F_m(x_1,\ x_2, \dots,\ x_m), \end{cases} \tag{3.1}$$

where

$$x = (x_1,\ x_2, \dots,\ x_m)^T \in R^{n_1} \times R^{n_2} \times \cdots \times R^{n_m} = R^n,$$

$$F_i : R^{n_1} \times R^{n_2} \times \cdots \times R^{n_i} \to R^{n_i}, \qquad i = 1,\ 2, \dots,\ m,$$

and

$$\sum_{i=1}^{m} n_i = n,$$

then we have to solve

$$\begin{cases} \frac{\partial F_1(x_1^k)}{\partial x_1} s_1^N = -F_1(x_1^k), \\ \frac{\partial F_2(x_1^k, x_2^k)}{\partial x_1} s_1^N + \frac{\partial F_2(x_1^k,\ x_2^k)}{\partial x_2} s_2^N = -F_2(x_1^k, x_2^k), \\ \vdots \\ \frac{\partial F_m(x_1^k, x_2^k, \dots, x_m^k)}{\partial x_1} s_1^N + \frac{\partial F_m(x_1^k,\ x_2^k, \dots, x_m^k)}{\partial x_2} s_2^N + \cdots + \frac{\partial F_m(x_1^k,\ x_2^k, \dots, x_m^k)}{\partial x_m} s_m^N \\ \quad = -F_m(x_1^k,\ x_2^k, \dots, x_m^k) \end{cases} \tag{3.2}$$

and set $x_i^{k+1} = x_i^k + s_i^N$ for $i = 1, \dots, m$ and $k = 0, 1, 2, \dots$.

We should use forward block substitution, which is the same as the block Gauss–Seidel linear iteration, to compute the Newton step by

$$\frac{\partial F_1(x_1^k)}{\partial x_1} s_1^N = -F_1(x_1^k), \tag{3.3}$$

and for $i = 2, \dots, m$, s_i^N comes from the $n_i \times n_i$ linear system

$$\frac{\partial F_i(x_1^k, \dots, x_i^k)}{\partial x_i} s_i^N = -\left(F_i(x_1^k, \dots, x_i^k) + \sum_{j=1}^{i-1} \frac{\partial F_i(x_1^k, \dots, x_i^k)}{\partial x_j} s_j^N \right). \tag{3.4}$$

But now the key observation is that the right sides of (3.4) are just the first-order Taylor approximations to $-F_i(x_1^k + s_1^N, x_2^k + s_2^N, \dots, x_{i-1}^k + s_{i-1}^N, x_i^k)$. Thus, if we can compute values of F_i independently, it must surely be better to do so than to build a Taylor approximation to it at the expense of computing the strict lower triangular part of the Jacobian. This leads us to suggest the block version of the Gauss–Seidel–Newton method given by

$$\frac{\partial F_i(x_1^k, \dots, x_i^k)}{\partial x_i} s_i = -F_i(x_1^k + s_1, x_2^k + s_2, \dots, x_{i-1}^k + s_{i-1}, x_i^k), \tag{3.5}$$

where $x_i^{k+1} = x_i^k + s_i$, $i = 1, \ldots, m$ and $k = 0, 1, 2, \ldots$.

It is worth noting here that the fact that the lower triangular parts of the Jacobian do not appear in the algorithm has the effect that the local quadratic convergence analysis given below does not require the lower triangular parts of the Jacobian even to exist. In contrast, for Newton's method, they must exist and be smooth so that the Taylor approximation (3.4) is sufficiently accurate not to impede convergence.

Finally, let us set some more terminology. The methods fit the general framework of p. 214 of Ortega and Rheinboldt [14]. Our methods are only for the block lower triangular case. But, we were unable to find a naming scheme to reflect that which did not eventually get out of hand when we applied it to the variants given in the next section. Thus, we will base our notation on Ortega and Rheinboldt [14], but we will not use the word "block" in any of the names, since it would have exactly the same obvious meaning in all the names. Notice that for (3.1), Newton's method and the Newton–Gauss–Seidel method are both given by (3.4), and so they are identical. The reason for such attention to names is that things are about to get complicated in the next section.

4. Variants of nonlinear Gauss–Seidel.

In this section, we will look at some modifications of the Gauss–Seidel–Newton method given by (3.5). These will consist in using modifications of Newton's method on each block.

First note that for all these variants, besides the obvious advantages of not needing to compute the strict lower triangle of the Jacobian, it would be possible to apply only to the diagonal blocks graph coloring heuristics to compute the derivatives by finite differences or automatic differentiation. And so, based again on the assumption that the F_i can be evaluated independently and assuming also that the cost of $F(x)$ is the same as the sum of the costs of an evaluation of each F_i, one could not do worse than the result of coloring the entire graph. On a parallel machine, there are obvious possibilities for savings by considering coloring, derivative calculations, and factorizations independently for each diagonal block.

4.1. Nonlinear Gauss–Seidel.

Steward [15] and Erickson [9] suggest the limiting case of the class of methods we will consider. It is the block version of what Ortega and Rheinboldt [14] call the nonlinear Gauss–Seidel method. In this method, one solves the block systems successively starting from the top left corner and using the newest values as they are found:

$$\text{(4.1)} \qquad \text{solve } F_i(x_1^*, \ldots, x_{i-1}^*, x_i) = 0 \quad \text{for } x_i^*, \quad i = 1, 2, \ldots, m.$$

This method is defined independently of the particular solution technique applied to the blocks, but Erickson [9] used Newton's method to solve for each x_i^* for his comparisons with Newton's method (3.4). He ran experiments on a 66-variable nonlinear block triangular equation, where the sizes of the 2 diagonal blocks are 44 and 22, respectively.

The ratio he reported of the computing time between Newton's method and the nonlinear Gauss-Seidel method was 2.7. Since Erickson does not specify whether he takes advantage of the block forward substitution approach for solving the linear systems for the Newton step, and since this ratio is very close to 3, which could be estimated by $\sum_{i=1}^m n_i^3 : (\sum_{i=1}^m n_i)^3$ for the savings from doing only the block factorizations rather than the entire matrix, we are not sure how to interpret these results.

4.2. The Gauss–Seidel–Newton method. The Gauss–Seidel principle is to use new information as soon as it is available in order to achieve fast convergence, and this should be especially advantageous for block lower triangular systems.

$$(4.2)\quad \begin{aligned} &x_i^{k+1} = x_i^k - [J_i(x_i^{k,i})]^{-1} F_i(x_i^{k,i}),\\ &\text{where } x_i^{k,i} = (x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i^k), \quad J_i(\cdot) = \frac{\partial F_i(\cdot)}{\partial x_i}, \quad i = 1, \ldots, m. \end{aligned}$$

Below, we will use $x_i^{k,i}$ and J_i without further definition.

It will not surprise the reader that for all the examples we tried, this method converged in less time than Newton's method. There are several reasons for this. First, there is the probability that we will use fewer iterations because of the Gauss–Seidel principle. But even if we use the same number of iterations, we do less function work because only diagonal Jacobian blocks are evaluated rather than the whole matrix. The only savings in linear algebra required for our method compared to Newton's method is that Newton's method has to form the linear combinations of products of blocks in the strict lower triangular part of the Jacobian with already determined components of the Newton step in order to correct the right-hand side of the linear system to be solved for the next component. Test results are given in §5.

4.3. The Gauss–Seidel–Newton stationary (q-step) method. In practice, applying more than one inner iteration to each block can reduce the total number of outer sweeps and the total computing time. Of course, too many inner iterations means that one is doing nonlinear Gauss–Seidel, which has a slower convergence rate than Newton's method.

There is another very interesting issue to consider in deciding how many inner iterations to use. Suppose, for the sake of argument, that $x^0 = (x_1^0, x_2^0)$ has the property that $x_1^0 \neq x_1^*$, but $x_2^0 = x_2^*$. Then, unless one iterates on the first block until convergence, x_1^1 is not x_1^*. Thus, each inner iteration on the second block will be taking us further from x_2^* because it will be trying to converge to a solution to $F_2(x_1^1, x_2) = 0$.

This is an extreme case, but in general one can extend this way of thinking to see that when we begin iterating on a block, it is probably true that the earlier blocks have improved their variables. Therefore, we expect that as we iterate on the ith block toward a solution to $F_i(\bar{x}_{i-1}^{k+1}, x_i) = 0$, initially we will improve our approximation to x_i^*. However, if we keep doing inner iterations, that improvement will end and we will draw further from x_i^* as we continue toward the root of $F_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i) = 0$.

The experimental results of several such cases are in §5, including one case in which we diverge if we take more than two inner iterations, but not enough so that we are doing nonlinear Gauss–Seidel.

We found it best to implement the inner iterations in the cheapest way; we use a stationary Newton method in which we do a new block function evaluation at each inner iteration, but only one Jacobian evaluation and factorization per block per outer sweep. The algorithm for q inner stationary Newton steps on each block is as follows:

$$(4.3)\quad \begin{aligned} &x_i^{k(\nu+1)} = x_i^{k(\nu)} - [J_i(x_i^{k,i})]^{-1} F_i(x_i^{k(\nu),i}),\\ &\text{for } \nu = 0, \ldots, q-1, \quad i = 1, \ldots, m, \quad k = 0, 1, 2, \ldots,\\ &\text{and } x_i^{k(\nu),i} = (x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i^{k(\nu)}) \quad \text{with } x_i^{k(0)} = x_i^k, \quad x_i^{k(q)} = x_i^{k+1}. \end{aligned}$$

4.4. The modified Gauss–Seidel–Newton stationary (q-step) method. Another variation, which is well suited to parallel computation, is to evaluate and factor the diagonal blocks concurrently at x^k. We will show in the next section that this modified method and its variations are quadratically convergent. The algorithm for q inner stationary Newton steps on each block is as follows:

$$x_i^{k(\nu+1)} = x_i^{k(\nu)} - [J_i(x^k)]^{-1} F_i(x_i^{k(\nu),i}),\quad \text{for } \nu = 0,\ldots,q-1,\quad i = 1,\ldots,m,\quad k = 0,1,2,\ldots. \tag{4.4}$$

4.5. Gauss–Seidel–quasi-Newton q-step methods. The more general form for the methods we consider, as well as some others we will consider in future work, is

$$\begin{gathered} x_i^{k(\nu+1)} = x_i^{k(\nu)} - [B_i^{k(\nu)}]^{-1} F_i(x_i^{k(\nu),i}),\quad B_i^{k(\nu)} \text{ a nonsingular } n_i \times n_i \text{ matrix,} \\ \text{and } x_i^{k(\nu),i} = (x_1^{k+1},\ldots,x_{i-1}^{k+1}, x_i^{k(\nu)}) \text{ with } x_i^{k(0)} = x_i^k,\quad x_i^{k(q)} = x_i^{k+1}, \\ \text{for } \nu = 0,\ldots,q-1,\quad i = 1,\ldots,m,\quad k = 0,1,2,\ldots. \end{gathered} \tag{4.5}$$

The notation

$$\bar{x}_i = (x_1,\ldots,x_i) \in R^{n_1} \times R^{n_2} \times \cdots \times R^{n_i}, \qquad i = 1,2,\ldots,m$$

will be useful in the analysis. Thus, method (4.2) corresponds to (4.5) with

$$B_i^{k(\nu)} = J_i(x_1^{k+1},\ldots,x_{i-1}^{k+1}, x_i^k), \qquad q = 1, \tag{4.6}$$

method (4.3) is (4.5) with

$$B_i^{k(\nu)} = J_i(x_1^{k+1},\ldots,x_{i-1}^{k+1}, x_i^k), \tag{4.7}$$

and method (4.4) is (4.5) with

$$B_i^{k(\nu)} = J_i(x_1^k,\ldots,x_i^k). \tag{4.8}$$

4.6. Block Jacobi–Newton. Finally, we include the well-known Jacobi–Newton method for completeness in our test results. It is not a forward nonlinear substitution or Gauss–Seidel method.

$$x_i^{k+1} = x_i^k - [J_i(x_1^k,\ldots,x_i^k)]^{-1} F_i(\bar{x}_i^k),\quad i = 1,\ldots,m,\quad k = 1,\ldots. \tag{4.9}$$

Note that there is no inner iteration; in order to obtain x^{k+1} from x^k, one solves the m independent linear blocks. The same argument is used for all the Jacobian blocks and function blocks.

Besides the simple structure for implementation, the Jacobi method also exhibits a high degree of parallelism. Its main drawback is its slow rate of convergence. We will see this in the analytical convergence results and the experimental results in the following two sections.

5. Convergence results. In this section, we will give some convergence theorems for (4.5). At first, they may seem surprisingly strong, but this is the power of putting the system in block lower triangular form before starting the nonlinear solution process.

Let $|\cdot|$ denote an arbitrary norm defined on each R^{n_i} as well as its subordinate matrix norm. For every i and any $\bar{x}_i \in R^{n_1} \times \cdots \times R^{n_i}$, $i = 1, \ldots, m$, we use the norm

$$\|\bar{x}_i\| = \sum_{j=1}^{i} |x_j|.$$

Remember that $x = \bar{x}_m$, $x^k = \bar{x}^k_m$, $x^* = \bar{x}^*_m$. The Jacobian of F_i with respect to x_i is denoted by $J_i(\bar{x}_i)$.

For all our results we will assume: There exist $x^* \in R^n$, $\epsilon_1 > 0$ such that $F(x)$ is well defined for all x such that $\|x - x^*\| \le \epsilon_1$, and

$$F(x^*) = 0. \tag{5.1}$$

The Jacobian matrix $J_i(\bar{x}_i)$ exists for all $\bar{x}_i$ such that $\|\bar{x}_i - \bar{x}^*_i\| \le \epsilon_1$. Moreover, $J_i(\bar{x}_i)$ is continuous at $\bar{x}^*_i$ and $J_i(\bar{x}^*_i)$ is nonsingular for all $i = 1, \ldots, m$. We also assume that for some Γ

$$|J_i(\bar{x}_i)^{-1}| < \Gamma \tag{5.2}$$

for all $i = 1, \ldots, m$ and $\|\bar{x}_i - \bar{x}^*_i\| \le \epsilon_1$. There exists $L > 0$ such that

$$|J_i(\bar{x}_i) - J_i(\bar{x}^*_i)| \le L\|\bar{x}_i - \bar{x}^*_i\| \tag{5.3}$$

for all $i = 1, \ldots, m$ and $\|\bar{x}_i - \bar{x}^*_i\| \le \epsilon_1$. There exists $\alpha \ge 0$ such that

$$|F_i(x_1, \ldots, x_j, \ldots, x_i) - F_i(x_1, \ldots, x^*_j, \ldots, x_i)| \le \alpha |x_j - x^*_j| \tag{5.4}$$

for all $j \le i = 1, \ldots, m$ and $\|\bar{x}_i - \bar{x}^*_i\| \le \epsilon_1$.

LEMMA 5.1.

$$|F_i(\bar{x}_{i-1}, x_i) - F_i(\bar{x}^*_{i-1}, x_i)| \le \alpha \|\bar{x}_{i-1} - \bar{x}^*_{i-1}\| \tag{5.5}$$

for all $i = 1, \ldots, m$, $\|\bar{x}_i - \bar{x}^*_i\| \le \epsilon_1$.

Proof. This property can be easily obtained by using (5.4). □

Remark. Observe that we are not assuming differentiability of F_i with respect to variables x_j, $j \ne i$. This allows us to include under our analysis more general functions than the ones considered in the usual convergence analysis for Newton methods.

THEOREM 5.2. *Let* $r \in (0, 1)$. *There exist* ϵ_2 , $\delta > 0$ *such that if*

$$\|x^0 - x^*\| \le \epsilon_2 \tag{5.6}$$

and

$$|B_i^{k(\nu)} - J_i(\bar{x}^*_i)| \le \delta \tag{5.7}$$

for all $\nu = 0, \ldots, q-1$, $i = 1, \ldots, m$, $k = 0, 1, 2, \ldots$, *then the sequence* $\{x^k\}$ *generated by* (4.5) *is well defined, converges to* x^*, *and satisfies*

$$|x_i^{k(\nu)} - x^*_i| \le \frac{r}{m}\|\bar{x}^k_i - \bar{x}^*_i\|, \qquad \nu = 1, \ldots, q, \tag{5.8}$$

$$|x_i^{k+1} - x^*_i| \le \frac{r}{m}\|\bar{x}^k_i - \bar{x}^*_i\|, \qquad i = 1, \ldots, m, \tag{5.9}$$

and

$$\|x^{k+1} - x^*\| \leq r\|x^k - x^*\|. \tag{5.10}$$

Moreover, for all $\nu = 0, 1, \ldots, q-1$, $i = 1, \ldots, m$ *and* $k = 0, 1, 2, \ldots$, *we have*

(5.11)
$$|x_i^{k(\nu+1)} - x_i^*| \leq \Gamma(|B_i^{k(\nu)} - J_i(\bar{x}_i^*)| + L|x_i^{k(\nu)} - x_i^*|)|x_i^{k(\nu)} - x_i^*| + \Gamma\alpha\|\bar{x}_{i-1}^{k+1} - \bar{x}_{i-1}^*\|.$$

Proof. See the Appendix.

The following result applies to the Gauss–Seidel–Newton (4.3) and modified Gauss–Seidel–Newton methods (4.4) with q inner iterations.

THEOREM 5.3. *Assume that the choice of* $B_i^{k(\nu)}$ *in* (4.5) *is given by* (4.7) (*the Gauss–Seidel–Newton*) *or by* (4.8) (*the modified Gauss–Seidel–Newton*). *Then, there exists* $\epsilon_3 > 0$ *such that if*

$$\|x^0 - x^*\| \leq \epsilon_3,$$

the sequence generated by (4.5) *is well defined and converges to* x^*. *Furthermore, there exists* c_1 *such that*

$$|x_i^{k(\nu)} - x_i^*| \leq c_1\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+1} \tag{5.12}$$

so that

$$|x_i^{k+1} - x_i^*| \leq c_1\|\bar{x}_i^k - \bar{x}_i^*\|^{q+1} \tag{5.13}$$

and

$$\|x^{k+1} - x^*\| \leq c_1\|x^k - x^*\|^{q+1}, \tag{5.14}$$

where $\nu = 0, \ldots, q$, $i = 1, \ldots, m$, *and* $k = 0, 1, 2, \ldots$.

Proof. See the Appendix.

The rest of this section is dedicated to an analysis of the Jacobi–Newton method discussed in §3. Recall that an iteration of this method is defined by

$$x_i^{k+1} = x_i^k - J_i(\bar{x}_i^k)^{-1}F_i(\bar{x}_i{}^k) \tag{5.15}$$

for $i = 1, \ldots, m$. It turns out that in a sense, the method is m-step quadratically convergent.

THEOREM 5.4. There exist $\epsilon_4, c_2 > 0$ such that, if

$$\|x^0 - x^*\| \leq \epsilon_4, \tag{5.16}$$

the sequence generated by the Jacobi–Newton method (5.15) is well defined, converges to x^*, and satisfies

$$|x_i^{k+1} - x_i^*| \leq c_2(|x_i^k - x_i^*|^2 + \cdots + |x_1^{k-i+1} - x_1^*|^2) \tag{5.17}$$

for all $i = 1, \ldots, m$ and $k \geq m-1$.

Moreover, defining

$$y^k = (x_1^k, x_2^{k+1}, \ldots, x_m^{k+m-1}),$$

there exists a constant $c_3 > 0$ such that

$$\|y^{k+1} - x^*\| \leq c_3\|y^k - x^*\|^2 \tag{5.18}$$

for all $k = 0, 1, 2, \ldots$.

Proof. See the Appendix.

6. Experimental results. We generated our test problems from the Moré–Garbow–Hillstrom [13] test set. Some are polynomial and some are combinations of polynomial and trigonometric functions. Such simple functions would not favor the methods studied here because they are so cheap to differentiate. Thus, Newton's method does not have to pay much of a penalty for the extraneous derivative calculations.

In all cases, the diagonal blocks were fairly large, which again means that Newton's method does not have to do as much extraneous differentiation in computing the strict lower triangular part of the Jacobian as if the diagonal blocks were small. But the diagonal blocks were the same size, which our intuition says might favor a Gauss–Seidel method because it would maximize the average amount of new information available to each block iteration. However, all diagonal blocks of the same size would minimize the extra linear algebra needed by Newton's method for correcting the right-hand sides in computing the Newton step.

The computer we used is a single processor of Intel 80386 in the Sequent Symmetry system.

We wanted to avoid having to code Tarjan's algorithm [16] just to test our ideas, and so we used systems already in block triangular form with known solutions.

The test problems we used were generated in a simple way from some variable-dimension problems in Moré, Garbow, and Hillstrom [13]. Given any block dimension n and number of blocks m, we generated some polynomial test functions from two variable-dimension test problems as follows: Let the function $F_a : R^n \to R^n$ be the block of nonlinear equations with n unknowns $y = (y_1, y_2, \dots, y_n)$:

$$f_i(y) = y_i + \sum_{j=1}^{n} y_j - (n+1), \qquad 1 \le i \le n-1,$$

$$f_n(y) = \left(\prod_{j=1}^{n} y_j \right) - 1.$$

Let $F_b : R^n \to R^n$ be another block of nonlinear equations with n unknowns:

$$f_i(y) = (3 - 2y_i)y_i - y_{i-1} - 2y_{i+1} + 1, \qquad i = 1, \dots, n.$$

Now generate an $n \cdot m \times n \cdot m$ system in the blocks of variables $x = (x_1, x_2, \dots, x_m)$ by

$$F(x) = \begin{cases} F_1(x_1) = F_a(x_1), \\ F_2(x_1, x_2) = F_a(x_1) + F_b(x_2), \\ \vdots \\ F_{2\ell}(x_1, x_2, \dots, x_{2\ell}) = F_a(x_1) + \prod_{j=1}^{\ell-1} F_b(x_{2j})F_a(x_{2j+1}) + F_b(x_{2\ell}), \\ F_{2\ell+1}(x_1, x_2, \dots, x_{2\ell+1}) = F_a(x_1) + [\prod_{j=1}^{\ell-1} F_b(x_{2j})F_a(x_{2j+1})]F_b(x_{2\ell}) \\ \qquad\qquad + F_a(x_{2\ell+1}), \\ \vdots \\ F_m(x_1, x_2, \dots, x_m). \end{cases}$$

The second group of functions were generated from F_a, F_b, and the trigonometric system F_c with n unknowns:

$$f_i(y) = n - \sum_{j=1}^{n} \cos\, y_j + i(1 - \cos\, y_i) - \sin\, y_i, \qquad i = 1, \dots, n$$

as follows:

$$F(x) = \left\{ \begin{array}{l} F_1(x_1) = F_a(x_1), \\ F_2(x_1, x_2) = F_a(x_1) + F_b(x_2), \\ F_3(x_1, x_2, x_3) = F_a(x_1) + F_b(x_2) + F_c(x_3), \\ \vdots \\ F_{3\ell+1}(x_1, \ldots, x_{3\ell+1}) \\ \quad = F_a(x_1) + [\prod_{j=1}^{\ell-1} F_b(x_{3j-1}) F_c(x_{3j}) F_a(x_{3j+1})] F_b(x_{3\ell-1}) F_c(x_{3\ell}) \\ \qquad\qquad + F_a(x_{3\ell+1}) \\ F_{3\ell+2}(x_1, \ldots, x_{3\ell+2}) \\ \quad = F_a(x_1) + \prod_{j=1}^{\ell} F_b(x_{3j-1}) F_c(x_{3j}) F_a(x_{3j+1}) + F_b(x_{3\ell+2}) \\ F_{3\ell+3}(x_1, \ldots, x_{3\ell+3}) \\ \quad = F_a(x_1) + [\prod_{j=1}^{\ell} F_b(x_{3j-1}) F_c(x_{3j}) F_a(x_{3j+1})] F_b(x_{3\ell+2}) \\ \qquad\qquad + F_c(x_{3\ell+3}) \\ \vdots \\ F_m(x_1, x_2, \ldots, x_m) \end{array} \right\} = 0.$$

Here we report results on a single processor to demonstrate that the methods given here are very good sequential algorithms. In later work, we will test parallel variants of the Gauss–Seidel–Newton approach for systems of the form (3.1).

In all cases, to keep things simple the starting value of x^0 was reasonably close to the solution, and no globalizing strategy was used. The stopping criterion was to reduce the total function ℓ_2 norm below 10^{-12}.

First, we compare the performance of the five methods on a 600×600 polynomial block triangular system with six blocks of 100 variables (see Fig. 1).

As expected, the Jacobi–Newton method took the most time to converge. The nonlinear Gauss–Seidel method took 13 iterations for each block, and total computing time was impressively small. The Gauss–Seidel–Newton method used one iteration less than the Newton's method, but the computing time was about 27% because of less time spent in Jacobian evaluation. We do not compute the strict lower triangle, and in doing linear algebra, Newton must correct the right-hand sides in solving for the Newton step.

The Gauss–Seidel–Newton method with more than one inner iteration was the most effective one among all the methods for this problem. The experimental results show that the number of outer iterations is sharply decreased from 13 to 5 when the number of inner iterations is increased to 2. However, the number of outer iterations decreases more slowly as the inner number of iterations increases further. The optimal number of stationary inner iterations is problem dependent. Our experiments show that when $q = 4$, then $k = 3$ was needed, and this minimized the computing time for this system.

Next we tested a 1600×1600 polynomial block triangular problem with 16 blocks of 100 variables (see Fig. 2). Unfortunately, the Jacobi–Newton method could not converge to a solution from the same x^0 as the other methods used. We modified the initial value so that it was closer to a solution, and we found the solution by the Jacobi–Newton method. This group of experiments also showed that the nonlinear Gauss–Seidel method and the Gauss–Seidel–Newton with stationary inner iterations converged faster than the Newton's method.

We also tested the five methods on the second group of nonlinear block triangular problems with combined polynomial and trigonometric functions with 8 blocks of 100

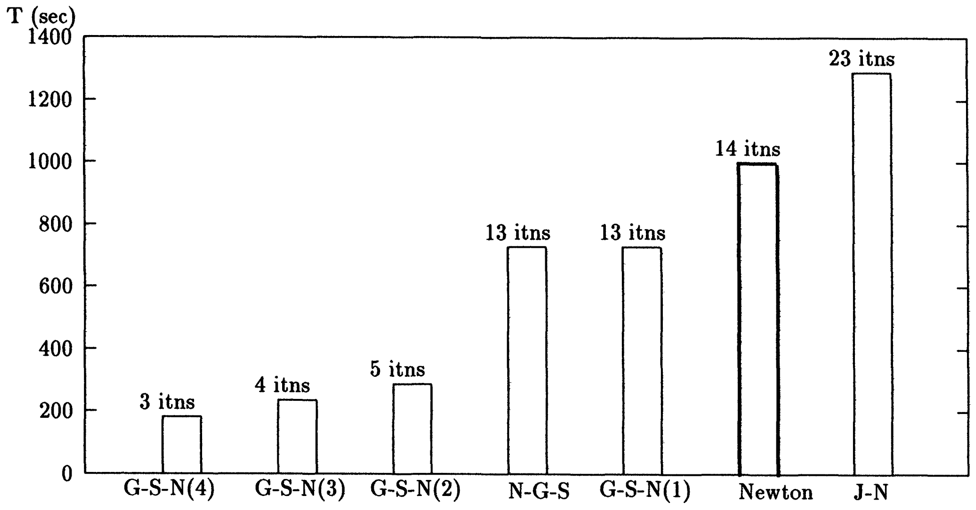

FIG. 1. *One-processor times for different methods on a* 600 × 600 *polynomial system.*

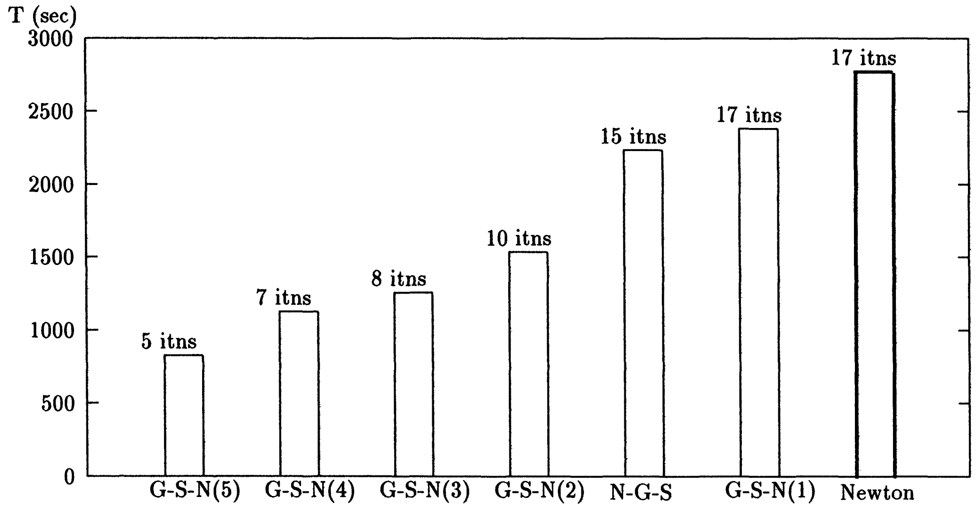

FIG. 2. *One-processor times for different methods on a* 1600 × 1600 *polynomial system.*

variables and 16 blocks of 100 variables, respectively (see Figs. 3 and 4). Performance was similar to the first tests, but the Jacobi–Newton method again failed to converge on the larger problem of the second group. Also, the Gauss–Seidel–Newton method was not as effective as before. The largest number of inner iterations successfully applied was 2, except for the nonlinear Gauss–Seidel method which in a sense has infinitely many inner iterations. When more inner iterations than 2 were applied, the method did not converge to the solution (see Figs. 3 and 4).

In general, more inner iterations should make the Gauss–Seidel–Newton method more likely to converge. But, as we discussed in §3, this is not always true. The experiments presented in Figs. 3 and 4 are examples of this point.

The best way to handle this situation is to adaptively choose the number of inner iterations for each block by some easily imagined strategies, such as reaching a target percentage reduction in the block function norms. It is not our purpose in this paper

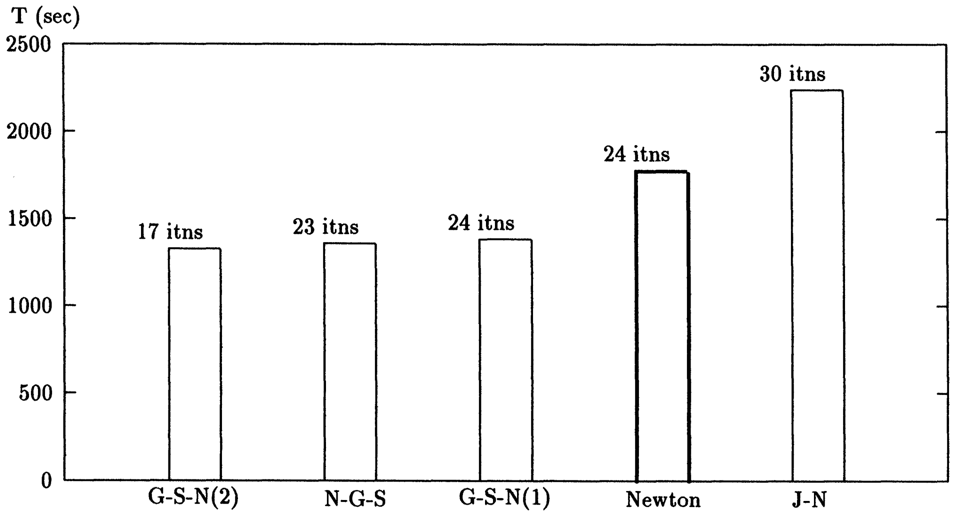

FIG. 3. *One-processor times for different methods on a* 800 × 800 *polynomial, trigonometric coupled variable function.*

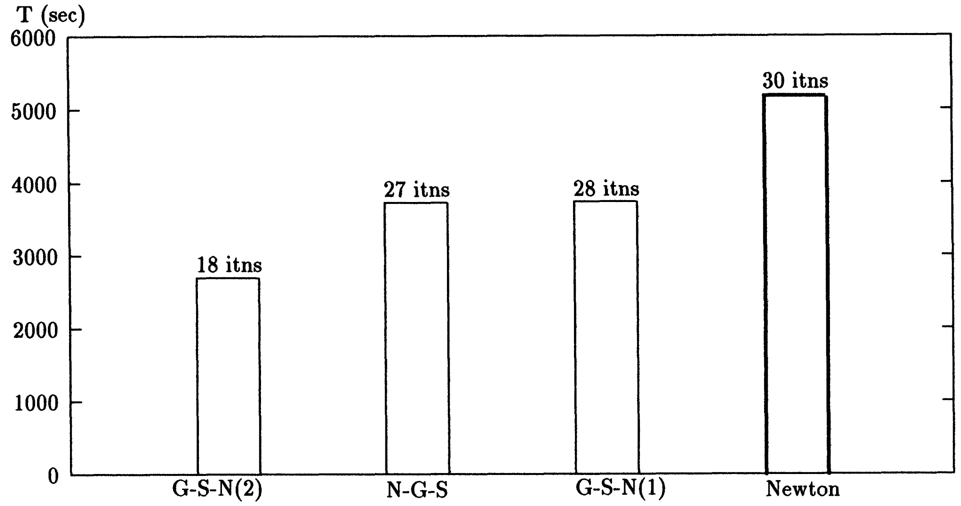

FIG. 4. *One-processor times for different methods on a* 1600 × 1600 *polynomial, trigonometric coupled variable function.*

to complicate the basic idea with such implementational details, however important to a production version they might be. But to illustrate the point, we did some experiments to show how convergence can vary with the choice of q_i.

For example, the best result we got for the 800 × 800 problem with eight blocks of 100 variables was to use in the first and second outer iterations, respectively, 1, 2, 3, 2, 2, 2, 3, 3 inner iterations. In the third outer iteration, 2 inner iterations were applied to the first block, and 3 to the other blocks. We obtained convergence with five more outer iterations, with 3 inner iterations applied to each of the blocks.

This sums to 8 total outer iterations and 1024 seconds spent, which reduced the computing time approximately 25% over the solution by the Gauss–Seidel–Newton method with the best uniform number of inner iterations. Similarly, for the 1600 × 1600 problem with 16 blocks of 100 variables, 10 outer iterations and 1995 seconds were used with varying numbers of inner iterations, which was approximately a 30% reduction in computing time over the best performance by applying a fixed number of inner iterations to each block.

7. Summary and future research. We have shown that Gauss–Seidel–Newton algorithms can be much more effective than Newton's method applied to nonlinear systems put into block triangular form by standard graph algorithms used to permute general sparse linear systems to block lower triangular form.

We have introduced the notion of allowing stationary inner iterations in the Gauss–Seidel–Newton method, and we have seen that the Jacobi–Newton method, which has the highest degree of parallelism among the methods, has the slowest convergence rate, and seems less effective.

The development of secant approximations to the diagonal blocks of a block triangular Jacobian is an attractive research topic, since it would allow cheap approximations and it would allow the factorization of the Jacobian approximation to be updated efficiently. We have done some analytical studies on quasi-Newton methods in which the block diagonals are approximated by different updates such as the Broyden update (see Broyden [1]). We plan to implement these methods on parallel computers and to test the performance.

The algorithms we considered here all have various synchronization points in them. For Jacobi and Gauss–Seidel methods, a potentially attractive alternative is to allow parallel processors to proceed asynchronously. We will develop synchronous and asynchronous parallel methods based on the sequential methods we have studied in this paper.

A. Proofs of the convergence results.

LEMMA A1. *Suppose that*

$$\{e_i^\nu,\ i = 1, \ldots, m,\ \nu = 0, 1, \ldots, q\} \tag{A.1}$$

is a set of positive real numbers such that

$$\sum_{i=1}^{m} e_i^0 \le \epsilon, \tag{A.2}$$

$$e_i^{\nu+1} \le \Gamma(\delta + L e_i^\nu) e_i^\nu + \Gamma\alpha \sum_{j=1}^{i-1} e_j^q \tag{A.3}$$

for $i = 1, \ldots, m$ *and* $\nu = 0, 1, \ldots, q-1$. *By convention* $\sum_{j=1}^{0} = 0$.

Then, there exist continuous functions $\phi_{i,\nu}(\epsilon, \delta)$, $i = 1, \ldots, m$, $\nu = 1, \ldots, q$ *such that*

$$\phi_{i,\nu}(0, 0) = 0 \tag{A.4}$$

and

$$e_i^\nu \le \phi_{i,\nu}(\epsilon, \delta) \sum_{j=1}^{i} e_j^0 \tag{A.5}$$

for all $i = 1, \ldots, m$ *and* $\nu = 1, \ldots, q$.

Proof. We proceed by induction on i. Let us first prove (A.5) for $i = 1$ by induction on ν. For $\nu = 1$, we have by (A.2) and (A.3) that

$$e_1^\nu = e_1^1 \le \Gamma(\delta + Le_1^0)e_1^0 \le \Gamma(\delta + L\epsilon)e_1^0 .$$

Therefore, (A.5) holds with

$$\phi_{1,1}(\epsilon, \delta) = \Gamma(\delta + L\epsilon) . \tag{A.6}$$

Now let us prove (A.4) and (A.5) for $i = 1$ and $\nu \in \{1, \dots, q\}$. The inductive hypothesis is that $\phi_{1,\ell}$ is defined and satisfies

$$\phi_{1,\ell}(0,0) = 0$$

and

$$e_1^\ell \le \phi_{1,\ell}(\epsilon,\delta)e_1^0$$

for $\ell = 1, \dots, \nu - 1$.

By (A.3) and the inductive hypothesis, we have that

$$\begin{aligned} e_1^\nu &\le \Gamma(\delta + Le_1^{\nu-1})e_1^{\nu-1} \\ &\le \Gamma\left(\delta + L\phi_{1,\nu-1}(\epsilon,\delta)e_1^0\right) \phi_{i,\nu-1}(\epsilon,\delta)e_1^0 \\ &\le \left[\Gamma\left(\delta + L\phi_{1,\nu-1}(\epsilon,\delta)\epsilon\right)\right] \phi_{1,\nu-1}(\epsilon,\delta)e_1^0. \end{aligned}$$

Thus (A.5) holds for $i = 1$ defining

$$\phi_{1,\nu}(\epsilon,\delta) = \Gamma(\delta + L\phi_{1,\nu-1}(\epsilon,\delta)\epsilon)\phi_{1,\nu-1}(\epsilon,\delta).$$

By the inductive hypothesis, $\phi_{1,\nu}$ is continuous and $\phi_{1,\nu}(0,0) = 0$. Therefore, (A.5) is proved for $i = 1, \nu = 1, \dots, q$.

Assume, by inductive hypothesis, that $\phi_{j,\ell}$ is defined and satisfies

$$\phi_{j,\ell}(0,0) = 0$$

and

$$e_j^\ell \le \phi_{j,\ell}(\epsilon,\delta)\sum_{s=1}^{j} e_s^0$$

for all $j \in \{1, \dots, i-1\}$, $\ell = 1, \dots, q$.

Let us prove (A.4) and (A.5) for a given $i > 1$, $\nu = 1, \dots, q$. We proceed by induction on ν. If $\nu = 1$, we have, by (A.3) and the inductive hypothesis, that

$$\begin{aligned} e_i^1 &\le \Gamma(\delta + Le_i^0)e_i^0 + \Gamma\alpha\sum_{j=1}^{i-1} e_j^q \\ &\le \Gamma(\delta + L\epsilon)e_i^0 + \Gamma\alpha\sum_{j=1}^{i-1} \phi_{j,q}(\epsilon,\delta)\sum_{\ell=1}^{j} e_\ell^0 \\ &\le \Gamma(\delta + L\epsilon)\sum_{j=1}^{i} e_j^0 + \Gamma\alpha\left(\sum_{j=1}^{i-1} \phi_{j,q}(\epsilon,\delta)\right)\left(\sum_{\ell=1}^{i-1} e_\ell^0\right) \\ &\le \left[\Gamma(\delta + L\epsilon) + \Gamma\alpha\sum_{j=1}^{i-1} \phi_{j,q}(\epsilon,\delta)\right]\sum_{\ell=1}^{i} e_\ell^0. \end{aligned}$$

Thus (A.5) holds with

$$\phi_{i,1}(\epsilon,\delta) = \Gamma(\delta + L\epsilon) + \Gamma\alpha \sum_{j=1}^{i-1} \phi_{j,q}(\epsilon,\delta) \, .$$

If $\nu > 1$, we have by (A.3) and the inductive hypothesis that

$$\begin{aligned}
e_i^\nu &\le \Gamma(\delta + Le_i^{\nu-1})e_i^{\nu-1} + \Gamma\alpha \sum_{j=1}^{i-1} e_j^q \\
&\le \Gamma\left(\delta + L\phi_{i,\nu-1}(\epsilon,\delta)\sum_{j=1}^{i} e_j^0\right)\phi_{i,\nu-1}(\epsilon,\delta)\sum_{j=1}^{i} e_j^0 \\
&\quad + \Gamma\alpha \sum_{j=1}^{i-1} \phi_{j,q}(\epsilon,\delta)\sum_{\ell=1}^{j} e_\ell^0 \\
&\le \Gamma\left(\delta + L\phi_{i,\nu-1}(\epsilon,\delta)\epsilon\right)\phi_{i,\nu-1}(\epsilon,\delta)\left(\sum_{j=1}^{i} e_j^0\right) \\
&\quad + \Gamma\alpha\left(\sum_{j=1}^{i-1} \phi_{j,q}(\epsilon,\delta)\right)\left(\sum_{j=1}^{i} e_j^0\right) \\
&\le \left[\Gamma\left(\delta + L\phi_{i,\nu-1}(\epsilon,\delta)\epsilon\right)\phi_{i,\nu-1}(\epsilon,\delta) + \Gamma\alpha\left(\sum_{j=1}^{i-1} \phi_{j,q}(\epsilon,\delta)\right)\right]\sum_{j=1}^{i} e_j^0 .
\end{aligned}$$

Thus (A.5) holds, defining

$$\phi_{i,\nu}(\epsilon,\delta) = \Gamma(\delta + L\phi_{i,\nu-1}(\epsilon,\delta)\epsilon)\phi_{i,\nu-1}(\epsilon,\delta) + \Gamma\alpha \sum_{j=1}^{i-1} \phi_{j,q}(\epsilon,\delta) \, .$$

By the inductive hypothesis, $\phi_{i,\nu}$ is continuous and $\phi_{i,\nu}(0,0) = 0$. Therefore, the existence of the functions $\phi_{i,\nu}$ such that (A.4) and (A.5) hold is proved. □

Proof of Theorem 5.2. By (4.2), $|J_i(\bar{x}_i^*)^{-1}| < \Gamma$. Therefore, we can define $\delta_1 > 0$ so that $|B_i^{-1}| \le \Gamma$ for all $B_i \in \mathbb{R}^{n_i \times n_i}$ such that $|B_i - J_i(\bar{x}_i^*)| \le \delta_1$ for $i = 1, \dots, m$.

Let ϵ_2, δ be such that $0 < \epsilon_2 \le \epsilon_1$, $0 < \delta \le \delta_1$, and

$$\text{(A.7)} \qquad \phi_{i,\nu}(\epsilon_2, \delta) \le \frac{r}{m}$$

for all $i = 1, \dots, m$ and $\nu = 1, \dots, q$, where the functions $\phi_{i,\nu}$ are given by Lemma A1.

Assume (4.6) and (4.7). Let us prove inductively that

$$\text{(A.8)} \qquad \|(x_i^{0(\nu),i}) - \bar{x}_i^*\| \le \epsilon_2$$

for all $i = 1, \dots, m$, $\nu = 0, 1, \dots, q$ and that

$$\text{(A.9)} \qquad |x_i^{0(\nu)} - x_i^*| \le \frac{r}{m} \sum_{j=1}^{i} |x_j^0 - x_j^*|$$

for all $i = 1,\dots,m$, $\nu = 1,\dots,q$.

We proceed by induction on i. So, let us first prove (A.8) and (A.9) for $i = 1$. If $i = 1$ and $\nu = 0$, (A.8) is trivial. If $i = 1$ and $\nu = 1$, we have, by (4.3), (A.1), (A.6), (A.7), and (4.6), that

$$\begin{aligned}|x_1^{0(1)} - x_1^*| &= |x_1^0 - x_1^* - (B_1^{0(0)})^{-1}F_1(x_1^0)| \\ &= \left|x_1^0 - x_1^* - (B_1^{0(0)})^{-1}\left(\int_0^1 J_1\left(x_1^* + t(x_1^0 - x_1^*)\right)dt\right)(x_1^0 - x_1^*)\right| \\ &= \left|I - (B_1^{0(0)})^{-1}J_1(x_1^*)\right| |x_1^0 - x_1^*| + \Gamma L|x_1^0 - x_1^*|^2 \\ &\le \Gamma\delta|x_1^0 - x_1^*| + \Gamma L|x_1^0 - x_1^*|^2 \\ &\le \Gamma(\delta + L\epsilon_2)|x_1^0 - x_1^*| = \phi_{1,1}(\delta,\epsilon_2)|x_1^0 - x_1^*| \\ &\le \frac{r}{m}|x_1^0 - x_1^*| \le \epsilon_2 \,.\end{aligned}$$

So, (A.8) and (A.9) are proved for $i = 1$, $\nu = 1$.

Assuming that (A.8) and (A.9) are true for $i = 1$ and $\nu \in \{1,\dots,q-1\}$, let us prove these two inequalities for $i = 1$, replacing ν by $\nu + 1$.

By (4.3), (A.1), (A.6), (A.7), (4.6), (4.5), and the inductive hypothesis, we have that

$$\begin{aligned}|x_1^{0(\nu+1)} - x_1^*| &= |x_1^{0(\nu)} - x_1^* - (B_1^{0(\nu)})^{-1}F_1(x_1^{0(\nu)})| \\ &\le \left|x_1^{0(\nu)} - x_1^* - (B_1^{0(\nu)})^{-1}\left(\int_0^1 J_1(x_1^* + t(x_1^{0(\nu)} - x_1^*))dt\right)(x_1^{0(\nu)} - x_1^*)\right| \\ &\le |I - (B_1^{0(\nu)})^{-1}J_1(x_1^*)|\,|x_1^{0(\nu)} - x_1^*| \\ &\quad + \Gamma L|x_1^{0(\nu)} - x_1^*|^2 \\ &\le \Gamma\left(|B_1^{0(\nu)} - J_1(x_1^*)| + L|x_1^{0(\nu)} - x_1^*|\right)|x_1^{0(\nu)} - x_1^*| \\ &\le \Gamma(\delta + L|x_1^{0(\nu)} - x_1^*|)|x_1^{0(\nu)} - x_1^*|\end{aligned}$$

So, by (A.3), (A.5), (4.6), and (A.7),

$$|x_1^{0(\nu+1)} - x_1^*| \le \phi_{1,\nu+1}(\epsilon_2,\delta)|x_1^0 - x_1^*| \le \frac{r}{m}|x_1^0 - x_1^*| \,.$$

Therefore, (A.8) and (A.9) are proved for $i = 1$ and $\nu = 1,\dots,q$.

Let us prove now that (A.8) and (A.9) hold for an arbitrary $i \in \{2,\dots,m\}$. Assume, as the inductive hypothesis, that

$$\|(x_j^{0(\nu),j}) - \bar{x}_j^*\| \le \epsilon_2$$

for all $j = 1,\dots,i-1$, $\nu = 0,1,\dots,q$ and that

$$|x_j^{0(\nu)} - x_j^*| \le \frac{r}{m}\sum_{\ell=1}^{j}|x_\ell^0 - x_\ell^*|$$

for all $j = 1,\dots,i-1$, $\nu = 1,\dots,q$.

We first prove (A.8) for $\nu = 0$. By the inductive hypothesis and (4.6), we have that

$$\|x_i^{0,i} - \bar{x}_i^*\| = \|(\bar{x}_{i-1}^1, x_i^0) - (\bar{x}_{i-1}^*, x_i^*)\|$$

$$
\begin{aligned}
&= \|\bar{x}_{i-1}^1 - \bar{x}_{i-1}^*\| + |x_i^0 - x_i^*| \\
&= \sum_{j=1}^{i-1} |x_j^1 - x_j^*| + |x_i^0 - x_i^*| \\
&\le \sum_{j=1}^{i-1} \frac{r}{m} \sum_{\ell=1}^{j} |x_\ell^0 - x_\ell^*| + |x_i^0 - x_i^*| \\
&\le \sum_{j=1}^{i-1} \frac{r}{m} \sum_{\ell=1}^{i-1} |x_\ell^0 - x_\ell^*| + |x_i^0 - x_i^*| \\
&\le \sum_{\ell=1}^{i-1} |x_\ell^0 - x_\ell^*| \sum_{j=1}^{i-1} \frac{r}{m} + |x_i^0 - x_i^*| \\
&\le \sum_{\ell=1}^{i-1} |x_\ell^0 - x_\ell^*| + |x_i^0 - x_i^*| = \|\bar{x}_i^0 - \bar{x}_i^*\| \le \epsilon_2 \,,
\end{aligned}
$$

since $i \le m$ and $r < 1$.

Now let us prove (A.8) and (A.9), replacing ν by $\nu + 1$. The proof of (A.8) and (A.9) for $\nu = 1$ is the same as the following proof, with the obvious substitutions.

By (4.3), (A.1), (A.6), (A.7), (4.6), (4.5), and the inductive hypothesis, we have that

(A.10)

$$
\begin{aligned}
|x_i^{0(\nu+1)} - x_i^*| &= |x_i^{0(\nu)} - x_i^* - (B_i^{0(\nu)})^{-1} F_i(x_i^{0(\nu),i})| \\
&\le |x_i^{0,\nu} - x_i^* - (B_i^{0(\nu)})^{-1} F_i(\bar{x}_{i-1}^*, x_i^{0(\nu)})| \\
&\quad + \Gamma |F_i(\bar{x}_{i-1}^1, x_i^{0(\nu)}) - F_i(\bar{x}_{i-1}^*, x_i^{0(\nu)})| \\
&\le \left| x_i^{0(\nu)} - x_i^* - (B_i^{0(\nu)})^{-1} \left(\int_0^1 J_i(\bar{x}_{i-1}^*, x_i^* + t(x_i^{0(\nu)} - x_i^*)) dt \right) \right. \\
&\quad \left. (x_i^{0(\nu)} - x_i^*) \right| + \Gamma\alpha \|\bar{x}_{i-1}^1 - \bar{x}_{i-1}^*\| \\
&\le |I - (B_i^{0(\nu)})^{-1} J_i(\bar{x}_i^*)| \, |x_i^{0(\nu)} - x_i^*| \\
&\quad + \Gamma L |x_i^{0(\nu)} - x_i^*|^2 + \Gamma\alpha \|\bar{x}_{i-1}^1 - \bar{x}_{i-1}^*\| \\
&\le \Gamma \left(|B_i^{0(\nu)} - J_i(\bar{x}_i^*)| + L |x_i^{0(\nu)} - x_i^*| \right) |x_i^{0(\nu)} - x_i^*| \\
&\quad + \Gamma\alpha \|\bar{x}_{i-1}^1 - \bar{x}_{i-1}^*\| \\
&\le \Gamma(\delta + L|x_i^{0(\nu)} - x_i^*|)|x_i^{0(\nu)} - x_i^*| + \Gamma\alpha \sum_{j=1}^{i-1} |x_j^1 - x_j^*|.
\end{aligned}
$$

So, by (A.3), (A.5), (4.6), (A.7), and the inductive hypothesis,

$$
|x_i^{0(\nu+1)} - x_i^*| \le \phi_{i,\nu+1}(\epsilon_2, \delta) \sum_{j=1}^{i} |x_j^0 - x_j^*| \le \frac{r}{m} \sum_{j=1}^{i} |x_j^0 - x_j^*|. \tag{A.11}
$$

But, by the inductive hypothesis,

$$|x_j^1 - x_j^*| \le \frac{r}{m}\sum_{\ell=1}^{j}|x_\ell^0 - x_\ell^*| \le \frac{r}{m}\sum_{\ell=1}^{i}|x_\ell^0 - x_\ell^*| \tag{A.12}$$

for $j = 1, \dots, i-1$.

Adding the inequalities (A.11) and (A.12), we obtain

$$\begin{aligned}\sum_{j=1}^{i-1}|x_j^1 - x_j^*| + |x_i^{0(\nu+1)} - x_i^*| &\le \frac{(i-1)r}{m}\sum_{j=1}^{i}|x_j^0 - x_j^*| \\ &+ \frac{r}{m}\sum_{j=1}^{i}|x_j^0 - x_j^*| = \frac{ir}{m}\|\bar{x}_i^0 - \bar{x}_i^*\| \le \epsilon_2.\end{aligned} \tag{A.13}$$

With (A.11) and (A.13), the inductive proof of (A.8) and (A.9) is complete. Therefore,

$$|x_i^1 - x_i^*| \le \frac{r}{m}\sum_{j=1}^{m}|x_j^0 - x_j^*| = \frac{r}{m}\|x^0 - x^*\|$$

for all $i = 1, \dots, m$.

So,

$$\|x^1 - x^*\| \le r\|x^0 - x^*\|\,.$$

Repeating the argument used for $k = 0$ we prove that (4.8)–(4.11) holds for $k = 0, 1, 2, \dots$. This completes the proof of the theorem. □

Proof of Theorem 5.3. Let ϵ_2 and δ be as given by Theorem 5.2 for $r = \frac{1}{2}$. Let $\epsilon_3 \le \epsilon_2$ be such that

$$|J_i(\bar{x}_i) - J_i(\bar{x}_i^*)| \le \delta$$

for all $\bar{x}_i$ such that $\|\bar{x}_i - \bar{x}_i^*\| \le \epsilon_3$. With this choice, the convergence of the sequences generated by the Gauss–Seidel–Newton method and the modified Gauss–Seidel–Newton method follows from Theorem 5.2.

Let us prove by induction that there exist constants $c_{i,\nu}$ that only depend on Γ, L, ϵ_3, and α such that

$$|x_i^{k(\nu)} - x_i^*| \le c_{i,\nu}\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+1} \tag{A.14}$$

for $i = 1, \dots, m$, $\nu = 0, 1, 2, \dots, q$.

First, we prove (A.14) for the choice (3.7) of $B_i^{k(\nu)}$. We proceed by induction on i. For $i = 1$, we prove (A.14) by induction on ν.

For $\nu = 0$, (A.14) is trivial with $c_{1,\nu} = 1$. Let us make the inductive step on ν. By (5.11) and (4.6) we have

$$\begin{aligned}|x_1^{k(\nu+1)} - x_1^*| \le \Gamma(&|J_1(x_1^{k,1}) - J_1(x_1^*)| \\ &+ L|x_1^{k(\nu)} - x_1^*|)|x_1^{k(\nu)} - x_1^*|.\end{aligned}$$

So, by (5.3),

$$\begin{aligned}|x_1^{k(\nu+1)} - x_1^*| &\le \Gamma L(|x_1^k - x_1^*| + |x_1^{k(\nu)} - x_1^*|)|x_1^{k(\nu)} - x_1^*|\\ &\le \Gamma L(|x_1^k - x_1^*| + c_{1,\nu}\|x_1^k - x_1^*\|^{\nu+1})c_{1,\nu}\|x_1^k - x_1^*\|^{\nu+1}\\ &\le \Gamma L(|x_1^k - x_1^*| + c_{1,\nu}\|x_1^k - x_1^*\|^{\nu+1})c_{1,\nu}\|x_1^k - x_1^*\|^{\nu+1}\\ &\le \Gamma L(1 + c_{1,\nu}\|x_1^k - x_1^*\|^{\nu})c_{1,\nu}\|x_1^k - x_1^*\|^{\nu+2}.\end{aligned}$$

Therefore, the inductive step is complete, defining

$$c_{1,\nu+1} = \Gamma L(1 + c_{1,\nu}\epsilon_3^{\nu})c_{1,\nu}.$$

Assume now that $|x_j^{k(\nu)} - x_j^*| \le c_{j,\nu}\|\bar{x}_j^k - \bar{x}_j^*\|^{\nu+1}$ for all $j \in \{1,\dots,i-1\}, \nu \in \{0,1,2,\dots,q\}$. Let us prove that (A.14) also holds for i. We proceed by induction on ν. For $\nu = 0$, (A.14) is trivial with $c_{i,\nu} = 1$. Let us make the inductive step. By (5.11) and (4.6) we have

$$\begin{aligned}|x_i^{k(\nu+1)} - x_i^*| \le \Gamma(&|J_i(x_i^{k,i}) - J_i(\bar{x}_i^*)|\\ &+ L|x_i^{k(\nu)} - x_i^*|)|x_i^{k(\nu)} - x_i^*| + \Gamma\alpha\|\bar{x}_{i-1}^{k+1} - \bar{x}_{i-1}^*\|.\end{aligned}$$

So, by (5.3),

$$\begin{aligned}|x_i^{k(\nu+1)} - x_i^*| &\le \Gamma L(\|\bar{x}_{i-1}^{k+1} - \bar{x}_{i-1}^*\| + |x_i^k - x_i^*|\\ &\qquad + |x_i^{k(\nu)} - x_i^*|)|x_i^{k(\nu)} - x_i^*| + \Gamma\alpha\sum_{j=1}^{i-1}|x_j^{k+1} - x_j^*|\\ &\le \Gamma L\left(\sum_{j=1}^{i-1}|x_j^{k+1} - x_j^*| + |x_i^k - x_i^*| + c_{i,\nu}\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+1}\right)\\ &\qquad \times c_{i,\nu}\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+1} + \Gamma\alpha\sum_{j=1}^{i-1}|x_j^{k(q)} - x_j^*|\\ &\le \Gamma L\left(\sum_{j=1}^{i-1}c_{j,q}\|\bar{x}_j^k - \bar{x}_j^*\|^{q+1} + |x_i^k - x_i^*|\right.\\ &\qquad \left. + c_{i,\nu}\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+1}\right)c_{i,\nu}\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+1}\\ &\quad + \Gamma\alpha\sum_{j=1}^{i-1}c_{j,q}\|\bar{x}_j^k - \bar{x}_j^*\|^{q+1}\\ &\le \Gamma L\left(\sum_{j=1}^{i-1}c_{j,q}\|\bar{x}_i^k - \bar{x}_i^*\|^{q} + 1 + c_{i,\nu}\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu}\right)c_{i,\nu}\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+2}\\ &\quad + \Gamma\alpha\left(\sum_{j=1}^{i-1}c_{j,q}\right)(\|\bar{x}_i^k - \bar{x}_i^*\|^{q-(\nu+1)})\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+2}.\end{aligned}$$

Therefore, the inductive step is complete, defining

$$c_{i,\nu+1} = \Gamma L\left(\left(\sum_{j=1}^{i-1}c_{j,q}\right)\epsilon_3^{q} + 1 + c_{i,\nu}\epsilon_3^{\nu}\right)c_{i,\nu}$$

(A.15)
$$+\Gamma\alpha\left(\sum_{j=1}^{i-1} c_{j,q}\right)\epsilon_3^{q-(\nu+1)}.$$

Let us now prove (A.14) for the case where B_i^k is chosen by (4.8) (the modified Gauss–Seidel–Newton). We also proceed by induction on i. As before, let us prove (A.14) first for $i = 1$. For $i = 1$ and $\nu = 0$, (A.14) is also trivial. Let us prove the inductive step on ν.

$$\begin{aligned}
|x_1^{k(\nu+1)} - x_1^*| &\le \Gamma(|J_1(x_1^k) - J_1(x_1^*)| \\
&\quad + L|x_1^{k(\nu)} - x_1^*|)|x_1^{k(\nu)} - x_1^*| \\
&\le \Gamma L(\|x_1^k - x_1^*\| + L|x_1^{k(\nu)} - x_1^*|)|x_1^{k(\nu)} - x_1^*| \\
&\le \Gamma L(\|x_1^k - x_1^*\| + Lc_{1,\nu}\|x_1^k - x_1^*\|^{\nu+1})c_{1,\nu}\|x_1^k - x_1^*\|^{\nu+1} \\
&\le \Gamma L(1 + Lc_{1,\nu}\epsilon_3^\nu)c_{1,\nu}\|x_1^k - x_1^*\|^{\nu+2}.
\end{aligned}$$

So, in this case, we must define

(A.16)
$$c_{1,\nu+1} = \Gamma L(1 + Lc_{1,\nu}\epsilon_3^\nu)c_{i,\nu}.$$

Now let us prove (A.14) for the choice (3.8) for a generic $i \in \{1, \ldots, m\}$, assuming that $|x_j^{k(\nu)} - x_j^*| \le c_{j,\nu}\|\bar{x}_j^k - \bar{x}_j^*\|^{\nu+1}$ holds for all $j \in \{1, \ldots, i-1\}$. For $\nu = 0$, (A.14) is again trivial. For $\nu + 1$, we have that

$$\begin{aligned}
|x_i^{k(\nu+1)} - x_i^*| &\le \Gamma(|J_i(\bar{x}_i^k) - J_i(\bar{x}_i^*)| \\
&\quad + L|x_i^{k(\nu)} - x_i^*|)|x_i^{k(\nu)} - x_i^*| + \Gamma\alpha\|\bar{x}_{i-1}^{k+1} - \bar{x}_{i-1}^*\| \\
&\le \Gamma L(\|\bar{x}_i^k - \bar{x}_i^*\| + L|x_i^{k(\nu)} - x_i^*|)|x_i^{k(\nu)} - x_i^*| + \Gamma\alpha\|\bar{x}_{i-1}^{k+1} - \bar{x}_{i-1}^*\| \\
&\le \Gamma L(\|\bar{x}_i^k - \bar{x}_i^*\| + Lc_{i,\nu}\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+1})c_{i,\nu}\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+1} \\
&\quad + \Gamma\alpha\left(\sum_{j=1}^{i-1} c_{j,q}\right)\epsilon_3^{q-(\nu+1)}\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+2} \\
&\le \left(\Gamma L(1 + Lc_{i,\nu}\epsilon_3^\nu)c_{i,\nu} + \Gamma\alpha\left(\sum_{j=1}^{i-1} c_{j,q}\right)\epsilon_3^{q-(\nu+1)}\right)\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+2}.
\end{aligned}$$

So, in this case, we must define

(A.17)
$$c_{i,\nu+1} = \Gamma L(1 + Lc_{i,\nu}\epsilon_3^\nu)c_{i,\nu} + \Gamma\alpha\left(\sum_{j=1}^{i-1} c_{j,q}\right)\epsilon_3^{q-(\nu+1)}.$$

Therefore, the inductive proof is complete and (A.14) holds both for the choice (3.7) and for the choice (3.8) of B_i^k. Now, defining $c_1' = \max_{i,\nu}\{c_{i,\nu}\}$, we have that

(A.18)
$$|x_i^{k(\nu)} - x_i^*| \le c_1'\|\bar{x}_i^k - \bar{x}_i^*\|^{\nu+1}$$

for $\nu = 1, \ldots, q$, $i = 1, \ldots, m$, $k = 0, 1, 2, \ldots$. So,

(A.19)
$$|x_i^{k+1} - x_i^*| = |x_i^{k(q)} - x_i^*| \le c_1'\|\bar{x}_i^k - \bar{x}_i^*\|^{q+1} \le c_1'\|x^k - x^*\|^{q+1}$$

for $i = 1, \dots, m$. Adding the inequalities (A.19) for $i = 1, \dots, m$, we obtain

$$\|x^{k+1} - x^*\| \le mc_1' \|x^k - x^*\|^{q+1}.$$

Hence, (5.12), (5.13), and (5.14) follow, defining $c_1 = mc_1'$. This completes the proof of the theorem. □

We need an auxiliary lemma to prove Theorem 5.4.

LEMMA A2. *Let $e_i^k, i = 1, \dots, m, k = 0, 1, 2, \dots,$ $\beta_1, \beta_2, \beta_3 > 0$ be real numbers such that $e_i^k \ge 0$ for all $i = 1, \dots, m, k = 0, 1, 2, \dots,$ and*

$$e_i^{k+1} \le \beta_1 (e_i^k)^2 + (\beta_2 + \beta_3 e_i^k) \sum_{j=1}^{i-1} e_j^k. \tag{A.20}$$

Let $\epsilon > 0$. There exists $\epsilon' \in (0, \epsilon]$ such that if

$$\sum_{i=1}^{m} e_i^0 \le \epsilon' \tag{A.21}$$

we have

$$\sum_{i=1}^{m} e_i^k \le \epsilon \tag{A.22}$$

for all $k = 0, 1, 2, \dots$ and

$$\lim_{k \to \infty} e_i^k = 0 \tag{A.23}$$

for $i = 1, \dots, m$. Moreover, there exists $c > 0$ such that

$$e_i^{k+1} \le c[(e_i^k)^2 + \cdots + (e_1^{k-i+1})^2] \tag{A.24}$$

for all i, k such that $k \ge m - 1, i = 1, \dots, m$.

Proof. Define

$$w_i^k = e_i^{k+i-1} \tag{A.25}$$

for $i = 1, \dots, m$, $k = 0, 1, 2, \dots,$

$$w^k = (w_1^k, \dots, w_m^k). \tag{A.26}$$

Let us consider a fixed k. We will prove that

$$w_i^{k+\nu} \le \sum_{\lambda \in \Upsilon(i,\nu)} \phi_\lambda(w_1^k, \dots, w_i^k), \qquad \nu = 1, \dots, m, \tag{A.27}$$

where $\phi_\lambda(y_1, \dots, y_i)$ is a monomial of degree at least 2 whose coefficients are positive and independent of k, $\Upsilon(i, \nu)$ is a set of indices, and $\vartheta(i, \nu)$ is the number of monomials that occur in (A.27).

We proceed by induction on i. So, let us first prove (A.27) for $i = 1$. In fact, by (A.20), we have that

$$w_1^{k+\nu} = e_1^{k+\nu} \le \beta_1^\nu (e_1^k)^{2\nu} = \beta_1^\nu (w_1^k)^{2\nu}. \tag{A.28}$$

So, (A.27) holds for $i = 1$ with $\vartheta(1,\nu) = 1$ for all ν, and $\phi_\lambda(w_1^k) = \beta_1^\nu (w_1^k)^{2\nu}$.

Now, let us assume that $w_j^{k+\nu} \le \sum_{\lambda\in\Upsilon(j,\nu)} \phi_\lambda(w_1^k,\ldots,w_j^k)$ for all $j = 1,\ldots,i-1$, $\nu \ge 1$. Using this, we are going to prove (A.27). We proceed by induction on ν. If $\nu = 1$, we have, by (A.20) and the inductive hypothesis, that

$$\begin{aligned}
w_i^{k+1} = e_i^{k+i} &\le \beta_1(e_i^{k+i-1})^2 + (\beta_2 + \beta_3 e_i^{k+i-1})\sum_{j=1}^{i-1} e_j^{k+i-1} \\
&\le \beta_1(e_i^{k+i-1})^2 + (\beta_2 + \beta_3 e_i^{k+i-1})\left(\sum_{j=1}^{i-1} e_j^{k+i-1}\right) \\
&= \beta_1(w_i^k)^2 + (\beta_2 + \beta_3 w_i^k)\left(\sum_{j=1}^{i-1} w_j^{k+j}\right) \\
&= \beta_1(w_i^k)^2 + (\beta_2 + \beta_3 w_i^k)\left(\sum_{j=1}^{i-1}\sum_{\lambda\in\Upsilon(j,j)} \phi_\lambda(w_1^k,\ldots,w_j^k)\right).
\end{aligned}$$

The right-hand side of the last inequality has the form of the right-hand side of (A.27). So, (A.27) is proved for $\nu = 1$. Finally, assume that

$$w_j^{k+\ell} \le \sum_{\lambda\in\Upsilon(j,\ell)} \phi_\lambda(w_1^k,\ldots,w_j^k)$$

for all $j = 1,\ldots,i-1$ and for $j = i$, $\ell = 1,\ldots,\nu-1$. Then, by (A.20),

$$\begin{aligned}
w_i^{k+\nu} = e_i^{k+\nu+i-1} &\le \beta_1(e_i^{k+\nu+i-2})^2 + (\beta_2 + \beta_3 e_i^{k+\nu+i-2})\sum_{j=1}^{i-1} e_j^{k+\nu+i-2} \\
&\le \beta_1(w_i^{k+\nu-1})^2 + (\beta_2 + \beta_3 w_i^{k+\nu-1})\sum_{j=1}^{i-1} w_j^{k+\nu+j-1} \\
&\le \beta_1\left[\sum_{\lambda\in\Upsilon(i,\nu-1)} \phi_\lambda(w_1^k,\ldots,w_i^k)\right]^2 \\
&\quad + \left(\beta_2 + \beta_3 \sum_{\lambda\in\Upsilon(i,\nu-1)} \phi_\lambda(w_1^k,\ldots,w_i^k)\right)\left(\sum_{j=1}^{i-1}\sum_{\lambda\in\Upsilon(j,\nu+j-1)} \phi_\lambda(w_1^k,\ldots,w_j^k)\right) \\
&= \sum_{\lambda\in\Upsilon(i,\nu)} \phi_\lambda(w_1^k,\ldots,w_i^k)
\end{aligned}$$

for a suitable definition of $\Upsilon(i,\nu)$. From the construction, we easily see that the coefficients and indices involved in the monomials at (A.26) are independent of k.

In particular, for $\nu = 1$, we have that

$$w_i^{k+1} \le \sum_{\lambda\in\Upsilon(i,1)} \phi_\lambda(w_1^k,\ldots,w_i^k) \tag{A.29}$$

for all $i = 1,\ldots,m$, $k = 0,1,2,\ldots$. But, since $w_j^k \le (w_1^2 + \cdots + w_i^2)^{\frac{1}{2}} \le \|w^k\|_2$ for all $i = 1,\ldots,m$, $j = 1,\ldots,i$, (A.28) implies that

$$w_i^{k+1} \le (w_1^2 + \cdots + w_i^2)P(\|w^k\|_2), \tag{A.30}$$

where P is a polynomial. By (A.30) and (A.20), $\|w^0\|$ can be made as small as desired if $\|e^0\|$ is small enough. Therefore, it holds that (A.23) and (A.24) follow from (A.30) and (A.20). □

Proof of Theorem 5.4. Define

$$e_i^k = |x_i^k - x_i^*| \tag{A.31}$$

for $i = 1, \dots, m$, $k = 0, 1, 2, \dots$. Then by (5.5) and (5.3), we have, if $\|x^k - x^*\| \le \epsilon_1$, that

(A.32)

$$\begin{aligned}
e_i^{k+1} &= |x_i^{k+1} - x_i^*| = |x_i^k - x_i^* - J_i(x_1^k, \dots, x_i^k)^{-1} F_i(x_1^k, \dots, x_i^k)| \\
&\le |x_i^k - x_i^* - J_i(\bar{x}_i^k)^{-1} F_i(\bar{x}_{i-1}^*, x_i^k)| + |J_i(\bar{x}_i^k)^{-1}| \cdot |F_i(\bar{x}_{i-1}^*, x_i^k) - F_i(\bar{x}_i^k)| \\
&\le |x_i^k - x_i^* - J_i(\bar{x}_i^k)^{-1} \left(\int_0^1 J_i(\bar{x}_{i-1}^*, x_i^* + t(x_i^k - x_i^*)) dt \right) (x_i^k - x_i^*)| \\
&\quad + \alpha |J_i(\bar{x}_i^k)^{-1}| \cdot \|\bar{x}_{i-1}^k - \bar{x}_{i-1}^*\| \\
&\le \Gamma |J_i(\bar{x}_i^k) - J_i(\bar{x}_i^*)| |x_i^k - x_i^*| + \Gamma L |x_i^k - x_i^*|^2 + \alpha\Gamma \|\bar{x}_{i-1}^k - \bar{x}_{i-1}^*\| \\
&\le \Gamma L \|\bar{x}_i^k - \bar{x}_i^*\| \cdot |x_i^k - x_i^*| + \Gamma L |x_i^k - x_i^*|^2 + \alpha\Gamma \|\bar{x}_{i-1}^k - \bar{x}_{i-1}^*\| \\
&\le 2\Gamma L |x_i^k - x_i^*|^2 + (\alpha\Gamma + \Gamma L |x_i^k - x_i^*|) \sum_{j=1}^{i-1} |x_j^k - x_j^*| \\
&= 2\Gamma L (e_i^k)^2 + (\alpha\Gamma + \Gamma L e_i^k) \sum_{j=1}^{i-1} e_j^k.
\end{aligned}$$

Hence, by (A.31) and (A.32), the inequality (A.20) holds if $\|x^k - x^*\|$ is small enough. Therefore, the thesis follows by Lemma A2 using a straightforward inductive argument. □

REFERENCES

[1] C. G. Broyden, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp., 19 (1965), pp. 577–593.

[2] T. F. Coleman, B. S. Garbow, and J. J. Moré, *Software for estimating sparse Jacobian matrices*, ACM Trans. Math. Software, 11 (1984), pp. 363–378.

[3] T. F. Coleman and J. J. Moré, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. Numer. Anal., 20 (1983), pp. 187–209.

[4] A. R. Curtis, M. J. D. Powell, and J. K. Reid, *On the estimation of sparse Jacobian matrices*, J. Inst. Math. Appl., 13 (1974), pp. 117–120.

[5] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[6] I. S. Duff, *On permutations to block triangular form*, J. Inst. Math. Appl., 19 (1977), pp. 339–342.

[7] ———, MA28—*a set of Fortran subroutines for sparse unsymmetric linear equations*, Report, AERE R8730, Her Majesty's Stationery Office, London, 1977.

[8] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford Scientific Publications, 1989.

[9] J. Erickson, *A note on the decomposition of systems of sparse non-linear equations*, BIT, 16 (1976), pp. 462–465.

[10] F. Harary, *Sparse matrices and graph theory*, in Large Sparse Sets of Linear Equations, J. K. Reid, ed., Academic Press, New York, 1971.

[11] T. Lovett and S. Thakkar, *The symmetry multiprocessor system*, in Proceedings of 1988 International Conference of Parallel Processing, August 1988, CRC Press, Boca Raton, FL, pp. 303–310.

[12] J. MORÉ AND M. COSNARD, *Numerical solution of nonlinear equations*, ACM Trans. Math. Software, 5 (1979), pp. 64–85.
[13] J. MORÉ, B. S. GARBOW, AND K. E. HILLSTROM, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.
[14] J. M. ORTEGA AND W. G. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.
[15] D. V. STEWARD, *On a approach to techniques for the analysis of the structure of large systems of equations*, SIAM Rev., 4 (1962), pp. 321–342.
[16] R. E. TARJAN, *Depth-first search and linear graph algorithms*, SIAM J. Comput., 1 (1972), pp. 146–160.