AD for optimization in electromagnetism applied to semi analytical models combining composed functions

Petre Enciu, Frédéric Wurtz, Laurent Gerbaud, Benoît Delinchant

AD for Optimization in Electromagnetism Applied to Semi-Analytical Models Combining Composed Functions

P. Enciu[*], F. Wurtz[*], L. Gerbaud[*], B. Delinchant[*]

(*) Laboratoire de Génie Electrique de Grenoble (G2ELab)
UMR5269, CNRS/UJF/INPG
ENSE^3, Domaine Universitaire, BP 46,
F-38402 Saint Martin d'Hères Cedex, FRANCE
E-mail: Petre.ENCIU@g2elab.grenoble-inp.fr

Purpose:

This paper deals with Automatic Differentiation, for the device sizing in electromagnetism by using gradient constrained optimization. CADES framework (Component Architecture for the Design of Engineering Systems), previously described, is presented here with extended features.

Design/Methodology/Approach

The paper is subject to further usage for optimization of Automatic Differentiation (also named Algorithmic Differentiation) which is a powerful technique that computes derivatives of functions described as computer programs in a programming language like C/C++, FORTRAN.

Findings

Indeed, analytical modeling is well suited regarding optimization procedure, but the modeling of complex devices needs sometimes numerical formulations. This paper then reviews the "External Functions" and the so called "External Function of Function" modeling concepts implemented in CADES which aim to manage the interactions of analytical and numerical modeling inside of gradient based optimization procedure. Finally, the paper shows that Automatic Differentiation has no limit for the input program complexity, or gradients accuracy, in the context of constrained optimization of an electromagnetic actuator.

Originality/value

Automatic Differentiated is employed for a large and complex numerical code computing multidimensional integrals of functions. Thus, the paper intends to prove the Automatic Differentiation capabilities in the context of electromagnetic device sizing by means of gradient optimization. The code complexity as also as the implications of Automatic Differentiation usage may stand as a good reference for the researchers in this field area.

Keywords: Automatic Differentiation, Gradient Constrained Optimization, Software Environment for Optimization, Semi-analytical models.

I. INTRODUCTION

In electromagnetism, the device sizing has often to be carried out by using optimization procedures. The models are very constrained, with numerous parameters, so, the gradient based optimization algorithms are considered e.g. the Sequential Quadratic Programming (SQP) (Gloud and Robinson, 2009). Such algorithms need accurately valued gradients.

Basically, the designer may use explicit analytical formulas to describe sizing mathematical models. In large-scale computational models, the objective functions may not be available in analytical form, but given by a computer program or by routines implementing numerical *algorithms*. These algorithms can implement numerical methods like integration of functions, solvers of non-linear implicit systems of equations, etc… The analytical parts of models interacting with the numerical modules, yields a *semi-analytical* model. A real difficulty may happen to calculate those partial derivatives of the constrained output variables with respect to optimizable inputs when they are linked by a numerical algorithm or computer program. For example, computing the exact derivatives of an objective function computed by a computer program with thousands of code lines simulating the electromagnetic field in a 3D given geometry may be the origin of serious problems.

Several large scale differentiation techniques may be used to accomplish the differentiation task.

The traditional way in electromagnetism to compute derivatives is the *Finite Differences* method such as:

$$f'(x) \cong \frac{f(x+h) - f(x)}{h} \tag{1}$$

This method seems simple, however a suitably chosen step is far to be simple, playing a crucial role for derivatives accuracy. Some studies (Delinchant *et al.*, 2004) show that if the derivatives are not accurately valued, the efficiency of gradient based algorithms decreases considerably.

The *symbolic differentiation* can be used (*Mathematica* (Wolfram, 1999), *Macsyma* (Petti, 1997), *Maple* (Kofler, 1997)). Whereas accurate, this approach is limited by the fact that the representation of the differentiated expression grows fast with the number of input variables. Furthermore, this approach is limited to symbolic expressions and cannot differentiate algorithms.

Contrary to the previously presented methods, the *Automatic Differentiation* (AD) (Griewank, 2000; Bucker and Hovland, 2000) is accurate up to machine precision, minimal in

terms of human computation effort, efficient in terms of running time and applicable for large algorithms. The purpose of this work is to use AD in the context of electromagnetic devices sizing like in (Fischer *et al.*, 2005), which is a study of automatic differentiation application for optimization of electrical devices where only explicit defined mathematical equations are treated. The present paper deals with the further application of AD, especially to differentiate algorithms. Firstly, a brief presentation of the existing AD techniques will be made. Secondly, a special architecture implemented in a software for optimization, called CADES (Delinchant *et al.*, 2007), is presented. It combines both analytical and numerical parts of a sizing model. Finally, a gradient constrained optimization application of a *semi-analytical* electromagnetic model is presented.

II. Introduction to Automatic Differentiation

The *Automatic Differentiation* (AD), is a term applied to a collection of techniques for automatically producing derivative codes of functions implemented in a high-level computer language, such C/C++ or FORTRAN. AD analyses the program functions and the operators and systematically applies the differentiation rules, notably the chain rule, to value error free derivatives.

An automatic differentiator could be a pre-compiler that analyzes the source code and generates a new program that computes both output and derivative values for any user defined set of inputs. Such a technique namely *Automatic Differentiation* by *Source-to-Source* (see Fig. 1) is implemented in packages such TAPENADE (Hascoet and Pascual, 2004), ADIC (Bischof *et al.*, 1997), ADIFOR (Bischof *et al.*, 1996)…

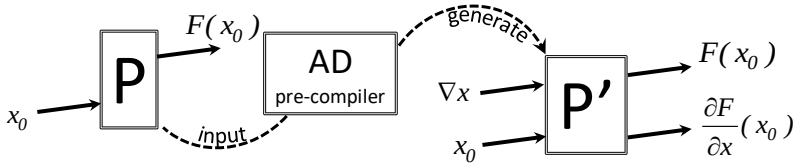


Fig. 1. Source-to-source AD technique.

An automatic differentiator could instrument the routines of an initial code in order to be differentiated. Such tools demand some minor modifications of the initial source. Generally these tools implement this strategy by using the *operator overloading* capacities of certain high-level programming languages such C++, FORTRAN 90. The overloaded operators and operands of each arithmetic operation and elementary function are recorded on a *"tape"*

(Fig. 2). Tapes are subsequently interpreted to provide the required derivatives. The most straightforward AD technique is perhaps based on operator overloading. Such representative tools are CppAD (CppAD 2008), FADBAD/TADIFF (Bendtsen and Stauning, 1996), ADOL-C (Walter and Griewank, 2008) used in the paper…

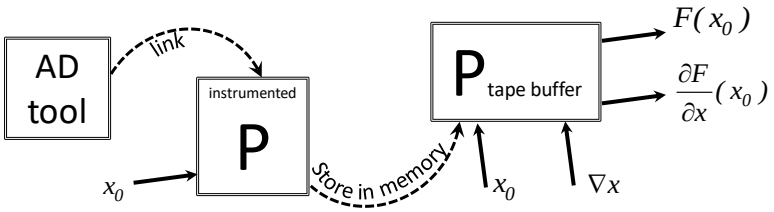
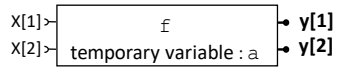


Fig. 2. *Tape storage AD technique*

Almost all the AD tools existing in the literature implement two modes of differentiation, i.e. the *forward mode* and the *reverse mode*. In the following sub-sections, only the forward mode is explained using ADOL-C.

A. Automatic Differentiation –Forward Mode.

Assume the sample program in Fig. 3.a, representing a function $f: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ containing a loop, a conditional branch and an intermediate variable, a. Before computing the derivatives, the initial program needs to be instrumented with ADOL-C, resulting in Fig. 3.b. Here all the variables, including the temporary ones, are of *adouble* data type. The active code is bounded by calling functions *"trace_on()* and *trace_off()"*. These functions are used to build the differentiation tape. Inside the tape, the vectors x and y are defined independent and respectively dependent variables by using operators :**<<=** and **>>=**.




```
function f(double x[]):
return double y[]
```

```
begin function f:
double a, y[2];
int i;
    if(x[1]>2)
       a = x[1] + x[2];
    else
       a = x[1]*x[2];
    for(i = 1; i<=2; i++)
       a = a * x[i];
    y[1] = a/x[2];
    y[2] = sin(x[2]);
end function f;
```

a.

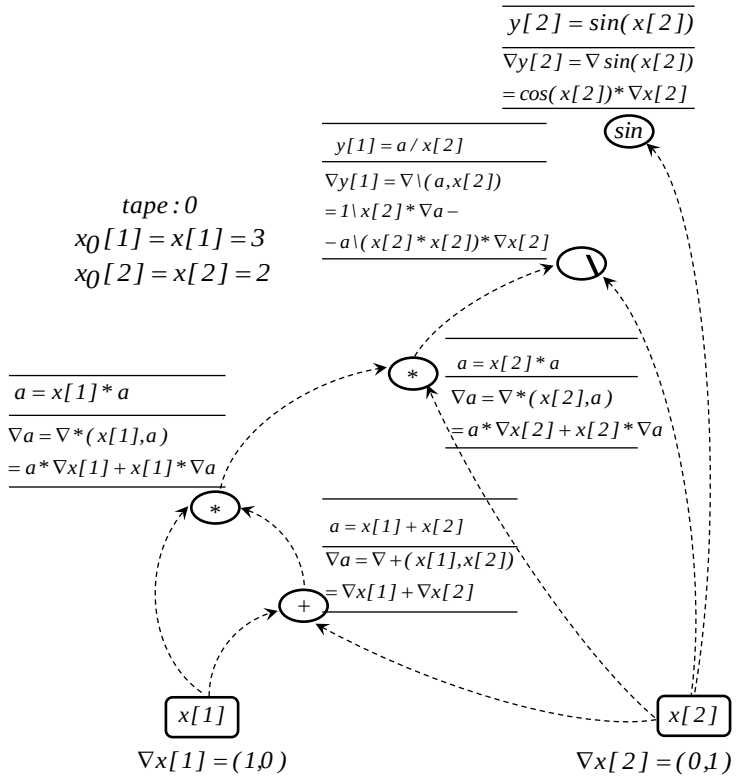
```
function f(adouble x[],double x0[]):
return double y0[]
```

```
begin function f:
adouble a, y[2];
int i;
    trace_on(0);//starting tape buffer
    x[1]<<=x0[1]; x[2]<<=x0[2];//independents
    if(x[1]>2)
       a = x[1] + x[2];
    else
       a = x[1]*x[2];
    for(i = 1; i<=2; i++)
       a = a * x[i];
    y[1] = a/x[2];
    y[2] = sin(x[2]);
    y[1]<<=y0[1]; y[2]<<=y0[2];//dependents
    trace_off();//end tape buffer
end function f;
```

b.

Fig. 3. Example code for AD. *a.:* Initial Code *b.:* The instrumented code with ADOL-C

The tape is the computer representation of the calling graph (see Fig. 4.a.). Since the tape construction is made at run time, it contains no branch and all loops are unrolled. The forward mode of differentiation means that both function and derivatives valuation are made in the same direction (from bottom to the top of the function graph).



a.

```
fov_forward(int traceNb,
 int nbDeps, int nbIndeps, int
keep, double *x0, double **X,
double *y0, double **J)
•traceNb = 0;
•nbDeps = 2;//number of
dependent variables
•nbIndeps = 2;//number of
independent variables
•int keep = 0;//only for reverse
mode
•x0 = (3,2);//user defined set of
inputs;
•double **X = ()=(1,0)(0,1);
//the directions for
differentiation
•double *y0=y; //the output
values;
•double **J = ();//the jacobain
matrix;
```

b.

Fig. 4. The AD tape. *a.:* The tape containing the overloaded operators for the forward mode. *b.:* Procedure used to reuse the tape in order to value derivatives

The trace building is slow (Gay, 2006), but once it is constructed, it may be reused further with different sets of inputs. In ADOL-C this may be done by calling special functions, e.g., *fov_forward*, Fig. 4.b (for *First Order Vectorial* differentiation *in forward* mode*)* that succeeds only if all logical operations of the initial code yield the same result (e.g.: the tape in Fig. 4.a changes if $x[1] = 1$). Otherwise, the retaping is imposed by recalling the routine presented in Fig. 3.b. So, special care must be taken for piecewise differentiable functions. ADOL-C implements the so called mechanism of *branch switching detection.* Thanks to it, the necessity of *retaping* the active program is known.

This discussion shows that the principles underlying AD are not complicated for the computation of first order derivatives, but they can also be easily generalized to the computation of univariate Taylor series or Hessians and multivariate high-order derivatives.

III. ARCHITECTURE STRATEGY FOR SEMI-ANALYTICAL MODELS

In paper (Delinchant *et al., 2007*), the authors describe CADES framework (*Component Architecture for Design of Engineering Systems*), a software environment designed to size and optimize engineering devices and systems. Its design pattern (Fig. 5) that formalizes the entire design process is made of: a *mathematical model* description in the language *sml (*for *System Modelling Language*) (Delinchant *et al.*, 2007),, a model analysis by *Generators* that create software Calculation *Components (models)* and the use of these components in *services (e.g. optimisation, calculation*…).

The *sml* language syntax is based on the *analytical* model description made of algebraic equations and explicitly defined functions. However, this simple description may be limited in electromagnetics. Models using *numerical* methods, or kind of various *numerical algorithms* can be seen as a further model completion. The paper names them *semi-analytical* models.

This section focuses on the features of *numerically* computed *external functions* and *external functions of functions* that interact with the analytical model.

Also, one may need a model capable to interact or to drive simulation environments like FLUX, P-SPICE, MATLAB/SIMULINK or to read and use some measure data bases.

The analytical model of an electrical resistor which resistance depends on temperature is presented to introduce the semi-analytical model syntax in CADES (see Fig. 5)

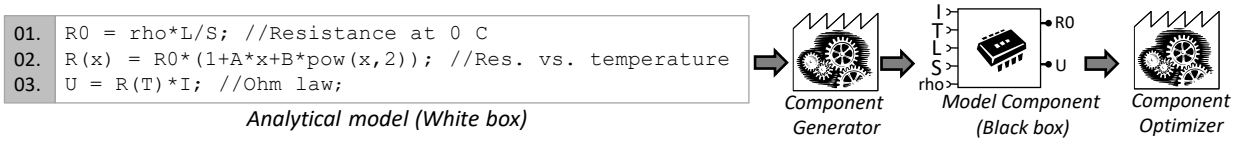


Fig. 5. The chain generation of an analytical model.

The description of analytical models is carried out in the sml language, with the following instructions:

- *scalar constants and variables* of double precision (*R0, rho, L, S, etc…*).
- basic arithmetic operators (+, -, *, /);
- basic mathematical functions (trigonometric, logarithmic, etc… functions);
- user defined internal functions (like *R(x)*);

Model components which are required to carry out the model computation, are automatically produced with respect to a set of rules according to *ICAr* norm (Delinchant *et al.*, 2007).

A. Semi-analytical modeling using external functions

From the previous example, the resistance dependence on temperature can be defined with a numerical algorithm that interpolates some measurement values.

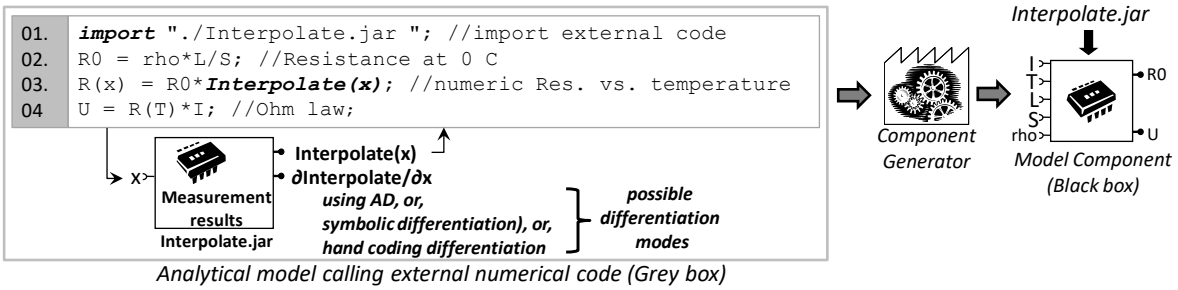
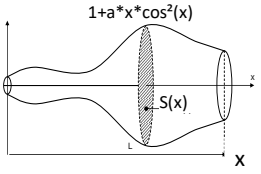


Fig. 6. Semi-analytical model calling external code

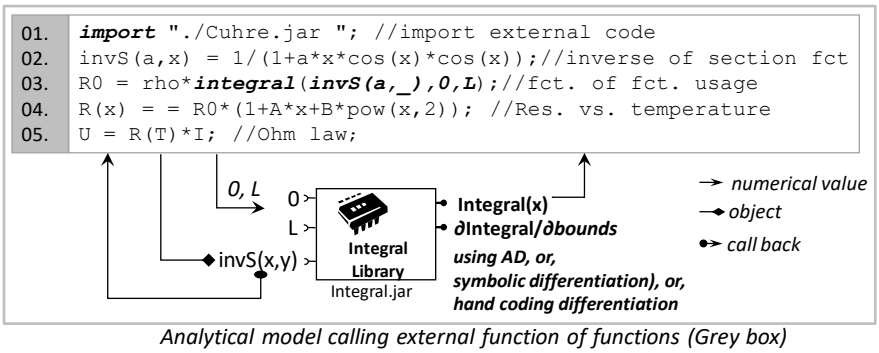
As shown in Fig. 6, an *external function* is a function numerically computed outside the analytical model (e.g. in C or Java language). Note that for further optimization with gradient optimization algorithms like SQP, the jacobian of such an external numerical code is needed and several approaches might be used as shown in Fig. 6.

B. Semi-analytical modeling using the external functions of functions.

Another semi-analytical form lays on *external functions of functions* (extension of the previous formalism). On the example of Fig. 5, suppose that the conductor surface changes like in Fig. 7.a.



a.



b.

Fig. 7. The external functions of functions. *a.*: Conductor shape; application to integral. *b.*: Semi-analytical model calling external function of functions.

The external numerical code computes the integral functional in (2).

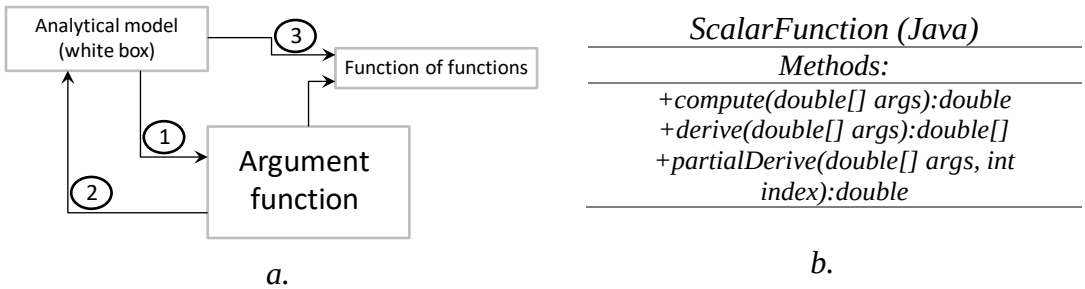
$$\int_0^L invS(a,x)dx \equiv \int_0^L \frac{1}{1+a*\cos^2(x)}dx \qquad (2)$$

Such a *function of functions is translated into sml* language as (3):

$$integral(invS(a,\_),0,L) \tag{3}$$

Note that this syntax approach is formalized and the concerned elements are :

- *integral* - the name of the *function of functions;* this function is implementing a numerical algorithm.
- *invS – the argument function* defined into the analytical model (*arrow 1* in Fig. 8.a). In the integral case, this function represents the integrand.



a. *b.*

Fig. 8. Argument function: a.: strategy b.: architecture

This special argument function is represented by an instance of a class implementing the interface of Fig. 8.b. The methods such *"compute()"*(like in (4)):

$$\begin{aligned} invS(a,\mathbf{0}) &\equiv invSObject.compute(\{\mathbf{0}\});\\ invS(a,\boldsymbol{L}) &\equiv invSObject.compute(\{\boldsymbol{L}\}); \end{aligned} \tag{4}$$

or *"derive()"* (like in (5)):

$$JinvS(a,0) = \left\{\frac{\partial invS}{\partial arg_1}(a,\mathbf{0}), \frac{\partial invS}{\partial arg_2}(a,\mathbf{0})\right\} \equiv invSObj.derive(\{\mathbf{0}\}) \tag{5}$$

permit to calculate this function values and partial derivatives respectively at different points (e.g.: in (4) and (5) at the integration bounds, *0* and *"L"*) required by the numerical routine (here *"integral"*). Thus, a connection is established back (here from *"integral"*) with the analytical model (here "*invS", arrow 2* in Fig. 8.a) and thus *the function of functions* strategy is based on the *call back functions* formalism.

- *"_"* – is used, in the *sml language*, to notify the *variable argument position* among other arguments of argument function (in Fig. 9 the second argument of *"invS"*).

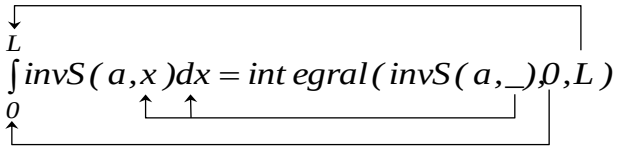


Fig. 9. Variables mapping in the integral case

- *"a, 0, L" – global variables* (G_V). To explicit these variables, consider the formulation in (4) as *F*:

  $$F(a, 0, L) = integral(invS(a, \_), 0, L) \quad (6)$$

  Thus, the variables underlying within this equation are global variables of the complete sizing model. So, the derivatives of *F* in (6) may be needed, if constraints are considered further in optimization.

The jacobian of any *function of function Fof* is:

$$JFof \equiv \left\{\frac{\partial Fof}{\partial G_V}\right\} \quad (7)$$

C. The integral differentiation reference

The numerical integration algorithms used in this paper are Cuhre (for *n>1* dimensions) and Sympson (for *n=1* dimensions), both implemented in Cuba library (Hahn, 2009). This subsection aims to define exact formulations for the partial derivatives of such integrals in order to achieve a strong reference.

Let consider the general *k*-dimension integral in (8), where $I(l_1, u_1, \dots l_k, u_k, T)$ represents the integral result, being a functional depending on the integration bounds and on the *non*-integrable parameters, $T \in \mathbb{R}^t$:

$$I(l_1, u_1, \dots l_k, u_k, T) = \int_{l_1}^{u_1} \dots \int_{l_k}^{u_k} f(x_1, \dots, x_k, T)\, dx_1 \dots dx_k \quad (8)$$

Translated into sml language as:

$$I(l, u, T) = integral(f(\_, \dots, \_, T), l_1, u_1, \dots l_k, u_k)$$

Here, the paper focuses on computing the partial derivatives of the integral with respect to its bounds and to the *non*-integrable parameters, like in (9) and (10).

$$\frac{\partial I}{\partial b_i} = -1^k \int_{l_1}^{u_1} .. \int_{l_{i-1}}^{u_{i-1}} \int_{l_{i+1}}^{u_{i+1}} .. \int_{l_k}^{u_k} f(x_1, .., x_{i-1}, b_i, x_{i+1}, .., x_k, T)\, dx_1 .. dx_{i-1} dx_{i+1} .. dx_k \quad (9)$$

$$\forall\, b_i \in \{l_i, u_i\}, k = \begin{cases} 1, b_i = l_i \\ 0, b_i = u_i \end{cases}$$

$$\frac{\partial I}{\partial T_j} = \int_{l_1}^{u_1} \dots \int_{l_k}^{u_k} \frac{\partial f(x_1, \dots, x_k, T)}{\partial T_j} dx_1 \dots dx_k, \forall j = 1..t \tag{10}$$

Resuming the Cuhre algorithm, it consists in several integrand valuations at discrete points between the lower and the upper bounds. Thanks to functions of functions formalism, the value of the integrand is:

$$fObj.compute(\{l_1, l_2, \dots, l_{i-1}, b_i, l_{i+1}, \dots, l_k\}), \forall b_i \in \{l_1, \dots, l_k\}$$

On the other hand, the integrand in (10), at the lower bound, is:

$$fObj.partialDerive(\{l_1, l_2, \dots, l_i, \dots, l_k\}, k + j), \forall j = 1..t$$

A special attention was carried out for the case when the bounds are functions of *non*-integrable parameters. A new formulation of the relation in (10) yields in this case, to:

$$\frac{\partial I}{\partial T_j} = \int_{l_1}^{u_1} \dots \int_{l_k}^{u_k} \frac{\partial f(x_1, \dots x_k, T)}{\partial T_j} dx_1 .. dx_k + \sum_{i=1}^{k} \frac{\partial u_i}{\partial T_j} \cdot f(x_1 \dots x_k, T) - \sum_{i=1}^{k} \frac{\partial l_i}{\partial T_j} \cdot f, (x_1, \dots x_k, T)$$

IV. THE DESIGN OF AN ELECTROMAGNTIC DEVICE

Here, an electromagnetic actuator applied for a deformable mirror for astronomy adaptive optic (Delinchant *et al.*, 2008; Cugat *et al.*, 2001) is proposed for being sized by gradient constrained optimization. Its model is *semi-analytical* with numerical integrals, using the Cuba library. Two approaches are considered for differentiation.

First, the analytical part of the sizing model is symbolically differentiated, using the differentiation engine implemented in CADES. The numerical integration algorithm is differentiated by hand (see subsection *C* above). This approach is considered the reference for the optimization final solutions.

Second, the entire *semi-analytical* model is differentiated using an AD tool (i.e. ADOL-C).

The same SQP optimization algorithm (i.e. VF13 (Gloud, 2009)) is used for optimization. So, the approaches differ only for differentiation.

A. Micro-actuator for deformable mirror

Here, a deformable mirror reflector actuated by a matrix of several identical micro-actuators is considered. Each actuator, separately controlled, is supposed to avoid image deformations. This device represents a specific application for adaptive optic (Cugat, 2001).

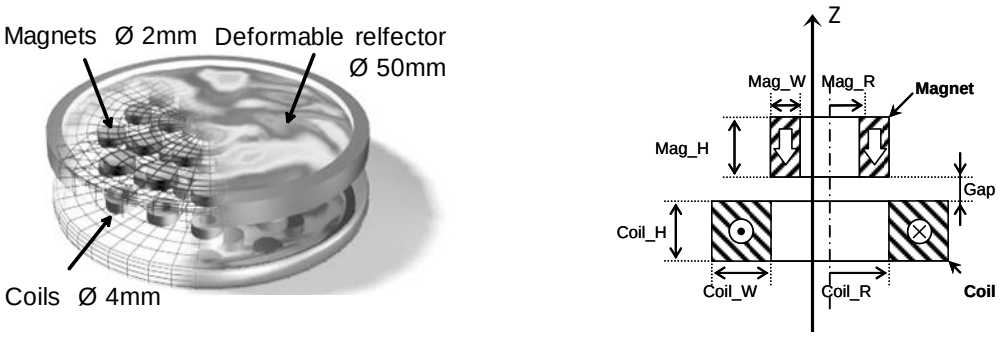


Fig. 10. Deformable mirror system and micro-actuator.

The semi-analytical modeling of each electromagnetic micro-actuator is based on integral formulations for the force acting on the magnet in the electromagnetic field produced by the coil. For the coil shape (see Fig. 10), the Biot-Savard law calculates the magnetic field. We are particularly interested by the field component in the *z-axis*:

$$H_{ZCoil} = \frac{j_\theta}{4\pi} \int_0^{2\pi} g(x,y,z,\theta)\, d\theta \qquad (11)$$

where g represents a functional related to coil shape.

The force acting on the magnet, produced by the field in (11) is calculated by applying the equivalent surface charges method (De Medeiros, 1998), like electrostatic devices, as in (12):

$$\overrightarrow{F_{Mag}} = \iint_S \overrightarrow{\sigma_s} \times \overrightarrow{H_{coil}}\, dS \qquad (12)$$

Only the *z*-component of this force will be used in the modeling approach.

Note the influence of the coil field on the magnet is completely ignored. So the magnet magnetization is considered homogenous and constant, making this formulation deficient in terms of accuracy.

B. The optimization goal

The aim of the optimization of the electromagnetic device in Fig. 10, is to reach a fixed actuating force at air Gap=0.5mm of 30mN, in order to limit the mirror deformability, with a minimal magnet volume. The variation range of geometrical parameters is restricted to the device validity domain. Without such safety measures, the optimization may go out of scope. The initial values, as also the specifications, are reported in the table below.

TABLE I THE OPTIMIZATION SPECIFICATIONS

Parameter	*Description*	*Unit*	*Value*	*Type*
Coil_W	Coil Width	mm	1	fixed
Coil_H	Coil High	mm	1	fixed
Coil_R	Coil Radius	mm	1	fixed
J	Current Density	A/mm	100	fixed
Mz	Magnetization	T	1	fixed
Gap	Air gap	mm	0.5	fixed
Mag_R	Magnet Radius	mm	0.5	Constrained [0.001,1]
Mag_W	Magnet Width	mm	0.5	Constrained [0.1, 3]
Mag_FF	Magnet Form Factor	-	0.25	Constrained [0.01, 1]
prec	Integration accuracy	-	1e-5	fixed
Mag_force	The force on magnet	mN	30	fixed

C. Automatic Differentiation strategy of the integration algorithm

The integration algorithms – the Cuhre and Sympson routines of the *Cuba* integration library, are instrumented with ADOL-C tool package, in order to be differentiated in the sense of calculating the partial derivatives in (9) and (10). The major changes made to the initial algorithm for differentiation, are listed below:

- ***Convert to C++.*** Certain header files (originally written in *C*) were converted to a form acceptable by C++.
- ***Changes to derivative type.*** All the variable types was changed from (*float/double*) to ADOL-C derivative type, *adouble.*
- ***Declaring the independent variables.*** The variables defining the integration bounds ($l_1, u_1, \dots, l_k, u_k$) were declared as *active independent variables*.
- ***Taping the active section.*** The integration algorithm was called between the special functions *"trace_on()"* and "*trace_off()".*
- ***Declaring the dependent variables.*** The variables defining the integration result were defined as active *dependent variables*.

Regarding the complexity of the integration algorithm used in this paper (Hahn, 2009), we tend to consider it enough to demonstrate the actual capacities of AD. Many imbricated loops, conditional branches, integrands called thousands of times since the desired

convergence is achieved, represent a short review that could measure this algorithm complexity.

D. Optimization results

In this subsection, a comparative study is done between the two differentiation strategies applied for the integrals in (11) and (12) that define the constrained force acting on the magnet. This formulation consists in calculating a two dimension integral of an integrand function valuating another one dimension integral.

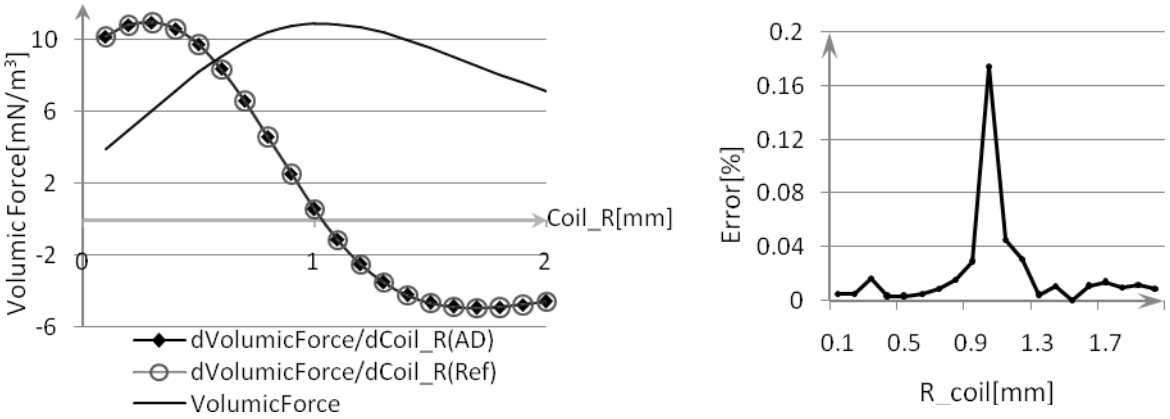


Fig. 11. Partial derivatives and errors obtained in the both considered approaches

For a set of model parameters arbitrary chosen in Fig. 11, it is obvious that AD yields the same partial derivatives, as the considered reference approach. The maximum relative error for plotted partial derivatives in Fig. 11, is 0.18%. Such error is acceptable for a gradient based optimization algorithm, its global convergence being not affected.

In Fig. 12, the initial and the final configuration obtained for the studied device are indicated. The objective value is obtained with a precision of 10^{-4}. The optimization converges in the both approaches with the same iterations number to almost the same value of the objective function.

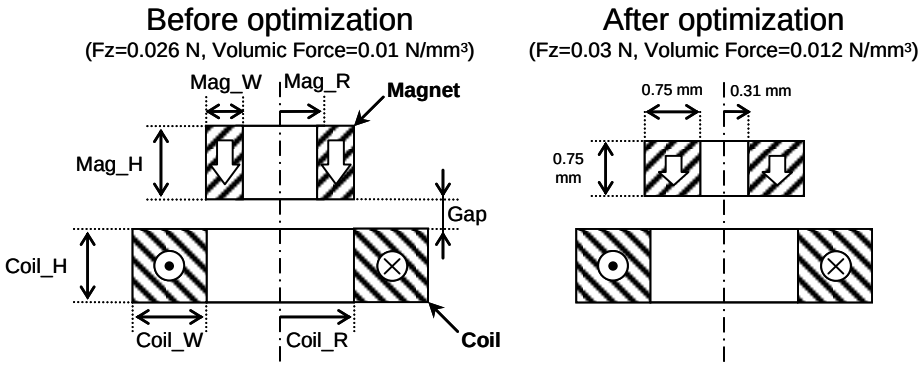
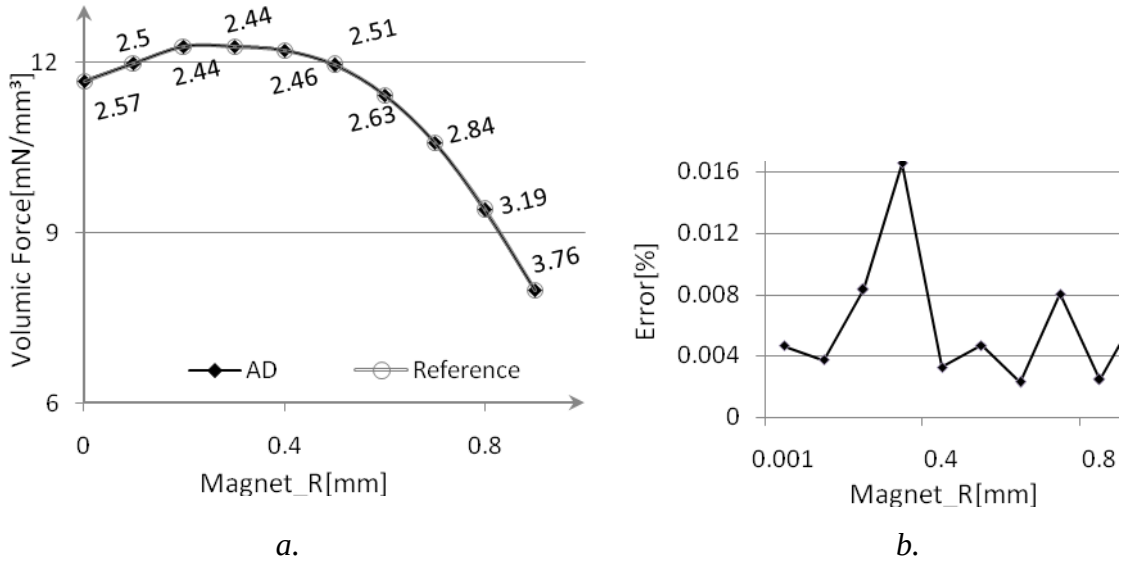


Fig. 12. The actuator structure before and after the optimization

In order to check the robustness of the Automatic Differentiation capabilities, successive optimizations were applied for the same structure with the same specifications varying the coil radius in the range 0…0.9 mm. Outside this range, no solution was found for the both differentiation approaches since the validity domain of the device is no longer respected. In Fig. 13 are reported the optimization results by differentiating the integrals in (12) and (13) using AD and the reference approach. There are no major errors between the optimization solutions.



a. *b.*

Fig. 13. Optimization results for the robustness test: *a.*: The variation of volumic force and objective function when varying the magnet radius. *b.*: The error obtained for the objective function (magnet voulume) within the both differentiation approaches.

V. CONCLUSION

This paper has illustrated AD as a new technology for accurate gradient valuation applied for gradient based optimization of electromagnetic devices. Contrary to symbolic differentiation, AD makes possible to differentiate more than explicitly defined analytical formulas. It might deal with differentiation of numerical codes that are ubiquitous when defining a sizing model in electromagnetism. AD saves work contrarily to hand coding of derivatives and avoids troubles caused by the inaccurate finite differentiation method. In the paper, it is shown that a minimal human computational effort is required when differentiating a numerical code. More, no user knowledge is required for the algorithm in cause. The

gradient accuracy obtained with AD is validated within the optimization of an electromagnetic actuator. In comparison with a reference approach, the values of the derivatives computed in double precision by AD agree to at least four digits. Such accuracy is acceptable for an SQP optimization algorithm since the solution as also as the iterations number is quite identical.
However, the running of the derivative code implemented with ADOL-C tends to be slower than the reference approach case.

A special architecture is proposed in this paper to handle the interaction between the analytical part describing a sizing model and the numerical codes within a software environment designed for sizing. Situations when such functions of functions are useful might be when the external function is supposed to compute an integral, or an implicit solver (Enciu *et al.*, 2008), minimizations/maximizations of functions…

VI. REFERENCE

Gloud, N. I. M. and Robinson, D. P. (2009), "A second derivative SQP method: local convergence", available at http://www.numerical.rl.ac.uk/reports/reports.shtml, (accessed 08 March 2009).

Delinchant, B., Wurtz, F. and Atienza, E. (2004), "Reducing Sensitivity Analysis Time-Cost of the Compound Model", *IEEE Transactions on Magnetics*, vol. 40, No. 2, pp. 1216-1219.

Wolfram, S. (1999) *The Mathematica Book*, Cambridge University Press, 4th Edition.

Petti, R. J. (1997) *Introduction to Macsyma*, Publishers, Inc.

Kofler, M. (1997) Maple: An Introduction and Reference, Addison Wesley.

Griewank, A. (2000) *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers in Applied Mathematics no. 19, SIAM, Philadephia.

Bücker, M. and Hovland, P. (2000) "Automatic Diffrentiation", available at www.autodiff.org, (accessed 08 March 2009)

Fischer, V., Gerbaud, L., Wurtz, F., Leconte, V. and Dorschner, F. (2005), "Using Automatic Code Differentiation for Optimization", *IEEE Transactions on Magnetics*, vol. 41, No. 5, pp. 1812 – 1815.

Delinchant, B., Duret, D., Estrabaut, L., Gerbaud, L., Nguyen, H. H., Du Peloux, B., Rakotoarison, H.L., Verdiere, F. and Wurtz, F. (2007) "An optimizer using the software component paradigm for the optimization of engineering systems", *COMPEL*, Vol. 26, Issue 2, Page: 368 - 379.

Hascoet, L. and Pascual, V. (2004), working paper, "TAPENADE 2.1 user's guide", Institut National de Recherche en Informatique et en Automatique, No. 0300, France, Sept.

Bischof, C., Roh, L. and Mauer, A. (1997), working paper, "ADIC: An Extensible Automatic Differentiation Tool for ANSI-C", Center of Research on Parallel Computation, Rice University, January.

Bischof, C. H., Carle, A., Khademi, P. and Mauer, A. (1996), "ADIFOR 2.0: Automatic Differentiation of FORTRAN 77 Programs", *IEEE Computational Science & Engineering*, 3(3): 18-32.

CppAD (2008), available at, http://www.coin-or.org/CppAD/ (accessed at 08 March 2009).

Bendtsen, C. and Stauning, Ole (1996), working paper, "FADBAD, a Flexible C++ Package for Automatic Differentiation, Department of Mathematical Modelling", Technical University of Denmark.

Walter, A. and Griewank, A. (2008), working paper, ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++, Institute of Scientific Computing, Technical University Dresden, D-01062, December.

Gay, D. M. (2006), "Semiautomatic Differentiation for Efficient Gradient Computations", *Automatic Differentiation: Applications, Theory and Implementations*, Springer.

Hahn, T. (2009), working paper, *"Cuba: a library for multidimensional numerical integration"*, Max-Planck-Institut fur Physik, Munich, Germany.

Enciu, P., Gerbaud, L. and Wurtz, F. (2008), "Automatic Differentiation for Sensitivity Calculation in Electromagnetism: Application to Algorithms", in *IEEE Conference on Electromagnetic Field Computation proceedings of International conference*, Athens, Greece, May.

Delinchant, B., Gruosso, G. and Wurtz, F. (2008), "Two-levels modelling for the optimization of electromagnetic actuators", IEEE Transactions on Magnetics (to be published).

Cugat, O., Basrour, S., Divoux, C., Mounaix, P. and Reyne, G. (2001), "Deformable magnetic mirror for adaptive optics: technological aspects", *Sensors and Actuators A: Physical*, Volume 89, Issues 1-2, 20, pp. 1-9.

De Medeiros, L. H., Reyne, G. and Meunier, G. (1998), "Comparison of Global Force Calculations on Permanent Magnets", *IEEE Transactions on Magnetics*, VOL.34, NO. 5.