95

A Quasi-Exact Interface Condition for Implicit Multiblock Euler and Navier-Stokes Calculations

Guy De Spiegeleer and Alain Lerat

1 Introduction

In most domain decomposition methods for Computational Fluid Dynamics, the interface conditions are treated explicitly. When using implicit schemes, this causes stability or efficiency limitations (see [Rai86, Jen94]). For implicit schemes leading to the solution of block-tridiagonal linear systems, one can use the parallel factorisation technique developed in [Wan81] or in [Bru91]. However, this technique needs the sequential computation of a reduced system, which slightly degrades the parallelisation rate of the algorithm.

In this paper, we introduce a completely parallelizable algorithm which yields accurate interface conditions at a quite low cost. It is deduced from the Schwarz alternating method with a fictitious overlapping mesh [Lio88]. We transform this algorithm into another much more efficient one since the latter is equivalent to a one-block overlapping Schwarz algorithm. It allows a strict parallelization because all computations are done independently in each block. Communications are limited to neighbouring blocks. The efficiency of the multiblock implicit solver is assessed by numerical calculations of inviscid and viscous compressible flows around a NACA0012 airfoil.

2 Numerical Scheme

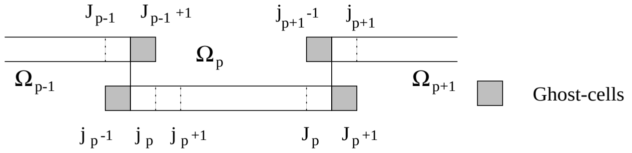
We consider the one-dimensional conservation law system :

$$w_t + f_x = 0, \qquad x \in \Omega \subset \mathbb{R} \tag{2.1}$$

where $w(x,t) \in \mathbb{R}^m$ denotes the vector of conservative variables and f is the flux vector. The flux Jacobian matrix $A = df/dw$ has real eigenvalues and a complete set of eigenvectors.

Ninth International Conference on Domain Decomposition Methods
Editor Petter E. Bjørstad, Magne S. Espedal and David E. Keyes

Figure 1 Multiblock mesh and ghost-cells definition.



Implicit Single-Block Scheme

The numerical solution is denoted by $w_j^n \approx w\,(j\delta x, n\Delta t)$, for $j \in \mathcal{J}_0 = \{j_0, \ldots, J_0\}$, Δt and δx are respectively the time-step and the space-increment, $\sigma = \Delta t/\delta x$. System (2.1) is approximated by a three-point implicit scheme which involves two time levels. The explicit stage reads :

$$\Delta w_j^{exp} = -\sigma\left[h\left(w_{j+1}, w_j\right) - h\left(w_j, w_{j-1}\right)\right],\ j \in \mathcal{J}_0$$

where h is the numerical flux associated with f. Note that without significant change, the explicit stage could involve more than three points. The linear implicit stage is given by :

$$n_j^- \Delta w_{j-1} + n_j^0 \Delta w_j + n_j^+ \Delta w_{j+1} = \Delta w_j^{exp},\ j \in \mathcal{J}_0 \tag{2.2}$$

where $\Delta w_j = w_j^{n+1} - w_j^n$. The $m \times m$ matrix blocks n_j^-, n_j^0 and n_j^+ depend on w_{j-1}^n, w_j^n and w_{j+1}^n. The single-block computation of the implicit stage consists in solving the $(J_0 - j_0 + 1) \times (J_0 - j_0 + 1)$ block-tridiagonal matrix $\mathcal{N}_0 = [n_j^-, n_j^0, n_j^+]$, with the Thomas algorithm. Here the vector values Δw_{j_0-1} and Δw_{J_0+1} are assumed to be null for convenience.

Implicit Multiblock Scheme

The computational domain Ω is partitioned into P blocks Ω_p, $p \in \{1, \ldots, P\}$. The block Ω_p is composed of adjacent mesh-cells c_j, $j \in \mathcal{J}_p = \{j_p, \ldots, J_p\}$. A fictitious overlapping is introduced so that the ghost-cells c_{j_p-1} and c_{J_p+1} respectively overlap the interior cells $c_{J_{p-1}}$ and $c_{j_{p+1}}$ (see Fig. 1).

In order to ensure the solution uniqueness, we need interface conditions at the ghost-cells c_{j_p-1} and c_{J_p+1}. The definition of the exact explicit interface condition easily comes from the identification of the multiblock computation using ghost-cells with the equivalent single-block computation. At $j = j_p$ we have for the single-block computation :

$$\Delta w_{j_p}^{exp} = -\sigma\left[h\left(w_{j_p+1}, w_{j_p}\right) - h\left(w_{j_p}, w_{j_p-1}\right)\right]$$

and for the multiblock computation :

$$\Delta w_{j_p}^{exp} = -\sigma\left[h\left(w_{j_p+1}, w_{j_p}\right) - h\left(w_{j_p}, w_{J_{p-1}}\right)\right]$$

Identifying at $j = J_p$, we deduce the exact explicit interface condition :

$$w_{j_p-1}^n = w_{J_{p-1}}^n \text{ and } w_{J_p+1}^n = w_{j_{p+1}}^n, \qquad p \in \{1, \ldots, P\} \tag{2.3}$$

In the same way, we obtain the exact implicit interface condition :

$$\Delta w_{j_p-1} = \Delta w_{J_{p-1}} \text{ and } \Delta w_{J_p+1} = \Delta w_{j_{p+1}}, \qquad p \in \{1, \ldots, P\} \tag{2.4}$$

The implicit interface condition (2.4) couples the implicit stage systems of the P blocks. Consequently, we cannot compute the implicit scheme independently in each block. To uncouple the systems, one solution consists in lagging in time the interface values on the adjacent block so that the interface condition becomes :

$$\begin{array}{lcl} w^n_{j_p-1} = w^{n-1}_{J_{p-1}} & \text{and} & w^n_{J_p+1} = w^{n-1}_{j_{p+1}} \\ \Delta w_{j_p-1} = \Delta w^{n-1}_{J_{p-1}} & \text{and} & \Delta w_{J_p+1} = \Delta w^{n-1}_{j_{p+1}} \end{array}, \qquad p \in \{1, \ldots, P\} \tag{2.5}$$

where $\Delta w^{n-1} = w^n - w^{n-1}$. This time-lagging condition has been proved not to jeopardize the linear stability of the interface problem [LW96]. However, the interface condition (2.5) causes a partition dependency which can degrade the solver efficiency for a large number of blocks.

3 New Implicit Interface Condition

The reduced system

The construction of the implicit interface condition proposed in this paper begins like a classical parallelizable tridiagonal-linear system solver. We compute the influence of the ghost-cells on the interior-cell time-increments. The implicit multiblock scheme using ghost-cells is written in each block $p \in \{1, \ldots, P\}$ as :

$$\underbrace{\begin{bmatrix} n^0_{j_p} & n^+_{j_p} & & \\ n^-_{j_p+1} & \ddots & \ddots & \\ & \ddots & \ddots & n^+_{J_p-1} \\ & & n^-_{J_p} & n^0_{J_p} \end{bmatrix}}_{\mathcal{N}_p} \begin{bmatrix} \Delta w_{j_p} \\ \Delta w_{j_p+1} \\ \vdots \\ \Delta w_{J_p-1} \\ \Delta w_{J_p} \end{bmatrix} = \begin{bmatrix} \Delta w^{exp}_{j_p} \\ \Delta w^{exp}_{j_p+1} \\ \vdots \\ \Delta w^{exp}_{J_p-1} \\ \Delta w^{exp}_{J_p} \end{bmatrix} - \begin{bmatrix} n^-_{j_p} \Delta w_{j_p-1} \\ 0 \\ \vdots \\ 0 \\ n^+_{J_p} \Delta w_{J_p+1} \end{bmatrix} \tag{3.6}$$

The vector values Δw_{j_1-1} and Δw_{J_P+1} are assumed to be null. The inversion of the matrix $\mathcal{N}_p$ shows that the linear system (3.6) is a linear form which links the ghost-cell values Δw_{j_p-1} and Δw_{J_p+1} to the time-increments Δw_j, $j \in \{j_p, \ldots, J_p\}$:

$$\Delta w_j = \Delta \overline{w}_j + B^-_j \Delta w_{j_p-1} + B^+_j \Delta w_{J_p+1}, \qquad j \in \mathcal{J}_p \tag{3.7}$$

where $\Delta \overline{w}_j$ is the numerical value of the time-increment in Ω_p assuming a zero value at each interface, and the matrices B^-_j and B^+_j are the *influence matrices*. The computation of $\Delta \overline{w}_j$, B^-_j and B^+_j only depends on the interior data in Ω_p. They are computed independently in each block using the following algorithm adapted from the Thomas algorithm.

$$
\left|
\begin{array}{l}
U_{j_p-1} = 0 \\
C_{j_p-1} = I \\
\overline{y}_{j_p-1} = 0 \\
\textbf{DO } j = j_p, J_p \\
\quad D_j = -n_j^- U_{j-1} + n_j^0 \\
\quad U_j = (D_j)^{-1} n_j^+ \\
\quad C_j = -(D_j)^{-1} n_j^- C_{j-1} \\
\quad \overline{y}_j = (D_j)^{-1} \left(\Delta w_j^{exp} - n_j^- \overline{y}_{j-1} \right) \\
\textbf{ENDDO} \\
\Delta \overline{w}_{J_p} = \overline{y}_{J_p} \\
B_{J_p}^- = C_{J_p} \\
B_{J_p}^+ = -U_{J_p} \\
\textbf{DO } j = J_p - 1, j_p \\
\quad \Delta \overline{w}_j = \overline{y}_j - U_j \Delta \overline{w}_{j+1} \\
\quad B_j^- = C_j - U_j B_{j+1}^- \\
\quad B_j^+ = -U_j B_{j+1}^+ \\
\textbf{ENDDO}
\end{array}
\right.
$$

Now we assemble the interface problem (or *reduced system* or *global Schur complement operator*) whose unknowns are the ghost-cell values. It corresponds to the linear form (3.7) for $j = J_1, j_2, \ldots, j_p, J_p, \ldots, J_{P-1}, j_P$, where the exact implicit interface condition (2.4) has been applied :

$$
\left\{
\begin{array}{llll}
\Delta w_{J_1} & = \Delta \overline{w}_{J_1} & & + B_{J_1}^+ \Delta w_{j_2} \\
\Delta w_{j_2} & = \Delta \overline{w}_{j_2} & + B_{j_2}^- \Delta w_{J_1} & + B_{j_2}^+ \Delta w_{j_3} \\
 & \ldots & & \\
\Delta w_{j_p} & = \Delta \overline{w}_{j_p} & + B_{j_p}^- \Delta w_{J_{p-1}} & + B_{j_p}^+ \Delta w_{j_{p+1}} \\
\Delta w_{J_p} & = \Delta \overline{w}_{J_p} & + B_{J_p}^- \Delta w_{J_{p-1}} & + B_{J_p}^+ \Delta w_{j_{p+1}} \\
 & \ldots & & \\
\Delta w_{J_{P-1}} & = \Delta \overline{w}_{J_{P-1}} & + B_{J_{P-1}}^- \Delta w_{J_{P-2}} & + B_{J_{P-1}}^+ \Delta w_{j_P} \\
\Delta w_{j_P} & = \Delta \overline{w}_{j_P} & + B_{j_P}^- \Delta w_{J_{p-1}} &
\end{array}
\right. \tag{3.8}
$$

This reduced system has $2(P-1)$ equations and $2(P-1)$ unknowns. It can be solved with the Wang's direct algorithm [Wan81]. But this algorithm is only efficient when the cell number in each block is much larger than the block number.

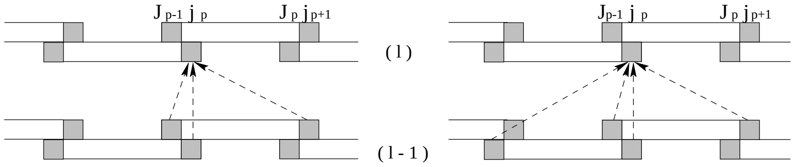
The iterative process

Taking advantage of the particular structure of our problem, we have easily formed the reduced system. Thus we can completely relax system (3.8) by introducing the following iterative algorithm :

$$
\begin{array}{|l}
\text{Initialization of } \Delta w_{j_p}^{(0)} = 0 \text{ and } \Delta w_{J_p}^{(0)} = 0 \\
\textbf{DO } l = 1, L \\
\quad \Delta w_{j_p}^{(l)} = \Delta \overline{w}_{j_p} + B_{j_p}^{-} \Delta w_{J_{p-1}}^{(l-1)} + B_{j_p}^{+} \Delta w_{j_{p+1}}^{(l-1)} \\
\quad \Delta w_{J_p}^{(l)} = \Delta \overline{w}_{J_p} + B_{J_p}^{-} \Delta w_{J_{p-1}}^{(l-1)} + B_{J_p}^{+} \Delta w_{j_{p+1}}^{(l-1)} \\
\textbf{ENDDO} \\
\Delta w_{j_p - 1} = \Delta w_{J_{p-1}}^{(L)},\ \Delta w_{J_p + 1} = \Delta w_{j_{p+1}}^{(L)}
\end{array}
$$

This algorithm consumes very little CPU time because an iteration represents only four matrix-vector multiplications independently computed in each block. Nevertheless, its convergence rate is proportional to the overlapping size. On Fig. 2 -left, we represent by an arrow the geometrical dependency of the ghost-cells from iteration $(l-1)$ to iteration (l). Since it is equivalent to a one-cell overlapping Schwarz algorithm, this algorithm requires many iterations to reach a perfect accuracy.

Figure 2 Left : Slow iterative algorithm. Right : Fast iterative algorithm.



In order to dramatically increase this low efficiency, we suggest a second algorithm. We now relax system (3.8) according to the following relation :

$$
\begin{array}{llll}
\Delta w_{j_p}^{(l)} & = \Delta \overline{w}_{j_p} & + B_{j_p}^{-} \Delta w_{J_{p-1}}^{(l)} & + B_{j_p}^{+} \Delta w_{j_{p+1}}^{(l-1)} \\
\Delta w_{J_p}^{(l)} & = \Delta \overline{w}_{J_p} & + B_{J_p}^{-} \Delta w_{J_{p-1}}^{(l-1)} & + B_{J_p}^{+} \Delta w_{j_{P+1}}^{(l)}
\end{array}
\quad , \qquad p \in \{1, \ldots, P\}
$$

We find our new implicit interface algorithm :

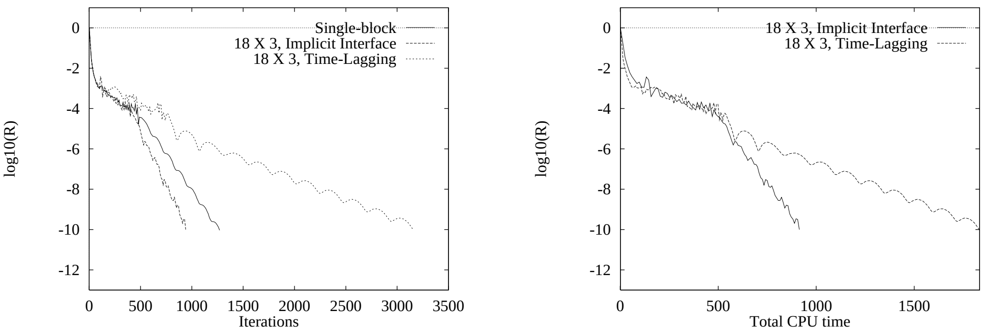
$$
\begin{array}{|l}
\text{Initialization of } \Delta w_{j_p}^{(0)} = 0 \text{ and } \Delta w_{J_p}^{(0)} = 0 \\
\textbf{DO } l = 1, L \\
\quad \Delta w_{j_p}^{(l)} = \Delta \overline{\overline{w}}_{j_p} + \overline{B}_{j_p}^{-} \Delta w_{J_{p-2}}^{(l-1)} + \overline{B}_{j_p}^{+} \Delta w_{j_{p+1}}^{(l-1)} \\
\quad \Delta w_{J_p}^{(l)} = \Delta \overline{\overline{w}}_{J_p} + \overline{B}_{J_p}^{-} \Delta w_{J_{p-1}}^{(l-1)} + \overline{B}_{J_p}^{+} \Delta w_{j_{p+2}}^{(l-1)} \\
\textbf{ENDDO} \\
\Delta w_{j_p - 1} = \Delta w_{J_{p-1}}^{(L)},\ \Delta w_{J_p + 1} = \Delta w_{j_{p+1}}^{(L)}
\end{array}
$$

where the new influence matrices are computed in parallel before the first iteration as :

$$
\begin{array}{|ll}
\Delta \overline{\overline{w}}_{j_p} & = M_{j_p} \left(\Delta \overline{w}_{j_p} + B_{j_p}^{-} \Delta \overline{w}_{J_{p-1}} \right) \\
\overline{B}_{j_p}^{-} & = M_{j_p} B_{j_p}^{-} B_{J_{p-1}}^{-} \\
\overline{B}_{j_p}^{+} & = M_{j_p} B_{j_p}^{+}
\end{array}
\qquad
\begin{array}{|ll}
\Delta \overline{\overline{w}}_{J_p} & = M_{J_p} \left(\Delta \overline{w}_{J_p} + B_{J_p}^{+} \Delta \overline{w}_{j_{p+1}} \right) \\
\overline{B}_{J_p}^{-} & = M_{J_p} B_{J_p}^{-} \\
\overline{B}_{J_p}^{+} & = M_{J_p} B_{J_p}^{+} B_{j_{p+1}}^{+}
\end{array}
$$

$$
\text{with } M_{j_p} = \left(I - B_{j_p}^{-} B_{J_{p-1}}^{+} \right)^{-1} \qquad \text{with } M_{J_p} = \left(I - B_{J_p}^{+} B_{j_{p+1}}^{-} \right)^{-1}
$$

Figure 3 Inviscid transonic flow ($M_\infty = 0.85$, $\alpha = 1^o$). Convergence history in terms of iterations and CPU time.



This iterative process is now equivalent to a one-block overlapping Schwarz algorithm (see Fig. 2 -right). The convergence rate of the latter algorithm (called the fast algorithm) is much greater than the convergence rate of the previous one because the overlapping distance has been increased from one cell to one block. It offsets the increasing cost due to the computation of the new influence matrices.

4 Numerical Applications

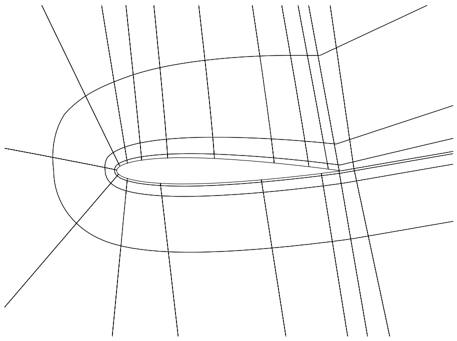
For the numerical applications, we use the centered implicit dissipative Lerat scheme [Ler83]. This inviscid method is second-order accurate, unconditionally stable and dissipative in the sense of Kreiss. For viscous flow computations, we use an extension due to Hollanders *et al.* [HLP83]. The implicit system is solved using a line Jacobi relaxation procedure so that the computation of the two-dimensional scheme is equivalent to two computations of the one-dimensional scheme (2.2). A local time step is used so that the CFL number is uniform all over the mesh. The computations start from an uniform flow. All computations have been made without artificial viscosity.

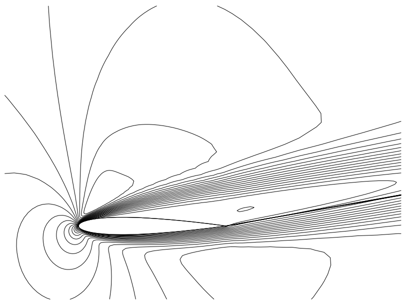
The present multiblock implicit solver has been implemented on a Cray J916. Its efficiency is assessed by computing two external steady flows around a NACA 0012 airfoil and comparing the convergence histories of the single and multiblock methods using either the time-lagging interface condition or the new implicit interface condition with only two iterations of the fast iterative algorithm. The comparison is made in terms of iterations and also of CPU times for the multiblock solvers.

Inviscid transonic flow

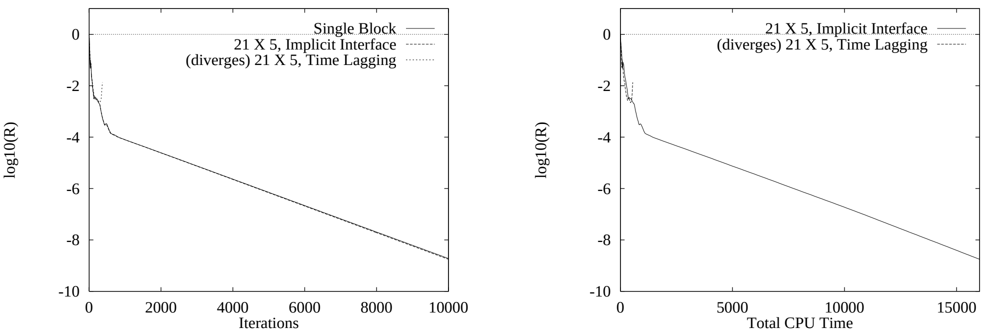
The first application is an inviscid transonic flow at freestream Mach number $M_\infty = 0.85$ with a 1^o angle of attack. The calculation is run on a 188×24 C-mesh which is partitioned into $18 \times 3 = 54$ blocks of approximately 10×8 cells each, the CFL number is equal to 20.

Figure 4 Viscous subsonic flow ($M_\infty = 0.8$, $\alpha = 10^o$). Convergence history in terms of iterations and CPU time.





Mesh partitioning in 21×5 blocks. Mach-number field ($\Delta M = 0.05$).



The convergence histories are shown on Fig. 1.3. In terms of time iterations, all the methods behave similarly at the beginning of the transient phase, but after 500 iterations the implicit interface condition is much more efficient than the time lagging condition. The number of iterations to reach a L_2 - residual of 10^{-10} is reduced by a factor 3.3. The CPU-time per iteration depends on the coding and the load balancing between the explicit and implicit stage of the scheme. The extra-cost per iteration (over time-lagging) decreases when the flow modelization is improved and varies from 65 % for the present Euler computation to 5% for a turbulent Navier-Stokes computation. Therefore, in the present Euler test-case, the total CPU-time to reach the 10^{-10} residual is reduced by a factor 2 (see Fig. 1.3).

Furthermore, Fig. 1.3-left shows that the implicit multiblock solver converges faster than the single block solver. This surprising result is due to a better treatment of the cut line issuing from the trailing edge. Actually the matching is time lagged in the single block computation while it is accurately implicited in the proposed multiblock treatment.

Viscous subsonic flow

The second application is a viscous subsonic laminar flow ($Re = 500$) at free stream Mach number $M_\infty = 0.8$ with a 10^o angle of attack. The C-mesh is composed of 254×58 cells. It is partitioned into $21 \times 5 = 105$ blocks of roughly 12×12 cells (see Fig. 4). The CFL number is equal to 35. On the Mach number field, we observe a large recirculation region.

The multiblock solver using the time-lagging interface condition diverges while the use of the new implicit interface condition yields the single-block solver efficiency. Note that even if there are numerous blocks into the boundary layer, no numerical problem has been found with for the multiblock calculation.

5 Conclusion

A new implicit interface condition has been developed for the solution of the Euler and the Navier-Stokes equations on multiblock structured meshes. This method is perfectly parallelizable and as accurate as an exact implicit interface condition.

The present method reduces the CPU cost of a steady flow computation with respect to a block-Jacobi solver or simply allows the convergence when the latter is not able to reach the steady-state.

REFERENCES

[Bru91] Brugnano L. (1991) A parallel solver for tridiagonal linear systems for distributed memory parallel computers. *Par. Comput.* 5: 1017–1023.

[HLP83] Hollanders H., Lerat A., and Peyret R. (1983) Three-dimensional calculation of transonic viscous flows by an implicit method. *AIAA P.* 83-1953. (1985) *AIAA J.* 23 : 1670–1678.

[Jen94] Jenssen C. B. (1994) Implicit multiblock Euler and Navier-Stokes calculations. *AIAA J.* 32(9): 1808–1814.

[Ler83] Lerat A. (1983) Implicit methods of second order accuracy for the Euler equations. *AIAA P.* 83-1925. (1985) *AIAA J.* 23: 33–40.

[Lio88] Lions P. L. (1988) On the Schwarz alternating method I. In Glowinski R., Golub G. H., Meurant G. A., and Périaux J. (eds) *Proc. First Int. Conf. on Domain Decomposition Meths.*, pages 1–42. SIAM, Philadelphia.

[LW96] Lerat A. and Wu Z. N. (1996) Stable conservative multidomain treatments for implicit Euler solvers. *J. Comp. Phys.* 123: 45–64.

[Rai86] Rai M. M. (1986) An implicit, conservative, zonal-boundary scheme for Euler equation calculations. *Comp. Fluids* 14(3): 295–319.

[Wan81] Wang H. (1981) A parallel method for tridiagonal equations. *ACM Trans. Math. Software* 7: 170–183.