ON THE ESTIMATION OF DRAG UNCERTAINTY

M. Martinelli, Praveen C. & R. Duvigneau
Opale Project-Team, INRIA Sophia-Antipolis Méditerranée (France)
Massimiliano.Martinelli@sophia.inria.fr
Praveen.Chandrashekarappa@sophia.inria.fr
Regis.Duvigneau@sophia.inria.fr

ABSTRACT

In this paper, we present different practical methods to estimate the drag uncertainty in Computational Fluid Dynamics (CFD). We aim at taking into account the variability of operational conditions, such as Mach number or angle of attack, in order to replace the classical computation of the drag by the estimation of statistics on the drag, such as mean and variance. We compare two methods: the use of an Automatic Differentiation (AD) software to compute the derivatives of the drag with respect to uncertain parameters and the use of meta-modeling techniques to represent the drag variability from a database. A practical testcase is presented to mesure the accuracy and the computational efficiency of the proposed methods.

INTRODUCTION

Simulation-based drag prediction has been an active research topic for the last years and is now applied for industrial cases in aerodynamics. All the methods developed in Computational Fluid Dynamics (CFD) assume a perfect knowledge of the parameters of the system of interest, such as Mach number, angle of attack, wall position, etc. Under this asumption, numerical methods have been developed, yielding an accurate drag prediction. However, everyday life is subject to uncertainty and the parameters of every systems are subject to random fluctuations. For instance the flight conditions can fluctuate according to the weather, 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 drag and in some cases will significantly degrade the performance of a system that has been optimized for some precise operational conditions.

Therefore, some numerical methods have been developed recently to estimate the drag uncertainty. According to AIAA (American Institute of Aeronautics ans Astronautics), the uncertainty in CFD is "a potential deficiency in any phase or activity of the modeling process that is due to lack of knowledge" [1]. This definition underlines the stochastic nature of uncertainty. One should not confuse with the numerical error. Error in CFD simulations is clearly deterministic and is defined as "a recognizable deficiency in any phase or activity of the modeling process that is not due to lack of knowledge" [1]. These two concepts sometimes lead to confusion.

A review of the numerical methods to estimate uncertainty in CFD is proposed by Huyse [21]. The most often used methods are : Monte-Carlo estimate [21], method of moments [3, 7, 14, 20], polynomial chaos expansion [9]. These methods consider uncertain parameters as random variables and assume that their Probability Density Function is known. Then, they propose different approaches to estimate some statistical quantities that characterize the aerodynamic coefficients, such as expectation or variance, by propagating uncertainty through the CFD code.

In this paper, we intend to compare for practical testcases two approaches commonly used in the CFD community and compare their accuracy and computational efficiency:

- The first method consist in using an Automatic Differentiation (AD) software to compute the first and second derivatives of the drag with respect to uncertain parameters. Then, we assume that the drag fluctuations can be described by a second-order Taylor series expansion and statistics are evaluated by analytic integration (method of moments).
- In the second method, we compute the drag for a few number of uncertain parameter values to build a database. It is then used to construct an inexpensive meta-model that describes the drag fluctuations. Statistics are computed by numerical integration or Monte-Carlo simulation on the basis of the meta-model.

We briefly describe the two approaches in the first sections of the paper. Then, we consider a practical testcase in the context of Eulerian flows : the subsonic flow around the wing of a business aircraft subject to uncertain operating conditions in term of Mach number and incidence.

UNCERTAINTY ESTIMATION USING AD

Computation of first-order derivatives

We aim at computing the derivatives of the aerodynamic coefficients with respect to uncertain variables [4, 11, 19]. We consider the aerodynamic coefficient of interest, such as drag of lift, as a functional j expressed as:

$$j : \gamma \mapsto j(\gamma) = J(\gamma, W) \in \mathbb{R},$$

where $\gamma \in \mathbb{R}^n$ are parameters subject to random fluctuations and the state variables $W = W(\gamma) \in \mathbb{R}^{N_S}$ satisfy a (nonlinear) state equation:

$$\Psi(\gamma, W) = 0.$$

Using the chain rule, the gradient of the functional with respect to each component of γ is given by:

$$\frac{dj}{d\gamma_i} = \frac{dj}{d\gamma} e_i = \frac{\partial J}{\partial \gamma_i} + \frac{\partial J}{\partial W}\frac{dW}{d\gamma_i}. \quad (1)$$

The differentiation of the state equation reads:

$$\frac{\partial \Psi}{\partial \gamma_i} + \frac{\partial \Psi}{\partial W}\frac{dW}{d\gamma_i} = 0. \quad (2)$$

This equation yields the computation of the flow sensitivities $\theta_i = \frac{dW}{d\gamma_i}$ by solving the following linear system:

$$\frac{\partial \Psi}{\partial W}\theta_i = -\frac{\partial \Psi}{\partial \gamma_i}. \tag{3}$$

The first-order derivatives of j with respect to uncertain parameters γ can be obtained by solving equation (3) to obtain the flow sensitivities first, and then by using (1). However, using such a method, we should solve one linear system for each uncertain parameter γ_i. It is more efficient to adopt a so-called adjoint approach. Combining equations (1) and (2) in transposed form, we get:

$$\left(\frac{dj}{d\gamma}\right)^\top = \left(\frac{\partial J}{\partial \gamma}\right)^\top - \left(\frac{\partial \Psi}{\partial \gamma}\right)^\top \left(\frac{\partial \Psi}{\partial W}\right)^{-\top} \left(\frac{\partial J}{\partial W}\right)^\top.$$

Then, we can easily obtain the gradient of j by solving the adjoint system first:

$$\left(\frac{\partial \Psi}{\partial W}\right)^\top \Pi = \left(\frac{\partial J}{\partial W}\right)^\top, \tag{4}$$

where Π are the adjoint variables, and then by computing:

$$\left(\frac{dj}{d\gamma}\right)^\top = \left(\frac{\partial J}{\partial \gamma}\right)^\top - \left(\frac{\partial \Psi}{\partial \gamma}\right)^\top \Pi. \tag{5}$$

Using the adjoint approach, only one linear system solving is required, whatever the number of uncertain variables.

Computation of second-order derivatives

Starting from the first-order derivative (1), we perform another differentiation with respect to the k-th component of γ, which reads:

$$\frac{d^2 j}{d\gamma_i d\gamma_k} = D^2_{i,k} J + \frac{\partial J}{\partial W}\frac{d^2 W}{d\gamma_i d\gamma_k}, \tag{6}$$

with:

$$\begin{aligned} D^2_{i,k} J =& \frac{\partial}{\partial \gamma}\left(\frac{\partial J}{\partial \gamma} e_i\right) e_k + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial \gamma} e_i\right)\frac{dW}{d\gamma_k} \\ &+ \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial \gamma} e_k\right)\frac{dW}{d\gamma_i} + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial W}\frac{dW}{d\gamma_i}\right)\frac{dW}{d\gamma_k}. \end{aligned}$$

Then, we differentiate equation (2) to obtain:

$$D^2_{i,k}\Psi + \frac{\partial \Psi}{\partial W}\frac{d^2 W}{d\gamma_i d\gamma_k} = 0, \tag{7}$$

with:

$$\begin{aligned} D^2_{i,k} \Psi =& \frac{\partial}{\partial \gamma}\left(\frac{\partial \Psi}{\partial \gamma} e_i\right) e_k + \frac{\partial}{\partial W}\left(\frac{\partial \Psi}{\partial \gamma} e_i\right)\frac{dW}{d\gamma_k} \\ &+ \frac{\partial}{\partial W}\left(\frac{\partial \Psi}{\partial \gamma} e_k\right)\frac{dW}{d\gamma_i} + \frac{\partial}{\partial W}\left(\frac{\partial \Psi}{\partial W}\frac{dW}{d\gamma_i}\right)\frac{dW}{d\gamma_k}. \end{aligned}$$

If we substitute the second-order derivatives of the state with respect to uncertain parameters $\frac{d^2 W}{d\gamma_i d\gamma_k}$ in equation (6) from equation (7), we get:

$$\begin{aligned} \frac{d^2 j}{d\gamma_i d\gamma_k} &= D^2_{i,k} J - \frac{\partial J}{\partial W}\left(\frac{\partial \Psi}{\partial W}\right)^{-1} D^2_{i,k}\Psi \quad (8) \\ &= D^2_{i,k} J - \Pi^\top D^2_{i,k}\Psi, \quad (9) \end{aligned}$$

where Π is the solution of the adjoint system (4). This approach was firstly proposed by [18] and is usually known as *Tangent on Tangent* (ToT) approach ([11]) or *Forward-on-Forward* ([5]), since we apply two direct differentiations to the functional.

Algorithm

The algorithm to compute the first and second derivatives is summarized in the next table :

1. Solve for the adjoint variables Π in:
   $\left(\frac{\partial \Psi}{\partial W}\right)^\top \Pi = \left(\frac{\partial J}{\partial W}\right)^\top$
2. Compute the gradient of j:
   $\left(\frac{dj}{d\gamma}\right)^\top = \left(\frac{\partial J}{\partial \gamma}\right)^\top - \left(\frac{\partial \Psi}{\partial \gamma}\right)^\top \Pi$
3. For each $i \in 1 \dots n$:
   Solve for the flow sensitivities θ_i in:
   $\left(\frac{\partial \Psi}{\partial W}\right)\theta_i = -\left(\frac{\partial \Psi}{\partial \gamma_i}\right)$
4. For each $i \in 1 \dots n$ and $k \in 1 \dots i$, compute:
   $\frac{d^2 j}{d\gamma_i d\gamma_k} = D^2_{i,k} J - \Pi^\top (D^2_{i,k}\Psi)$

Automatic differentiation

In order to obtain the terms that appear in the algorithm above and containing the first- and second-order derivatives, we need a differentiated version of the original CFD code and this differentiation, if performed "by hand" is tedious and error-prone. Then, we prefer to compute them using the automatic differentiation (AD) software Tapenade [6], developed by Tropics Project-Team at INRIA. This software is used to generate automatically a source code that computes the derivatives of an original FORTRAN code.

Consider a program that computes an output vector $v \in \mathbb{R}^m$ from an input vector $u \in \mathbb{R}^n$ as a function $v = \phi(u)$. The derivative of the function is provided by the Jacobian matrix $\frac{\partial \phi}{\partial u}$. The program is a sequence of elementary instructions that can be identified with a composition of elementary functions. The AD tool simply applies the chain rule to differentiate these elementary functions to obtain the desired Jacobian matrix. However, we are usually not interested by the knowledge of the full Jacobian matrix. Then, Tapenade has two differentiation modes, that allow to compute the product of the Jacobian matrix by a given vector. We can perform this matrix-by-vector product in a twofold manner: by right (*tangent mode*) or by left (*reverse mode*):

- the tangent mode allows to compute, from an arbitrary direction $\dot{u} \in \mathbb{R}^n$, the derivative in the direction $\dot{u}$:

$$u, \dot{u} \mapsto \left.\frac{\partial \phi}{\partial u}\right|_u \dot{u}$$

- the reverse mode allows to compute, from an arbitrary direction $\bar{\phi} \in \mathbb{R}^m$, the following product:

$$u, \bar{\phi} \mapsto \left(\left.\frac{\partial \phi}{\partial u}\right|_u\right)^\top \bar{\phi}$$

These two modes can be employed to easily compute the terms that are required for derivatives estimation.

Application of AD

Consider that the functional $j = J(\gamma, W)$ is computed in a FORTRAN subroutine `func`, whose input variables are `gamma` and `w` and output variable is `j`:

$$\underset{\downarrow J}{\texttt{func}}(\texttt{j}, \overset{\gamma}{\underset{\downarrow}{\texttt{gamma}}}, \overset{W}{\underset{\downarrow}{\texttt{w}}}).$$

If we perform a reverse mode differentiation with respect to the input variables `gamma` and `w`, we obtain a new subroutine:

$$\underset{\downarrow J}{\texttt{func\_b}}(\texttt{j}, \overset{\bar{J}}{\underset{\downarrow}{\texttt{jb}}}, \overset{\gamma}{\underset{\downarrow}{\texttt{gamma}}}, \underset{\downarrow \bar{\gamma}}{\texttt{gammab}}, \overset{W}{\underset{\downarrow}{\texttt{w}}}, \underset{\downarrow \bar{W}}{\texttt{wb}}),$$

where `jb` a new input variable and `gammab` and `wb` are new output variables defined as:

$$\bar{\gamma} = \left(\frac{\partial J}{\partial \gamma}\right)^\top \bar{J} \qquad \bar{W} = \left(\frac{\partial J}{\partial W}\right)^\top \bar{J}. \tag{10}$$

If we evaluate the above subroutine with the input $\bar{J} = 1$, the quantities in (10) are the first term in the right hand side (r.h.s.) of (5) and the r.h.s. of the adjoint equation (4).

Now consider that the subroutine that computes the state residuals $\Psi(\gamma, W)$ is `state`, whose input variables are `gamma` and `w` and output variable is `psi`:

$$\underset{\downarrow \Psi}{\texttt{state}}(\texttt{psi}, \overset{\gamma}{\underset{\downarrow}{\texttt{gamma}}}, \overset{W}{\underset{\downarrow}{\texttt{w}}}).$$

If we perform a tangent mode differentiation, we can easily compute the product of the derivatives of the state residuals with respect to state variables $\frac{\partial \Psi}{\partial W}$ with a given vector, or the derivatives of the state residuals with respect to the uncertain parameters $\frac{\partial \Psi}{\partial \gamma_i}$, required to compute the flow sensitivities in (3). For the first quantity we need to differentiate with respect to the input variable `w`, namely:

$$\underset{\downarrow \Psi \quad \downarrow \dot{\Psi}}{\texttt{state\_dw\_d}}(\texttt{psi}, \texttt{psid}, \overset{\gamma}{\underset{\downarrow}{\texttt{gamma}}}, \overset{W}{\underset{\downarrow}{\texttt{w}}}, \overset{\dot{W}}{\underset{\downarrow}{\texttt{wd}}}),$$

where the new output variable `psid` contains the directional derivative $\dot{\Psi} = \frac{\partial \Psi}{\partial W}\dot{W}$. Similarly, for $\frac{\partial \Psi}{\partial \gamma_i}$ we need to differentiate with respect to the input variable `gamma`:

$$\underset{\downarrow \Psi \quad \downarrow \dot{\Psi}}{\texttt{state\_dgamma\_d}}(\texttt{psi}, \texttt{psid}, \overset{\gamma}{\underset{\downarrow}{\texttt{gamma}}}, \overset{\dot{\gamma}}{\underset{\downarrow}{\texttt{gammad}}}, \overset{W}{\underset{\downarrow}{\texttt{w}}}),$$

and now the new output variable `psid` is $\dot{\Psi} = \frac{\partial \Psi}{\partial \gamma}\dot{\gamma}$. Thus, to compute $\frac{\partial \Psi}{\partial \gamma_i}$ is sufficient to set $\dot{\gamma} = e_i$, where e_i is the i-th vector of the canonical basis.

If we perform a reverse mode differentiation with respect to the input variables, we obtain the following new subroutine:

$$\underset{\downarrow \Psi}{\texttt{state\_b}}(\texttt{psi}, \overset{\bar{\Psi}}{\underset{\downarrow}{\texttt{psib}}}, \overset{\gamma}{\underset{\downarrow}{\texttt{gamma}}}, \underset{\downarrow \bar{\gamma}}{\texttt{gammab}}, \overset{W}{\underset{\downarrow}{\texttt{w}}}, \underset{\downarrow \bar{W}}{\texttt{wb}}),$$

where `gammab` and `wb` are new output variables and `psib` a new input variable. A call to this subroutine allows to compute the matrix-by-vector products:

$$\bar{\gamma} = \left(\frac{\partial \Psi}{\partial \gamma}\right)^\top \bar{\Psi} \quad \bar{W} = \left(\frac{\partial \Psi}{\partial W}\right)^\top \bar{\Psi}. \tag{11}$$

A similar subroutine (without the $\bar{\gamma}$ term) is used to solve the adjoint equation (4). Indeed, using an iterative matrix-free linear solver (e.g. GMRES) in which the matrix-by-vector products are obtained by the differentiated routine (i.e. the $\bar{W}$ term in (11)), we can easily solve the linear system (4) without to store the Jacobian $\left(\frac{\partial \Psi}{\partial W}\right)^\top$.

To compute the terms required to estimate the second-order derivatives (9), we need to perform two successive tangent-mode differentiations. For example, considering the tangent-mode differentiation of the subroutine `func` with respect to `gamma` and `w` we obtain:

$$\underset{\downarrow J \quad \downarrow \dot{j}}{\texttt{func\_d}}(\texttt{j}, \texttt{jd}, \overset{\gamma}{\underset{\downarrow}{\texttt{gamma}}}, \overset{\dot{\gamma}}{\underset{\downarrow}{\texttt{gammad}}}, \overset{W}{\underset{\downarrow}{\texttt{w}}}, \overset{\dot{W}}{\underset{\downarrow}{\texttt{wd}}}),$$

where `gammad` and `wd` are new input variables and `jd` a new output variable. These input variables are provided by the user, whereas the output variable is the directional derivative:

$$\dot{J} = \frac{\partial J}{\partial \gamma}\dot{\gamma} + \frac{\partial J}{\partial W}\dot{W}.$$

Then, the differentiation of the output variable `jd` in the subroutine `func_d` with respect to input variables `gamma` and `w` gives us:

$$\underset{\downarrow J \quad \downarrow \dot{j} \quad \downarrow \ddot{j}}{\texttt{func\_dd}}(\texttt{j}, \texttt{jd}, \texttt{jdd}, \overset{\gamma}{\underset{\downarrow}{\texttt{gamma}}}, \overset{\dot{\gamma}_0}{\underset{\downarrow}{\texttt{gammad0}}}, \overset{\dot{\gamma}}{\underset{\downarrow}{\texttt{gammad}}}, \overset{W}{\underset{\downarrow}{\texttt{w}}}, \overset{\dot{W}_0}{\underset{\downarrow}{\texttt{wd0}}}, \overset{\dot{W}}{\underset{\downarrow}{\texttt{wd}}}),$$

where `jdd` is the new output variable, that represents:

$$\begin{aligned}\ddot{J} =& \frac{\partial}{\partial \gamma}\left(\frac{\partial J}{\partial \gamma}\dot{\gamma}\right)\dot{\gamma}_0 + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial \gamma}\dot{\gamma}\right)\dot{W}_0 \\ &+ \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial \gamma}\dot{\gamma}_0\right)\dot{W} + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial W}\dot{W}\right)\dot{W}_0.\end{aligned} \tag{12}$$

If one calls this subroutine with the following input parameters:

$$\dot{\gamma}_0 = e_k \quad \dot{\gamma} = e_i \quad \dot{W}_0 = \frac{dW}{d\gamma_k} \quad \dot{W} = \frac{dW}{d\gamma_i}, \tag{13}$$

one obtains as output variable $\ddot{J} = D^2_{i,k}J$. Then this routine is used to compute the second order derivatives according to (9). Note that the flow sensitivities $\dot{W} = \frac{dW}{d\gamma}$ should be computed and stored before the evaluation of $D^2_{i,k}J$, according to the algorithm presented previously. The term $D^2_{i,k}\Psi$ can be computed in the same way, by applying a tangent differentiation mode to the subroutine `state_d`. As conclusion, AD can be used in an efficient way to compute the first and second derivatives of the functional, according to the algorithm presented in a previous section.

Uncertainty estimation

The derivatives of the drag with respect to uncertain parameters are now used to estimate drag uncertainty. We assume that the drag variation due to uncertain parameters fluctuations can be described by a first- or second-order Taylor series expansion around the mean value μ_γ of uncertain parameters:

$$j(\gamma) \approx j(\mu_\gamma) + \sum_{i=1}^{n} \frac{dj}{d\gamma_i}(\gamma_i - \mu_{\gamma_i}), \tag{14}$$

or:

$$j(\gamma) \approx j(\mu_\gamma) + \sum_{i=1}^{n} \frac{dj}{d\gamma_i}(\gamma_i - \mu_{\gamma_i}) + \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n} \frac{d^2 j}{d\gamma_i d\gamma_k}(\gamma_i - \mu_{\gamma_i})(\gamma_k - \mu_{\gamma_k}). \tag{15}$$

Since the drag is now considered as a random variable, we would like to characterize it by estimating some statistical quantities, such as expectation or variance. The expectation of the drag is:

$$\mu_j = \int_\Omega j(\gamma)\rho(\gamma)d\gamma, \tag{16}$$

where Ω and ρ are respectively the range and the probability density function (PDF) of the uncertain parameters γ.

If we use the first-order Taylor series expansion (14) to estimate the previous integral, we obtain:

$$\mu_j \approx j(\mu_\gamma), \tag{17}$$

In this case, the mean value of the drag is the drag estimated at the mean value of uncertain parameters. If we use the second-order Taylor series expansion (15) and assume the variables γ_i, γ_k to be independent, we obtain:

$$\mu_j \approx j(\mu_\gamma) + \frac{1}{2}\sum_{i=1}^{n} \frac{d^2 j}{d\gamma_i d\gamma_i}\sigma^2_{\gamma_i}, \tag{18}$$

where $\sigma^2_{\gamma_i}$ is the variance of the uncertain parameter γ_i. One can notice that the mean value of the drag is not the drag estimated at the mean value of uncertain parameters.

The variance of the drag is defined by:

$$\sigma_j^2 = \int_\Omega (j(\gamma) - \mu_j)^2 \rho(\gamma)d\gamma. \tag{19}$$

If we use the first-order Taylor series expansion (14) and the mean value (17) to estimate the previous integral, we obtain:

$$\sigma_j^2 \approx \sum_{i=1}^{n} \left(\frac{dj}{d\gamma_i}\right)^2 \sigma^2_{\gamma_i}. \tag{20}$$

If we use the second-order Taylor series expansion (15) and the mean value (18) to estimate the previous integral, we obtain:

$$\sigma_j^2 \approx \sum_{i=1}^{n} \left(\frac{dj}{d\gamma_i}\right)^2 \sigma^2_{\gamma_i} + \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n} \left(\frac{d^2 j}{d\gamma_i d\gamma_k}\right)^2 \sigma^2_{\gamma_i}\sigma^2_{\gamma_k}. \tag{21}$$

Note that this formula is restricted to the case the PDFs of uncertain parameters are Gaussian. Finally, according to the first and second derivatives computed using AD, the drag uncertainty can be characterized by estimating its expectation and variance using (17) and (20) or (18) and (21).

UNCERTAINTY ESTIMATION USING META-MODELS

Meta-models

Meta-models are surrogate models whose evaluations are far less expensive than the original model (PDEs for instance). Then, they can be used for a negligible cost to remplace some evaluations of the original model or to provide additional information.

Meta-models are constructed according to available data that are stored in a database. It consists in using these data (drag computed for some parameter sets) to predict the drag for new parameters. This database can be generated separately or compiled during an optimization procedure for instance. meta-models mostly used for data fitting are:

- polynomial fitting (least-squares approximation) ;
- artificial neural networks (multi-layer perceptrons) [15];
- radial basis functions [13] ;
- Kriging methods (Gaussian process models) [17].

The last three options are well suited to highly non-linear behaviors, such as those encountered in aerodynamics.

For uncertainty propagation purpose, meta-models can be employed to estimate the drag modification due to random fluctuations of the uncertain parameters. Since these estimations are inexpensive, classical Monte-Carlo simulation or numerical integration of PDF can then be used to compute drag statistics.

The choice of the database point is critical for the accuracy of the meta-model, whatever it is. Several methods can be found in the literature that describe how to choose the training points[8]. In the present study, the database points are generated using either a regular grid, or a Latin Hypercube Sampling (LHS) of size N. Latin hypercube samples have the property that any projection along one variable yields a regular distribution. In order to avoid to have an ill-conditioned database (for instance, a diagonal-like sample), several LHS are performed in practice and we consider as database the one that maximizes the distances between the points.

Radial basis functions

RBFs are non-polynomial interpolation methods for scattered data [13]. They have been found to be very accurate for highly non-linear data in high dimension [2]. RBFs seek an approximation of the function $j(\gamma)$, $\gamma \in \mathbb{R}^n$ of the form:

$$\hat{j}(\gamma) = \sum_{i=1}^{N} \omega_i \, \phi_i(\gamma), \tag{22}$$

where:

$$\phi_i(\gamma) = \Phi(\|\gamma - \gamma^i\|). \tag{23}$$

$\Gamma^N = (\gamma^i)_{i=1,\ldots,N}$ are called RBFs centers and correspond to the points stored in the database. Several radial functions Φ can be considered. For the present study a Gaussian function is employed:

$$\Phi(r) = e^{-\frac{r^2}{s^2}}, \tag{24}$$

where s is a parameter called attenuation factor, that controls the extend of the basis functions. Therefore, the evaluation of the RBFs using (22-24) has a negligible computational cost.

The training of the RBFs consists in determining the weights $(\omega_i)_{i=1,\dots,N}$ to fit the data. Suppose that the function value is known for a set of N points that correspond to the RBF centers $(\gamma^k)_{k=1,\dots,N}$. Then, the weights $(\omega_i)_{i=1,\dots,N}$ are determined from the interpolation conditions:

$$j(\gamma^k) = \sum_{i=1}^{N} \omega_i \, \phi_i(\gamma^k) \quad k = 1, \dots, N. \tag{25}$$

Thus, $(\omega_i)_{i=1,\dots,N}$ is the solution of the following linear system:

$$\begin{pmatrix} \phi_1(\gamma^1) & \dots & \phi_N(\gamma^1) \\ \phi_1(\gamma^2) & \dots & \phi_N(\gamma^2) \\ \dots & \dots & \dots \\ \phi_1(\gamma^N) & \dots & \phi_N(\gamma^N) \end{pmatrix} \begin{Bmatrix} \omega_1 \\ \omega_2 \\ \dots \\ \omega_N \end{Bmatrix} = \begin{Bmatrix} j(\gamma^1) \\ j(\gamma^2) \\ \dots \\ j(\gamma^N) \end{Bmatrix}. \tag{26}$$

The matrix of the system is obviously symmetric. It is also positive-definite if the RBF centers $(\gamma^i)_{i=1,\dots,N}$ are distinct. One can notice that it is a full matrix. However, its dimension N is usually moderate in practice. Then, the computational cost of its inversion is far lower than that of PDEs solving. The attenuation factor s can be determined experimentally by the user. An empirical formula is also proposed in [12]: However, the choice of the attenuation factor can be tedious for some applications. Then, we propose to optimize it by minimizing an error estimation functional. According to [16], we employ the *leave-one-out* technique: one point of the data set is ignored during the training. This point is then used to estimate the fitting error. By considering successively all the points of the data set, one can estimate a global error for a given attenuation factor. Thus, a Particle Swarm Optimization (PSO) algorithm is used to determine the attenuation factor that minimises this error.

Kriging

Gaussian process models [10, 17] (also called kriging) treat the response of some experiment as if it were a realization of a stochastic process. In the following, we adopt the Bayesian viewpoint of Gaussian processes [10]. The vector of known function values $J^N = (j(\gamma^i))_{i=1,\dots,N}$ is assumed to be one realization of a multivariate Gaussian process with joint probability density:

$$p(J^N|\Gamma^N) = \frac{\exp\left(-\frac{1}{2} {J^N}^\top C_N^{-1} J^N\right)}{\sqrt{(2\pi)^N \det(C_N)}} \tag{27}$$

where C_N is the $N \times N$ covariance matrix. The element C_{ik} of the covariance matrix gives the correlation between the function values $j^i = j(\gamma^i)$ and $j^k = j(\gamma^k)$. This is expressed in terms of a covariance function c, i.e., $C_{ik} = c(\gamma^i, \gamma^k)$. When adding a new point γ^{N+1}, the resulting vector of function values J^{N+1} is assumed to be a realization of the $(N+1)$-variable Gaussian process with joint probability density:

$$p(J^{N+1}|\Gamma^{N+1}) = \frac{\exp\left(-\frac{1}{2} {J^{N+1}}^\top C_{N+1}^{-1} J^{N+1}\right)}{\sqrt{(2\pi)^{N+1} \det(C_{N+1})}}. \tag{28}$$

Using the rule of conditional probabilities $p(A|B) = p(A,B)/p(B)$ we can write the probability density for the unknown function value f_{N+1}, given the data (Γ^N, J^N) as:

$$p(j^{N+1}|\Gamma^{N+1}, J^N) = \frac{p(J^{N+1}|\Gamma^{N+1})}{p(J^N|\Gamma^N)}. \tag{29}$$

In order to simplify this expression we notice that the $(N+1)$-variable covariance matrix C_{N+1} can be written as:

$$C_{N+1} = \begin{bmatrix} C_N & k \\ k^\top & \kappa \end{bmatrix}$$

where $k = [c(\gamma^1, \gamma^{N+1}), c(\gamma^2, \gamma^{N+1}), \dots, c(\gamma^N, \gamma^{N+1})]^\top$ and $\kappa = c(\gamma^{N+1}, \gamma^{N+1})$. This gives:

$$C_{N+1}^{-1} = \begin{bmatrix} M & m \\ m^\top & \mu \end{bmatrix}$$

where:

$$M = C_N^{-1} + \frac{1}{\mu} m m^\top \; m = -\mu C_N^{-1} k \; \mu = (\kappa - k^\top C_N^{-1} k)^{-1}$$

Then, using (27) and (27) the probability density for the function value at the new point is:

$$p(j^{N+1}|\Gamma^{N+1}, J^N) \propto \exp\left[-\frac{(j^{N+1} - \hat{j}^{N+1})^2}{2\sigma^2_{j^{N+1}}}\right]$$

where:

$$\hat{j}^{N+1} = k^\top C_N^{-1} J^N, \qquad \sigma^2_{j^{N+1}} = \kappa - k^\top C_N^{-1} k \tag{30}$$

Thus the probability density for the function value at the new point γ^{N+1} is also Gaussian with mean $\hat{j}^{N+1}$ and standard deviation $\sigma_{j^{N+1}}$. Therefore, the most likely value for the function at the new point is $\hat{j}^{N+1}$. The availability of the variance of the estimated function value is an important advantage of this method, since it represent a kind of error estimate of the prediction.

The covariance function must reflect the characteristics of the output of the computer code. For a smooth response, a covariance function with sufficient number of derivatives is preferable, whereas an irregular response might require a covariance function with no derivatives. In the absence of any knowledge regarding the unknown function, the most commonly used correlation function is an exponential:

$$c(\gamma^i, \gamma^k) = \theta_1 \exp\left[-\frac{1}{2} \sum_{l=1}^{n} \frac{(\gamma_l^i - \gamma_l^k)^2}{r_l^2}\right] + \theta_2$$

where $\Theta = (\theta_1, \theta_2, r_1, r_2, \dots, r_n)$ are some parameters to be determined. The first term is a distance-dependent correlation between the function values at two data points; if their distance is small compared to the length scales r_i, the exponential term is close to one while it decays exponentially as their distance increases. The parameter θ_1 scales this correlation. In the second term, θ_2 gives an offset of the function values from zero. It is also common to use a single length scale r which reduces the number of parameters to three irrespective of the dimension n of the independent variables. This choice may be sufficient for constructing local kriging models.

It now remains to find the parameters Θ in the correlation function. These parameters are determined by maximizing the joint probability density $p(J^N|\Gamma^N)$. This is equivalent to minimizing the log-likelihood function given by

$$\mathfrak{L} = {J^N}^\top C_N^{-1} J^N + \log\det(C_N)$$

This function is known to be multi-modal; hence robust techniques like genetic algorithms or Particle Swarm Optimization (PSO) might be preferable. Many researchers have combined such techniques with a final gradient search to locate the minimum more accurately. In this work we have used PSO technique but without any gradient search.

Uncertainty estimation

Using the meta-models described above, one can obtain an inexpensive evaluation of the drag as the uncertain parameters vary, on the basis of a few number of CFD evaluations that correspond to different parameter values. Then, the statistical quantities (16) and (19) can be easily estimated using numerical integration or Monte-Carlo simulation on the basis of the meta-model approximation:

$$\mu_j \approx \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} \hat{j}(\gamma^i) \tag{31}$$

$$\sigma_j \approx \frac{1}{N_{MC}-1} \sum_{i=1}^{N_{MC}} (\hat{j}(\gamma^i) - \mu_j)^2, \tag{32}$$

where N_{MC} is the size of the sample considered for the Monte-Carlo simulation.

FLOW SOLVER

Flow equations

This study is restricted to three-dimensional inviscid compressible flows governed by the Euler equations. Then, the state equations can be written in the conservative form:

$$\frac{\partial W}{\partial t} + \frac{\partial F_1(W)}{\partial x} + \frac{\partial F_2(W)}{\partial y} + \frac{\partial F_3(W)}{\partial z} = 0, \tag{33}$$

where W are the conservative flow variables $(\rho, \rho u, \rho v, \rho w, E)$, with ρ the density, $\overrightarrow{U} = (u, v, w)$ the velocity vector and E the total energy per unit of volume. $\overrightarrow{F} = (F_1(W), F_2(W), F_3(W))$ is the vector of the convective fluxes, whose components are given by:

$$F_1(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E+p) \end{pmatrix} \quad F_2(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(E+p) \end{pmatrix} \quad F_3(W) = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(E+p) \end{pmatrix}. \tag{34}$$

The pressure p is obtained from the perfect gas state equation:

$$p = (\gamma - 1)(E - \frac{1}{2}\rho \|\overrightarrow{U}\|^2), \tag{35}$$

where $\gamma = 1.4$ is the ratio of the specific heat coefficients.

Spatial discretization

Provided that the flow domain Ω is discretized by a tetrahedrization $\mathcal{T}_h$, a discretization of equation (33) at the mesh node s_i is obtained by integrating (33) over the volume C_i, that is built around the node s_i by joining barycenters of the tetrahedra and triangles containing s_i and midpoints of the edges adjacent to s_i:

$$Vol_i \frac{\partial W_i}{\partial t} + \sum_{j \in N(i)} \Phi(W_i, W_j, \overrightarrow{\sigma}_{ij}) = 0, \tag{36}$$

where W_i represents the cell averaged state and Vol_i the volume of the cell C_i. $N(i)$ is the set of the neighboring nodes. $\Phi(W_i, W_j, \overrightarrow{\sigma}_{ij})$ is an approximation of the integral of the fluxes (34) over the boundary ∂C_{ij} between C_i and C_j, which depends on W_i, W_j and $\overrightarrow{\sigma}_{ij}$ the integral of a unit normal vector over ∂C_{ij}. These numerical fluxes are evaluated using upwinding, according to the approximate Riemann solver of Roe:

$$\begin{aligned} \Phi(W_i, W_j, \overrightarrow{\sigma}_{ij}) = & \frac{\overrightarrow{F}(W_i) + \overrightarrow{F}(W_j)}{2} \cdot \overrightarrow{\sigma}_{ij} \\ & - |A_R(W_i, W_j, \overrightarrow{\sigma}_{ij})| \frac{W_j - W_i}{2}. \end{aligned} \tag{37}$$

A_R is the jacobian matrix of the fluxes for the Roe average state and verifies:

$$A_R(W_i, W_j, \overrightarrow{\sigma}_{ij})(W_j - W_i) = (\overrightarrow{F}(W_j) - \overrightarrow{F}(W_i)) \cdot \overrightarrow{\sigma}_{ij}. \tag{38}$$

A high order scheme is obtained by interpolating linearly the physical variables from s_i to the midpoint of $[s_i s_j]$, before equation (37) is employed to evaluate the fluxes. Nodal gradients are obtained from a weighting average of the P1 Galerkin gradients computed on each tetrahedron containing s_i. In order to avoid spurious oscillations of the solution in the vicinity of the shock, a slope limitation procedure using the Van-Leer limiter is introduced. The resulting discretization scheme exhibits a high-order accuracy in the regions where the solution is regular.

Time integration

A first order implicit backward scheme is employed for the time integration of (36), which yields :

$$\frac{Vol_i}{\Delta t} \delta W_i + \sum_{j \in N(i)} \Phi(W_i^{n+1}, W_j^{n+1}, \overrightarrow{\sigma}_{ij}) = 0, \tag{39}$$

with $\delta W_i = W_i^{n+1} - W_i^n$. Then, the linearization of the numerical fluxes provides the following integration scheme :

$$\left(\frac{Vol_i}{\Delta t} + J_i^n\right) \delta W_i = - \sum_{j \in N(i)} \Phi(W_i^n, W_j^n, \overrightarrow{\sigma}_{ij}). \tag{40}$$

Here, J_i^n is the Jacobian matrix of the first order numerical fluxes, whereas the right hand side of (40) is evaluated using high order approximations. The resulting integration scheme provides a high order solution of the steady problem.

APPLICATION TO SUBSONIC FLOW

Testcase description

The testcase considered here corresponds to the flow around the wing of a business aircraft (courtesy of Piaggio Aero Ind.), for a subsonic regime. The nominal operating conditions are defined by the free-stream Mach number

$M_\infty = 0.65$ and the incidence $\alpha = 2°$. The wing section is supposed to correspond to the NACA 0012 airfoil. The wing shape and the mesh in the symmetry plane are depicted in figure (1). The mesh employed counts 31124 nodes.

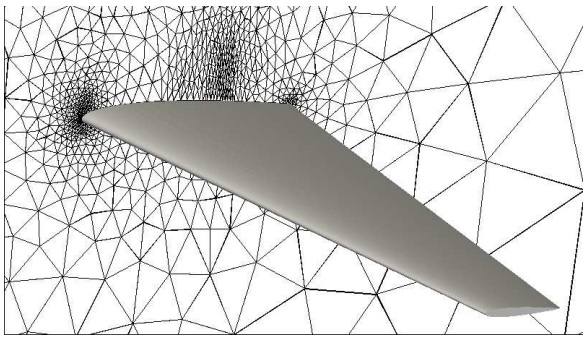

Figure 1: Wing shape and mesh in the symmetry plane.

We suppose that the free-stream Mach number and the angle of attack are subject to random fluctuations. For the sake of simplicity, we assume that their PDFs are Gaussian and uncorrelated. They are characterized by :

	Mach	Incidence (deg.)
Mean	0.65	2
Standard deviation	0.01666	0.33333

We aim at using the methods presented above to estimate the statistics on the drag.

Reference results

We compute first some reference results, obtained by performing 21×21 CFD analyses as seen in figure (2):

Reference expectation	$6.857\,10^{-3}$
Reference variance	$1.553\,10^{-7}$

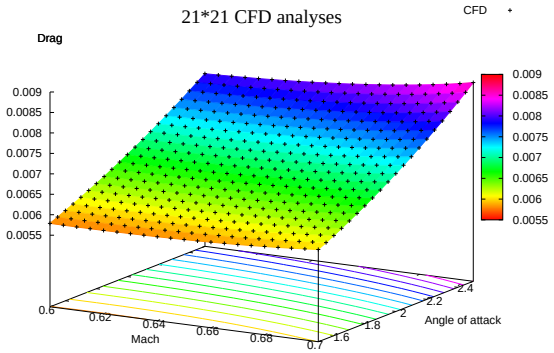

Figure 2: Drag for 21×21 CFD analyses.

Results with meta-models (regular grid)

We construct RBF and kriging meta-models for different database sizes that correspond to regular grids 3×3, 4×4 and 5×5. Then, Monte-Carlo simulations based on these meta-models are performed. The results in terms of drag expectation, variance and errors with respect to reference results are summarized in the following tables:

Metamodel	Points	Expectation	Expectation error
RBF	4	$7.09309\,10^{-3}$	$2.35579\,10^{-4}$
RBF	9	$6.85813\,10^{-3}$	$6.21914\,10^{-7}$
RBF	25	$6.85756\,10^{-3}$	$5.03367\,10^{-8}$
KRG	4	$7.09309\,10^{-3}$	$2.35579\,10^{-4}$
KRG	9	$6.85932\,10^{-3}$	$1.80872\,10^{-6}$
KRG	25	$6.85756\,10^{-3}$	$5.44201\,10^{-8}$

Metamodel	Points	Variance	Variance error
RBF	4	$1.83171\,10^{-7}$	$2.78367\,10^{-8}$
RBF	9	$1.57570\,10^{-7}$	$2.23580\,10^{-9}$
RBF	25	$1.56382\,10^{-7}$	$1.04780\,10^{-9}$
KRG	4	$8.08668\,10^{-10}$	$1.54525\,10^{-7}$
KRG	9	$1.77749\,10^{-7}$	$2.24148\,10^{-8}$
KRG	25	$1.56307\,10^{-7}$	$9.72773\,10^{-10}$

As can be seen, results converge quickly as the size of the database increases.

Results with meta-models (LHS)

Then we construct RBF and kriging meta-models for databases generated by latin hypercube sampling. The databases also include the corners of the variation domain. Results are slighty less accurate for the mean estimate, but not for the variance estimate. Anyway, the difference is not significative for practical applications. Figure (3) shows the drag evolution with respect to the uncertain parameters obtained for a RBF meta-model with 8 points, as well as the error computed at 21×21 points. As seen, the error on the drag is less than 0.5%.

Metamodel	Points	Expectation	Expectation error
RBF	8	$6.85507\,10^{-3}$	$2.43491\,10^{-6}$
RBF	23	$6.85762\,10^{-3}$	$1.05410\,10^{-7}$
KRG	8	$6.85257\,10^{-3}$	$4.94454\,10^{-6}$
KRG	23	$6.85762\,10^{-3}$	$1.14150\,10^{-7}$

Metamodel	Points	Variance	Variance error
RBF	8	$1.56176\,10^{-7}$	$8.41631\,10^{-10}$
RBF	23	$1.56595\,10^{-7}$	$1.26098\,10^{-9}$
KRG	8	$1.55076\,10^{-7}$	$2.57555\,10^{-10}$
KRG	23	$1.56575\,10^{-7}$	$1.24084\,10^{-9}$

Results with AD

AD is then used to estimate drag statistics. Contrary to the previous case, the flow is only computed at the mean values of the uncertain parameters, as well as the derivatives. The statistics obtained using first- and second-order Taylor series are given in the next tables:

	Expectation	Expectation error
First-order	$6.83049\,10^{-3}$	$2.70266\,10^{-5}$
Second-order	$6.86056\,10^{-3}$	$3.04649\,10^{-6}$

	Variance	Variance error
First-order	$1.54665\,10^{-7}$	$6.69276\,10^{-10}$
Second-order	$1.55758\,10^{-7}$	$4.23060\,10^{-10}$

The accuracy of the results is similar to the one obtained with metamodels with 8 or 9 points. Figures (4) and (5) show the drag evolution with respect to the uncertain parameters obtained using AD, as well as the error computed at 21×21

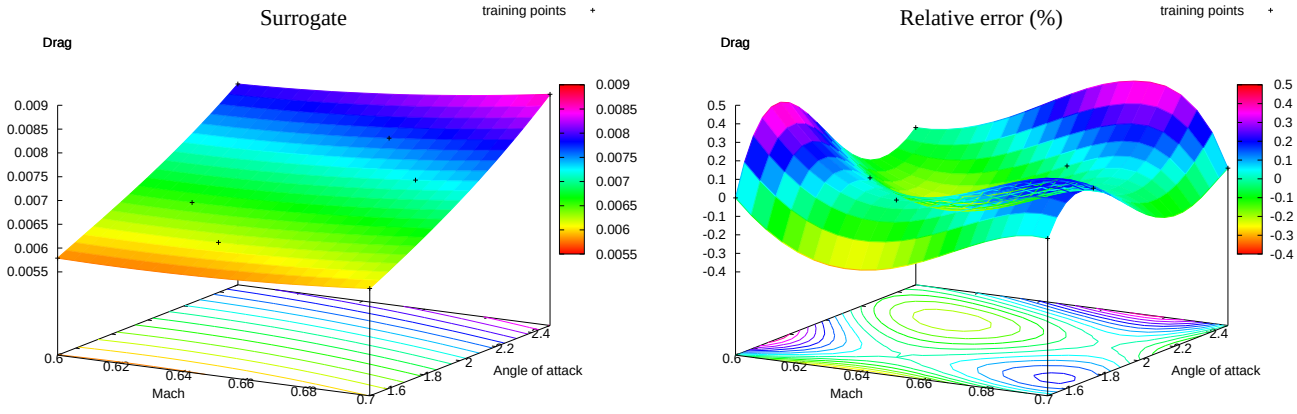

Figure 3: Drag value and error in % using a RBF meta-model with 8 points (LHS sampling).

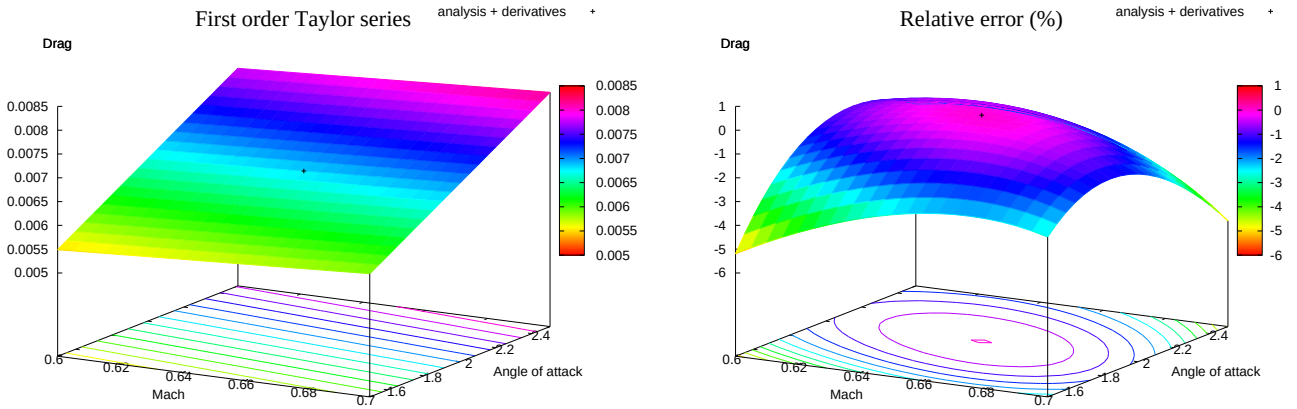

Figure 4: Drag value and error in % using AD (first-order).

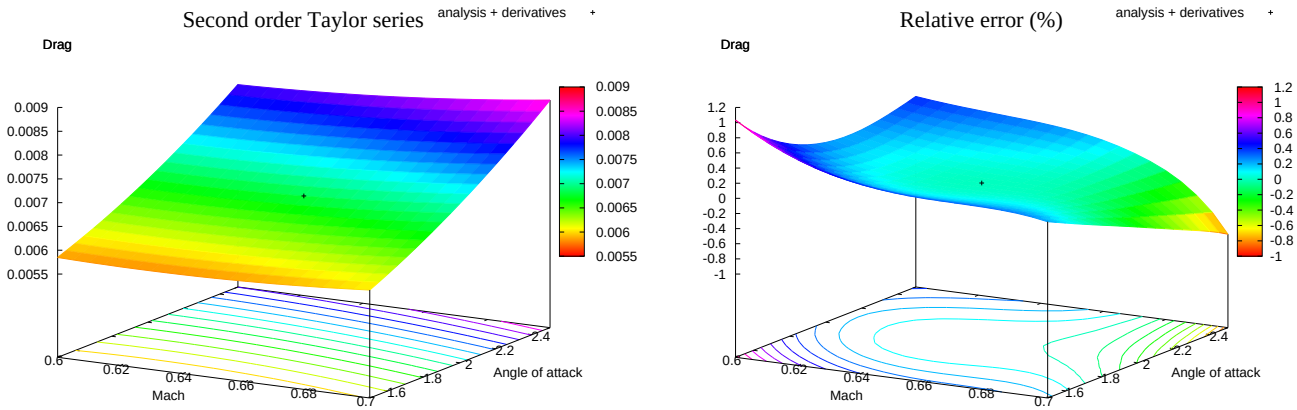

Figure 5: Drag value and error in % using AD (second-order).

points. One can observe that the error is larger at the corners than using meta-models. However, this is not critical for statistics estimation, since the PDFs of uncertain parameters become smaller and smaller as one moves away from nominal operational conditions.

Comparison of computational performance

Since the two proposed methods are essentially different, it is interesting to compare also their computational performance, in terms of CPU time and memory requirements. The following table details the memory used by the AD-based approach:

	Memory in Mb
Flow solver	130
First derivatives	250
Second derivatives	120
GMRES linear solver	250
Preconditionners	340
Total	1090

As seen, the computation of the flow solution with the first and second derivatives requires about 10 times more memory than the flow solution alone. However, this result can be improved using dynamic memory allocation or advanced programming tricks.

The computational time required is given in the next table:

	CPU time in second
Flow solver	403
Gradient	255
Hessian	278
Total	936

These results are obtained with an Intel Xeon 2.66 GHz. The AD-based approach is particularly efficient in this case, since the CPU time only increases about twice to obtain the gradient and Hessian required to compute the statistics.

Concerning the method based on meta-models, the costs are mainly related to the construction of the database. If it is built sequentially, the memory required is the same as the one used by the flow solver alone, whereas the CPU time increases linearly. If it is built using parallel computing, with a number of processors equal to the database size, the CPU time remains more or less similar to a single flow solver run. For instance, we obtain for a database with 8 points:

	CPU time in second
Sequential database	3250
Parallel database	440

In conclusion, the method based on meta-models is more expensive in terms of CPU time, except if one has the capability to build it using parallel computing.

CURRENT CHALLENGES

The results presented above are promising, but one should underline that the problem considered here remains quite simple. Indeed, only two uncertain parameters are under consideration. If this number increases, the CPU time and memory requirements will increase more than linearly, whatever the method considered. Therefore, the current challenge is to be able to manage a large number of uncertain parameters. To succeed, AD tools should clearly improve their memory management, whereas meta-models should improve

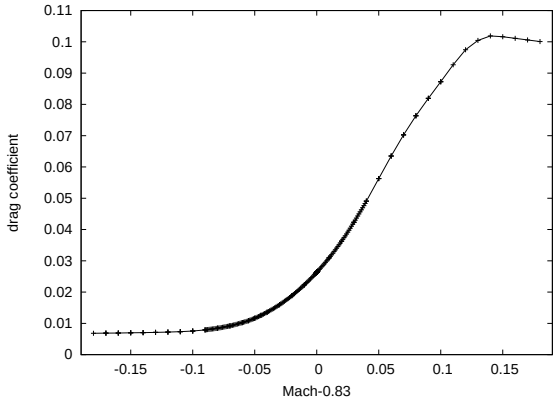

Figure 6: Drag with respect to Mach number in transonic regime.

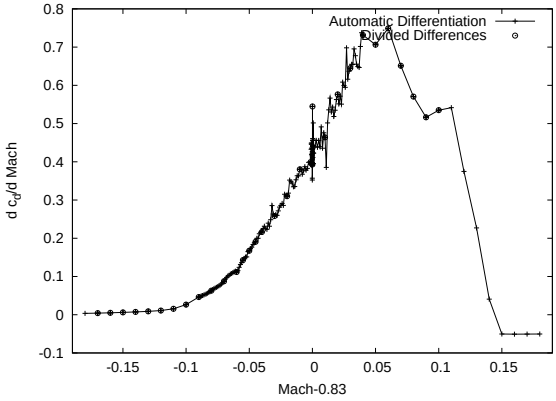

Figure 7: Drag derivatives with respect to Mach number in transonic regime.

their accuracy when using small databases in large dimension spaces.

AD is also faced to another difficulty, arising from non-differentiable programs. For instance, if one considers the previous problem in transonic regime, one observes a noisy gradient computation, although the drag seems to be smooth (see figure (6) and (7)). This result is due to the presence of limiters in the fluxes computation, that are not differentiable at some points. This difficulty can be overcome by modifying the limiters, but it can be a tedious task.

CONCLUSION

In this paper, we have presented and compared two methods to estimate drag uncertainty for practical testcases: automatic differentiation has been applied to a 3D Eulerian flow solver to compute first and second derivatives of the drag with respect to uncertain parameters, in the framework of the method of moments. Metamodels have been used in conjunction with Monte-Carlo simulations to estimate statistics.

Satisfactory results in terms of accuracy have been obtained for a rather simple testcase. However, these methodologies still need to mature in order to be able to manage a large number of uncertain parameters.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the scientific commitee of IDRIS (project 72906) and CINES (project sop2703) for the attribution of CPU time.

This study has been partially supported by the RNTL-ANR “OMD” project.

This work was partially supported by the project NODESIM-CFD "Non-Deterministic Simulation for CFD-based Design Methodologies" funded by the European Community represented by the CEC, Research Directorate-General, in the 6th Framework Programme, under Contract No. AST5-CT-2006-030959.

References

[1] Guide for the verification and validation of Computational Fluid Dynamics simulation. AIAA guide G-077-1998, 1998.

[2] M.D. Buhmann. Radial basis functions. *Acta Numerica*, 9:1–38, 2000.

[3] R. Duvigneau and D. Pelletier. A sensitivity equation method for fast evaluation of nearby flows and uncertainty analysis for shape parameters. *Int. J. of Computational Fluid Dynamics*, 20(7):497–512, August 2006.

[4] D. Ghate and M. B. Giles. *Inexpensive Monte Carlo uncertainty analysis*, pages 203–210. Recent Trends in Aerospace Design and Optimization. Tata McGraw-Hill, New Delhi, 2006.

[5] D. Ghate and M. B. Giles. Efficient Hessian Calculation Using Automatic Differentiation. Number 2007-4059. AIAA, June 2007. 25th Applied Aerodynamics Conference, Miami (Florida).

[6] L. Hascoët and V. Pascual. TAPENADE 2.1 user's guide. Technical Report 0300, INRIA, Sep 2004.

[7] L. Huyse. Free-form airfoil shape optimization under uncertainty using maximum expected value and second order second moment strategies. Technical Report 2001–18, ICASE, June 2001.

[8] A.J. Keane and P.B. Nair. *Computational Approaches for Aerospace Design: The Pursuit of Excellence*. John-Wiley and Sons, 2005.

[9] O.M. Knio and O.P. Le Maitre. Uncertainty propagation in CFD using polynomial chaos decomposition. *Fluid Dynamics Research*, 38(9):616–640, September 2006.

[10] D. Mackay. 'gaussian processes: A replacement for supervised neural networks ? Lecture notes for a tutorial at NIPS 1997, 1997. www.inference.phy.cam.ac.uk/mackay/BayesGP.html.

[11] M Martinelli, A. Dervieux, and L. Hascoët. Strategies for computing second-order derivatives in CFD design problems. In *Proceedings of WEHSFF2007*, 2007.

[12] H. Nakayama, M. Arakawa, and R. Sasaki. A computational intelligence approach to optimization with unknown objective functions. In *ICANN 2001 : international conference on artificial neural networks, Vienna , Austria*, pages 73–80, 2001.

[13] M.J.D. Powell. Radial basis function methods for interpolation to functions of many variables. In *Fifth hellenic-European conference on Computer Mathematics and its applications*, 2001.

[14] M.M. Putko, P.A. Newman, A.C. Taylor, and L.L. Green. Approach for uncertainty propagation and robust design in cfd using sensitivity derivatives. In *15th AIAA Computational Fluid Dynamics Conference*, Anaheim, CA, June 2001. AIAA Paper 2001-2528.

[15] M. Riedmiller. Advanced supervised learning in multi-layer perceptron - from backpropagation to adaptative learning algorithms. *Computer standards and interfaces*, (5), 1994.

[16] S. Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advanced Computational Mathematics*, 11, 1999.

[17] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

[18] L. L. Sherman, A. C. Taylor III, L. L. Green, and P. A. Newman. First and second-order aerodynamic sensitivity derivatives via automatic differentiation with incremental iterative methods. *Journal of Computational Physics*, 129:307–331, 1996.

[19] A. C. Taylor, L. L. Green, P. A. Newman, and M. M. Putko. Some advanced concepts in discrete aerodynamic sensitivity analysis. *AIAA Journal*, 41(7):1224–1229, 2003.

[20] É. Turgeon, D. Pelletier, and J. Borggaard. Sensitivity and uncertainty analysis for variable property flows. In *39th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, Jan. 2001. AIAA Paper 2001-0139.

[21] R.W. Walter and L. Huyse. Uncertainty analysis for fluid mechanics with applications. Technical Report 2002–1, ICASE, February 2002.