Activity Report 2019

Project-Team ECUADOR

Program transformations for scientific computing

RESEARCH CENTER
Sophia Antipolis - Méditerranée

THEME
Numerical schemes and simulations

Table of contents

Project-Team ECUADOR

Creation of the Project-Team: 2014 January 01

Keywords:

Computer Science and Digital Science:

A2.1.1. - Semantics of programming languages
A2.2.1. - Static analysis
A2.5. - Software engineering
A6.1.1. - Continuous Modeling (PDE, ODE)
A6.2.6. - Optimization
A6.2.7. - High performance computing
A6.3.1. - Inverse problems
A6.3.2. - Data assimilation

Other Research Topics and Application Domains:

B1.1.2. - Molecular and cellular biology
B3.2. - Climate and meteorology
B3.3.2. - Water: sea & ocean, lake & river
B3.3.4. - Atmosphere
B5.2.3. - Aviation
B5.2.4. - Aerospace
B9.6.3. - Economy, Finance

1. Team, Visitors, External Collaborators

Research Scientists

Laurent Hascoët [Team leader, Inria, Senior Researcher, HDR]
Alain Dervieux [Inria, Emeritus, HDR]
Valérie Pascual [Inria, Researcher]

Administrative Assistant

Christine Claux [Inria, Administrative Assistant]

External Collaborator

Bruno Koobus [Univ Montpellier II (sciences et techniques du Languedoc)]

2. Overall Objectives

2.1. Overall Objectives

Team Ecuador studies Algorithmic Differentiation (AD) of computer programs, blending :

- **AD theory:** We study software engineering techniques, to analyze and transform programs mechanically. Algorithmic Differentiation (AD) transforms a program `P` that computes a function F, into a program `P'` that computes analytical derivatives of F. We put emphasis on the *adjoint mode* of AD, a sophisticated transformation that yields gradients for optimization at a remarkably low cost.
- **AD application to Scientific Computing:** We adapt the strategies of Scientific Computing to take full advantage of AD. We validate our work on real-size applications.

We aim to produce AD code that can compete with hand-written sensitivity and adjoint programs used in the industry. We implement our algorithms into the tool Tapenade, one of the most popular AD tools at present.

Our research directions :

- Efficient adjoint AD of frequent dialects e.g. Fixed-Point loops.
- Development of the adjoint AD model towards Dynamic Memory Management.
- Evolution of the adjoint AD model to keep in pace with with modern programming languages constructs.
- Optimal shape design and optimal control for steady and unsteady simulations. Higher-order derivatives for uncertainty quantification.
- Adjoint-driven mesh adaptation.

3. Research Program

3.1. Algorithmic Differentiation

Participants: Laurent Hascoët, Valérie Pascual.

algorithmic differentiation (AD, aka Automatic Differentiation) Transformation of a program, that returns a new program that computes derivatives of the initial program, i.e. some combination of the partial derivatives of the program's outputs with respect to its inputs.

adjoint Mathematical manipulation of the Partial Differential Equations that define a problem, obtaining new differential equations that define the gradient of the original problem's solution.

checkpointing General trade-off technique, used in adjoint AD, that trades duplicate execution of a part of the program to save some memory space that was used to save intermediate results.

Algorithmic Differentiation (AD) differentiates *programs*. The input of AD is a source program P that, given some $X \in \mathbb{R}^n$, returns some $Y = F(X) \in \mathbb{R}^m$, for a differentiable F. AD generates a new source program P' that, given X, computes some derivatives of F [4].

Any execution of P amounts to a sequence of instructions, which is identified with a composition of vector functions. Thus, if

$$\begin{array}{lll} P & \text{runs} & \{I_1; I_2; \cdots I_p;\}, \\ F & \text{then is} & f_p \circ f_{p-1} \circ \cdots \circ f_1, \end{array} \tag{1}$$

where each f_k is the elementary function implemented by instruction I_k. AD applies the chain rule to obtain derivatives of F. Calling X_k the values of all variables after instruction I_k, i.e. $X_0 = X$ and $X_k = f_k(X_{k-1})$, the Jacobian of F is

$$F'(X) = f'_p(X_{p-1}) \,.\, f'_{p-1}(X_{p-2}) \,.\, \cdots \,.\, f'_1(X_0) \tag{2}$$

which can be mechanically written as a sequence of instructions I'_k. This can be generalized to higher level derivatives, Taylor series, etc. Combining the I'_k with the control of P yields P', and therefore this differentiation is piecewise.

The above computation of $F'(X)$, albeit simple and mechanical, can be prohibitively expensive on large codes. In practice, many applications only need cheaper projections of $F'(X)$ such as:

- **Sensitivities**, defined for a given direction $\dot{X}$ in the input space as:

$$F'(X).\dot{X} = f'_p(X_{p-1}) \,.\, f'_{p-1}(X_{p-2}) \,.\, \cdots \,.\, f'_1(X_0) \,.\, \dot{X} \quad . \tag{3}$$

This expression is easily computed from right to left, interleaved with the original program instructions. This is the *tangent mode* of AD.

- **Adjoints**, defined after transposition (F'^*), for a given weighting $\overline{Y}$ of the outputs as:

$$F'^*(X).\overline{Y} = f_1'^*(X_0).f_2'^*(X_1).\ \cdots\ .f_{p-1}'^*(X_{p-2}).f_p'^*(X_{p-1}).\overline{Y} \quad . \tag{4}$$

This expression is most efficiently computed from right to left, because matrix×vector products are cheaper than matrix×matrix products. This is the *adjoint mode* of AD, most effective for optimization, data assimilation [31], adjoint problems [25], or inverse problems.

Adjoint AD builds a very efficient program [27], which computes the gradient in a time independent from the number of parameters n. In contrast, computing the same gradient with the *tangent mode* would require running the tangent differentiated program n times.

However, the X_k are required in the *inverse* of their computation order. If the original program *overwrites* a part of X_k, the differentiated program must restore X_k before it is used by $f_{k+1}'^*(X_k)$. Therefore, the central research problem of adjoint AD is to make the X_k available in reverse order at the cheapest cost, using strategies that combine storage, repeated forward computation from available previous values, or even inverted computation from available later values.

Another research issue is to make the AD model cope with the constant evolution of modern language constructs. From the old days of Fortran77, novelties include pointers and dynamic allocation, modularity, structured data types, objects, vectorial notation and parallel programming. We keep developing our models and tools to handle these new constructs.

3.2. Static Analysis and Transformation of programs

Participants: Laurent Hascoët, Valérie Pascual.

abstract syntax tree Tree representation of a computer program, that keeps only the semantically significant information and abstracts away syntactic sugar such as indentation, parentheses, or separators.

control flow graph Representation of a procedure body as a directed graph, whose nodes, known as basic blocks, each contain a sequence of instructions and whose arrows represent all possible control jumps that can occur at run-time.

abstract interpretation Model that describes program static analysis as a special sort of execution, in which all branches of control switches are taken concurrently, and where computed values are replaced by abstract values from a given *semantic domain*. Each particular analysis gives birth to a specific semantic domain.

data flow analysis Program analysis that studies how a given property of variables evolves with execution of the program. Data Flow analysis is static, therefore studying all possible run-time behaviors and making conservative approximations. A typical data-flow analysis is to detect, at any location in the source program, whether a variable is initialized or not.

The most obvious example of a program transformation tool is certainly a compiler. Other examples are program translators, that go from one language or formalism to another, or optimizers, that transform a program to make it run better. AD is just one such transformation. These tools share the technological basis that lets them implement the sophisticated analyses [14] required. In particular there are common mathematical models to specify these analyses and analyze their properties.

An important principle is *abstraction*: the core of a compiler should not bother about syntactic details of the compiled program. The optimization and code generation phases must be independent from the particular input programming language. This is generally achieved using language-specific *front-ends*, language-independent *middle-ends*, and target-specific *back-ends*. In the middle-end, analysis can concentrate on the semantics of a reduced set of constructs. This analysis operates on an abstract representation of programs made of one *call graph*, whose nodes are themselves *flow graphs* whose nodes (*basic blocks*) contain abstract *syntax trees* for the individual atomic instructions. To each level are attached symbol tables, nested to capture scoping.

Static program analysis can be defined on this internal representation, which is largely language independent. The simplest analyses on trees can be specified with inference rules [18], [28], [15]. But many *data-flow analyses* are more complex, and better defined on graphs than on trees. Since both call graphs and flow graphs may be cyclic, these global analyses will be solved iteratively. *Abstract Interpretation* [19] is a theoretical framework to study complexity and termination of these analyses.

Data flow analyses must be carefully designed to avoid or control combinatorial explosion. At the call graph level, they can run bottom-up or top-down, and they yield more accurate results when they take into account the different call sites of each procedure, which is called *context sensitivity*. At the flow graph level, they can run forwards or backwards, and yield more accurate results when they take into account only the possible execution flows resulting from possible control, which is called *flow sensitivity*.

Even then, data flow analyses are limited, because they are static and thus have very little knowledge of actual run-time values. Far before reaching the very theoretical limit of *undecidability*, one reaches practical limitations to how much information one can infer from programs that use arrays [34], [20] or pointers. Therefore, conservative *over-approximations* must be made, leading to derivative code less efficient than ideal.

3.3. Algorithmic Differentiation and Scientific Computing

Participants: Alain Dervieux, Laurent Hascoët, Bruno Koobus, Eléonore Gauci, Emmanuelle Itam, Olivier Allain, Stephen Wornom.

linearization In Scientific Computing, the mathematical model often consists of Partial Differential Equations, that are discretized and then solved by a computer program. Linearization of these equations, or alternatively linearization of the computer program, predict the behavior of the model when small perturbations are applied. This is useful when the perturbations are effectively small, as in acoustics, or when one wants the sensitivity of the system with respect to one parameter, as in optimization.

adjoint state Consider a system of Partial Differential Equations that define some characteristics of a system with respect to some parameters. Consider one particular scalar characteristic. Its sensitivity (or gradient) with respect to the parameters can be defined by means of *adjoint* equations, deduced from the original equations through linearization and transposition. The solution of the adjoint equations is known as the adjoint state.

Scientific Computing provides reliable simulations of complex systems. For example it is possible to *simulate* the steady or unsteady 3D air flow around a plane that captures the physical phenomena of shocks and turbulence. Next comes *optimization*, one degree higher in complexity because it repeatedly simulates and applies gradient-based optimization steps until an optimum is reached. The next sophistication is *robustness*, that detects undesirable solutions which, although maybe optimal, are very sensitive to uncertainty on design parameters or on manufacturing tolerances. This makes second derivatives come into play. Similarly *Uncertainty Quantification* can use second derivatives to evaluate how uncertainty on the simulation inputs imply uncertainty on its outputs.

To obtain this gradient and possibly higher derivatives, we advocate adjoint AD (*cf* 3.1) of the program that discretizes and solves the direct system. This gives the exact gradient of the discrete function computed by the program, which is quicker and more sound than differentiating the original mathematical equations [25]. Theoretical results [24] guarantee convergence of these derivatives when the direct program converges. This approach is highly mechanizable. However, it requires careful study and special developments of the AD

model [29], [32] to master possibly heavy memory usage. Among these additional developments, we promote in particular specialized AD models for Fixed-Point iterations [26], [17], efficient adjoints for linear algebra operators such as solvers, or exploitation of parallel properties of the adjoint code.

4. Application Domains

4.1. Algorithmic Differentiation

Algorithmic Differentiation of programs gives sensitivities or gradients, useful for instance for :

- optimum shape design under constraints, multidisciplinary optimization, and more generally any algorithm based on local linearization,
- inverse problems, such as parameter estimation and in particular 4Dvar data assimilation in climate sciences (meteorology, oceanography),
- first-order linearization of complex systems, or higher-order simulations, yielding reduced models for simulation of complex systems around a given state,
- adaption of parameters for classification tools such as Machine Learning systems, in which Adjoint Differentiation is also known as *backpropagation*.
- mesh adaptation and mesh optimization with gradients or adjoints,
- equation solving with the Newton method,
- sensitivity analysis, propagation of truncation errors.

4.2. Multidisciplinary optimization

A CFD program computes the flow around a shape, starting from a number of inputs that define the shape and other parameters. On this flow one can define optimization criteria e.g. the lift of an aircraft. To optimize a criterion by a gradient descent, one needs the gradient of the criterion with respect to all inputs, and possibly additional gradients when there are constraints. Adjoint AD is the most efficient way to compute these gradients.

4.3. Inverse problems and Data Assimilation

Inverse problems aim at estimating the value of hidden parameters from other measurable values, that depend on the hidden parameters through a system of equations. For example, the hidden parameter might be the shape of the ocean floor, and the measurable values of the altitude and velocities of the surface. Figure 1 shows an example of an inverse problem using the glaciology code ALIF (a pure C version of ISSM [30]) and its AD-adjoint produced by Tapenade.

One particular case of inverse problems is *data assimilation* [31] in weather forecasting or in oceanography. The quality of the initial state of the simulation conditions the quality of the prediction. But this initial state is not well known. Only some measurements at arbitrary places and times are available. A good initial state is found by solving a least squares problem between the measurements and a guessed initial state which itself must verify the equations of meteorology. This boils down to solving an adjoint problem, which can be done though AD [33]. The special case of *4Dvar* data assimilation is particularly challenging. The 4^{th} dimension in "4D" is time, as available measurements are distributed over a given assimilation period. Therefore the least squares mechanism must be applied to a simulation over time that follows the time evolution model. This process gives a much better estimation of the initial state, because both position and time of measurements are taken into account. On the other hand, the adjoint problem involved is more complex, because it must run (backwards) over many time steps. This demanding application of AD justifies our efforts in reducing the runtime and memory costs of AD adjoint codes.

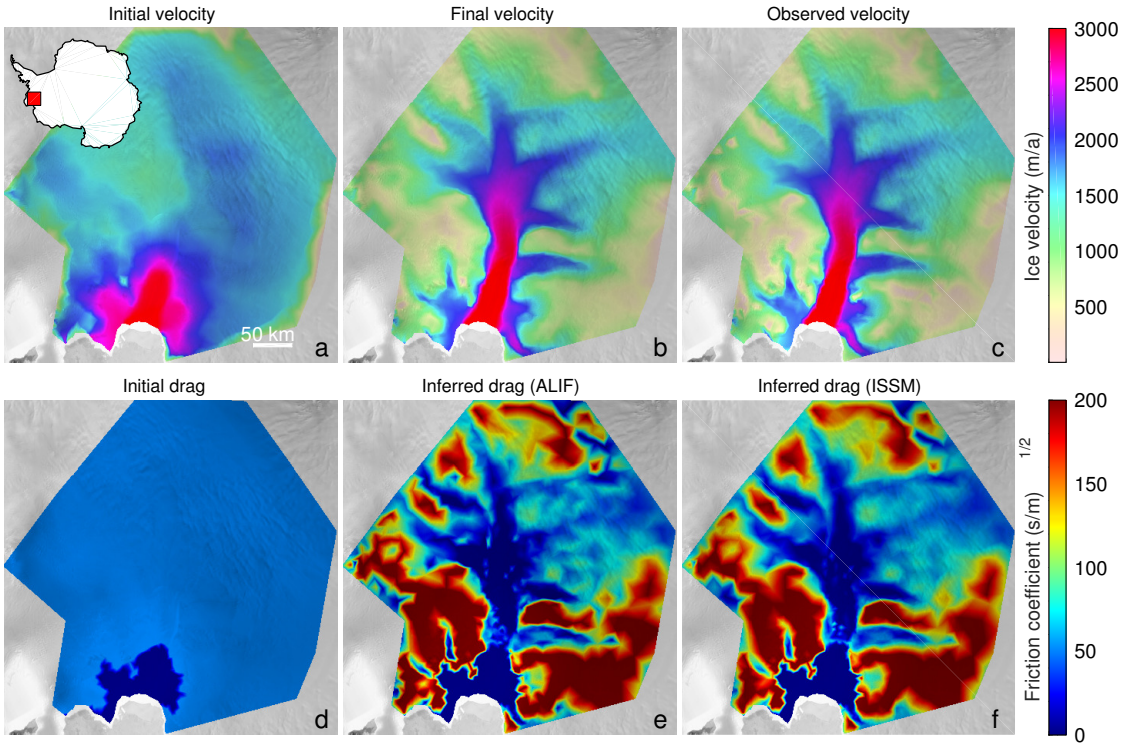

Figure 1. Assimilation of the basal friction under Pine Island glacier, West Antarctica. The final simulated surface velocity (b) is made to match the observed surface velocity (c), by estimation of the basal friction (e). A reference basal friction (f) is obtained by another data assimilation using the hand=written adjoint of ISSM

4.4. Linearization

Simulating a complex system often requires solving a system of Partial Differential Equations. This can be too expensive, in particular for real-time simulations. When one wants to simulate the reaction of this complex system to small perturbations around a fixed set of parameters, there is an efficient approximation: just suppose that the system is linear in a small neighborhood of the current set of parameters. The reaction of the system is thus approximated by a simple product of the variation of the parameters with the Jacobian matrix of the system. This Jacobian matrix can be obtained by AD. This is especially cheap when the Jacobian matrix is sparse. The simulation can be improved further by introducing higher-order derivatives, such as Taylor expansions, which can also be computed through AD. The result is often called a *reduced model*.

4.5. Mesh adaptation

Some approximation errors can be expressed by an adjoint state. Mesh adaptation can benefit from this. The classical optimization step can give an optimization direction not only for the control parameters, but also for the approximation parameters, and in particular the mesh geometry. The ultimate goal is to obtain optimal control parameters up to a precision prescribed in advance.

5. New Software and Platforms

5.1. AIRONUM

KEYWORDS: Computational Fluid Dynamics - Turbulence

FUNCTIONAL DESCRIPTION: Aironum is an experimental software that solves the unsteady compressible Navier-Stokes equations with k-epsilon, LES-VMS and hybrid turbulence modelling on parallel platforms, using MPI. The mesh model is unstructured tetrahedrization, with possible mesh motion.

- Participant: Alain Dervieux
- Contact: Alain Dervieux
- URL: http://www-sop.inria.fr/tropics/aironum

5.2. TAPENADE

KEYWORDS: Static analysis - Optimization - Compilation - Gradients

SCIENTIFIC DESCRIPTION: Tapenade implements the results of our research about models and static analyses for AD. Tapenade can be downloaded and installed on most architectures. Alternatively, it can be used as a web server. Higher-order derivatives can be obtained through repeated application.

Tapenade performs sophisticated data-flow analysis, flow-sensitive and context-sensitive, on the complete source program to produce an efficient differentiated code. Analyses include Type-Checking, Read-Write analysis, and Pointer analysis. AD-specific analyses include the so-called Activity analysis, Adjoint Liveness analysis, and TBR analysis.

FUNCTIONAL DESCRIPTION: Tapenade is an Algorithmic Differentiation tool that transforms an original program into a new program that computes derivatives of the original program. Algorithmic Differentiation produces analytical derivatives, that are exact up to machine precision. Adjoint-mode AD can compute gradients at a cost which is independent from the number of input variables. Tapenade accepts source programs written in Fortran77, Fortran90, or C. It provides differentiation in the following modes: tangent, vector tangent, adjoint, and vector adjoint.

NEWS OF THE YEAR: - Continued development of multi-language capacity: AD of codes mixing Fortran and C - Continued front-end for C++ (using Clang-LLVM) - Preliminary work, including refactoring, in view of future Open-Source distribution

- Participants: Laurent Hascoët and Valérie Pascual
- Contact: Laurent Hascoët
- URL: http://www-sop.inria.fr/tropics/tapenade.html

6. New Results

6.1. Towards Algorithmic Differentiation of C++

Participants: Laurent Hascoët, Valérie Pascual, Frederic Cazals [ABS team, Inria Sophia-Antipolis].

Our goal is to extend Tapenade for C++. We further developed our external parser for C++, built on top of Clang-LLVM https://clang.llvm.org/. This parser is now connected to the input formalism "IL" of Tapenade. Tapenade now manages enough constructs of Object languages to be able to build its own Internal Representation (IR) and to regenerate back from the IR a non-transformed C++ source.

In the present development stage, this back-and-forth chain works on several small C++ codes. Still, many issues remain on the large example provided by the ABS team. We are working to solve those progressively. The lack of serious development documentation on the Clang internals obviously doesn't help.

The next development stage will be to adapt the analysis and differentiation components of Tapenade to the new Object constructs of the IR. This development has not started yet. Upstream this development, we need to devise an extended AD model correspondingly. This research part is in progress.

6.2. AD of mixed-language codes

Participants: Valérie Pascual, Laurent Hascoët.

We extend Tapenade to differentiate codes that mix different languages, for both tangent and adjoint modes of AD. Our motivating application is Calculix, a 3-D Structural Finite Element code that mixes Fortran and C. This year we improved the memory representation of Tapenade's IR to handle the C-Fortran memory correspondence (commons, structs...) defined by the Fortran standard.

C files (aka "translation units") and Fortran modules are two instances of the more general notion of "package" for which we have developed a unified representation in Tapenade, that also handles C++ namespaces.

6.3. Application to large industrial codes

Participants: Valérie Pascual, Laurent Hascoët, Bruno Maugars [ONERA], Sébastien Bourasseau [ONERA], Cédric Content [ONERA], Jose I. Cardesa [IMFT], Christophe Airiau [IMFT].

We support industrial users with their first experiments of Algorithmic Differentiation of large in-house codes. This concerned two industrial codes this year.

One application is with ONERA on their ElsA CFD platform (Fortran 90). This is the continuation of a collaboration started in 2018. Both tangent and adjoint models of the kernel of ElsA were built successfully with Tapenade. This year's work was mostly about improving efficiency. It is worth noticing that this application was performed inside ONERA by ONERA engineers (Bruno Maugars, Sébastien Bourasseau, Cédric Content) with no need for installation of ElsA inside Inria. We take this as a sign of maturity of Tapenade. Our contribution is driven by development meetings, in which we point out some strategies and tool options to improve efficiency of the adjoint code. As a result from these discussions, we developed improved strategies and AD model, that will be useful to other tools. These improvements deal mostly with the adjoint of vectorized code. We prepared together an aticle that describes the architecture of a modular and AD-friendly ElsA, together with the corresponding extensions of the AD model of Tapenade. This article has been submitted to "Computers and Fluids".

The other application ultimately targets AD of the "Jaguar" code, developed jointly by ONERA, CERFACS, and IMFT in Toulouse. This is a collaboration with Jose I. Cardesa and Christophe Airiau, both at IMFT. After a relatively easy tangent differentiation, most of the effort was devoted to obtaining a running and efficient adjoint code. Not too surprisingly, the main source of trouble was the Message-Passing parallel aspect on the application code. This underlined a lack of debugging support for adjoint-differentiated code. Given the run-time of the simulation that we consider (from hours to a few days on a 4110 processors platform), efficiency is crucial. We used the optimal binomial checkpointing scheme at the time-stepping level. However, performance of the adjoint code can probably be improved further with a better checkpointing scheme on the call tree. This calls in particular for AD-specific performance profiling tools, that we are planning to develop. We prepared together an article that describes this succesful experiment, which is now submitted to "Journal of Computational Science".

Two collaborations are in preparation for next year, one with Jan Hueckelheim at Argonne National Lab. about SIMD parallel codes, and one with Stefano Carli at KU Leuven about adjoint AD of the plasma code SOLPS-ITER.

6.4. Control of approximation errors

Participants: Alain Dervieux, Loic Frazza [Gamma3 team, Inria-Saclay], Adrien Loseille [Gamma3 team, Inria-Saclay], Frédéric Alauzet [Gamma3 team, Inria-Saclay], Anca Belme [university of Paris 6], Alexandre Carabias [Lemma].

Reducing approximation errors as much as possible is a particular kind of optimal control problem. We formulate it exactly this way when we look for the optimal metric of the mesh, which minimizes a user-specified functional (goal-oriented mesh adaptation). In that case, the usual methods of optimal control apply, using adjoint states that can be produced by Algorithmic Differentiation.

This year, we published the final revised versions of two conference papers [23], [21], we published in a journal the final version of the adjoint-based mesh adaptation for Navier-Stokes flows [16]), and we published in "Numerical Methods in Fluids" a work on nonlinear correctors extending [22]. Let us also mention the final publication of the book "Uncertainty Management for Robust Industrial Design in Aeronautics", edited by C. Hirsch et al. in the Springer series Notes on Numerical Fluid Mechanics and Multidisciplinary Design (2019) in which we have contributed chapters 20, 21, 45, and 48.

The monography on mesh adaptation currently being written by Alauzet, Loseille, Koobus and Dervieux now involves all its chapters (14 chapters) and is being finalized.

6.5. Turbulence models

Participants: Alain Dervieux, Bruno Koobus, Stephen Wornom.

Modeling turbulence is an essential aspect of CFD. The purpose of our work in hybrid RANS/LES (Reynolds Averaged Navier-Stokes / Large Eddy Simulation) is to develop new approaches for industrial applications of LES-based analyses. In the applications targetted (aeronautics, hydraulics), the Reynolds number can be as high as several tens of millions, far too high for pure LES models. However, certain regions in the flow can be predicted better with LES than with usual statistical RANS models. These are mainly vortical separated regions as assumed in one of the most popular hybrid models, the hybrid Detached Eddy Simulation (DES) model. Here, "hybrid" means that a blending is applied between LES and RANS. An important difference between a real life flow and a wind tunnel or basin is that the turbulence of the flow upstream of each body is not well known.

The development of hybrid models, in particular DES in the litterature, has raised the question of the domain of validity of these models. According to theory, these models should not be applied to flow involving laminar boundary layers (BL). But industrial flows are complex flows and often present regions of laminar BL, regions of fully developed turbulent BL and regions of non-equilibrium vortical BL. It is then mandatory for industrial use that the new hybrid models give a reasonable prediction for all these types of flow. We concentrated on evaluating the behavior of hybrid models for laminar BL and for vortical wakes. While less predictive than pure LES on laminar BL, some hybrid models still give reasonable predictions for rather low Reynolds numbers.

We have developed a new model relying on the hybridation of a DES model based on a k-ϵ closure with our dynamic VMS model [13] [11]. Our purpose is to propose a model rather predictive in condition where the engineer has not much information concerning the turbulence in the flow under study. This year, we continued to improve this model and to test it for a large set of configurations with Reynolds numbers ranging from low (laminar flows) to very large.

6.6. High order approximations

Participants: Alain Dervieux, Bruno Koobus, Stephen Wornom, Tanya Kozubskaya [Keldysh Institute of Russian Academy].

High order approximations for compressible flows on unstructured meshes are facing many constraints that increase their complexity i.e. their computational cost. This is clear for the largest class of approximation, the class of k-exact schemes, which rely on a local polynomial representation of degree k. We are investigating schemes which would solve as efficiently as possible the dilemma of choosing between an approximation with a representation inside macro-elements which finally constrains the mesh, and a representation around each individual cell, as in vertex formulations. This is a cooperation with the Keldysh Institute of Russian Academy which whom we have already developed several families of superconvergent schemes.

6.7. Aeroacoustics

Participants: Alain Dervieux, Bruno Koobus, Stephen Wornom, Tanya Kozubskaya [Keldysh Institute of Russian Academy].

The progress in highly accurate schemes for compressible flows on unstructured meshes (together with advances in massive parallelization of these schemes) allows us to solve problems previously out of reach. The three teams of Montpellier university (coordinator), Inria-Sophia and Keldysh Institute of Moscow have written a proposal for cooperation on the subject of the extension of these methods to simulate the noise emission of rotating machines (helicopters, future aerial vehicles, unmanned aerial vehicles, wind turbines...). The proposal has been selected by ANR and RSF (Russian Science Foundation) for support for a program duration of 4 years.

7. Dissemination

7.1. Promoting Scientific Activities

7.1.1. Scientific Events: Organisation

7.1.1.1. Member of the organizing committees

- Laurent Hascoët is on the organizing commitee of the EuroAD Workshops on Algorithmic Differentiation (http://www.autodiff.org).
- Laurent Hascoët was on the organizing and program committees of the workshop “Program Transformations for Machine Learning” at NeurIPS2019, Vancouver Canada, December 14th.

7.1.2. Invited Talks

Laurent Hascoët was invited to give a talk on AD for the “GdR Calcul”, at “Institut de Physique du Globe”, Paris, January 24th.

7.1.3. Scientific Expertise

Alain Dervieux is Scientific Director for the LEMMA company.

8. Bibliography

Major publications by the team in recent years

[1] F. COURTY, A. DERVIEUX, B. KOOBUS, L. HASCOËT. *Reverse automatic differentiation for optimum design: from adjoint state assembly to gradient computation*, in "Optimization Methods and Software", 2003, vol. 18, n^{o} 5, pp. 615-627

[2] B. DAUVERGNE, L. HASCOËT. *The Data-Flow Equations of Checkpointing in reverse Automatic Differentiation*, in "International Conference on Computational Science, ICCS 2006, Reading, UK", 2006

[3] D. GOLDBERG, S. H. K. NARAYANAN, L. HASCOËT, J. UTKE. *An optimized treatment for algorithmic differentiation of an important glaciological fixed-point problem*, in "Geoscientific Model Development", 2016, vol. 9, n^{o} 5, 27 p. , https://hal.inria.fr/hal-01413295

[4] L. HASCOËT. *Adjoints by Automatic Differentiation*, in "Advanced data assimilation for geosciences", Oxford University Press, 2014, https://hal.inria.fr/hal-01109881

[5] L. HASCOËT, U. NAUMANN, V. PASCUAL. *"To Be Recorded" Analysis in Reverse-Mode Automatic Differentiation*, in "Future Generation Computer Systems", 2004, vol. 21, n^{o} 8

[6] L. HASCOËT, J. UTKE, U. NAUMANN. *Cheaper Adjoints by Reversing Address Computations*, in "Scientific Programming", 2008, vol. 16, n^{o} 1, pp. 81–92

[7] L. HASCOËT, M. VÁZQUEZ, B. KOOBUS, A. DERVIEUX. *A Framework for Adjoint-based Shape Design and Error Control*, in "Computational Fluid Dynamics Journal", 2008, vol. 16, n^{o} 4, pp. 454-464

[8] L. HASCOËT, V. PASCUAL. *The Tapenade Automatic Differentiation tool: Principles, Model, and Specification*, in "ACM Transactions On Mathematical Software", 2013, vol. 39, n^{o} 3, http://dx.doi.org/10.1145/2450153.2450158

[9] L. HASCOËT, J. UTKE. *Programming language features, usage patterns, and the efficiency of generated adjoint code*, in "Optimization Methods and Software", 2016, vol. 31, pp. 885–903 [*DOI :* 10.1080/10556788.2016.1146269], https://hal.inria.fr/hal-01413332

[10] J. C. HUECKELHEIM, L. HASCOËT, J.-D. MÜLLER. *Algorithmic differentiation of code with multiple context-specific activities*, in "ACM Transactions on Mathematical Software", 2016, https://hal.inria.fr/hal-01413321

Publications of the year

Articles in International Peer-Reviewed Journals

[11] E. ITAM, S. F. WORNOM, B. KOOBUS, A. DERVIEUX. *A Volume-agglomeration multirate time advancing for high Reynolds number flow simulation*, in "International Journal for Numerical Methods in Fluids", 2019, vol. 89, n^{o} 8, pp. 326-341 [*DOI :* 10.1002/FLD.4702], https://hal.inria.fr/hal-01928223

[12] P. MOHANAMURALY, L. HASCOËT, J.-D. MÜLLER. *Seeding and adjoining zero-halo partitioned parallel scientific codes*, in "Optimization Methods and Software", April 2019, pp. 1-20 [*DOI :* 10.1080/10556788.2019.1591404], https://hal.inria.fr/hal-02368107

International Conferences with Proceedings

[13] E. ITAM, S. F. WORNOM, B. KOOBUS, A. DERVIEUX. *Hybrid versus pure-les models comparison for subcritical cylinder flows*, in "DLES11 2017 - ERCOFTAC Workshop Direct and Large-Eddy Simulation 11", Pisa, Italy, M. SALVETTI, V. ARMENIO, J. FRÖHLICH, B. GEURTS, H. KUERTEN (editors), ERCOFTAC Series, Springer, Cham, 2019, vol. 25, pp. 445-451 [*DOI :* 10.1007/978-3-030-04915-7_59], https://hal.inria.fr/hal-01655171

References in notes

[14] A. AHO, R. SETHI, J. ULLMAN. *Compilers: Principles, Techniques and Tools*, Addison-Wesley, 1986

[15] I. ATTALI, V. PASCUAL, C. ROUDET. *A language and an integrated environment for program transformations*, Inria, 1997, n^{o} 3313, http://hal.inria.fr/inria-00073376

[16] A. BELME, F. ALAUZET, A. DERVIEUX. *An a priori anisotropic Goal-Oriented Error Estimate for Viscous Compressible Flow and Application to Mesh Adaptation*, in "J. Comp. Phys.", 2019, pp. 1051-1088, https://hal.inria.fr/hal-01927113

[17] B. CHRISTIANSON. *Reverse accumulation and implicit functions*, in "Optimization Methods and Software", 1998, vol. 9, n^{o} 4, pp. 307–322

[18] D. CLÉMENT, J. DESPEYROUX, L. HASCOËT, G. KAHN. *Natural semantics on the computer*, in "Proceedings, France-Japan AI and CS Symposium, ICOT", 1986, pp. 49-89, Also, Information Processing Society of Japan, Technical Memorandum PL-86-6. Also Inria research report # 416, http://hal.inria.fr/inria-00076140

[19] P. COUSOT. *Abstract Interpretation*, in "ACM Computing Surveys", 1996, vol. 28, n^{o} 1, pp. 324-328

[20] B. CREUSILLET, F. IRIGOIN. *Interprocedural Array Region Analyses*, in "International Journal of Parallel Programming", 1996, vol. 24, n^{o} 6, pp. 513–546

[21] A. DERVIEUX, E. GAUCI, L. FRAZZA, A. BELME, A. CARABIAS, A. LOSEILLE, F. ALAUZET. *Mesh adaptation for k-exact CFD approximations*, in "Numerical Methods for Flows, FEF 2017 Selected Contributions", 2020, Lecture Notes in Computational Science and Engineering, Springer, https://hal.inria.fr/hal-01927145

[22] L. FRAZZA, A. LOSEILLE, F. ALAUZET, A. DERVIEUX. *Nonlinear corrector for RANS equations*, in "Int. J. Numer. Meth. Fluids", 2019, n^{o} 11, pp. 567-586, https://hal.inria.fr/hal-01962171

[23] E. GAUCI, A. BELME, A. CARABIAS, A. LOSEILLE, F. ALAUZET, A. DERVIEUX. *A priori error-based mesh adaptation in CFD*, International Press, 2019, Methods and Applications of Analysis, https://hal.inria.fr/hal-01928249

[24] J. GILBERT. *Automatic differentiation and iterative processes*, in "Optimization Methods and Software", 1992, vol. 1, pp. 13–21

[25] M.-B. GILES. *Adjoint methods for aeronautical design*, in "Proceedings of the ECCOMAS CFD Conference", 2001

[26] A. GRIEWANK, C. FAURE. *Reduced Gradients and Hessians from Fixed Point Iteration for State Equations*, in "Numerical Algorithms", 2002, vol. 30(2), pp. 113–139

[27] A. GRIEWANK, A. WALTHER. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, 2nd, SIAM, Other Titles in Applied Mathematics, 2008

[28] L. HASCOËT. *Transformations automatiques de spécifications sémantiques: application: Un vérificateur de types incremental*, Université de Nice Sophia-Antipolis, 1987

[29] P. HOVLAND, B. MOHAMMADI, C. BISCHOF. *Automatic Differentiation of Navier-Stokes computations*, Argonne National Laboratory, 1997, n^{o} MCS-P687-0997

[30] E. LAROUR, J. UTKE, B. CSATHO, A. SCHENK, H. SEROUSSI, M. MORLIGHEM, E. RIGNOT, N. SCHLEGEL, A. KHAZENDAR. *Inferred basal friction and surface mass balance of the Northeast Greenland Ice Stream using data assimilation of ICESat (Ice Cloud and land Elevation Satellite) surface altimetry and ISSM (Ice Sheet System Model)*, in "Cryosphere", 2014, vol. 8, n^{o} 6, pp. 2335-2351 [*DOI :* 10.5194/TC-8-2335-2014], http://www.the-cryosphere.net/8/2335/2014/

[31] F.-X. LE DIMET, O. TALAGRAND. *Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects*, in "Tellus", 1986, vol. 38A, pp. 97-110

[32] B. MOHAMMADI. *Practical application to fluid flows of automatic differentiation for design problems*, in "Von Karman Lecture Series", 1997

[33] N. ROSTAING. *Différentiation Automatique: application à un problème d'optimisation en météorologie*, université de Nice Sophia-Antipolis, 1993

[34] R. RUGINA, M. RINARD. *Symbolic Bounds Analysis of Pointers, Array Indices, and Accessed Memory Regions*, in "Proceedings of the ACM SIGPLAN'00 Conference on Programming Language Design and Implementation", ACM, 2000