Activity Report 2014

Project-Team ECUADOR

Algorithmic Differentiation for Scientific Computing

RESEARCH CENTER
Sophia Antipolis - Méditerranée

THEME
Numerical schemes and simulations

Table of contents

Project-Team ECUADOR

Keywords: Scientific Computation, Fluid Dynamics, Automatic Differentiation, Program Transformation, Parallelism

Creation of the Project-Team: 2014 January 01.

1. Members

Research Scientists

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

PhD Students

Gautier Brèthes [Inria, LEMMA]
Eléonore Gauci [Inria, from Apr 2014]
Ala Taftaf [Inria]

Visiting Scientists

Bruno Koobus [University Montpellier II]
Stephen Wornom [LEMMA]

Administrative Assistant

Claire Senica [Inria]

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 Derivative Equations that define a problem, obtaining new differential equations that define the gradient of the original problem's solution.

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 [14].

The resulting P' reuses the control of P. For any given control, P is equivalent to a sequence of instructions, which is identified with a composition of vector functions. Thus, if

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

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. Combining the I'_k with the control of P yields P', and therefore this differentiation is piecewise.

AD can be generalized to higher level derivatives, Taylor series, etc. In practice, many applications only need cheaper projections of $F'(X)$ such as:

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

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

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

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

$$F'^*(X).\overline{Y} = f'^*_1(X_0).f'^*_2(X_1). \cdots .f'^*_{p-1}(X_{p-2}).f'^*_p(X_{p-1}).\overline{Y} \quad . \tag{4}$$

  This expression is most efficiently computed from right to left, because matrix×vector products are cheaper than matrix×matrix products. This defines the *adjoint mode* of AD, most effective for optimization, data assimilation [28], adjoint problems [23], or inverse problems.

Adjoint-mode AD turns out to make a very efficient program, at least theoretically [25]. The computation time required for the gradient is only a small multiple of the run-time of P. It is 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-mode 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 communication. 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.

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.

data dependence analysis Program analysis that studies the itinerary of values during program execution, from the place where a value is defined to the places where it is used, and finally to the place where it is overwritten. The collection of all these itineraries is stored as *Def-Use and Use-Def chains* or as a *data dependence graph*, and data flow analysis most often rely on this graph.

data dependence graph Directed graph that relates accesses to program variables, from the write access that defines a new value to the read accesses that use this value, and from the read accesses to the write access that overwrites this value. Dependences express a partial order between operations, that must be preserved to preserve the program's result.

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 use sophisticated analysis [15]. These tools share their technological basis. More importantly, there are common mathematical models to specify and analyze them.

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* and *back-ends*. Abstraction can go further: the internal representation becomes more language independent, and semantic constructs can be unified. Analysis can then concentrate on the semantics of a small set of constructs. We advocate an internal representation composed of three levels.

- At the top level is the *call graph*, whose nodes are modules and procedures. Arrows relate nodes that call or import one another. Recursion leads to cycles.

- At the middle level is the *flow graph*, one per procedure. It captures the control flow between atomic instructions. Loops lead to cycles.
- At the lowest level are abstract *syntax trees* for the individual atomic instructions. Semantic transformations can benefit from the representation of expressions as directed acyclic graphs, sharing common sub-expressions.

To each level belong 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], [26], [16]. But many analyses are more complex, and better defined on graphs than on trees. This is the case for *data-flow analyses*, that look for run-time properties of variables. Since flow graphs may be cyclic, these global analyses generally require an iterative resolution. *Data flow equations* is a practical formalism to describe data-flow analyses. Another formalism is described in [19], which is more precise because it can distinguish separate *instances* of instructions. However it is still based on trees, and its cost forbids application to large codes. *Abstract Interpretation* [20] 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], [21] or pointers. In general, conservative *over-approximations* are always made that lead to derivative code that is less efficient than possibly achievable.

3.3. Algorithmic Differentiation and Scientific Computing

Participants: Alain Dervieux, Laurent Hascoët, Bruno Koobus.

linearization In Scientific Computing, the mathematical model often consists of Partial Derivative 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 Derivative 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 optimization steps until an optimum is reached. We focus on gradient-based optimization.

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 mathematically sound [23], but very costly in development time. It also does not produce an exact gradient of the discrete function, and this can be a problem if using optimization methods based on descent directions.

- One can apply adjoint-mode AD (*cf* 3.1) on the program that discretizes and solves the direct system. This gives in fact the adjoint of the discrete function computed by the program. Theoretical results [22] 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 [27], [30] to reduce memory usage.

If for instance the model is steady, or more generally when the computation uses a Fixed-Point iteration, tradeoffs exist between these two extremes [24], [17] 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 output criterion with respect to all the inputs, and possibly additional gradients when there are constraints. Adjoint-mode 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 the altitude and speed of the surface.

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]. Figure 1 shows an example of a data assimilation exercise using the oceanography code OPA [29] and its AD adjoint produced by Tapenade.

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.

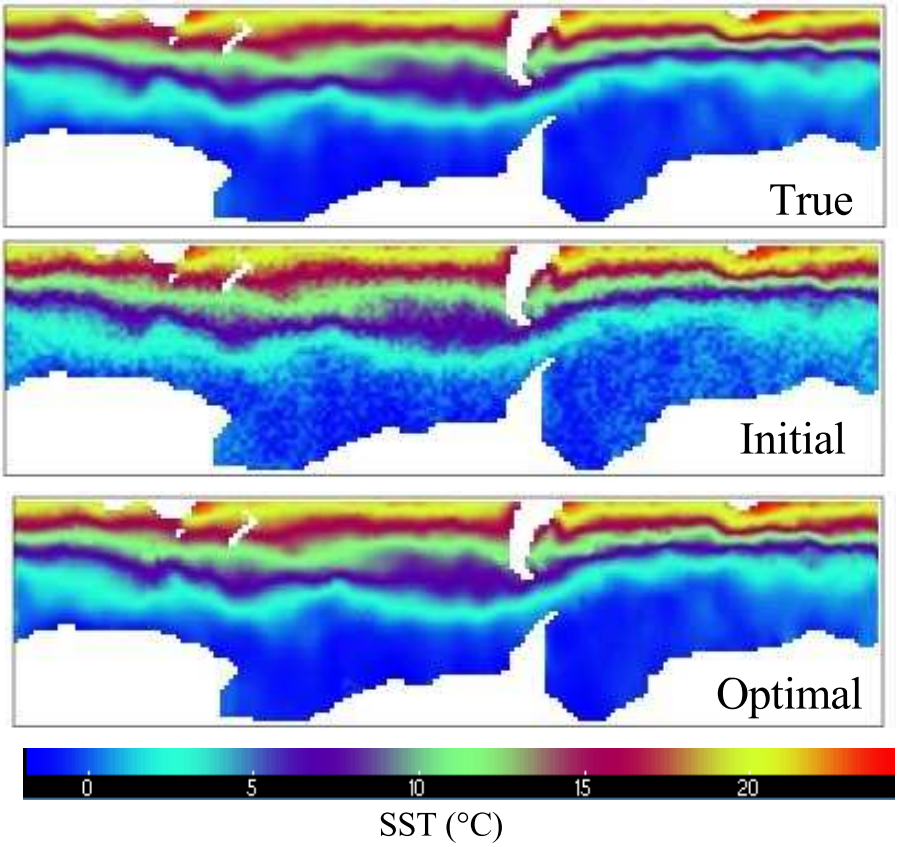

Figure 1. Twin experiment using the adjoint of OPA. Random noise was added to a simulation of the sea surface temperature around the Antarctic, and we remove this noise by minimizing the discrepancy with the physical model

4.4. Linearization

Simulating a complex system often requires solving a system of Partial Differential Equations. This can be too expensive, in particular in the context of real time. 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

Participant: Alain Dervieux [correspondant].

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. See http://www-sop.inria.fr/tropics/aironum

- Version: v 1.0
- Programming language: Fortran95 (mostly). About 100,000 lines.

Aironum was developed by Inria and University of Montpellier. It is used by Inria, University of Montpellier and University of Pisa (I). Aironum is used as an experimental platform for:

- Numerical approximation of compressible flows, such as upwind mixed element volume approximation with superconvergence on regular meshes.
- Numerical solution algorithms for the implicit time advancing of the compressible Navier-Stokes equations, such as parallel scalable deflated additive Schwarz algorithms.
- Turbulence modelling such as the Variational Multiscale Large eddy Simulation and its hybridization with RANS statistical models.

5.2. TAPENADE

Participants: Laurent Hascoët [correspondant], Valérie Pascual, Ala Taftaf, Jan Hueckelheim [Queen Mary University of London].

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. Documentation is provided on http://www-sop.inria.fr/tropics/tapenade.html, in Inria technical report RT-0300, and in [9].

- Version: v3.9, r5092, February 2014
- ACM: D.3.4 Compilers; G.1.0 Numerical algorithms; G.1.4 Automatic differentiation; I.1.2 Analysis of algorithms
- AMS: 65K10; 68N20
- APP: IDDN.FR.001.040038.002.S.P.2002.000.10600
- Keywords: algorithmic differentiation, adjoint, gradient, optimisation, inverse problems, static analysis, data-flow analysis, compilation
- Programming language: Java

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.

Tapenade is not open-source. Academic usage is free for one year. Other usages require a paying license, as detailled on the web page. Ten industrial licences have been sold. Tapenade has been downloaded several hundred times, and the web tool served several thousands of true connections (robots and crawlers excluded) The tapenade-users mailing list is over one hundred registered users.

6. New Results

6.1. Resolution of linearised systems

Participants: Olivier Allain [Lemma], Gautier Brèthes, Alain Dervieux, Bruno Koobus, Stephen Wornom.

For Fluid Mechanics as well as for Structural Mechanics, implicit time-advancing is mandatory. It can be applied efficiently if the large systems involved are solved with a good parallel algorithm. In the 90's, a generation of solution algorithms was devised on the basis of Domain Decomposition Methods (DDM). For complex models such as compressible flows, Schwarz DDM were combined with quasi-Newton algorithms like GMRES. For example in the Aironum tool, we use Restrictive Additive Schwarz (RAS, developed by Cai and Farhat). RAS is an ancestor of the widely used class of Newton-Krylov-Schwarz (NKS) algorithms. NKS are, in some versions including RAS, almost scalable *i.e.* their convergence rate is independant of the number of processors. But scalability degrades over a thousand processors. During the ANR ECINADS, coordinated by Ecuador, a Coarse-Grid Deflated RAS was developed. The algorithmic scalability (iteration-wise) holds for all part, except for the coarse grid direct solver, which concerns a much smaller problem. Effective Convergence Scalability (ECS) was confirmed up to 2048 processors. After this level, the asymptotic complexity of the coarse-grid direct solver become predominant and ECS is lost. In other words, with a Coarse-Grid Deflated RAS, the size of the coarse grid problem must be limited in order to enjoy ECS.

In the thesis of Gautier Brèthes, we now study a further step towards super-massive scalability: we use a number of fine and medium grids in order to solve the target large system by a multi-mesh multigrid (MG) algorithm. An important novelty is that the complete FMG technology is applied, with a new stopping criterion avoiding useless cycling [12]. It is well-known that parallel MG is limited to "no-too-coarse" coarse levels due to an excessive ratio between communication and computation. Now our parallel MG can be complemented by the previous version of the solver (deflated RAS) for this no-too-coarse coarse level.

6.2. Algorithmic Differentiation of a CFD code

Participants: Valérie Pascual, Laurent Hascoët, Alain Dervieux.

This activity continues in collaboration with the partners of the FP7 project UMRIDA. The team is assisting Alenia-Aermacchi in the efficient differentiation of its Euler/Navier Stokes UNS3D code in tangent mode, using in particular a differentiable extension of the MPI library.

Inside a collaboration with EDF, Valérie Pascual is also applying Tapenade to produce various adjoint differentiations of the hydrographic code Mascaret.

6.3. Control of approximation errors

Participants: Gautier Brèthes, Eléonore Gauci, Alain Dervieux, Adrien Loseille [Gamma team, Inria-Rocquencourt], Frederic Alauzet [Gamma team, Inria-Rocquencourt], Stephen Wornom, Olivier Allain [Lemma], Anca Belme [University Paris VI].

A study of an interesting combination of Full Multigrid (FMG) and Anisotropic mesh Adaptation (AA) started last year, with the beginning of the thesis of Gautier Brèthes. FMG is one of the (very) few algorithm giving N results by consuming kN floats. Anisotropic adaptation produces approximation errors less than ε with $N = \varepsilon^{-\frac{1}{dim}}$ nodes, this for smooth and non-smooth solution fields. Anisotropic adaptative FMG may produce approximation errors less than ε by consuming $k\varepsilon^{-\frac{1}{dim}}$ floats. Moreover, theory and experiments show that FMG works better when combined with AA. A first AA-FMG platform has been developed. It combines several mesh-adaptation modules developed by Gamma and Distene. It is used for testing new adaptation criteria.

Third-order mesh adaptation was the main topic of last year in error control. The scheme is the ENO finite-volume formulation with quadratic reconstruction. An article describing our results for 2D applications is being written. A 3D version is developed in the Aironum CFD platform. A cooperation with Lemma is also running, with Eléonore Gauci, to apply the scheme to fluid-gas interfaces. Further studies of mesh adaptation for viscous flows are ongoing and an article in collaboration with Gamma3 and University Paris VI (Anca Belme) is being written.

An important novelty in mesh adaptation is the norm-oriented AA method. The method relies on the definition of ad hoc correctors. It has been developed in the academic FMG platform for elliptic problems. Another version is developed by Gamma, in collaboration with Ecuador, for the compressible flow models. The purpose is to devise a composite algorithm in which an approximation error norm can be specified by the user. The introduction of the norm-oriented idea considerably amplifies the impact of adjoint-based AA. The applied mathematician and the engineer now have methods when faced to mesh adaptation for the simulation of a complex PDE system, since they can specify which error norm level they wish, and for which norm. Eléonore Gauci starts a thesis, co-advised by Alain Dervieux and Frédéric Alauzet, on the norm-oriented criteria for CFD and coupled CSM-CFD systems. She also works on a new version of the mesh adaptive CFD demonstrator of Gamma3. This new version improves the resolution of curved features. A cooperation is also starting between Gautier Brèthes and Thierry Coupez (Ecole Centrale de Nantes) on discrete metrics.

These studies are supported by an European FP7 project UMRIDA which deals with the application of AA to approximation error modelling and control, and by ANR project MAIDESC coordinated by Ecuador, which deals with meshes for interfaces, third-order accuracy, meshes for boundary layers, and curved meshes.

6.4. Turbulence models

Participants: Emmanuelle Itam [University Montpellier II], Alain Dervieux, Bruno Koobus, Carine Moussaed [University Montpellier II], Maria-Vittoria Salvetti [University of Pisa], Stephen Wornom, Bruno Sainte-Rose [Lemma].

The purpose of our work in hybrid RANS/LES is to develop new approaches for industrial applications of LES-based analyses. This year, many experiments have validated the dynamic version of our VMS-LES. The quality of simulations is either comparable to non-dynamic, or better. In the applications targetted (aeronautics, hydraulics), the Reynolds number can be as high as several tenth millions, far too high for pure LES models. However, certain regions in the flow can be better predicted 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 model, 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. This year, we have started the study of multiple-body flows. A typical case is the interaction between two parallel cylinders, one being in the wake of the other. A recent workshop showed the rather disastrous predictions of LES models. Most hybrid models behave better, mainly for the first cylinder. We are progressively extending and validating our VMS-LES model and our hybrid ones ([11]).

6.5. AD tools infrastructure

Participants: Laurent Hascoët, Paul Hovland [Argonne National Lab. (Illinois, USA)], Sri Hari Krishna Narayanan [Argonne National Lab. (Illinois, USA)].

We have an ongoing partnership with Paul Hovland's team at Argonne National Lab, formalized by joint participation in the Inria-Illinois joint lab on petascale computing and with travels funded by the Partner University Fund (PUF) of the French embassy in the USA.

In this framework, we worked on the goal of blending our AD tool Tapenade with Argonne's tool OpenAD, buy developing bridges between their internal representations, through a common external representation of analyzed programs. This representation called XAIF is based on XML. We have developed running prototypes of these bridges in both directions, that run on a few examples and that need further development to allow each tool to take advantage of the other's analyses and models. This was supported by two visits of Krishna Narayanan to Inria and one of Laurent Hascoët to Argonne.

We also continued joint development of the Adjoinable-MPI library (AMPI) that provides efficient tangent and adjoint differentiation for MPI-parallel codes, independently of the AD tool used (now AdolC, dco, OpenAD, Tapenade).

We also extracted from Tapenade a standalone kernel (with documented API) for program parsing, analysis, and unparsing, which is not specific to AD and which could be used to develop other source-to-source code transformations. Paul Hovland's team and another Argonne team have shown interest for this library.

6.6. Algorithmic Differentiation and Dynamic Memory

Participants: Laurent Hascoët, Sri Hari Krishna Narayanan [Argonne National Lab. (Illinois, USA)].

In the same framework as in section 6.5, we made progress in the development of the adjoint AD model for programs that use dynamic memory. Adjoint differentiated code obtained by source transformation (OpenAD, Tapenade...) consists of a forward sweep that essentially copies the original code, and a backward sweep that computes the derivatives, These two sweeps must have the same control-flow shape, only reversed. The allocation and deallocation of dynamic memory inside the forward sweep requires a similar pattern in the backward sweep. However, allocations do not always return the same memory chunk, and therefore all memory addresses must be updated to preserve their consistency in the backward sweep.

This problem can only be solved at run-time. A compile-time analysis simply cannot extract the information needed. Our approach is thus to design a library that encapsulates all calls to memory allocation primitives (`malloc, free`...) in order to register the allocated addresses and to restore consistency of pointers during the backward sweep. This strategy is similar to the one we use for MPI calls, *cf* 6.5, and is actually an ingredient in our AMPI strategy.

This approach was tested with success on a medium-size industrial application in structural mechanics. For this unsteady simulation the C code allocates and frees memory repeatedly at each time step. The tangent and adjoint differentiated C codes, as produced by Tapenade, have been adapted by hand to run the new model, showing promising performance. Obviously, the next step is to update the Tapenade AD model to automate this hand adaptation.

6.7. Algorithmic Differentiation and Iterative Processes

Participants: Laurent Hascoët, Ala Taftaf.

Adjoint codes naturally propagate partial gradients backwards from the result of the simulation. However, this uses the data flow of the simulation in reverse order, at a cost that increases with the length of the simulation. In the special case of iterative Fixed-Point loops, it is clear that the first iterations operate on a meaningless "initial guess" state vector, and that reversing the corresponding data-flow is wasted effort. An adapted adjoint strategy for the iterative process should consider only the last or the few last iterations. Also the adjoint loop, which is itself a Fixed-Point iteration, must have its own stopping criterion and not merely run as many times as the direct Fixed-Point loop. We selected the strategy proposed by Bruce Christianson [17] and this year we implemented it in Tapenade. This strategy is triggered by differentiation directives that we defined. We tested this strategy with success on the medium-size testcase provided by Queen Mary University for the AboutFlow project.

Ala Taftaf presented her results at the WCCM congress during the Eccomas conference in Barcelona [13], july 21-25. Ala Taftaf did a two-months secondment for her Marie Curie PhD grant, with our partner team of Queen Mary University of London, during which she helped them take advantage of the latest developments in Tapenade and of her developments about Fixed-Point adjoints.

6.8. Multi-Activity specialized Differentiation

Participants: Laurent Hascoët, Ian Hueckelheim [Queen Mary University of London].

Up to this year, Tapenade did not allow for specialization of differentiated routines for different "activity" patterns. If a procedure must be differentiated once with respect to some of its arguments, and once with respect to another subset of arguments, then only one generalized differentiated procedure is created, with respect to the union of all subsets of active arguments. This incurs an efficiency penalty, but avoids a combinatorial explosion of the differentiated code.

However, there are cases where the efficiency penalty is high, and some users want more flexibility. Also the specialized adjoint for Fixed-Point iterations *cf* 6.7 uses two distinct activity patterns for the Fixed-Point loop body, and merging them looses some of the benefits of the approach. We have modified Tapenade to perform activity-specialized differentiation, giving the end-user a complete control through differentiation directives.

The experiments on a non-contrived industrial testcase of the AboutFlow project showed a non-negligible improvement between 5 to 10%. Work is still in progress to incorporate this new functionality into the mainstream distributed Tapenade. Ian Hueckelheim presented these results at the 16th EuroAD workshop in Jena, Germany, December 8-9.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

- Ecuador and Lemma share the results of Gautier Brèthes' thesis, which is partly supported by Lemma, the other part being supported by a PACA region fellowship.
- Ecuador and Lemma have a bilateral contract to share the results of Stephen Wornom.
- Ecuador and EDF have a bilateral contract on AD of the hydrology code "Mascaret". The correspondent on the Ecuador side is Valérie Pascual.

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 Gamma team, 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.

8.2. European Initiatives

8.2.1. FP7 & H2020 Projects

8.2.1.1. AboutFlow

Type: PEOPLE

Instrument: Initial Training Network

Duration: 2012-2016

Coordinator: Jens-Dominik Mueller

Partner: Queen Mary University of London (UK)

Inria contact: Laurent Hascoët

Abstract: The aim of AboutFlow is to develop robust gradient-based optimisation methods using adjoint sensitivities for numerical optimisation of flows. http://aboutflow.sems.qmul.ac.uk/

8.2.1.2. UMRIDA

Type:AAT

Instrument:Aeronautics and Air Transport

Duration: 2013-2016

Coordinator: Charles Hirsch

Partner: Numeca S.A. (Belgium)

Inria contact: Alain Dervieux

Abstract: UMRIDA addresses major research challenges in Uncertainty Quantification and Robust Design: develop new methods that handle large numbers of simultaneous uncertainties and generalized geometrical uncertainties. The turn-around time must be acceptable for industrial readiness. UMRIDA will apply these methods to representative industrial configurations.

8.3. International Initiatives

8.3.1. Inria International Labs

Ecuador participates in the Joint Laboratory for Petascale Computing (JLPC) together with our colleagues at Argonne National Laboratory. In 2014, Ecuador was local organizer of the 11th workshop of the JLPC in Sophia-Antipolis, june 9-11, and of the PUF summer school on HPC systems, june 12-13.

8.4. International Research Visitors

8.4.1. Visits of International Scientists

- Krishna Narayanan, from Argonne National Laboratory, visited Ecuador twice, on april 14-18 and on november 20-28
- Trond Steihaug, from University of Bergen (Norway), visited Ecuador from june 2 to june 27.
- Jan Hueckelheim, from Queen Mary University of London, did a secondment for the AboutFlow project with the Ecuador team from september 22 to november 21.

8.4.2. Visits to International Teams

- Laurent Hascoët visited Argonne National Laboratory from may 13 to may 23.
- Ala Taftaf did a secondment for the AboutFlow project with Queen Mary University of London from april 7 to june 6.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific events organisation

9.1.1.1. Member of the organizing committee

- Laurent Hascoët is on the organizing commitee of the EuroAD Workshops on Algorithmic Differentiation. The 15th EuroAD workshop was organized and hosted by the team in Sophia-Antipolis, june 16-17.
- Ecuador was local organizer of the 11th workshop of the Inria-Illinois JLPC in Sophia-Antipolis, june 9-11, and of the PUF summer school on HPC systems, June 12-13.

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 in progress : Gautier Brèthes, “Multigrilles anisotropes adaptatives”, started october 2012, advisor A. Dervieux

PhD in progress : Ala Taftaf, “Adjoint Automatic Differentiation on High-performance codes”, started july 2013, advisor L. Hascoët.

9.2.3. Juries

- Alain Dervieux, jury, PhD defense of Fernando Grossi, University of Toulouse, february 11.
- Alain Dervieux, jury, HDR defense of Elie Hachem, University of Nice, may 20.
- Laurent Hascoët, opponent, PhD defense of Mikko Auvinen, Aalto University, Finland, october 10.

9.3. Popularization

- Laurent Hascoët made a presentation on “Adjoint Automatic Differentiation for Data Assimilation” at the EGC 2014 conference in Rennes, January 28.
- Laurent Hascoët made a presentation on “Using Algorithmic Differentiation Tools to Compute Derivatives” at the ATOC Adjoint 2014 workshop organized by the British Antarctic Survey, Cambridge, July 1-2.

- Laurent Hascoët was invited to a seminar on adjoint methods http://www.dagstuhl.de/program/calendar/partlist/?semnr=14371&SUOG in Dagstuhl, Germany, september 7-12.

10. Bibliography

Major publications by the team in recent years

[1] F. COURTY, A. DERVIEUX. *Multilevel functional Preconditioning for shape optimisation*, in "International Journal of CFD", 2006, vol. 20, n° 7, pp. 481-490

[2] 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° 5, pp. 615-627

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

[4] 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

[5] L. HASCOËT, S. FIDANOVA, C. HELD. *Adjoining Independent Computations*, in "Automatic Differentiation of Algorithms: From Simulation to Optimization", New York, NY, G. CORLISS, C. FAURE, A. GRIEWANK, L. HASCOËT, U. NAUMANN (editors), Computer and Information Science, Springer, 2001, chap. 35, pp. 299-304

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

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

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

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

[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

Articles in International Peer-Reviewed Journals

[11] C. MOUSSAED, V. MARIA, S. WORNOM, B. KOOBUS, A. DERVIEUX. *Simulation of the flow past a circular cylinder in the supercritical regime by blending RANS and variational-multiscale LES models*, in "Journal

of Fluids and Structures", 2014, vol. 47, pp. 114 - 123 [*DOI :* 10.1016/J.JFLUIDSTRUCTS.2013.11.006], https://hal.inria.fr/hal-01083801

International Conferences with Proceedings

[12] G. BRETHES, A. DERVIEUX. *Main issues in Anisotropic Mesh Adaptive FMG*, in "World Congress of Computational Mechanics", Barcelone, Spain, July 2014, https://hal.inria.fr/hal-01110267

[13] A. TAFTAF, V. PASCUAL, L. HASCOËT. *Adjoints of fixed-point iterations*, in "11th World Congress on Computational Mechanics (WCCM XI)", Barcelona, Spain, July 2014, https://hal.inria.fr/hal-01079185

Scientific Books (or Scientific Book chapters)

[14] L. HASCOËT. *Adjoints by Automatic Differentiation*, in "Advanced Data Assimilation for Geosciences", Oxford University Press, 2014, https://hal.inria.fr/hal-01109881

References in notes

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

[16] 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

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

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

[19] J.-F. COLLARD. *Reasoning about program transformations*, Springer, 2002

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

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

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

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

[24] 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

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

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

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

[28] 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

[29] G. MADEC, P. DELECLUSE, M. IMBARD, C. LEVY. *OPA8.1 ocean general circulation model reference manual*, Pole de Modelisation, IPSL, 1998

[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*, université de Nice Sophia-Antipolis, 1993

[32] 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