# A code-independent technique for computational verification of fluid mechanics and heat transfer problems

M. Garbey † and C. Picard †

† Department of Computer Science, University of Houston, Houston, TX 77204, USA

## 1 Introduction

This paper addresses the challenge of Solution Verification (SV) and accuracy assessment for computing complex Partial Differential Equation (PDE) model. Our goal is to provide a postprocessing software infrastructure that can be connected to any existing numerical simulation software, for example, widely used commercial applications such as ADINA, Ansys, Fluent, Numeca, Star-CD etc... and provide an *a posteriori* error estimate to their simulation. Important design decisions are based on simulation done with these software. Unfortunately, we know that to verify a numerical solution, that is to provide a quantitative assessment on the numerical accuracy of the solution, is difficult.

The problem of accuracy assessment is a necessary step that should be treated between the code verification step and the code validation step to complete the global task of providing a reliable virtual experiment tool [1, 2].

Our major goal in this paper is to pursue our work on the design of a new method that offers a general framework to do solution verification efficiently [3, 4, 5]. The standard approach in applied mathematics to handle the problem of solution verification is to work on the approximation theory of the PDE. For each specific PDE problem, the right Finite Element (FE) approximation may provide the correct *a posteriori* error estimate [6]. Unfortunately, this approach may require a complete rewriting of an existing Computational Fluid Dynamic (CFD) application based on Finite Volume (FV) for example, and lack generality. Usually *a posteriori* estimators fails if the (nonlinear) PDE solution is stiff or if the grid resolution is not adequate. Since grid refinement itself is based on *a posteriori* estimator, this leads to an obvious problem. Large Reynolds number flow are common in many applications, including turbulence problems. For those applications rigorous solution verification may not be achievable by the current state of the art of numerical analysis. The difficulty of SV is even greater for complex multi-physic coupling. The general practice in scientific computing is to simulate PDEs, for which neither applied mathematics, nor numerical analysis, guaranty the result.

Because of the time lag between the development of rigorous mathematical tools and the common scientific computing practice, our goal is to improve existing SV tools such as the convergence index of Roache et Al, and the Richardson Extrapolation (RE) technique [7, 8], that are used daily by practitioner, by something more elaborate and

reliable that can take both advantage of existing *a posteriori* estimators when they are available, and new distributed computing tools since SV is computer intensive.

Our method for *a posteriori* error estimate relies on four main ideas that are :

- Reformulate the problem: in place of solving the problem of error estimation at once, it is embedded in an optimum design framework. This latter will determine the best possible solution form a reduced set of two to three existing numerical results of a same problem produced by an independent simulation tool.
- Independence form approximation theory framework: in most of the case, detailed knowledge of the approximation theory framework used to modelize the PDE's and produce the numerical solutions is at best limited and in the worst case completely inaccessible. A better solution will be to permit the processing of the underlying set of discrete (non) linear equations without using *a priori* information on the approximation theory framework.
- Integration with available *a posteriori* estimators : industrial and research lab simulation tools might already make use of some form of error estimators. They have been tested and proved to be efficient. The optimization procedure of our framework should enable the use of this existing estimators when possible.
- Support distributed computing : like most of the software involving numerical computations, error estimation can be a fairly expensive procedure in term of computer operations and time. As a mean of decreasing the time cost, the calculation should be performed in a distributed environment when computers are in their idle state.

The plan of this paper is as follows: Section 2 describes the general method; Section 3 gives some examples with two benchmark problems; Section 4 gives a description of our postprocessing module; Section 5 discusses some limits of the method and the conclusion.

# 2 Method

We describe in the following the main ideas without seeking an exact formal mathematical description of a given specific PDE problem.

We consider a boundary value problem ($\Omega$ is a polygonal domain and $n = 2$ or 3) :

$$N[U(x)] = S(x),\ x \in \Omega \subset \mathbb{R}^n,\ U = g \text{ on } \partial\Omega. \tag{1}$$

We assume that the PDE problem is well posed and has a unique smooth solution. Let $(E, ||.||_E)$ and $(F, ||.||_F)$ be two normed linear space, $N_h : E \to F$ be the operator corresponding to the problem solved by the code. It can be a finite volume approximation of (1) on a family of meshes $M(h)$ parametrized by $h > 0$ a small parameter. In practice we look for an approximation of the accuracy of the solution $U_h$ on the mesh $M(h)$ produced by the code $\mathcal{C}$ that operates on the data $S_h$:

$$\mathcal{C} :\ S_h\ \to U_h$$

The smaller $h$, the finer should be the discretization.
Let $p_h$ denotes the projection of the continuous solution $U$ onto the mesh $M(h)$. We assume *a priori*:

$$||U_h - p_h(U)||_E \ \rightarrow \ 0, \text{ as } h \ \rightarrow \ 0, \tag{2}$$

We assume, therefore, that the code has been verified, and that the convergence to the exact continuous solution is satisfied.

The objective is to verify the solution produce by the code, not the code itself. We will get an error estimate versus a very fine grid solution $U_\infty$ that is never computed, because the cost is prohibitive. We will skip the index $h$ when it is not essential. The space $E$, $F$ have in practice (very large) finite dimensions when they are for the discrete solutions on $M(h_\infty)$, and discrete data $S_{h_\infty}$.

We assume that the code $\mathcal{C}$ has a procedure that provides the residual, i.e $V \rightarrow \rho = N(U_h) - N(V)$, where $V \in E$, $\rho \in F$. We note that most of the commercial code offer this feature or either provides a (first order explicit) time stepping procedure:

$$\frac{U_h^{n+1} - U_h^n}{dt} = N(U_h^n) - S, \tag{3}$$

The residual is then $\rho = \frac{U_h^1 - U_h}{dt}$. We assume that the following problem

$$N(u) \ = \ s, \ \forall s \in B(S,d).$$

is well posed for $s \in B(S,r)$, where $B$ is a ball of center $S$ and radius $r$ in $(F, ||.||_F)$. There should exist a unique solution for all data in $B(S,r)$ and the dependency of the solution on these data is supposed to be smooth enough to use a second order Taylor expansion.

Let us suppose that $N(U_h) \in B(S,r)$, that is

$$||\rho||_F = ||N(U_h) - S||_F < d. \tag{4}$$

We would like to get an error estimate on $e = U_h - U_\infty = \mathcal{C}(S+\rho) - \mathcal{C}(S)$. A Taylor expansion writes

$$\begin{aligned} \mathcal{C}(S) = \mathcal{C}(S+\rho) - (\rho \cdot \nabla_s)\mathcal{C}(S+\rho) \\ + \frac{1}{2}\rho \cdot [\rho \cdot R(S)] \\ \text{where } ||R(S)||_E \leq K = \sup_{s \in B(S,d)} ||\nabla_s^2 \mathcal{C}(s)||_E. \end{aligned} \tag{5}$$

Therefore

$$||e||_E \ \leq \ ||\rho||_F \ (||\nabla_s \mathcal{C}(S+\rho)||_E \ + \ \frac{K}{2}||\rho||_F). \tag{6}$$

This completely general error estimate point out to two different tasks:

• Task 1: compute an accurate upper bound on $||\nabla_S \mathcal{C}(S+\rho)||$

• Task 2: obtain a solution $U_\infty + e$ that gives a residual $||\rho||$ small enough to make the estimate useful, i.e compatible with (4).

Task 2 is the purpose of the Optimized Extrapolation Solution (OES) method, while Task 1 can be achieved by a sensitivity analysis of $\mathcal{C}$.

## 2.1 Task 1: Stability Estimate:

Let $\{b_i^E,\ i = 1..N\}$, (respt. $\{b_i^F,\ i = 1..N\}$) be a basis of $E_h$, (respt. $F_h$) and $\varepsilon \in I\!R$ such that $\varepsilon = o(1)$. Let $(V_i^{\mp})_{i=1..N}$, be the family of solutions of the following problems:

$$N(U_h \mp \varepsilon V_i) = S + \rho \mp \varepsilon b_i$$

We get from finite differences the approximation

$$\begin{aligned} C_{h_\infty} &= ||\nabla_S \mathcal{C}(S+\rho)|| \\ &\approx ||(\frac{1}{2}(V_j^+ - V_j^-))_{j=1..N}|| + O(\varepsilon^2). \end{aligned} \tag{7}$$

We can get in a similar manner an approximation of the norm of the Hessian $\nabla_s^2 \mathcal{C}(S+\rho)$. For $\rho$ small enough, we can verify that the upper bound is given at first approximation by:

$$||e||_E \ \leq \ C_{h_\infty} ||\rho||_F. \tag{8}$$

The column vectors $V_j^{\mp}$ can be computed with embarrassing parallelism. It is, however, unrealistic to compute these solutions that lies on the fine grid $M(h_\infty)$.

To make this task manageable, we have to reduce the dimension of the problem. We use the following two observations: while the solution of a CFD problem can be very much grid dependent, the conditioning number of the problem is in general much less sensitive to the grid. The idea is then to compute an approximation of $C_{h_\infty}$ by extrapolation from an estimate of two or three coarse grid computation of $C_{h_j}$. Further, let us assume that the fine grid $M(h_\infty)$ is a regular Cartesian grid. The number of terms to represent accurately the projected solution $\tilde{U}_j,\ j = 1..3$ with a spectral expansion or a wavelet approximation at a given accuracy is much less than the dimension of the coarse grid used in a Finite Element/Finite Volume computation. We propose to use preferably a grid $M_{h\infty}$ that has enough regularity to allow a representation of the solution $U_\infty$ with some form of reduced representation, using either trigonometric expansion or wavelets.

The grid $M_{h_\infty}$ may have many more grid points than necessary. Therefore, it might not be computationally efficient to perform a true fine grid computation, but we do not have to do this computation anyway.

Further, regular grids are far more easy to construct. If for some reasons $M_{h\infty}$ has to be unstructured, we can also use spectral elements. An ideal method might be to use a proper orthogonal decomposition that captures the main feature of the solution [9].

Let us denote $\hat{E}$ and $\hat{F}$ the spaces corresponding to one of these compact representation of the solution and residual. Let $(\hat{b}_j^{E/F}, j = 1..\hat{N})$, be the corresponding base with $\hat{N} \ll N$. Let $\hat{q}_{E/F}$ be a mapping $E/F \rightarrow \hat{E}/\hat{F}$, respectively $q_{\hat{E}/\hat{F}}$ be a mapping $\hat{E}/\hat{F} \rightarrow E/F$, and let $\hat{\mathcal{C}}: \ \hat{S}_h \rightarrow \hat{U}_h$ be the code that uses this postprocessing of the residual and solution.

Figure 1 illustrates the relation between the different discrete spaces, along with the corresponding mappings. The mapping $\hat{q}_E$ can be a least square approximation of the solution $u$ into $\hat{E}$, acting as a filter on the solution, while $q_{\hat{E}}$ is a projection onto $E$.

The construction of $q_E$ and $\hat{q}_E$, respectively $q_F$ and $\hat{q}_F$ does not consider the nature of the true approximation space used in the code $\mathcal{C}$ since the implementation details are

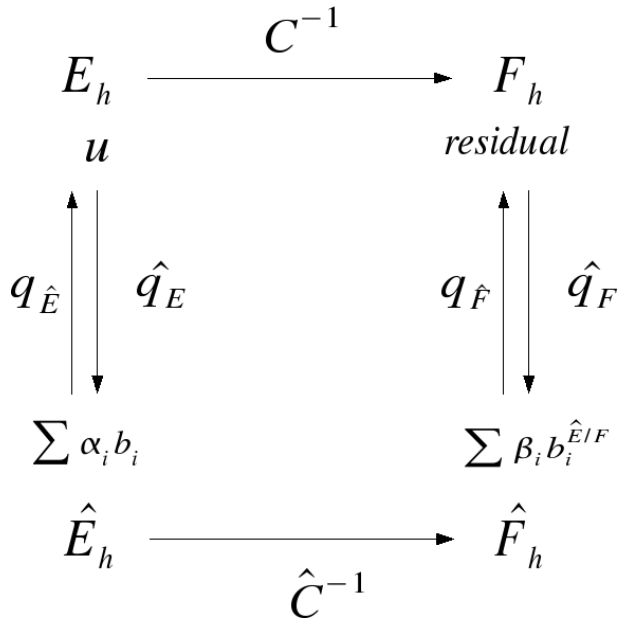


Figure 1: Mappings and vector spaces.

most of the time unavailable: the mappings involve only the discrete representations of the functions.

To summarize the procedure for Task 1, the estimate on $\mathcal{C}_{h_\infty}$ will be applied to verify the code $\hat{\mathcal{C}}$ based on the computation of $(\hat{V}_j^{\mp}, j = 1..\hat{N})$ vectors on the coarse grids $M(h_j),\ j = 1..3$ done by the code $\hat{\mathcal{C}}$. We notice that the computation of the vector $\hat{V}_j^{\mp}$ can be done with embarrassing parallelism. Further, because $\varepsilon$ is small the code $\hat{\mathcal{C}}$ can use as an initial guess in its iterative process the solution $U_h$ that is hopefully very close to the unknown $\hat{U}_h \pm \epsilon \hat{V}_j^{\mp}$.

There are several issues that needs to be carefully investigated when applying this procedure. Let us mention two of them. First, our method assumes that the spectral properties of the operator follows some fairly regular asymptotic properties for the coarse meshes under consideration as $h \to 0$. This hypothesis should be verified and/or may not be true. Second, $\varepsilon$ must be chosen carefully as a function of the mesh size, in order to avoid a dramatic inaccuracy on the stability estimate of $C_{h_\infty}$. We recall, however, that we are only looking for an order of magnitude of $C_{h_\infty}$ and not its true value.

Let us discuss our second task that is to compute a solution on the fine grid that is good enough to recover an error estimate.

## 2.2 Task 2: Optimized Extrapolation :

Let $M(h_1)$ and $M(h_2)$ be two different meshes used to build two approximations $U_1$ and $U_2$ of the PDE problem (1). A consistent linear extrapolation formula should have the form

$$\alpha U_1 + (1-\alpha)U_2,$$

where $\alpha$ is a weight function. The consistency results from the fact that for carefully chosen $\alpha$, the approximations $U_1$ and $U_2$ can also be expressed with the same relation.

In classical RE the $\alpha$ function is a constant. In our optimized extrapolation method $\alpha$ is an unknown space dependent function solution of the following optimization problem, where $G$ is an objective function to be defined:

$$P_\alpha\text{: } \textit{Find } \alpha \in \Lambda(\Omega) \subset L_\infty \textit{ such that } G(\alpha U_1 + (1-\alpha)U_2) \textit{ is minimum.}$$

The OES, if it exists, is denoted $V_e = \alpha U_1 + (1-\alpha)U_2$.

Whenever $U_1 - U_2 << U - U_2$ in some set of non zero measure, it is necessary to modified the extrapolation formula. Indeed, outliers should not affect globally the least square extrapolation and $\alpha$ is chosen to be a bounded function, independent of the spatial discretization. A potentially more robust approximation procedure is introduce a third approximation $U_3$. The three level problem to be solved is as follows:

$$P_{\alpha,\beta}\text{: } \textit{Find } \alpha, \beta \in \Lambda(\Omega) \subset L_\infty \textit{ such that } G(\alpha U_1 + \beta U_2 + (1-\alpha-\beta)U_3) \textit{ is minimum.}$$

If the three approximations $U_j, j = 1 \ldots 3$ coincide at the same spatial point, the local accuracy cannot be improved using extrapolation method only.

For computational efficiency, $\Lambda(\Omega)$ should be a finite vector space of very small dimension. The objective function $G$ might be derived from any existing *a posteriori* error estimators, if possible. Our ambition is to provide a numerical estimate on $||U_j - U_\infty||$, $j = 1, 2$, without computing $U_\infty$ explicitly. The solution $U_j$ can be verified then assuming (2).

The fine mesh $M(h_\infty)$ should be set such that it captures all the scales of the continuous solution with the level of accuracy required by the application. We have *a priori* $h_\infty << h_1,\ h_2$. Both coarse grid solutions $U_1$ and $U_2$ must be projected onto $M(h_\infty)$. We will denote $\tilde{U}_1$ and $\tilde{U}_2$ the corresponding functions. We choose then to minimize the consistency error for the numerical approximation of (1) on a fine mesh $M(h_\infty)$ . The objective function is then

$$G(U^\alpha) = ||N_{h_\infty}\ U^\alpha - F_{h_\infty}||, \tag{9}$$

where $U^\alpha = \alpha\ \tilde{U}_1\ +\ (1-\alpha)\ \tilde{U}_2$.

To reduce the dimension of this problem we search for the unknown weight functions in a small space that can be described either by trigonometric expansion, wavelet expansion, or possibly spectral elements. If $\Omega$ is the physical domain for the CFD solution, the unknown weight function can be extended to a square domain $(a,b)^n$, such that $\Omega \subset (a,b)^n$. As a matter of fact, no boundary conditions are required on the unknown weight functions. Let $\{\theta_j,\ j = 1..m\}$ be the set of basis function of $\Lambda(\Omega)$. We look for the solution of the optimization problem

$$\begin{aligned} &\text{Find } (\alpha_j)_j \in I\!R^m, \text{ such that} \\ &||G([\sum_{j=1..m} \alpha_j \Theta_j]\tilde{U}_1 + [1 - \sum_{j=1..m} \alpha_j \Theta_j]\tilde{U}_2)||_F \\ &\text{is minimum .} \end{aligned} \tag{10}$$

We have a similar formulation for the three level OES described in [3, 4], that is:

$$\begin{aligned}&\text{Find } (\alpha_j)_j(\beta_j)_j \in I\!R^m, \text{ such that}\\ &||G([\sum_{j=1..m}\alpha_j\Theta_j]\tilde{U_1} + \sum_{j=1..m}\beta_j\Theta_j\tilde{U_2} + \sum_{j=1..m}[1-\alpha_j-\beta_j]\Theta_j\tilde{U_3})||_F\\ &\text{is minimum .}\end{aligned} \tag{11}$$

Further, we need a filtering process of the solution to have this minimization procedure numerically efficient. In practice the interpolation of the coarse grid solution to the fine grid $M_{h_\infty}$ introduces spurious high order oscillations that may make the optimization process (10) unreliable. The postprocessing $\hat{q}_F$ regularized the problem. We can obtain easily the result when the weight function is a scalar function. To make this computation robust we use a surface response methodology [10] that is rather trivial in the scalar case. This procedure consist to compute a lower order polynomial best fit of the function $||G(\alpha\tilde{U_1} + (1-\alpha)\tilde{U_2})||$ by sampling $\alpha$ according to the expected convergence order range of the code. The minimization on $\alpha$ is then done on this polynomial approximation by a standard method. The sampling process is a cumbersome embarrassing parallel process that can take advantage of a computational grid [11].

It is, however, impractical when the dimension of the problem for the $\alpha$ search is more than few units. One may use a combination of genetic algorithm and local optimization search to solve the nonlinear optimization problem (10). There is an extensive knowledge and set of available optimization software [12] that we can reuse indeed. We have observed in our experiments that if the solution provided by the CFD code is very coarse, the use of space dependent weight functions might not be required. Similarly, if the solution is very accurate and the code has uniform convergence, we do not need either space dependent weight function. But, in the case of stiff problems we would like to get some adaptivity on the construction of the weight function. This is an open problem that we are currently working on [13]. After exposing the theoretical concepts behind the optimized extrapolation method, we are now going to present different numerical illustrations of this work.

# 3 Applications

We have selected two test cases, one in fluid dynamic and one in heat transfer. Further details on these numerical simulations can be found in [14].

We will carry through our method starting from two coarse grid calculations only. The minimization problem is the following:

$$P_\alpha\text{: } \textit{Find } \alpha \in I\!R \textit{ such that } G(\alpha U_1 + (1-\alpha)U_2) \textit{ is minimum.}$$

For the sake of simplicity, we restrict ourselves to $\alpha$ being a constant function on the domain ($\alpha \in I\!R$).

Both test cases are solved with the finite element commercial package named ADINA. We could have used indeed another commercial package for this demonstration. ADINA is a comprehensive finite element software that enable analysis of structures, fluid

simulations , and fluid flow simulations with structural interactions. More information can be found at http://www.adina.com/.

## 3.1 Backward facing step flow

We are going to consider first a steady incompressible viscous flow in the backward facing step configuration.

The incompressible flow is governed by the following equations

$$\left\{ \begin{array}{rcl} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}.\nabla\mathbf{u} + \nabla\mathbf{p} & = & \nu\Delta\mathbf{u} \\ \nabla\mathbf{u} & = & 0 \end{array} \right. \tag{12}$$

where $\nu$ is the viscosity of the fluid.

Figure 2 shows an example of an unstructured coarse mesh used for the calculation of the backward-facing step flow at Reynolds number 500.

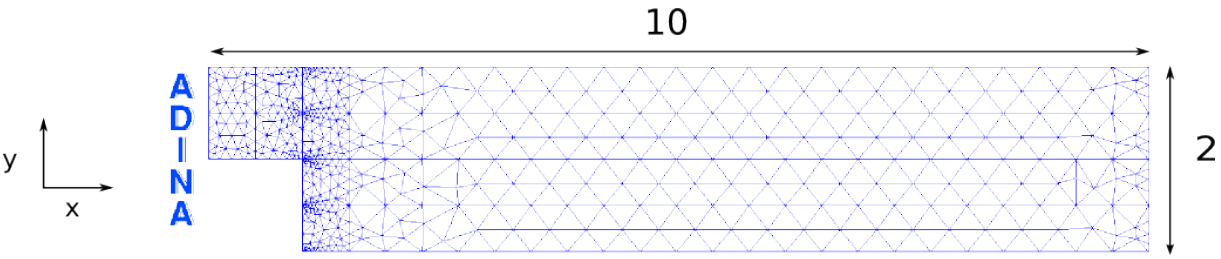


Figure 2: Coarse unstructured mesh for the backward facing step test case generated by Adina.

The length of the cavity is $L_x = 10$, the width is $L_y = 2$, and the size of the step is 1. The inflow is set to be a Poiseuille flow with maximum velocity 1 *U.I.* The outflow is left free. We assume no slip boundary conditions on the walls. In this simulation, the number of quad elements are respectively 10347 on the fine grid $G^\infty$ , 1260 on the coarse grid $G_1$ , and 2630 on the coarse grid $G_2$ .

To provide a rigorous measurement of the error we construct a near by manufactured solution [15] that is within 1/1000 of a very fine grid solution. This procedure forces the RHS and boundary conditions of the back-step problem in such a way that we get an exact solution to NS that is very close to the back step problem. We obtain then an exact measurement of the numerical error for this manufactured solution. The manufactured solutions are constructed from the following set of basis functions

$$\mathbf{u_{k_1}} = \left\{ \begin{array}{c} cos(k_1\pi\frac{x}{L_x})sin(k_1\pi\frac{y}{L_y}) \\ -sin(k_1\pi\frac{x}{L_x})cos(k_1\pi\frac{y}{L_y}) \end{array} \right. \tag{13}$$

that satisfies the divergence free condition and

$$p_{k_2} = cos(k_2\pi\frac{x}{L_x})cos(k_2\pi\frac{y}{L_y}). \tag{14}$$

That is defining a reduced representation space for the solution verification procedure.

Figure 3 shows the evolution of the stability constant as the size of the reduced representation space increase. This results implies that only a limited number of basis functions are needed to represent the numerical solution. Figure 4 gives the error for the OES versus the fine grid solution in $L_2$ norm computed in a reduced space generated by trigonometric functions. The low horizontal line is the true error in $L_2$ norm for the OES obtained previously. One observes that the extrapolation on the stability constant that combines the calculation of the estimate on both coarse grids with the best $\alpha$ obtained in the OES process does improve the accuracy of the error estimate. The fundamental result in this figure is that we can provide an upper bound on the numerical error of the simulation for each coarse grid calculation within 10% of the 'true' error.

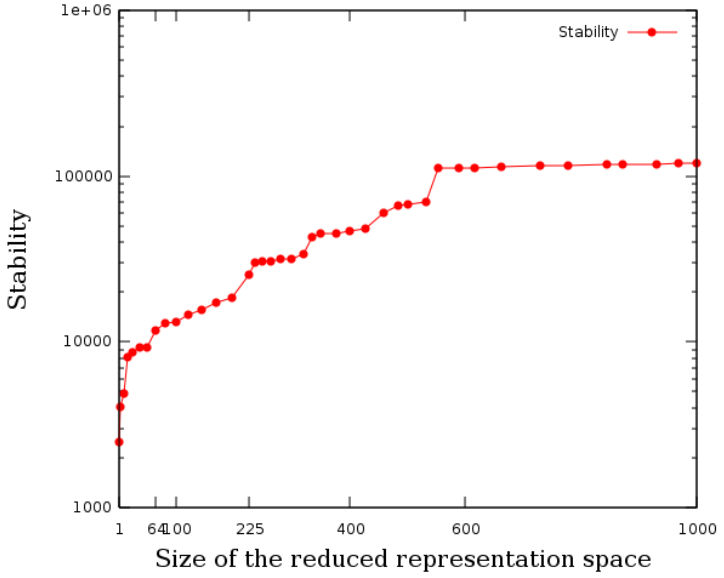


Figure 3: Evaluation of the stability constant for the modified NS problem in $L_2$ norm.

## 3.2 Battery

The model, for our second example is governed by the energy equation:

$$\frac{\partial}{\partial x_i}\left(k_{ij}(T)\frac{\partial T}{\partial x_j}\right) + Q(T) = \rho c_p(T)\frac{\partial T}{\partial t} \quad \text{on } \Omega \times (0,t) \tag{15}$$

with $i$, $j$ running from 1 to 2 for this model. This two dimensional problem is solved in a square domain $\Omega = (0, L_x) \times (0, L_y)$. $T$ is the temperature, $t$ is time, $\rho$ is the material density, $c_p$ is the specific heat as a function of $T$, $Q$ is the volumetric heat source as a function of $t$, and $k_{ij}$ is the thermal conductivity tensor as a function of $T$.

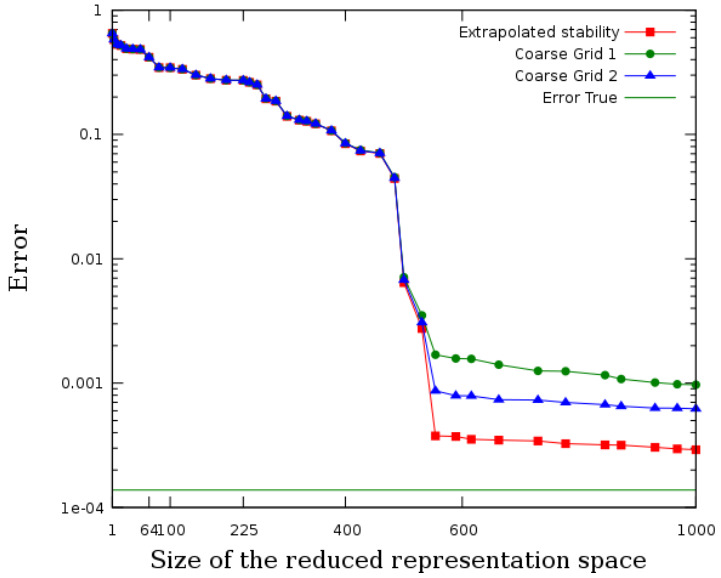


Figure 4: Evolution of the error versus the manufactured solution with the $L_2$ norm for the modified NS problem.

The boundary conditions are :

$$\bullet\ -\left(k_{ij}\frac{\partial T}{\partial x_j}\right)\cdot n = h_1(T)(T-T_\infty) + \sigma\varepsilon_1\left(T^4-T_\infty^4\right) \text{ on } \Gamma_{N_1}$$

(radiation, convection)

$$\bullet\ -\left(k_{ij}\frac{\partial T}{\partial x_j}\right)\cdot n = h_2(T)(T-T_\infty) + \sigma\varepsilon_2\left(T^4-T_\infty^4\right) \text{ on } \Gamma_{N_2}$$

(radiation, convection)

$$\bullet\ -\left(k_{ij}\frac{\partial T}{\partial x_j}\right)\cdot n = h_3(T)(T-T_\infty)$$

on $\Gamma_{N_3}$ (convection)

where $T_\infty = 313.0K$ , $\varepsilon_1 = \varepsilon_2 = 0.25$, $\sigma = 5.670 \times 10^{-8}$ is the Stefan-Boltzmann constant $(W \cdot m^{-2} \cdot K^{-1})$, and $h_3 = 1.0$ $(W \cdot m^{-2} \cdot K^{-1})$. The functions $h_1(T)$, $h_2(T)$, and $c_p$ are given by tables.

The temperature is initialized to $T_0 = 313.0K$ in the structure. The difficulty of this study is due to the fact that the structure is compounded of different materials, for which coefficients might depend on space and temperature, and finally, two regions will undergo a phase change. The problem is therefore very stiff and the solution is almost discontinuous near the wall as shown in Figure 5. To simulate the heat transfer in this structure, we have used quad elements in each physical subdomain. The total number of

elements for the fine grid $G^{\infty}$ is 57258 and the coarse grids $G_1$ and $G_2$ have respectively 8767 and 21072.

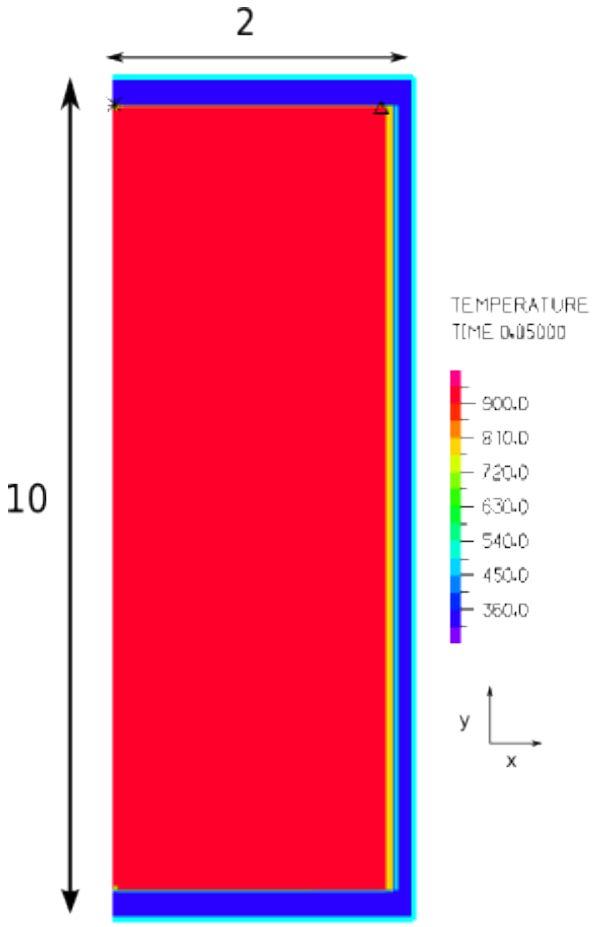


Figure 5: Steady state solution of the heat transfer problem.

Figure 6 gives the numerical error in $L_2$ norm as the function of the number of basis functions used. We need of the order of 400 basis functions to reach a plateau in the error estimate. Once again we obtain an *a posteriori* error estimate on the coarse grid solution within 10% of the 'true' error versus the reference solution on $M(h_{\infty})$. To be more specific, we are able to estimate a solution with an absolute error of 0.2 Kelvin, using coarse grid solutions with absolute errors of 10 Kelvin.

# 4 Software implementation

## 4.1 Algorithm

The algorithm of our method writes in its simplest version

1. *Call coarse Mesh* : generate the (coarse) meshes $M(h_1)$ and $M(h_2)$. If $h_i$ is the average space step for the grid $M(h_i)$ we should have $h_2 < h_1$ but this is not

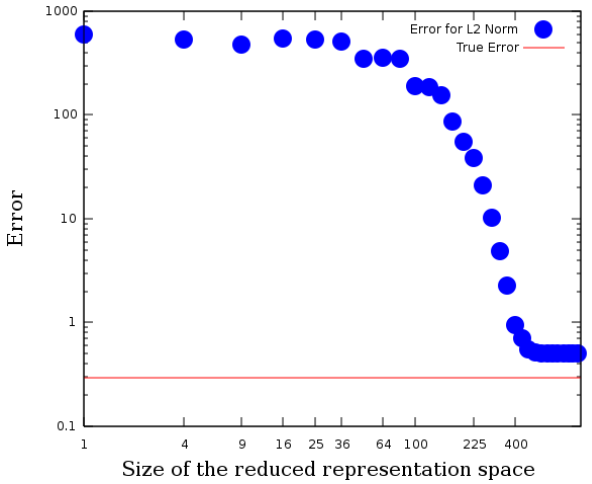


Figure 6: Evolution of the error versus the fine grid solution for the heat transfer problem

necessary.

2. *Call fine Mesh* : generate a fine mesh $M(h_\infty)$ that is supposed to solve all the scales of the problem. $M(h_\infty)$ is preferably a structured mesh. We must have $h_\infty << h_1, h_2$.

3. *Call Solver* : solve the problem on $M(h_1)$ and $M(h_2)$, possibly in parallel.

4. *Call Projection* : project the coarse solutions $U_1$ and $U_2$ onto $M(h_\infty)$ and post-process them to avoid spurious oscillations due to the interpolating function.

5. *Solve minimization problem* : we can create, for example, sample solutions $U_\alpha = [\alpha\tilde{U}_1 + (1-\alpha)\tilde{U}_2]$ and/or use an off the shelf optimization package.

6. *Get Stability Constant*: compute in parallel an increasing set of perturbed solutions $U_h \pm \varepsilon V_i^{\pm}$ until eventually convergence of the stability estimate.

## 4.2 Performance issues

Step 5 and Step 6 of our algorithm are source of large set of cumbersome computations. Looking at figures 3 and 4, one can observed that about 1000 basis computations are required for a proper evaluation. Each of this computation can take several minutes depending on the size of the problem at hand. This computation time take into account the few explicit time step needed to smooth the high frequency components of the projected approximation on the fine grid. Thus a simple sequential implementation might required several days. The key feature that make our solution verification procedure effective is its natural parallelism. Indeed, each of the computation can be done independently of the others since they are not sharing any information. So one can schedule the tasks to be perform in parallel. In order to perform this task efficiently, we need to keep track of which tasks have been completed, and how to distribute the remaining tasks. This information

can be centralized on a master node that will allow a natural load balancing of the tasks among the computational nodes. This is an important element for the distributed computation, because it dynamically adjusts the workload of the computers depending on their availability and the calculation to perform. The last level to take into consideration is the user control interface. Even though the procedure can be parallelized, the execution time might not always be negligible. So the user should be able to check and control remotely the execution of the verification procedure. For this reason, a software interface that does not need to run on the master node, or the computational node has been developed.

Figure 7 reflects the interaction between the components of the system. The idea is to take advantage of a network of computers by limiting the communications to few megabytes of data only, and only once computations are finished. Those computations are done on slave nodes that have knowledge only of the existence of the master node. A computational nexus distributes the task dynamically, self-adjusting to the deficiencies of the network or the availability of the computational nodes. The limiting factor of the process is the number of slave nodes we can access simultaneously, since the computation is virtually scalable. More over, each task in figure 7 make use of a different programming language. The user interface is written in C#. The communications protocol on the master and computational nodes are handled by java code for portability reasons. The preprocessing interfaces that are the core of our method are written in C++. And finally, the computational code, that is the one providing solutions to be verified, can be of any kind.

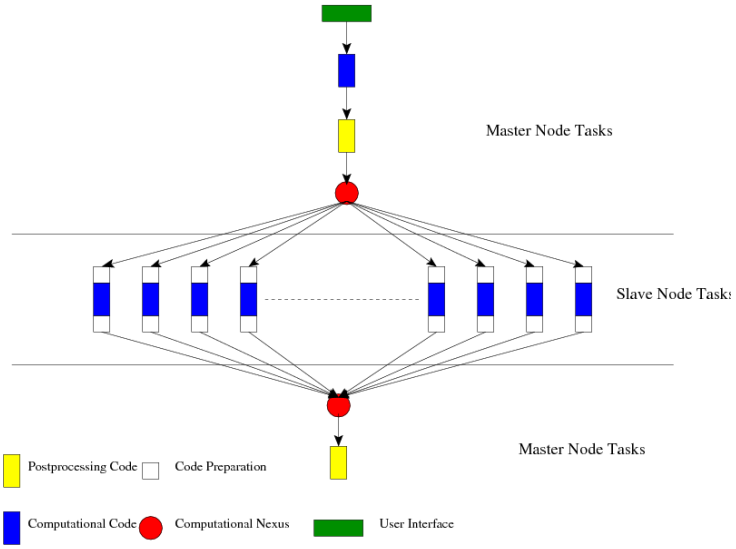


Figure 7: Task distribution for effective distributed computation in OES context

We use a standard three-tier client/server/slave architecture. The *a posteriori* estimate takes then a very small fraction of the time that it would take to compute the fine grid solution. More details on this parallel implementation can be found in [11].

Figure 8 shows the performance improvement for three setups using a three-tier architecture. This first column is for a sequential execution, the second column is for an

heterogeneous network of computers, and the third column is the expected execution time with a parallelized interpolation.

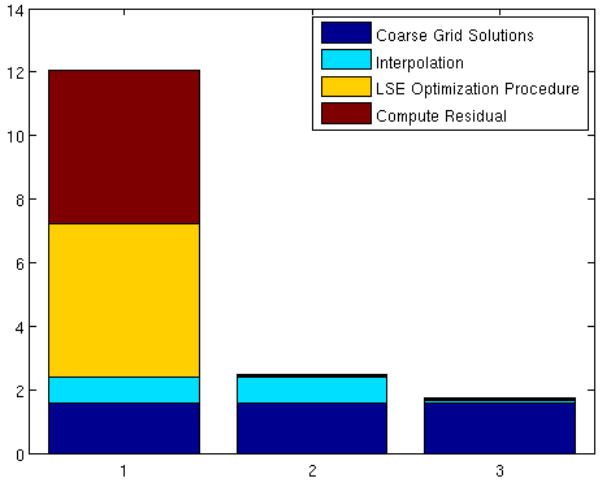


Figure 8: Overview of performance for a simple parallel implementation

This implementation of the solution verification procedure is independent of the operating system or architecture use. The java communication interfaces only requires the Java Runtime Environment to be present on the target machine.

While the architecture of the system is fairly straightforward, one has to be concerned by several other aspects, starting with security and error checking. The data to be exchange between the control interface,the master node, and the computational nodes can have a sensitive purpose, from medical data relative to a patient condition to physical data for DOD projects. It is therefore necessary to pay special attention on how the data is transferred.

## 4.3 Error control

In the three-tier architecture on which our method relies to be cost effective make use of computers and network that can be completely different by nature (but supporting java technology). Also, because of the mix of computational intensive and communication processes, it is important to access what are the problems and which components are involved. And this on each entity. Figure 9 outlines the control points of our software and 10 gives a short description of each element.

## 4.4 Security

The implementation is done using SSL Sockets instead of simple sockets. That means the communication between the different components is encrypted based on SSL public/private keys to generate the 'session key'. The control interface owns the public key, the master node own both the private and the public key and the computational nodes own

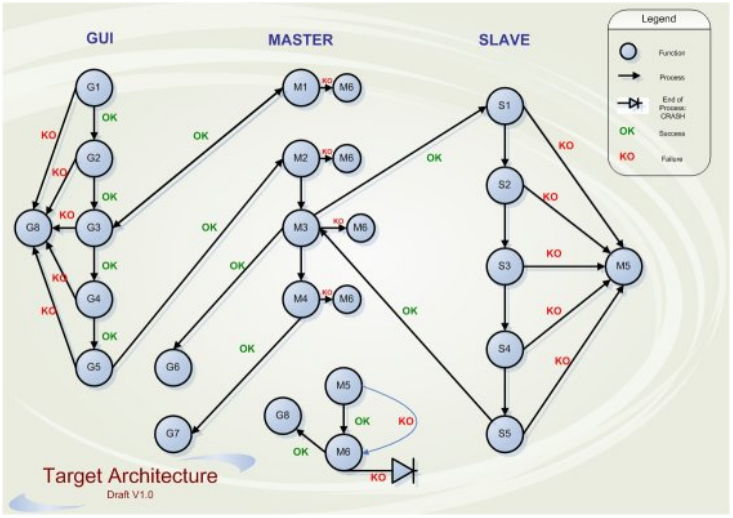


Figure 9: Overview on error control and secure data transfer

the public key. Security issues dictate this distribution of the keys. The question remains of where the keys should be stored.

In the SSL scheme, the communication is secure but any computational node can obtain the public key to exchange information with the master. It is the confidentiality, integrity and authentication concept for data security. Both Confidentiality and Integrity is realized withe the SSL sockets, but there is no control on the authentication. That is why an authentication method has been implemented between the master node and the slave nodes to ensure that only 'allowed slaves' can communicate with the master node.

In our architecture, login and password are encrypted and stored in a file,thus providing the authentication source for the Master. When a computational node wants to communicate with the master, the SSL is initiated, then the master asks the computational node a login and a password. The computational node find there authentication data in an other encrypted file. This file is created either the first time the computational node socket initiates, or when the socket loads and does not find the file in its root directory. The administrator that launches the socket will then be prompted for the login and the password. In the authentication process, if a computational node has a wrong login or a wrong password, the control interface is notified by some data sent on the ssl socket, the SSL tunnel is closed, and a pop-up notifies the user of the console that one or more slaves did not authenticate properly (data displayed on the result graph). Figure 11 shows the communication process between the Master and the Slaves regarding the establishment of the SSL tunnel and the authentication process.

The overhead generated by this control statement and by the SSL sockets encryption is negligible with respect to the time needed to execute a task on the computational node. More specific mechanism should be applied when computational nodes are in cluster with private IP address. One can for example create a new level in the hierarchy by giving some scheduling control to the node of the cluster that is owning a public IP address.

The next section presents our conclusion on the optimized extrapolation method and its implementation using distributed computing.

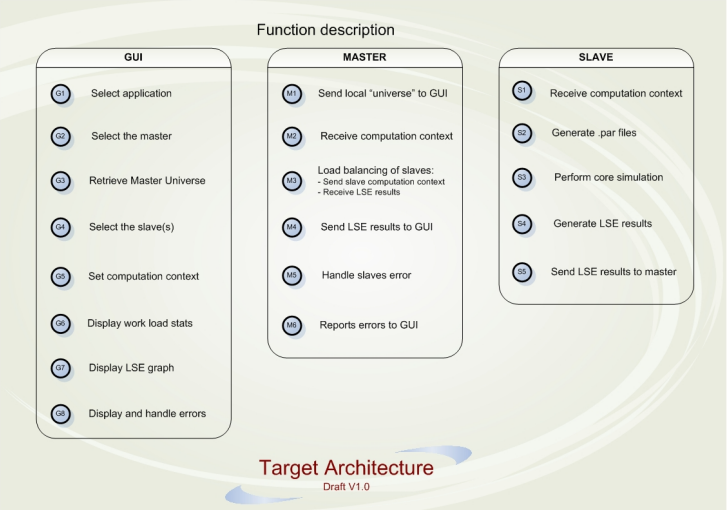


Figure 10: Description of controls

# 5 Discussion and conclusion

In this paper, we have explored a new framework to provide *a posteriori* error estimates for CFD simulations produced with a commercial package. The challenge comes from the fact we do not have access to the source code, neither we may not know precisely what approximation is used. In such context the only method that is used by practitioner is mesh refinement. This process is time consuming and may not even be possible for complex PDE problems. We have proposed an alternative solution that still uses a fine mesh as a reference, but do not require the calculation of the CFD solution on this fine mesh.

There are indeed several limits to our method. Let us mention two of them.

The first drawback of our method is that it is computationally intensive and requires hundreds of evaluations of a residual. On the other hand, this process has embarrassing parallelism and constitute an effective way of using a network of distributed computers. Both OES and sensitivity analysis presented in this paper can be speedup by using state of the art optimizers. This new development is part of our ongoing research.

The second drawback is that if both coarse grid solutions are dramatically inaccurate we should not be able indeed to retrieve an accurate solution from OES. Hopefully, the sensitivity analysis might be able to catch this failure as it is the case, for example, for meta-stable solution with the steady viscous burgers problem.

*Thanks:* The authors would like to thank Guillaume Garbey for is contribution on the security distributed implementation. This research was sponsored by Sandia Nat. Lab. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

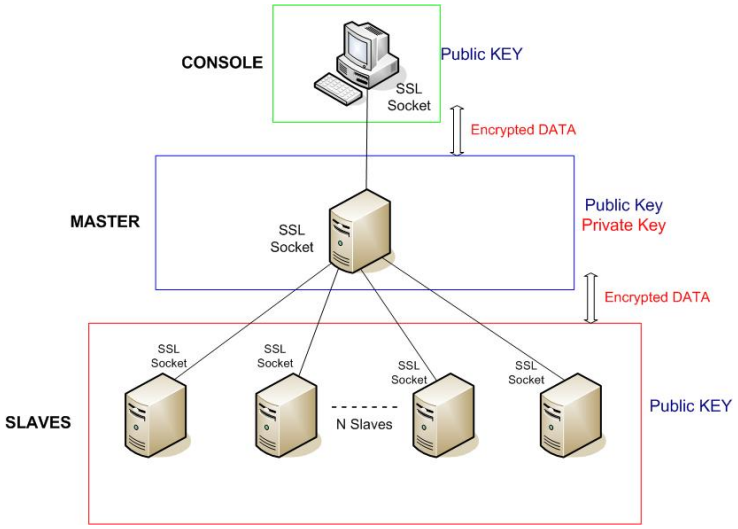


Figure 11: SSL communication scheme

# References

[1] W. L. Oberkampf, F. G. Blottner, and D. P. Aeshliman. Methodology for computational fluid dynamics code verification/validation. In *26th AIAA Fluid Dynamics Conference*, number AIAA 95-2226, 1995.

[2] W. L. Oberkampf and T. G. Trucano. Verification and validation in computational fluid dynamics. Technical report, Sandia National Laboratory, 2002.

[3] M. Garbey and W. Shyy. A least square extrapolation method for improving solution accuracy of pde computations. *JCP*, 186(1):1–23, 2003.

[4] M. Garbey and W. Shyy. A least square extrapolation method for the a posteriori error estimate of the incompressible navier stokes problem. *International Journal for Numerical Methods in Fluids*, 48(1):43–59, 2005.

[5] M. Garbey and C. Picard. A least square extrapolation method for the a posteriori error estimate of cfd and heat transfer problem. In *Sixth European Conference on Structural Dynamics, Paris*, 2005.

[6] J. Tinsley Oden. A posteriori error estimation. In *Verification and Validation in Computational Solid Mechanics*. Hans Maier, ASME/USACM Standards, 2002.

[7] P.J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, New Mexico, 1998.

[8] P. J. Roache. Code verification by the method of manufactured solutions. *Journal of Fluids Engineering*, 124:??, 2002.

[9] P.Holmes G.Berkooz and J.L.Humley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechnics*, 25:539–575, 1993.

[10] Raymond H. Myers and Douglas C. Montgomery. *Response Surface Methodology: Process and Product in Optimization Using Designed Experiments.* John Wiley & Sons, Inc., 1995.

[11] C. Picard, M. Garbey, and V. Subramanian. Mapping lse on a grid: Software architecture and performance gains. In *International Conference on Parallel Computational Fluid Dynamics*, 2005.

[12] Jorge J. More and Stephen J. Wright. *Optimization Software Guide.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1993.

[13] M. Garbey and W. Shyy. On optimized extrapolation method for elliptic problems with coefficients having large variation. Technical report, preprint UH-CS-05-17 submitted., Depart. of Comput. Science, University of Houston, 2005.

[14] M.Garbey and C.Picard. *Toward a General Solution Verification Method for Complex PDE problem with Hands off Coding,.* Technical report, preprint UH-CS-07-2 submitted., Depart. of Comput. Science, University of Houston, 2007.

[15] C. J. Roy and M. M. Hopkins. Discretization error estimates using exact solutions to nearby problems. In *41st AIAA Aerospace Sciences Meeting & Exhibit*, 2003.

[16] L. Machiels, J. Peraire, and A. T. Patera. A posteriori finite element output bounds for incompressible Navier-Stokes equations; application to a natural convection problem. *Journal of Computational Physics*, 172(2):401–425, 2001.