RESEARCH CENTRE

Sophia Antipolis - Méditerranée

2021 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]
- Valérie Pascual [Inria, Researcher, until Jul 2021]

PhD Students

- Matthieu Gschwend [Inria, until Jul 2021]
- Bastien Sauvage [Inria, from Oct 2021]

Administrative Assistant

- Christine Claux [Inria]

External Collaborator

- Bruno Koobus [Univ de Montpellier]

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, Valérie Pascual.

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

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

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 [28], adjoint problems [22], or inverse problems.

Adjoint AD builds a very efficient program [24, 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, Valérie Pascual.

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 [15] 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, 25, 16]. 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 [32, 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, Matthieu Gschwend, 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 [22]. Theoretical results [21] 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 [26, 30] to master possibly heavy memory usage. Among these

additional developments, we promote in particular specialized AD models for Fixed-Point iterations [23, 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,
- 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 [27]) and its AD-adjoint produced by Tapenade.

One particular case of inverse problems is *data assimilation* [28] 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 [31]. 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.

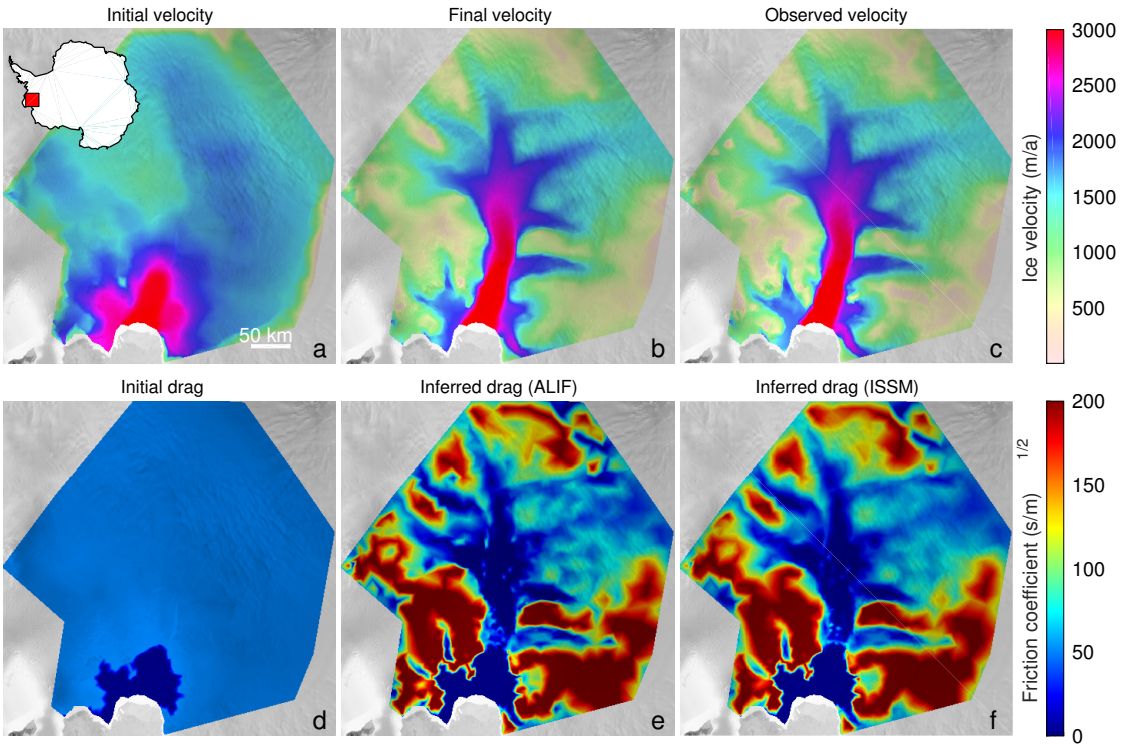

Figure 1: Assimilation of the basal friction under Pine Island glacier, West Antarctica. The final simulated surface velocity (b) is made to match the observed surface velocity (c), by estimation of the basal friction (e). A reference basal friction (f) is obtained by another data assimilation using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

Here we describe new or upgraded software.

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): at this stage only for tangent AD. Various extensions requested by end-users: C++ "&" reference notation, Fortran2003 object constructs. Extension to OpenMP: support for experimental proof of absence of increment-increment conflicts. Continued refactoring and bug fixes. Draft developer's documentation.

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

Contact: Laurent Hascoët

Participants: Laurent Hascoët, Valerie Pascual

7 New results

7.1 Algorithmic Differentiation of OpenMP

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

For applications that are parallelized for multi-core CPUs or GPUs using OpenMP, it is desirable to also compute the gradients in parallel. We extended 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, one of the most commonly used OpenMP features, in tangent-linear and adjoint mode. This work was published in ACM TOMS [12].

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 is as always limited. We propose to gather information about the memory access patterns of the original code, which is assumed correct and therefore free of data races, and to reuse this information into a theorem prover with which we check the safety of the shared memory accesses of the adjoint code.

A poster was accepted for presentation at PPoPP'22. An article is in preparation.

7.2 Algorithmic Differentiation of CUDA

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

In collaboration with ONERA, we study extension of Tapenade to the parallel constructs of CUDA. The industrial objective is to include Adjoint AD natively into the successor of ONERA's "Elsa" CFD platform. Our research objective is to extend our adjoint AD model to CUDA code. While this extension bears similarity with our work on OpenMP, several specific aspects of CUDA deserve a specific treatment. For instance the stack storage mechanism central to our adjoint AD model must be redesigned to take into accout the memory limitations of GPU code sections.

This year, we delivered to ONERA a version of Tapenade that can parse and regenerate CUDA C source, and that can differentiate in tangent mode a few of the test cases provided by ONERA.

A joint article is in preparation about the general architecture of ONERA's new CFD solver, that includes a section on integrating AD into the compilation chain and on the needed adaptions of Tapenade.

7.3 Adjoint Differentiation and Garbage Collection

Participants: Laurent Hascoët.

Data-Flow reversal is at the heart of our model of Source-Transformation Adjoint Algorithmic Differentiation (Adjoint ST-AD). The presence of addresses, pointers, and pointer arithmetics in the target language poses challenges to Data-Flow reversal, which we can deal with in languages such as C. However, when the target language uses Garbage Collection (GC), for instance Java or Python, the notion of address that is used by Data-Flow reversal disappears. Moreover, GC is asynchronous and does not appear explicitly in the source. We studied an extension to the model of Adjoint ST-AD suitable for a language with GC. We validated this approach on a Java implementation of a simple Navier-Stokes solver. We compared performance with alternative models of Adjoint AD, such as Overloading-based AD (e.g. ADOL-C) which by design could handle GC more easily.

An article has been written and is currently under review with ACM TOMS. A research report has been published [14].

7.4 Application to large industrial codes

Participants: Laurent Hascoët, Shreyas Gaikwad *(U. of Texas, Austin, USA)*, Sri Hari Krishna Narayanan *(Argonne National Lab. (Illinois, USA))*, Hervé Guillard *(CASTOR, INRIA Sophia-Antipolis)*.

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

One application was done by Shreyas Gaikwad, University of Texas at Austin, PhD student supervised by Patrick Heimbach. His goal is to produce the adjoint of glaciology codes such as SICOPOLIS and the glaciology component of the MIT GCM. Both are Fortran 90 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 tool itself. It rather underlined interface and documentation difficulties to apply special recommended strategies e.g. for adjoint AD of linear solvers.

The other code this year is CTFEM, an element of the code suite developed by the CASTOR team of INRIA and University of Nice for the ITER project. CTFEM is a plasma simulation code written in Fortran90. Together with Hervé Guillard, we applied Tapenade to produce the adjoint code of CTFEM. In addition to several Tapenade bugs, now fixed, we encountered two more interesting issues. One is that CTFEM introduces memory aliasing at a few locations. Classically, memory aliasing is adverse to adjoint AD and should be avoided. In general, the AD tool emits a warning message when potential aliasing is detected. It unfortunately failed to do so in a few particular cases. The other issue is about array notation: Fortran90 actual parameters of calls that are arrays are in principle passed by reference, allowing the called procedure to modify the actual parameter. This is also true for array sections, i.e. actual parameters of the form `T(0:10:2)`. On the other hand, an array section that uses an indirection

such as `T(ind(0:10:2))` appears to be passed by value, although we found no literature on the subject. Tapenade now points out this adverse situation, which can be solved by a simple local code rewrite.

7.5 Aeroacoustics

Participants: Alain Dervieux, Matthieu Gschwend, 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 the extension of Computational AeroAcoustics methods to the simulation the noise emission by rotating machines (helicopters, future aerial taxis, unmanned aerial vehicles, wind turbines...). The tasks of INRIA in this Russian-French cooperation program are:

- the advancement of two numerical techniques, namely:
  - a higher-order approximation scheme for compressible Navier-Stokes equations, and
  - mesh adaptation methods for the flows under study (DES modeled flows able to model noise generation).
- a contribution to the Norma test cases, in particular the quad-rotor drone.

Among this year's results:

- We delivered a review on High-Order methods for compressible CFD.
- We defined CENO3D, a 3D finite-volume scheme relying on cell-based reconstruction and fourth-order accurate, and we developed and tested it in a CFD software.
- We installed an anisotropic metric-based mesh adaptation for rotating machines and we validated it on a first test case, a mixing device rotating in a cylinder. The study of the next test case, the Caradonna-Tung helix (Norma test case 1.1.) is under progress.
- We started the design of a new mesh adaptation based on H-P principle (simultaneous adaptation of the mesh and of the scheme accuracy), in order to combine the higher-order scheme CENO3D with our anisotropic metric-based mesh adaptation technology.

7.6 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)*.

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 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 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.

During the first phase of Norma, Montpellier and Moscow are computing a series of initial test cases in order to control the consistancy of the results produced by the two platforms of CFD, namely Noisette for Moscow, and Aironum for Montpellier.

A communication in seminar was presented by Florian Miralles [29]

7.7 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). In concertation with Moscow, Montpellier is introducing a novel IBM in the CFD code Aironum. The Ecuador team is studying in cooperation with Lemma engineering (Sophia Antipolis) a novel sliding/chimera method.

7.8 High order approximations

Participants: Alain Dervieux, Matthieu Gschwend, Bastien Sauvage.

After many decades of research which we suumarized in [11], approximation of unstructured meshes have become the common practice in CFD. 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. For this purpose, we extend the Central Essentially Non Oscillating (CENO) family of schemes. We have developed a fourth-order accurate three-dimensional CENO. This work is documented in a research report [13].

7.9 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 changing 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.

This year, we extended mesh adaptation methods to the simulation of rotating machines is a special unsteady periodic flow. We also continued our new analysis for h-p anisotropic mesh adaptation.

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. This contract will finance the work of a development engineer for two years. The contract also includes support to the ONERA development team on advanced use of AD, e.g., fixed-point adjoints and binomial checkpointing.

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.

We gave a short presentation at the last JLESC meeting, about research on Adjoint AD of OpenMP. This is a joint work with Jan Hueckelheim, Argonne National Lab.

10 Dissemination

Participants: Laurent Hascoët.

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

Together with the MCS division of Argonne National Lab, the Ecuador team co-organized a 2-days tutorial "Automatic Differentiation as a Tool for Computational Science" at the SIAM CSE conference, on March 1-5. Laurent Hascoët gave three presentations there.

Laurent Hascoët is on the organizing commitee of the EuroAD Workshops on Algorithmic Differentiation (www.autodiff.org), taking place twice a year, with exceptions. Organization rotates between RWTH Aachen, University of Jena, Humboldt University Berlin, INRIA Sophia-Antipolis, and Argonne National Lab. The 24th EuroAD workshop was organized by the Ecuador team, remotely, on December 14-16.

10.1.2 Scientific events: selection

Laurent Hascoët was on the program committee of the "Differentiable Programming Workshop" at NeurIPS2021, December 6-14.

11 Scientific production

11.1 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.

11.2 Publications of the year

International journals

[11] A. Dervieux. 'To be structured, or unstructured, fifty years of slings and arrows'. In: *Comptes Rendus de l'Académie des sciences Série 2 - Mécanique-physique, Chimie, Sciences de l'univers, Sciences de la Terre* (2022). URL: https://hal.inria.fr/hal-03515544.

[12] J. Hückelheim and L. Hascoët. 'Source-to-Source Automatic Differentiation of OpenMP Parallel Loops'. In: *ACM Transactions on Mathematical Software* (2021). URL: https://hal.inria.fr/hal-03429677.

Reports & preprints

[13] M. Gschwend. *A 3D vertex centered CENO scheme for advection*. RR-9415. Inria Sophia Antipolis - Méditerranée, 12th July 2021, p. 37. URL: https://hal.inria.fr/hal-03284753.

[14] L. Hascoet. *Data-Flow reversal and Garbage Collection*. RR-9416. Inria Sophia Antipolis - Méditerranée, July 2021, p. 18. URL: https://hal.inria.fr/hal-03291836.

11.3 Cited publications

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

[16] 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.

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

[18] 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.

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

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

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

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

[23] 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.

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

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

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

[27] 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/.

[28] 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.

[29] F. Miralles, S. Wornom, B. Koobus and A. Dervieux. 'Hybrid 3D simulations on circular cylinder at Re=1M. Comparison between DDES, RANS/DVMS, DDES/DVMS'. In: 33rd Nordic Seminar on Computational Mechanics, Jönköping, 25-26 november. 2021.

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

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

[32] 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.