A three-level parallelization strategy for robust design in aerodynamics

R. Duvigneau [a], T. Kloczko[b], and Praveen C.[a].

[a]INRIA Sophia-Antipolis Méditerranée, OPALE Project-Team
06902 Sophia-Antipolis, France

[b]INRIA Sophia-Antipolis Méditerranée, SMASH Project-Team
06902 Sophia-Antipolis, France

A three-level parallelization strategy is developed to achieve an efficient robust design procedure in aerodynamics. We first parallelize the optimization algorithm, that computes the aerodynamic performance for different shapes. Then we parallelize the procedure that performs a statistical fitness estimation for robustness purpose. Finally, a parallel CFD solver is used to simulate the flows. The proposed method is demonstrated on a realistic testcase and we analyze the computational efficiency in order to provide some practical guidelines.

1. SHAPE OPTIMIZATION IN AERODYNAMICS

A typical shape optimization problem in aerodynamics consists in minimizing a cost functional, that depends on design variables (dimension n) and flow variables, submitted to some constraints. For instance, one considers a wing shape optimization by minimizing the drag with a lift constraint. Several optimization strategies have been developed in the past to solve such a problem, ranging from gradient-based methods to semi-stochastic methods like genetic algorithms. Gradient-based methods are moderately expensive, in terms of function evaluations but can be trapped into local minima, whereas semi-stochastic methods have the capability to avoid such local minima, but are far more expensive.

Since aerodynamics is highly non-linear and often generates multi-modal cost functionals, semi-stochastic methods are particularly popular and will be considered in this study. These methods require from several hundreds to several thousands of simulations, before converging to the optimum shape. Hopefully, some of these simulations can be run independently and the computational cost can be significantly reduced by implementing a parallel computation strategy. Most semi-stochastic methods can be written as:

1. initialize a population of N shapes
2. compute the cost function for the N shapes (requires N simulations)
3. use an evolution strategy to update the population
4. goto step (2)

Obviously, step (2) can be parallelized, by running the flow solver that computes the cost function for the different shapes using different processes. This strategy is particularly efficient,

since the amount of communication is very small. In practice, only the cost function value computed after each simulation by a given process has to be communicated to other processes. However, the computational time required by each flow solver run can vary, since the convergence is not the same for all the shapes. Since the previous procedure is synchronous, this yields a (moderate) waste of computational time.

In the present study, we use a Particle-Swarm Optimization (PSO) method[1]. PSO algorithm was first introduced by Kennedy and Eberhart [3], as a simplified social model. It mimics the behavior of birds flocking and is based on underlying rules that enable sudden direction changes, scattering, regrouping, etc. These moves are motivated in nature by food seeking or predators avoiding and can be implemented in a simple algorithm for global optimization purpose [6]. The algorithm consists in building the trajectories of N particles of a swarm in the search space of dimension n (the position of each particle represents a shape). These trajectories are adjusted dynamically to take into account the information collected about the cost function value in order to converge to the optimum point.

2. ROBUST DESIGN

All the methods developed in CFD assume a perfect knowledge of the parameters of the system of interest, such as Mach number, angle of attack, wall position, etc. However, everyday life is subject to uncertainty and the parameters of every system are subject to random fluctuations. For instance the flight conditions can fluctuate according to the weather conditions, the wing design can vary because of manufacturing tolerances, the angle of attack can be modified due to wind fluctuations, etc. These uncertainties will modify the flow and in some cases will significantly degrade the performance of a system that has been optimized for some precise operational conditions.

Therefore, we have developed a *robust design* approach, that rely on a statistical estimation of the cost function[2]. Instead of simulating the flow only at nominal operational conditions to compute the cost function value, we estimate the mean and the variance of the cost function by taking into account the variability of operational conditions. In practice, we simulate the flow for a few number of different operational condition values, such as Mach number and incidence, and results are stored into a database. Then, we construct a surrogate model[5], that is used to integrate the cost function over the range of operational conditions to derive statistics. For each shape, the procedure can be written as:

1. compute and store the cost function for P operational condition values (requires P simulations)
2. construct a surrogate model based on the database
3. estimate statistics based on the surrogate model

Again, step (1) can be parallelized, by running the flow solver that computes the cost functions for the different operational conditions using different processes. This parallelization strategy is similar to the previous one, except that the information communicated does not concern different shapes, but different operational conditions for the same shape. The two strategies can then be combined.

3. FLOW SOLVER

The flow solver used in this study is the NUM3SIS code developed at INRIA Sophia-Antipolis. It solves compressible Euler / Navier-Stokes equations on 3D tetrahedral grids using a mixed

finite-volume finite-element method with a cell-vertex approach. Convective fluxes are discretized using approximate Riemann solvers, such as HLLC or AUSM+ methods. A MUSCL interpolation is employed on the dual mesh to obtain high-order schemes. The diffusive terms are computed using a P1-Galerkin approach in a classical finite-element framework. The time integration is achieved using a first-order implicit backward scheme, that is based on a matrix-free implementation.

The NUM3SIS code is parallelized using a domain decomposition approach [4]. It is based on a simple overlapping strategy, since only boundary nodes are shared by adjacent domains. For these nodes, fluxes are computed independently in each domain. Then, communications allow to sum all contributions from the different domains. Finally, the state update for shared nodes is performed in all domains.

The performance of this code for parallel computations will be analyzed in depth in [4].

4. PARALLELIZATION STRATEGY

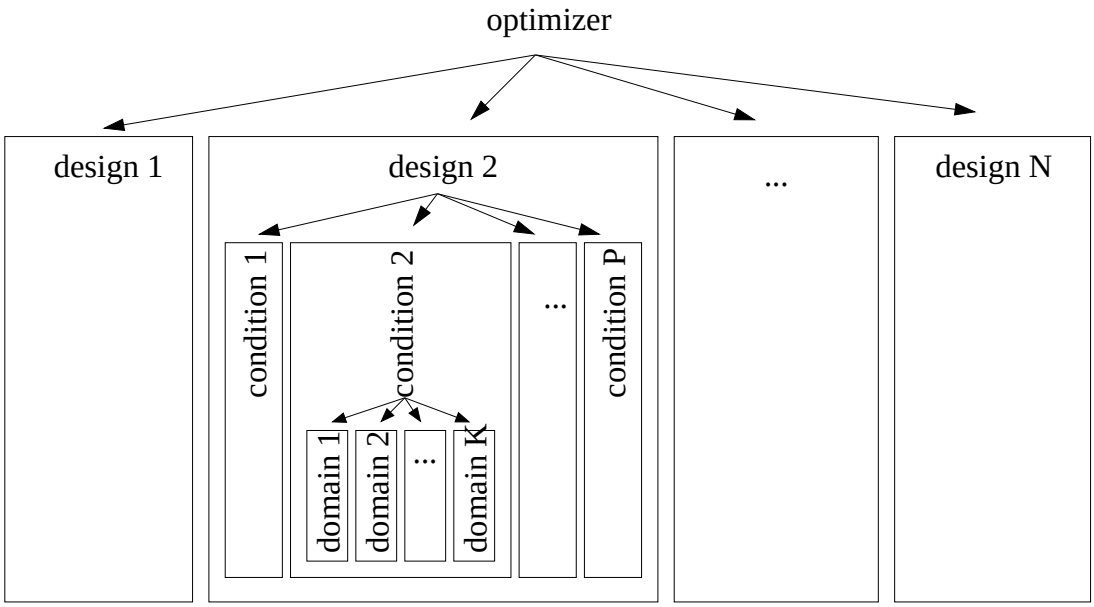

Figure 1. The three-level parallelization strategy.

The three-level parallelization strategy is described in figure 1. It is implemented using the MPI library. In practice, two MPI programs are used: a first one manages the optimization procedure and the two first parallelization levels (distribution of the CFD code). The parallel CFD code is a second program, that is called by the first one using scripts.

Using such a strategy, a large number of processors could be used: a typical application would involve 50 individuals for the optimizer, 10 different operating conditions and 20 CFD domains, yielding a possible use of 10 000 processors.

In practice, such a large computational facility is not yet available. Therefore, the objective of this paper is to compare different strategies in order to maximize the computational efficiency

for a given hardware configuration, and give some guidelines. For instance, is it more efficient to parallelize only the flow solver, or to parallelize only the optimization procedure ? When both options are possible, which one should be encouraged ?

5. RESULTS

The test-case considered here corresponds to the optimization of the wing shape of a business aircraft (courtesy of Piaggio Aero Ind.), for a transonic regime. The nominal operating conditions are defined by the free-stream Mach number $M_\infty = 0.83$ and the incidence $\alpha = 2°$. We suppose that the free-stream Mach number as well as the incidence are subject to random fluctuations. The goal of the optimization is to reduce the mean and the variance of the drag coefficient C_D (robust design), subject to a lift constraint.

The aerodynamic coefficients are computed by simulating the Eulerian flow around the wing using the NUM3SIS flow solver. The mesh employed contains 31124 nodes and includes a refined area to accurately capture the shock wave. The wing shape is parameterized using $n = 32$ degrees of freedom. A PSO algorithm is used as optimizer.

The optimizer evaluates the performance of 32 shapes simultaneously and each performance estimation requires 4 simulations corresponding to 4 different values of Mach number and incidence. Then, 128 processors could be employed at each optimization step. Besides, the flow solver can also be run in parallel.

We have solved the problem using from 1 to 128 processors, with the following strategies:

- parallelize the optimization procedure (two-level parallelization) and use a sequential flow solver.
- parallelize the flow solver (one-level parallelization) and use a sequential optimizer.
- parallelize both the flow solver and the optimizer (three-level parallelization).

The results obtained are shown in figure 2 and table 1. These computations are performed on a cluster with processors Xeon 2.66Ghz. Only three optimization steps are considered here for these tests.

As seen, the parallelization of the solver is particularly inefficient in this case, since the mesh size is very small.

Parallelization of the optimizer is more efficient, even if some delay is observed, due to the fact that the different simulations do not converge with the same CPU cost.

The hybrid approach is more efficient if a large number of processors is used, since each parallelization task is achieved with a lower number of processes.

6. PROSPECTS

We are currently performing the same study including a test-case with a larger grid size. Then, the parallelization of the solver will be more efficient and the conclusions will be drawn in a more suitable framework.

REFERENCES

1. R. Duvigneau. A multi-level particle swarm optimization strategy for aerodynamic shape optimization. In *EUROGEN 2007, Evolutionary Methods for Design, Optimization and Control*, Jyvaskyla, Finland, June 2007.

Proc. for solver	Proc. for optimizer	CPU time (s)	Speed up
1	1	364094	1.
1	8	45798	7.95
1	16	24029	15.247
1	32	13141	27.880
1	64	8597	42.617
1	128	5565	65.425
8	1	68188	5.3731
16	1	57882	6.3298
32	1	58335	6.2806
64	1	59492	6.1585
4	8	63145	5.765
4	16	32850	11.083
4	32	16895	21.550
4	64	8803	41.360
4	128	4770	76.329

Table 1
CPU time for the different strategies

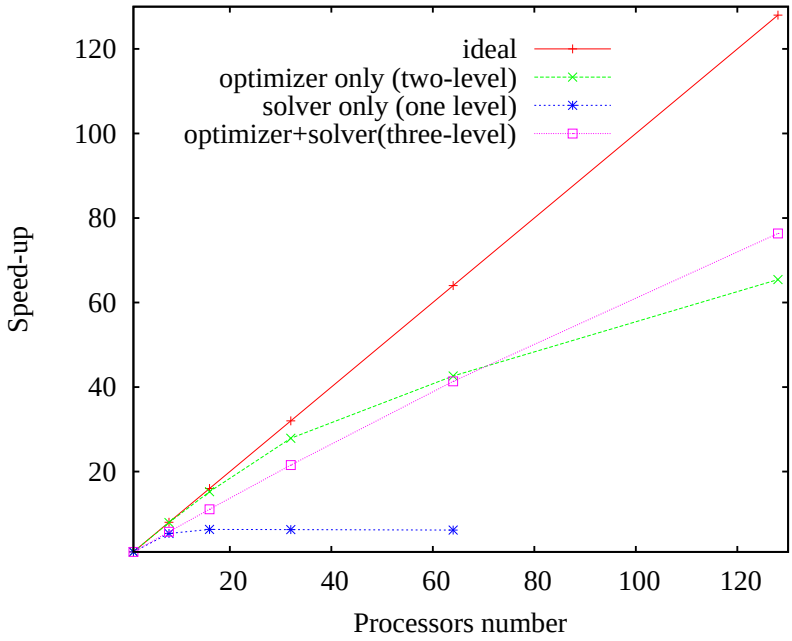

Figure 2. Speed-up for the different strategies

2. R. Duvigneau. Robust design of a transonic wing with uncertain mach number. In *EUROGEN 2007, Evolutionary Methods for Design, Optimization and Control*, Jyvaskyla, Finland, June 2007.
3. J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *1995 IEEE International*

Conference on neural networks, Perth, Australia, 1995.
4. T. Kloczko. Concept, architecture and performance study for a parallel code in cfd. In *20th International Conference on Parallel Computational Fluid Dynamics*, May 2008.
5. C. Praveen and R. Duvigneau. Radial basis functions and kriging metamodels for aerodynamic optimization. Research Report 6151, INRIA, 03 2007.
6. G. Venter and J. Sobieszczanski-Sobieski. Particle swarm optimization. *AIAA Journal*, 41(8):1583–1589, August 2003.