An Efficient Parallel Algorithm with Application to Computational Fluid Dynamics

Wilson Rivera, Jianping Zhu*, and David Huddleston
Mississippi State University, USA

Abstract

When solving time dependent partial differential equations on parallel computers using non-overlapping domain decomposition methods, one often needs numerical boundary conditions on the boundaries between subdomains. These numerical boundary conditions can significantly affect the stability and accuracy of the final algorithm.

In this paper, a stability and accuracy analysis of the existing methods for generating numerical boundary conditions will be presented, and a new approach based on explicit predictors and implicit correctors will be used to solve convection-diffusion equations on parallel computers, with application to aerospace engineering for the solution of Euler equations in computational fluid dynamics simulations. Both theoretical analyses and numerical results demonstrate significant improvement in stability and accuracy by using the new approach.

*Corresponding author: Jianping Zhu, Department of Mathematics and Statistics, Mississippi State University, Mississippi State, MS 39762, U.S.A. email: jzhumath.msstate.edu

1 Introduction

Convection-diffusion equations in the form of

$$u_t + \alpha \nabla u = \nabla \cdot (\beta \nabla u), \quad x \in \Omega, \quad t > 0, \tag{1}$$

$$u(x,t) = f(x,t), \qquad x \in \partial\Omega, \qquad t > 0$$

$$u(x,0) = g(x), \qquad x \in \Omega,$$

where Ω is the spatial domain and $\partial\Omega$ is the boundary of Ω, are widely used in science and engineering as mathematical models for computational simulations, such as in oil reservoir simulations, analysis of flow field around airplanes, transport of solutes in groundwater, and global weather prediction. In particular, when $\beta = 0$, Equation (1) becomes a pure convection equation.

For large-scale problems, particularly those defined in two- or three-dimensional spatial domains, the computation of solutions may require substantial CPU time. It is therefore desirable to use multiprocessor parallel computers to calculate solutions. One way to parallelize an implicit algorithm for solving time dependent PDEs is to use a parallel linear algebraic equation solver. There are various parallel algorithms for solving linear algebraic equation systems using multiprocessor computers, notably the nested dissection method [7], the cyclic reduction method [7], and the parallel diagonal dominant method [25]. These parallel algorithms either have a higher computational complexity than the sequential algorithm, or are applicable only to a special class of matrices, such as diagonally dominant Toeplitz matrices [9, 21]. For problems involving Neumann boundary conditions or convection terms, the coefficient matrix resulting from discretization of Equation (1) may not be a diagonally dominant Toeplitz matrix.

Another widely used method for solving time dependent PDEs on parallel computers is domain decomposition [5]. It dates back to the classical Schwarz alternating algorithm with overlapping subdomains [16, 24] for solving elliptic boundary value problems. Note that the original motivation for using domain decomposition method was to deal with complex geometries, equations that exhibit different behaviors in different regions of the domain, and memory restriction for solving large scale problems.

When solving time dependent PDEs with non-overlapping subdomains on parallel computers, the domain decomposition method could either be used as a preconditioner for Krylov type algorithms [2, 5], or as a means to decompose the original domain into subdomains and solve the PDEs defined in different subdomains concurrently [4, 8, 12, 17, 19]. When it is used as a preconditioner, the relevant PDE is discretized over the entire original domain to form a large system of algebraic equations, which is then solved by Krylov type iterative algorithms. The preconditioning step and the inner products involved in the solution process often incur a significant amount of communication overhead that could significantly affect the scalability of the solution algorithms.

On the other hand, if the original domain Ω is decomposed into a set of non-overlapping subdomains $\Omega_k, k = 1, \cdots, M$, it would be ideal that the PDEs defined in different subdomains could be solved on different processors concurrently. This often requires numerical boundary conditions at the boundaries between subdomains. These numerical

boundary conditions are not part of the original mathematical model and the physical problem. One way to generate those numerical boundary conditions is to use the solution values from the previous time step t_n to calculate the solutions at t_{n+1} [1, 17, 20]. This is often referred to as time lagging (TL). The other way to generate numerical boundary conditions is to use an explicit algorithm to calculate the solutions at the boundaries between subdomains, using the solutions from the previous time step, and then solve the PDEs defined on different subdomains concurrently using an implicit method [6, 14]. This is referred to as the explicit predictor (EP) method in this paper. In an earlier paper [22], Zhu et al. showed, in the context of the numerical solution of one-dimensional linear heat equation, that the stability and accuracy of the solution algorithm can be significantly affected by the TL and EP methods. A new method based on explicit predictor and implicit corrector (EPIC) for generating numerical boundary conditions was discussed in [22]. Preliminary numerical experiments with a one-dimensional linear heat equation have demonstrated significant improvement in stability and accuracy using this new method.

In this paper, a more systematic stability analysis for the TL, EP, and EPIC methods will be presented for solving more general equations, i. e. the convection-diffusion equations. Practical application of the methods to nonlinear system of Euler equations for the flow field calculation of an airfoil on parallel computers will also be discussed.

The next section is devoted to the analysis of the TL and EP methods. The EPIC method will be discussed in Section 3. Parallel implementation of the algorithm and numerical experiment will be presented in Section 4. Application to the solution of nonlinear Euler equations will be given in Section 5, followed by the conclusions in Section 6.

2 Analysis of the TL and EP Methods

For simplicity of the discussion, the following one-dimensional linear model with constant coefficients and homogeneous boundary conditions is used here to analyze different methods for generating numerical boundary conditions:

$$\begin{aligned} &u_t + \alpha u_x = \beta u_{xx}, \quad 0 < x < 1, \quad 0 < t \le T, \\ &u(0,t) = u(1,t) = 0, \quad t > 0, \\ &u(x,0) = g(x), \quad 0 \le x \le 1. \end{aligned} \tag{2}$$

The results are applicable to higher dimensional models with nonhomogeneous boundary conditions.

As shown in Fig. 1, the original spatial domain $\Omega = [0,1]$ is discretized by a set of grid points $x_i, i = 0, \ldots, L$, uniformly distributed with $\Delta x = x_i - x_{i-1} = \frac{1}{L}$. The temporal domain $[0,T]$ is discretized by a set of discrete time steps $t_n, n = 0, \cdots, N$, with $\Delta t = t_n - t_{n-1} = \frac{T}{N}$. The numerical solution $u(x_i, t_n)$ is denoted by u_i^n, and the original spatial domain is decomposed into M subdomains $\Omega_k, k = 1, \ldots, M$, where the two end points of subdomain Ω_k are denoted as r_{k-1} and r_k, respectively. Each

subdomain Ω_k has $m+1$ points including the two end points r_{k-1} and r_k. Since only two physical boundary conditions are available at the points r_0 and r_M, numerical boundary conditions are needed at points r_k, $k = 1, \cdots, M-1$, if the PDEs defined in different subdomains are to be solved concurrently using an implicit algorithm.

Various finite difference algorithms are available for discretizing Eq. (2). The forward time central difference (FTCS) scheme is given by

$$\begin{aligned} &\frac{u_i^{n+1} - u_i^n}{\Delta t} + \alpha \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} = \beta \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}, \\ &i = 1, ..., L-1, \quad n = 0, ..., N-1, \end{aligned} \tag{3}$$

or equivalently

$$\begin{aligned} &u_i^{n+1} = (r + \frac{R}{2})u_{i-1}^n + (1-2r)u_i^n + (r - \frac{R}{2})u_{i+1}^n, \\ &i = 1, ..., L-1, \quad n = 0, ..., N-1, \end{aligned} \tag{4}$$

where $R = \alpha \frac{\Delta t}{\Delta x}$ and $r = \beta \frac{\Delta t}{\Delta x^2}$.

The stability condition for this FTCS scheme is given by [26]

$$\frac{R^2}{2} \leq r \leq \frac{1}{2}. \tag{5}$$

When the physical process is convection dominant, the stability condition in (5) may be difficult to satisfy and oscillations could occur in the numerical solutions obtained with the FTCS scheme. In this case, it is usually better to use upwind difference for the convective term to improve stability and reduce oscillations. If $\alpha < 0$, the upwind scheme is

$$\begin{aligned} &\frac{u_i^{n+1} - u_i^n}{\Delta t} + \alpha \frac{u_{i+1}^n - u_i^n}{\Delta x} = \beta \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}, \\ &i = 1, ..., L-1, \quad n = 0, ..., N-1, \end{aligned} \tag{6}$$

or equivalently

$$\begin{aligned} &u_i^{n+1} = ru_{i-1}^n + (1 + R - 2r)u_i^n + (r - R)u_{i+1}^n, \\ &i = 1, ..., L-1, \quad n = 0, ..., N-1. \end{aligned} \tag{7}$$

Otherwise, if $\alpha > 0$ then the upwind scheme is

$$\begin{aligned} &\frac{u_i^{n+1} - u_i^n}{\Delta t} + \alpha \frac{u_i^n - u_{i-1}^n}{\Delta x} = \beta \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}, \\ &i = 1, ..., L-1, \quad n = 0, ..., N-1, \end{aligned} \tag{8}$$

or equivalently

$$\begin{aligned} &u_i^{n+1} = (r + R)u_{i-1}^n + (1 - R - 2r)u_i^n + ru_{i+1}^n, \\ &i = 1, ..., L-1, \quad n = 0, ..., N-1. \end{aligned} \tag{9}$$

For implicit schemes, the BTCS scheme is given by

$$\frac{u_i^{n+1}-u_i^n}{\Delta t}+\alpha\frac{u_{i+1}^{n+1}-u_{i-1}^{n+1}}{2\Delta x}=\beta\frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{\Delta x^2}, \qquad (10)$$
$$i=1,...,L-1, \quad n=0,...,N-1,$$

or equivalently

$$-(r+\frac{R}{2})u_{i-1}^{n+1}+(1+2r)u_i^{n+1}-(r-\frac{R}{2})u_{i+1}^{n+1}=u_i^n, \qquad (11)$$
$$i=1,...,L-1, \quad n=0,...,N-1.$$

The implicit version of the upwind schemes is

$$-ru_{i-1}^{n+1}+(1-R+2r)u_i^{n+1}-(r-R)u_{i+1}^{n+1}=u_i^n, \qquad (12)$$
$$i=1,...,L-1, \quad n=0,...,N-1$$

for $\alpha<0$, and

$$-(r+R)u_{i-1}^{n+1}+(1+R+2r)u_i^{n+1}-ru_{i+1}^{n+1}=u_i^n, \qquad (13)$$
$$i=1,...,L-1, \quad n=0,...,N-1,$$

for $\alpha>0$.

2.1 Time-Lagging (TL) Method

For the TL method, the boundary conditions between subdomains are generated by setting

$$\begin{aligned}\bar{u}_{r_k-1}^{n+1}&=u_{r_k-1}^n,\\ \bar{u}_{r_k}^{n+1}&=u_{r_k}^n, \qquad k=1,\ldots,M-1.\end{aligned} \qquad (14)$$

Note that the left side of subdomain Ω_k, $k=2,\ldots,M$, is extended to $r_{k-1}-1$ in order to advance the solution value at the point r_{k-1} to the next time level. An implicit scheme is then used to solve the PDE in each subdomain concurrently. This process can be illustrated using a simple example shown in Fig. 2, in which $\Omega=\{x_i, i=0,\cdots,8\}$. There are two subdomains $\Omega_1=\{x_i, i=0,\cdots,4\}$ and $\Omega_2=\{x_i, i=3,\cdots,8\}$.

The matrix representation of the solution algorithm given in (10) and (14) can be written as

$$\begin{bmatrix} a_0 & a_1 & & & & & \\ a_2 & a_0 & a_1 & & & & \\ & a_2 & a_0 & & & & \\ & & & a_0 & a_1 & & \\ & & & a_2 & a_0 & a_1 & \\ & & & & a_2 & a_0 & a_1 \\ & & & & & a_2 & a_0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix}^{n+1} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & -a_1 & & & \\ & & -a_2 & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix}^{n} \qquad (15)$$

where the coefficients a_0, a_1 and a_2 depend on the discretization method used. For example, with the BTCS algorithm, we have $a_0 = 1 + 2r$, $a_1 = -(r - \frac{R}{2})$, and $a_2 = -(r + \frac{R}{2})$.

For a general domain with $\Omega = \{x_i, i = 0, \ldots, L\}$ and M subdomains with equal number of grid points, the matrix representation of the algorithm is given by

$$\begin{bmatrix} A' & & & & \\ & A & & & \\ & & \ddots & & \\ & & & A & \\ & & & & A \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{M-1} \\ \mathbf{u}_M \end{bmatrix}^{n+1} = \begin{bmatrix} I' & B_1' & & & \\ B_2' & I & B_1 & & \\ & \ddots & \ddots & \ddots & \\ & & B_2 & I & B_1 \\ & & & B_2 & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{M-1} \\ \mathbf{u}_M \end{bmatrix}^{n} . \tag{16}$$

The vectors $\mathbf{u}_k, k = 1, \ldots, M$, represent the solution vector in the interior of the k-th subdomain Ω_k plus the solution at the left end point $u_{r_{k-1}}$, except for $\mathbf{u}_1$ that includes only the solutions at the interior points of Ω_1. The block matrices A' and A are of the form

$$\begin{bmatrix} a_0 & a_1 & 0 & \cdots & 0 \\ a_2 & a_0 & a_1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & a_1 \\ 0 & \cdots & 0 & a_2 & a_0 \end{bmatrix}, \tag{17}$$

The order of A' is $m - 2$, and that of A is $m - 1$. Each subdomain Ω_k has $m + 1$ points including the boundary points r_{k-1} and r_k. Both I' and I are identity matrices with the same dimensions as that of A' and A, respectively. The elements in matrices B_1, B_1', B_2 and B_2' are all zero, except for the element $-a_1$ at the lower left corner of B_1 and B_1', and the element $-a_2$ at the upper right corner of B_2 and B_2'.

Since the standard Von Neumann stability analysis is based on Fourier transform, it can only be used to analyze problems with periodic boundary conditions. To consider the effect of different boundary conditions on the stability of the solution algorithms, matrix analysis is used here to analyze various numerical boundary conditions.

The compact form of the matrix representation of the TL method in (16) is

$$C\mathbf{u}^{n+1} = F\mathbf{u}^n. \tag{18}$$

It is well-known that the necessary condition for stability is $\rho(C^{-1}F) = max|\lambda_j| \leq 1, j = 1, \ldots, L-1$, where λ_j's are the eigenvalues of matrix $C^{-1}F$ [13, 26]. Since it is difficult to obtain an analytic expression of the eigenvalues for this matrix, the software package MATLAB has been used to calculate the magnitude of the largest eigenvalues.

Fig. 3 shows the contour plot of $\rho(C^{-1}F)$ vs. (R, r) with $L = 200$, $M = 10$, $-10^{-4} \leq R \leq 10^4$, and $0 \leq r \leq 10^4$ for the TL method with the BTCS scheme in (11). We refer to this method as TL method with central difference. Note that the maximum value of the contour lines is 1.0, showing that the necessary condition for stability is satisfied. Actual numerical experiments also demonstrate that the computation is stable.

Fig. 4 shows the contour plot of $\rho(C^{-1}F)$ vs. (R, r) with $L = 200$, $M = 10$, $-10^{-4} \leq R \leq 0$, and $0 \leq r \leq 10^4$, for the TL method using the upwind scheme in (12) with $\alpha < 0$. In this case $a_0 = 1 - R + 2r$, $a_1 = -(r - R)$, and $a_2 = -r$. The maximum spectral radius in the figure is 0.9976

The effect of the number of subdomains M on the spectral radius is demonstrated in Fig. 5, with $0 \leq M \leq 100$, $L = 500$, $R = 1000$, and $r = 1000$ for the TL method with central and upwind schemes. Even though the spectral radius increases as the number of subdomains increases, it is still less than 1 with 100 subdomains. In practical computations, it is unlikely that a domain with only 500 grid points will be decomposed into 100 subdomains. Each subdomain will have a lot more grid points, which helps keep the spectral radius less than 1.

In all cases, it is shown that the necessary condition for stability is satisfied. Although this is not a rigorous mathematical proof of $\rho(C^{-1}F) \leq 1$ for all cases, it does demonstrate $\rho(C^{-1}F) \leq 1$ for practical purpose, since the cases cover a wide range of values of r, R, and M.

To estimate the additional temporal error caused by the the numerical boundary condition from the TL method, we substitute the Taylor series expansion into the finite difference formula (10) with time lagging at the point $r_k - 1$:

$$
\begin{aligned}
&\left\{\frac{u^{n+1}_{r_k-1} - u^{n}_{r_k-1}}{\Delta t}\right\} + \alpha\left\{\frac{u^{n}_{r_k} - u^{n+1}_{r_k-2}}{2\Delta x}\right\} - \beta\left\{\frac{u^{n}_{r_k} - 2u^{n+1}_{r_k-1} + u^{n+1}_{r_k-2}}{\Delta x^2}\right\} = \\
&\left\{\frac{u^{n+1}_{r_k-1} - u^{n}_{r_k-1}}{\Delta t}\right\} + \alpha\left\{\frac{u^{n}_{r_k} - u^{n+1}_{r_k} + u^{n+1}_{r_k} - u^{n+1}_{r_k-2}}{2\Delta x}\right\} \\
&-\beta\left\{\frac{u^{n}_{r_k} - u^{n+1}_{r_k} + u^{n+1}_{r_k} - 2u^{n+1}_{r_k-1} + u^{n+1}_{r_k-2}}{\Delta x^2}\right\} = \\
&\left\{\frac{u^{n+1}_{r_k-1} - u^{n}_{r_k-1}}{\Delta t}\right\} + \alpha\left\{\frac{u^{n}_{r_k} - u^{n+1}_{r_k}}{2\Delta x}\right\} + \alpha\left\{\frac{u^{n+1}_{r_k} - u^{n+1}_{r_k-2}}{2\Delta x}\right\} \\
&-\beta\left\{\frac{u^{n}_{r_k} - u^{n+1}_{r_k}}{\Delta x^2}\right\} - \beta\left\{\frac{u^{n+1}_{r_k} - 2u^{n+1}_{r_k-1} + u^{n+1}_{r_k-2}}{\Delta x^2}\right\} = \\
&\left\{u^{n+1}_{t} - \frac{(\Delta t)}{2}u^{n+1}_{tt} + \cdots\right\}_{r_k-1} \\
&+\frac{\alpha}{2(\Delta x)}\left\{-(\Delta t)u^{n+1}_{t} + \frac{(\Delta t)^2}{2}u^{n+1}_{tt} + \cdots\right\}_{r_k} \\
&+\frac{\alpha}{2(\Delta x)}\left\{2(\Delta x)u^{n+1}_{x} + \frac{(\Delta x)^3}{3}u^{n+1}_{xxx} + \cdots\right\}_{r_k-1} \\
&-\frac{\beta}{(\Delta x)^2}\left\{-(\Delta t)u^{n+1}_{t} + \frac{(\Delta t)^2}{2}u^{n+1}_{tt} + \cdots\right\}_{r_k} \\
&-\frac{\beta}{(\Delta x)^2}\left\{(\Delta x)^2 u^{n+1}_{xx} + \frac{(\Delta x)^4}{12}u^{n+1}_{xxxx} + \cdots\right\}_{r_k-1}.
\end{aligned}
$$

Since we have

$$\left\{u_t^{n+1} + \alpha u_x^{n+1} - \beta u_{xx}^{n+1}\right\}_{r_k-1} = 0, \tag{19}$$

the truncation error is

$$\begin{aligned}
&\left\{ -\frac{(\Delta t)}{2} u_{tt}^{n+1} + \cdots \right\}_{r_k-1} \\
&+ \left\{ -\frac{\alpha(\Delta t)}{2(\Delta x)} u_t^{n+1} + \frac{\alpha(\Delta t)^2}{4(\Delta x)} u_{tt}^{n+1} + \cdots \right\}_{r_k} \\
&+ \left\{ \frac{\alpha(\Delta x)^2}{6} u_{xxx}^{n+1} + \cdots \right\}_{r_k-1} \\
&- \left\{ -\frac{\beta(\Delta t)}{(\Delta x)^2} u_t^{n+1} + \frac{\beta(\Delta t)^2}{(\Delta t)^2} u_{tt}^{n+1} + \cdots \right\}_{r_k} \\
&- \left\{ \frac{\beta(\Delta x)^2}{12} u_{xxxx}^{n+1} + \cdots \right\}_{r_k-1},
\end{aligned} \tag{20}$$

in which the additional truncation error caused by time lagging is represented by the terms in the second and the fourth pairs of brackets. If $\beta \neq 0$, the additional error is dominated by the term $\frac{\Delta t}{\Delta x^2}$. This indicates that the TL method causes an additional truncation error of order $\mathcal{O}[\frac{\Delta t}{\Delta x^2}]$ at the boundaries between subdomains. Notice that if $\beta = 0$, that is the PDE is a pure convective equation, then the additional truncation error is dominated by the term $\frac{\Delta t}{\Delta x}$ at the boundary.

By using the same procedure, it can be shown that the additional truncation errors of the TL method with upwind difference is also dominated by the term $\frac{\Delta t}{\Delta x^2}$ for the convection-diffusion equation, and by the term $\frac{\Delta t}{\Delta x}$ for the pure convection equation.

Gustafsson [10] has proved that if the order of accuracy of the difference equation approximating a given hyerbolic partial differential equation is $p \geq 1$, the order of accuracy of the boundary conditions is $p-1$, and the scheme is stable, then the computation will converge with order m. This result can be interpreted as that the truncation error for the solution inside the spatial domain is one order higher than that at the boundaries. Thus, since the additional error caused by the TL method applied to the pure convection equation is $\mathcal{O}[\frac{\Delta t}{\Delta x}]$ at the boundaries, the additional error inside the spatial domain is of the order of $\mathcal{O}[\frac{\Delta t}{\Delta x}\Delta x] = \mathcal{O}[\Delta t]$.

Although Gustafsson's proof applies only to pure convection equations, it will be demonstrated later in our numerical experiment that this result also holds for parabolic equations. Since the additional error caused by the TL method for the convection-diffusion equation at the boundaries is of order $\mathcal{O}[\frac{\Delta t}{\Delta x^2}]$, the additional error inside the domain is of order $\mathcal{O}[\frac{\Delta t}{\Delta x}]$. This explains why the maximum error for the TL method in Table 3 remains roughly a constant when the grid is refined with Δx proportional to Δt.

2.2 Explicit-Predictor (EP) Method

The matrix representation of the EP method for a one-dimensional domain with uniformly distributed grid points x_i, $i = 0, \ldots, L$, and M subdomains with equal number

of grid points can be written as

$$\begin{bmatrix} A & v & & & & & \\ & 1 & & & & & \\ & w & A & v & & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & w & A & v \\ & & & & & & 1 \\ & & & & & & w & A \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ u_{r_1} \\ \mathbf{u}_2 \\ u_{r_2} \\ \vdots \\ \mathbf{u}_{M-1} \\ u_{r_{M-1}} \\ \mathbf{u}_M \end{bmatrix}^{n+1} = \begin{bmatrix} I & & & & & & & \\ y & \sigma & z & & & & & \\ & & I & & & & & \\ & & y & \sigma & z & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & & I & & \\ & & & & & y & \sigma & z \\ & & & & & & & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ u_{r_1} \\ \mathbf{u}_2 \\ u_{r_2} \\ \vdots \\ \mathbf{u}_{M-1} \\ u_{r_{M-1}} \\ \mathbf{u}_M \end{bmatrix}^{n} \tag{21}$$

where A is a matrix of order $m-2$ with similar structure as that in (17), I is an identity matrix of order $m-2$, $\mathbf{u}_k$, $k = 1, \ldots, M$, represent the solution vector in the interior of the k-th subdomain Ω_k without including the two end points $u_{r_{k-1}}$ and u_{r_k}, the parameters a_0, a_1, a_2 and σ depend on the discretization methods used, and the vectors v, w, y, z are defined as

$$v = \{0, \cdots, 0, a_1\}^T, \qquad w = \{a_2, 0, \cdots, 0\}^T, \qquad y = -v^T, \qquad z = -w^T.$$

The compact form of (21) is

$$C\mathbf{u}^{n+1} = F\mathbf{u}^n. \tag{22}$$

For the EP method with central difference for both the convection and diffusion term, the algorithm can be written as

- Predictor

$$\bar{u}_{r_k}^{n+1} = (r + \frac{R}{2})u_{r_k-1}^n + (1-2r)u_{r_k}^n + (r - \frac{R}{2})u_{r_k+1}^n, \quad k = 1, \cdots, M-1, \tag{23}$$

- Calculate solution in each subdomain

$$\begin{bmatrix} a_0 & a_1 & 0 & \cdots & 0 \\ a_2 & a_0 & a_1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & a_1 \\ 0 & \cdots & 0 & a_2 & a_0 \end{bmatrix} \begin{bmatrix} u_{r_{k-1}+1}^{n+1} \\ u_{r_{k-1}+2}^{n+1} \\ \vdots \\ u_{r_k-2}^{n+1} \\ u_{r_k-1}^{n+1} \end{bmatrix} = \begin{bmatrix} u_{r_{k-1}+1}^n - a_2\bar{u}_{r_{k-1}}^{n+1} \\ u_{r_{k-1}+2}^n \\ \vdots \\ u_{r_k-2}^n \\ u_{r_k-1}^n - a_1\bar{u}_{r_k}^{n+1} \end{bmatrix}, \tag{24}$$

$$k = 1, \cdots, M,$$

which can be assembled in the form of system (21) with $a_0 = 1 + 2r$, $a_1 = -(r - \frac{R}{2})$, $a_2 = -(r + \frac{R}{2})$, and $\sigma = 1 - 2r$.

Fig. 6 shows the contour plot of $\rho(C^{-1}F)$ vs. (R, r) with $L = 200$, and $M = 10$ for the EP method with central difference. In this case $-2 \le R \le 2$ and $0 \le r \le 2$. It is clear that the method is only conditionally stable. For example, when $R \ge 1$, the scheme is unstable.

Similarly, Fig. 7 shows that the EP method with upwind difference is also only conditionally stable. In this case $\alpha < 0$, $L = 200$, $M = 10$, $-2 \le R \le 0$, and $0 \le r \le 2$. The method is a combination of scheme (7) as a predictor and scheme (12) as the algorithm for the interior points in all subdomains, which corresponds to $a_0 = 1 - R + 2r$, $a_1 = -(r - R)$, $a_2 = -r$, and $\sigma = 1 + R - 2r$.

The additional truncation error caused by the numerical boundary condition from the EP method can be analyzed in a similar way to what we did for the TL method. With central difference for both the convection and diffusion terms, we have

$$\begin{aligned}
&\left\{\frac{u_{r_k-1}^{n+1} - u_{r_k-1}^{n}}{\Delta t}\right\} + \alpha\left\{\frac{\bar{u}_{r_k}^{n+1} - u_{r_k-2}^{n+1}}{2\Delta x}\right\} - \beta\left\{\frac{\bar{u}_{r_k}^{n+1} - 2u_{r_k-1}^{n+1} + u_{r_k-2}^{n+1}}{\Delta x^2}\right\} = \\
&\left\{\frac{u_{r_k-1}^{n+1} - u_{r_k-1}^{n}}{\Delta t}\right\} + \alpha\left\{\frac{\bar{u}_{r_k}^{n+1} - u_{r_k}^{n+1} + u_{r_k}^{n+1} - u_{r_k-2}^{n+1}}{2\Delta x}\right\} \\
&- \beta\left\{\frac{\bar{u}_{r_k}^{n+1} - u_{r_k}^{n+1} + u_{r_k}^{n+1} - 2u_{r_k-1}^{n+1} + u_{r_k-2}^{n+1}}{\Delta x^2}\right\} = \\
&\left\{\frac{u_{r_k-1}^{n+1} - u_{r_k-1}^{n}}{\Delta t}\right\} + \alpha\left\{\frac{\bar{u}_{r_k}^{n+1} - u_{r_k}^{n+1}}{2\Delta x}\right\} + \alpha\left\{\frac{u_{r_k}^{n+1} - u_{r_k-2}^{n+1}}{2\Delta x}\right\} \\
&- \beta\left\{\frac{\bar{u}_{r_k}^{n+1} - u_{r_k}^{n+1}}{\Delta x^2}\right\} - \beta\left\{\frac{u_{r_k}^{n+1} - 2u_{r_k-1}^{n+1} + u_{r_k-2}^{n+1}}{\Delta x^2}\right\}.
\end{aligned} \tag{25}$$

Note that we have

$$\begin{aligned}
\bar{u}_{r_k}^{n+1} &= u_{r_k} - \frac{\alpha(\Delta t)}{2(\Delta x)}\left\{u_{r_k+1}^{n} - u_{r_k-1}^{n}\right\} + \frac{\beta(\Delta t)}{(\Delta x)^2}\left\{u_{r_k+1}^{n} - 2u_{r_k}^{n} + u_{r_k-1}^{n}\right\} \\
&= u_{r_k}^{n+1} + \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta t \Delta x^2).
\end{aligned} \tag{26}$$

Substituting (26) into (25) and expanding by Taylor series as we did for the TL method, we found that the additional error caused by the EP method with central difference is dominated by the term of order $\mathcal{O}[\frac{\Delta t^2}{\Delta x^2}]$ at the boundaries. If $\beta = 0$ then the additional error at the boundaries is reduced to $\mathcal{O}[\frac{\Delta t^2}{\Delta x}]$. The same conclusions also apply to the EP method with upwind difference.

Similar to the discussion for the TL method, we can show, based on Gustafsson's proof and numerical experiment, that the additional error in the final solution caused by the EP method is of order $\mathcal{O}[\Delta t^2]$ for the pure convection equation with $\beta = 0$. For the convection-diffusion equations, numerical experiments have shown that the additional error in the final solution is of order $\mathcal{O}[\frac{(\Delta t)^2}{(\Delta x)}]$. In particular, if the grid size Δt and Δx is refined proportionally, i. e. $\Delta t = c\Delta x$, then the error is of order $O(\Delta t)$, which explains why the results from the EP method in our numerical experiment is more accurate than the TL method when it is in the stable region.

3 Explicit-Predictor Implicit-Corrector (EPIC)

Based on the analysis of the TL and EP methods, it is clear that a method that combines the advantages of both the TL (stability) and EP (accuracy) methods would be

very desirable for solving convection-diffusion equations on parallel computers. An approach based on explicit predictor and implicit corrector (EPIC) was first proposed in [22] for solving linear heat equations, with some preliminary numerical results. Although the concept of explicit predictor and implicit corrector has long been used to develop numerical algorithms for solving nonlinear ODEs, it has never been used to generate numerical boundary conditions for solving PDEs using domain decomposition algorithms. We will extend this method and the analysis here to the solution of convection-diffusion equations on parallel computers. The following are the main steps of the EPIC method with central difference for both the convection and diffusion terms:

- Use an explicit predictor, such as the FTCS algorithm, to generate the numerical boundary conditions at the end points r_k, $k = 1, \cdots, M-1$, of all subdomains:

$$\bar{u}_{r_k}^{n+1} = (r+R)u_{r_k-1}^n + (1-2r)u_{r_k}^n + (r-R)u_{r_k+1}^n, \quad k = 1, ..., M-1, \tag{27}$$

- Solve the systems of equations in all subdomains Ω_k, $k = 1, \cdots, M-1$, concurrently:

$$\begin{bmatrix} a_0 & a_1 & 0 & \cdots & 0 \\ a_2 & a_0 & a_1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & a_1 \\ 0 & \cdots & 0 & a_2 & a_0 \end{bmatrix} \begin{bmatrix} u_{r_{k-1}+1}^{n+1} \\ u_{r_{k-1}+2}^{n+1} \\ \vdots \\ u_{r_k-2}^{n+1} \\ u_{r_k-1}^{n+1} \end{bmatrix} = \begin{bmatrix} u_{r_{k-1}+1}^n - a_2 \bar{u}_{r_{k-1}}^{n+1} \\ u_{r_{k-1}+2}^n \\ \vdots \\ u_{r_k-2}^n \\ u_{r_k-1}^n - a_1 \bar{u}_{r_k}^{n+1} \end{bmatrix}, \tag{28}$$

- Update the numerical boundary conditions at points r_k, $k = 1, \cdots, M-1$, using an implicit corrector, such as the BTCS algorithm:

$$u_{r_k}^{n+1} = \frac{(r+R)}{(1+2r)} u_{r_k-1}^n + \frac{1}{(1+2r)} u_{r_k}^n + \frac{(r-R)}{(1+2r)} u_{r_k+1}^{n+1}. \tag{29}$$

The matrix representation of the first two steps for a general domain $\Omega = \{x_i, i = 0, \ldots, L\}$ having M subdomains with equal number of $m+1$ grid points will be similar to those as given in (21), that is

$$\begin{bmatrix} A & v & & & & & & \\ & 1 & & & & & & \\ & w & A & v & & & & \\ & & & 1 & & & & \\ & & & & \ddots & & & \\ & & & & w & A & v & \\ & & & & & & 1 & \\ & & & & & & w & A \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}}_1 \\ \tilde{u}_{r_1} \\ \tilde{\mathbf{u}}_2 \\ \tilde{u}_{r_2} \\ \vdots \\ \tilde{\mathbf{u}}_{M-1} \\ \tilde{u}_{r_{M-1}} \\ \tilde{\mathbf{u}}_M \end{bmatrix}^{n+1} = \begin{bmatrix} I & & & & & & & \\ y & \sigma & z & & & & & \\ & & I & & & & & \\ & & & y & \sigma & z & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & & I & \\ & & & & & y & \sigma & z \\ & & & & & & & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ u_{r_1} \\ \mathbf{u}_2 \\ u_{r_2} \\ \vdots \\ \mathbf{u}_{M-1} \\ u_{r_{M-1}} \\ \mathbf{u}_M \end{bmatrix}^n,$$

or

$$C\tilde{\mathbf{u}}^{n+1} = F\mathbf{u}^n. \tag{30}$$

The vector $\tilde{\mathbf{u}}^{n+1}$ represents the intermediate solution obtained without the corrector step. The matrix representation of the corrector step is given by

$$\begin{bmatrix} \mathbf{u}_1 \\ u_{r_1} \\ \mathbf{u}_2 \\ u_{r_2} \\ \vdots \\ \mathbf{u}_{M-1} \\ u_{r_{M-1}} \\ \mathbf{u}_M \end{bmatrix}^{n+1} = \begin{bmatrix} O & & & & & & & \\ & \bar{\sigma} & & & & & & \\ & & O & & & & & \\ & & & \bar{\sigma} & & & & \\ & & & & \ddots & & & \\ & & & & & \bar{\sigma} & & \\ & & & & & & O & \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ u_{r_1} \\ \mathbf{u}_2 \\ u_{r_2} \\ \vdots \\ \mathbf{u}_{M-1} \\ u_{r_{M-1}} \\ \mathbf{u}_M \end{bmatrix}^{n}$$

$$+ \begin{bmatrix} I & & & & & & & \\ \bar{v} & 0 & \bar{w} & & & & & \\ & & I & & & & & \\ & & \bar{v} & 0 & \bar{w} & & & \\ & & & & \ddots & & & \\ & & & & \bar{v} & 0 & \bar{w} & \\ & & & & & & I & \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}}_1 \\ \tilde{u}_{r_1} \\ \tilde{\mathbf{u}}_2 \\ \tilde{u}_{r_2} \\ \vdots \\ \tilde{\mathbf{u}}_{M-1} \\ \tilde{u}_{r_{M-1}} \\ \tilde{\mathbf{u}}_M \end{bmatrix}^{n+1}, \tag{31}$$

or

$$\mathbf{u}^{n+1} = E\mathbf{u}^n + G\tilde{\mathbf{u}}^{n+1}, \tag{32}$$

where O is the zero matrix, $\bar{v}$, $\bar{w}$ and $\bar{\sigma}$ depend on the discretization method used for the corrector. For the corrector given in (29), we have $\bar{\sigma} = \frac{1}{1+2r}$ and the vectors $\bar{v}$ and $\bar{w}$ of dimension $m-2$ are defined as

$$\bar{v} = \{0, \cdots, 0, \frac{r+R}{1+2r}\}, \qquad \bar{w} = \{\frac{r-R}{1+2r}, 0, \cdots, 0\}, \quad k = 1, ..., M-1.$$

The final equation that connects $\mathbf{u}^n$ with $\mathbf{u}^{n+1}$ is then

$$\mathbf{u}^{n+1} = E\mathbf{u}^n + G\tilde{\mathbf{u}}^{n+1} = (E + GC^{-1}F)\mathbf{u}^n, \tag{33}$$

and the necessary condition for stability is $\rho(E + GC^{-1}F) \leq 1$.

Fig. 8 shows the contour plot of $\rho(E + GC^{-1}F)$ vs. (R, r) with $-10^{-4} \leq R \leq 10^4$, $0 \leq r \leq 10^4$, $L = 200$, and $M = 10$. Unlike the case for the pure heat equation discussed in [22], the EPIC method based on central difference for both the convection and diffusion terms is only conditionally stable. It is clear from Fig. 8 that the value of $\rho(E+GC^{-1}F)$ becomes much larger than 1 when $|R|$ is close to 10^4.

Fig. 9 shows, on the other hand, much better stability of the EPIC method using upwind difference for the convection term, with $\alpha < 0$, $-10^{-4} \leq R \leq 0$, $0 \leq r \leq 10^4$, $L = 200$, and $M = 10$. In all cases, the spectral radiums is less than 1.

The analysis of the additional error caused by the numerical boundary condition from the EPIC method is similar to that of the EP method, since the first two steps in the

EPIC method is the same as the EP method and the corrector step does not introduce any additional truncation error to the scheme. Thus, the EPIC method with either central difference for both the convection and diffusion terms, or upwind difference for the convection term causes a truncation error of order $\mathcal{O}[\frac{\Delta t^2}{\Delta x^2}]$ at the boundaries for $\beta \neq 0$. For the pure convection equation with $\beta = 0$, the additional error is of order $\mathcal{O}[\frac{\Delta t}{\Delta x}]$. The accuracy of the solution in the interior of the subdomain is $\mathcal{O}[\Delta t^2]$ for $\beta = 0$ and $\mathcal{O}[\frac{\Delta t^2}{\Delta x}]$ for $\beta \neq 0$. In particular, if the grid size Δt and Δx are refined proportionally, the accuracy is $\mathcal{O}[\Delta t]$ for the convection diffusion equation with $\beta \neq 0$.

4 Parallel Implementation and Numerical Experiment

The EPIC method is highly parallel. The message passing standard MPI [11] is used in the code implementation to ensure maximum portability to a wide range of architectures, including both distributed and shared memory parallel computers, as well as clusters of workstations and personal computers.

For the example discussed in this paper, the processors are configured as a one-dimensional chain. The number of processors equals the number of subdomains, with each processor assigned to one subdomain.

In the first step of the computation, each processor calculates the numerical boundary condition(s) needed for its subdomain using an explicit method, such as (7). Note that the first and the last processor only need to calculate one numerical boundary condition, while other processors need to calculate two numerical boundary conditions. These computations can be done concurrently on all processors.

In the second step, each processor forms the system of linear algebraic equations similar to that in (12) using the numerical boundary conditions calculated at the first step, and then solves the system of equations. These computations can also be done concurrently on all processors.

After the solutions have been calculated, each processor must send to and receive from its neighboring processors the solutions at the points next to the end points of the subdomain. For the processor holding subdomain Ω_k with the end points r_{k-1} and r_k, it must send the calculated solutions at $r_{k-1}+1$ and r_k-1 to the left and right neighboring processors, respectively, and receive the newly calculated solutions at the points $r_{k-1}-1$ and r_k+1 from the left and right neighboring processors, respectively. Note that the first and the last processors on the chain only need to communicate with one neighboring processor. This is the only communication step involved in the EPIC method.

In the last step of the computation, each processor corrects the numerical boundary condition(s) at the end of its subdomain using an implicit method, such as (12). This can again be done in parallel.

The pseudo-code of the EPIC method for the p-th processor is given below:

```
Initialization
Read input data
```

Do $n = 0, ..., N-1$ (Loop over time steps)

1. *Do* $k = 1, ..., M-1$ (Loop over subdomain boundary points)
   If r_k is one of the two end points of the subdomain assigned to the p-th processor, then calculate $u_{r_k}^{n+1}$ using an explicit predictor
   End do
2. *Do* $k = 1, ..., M$ (Loop over subdomains)
   If Ω_k is assigned to the p-th processor, then
   - Form the system of equations for Ω_k
   - Solve the system of equations

   End do
3. *Do* $k = 1, ..., M-1$ (Loop over subdomain boundary points)
   If r_k is the left end point of the subdomain assigned to the p-th processor, then send $u_{r_k+1}^{n+1}$ to and receive $u_{r_k-1}^{n+1}$ from the left neighboring processor. Otherwise, if r_k is the right end point of the subdomain assigned to the p-th processor, then send $u_{r_k-1}^{n+1}$ to and receive $u_{r_k+1}^{n+1}$ from the right neighboring processor.
   End do
4. *Do* $k = 1, ..., M-1$ (Loop over subdomain boundary points)
   If r_k is one of the two end points of the subdomain assigned to the p-th processor, then update $u_{r_k}^{n+1}$ using an implicit corrector
   End do

End do

It is obvious from the above algorithm that the computations in step 1, 2, and 4 on different processors are completely independent to each other. Additionally, the inter-processor communications in step 3 can be grouped into pairs, which allow different pairs to exchange solutions concurrently, as shown in Fig. 10.

The following equation is used in our numerical experiment:

$$\begin{aligned} u_t + u_x &= \frac{1}{\pi^2} u_{xx}, \quad 0 < x < 1, \quad t > 0, \\ u(x,0) &= sin(\pi x), \quad x \in [0,1], \\ u(0,t) &= -e^{-t} sin(\pi t), \quad t > 0, \\ u(1,t) &= e^{-t} sin(\pi(1-t)), \quad t > 0, \end{aligned} \tag{34}$$

with an exact solution of $u^*(x,t) = e^{-t} sin(\pi(x-t))$.

Four different algorithms are used to calculate numerical solutions. The BTUS method refers to the use of the implicit algorithm (12) without domain decomposition. There is no need for numerical boundary condition in this case. TL, EP, and EPIC refer to

the use of time lagging, explicit predictor, and explicit predictor and implicit corrector method, respectively, to generate a numerical boundary condition at the middle point of the domain Ω, which is decomposed into two subdomains.

Table 1 shows the maximum errors of the solutions obtained using the BTUS, TL, EP and EPIC methods. Note that in this particular case, the errors from all methods are similar, with TL method being slightly more inaccurate. This is because the additional error terms caused by the numerical boundary conditions depend on the time derivative of the solution, which approaches zero as time t goes to infinity. At $t = 10.0$, the solution can be considered as having reached steady state within machine accuracy. Therefore, the additional errors caused by the numerical boundary conditions are negligible, and all methods appear to be reasonably accurate with similar spatial accuracy.

Table 2 has similar contents as those in Table 1, except that the solution is calculated to the time level of $T = 1.0$. It now appears that the EP method, with similar errors as those from the BTUS algorithm, is more accurate than the TL method. However, the results from Table 3, also calculated to $T = 1.0$ using $\Delta t = \Delta x$, shows that both EP and TL fail to deliver solutions with similar accuracy as those from the BTUS algorithm. The errors from the TL method roughly remain at a constant level as the grid size Δt and Δx are refined proportionally, while that from the EP method indicate that it has lost stability. The results from the EPIC method, on the other hand, have similar accuracy as that from the BTUS method applied to the entire domain without decomposition.

Table 1: Maximum errors: $\Delta x^2 = \Delta t$, $T = 10.0$, $M = 2$, $CFL = \Delta x$.

Δx	**BTUS**	**TL**	**EP**	**EPIC**
0.1000	0.261e-05	0.266e-05	0.235e-05	0.245e-05
0.0500	0.656e-06	0.673e-06	0.620e-06	0.634e-06
0.0250	0.164e-06	0.107e-06	0.159e-06	0.161e-06
0.0100	0.263e-07	0.244e-07	0.260e-07	0.261e-07
0.0050	0.658e-08	0.615e-08	0.654e-08	0.656e-08
0.0025	0.164e-08	0.260e-08	0.164e-08	0.164e-08
0.0010	0.834e-09	0.813e-09	0.833e-09	0.833e-09
0.0005	0.163e-09	0.212e-09	0.171e-09	0.162e-09

Fig. 11 shows that the error from the EPIC method with upwind difference for the convection term does not increase as the number of subdomains increases. In this particular case, we have $\Delta x = 0.005$, $\Delta t = 0.0005$ and the numerical solution is advanced to $T = 0.5$. Similar results can be obtained for the cases when the convection term is discretized by the central finite difference scheme.

Table 2: Maximum errors: $(\Delta x)^2 = \Delta t$, $T = 1.0$, $M = 2$, $CFL = \Delta x$.

Δt	**BTUS**	**TL**	**EP**	**EPIC**
0.1000	0.238e-01	0.422e-01	0.211e-01	0.222e-01
0.0500	0.587e-02	0.154e-01	0.552e-02	0.566e-02
0.0250	0.146e-02	0.626e-02	0.141e-02	0.143e-02
0.0100	0.234e-03	0.215e-02	0.231e-03	0.232e-03
0.0050	0.585e-04	0.101e-02	0.581e-04	0.583e-04
0.0025	0.146e-04	0.493e-03	0.145e-04	0.146e-04
0.0010	0.234e-05	0.193e-03	0.233e-05	0.233e-05
0.0005	0.129e-05	0.411e-04	0.198e-05	0.177e-05

Table 3: Maximum errors: $\Delta x = \Delta t$, $T = 1.0$, $M = 2$, $CFL = 1.0$.

Δx	**BTUS**	**TL**	**EP**	**EPIC**
0.1000	0.146e+0	0.330e+0	0.106e+0	0.810e-1
0.0500	0.816e-1	0.281e+0	0.842e+3	0.447e-1
0.0250	0.434e-1	0.248e+0	0.281e+16	0.253e-1
0.0100	0.180e-1	0.226e+0	∞	0.109e-2
0.0050	0.917e-2	0.218e+0	∞	0.563e-2
0.0025	0.462e-2	0.214e+0	∞	0.285e-2
0.0010	0.185e-2	0.212e+0	∞	0.115e-2
0.0005	0.929e-3	0.211e+0	∞	0.577e-3

5 Application to Euler Equations

In this section, we discuss the application of the EPIC method to the solution of the Euler equations in the flow field calculation around an airfoil. The two-dimensional Euler equations expressed in Cartesian coordinates and conservation form are given by

$$\frac{\partial \hat{Q}}{\partial t} + \frac{\partial \hat{E}}{\partial x} + \frac{\partial \hat{F}}{\partial y} = 0, \tag{35}$$

where

$$\hat{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad \hat{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{bmatrix}, \quad \hat{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \end{bmatrix}, \tag{36}$$

with the equation of state given by

$$p = (\gamma - 1)\{e - \frac{p}{2}(u^2 + v^2)\}, \tag{37}$$

where ρ is the mass density, u and v are the velocity components in the x and y directions, respectively, e is the total specific energy of the fluid, p is the pressure, and γ is the ratio of specific heats.

For complex geometry, this system of equations need to be transformed using a general body-fitted curvelinear coordinate systems:

$$\begin{aligned} \xi &= \xi(x, y, t), \\ \eta &= \eta(x, y, t), \\ \tau &= t. \end{aligned} \tag{38}$$

The transformed Euler equations can then be written as

$$\frac{\partial Q}{\partial \tau} + \frac{\partial E}{\partial \xi} + \frac{\partial F}{\partial \eta} = 0, \tag{39}$$

where

$$\begin{aligned} Q &= J^{-1}\hat{Q}, \\ E &= J^{-1}(\xi_x \hat{E} + \xi_y \hat{F}), \\ F &= J^{-1}(\eta_x \hat{E} + \eta_y \hat{F}). \end{aligned} \tag{40}$$

The Jacobian J of the transformation is given by

$$J = \xi_x \eta_y - \xi_y \eta_x, \tag{41}$$

and the metric terms are

$$\begin{aligned} \xi_x &= y_\eta J, \qquad \eta_x = -y_\xi J, \\ \xi_y &= -x_\eta J, \qquad \eta_y = x_\xi J. \end{aligned} \tag{42}$$

The algorithm applied to the entire domain without decomposition is an implicit finite volume scheme that is first order accurate in time and up to third order accurate in space using Roe's approximate Riemann solver [23]. The original sequential code was provided to the authors by the Computational Fluid Dynamics Laboratory at Mississippi State University.

For parallel processing, the computational domain was decomposed along the ξ dimension. All processors read the same input files concurrently to obtain grid data and initial conditions. Each processor performs the calculations on assigned subdomains, and the partial solutions from different processors are collected and assembled into an external file by a master processor.

Since the computational domain is decomposed only in the ξ-direction, the loops in the η-direction in the original code are not affected. Thus, only the indices corresponding to the ξ-direction need to be modified to distribute computations in different subdomains to different processors. Interprocessor communication between processors is carried out using the MPI standard library [11] to ensure maximum portability.

In this application, the MacCormack scheme [18] is used for the predictor step. The Euler solver based on the Roe's scheme [28] is used to calculate solutions in all subdomains

concurrently, and to update the boundary data between subdomains. Fig. 12 shows the flow chart of the computational process described here.

In order to demonstrate applicability, accuracy, and performance of the EPIC method for solving Euler equations, a series of test cases for transonic flow calculation around a NACA0012 airfoil were carried out. Fig. 13 shows the grid used for these calculations, which is a 290×81 C-grid. Fig. 14 illustrates the decomposition of the original grid into four subdomains.

The steady state calculations are for the NACA0012 airfoil at Mach number $M_\infty = 0.85$ and angle of attack $\alpha = 1.0$, characterized by the presence of a strong shock on the upper surface of the airfoil and a weaker, but significant, shock on the lower surface. The numerical results are compared with the experimental data in the AGARD Report [27]. Since the EP method suffers from a severe stability restriction in this application, only the results from three methods are compared with the experimental data: 1) Single domain compuation without decomposition, 2) TL method with domain decomposition, and 3) EPIC method with domain decomposition. Fig. 15 shows the pressure distributions obtained after 1000 local time steps using CFL=15 from all three methods, as well as the experimental data from the AGARD report. Four subdomains were used for the TL and EPIC methods. Note that the results from all three methods are very close to the experimental data, which is not surprising based on the previous analysis.

The unsteady calculations correspond to the flow around the NACA0012 airfoil pitching about the quarter chord point [3]. The movement of the airfoil is prescribed such that the angle of attack varies sinusoidally according to the following relation

$$\alpha(t) = \alpha_m + \alpha_0 \sin(M_\infty k t), \tag{43}$$

where α_m is the mean angle of attack, α_0 is the amplitude of the unsteady angle of attack, and k is the reduced frequency defined as

$$k = \frac{wc}{V_\infty}, \tag{44}$$

where w is the frequency, c is the chord length, and V_∞ is the freestream velocity. In this test case, the NACA0012 airfoil is assumed to be pitching at $M_\infty = 0.755$, $k = 0.1628$, $\alpha_m = 0.016$, and $\alpha_0 = 2.51$. The numerical results are compared with the experimental data by Landon [15].

The unsteady calculations were started from a converged steady state solution and afterward the CFL number is kept at 1000 during the simulation. Figures 16 and 17 show the pressure distribution for different angles of attack. Eight subdomains are used in the computation for the TL and EPIC methods. It is clear from figures 16 and 17 that the pressure distributions obtained using the EPIC method matches very well with those obtained using the Roe's approximate Riemann solver without domain decomposition, as well as the experimental data. The TL method, on the other hand, produces a considerable error in shock locations for both angles of attack.

The numerical results in these test cases show that both the TL and EPIC method are acceptable for the steady state computations. However, for unsteady state calculations,

the EPIC method yields significantly more accurate solutions when domain decomposition is used.

Fig. 18 shows the speedup for the calculation of steady state solution using the TL and EPIC methods. The computations were carried out on an SGI Power Challenge XL parallel computer with 16 processors. It is clear from the figure that both methods are highly scalable with almost ideal speedup, which can be maintained on more procesors by increasing the problem size.

6 Conclusion

Both the TL and the EP methods, in particular the TL method, have been widely used in solving time dependent PDEs combined with domain decomposition or multiblock grids. Detailed stability and accuracy analyses in this paper show that, for convection-diffusion equations, the TL algorithm is stable, but in general reduces accuracy for calculating unsteady (transient) solutions. On the other hand, the EP method is accurate, but only conditionally stable. The EPIC method using upwind finite difference for the convection term combines the advantages of both the TL (stability) and EP (accuracy) methods. Unlike the case for pure diffusion equations, the EPIC method using central difference for both the convection and diffusion terms is only conditionally stable.

Application to the solution of nonlinear system of Euler equations in flow simulation of an airfoil shows that the analysis and conclusion drawn from the one-dimensional linear model equations about the TL, EP, and EPIC methods is also applicable to systems of higher dimensional nonlinear equations. Numerical results demonstrate that the EPIC method is as stable and scalable as the TL method, and is more accurate than the TL method for calculating unsteady solutions on parallel computers.

References

[1] S. Barnard, S. Saini, R. Van der Wijngaart, M. Yarrow, L. Zechtzer, I. Foster, and O. Larsson, "Large-scale distributed computational fluid dynamics on the information power grid using Globus," in Proceedings of the 7th Symposium on the Frontiers of Massively Parallel Computation, 1999.

[2] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods," SIAM: Philadelphia, 1994.

[3] D. M. Belk, *Three-dimensional Euler Equations Solutions on Dynamic Blocked Grids*. PhD thesis, Mississippi State University, 1986.

[4] E. Brakkee, A. Segal, and C. G. M. Kassels, "A parallel domain decomposition algorithm for the incompressible Naiver-Stokes equations," *Simulation Practice and Theory,* 3 (1995), pp. 185-205.

[5] T. F. Chan and T. P. Mathew, "Domain decomposition algorithms," *Acta Numerica,* 3 (1994), pp. 61-143.

[6] C. N. Dawson, Q. Du, and T. F. Dupont, "A finite difference domain decomposition algorithm for numerical solution of the heat equations," *Mathematics of Computation*, **57** (1991), pp. 63 - 71.

[7] J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. Van der Vorst, "Numerical Linear Algebra for High-Performance Computers," SIAM: Philadelphia, 1998.

[8] D. Drikakis and E. Schreck, "Development of parallel implicit Navier-Stokes solvers on MIMD multi-processor systems," *AIAA* 93-0062.

[9] G. Golub and J. M. Ortega, Scientific Computing, An Introduction with Parallel Computing, Academic Press, San Diego, CA, 1993.

[10] B. Gustafsson, "The convergence rate for difference approximations to mixed initial boundary value problems," *Mathematics of Computation*, vol. 29, pp. 396–406, 1975.

[11] W. Gropp, M. Snir, B. Nitzberg, E. Lusk, "MPI: The Complete Reference," MIT Press, Cambridge, 1998.

[12] B. Heise, "Nonlinear simulation of electromagnetic fields with domain decomposition methods on MIMD parallel Computers," *Journal of Computational and Applied m Mathematics,* 63 (1995), pp. 373-381.

[13] C. Hirsch, "Numerical Computation of Internal and External Flows," Volume I, John Wiley and Sons: New York, 1988.

[14] Y. A. Kuznetsov, "New algorithms for approximate realization of implicit difference schemes," *Sovietic Journal of Numerical Analysis and Mathematical Modeling,* 3 (1988), pp. 99-114.

[15] R. H. Landon, "Naca0012 oscillation and transient pitching," in Compendium of Unsteady Aerodynamic Measurements, Advisory Report 702, AGARD, 1982.

[16] P. L. Lions, "On the Schwartz alternating method I," *First International Symposium on Domain Decomposition Methods for Partial Differential Equations,* R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., SIAM, Philadelphia, 1988, pp. 1-42.

[17] Y. Lu and C. Y. Shen, "A domain decomposition finite-difference method for parallel numerical implementation of time dependent Maxwell's equations," *IEEE Transactions on Antennas and Propagation,* 45 (1997), pp. 1261.

[18] R. W. MacCormack, "The effects of viscosity in hypervelocity impact cratering," AIAA 69-354, 1969.

[19] S. Oh, S. Paik, and H. D. Nguyen, "Domain decomposition method for heat transfer problem using parallel distributed computing," *Journal of Scientific Computing,* 12 (1997), pp. 187-204.

[20] R. Pankajakshan and W. R. Briley, "Parallel solution of viscous incompressible flow on multi-block structured grids using MPI," *Parallel Computational Fluid Dynamics: Implementation and Results Using Parallel Computers,* A. Ecer, J. Periaux, N. Satofuka, and S. Taylor, ed., Elsevier Science, Amsterdam, 1996, pp. 601-608.

[21] A. Povitsky and P. J. Morris, "A parallel compact multi-dimensional numerical algorithm with aeroacustics application," ICASE Report 99-34, 1999.

[22] H. Qian and J. Zhu, "On an efficient parallel algorithm for solving time dependent partial differential equations," *Proceedings of the 1998 International Conference on Parallel and Distributed Processing Technology and Applications*, H. R. Arabnia Ed., CSREA Press, Athens, GA 1998, pp. 394 - 401.

[23] P. L. Roe, "Approximate Riemann solvers, parameter vector, and difference schemes," *Journal of Computational Physics*, vol. 43, pp. 357–372, 1981.

[24] H.W. Schwarz, "Gesammelete Mathematische Abhandlungen," 2 (1890), pp. 133-143.

[25] X. H. Sun, Application and accuracy of the parallel diagonal dominant algorithm, Parallel Computing, **21**(8), 1241 - 1268, 1995.

[26] J. W. Thomas, "Numerical Partial Differential Equations: Finite Difference Methods," Springer-Verlag: New York, 1995.

[27] H. Viviand, "Numerical solutions of two-dimensional reference test cases," in Test Cases for Inviscid Flow Field Methods, Advisory Report 211, AGARD, 1985.

[28] D. L. Whitfield, J. M. Janus, and L. B. Simpson, "Implicit finite volume high resolution wave split scheme for solving the unsteady three-dimensional Euler and Navier-Stokes equations on stationary or dynamic grids," Engineering and Industrial Research Report MSSU-EIRS-ASE-88-2, Mississippi State University, 1988.

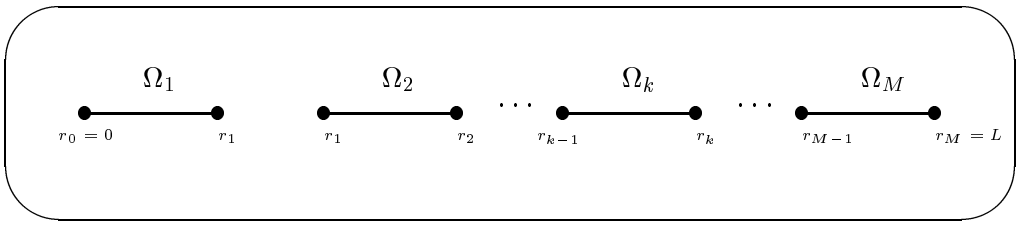


Figure 1: The original domain Ω is decomposed into M subdomains $\Omega_k, k = 1, \ldots, M$.

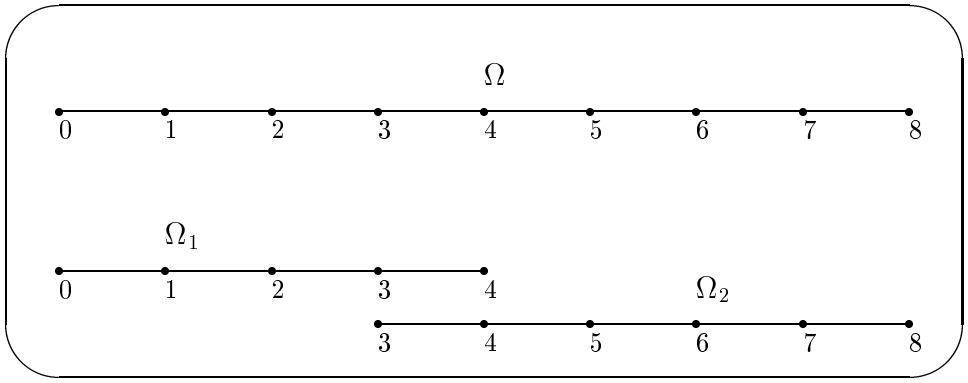


Figure 2: Time-lagging domain decomposition.

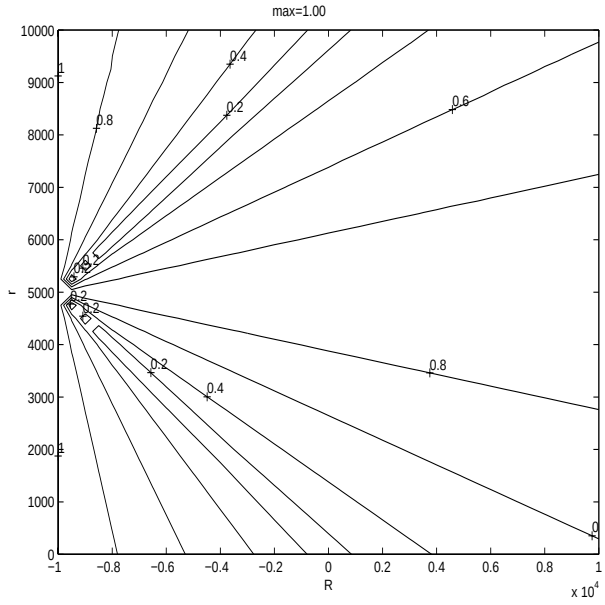


Figure 3: Spectral radius ρ vs. (R, r): TL method with central difference, $L = 200$, $M = 10$.

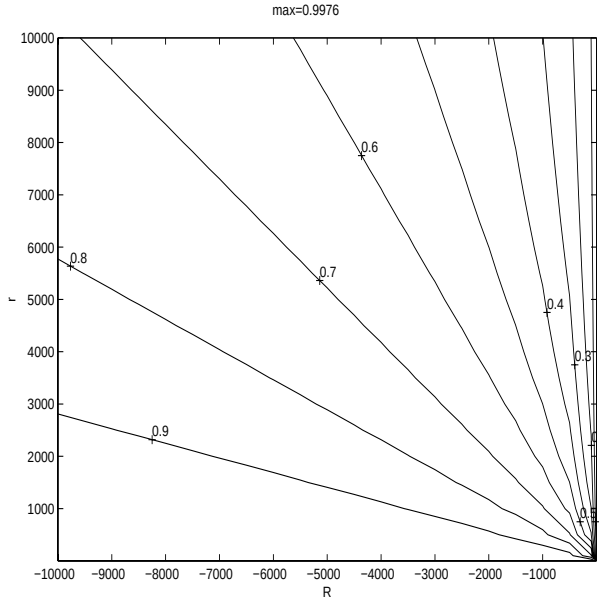


Figure 4: Spectral radius ρ vs. (R, r): TL method with upwind (forward) difference, $L = 200$, $M = 10$.

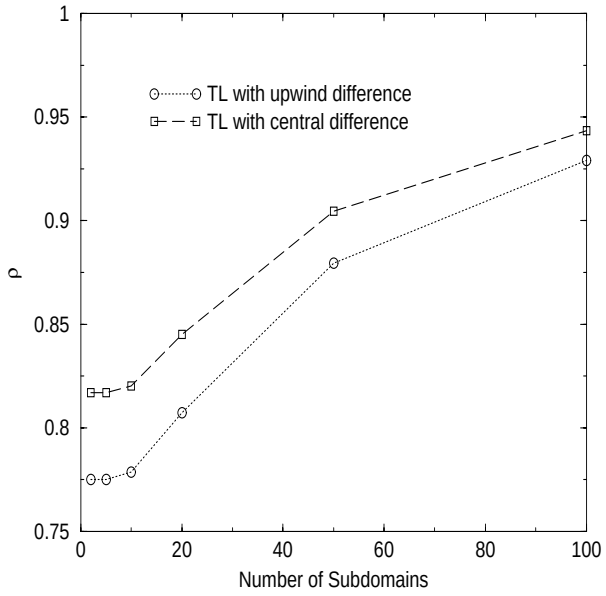


Figure 5: Spectral radius ρ vs. number of subdomains: TL method, $L = 1000$, $R = 500$, $r = 1000$.

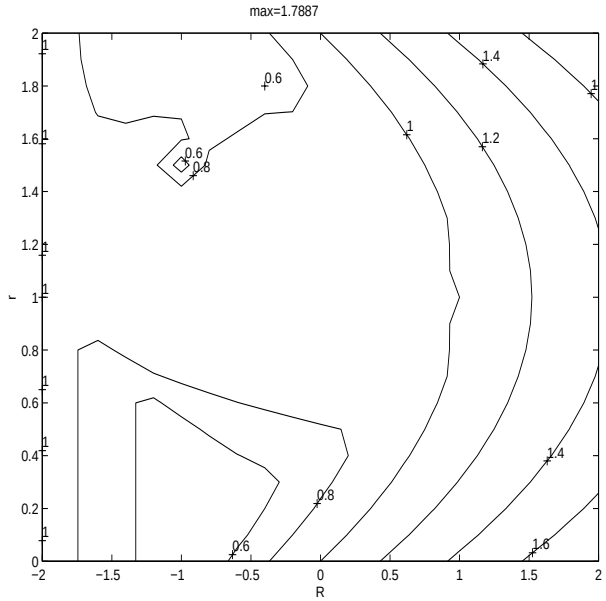


Figure 6: Spectral radius ρ vs. (R, r): EP method with central difference, $L = 200$, $M = 10$.

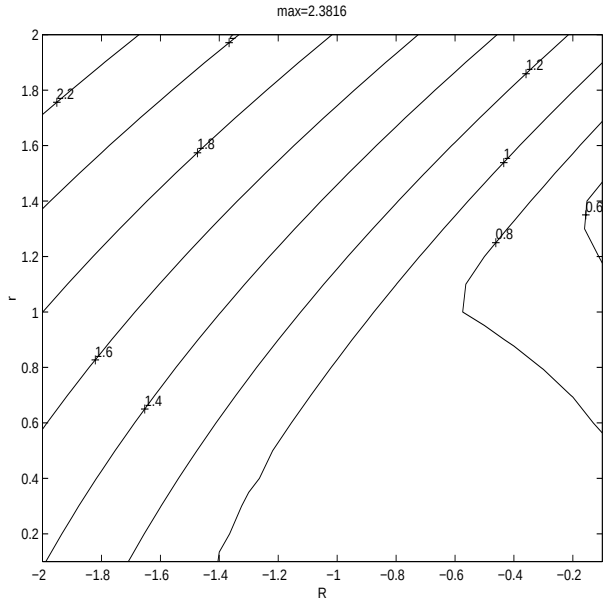


Figure 7: Spectral radius ρ vs. (R, r): EP method with upwind (forward) difference, $L = 200$, $M = 10$.

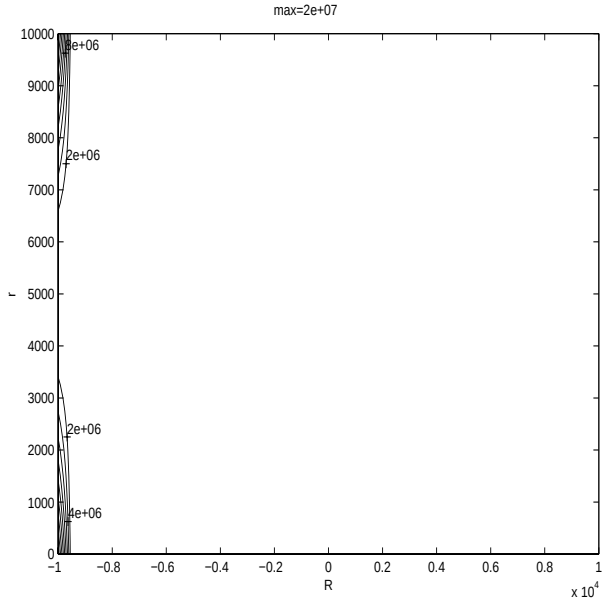


Figure 8: Spectral radius ρ vs. (R, r): EPIC method with central difference, $L = 200$, $M = 10$.

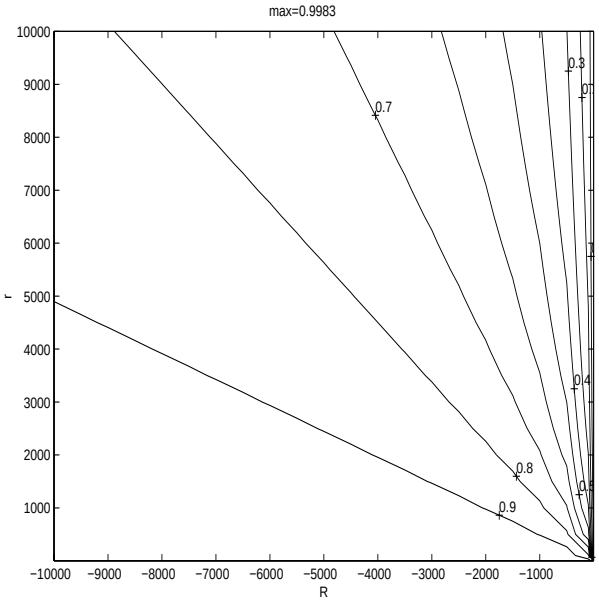
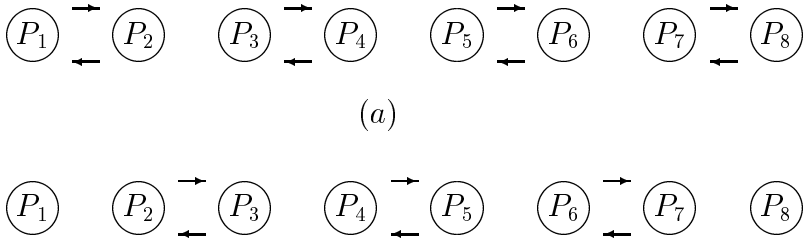


Figure 9: Spectral radius ρ vs. (R, r): EPIC method with upwind (forward) difference, $L = 200$, $M = 10$.



(a)

(b)

Figure 10: Interprocessor communications between 8 processors. In (a), communications between 4 pairs of processors can be done in parallel, and in (b), communications between 3 pairs of processors can be done in parallel.

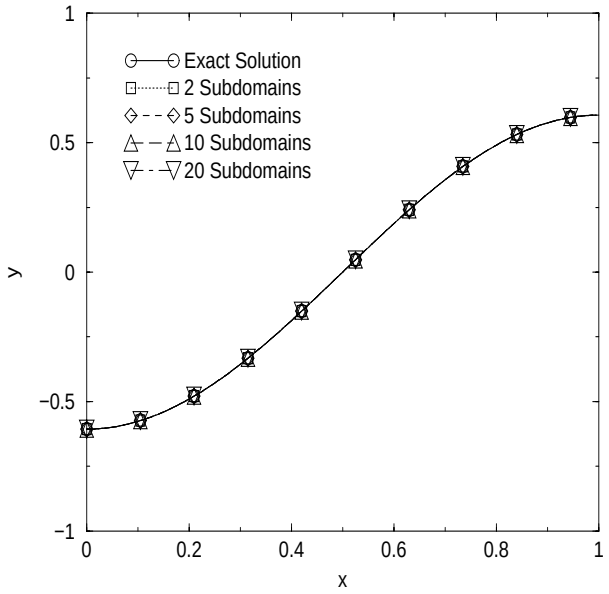


Figure 11: Maximum error (y) vs. number of subdomains: EPIC method, $\Delta x = 0.005$, $\Delta t = 0.0005$, $T = 0.5$.

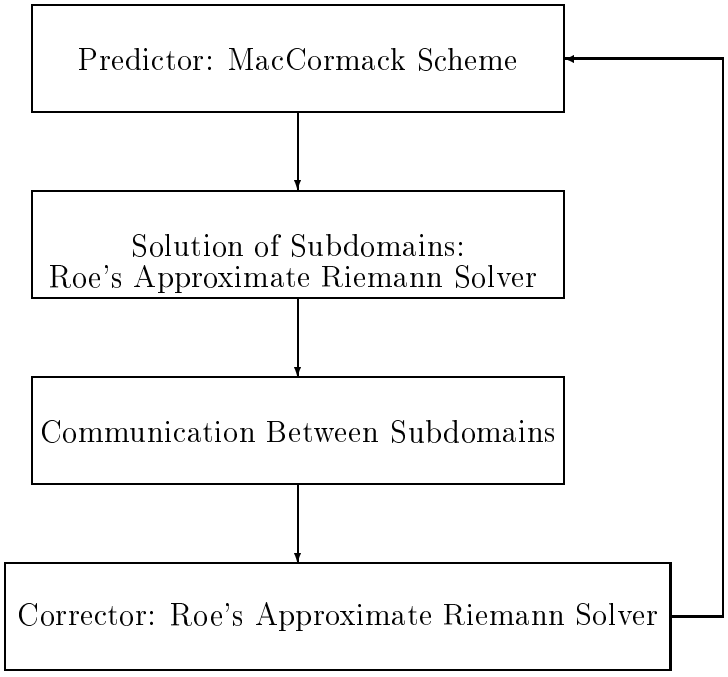


Figure 12: Flow Chart of the parallel code for solving the Euler equations.

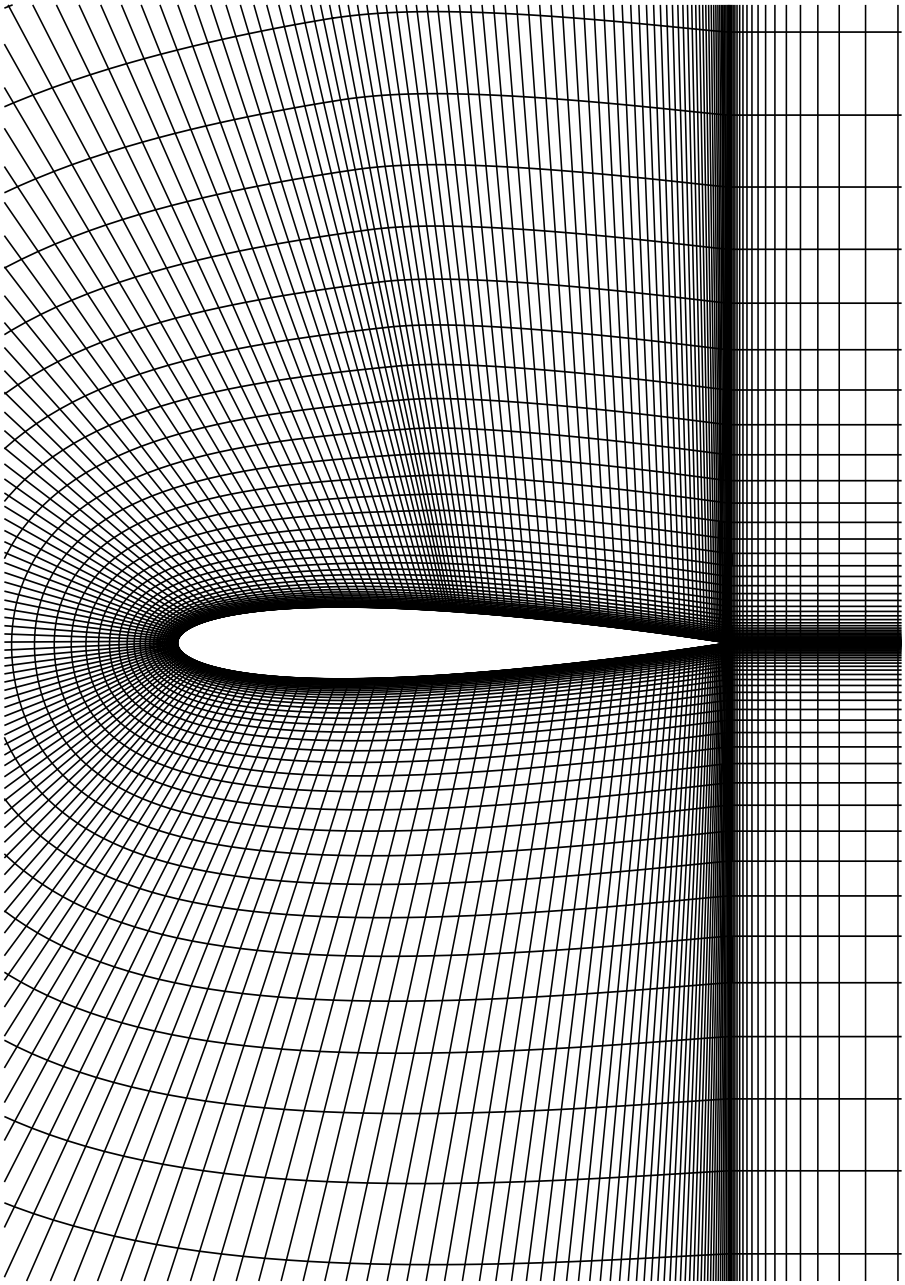


Figure 13: A 290×81 grid for the NACA0012 airfoil.

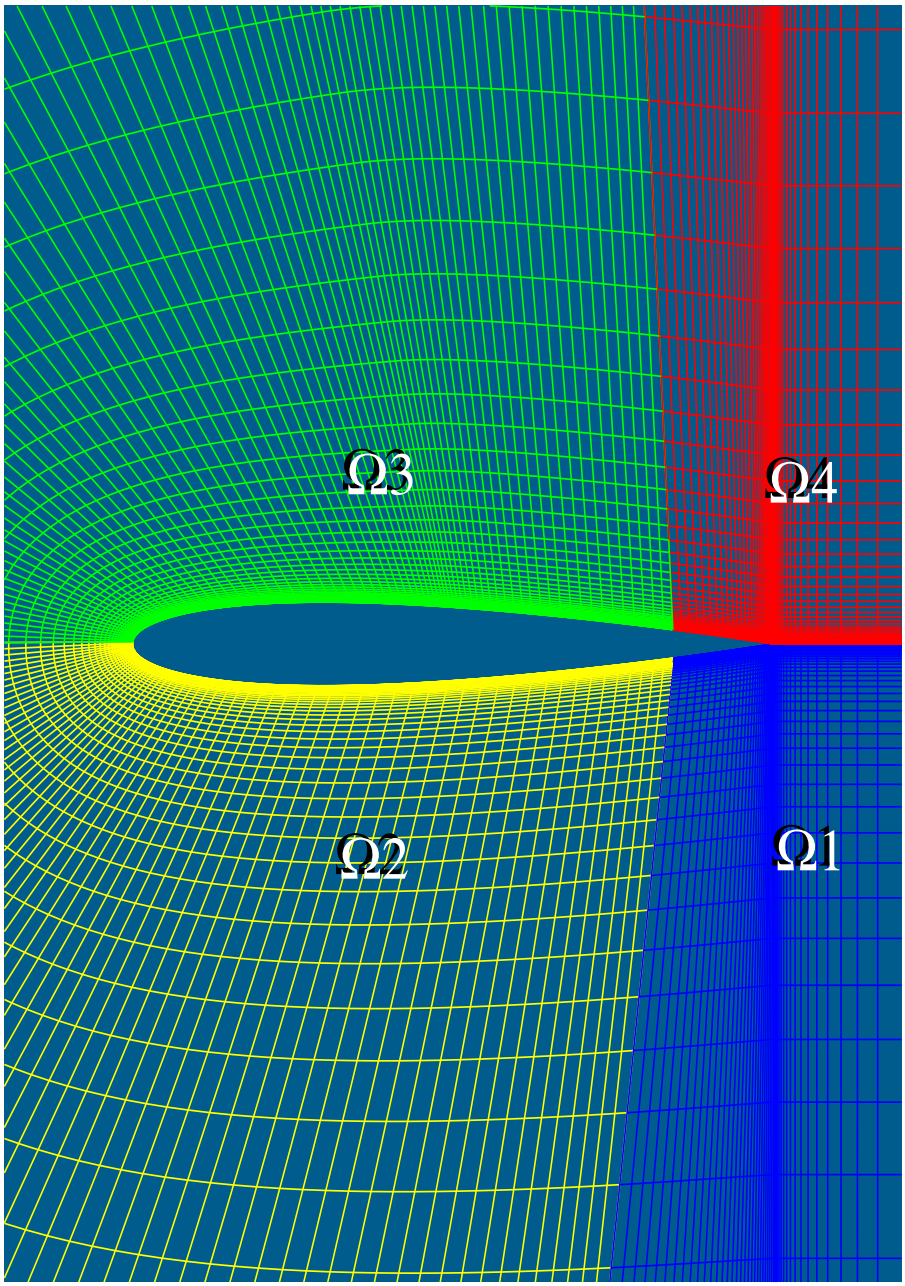


Figure 14: A 4-subdomain decomposition for the NACA0012 airfoil.

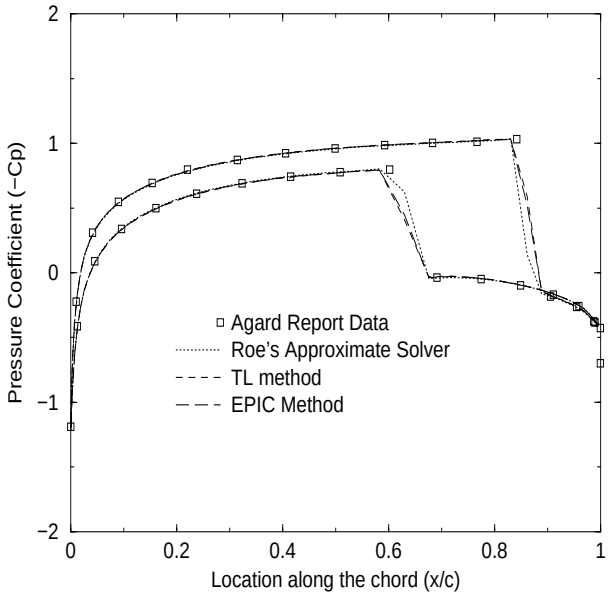


Figure 15: NACA 0012 steady pressure distribution: $M_\infty = 0.85$, $\alpha = 1.0$, $CFL = 15$, 4 Subdomains.

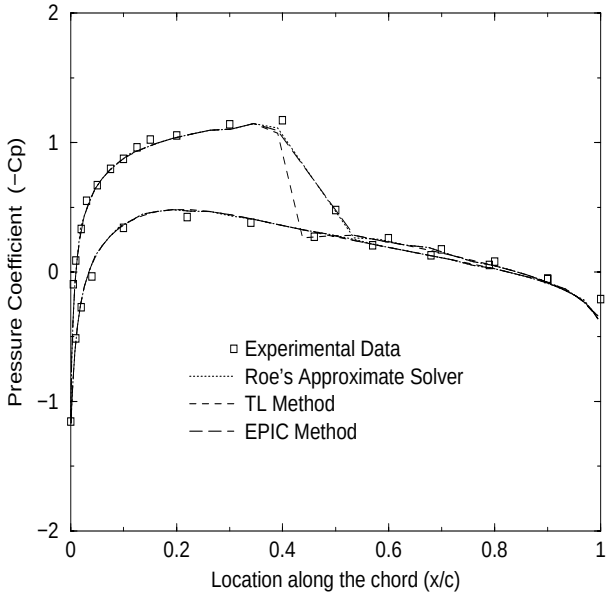


Figure 16: NACA 0012 unsteady pressure distribution: $\alpha = 2.34$.

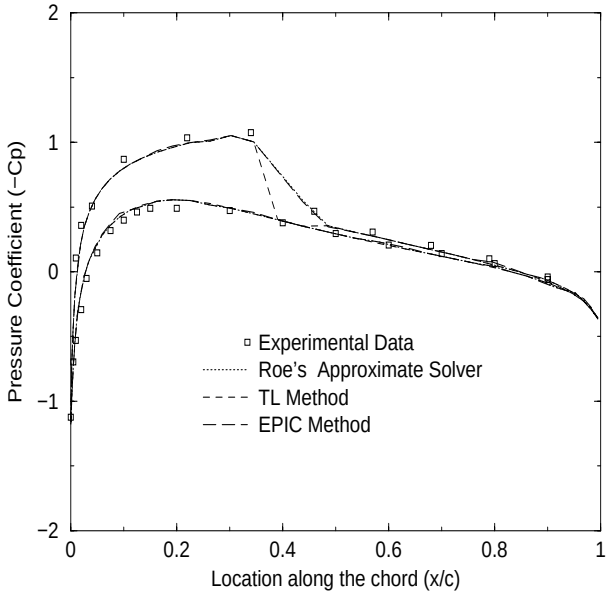


Figure 17: NACA 0012 unsteady pressure distribution: $\alpha = -2.41$.

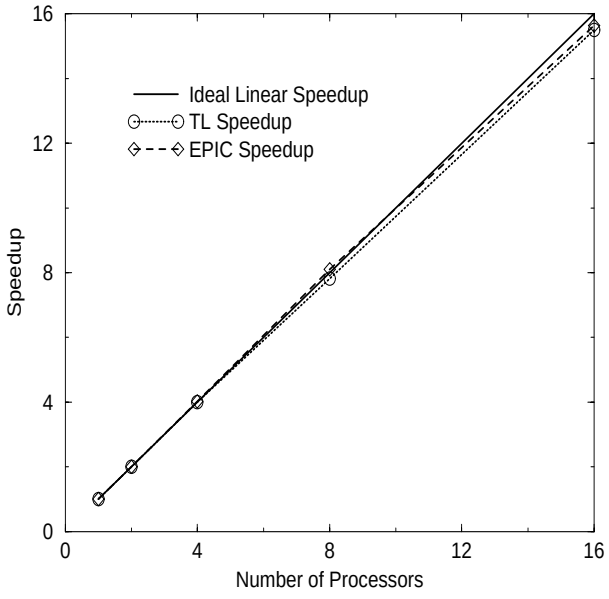


Figure 18: Speedup: 290×81 grid for the NACA0012 airfoil, steady state flow calculation.