Isogeometric shape sensitivity analysis

F. Navarrina, H. Gómez, J. París, I. Colominas,
X. Nogueira & M. Casteleiro
*GMNI — Group of Numerical Methods in Engineering,
Civil Engineering School, University of A Coruña, Spain*

Abstract

On a regular basis, engineering analysis requires stating and solving systems of partial differential equations (PDEs). The most powerful and widely extended techniques for solving PDEs are the so-called Weighted Residuals Methods. To this group belong, among others, the Finite Element Method (FEM), the Boundary Element Method (BEM), the Finite Volume Method (FVM) and the Mesh-Free Method (MFM), as well as the many different formulations included in each of these categories. The new Isogeometric Analysis (IGA) methods were proposed by Hughes *et al.* in 2005, and it is our belief that they really deserve special attention. The key idea of IGA is to use a previously generated CAD model for discretizing both the geometry and the solution to the problem being analyzed. In return for some minor drawbacks, IGA offers a number of major advantages that make the technique specially attractive and promising in comparison with standard FEM, BEM, FVM and MFM formulations.

In this presentation we will state a general formulation for the sensitivity analysis of Weighted Residual Methods, both for linear and non-linear problems with constant or varying geometry. The effects due to variation of geometry are addressed by defining a generic procedure for integration in manifolds on the basis of the metric tensor concept. The proposed approach leads to compact and relatively simple expressions to obtain directional derivatives of arbitrarily high order. The resulting scheme can be easily applied to IGA formulations, its implementation being quite a straightforward task. Finally, one application example is presented.
Keywords: IGA, isogeometric analysis, high order shape sensitivity analysis.

1 Introduction

As a general rule, engineers are required to state and manipulate quite complicated computational models in order to make proper decisions regarding the design and the operation of structures, devices or systems of any kind.

We can think of the actual computational model being used for analyzing a certain case as a black box (that will be called analysis module from here on) type

$$\xrightarrow{\boldsymbol{\alpha}} \boxed{\text{ANALYSIS MODULE}} \xrightarrow{\boldsymbol{\omega}} \tag{1}$$

where $\boldsymbol{\alpha}$ represents the set of input data that are provided by the user, and $\boldsymbol{\omega}$ represents the corresponding set of output results that are provided by the computational model for each given case. We will say that a direct problem is being solved when the value of the outputs $\boldsymbol{\omega}$ are to be obtained for given values of the inputs $\boldsymbol{\alpha}$. This is what the analysis module is originally made for, and this might be all what was needed in certain real cases. But not always, indeed.

If we were dealing with a so-called calibration problem, our aim would be to set up the values of some (or all) of the adjustable parameters $\boldsymbol{\alpha}$ for the results $\boldsymbol{\omega}$ to reach a desired target value. On the other hand, if we were dealing with a so-called parameter estimation problem (also called inverse problem, as opposed to the concept of direct problem), our aim would be to find out the values of some (or all) of the hidden parameters $\boldsymbol{\alpha}$ for the computed outputs $\boldsymbol{\omega}$ to fit some real measures (taken from the real structure, device or system under consideration). Finally, if we were dealing with a so-called optimum design problem, our aim would be to choose the optimum values of some (or all) of the inputs $\boldsymbol{\alpha}$ for the results $\boldsymbol{\omega}$ to minimize a suitable objective function with some additional constraints.

The solution to a problem in any of the above mentioned categories (calibration, parameter estimation and optimum design) could be tackled by trial and error or by means of some heuristics. If we proceed in this way, the analysis module will be simply used (as a black box) for solving a sequence of direct problems. This is what we call a zeroth-order approach. Zeroth-order approaches are conceptually simple and easy to implement. But the price to pay for these advantages is a very poor rate of convergence (kind of 1st-order). For this reason, an extremely large (if not completely out of range) number of iterations is expected as a general rule. This effect strongly limits the accuracy and the efficiency of zeroth-order methods for solving many real problems, what precludes their use in a number of cases. Other more sophisticated (but still zeroth-order) techniques of this type would be the bisection method (in the case of one-dimensional calibration problems) and the golden search method (in the case of one-dimensional optimum design problems.

The above discussion makes clear why higher order approaches (type Newton-Raphson, with a better rate of convergence) are called for solving calibration, parameter estimation and optimum design problems. But the price to pay for a higher rate of convergence (2nd-order) is the need for computing the derivatives of the outputs $\boldsymbol{\omega}$ with respect to the inputs $\boldsymbol{\alpha}$, what is called the sensitivity analysis.

2 Sensitivity analysis

For a given analysis module (1), our aim is to compute the derivatives

$$\frac{d\boldsymbol{\omega}}{d\boldsymbol{\alpha}}, \frac{d^2\boldsymbol{\omega}}{d\boldsymbol{\alpha}^2}, \dots \frac{d^i\boldsymbol{\omega}}{d\boldsymbol{\alpha}^i}, \dots \frac{d^n\boldsymbol{\omega}}{d\boldsymbol{\alpha}^n}, \tag{2}$$

up to a certain order n. In real engineering problems both the number of inputs $\dim(\boldsymbol{\alpha})$ and the number of outputs $\dim(\boldsymbol{\omega})$ are expected do be quite large. On the other hand, the number of derivatives (2) increases exponentially with the order of derivation n. Thus, both the computational cost and the amount of memory storage needed for performing a complete nth order sensitivity analysis must be considered unaffordable in most real cases.

However, not all derivatives (2) will be needed as a general rule, since part of the inputs $\boldsymbol{\alpha}$ are expected to be constant and part of the outputs $\boldsymbol{\omega}$ are expected to be irrelevant for the particular case being considered. On the other hand, the derivatives of a function and the function itself belong to a different class (both from the mathematical and from the computer science point of view). Thus, the ith order derivative of $\boldsymbol{\omega}$ with respect to $\boldsymbol{\alpha}$ in (2) is a multidimensional array with $i+1$ indices. This effect poses additional difficulties at the time of implementing the sensitivity analysis in a computer code. We conclude that the sensitivity analysis procedure will be more suitable for each particular case if it allows to select which derivatives are to be computed. Moreover, the implementation of the procedure will be definitely easier if the results $\boldsymbol{\omega}$ and the selected derivatives to be computed are unidimensional arrays of the same size.

The answer for achieving both objectives lies with directional derivatives. For a known value of the first order directional derivative of the inputs $\boldsymbol{\alpha}$ along a certain vector $\boldsymbol{s}$, the first order directional derivative of the outputs $\boldsymbol{\omega}$ along the same vector is given by

$$D_s\ \boldsymbol{\omega} = \left[\frac{\partial\boldsymbol{\omega}}{\partial\boldsymbol{\alpha}}\right]\ D_s\ \boldsymbol{\alpha}. \tag{3}$$

The concept can be extended for higher order derivatives. Thus, the nth order directional derivative of the outputs $\boldsymbol{\omega}$ along vectors $\{\boldsymbol{s}_1, \boldsymbol{s}_2, \dots, \boldsymbol{s}_n\}$ is given by

$$D^n_{s_1,s_2,\dots,s_n}\ \boldsymbol{\omega} = D_{s_n}\ \left(\dots D_{s_2}\ \left(D_{s_1}\ \boldsymbol{\omega}\right)\right). \tag{4}$$

Now it seems clear that the information provided by the high order sensitivity analysis allows for computing approximations to the outputs $\boldsymbol{\omega}$ without the need to repeat the analysis when the inputs $\boldsymbol{\alpha}$ are modified. Thus, if the inputs $\boldsymbol{\alpha}$ are modified as

$$\widetilde{\alpha}(\theta) = \boldsymbol{\alpha} + \sum_{k=1}^{n} \frac{1}{k!}\ \left(D^k_{s^k}\ \boldsymbol{\alpha}\right)\ \theta^k + \mathcal{O}(\theta^{n+1}), \tag{5}$$

in terms of a parameter θ, then the corresponding outputs $\boldsymbol{\omega}$ take the values

$$\widetilde{\omega}(\theta) = \boldsymbol{\omega} + \sum_{k=1}^{n} \frac{1}{k!}\ \left(D^k_{s^k}\ \boldsymbol{\omega}\right)\ \theta^k + \mathcal{O}(\theta^{n+1}). \tag{6}$$

This is what Lucien Schmidt [1] called "Structural Synthesis Concept" in his pioneering works on structural optimization. And this is the essential point for understanding why sensitivity analysis plays a key role for solving calibration, parameter estimation and optimum design problems in engineering (as well as control problems, that are basically time-dependent problems of any of the former three categories).

Finally, it seems also clear that numerical differentiation should be avoided for at least two reasons: the lack of accuracy and the high computational cost. To illustrate this point, let's suppose that the following procedure is used for approximating the first order derivative of a scalar function of one variable $f(x)$:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h). \tag{7}$$

It is obvious that the correct balance between round-off and truncation errors is difficult to achieve in practice, since it depends on the value of the increment h. On the other hand, for computing the single derivative $f'(x)$ by means of approximation (5) the original function must be evaluated twice. These effects are more noticeable indeed in the case of multiple variables.

3 Linear and non-linear analysis

For the sake of simplicity, we can think of the analysis module (1) as a computer implementation of a numerical method that states and solves the so-called state equation

$$\boldsymbol{\psi}(\boldsymbol{\alpha}, \boldsymbol{\omega}) = \mathbf{0}. \tag{8}$$

However, the forthcoming concepts can be also quite easily extended to time-dependent and eigenvalue problems.

If the state function $\boldsymbol{\psi}(\boldsymbol{\alpha}, \boldsymbol{\omega})$ is linear

$$\boldsymbol{\psi}(\boldsymbol{\alpha}, \boldsymbol{\omega}) = \boldsymbol{K}(\boldsymbol{\alpha})\, \boldsymbol{\omega} - \boldsymbol{b}(\boldsymbol{\alpha}) = \mathbf{0}, \tag{9}$$

then the analysis module is prepared for solving linear systems type

$$\boldsymbol{K}(\boldsymbol{\alpha})\, \boldsymbol{\omega} = \boldsymbol{b}(\boldsymbol{\alpha}) \tag{10}$$

by means of a direct solver (very likely a sparse matrix factorization solver type Cholesky or Crout) or an iterative solver (type PCG o GMRES).

On the other hand, if the state function $\boldsymbol{\psi}(\boldsymbol{\alpha}, \boldsymbol{\omega})$ is non-linear the analysis module is prepared for performing iterations until convergence type Newton-Raphson's method

$$\begin{cases} \text{solve} & \boldsymbol{K}(\boldsymbol{\alpha},\boldsymbol{\omega}_\kappa)\,\Delta\boldsymbol{\omega}_\kappa = \boldsymbol{b}(\boldsymbol{\alpha},\boldsymbol{\omega}_\kappa), \\ \text{update} & \boldsymbol{\omega}_{\kappa+1} = \boldsymbol{\omega}_\kappa + \Delta\boldsymbol{\omega}_\kappa, \end{cases} \tag{11}$$

where

$$\begin{cases} \boldsymbol{K}(\boldsymbol{\alpha},\boldsymbol{\omega}) = \left[\dfrac{\partial\boldsymbol{\psi}}{\partial\boldsymbol{\omega}}\right], \\ \boldsymbol{b}(\boldsymbol{\alpha},\boldsymbol{\omega}) = -\boldsymbol{\psi}(\boldsymbol{\alpha},\boldsymbol{\omega}). \end{cases} \tag{12}$$

Therefore, each non linear step (iteration) is roughly equivalent to a full linear analysis.

4 Sensitivity analysis master techniques

Direct differentiation of the state equation (8) yields the linear system

$$\left[\frac{\partial\boldsymbol{\psi}}{\partial\boldsymbol{\omega}}\right]\ D_s\,\boldsymbol{\omega} = -\left(\frac{\partial\boldsymbol{\psi}}{\partial\boldsymbol{\alpha}}\ D_s\,\boldsymbol{\alpha}\right). \tag{13}$$

If the state function $\boldsymbol{\psi}(\boldsymbol{\alpha},\boldsymbol{\omega})$ is linear as in (9) then

$$\left[\frac{\partial\boldsymbol{\psi}}{\partial\boldsymbol{\omega}}\right] = \boldsymbol{K}(\boldsymbol{\alpha}), \quad \frac{\partial\boldsymbol{\psi}}{\partial\boldsymbol{\alpha}}\ D_s\,\boldsymbol{\alpha} = D_s\,\boldsymbol{b} - (D_s\,\boldsymbol{K})\ \boldsymbol{\omega}. \tag{14}$$

Hence, the sensitivity analysis consists in solving the linear system

$$\boldsymbol{K}(\boldsymbol{\alpha})\ D_s\,\boldsymbol{\omega} = D_s\,\boldsymbol{b} - (D_s\,\boldsymbol{K})\ \boldsymbol{\omega} \tag{15}$$

which coefficient matrix is the same as in (10). Therefore, if the analysis module is based on a direct solver, the computing overload associated to the sensitivity analysis is expected to be small in comparison with the analysis itself (about the cost of calling for one additional load case).

On the other hand, if the state function $\boldsymbol{\psi}(\boldsymbol{\alpha},\boldsymbol{\omega})$ is non-linear, we notice that (13) is still a linear system with the same coefficients matrix that was used in the last iteration of Newton-Raphson's method (11). Therefore, if the analysis module is based on a direct solver, the computing overload associated to the sensitivity analysis is expected to be also small in comparison with the analysis itself (about the cost of performing one more Newton-Raphson iteration in the analysis).

The first order sensitivity analysis procedure above explained is known as the Direct Differentiation Method. Depending on other considerations (see [1, 2] for more details), the operations involved can be conveniently rearranged. This leads to the so-called Adjoint State Method [3, 4].

Finally, deriving a high order sensitivity analysis procedure by means of the same principles above outlined is just a matter of algebra. Thus, if the state function $\boldsymbol{\psi}(\boldsymbol{\alpha},\boldsymbol{\omega})$ is linear as in (9) recursive differentiation of the state equation (8) yields linear systems type

$$\boldsymbol{K}(\boldsymbol{\alpha})\, D^n_{s^n}\, \boldsymbol{\omega} = D^n_{s^n}\, \boldsymbol{b} - \left(D^n_{s^n}\, \boldsymbol{K}\right) \boldsymbol{\omega} - \sum_{i=1}^{n-1} \binom{n}{i} \left(D^i_{s^i}\, \boldsymbol{K}\right) D^{n-i}_{s^{n-i}}\, \boldsymbol{\omega}. \quad (16)$$

A similar (but more cumbersome) expression can be derived for the non-linear case. One again, we notice that the coefficient matrix is the same as in (10) and in (15). Therefore, if the analysis module is based on a direct solver, the computing overload associated to the high order sensitivity analysis is still expected to be small in comparison with the analysis itself.

We conclude that in practice it will be realistic to ask for the first directional derivative (or the full gradient, if necessary) of the outputs $\boldsymbol{\omega}$. And it will also be realistic to ask for their second (or higher) order directional derivative. But it will not be realistic, as a general rule, to call for the full hessian (or higher order full set of derivatives) of the outputs $\boldsymbol{\omega}$ due to the large number of items to be computed and to the associated amount of memory storage that would be required.

5 Isogeometric analysis (IGA)

Very likely, the Analysis Module (1) will consist on a FEM, BEM, FVM, MFM or IGA code. Hence, we are talking about a quite sophisticated intensive CPU time consuming tool.

All these formulations belong to the group of the so-called Weighted Residuals Methods. Describing and enumerating the similarities and the differences between all of them is far beyond the scope of this paper. But, as far as we are concerned in this section, the aim of the Weighted Residual Methods is to obtain an approximate numerical solution for a given partial differential equation in a given domain. For achieving this objective, the weighted residuals principle (and other techniques, such as the Divergence Theorem) are used to convert the original partial differential equation into a suitable variational weak form. Then, the solution to the problem is discretized in order to convert the above mentioned variational weak form into an algebraic system of linear or non-linear equations type (8). Finally, the geometry of the domain is discretized in order to compute the integral terms that are introduced by the weighted residuals principle. Quite frequently both, the solution to the problem and the geometry of the domain, are discretized by means of the same interpolation techniques (isoparametric interpolation), what contributes some computational advantages.

FEM, BEM, FVM and MFM formulations are based on classic interpolation techniques (type Lagrangian interpolation, least squares, etc.). On the contrary, IGA formulations are based on the much more powerful and versatile CAD interpolation techniques, namely B-Splines, NURBS and (more recently) T-Splines. The IGA concept was first introduced in 2005 by Hughes *et al.* [5]. The original aim of IGA was to avoid the need for producing the coarse and non smooth FEM type discretizations that were needed for the analysis, while the much more sophisticated and smooth discretizations that were generated by CAD systems had to be systematically discarded. The key idea is quite simple: just import the discretization generated by the CAD system into the analysis module, and use it for discretizing both, the geometry and the solution to the problem.

B-Splines, NURBS and T-Splines belong to the same family of interpolation methods. Again, describing and enumerating the similarities and the differences between all of them is far beyond the scope of this paper. Essentially NURBS are B-Splines in homogeneous coordinates [6]. Although this slight difference gratefully improves the power of the interpolation technique, the underlying concepts are essentially identical. On the other hand, the aim of the T-Splines formulation is to provide local h-refinement in bi-dimensional and three-dimensional discretizations. The T-Spline formulation is more complicated (and also more versatile and powerful) than the B-Spline formulations, but the underlying concepts regarding IGA are basically the same. For these reasons we will focus on B-Splines.

Let the definition domain of a certain problem be a curve embedded in the three-dimensional material space. The IGA B-Spline one-dimensional interpolation for this case can be written as

$$\begin{cases} \boldsymbol{r}^h(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi)\, \boldsymbol{r}_i, \\ u^h(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi)\, u_i, \end{cases} \qquad \xi \in [0,1]. \tag{17}$$

For each value of the so-called reference coordinate ξ, the above expressions give the interpolated position of each point $\boldsymbol{r}^h(\xi)$ within the domain and the interpolated value of the solution $u^h(\xi)$ to the problem at that point. The so-called control points $\{\boldsymbol{r}_i\}_{i=1,\dots,n}$ will be provided by the CAD system and define the geometry of the domain (the curve itself) of the problem being solved. On the other hand, the unknowns $\{u_i\}_{i=1,\dots,n}$ must be computed by the analysis module in order to approximate the solution to the problem.

The base of shape functions $N_{i,p}(\xi)$ in the above expression are given by the Cox-De Boor recursion formula [7]

$$\begin{cases} N_{i,q}(\xi) = \begin{cases} 1 & \text{if } \xi \le \xi_{i+1}, \\ 0 & \text{otherwise,} \end{cases} & \begin{cases} q = 0, \\ i = 1, \dots, n + (p - q). \end{cases} \\ N_{i,q}(\xi) = \left(\dfrac{\xi - \xi_i}{\xi_{i+q} - \xi_i} \right) N_{i,q-1}(\xi) & \\ \qquad + \left(\dfrac{\xi_{i+q+1} - \xi}{\xi_{i+q+1} - \xi_{i+1}} \right) N_{i+1,q-1}(\xi), & \begin{cases} q = 1, \dots, p, \\ i = 1, \dots, n + (p - q). \end{cases} \end{cases} \tag{18}$$

The above computations are normally implemented by means of the Piegl and Tiller Algorithms [8], that are considered as the best numerically stable and fully optimized routines to perform the Cox-De Boor recursion. Functions $N_{i,p}(\xi)$ generated by this procedure are piecewise polynomials in terms of the variable ξ. We also notice that for $\xi_1 \leq \xi \leq \xi_{n+p+1}$, the following assertions hold [7]:

$$\begin{cases} \sum_{i=1}^{n} N_{i,p}(\xi) = 1 & \text{(partition of unity),} \\ N_{i,p}(\xi) \geq 0, & \\ N_{i,p}(\xi) \in \mathcal{C}^{p-1} & \text{(if knots are not repeated),} \\ N_{i,p}(\xi) \neq 0 & \text{in } (p+1) \text{ knot spans at most.} \end{cases} \tag{19}$$

The partition of unity assertion in (19) guarantees that the IGA approximation type (17) for $u^h(\xi)$ is able to represent (exactly) a constant function (what happens when u_i takes the same value for all i). This is an essential requirement for an interpolation technique to be acceptable in computational mechanics as a general rule. On the other hand, the last assertion in (19) is quite surprising, but also important and beneficial, since it guarantees that the bandwidth of the final system type (8) will not be too large.

The degree and the continuity of the piecewise polynomials $N_{i,p}(\xi)$ are controlled by the so called knot vector [7]

$$\begin{aligned} &\Xi = [\xi_1, \ldots, \xi_i, \xi_{i+1}, \ldots, \xi_{n+p+1}], \\ &\text{with } \xi_j \in \mathrm{I\!R}\ \forall j, \text{ and } \quad \xi_i \leq \xi_{i+1} \text{ for } 1 \leq i \leq n+p. \end{aligned} \tag{20}$$

where the ξ_j are called knots, $[\xi_i, \xi_{i+1}]$ is a so-called knot span, n is the number of basis functions and p is the polynomial order. We notice that knots must form a non decreasing sequence, although multiplicity is allowed (and has some important effects). As a general rule, the thus defined IGA approximation (17) is not guaranteed to be "interpolatory" in the strict sense, since it is possible that $\boldsymbol{r}^h(\xi)$ and $u^h(\xi)$ will not match the values of $\boldsymbol{r}_i$ and u_i (respectively) for any value of the reference coordinate ξ. However, if a knot is repeated, the approximation looses one order of continuity for each repetition of the knot at the corresponding value of ξ . Therefore, the approximation becomes discontinuous if the knot is repeated exactly $p+1$ times. Furthermore, it becomes interpolatory if the knot is repeated exactly p times. In many practical applications the knot vector is said to be open and uniform. This means that $\xi_1 = \xi_2 = \ldots = \xi_{p+1} = 0$, $\xi_{n+1} = \xi_{n+2} = \ldots = \xi_{n+p+1} = 1$ and knots $\{\xi_{p+1}, \ldots, \xi_{n+1}\}$ are uniformly distributed in the interval $[0, 1]$. The thus defined IGA approximation is interpolatory only at the first and at the last control point [7].

Another important issue in favor of IGA is the fact that the so-called Oslo Knot Insertion procedure [7] allows for creating new knots while the domain geometry remains unchanged. Therefore, the functional interpolation can be improved as much as desired without affecting the geometry of the problem. This makes h-refinement a trivial automatic task that can be done on the fly by the analysis code, without the need for a mesh refinement preprocessor. However, knot insertion in bi-dimensional and three-dimensional problems makes h-refinement non-local (and this is one of the reasons why there is so much interest and research on T-Splines at present)

Finally, bi-dimensional and three-dimensional B-Spline interpolations can be easily generated by means of a tensor product of several one-dimensional B-Spline interpolations type (17), thus giving [7]

$$\left\{\begin{aligned} \boldsymbol{r}^h(\xi,\eta,\chi) &= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{\ell} N^{\xi}_{i,p_\xi}(\xi)\, N^{\eta}_{j,p_\eta}(\eta)\, N^{\chi}_{k,p_\chi}(\chi)\, \boldsymbol{r}_{i,j,k}, \\ u^h(\xi,\eta,\chi) &= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{\ell} N^{\xi}_{i,p_\xi}(\xi)\, N^{\eta}_{j,p_\eta}(\eta)\, N^{\chi}_{k,p_\chi}(\chi)\, u_{i,j,k} \end{aligned}\right. \quad (\xi,\eta,\chi)\in[0,1]^3. \tag{21}$$

IGA can be quite easily implemented within an existing FEM code. The rules to do so are quite simple:

1. ELEMENTS are substituted by KNOT SPANS.
2. NODES are substituted by CONTROL POINTS.
3. SHAPE FUNCTIONS are substituted by the PIEGL AND TILLER ALGORITHMS.
4. The rest remain unchanged (same weak form, same general organization of the code, same integration formulas, etc.).

It is true that IGA has some drawbacks in comparison with FEM, BEM, FVM, MFM formulations since unknowns no longer represent nodal values, essential boundary conditions may become non-trivial and geometric modelling is much less intuitive. But IGA also offers a number of major advantages, namely: CAD models are expected to be analyzed without the need for mesh generation (although this has not been fully accomplished because CAD systems are designed to model surfaces mainly, instead of solids); mesh refinement becomes (almost) trivial, and geometry is not modified in the process; continuity can be arbitrarily raised without heavy tolls to pay for in return; the capacity of the formulation for approximating complex behaviors is dramatically increased; and higher order problems can be addressed (such as Phase-Field models type Cahn-Hilliard and Navier-Stokes-Korteweg equations, and many others).

6 Sensitivity analysis specifics for IGA

In Weighted Residuals based formulations (such as IGA) state equation (8) takes the general form

$$\boldsymbol{\psi}(\boldsymbol{\alpha},\boldsymbol{\omega}) = \int_{\boldsymbol{r}\in\Omega(\boldsymbol{\alpha})} \boldsymbol{\mathcal{P}}(\boldsymbol{r},\boldsymbol{\alpha},\boldsymbol{\omega})\, d\Omega, \tag{22}$$

where $\boldsymbol{\alpha}$ are the input data to the IGA analysis module (geometry, physical properties, loads, etc.), $\boldsymbol{\omega}$ are the so called state variables or output results (deformation–stress, velocity-pressure, etc.), Ω is the definition domain of the problem embedded in the material space (normally $\mathbb{R}^3$) and $\boldsymbol{r}$ are the coordinates of an arbitrary point in Ω. For the sake of simplicity, we assume that geometry (and hence, domain Ω) does not depend on $\boldsymbol{\omega}$. Otherwise, the suitable corrections should be included.

Therefore, the sensitivity analysis of this kind of formulations requires taking derivatives of functions that are defined by integration in arbitrary domains. If the geometry of the problem being solved is constant, so will be the integration domains. In these conditions it is fairly straightforward to state the sensitivity analysis by means of fully analytical techniques. On the contrary, if the geometry of the problem being solved is not constant (i.e. in shape optimization or shape parameter estimation problems, non-confined flow problems, etc.) stating the sensitivity analysis is not immediate. For this reason, the sensitivity analysis for varying-geometry problems has been mainly addressed by means of low order finite difference approximations, that are known to be inaccurate and difficult to calibrate. In an attempt to overcome this drawbacks, a number of analytical approaches have been proposed to address the sensitivity of 1) the numerical implementation, 2) the analytical model in weak form, and 3) the discretized formulation corresponding to the numerical method being used [1, 3]. The latter approach is in perfect agreement with the analysis model being used, and gives rise to a most compact and flexible formulation than the other options. For those reasons, this will be the approach that we will follow in this paper.

As shown in (13) we need to compute sensitivity terms type

$$\left[\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\omega}}\right] = \frac{\partial}{\partial \boldsymbol{\omega}} \int_{\boldsymbol{r}\in\Omega(\boldsymbol{\alpha})} \boldsymbol{\mathcal{P}}(\boldsymbol{r}, \boldsymbol{\alpha}, \boldsymbol{\omega})\, d\Omega \tag{23}$$

and

$$D_s^{\alpha}\, \boldsymbol{\psi} = D_s^{\alpha} \int_{\boldsymbol{r}\in\Omega(\boldsymbol{\alpha})} \boldsymbol{\mathcal{P}}(\boldsymbol{r}, \boldsymbol{\alpha}, \boldsymbol{\omega})\, d\Omega, \quad \text{being } D_s^{\alpha}\, \Box = \frac{\partial \Box}{\partial \boldsymbol{\alpha}}\, D_{\mathrm{s}}\boldsymbol{\alpha}. \tag{24}$$

Terms type (23) can be considered immediate, since integration domain $\Omega(\boldsymbol{\alpha})$ does not vary with $\boldsymbol{\omega}$. Thus,

$$\left[\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\omega}}\right] = \int_{\boldsymbol{r}\in\Omega(\boldsymbol{\alpha})} \frac{\partial}{\partial \boldsymbol{\omega}}\, \boldsymbol{\mathcal{P}}(\boldsymbol{r}, \boldsymbol{\alpha}, \boldsymbol{\omega})\, d\Omega \tag{25}$$

On the other hand, terms type (24) can not be considered immediate, since integration domain $\Omega(\boldsymbol{\alpha})$ does vary with $\boldsymbol{\omega}$. This is the reason why a traditional distinction is made between sizing optimization (when $\Omega(\boldsymbol{\alpha})$ is constant) and shape optimization (when $\Omega(\boldsymbol{\alpha})$ is variable). However, it can be proved [3] that

$$D_s^{\alpha}\, \boldsymbol{\psi} = \int_{\boldsymbol{r}\in\Omega(\boldsymbol{\alpha})} \mathcal{D}_{\mathrm{s}}\, \boldsymbol{\mathcal{P}}(\boldsymbol{r}, \boldsymbol{\alpha}, \boldsymbol{\omega})\, d\Omega \tag{26}$$

where,

$$\mathcal{D}_{\mathrm{s}}\, \Box = D_s^{\alpha}\, \Box + \Box\, \frac{1}{2}\, \mathrm{Tr}\left[\boldsymbol{G}^{-1}\, D_s\, \boldsymbol{G}\right], \tag{27}$$

being $\boldsymbol{G}$ the so-called metric tensor of the IGA mapping type (21). The details on how to compute and manipulate the metric tensor can be found in [3].

We notice that expressions (25) and (26) are analogous to the previous expression (22). Therefore, the same integration techniques can be used for the three of them. Furthermore, the corresponding high order sensitivity terms can be generated by recurrence.

7 Application example

In this section we will present some results obtained for a concrete roof spanning over a square room and built-in at its four vertices. Because of symmetry, only one quarter of the structure is analyzed. The complete specifications of the structure are given in [3]. In particular, the height of the mid-surface at the keystone is 1:839920 m. The analysis is performed by means of a linear elastic three-dimensional isogeometric model with uniform open knot vectors, B-Spline interpolating functions (p = 2) null displacements at the built-in supports, symmetry conditions where applicable, and integration by means of 3-point Gauss quadratures.

The geometry of the roof is defined by a total of 27 control points (see left side of Fig. 1). An h-refined mesh is automatically generated for the analysis by knot insertion, giving a total of $32 \times 32 \times 4 = 4,096$ control points (nodes) and $30 \times 30 \times 2 = 1,800$ knot spans (elements). We notice that this IGA model reproduces the exact geometry of the roof (as defined in [3]) for all and any level of h-refinement being applied.

Fig. 1 (right side) shows how the 1st principal stress in the vicinity of control point # 1 changes when the design in modified. In this figure we compare the IGA computed results with the IGA 3rd order predicted values. The former were obtained by recalculating the structure each time. The latter were obtained by approximating the structural response in terms of a Taylor expansion, taking into account the information given by a 3rd order directional sensitivity analysis of the unmodified design.

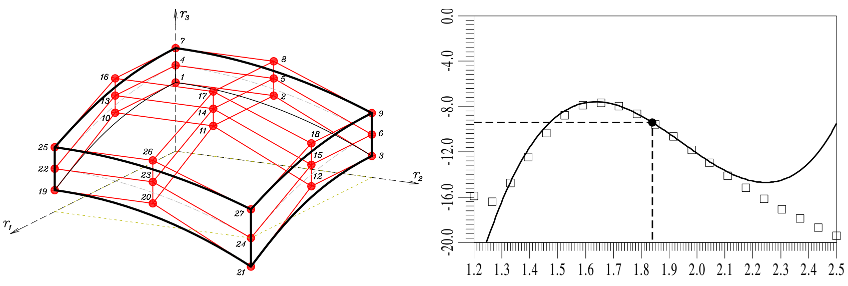

Figure 1: Left: IGA model. Right: 1st principal stress [Pa] at a given point for different heights [m] of the mid-surface at the keystone; 3rd order predicted (line) vs. computed results (squares).

8 Conclusions

A unified approach for the high order sensitivity analysis of IGA formulations has been presented. The technique is based on the IGA mapping that links material and reference coordinates. High order directional sensitivity derivatives are given by a single, unified and compact expression. The 1st order sensitivity analysis is cheap in terms of computing time, and may contribute essential information for decision-making. The 2nd order sensitivity analysis is relatively cheap in terms of computing time, and may contribute important information for improving algorithms. Higher order information is more expensive in terms of computing time, although it contributes to improve the quality of the approximations being used. The scope of future versions of IGA-based codes will be gratefully expanded if support for high order directional sensitivity analysis is provided as a standard feature.

Acknowledgements

This research work has been partially supported by FEDER funds of the European Union, by Grants # DPI2009-14546-C02-01 and # DPI2010-16496 of the *"Ministerio de Ciencia e Innovación"* of the Spanish Government, by Grants #CEOU 2007/09, # PGDIT09MDS00718PR and # PGDIT09REM005118PR of the *"Subdirección Xeral de I+D"* of the *"Xunta de Galicia"* and by research fellowships of *"Universidad de A Coruña"* and *"Fundación de la Ingeniería Civil de Galicia"*.

References

[1] Navarrina F. and Casteleiro M., A General Methodologycal Analysis for Optimum Design, *International Journal for Numerical Methods in Engineering*, **31(1)**, pp. 85–111, 1991.
[2] Haug J.E., Choi K.K. and Komkov V., *Design Sensitivity Analysis of Structural Systems*, Academic Press: Orlando, 1986.
[3] Navarrina F., López S., Colominas I., Bendito E. and Casteleiro M., High Order Shape Design Sensitivity: A Unified Approach, *Computer Methods in Applied Mechanics and Engineering*, **188(4)**, pp. 681–696, 2000.
[4] Choi K.K. and Kim N.H., *Structural Sensitivity Analysis and Optimization (Vol. 1 & 2)*, Springer Mechanical Engineering Series: New York, 2005.
[5] Hughes T.J.R., Cottrell J.A. and Bazilevs Y., Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, *Computer Methods in Applied Mechanics and Engineering*, **194**, pp. 4135–4195, 2005.
[6] Rogers D.F., *An Introduction to NURBS*, Academic Press: San Diego, 2001.
[7] Cottrell J.A., Hughes T.J.R. and Bazilevs Y., *ISOGEOMETRIC ANALYSIS. Towards Integration of CAD and FEA*, John Wiley & Sons Ltd.: Chichester, 2009.
[8] Piegl L., Tiller W., *The NURBS Book*, Monographs in Visual Communication, Springer-Verlag: Berlin, 1997.