Activity Report 2017

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 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
- B9.5.3. - Economy, Finance

1. Personnel

Research Scientists

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

PhD Students

Eléonore Gauci [Inria, until Aug 2017]
Emmanuelle Itam [Univ de Montpellier]
Ala Taftaf [Inria, until Jan 2017]

Intern

Benoit Dufumier [Inria, from Jun 2017 until Aug 2017]

Administrative Assistant

Christine Claux [Inria]

Visiting Scientist

Olivier Allain

External Collaborators

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

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 want 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 now.

Our research directions :

- Efficient adjoint AD of frequent dialects e.g. Fixed-Point loops.
- Development of the adjoint AD model towards Dynamic Memory Management.
- Development of the adjoint AD model towards Parallel Languages.
- 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, Ala Taftaf.

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

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

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.

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$\times$vector products are cheaper than matrix$\times$matrix products. This is the *adjoint mode* of AD, most effective for optimization, data assimilation [30], adjoint problems [24], or inverse problems.

Adjoint AD builds a very efficient program [26], 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, Ala Taftaf, Benoit Dufumier.

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 [17] 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 [20], [27], [18]. 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* [21] 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 [33], [22] 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 input parameters. Consider one particular scalar characteristic. Its sensitivity, (or gradient) with respect to the input parameters can be defined as the solution 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* i.e. to detect and to lower preference to a solution which, although maybe optimal, is very sensitive to uncertainty on design parameters or on manufacturing tolerances. This makes second derivative come into play. Similarly *Uncertainty Quantification* can use second derivatives to evaluate how uncertainty on the simulation inputs imply uncertainty on its outputs.

We investigate several approaches to obtain the gradient, between two extremes:

- One can write an *adjoint system* of mathematical equations, then discretize it and program it by hand. This is time consuming. Although this looks mathematically sound [24], this does not provide the gradient of the discretized function itself, thus degrading the final convergence of gradient-descent optimization.
- One can apply adjoint AD (*cf* 3.1) on the program that discretizes and solves the direct system. This gives exactly the adjoint of the discrete function computed by the program. Theoretical results [23] guarantee convergence of these derivatives when the direct program converges. This approach is highly mechanizable, but leads to massive use of storage and may require code transformation by hand [28], [31] to reduce memory usage.

If for instance the model is steady, or when the computation uses a Fixed-Point iteration, tradeoffs exist between these two extremes [25], [19] that combine low storage consumption with possible automated adjoint generation. We advocate incorporating them into the AD model and into the AD tools.

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,
- 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 [29]) and its AD-adjoint produced by Tapenade.

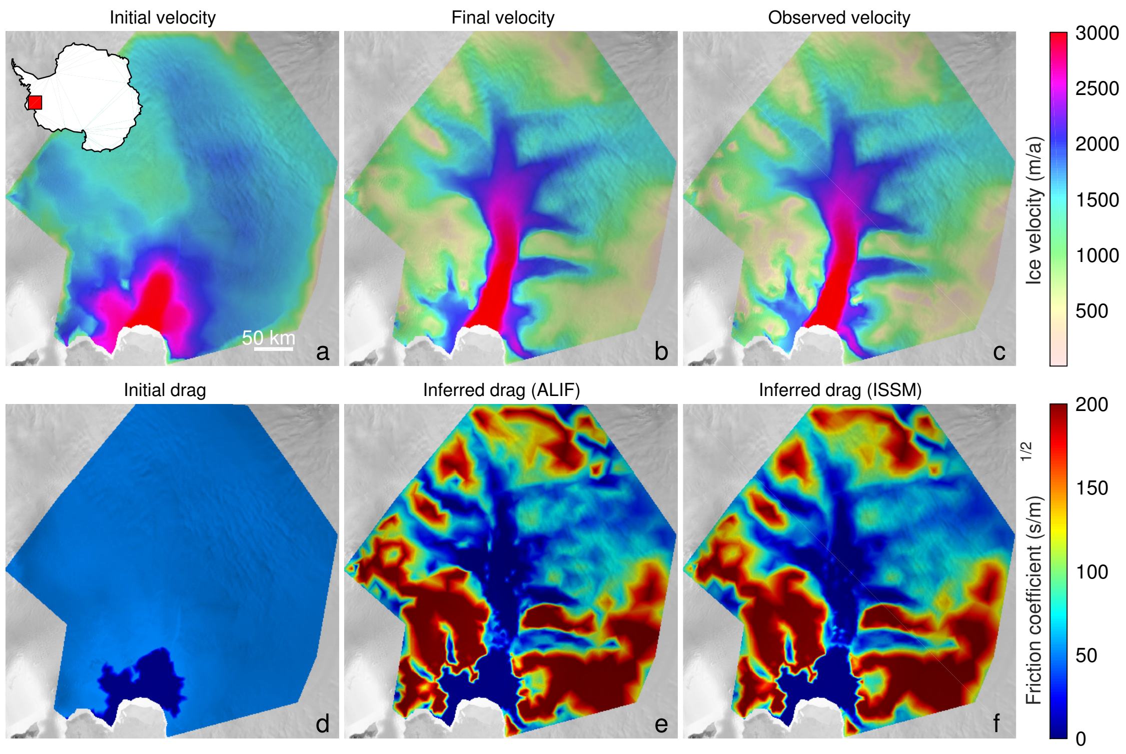

Figure 1. Assimilation of the basal friction under Pine Island glacier, West Antarctica. The final simulated surface velocity (b) is made to match the observed surface velocity (c), by estimation of the basal friction (e). A reference basal friction (f) is obtained by another data assimilation using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

One particular case of inverse problems is *data assimilation* [30] 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 [32]. The special case of *4Dvar* data assimilation is particularly challenging. The 4th 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.

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:

Activity analysis: Detects variables whose derivative is either null or useless, to reduce the number of derivative instructions.

Adjoint Liveness analysis: Detects the source statements that are dead code for the computation of derivatives.

TBR analysis: In adjoint-mode AD, reduces the set of source variables that need to be recovered.
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 - Creation of a front-end for C++ based on Clang - Improved support for the current frontiers of Source-Transformation AD: Dynamic Memory, and MPI

- Participants: Benoît Dufumier, Louis Becquey, 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, Benoit Dufumier, Frederic Cazals [ABS team, Inria Sophia-Antipolis], Louis Becquey [ABS team, Inria Sophia-Antipolis], Valérie Pascual.

We started the extension of Tapenade for C++. This year's first step is to connect an external C++ parser to the input formalism of Tapenade: IL. IL is an abstract language that contains the usual constructs of imperative languages. For example, the three existing parsers for Tapenade (Fortran, Fortran 95, and C) all produce Tapenade input in the form of an IL tree. Our goal was therefore to select a C++ parser or front-end and to make it produce IL trees. In parallel, our goal was also to identify the new operators, specific to Object-Oriented languages, that we must add to IL to capture C++ codes.

During their summer internship, students Benoit Dufumier from SUPELEC and Louis Becquey from INSA Lyon have drafted a C++ front-end for Tapenade, based on the tool "Clang-LLVM". They also added the new operators required into IL, and started the developments in Tapenade to manage them. At present, Tapenade is still unable to differentiate a C++ code, but it can parse and analyze a few toy C++ codes. Still, the latest release 3.13 of Tapenade does not provide any meaningful result for C++ codes yet. This work is going on.

This work benefited from the expertise in C++ of Frederic Cazals (Inria ABS team). Frederic Cazals jointly supervised both students during their internship, funded by the local Inria programme "masters transverses". The ABS team also provided the first C++ test codes and will eventually provide a large application code for Molecular Dynamics.

6.2. AD of mixed-language codes

Participants: Valérie Pascual, Laurent Hascoët.

The tangent differentiated code of Calculix (Three-Dimensional Structural Finite Element code), has been built and validated. Adjoint Differentiation in in progress. Driven by this application to Calculix, Tapenade is now able to differentiate mixed-language source that uses either the old style conventions or the newer Fortran 2003 primitives for interoperability with C.

Unsurprisingly, an application to such a large code uncovered a few limitations of our AD tool. One was a faulty treatment of C translation units (i.e. files), which is now fixed. C translation units or Fortran modules are two instances of the more general notion of “package” for which we need to develop more generic support in Tapenade.

6.3. AD-adjoints and C dynamic memory management

Participants: Laurent Hascoët, Sri Hari Krishna Narayanan [Argonne National Lab. (Illinois, USA)], Mathieu Morlighem [University of California at Irvine (USA)].

One of the current frontiers of AD research is the definition of an adjoint AD model that can cope with dynamic memory management. This research is central to provide reliable adjoint differentiation of C, and for our distant goal of AD of C++. This research is conducted in collaboration with the MCS department of Argonne National Lab. Our partnership is formalized by joint participation in the Inria joint lab JLESC, and partly funded by the Partner University Fund (PUF) of the French embassy in the USA.

Adjoint AD must reproduce in reverse order the control decisions of the original code. In languages such as C, allocation of dynamic memory and pointer management form a significant part of these control decisions. Reproducing memory allocation in reverse means reallocating memory, possibly receiving a different memory chunk. Reproducing pointer addresses in reverse thus require to convert addresses in the former memory chunks into equivalent addresses in the new reallocated chunks. Together with Krishna Narayanan from Argonne, we experiment on real applications to find the most efficient solution to this address conversion problem. We jointly develop a library (called ADMM, ADjoint Memory Management) whose primitives are used in AD adjoint code to handle this address conversion. Both our AD tool Tapenade and Argonne’s tool OpenAD use ADMM in the adjoint code they produce.

This year, the ADMM library has been improved, and its API has been redesigned to be called from both C and Fortran. The architecture of ADMM has also been simplified, removing circular dependencies with other parts of Tapenade. The latest release 3.13 of Tapenade automatically places calls to ADMM where needed, whether the application language is C or Fortran 95. Since ADMM is a C library, the differentiated code in the Fortran case uses the Fortran 2003 standardized interoperability primitives.

In parallel, we improved the Tapenade adjoint of the “ALIF” code. ALIF is developed by Mathieu Morlighem from UC Irvine, jointly with Eric Larour from JPL. This glaciology code is a C clone of the C++ “ISSM” code from JPL. One challenge is the intensive use of dynamic memory in ALIF, following the programming style of its model C++ code ISSM. Although sucessful, the usage of the ADMM library incurred some overhead. We developed a static data-flow analysis to reduce the number of calls to ADMM. This work is discussed in an article published in journal “Optimization Methods and Software”[14]

6.4. Application to large industrial codes

Participants: Valérie Pascual, Laurent Hascoët, Nicole Goutal [EDF-LNHE], Andrea Piacentini [CERFACS-GLOBC], Charlotte Kotas [Oak Ridge National Lab. (Tennessee, USA)].

We support industrial users with their first experiments of Algorithmic Differentiation of large in-house codes.

A previous collaboration with EDF and CERFACS has been continued by a new three-months contract, with the objective of improving the AD adjoint of the hydrodynamic code Mascaret. The tangent and adjoint differentiated codes have been built for the calculation of steady subcritical flow (“Sarap” kernel) with the latest Mascaret Version 8.1. The differentiation process has been simplified and it exploits the latest capacities of Tapenade on Fortran 95. In particular, the differentiated code manages Fortran 95 dynamic memory through our library ADMM. Connection with the C-written ADMM uses the Fortran 2003 standardized interoperability primitives. Validation was conducted on two test cases (named “Garonne” and “Oraison”).

We support AD experiments taking place at Oak Ridge National Laboratory, targetted at building the adjoint of a large CFD application called “Rex”. After one year of collaboration, we reached a first milestone with a working tangent differentiation of a sequential (i.e. non MPI) version of the code. Differentiation in tangent mode is significantly easier than in adjoint mode. Therefore it is a good practice to differentiate first in tangent mode even when the final goal is to produce gradients, which require adjoint mode. Moreover, a validated tangent code is very helpful to validate and debug an adjoint code. The next step will be extension to the MPI-parallel version of the code. This will exploit and develop the AMPI library, co-developped with Argonne National Lab, for automated differentiation of MPI communication routines.

6.5. Multirate methods

Participants: Alain Dervieux, Bruno Koobus, Emmanuelle Itam, Stephen Wornom.

This study is performed in collaboration with IMAG-Montpellier II. It addresses an important complexity issue in unsteady mesh adaptation and takes place in the work done in the ANR Maidesc. Unsteady high-Reynolds computations are strongly penalized by the very small time step imposed by accuracy requirements on regions involving small space-time scales. Unfortunately, this is also true for sophisticated unsteady mesh adaptive calculations. This small time step is an important computational penalty for mesh adaptive methods of AMR type. This is also the case for the Unsteady Fixed-Point mesh adaptive methods developed by Ecuador in cooperation with the Gamma3 team of Inria-Saclay. In the latter method, the loss of efficiency is even more crucial when the anisotropic mesh is locally strongly streched. This loss is evaluated as limiting the numerical convergence order for discontinuities to 8/5 instead of second-order convergence. An obvious remedy is to design time-consistent methods using different time steps on different parts of the mesh, as far as they are efficient and not too complex. The family of time-advancing methods in which unsteady phenomena are computed with different time steps in different regions is referred to as the multirate methods. In our collaboration with university of Montpellier, a novel multirate method using cell agglomeration has been designed and developed in our AIRONUM CFD platform. A series of large-scale test cases show that the new method is much more efficient than an explicit method, while retaining a similar time accuracy over the whole computational domain. The comparison with an implicit scheme shows that the implicit scheme is in most cases one order less accurate due to higher time steps and higher dissipation. For the applications to massively parallel computing, an accurate study has been undertaken in order to analyse the impact of the mesh partitioning on the parallel efficiency. Three options have been considered. The usual partition, minimizing communication under the unique constraint of uniform load over the whole domain is optimal for a part of the algorithm but performs very poorly for the other part. We have also applied the multi-constraint partitioning of Metis which relies on both whole domain balancing and fine-mesh subdomain balancing. This strategy significantly improves the efficiency, but we observed that the balancing of the whole domain phase was not perfect. A third set of experiments relied on a geometrical-based optimal multi-contraint partition which we could apply to most of our geometries and which gave a notable further improvement. An article is submitted to a journal on the basis of the second part of the thesis of Emmanuelle Itam.

6.6. Control of approximation errors

Participants: Eléonore Gauci, Alain Dervieux, Adrien Loseille [Gamma3 team, Inria-Rocquencourt], Frédéric Alauzet [Gamma3 team, Inria-Rocquencourt], Anca Belme [university of Paris 6], Gautier Brèthes [university of Montreal], 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.

Our theoretical studies in mesh adaptation are supported by the ANR project MAIDESC coordinated by ECUADOR and Gamma3, which deals with meshes for interfaces, third-order accuracy, meshes for boundary layers, and curved meshes.

During this year, two works, one on the tensorial metric method started during the thesis of Gautier Brèthes [12], and one on mesh adaptation for third order approximation [13], were completed and published in journals.

Further studies of mesh adaptation for viscous flows are currently performed and a paper in collaboration with Gamma3 and university of Paris 6 (Anca Belme) has been submitted to a Journal.

6.7. Turbulence models

Participants: Alain Dervieux, Bruno Koobus, Stephen Wornom, Maria-Vittoria Salvetti [University of Pisa].

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 (Reynolds averaged Navier-Stokes) models. These are mainly vortical separated regions as assumed in one of the most popular hybrid models, the hybrid Detached Eddy Simulation 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. A little surprisingly, the prediction of vortical wakes needs some improvement. For this improvement, we propose a hybrid formulation involving locally a sophisticated LES-VMS (Large Eddy Simulation - Variational Multi-Scale) model combined with the dynamic local limitation of Germano-Piomelli. This model can be hybridized with a RANS model with some positive outputs. It can also be hybridized with a DDES model with larger benefits. The prediction is better than with RANS and also better than with a pure DDES model. A communication has been presented in the DLES11 conference [15] and an extended article from a conference held last year in Strasbourg has been written (with 2018 results), submitted and accepted for publication in a Springer book entitled “Progress in Hybrid RANS-LES Modelling”(2018).

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

- Ecuador and Lemma have a bilateral contract to share the results of Stephen Wornom, Lemma engineer provided to Inria and hosted by Inria under a Inria-Lemma contract.
- Ecuador and EDF had a three months contract to study the adjoint differentiation of the hydrology code Mascaret.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. ANR

8.1.1.1. MAIDESC

Ecuador is coordinator of the ANR project MAIDESC, with Inria team Gamma3, University of Montpellier II, CEMEF-Ecole des Mines, Inria-Bordeaux, Lemma and Transvalor. MAIDESC concentrates on mesh adaptation and in particular meshes for interfaces, third-order accuracy, meshes for boundary layers, and curved meshes. Project MAIDESC terminated in november 2017.

8.2. International Initiatives

8.2.1. Inria International Labs

Ecuador participates in the Joint Laboratory for Exascale Computing (JLESC) together with colleagues at Argonne National Laboratory. Laurent Hascoët visited Argonne National Laboratory, december 11-18.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific events organisation

9.1.1.1. Member of the organizing committees

- Laurent Hascoët was the local organizer, and one of the speakers/teachers during the 2-days training of the Marie Curie FP7 programme "IODA" on Algorithmic Differentiation, http://ioda.sems.qmul.ac.uk/events/nice2017, february 13-14.
- Laurent Hascoët is on the organizing commitee of the EuroAD Workshops on Algorithmic Differentiation. This year, the team organized the 20th EuroAD workshop at Inria Sophia-Antipolis, http://www.autodiff.org/?module=Workshops&submenu=EuroAD/20/main, february 16-17.
- Alain Dervieux organized two workshops for the ANR project MAIDESC, on may 11th and november 16th.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Master : Laurent Hascoët, Optimisation avancée, 15 h, M2, University of Nice

9.2.2. Supervision

PhD : Ala Taftaf, "Extensions of Algorithmic Differentiation by Source Transformation to meet some needs of Scientific Computing", defended january 17, advisor L. Hascoët

PhD in progress : Éléonore Gauci, "Norm-oriented criteria for CFD and coupled CSM-CFD systems", started october 2014, advisor A. Dervieux

PhD : Emmanuelle Itam, "Simulation numérique d'écoulements autour de corps non profilés par des modèles de turbulence hybrides et un schéma multi-rate", defended november 30, co-advisor A. Dervieux

9.2.3. Juries

- Stephen Wornom, jury, PhD defense of Emmanuelle Itam, Université Montpellier II, november 30.

10. 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] 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

[3] L. HASCOËT, M. ARAYA-POLO. *The Adjoint Data-Flow Analyses: Formalization, Properties, and Applications*, in "Automatic Differentiation: Applications, Theory, and Tools", H. M. BÜCKER, G. CORLISS, P. HOVLAND, U. NAUMANN, B. NORRIS (editors), Lecture Notes in Computational Science and Engineering, Springer, 2005

[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, 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

[8] 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

[9] 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

[10] M. VÁZQUEZ, A. DERVIEUX, B. KOOBUS. *Multilevel optimization of a supersonic aircraft*, in "Finite Elements in Analysis and Design", 2004, vol. 40, pp. 2101-2124

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] A. TAFTAF. *Extensions of algorithmic differentiation by source transformation inspired by modern scientific computing*, Université Côte d'Azur, January 2017, https://tel.archives-ouvertes.fr/tel-01503507

Articles in International Peer-Reviewed Journals

[12] G. BRÈTHES, A. DERVIEUX. *A tensorial-based Mesh Adaptation for a Poisson problem*, in "Revue Européenne de Mécanique Numérique/European Journal of Computational Mechanics", March 2017 [*DOI :* 10.1080/17797179.2017.1310648], https://hal.inria.fr/hal-01512995

[13] A. CARABIAS, A. BELME, A. LOSEILLE, A. DERVIEUX. *Anisotropic Goal-oriented error analysis for a third-order accurate CENO Euler discretization*, in "International Journal for Numerical Methods in Fluids", 2017 [*DOI :* 10.1002/FLD.4423], http://hal.upmc.fr/hal-01579998

[14] L. HASCOET, M. MORLIGHEM. *Source-to-source adjoint Algorithmic Differentiation of an ice sheet model written in C*, in "Optimization Methods and Software", November 2017, pp. 1 - 15 [*DOI :* 10.1080/10556788.2017.1396600], https://hal.inria.fr/hal-01651963

International Conferences with Proceedings

[15] 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, May 2017, pp. 1-6, https://hal.inria.fr/hal-01655171

Conferences without Proceedings

[16] L. HASCOËT. *Some highlights on Source-to-Source Adjoint AD*, in "NIPS 2017 - workshop The future of gradient-based machine learning software & techniques", Long Beach, Californie, United States, December 2017, pp. 1-5, https://hal.inria.fr/hal-01655085

References in notes

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

[18] 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

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

[20] 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

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

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

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

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

[25] 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

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

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

[28] P. HOVLAND, B. MOHAMMADI, C. BISCHOF. *Automatic Differentiation of Navier-Stokes computations*, Argonne National Laboratory, 1997, n° MCS-P687-0997

[29] 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° 6, pp. 2335-2351 [*DOI :* 10.5194/TC-8-2335-2014], http://www.the-cryosphere.net/8/2335/2014/

[30] 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

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

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

[33] 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