Globally Convergant Parallel Algorithms for Solving Block Bordered Systems fo Nonlinear Equations

Dan Feng and Robert B. Schnable

CU-CS-633-92 December 1992

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Abstract

Block bordered systems of nonlinear equations are systems whose Jacobian matrix consists of a series of diagonal blocks, plus a border of possibly dense rows and columns. Large systems of this type occur in many applications in science and engineering, and are attractive candidates for solution on parallel computers. Recently, Zhang, Byrd, and Schnabel developed a new class of algorithms for solving such systems that has significant computational advantages over previous methods on sequential computers, and even greater advantages on parallel computers. Its main feature is that the methods perform multiple inner iterations on the diagonal blocks for each outer iteration on the overall system, in a new way that retains fast local convergence. This paper investigates whether one can develop related algorithms that retain these advantages of parallelizability and fast local convergence, and are also globally convergent and computationally robust on a broad class of problems, including those where the Jacobian matrix or any diagonal block is singular or ill-conditioned. We introduce related new algorithms for solving such systems, and show that they have strong global convergence properties under very mild assumptions, and retain parallelizability and fast local convergence.

1 Introduction

Efficient numerical methods for solving systems of nonlinear equations find wide application in science and engineering. Newton's method is the basic general purpose iterative approach for solving such systems. At each iteration, Newton's method creates a linear approximation to the nonlinear system of equations based upon the current Jacobian matrix. On parallel computers, an efficient parallel method for solving this system of linear equations becomes an important consideration [5].

In many practical problems, systems of nonlinear equations are large and have special structure. A common structure is large, "block bordered" systems of nonlinear equations. In such problems, the Jacobian matrix consists of a number of square, non-overlapping diagonal blocks, plus a final set of possibly dense rows and columns. Block bordered nonlinear equations are common in VLSI design, structural engineering, and many other areas [6, 7].

Due to their near separability, block bordered systems of nonlinear equations are an attractive candidate for parallel computation. This issue is explored in Zhang, Byrd and Schnabel[10] and Zhang[8]. An obvious approach is to parallelize Newton's method, and utilize the block bordered structure in solving the resultant block bordered system of linear equations that arises at each iteration. More parallelism can be obtained, however, if more work is done independently at each iteration on the subsets of equations corresponding to the diagonal blocks, i.e. by performing more "inner iterations" on these subsystems per each "outer iteration" on the entire system. Zhang, Byrd and Schnabel[10] showed how to perform such extra, perfectly parallelizable inner iterations at each iteration, while retaining the fast local convergence of Newton's method through a "correction" step. They also showed that the extra inner iterations improve the computational performance of the algorithms on practical problems on both sequential and parallel computers.

Two important issues that arise from this approach are how to make the inner iterations consistent with global convergence, and how to handle possible singularity or ill-conditioning of either the entire Jacobian matrix or any diagonal submatrix in a way that is consistent with global convergence and does not harm parallelizability. This paper addresses these issues. We give a new, computationally practical algorithm that addresses singularity in a way that is consistent with global convergence, parallel efficiency, and the sparsity structure of the problem. The algorithm is related to the methods of Zhang, Byrd and Schnabel[10], but incorporates a new way of dealing with singularity, and a revised way of performing the inner iterations. We show that this algorithm is globally convergent to a root or critical point of the system of nonlinear equations under very mild assumptions, namely a continuously differentiable set of equations with a Lipschitz continuous Jacobian matrix whose norm is bounded above, and that it retains the fast local convergence properties of the previous methods.

The remainder of this paper is organized as follows. Block bordered systems of nonlinear equations and current approaches to solving them are reviewed in section 2. In section 3, two new algorithms, a basic algorithm for the nonsingular case and a robust, practical algorithm for the general case, are introduced, and their local convergence properties are discussed. Then in section 4, the global convergence properties of the new algorithms are analyzed. Finally, in section 5, we make some brief comments about some possible difficulties and extensions of our algorithms.

2 Background on block bordered nonlinear problems

Consider the system of nonlinear equations

$$\begin{aligned} f_i(x_i, x_{q+1}) = 0;\ i = 1, \cdots, q \\ f_{q+1}(x_1, \cdots, x_{q+1}) = 0, \end{aligned} \tag{2.1a}$$

where

$$x_i \in \Re^{n_i},\ f_i \in \Re^{n_i},\ i = 1, \cdots, q+1, \tag{2.1b}$$

and

$$\sum_{i=1}^{q+1} n_i = n. \tag{2.1c}$$

Let

$$\begin{aligned} x \in \Re^n &= (x_1^T, \cdots, x_q^T, x_{q+1}^T)^T & (2.1d) \\ F(x) : \Re^n \to \Re^n &= (f_1^T(x), \cdots, f_q^T(x), f_{q+1}^T(x))^T. & (2.1e) \end{aligned}$$

The Jacobian matrix

$$J(x) = \begin{pmatrix} A_1 & & & & B_1 \\ & A_2 & & & B_2 \\ & & \ddots & & \vdots \\ & & & A_q & B_q \\ C_1 & C_2 & \cdots & C_q & P \end{pmatrix}, \tag{2.1f}$$

has a block bordered structure, where

$$A_i = \frac{\partial f_i(x)}{\partial x_i} \in \Re^{n_i \times n_i},\ i = 1, \cdots, q, \tag{2.1g}$$

$$B_i = \frac{\partial f_i(x)}{\partial x_{q+1}} \in \Re^{n_i \times n_{q+1}},\ i = 1, \cdots, q, \tag{2.1h}$$

$$C_i = \frac{\partial f_{q+1}(x)}{\partial x_i} \in \Re^{n_{q+1} \times n_i},\ i = 1, \cdots, q \tag{2.1i}$$

and

$$P = \frac{\partial f_{q+1}(x)}{\partial x_{q+1}} \in \Re^{n_{q+1} \times n_{q+1}}. \tag{2.1j}$$

Block bordered problems of this form arise in many areas of science and engineering including VLSI design and structural engineering. Efficient algorithms for solving linear systems of this special structure have been studied in Christara [1], Christara and Houstis [2], Farhat and Wilson [6], and Mu and Rice [7]. Zhang, Byrd and Schnabel [9, 10] considered algorithms for solving nonlinear systems of this type. They studied two types of iterative methods, explicit and implicit methods. These methods are both related to Newton's method. They both try to maintain the advantages of Newton's method while taking advantage of the sparsity of the problem, but differ in how fully they utilize the near separability of the equations.

The explicit method is obtained by simply applying Newton's method to the problem defined in (2.1a)–(2.1j), resulting in the system of linear equations

$$J(x^k)\Delta x^k = -F(x^k).$$

This block bordered system of linear equations can be solved efficiently by first factoring the diagonal blocks A_i, $i = 1, \cdots, q$, which is essentially the same as using the ith set of equations to express Δx_i in terms of Δx_{q+1} for each $1 \leq i \leq q$. Then one completes the factorization by forming and factoring the Schur complement of the bottom block,

$$\hat{J} = P - \sum_{i=1}^{q} C_i A_i^{-1} B_i, \tag{2.2}$$

after which Δx_{q+1} can be solved for. Finally, one can solve for each Δx_i, $i = 1, \cdots, q$, using the value of Δx_{q+1} and the factorization of A_i. The solution Δx^k is identical to the standard Newton step, and the method can be made globally convergent by utilizing a line search (assuming $J(x^k)$ and the diagonal block are nonsingular).

There is perfect parallelism available in two portions of the implicit method, the factorization of the q diagonal blocks A_i and the backsolves for the q Δx_i's after Δx_{q+1} is calculated. A bottleneck to parallelism exists, however, in the formation and factorization of the bottom block $\hat{J}$, and the solution for Δx_{q+1} and its communication to other processors. Therefore from the point of view of parallel computation, it would be desirable to do more work on the equations that correspond to the diagonal blocks and the x_i's, and less work on the final set of equations and x_{q+1}. The implicit method attempts to achieve this goal in a way that is beneficial to the overall speed of the method.

The basic philosophy behind the implicit approach (but not its implementation) is as follows. First each of the q systems of nonlinear equations

$$f_i(x_i, x_{q+1}) = 0, \; i = 1, \cdots, q \tag{2.3}$$

could be used to solve for x_i for each value of x_{q+1}. This would mean that each of the x_i's would be implicitly given by a function of x_{q+1}. Then the last system of equations could be viewed as a system in terms of x_{q+1} only,

$$f_{q+1}(x_1(x_{q+1}), \cdots, x_q(x_{q+1}), x_{q+1}) = 0, \tag{2.4}$$

which is solved for x_{q+1}. Note that the Jacobian matrix of (2.4) is given by (2.2).

A practical computational procedure related to this idea is to first apply several iterations of Newton's method to *approximately* solve each (2.3) for $x_i(x_{q+1})$ given the current value of x_{q+1}, then to apply one iteration of a second Newton's method to (2.4) to determine a new value of x_{q+1} using these new values of x_i, and finally to combine these results to determine a search direction for the next iterate. The first set of Newton iterations, which we call "inner iterations", can be fully parallelized (because the q calculations of the new x_i's are independent of each other), while the second Newton iteration is generally performed sequentially because the system (2.4) is generally fairly small. Thus in one overall "outer" iteration of the implicit method, one would like to apply multiple inner iterations to each (2.3) more accurately, because of the parallelism available in these calculations, hopefully in exchange for requiring fewer overall iterations, and consequently less total work, for solving the overall problem.

A subtlety involved in applying the implicit approach is that before using the step $(\Delta x_1, \cdots, \Delta x_q, \Delta x_{q+1})$ calculated by the above procedure as a search direction, a "correction" needs to be made to each Δx_i $i = 1, \cdots, q$ to account for the change in x_{q+1}. This correction, which was introduced in [9, 10], consists of adding $-A_i^{-1} B_i \Delta x_{q+1}$ to each Δx_i for $1 \leq i \leq q$. With this correction, the outer iterations of the implicit method retain the local quadratic convergence of the explicit method, whereas without the correction they do not. Also, we show in Section 3 that with one new modification to the previous corrected implicit method which does not alter its local quadratic convergence properties, the implicit method with one inner iteration per outer iteration is the same as the explicit method.

In practice, however, the advantage of the implicit method comes from using more than one inner iteration per outer iteration. In [9] and [10] it is shown that on a set of practical problems, performing several inner iterations per outer iteration reduces the number of overall (outer) iterations required to solve the problem, and the total time required to solve the problems. Since the inner iterations parallelize fully, this leads to an even larger advantage over the explicit method (i.e. the conventional Newton's method) on parallel computers.

Therefore implicit methods seem to offer a promising computational approach to solving block bordered systems of nonlinear equations, and it seems important to create versions of them that are computationally robust in the face of possible singularity or ill-conditioning of the Jacobian matrix or its diagonal blocks, and that are known to have strong global convergence properties. This has not previously been accomplished, and is the contribution of the remainder of this paper.

3 New algorithms

In this section we introduce two new implicit methods for solving block bordered systems of nonlinear equations. The first, the basic method given in Algorithm A, is the method that is used as long as the Jacobian matrix and all of its diagonal blocks are nonsingular and sufficiently well-conditioned. Presenting this algorithm first allows us to introduce several new features that are used in both new methods, in a simplified setting. The second new method, the robust method given in Algorithm B, is the complete new algorithm including provisions for dealing with singularity or ill-conditioning of either the entire Jacobian matrix or any of its diagonal blocks. It includes Algorithm A as a sub-case. In Section 4 we will first prove the global convergence of Algorithm A (for the case when the Jacobian matrix and its diagonal blocks are uniformly nonsingular), and then build upon this to prove the global convergence of Algorithm B (without any assumption about the singularity of the Jacobian matrix or its diagonal blocks).

3.1 A basic algorithm

The basic method is fairly similar to the corrected implicit method introduced in [9, 10]. Its framework is as follows. Like all the previous and new implicit methods, it has inner and outer iterations. Its inner iterations are the same as in the corrected implicit method of [10] except for their stopping conditions, which are discussed later in this subsection. At iteration k, for each $i = 1, \cdots, q$, the inner iterations consist of solving

$$A_i \Delta x_i^{k,j-1} = -f_i(x_i^{k,j-1}, x_{q+1}^k),\ j = 1, \cdots, j_i$$

for some number of inner iterations j_i that is determined by the stopping conditions. Here $x_i^{k,0} = x_i^k$, and at the end of each inner iteration, $x_i^{k,j-1}$ is updated by

$$x_i^{k,j} = x_i^{k,j-1} + \Delta x_i^{k,j-1}.$$

We exit the inner iterations by setting

$$x_i^{k+1} = x_i^{k,j_i}.$$

Note that the $x_i^{k,j}$ are useful for describing the algorithm but do not need to be stored once the next inner iteration is completed. Note also that as in [10], A_i is kept fixed at $A_i(x^k)$ throughout the inner iterations for a given outer iteration, rather than re-evaluating it at each $x_i^{k,j}$, because this saves the cost of multiple evaluations and factorizations of the diagonal blocks per outer iteration without harming the rate of convergence of the outer iterations.

In the outer iteration, instead of solving

$$\hat{J}\Delta x_{q+1}^k = -f_{q+1}(x_1^{k+1}, \cdots, x_q^{k+1}, x_{q+1}^k)$$

($\hat{J}$ given by (2.2)) as in [10], we solve

$$\hat{J}\Delta x_{q+1}^k = -f_{q+1}(x_1^k, \cdots, x_q^k, x_{q+1}^k) - \sum_{i=1}^{q} C_i(x_i^{k+1} - x_i^k). \tag{3.1}$$

It is easy to see that the right hand side of (3.1) is a linear Taylor series approximation to $-f_{q+1}(x_1^{k+1}, \cdots, x_q^{k+1}, x_{q+1}^k)$. An obvious advantage of using this approximation is the saving of a function evaluation in the outer iteration, which is desirable because this evaluation may be expensive and because it would likely be performed sequentially and thus increase the parallel bottleneck of the outer iteration. Even more importantly, this modification has significant advantages with respect to global convergence that are discussed later in this subsection.

Finally, as in the corrected implicit method of [10], the correction

$$\delta_i = -A_i^{-1}B_i\Delta x_{q+1}^k$$

is calculated and added to each increment of Δx_i obtained in the inner iterations before the line search is performed.

We now present the entire algorithm and then discuss the remainder of its features. Note that since we assume for Algorithm A that the Jacobian matrix and its diagonal blocks are nonsingular, the Schur complement $\hat{J} = P - \sum_{i=1}^{q} C_i A_i^{-1} B_i$ is defined and nonsingular. In general, the Schur complement will also be well-conditioned if the entire Jacobian matrix and its diagonal blocks are well-conditioned, although we mention in Section 5 that there do exist somewhat pathological counterexamples.

Algorithm A: Iteration k of the Basic Method

Given $x^k = (x_1^k, \cdots, x_q^k, x_{q+1}^k)$ and A_i, B_i, C_i, $i = 1, \cdots, q$, P, $f_i(x)$, $i = 1, \cdots, q+1$, $F(x)$ and $J(x)$ as defined in (2.1a)–(2.1j), and constants α, β, τ_1, τ_2, j_{max} with $0 < \alpha < \beta < \frac{1}{2}$, $0 < \tau_1 \leq 1$, $\tau_2 \geq 1$, $j_{max} \geq 0$.

```
INNER ITERATIONS

  For i=1,q Do
      Solve $A_i\Delta x_i^k = -f_i(x_i^k, x_{q+1}^k)$ for $\Delta x_i^k$;
      $x_i^{k,0} = x_i^k$; $s_i = \Delta x_i^k$;
```

```
ADDITIONAL INNER ITERATIONS SUBALGORITHM
  $j = 0$;
  While $s_i^T A_i^T f_i(x_i^k, x_{q+1}^k) \leq -\tau_1 \|f_i(x_i^k, x_{q+1}^k)\|^2$
        and $\|A_i s_i\| \leq \tau_2 \|f_i(x_i^k, x_{q+1}^k)\|$ and $j \leq j_{max}$
    $\Delta x_i^k = s_i$; $j = j + 1$; $x_i^{k,j} = x_i^k + s_i$;
    Solve $A_i \delta_i = -f_i(x_i^{k,j}, x_{q+1}^k)$ for $\delta_i$;
    $s_i = s_i + \delta_i$;
  End{While}
  $j_i = j - 1$
End{Additional Inner Iterations Subalgorithm}
$t_i = -C_i \Delta x_i^k$.

OUTER ITERATION
  Let $\bar{F}(x^k) = \begin{pmatrix} \sum_{j=0}^{j_1} f_1(x_1^{k,j}, x_{q+1}^k) \\ \sum_{j=0}^{j_2} f_2(x_2^{k,j}, x_{q+1}^k) \\ \vdots \\ \sum_{j=0}^{j_q} f_q(x_q^{k,j}, x_{q+1}^k) \\ f_{q+1}(x^k) \end{pmatrix}$;
  Form $\hat{J} = P - \sum_{i=1}^{q} C_i A_i^{-1} B_i$;
  Solve $\hat{J} \Delta x_{q+1}^k = -f_{q+1}(x^k) + \sum_{i=1}^{q} t_i$ for $\Delta x_{q+1}^k$;
  Do the correction: Set $\Delta x_i^k = \Delta x_i^k - A_i^{-1} B_i \Delta x_{q+1}^k$, $i = 1, \cdots, q$;
  Perform Line Search: Find $\lambda^k$ so that
    $\|F(x^k + \lambda^k \Delta x_k)\|^2 - \|F(x^k)\|^2 \leq -\alpha \lambda^k F(x^k)^T \bar{F}(x^k)$ and
    $\|F(x^k + \lambda^k \Delta x_k)\|^2 - \|F(x^k)\|^2 \geq -\beta \lambda^k F(x^k)^T \bar{F}(x^k)$;
  $x^{k+1} = x^k + \lambda^k \Delta x^k$. □
```

The conditions enforced in the while loop for the inner iterations of Algorithm A will be seen, in its global convergence proof, to ensure that the step generated by the algorithm is a sufficient descent direction (i.e. giving at least τ_1 times the descent of the Newton direction) and that its length is bounded above (by τ_2 times the length of the Newton step). After one inner iteration, $s_i = -A_i^{-1} f_i(x_i^k, x_{q+1}^k)$, which gives $s_i^T A_i^T f_i(x_i^k, x_{q+1}^k) = -\|f_i(x_i^k, x_{q+1}^k)\|^2$, hence $s_i^T A_i^T f_i(x_i^k, x_{q+1}^k) \leq -\tau_1 \|f_i(x_i^k, x_{q+1}^k)\|^2$ is satisfied for any $0 < \tau_1 \leq 1$. In addition, $\|A_i s_i\| = \|f_i(x_i^k, x_{q+1}^k)\|$, hence $\|A_i s_i\| \leq \tau_2 \|f_i(x_i^k, x_{q+1}^k)\|$ is satisfied for any $\tau_2 \geq 1$. Therefore the while loop conditions always accept the step that is produced after one inner iteration, which by Lemma 3.1 below corresponds to the standard Newton step. The upper bound on the number of inner iterations, j_{max}, is not required by the theoretical analysis but would be imposed in any practical implementation. The optimal choices of τ_1 and τ_2 would need to be determined by experimentation on practical problems.

We now show in the following lemma that the search direction Δx^k generated by Algorithm A satisfies $J(x^k)\Delta x^k = -\bar{F}(x^k)$. In Section 4, this will be shown to ensure that Δx^k is a descent direction for $\frac{1}{2}\|F(x)\|^2$, since Lemma 3.1 shows that the dot product of Δx^k and the gradient of $\frac{1}{2}\|F(x)\|^2$ at x^k, $J(x^k)^T F(x^k)$, is equal to $(-J(x^k)^T\bar{F}(x^k))^T(J(x^k)^T F(x^k))$ $= \bar{F}(x^k)^T F(x^k)$, which from the first termination condition of the while loop of Algorithm A will be shown to be no greater than $-\tau_1\|F(x)\|^2$.

Lemma 3.1 *The search direction Δx^k generated by Algorithm A at iteration k satisfies $J(x^k)\Delta x^k = -\bar{F}(x^k)$.*

Proof. After the inner iterations

$$\Delta x_i^k = -A_i^{-1}\sum_{j=0}^{j_i} f(x_i^{k,j}, x_{q+1}^k),$$

and

$$t_i = -C_i\Delta x_i^k = C_iA_i^{-1}\sum_{j=0}^{j_i} f(x_i^{k,j}, x_{q+1}^k). \tag{3.2}$$

Hence, after the correction

$$\Delta x_i^k = -A_i^{-1}\sum_{j=0}^{j_i} f(x_i^{k,j}, x_{q+1}^k) - A_i^{-1}B_i\Delta x_{q+1}^k, \tag{3.3}$$

or,

$$A_i\Delta x_i^k + B_i\Delta x_{q+1}^k = -\sum_{j=0}^{j_i} f(x_i^{k,j}, x_{q+1}^k). \tag{3.4}$$

On the other hand, from the outer iteration Δx_{q+1}^k solves

$$\hat{J}\Delta x_{q+1}^k = -f_{q+1}(x_1^k, \cdots, x_q^k, x_{q+1}^k) + \sum_{i+1}^{q} t_i. \tag{3.5}$$

Plugging (3.2) in (3.5) and recalling $\hat{J} = P + \sum_{i=1}^q C_iA_i^{-1}B_i$, we have

$$P\Delta x_{q+1}^k + \sum_{i=1}^{q} C_iA_i^{-1}(-B_i\Delta x_{q+1}^k - \sum_{j=0}^{j_i} f(x_i^{k,j}, x_{q+1}^k)) = -f_{q+1}(x_1^k, \cdots, x_q^k, x_{q+1}^k),$$

which from (3.4) gives

$$P\Delta x_{q+1}^k + \sum_{i=1}^{q} C_i \Delta x_i^k = -f_{q+1}(x_1^k, \cdots, x_q^k, x_{q+1}^k). \tag{3.6}$$

Combining (3.4) and (3.6) yields $J(x^k)\Delta x^k = -\bar{F}(x^k)$ and completes the proof. □

A corollary of Lemma 3.1 is that Algorithm A with each $j_i = 0$ gives the Newton step, while for the corrected implicit method of [10], this is only true if f_{q+1} is linear. The difference arises because the previous algorithm evaluates f_{q+1} at the intermediate point, while the new algorithm evaluates the linear approximation to f_{q+1} at the intermediate point (equation (3.1)). This difference, and the analogous difference when multiple inner iterations are used which leads to a guarantee of sufficient descent as discussed above, is advantageous to the global convergence properties of the method. It does not affect the local convergence properties of the new algorithm, however, because from the theory of Dennis and Moré [3], it is easy to show that asymptotically the steps generated by the new method and the method of [8] are arbitrarily close to each other, and to the Newton step, no matter how many inner iterations are used. Thus by applying the theory in [8], the new algorithm, like the method of [8], is locally quadratically convergent to roots where the Jacobian matrix and all its diagonal submatrices are nonsingular.

3.2 The robust algorithm

If any of the diagonal blocks A_i, or $\hat{J}$, is (nearly) singular, Algorithm A needs to be modified to be numerically stable and produce a reasonable step. A simple way of dealing with this situation is suggested in [10]. Their approach is as follows. If any of these matrices, say G, is nearly singular, then based upon the general purpose approach for dealing with singular Jacobian matrices described in Dennis and Schnabel [4], they replace G^{-1} by $(G^TG + \mu I)^{-1}G^T$ in their computations, where μ is a small number that is a function of the norm of G.

In the context of the block bordered algorithm, however, the affect of this perturbation strategy is unclear. In particular, it is not clear that it is consistent with global convergence. An alternative would be to perturb the overall Jacobian matrix J of the problem in this manner if any A_i or $\hat{J}$ were (nearly) singular. This would lead to the solution of a linear system with the matrix is $J^TJ + \mu I$ for some $\mu > 0$. However $J^TJ + \mu I$ loses the block bordered sparsity structure of J. Hence using this perturbation directly is computationally inefficient, especially for parallel computation.

Our approach is related to both of the above approaches. It can be interpreted along the lines of perturbing the Jacobian matrix as a whole. However we do not solve

$$(J(x^k)^TJ(x^k) + \mu I)\Delta x^k = -J(x^k)^TF(x^k)$$

directly. To motivate our approach, first let us abbreviate the Jacobian matrix of the block bordered system at x^k as

$$J = \begin{pmatrix} A & B \\ C & P \end{pmatrix}, \tag{3.7}$$

where

$$A = diag(A_1, \cdots, A_q), B = [B_1^T, \cdots, B_q^T]^T, C = [C_1, \cdots, C_q]. \tag{3.8}$$

Then the system of $(J^TJ + D)\Delta x = -J^TF$ can be written as

$$\left(\begin{pmatrix} A & B \\ C & P \end{pmatrix}^T \begin{pmatrix} A & B \\ C & P \end{pmatrix} + \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix}\right) \begin{pmatrix} \Delta X_1 \\ \Delta X_2 \end{pmatrix} = -\begin{pmatrix} A & B \\ C & P \end{pmatrix}^T \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}, \tag{3.9}$$

where

$$\Delta X_1 = [\Delta x_1^T, \cdots, \Delta x_q^T]^T, \Delta X_2 = \Delta x_{q+1}, F_1 = [f_1^T(x^k), \cdots, f_q^T(x^k)]^T, F_2 = f_{q+1}(x^k), \tag{3.10}$$

and D_1, D_2 are non-negative diagonal matrices. Since J^TJ is at least positive semidefinite, the diagonal elements of D_1 and D_2 can be chosen so that the coefficient matrix is positive definite.

Multiplying (3.9) out gives

$$\begin{aligned} (A^TA + C^TC + D_1)\Delta X_1 + (A^TB + C^TP)\Delta X_2 &= -(A^TF_1 + C^TF_2) \\ (B^TA + P^TC)\Delta X_1 + (B^TB + P^TP + D_2)\Delta X_2 &= -(B^TF_1 + P^TF_2). \end{aligned}$$

Introducing $v = C\Delta X_1 + P\Delta X_2 + F_2$ leads to a corresponding symmetric system of linear equations

$$\begin{pmatrix} A^TA + D_1 & A^TB & C^T \\ B^TA & B^TB + D_2 & P^T \\ C & P & -I \end{pmatrix} \begin{pmatrix} \Delta X_1 \\ \Delta X_2 \\ v \end{pmatrix} = -\begin{pmatrix} A^TF_1 \\ B^TF_1 \\ F_2 \end{pmatrix}. \tag{3.11}$$

Obviously, (3.11) has the same solution as (3.9) for ΔX_1 and ΔX_2. In addition, the attractive property of (3.11) is that the coefficient matrix is a block bordered system not much larger than (3.9). The size of its diagonal blocks are the same as those of J, and the size of its border is double the size of the border of the original system. Also, the diagonal blocks of the matrix in (3.11) are of the form $A_i^TA_i + (D_1)_i$, showing that there is a close connection between the new approach and an approach that modifies the inner iterations directly. Our new robust algorithm will be based on solving (3.11) when any A_i, or $\hat{J}$, is (nearly) singular.

We now present the robust algorithm, and then discuss its remaining features. For the sake of convenience, we will abbreviate (3.11) as

$$\begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{B}^T & \tilde{P} \end{pmatrix} \begin{pmatrix} \Delta\tilde{X}_1 \\ \Delta\tilde{X}_2 \end{pmatrix} = -\begin{pmatrix} \tilde{F}_1 \\ \tilde{F}_2 \end{pmatrix}, \tag{3.12}$$

where

$$\tilde{A} = A^TA + D_1, \tilde{B} = [A^TB \ \ C^T], \tilde{P} = \begin{pmatrix} B^TB + D_2 & P^T \\ P & -I \end{pmatrix}, \tag{3.13}$$

$$\Delta\tilde{X}_2 = [\Delta X_2 \ \ v]^T, \tilde{F}_1 = A^TF_1, \tilde{F}_2 = [B^TF_1 \ \ F_2]^T. \tag{3.14}$$

That is, $\tilde{A}$ contains the diagonal blocks, $\tilde{B}$ is the vertical border, and $\tilde{B}^T$ is the horizontal border. Looked at in this way, the augmented system has the same form as the original block bordered matrix and can be solved using the same algebraic techniques, which is done in the Singular Case Subalgorithm below.

Algorithm B: Iteration k of the Robust Method

Given $x^k = (x_1^k, \cdots, x_q^k, x_{q+1}^k)$ and A_i, B_i, C_i, $i = 1, \cdots, q$, P, $f_i(x)$, $i = 1, \cdots, q+1$, $F(x)$ and $J(x)$ as defined in (2.1a)–(2.1j), and constants α, β, τ_1, τ_2, j_{max}, ϵ_1, ϵ_2, μ_1, μ_2, γ, σ, with $0 < \alpha < \beta < \frac{1}{2}$, $0 < \tau_1 \leq 1$, $\tau_2 \geq 1$, $j_{max} \geq 0$, $0 < \epsilon_1 < \mu_1$, $0 < \epsilon_2 < \mu_2$, $\gamma \geq 0$, $0 < \sigma \leq 1$.

```
INNER ITERATIONS
  For $i = 1, q$ Do
    If $||A_i^{-1}|| \leq \gamma$
    Then
      $\omega_i = 0$; (* Comment: $(D_1)_i = \omega_i I$ *)
      Solve $A_i \Delta x_i^k = -f_i(x_i^k, x_{q+1}^k)$ for $\Delta x_i^k$;
      $x_i^{k,0} = x_i^k$; $\Delta x_{iN}^k = s_i = \Delta x_i^k$;
      (* Comment: $\Delta x_{iN}^k$ will be used to form the Newton step if needed *)
      Perform ADDITIONAL INNER ITERATIONS SUBALGORITHM
        from Algorithm A to find $\Delta x_i^k = s_i$ so that
          $s_i^T A_i^T f_i(x_i^k, x_{q+1}^k) \leq -\tau_1 ||f_i(x_i^k, x_{q+1}^k)||^2$ and
          $||A_i s_i|| \leq \tau_2 ||f_i(x_i^k, x_{q+1}^k)||$;
      $t_i = -C_i \Delta x_i^k$;
    Else
      Choose $\omega_i$ with $\epsilon_1 \leq \omega_i \leq \mu_1$;
      Solve $(A_i^T A_i + \omega_i I)\Delta x_i^k = -A_i^T f_i(x_i^k, x_{q+1}^k)$ for $\Delta x_i^k$;
      $x_i^{k,0} = x_i^k$; $\Delta x_{iN}^k = s_i = \Delta x_i^k$;
      Perform ADDITIONAL INNER ITERATIONS SUBALGORITHM
        from Algorithm A, with the Solve statement changed to
        Solve $(A_i^T A_i + \omega_i I)\delta_i = -A_i^T f_i(x_i^{k,j}, x_{q+1}^k)$ for $\delta_i$
      and the While condition changed to
        While $s_i^T A_i^T f_i(x_i^k, x_{q+1}^k) < 0$ and $j \le j_{max}$
      to find $\Delta x_i^k = s_i$ so that $s_i^T A_i^T f_i(x_i^k, x_{q+1}^k) < 0$.
```

OUTER ITERATION

```
  If $\|A_i^{-1}\| \le \gamma$ for each $1 \le i \le q$
  Then Form $\hat{J} = P - B^T A^{-1} B$;
  If $\|A_i^{-1}\| \le \gamma$ for each $1 \le i \le q$ and $\|\hat{J}^{-1}\| \le \gamma$
  Then
      Solve $\hat{J}\Delta x_{q+1}^k = -f_{q+1}(x^k) + \sum_{i=1}^q t_i$ for $\Delta x_{q+1}^k$;
      Do the correction: Set $\Delta x_i^k = \Delta x_i^k - A_i^{-1} B_i^{-1} \Delta x_{q+1}^k$, $i = 1, \cdots, q$
  Else
      Perform SINGULAR CASE SUBALGORITHM FOR SEARCH
        DIRECTION COMPUTATION;
  Perform Line Search: Find $\lambda^k$ so that
      $\|F(x^k + \lambda^k \Delta x_k)\|^2 - \|F(x^k)\|^2 \le -\alpha\lambda^k \Delta x_k^T J(x^k)^T F(x^k)$ and
      $\|F(x^k + \lambda^k \Delta x_k)\|^2 - \|F(x^k)\|^2 \ge -\beta\lambda^k \Delta x_k^T J(x^k)^T F(x^k)$;
  $x^{k+1} = x^k + \lambda^k \Delta x^k$. □
```

SINGULAR CASE SUBALGORITHM FOR SEARCH DIRECTION COMPUTATION

```
  Let $A$, $B$, $C$ be defined by (3.7) and (3.8), $\Delta X_2$, $F_1$, $F_2$ be defined by (3.9), $\tilde{A}$,
      $\tilde{B}$, $\tilde{P}$, $\Delta\tilde{X}_2$, $\tilde{F}_1$, $\tilde{F}_2$ be defined by (3.12) to (3.14);
  Let $(\Delta X_1)^1 = (\Delta x_1, \cdots, \Delta x_q)$, $(\Delta X_1)^2 = (\Delta x_{1N}, \cdots, \Delta x_{qN})$;
  Let $\bar{\tilde{F}}_1 = \begin{pmatrix} \sum_{j=0}^{j_1} A_1^T f_1(x_1^{k,j}, x_{q+1}^k) \\ \sum_{j=0}^{j_2} A_2^T f_2(x_2^{k,j}, x_{q+1}^k) \\ \vdots \\ \sum_{j=0}^{j_q} A_q^T f_q(x_q^{k,j}, x_{q+1}^k) \end{pmatrix}$;
  Form $\tilde{J} = \tilde{P} - \tilde{B}^T \tilde{A}^{-1} \tilde{B}$ with $D_1 = diag(\omega_1 I, \cdots, \omega_q I)$ and $D_2 = 0$ if this results
      in $\|\tilde{J}^{-1}\| \le \gamma$, otherwise $D_2 = \omega_{q+1} I$ for some $\omega_{q+1}$ with $\epsilon_2 \le \omega_{q+1} \le \mu_2$;
```

```
Solve $\tilde{J}(\Delta\tilde{X}_2)^1 = -\tilde{F}_2 + \tilde{B}^T\tilde{A}^{-1}\bar{\tilde{F}}_1$ for $(\Delta\tilde{X}_2)^1$
  and $\tilde{J}(\Delta\tilde{X}_2)^2 = -\tilde{F}_2 + \tilde{B}^T\tilde{A}^{-1}\tilde{F}_1$ for $(\Delta\tilde{X}_2)^2$;
Do the correction: Set $(\Delta X_1)^1 = (\Delta X_1)^1 - \tilde{A}^{-1}\tilde{B}(\Delta\tilde{X}_2)^1$
  and $(\Delta X_1)^2 = (\Delta X_1)^2 - \tilde{A}^{-1}\tilde{B}(\Delta\tilde{X}_2)^2$;
Let $\Delta X^1 = ((\Delta X_1)^1, (\Delta X_2)^1)$, $\Delta X^2 = ((\Delta X_1)^2, (\Delta X_2)^2)$, where $(\Delta X_2)^1$,
    $(\Delta X_2)^2$ are the first $n_{q+1}$ components of $(\Delta\tilde{X}_2)^1$, $(\Delta\tilde{X}_2)^2$ respectively
    as defined in (3.14).
If $(\Delta X^1)^T J(x^k)^T F(x^k) < -\sigma\|\Delta X^1\|\|J(x^k)^T F(x^k)\|$
    Then $\Delta x^k = \Delta X^1$
    Else $\Delta x^k = \Delta X^2$. □
```

The first thing to note about Algorithm B is that if J and its diagonal blocks are sufficiently well-conditioned, in the sense that the inverses of each A_i and $\hat{J}$ are not too large, then Algorithm B is identical to Algorithm A. Thus it has the same fast local convergence properties as Algorithm A for problems where the Jacobian matrix at the solution is well-conditioned in this sense. The reason we have based the algorithmic decisions about near-singularity on the norms of the inverses of the blocks, rather than upon their condition numbers, is that each block being well-conditioned does not imply that the entire matrix is well-conditioned. If the elements of J vary widely in magnitude, then the norms of A_i and $\hat{J}$ probably should also be considered in the determination of near-singularity in Algorithm B. From the point of view of the theory, the heuristics that are used to determine near-singularity do not affect the convergence properties, all that matters is that a singular A_i or $\hat{J}$ is perturbed. Thus many practical variations could be introduced into the decision in Algorithm B about when A_i or $\hat{J}$ is perturbed without affecting its theoretical properties.

Note also that in the singular case (when either some A_i or $\hat{J}$ is (nearly) singular), Algorithm B produces two intermediate trial directions, ΔX^1 and ΔX^2, with ΔX^2 corresponding to one inner iteration for each diagonal block and ΔX^1 corresponding to multiple inner iterations. The reasons for this are as follows. First, as in Algorithm A, we will see in Lemma 3.2 below that the step with one inner iteration, ΔX^2, is a perturbed Newton step and thus a good step from the point of view of global convergence. Second, while in the well-conditioned case, we can produce independent termination conditions for the inner iterations that guarantee that the overall step direction that results from using multiple inner iterations will be a good one, in the poorly-conditioned case where some part of the matrix is perturbed, there does not seem to be a good way to guarantee this. Intuitively, the problem stems in part from the fact that we may further perturb the matrix after the inner iterations. So instead, we keep two trial steps, knowing that the second (ΔX^2) is guaranteed to be a good descent direction, and that the first (ΔX^1) may often be better in practice but is not guaranteed to be a good direction. Then we simply test whether the first direction provides sufficient descent, and if so we use it in the line search, otherwise we use the second direction. The extra expense of this procedure is minimal, just the storage of one extra trial direction vector and a small amount of extra algebraic work. A related interesting property of Algorithm B is that if we always used only one inner iteration, it would become a globally convergent algorithm for the explicit method that is consistent with sparsity, parallelism, and possible singularity, which by itself is a useful contribution for solving block bordered systems of nonlinear equations.

It is also easy to verify that Algorithm B parallelizes quite well. In particular, the major work in the singular case, the formation of $\tilde{J}$ and of the terms $\tilde{B}^T\tilde{A}^{-1}\bar{\tilde{F}}_1$, $\tilde{B}^T\tilde{A}^{-1}\tilde{F}_1$, $\tilde{A}^{-1}\tilde{B}(\Delta\tilde{X}_2)^1$, and $\tilde{A}^{-1}\tilde{B}(\Delta\tilde{X}_2)^2$, is analogous algebraically to calculations in Algorithm A and can also be performed efficiently in parallel. The remainder of the calculations are the same, or virtually the same, as in Algorithm A and have similar properties for parallelism.

In analogy to Lemma 3.1, Lemma 3.2 shows that the search directions generated by Algorithm B are the solutions to perturbed Newton equations. This lemma will be a key to the global convergence analysis of the algorithm in Section 4. The other key will be establishing that the matrix $J^TJ + D$ produced by Algorithm B is nonsingular and that the norm of its inverse is uniformly bounded above.

Lemma 3.2 *The vectors* $(\Delta\tilde{X})^i = ((\Delta X_1)^i, (\Delta\tilde{X}_2)^i)$, $i = 1, 2$ *obtained by Algorithm B at iteration* k *satisfy*

$$\begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{B}^T & \tilde{P} \end{pmatrix} \begin{pmatrix} \Delta(X_1)^i \\ \Delta(\tilde{X}_2)^i \end{pmatrix} = -\begin{pmatrix} (\hat{F}_1)^i \\ \tilde{F}_2 \end{pmatrix},$$

where $(\hat{F}_1)^1 = \bar{\tilde{F}}_1$ *and* $(\hat{F}_1)^2 = \tilde{F}_1$.

Proof. By an almost identical proof as for (3.4) in Lemma 3.1, for either $i = 1$ or 2, after the correction, $(\Delta X_1)^i$ and $(\Delta\tilde{X}_2)^i$ satisfy

$$\tilde{A}(\Delta X_1)^i + \tilde{B}(\Delta\tilde{X}_2)^i = -(\hat{F}_1)^i. \tag{3.15}$$

On the other hand, in the outer iteration, $(\Delta\tilde{X}_2)^i$ satisfies

$$(\tilde{P} - \tilde{B}^T\tilde{A}^{-1}\tilde{B})(\Delta\tilde{X}_2)^i = -\tilde{F}_2 + \tilde{B}^T\tilde{A}^{-1}(\hat{F}_1)^i,$$

which implies

$$\tilde{P}(\Delta\tilde{X}_2)^i + \tilde{B}^T\tilde{A}^{-1}(-\tilde{B}(\Delta\tilde{X}_2)^i - (\hat{F}_1)^i) = -\tilde{F}_2,$$

which in turn gives

$$\tilde{P}(\Delta\tilde{X}_2)^i + \tilde{B}^T(\Delta X_1)^i = -\tilde{F}_2 \tag{3.16}$$

by using (3.15). Combining (3.15) and (3.16) completes the proof. □

4 Global convergence analysis of the algorithms

In this section we prove the global convergence of both algorithms from the previous section. Section 4.1 gives the rather simple proof of the global convergence of Algorithm A. Section 4.2 then gives the more complicated proof of the global convergence of Algorithm B. Some of the lemmas, and the main theorem, for Algorithm B essentially use the proof for Algorithm A as a sub-case, since when the diagonal blocks of the Jacobian matrix and its Schur complement are nonsingular and the norms of their inverses are not too large, the step taken by Algorithm B is the same as the step taken in Algorithm A. The norm $||.||$ is assumed to be the l_2 norm throughout.

4.1 Global convergence of the basic method

Recall that Algorithm A generates a step that is the solution to the system of equations,

$$\begin{pmatrix} A_1 & & & & B_1 \\ & A_2 & & & B_2 \\ & & \ddots & & \vdots \\ & & & A_q & B_q \\ C_1 & C_2 & \cdots & C_q & P \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_q \\ \Delta x_{q+1} \end{pmatrix} = - \begin{pmatrix} \sum_{j=0}^{j_1} f_1(x_1^{k,j}, x_{q+1}^k) \\ \sum_{j=0}^{j_2} f_2(x_2^{k,j}, x_{q+1}^k) \\ \vdots \\ \sum_{j=0}^{j_q} f_q(x_q^{k,j}, x_{q+1}^k) \\ f_{q+1}(x^k) \end{pmatrix},$$

where each $(j_i + 1)$, $1 \le i \le q$ is the actual number of inner iterations executed for ith set of equations. Also recall that the right hand side of this equation is denoted by $\bar{F}(x^k)$. We first show that the step satisfying this equation is a descent direction on $\frac{1}{2}||F(x)||^2$. We assume in this subsection that each A_i, and J, are uniformly nonsingular, so that the step of Algorithm A is well defined.

Lemma 4.1 *The step direction Δx^k generated by Algorithm A is a descent direction on $\frac{1}{2}||F(x)||^2$, as long as each $A_i(x^k)$, $i = 1, \cdots, q$ and $J(x^k)$ are nonsingular and $F(x^k) \neq 0$.*

Proof. Note that the gradient of $\frac{1}{2}||F(x)||^2$ at x^k is $J(x^k)^T F(x^k)$, and by Lemma 3.1, the step direction generated by Algorithm A is $-J(x^k)^{-1}\bar{F}(x^k)$ as long as $J(x^k)$ is nonsingular. The dot product of the gradient and the step direction is

$$\begin{aligned} & -F(x^k)^T J(x^k) J(x^k)^{-1} \bar{F}(x^k) \\ = & -F(x^k)^T \bar{F}(x^k) \\ = & -(\sum_{i=1}^{q} f(x_i^k, x_{q+1}^k)^T (\sum_{j=0}^{j_i} f_i(x_i^{k,j}, x_{q+1}^k))) - f_{q+1}(x^k)^T f_{q+1}(x^k) \qquad (4.1) \\ \le & -(\sum_{i=1}^{q} \tau_1 ||f_i(x_i^k, x_{q+1}^k)||^2) - f_{q+1}(x^k)^T f_{q+1}(x^k) \qquad (4.2) \\ < & \ 0, \end{aligned}$$

with the inequality between (4.1) and (4.2) coming from the relation $A_i s_i = -\sum_{j=0}^{j_i} f_i(x_i^{k,j}, x_{q+1}^k)$ and the condition $s_i^T A_i^T f_i(x_i^k, x_{q+1}^k) \leq -\tau_1 \|f_i(x_i^k, x_{q+1}^k)\|^2$ in Algorithm A for $1 \leq i \leq q$, and the final inequality coming from $\tau_1 > 0$ and $F(x^k) \neq 0$. This completes the proof. □

Now we conclude our analysis of Algorithm A by proving Theorem 4.2.

Theorem 4.2 *If $\|J(x)^{-1}\|$ is uniformly bounded above, $A_i(x)$ is uniformly nonsingular, $i = 1, \cdots, q$, and $f_i(x)$ is continuously differentiable and $\nabla f_i(x)$ is Lipschitz continuous for $1 \leq i \leq q+1$, then the sequence generated by Algorithm A obeys either*

(a) $J(x^k)^T F(x^k) = 0$ *for some* $k \geq 0$, *or*

(b) $\lim_{k\to\infty} F(x^k) = 0$.

Proof. By Lemma 4.1, at each iteration, the search direction generated by Algorithm A is a descent direction on $\frac{1}{2}\|F(x)\|^2$. By Theorem 6.3.2 of [4], there exists $\lambda^k > 0$ such that the line search conditions in the outer iteration of Algorithm A are satisfied at each iteration. Applying Theorem 6.3.3 of [4] to the function $\frac{1}{2}\|F(x)\|^2$ and the iterates generated by Algorithm A gives either $J(x^k)^T F(x^k) = 0$ for some $k \geq 0$, or

$$\lim_{k\to\infty} \frac{(J(x^k)^T F(x^k))^T(-\lambda^k J(x^k)^{-1}\bar{F}(x^k))}{\|-\lambda^k J(x^k)^{-1}\bar{F}(x^k)\|} = 0, \tag{4.3}$$

since $\frac{1}{2}\|F(x)\|^2$ is bounded below by zero. But for each k

$$\begin{aligned}
& \frac{|(J(x^k)^T F(x^k))^T(-\lambda^k J(x^k)^{-1}\bar{F}(x^k))|}{\|-\lambda^k J(x^k)^{-1}\bar{F}(x^k)\|} \\
= & \frac{|F(x^k)^T \bar{F}(x^k)|}{\|J(x^k)^{-1}\bar{F}(x^k)\|} \\
= & \frac{|\sum_{i=1}^q f(x_i^k, x_{q+1}^k)^T(\sum_{j=0}^{j_i} f_i(x_i^{k,j}, x_{q+1}^k)) + \|f_{q+1}(x^k)\|^2|}{\|J(x^k)^{-1}\bar{F}(x^k)\|} \\
\geq & \frac{\tau_1\|F_1(x^k)\|^2 + \|f_{q+1}(x^k)\|^2}{\|J(x^k)^{-1}\|\|\bar{F}(x^k)\|} \\
\geq & \frac{\tau_1\|F(x^k)\|^2}{\tau_2\|J(x^k)^{-1}\|\|F(x^k)\|} && (4.4) \\
= & \frac{\tau_1\|F(x^k)\|}{\tau_2\|J(x^k)^{-1}\|}. && (4.5)
\end{aligned}$$

where we have again used the inequality between (4.1) and (4.2) to obtain the numerator in (4.4), and the relation $\|A_i s_i\| = \|\sum_{j=0}^{j_i} f_i(x_i^{k,j}, x_{q+1}^k)\| \leq \tau_2\|f_i(x_i^k, x_{q+1}^k)\|$, $i = 1, \cdots, q$, from Algorithm A (i.e. $\|(\bar{F}(x^k))_i\| \leq \tau_2\|(F(x^k))_i\|$) along with the definition of $\bar{F}(x^k)$ to obtain

$\|\bar{F}(x^k)\| \leq \tau_2 \|F(x^k)\|$ for the denominator of (4.4). Since $\|J(x)^{-1}\|$ is uniformly bounded, and $\tau_1, \tau_2 > 0$ are positive constants, (4.3) and (4.5) imply

$$\lim_{k \to \infty} F(x^k) = 0,$$

which completes the proof. $\square$

4.2 Global convergence of the robust method

Recall that the step direction taken by Algorithm B is either the same as in Algorithm A (if each $\|A_i\| \leq \gamma$ and $\|\hat{J}^{-1}\| < \gamma$), or the step ΔX^1 from the singular case subalgorithm if it is a direction of sufficient descent, or the step $\Delta X^2 = -(J(x^k)^T J(x^k) + D)^{-1} J(x^k)^T F(x^k)$ otherwise, where $D = diag(\omega_1 I, \cdots, \omega_q I, \omega_{q+1} I)$ is chosen at iteration k. The first case has been analyzed in the previous subsection, and the second case is straightforward since the search direction is a sufficient descent direction. Thus the key to the global convergence analysis of Algorithm B turns out to be to show that $\|(J^T J + D)^{-1}\|$ is bounded above in order to analyze the third case. This is established as follows. We first establish the relationship between the inverse of the augmented matrix

$$\begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{B}^T & \tilde{P} \end{pmatrix} \tag{4.6}$$

and the inverse of $J^T J + D$ by showing in Lemma 4.9 that $\|(J^T J + D)^{-1}\|$ is bounded above if the norm of the inverse of (4.6) is bounded above. Then we show in Lemma 4.10 that the norm of the inverse of (4.6) is bounded above if $\|J\|$, $\|\tilde{A}^{-1}\|$ and the norm of the inverse of $(\tilde{P} - \tilde{B}^T \tilde{A}^{-1} \tilde{B})$, the Schur complement of (4.6) with respect to $\tilde{A}$, are bounded above. In Lemmas 4.11 and 4.12, we establish separately that $\|\tilde{A}^{-1}\|$ and $\|(\tilde{P} - \tilde{B}^T \tilde{A}^{-1} \tilde{B})^{-1}\|$ are bounded above. These lemmas are sufficient for establishing the boundedness of $\|(J^T J + D)^{-1}\|$, which is done in Lemma 4.13. This lemma also shows that if Algorithm B takes the step of Algorithm A (i.e. sets $D = 0$), then $\|J^{-1}\|$ is bounded.

Using these results we show that the step generated by Algorithm B is a descent direction on $\frac{1}{2}\|F(x)\|^2$ in Lemma 4.14, and finish our analysis by presenting the main result of this paper, the global convergence of Algorithm B, in Theorem 4.15.

Before going into this analysis, in Lemmas 4.3 to 4.8 we present some basic results related to Schur complements of matrices, that will be used in the proofs of Lemmas 4.9 to 4.13.

Consider the square matrix

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}, \tag{4.7}$$

where A and D also are square. First we state the following three well-known facts about this matrix as lemmas without proof.

Lemma 4.3 *If A is nonsingular and the Schur complement $D - CA^{-1}B$ is nonsingular, then the inverse of (4.7) exists and is given by*

$$\begin{pmatrix} A^{-1} + A^{-1}BSCA^{-1} & -A^{-1}BS \\ -SCA^{-1} & S \end{pmatrix},$$

where $S = (D - CA^{-1}B)^{-1}$.

Parallel to Lemma 4.3 is the following fact.

Lemma 4.4 *If D is nonsingular and $A - BD^{-1}C$ is nonsingular, then the inverse of (4.7) exists and is given by*

$$\begin{pmatrix} \tilde{S} & -\tilde{S}BD^{-1} \\ -D^{-1}C\tilde{S} & D^{-1} + D^{-1}C\tilde{S}BD^{-1} \end{pmatrix},$$

where $\tilde{S} = (A - BD^{-1}C)^{-1}$.

Closely related to Lemma 4.4 is the following fact which will be useful to us.

Lemma 4.5 *If the matrix (4.7) and D are nonsingular then $A - CD^{-1}B$ is nonsingular.*

Using the above lemmas we now establish three additional lemmas that relate the norm of the inverse of a matrix to the norm of the inverse of certain submatrices and Schur complements.

Lemma 4.6 *If $\|A^{-1}\|$, $\|B\|$, $\|C\|$ and $\|(D-CA^{-1}B)^{-1}\|$ are bounded above, then the norm of the inverse of (4.7) is bounded above.*

Proof. From Lemma 4.3, the inverse of (4.7) involves A^{-1}, B, C, $(D - CA^{-1}B)^{-1}$ and arithmetic combinations of them. Since $\|A^{-1}\|$, $\|B\|$, $\|C\|$ and $\|(D - CA^{-1}B)^{-1}\|$ are all bounded above, the norm of the inverse of (4.7) is bounded above, which completes the proof. □

Parallel to Lemma 4.6 is the following lemma.

Lemma 4.7 *If $\|D^{-1}\|$, $\|B\|$, $\|C\|$ and $\|(A-BD^{-1}C)^{-1}\|$ are bounded above, then the norm of the inverse of (4.7) is bounded above.*

The following lemma deals with matrices of a special form that we will have use for.

Lemma 4.8 *For any matrices X, Y with X square, if*

$$\left\| \begin{pmatrix} X & Y^T \\ Y & -I \end{pmatrix}^{-1} \right\| \tag{4.8}$$

is bounded above, then $\|(X + Y^TY)^{-1}\|$ is bounded above.

Proof. First $(X+Y^TY)^{-1}$ exists by Lemma 4.5. By Lemma 4.4,

$$\begin{pmatrix} X & Y^T \\ Y & -I \end{pmatrix}^{-1} = \begin{pmatrix} (X+Y^TY)^{-1} & (X+Y^TY)^{-1}Y^T \\ Y(X+Y^TY)^{-1} & -I+Y(X+Y^TY)^{-1}Y^T \end{pmatrix}.$$

Hence $\|(X+Y^TY)^{-1}\|$ is bounded above because otherwise (4.8) is not bounded above. □

Now we present the lemmas that allow us to bound $\|(J^TJ+D)^{-1}\|$.

Lemma 4.9 *Let J be given by (3.7) with A and P square, $D = diag(D_1, D_2)$ be a nonsingular diagonal matrix, and $\tilde{A}$, $\tilde{B}$, $\tilde{P}$ be given by (3.13). Then $\|(J^TJ+D)^{-1}\|$ is bounded above if*

$$\left\| \begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{B}^T & \tilde{P} \end{pmatrix}^{-1} \right\|$$

is bounded above.

Proof. Recall that

$$\begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{B}^T & \tilde{P} \end{pmatrix} = \begin{pmatrix} A^TA+D_1 & A^TB & C^T \\ B^TA & B^TB+D_2 & P^T \\ C & P & -I \end{pmatrix}. \tag{4.9}$$

Let

$$X = \begin{pmatrix} A^TA+D_1 & A^TB \\ B^TA & B^TB+D_2 \end{pmatrix},$$

and $Y = [C \;\; P]$. Note that $X+Y^TY = J^TJ+D$. Applying Lemma 4.8 to the right hand side of (4.9) completes the proof. □

Lemma 4.10 *Let J, $\tilde{A}$, $\tilde{B}$, $\tilde{P}$ be defined as in Lemma 4.9. If $\|J\|$, $\|\tilde{A}^{-1}\|$, and $\|(\tilde{P} - \tilde{B}^T\tilde{A}^{-1}\tilde{B})^{-1}\|$ are bounded above then*

$$\left\| \begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{B}^T & \tilde{P} \end{pmatrix}^{-1} \right\|$$

is bounded above.

Proof. Simply note that $\tilde{B} = [A^TB \;\; C^T]$. Since $\|J\|$ is bounded above, $\|A\|$, $\|B\|$ and $\|C\|$ are bounded above as well. Hence $\tilde{B}$ is bounded above. Applying Lemma 4.6 completes the proof. □

Lemma 4.11 *After the inner iteration in Algorithm B, $\|(A^TA+D_1)^{-1}\|$ is bounded above by $\max\{\gamma^2, \frac{1}{\epsilon_1}\}$, where $D_1 = diag(\omega_1 I, \cdots, \omega_q I)$.*

Proof. We have $(A^TA+D_1)^{-1} = diag((A_1^TA_1+\omega_1 I)^{-1},\cdots,(A_q^TA_q+\omega_q I)^{-1})$. For each $1 \le i \le q$, if $\omega_i = 0$ then $\|(A_i^TA_i+\omega_i I)^{-1}\| = \|A_i^{-1}\|^2 \le \gamma^2$, since in this case $\|A_i^{-1}\| \le \gamma$ by Algorithm B. If $\omega_i \ne 0$ then $\|(A_i^TA_i+\omega_i I)^{-1}\| \le \frac{1}{\omega_i} \le \frac{1}{\epsilon_1}$ by Algorithm B. Hence $\|(A^TA+D_1)^{-1}\|$ is bounded above by $\max(\gamma^2, 1/\epsilon_1)$. □

Lemma 4.12 *After the outer iteration of Algorithm B, if the Singular Case Subalgorithm is used,* $\|(\tilde{P}-\tilde{B}^T\tilde{A}^{-1}\tilde{B})^{-1}\|$ *is bounded above by the function of* $\|A\|$, $\|B\|$, $\|C\|$, $\|P\|$, ϵ_1, ϵ_2 *and* γ *given in (4.10) below.*

Proof. Recall that

$$\begin{aligned}
&\tilde{P}-\tilde{B}^T\tilde{A}^{-1}\tilde{B}\\
=&\begin{pmatrix} B^TB+D_2 & P^T \\ P & -I \end{pmatrix} - \begin{pmatrix} B^TA \\ C \end{pmatrix}(A^TA+D_1)^{-1}(A^TB \;\; C^T)\\
=&\begin{pmatrix} B^TB+\omega_{q+1}I - B^TA(A^TA+D_1)^{-1}A^TB & P^T - B^TA(A^TA+D_1)^{-1}C^T \\ P-C(A^TA+D_1)^{-1}A^TB & -I-C(A^TA+D_1)^{-1}C^T \end{pmatrix},
\end{aligned}$$

and from Algorithm B that if $\omega_{q+1}=0$, $\|(\tilde{P}-\tilde{B}^T\tilde{A}^{-1}\tilde{B})^{-1}\| \le \gamma$. Assume for now that $\omega_{q+1} \ne 0$. Let

$$\begin{aligned}
M &= B^TB+\omega_{q+1}I - B^TA(A^TA+D_1)^{-1}A^TB,\\
N &= P-C(A^TA+D_1)^{-1}A^TB,\\
W &= -I-C(A^TA+D_1)^{-1}C^T.
\end{aligned}$$

Observe that

$$M = \omega_{q+1}I + B^T(I-A(A^TA+D_1)^{-1}A^T)B,$$

and that $B^T(I-A(A^TA+D_1)^{-1}A^T)B$ is at least positive semidefinite. Also $-W$ is positive definite with $\|W^{-1}\| \le 1$. Furthermore,

$$\begin{aligned}
&M - N^TW^{-1}N\\
=&\ \omega_{q+1}I + B^T(I-A(A^TA+D_1)^{-1}A^T)B + N^T(I+C(A^TA+D_1)^{-1}C^T)N,
\end{aligned}$$

which is positive definite with $\|(M-N^TW^{-1}N)^{-1}\| < \frac{1}{\omega_{q+1}} \le \frac{1}{\epsilon_2}$ since $(M-N^TW^{-1}N) - \omega_{q+1}I$ is at least positive semi-definite and $\omega_{q+1} \ge \epsilon_2$. Since also $(A^TA+D_1)^{-1}$ is bounded above by Lemma 4.11, Lemma 4.7 implies that

$$\left\|\begin{pmatrix} M & N^T \\ N & W \end{pmatrix}^{-1}\right\|$$

is bounded above. In addition, from $\|W^{-1}\| \le 1$, $\|(M-N^TW^{-1}N)^{-1}\| \le \frac{1}{\epsilon_2}$, and $\|N\| \le \|P\| + \|C\|\|A\|\|B\| \cdot \max\{\gamma^2, \frac{1}{\epsilon_1}\}$(using Lemma 4.11), it is straightforward to calculate from

Lemma 4.4 and the fact that an l_p norm of a matrix is bounded above by the sum of the norms of its submatrices that

$$\|(\tilde{P} - \tilde{B}^T\tilde{A}^{-1}\tilde{B})^{-1}\| \leq \max\{\gamma, 1 + \frac{[\|P\| + \|C\|\|A\|\|B\| \cdot \max\{\gamma^2, \frac{1}{\epsilon_1}\} + 1]^2}{\epsilon_2}\} \qquad (4.10)$$

with the first term in the right hand side of (4.10) accounting for the case when $\omega_{q+1} = 0$. □

Lemma 4.13 *If $\|J(x)\|$ is uniformly bounded above, then $\|(J(x^k)^TJ(x^k)+D)^{-1}\|$ produced by Algorithm B is uniformly bounded above, where $D = diag(\omega_1 I, \cdots, \omega_q I, \omega_{q+1}I)$.*

Proof. Assume first that the Singular Case Subalgorithm is used by Algorithm B at iteration k (i.e. $D \neq 0$). By Lemma 4.11, $\|\tilde{A}^{-1}\|$ is bounded above by a constant. Also, by Lemma 4.12 and $\|J(x^k)\|$ uniformly bounded (which implies that $\|A\|$, $\|B\|$, $\|C\|$, and $\|P\|$ are uniformly bounded above), $\|(\tilde{P} - \tilde{B}^T\tilde{A}^{-1}\tilde{B})^{-1}\|$ is bounded above by a constant. Then, by Lemma 4.10, and the definition of $\tilde{B}$ and the boundedness of $\|A\|$, $\|B\|$, and $\|C\|$,

$$\left\| \begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{B}^T & \tilde{P} \end{pmatrix}^{-1} \right\|$$

is bounded above, which in turn implies $\|(J(x^k)^TJ(x^k)+D)^{-1}\|$ is bounded above by Lemma 4.9.

Finally, if the Singular Case Subalgorithm is not used, i.e. $D = 0$, then $\|A_i^{-1}\| \leq \gamma$ for each $1 \leq i \leq q$, which implies $\|A^{-1}\| \leq \gamma$, and $\|\hat{J}\| = \|(P - CA^{-1}B)^{-1}\| \leq \gamma$. Thus by Lemma 4.3, $J(x^k)$ is nonsingular and from the norms of the submatrices of $J(x^k)^{-1}$,

$$\|J(x^k)^{-1}\| \leq 2\gamma + \gamma^2(\|B\| + \|C\|) + \gamma^3\|B\|\|C\|. \qquad (4.11)$$

Therefore $J(x^k)^TJ(x^k)$ is positive definite, and $(J(x^k)^TJ(x^k))^{-1}$ is uniformly bounded due to $\|(J(x^k)^TJ(x^k))^{-1}\| \leq \|J(x^k)^{-1}\|^2$, (4.11), and $\|B\|$, $\|C\|$ uniformly bounded. □

Lemma 4.14 *The direction generated by Algorithm B is a descent direction on the function $\frac{1}{2}\|F(x)\|^2$, if its gradient $J(x^k)^TF(x^k)$ is not equal to zero.*

Proof. If Algorithm B uses the Singular Case Subalgorithm it selects the step direction Δx^k to be either ΔX^1 or ΔX^2. If it selects $\Delta x^k = \Delta X^1$, then from the selection rule we have

$$(\Delta x^k)^TJ(x^k)^TF(x^k) < -\sigma\|\Delta x^k\|\|J(x^k)^TF(x^k)\| \qquad (4.12)$$

which shows that Δx^k is a descent direction on $\frac{1}{2}\|F(x)\|^2$ at x^k (and that $J(x^k)^TF(x^k) \neq 0$). If it selects $\Delta x^k = \Delta X^2 = ((\Delta X_1)^1, (\Delta X_2)^2)$ then from Lemma 3.2, $\Delta\tilde{X}^2 = ((\Delta X_1)^1, (\Delta X_2)^2, v)$ satisfies

$$\begin{pmatrix} \tilde{A} & \tilde{B} \\ \tilde{B}^T & \tilde{P} \end{pmatrix} \begin{pmatrix} \Delta(\tilde{X}_1)^2 \\ \Delta(\tilde{X}_2)^2 \end{pmatrix} = - \begin{pmatrix} \tilde{F}_1 \\ \tilde{F}_2 \end{pmatrix}.$$

which is the same as (3.11), and from the equivalence of (3.11) and (3.9), ΔX^2 satisfies (3.9) which is the same as

$$(J(x^k)^T J(x^k) + D)\Delta x^k = -J(x^k)^T F(x^k). \tag{4.13}$$

From Lemma 4.13, $(J(x^k)^T J(x^k) + D)^{-1}$ exists, and thus is positive definite since D is a non-negative diagonal matrix. Therefore,

$$\begin{aligned} & (\Delta x^k)^T J(x^k)^T F(x^k) \\ = \ & -(J(x^k)^T F(x^k))^T (J(x^k)^T J(x^k) + D)^{-1} J(x^k)^T F(x^k) < 0 \end{aligned}$$

since $J(x^k)^T F(x^k) \neq 0$. Hence Δx^k is a descent direction on $\frac{1}{2}||F(x)||^2$.

Finally, if Algorithm B does not use the Singular Case Subalgorithm, then $J(x^k)^{-1}$ exits from the proof of Lemma 4.13, and Δx^k is a descent direction on $\frac{1}{2}||F(x)||^2$ from Lemma 4.1. □

Now we conclude our analysis by giving the global convergence result for Algorithm B.

Theorem 4.15 *If $||J(x)||$ is uniformly bounded above, and $f_i(x)$ is continuously differentiable and $\nabla f_i(x)$ is Lipschitz continuous for $1 \leq i \leq q+1$, then the sequence generated by Algorithm B obeys either*

(a) $J(x^k)^T F(x^k) = 0$ *for some* $k \geq 0$, *or*

(b) $\lim_{k\to\infty} J(x^k)^T F(x^k) = 0$.

Proof. By Lemma 4.14, the step generated by Algorithm B is a descent direction at each iteration. Thus Theorem 6.3.2 of [4] ensures that there exists λ^k such that the line search conditions are satisfied at each iteration. Applying Theorem 6.3.3 of [4] to the function $\frac{1}{2}||F(x)||^2$ and the iterates generated by Algorithm B gives either $J(x^k)^T F(x^k) = 0$ for some $k \geq 0$, or

$$\lim_{k\to\infty} \frac{(J(x^k)^T F(x^k))^T \lambda^k \Delta x^k}{||\lambda^k \Delta x^k||} = 0 \tag{4.14}$$

since $\frac{1}{2}||F(x)||^2$ is bounded below by zero. At iteration k, if Δx^k is selected by the Singular Case Subalgorithm and satisfies (4.13) (i.e. $\Delta x^k = \Delta X^2$), then

$$\begin{aligned} & \frac{(J(x^k)^T F(x^k))^T \lambda^k \Delta x^k}{||\lambda^k \Delta x^k||} \\ = \ & \frac{(J(x^k)^T F(x^k))^T (-(J(x^k)^T J(x^k) + D)^{-1} J(x^k)^T F(x^k))}{||-(J(x^k)^T J(x^k) + D)^{-1} J(x^k)^T F(x^k)||}. \end{aligned}$$

Since $\|J(x^k)^T J(x^k) + D\|$ is bounded above due to $\|J\|$ bounded above and $\omega_1, \cdots, \omega_{q+1}$ bounded above, and $\|(J(x^k)^T J(x^k) + D)^{-1}\|$ is bounded above by Lemma 4.13, we have

$$\begin{aligned}
& \frac{|(J(x^k)^T F(x^k))^T(-(J(x^k)^T J(x^k) + D)^{-1} J(x^k)^T F(x^k))|}{\| - (J(x^k)^T J(x^k) + D)^{-1} J(x^k)^T F(x^k)\|} \\
\geq\ & \frac{\frac{1}{\|J(x^k)^T J(x^k)+D\|}\|J(x^k)^T F(x^k)\|^2}{\|(J(x^k)^T J(x^k) + D)^{-1}\|\|J(x^k)^T F(x^k)\|} \\
=\ & \frac{\|J(x^k)^T F(x^k)\|}{\|J(x^k)^T J(x^k) + D\|\|(J(x^k)^T J(x^k) + D)^{-1}\|} \\
\geq\ & \theta\|J(x^k)^T F(x^k)\|, \qquad (4.15)
\end{aligned}$$

where $\theta > 0$ is a positive constant.

On the other hand, if Δx^k is selected by Singular Case Subalgorithm and satisfies (4.12) at iteration k (i.e. $\Delta x^k = \Delta X^1$), then

$$\begin{aligned}
& \frac{|(J(x^k)^T F(x^k))^T \lambda^k \Delta x^k|}{\|\lambda^k \Delta x^k\|} \\
\geq\ & \frac{\sigma\|J(x^k)^T F(x^k)\|\|\Delta x^k\|}{\|\Delta x^k\|} \\
=\ & \sigma\|J(x^k)^T F(x^k)\|, \qquad (4.16)
\end{aligned}$$

where σ is a positive constant.

Finally, if Algorithm B does not use the Singular Case Subalgorithm at iteration k, then $\|J(x^k)^{-1}\|$ is uniformly bounded above from Lemma 4.13, and from the proof of Theorem 4.2,

$$\begin{aligned}
& \frac{|(J(x^k)^F(x^k))^T \lambda^k \Delta x^k|}{\|\lambda \Delta x^k\|} \\
\geq\ & \frac{\tau_1}{\tau_2}\frac{\|F(x^k)\|}{\|J(x^k)^{-1}\|} \geq \phi\|F(x^k)\| \\
\geq\ & \phi\frac{\|J(x^k)^T F(x^k)\|}{\|J(x^k)\|} \geq \hat{\phi}\|J(x^k)^T F(x^k)\| \qquad (4.17)
\end{aligned}$$

where the last inequality uses $\|J(x^k)\|$ bounded above, and ϕ, $\hat{\phi}$ are positive constants.

Combining (4.14), (4.15), (4.16) and (4.17) gives

$$\lim_{k\to\infty} J(x^k)^T F(x^k) = 0,$$

which completes the proof. □

5 Summary and discussion

We have shown that a computationally robust and efficiently parallelizable algorithm for solving block bordered systems of nonlinear equations, Algorithm B, is globally convergent under very mild conditions, in particular assuming only that the Jacobian matrix is Lipschitz continuous and uniformly bounded above. We have also shown that a simple sub-case of this method that is intended for well-conditioned problems, Algorithm A, is globally convergent under the stronger conditions that the Jacobian matrix and its diagonal blocks also are uniformly nonsingular. In addition, both algorithms are locally quadratically convergent on problems where the Jacobian matrix and its diagonal blocks are nonsingular and sufficiently well-conditioned at the solution.

One disadvantage of the new methods is that whereas a robust version of Newton's method is locally quadratically convergent on any problem where the Jacobian matrix at the solution is safely nonsingular, our algorithms also require the diagonal blocks to be safely nonsingular for fast local convergence. (Note that it is quite possible for the Jacobian matrix to be nonsingular even though a diagonal block is singular, see e.g. the example in [8].) The reason is that in order to make the computations on the diagonal blocks independent so that they can be parallelized efficiently, we make decisions about whether to perturb these blocks without knowing whether the overall Jacobian matrix is singular or ill-conditioned. A possible way to avoid this disadvantage would be to move any singularity that is detected in the diagonal blocks to the borders by a pivoting strategy. For example, one could use column pivoting to detect a maximal-size nonsingular submatrix of a given diagonal block A_i, and pivot the remaining columns and rows into the border. Such a strategy would not affect the global convergence properties of the algorithm, and would ensure that if the overall Jacobian matrix is nonsingular then the matrix does not have to be perturbed. Thus the algorithm would have strong global convergence properties and the same local convergence properties as a robust implementation of Newton's method. The disadvantages of such an approach are its added complexity, and the fact that if after pivoting extra rows to the border we discover that the enlarged Schur complement is still singular or ill-conditioned, then we have created extra computations and reduced parallel efficiency due to the increased size of the border.

Another possible disadvantage of our current approach that is not addressed by the discussion of the previous paragraph is that even if the whole Jacobian matrix and its the diagonal blocks are well-conditioned, the Schur complement could be ill-conditioned so that we might still have to perturb the matrix in some way. A simple example is

$$J = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & \alpha I & I & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & 0 & I \end{pmatrix},$$

in which $A_1 = I$, $A_2 = \alpha I$,

$$B = \begin{pmatrix} 0 & 0 \\ I & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix}.$$

It is easy to see that J, A_1 and A_2 are all well-conditioned. However, the Schur complement

$$P - CA^{-1}B = \begin{pmatrix} \alpha I & 0 \\ 0 & I \end{pmatrix},$$

can be arbitrarily ill-conditioned for small α. In our algorithms this example is not a problem since we have based our perturbations on the norms of the inverses of each A_i and of the Schur complement, but there may well be related examples where we would make perturbations but a robust Newton's method that works directly on the overall system would not. This is another possible price that we pay for working on the diagonal blocks independently, and making decisions about perturbing them independently, in order to attain good parallel efficiency.

Note finally that one unnatural aspect of Algorithm B is that in the singular case, it generates two intermediate trial steps, based on single and multiple inner iterations. An obvious question is whether one can produce stopping conditions for the inner iterations, as in Algorithm A, that would make this unnecessary. As we have discussed in section 3, this appears to be a challenging task and remains a topic for further research.

References

[1] C. Christara, Spline collocation methods, software and architecture for linear elliptic boundary value problems, Ph.D. dissertation, Computer Science Department, Purdue University (1988).

[2] C. Christara and E. Houstis, A domain decomposition spline collocation method for elliptic partial differential equations, in *Proc. 4th Conf. Hypercubes, Concurrent Computers and Applications*, Monterey, CA, (1989).

[3] J. E. Dennis Jr. and J. J. Moré, A characterization of superlinear convergence and its application to quasi-Newton methods, *Mathematics of Computation*, 28(1974), pp. 549-560.

[4] J. E. Dennis Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, New Jersey (1983).

[5] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM (1991).

[6] C. Farhat and E. Wilson, Concurrent iterative solution of large finite element systems, Tech. Report, Civil Engineering Department, University of California, Berkeley (1986).

[7] M. Mu and J. Rice, Solving linear systems with sparse matrices on hypercubes, Tech. Report CSD-TR-870, Computer Science Department, Purdue University (1989).

[8] X. Zhang, Parallel computation for the solution of nonlinear block bordered equations and their applications, Ph.D. dissertation, Department of Computer Science, University of Colorado (1989).

[9] X. Zhang, R. H. Byrd, R. B. Schnabel, Solving nonlinear block bordered circuit equations on hypercube multiprocessors, in *Pcoc. 4th Conf. Hypercubes, Concurrent Computers and Applications*, Monterey, CA, (1989), pp. 701-707.

[10] X. Zhang, R. H. Byrd, R. B. Schnabel, Parallel Methods for Solving Nonlinear Block Bordered Systems of Equations, *SIAM J. Sci. Stat. Comput.*, 13(1992), pp. 841-859.