New cooperative parallel strategy for massive CFD computations of aerodynamic data on multiprocessors

S. Peigin[a], B.Epstein[b]

[a] Israel Aircraft Industries, Israel

[b]The Academic College of Tel-Aviv Yaffo, Israel

1. INTRODUCTION

In practical aerodynamics, massive computations of aerodynamic data, especially the computations of aerodynamic force and moments, are required.

Aerodynamic data-bases of this kind include drag/lift polars at different free-stream Mach numbers and Reynolds numbers, Mach drag rise curves which are indicative of MDD (Mach Drag Divergence), lift vs. angle of attack graphs, C_M (pitching moment) vs. C_L (lift coefficient) and other moment/angle of attack/lift/drag combinations.

A usual way of attaining the above aerodynamic information is to perform massive CFD computations since the wind-tunnel tests are not able of supplying a quick response to the needs of the aircraft industry. Besides, the wind-tunnel tests are prohibitively expensive (in terms of both money and time), especially in the stage of the configuration design where a considerable amount of different configurations must be tested as fast as possible.

However even the computer-aided construction of parametrically dense aerodynamic data-bases is highly time-consuming (though of course it is much cheaper than the corresponding wind-tunnel tests).

It is especially true where geometrically complicated configurations are treated such as a complete aircraft or even simpler wing-body configurations.

For example, a single high-accuracy simulation of a complete aircraft may take a number of days (or even weeks for a high-accuracy computation) on presently available serial computers. To attain a comprehensive aerodynamic data-base, hundreds (or even thousands) of such simulations are needed.

Two approaches are presently in use in industrial environment intended to diminish the overall computation time.

With the first approach, the computations are performed by means of low-accuracy CFD codes which are based on incomplete gas-dynamic models (such as panel codes). This may give reasonably accurate results for lift evaluations in the subsonic flow regime but fails in more complicated flight conditions even in the case of lift/angle of attack computations.

The computation of drag and pitching moment in a flow regime where considerable viscous effects are present and/or in the presence of shock waves requires high-accuracy full Navier-Stokes simulations which (as already explained above) constitute a highly time-consuming task.

A more comprehensive approach is to make use of massive parallelization of CFD codes on distributed memory clusters. This may essentially decrease the computation time as the CFD parallelization allows for concurrent computation of a number of flight points. Still the overall computing time remains high in the following three practically crucial cases: 1) Construction of large-size aerodynamic data-bases, 2) Numerical testing of concurrent configurations and 3) Multi-objective aerodynamic design.

The present work makes use of usually neglected information which may be extracted from already computed (or partially computed) flow-fields. A sophisticated use of this (off-line and on-line) information may essentially reduce the number of iterations needed for convergence thus dramatically diminishing the overall time of multiple aerodynamic simulations.

The presented results show that the proposed technology may result in the parallel efficiency which is formally higher than 100%. This is of course due to the information exchange, infeasible in the case of serial runs.

Note, that though the proposed technology is especially efficient for the computation of large-size aerodynamic data-bases, it can be successfully employed for the computation of even relatively small data sets, such as a single lift/drag polar.

2. NEW COOPERATIVE STRATEGY FOR MASSIVE PARALLEL ITERATIVE COMPUTATIONS

In general, parallelization of any computational algorithm leads to distribution of total computational work among participating parallel processors and organization of message-passing between the processors. A crucial question raised by the parallel implementation of iterative algorithms is the efficiency of the data exchange between different processors.

It is widely accepted that the optimal strategy to reach the main goal of parallelization (to maximize parallel efficiency of the algorithm), is to minimize the CPU time related to the process of data transfer. Based on this point of view, we take into account the above CPU time as a loss only.

From a more general (more "philosophical") point of view, the incidence of the data transfer between processors it twofold. On one hand this results in a loss (increasing the total CPU time). But on the other hand it also results in a gain, since individual parallel processes gain additional information, thus improving the rate of convergence of the iterative algorithm, or providing a more accurate solution. Consequently, at least theoretically, the gain may exceed the loss. Of course, the most challenging question is how to recognize which information should be transferred in order that the gain surpasses the loss.

Apparently there exists no general answer to this question. In this paper, the problem is being solved in the context of massive iterative computations for the aerodynamic analysis and design.

The first example is the optimization of aerodynamic shapes by means of Genetic Algorithms (GAs). In this case, it is necessary to accelerate the optimum search. It appeared that this goal may be achieved through exchanging information related to the best individuals of parallel populations which essentially improves the convergence rate of GA iterative process.

The second example is the CFD analysis intended to the computations of large-size aerodynamic data. The computer aided aerodynamic flow analysis is performed by numerical solution of flow equations (which represent a non-linear Partial Differential Equations (PDE) system). The equations are usually solved by an iterative numerical algorithm which may involve (for high accuracy solutions) hundreds, and sometimes thousands, iterations. Thus the convergence acceleration may result in essential gain in the overall time performance of the algorithm.

It was found that in this case, the relevant exchange information consists in the (possibly partial) flow-field data related to a limited subset of the considered solutions. It is important to note that the dimension of this subfamily is essentially lower than the overall dimension of the whole parametric family.

3. IMPLEMENTATION OF THE COOPERATIVE STRATEGY FOR PARALLEL CFD COMPUTATIONS

3.1. Parallel GAs search for the aerodynamic shape optimization

The basic idea behind Genetic Algorithms is to mathematically imitate the evolution process of nature. They are semi-stochastic optimization methods that are conveniently presented using the metaphor of natural evolution.

A new population of individuals is generated using variations of "genetic" operators: selection, crossover and mutation. Parents are chosen by selection, and new offsprings are produced with crossover and mutation. All these operations include randomness.

In the optimization of aerodynamic shapes by means of GAs [1] the population represents a set of aerodynamic shapes.

In order to improve the optimal search and to accelerate the convergence to the optimum shape, the above cooperative strategy was implemented in the following way. The crucial question is what kind of information is to be exchanged. Fortunately, for GAs the situation is favourable to giving a positive constructive answer to this general question: by exchanging information related to the best individual of parallel populations, one can expect the convergence rate of GA iterative process to improve.

The parallel asynchronous GA algorithm involved the following main stages:

a) The initial population (based on random search) is independently obtained on each processor of the group.

b) Non-blocking data acquisition from other processors of the group. The received individuals are included in the current population. If the requested message has not arrived this step is skipped.

c) With a preassigned value of the exchange step L, the information related to the best individual in the generation is broadcast to the other processors of the group.

3.2. Parallel cooperative strategy for massive CFD computations of aerodynamic data-bases

First, it is necessary to explain why a straightforward CFD parallelization on distributed memory clusters is not sufficiently efficient where large-scale aerodynamic computations are performed.

Consider a parallelized CFD code aimed to solve the full Navier-Stokes equations. The full-scale computation of a complicated aircraft configuration is usually infeasible on available serial processors.

The code similar to NES [2] which exhibits a high parallel efficiency runs about 6 hours on 106 processors (see [3] - [4]). This means that an average aerodynamic user which is rarely eligible to more than 100-200 processors may get only 1-2 flight point computations in a time. A more economic way to perform these computations is to employ fewer processors (30-40) in a time but then, of course, the overall CPU time greatly increases.

Note that in the case of a popular commercial code FLUENT, the peak parallel performance is achieved (on a typical platform of HP J6000 PA 8600,550) on already 16 processors.

The computation time is almost proportional to the number of iterations (sometimes measured in Multigrid cycles). Due to the complexity of the problem involved, the number of these iterations (cycles) may be as great as 200-400 (depending on the flight conditions).

We propose a parallel cooperative technology which essentially decreases the number of cycles needed to get a sufficiently accurate solution. This is done by performing the computations in an order which allows making use of already performed computations on the one hand, and, by exchanging the information in the course of parallel runs in a sophisticated way, on the other hand.

The idea of the parallel algorithm may be illustrated as follows. Consider, for example, the problem of computing a number of lift/ drag polars for different Mach numbers. For example, 10 polars are needed, each of them containing 20 different angles of attack.

In the first stage, a very limited number of full-size Navier-Stokes calculations are performed. The number of these calculations may amount to 10-15 (of a total of 200 lift/drag curves). The flight points chosen parse the parametric space of Mach/ angle of attack by covering it in an approximately uniform way. (Of course, this may change according to specific flight conditions since harder flight points necessitate a denser covering).

Already in this stage, completed CFD runs may be employed as a starting point for the runs in queue. This can be done by initializing the initial flow-field by an available information taken from a different (than the current) flight point (in contrast to a standard computation which starts from an appropriate free-stream solution).

In this stage, a flexible amount of processors may be employed starting from a cluster of 15-20 processors. On the other hand, the computational volume of the stage makes it also efficient the use of large-size clusters including up to 1000 processors.

The next stage of computations deals with the rest of the tested flight points. The specific implementation depends on the number of parallel processors available. The flight points are distributed among the processors in such a way that the points which are close (in the metric of the parametric space of flight conditions) to those computed in the first stage, start immediately after the completion of the first stage.

If necessary, the second wave of computations start originating from the already existing solutions, the third group employs the solutions of the previous two groups and so on. The approach allows to essentially decrease the number of iterations (cycles) needed to get the converged solution and thus to decrease the overall computation time.

An additional important source of increasing the parallel efficiency and thus diminishing the overall CPU time is the inter-level data exchange. Specifically this means that the current solution is interpolated with more advanced solutions at the flight points close to the current one. For example, this may be done by weighting the solutions U_1 and U_2: $U_{new} = a \cdot U_1 + (1-a) \cdot U_2$, where the parameter $0 < a < 1$, in a way minimizing the current density residual of the Navier-Stokes equations.

This approach involves an insignificant amount of additional computations and a certain amount of additional message passing. The tests show that the approach can be highly efficient provided the frequency of exchange is tuned to the parallel configuration.

4. ANALYSIS OF RESULTS

4.1. Acceleration of GAs optimal search

The results presented in this section are related to the optimization of RAE2822 airfoil at fixed values of free-stream Mach and Reynolds at $C_L = 0.745$, $M = 0.75$. The flight conditions are those of a high transonic regime. At the considered values of target C_L, the initial flow is characterized by strong shock/boundary layer interaction.

It appeared that a parallelization of GA algorithms can essentially improve accuracy and robustness of the optimization search.

Thus we observe that the present approach of parallel cooperative implementation of a genetic algorithm provides an exemplary case of a situation where the gain due to exchange of information among processors improves the convergence rate of GA and surpasses the loss in CPU time related to the data exchange itself.

4.2. Massive parallel computations of aerodynamic data-bases

The results of this section are related to massive computations of aerodynamic data for VANTAGE wing-body configuration. The CFD computations were performed by a multiblock parallel full Navier-Stokes code NES [3]. The computational grids contained about 500000 grid points united in three Multigrid levels, each of the levels included 16 blocks.

We constructed the aerodynamic data bases corresponding to all the possible combinations of the following flight conditions:$M = 0.60 - 0.70$, $\alpha = -2.0° - 4.0°$, $Re = 3 \cdot 10^6$, $Re = 5 \cdot 10^6$. A total of 286 flight points were computed.

The conventional strategy consisted in the computation at all the angles of attack, listed above. In this case, each converged solution required 60 Multigrid cycles on the fine level. Thus achieved results serve us as the exact reference. Two versions of the cooperative parallel strategy were applied. In the first case, 30 Multigrid cycles were performed at $\alpha = -0.5°$, $\alpha = 1.5°$ and $\alpha = 3.0°$. In the next stage, the information on the flow-fields, obtained in the first stage was transferred to the processors responsible for the computation of the remaining angles of attack. Then, at each angle of attack, 30 Multigrid cycles were performed.

In the second case, the first stage was reduced to the 30 cycles computation at only one angle of attack $\alpha = 0.0°$. In the next stage, the information on this flow-field, was sent to the processors responsible for the computation of the remaining angles of attack. Then, once again, at each angle of attack, 30 Multigrid cycles were performed.

Since the computational cost of the interstage data transfer is negligible, the overall computational volume needed for the construction of the whole data-base (a complete drag polar in this case) can be estimated in terms of Multigrid (MG) cycles. The conventional approach necessitated 780 MG cycles while the first and the second versions of the cooperative strategy required 480 and 420 MG cycles, respectively.

It may be be assessed from the computed data that the results achieved by the three computations are virtually identical.

5. CONCLUSIONS

A new efficient cooperative strategy for massive parallel CFD computations is proposed. The approach was implemented for 1) computation of large-size aerodynamic data-bases by means of high-accuracy Navier-Stokes simulations and 2) optimal GAs search in the framework of the aerodynamic shapes optimization. The results indicate an essential improvement in the computational efficiency and allow to significantly reduce the CPU time needed for large scale aerodynamic simulations.

REFERENCES

1. Epstein, B., and Peigin, S., "A Robust Hybrid GA/ROM Approach to Multiobjective Constrained Optimization in Aerodynamics" *AIAA Journal*, V.42, No. 8, pp.1572–1581, 2004.
2. Epstein, B., Rubin, T., and Seror, S., "Accurate Multiblock Navier-Stokes Solver for Complex Aerodynamic Configurations", *AIAA Journal*, V.41, No.4, pp.582–584, 2003.
3. Peigin, S., Epstein, B., Rubin, T., and Seror, S., "Parallel Large Scale High Accuracy Navier-Stokes Computations on Distributed Memory Clusters", *The Journal of Supercomputing*, Vol. 27, No.1, pp.49–68, 2003.
4. Peigin, S., Epstein, B., and Gali, S., "Multilevel Parallelization Strategy for Optimization of Aerodynamic Shapes", in *Parallel Computational Fluid Dynamics, New Frontiers and Multi-disciplinary Applications*, Elsevier, 2004.