A Parallel Adaptive Nonlinear Elimination Preconditioned Inexact Newton Method for Transonic Full Potential Equation

Feng-Nan Hwang[a,*], Yi-Cheng Su[a], Xiao-Chuan Cai[b]

[a] *Department of Mathematics, National Central University, Jhongli 32001, Taiwan*
[b] *Department of Computer Science, University of Colorado, Boulder, CO 80309, USA*

Abstract

We propose and study a right-preconditioned inexact Newton method for the numerical solution of large sparse nonlinear system of equations. The target applications are nonlinear problems whose derivatives have some local discontinuities such that the traditional inexact Newton method suffers from slow or no convergence even with globalization techniques. The proposed adaptive nonlinear elimination preconditioned inexact Newton method consists of three major ingredients: a subspace correction, a global update, and an adaptive partitioning strategy. The key idea is to remove the local high nonlinearity before performing the global Newton update. The partition used to define the subspace nonlinear problem is chosen adaptively based on the information derived from the intermediate Newton solution. Some numerical experiments are presented to demonstrate the robustness and efficiency of the algorithm compared to the classical inexact Newton method. Some parallel performance results obtained on a cluster of PCs are reported.

Key words: Transonic flow, adaptive nonlinear elimination, inexact Newton, local high nonlinearity, density upwinding finite difference, shock wave

1. Introduction

The class of inexact Newton method (IN) [8, 15] is popular for solving large sparse nonlinear system of equations arising from discretization of partial differential equations (PDEs). IN is quite robust and efficient for smooth nonlinear problems, but if the solution of the problem or its derivatives has certain discontinuity, the convergence rate of IN degrades, and the method may fail to converge even used together with globalization techniques, such as linesearch or trust region [8, 15]. Such problems appear often in computational fluid dynamics involving, for examples, shock waves, boundary layers, and corner singularities. To overcome the problem, we develop a nonlinear preconditioning technique in this paper.

Nonlinear preconditioning can be applied on the left or on the right of the nonlinear function. The basic idea of left preconditioning is to change the function of the system to a more balanced system and then solve the new system by IN. The additive Schwarz preconditioned inexact Newton algorithm (ASPIN) [3, 11] belongs to this class. ASPIN has been applied successfully to incompressible high Reynolds number flows [3, 4, 6, 11, 12], transonic compressible flows [5, 13, 20], flows in porous media [19], unconstrained optimization problems arising in nonlinear elasticity problems [9], and image processing [22]. On the other hand, right preconditioning is to modify the variable of the nonlinear system. For example, Hwang et al. [13] employs a nonlinear elimination (NE) technique [14] as a right preconditioner for a quasi one-dimensional shocked duct flow calculation. The key idea of NE is to implicitly remove these components that cause trouble for IN. In order to use the algorithm proposed in [13], one has to assume that the components to be eliminated are known in advance. However, in practice it may not always be possible to determine these components. The

*Corresponding author. Tel: +886-3-422-7151 Ext. 65110; Fax +886-3-425-7379
Email addresses: hwangf@math.ncu.edu.tw (Feng-Nan Hwang)

main contribution of this paper is to propose a new algorithm, namely an adaptive nonlinear elimination (ANE) preconditioned inexact Newton method that does not require this assumption. In the proposed algorithm, we use the intermediate IN solution to identify these components to be eliminated before a new global IN iteration. One potential application of the proposed algorithm is for the time-dependent PDE problems solved by a fully implicit scheme. In this paper, we focus only on a steady-state problem, namely the full potential equation in two different computational domain. The subspace correction phase can be done before a global Newton iteration is performed so that the overall performance of IN-based kernel solver is improved.

The rest of the paper is organized as follows. The next section describes the full potential equation discretized using a finite difference method with density upwinding. Section 3 provides a detailed description of the proposed algorithm. Section 4 presents the numerical results, including parallel performance of the proposed algorithm. Section 5 summarizes the main contributions of this paper and points out some potential applications of the algorithm.

2. Full potential flow equation and its discretization

We consider the full potential flow equation [10, 18], which is often used for modeling transonic flows passing an airfoil,

$$\nabla \cdot (\rho(\phi)\nabla\phi) = 0, \tag{1}$$

where ϕ is the velocity potential, $(u_1, u_2) = \nabla\phi$ is the velocity field, and the density function ρ is given as

$$\rho(\phi) = \rho_\infty \left(1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - \frac{\|\nabla\phi\|_2^2}{q_\infty^2})\right)^{1/(\gamma-1)}. \tag{2}$$

Here, $\gamma = 1.4$ is the specific heat for air. The constants ρ_∞, M_∞ and q_∞ represent the density, the Mach number, and the speed at the far field, respectively. In this work, two test cases, namely a flow passing the NACA0012 airfoil case [2, 18], and a channel flow passing through a circular bump [7, 16].

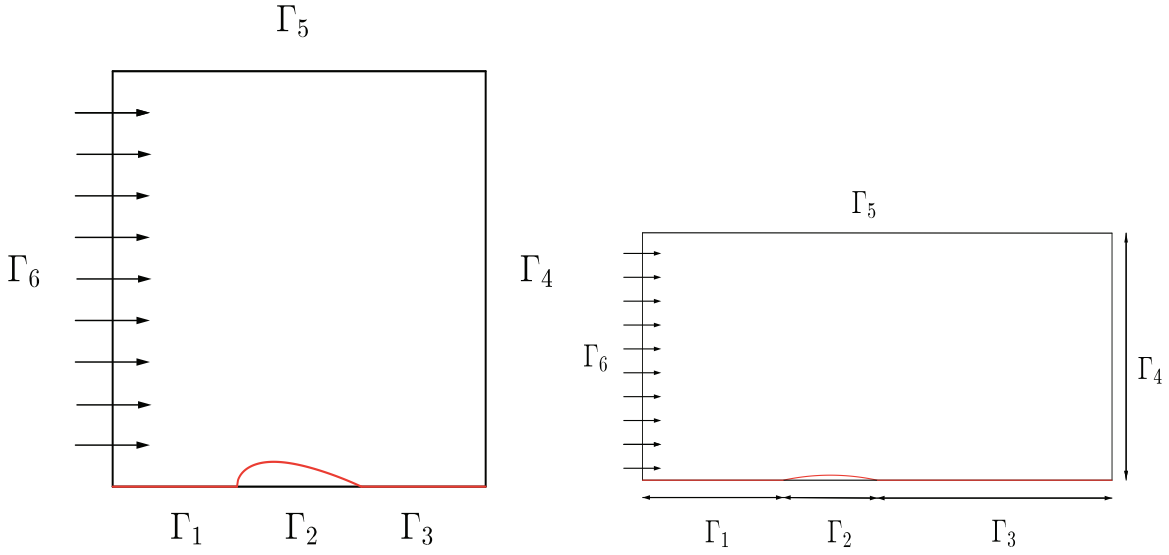

Figure 1: The geometrical configuration for the airfoil problem (left) and the internal channel flow with a circular bump problem (right)

The geometrical configuration for a transonic flow passing the NACA0012 airfoil is shown in the left figure of Figure 1, where the computational domain is $[0, 1] \times [0, 1]$ and the shape of the NACA0012 model

is described by the function

$$f(x) = 0.17814(\sqrt{x} - x) + 0.10128(x(1-x)) - 0.10968x^2(1-x) + 0.06090x^3(1-x),$$

for $x \in (0,1)$ and then re-scaled into $[1/3, 2/3]$ through $x = 3t - 1$, for $t \in [1/3, 2/3]$. The boundary conditions are specified as follows.

1. $\phi = 0$ on the inflow boundary Γ_6, $\phi = q_\infty$ on the outflow boundary Γ_4, and the freestream boundary on Γ_5, which are described by $\phi = \phi_\infty = \int_x q_\infty dx$. The freestream speed q_∞ is normalized to be 1.
2. A homogenous Neumann boundary condition is imposed on Γ_1, and Γ_3, i.e., $\frac{\partial \phi}{\partial y} = 0$. This condition implies that the flow is symmetric with respect to the boundaries and no flow penetrate through the boundaries.
3. A transpiration boundary condition is given on Γ_2 by $\frac{\partial \phi}{\partial y} = -q_\infty f'(x)$.

As a second example, we consider a channel flow as shown in the right figure of Figure 1. The computational domain is defined as $[-1.0, 4.0] \times [0.0, 2.073]$. The shape of the bump is described by the function.

$$f(x) = 4tx(1-x)$$

for $0 \le x \le 1$, $t = 0.042$. The settings of the boundary conditions are similar to the airfoil case, except that the freestream boundary condition on Γ_5 is replaced by a homogenous Neumann boundary condition as on Γ_1 and Γ_3 and $\phi = 0$ and $\phi = 1$ on Γ_6 and Γ_4, respectively.

To discretize (1) by a finite difference method with density upwinding [10], we begin by introducing a set of mesh points, (x_i, y_j), $0 \le i \le n_x$ and $0 \le j \le n_y$ with the mesh size $h_x = l_x/n_x$ and $h_y = l_y/n_y$, where l_x and l_y are the lengths of the computational domain in the x- and y-directions, respectively. Let $\Phi = [\phi_{i,j}]^T$ be the numerical approximations at mesh points (including the Dirichlet and Neumann boundary points) in the natural ordering. We denote $x_{i+1/2}$ and $y_{i+1/2}$, as the midpoints of subintervals $[x_i, x_{i+1}]$ and $[y_j, y_{j+1}]$, respectively. We discretize the full potential equation (1) at the interior point (x_i, y_j) using a second-order centered finite difference method, i.e.,

$$F_I(\Phi) \equiv h_y \left[\rho_{i+\frac{1}{2},j}(u_1)_{i+\frac{1}{2},j} - \rho_{i-\frac{1}{2},j}(u_1)_{i-\frac{1}{2},j}\right] + h_x \left[\rho_{i,j+\frac{1}{2}}(u_2)_{i,j+\frac{1}{2}} - \rho_{i,j-\frac{1}{2}}(u_2)_{i,j-\frac{1}{2}}\right] = 0,$$

where the velocity components u_1 and u_2 at $(x_{i+1/2}, y_j)$ and $(x_i, y_{j+1/2})$, respectively, are approximated as

$$(u_1)_{i+1/2,j} \approx (\phi_{i+1,j} - \phi_{i,j})/h_x \tag{3}$$

$$(u_2)_{i,j+1/2} \approx (\phi_{i,j+1} - \phi_{i,j})/h_y \tag{4}$$

and $(u_1)_{i-1/2,j}$ and $(u_2)_{i,j-1/2}$ are approximated similarly. For purely subsonic flows, using (2) for calculating the flow density at $(x_{i\pm1/2}, y_j)$ and $(x_i, y_{j\pm1/2})$, i.e., $\rho_{i\pm1/2,j} = \rho(\|q\|_{i\pm1/2,j})$ and $\rho_{i,j\pm1/2} = \rho(\|q\|_{i,j\pm1/2})$, is sufficient. Here, $q_{i+1/2,j} = \sqrt{(u_1)^2_{i+1/2,j} + (u_2)^2_{i+1/2,j}}$ and $q_{i,j+1/2} = \sqrt{(u_1)^2_{i,j+1/2} + (u_2)^2_{i,j+1/2}}$ with $(u_1)_{i+1/2,j}$ and $(u_2)_{i,j+1/2}$ defined as (3) and (4), and

$$\begin{aligned}(u_2)_{i+1/2,j} &\approx (\phi_{i+1,j+1} + \phi_{i,j+1} - \phi_{i+1,j} - \phi_{i,j})/(2h_y)\\ (u_1)_{i,j+1/2} &\approx (\phi_{i+1,j+1} + \phi_{i+1,j} - \phi_{i,j+1} - \phi_{i,j})/(2h_x).\end{aligned}$$

However, for transonic flows, this formulation needs to be modified in order to capture the shock. By applying a first-order density upwinding scheme as suggested by Young *et al.* [2, 21], a modified flow density value at the points $(x_{i+1/2}, y_j)$ and $(x_i, y_{j+1/2})$ are expressed as

$$\rho_{i+1/2,j} = \begin{cases} \rho_{i+1/2,j} - \widetilde{\mu}_{i,j}(\rho_{i+1/2,j} - \rho_{i-1/2,j}) & \text{if } (u_1)_{i+1/2,j} > 0, \\ \rho_{i+1/2,j} + \widetilde{\mu}_{i+1,j}(\rho_{i+1/2,j} - \rho_{i+3/2,j}) & \text{if } (u_1)_{i+1/2,j} < 0, \end{cases}$$

and

$$\rho_{i,j+1/2} = \begin{cases} \rho_{i,j+1/2} - \widetilde{\mu}_{i,j}(\rho_{i,j+1/2} - \rho_{i,j-1/2}) & \text{if } (u_2)_{i,j+1/2} > 0, \\ \rho_{i,j+1/2} + \widetilde{\mu}_{i,j+1}(\rho_{i,j+1/2} - \rho_{i,j+3/2}) & \text{if } (u_2)_{i,j+1/2} < 0, \end{cases}$$

where the first-order switching parameter $\widetilde{\mu}_{i,j}$ is defined as $\widetilde{\mu}_{i,j} = \max\{\mu_{s,t}\}$, for $s = i-1, i, i+1$ $(i = 1, ..., m)$ and $t = j-1, j, j+1$ $(j = 1, ..., n)$ with the zeroth-order switching parameter $\mu_{s,t} = \max\{0, 1 - \hat{M}_c^2/M_{s,t}^2\}$. Here $\hat{M}_c$ is called a pre-selected cutoff Mach number and $M_{s,t}$ is the numerical local Mach number at (x_s, y_t) given by

$$M_{s,t} \approx M_\infty(q_{s,t}/\rho_{s,t}^{\frac{1}{\gamma-1}}) \tag{5}$$

with $q_{s,t} = (q_{s+1/2,t} + q_{s-1/2,t} + q_{s,t+1/2} + q_{s,t-1/2})/4$ and $\rho_{s,t} = \rho(q_{s,t})$. To close the system, we impose the boundary condition at the leftmost and rightmost mesh points, for $j = 0, ..., (n_y - 1)$

$$F_L(\Phi) \equiv \phi_{0,j} = 0$$

and

$$F_R(\Phi) \equiv \phi_{n_x,j} - 1.0 = 0.$$

For the topmost mesh points, we have, for $i = 1, ..., (n_x - 1)$

$$F_T(\Phi) \equiv \phi_{i,n_y} - ih_x = 0, \text{ (The airfoil case)}$$

or

$$F_T(\Phi) \equiv \phi_{i,n_y} - \phi_{i,n_y-1} = 0, \text{ (The channel flows case).}$$

For the bottommost mesh points, when $1/3 \le ih_x \le 2/3$ for the NACA0012 case (or $0 \le ih_x \le 1$ for the channel flow case), we have

$$F_B(\Phi) \equiv \phi_{i,1} - \phi_{i,-1} - 2h_y q_\infty f'(jh_x) = 0$$

else

$$F_B(\Phi) \equiv \phi_{i,1} - \phi_{i,0} = 0,$$

for $i = 1, ..., (n_x - 1)$. Here, a horizontal layer of ghost points $(j = -1)$ is included to avoid unphysical solution. In summary, the discretized transonic full potential flow problem is written as a large sparse nonlinear system of algebraic equations,

$$F(x) = 0, \tag{6}$$

where $x = [\phi_{i,j}]^T$ is a solution vector containing the numerical approximations at the mesh points in the natural ordering. Here $F = (F_1, \ldots, F_n)^T$ and $F_i = F_i(x_1, \ldots, x_n)$.

3. Inexact Newton and nonlinear preconditioning

We briefly review the classical inexact Newton method with backtracking (INB) [8, 15] which will be used as the basis of the proposed algorithm.

Algorithm 1. [Inexact Newton with Backtracking (INB)]

Given an initial guess $x^{(0)}$
Evaluate $F(x^{(0)})$ *and* $\|F(x^{(0)})\|$
Set $k = 0$
While $\|F(x^{(k)})\| > \varepsilon_{global\_nonlinear\_atol}$ *do*
  Compute the Jacobian matrix $F'(x^{(k)})$
  Inexactly solve the Jacobian system $F'(x^{(k)})s^{(k)} = -F(x^{(k)})$
  Update $x^{(k+1)} = x^{(k)} + \lambda^{(k)}s^{(k)}$, *where* $\lambda^{(k)} \in (0, 1]$ *is determined to satisfy*
    $\|F(x^{(k)} + \lambda^{(k)}s^{(k)})\| \le (1 - \alpha\lambda^{(k)})\|F(x^{(k)})\|$
  Set $k = k + 1$
End While

In INB, we first find the search direction by solving inexactly the Jacobian system, then compute the next approximate solution along this search direction. How far we should go from the current approximation is determined by the damping scalar, $\lambda^{(k)}$. $\varepsilon_{global\_nonlinear\_atol}$ is the absolute tolerance and α is employed to assure that the reduction of $\|F(x)\|_2$ is sufficient. Observed from many numerical experiments, the size of $\lambda^{(k)}$ could be very small due to the existence of some bad components in the function $F(x)$. These bad components are often associated with certain interesting physics of the solution, e.g., the shock wave located in a small region for the transonic full potential flow problem. On the other hand, in our proposed algorithm, NE is a subproblem solver inside a global INB that is designed to smooth out these "bad components" so that the total number of global INB is reduced. It is important to note that by nonlinear elimination we change the sequence $x^{(0)}$, $x^{(1)}$, ..., but we do not change the final solution obtained by INB.

To define a nonlinear elimination based preconditioner, we introduce some notations. Let $S = \{1, 2, ..., n\}$ be an index set and each index corresponds to an unknown component x_i and a nonlinear residual component, F_i. We classify the nonlinear residual components, $F_1, F_2, ..., F_n$, into two groups for the "good components" and "bad components". Let k be the global Newton iteration number. Assume that $S_b^{(k)}$ ("b" for bad) is a subset of S with $m^{(k)}$ components and $S_g^{(k)}$ ("g" for good) with $(n - m^{(k)})$ components is its complement; that is

$$S = S_b^{(k)} \cup S_g^{(k)}. \tag{7}$$

Usually $m^{(k)} \ll n$. For this partition, we define two subspaces

$$V_b^{(k)} = \{v | v = (v_1, ..., v_n)^T \in R^n, v_i = 0 \text{ if } i \notin S_b^{(k)}\}$$

and

$$V_g^{(k)} = \{v | v = (v_1, ..., v_n)^T \in R^n, v_i = 0 \text{ if } i \notin S_g^{(k)}\},$$

respectively, and the corresponding restriction operators, $R_b^{(k)}$ and $R_g^{(k)}$, which map vectors from R^n to $V_b^{(k)}$ and $V_g^{(k)}$, respectively. Using the restriction operator $R_b^{(k)}$, we define a nonlinear function $F_{S_1^b} : R^n \to V_b^{(k)}$ as

$$F_{S_b^{(k)}}(x) = R_b^{(k)}(F(x)).$$

For any given $x \in R^n$, $T_b^{(k)}(x)$: $R^n \to V_b^{(k)}$ is defined as the solution of the following subspace nonlinear system,

$$F_{S_b^{(k)}}(R_g^{(k)} x + T_b^{(k)}(x)) = 0. \tag{8}$$

Using the subspace mapping functions, we introduce a new global nonlinear function,

$$y = G^{(k)}(x) \equiv R_g^{(k)} x + T_b^{(k)}(x).$$

Note that for a given x, the evaluation of $G^{(k)}(x)$ is not straightforward, a nonlinear system corresponding to the subspace V_1^b has to be solved using typically the classical INB algorithm. The adaptivity is reflected in the fact that this nonlinear function changes with k. Now INB in conjunction with ANE can be described as follows.

Algorithm 2. [INB-ANE]

Given an initial guess $x^{(0)}$.
Initialize the partition: $S_b^{(0)} = \emptyset$ *and* $S_g^{(0)} = S$, $k = 0$
Evaluate $F(x^{(0)})$ *and* $\|F(x^{(0)})\|$
While $(\|F(x^{(k)})\| > \varepsilon_{global\_nonlinear\_atol})$ *do*
Subspace correction:
Given $x^{(k)}$, *solve the subspace problem* $F_{S_b^{(k)}}(R_g^{(k)} x^{(k)} + T_b^{(k)}) = 0$ *for* $T_b^{(k)}$.

Compute $z = G^{(k)}(x^{(k)}) = R_g^{(k)} x^{(k)} + T_b^{(k)}$.

Global update:

Compute $J(z)$

Approximately solve $J(z)s^{(k)} = -F(z)$

Update $x^{(k+1)} = z + \lambda^{(k)} s^{(k)}$, *where* $\lambda^{(k)}$ *is determined to satisfy*

$\|F(z + \lambda^{(k)} s^{(k)}))\| \leq (1 - \alpha\lambda^{(k)})\|F(z)\|$

Determine a new partition: $S = S_b^{(k)} \cup S_g^{(k)}$.

Set $k = k + 1$

End While

Remarks:

1. The basic idea of INB-ANE is to approximately eliminate the local high nonlinearities before applying a global Newton iteration. Skipping the subspace correction phase, INB-ANE is reduced to the classical INB.
2. When the "bad" subspace $S_b^{(k)}$ is empty, it does not mean all the equations are good. It may indicate that all the equations are bad; bad in a global way.
3. A physics-based strategy can be used to determine the subset of indices containing all the bad components, $S_b^{(k)}$. For the applications considered in the paper, the bad components correspond to the transonic flow region. In our implementation, we calculate the local Mach number $M_{s,t}$ using (5) for each Newton iteration. The bad components are therefore defined by the local Mach number, if $M_c < M_{s,t}$, whereas the good components correspond to the subsonic flow region, i.e., $M_{s,t} < M_c$. Here, M_c is a pre-selected cuf-off Mach number for ANE. The effect of different values of M_c on the overall performance of INB-ANE will be investigated in Section 4.2.
4. The partition for the nonlinear elimination changes with k. As shown in Section 4.2 the bad components might not be clearly identified at the beginning, since the early approximate solution is far from the final solution. But as Newton progresses toward the final solution, the solution provides more accurate information for the partition.

An important feature of the right-nonlinear preconditioner is that there are some flexibilities to select the global Jacobian solver and it is easier to develop preconditioning strategies for the linear part of the algorithm to enhance the overall scalability of the algorithm. To define the parallel restricted Schwarz preconditioner for the global Jacobian system, we introduce another non-overlapping partition of the index set S defined as in Section 3.2, i.e.,

$$S = \cup_{i=1}^{N_s} S_i, \ S_i \cap S_j = \emptyset \text{ if } i \neq j, \text{ and } S_i \subset S,$$

where n_i is the dimension of S_i and $\sum_{i=1}^{N_s} n_i = n$. Here N_s is the number of processors of the parallel computer. Note that this partition is needed for the Schwarz preconditioning, and it is not related to the "good/bad" partition, which is for the purpose of removing the local high nonlinearity. To obtain overlapping subdomains, we expand each S_i to a larger S_i^δ, Here δ is an integer indicating the level of overlap. The meaning of δ is explained as follows. S_i^δ contains the mesh points in S, whose distance to S_i is less than or equal to δ and the all points in S_i. When $\delta = 0$, we have $S_i^0 = S_i$. Using the overlapping partition of S, we introduce some subspaces of R^n and the corresponding restriction and extension operators. For each S_i^δ, we define $V_i^\delta \subset R^n$ as

$$V_i^\delta = \{v | v = (v_1, v_2, ..., v_n)^T \in R^n, v_k = 0 \text{ if } k \notin S_i^\delta\}$$

and a $n \times n$ restriction matrix and also an extension matrix, R_i^δ, whose diagonal element, $(R_i^\delta)_{kk} = 1$ if $k \in S_i^\delta$; otherwise, $(R_i^\delta)_{kk} = 0$. Using the restriction and extension matrices, we define the one-level restricted additive Schwarz preconditioner as

$$M_{RAS1}^{-1} = \sum_{i=1}^{N_s} R_i^0 J_i^{-1} R_i^\delta, \tag{9}$$

where J_i is the subdomain Jacobian matrix evaluated at $x^{(k)}$, $J_i = R_i^\delta J(x^{(k)}) R_i^\delta$. To enhance the scalability of the algorithm, a coarse space is necessary. Let S^c be an index set corresponding to the coarse mesh points and V^c be the coarse subspace of V. Similar for the subdomains, we introduce another pair of restriction and extension operator to map the data between the spaces of coarse mesh and fine mesh, V^c and V, denoted by I_h^H and I_H^h, respectively. Here, the multiplication of the symmetrized multiplicative two-level restricted Schwarz preconditioner with a given vector r is carried out in the following steps:

1. $u^{(1)} \leftarrow \sum_{i=1}^{N_s} R_i^0 J_i^{-1} R_i^\delta r$
2. $r^{(0)} \leftarrow I_h^H (r - J u^{(1)})$
3. $u^{(0)} \leftarrow J_c^{-1} r^{(0)}$
4. $u^{(1)} \leftarrow u^{(1)} + I_H^h u^{(0)}$
5. $u^{(1)} + \sum_{i=1}^{N_s} R_i^0 J_i^{-1} R_i^\delta (r - J u^{(1)})$

Here, J_c is the coarse Jacobian matrix obtained by the Galerkin formulation, i.e. $J_c = I_h^H J I_H^h$. The above procedure can be understood as a two-grid correction scheme, where the one-level restricted additive Schwarz preconditioner is used as the pre- and post-smoothers.

4. Numerical results and discussion

In this section, we present some numerical results for solving the transonic full potential flow problem (1) using the classical inexact Newton method and the new algorithm proposed in the previous section. The stopping condition

$$\|F(x^{(k)})\| \leq 10^{-8},$$

and a zero initial guess are used for both methods and for all test cases. For the INB-ANE algorithm, the subspace nonlinear problem,

$$G(z) \equiv F_{S_b^{(k)}}(R_g^{(k)} x + z) = 0.$$

is inexactly solved by INB until the stopping condition,

$$\|G(z^{(l)}))\| \leq \max\{\varepsilon_{local\_nonlinear\_rtol} \|G(z^{(0)})\|, 10^{-10}\},$$

is satisfied. A left-preconditioned restarted GMRES [17] is used for solving the global Jacobian system with zero initial guess and the restarting number is set to be 200. The stopping condition for GMRES is

$$\|M_k^{-1}(F(x^{(k)}) + J(x^{(k)}))s^{(k)}\| \leq \max\{\eta \|F(x^{(k)})\|, 10^{-10}\}.$$

Here $\eta = 10^{-6}$ and M_k^{-1} is the one-level or symmetrized multiplicative two-level restricted Schwarz preconditioner. In the tests, we partition the computational domain in the regular checkerboard fashion. Each subdomain problem is assigned to a compute core and is solved by the sparse direct LU decomposition. The global Newton step is updated by

$$x^{(k+1)} = x^{(k)} + \lambda^{(k)} s^{(k)}.$$

The step length, $\lambda^{(k)} \in [\lambda_{\min}, \lambda_{\max}] \subset (0, 1]$, is selected so that

$$\|F(x^{(k)} + \lambda^{(k)} s^{(k)})\| \leq (1 - \alpha \lambda^{(k)}) \|F(x^{(k)})\|,$$

where the two parameters $\lambda_{\min}$ and $\lambda_{\max}$ act as safeguards, which are required for strong global convergence and the parameter α is used to assure that the reduction of $\|F\|$ is sufficient. Here, a cubic linesearch technique [8] is employed to determine the step length $\lambda^{(k)}$, with $\alpha = 10^{-4}$, $\lambda_{\min} = 1/10$ and $\lambda_{\max} = 1/2$. We use the Portable, Extensible Toolkits for Scientific computation (PETSc) [1] for the parallel implementation. The numerical results are obtained on a cluster of computers.

4.1. Validation of algorithm and software

In the case of linear preconditioning, it is obvious that the preconditioned system and the original system have the same solution, but in the case of nonlinear preconditioning, the situation is not obvious, especially for the full potential equation whose uniqueness theory has not been established in the transonic regime. To validate the proposed approach, we first perform a few experiments for different values of $M_\infty = 0.1, 0.3, 0.5,$ and 0.8 for the airfoil case. The mesh size changes from $h = 1/64$ to $h = 1/1024$. The nonlinear systems are solved in parallel with the number of processors (np), $np = 16$. Figure 2 shows the computed potential curves along the vertical midline $x = 0.5$ and it is clear that the computed solutions converge as the mesh is refined. Figure 3 shows a comparison of the potential contours, the Mach number contours and the pressure coefficients along the airfoil obtained from the numerical solutions by INB (left) and INB-ANE (right), where the pressure coefficient, C_p, is calculated using

$$C_p = \frac{2}{\gamma M_\infty^2}\left(\left[1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - q^2)\right]^{\gamma/(\gamma-1)} - 1\right).$$

We see that these two set of results are almost indistinguishable. This provides a numerical evidence that the numerical solution is not altered with the proposed nonlinear preconditioner. In addition, Figures 4 and 5 provide a comparison of the Mach number contour and the pressure coefficient distribution for the channel flow case along the circular bump wall with four different values of M_∞, i.e., 0.8, 0.835, 0.8435, and 0.85. For the $M_\infty = 0.8$ case, almost all region is subsonic. As the value of M_∞ increases to near the critical value, e.g., $M_\infty = 0.835$, a shock appears near the circular bump.

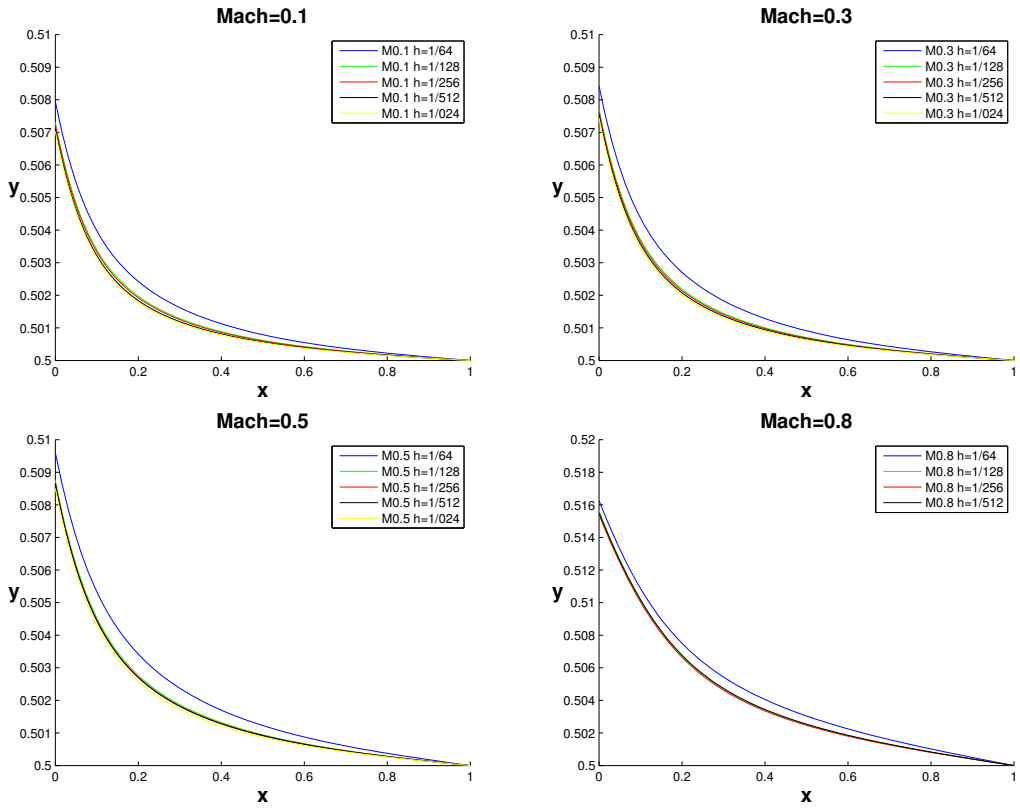

Figure 2: The airfoil case. Computed potential curves along the vertical midline at $x = 0.5$ for different values of $M_\infty = 0.1$ (top-left), 0.3 (top-right) 0.5 (bottom-left), and 0.8 (bottom-right) and different mesh sizes.

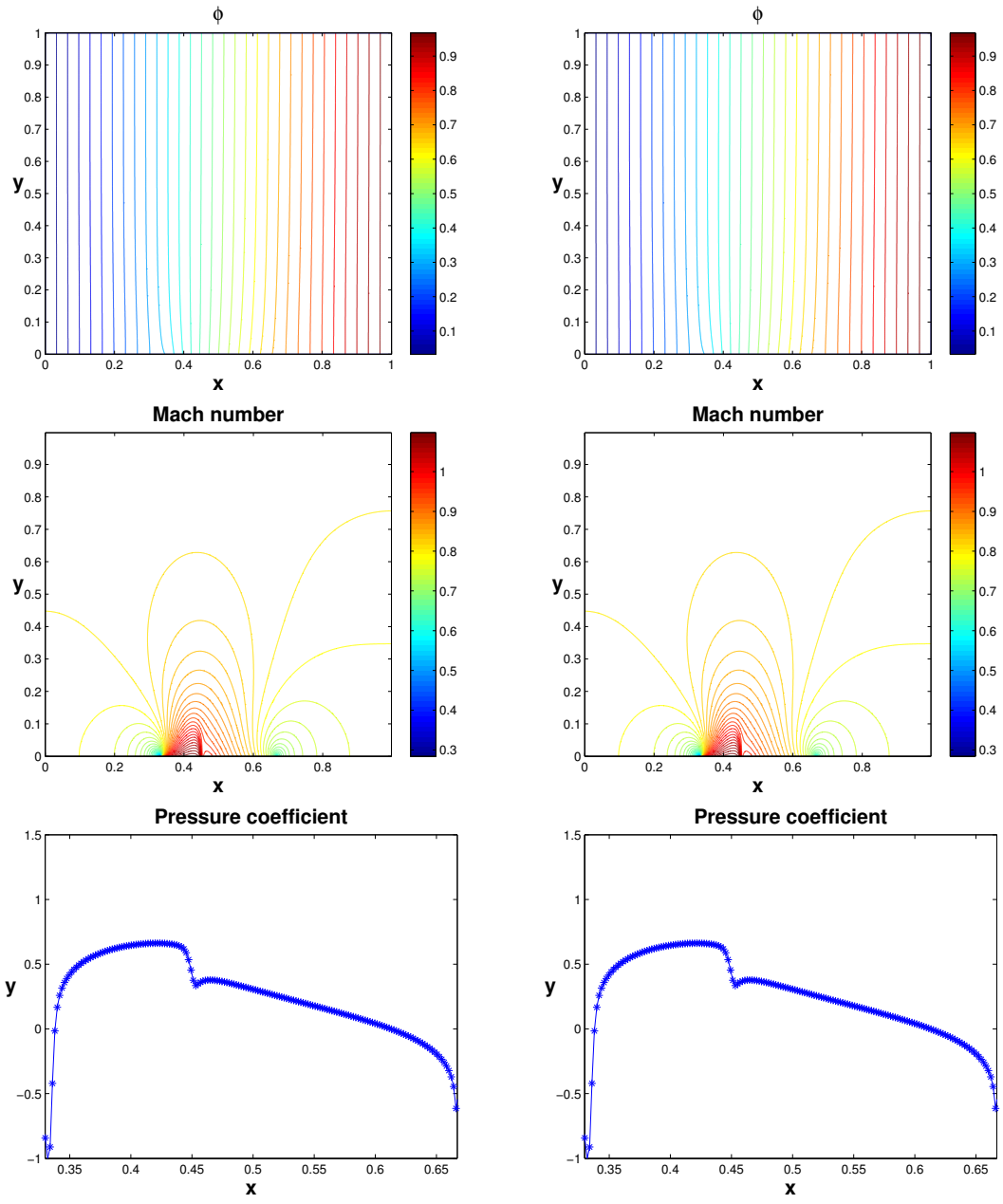

Figure 3: The airfoil case. A comparison of potential contours (1^{st} row), the Mach number contours (2^{nd} row), and the pressure coefficients (3^{rd} row) obtained by the numerical solution using INB (left column) and INB-ANE (right column). $M_\infty = 0.8$ and $h = 1/512$.

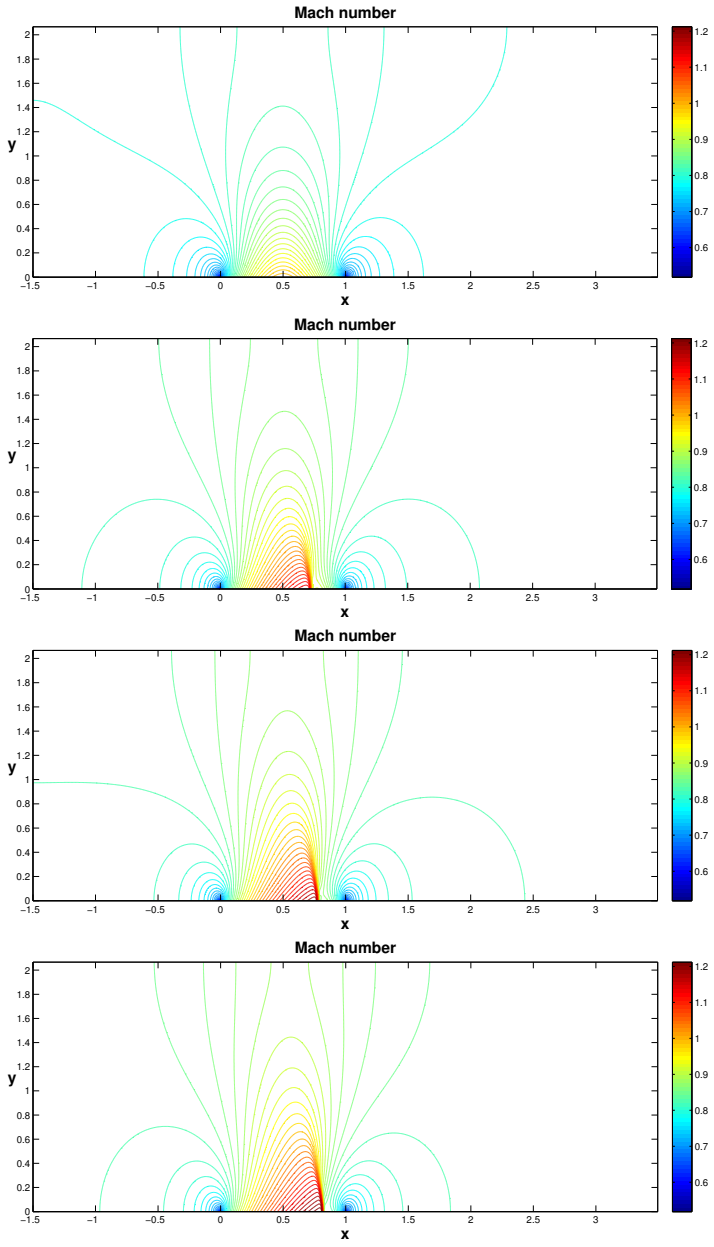

Figure 4: The channel flow case. The Mach number contours for $M_\infty = 0.8, 0.835, 0.8435$, and 0.85 (from top to bottom) obtained by the numerical solution using INB-ANE on a mesh with $h = 1/160$.

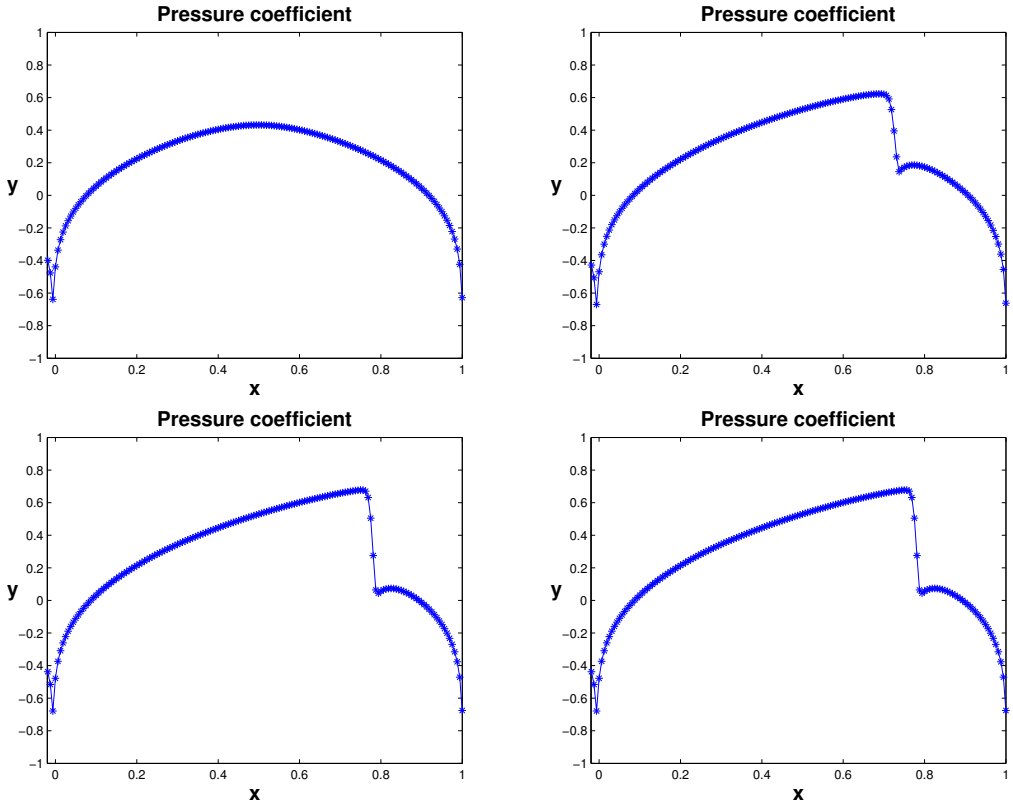

Figure 5: The channel flow case. The plots of the pressure coefficients for $M_\infty = 0.8$ (top-left), $M_\infty = 0.835$ (top-right), $M_\infty = 0.8435$ (bottom-left), and $M_\infty = 0.85$ (bottom-right) obtained by the numerical solution using INB-ANE on a mesh with $h = 1/160$.

4.2. A comparison of INB and INB-ANE

In Table 1, we summarize the number of Newton iterations on five different meshes, 1/64, 1/128, 1/256, 1/512, and 1/1024 with four different values of $M_\infty = 0.1, 0.3, 0.5$, and 0.8 for the airfoil case and with four different values of $M_\infty = 0.8$, 0.835, 0.8435, and 0.85 for the channel flow case. As expected, INB works quite well for the cases of subsonic flows, $M_\infty = 0.1, 0.3, 0.5$ (the airfoil case) and $M_\infty = 0.8$ (the channel flow case), which are mathematically classified as nonlinear elliptic problems. On the other hand, for the airfoil case, when a shock shows up in the solution when $M_\infty = 0.8$, the number of INB iterations increases quickly as we increase the resolution. When the strong discontinuity of the solution is resolved, INB often fails to converge. Note that for the channel flow case with M_∞=0.85, the transonic region covers almost everywhere. Hence, the cost for the subspace correction phase in INB-ANE becomes too high. For this case, the INB-RAS algorithm proposed in [6] might be more appropriate. In the following discussion, we restrict ourselves to the cases of $M_\infty = 0.835$ and 0.8435.

	Airfoil case				
Mesh sizes (h)	1/64	1/128	1/256	1/512	1/1024
$M_\infty = 0.1$	2	4	4	4	4
$M_\infty = 0.3$	3	4	3	4	4
$M_\infty = 0.5$	3	4	3	4	4
$M_\infty = 0.8$	7	10	17	31	F
	Channel flow case				
Mesh sizes (h)	1/40	1/80	1/160	1/320	
$M_\infty = 0.8$	6	6	5	6	
$M_\infty = 0.835$	22	37	74	F	
$M_\infty = 0.8435$	32	55	109	F	
$M_\infty = 0.85$	42	82	F	F	

Table 1: Number of Newton iterations when using INB for different values of M_∞ and different mesh size. "F" means that INB fails to converge.

We next focus on the cases that INB has some trouble to converge to the desired solution. Figure 6 shows the histories of the nonlinear residuals of INB and INB-ANE with different mesh sizes for both test cases. We observe from the figures that (1) INB-ANE converges for all mesh sizes; and (2) the number of Newton iterations is independent of the mesh sizes.

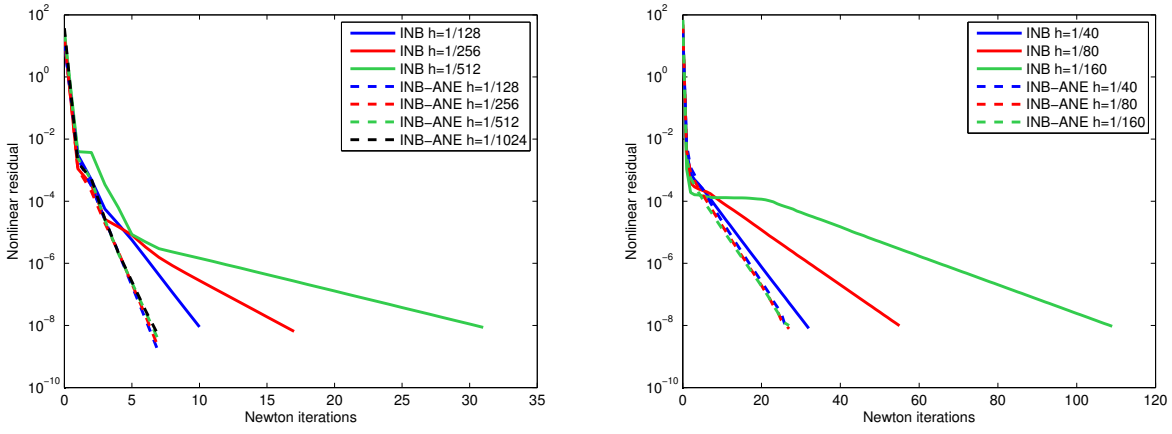

Figure 6: Convergence histories of INB and INB-ANE for different mesh sizes, the airfoil case, M_∞=0.8 (left) and the channel flow case, M_∞=0.8435 (right).

We further investigate how the "bad" region changes during the iterations of NB-ANE. As shown in Figure 7 for the case of $h = 1/256$. the distribution of the bad components to be eliminated is marked

as red in the first six global Newton iterations. Note that the total number of bad components remains unchanged since the 5th global nonlinear iteration. We observe that by comparing with the converged final Mach number contour plots in Figure 3, INB-ANE is able to identify the transonic region correctly within the first few iterations by computing the local Mach number obtained from intermediate Newton solutions. Furthermore, the percentages of the number of bad components compared to the total number of unknowns are quite low, only about 5%. Hence the overhead due to the subspace solution is relatively small. On the other hand, Figure 8 shows the bad components at the final iterations for the channel flow case, $M_\infty = 0.835$ and 0.8435. Obviously, the percentages of bad components are about 20% for these cases, which is larger than the airfoil case.

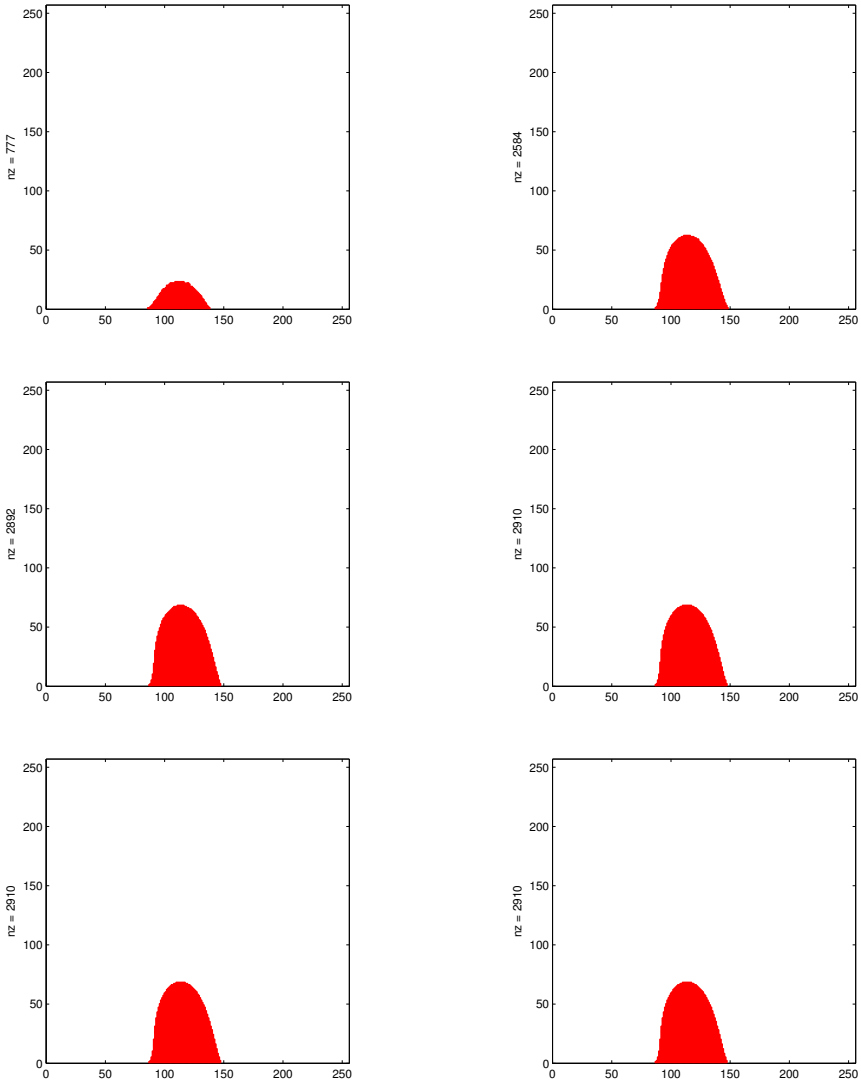

Figure 7: The airfoil case. The evolution of the distribution of the "bad" components during the global Newton iterations. The red area means the collection of the bad components. Mesh size, $h = 1/256$. "nz" is the total number of bad components.

Next we study other parameters that impact the performance of INB-ANE. These parameters include the cutoff Mach number (M_c) (Table 2) and the subspace nonlinear stopping condition ($\varepsilon_{subspace-nonlinear-rtol}$)

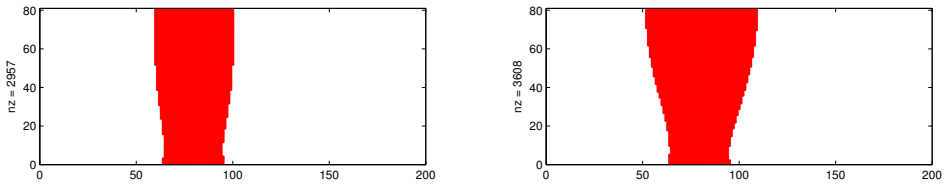

Figure 8: The channel flow case. The final distribution of the bad components. $h = 1/40$ with $M_\infty = 0.835$ (left) and $M_\infty = 0.8435$ (right), $M_c = 0.85$. The red region means the collection of bad components.

(Table 3).

The major role of M_c is to balance the costs of solving the global and subspace nonlinear problems. When the value of M_c is large, the number of global Newton iterations decreases. But the dimension of the subspace nonlinear problem increases, hence the cost for solving subspace nonlinear problems increases. Observed from the table, we find that $M_c = 0.85$ is appropriate for most cases that compromises the costs between the global and subspace solves to minimize the overall computing time.

From the numerical experiments, we find that the number of global Newton iterations is not sensitive to the stopping condition for the subspace nonlinear problem ($\varepsilon_{subspace-nonlinear-rtol}$). A very loose tolerance, say 10^{-1} is sufficient. This is very different from the case of left nonlinear preconditioning, e.g. ASPIN [3, 11]. For ASPIN, the subspace nonlinear problems need to be solved accurately enough in order not to change the solution of targeted nonlinear problems. It is worth mentioning that even performing only one Newton iteration for the subspace nonlinear problem helps the overall convergence of the global Newton iterations comparing to INB, sometimes it is even faster than the case with several more subspace Newton iterations. However, some dependency on the mesh size is observed. Hence, for numerical experiments reported in the next section, we select $\varepsilon_{subspace-nonlinear-rtol}$ to be 10^{-1} with $M_c = 0.85$.

4.3. Parallel performance study

An important feature of INB-ANE is that the selections of linear preconditioners for the solution of the global/subspace Jacobian systems are more flexible than ASPIN, since we do not have to worry about changing the solution of the original nonlinear system. Here, we consider two preconditioned GMRES for solving the global and local Jacobian systems, i.e. one-level restricted additive Schwarz and two-level Schwarz preconditioners. For the two-level method, a redundant sparse direct approach is used for the solution of the coarse mesh problem, i.e., the coarse mesh problem is distributed to each processor and solved redundantly by the LU decomposition. From Table 4 to Table 7, we summarize the performance of one- and two-level INB-ANE as well as INB if it converges successfully (Tables 4 and 5 for the NACA0012 case; Tables 6 and 7 for the channel flow case). The tables present the total computing time, the number of global nonlinear iterations and the number of subspace nonlinear iterations, and in the parenthesis, the average number of iterations for solving the global and subspace Jacobian systems, respectively.

Several observations are summarized as follows.

1. As expected, both INB and INB-ANE are nonlinearly scalable but not linearly scalable without a coarse space, in terms of the number of iterations. On the other hand, when including a coarse mesh correction, the number of iterations for solving the Jacobian systems is almost independent of the number of processors in the two-level INB-ANE algorithm.

		INB	**INB-ANE**		
Mesh sizes (h)		$M_c =$	0.8	0.85	0.9
Airfoil case, $M_\infty = 0.8$					
1/128	global	10(33.7)	5(33.6)	7(34.4)	8(34.5)
	subspace	—	10(22.0)	11(6.3)	13(3.0)
	time (secs)	0.5	0.7	1.8	0.7
1/256	global	17(45.1)	6(46.0)	7(47.7)	8(47.0)
	subspace	—	20(29.3)	23(7.7)	23(4.8)
	time (secs)	4.7	5.6	4.7	4.5
1/512	global	31(61.0)	6(62.2)	7(64.6)	9(64.4)
	subspace	—	32(38.9)	41(10.1)	49(5.6)
	time (secs)	55.1	52.6	37.0	39.5
Channel flow case, $M_\infty = 0.835$					
Mesh sizes (h)		$M_c =$	0.85	0.87	0.90
1/40	global	22(33.8)	15(34.5)	17(33.8)	21(33.8)
	subspace	—	28(13.5)	29(9.8)	36(5.9)
	time (secs)	1.6	2.6	2.6	3.0
1/80	global	37(44.9)	15(46.5)	17(46.4)	23(45.4)
	subspace	—	58(17.0)	55(12.8)	60(6.4)
	time (secs)	16.8	20.3	19.2	20.6
1/160	global	71(61.1)	15(63.4)	17(62.5)	24(61.0)
	subspace	—	107(22.4)	111(16.0)	118(7.2)
	time (secs)	215.7	195.8	182.2	161.0
Channel flow case, $M_\infty = 0.8435$					
Mesh sizes (h)		$M_c =$	0.85	0.87	0.90
1/40	global	32(34.8)	27(34.7)	30(34.8)	33(34.8)
	subspace	—	50(14.6)	53(12.6)	36(6.8)
	time	2.6	4.6	4.8	4.0
1/80	global	56(46.0)	27(46.8)	39(46.8)	54(46.1)
	subspace	—	96(18.7)	85(15.7)	67(9.2)
	time	25.5	37.9	37.9	36.6
1/160	global	109(62.2)	27(63.7)	64(63.8)	88(62.9)
	subspace	—	190(23.2)	165(20.3)	173(11.6)
	time	350.7	375.4	401.6	429.7

Table 2: A comparison of the cutoff Mach number for INB-ANE for the airfoil case and the channel flow case for different mesh sizes.

		INB	**INB-ANE**				
Mesh sizes (h)		$\varepsilon_{subspace-nonlinear-rtol}$ =	10^{-6}	10^{-4}	10^{-2}	10^{-1}	Only 1 ite
Airfoil case $M_\infty = 0.8$							
1/128	global	10(33.7)	6(34.3)	6(34.3)	6(34.3)	7(34.4)	7(34.6)
	subspace	—	43(6.0)	30(5.9)	17(5.9)	11(6.3)	6(6.2)
	time (secs)	0.5	1.3	1.0	0.7	1.8	0.5
1/256	global	17(45.1)	6(47.5)	6(47.5)	6(47.5)	7(47.7)	10(64.9)
	subspace	—	88(7.7)	59(7.7)	32(7.7)	23(7.7)	9(7.7)
	time (secs)	4.7	11.2	8.1	5.3	4.7	4.3
1/512	global	31(61.0)	6(64.0)	6(64.0)	6(64.0)	7(64.6)	18(63.1)
	subspace	—	194(10.0)	128(10.0)	61(10.0)	41(10.1)	17(10.0)
	time (secs)	55.1	130.0	87.7	48.4	37.0	44.5
Channel flow case, $M_\infty = 0.835$							
1/40	global	22(33.8)	15(34.5)	15(34.5)	15(34.5)	15(34.5)	16(33.9)
	subspace	—	65(13.4)	62(13.5)	45(13.6)	28(13.5)	15(13.5)
	time (secs)	1.6	4.0	3.9	3.1	2.6	1.9
1/80	global	37(44.9)	15(46.5)	15(46.5)	15(46.5)	15(46.5)	22(46.6)
	subspace	—	129(17.0)	125(17.0)	89(17.0)	58(17.0)	21(17.4)
	time (secs)	16.8	37.2	36.2	28.6	20.3	15.2
1/160	global	71(61.1)	14(63.4)	14(63.4)	14(63.4)	15(63.4)	35(62.8)
	subspace	—	243(22.3)	240(22.3)	172(22.3)	107(22.4)	34(22.7)
	time (secs)	215.7	353.6	352.5	264.1	195.8	171.6
Channel flow case, $M_\infty = 0.8435$							
1/40	global	32(34.8)	27(34.7)	27(34.7)	26(34.7)	27(34.7)	27(34.9)
	subspace	—	101(14.5)	99(14.5)	70(14.6)	50(14.6)	26(14.7)
	time (secs)	2.6	6.7	6.5	5.2	4.6	3.5
1/80	global	56(46.0)	26(46.8)	26(46.8)	26(46.8)	27(46.8)	35(46.8)
	subspace	—	217(18.2)	212(18.3)	147(18.4)	96(18.7)	34(18.7)
	time (secs)	25.5	64.9	66.1	49.7	37.9	24.8
1/160	global	109(62.2)	26(63.7)	26(63.7)	26(63.7)	27(63.7)	58(61.9)
	subspace	—	427(22.6)	422(22.7)	298(22.8)	190(23.2)	57(24.0)
	time (secs)	350.7	688.8	681.5	506.6	375.4	293.7

Table 3: The airfoil case. Mesh sizes, $h = 1/128$ $1/256$ and $1/512$ with different values of M_∞. A comparison of subspace nonlinear stopping conditions for the one-level INB-ANE.

2. In general, for a fixed number of processors, the two-level INB-ANE is about 33% or more faster than INB for the airfoil case, and 13% faster for the channel flow case. The only exception is the case of channel flow, $M_\infty = 0.8435$ with $h = 1/160$. Notice that for $M_\infty = 0.8435$, the number of INB iterations is quite sensitive to the Jacobian solution, which is the Newton search direction. The number of Newton iterations is down to 26 from 109, when the number of processors increases from 40 to 160. Hence, the gain by using nonlinear preconditioning is marginal.
3. If the problem size is not large enough, e.g., the case shown in Tables 4 and 6, the parallel performance of the two-level INB-ANE with 128 processors is degraded. The benefit of using the two-level method for problems of larger size is more obvious. The main reason is that the two-level INB-ANE method consists of a sequential component, i.e., a redundant LU solve is employed as the coarse mesh solver, and the cost of that is not negligible for smaller problems. A scalable parallel coarse mesh solver is needed. In addition, the issue of load balancing is important in parallel computing, but is not addressed in this paper. For the ease of implementation, we do not redistribute dynamically the subspace nonlinear problems. As a result, the heavy computing is concentrated on a subset of processors. The parallel efficiency of the algorithm is expected to be improved when the subspace nonlinear problem is evenly redistributed to processors and then solved in parallel.

	INB		**One-level INB-ANE**			**Two-level INB-ANE**		
np	its	time	global its	subspace its	time	global its	subspace its	time
4	31(39.8)	271.9	7(40.0)	41(9.8)	119.8	7(6.0)	41(9.8)	112.9
16	31(61.0)	55.1	7(64.6)	41(10.1)	37.0	7(6.3)	41(10.1)	33.8
32	31(83.7)	34.9	7(89.3)	41(19.2)	24.2	7(6.3)	41(19.3)	22.0
64	31(90.0)	17.6	7(97.0)	41(19.5)	13.0	7(6.3)	41(19.5)	11.3
128	31(123.9)	12.1	7(132.1)	41(29.6)	8.3	7(8.0)	41(30.4)	8.3

Table 4: The airfoil case. Mesh size $h = 1/512$ (and $H = 1/64$ for the two-level method) with $M_\infty = 0.8$. A comparison of the parallel performance for INB, one-level INB-ANE, and two-level INB-ANE.

	One-level INB-ANE			**Two-level INB-ANE** ($H = 1/128$)			**Two-level INB-ANE** ($H = 1/32$)		
np	global its	subspace its	time	global its	subspace its	time	global its	subspace its	time
4	7(53.9)	87(12.8)	1259.5	7(6.1)	86(12.8)	984.6	7(10.1)	89(12.8)	1101.9
16	7(88.0)	86(13.1)	392.4	7(6.1)	85(13.1)	358.6	7(10.7)	89(13.2)	372.3
32	7(121.9)	87(25.4)	261.2	7(6.6)	85(25.4)	255.0	7(11.1)	89(25.4)	266.4
64	7(132.1)	84(25.6)	133.8	7(6.7)	88(25.7)	118.5	7(11.4)	89(26.5)	126.4
128	7(187.1)	87(42.6)	83.0	7(8.3)	85(42.5)	77.2	7(13.1)	89(42.5)	80.8
256	7(209.4)	88(43.6)	41.2	7(8.7)	87(43.5)	42.9	7(14.7)	90(43.6)	38.8

Table 5: The airfoil case. Mesh size $h = 1/1024$ ($H = 1/128$ and $H = 1/32$ for the two-level method) with $M_\infty = 0.8$. A comparison of the parallel performance for one-level INB-ANE, and two-level INB-ANE.

	INB		**One-level INB-ANE**			**Two-level INB-ANE**		
				$M_\infty = 0.835$				
np	its	time	global its	subspace its	time	global its	subspace its	time
10	71(61.1)	215.7	15(63.4)	107(22.4)	195.8	15(11.5)	118(22.3)	181.7
40	69(103.6)	79.7	15(107.7)	107(33.7)	70.7	15(12.5)	118(33.7)	61.1
160	72(165.8)	31.5	15(166.5)	107(61.4)	25.4	15(22.7)	118(61.4)	25.7
				$M_\infty = 0.8435$				
10	109(62.2)	350.7	27(63.7)	190(23.2)	375.4	26(12.1)	205(23.1)	351.5
40	109(106.1)	137.3	27(108.9)	190(38.2)	129.8	26(16.4)	205(38.2)	111.9
160	26(173.5)	13.3	9(190.8)	39(70.6)	13.0	9(23.8)	41(70.9)	12.4

Table 6: The channel flow case. Mesh size $h = 1/160$ and $H = 1/20$ for the two-level method with $M_\infty = 0.835$ and 0.8435. A comparison of the parallel performance for INB, one-level INB-ANE, and two-level INB-ANE.

	One-level INB-ANE			Two-level INB-ANE		
			$M_\infty = 0.835$			
np	global its	subspace its	time	global its	subspace its	time
10	15(236.7)	208(29.5)	2440.9	14(10.4)	224(29.4)	1957.3
40	15(186.8)	208(44.2)	750.3	14(13.8)	224(44.2)	666.7
160	15(341.2)	208(92.5)	305.3	14(27.4)	224(92.4)	271.7
			$M_\infty = 0.8435$			
10	28(255.7)	403(30.7)	7294.7	26(11.0)	397(30.2)	4301.0
40	29(233.9)	366(50.4)	1552.6	26(16.1)	409(50.4)	1279.0
160	10(400.1)	78(107.3)	153.1	11(29.1)	84(109.5)	119.5

Table 7: The channel flow case. Mesh size $h = 1/320$ ($H = 1/40$ for the two-level method) with $M_\infty = 0.835$ and 0.8435. A comparison of the parallel performance for INB, one-level INB-ANE, and two-level INB-ANE.

5. Conclusions

In this work, we proposed a new algorithm, namely a parallel adaptive nonlinear elimination preconditioned inexact Newton method. The key idea of the method is to remove the unbalanced nonlinearity before performing a global Newton update. The effective identification of the bad components to be eliminated plays an important role in the success of the algorithm. We use information obtained from an intermediate solution of INB to select adaptively these to-be-eliminated components. For some transonic flow problems we studied the parallel performance of the new algorithm and found that INB-ANE is nonlinearly scalable with respect to the number of processors. Due to the flexibility for the selection of the global Jacobian solver, near linear scalability is achieved by a two-level Schwarz preconditioned Krylov subspace method. For the transonic full potential flow problems considered in this paper, the criteria for determining the bad components is based on a physical approach that identifies the components corresponding the transonic flow region. But for other applications, different criteria may be necessary.

Acknowledgements

The first two authors were supported in part by the Ministry of Science and Technology of Taiwan, NSC-100-2115-M-008-008-MY2 and the last author was supported in part by NSF, DMS-0913089 and CCF-1216314, and DOE under grant DE-SC0001994.

References

[1] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L. Curfman McInnes, B.F. Smith, and H. Zhang. PETSc Webpage, 2013. http://www.mcs.anl.gov/petsc.

[2] X.-C. Cai, W.D. Gropp, D.E. Keyes, R.G. Melvin, and D.P. Young. Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation. *SIAM J. Sci. Comput.*, 19:246–265, 1998.

[3] X.-C. Cai and D.E. Keyes. Nonlinearly preconditioned inexact Newton algorithms. *SIAM J. Sci. Comput.*, 24:183–200, 2002.

[4] X.-C. Cai, D.E. Keyes, and L. Marcinkowski. Nonlinear additive Schwarz preconditioners and applications in computational fluid dynamics. *Int. J. Numer. Meth. Fluids*, 40:1463–1470, 2002.

[5] X.-C. Cai, D.E. Keyes, and D.P. Young. A nonlinear additive Schwarz preconditioned inexact Newton method for shocked duct flow. In *Domain Decomposition Methods in Science and Engineering*. CIMNE, 2002.

[6] X.-C. Cai and X. Li. Inexact Newton methods with restricted additive Schwarz based nonlinear elimination for problems with high local nonlinearity. *SIAM J. Sci. Comput.*, 33:746–762, 2011.

[7] H. Deconinck and C. Hirsch. A multigrid method for the transonic full potential equation discretized with finite elements on an arbitrary body fitted mesh. *J. Comput. Phys.*, 48:344–365, 1982.

[8] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, 1996.

[9] C. Groß and R. Krause. On the globalization of ASPIN employing trust-region control strategies–convergence analysis and numerical examples. *preprint*, 2011.

[10] C. Hirsch. *Numerical Computation of Internal and External Flows, Vol. 2*. Wiley, 1990.

[11] F.-N. Hwang and X.-C. Cai. A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier-Stokes equations. *J. Comput. Phys.*, 204:666–691, 2005.

[12] F.-N. Hwang and X.-C. Cai. A class of parallel two-level nonlinear Schwarz preconditioned inexact Newton algorithms. *Comput. Meth. Appl. Mech. Eng.*, 196:1603–1611, 2007.

[13] F-.N. Hwang, H.-L. Lin, and X.-C. Cai. Two-level nonlinear elimination-based preconditioners for inexact Newton methods with application in shoched duct flow calculation. *Electron. Trans. Numer. Anal.*, 37:239–251, 2010.

[14] P.J. Lanzkron, D.J. Rose, and J.T. Wilkes. An analysis of approximate nonlinear elimination. *SIAM, J. Sci. Comput.*, 17:538–559, 1996.

[15] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Verlag, New York, 1999.

[16] A. Rizzi and H. Viviand, editors. *Numerical Methods for the Computation of Inviscid Transonic Flows with Shock Waves: A GAMM Workshop*. Vieweg, Brunswick, 1981.

[17] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[18] S. Shitrit, D. Sidilkover, and A. Gelfgat. An algebraic multigrid solver for transonic flow problems. *J. Comput. Phys.*, 230:1707–1729, 2011.

[19] J.O. Skogestad, E. Keilegavlen, and J.M. Nordbotten. Domain decomposition strategies for nonlinear flow problems in porous media. *J. Comput. Phys.*, 234:439–451, 2013.

[20] D.P. Young, W.P. Huffman, R.G. Melvin, C.L. Hilmes, and F. T. Johnson. Nonlinear elimination in aerodynamic analysis and design optimization. In L.T. Biegler, O. Ghattas, Heinkenschloss M., and B. van Bloemen Waanders, editors, *Large-Scale PDE-Constrained Optimization*, Lect. Notes in Comp. Sci., pages 17–44. Springer-Verlag, 2003.

[21] D.P. Young, R.G. Melvin, M.B. Bieterman, F.T. Johnson, S.S. Samant, and J.E. Bussoletti. A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics. *J. Comput. Phys.*, 92:1–66, 1991.

[22] M. Ziani. *Accélération de la convergence des méthodes de type Newton pour la résolution des systèmes non-linéaires*. PhD thesis, Université de Rennes 1, 2009.