RESEARCH CENTRE

Inria Center at Université Côte d'Azur

2022

ACTIVITY REPORT

Project-Team

ECUADOR

Program transformations for scientific computing

DOMAIN

Applied Mathematics, Computation and Simulation

THEME

Numerical schemes and simulations

Contents

Project-Team ECUADOR

Creation of the Project-Team: 2014 January 01

Keywords

Computer sciences and digital sciences

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 members, visitors, external collaborators

Research Scientists

- Laurent Hascoët [Team leader, INRIA, Senior Researcher, HDR]
- Alain Dervieux [INRIA, Emeritus, HDR]

Faculty Member

- Bruno Koobus [UNIV MONTPELLIER]

PhD Students

- Bastien Sauvage [INRIA]
- Stephen Wornom [UNIV MONTPELLIER]

Administrative Assistant

- Christine Claux [INRIA]

2 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.

Glossary

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

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

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

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

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:

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

  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:

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

  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 [35], adjoint problems [28], or inverse problems.

Adjoint AD builds a very efficient program [30, Section 3.3], 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.

Glossary

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 [21] 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 [24, 31, 22]. 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* [25] 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 [39, 26] 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, Stephen Wornom.

Glossary

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 [28]. Theoretical results [27] 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 [32, 37] to master possibly heavy memory usage. Among these additional developments, we promote in particular specialized AD models for Fixed-Point iterations [29, 23], 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,
- adaptation 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 [34]) and its AD-adjoint produced by Tapenade.

One particular case of inverse problems is *data assimilation* [35] 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 [38]. 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.

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

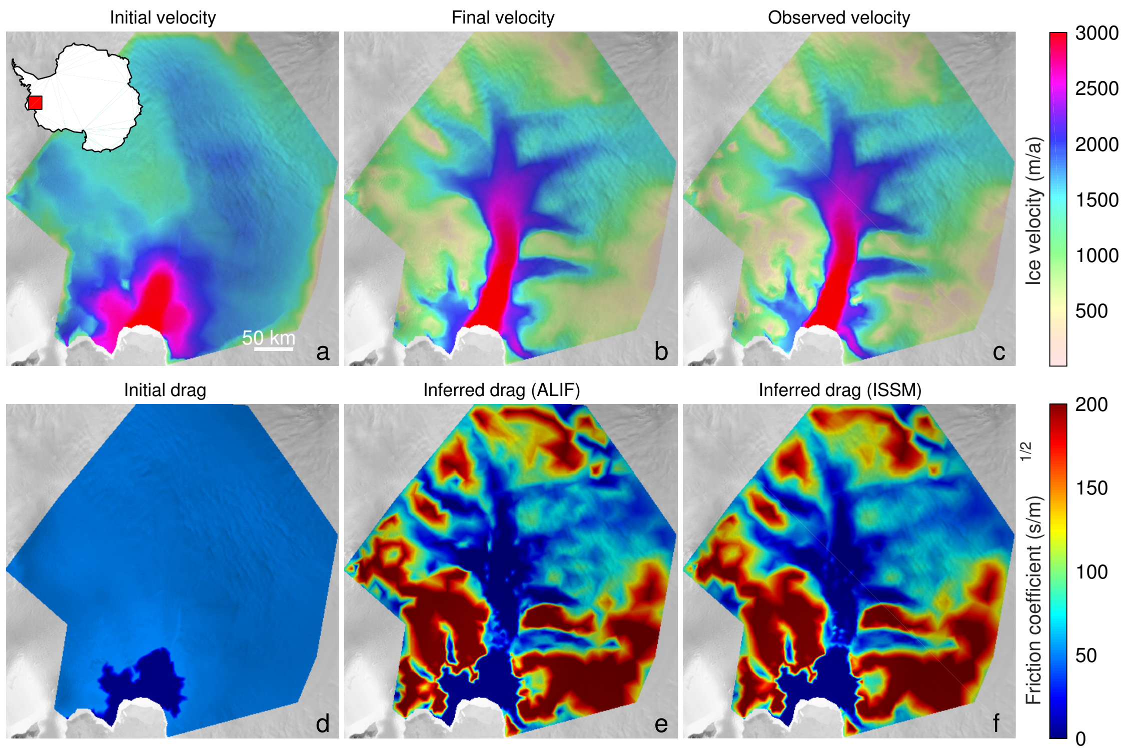

Figure 1: Assimilation of the basal friction under Pine Island glacier, West Antarctica. The final simulated surface velocity (b) is made to match the observed surface velocity (c), by estimation of the basal friction (e). A reference basal friction (f) is obtained by another data assimilation using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

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 Social and environmental responsibility

5.1 Impact of research results

Our research has an impact on environmental research: in Earth sciences, our gradients are used in inverse problems, to determine key properties in oceanography, glaciology, or climate models. For instance they determine basal friction coefficients of glaciers that are necessary to simulate their future evolution. Another example is to locate sources and sinks of CO2 by coupling atmospheric models and remote measurements.

6 New software and platforms

6.1 New software

6.1.1 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.

URL: https://imag.umontpellier.fr/~koobus/norma.html

Contact: Alain Dervieux

Participant: Alain Dervieux

6.1.2 TAPENADE

Name: Tapenade Automatic Differentiation Engine

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: Extension to CUDA (in C) for tangent and adjoint AD. Extension to OpenMP: native usage of Z3 to reduce the need for atomic increments. Usage as a preprocessor to use the Complex-Step tangent AD method. Various extensions requested by end-users: correct adjoint AD of complex arithmetic, in Fortran and in C(99). Compatibility with Fortran sources with fpp pragma. Continued refactoring and bug fixes.

URL: `https://team.inria.fr/ecuador/en/tapenade/`

Contact: Laurent Hascoët

Participants: Laurent Hascoët, Jan Hueckelheim, Michael Vossbeck

7 New results

7.1 Algorithmic Differentiation of OpenMP

Participants: Laurent Hascoët, Jan Hueckelheim *(Argonne National Lab. (Illinois, USA))*.

For multithreaded parallel applications that use OpenMP, it is desirable to also compute the gradients in parallel. Last year we started extension of the AD model of Tapenade (source transformation, association by address, storage on tape of intermediate values) towards correct and efficient differentiation of OpenMP parallel worksharing loops. These loops are the most common idiom in OpenMP programs. This extension concerns the relatively easy tangent-linear mode of AD, but also the more sophisticated adjoint mode where the most interesting problems lie.

The major issue raised by the adjoint mode is the transformation of variable reads into adjoint variable overwrites, more accurately into increments. While there is no parallel conflict between two reads, two concurrent increments can cause a data race, unless they are both atomic. Classical automated detection of independence, for instance through the polyhedral methods used in parallelizing compilers, is as always limited. This year we have pushed forward the idea of extracting information about the memory access patterns of the original code (which we may assume correct and therefore free of data races) and of feeding this information to a theorem prover to which we then ask questions about independence of access patterns of the adjoint code. As a side benefit, we point out that the thorem prover can very well use other clues to answer the question, and can in particular implement its own polyhedral approach like in a parallellizing compiler. Therefore this approach nicely blends two sources of improvement.

Technically, all the information extraction phase is implemented inside Tapenade, as well as gathering the list of independence questions to be asked. The theorem proving part is delegated to a plugin of the Z3 tool, which is now systematically added into Tapenade. The issues of interfacing between different API's are thus reduced to a minimum. This new functionality is available in all distributions of Tapenade.

This extension was presented this year as a poster at the PPoPP'22 conference [20], and later as a full paper at the ICPP'22 conference [12].

7.2 Algorithmic Differentiation of CUDA

Participants: Laurent Hascoët, Sébastien Bourasseau *(ONERA)*, Cedric Content *(ONERA)*, Bruno Maugars *(ONERA)*.

We continued this year our collaboration with ONERA on differentiation of CUDA source. More generally we extended our AD model towards integration of AD as a fundamental component of SoNice. SoNice is the new CFD platform developed at ONERA by Bourasseau, Maugars, Content and colleagues, as a successor to Elsa. This work is supported by a contract between ONERA and INRIA. This contract officially completed in July 2022 but collaboration continues.

This year, we delivered to ONERA a version of Tapenade that can differentiate most of the test cases provided by ONERA, in tangent as well as adjoint mode.

On the issue of differentiation of the CUDA kernels of SoNice, we continued the work by investigating the interest of the formal methods studied in 7.1. This approach can bring the same benefits for CUDA code as for OpenMP, as we showed on example codes with regular array accesses. However, SoNice favors unsructured grids and therefore irregular array indices that do not lend themselves to this approach. The alternative approach is a mesh coloring technique. Unlike a classical mesh coloring that assigns colors to avoid read-write and write-write conflicts inside a given color, the adapted coloring also avoids read-read situations in the original source. This may slightly increase the number of colors, but in return guarantees the absence of conflicts in the adjoint code, and therefore avoids atomic pragmas in the adjoint CUDA code. The reason is that adjoint AD turns reads into increments, and thus read-read situations into write-write conflicts.

Still on the CUDA aspect, we improved the adjoint AD model to further reduce the need for storage of intermediate values in the adjoint CUDA kernels. Adjoint code reuses intermediate values from the primal computation in reverse order, and this is often arranged for through storage. However, CUDA storage capacity is limited and costly. One workaround is to promote recomputation instead of storage. We pushed the capacity of Tapenade to detect possible recomputation, to a level that is sufficient for SoNice. Still, finding an optimal mix of recomputation and storage is an open question.

Other developments required for SoNice have been:

- a better handling by the AD tool of `fpp` directives in Fortran source, similar to the existing handling of `cpp` directives in C source,
- a labeling mechanism to exactly identify the derivative counterpart of any source code fragment,
- an experimental variant of the multi-directional modes of AD where the needed extra dimension is added to arrays in first position instead of last,
- a variant of the tangent mode of AD that automates the manual preparation of a code, needed to apply the popular “complex step” tangent AD method [36].

A joint article has been presented at the ECCOMAS’22 conference in Oslo [13].

7.3 Application to large industrial codes

Participants: Laurent Hascoët, Shreyas Gaikwad *(U. of Texas, Austin, USA)*, Sri Hari Krishna Narayanan *(Argonne National Lab. (Illinois, USA))*, Daniel Goldberg *(University of Edinburgh, UK)*, Michael Vossbeck *(The Inversion Lab, Hamburg, Germany)*.

We support users with their first experiments of Algorithmic Differentiation of large codes. This concerned several Earth Science codes this year.

One set of target applications was addressed by Shreyas Gaikwad, University of Texas at Austin, PhD student supervised by Patrick Heimbach. His goal is to produce the adjoint of glaciology codes. Last year, Shreyas differentiated the SICOPOLIS code in adjoint mode with Tapenade. After some polishing this year, this work is now described in the main documentation of SICOPOLIS, hosted by `readthedocs.io`. An article was written and submitted to JOSS (Journal of Open Source Software).

This year, Shreyas successfuly differentiated the MIT GCM code, in tangent and adjoint mode, with Tapenade. All these codes are Fortran90 codes that have been previously differentiated with OpenAD, the former AD tool developed by Argonne National Lab. Krishna Narayanan provided crucial help and expertise, as he had been in charge of the previous differentiation with OpenAD. Indeed, differentiation with Tapenade did not encounter bugs in the AD tool itself. It also substantially simplified the tool chain that was necessary with OpenAD, a consequence of the different user interface choices of the two AD tools. The new tool chain was verified on most of the standard test cases of the MIT GCM.

A "pull request" of these new developments was submitted in December to the MIT GCM github repository, and is waiting for approval before final integration in the GCM. Documentation of this work was also prepared and submitted in the main documentation of the MIT GCM, hosted by `readthedocs.io`.

Shreyas Gaikwad presented this work at the Fall meeting of American Geophysical Union, 12-16 December 2022, in Chicago.

A sequel to this work is just starting, with Dan Goldberg, about differentiation of the STREAMICE package of the GCM, with Tapenade.

Independently from the above, we supported adjoint differentiation of the BEPS [33] code, a "biosphere" Earth Science simulation model taking vegetation into accout. This code is developed in part by the small company The Inversion Lab in Hamburg, Germany. We collaborate with Michael Vossbeck, engineer with The Inversion Lab, to provide the adjoint of BEPS by Tapenade. This year, we improved the algorithms and differentiation options of Tapenade to simplify the work chain that produces the adjoint of BEPS. This significantly simplifies future maintenance of the system. This work required extensions of Tapenade towards Fortran 2003 features that were not supported before. Differentiation with Tapenade is becoming a strategic part of the business of The Inversion Lab. Michael Vossbeck is progressively getting acquainted with the source of Tapenade, with the objective of contributing to Tapenade development, and of being able to maintain it.

7.4 Aeroacoustics

Participants: Alain Dervieux, Bastien Sauvage, Bruno Koobus *(IMAG, U. of Montpellier)*, Florian Miralles *(IMAG, U. of Montpellier)*, Stephen Wornom *(IMAG, U. of Montpellier)*, Tanya Kozubskaya *(CAALAB, Moscow)*.

The progress in highly accurate schemes for compressible flows on unstructured meshes (together with advances in massive parallelization of these schemes) allows to solve problems previously out of reach. The four-years programme Norma, associating:

- IMAG of Montpellier University (B. Koobus, coordinator),
- Computational AeroAcoustics Laboratory (CAALAB) of Keldysh Institute of Moscow (T. Kozubskaya, head), and
- Ecuador of INRIA Sophia-Antipolis.

is supported by the French ANR and by the Russian Science Foundation. Norma is a cooperation on the subject of extending Computational AeroAcoustics methods to simulate the noise emited by rotating machines (helicopters, aerial vehicles, unmanned aerial vehicles, wind turbines...). A detailed description with progress reports is available at this location.

Sections 7.5, 7.6, 7.7, 7.8 describe the 2022 contributions of Ecuador to the Norma programme.

7.5 Turbulence models

Participants: Alain Dervieux, Bruno Koobus *(IMAG, U. of Montpellier)*, Florian Miralles *(IMAG, U. of Montpellier)*, Stephen Wornom *(IMAG, U. of Montpellier)*, Tanya Kozubskaya *(CAALAB, Moscow)*.

Modelling 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 targeted (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 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 flows involving laminar boundary layers (BL). But industrial flows are complex flows and often combine in a single flow 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.

During 2022, several advances have been obtained. B. Sauvage developed a study of the NACA0021 test case at high angle of attack, an unsteady flow adressed with a DDES model and mesh adaptation. He also focused on the new test case chosen by the Norma consortium, the flow past a NACA0018 at various angles of attack, for which several measurements are available in the literature. A part of the work of B. Sauvage has been presented in CMMF'22 in Budapest and in DLES13 in Udine [15]. B. Sauvage also made studies on approximation, with comparison of several Riemann solvers such as Roe flux difference splitting (FDS), HLLC flux vector splitting (FVS), and a new variant of the Toro-Vazquez FVS. For pure advection, the Toro-Vazquez FVS proved to be less diffusive than HLLC, of comparable diffusion to Roe, while being much more robust than Roe FDS. These properties are important when mesh adaptation algorithms are applied, since these algorithms are more challenging for the approximation.

7.6 Rotating machines

Participants: Alain Dervieux, Didier Chargy *(Lemma, Sophia-Antipolis)*, Bastien Sauvage, Bruno Koobus *(IMAG, U. of Montpellier)*, Florian Miralles *(IMAG, U. of Montpellier)*, Tanya Kozubskaya *(CAALAB, Moscow)*.

The physical problem addressed by Norma involves a computational domain made of at least two components having different rotative motions. The numerical problem of their combination gave birth to many specialized schemes, such as the so-called sliding method, chimera method, immersed boundary method (IBM). The Ecuador team is studying a novel Chimera method, in cooperation with Lemma engineering (Sophia-Antipolis).

In 2022, B. Sauvage continued his study of mesh adaptive calculation of rotation flows. He developed a Multiple Reference Frame method [18] satisfying the Discrete Geometric Conservation Law. The new scheme together with the Chimera scheme are compared [19] with the calculation of the Caradonna rotor, both with mesh adaptation. The results have been presented in two communications, at ECCOMAS [14] in Oslo and CMFF22 [16] in Budapest. B. Sauvage and Didier Chargy made progresses in the computation of the Caradonna-Tung rotor, with a new adapted calculation on 3 million vertices.

7.7 High order approximations

Participants: Alain Dervieux, Bastien Sauvage.

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

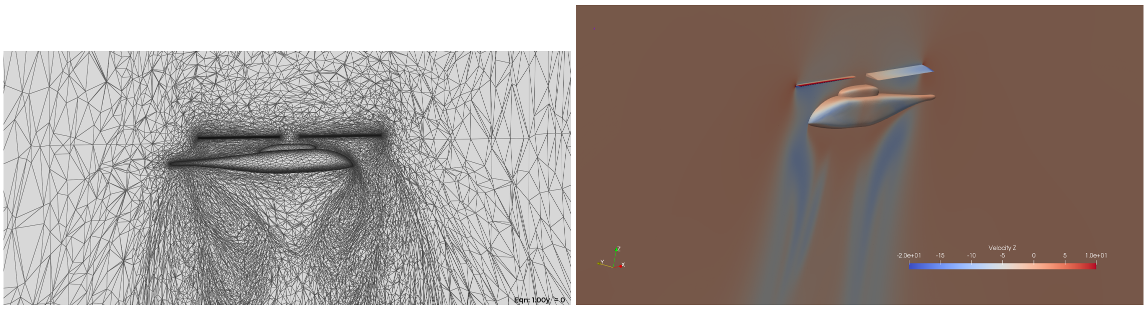

Figure 2: Adaptive calculation in rotor+stator geometry around a Robin+Caradonna-Tung configuration. Adapted mesh (left), fluid velocity (right).

an approximation with a representation inside macro-elements which finally constrains the mesh, and a representation around each individual cell, as in vertex formulations. For this purpose, we extend the Central Essentially Non Oscillating (CENO) family of schemes. We have developed a fourth-order accurate three-dimensional CENO.

In 2022, the consolidation and validation of the fourth-order accurate CENO in LEMMA's code NiceFlow has been advanced further. Alain Dervieux's article [11] on approximation on unstructured meshes has been finally published this year.

7.8 Control of approximation errors

Participants: Alain Dervieux, Bastien Sauvage, Adrien Loseille *(Gamma3 team, INRIA-Saclay)*, Frédéric Alauzet *(Gamma3 team, INRIA-Saclay)*.

Reducing approximation errors as much as possible by modifying the mesh 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.

The first volume of the book [17] on mesh adaptation by Alauzet, Loseille, Koobus and Dervieux has been published by Wiley this year.

B. Sauvage and A. Dervieux made advances in the theory of space-and-time mesh adaptation with the transient fixed point algorithm, in cooperation with F. Alauzet. Adapting the time step is an important issue for aerodynamic LES and hybrid RANS-LES calculations since implicit time advancing with large time steps is used for efficiency reasons while time truncation errors have to be mastered by identifying the best time step.

8 Bilateral contracts and grants with industry

8.1 Bilateral contracts with industry

Participants: Laurent Hascoët, Sébastien Bourasseau *(ONERA)*, Cedric Content *(ONERA)*, Bruno Maugars *(ONERA)*.

The team has a contract with ONERA to assist in the Algorithmic Differentiation of ONERA's new CFD platform "SoNICS". The main objective is adjoint AD of the CUDA parts of the SoNICS source. The contract also includes support to the ONERA development team on advanced use of AD, e.g., fixed-point adjoints and binomial checkpointing. The contract ended formally in July this year.

9 Partnerships and cooperations

9.1 International initiatives

9.1.1 Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program

Participants: Laurent Hascoët, Paul Hovland *(Argonne National Lab)*, Jan Hueckelheim *(Argonne National Lab)*, Sri Hari Krishna Narayanan *(Argonne National Lab)*.

Ecuador participates in the Joint Laboratory for Exascale Computing (JLESC) together with colleagues at Argonne National Laboratory. Laurent Hascoët gave a presentation at the 13th JLESC Workshop (virtual) Dec 14 to 16, 2021, about progress on AD of OpenMP.

10 Scientific production

10.1 Major publications

[1] F. Courty, A. Dervieux, B. Koobus and L. Hascoët. 'Reverse automatic differentiation for optimum design: from adjoint state assembly to gradient computation'. In: *Optimization Methods and Software* 18.5 (2003), pp. 615–627.

[2] B. Dauvergne and 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 and J. Utke. 'An optimized treatment for algorithmic differentiation of an important glaciological fixed-point problem'. In: *Geoscientific Model Development* 9.5 (2016), p. 27. URL: https://hal.inria.fr/hal-01413295.

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

[5] L. Hascoët, U. Naumann and V. Pascual. '"To Be Recorded" Analysis in Reverse-Mode Automatic Differentiation'. In: *Future Generation Computer Systems* 21.8 (2004).

[6] L. Hascoët and V. Pascual. 'The Tapenade Automatic Differentiation tool: Principles, Model, and Specification'. In: *ACM Transactions On Mathematical Software* 39.3 (2013). URL: http://dx.doi.org/10.1145/2450153.2450158.

[7] L. Hascoët, J. Utke and U. Naumann. 'Cheaper Adjoints by Reversing Address Computations'. In: *Scientific Programming* 16.1 (2008), pp. 81–92.

[8] L. Hascoët, M. Vázquez, B. Koobus and A. Dervieux. 'A Framework for Adjoint-based Shape Design and Error Control'. In: *Computational Fluid Dynamics Journal* 16.4 (2008), pp. 454–464.

[9] L. Hascoët and J. Utke. 'Programming language features, usage patterns, and the efficiency of generated adjoint code'. In: *Optimization Methods and Software* 31 (2016), pp. 885–903. DOI: 10.1080/10556788.2016.1146269. URL: https://hal.inria.fr/hal-01413332.

[10] J. C. Hueckelheim, L. Hascoët and J.-D. Müller. 'Algorithmic differentiation of code with multiple context-specific activities'. In: *ACM Transactions on Mathematical Software* (2016). URL: https://hal.inria.fr/hal-01413321.

10.2 Publications of the year

International journals

[11] A. Dervieux. 'To be structured, or unstructured, fifty years of slings and arrows'. In: *Comptes Rendus. Mécanique* (2022). URL: https://hal.inria.fr/hal-03515544.

International peer-reviewed conferences

[12] J. Hückelheim and L. Hascoët. 'Automatic Differentiation of Parallel Loops with Formal Methods'. In: ICPP 2022 - 51st International Conference on Parallel Processing. Bordeaux, France, 29th Aug. 2022. DOI: 10.1145/3545008.3545089. URL: https://hal.inria.fr/hal-03923346.

[13] B. Maugars, S. Bourasseau, C. Content, B. Michel, B. Berthoul, J. N. Ramirez, P. Raud and L. Hascoët. 'Algorithmic Differentiation for an efficient CFD solver'. In: ECCOMAS 2022 - 8th European Congress on Computational Methods in Applied Sciences and Engineering. Oslo, Norway, 5th June 2022. URL: https://hal.archives-ouvertes.fr/hal-03759125.

[14] F. Miralles, B. Sauvage, A. Duben, V. Bobkov, T. Kozubskaya, S. Wornom, B. Koobus and A. Dervieux. 'SIMULATION OF MASSIVELY SEPARATED FLOWS AND ROTATING MACHINE FLOWS USING HYBRID MODELS'. In: European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS). Oslo, Norway, 5th June 2022. URL: https://hal.inria.fr/hal-03929765.

[15] F. Miralles, B. Sauvage, S. Wornom, F. Alauzet, B. Koobus and A. Dervieux. 'SIMULATION OF MASSIVELY SEPARATED FLOWS USING HYBRID TURBULENCE MODELS AND MESH ADAPTATION'. In: Direct and Large-Eddy Simulation 13. Udine, Italy, 26th Oct. 2022. URL: https://hal.inria.fr/hal-03929769.

Conferences without proceedings

[16] F. Miralles, B. Sauvage, S. Wornom, B. Koobus and A. Dervieux. 'Application of hybrid RANS/VMS modeling to rotating machines'. In: CMF 2022 - The 18th International Conference on Modelling Fluid Flow. Budapest, Hungary, 30th Aug. 2022. URL: https://hal.inria.fr/hal-03922768.

Scientific books

[17] A. Dervieux, F. Alauzet, A. Loseille and B. Koobus. *Mesh Adaptation for Computational Fluid Dynamics 1*. 2023. URL: https://hal.inria.fr/hal-03930979.

Reports & preprints

[18] D. Chargy and B. Sauvage. *A mesh adaptative method for rotating machines*. RR-9471. Inria, 2022, p. 28. URL: https://hal.inria.fr/hal-03684715.

[19] F. Miralles, B. Sauvage, S. F. Wornom, B. Koobus and A. Dervieux. *Hybrid RANS/DVMS modeling for static and rotating obstacles*. RR-9464. INRIA, 11th Mar. 2022, p. 26. URL: https://hal.inria.fr/hal-03605874.

Other scientific publications

[20] J. Hückelheim and L. Hascoët. 'Automatic differentiation of parallel loops with formal methods'. In: PPoPP 2022 / 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Seoul, South Korea: ACM, 2nd Apr. 2022, pp. 463–464. DOI: 10.1145/3503221.3508442. URL: https://hal.inria.fr/hal-03923261.

10.3 Cited publications

[21] A. Aho, R. Sethi and J. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1986.

[22] I. Attali, V. Pascual and C. Roudet. *A language and an integrated environment for program transformations*. research report 3313. INRIA, 1997. URL: http://hal.inria.fr/inria-00073376.

[23] B. Christianson. 'Reverse accumulation and implicit functions'. In: *Optimization Methods and Software* 9.4 (1998), pp. 307–322.

[24] D. Clément, J. Despeyroux, L. Hascoët and G. Kahn. 'Natural semantics on the computer'. In: *Proceedings, France-Japan AI and CS Symposium, ICOT* (1986). Ed. by K. Fuchi and M. Nivat. Also, Information Processing Society of Japan, Technical Memorandum PL-86-6. Also INRIA research report # 416, pp. 49–89. URL: http://hal.inria.fr/inria-00076140.

[25] P. Cousot. 'Abstract Interpretation'. In: *ACM Computing Surveys* 28.1 (1996), pp. 324–328.

[26] B. Creusillet and F. Irigoin. 'Interprocedural Array Region Analyses'. In: *International Journal of Parallel Programming* 24.6 (1996), pp. 513–546.

[27] J. Gilbert. 'Automatic differentiation and iterative processes'. In: *Optimization Methods and Software* 1 (1992), pp. 13–21.

[28] M.-B. Giles. 'Adjoint methods for aeronautical design'. In: *Proceedings of the ECCOMAS CFD Conference*. Swansea, U.K., 2001.

[29] A. Griewank and C. Faure. 'Reduced Gradients and Hessians from Fixed Point Iteration for State Equations'. In: *Numerical Algorithms* 30(2) (2002), pp. 113–139.

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

[31] L. Hascoët. 'Transformations automatiques de spécifications sémantiques: application: Un vérificateur de types incremental'. PhD thesis. Université de Nice Sophia-Antipolis, 1987.

[32] P. Hovland, B. Mohammadi and C. Bischof. *Automatic Differentiation of Navier-Stokes computations*. Tech. rep. MCS-P687-0997. Argonne National Laboratory, 1997.

[33] W. Ju, J. Chen, T. Black, A. Barr, J. Liu and B. Chen. 'Modelling multi-year coupled carbon and water fluxes in a boreal aspen forest'. In: *Agric. For. Meteorol.* 140.1-4 (2006), pp. 136–151. DOI: 10.1016/j.agrformet.2006.08.008.

[34] E. Larour, J. Utke, B. Csatho, A. Schenk, H. Seroussi, M. Morlighem, E. Rignot, N. Schlegel and 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* 8.6 (2014), pp. 2335–2351. DOI: 10.5194/tc-8-2335-2014. URL: http://www.the-cryosphere.net/8/2335/2014/.

[35] F.-X. Le Dimet and O. Talagrand. 'Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects'. In: *Tellus* 38A (1986), pp. 97–110.

[36] J. Martins and A. Ning. *Engineering design optimization*. Cambridge University press, 2021. DOI: https://doi.org/10.1017/9781108980647.

[37] B. Mohammadi. 'Practical application to fluid flows of automatic differentiation for design problems'. In: *Von Karman Lecture Series* (1997).

[38] N. Rostaing. 'Différentiation Automatique: application à un problème d'optimisation en météorologie'. PhD thesis. université de Nice Sophia-Antipolis, 1993.

[39] R. Rugina and 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.