Simulation of transonic flows using quad-core OpenMP Euler, flux modified transonic small disturbance, and Fluent codes

E. Ly[1] D. Norrison[2] A. R. Barrett[3]

(Received 12 November 2009; revised 21 April 2010)

Abstract

This article investigates what performance can be achieved when executing an aerodynamic code on a quad-core based personal computer with an OpenMP compiler in a Windows environment. Euler equations are solved in two dimensions for simplicity, to predict flow field around an aerofoil on an O-type boundary fitted structured grid. The grid is generated by solving a system of Poisson's equations utilising an efficient method of false transients, coupled with an approximate factorisation technique and variable time steps cycling process. Comparison between the OpenMP Euler, flux modified transonic small disturbance and Fluent results for transonic flow fields, with shock waves, around a NACA0012 aerofoil is presented.

http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/2658 gives this article, © Austral. Mathematical Soc. 2010. Published April 30, 2010. ISSN 1446-8735. (Print two pages per sheet of paper.)

Contents

1 Introduction

Inadequate and unaffordable computer power has plagued computational fluid dynamics (CFD) research and applications since its inception. Consequently, the claim made by Harvard Lomax [12] in the late 1960's that CFD would replace wind tunnel experiments has not yet materialised. Instead, problems that required investigation were simplified (either mathematically or physically), so that reasonable solutions were obtained within an acceptable turnaround time with present computing platforms. Often, the only way to obtain results in a timely manner, for demanding high level aerodynamics solvers based on Euler or Navier–Stokes equations, required employing a very expensive supercomputer or cluster computer system. As such, researchers predominantly focused on researching large scale computations and associated issues using such systems [1, 5]. Recent developments offer a tantalising insight into new computation accelerating technologies for ordinary desktop personal computers (PCs). These include using relatively inexpensive high-end graphics card based solutions with AMD/ATI's Close-To-The-Metal and Nvidia's Compute Unified Device Architecture, which offer performance of up to 500 Gflops per card for single precision calculations.

Here we ask what sort of performance can a regular PC offer for less demanding CFD applications, particularly given that multiple processing cores are now be accommodated in a single-socket rather than resorting to the multi-socket architecture of the past? To answer this question, a finite difference based scheme, implementing the Van Leer flux blending [14], was developed to solve the two dimensional Euler equations, in double precision accuracy, on an O-type structured grid system. The grid generation code implements an efficient method of false transients, coupled with an approximate factorisation (AF) technique and a variable time step cycling process with repeated end points. At closure, comparison between the computed results obtained from the OpenMP Euler code, flux modified transonic small disturbance (TSD) code and commercial CFD code Fluent, for transonic flow fields around a NACA 0012 aerofoil is presented.

2 Numerical solution procedure

2.1 Flux blending technique based Euler solver

The first step in solving the Euler equations involves establishing a smooth boundary fitted grid around a body of interest, which in this case is an aerofoil. The cartesian coordinates in the physical domain, $\mathbf{r} = (x, z)$, are mapped to general curvilinear coordinates in the computational domain, $\boldsymbol{\vartheta} = (\xi, \zeta)$, via a relation $\boldsymbol{\vartheta} = \mathbf{r}$ (quantities in boldface represent vectors). The mapping is constructed by specifying desired grid points on the boundaries, with the interior point distribution determined through the solution of a system of Poisson's equations [10]. The grid equations are solved by the method of false transients coupled with an AF technique [9] and a variable time step cycling process [8]. The latter process is incorporated to further enhance the convergence rate of the grid generation process, and overall the process is efficient and robust.

The adiabatic flow field around the aerofoil, without body forces, is assumed to be governed by the unsteady Euler equations, written in a strong conservation law form with flux variables as the dependent variables [4]. The equations were non-dimensionalised with free stream fluid density and speed, and aerofoil chord length as the characteristic length scale, yielding

$$\frac{\partial \widehat{\mathbf{Q}}}{\partial \tau} + \frac{\partial \widehat{\mathbf{E}}}{\partial \xi} + \frac{\partial \widehat{\mathbf{G}}}{\partial \zeta} = \mathbf{0}\,, \tag{1}$$

where $J\widehat{\mathbf{Q}} = \mathbf{Q}$, $J\widehat{\mathbf{E}} = \mathbf{E}\xi_x + \mathbf{G}\xi_z$, $J\widehat{\mathbf{G}} = \mathbf{E}\zeta_x + \mathbf{G}\zeta_z$, J is the Jacobian of transformation [10],

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho w \\ \rho e_t \end{bmatrix}, \qquad \mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u w \\ u(\rho e_t + p) \end{bmatrix} \quad \text{and} \quad \mathbf{G} = \begin{bmatrix} \rho w \\ \rho w u \\ \rho w^2 + p \\ w(\rho e_t + p) \end{bmatrix}. \tag{2}$$

Here $\mathbf{Q}_\tau$ represents $\partial\mathbf{Q}/\partial\tau$, ρ is the fluid density, (u, w) are the cartesian fluid velocity components in the x- and z-direction, respectively, e_t is the total energy per unit mass and p is the fluid pressure.

The flux vectors are split according to $\widehat{\mathbf{E}} = \widehat{\mathbf{E}}^+ + \widehat{\mathbf{E}}^-$ and $\widehat{\mathbf{G}} = \widehat{\mathbf{G}}^+ + \widehat{\mathbf{G}}^-$. In the finite difference scheme, forward flux terms (plus superscript) are discretised using a backward difference rule, and a forward difference rule for backward flux terms (minus). When compared to centred schemes this brings some advantages, such as superior dissipation and dispersive properties, and up to twice the stability bound [13]. Van Leer flux blending [14] is employed to solve the equation, since it overcomes spurious oscillations (generated by the Steger and Warming flux splitting [6] technique) around the sonic transition regions and stagnation points.

Equation (1) is hyperbolic in time, and for steady flow simulations the solution process is marched in time until a steady state solution is obtained.

For simplicity, and since explicit schemes are well suited to parallel execution, a second order accurate (in both space and time) flux splitting version of MacCormack's scheme [4] is utilised. MacCormack's scheme involves a predictor-corrector sequence at each time level. In the predictor step, the time derivative is approximated by a first order forward time difference rule while all spatial derivatives are approximated by first order backward and forward difference rules as appropriate. The corrector step is more involved, and was presented by Steger and Warming [13]. The MacCormack scheme has been shown to be conditionally stable [4, 13] provided that the time step is not too large. A time step in the order of a tenth of millisecond seemed to provide a numerically stable solution with the boundary conditions used.

2.2 Flux modified transonic small disturbance solver

In the TSD code, the isentropic and inviscid flow over a thin aerofoil is assumed to be governed by the general frequency unsteady TSD equation [9],

$$\frac{\partial}{\partial t}\left[M_\infty^2\phi_t + 2M_\infty^2\phi_x\right] + \frac{\partial}{\partial x}\left[\tfrac{1}{2}M_\infty^2(\gamma+1)W^2\right] - \frac{\partial}{\partial z}\phi_z = 0\,, \tag{3}$$

where

$$W = \frac{1-M_\infty^2}{M_\infty^2\,(\gamma+1)} - \frac{\partial\phi}{\partial x}\,, \tag{4}$$

$\phi(x,z,t)$ is the reduced potential, M_∞ is the free stream Mach number and γ is the ratio of specific heats (about 1.4 for ambient air). Equation (3) is locally of elliptic/hyperbolic type representing local subsonic/supersonic flow when W is positive/negative, and its solution contains discontinuous jumps that approximate shock waves. Nonreflecting far field boundary conditions derived from the theory of wave propagation are imposed, and serve to simulate the disturbances that propagate outward from the aerofoil to infinity. This allows the solution to propagate through the artificial computational boundaries as if there are no boundaries present. This allows the far field boundaries to be moved closer to the aerofoil, and offer greater freedom in tradeoffs among grid density, accuracy and computational cost. Any shock wave that exists in the flow field must satisfy the shock jump condition derived from (3).

Rotational effects become significant when strong shock waves exist in the flow field, since vorticity is generated due to the entropy changes along the shock. Such effects are excluded in the conventional TSD theory [3, 8], because of the irrotationality assumption necessary for the existence of a velocity potential. Hence, inclusion of the shock generated entropy and vorticity effects is necessary when modelling such strong shock waves. To include the entropy effect, the streamwise flux in (3) is modified to

$$W = \frac{\psi^4 - 1}{2\psi} - \frac{\psi(1+\psi^2)\varphi_x}{1+\psi^2+\varphi_x} \quad \text{and} \quad \psi^4 = \frac{2+(\gamma-1)M_\infty^2}{(\gamma+1)M_\infty^2}\,. \tag{5}$$

To include vorticity effects, the velocity vector is treated as a sum of potential and rotational components, and the rotational component assumed to exist only in the region downstream of the shock wave. Since entropy is constant for steady flow, and imposing that the shock curvature is negligibly small, the fluid speed of the grid points behind the shock wave is modified to include an entropy jump, experienced by a fluid particle as it passes through the shock.

The numerical solution procedure involves applying the method of false transients coupled with an AF technique [3, 8, 9] to solve for the reduced potential. In the method of false transients for steady state solution, the time derivative in Equation (3) is replaced by an artificial time derivative written in a Padé form, which is approximated by a general time difference rule in the finite difference scheme. The spatial terms of Equation (3) are approximately factorised, and the time step sizes are cycled in a geometric manner with repeated end points to enhance the convergence rate of the scheme. The first ξ-derivative and all ζ-derivatives are differenced using standard second order accurate upwind and central differencing, respectively, while second order accurate Engquist–Osher type-dependent differencing [2] is applied to the second ξ-derivative. As the flow changes from subsonic to supersonic, Engquist–Osher operators smoothly change from central differencing (elliptic region) to upwind differencing (hyperbolic region). This ensures a smooth transition from subsonic to supersonic flow. Thence, entropy violating decompression shocks will not develop. As the flow changes from supersonic to subsonic, Engquist–Osher operators change to an appropriate shock point operator [11], implementing a shock capturing technique.

3 Simulation examples

In terms of software, two options exist for parallel programming, notably Open Multi-Processing (OpenMP) and message passing solutions such as the Message Passing Interface (MPI). The latter is used with cluster computer systems, where data is transferred between nodes. While it can be used for multiple processors on a single node, the programming effort required is much higher than for OpenMP, which is designed to share memory on an individual node. For this reason, OpenMP is used here.

In the present Euler code, the compiler directive,

```
!$OMP PARALLEL DEFAULT(PRIVATE) SHARED(Qhat,...)
```

marked the start of the parallel section inside the iteration loop. The thread numbers were limited with `!$OMP NUM_THREADS()`. The directives `!$OMP DO` and `!$OMP END DO` were placed at the start and end of each of the predictor and corrector loop blocks, where intensive computations took place to calculate the flow variables for each grid point at each time level. The parallel section closed after the second loop with the `!$OMP END PARALLEL` directive.

For investigation purposes, flow fields around a NACA 0012 aerofoil were computed on an O-type structured grid with coarse (101 streamwise points by 50 radial points), medium (201×101) and fine (401×201) meshes. The left plot of Figure 1 presents the comparison of the present OpenMP Euler result with the well known AGARD's (Advisory Group for Aerospace Research and Development) benchmark result by Lock [7] for the aerofoil at two de-

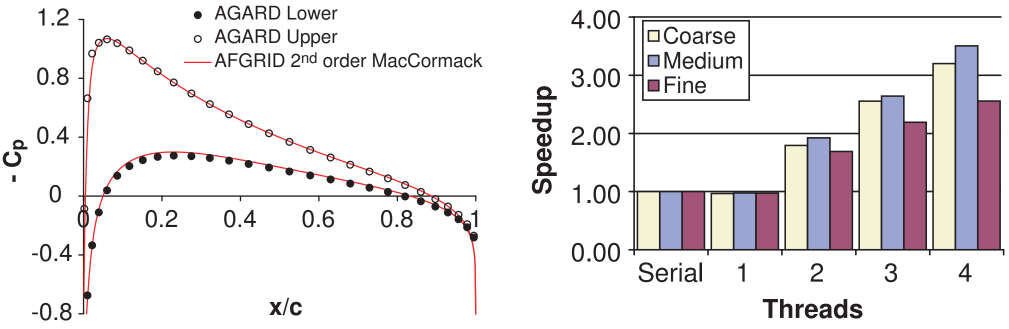
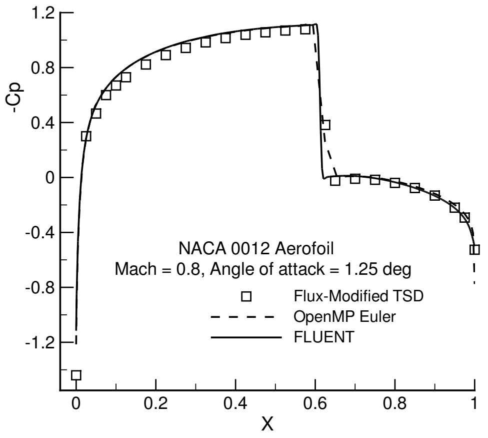


Figure 1: (left) Steady pressure coefficient distributions; (right) Computing speedups of the OpenMP Euler code for different thread modes.

gree angle of attack and at free stream Mach number of 0.63 (Reynolds number of 14.67 million). The Euler computation was performed on a personal computer consisting of an Intel 2.4 GHz Core 2 Quad Q6600 processor with 8 Mb L2 cache and 1,066 MHz front side bus, Gigabyte motherboard with an Intel G965 chipset and Corsair 2x1 Gb DDR2 800 MHz dual-channel memory, running Microsoft Windows XP Professional Service Pack 2. The pressure distributions for both the upperside and lowerside of the aerofoil are in excellent agreement. Adopting the serial code computation time as the benchmark, the performance results illustrated by the right plot of Figure 1 show that a substantial reduction in execution time can be achieved with a multicore PC. However, with only one thread the speedup is reduced by 2.5% to 3.3% due to the overhead associated with enabling OpenMP.

As the number of threads increased, the speedup associated with the quad-core processor varied approximately linearly with the thread number. Furthermore, a larger speedup is observed with the coarse and medium grids when compared to the fine grid. In regards to the coarse and medium grid results, a maximum speedup of 350% (which is 12.5% below the ideal value of 400%, most likely due to the necessary communication overheads) was achieved using four threads. The disparity between the two smaller grids is attributed to the OpenMP overhead to computation ratio per iteration



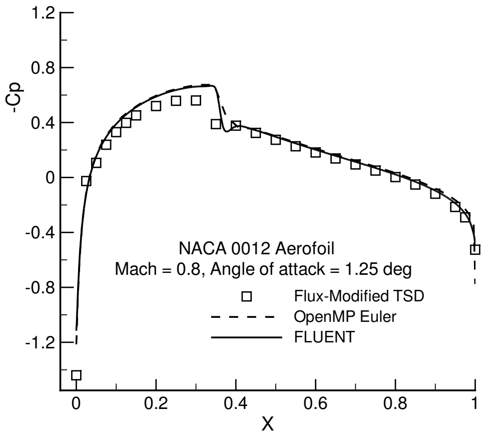


Figure 2: Steady pressure distributions: (left) upperside; (right) lowerside.

being higher for the coarse grid than for the medium grid. In other words, each outer loop iteration performs computations for all of the grid points in the streamwise direction along a given radial level. Since the medium grid contains more nodes than the coarse grid, the computation work done per iteration is greater for the medium grid while the OpenMP overhead is the same for each grids. Therefore, the proportion of time per iterations pent by OpenMP to distribute the workload is larger for the coarse grid compared to the medium grid, and this results in a slight reduction in the speedup for the coarse grid. As for the fine grid, the speedup rose to a maximum of 256%. This significant reduction in performance, compared to the coarser grids, is suspected to be caused by inadequate front side bus bandwidth. The coarse and medium grids occupied approximately 1.5 and 6.1 Mb of memory, respectively, which the faster cache memory could accommodate. Thus, the computations were limited by processor speed, whereas the fine grid required about 24 Mb, which could only fit in the slower main memory whose accessibility restricted the computation speed. The results for the smaller grids demonstrate that the new multicore architecture offers similar scaling performance to older shared memory multiprocessor architecture [5].

The steady state pressure distributions computed by the OpenMP Euler code, flux modifed TSD code and those obtained from the commercial CFD package Fluent, for a NACA 0012 aerofoil at 1.25 degree angle of attack and at free stream Mach number of 0.8, are compared in Figure 2. Fluent is a general purpose CFD code based on the finite volume method on a collocated grid. The present code employed four threads and medium grid for the computation, and the solution was obtained within an acceptable turnaround times (about ten minutes) on a descent desktop PC. While the TSD code employed an algebraic grid system with $18,000$ grid points, and Fluent employed a structured C-type grid system with $80,000$ grid points, generated using a commercial meshing package GAMBIT. GAMBIT is a geometric modelling and grid generation tool to automatically mesh surfaces and volumes while allowing the user to control the mesh through the use of sizing functions and boundary layer meshing. All grids involved clustering of grid lines toward the aerofoil in region adjacent to the aerofoil. All codes generated a shock wave as shown in Figure 2, which is represented by the discontinuous jump in the pressure distribution. The shock profile spreads over a maximum of three grid points (two computational cells) leading to an excellent agreement in terms of shock position and strength, except for the lowerside where the TSD code generated a slightly weaker shock. This is acceptable, since the TSD code solved the general frequency TSD equation, which is derived on the assumption of potential flow, generates a weaker shock wave. Even though the streamwise flux of the TSD equation was modified to include the shock generated entropy and vorticity effects, these modifications overall do not represent the exact effects that would otherwise produced by the Euler solution. Figure 3 presents a contour plot of steady pressure coefficient generated by the present Euler code, clearly showing the formation of the shock waves. Since the present code solves a system of flow equations, it requires more memory during computation as compared to the TSD solver (where it solves only a single potential equation), but it is not restricted to thin body application as it is not based on the potential theory.

As computational power increases, more versatile but less computationally

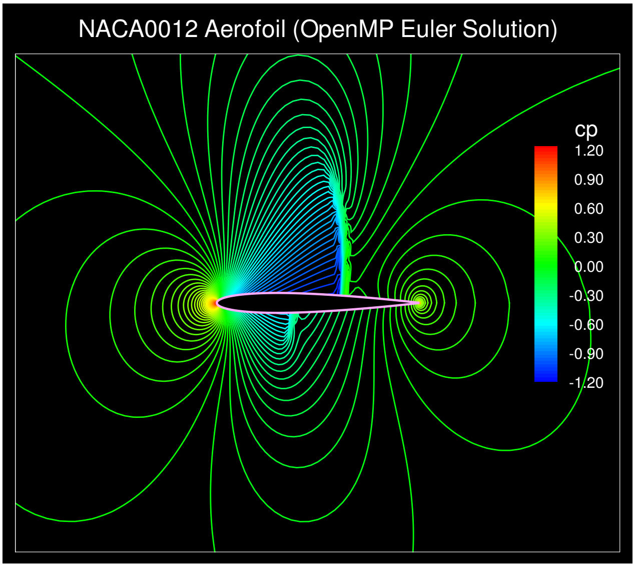


Figure 3: Contour plot of steady pressure coefficient.

efficient codes will be adopted. For example, with the processing speed of a quad-core PC, an Euler solver could replace solvers that solve the subsonic and supersonic small disturbance equations, full potential equation and general frequency TSD equation, which are limited to the potential flow regime, supercritical flows with weak embedded shock waves, thin aerofoils (except for the full potential equation) and small angle of attacks.

4 Concluding remarks

The scaling performance of an Intel quad-core processor using OpenMP to accelerate a two dimensional CFD problem on a PC was studied. With coarse grids, an excellent speedup of 350% was achieved, while for fine grids the speedup was 256%. In future work, processors with larger core numbers, as well as graphics card based solutions, may be investigated to establish their associated speedups in newer high performance PCs. Further reduction in turnaround times of the present code will make it attractive in aeroelasticity analysis work (computational aeroelasticity), where a large number of simulations need to be executed; improving the accuracy of aerodynamic computations (as compared with the modified TSD) in the preliminary design of aircraft wings; or may even be used as an educational tool for students studying CFD, computational aerodynamics or computational mathematics.

References

[1] Bailey, D., Barszcz, E., Barton, J., Browning, D., Carter, R., Dagum, L., Fatoohi, R., Fineberg, S., Frederickson, P., Lasinski, T., Schreiber, R., Simon, H., Venkatakrishnan, V., and Weeratunga, S., *The NAS Parallel Benchmarks*, Contractor Report 203186, NASA, USA, 1994. C156

[2] Engquist, B., and Osher, S., Stable and Entropy Satisfying Approximations for Transonic Flow Calculations, *Mathematics of Computation*, **34**, 149, January 1980, pp. 45–75. C160

[3] Gear, J., Ly, E., and Phillips, N. J. T., A Time Marching, Type-Dependent, Finite Difference Algorithm for the Modified Transonic Small Disturbance Equation, *Proceedings of the 21st Congress of the International Council of the Aeronautical Sciences (ICAS98)*, ICAS and AIAA, Melbourne, Australia, 1998, Paper A98-31513. C160

[4] Hirsch, C., *Numerical Computation of Internal and External Flows: Computational Methods for Inviscid and Viscous Flows*, Volume 2, John Wiley and Sons, Great Britain, 1992. C158, C159

[5] Jin, H., Frumkin, M., and Yan, J., *The OpenMP Implementation of NAS Parallel Benchmarks and its Performance*, Technical Report NAS-99-011, NASA, Moffett Field, USA, 1999. C156, C163

[6] Liu, Y., and Vinokur, M., *Nonequilibrium Flow Computations 1: An Analysis of Numerical Formulations of Conservation Laws*, Contractor Report 177489, NASA, California, USA, 1988. C158

[7] Lock, R. C., *Test Cases for Numerical Methods in Two-Dimensional Transonic Flows*, Report Number 575, AGARD, 1970. C161

[8] Ly, E., Improved Approximate Factorisation Algorithm for the Steady Subsonic and Transonic Flow over an Aircraft Wing, *Proceedings of the 21st Congress of the International Council of the Aeronautical Sciences (ICAS98)*, ICAS and AIAA, Melbourne, Australia, 1998, Paper Number A98-31699. C157, C160

[9] Ly, E., and Nakamichi, J., Time-Linearised Transonic Computations Including Entropy, Vorticity and Shock Wave Motion Effects, *The Aeronautical Journal*, November 2003, pp. 687–695. C157, C159, C160

[10] Ly, E., and Norrison, D., Automatic Elliptic Grid Generation by an Approximate Factorisation Algorithm, *ANZIAM Journal*, **48** (CTAC2006), pp. C188–C202, July 2007. C157, C158

[11] Murman, E., Analysis of Embedded Shock Waves Calculated by Relaxation Methods, *AIAA Journal*, **12**, 5, May 1974, pp. 626–633. C161

[12] Pulliam, T., Kutler, P., and Rossow, V., *Harvard Lomax: His Quiet Legacy to Computational Fluid Dynamics*, 14th AIAA Computational Fluid Dynamics Conference, Norfolk, VA, June 1999. C156

[13] Steger, J.L., and Warming, R. F., *Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite Difference Methods*, Technical Memorandum TM-78605, NASA, California, USA, 1979. C158, C159

[14] Van Leer, B., Flux-Vector Splitting for the Euler Equations, *Proceedings of the 8th International Conference on Numerical Methods in Fluid Dynamics*, Aachen, West Germany, 1982. C157, C158

Author addresses

1. **E. Ly**, School of Mathematical and Geospatial Sciences (SMGS), College of Science, Engineering and Health (SEH), RMIT University, Melbourne, Victoria 3001, AUSTRALIA.
`mailto:eddie.ly@rmit.edu.au`
2. **D. Norrison**, School of Mathematical and Geospatial Sciences (SMGS), College of Science, Engineering and Health (SEH), RMIT University, Melbourne, Victoria 3001, AUSTRALIA.
3. **A. R. Barrett**, School of Mathematical and Geospatial Sciences (SMGS), College of Science, Engineering and Health (SEH), RMIT University, Melbourne, Victoria 3001, AUSTRALIA.