Mapping LSE method on a grid : Software architecture and Performance gains

Christophe Picard [a], Marc Garbey [a] and Venkat Subramaniam [a]

[a]Department of Computer Science, University of Houston, Houston, TX 77204, USA

In CFD community, a popular technique for a Posteriori error estimators consists of combining mesh refinements with Richardson Extrapolation (RE). Despite the fact this method is intuitive and simple, the validity and robustness remain questionable. In a series of papers ([1],[2]), the Least Square Extrapolation method (LSE), that provides a convenient framework to improve RE has been presented. Rather than an a priori Taylor expansion like model of the error, LSE makes use of the PDE discretization information, and thus on the research of the best linear combinations of coarse meshes solution. The computation of these solutions presents a strong asynchronous parallelism. On the other hand, there is an increasing interest in grid computation involving heterogeneous distribution of computing resources. In this paper, we will focus on the application of LSE method with a commercial CFD software on a network of heterogeneous computers. We will present out approach to software architecture to map the LSE on to the computer grid, and present measurement of performance gains compared to sequential execution.

1. Introduction and Motivations

In Computational Fluid Dynamic (CFD), *a Posteriori* error estimators are widely produced using Richardson extrapolation (RE) and variations of it ([6], [5], [7]). All these methods rely on the a priori existence of an asymptotic expansion of the error such as a Taylor formula, and make no direct use of the PDE formulation. As a consequence RE methods are extremely simple to implement.

But in practice, meshes might not be fine enough to satisfy accurately the a priori convergence estimates that are only asymptotic in nature. RE is then unreliable ([8]) or fairly unstable and sensitive to noisy data ([2]).

On the other hand, a Posteriori estimates in the framework of finite element analysis have been rigorously constructed. While most of the work has been limited to linear elliptic problems in order to drive adaptive mesh refinement, more recently a general framework for finite element a Posteriori error control that can be applied to linear and non-linear elliptic problem has been introduced in ([4]).

RE method can be embedded into an optimization framework to cope with RE's limitations while retaining its simplicity. Least Square Extrapolation (LSE) ([1], [2], [3]) uses grid meshes solutions that can be produced by any discretization. LSE sets the weights of the extrapolation formula as the solution of a minimization problem. This approach might be combined to existing a Posteriori estimate when they are available, but is still

applicable as a better alternative to straightforward RE when no such stability estimate is available.

The extrapolation procedure is simple to implement and should be incorporated into any computer code without requiring detailed knowledge of the source code. Its arithmetic cost should be modest compare to a direct computation of a very fine grid mesh solution. Finally, the procedure should overall enhance the accuracy and trust of a CFD application in the context of solution verification.

In this paper, we show how LSE can perform on a commercial code, without having any knowledge on the source code. We emphasize this approach by using a basic network of workstation to compute the weighted solutions and then solve the minimization problem over the produce results.

In Section 2, the general idea of LSE method for steady problems is recalled. In Section 3, we discuss how the network grid can be set to match LSE requirements. In Section 4, we present some general numerical results about LSE when using a commercial code and what are the advantages of using a grid of computers instead of performing a sequential resolution.

2. LSE Method

Let $G_i,\ i = 1..3$, be three embedded grid meshes that do not necessarily match, and their corresponding solutions U_i. Let M^0 be a regular mesh that is finer than the meshes G_i. Let $\tilde{U}_i$ be the coarse grid meshes solutions interpolated on the fine grid M^0.

The main idea of the LSE method is to look for a consistent extrapolation formula based on the interpolated coarse grid meshes solutions $\tilde{U}_i$ that minimizes the residual, resulting from $\tilde{U}_i$ on a grid M^0 that is fine enough to capture a good approximation of the continuous solution.

Defining the Navier-Stokes operator by $N(u) = 0$ for example, we can rewrite the problem as follow :

$P_{\alpha,\beta}$: Find α, β such that $N(\alpha\ U^1\ +\ \beta\ U^2\ +\ (1-\alpha-\beta)\ U^3\ )$ is minimum in L_2 in some Banach space.

Since minimizing the consistency error lead to the minimization of the solution error.

In practice, since we have access to solutions of discretized PDE problem, we work with *grid meshes functions*. The idea is now to use the PDE in the RE process to find an improved solution on a given fine grid mesh M^0, by post-processing the interpolated functions $\tilde{U}^i$, by few steps of the relaxation scheme

$$\frac{V^{k+1} - V^k}{\delta t} \ =\ L_h[V^k] - f_h,\ V^0 = \tilde{U}^i, \tag{1}$$

with appropriate artificial time step δt. For elliptic problems, this may readily smooth out the interpolant. For a more complete description of the method and mathematical aspects, one should refers to the work in [1], [2], [3], [8].

3. Mapping LSE method onto a grid of computer

The LSE method can be split in 4 main computational steps:

Step 1 - Consists on defining the problem and solve it on 2 (or 3) different meshes. While this step may be parallelized, in reality we may realize reduce error by solving it sequentially since the result of one computation can be used as a guess for the next one.

Step 2 - Once the solutions are obtained on coarse grids meshes, we interpolate them on a fine grid mesh G_0, and form linear combinations with 1 (or 2 parameters) as initial guess for step 3.

Step 3 - This is the most time and resource consuming step. The problems may be solved independently on different nodes, and the error and residual of solutions are the only output of interest.

Step 4 - In this part we find the minimum of residual and error with respect to parameters. We generate curve (or surface) responses. This should be kept open since it will be subject to modification and optimization.

We would like to be able to re-launch calculation based on the same fine grid mesh, but with different weights around the potential minimum, or plot the map of the residual/error for optimal parameters in order to perform manual mesh refinements.

To solve the problem in Step 3 we are employing a grid of computers. Our grid consists of a master node and several heterogeneous slave nodes. The steps 1 and 4 run on the master. Step 3 is distributed on several nodes, depending on the number of nodes available and the problem size.

The distribution of the computation across nodes involves some network challenges. The result of step 2 is a set of files that can be pretty large (any where from 7 MB to 21 MB or more). Moving such large files across the network will be a major bottleneck and will limit our capability to realizing any performance gains from parallelizing the computations across grids of computers. By reordering some computations from step 2 to step 3, we expect that the files size to be transmitted will be much smaller. Even though each node may have to perform some extra computation, we think that the overall gain from parallelization offsets that extra time spent.

In order to get a better utilization of nodes and to adjust for variations in performance and availability of nodes, we employ dynamic load balancing of the computational task across the nodes.

Figure 1 is a symbolic representation of the computational procedure describe in the previous paragraph.

4. Realizing LSE method

In order to apply LSE on a grid of computers, we are looking for some applications that can be used on different types of architectures and operating systems, and allowing us to use a scripting shell language to launch and manipulate the data. A first step in this process was to implement a 2D non-homogeneous diffusion equation.

The idea here is to have an implementation that is independent from the computational code. Fig .2 shows what are the languages used to perform the different tasks.

4.1. Application to Laplace equation

We have tried different test cases. Here we show some results for non homogeneous Laplace equations with a strong gradient in the diffusion coefficient (Fig. 4.1). The grid meshes are regular Cartesian meshes. We first compute the solutions with ten different

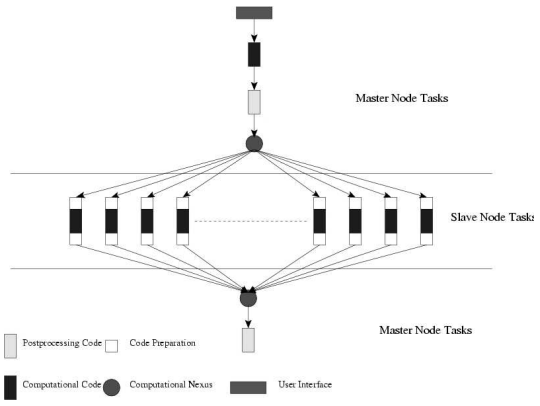

Figure 1. Software architecture realizing LSE

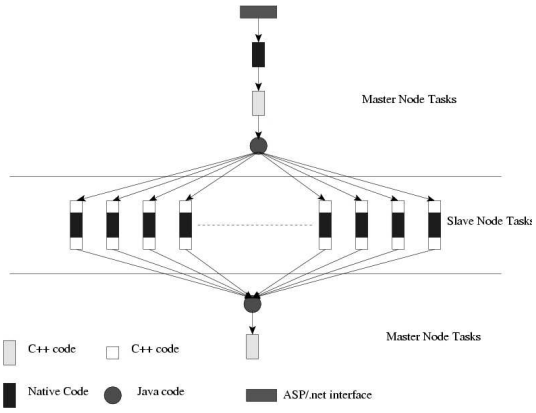

Figure 2. Language architecture realizing LSE

value for α. The grid meshes were chosen as follow : $G1 = 30 \times 30$, $G2 = 40 \times 40$, $G2 = 50 \times 50$ and $G0 = 100 \times 100$, where $G0$ is the fine grid.

To verify our solution, we also compute the solution on the finest grid mesh G_0. This solution allows us to compute the error we obtain using LSE versus the solution on the fine grid mesh. In figure 4.1 we show residuals obtained after few relaxations steps, and the corresponding error with the fine grid mesh solution. As we can see, there is an alpha that minimizes the residual, but also the error.

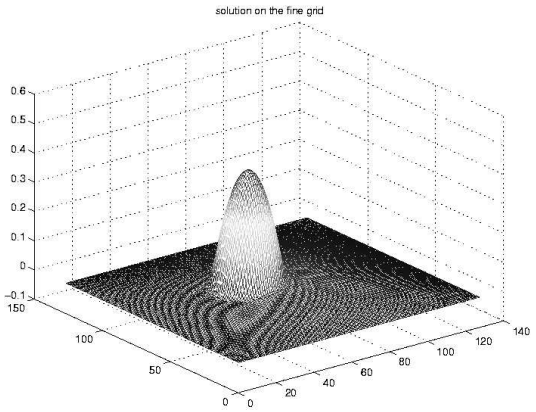

Figure 3. Solution of Laplace equation

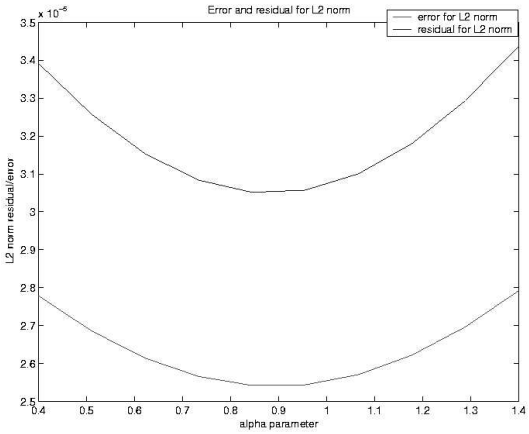

Figure 4. Residual and error curve for Laplace equation

For a two dimensions optimum design space for LSE optimization parameters, the results obtain are shown on figure. As we can see, to obtain a good resolution on the

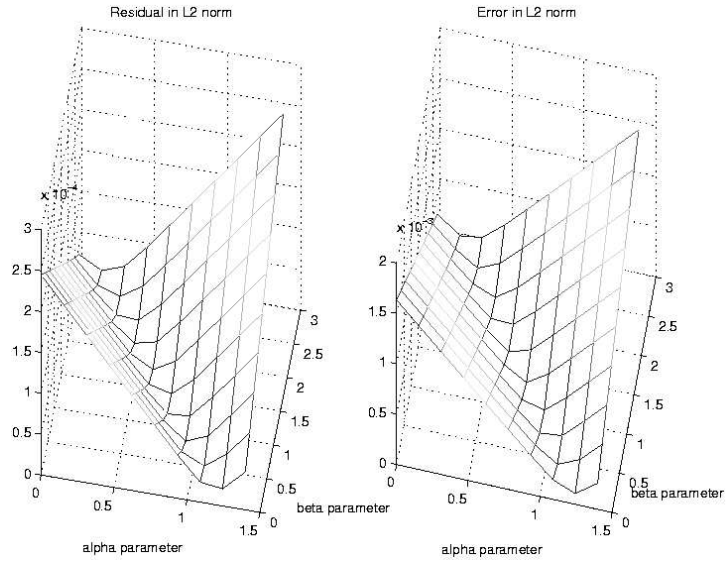

Figure 5. Residual and error curve for Laplace equation for a 2D optimum design space

value of the optimal parameters, the number of of initial step parameters can be important.

When solving LSE in a sequential way, with 100 parameters, the time required for this test case was 11.26 minutes , with roughly $80s$ for the Step 1, $5s$ for the Step 2, $600s$ for the Step 3, and $10s$ for the Step 4. It appears that Step 3 is a bottleneck in the method. Overall the solution verification procedure takes 615 s.

When mapping LSE onto a grid of computer, we manage to improve the performances of the method. In order to check the efficiency of this mapping, we run our code with three different scheme :

- on a single computer running Windows OS, using sequential LSE
- on a cluster of AMD Athlon running Linux OS, using a parallel version of LSE,
- on a heterogeneous grid running either Linux OS or Windows OS.

The speedup and scalability we obtain for the two parallel schemes are shown on fig. 4.1 and fig. 4.1. The speedup and scalability have a comportment close to the ideal one for the grid application. The main issue in that case is the quality of the network that is not homogeneous. For the cluster scheme, the comportment is becoming worst as we increase the number of nodes. Indeed, LSE required a lot of disk access since all the data have to be stored. Unfortunately, we didn't use a cluster with a parallel I/O system. Therefore, the scalability and the speedup reflect this characteristic.

The speed-up and scalability presented here are also relevant to the size of the grid meshes: given a number of weight, if we increase the size of the mesh the quality of the speedup should improve.

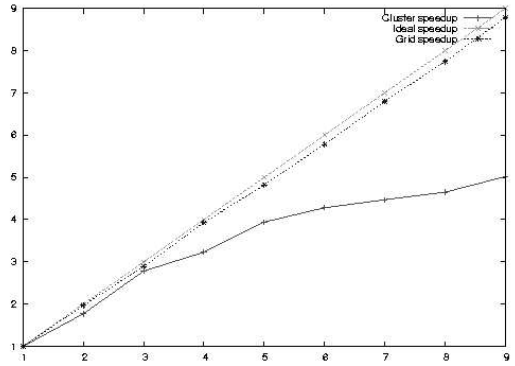

Figure 6. Speedup for LSE mapped onto a cluster and on a grid

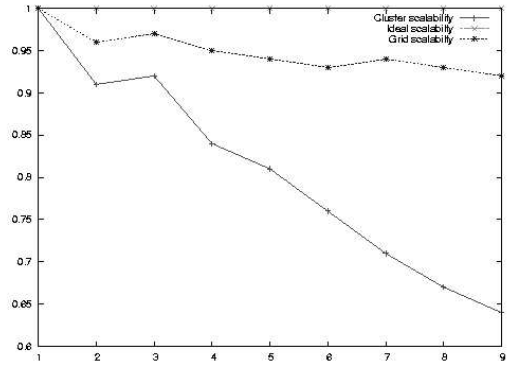

Figure 7. Scalability for LSE mapped onto a cluster and on a grid

4.2. Application to Navier-Stokes equations

The next step was to apply the same method to velocity equations in velocity-pressure formulation of Navier-Stokes equations. The software used to solve Navier-Stokes equation is ADINA R&D, a modeling software oriented to fluid-structure interaction. The test case we tried in the driven cavity flow in square box with a Reynold number of $Re = 100$ and a wall velocity of $v_0 = 1.0m.s^{-1}$. The grids meshes are three nodes finite elements meshes. We first compute the solutions with ten different value for α. The grid meshes were chosen as follow : $G1 = 30 \times 30$, $G2 = 40 \times 40$, $G2 = 50 \times 50$ and $G0 = 100 \times 100$, where $G0$ is the fine grid.

The figures 8 and 9 show the coarse grid mesh G_1 and the solution in velocity norm we obtained before any post-processing on step 1 of the presented algorithm. We obtained similar results for grids meshes G_2 and G_3.

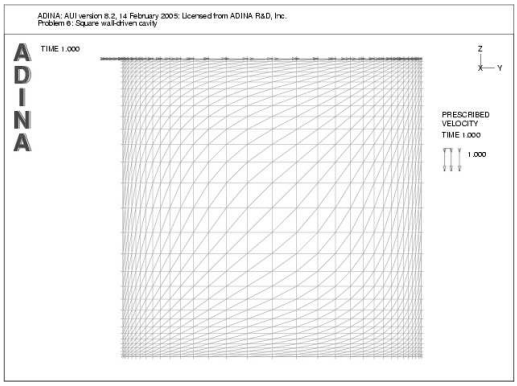

Figure 8. Coarse grid mesh G_1

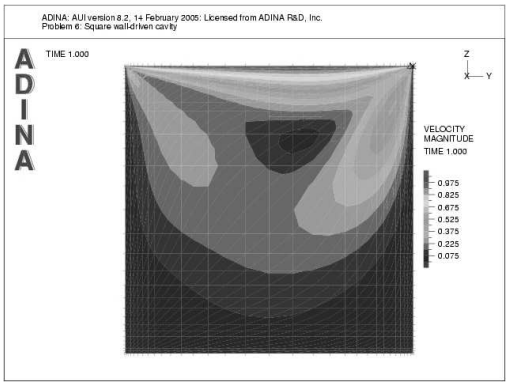

Figure 9. Solution on Grid G_1

To verify our solution, we also compute the solution on the finest grid mesh G_0. This solution allows us to compute the error we obtain using LSE versus the solution on the fine grid mesh. In figure 10 we show residuals obtained after few relaxations steps, and the corresponding error with the fine grid mesh solution. As we can see, there is an alpha that minimizes the residual, but also the error. On figure 11 the solution for the optimal parameter $\alpha = 1.5$ is presented.

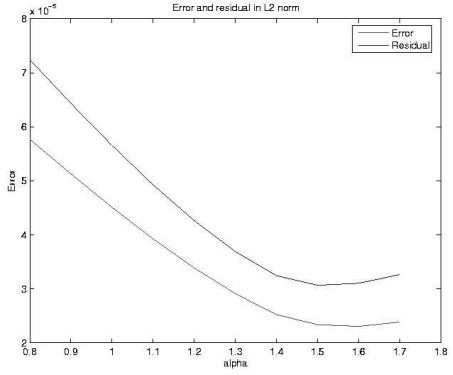

Figure 10. Error and residual in L_2 norm for $\alpha \in [0.8; 1.7]$

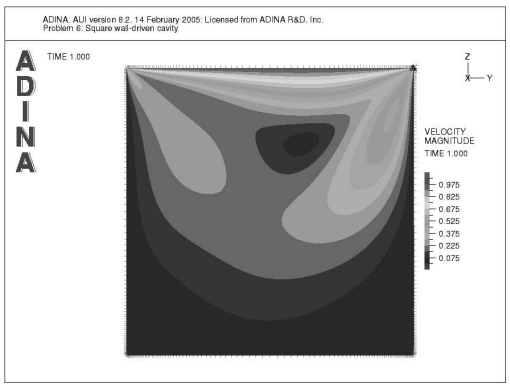

Figure 11. Solution on Grid G_0 for the optimal parameter

When solving LSE in the sequential time we obtained is 22 minutes, with roughly $130s$ for Step 1, $26s$ for step 2, $1200s$ for step 3 and $10s$ for step 4. For a parallelism on 10 nodes, Step 1 and Step 3 will have the same time as before, but the time for Step 2+3 will be $150s$, since we will compute the interpolation only once per node. We expect the speedup to be then 8.4 for the solution verification.

5. Conclusion

LSE is new extrapolation method for PDE that can be a solution verification method with hands off coding. It has be shown to be more efficient than Richardson extrapolation when the order of convergence of the CFD code is space dependent. In this paper, our objectives were to map the LSE method on a grid of computer and report the performance gains. We successfully managed to achieve those goal for a simple application.

This first step toward a grid application with Navier-Stokes equations is encouraging. It is obviously an interesting advantage to be able to perform a solution verification on 3D grid meshes using a grid of computer: most of the application time will then be spend into computation and not in communication. Some other aspects of LSE that are currently under consideration and that can influence the mapping onto a grid of computer are the optimum design space objective functions that can be higher order.

REFERENCES

1. M. Garbey. Some remarks on multilevel method, extrapolation and code verification. In N. Debit, M. Garbey, R. Hoppe, D. Keyes, Y. Kuznetsov, and J. Périaux, editors, *13th Int. Conf. on Domain Decomposition DD13, Domain Decomposition Methods in Science and Engineering*, pages 379–386. CIMNE, Barcelona, 2002.
2. M. Garbey and W. Shyy. A least square extrapolation method for improving solution accuracy of pde computations. *Journal of Computational Physics*, 186(1):1–23, 2003.
3. M. Garbey and W. Shyy. A least square extrapolation method for the a posteriori error estimate of the incompressible navier stokes problem. *Int. Journal of Fluid Dynamic*, to appear.
4. 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*, 2000.
5. W. L. Oberkampf and T. G. Trucano. Verification and validation in computational fluid dynamics. Technical report, Sandia National Laboratory, 2002.
6. W.L. Oberkampf, F.G. Blottner, and D.P. Aeshliman. Methodology for computational fluid dynamics code verification/validation. In *26th AIAA Fluid Dynamics Conference*, 1995.AIAA 95-2226.
7. P.J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, New Mexico, 1998.
8. W. Shyy, M. Garbey, A. Appukuttan, and J. Wu. Evaluation of richardson extrapolation in computational fluid dynamics. *Numerical Heat Transfer: Part B: Fundamentals*, 41(2):139 – 164, 2002.