Activity Report 2012

Team SCIPORT

Program transformations for scientific computing

RESEARCH CENTER
Sophia Antipolis - Méditerranée

THEME
Computational models and simulation

Table of contents

Team SCIPORT

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

Creation of the Team: January 01, 2012 .

1. Members

Research Scientists
Laurent Hascoët [Senior Researcher, Team leader, HdR]
Valérie Pascual [Junior Researcher]
Alain Dervieux [Senior Researcher, HdR]

Faculty Member
Bruno Koobus [Université Montpellier 2, HdR]

External Collaborators
Stephen Wornom [Lemma Company]
Trond Steihaug [Professor, University of Bergen, Norway, on sabbatical]

PhD Students
Hubert Alcin
Alexandre Carabias
Gautier Brethes [from October 2012]

Administrative Assistant
Claire Senica

2. Overall Objectives

2.1. Overall Objectives

Sciport is a temporary sequel to the former project Tropics. The team studies Automatic Differentiation (AD) of algorithms and programs. We work at the junction of two research domains :

- **AD theory:** We study software engineering techniques, to analyze and transform programs mechanically. Automatic 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 so-called *reverse* or *adjoint* mode of AD, a sophisticated transformation that yields gradients for optimization at a remarkably low cost.
- **AD application to Scientific Computing:** We apply the adjoint mode of AD to e.g. Computational Fluid Dynamics. We adapt the strategies of Scientific Computing to take full advantage of AD. We validate our work on real-size applications.

Each aspect of our work benefit to the other. 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 our tool TAPENADE, which is one of the most popular AD tools.

Our research directions are :

- Modern numerical methods for finite elements or finite differences : multigrid methods, mesh adaptation.
- Optimal shape design or optimal control in fluid dynamics for steady and unsteady simulations. Higher-order derivatives needed by robust optimization.
- Automatic Differentiation : AD-specific static data-flow analysis, strategies to reduce runtime and memory consumption of the adjoint mode for very large codes. Improved models for adjoint-mode AD, in particular coping with Message-Passing parallellism.

3. Scientific Foundations

3.1. Automatic Differentiation

Participants: Laurent Hascoët, Valérie Pascual.

automatic differentiation (AD) Automatic transformation of a program, that returns a new program that computes some derivatives of the given 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.

checkpointing General trade-off technique, used in adjoint-mode AD, that trades duplicate execution of a part of the program to save some memory space that was used to save intermediate results. Checkpointing a code fragment amounts to running this fragment without any storage of intermediate values, thus saving memory space. Later, when such an intermediate value is required, the fragment is run a second time to obtain the required values.

Automatic or Algorithmic Differentiation (AD) differentiates *programs*. An AD tool takes as input a source program P that, given a vector argument $X \in I\!R^n$, computes some vector result $Y = F(X) \in I\!R^m$. The AD tool generates a new source program P' that, given the argument X, computes some derivatives of F. 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

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

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 chain rule gives the Jacobian of F

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

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

In practice, the Jacobian $F'(X)$ is often too expensive to compute and store, but most applications only need projections of $F'(X)$ such as:

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

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

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

- **Adjoints**, defined for a given weighting $\overline{Y}$ of the outputs as:

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

Adjoints are 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 [33], adjoint problems [28], or inverse problems.

Adjoint-mode AD turns out to make a very efficient program, at least theoretically [30]. 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 regularly extend our models and tools to handle these new constructs.

3.2. Static Analysis and Transformation of programs

Participants: Laurent Hascoët, Valérie Pascual.

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

control flow graph Representation of a procedure body as a directed graph, whose nodes, known as basic blocks, contain each a list of instructions to be executed in sequence, and whose arcs 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 simultaneously, 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 whether a variable is initialized or not, at any location in the source program.

data dependence analysis Program analysis that studies the itinerary of values during program execution, from the place where a value is generated to the places where it is used, and finally to the place where it is overwritten. The collection of all these itineraries is often stored 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 conversely 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 [20] to improve the quality of the produced code. 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.
- 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 [23], [31], [21]. 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 are 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 [24], 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* [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. In addition to the very theoretical limit of *undecidability*, there are practical limitations to how much information one can infer from programs that use arrays [37], [26] or pointers. In general, conservative *over-approximations* are always made that lead to derivative code that is less efficient than possibly achievable.

3.3. Automatic 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 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 [28], 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 [27] 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 [32], [35] to reduce memory usage.

If for instance the model is steady, one tradeoff can use the iterated states in the direct order [29]. Another tradeoff can use only the fully converged final state. Since tradeoff approaches can be error-prone, we advocate incorporating them into the AD model and into the AD tools.

4. Application Domains

4.1. Automatic Differentiation

Automatic Differentiation of programs gives sensitivities or gradients, that are useful for many types of applications:

- 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. From 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 a promising 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* [33] 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 largely unknown. Only some measures at arbitrary places and times are available. A good initial state is found by solving a least squares problem between the measures 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 [36]. Figure 1 shows an example of a data assimilation exercise using the oceanography code OPA [34] and its AD adjoint produced by TAPENADE.

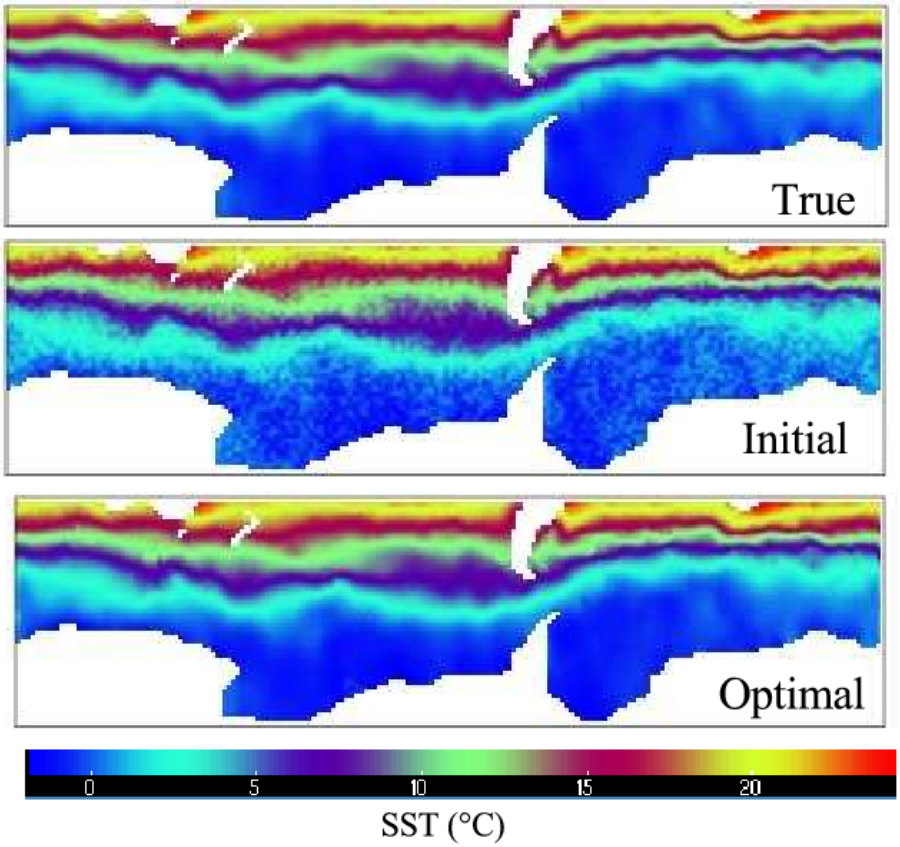

Figure 1. Twin experiment using the adjoint of OPA. We add random noise to a simulation of the ocean state around the Antarctic, and we remove this noise by minimizing the discrepancy with the physical model

The special case of *4Dvar* data assimilation is particularly challenging. The 4^{th} dimension in "4D" is time, as available measures 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 grows in complexity, 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 is sometimes 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 a very efficient approximate solution: 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. Software

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 with Mpi as parallel programming concept. The mesh model is unstructured tetrahedrization, with possible mesh motion. See also 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.

TAPENADE is an Automatic Differentiation tool that transforms an original program into a new program that computes derivatives of the original program. Automatic 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, and adjoint. Documentation is provided on the web site of the reserch team and as the Inria technical report RT-0300. TAPENADE runs under most operating systems and requires installation of Java jdk1.6 or upward. See also http://www-sop.inria.fr/tropics/

- Version: v3.6, r4343, February 2012
- 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: automatic 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. TAPENADE differentiates computer programs according to the model described in section 3.1 and in [19] Higher-order derivatives can be obtained through repeated application of tangent AD on tangent- and/or adjoint-mode AD.

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 analysis 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. Industrial or commercial usage require a paying license, as detailled on the team's web page. The software has been downloaded several hundred times, and the web tool served several thousands of true connections (not counting robots). The tapenade-users mailing list is over one hundred registered users.

6. New Results

6.1. Automatic Differentiation and parallel codes

Participants: Valérie Pascual, Laurent Hascoët, Hubert Alcin, Jean Utke [Argonne National Lab. (Illinois, USA)], Uwe Naumann [RWTH Aachen University (Germany)].

Together with colleagues in Argonne National Lab. and RWTH Aachen, we are studying how AD tools can handle MPI-parallel codes, especially in adjoint mode.

This year, we have presented our strategy [16] to extend Data-Flow analysis to Message-Passing communication. This strategy is specially designed for a program representation like that of TAPENADE, i.e. based on a Call-Graph whose nodes are indeed Flow-Graphs. This representation makes it easier to implement analyses in a way that is both context-sensitive and flow-sensitive. Our strategy also relies on the fixed-point implementation of the analyses, which uses a "wait-list".

At the same time, we continue the design of a adjoint-mode AD adapted to MPI communication. In our framework of AD by source transformation, we have pushed far in the direction of static data-flow analyses and static source transformation of individual MPI calls. We obtained results on classical cases of message-passing [38]. However, experience shows [11] that general usage of message-passing defies static analysis. A purely static analysis and transformation must resort too often to conservative choices, yielding a poor efficiency.

As a consequence, we are now going in the direction of a more dynamic, run-time treatment of adjoint MPI calls. This means designing a wrapper library "AMPI" on top of MPI, that takes care during execution of the adjoint code of the bookkeeping to send the adjoint messages in the reverse direction. This wrapper library should also be independent from the particular AD tool, as it will be used not only with TAPENADE but with the tools developed at Argonne and RWTH Aachen.

6.2. Finer control on AD transformation

Participants: Valérie Pascual, Laurent Hascoët.

We explore methods to provide the AD end-user with a better control on the AD transformation. We want to organize a progressive AD process in which the end-user can choose among a set of available AD code optimizations. In a first stage, the end-user may deactivate most of these optimizations, thus obtaining a differentiated code that is easier to understand and hopefully more robust. If problems do occur, this differentiated code is easier to debug with the debugging tools that we provide. In the next stages, the end-user may progressively turn the optimizations on, and at the same time check that the derivatives remain correct.

Another goal closely related is the comparison and evaluation of the existing corpus of AD code optimizations. TAPENADE is one of the AD tools that incorporate most of AD optimizations proposed in litterature. If a few missing optimizations are included, TAPENADE with its relatively large set of validation applications can be the common ground for a credible evaluation of the benefit brought by each optimization.

In this direction, we have extended TAPENADE to turn some classical optimizations that were automatically applied into optional optimizations. The emblematic example is activity analysis. This required some code cleanup. Also, we are extending TAPENADE to give the option of "association by address" instead of "association by name". This means bundling each variable with its derivative into a structured object, instead of creating new variables with new names to hold the derivatives. Which option is better is a difficult question, related to memory locality issues. This extension will allow us to make accurate measurements on our set of validation codes. This is also a step towards a better collaboration of TAPENADE with overloading-based AD tools, that natively use association by address.

6.3. Formal specification of AD

Participant: Laurent Hascoët.

There is very little formal specification of AD as a program transformation, and consequently no formal proof of its correctness. Correctness of tangent AD is problematic: if defined as equivalence of the tangent program semantics with the mathematical derivative of the semantics of the original code, correctness is mostly granted for simple staight-line programs, and in general not granted for programs with control. Therefore formal proofs of correctness appear unreachable at present. Fortunately, there is little concern about the practical relevance of tangent AD. The confidence of end-users regarding tangent AD is justified by everyday experience.

Adjoint-mode AD poses a different challenge. The adjoint AD transformation is by no means simple nor intuitive. Its specification is informal, so that end-users of AD cannot gain a strong confidence in the process. Moreover, the constant quest for efficiency of the adjoint code has introduced a number of improvements and tradeoffs that are defined informally. These improvements make the adjoint code intricate and sometimes interact to cause subtle bugs. On the other hand, the good news is that the difference between the adjoint code and the tangent code only lies in the order of the derivatives computations and not in their nature. A formal proof of semantic equivalence is thus conceivable.

The first step towards such a proof is a formal specification of both tangent-mode and adjoint-mode AD, including the specification of the program static data-flow analyses that the transformations require. We have provided this specification in terms of Data-Flow equations for the analyses, and in terms of Structural Operational Semantics (more precisely Natural Semantics) for the AD transformations themselves [19]. This specification will be the basis for future formal proofs of equivalence between tangent AD and adjoint AD.

6.4. Resolution of linearised systems

Participants: Hubert Alcin, Olivier Allain [Lemma], Marianna Braza [IMF-Toulouse], Alexandre Carabias, Alain Dervieux, Bruno Koobus [Université Montpellier 2], Carine Moussaed [Université Montpellier 2], Stephen Wornom [Lemma].

Increased sophistication of solution algorithms pose a challenge to Automatic Differentiation. Time-stepping iterations create numerous updates of the iterated solution vector. Other additional nonlinear iterative processes occur such as:

- the evaluation of an optimal step, which results at least from a homographic function of the unknown,
- the orthonormalisation of the updates (Gram-Schmidt method, Hessenberg method).

Adjoint-mode AD applied to these algorithms produces a "linearised iterative algorithm" which is transposed and therefore follows the original iterations in the reverse order, needing each of the iterated state solution vectors. One such extreme case is the simulation of unsteady phenomena with implicit numerical schemes: simulating high Reynolds turbulent flows by a Large Eddy Simulation (LES) and RANS-LES models requires hundreds of thousands time steps, each of them involving a modern iterative solution algorithm. This is the case targetted by the 4-year ANR project "ECINADS", jointly with university of Montpellier 2, the Institut de Mécanique des fluides de Toulouse and Lemma company, started in 2009.

In ECINADS, we design more efficient solution algorithms and we examine the questions risen by their adjoint differentiation. Our goal is practical scalability of the direct simulation and of its adjoint on a large number of processors. ECINADS also addresses the scalable solution of new approximations.

In 2012, the novel three-level method studied by H. Alcin on a model problem has been extended to compressible viscous flows by B. Koobus and C. Moussaed from university of Montpellier.

Hubert Alcin, Bruno Koobus, Olivier Allain and Alain Dervieux published their work on a two-level Schwarz algorithm in IJNMFD [12]. H. Alcin has presented his work in the Parallel CFD conference of Altlanta [14]. H. Alcin wrote his thesis [11], defended in december, on the three main subjects of ECINADS: the two- and three-level Schwarz algorithms, Automatic Differentiation and mesh adaptation.

6.5. Automatic Differentiation of a CFD code

Participants: Hubert Alcin, Valérie Pascual, Laurent Hascoët, Alain Dervieux.

The ECINADS workplan includes the building of an adjoint state for a CFD kernel. We have chosen AIRONUM 5.1, a real life kernel that combines two particular features:

- it uses intensively the Fortran95 dynamic memory allocation
- it uses MPI parallelization.

This work is reported in H. Alcin's PhD [11].

6.6. Perturbation Methods

Participants: Alain Dervieux, Laurent Hascoët.

In the context of the European project NODESIM-CFD (ended 2010), the contribution of Sciport involved mainly the derivation of perturbation methods and reduced order models for the management of uncertainties. These methods rely on Taylor series with second-order terms. The production of second derivative code is obtained through repeated application of Automatic Differentiation. Three strategies can be applied to obtain (elements of) the Hessian matrix, named Tangent-on-Tangent, Tangent-on-Adjoint, and Adjoint-on-Tangent. These new methods are promoted through short courses, e.g. by Alain Dervieux at an ERCOFTAC session (Chatou, 15-16 mai 2012). The application and extension of these methods are part of a FP7 proposal (Proposal UMRIDA, nov. 2012).

6.7. Control of approximation errors

Participants: Frédéric Alauzet [GAMMA team, Inria-Rocquencourt], Estelle Mbinky [GAMMA team, Inria-Rocquencourt], Olivier Allain [Lemma], Alexandre Carabias, Hubert Alcin, Alain Dervieux.

This is a joint research between Inria teams Gamma (Rocquencourt), Sciport, Castor and the Lemma company. Gamma brings mesh and approximation expertise, Sciport brings adjoint methods, and CFD applications are developed by CASTOR and Lemma.

The resolution of the optimum problem using adjoint-mode AD can be used in a slightly different context than optimal shape design, namely mesh adaptation. This will be possible if we can map the mesh adaptation problem into a differentiable optimal control problem. To this end, we express the mesh adaptation problem in a purely functional form: the mesh is reduced to a continuous property of the computational domain named the continuous metric. We minimize a continuous model of the error resulting from that metric. Thus the search of an adapted mesh is transformed into the search of an optimal metric.

In 2012, this activity is amplifying. A work on goal-oriented mesh adaptation for unsteady Euler flows submitted to the journal JCP has been accepted and published [13]. Its extension to the compressible Navier-Stokes model has been developed in 2D [22] and in 3D [11]. A further extension to Large Eddy Simulation has been defined and developed in the WOLF demonstrator. A communication at ECCOMAS (Vienna) has been presented and papers are being written for publication in journal.

The method is being extended to a third-order approximation, the Vertex-CENO. This approximation was defined collaboratively between university of Montpellier, IMM-Moscow and Sciport. A more accurate version is studied by Alexandre Carabias. A new mesh adaptation theory involving error estimates and criteria has been developed by Gamma and Sciport. The extension of the multiscale adaptation method is considered by Estelle Mbinky at Rocquencourt and has been presented at ECCOMAS (Vienna). The extension of the goal-oriented method is considered by Alexandre Carabias and first results were presented at ECCOMAS (Vienna). A cooperation with CEMEF and university of Nice is considered and a ERC common proposal, CMILE, has been built. Anisotropic mesh adaptation allows for better convergence towards continuous solutions, and in particular more accurate a posteriori error estimates and correctors. The synergy between correctors and mesh adaptation is the subject of a joint contribution (Gamma and Sciport) for the FP7 UMRIDA proposal (nov. 2012).

7. Partnerships and Cooperations

7.1. National Initiatives

7.1.1. ANR

7.1.1.1. ECINADS

Sciport is coordinator of the ANR project ECINADS, with CASTOR team, university Montpellier 2, Institut de Mécanique des Fluides de Toulouse and the Lemma company in Sophia-Antipolis. ECINADS concentrates on scalable parallel solution algorithms for state and adjoint systems in CFD, and on the use of this adjoint for mesh adaptation applied to unsteady turbulent flows.

7.2. European Initiatives

7.2.1. FP7 Projects

Program: FP7-PEOPLE-2012-ITN

Project acronym: About Flow

Project title: Adjoint-based optimisation of industrial and unsteady flows

Duration: Nov 2012 - Oct 2016

Coordinator: J.-D. Mueller, Queen Mary University of London

Other partners: Engys (UK), ESI-Group (F), Inria (F), National Technical University of Athens (GR), Rolls Royce (D), RWTH Aachen University (D), Volkswagen AG (D), Warsaw University of Technology (PL).

Abstract: Adjoint-based methods have become the most interesting approach in CFD optimisation due to their low computational cost compared to other approaches. The development of adjoint solvers has seen significant research interest, and a number of EC projects have been funded on adjoint-based optimisation. In particular, partners of this proposal are members of the EC FP7 project FlowHead which develops complete adjoint-based design methods for steady-state flows in automotive design. Integration of the currently available shape and topology modification approaches with the gradient-based optimisation approach will be addressed, in particular development of interfaces to return the optimised shape into CAD for further design and analysis, an aspect that currently requires manual interpretation by an expert user. In industrial practice most industrial flows have small levels of instability, which leads to a lack of robustness and instability of the adjoint, such as trailing edge vortex shedding in turbo-machinery. Many industrial applications are also partly unsteady such as bluff body separation in cars or fully unsteady such as vertical-axis wind turbines. In unsteady adjoints 'checkpoints' of the flow solution at previous timesteps need to be recorded and algorithms for an effective balance between storage and recomputation need to be implemented. The recomputation involves significant memory and runtime overheads for which efficient methods are developed and implemented. The results of the project will be applied to realistic mid-size and large-scale industrial optimisation problems supplied by the industrial project partners ranging from turbo-machinery, to automotive to wind-turbines. Training will be provided by academic, industrial and SME partners in methods development, industrial application and software management. A large programme of complementary training in professional skills will be provided with support from all partners.

7.3. International Initiatives

7.3.1. Inria International Partners

The team's collaboration with the Mathematics and Computer Science (MCS) division of Argonne National Laboratory is recognised by Inria as an "Inria International Partnership". This partnership is named "SARDINE" for "Sophia-Antipolis ARgonne DIfferentiation INitiativE".

7.3.2. Participation In International Programs

The team participates in the Joint Laboratory for Petascale Computing (Inria, University of Illinois at Urbana Champaign, Argonne National Laboratory). Laurent Hascoët gave talks at this year's meetings in Rennes and Argonne.

7.4. International Research Visitors

7.4.1. Visits of International Scientists

Trond Steihaug, professor at the University of Bergen, has spent a sabbatical period with the team, from september 2011 to may 2012. He worked on AD of Factorable and of Separable functions [15].

8. Dissemination

8.1. Scientific Animation

- Laurent Hascoët was on the program committee of the AD2012 conference, Fort Collins, Colorado, july 2012.
- Laurent Hascoët is on the organizing commitee of the EuroAD Workshops on Automatic Differentiation. No workshop was organized this year to avoid conflict with the AD2012 conference.
- Laurent Hascoët is a member of the internal "CDT" committee at Inria Sophia-Antipolis ("Comité Développement Technologique").

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Licence : Hubert Alcin, monitorat Analyse Numerique, 30 h, niveau L3, Université de Nice, France

Master : Laurent Hascoët, Optimisation avancée, 15 h, niveau M2, Université de Nice, France

8.2.2. Supervision

PhD : Hubert Alcin, "Résolution d'écoulements instationnaires et adjoints", Université de Nice, defended december 5th, 2012, advisors A. Dervieux and L. Hascoët

PhD in progress : Alexandre Carabias, "Adaptation de maillage pour calculs d'écoulements à l'ordre 3", started october 2011, advisor A. Dervieux

PhD in progress : Gauthier Brethes, "Multigrilles anisotropes adaptatives", started october 2012, advisor A. Dervieux

8.2.3. Juries

- Alain Dervieux was jury for the thesis of Olivier Rouch, Montrèal, and of Hubert Alcin, Nice.

8.3. Conferences and workshops

- Laurent Hascoët was invited to present Automatic Differentiation at NATIXIS, Paris, april 2012.
- Alexandre Carabias presented a seminar "Controle de la dispersion et de la dissipation pour un schema a reconstruction quadratique", during an ECINADS meeting, Inria Sophia Antipolis, april 2012.
- Alain Dervieux was invited to present "Goal-oriented anisotropic mesh adaptation based on a priori estimates" (F. Alauzet, A. Belme, A. Loseille, D. Guégan, A. Dervieux), at the Workshop on Adaptive Methods with Applications in Fluid Dynamics (ADAP-CFD12) (WIAS), Berlin, april 2012.
- Laurent Hascoët was invited to present advances in Automatic Differentiation at CERFACS, Toulouse, may 2012.
- Hubert Alcin presented "On 2-level Schwarz algorithms for LES compressible flows" (H. Alcin, O. Allain, B. Koobus and A. Dervieux), at ParCFD2012, Atlanta, may 2012.
- Alexandre Carabias presented a seminar "Schema QV6 et adaptation", with team Gamma3, Inria Rocquencourt, may 2012.
- Alexandre Carabias presented a seminar "Controle de la dispersion et de la dissipation pour un schema a reconstruction quadratique" at "Colloque des doctorants de 2eme année", Université de Nice, may 2012.

- Alain Dervieux gave a short course “Sensitivity analysis by adjoint Automatic Differentiation and Application” (A. Belme, M. Martinelli, L. Hascoët, V. Pascual, A. Dervieux), at the ERCOFTAC Course on Uncertainty Management and Quantification in Industrial Analysis and Design, EDF , PARIS, may 2012.
- Laurent Hascoët gave an introductory course on Automatic Differentiation [17] during the “Advanced data assimilation for geosciences” course (org. E. Blayo, M. Bocquet, and E. Cosme), Les Houches school of physics, june 2012.
- Laurent Hascoët presented at the two meetings of the JLPC Inria-Illinois joint laboratory: “Toward Adjoinable MPI”, Rennes, june 2012, and “The Data-Dependence graph of Adjoint Codes”, Argonne Nat. Lab., november 2012.
- Laurent Hascoët presented [18] “Using Automatic Differentiation to study the sensitivity of a crop model” at AD2012 conference, Fort Collins, Colorado, july 2012.
- Valérie Pascual presented [16] “Native handling of Message-Passing communication in Data-Flow analysis” at AD2012 conference, Fort Collins, Colorado, july 2012.
- Laurent Hascoët presented an overview of Automatic Differentiation at “Ecole d’été UFA” (org. A. Desideri), Sophia-Antipolis, september 2012.
- Alain Dervieux presented “Goal-Oriented mesh adaptation for vortex shedding flows” (H. Alcin, A. Belme, A. Loseille, F. Alauzet, S. Wornom , A. Dervieux), at ECCOMAS, Vienna, Austria, september 2012.
- Carine Moussaed presented “A Dynamic VMS-LES model and its Hybrid extension for bluff body flows” (C. Moussaed, S. Wornom, B. Koobus, M.-V. Salvetti, A. Dervieux), at ECCOMAS, Vienna, Austria, september 2012.
- Alexandre Carabias presented “Anisotropic Goal-oriented estimate for a third-order accurate Euler model” (A. Carabias, A. Belme, F. Alauzet, A. Dervieux, A. Loseille), at ECCOMAS, Vienna, Austria, september 2012.
- Hubert Alcin has presented “Goal-Oriented mesh adaptation and 2-level Schwarz Algorithms”, at CERFACS seminar, Toulouse, october 2012.
- Laurent Hascoët, together with Andreas Griewank, served as evaluator for B. Pearlmutter team in NUI Maynooth, Ireland, november 2012.
- Laurent Hascoët spent two weeks at Argonne Nat. Lab. to work on an adjoinable MPI library with Jean Utke, november 2012.
- Alain Dervieux gave a short course “Indicateurs de raffinement et adaptation de maillage en simulation numérique pour la Mécanique des Fluides” (A. Dervieux, F. Alauzet), during the “Cours sur la Vérification des simulations numériques en Mécanique des Milieux Continus”, Collège X, Paris, november 2012.

9. 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^{o} 7, p. 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^{o} 5, p. 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] A. GRIEWANK. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Frontiers in Applied Mathematics, 2000.

[5] 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.

[6] 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, New York, NY, 2001, chap. 35, p. 299-304.

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

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

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

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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] H. ALCIN. *Résolution d'écoulements instationnaires et adjoints*, Université de Nice Sophia-Antipolis, 2012.

Articles in International Peer-Reviewed Journals

[12] H. ALCIN, B. KOOBUS, O. ALLAIN, A. DERVIEUX. *Efficiency and scalability of a two-level Schwarz algorithm for incompressible and compressible flows*, in "International Journal for Numerical Methods in Fluids", 2012, http://dx.doi.org/10.1002/fld.3733.

[13] A. BELME, A. DERVIEUX, F. ALAUZET. *Time Accurate Anisotropic Goal-Oriented Mesh Adaptation for Unsteady Flows*, in "J. Comp. Phys.", 2012, vol. 231, n^{o} 19, p. 6323-6348.

International Conferences with Proceedings

[14] H. ALCIN, B. KOOBUS, O. ALLAIN, A. DERVIEUX. *On 2-level Schwarz Algorithms for LES compressible flows*, in "Proceedings of Par-CFD", 2012.

[15] L. HASCOËT, S. HOSSAIN, T. STEIHAUG. *Structure in Optimization: Factorable Programming and Functions*, in "Proceedings of the 27th International Symposium on Computer and Information Sciences, ISCIS 2012", Springer, 2012, http://dx.doi.org/10.1007/978-1-4471-4594-3_46.

[16] V. PASCUAL, L. HASCOËT. *Native handling of Message-Passing communication in Data-Flow analysis*, in "Recent Advances in Algorithmic Differentiation", Lecture Notes in Computational Science and Engineering, Springer, 2012, p. 83-92, Selected papers from AD2012 Fort Collins, july 2012.

Scientific Books (or Scientific Book chapters)

[17] L. HASCOËT. *Adjoints by Automatic Differentiation*, in "Advanced data assimilation for geosciences", E. BLAYO, M. BOCQUET, E. COSME (editors), Oxford University Press, 2012, to appear.

[18] C. LAUVERNET, L. HASCOËT, F.-X. LE DIMET, F. BARRET. *Using Automatic Differentiation to study the sensitivity of a crop model*, in "Recent Advances in Algorithmic Differentiation", Lecture Notes in Computational Science and Engineering, Springer, 2012, p. 59-70, Selected papers from AD2012 Fort Collins, july 2012.

Research Reports

[19] L. HASCOËT, V. PASCUAL. *The Tapenade Automatic Differentiation tool: principles, model, and specification*, Inria, May 2012, n^{o} RR-7957, 53, http://hal.inria.fr/hal-00695839.

References in notes

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

[21] 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.

[22] A. BELME. *Unsteady Aerodynamics and Adjoint method*, Université de Nice Sophia-Antipolis, 2011.

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

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

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

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

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

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

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

[30] A. GRIEWANK, 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*, Université de Nice Sophia-Antipolis, 1987.

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

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

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

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

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

[37] 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.

[38] J. UTKE, L. HASCOËT, P. HEIMBACH, C. HILL, P. HOVLAND, U. NAUMANN. *Toward Adjoinable MPI*, in "Proceedings of the 10th IEEE International Workshop on Parallel and Distributed Scientific and Engineering, PDSEC'09", 2009.