Activity Report 2013

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: 2012 January 01, end of the Team: 2013 December 31.

1. Members

Research Scientists

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

PhD Students

Gautier Brethes [Inria, LEMMA]
Alexandre Carabias [Inria]
Ala Taftaf [Inria, from Jul 2013]

Visiting Scientists

Bruno Koobus [Univ. Montpellier II]
Stephen Wornom [LEMMA]

Administrative Assistant

Claire Senica [Inria]

2. Overall Objectives

2.1. Overall Objectives

Sciport is a temporary sequel to the former project Tropics. Sciport will stop at the end of 2013, when the new project Ecuador starts.

The team studies Automatic Differentiation (AD) of algorithms and programs, thus 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.

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 for robust optimization and uncertainty quantification.
- 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. Research Program

3.1. Automatic Differentiation

Participants: Laurent Hascoet, Valérie Pascual, Ala Taftaf.

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.

Automatic or Algorithmic Differentiation (AD) differentiates *programs*. An AD tool takes as input a source program P that, given a vector argument $X \in \mathbb{R}^n$, computes some vector result $Y = F(X) \in \mathbb{R}^m$. 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

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

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 is therefore a piecewise differentiation, which 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:

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

  Sensitivities are 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:

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

  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 [31], adjoint problems [25], or inverse problems.

Adjoint-mode AD turns out to make a very efficient program, at least theoretically [28]. 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 extending our models and tools to handle new constructs.

3.2. Static Analysis and Transformation of programs

Participants: Laurent Hascoet, 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, 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 [17] 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 [20], [29], [18]. 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 [21], 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* [22] 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 [35], [23] 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 Hascoet, 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 [25], 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 [24] 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 [30], [33] to reduce memory usage.

If for instance the model is steady, one tradeoff can use the iterated states in the direct order [26]. 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* [31] 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 [34]. Figure 1 shows an example of a data assimilation exercise using the oceanography code OPA [32] and its AD adjoint produced by TAPENADE.

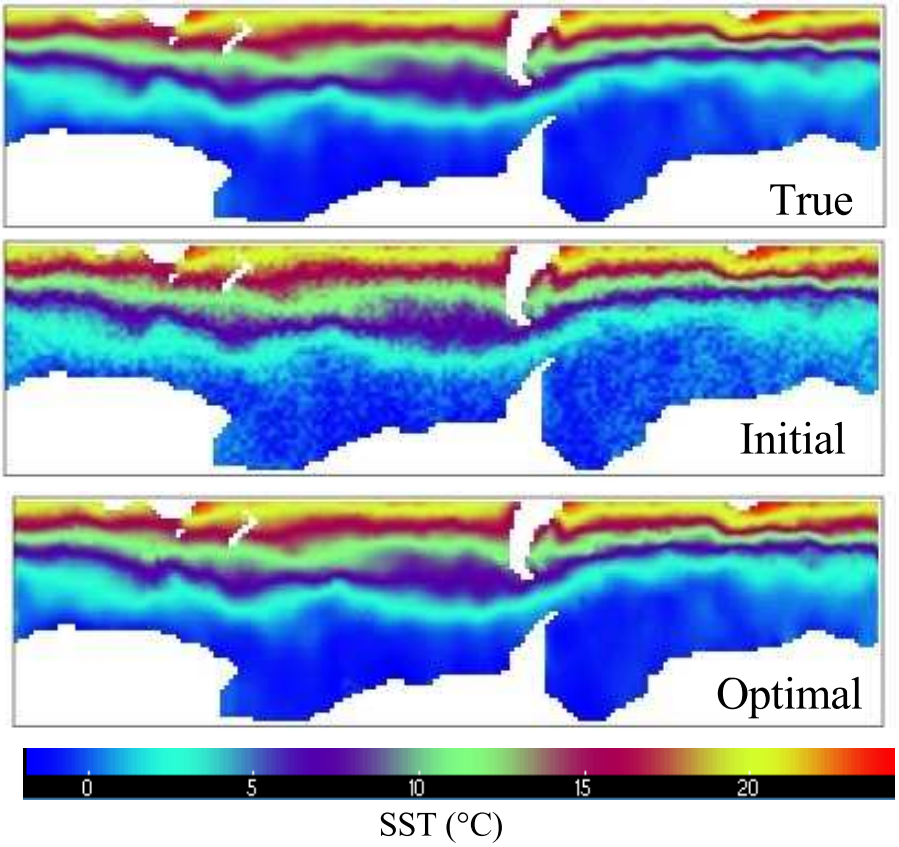

Figure 1. Twin experiment using the adjoint of OPA. We add random noise to a simulation of the ocean state around the Antarctic, and we remove this noise by minimizing the discrepancy with the physical model

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 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 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 Hascoet [correspondant], Valérie Pascual, Ala Taftaf.

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, adjoint, ans vector adjoint. Documentation is provided on the web site of the reserch team http://www-sop.inria.fr/tropics/, in Inria technical report RT-0300, and in [13]. TAPENADE runs under most operating systems and requires installation of Java jdk1.6 or upward.

- Version: v3.8, r4996, November 2013
- 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 [13] 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 Hascoet, Jean Utke [Argonne National Lab. (Illinois, USA)], Michel Schanen [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. Results are progressively incorporated into a library (AMPI, for Adjoinable-MPI) that is designed to provide efficient tangent and adjoint differentiation for MPI-parallel codes, independently of the AD tool used (AdolC, dco, OpenAD, TAPENADE ...). Primitives from the AMPI library dynamically orchestrate, at run-time, the MPI calls that are needed to compute the derivatives.

This year we studied issues raised by the collective reduction operations of MPI, and by the one-sided communications (i.e. remote memory access) offered by MPI-II.

The participants met on two occasions, two weeks in march in Sophia-Antipolis, and two weeks in october in Argonne.

This work was presented in particular at the meeting of the Inria-Illinois joint lab in june in Lyon. An article is in preparation.

6.2. Automatic Differentiation and Dynamic Memory

Participants: Valérie Pascual, Laurent Hascoet, Jean Utke [Argonne National Lab. (Illinois, USA)].

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 some 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 dynamically, at run-time. A compile-time analysis would have to be conservative, implying many overapproximations and in the end an unreasonably inefficient adjoint code. 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 be able to restore consistency of pointers during the backward sweep. This strategy is similar to the one we use for MPI calls, *cf* 6.1, and is actually needed in our AMPI strategy. All we can hope from a static analysis is to detect the simple cases where addresses could be recomputed instead of stored and updated. This may apply to a significant portion of memory manipulalions, and may thus reduce the overhead due to the dynamic updating.

We started developing this library, called ADMM for Adjoinable Dynamic Memory Management. TAPENADE will eventually produce adjoint code that calls these primitives instead of the standard memory management primitives.

6.3. Automatic Differentiation and iterative processes

Participants: Laurent Hascoet, 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. AD research looks for strategies to reduce this cost, taking advantage of the structures of the given program. One such structure is iterative fixed point loops, commonplace in numerical computation. They occur at the topmost level of steady-state simulations, as well as in unsteady simulations. They may also occur deeper in the simulation, for instance in linear solvers.

It clear that the first iterations of a fixed-point search operate on a meaningless 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. Furthermore, there is a discrete component to an iterative algorithm, namely the number of iterations, and this makes differentiability questionnable. For these reasons we are looking for a specific strategy for the adjoint, that reverses only the necessary data-flow, and that restores confidence in the validity of the derivative.

We seek inspiration in the strategies proposed by two authors [19], [27] to design one strategy that is amenable to implementation in a source-transformation AD tool such as TAPENADE. This will be triggered by user-given differentiation directives. We are also selecting example codes (a steady-state flow solver and a Newton solver) to benchmark and experiment.

Ala Taftaf presented her preliminary results at Queen Mary University in september, and at the 13th EuroAD workshop in Oxford, december 9-10. She attended a training on the CFD code OpenFOAM at Queen Mary, september 3-6.

6.4. Differentiation of third-party codes

Participants: Valérie Pascual, Laurent Hascoet, Alain Dervieux.

This year, we have differentiated two applications brought to us by academic colleagues. This is an important activity as it points us to problems that should be solved or interfaces that should be improved in TAPENADE.

- Striation simulates ionospheric plasma. It was developed in the University of Lille, then Toulouse. The Fortran90 source is relatively compact (10,000 lines). We obtained and validated the tangent and adjoint derivatives that were needed to solve an inverse problem i.e., identify the initial condition that causes an observed instability in plasma density. This work uncovered important AD issues when dynamic memory is used intensively, *cf* 6.2.
- Mascaret is a hydrodynamic simulation code developed and used by EDF to study river flows. Mascaret consists of 120,000 lines of Fortran90. In this first experiment, we differentiated only one of the three kernel solvers. We obtained validated tangent and adjoint derivatives. A further collaboration with EDF and CERFACS is planned next year.

In addition, Automatic Differentiation of the CFD code AIRONUM (*cf* 5.1) will continue in cooperation with the partners of the FP7 project UMRIDA.

6.5. Resolution of linearised systems

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

The work of Hubert Alcin for the ANR ECINADS on scalable parallel solvers based on coarse grids has been continued by Carine Moussaed and Bruno Koobus. This results in scalable computations up to 2048 processors.

Bruno Koobus and Carine Moussaed presented their results on “Un modèle VMS-LES dynamique pour la simulation d’écoulements autour d’obstacles” at CANUM congress, Super-Besse, France, May 21-25.

6.6. Control of approximation errors

Participants: Alexandre Carabias, Gautier Brethes, Alain Dervieux, Adrien Loseille [Gamma3 team, Inria-Rocquencourt], Frédéric Alauzet [Gamma3 team, Inria-Rocquencourt], Estelle Mbinky [Gamma3 team, Inria-Rocquencourt], Stephen Wornom [Lemma], Olivier Allain [Lemma], Anca Belme [university of Paris 6].

Third-order mesh adaptation was the main topic of the year in error control. Two PhD have been completed this year on third-order mesh adaptation:

- In team Gamma3, Estelle Mbinky has studied a method from Bernard Mourrain for transforming trilinear Taylor terms of the approximation error into a power of a bilinear term. Estelle Mbinky defended her thesis at Paris 6 on december 20.
- In our team, Alexandre Carabias (who spent most of this year with team Gamma3) has developed a 2D third-order scheme for the Euler model. The scheme is based on the ENO finite-volume formulation with quadratic reconstruction. Some effort was devoted to improve the performance of the scheme. The scheme is much less dissipative than an usual quadratic ENO scheme and of smaller cost. Implementation of a 3D version in AIRONUM (*cf* 5.1) is now starting. The 2D scheme has been the basis of an investigation of third order anisotropic mesh adaptation. Alexandre Carabias defended his thesis in Sophia-Antipolis on december 12.

A. Carabias and E. Mbinky presented their work on “A priori-based mesh adaptation for third-order accurate Euler simulation” at HONOM 2013, Bordeaux, France, March 18-22. We further studied mesh adaptation for viscous flows and we are preparing a journal article in collaboration with Gamma3 and University of Paris 6.

This year's new topic is the combination of Multi-Grid and anisotropic mesh adaption, with the starting PhD of Gautier Brèthes. The study involves several problematics, and in particular stopping criteria and construction of correctors. This was supported by the ANR project ECINADS, ended in november, but continues with the ANR project MAIDESC (started in october, coordinated by our team) following on mesh adaption and in particular meshes for interfaces, third-order accuracy, meshes for boundary layers, and curved meshes.

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. ECINADS ended in november.

7.1.1.2. MAIDESC

Sciport is coordinator of the ANR project MAIDESC, with Gamma team, university Montpellier 2, Cemef-Ecole des Mines, Inria-Bordeaux, Lemma and Transvalor. MAIDESC started in october. MAIDESC concentrates on mesh adaption and in particular meshes for interfaces, third-order accuracy, meshes for boundary layers, and curved meshes.

7.2. European Initiatives

7.2.1. FP7 Projects

7.2.1.1. AboutFlow

Type: PEOPLE

Instrument: Initial Training Network

Duration: November 2012 - October 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/

7.2.1.2. 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.

7.3. International Initiatives

7.3.1. Inria International Partners

7.3.1.1. SARDINE

Program: Inria International Partner

Title: Sophia-Antipolis ARgonne DIfferentiation INitiativE

Inria principal investigator: Laurent Hascoët

International Partner (Institution - Laboratory - Researcher):

Argonne National Laboratory (USA) - Math and Computer Science - Paul Hovland

Duration: 2012 - 2013

We study theoretical and computer science aspects of Automatic Differentiation (AD) by source transformation. In the context of the adjoint mode of AD, which computes gradients, we focus on the storage-recomputation tradeoffs that are the key to efficiency. We also focus on the correct AD of message-passing communication calls that are found in parallel application. A third goal is the use into Uncertainty Quantification of higher-order derivatives produced through AD. From the point of view of tool development, we aim at building interfaces to bridge between the AD tools of our teams, OpenAD and TAPENADE.

7.3.2. Inria International Labs

The team participates in the JLPC, together with our colleagues at Argonne National Laboratory. Laurent Hascoët attended the JLPC meeting in Lyon on june 12-14, and presented our works on the adjoint of MPI-II one-sided communications. The team co-organizes and will host the next JLPC meeting in june 2014 in Sophia-Antipolis.

7.4. International Research Visitors

7.4.1. Visits of International Scientists

- Jean Utke, Argonne National Laboratory (USA), march 11 to march 22.
- Michel Schanen, RWTH Aachen (Germany), march 11 to march 15.
- Trond Steihaug, from University of Bergen (Norway), june 3 to june 28.

7.4.2. Visits to International Teams

- Laurent Hascoët invited by Argonne National Laboratory (USA) from october 14th to october 25th.

8. Dissemination

8.1. Scientific Animation

- Laurent Hascoët is on the organizing commitee of the EuroAD Workshops on Automatic Differentiation. The 12th EuroAD workshop was organized and hosted by the team in Sophia-Antipolis, june 10-11.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

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

8.2.2. Supervision

PhD : Alexandre Carabias, “Adaptation de maillage pour calculs d’écoulements à l’ordre 3”, Université de Nice, defended december 12th, advisor A. Dervieux

PhD : Cédric Lachat, “Conception et validation d’algorithmes de remaillage parallèles à mémoire distribu’ee basés sur un remailleur séquentiel”, Université de Nice, defended december 13th, co-advisor L. Hascoët.

PhD in progress : Gauthier Brethes, “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.

8.3. Popularization

- Laurent Hascoët gave a presentation on adjoint Automatic Differentiation at the AboutFlow meeting in Tinos (Greece), may 27-31.
- Laurent Hascoët made a detailed presentation of Automatic Differentiation at Microsoft Research labs in Cambridge (UK) on december 6th.

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, 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^{o} 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] 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, 2001, chap. 35, pp. 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, pp. 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, pp. 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, pp. 2101-2124

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] A. CARABIAS. , *Analyse et adaptation de maillage pour des schemas non-oscillatoires d'ordre eleve*, Université Nice Sophia Antipolis, December 2013, http://hal.inria.fr/tel-00914214

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", 2013, vol. 72, n^{o} 1, pp. 69-89 [*DOI :* 10.1002/FLD.3733], http://hal.inria.fr/hal-00914000

[13] L. HASCOËT, V. PASCUAL. *The Tapenade Automatic Differentiation tool: principles, model, and specification*, in "ACM Transactions on Mathematical Software", 2013, vol. 39, n^{o} 3 [*DOI :* 10.1145/2450153.2450158], http://hal.inria.fr/hal-00913983

[14] C. MOUSSAED, S. WORNOM, B. KOOBUS, M.-V. SALVETTI, A. DERVIEUX. *Dynamic variational multi-scale LES of bluff body flows on unstructured grids*, in "World Academy of Science, Engineering and Technology", 2013, n^{o} 77, pp. 595-602, http://hal.inria.fr/hal-00914199

International Conferences with Proceedings

[15] C. MOUSSAED, S. WORNOM, B. KOOBUS, A. DERVIEUX, T. DELOZE, Y. HOARAU, M. ELHIMER, M. BRAZA. *VMS- and OES-based hybrid simulations of bluff body flows*, in "ERCOFTAC Symposium on Unsteady Separation in Fluid-Structure Interaction", Mykonos, Greece, 2013, http://hal.inria.fr/hal-00914023

Conferences without Proceedings

[16] G. BRÈTHES, A. DERVIEUX. *Adaptive Full-Multigrid algorithms based on Riemannian metrics*, in "2nd ECCOMAS Young Investigators Conference (YIC 2013)", Bordeaux, France, September 2013, http://hal.inria.fr/hal-00855886

References in notes

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

[18] I. ATTALI, V. PASCUAL, C. ROUDET. , *A language and an integrated environment for program transformations*, Inria, 1997, n^{o} 3313, http://hal.inria.fr/inria-00073376

[19] B. CHRISTIANSON. *Reverse accumulation and attractive fixed points*, in "Optimization Methods and Software", 1994, vol. 3, pp. 311-326

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

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

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

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

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

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

[26] 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

[27] A. GRIEWANK, C. FAURE. *Piggyback differentiation and optimization*, in "Large-scale PDE-constrained optimization", Springer, LNCSE #30, 2003, pp. 148–164

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

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

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

[31] 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

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

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

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

[35] 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