Sensitivity Analysis Based Automation of Computational Problems

Jože Korelc and Teja Melink

Abstract The paper describes automation of primal and sensitivity analysis of computational models formulated and solved by the finite element method. Based on the symbolic system *AceGen* (http://symech.fgg.uni-lj.si/), fast and reliable code can be created with minimum effort and immediately tested and verified by using the associated finite element program *AceFEM* . Automation of first- and second-order sensitivity analysis with respect to an arbitrary parameter is presented. In an example, it is shown how sensitivity analysis has become an indispensable part of modern computational algorithms.

1 Introduction

Contemporary finite element software is mostly handwritten and based on formulations that were derived by scientists and software engineers. The related process is slow and can take more than several weeks to derive for a new finite element. Derivations of complex tensor fields to obtain residuals and tangent matrices are also prone to errors. To reduce the effort of developing the related new source code, symbolic code generation has been developed over the past decade. It is in a stage where the automatically generated source code is as small as the handwritten code, it is efficient and reliable. In this paper, a general approach is described that can be applied to many different applications in engineering and science. The main advantage of using symbolic code development is that the development time, especially for complex materials or elements, reduces by orders of magnitude. The paper will mainly focus on solid and structural mechanics problems. However, the general potential of the automatic code generation goes far beyond these engineering applications.

Modern finite element simulations are often coupled with optimization procedures that require additionally to the solution of primal problem also the solution of

J. Korelc (✉) · T. Melink
Faculty of Civil and Geodetic Engineering, University of Ljubljana,
Jamova 2, 1000 Ljubljana, Slovenia
e-mail: jkorelc@fgg.uni-lj.si

© CISM International Centre for Mechanical Sciences 2020
J. Schröder and P. de Mattos Pimenta (eds.), *Novel Finite Element Technologies for Solids and Structures*, CISM International Centre for Mechanical Sciences 597,
https://doi.org/10.1007/978-3-030-33520-5_2

sensitivity problem. The aim of the sensitivity analysis is to calculate derivatives of an arbitrary response functional with respect to chosen parameters (see e.g., Kleiber et al. (1997), Keulen et al. (2005), Choi and Kim (2005a), or Choi and Kim (2005b)). Thus, any proposed method of automation should address automation of primal as well as sensitivity analysis. The response functional can depend on arbitrary analysis model inputs (material constants, load intensity and distribution, shape parameters, etc.) as well as on arbitrary intermediate or final results of the analysis (solution vectors, derived quantities such as stress tensor, integrated quantities such as damage, etc.). The complete automation of the sensitivity analysis is thus possible only if the automatic differentiation technology is applied on the complete simulation code. This is not possible for general finite element environments. Thus, a finite difference approximation of sensitivities is used for practical applications. However, a large variety of practical problems can still be solved by the classical finite element procedure, where all problem-dependent quantities are evaluated on the individual element level and then assembled on the global level. The established algorithm is then applied on the global level to obtain the derivatives of the response functional. A comprehensive overview of the possible approaches can be found in Keulen et al. (2005). In this case we can, with the use of methods of automation, obtain analytically exact sensitivities. The use of analytically exact sensitivity analysis can significantly improve optimization procedures Choi and Kim (2005b), Kristanic and Korelc (2008), multi-scale algorithms Solinc and Korelc (2015), Korelc and Zupan (2018) and implementation of nonlinear material models Korelc and Stupkiewicz (2014), Hudobivnik and Korelc (2016).

The paper will follow the automation procedure of an analytically exact first- and second-order sensitivity analysis. In the first chapter, the necessary tools will be described that can be used to automatically derive problem-dependent quantities at the individual element level. In the second chapter, the global sensitivity problem will be formulated and solved. The third chapter introduces a set of examples that demonstrate how sensitivity analysis can be used to improve modern computational algorithms.

2 Automatic Code Generation with AceGen

The problem of automation of computational methods has been explored by researches from the fields of mathematics, computer science, and computational mechanics, resulting in a variety of approaches (e.g., the hybrid object-oriented approach by Eyheramendy and Zimmermann (2000), Logg et al. (2012) and the hybrid symbolic-numeric approach by Korelc and Wriggers (2016)) and available software tools (e.g., computer algebra systems, AD tools by Griewank (2000), problem-solving environments, and numerical libraries). Automation can address all steps of the finite element solution procedure from the strong form of a boundary-value problem to the visualization of results, or it can be applied only to the automation of the selected steps of the whole procedure.

2.1 Hybrid Symbolic-Numerical System AceGen

Automation of primal and sensitivity analysis is *AceGen* (http://symech.fgg.uni-lj.si/) achieved through the hybrid symbolic-numeric approach to automation of finite element method that combines symbolic and algebraic capabilities of a general computer algebra system, e.g., *Mathematica* (www.wolfram.com), an automatic differentiation technique (AD) and an automatic code generation with the general-purpose finite element environment. The structure of the hybrid symbolic-numerical system *AceGen* for multi-language and multi-environment code generation introduced by Korelc (2002) is presented in Fig. 1.

General characteristics of *AceGen* code generator are the following:

- simultaneous optimization of expressions immediately after they have been derived,
- automatic differentiation technique,
- automatic selection of the appropriate intermediate variables,
- the whole program structure can be generated,

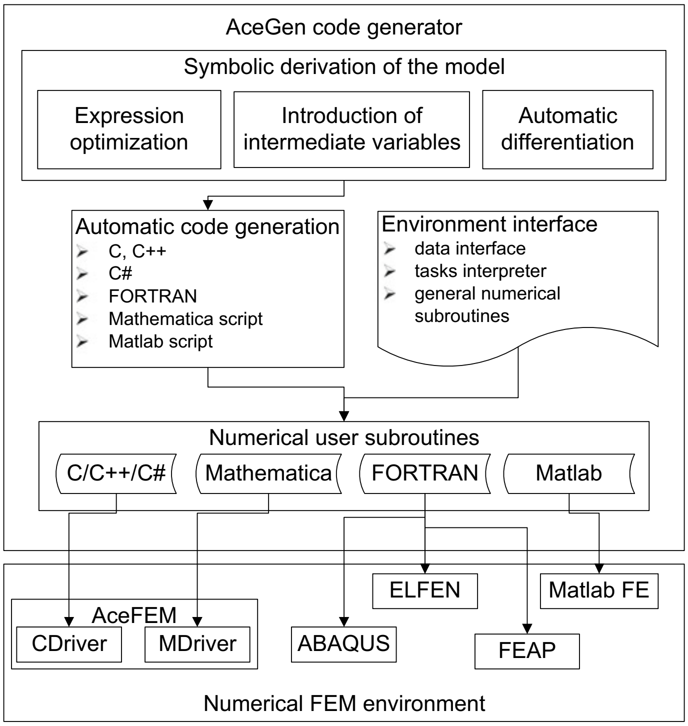


Fig. 1 Hybrid symbolic-numeric approach to automation of finite element method

- appropriate for large problems where also intermediate expressions can be subjected to uncontrolled swell,
- global expression optimization procedures with stochastic evaluation of expressions,
- differentiation with respect to indexed variables,
- automatic interface to other numerical environments,
- multi-language code generation (*Fortran*/*Fortran90*, C/C++, *Mathematica* language, *Matlab* language),
- advanced methods for exploring and debugging generated formulae.

The *AceGen* system is written in the symbolic language of *Mathematica*. A detailed description of the system can be found in Korelc and Wriggers (2016).

2.2 Simultaneous Simplification Procedure

Typical *AceGen* function takes the expression provided by the user, either interactively or in file, and returns an optimized version of the expression. Optimized version of the expression can result in a newly created auxiliary symbol, or in an original expression in parts replaced by previously created auxiliary symbols. In the first case, *AceGen* stores the new expression in an internal database. The procedure is presented in Fig. 2.

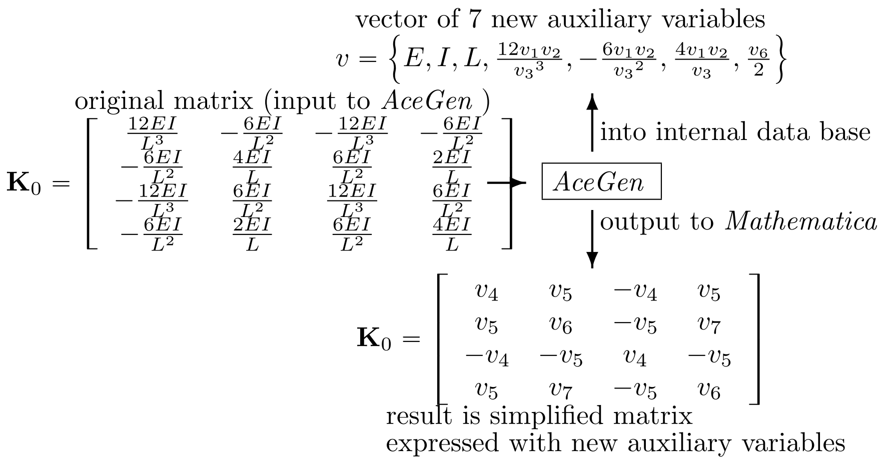

vector of 7 new auxiliary variables
$v = \left\{E, I, L, \frac{12 v_1 v_2}{v_3{}^3}, -\frac{6 v_1 v_2}{v_3{}^2}, \frac{4 v_1 v_2}{v_3}, \frac{v_6}{2}\right\}$
original matrix (input to AceGen)
$\mathbf{K}_0 = \begin{bmatrix} \frac{12EI}{L^3} & -\frac{6EI}{L^2} & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ -\frac{6EI}{L^2} & \frac{4EI}{L} & \frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{12EI}{L^3} & \frac{6EI}{L^2} & \frac{12EI}{L^3} & \frac{6EI}{L^2} \\ -\frac{6EI}{L^2} & \frac{2EI}{L} & \frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix}$
into internal data base
AceGen
output to Mathematica
$\mathbf{K}_0 = \begin{bmatrix} v_4 & v_5 & -v_4 & v_5 \\ v_5 & v_6 & -v_5 & v_7 \\ -v_4 & -v_5 & v_4 & -v_5 \\ v_5 & v_7 & -v_5 & v_6 \end{bmatrix}$
result is simplified matrix
expressed with new auxiliary variables

Fig. 2 Simultaneous simplification procedure

```
<< AceGen`;
SMSInitialize["DetJ", "Language" -> "C"];
SMSModule["DetJ", Real[X$$[2, 4], k$$, e$$, J$$]];
{ξ, η} ⊢ SMSReal[{k$$, e$$}];
{𝕏c, 𝕐c} ⊢ SMSReal[Array[X$$, {2, 4}]];
ℕh ⊨ {(1 - ξ) (1 - η), (1 + ξ) (1 - η), (1 + ξ) (1 + η), (1 - ξ) (1 + η)} / 4;
Jg ⊨ SMSD[{ℕh.𝕏c, ℕh.𝕐c}, {ξ, η}];
SMSExport[Det[Jg], J$$];
SMSWrite[];
```

Fig. 3 Typical *AceGen* input

2.3 *Typical Example of Automatic Code Generation with AceGen*

To illustrate the standard *AceGen* procedure, a simple example is considered. A typical numerical subprogram that returns a determinant of the JACOBI matrix of nonlinear transformation from the reference to initial configuration for quadrilateral element topology is derived. The syntax of the *AceGen* script language is the same as the syntax of the *Mathematica* script language with some additional functions. The input for *AceGen* is presented in Fig. 3. It can be divided into six characteristic steps:

- At the beginning of the session, the `SMSInitialize` function initializes the system.
- The `SMSModule` function defines the input and output parameters of the subroutine `DetJ`.
- The `SMSReal` function assigns the input parameters `X$$` and `k$$` and `e$$` of the subroutine to the standard *Mathematica* symbols. Double `$` character indicates that the symbol is an input or output parameter of the generated subroutine.
- During the description of the problem, special operators ($\vdash$, $\dashv$, $\models$) are used to perform the simultaneous optimization of expressions and the creation of new intermediate variables. The `SMSD` function performs an automatic differentiation of one or several expressions with respect to the arbitrary variable or the vector of variables by simultaneously enhancing the already derived code.
- The results of the derivation are assigned to the output parameter `J$$` of the subroutine by the `SMSExport` function.
- At the end of the session, the `SMSWrite` function writes the contents of the vector of the generated formulae to the file in a prescribed language format.

The generated subroutine in C language is presented in Fig. 4 and in FORTRAN language in Fig. 5.

```
/***************** S U B R O U T I N E *******************/
void DetJ(double v[5001],double X[2][4],double (*k),double (*e),double (*J))
{
v[20]=(-1e0+(*k))/4e0;
v[21]=(-1e0-(*k))/4e0;
v[22]=(1e0+(*e))/4e0;
v[19]=(-1e0+(*e))/4e0;
(*J)=(v[19]*(X[0][0]-X[0][1])+v[22]*(X[0][2]-X[0][3]))*
(v[21]*(X[1][1]-X[1][2])+v[20]*(X[1][0]
 -X[1][3]))-(v[21]*(X[0][1]-X[0][2])+v[20]*(X[0][0]-X[0][3]))*
 (v[19]*(X[1][0]-X[1][1])+v[22]*
 (X[1][2]-X[1][3]));
};
```

Fig. 4 Typical automatically generated subroutine in C language

```
      SUBROUTINE DetJ(v,X,k,e,J)
      IMPLICIT NONE
      include 'sms.h'
      DOUBLE PRECISION v(5001),X(2,4),k,e,J
      v(20)=((-1d0)+k)/4d0
      v(21)=((-1d0)-k)/4d0
      v(22)=(1d0+e)/4d0
      v(19)=((-1d0)+e)/4d0
      J=(v(19)*(X(1,1)-X(1,2))+v(22)*(X(1,3)-X(1,4)))*(v(21)*(X(2,2)
     &-X(2,3))+v(20)*(X(2,1)-X(2,4)))-(v(21)*(X(1,2)-X(1,3))+v(20)*(X
     &(1,1)-X(1,4)))*(v(19)*(X(2,1)-X(2,2))+v(22)*(X(2,3)-X(2,4)))
      END
```

Fig. 5 Typical automatically generated subroutine in FORTRAN language

2.4 Automatic Differentiation

Differentiation is the most important symbolic operation needed within the algorithmic treatment of the solution process for the nonlinear boundary-value problems. This is, for example, the case for finite element methods, where NEWTON- - RAPHSON algorithms are employed to solve the nonlinear algebraic equation systems. The automatic differentiation (AD) method is used in *AceGen* for the evaluation of the exact derivatives of any arbitrary complex function via chain rule and represents an alternative solution to the numerical differentiation and symbolic differentiation. Automatic differentiation techniques are based on the fact that every computer program executes a sequence of elementary operations with known derivatives, thus allowing the evaluation of exact derivatives via the chain rule for an arbitrary complex formulation. If one has a computer code, which allows to evaluate a function f and needs to compute the gradient ∇f of f with respect to arbitrary variables, then the automatic differentiation tools, see e.g., Griewank (2000), can be applied to generate the appropriate program code.

There are two approaches for the automatic differentiation of a computer program, often recalled as the forward and the backward mode of automatic differentiation. The procedure is illustrated on a simple example of function f defined by

$$f = b\,c \quad \text{with} \quad b = \sum_{i=1}^{n} a_i^2 \quad \text{and} \quad c = \mathrm{Sin}(b) \tag{1}$$

where $a_1, a_2, \ldots, a_n$ are n independent variables. The forward mode accumulates the derivatives of intermediate variables with respect to the independent variables as follows:

$$\begin{aligned}
\nabla b &= \left\{\tfrac{db}{da_i}\right\} = \{2\,a_i\} && i = 1, 2, \ldots, n\\
\nabla c &= \left\{\tfrac{dc}{da_i}\right\} = \{\mathrm{Cos}(b)\,\nabla b_i\} && i = 1, 2, \ldots, n\\
\nabla f &= \left\{\tfrac{df}{da_i}\right\} = \{\,\nabla b_i\,c + b\,\nabla c_i\,\} && i = 1, 2, \ldots, n
\end{aligned} \tag{2}$$

In contrast to the forward mode, the backward mode propagates adjoin $\bar{x} = \frac{\partial f}{\partial x}$, which are the derivatives of the final values, with respect to intermediate variables:

$$\begin{aligned}
&\bar{f} = \tfrac{df}{df} = 1 && 1\\
&\bar{c} = \tfrac{df}{dc} = \tfrac{\partial f}{\partial c}\bar{f} = b\,\bar{f} && 1\\
&\bar{b} = \tfrac{df}{db} = \tfrac{\partial f}{\partial b}\bar{f} + \tfrac{\partial c}{\partial b}\bar{c} = c\,\bar{f} + \mathrm{Cos}(b)\,\bar{c} && 1\\
&\nabla f = \{\bar{a}_i\} = \left\{\tfrac{\partial b}{\partial a_i}\,\bar{b}\right\} = \left\{2\,a_i\,\bar{b}\right\} && i = 1, 2, \ldots, n.
\end{aligned} \tag{3}$$

Although obviously numerically superior when the number of functions is small, the backward mode requires potential storage of a large amount of intermediate data during the evaluation of the function that can be as high as the number of numerical operations performed. Additionally, a complete reversal of the program flow is required. This is because the intermediate variables are used in reverse order when related to their computation. For the efficient automation of the FE method, it is desirable that both approaches are available and that the software tool used for the automation can automatically select the most efficient approach for a given task. There exist many strategies how the AD procedure can be implemented, see e.g., Bischof et al. (2002). The simplest approach is to use operator overloading and during the evaluation of function f create a trace of all numerical operations and their arguments, later used to evaluate gradient in forward or backward mode. More efficient is source-to-source transformation strategy that transforms the source code for computing a function into the source code for computing the derivatives of the function.

The result of the AD procedure is called "computational derivative" and is written as $\frac{\hat{\delta} f(\mathbf{a})}{\hat{\delta}\mathbf{a}}$. The AD operator $\frac{\hat{\delta} f(\mathbf{a})}{\hat{\delta}\mathbf{a}}$ represents partial differentiation of a function $f(\mathbf{a})$ with respect to variables $\mathbf{a}$. If, for example, alternative or additional dependencies for a set of intermediate variables $\mathbf{b}$ have to be considered for differentiation, then the AD exception is indicated by the following formalism:

$$\left.\frac{\hat{\delta} f(\mathbf{a}, \mathbf{b})}{\hat{\delta} \mathbf{a}}\right|_{\frac{D\mathbf{b}}{D\mathbf{a}}=\mathbf{M}}, \tag{4}$$

which indicates that during the AD procedure, the total derivatives of variables **b** with respect to variables **a** are set to be equal to matrix **M**. The automatic differentiation exceptions are the basis for the automatic differentiation or ADB formulation of computational problem. The ADB notation can be directly translated to the AceFEM code and is part of numerically efficient code automation. Details of the method and of the corresponding software AceGen can be found in Korelc (1997), Korelc (2009) and Korelc (2018).

2.5 *Automatic Differentiation and Finite Element Method*

Large finite element environment usually employs a large variety of finite elements, solution procedures, and they commonly use commercial numerical libraries for which the source codes are not readily available. In such a case, it would be difficult to directly apply the AD tools to get, for example, the global stiffness matrix of a large-scale problem. However, the AD technology can still be used for the evaluation of specific quantities that appear as a part of FE simulation. For example, one can use AD at the individual element level to evaluate element-specific quantities such as

- strain and stress tensors,
- nonlinear coordinate transformations,
- consistent tangent stiffness matrix,
- residual vector and
- sensitivity pseudo-load vectors.

3 Sensitivity Analysis

The procedures for the formulation and solution of primal and sensitivity problem for an arbitrary coupled path-dependent problem are presented in detail in Korelc (2009). Here, a summary of the primal and sensitivity analysis of hyper-elastic problems is given. Let us define a primal problem with the residual equation $\mathbf{R}(\mathbf{p}) = \mathbf{0}$, where $\mathbf{p}$ represents a set of nodal unknowns of the problem. The primal problem is solved by the standard Newton–Raphson iterative procedure. For sensitivity analysis, we define the residual and the vector of unknowns as a function of a vector of design parameters $\boldsymbol{\phi} = \{\phi_1, \ldots, \phi_n\}$ as

$$\mathbf{R}(\mathbf{p}(\boldsymbol{\phi}), \boldsymbol{\phi}) = \mathbf{0}. \tag{5}$$

The sensitivity problem can be obtained from the primal problem by differentiating (5) with respect to design parameter ϕ_I. Equation (6) represents a system of linear equations for the unknown sensitivities of the primal unknowns of the problem $\frac{D\mathbf{p}}{D\phi_I}$ (8). The right-hand side (7) is called "first-order sensitivity pseudo- load vector".

$$\frac{\partial \mathbf{R}}{\partial \mathbf{p}}\frac{D\mathbf{p}}{D\phi_I}+\frac{\partial \mathbf{R}}{\partial \phi_I}=\mathbf{0} \tag{6}$$

$${}^{I}\tilde{\mathbf{R}}=-\frac{\partial \mathbf{R}}{\partial \phi_I} \tag{7}$$

$$\mathbf{K}\frac{D\mathbf{p}}{D\phi_I}=-{}^{I}\tilde{\mathbf{R}} \tag{8}$$

The sensitivity problem that is solved after the convergence of the primal problem has been reached. The second-order sensitivity problem is obtained from the first-order problem by differentiating (6) with respect to design parameter ϕ_J. It results in

$$\frac{\partial^2 \mathbf{R}}{\partial \mathbf{p}^2}\frac{D\mathbf{p}}{D\phi_I}\frac{D\mathbf{p}}{D\phi_J}+\frac{\partial^2 \mathbf{R}}{\partial \mathbf{p}\partial \phi_J}\frac{D\mathbf{p}}{D\phi_I}+\frac{\partial^2 \mathbf{R}}{\partial \mathbf{p}\partial \phi_I}\frac{D\mathbf{p}}{D\phi_J}+\frac{\partial \mathbf{R}}{\partial \mathbf{p}}\frac{D^2\mathbf{p}}{D\phi_I D\phi_J}+\frac{\partial^2 \mathbf{R}}{\partial \phi_I\partial \phi_J}=\mathbf{0} \tag{9}$$

$$\mathbf{K}\frac{D^2\mathbf{p}}{D\phi_I D\phi_J}=-{}^{IJ}\tilde{\mathbf{R}} \tag{10}$$

where $\frac{D^2\mathbf{p}}{D\phi_I D\phi_J}$ are second-order sensitivities and ${}^{IJ}\tilde{\mathbf{R}}$ represents the "second- order sensitivity pseudo-load vector" (11).

$${}^{IJ}\tilde{\mathbf{R}}=\frac{\partial^2 \mathbf{R}}{\partial \mathbf{p}^2}\frac{D\mathbf{p}}{D\phi_I}\frac{D\mathbf{p}}{D\phi_J}+\frac{\partial^2 \mathbf{R}}{\partial \mathbf{p}\partial \phi_J}\frac{D\mathbf{p}}{D\phi_I}+\frac{\partial^2 \mathbf{R}}{\partial \mathbf{p}\partial \phi_I}\frac{D\mathbf{p}}{D\phi_J}+\frac{\partial^2 \mathbf{R}}{\partial \phi_I\partial \phi_J} \tag{11}$$

The global pseudo-load vectors ${}^{I}\tilde{\mathbf{R}}$ and ${}^{IJ}\tilde{\mathbf{R}}$ are obtained by the standard integration over the element domain and the standard finite element assembly procedure of element contributions to global vectors

$${}^{I}\tilde{\mathbf{R}}=\mathop{\mathsf{A}}_{e=1}^{n_e}\sum_{g=1}^{n_g} w_g\,{}^{I}\tilde{\mathbf{R}}_g,\quad {}^{IJ}\tilde{\mathbf{R}}=\mathop{\mathsf{A}}_{e=1}^{n_e}\sum_{g=1}^{n_g} w_g\,{}^{IJ}\tilde{\mathbf{R}}_g \tag{12}$$

where ${}^{I}\tilde{\mathbf{R}}_g$ and ${}^{IJ}\tilde{\mathbf{R}}_g$ represent integration point contributions to the element pseudo-load vectors and consequently to the global pseudo-load vectors and w_g is an integration point weight. The only part of the whole procedure that depends on specific element formulation is the evaluation of the integration point pseudo-load vectors. Consequently, for the automation of the complete sensitivity analysis procedure we only need a method for automatic derivation of integration point pseudo-load vectors ${}^{I}\tilde{\mathbf{R}}_g$ and ${}^{IJ}\tilde{\mathbf{R}}_g$. For an arbitrary finite element formulation, this can be

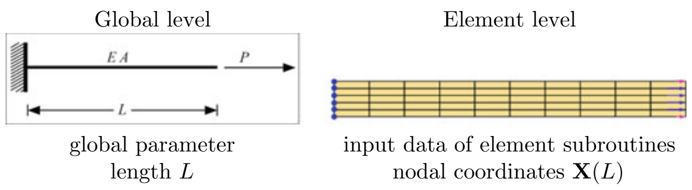

Global level
Element level
EA
P
L
global parameter length L
input data of element subroutines nodal coordinates $\mathbf{X}(L)$

Fig. 6 Parametrization of input data of continuum and discretized problem

achieved with automatic differentiation and code optimization as described in Korelc (2009).

The obvious problem in obtaining the right-hand sides ${}^{I}\tilde{\mathbf{R}}$ and ${}^{IJ}\tilde{\mathbf{R}}$ is that an arbitrary sensitivity parameter (e.g., length of the beam in Fig. 6) does not appear explicitly as an input parameter of the finite element solution procedure, either at the global level or at the level of user subroutines. The missing dependency between an arbitrary sensitivity parameter and the finite element code is defined by "design velocity field" (Korelc and Wriggers (2016)).

3.1 Design Velocity Field

For example, let us consider shape parameter L of the beam depicted in Fig. 6 as sensitivity parameter. The relation between shape parameter L and the coordinates of an arbitrary node $\mathbf{X}^J(L)$ can be an arbitrary complex function that, in general, cannot be input data of the finite element analysis. However, it is not the relation $\mathbf{X}^J(L)$ itself that is needed within the sensitivity analysis to obtain ${}^{I}\tilde{\mathbf{R}}$ and ${}^{IJ}\tilde{\mathbf{R}}$, but its first and second derivatives. The input data for the sensitivity analysis are thus the rate of change of nodal coordinates with the change of sensitivity parameter L. The rate of change of X_1 coordinate in all nodes represents the nodal values of a scalar field $\frac{DX_1}{DL}$. The $\frac{DX_1}{DL}$ field is traditionally called the design velocity field. The discretized design velocity field $\frac{D\mathbf{X}_1}{DL}$ is evaluated for the numeric values of the design sensitivity parameter L in all nodes and is the appropriate input data for sensitivity analysis related finite element subroutines.

Evaluation of sensitivity pseudo-load vectors ${}^{I}\tilde{\mathbf{R}}$ and ${}^{IJ}\tilde{\mathbf{R}}$ for the first- and second-order sensitivity analysis of the above example then follows as

$$ {}^{I}\tilde{\mathbf{R}}_g = \frac{\partial \mathbf{R}_g}{\partial \mathbf{X}_1^e}\frac{D\mathbf{X}_1^e}{DL} \tag{13}$$

$$ {}^{IJ}\tilde{\mathbf{R}}_g = \frac{\partial^2 \mathbf{R}_g}{\partial \mathbf{p}_e{}^2}\left(\frac{D\mathbf{p}_e}{DL}\right)^2 + 2\frac{\partial^2 \mathbf{R}_g}{\partial \mathbf{p}_e \partial \mathbf{X}_1^e}\frac{D\mathbf{X}_1^e}{DL}\frac{D\mathbf{p}_e}{DL} + \frac{\partial^2 \mathbf{R}_g}{\partial \mathbf{X}_1^{e2}}\frac{D^2\mathbf{X}_1^e}{DL^2} \tag{14}$$

where $\mathbf{X}_1^e$ is a vector of X_1 coordinates of element nodes. For the automation, we also need automatic differentiation based version of formulas (13) and (14) or the ADB notation (see Korelc (2009)). For the ADB notation, the partial derivatives are replaced with computational derivatives and the AD exceptions are added for the indirect dependencies $\mathbf{X}_1(L)$, leading to

$$^{I}\tilde{\mathbf{R}}_g = \left.\frac{\hat{\delta}\mathbf{R}_g}{\hat{\delta}L}\right|_{\frac{D\mathbf{X}_1^e}{DL}=\mathbf{V}_L} \tag{15}$$

$$^{IJ}\tilde{\mathbf{R}}_g = \left.\frac{\hat{\delta}}{\hat{\delta}L}\left(\left.\frac{\hat{\delta}\mathbf{R}_g}{\hat{\delta}L}\right|_{\frac{D\mathbf{X}_1^e}{DL}=\mathbf{V}_L,\ \frac{D\mathbf{p}_e}{DL}=\mathbf{S}_L}\right)\right|_{\frac{D\mathbf{X}_1^e}{DL}=\mathbf{V}_L,\ \frac{D\mathbf{V}_L}{DL}=\mathbf{V}_{LL},\ \frac{D\mathbf{p}_e}{DL}=\mathbf{S}_L} \tag{16}$$

where matrices $\mathbf{V}_L = \frac{D\mathbf{X}_1^e}{DL}$ and $\mathbf{V}_{LL} = \frac{D^2\mathbf{X}_1^e}{DL^2}$ are simulation input data that represent the first- and second-order velocity fields. Components of matrix $\mathbf{S}_L = \frac{D\mathbf{p}_e}{DL}$ are zero for the DOF's with prescribed essential boundary conditions and are set to already calculated first-order sensitivities for the true DOF's. Consequently, all the first-order sensitivities have to be calculated first in order to be able to calculate the second-order sensitivities.

Shape sensitivity parameters (shape sensitivity analysis) Symbol L in (15) and (16) is a global quantity. Thus, it does not actually appear explicitly as a part of Gauss point residual $\mathbf{R}_g$. Consequently, in formulas (15) and (16), symbol L has no meaning and it can be replaced by any symbol. Let ϕ_I and ϕ_J be an arbitrary shape parameters and $\mathbf{X}_e$ nodal wise ordered nested set of all coordinates of all element nodes ($\mathbf{X}_e = \mathbf{X}_e(\phi_I, \phi_J)$). A general ADB notation of the first- and second-order shape sensitivity analysis then follows as

$$^{I}\tilde{\mathbf{R}}_g = \left.\frac{\hat{\delta}\mathbf{R}_g}{\hat{\delta}\phi_I}\right|_{\frac{D\mathbf{X}_e}{D\phi_I}=\mathbf{V}_I} \tag{17}$$

$$^{IJ}\tilde{\mathbf{R}}_g = \left.\frac{\hat{\delta}}{\hat{\delta}\phi_J}\left(\left.\frac{\hat{\delta}\mathbf{R}_g}{\hat{\delta}\phi_I}\right|_{\frac{D\mathbf{X}_e}{D\phi_I}=\mathbf{V}_I,\ \frac{D\mathbf{p}_e}{D\phi_I}=\mathbf{S}_I}\right)\right|_{\frac{D\mathbf{X}_e}{D\phi_J}=\mathbf{V}_J,\ \frac{D\mathbf{V}_I}{D\phi_J}=\mathbf{V}_{IJ},\ \frac{D\mathbf{p}_e}{D\phi_J}=\mathbf{S}_J} . \tag{18}$$

The sensitivity-dependent analysis input data in (17) and (18) are matrices $\mathbf{V}_I = D\mathbf{X}_e/D\phi_I$, $\mathbf{V}_J = D\mathbf{X}_e/D\phi_I$ and $\mathbf{V}_{IJ} = D^2\mathbf{X}_e/D\phi_I D\phi_J$ that represent the first- and second- order shape design velocity fields, and $\mathbf{S}_I = \frac{D\mathbf{p}_e}{D\phi_I}$, $\mathbf{S}_J = \frac{D\mathbf{p}_e}{D\phi_J}$ are already calculated first-order sensitivities of element DOF's.

3.2 Arbitrary Sensitivity Parameters

The formulation can be extended to arbitrary sensitivity parameters. In Fig. 7, the parametrization of a general continuum problem to be solved using the finite element model is presented. Additionally to the nodal coordinates, the input data of the typical finite element procedures are material parameters and boundary conditions. The goal of automation is to preserve the standard finite element technology paradigm, where all the physical problem dependent quantities are calculated at the individual finite element level and then assembled at the global level. For the purpose of automation, each analysis input data is considered as a field defined over the domain of the problem that depends on specific sensitivity parameters, as depicted in Fig. 7. Fields and the corresponding design velocity fields are classified according to their actual appearance (or lack of it) in the formulation of the finite element problem. FE analysis input data can be, for the purpose of automation of sensitivity analysis, classified into several classes:

1. parameter (material) input data with corresponding parameter sensitivity analysis and parameter design velocity fields (e.g., E^J and $\frac{DE^J}{DE_\sigma}$),
2. nodal spatial coordinates with corresponding shape sensitivity analysis and shape design velocity fields (e.g., X_1^J and $\frac{DX_1^J}{DL}$),

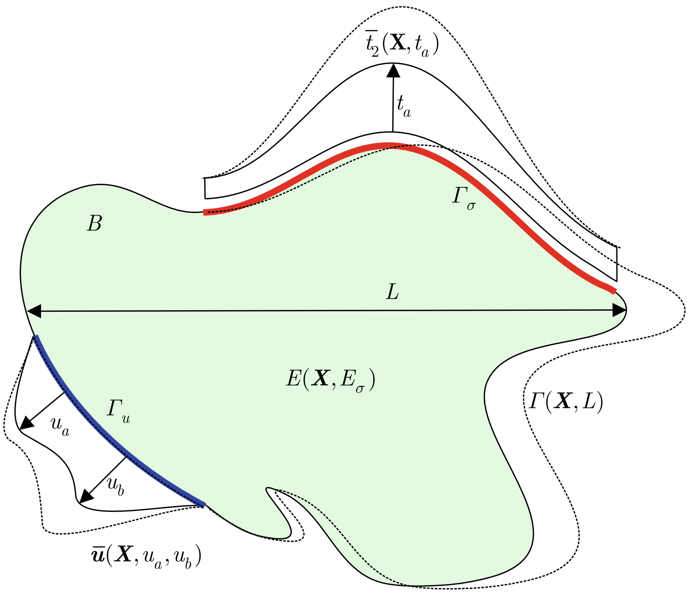

$\bar{t}_2(\mathbf{X}, t_a)$
t_a
Γ_σ
B
L
$E(\boldsymbol{X}, E_\sigma)$
Γ_u
$\Gamma(\boldsymbol{X}, L)$
u_a
u_b
$\bar{\boldsymbol{u}}(\boldsymbol{X}, u_a, u_b)$

Fig. 7 Parametrization of a general continuum problem to be solved using the finite element model

3. nodal essential boundary conditions with corresponding essential boundary condition sensitivity analysis and essential boundary condition velocity fields (e.g., $\bar{u}_1^J$ and $\frac{D\bar{u}_1^J}{Du_a}, \frac{D\bar{u}_1^J}{Du_b}$),
4. nodal natural boundary conditions with corresponding natural boundary condition sensitivity analysis and natural boundary condition velocity fields (e.g., P_2^J and $\frac{DP_2^J}{Dt_a}, \frac{DP_2^J}{DL}$).

Shape sensitivity analysis Shape sensitivity analysis is described in Sect. 3.1.

Essential boundary condition sensitivity analysis Essential boundary condition sensitivity parameters are used to parametrize the distribution of the essential boundary conditions at the boundary of the problem domain (e.g. u_a and u_b are used to parametrize $\bar{\mathbf{u}}$ in Fig. 7). Let ϕ_I and ϕ_J be arbitrary essential boundary condition sensitivity parameters and $\bar{\mathbf{p}}_e$ a set of element DOF with prescribed essential boundary condition, thus $\bar{\mathbf{p}}_e \subset \mathbf{p}_e$. The $\mathbf{p}_e$ set includes both degrees of freedom with prescribed essential boundary condition and true degrees of freedom, because they are at the element-level indistinguishable. The corresponding first- and second-order essential boundary condition velocity fields are defined by

$$\mathbf{V}_I = \begin{cases} \frac{D\bar{p}_{eJ}}{D\phi_I} & \text{if } p_{eJ} \in \bar{\mathbf{p}}_e \\ 0 & \text{if } p_{eJ} \in \mathbf{p}_e \backslash \bar{\mathbf{p}}_e \end{cases} : J = 1, \ldots, n_p, \tag{19}$$

$$\mathbf{V}_{IJ} = \begin{cases} \frac{D^2\bar{p}_{eJ}}{D\phi_I D\phi_J} & \text{if } p_{eJ} \in \bar{\mathbf{p}}_e \\ 0 & \text{if } p_{eJ} \in \mathbf{p}_e \backslash \bar{\mathbf{p}}_e \end{cases} : J = 1, \ldots, n_p \tag{20}$$

where n_p is the total number of element nodal DOF. Velocity field is zero for the true degrees of freedom. Thus, proper definition of velocity fields is sufficient to make the difference between the degrees of freedom with prescribed essential boundary condition and true degrees of freedom. A general ADB notation of Gauss point contribution to the first- and second-order essential boundary condition pseudo-load vectors then follows as

$$^{I}\tilde{\mathbf{R}}_g = \left.\frac{\hat{\delta}\mathbf{R}_g}{\hat{\delta}\phi_I}\right|_{\frac{D\mathbf{p}_e}{D\phi_I}=\mathbf{V}_I} \tag{21}$$

$$^{IJ}\tilde{\mathbf{R}}_g = \left.\frac{\hat{\delta}}{\hat{\delta}\phi_J}\left(\left.\frac{\hat{\delta}\mathbf{R}_g}{\hat{\delta}\phi_I}\right|_{\frac{D\mathbf{p}_e}{D\phi_I}=\mathbf{S}_I}\right)\right|_{\frac{D\mathbf{S}_I}{D\phi_J}=\mathbf{V}_{IJ},\, \frac{D\mathbf{p}_e}{D\phi_J}=\mathbf{S}_J} . \tag{22}$$

The sensitivity-dependent analysis input data in (21) and (22) are matrices $\mathbf{V}_I$ and $\mathbf{V}_{IJ}$. Matrices $\mathbf{S}_I$ and $\mathbf{S}_J$ are composed of the components of velocity fields for the DOF's with prescribed essential boundary conditions and already calculated first-order sensitivities for the true DOFs (23).

$$\mathbf{S}_I = \begin{cases} \frac{D\bar{p}_{eJ}}{D\phi_I} & \text{if } p_{eJ} \in \bar{\mathbf{p}}_e \\ \frac{Dp_{eJ}}{D\phi_I} & \text{if } p_{eJ} \in \mathbf{p}_e \backslash \bar{\mathbf{p}}_e \end{cases} : J = 1, \ldots, n_p \tag{23}$$

Material sensitivity parameters (parameter sensitivity analysis) Input parameters of the finite element procedures can be scalars (e.g., elastic modulus E), discretized scalar fields (e.g., nodal temperatures), and discretized vector fields (e.g., nodal spatial coordinates $\mathbf{X}^J$). Without losing the generality of the formulation, a scalar can be considered as a constant scalar field discretized by its constant nodal values and a vector field can be considered component-wise. Most of the input data of the finite element procedures are associated with nodes. However some quantities, such as material constants ($E_g = E_g(E_\sigma)$ and $\nu_g = \nu_0$), are associated with integration points. Again, the integration point based quantities can be obtained from the appropriate nodal-based quantities using standard finite element interpolation techniques. Consequently, integration point based quantities are also represented as a discretized scalar field unifying all sensitivity parameters within the same framework. Let ψ_e be a set of parameters on which element residual explicitly depends ($\mathbf{R}_g = \mathbf{R}_g(\psi_e)$). A general ADB notation of GAUSS point contribution to the first- and second-order parameter pseudo-load vectors then follows as

$$^I\tilde{\mathbf{R}}_g = \left.\frac{\hat{\delta}\mathbf{R}_g}{\hat{\delta}\phi_I}\right|_{\frac{D\psi_e}{D\phi_I}=\mathbf{V}_I} \tag{24}$$

$$^{IJ}\tilde{\mathbf{R}}_g = \left.\frac{\hat{\delta}}{\hat{\delta}\phi_J}\left(\left.\frac{\hat{\delta}\mathbf{R}_g}{\hat{\delta}\phi_I}\right|_{\frac{D\psi_e}{D\phi_I}=\mathbf{V}_I, \frac{D\mathbf{p}_e}{D\phi_I}=\mathbf{S}_I}\right)\right|_{\frac{D\psi_e}{D\phi_J}=\mathbf{V}_J, \frac{D\mathbf{V}_I}{D\phi_J}=\mathbf{V}_{IJ}, \frac{D\mathbf{p}_e}{D\phi_J}=\mathbf{S}_J} . \tag{25}$$

The sensitivity-dependent analysis input data in (24) and (25) are matrices $\mathbf{V}_I = D\psi_e/D\phi_I$, $\mathbf{V}_J = D\psi_e/D\phi_I$ and $\mathbf{V}_{IJ} = D^2\psi_e/D\phi_I D\phi_J$ that represent the first- and second-order parameter design velocity fields. $\mathbf{S}_I = \frac{D\mathbf{p}_e}{D\phi_I}$ and $\mathbf{S}_J = \frac{D\mathbf{p}_e}{D\phi_J}$ are the already calculated first-order sensitivities of element DOFs.

Natural boundary condition sensitivity parameters (natural boundary condition sensitivity analysis) Problems in solid mechanics and nonlinear structural mechanics, subjected to quasi-static proportional load, are frequently formulated as

$$\mathbf{R} = \mathbf{R}^{\text{int}} - \lambda\, \mathbf{R}^{\text{ref}} = \mathbf{0} \tag{26}$$

where $\mathbf{R}^{\text{int}}$ denotes the contribution of the internal forces to the global residual vector. Vector $\mathbf{R}^{\text{ref}}$ is the reference load vector associated with the pattern of the applied nodal forces (natural boundary condition input data) and λ is the loading parameter. Load vector $\lambda\, \mathbf{R}^{\text{ref}}$ is subtracted from the internal force vector and thus does not affect directly the residual vectors of the finite elements at local element level. Consequently, the contribution of variation of natural boundary conditions has to be formulated within the global solution algorithm and it does not follow the standard sensitivity analysis procedures as described in previous sections. If the contribution of the natural boundary conditions to the global residual $\mathbf{R}$ is accounted for by a special generalized finite elements then the natural boundary condition input data can be considered as a part of general input parameters $\boldsymbol{\psi}_e$ and treated accordingly.

The general equation (26) leads for an arbitrary time-dependent problem and for an arbitrary sensitivity parameter ϕ_I, ϕ_J to

$$\mathbf{R}^{\text{int}}(\mathbf{p}(\phi_I,\phi_J)) - \lambda\, \mathbf{R}^{\text{ref}}(\phi_I,\phi_J) = \mathbf{0}. \tag{27}$$

Direct differentiation of (27) with respect to ϕ_I yields the first-order pseudo-load vector and sensitivity of the response $\frac{D\mathbf{p}}{D\phi_I}$ by the solution of the linear equation systems (28).

$${}^{I}\tilde{\mathbf{R}} = -\lambda \frac{D\mathbf{R}^{\text{ref}}}{D\phi_I}, \quad \mathbf{K}\frac{D\mathbf{p}}{D\phi_I} = -{}^{I}\tilde{\mathbf{R}} \tag{28}$$

Second derivative of (27) yields

$${}^{IJ}\tilde{\mathbf{R}} = \frac{\partial^2\mathbf{R}}{\partial\mathbf{p}^2}\frac{D\mathbf{p}}{D\phi_I}\frac{D\mathbf{p}}{D\phi_J} + \frac{\partial^2\mathbf{R}}{\partial\mathbf{p}\partial\phi_J}\frac{D\mathbf{p}}{D\phi_I} + \frac{\partial^2\mathbf{R}}{\partial\mathbf{p}\partial\phi_I}\frac{D\mathbf{p}}{D\phi_J} - \lambda\,\frac{D^2\mathbf{R}^{\text{ref}}}{D\phi_I D\phi_J}. \tag{29}$$

Equation (29) has parts that depend on internal forces and a part that depends on reference load vector. Consequently, it has to be split into parts, one that is formed globally ${}^{IJ}\tilde{\mathbf{R}}^{\text{ref}}$ (30) and one that is formed by an element-based assembly procedure ${}^{IJ}\tilde{\mathbf{R}}^{\text{int}}$ (31).

$${}^{IJ}\tilde{\mathbf{R}}^{\text{ref}} = -\lambda\,\frac{D^2\mathbf{R}^{\text{ref}}}{D\phi_I D\phi_J}. \tag{30}$$

$${}^{IJ}\tilde{\mathbf{R}}^{\text{int}} = \frac{\partial^2\mathbf{R}}{\partial\mathbf{p}^2}\frac{D\mathbf{p}}{D\phi_I}\frac{D\mathbf{p}}{D\phi_J} + \frac{\partial^2\mathbf{R}}{\partial\mathbf{p}\partial\phi_J}\frac{D\mathbf{p}}{D\phi_I} + \frac{\partial^2\mathbf{R}}{\partial\mathbf{p}\partial\phi_I}\frac{D\mathbf{p}}{D\phi_J}. \tag{31}$$

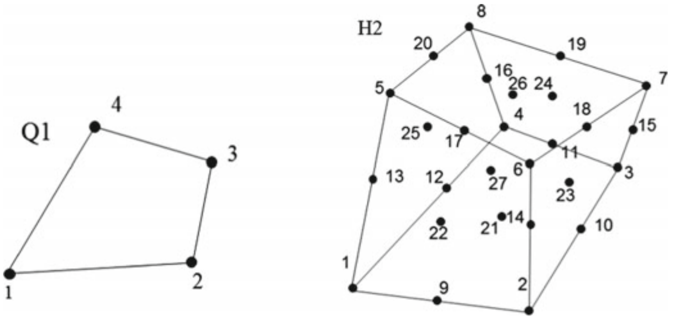
A general ADB notation of GAUSS point contribution to the ${}^{IJ}\tilde{\mathbf{R}}^{\text{int}}$ pseudo-load vector then follows as

$${}^{IJ}\tilde{\mathbf{R}}_g^{\text{int}} = \left.\frac{\hat{\delta}}{\hat{\delta}\phi_J}\left(\left.\frac{\hat{\delta}\mathbf{R}_g}{\hat{\delta}\phi_I}\right|_{\frac{D\mathbf{p}_e}{D\phi_I}=\mathbf{S}_I}\right)\right|_{\frac{D\mathbf{p}_e}{D\phi_J}=\mathbf{S}_J}. \tag{32}$$

At the end, the second-order sensitivity of the response $\frac{D^2\mathbf{p}}{D\phi_I D\phi_J}$ leads from the solution of the linear equation systems

$$\mathbf{K}\frac{D^2\mathbf{p}}{D\phi_I D\phi_J} = -({}^{IJ}\tilde{\mathbf{R}}^{\text{ref}} + {}^{IJ}\tilde{\mathbf{R}}^{\text{int}}). \tag{33}$$

Table 1 Comparison of code size and AceGen evaluation time



Q1 - linear elastic

Task	Code	*AceGen*
$\mathbf{K}_e \& \mathbf{R}_e$	5 kB	4 s
$+{}^I\tilde{\mathbf{R}}_e$	8 kB	5 s
$+{}^{IJ}\tilde{\mathbf{R}}_e$	9 kB	7 s

H2 - finite strain elasto-plastic

Task	Code	*AceGen*
$\mathbf{K}_e \& \mathbf{R}_e$	127 kB	57 s
$+{}^I\tilde{\mathbf{R}}_e$	270 kB	197 s
$+{}^{IJ}\tilde{\mathbf{R}}_e$	376 kB	377 s

3.3 Sensitivity Analysis—Code Complexity of AceGen Codes

The concept of design sensitivity velocity fields can be extended to general input parameters (e.g., nodal coordinates, material parameters, essential boundary conditions, and natural boundary conditions). For details see Korelc and Wriggers (2016). Any approach to automation is feasible only when the physical size of the generated codes stays within reasonable limits allowed by compilers and when the time to generate the code also stays within reasonable limits.

In Table 1, the code size and the *AceGen* evaluation time are compared for different finite element formulations. Two extreme cases are compared: simple two-dimensional linear elastic element and three-dimensional, finite strain, elastoplastic, 27-node brick element. For each required quantity (tangent and residual, the first-order sensitivity pseudo-load vector and the second-order sensitivity pseudo-load vector), the actual size of the code generated and the time used to generate the code are presented. We can see that also for the most complicated element the size of the code and the time to generate the code remain moderate.

4 Applications of Sensitivity Analysis

It is common for all applications of sensitivity analysis that once the element code that supports primal and sensitivity analysis for all input parameters of the individual finite elements is generated, then the only unanswered question remains "WHAT IS

THE VELOCITY FIELD OF THE PROBLEM?”. In this chapter, several examples are presented and the corresponding velocity fields are identified.

4.1 Sensitivity Analysis Based Stochastic Analysis

When an input parameter of the problem is random and it also randomly varies over the domain, it can be modeled as stochastic field. A stochastic field is defined with probability density function and covariance function. Probability density function specifies the probability of the random variable falling within a particular range of values. Covariance function describes how much a variable changes along the domain. In mechanical problems, most often used is exponential covariance function $C(\boldsymbol{X}_1, \boldsymbol{X}_2) = \sigma^2 e^{-\frac{||X_2 - X_1||}{l_c}}$, where $\boldsymbol{X}_i$ is a position vector over the physical domain, σ is standard deviation and l_c is correlation length. The bigger l_c is, the higher correlated is stochastic field (see e.g. Ghanem and Spanos (2003)).

The representation of the Gaussian stochastic field can be done with Karhunen–Loeve expansion, which is truncated after first M terms as

$$w(\boldsymbol{X}, \theta) = \bar{w}(\boldsymbol{X}) + \sum_{k=1}^{M} \sqrt{\lambda_k} f_k(\boldsymbol{X})\, \xi_k(\theta) \tag{34}$$

where $\boldsymbol{X}$ is a position vector over the physical domain, θ is an event of the space of random events, $\bar{w}(\boldsymbol{X})$ is expected value of the stochastic field and $\xi_k(\theta)$ are normalized uncorrelated Gaussian random variables with zero mean and unit variance. λ_k and $f_k(\boldsymbol{X})$ are the eigenvalues and eigenvectors, respectively, obtained as the solution of the homogeneus Fredholm integral equation ($\int_D C(\boldsymbol{X}_1, \boldsymbol{X}_2) f_k(\boldsymbol{X}_1)\, d\boldsymbol{X}_1 = \lambda_k f_k(\boldsymbol{X}_2)$) of the second kind with covariance function $C(\boldsymbol{X}_1, \boldsymbol{X}_2)$ as kernel. Galerkin procedure can be used to solve this equation numerically (see e.g., Melink and Korelc (2014)). The result is an approximated and discretized stochastic field according to (34).

When at least one of the input parameters is random, the response of the system is also random. The final goal of stochastic analysis is to calculate statistics (e.g., expected value and standard deviation) of the response. The response of the system is a function of a set of uncorrelated Gaussian random variables $\xi_k(\theta)$. In general, Monte Carlo method can be used to get statistics of the response for an arbitrary problem. However, Monte Carlo method requires a large number of direct simulations to be performed. An alternative approach is to use the second-order sensitivity analysis to the get second-order approximation of the response. In this case, only one direct simulation is needed.

In the presented stochastic approach, the response of the problem is approximated with a finite number of its Taylor series around the expected values of random variables (${}^0\boldsymbol{\xi} = \{{}^0\xi_1, {}^0\xi_2, \ldots {}^0\xi_M\}$), which resembles higher order sensitivity analysis.

In case of Gaussian stochastic field ($^0\boldsymbol{\xi} = \{0, 0, 0, \dots\}$) and second-order sensitivity analysis, we get

$$\mathbf{p}(\xi_1, \xi_2, \dots \xi_M) = \mathbf{p}(0, 0, 0, \dots) + \sum_{i=1}^{M} \frac{\partial \mathbf{p}}{\partial \xi_i} \xi_i + \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \frac{\partial^2 \mathbf{p}}{\partial \xi_i \partial \xi_j} \qquad (35)$$

where $\mathbf{p}$ is solution vector (in mechanics, $\mathbf{p}$ is usually vector of displacements). Derivatives of solution vector $\mathbf{p}$ with respect to random variables are calculated with sensitivity analysis. Thus, a set of sensitivity parameters of the problem is $\boldsymbol{\phi} = \boldsymbol{\xi}$. The approximation of the response is now closed-form polynomial formula. Thus, the statistics of the response (expected value and standard deviation) can be cheaply obtained either analytically or with the use of standard statistical functions in *Mathematica* . All we need to complete the derivation is the design velocity field of the problem.

A numerical example of bended clamped sinusoidal double skin cladding is chosen (see Fig. 8) to demonstrate the use of the above-described automation of the stochastic finite element method. The cladding is modeled by two-dimensional, four-node, finite strain elements. The shape of the cladding is sinusoidal with n wavelengths and constant thickness of the skin and foam. The amplitude of waves h is presumed to change stochastically along the X axis. Therefore, one-dimensional stochastic field $h(X, \boldsymbol{\xi})$ of the wave amplitude is considered. The Y coordinate of the central line nodes is then given by

$$Y(X, \boldsymbol{\xi}) = h(X, \boldsymbol{\xi}) \sin \frac{n\pi X}{L}, \qquad (36)$$

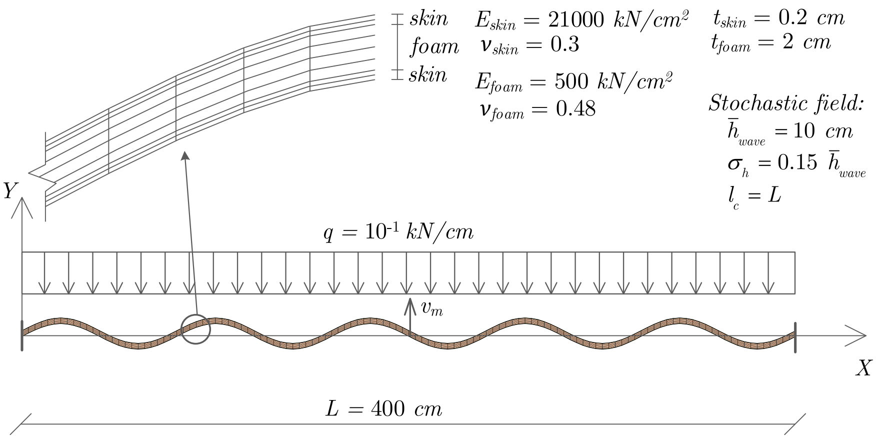


Fig. 8 Sinusoidal double skin cladding

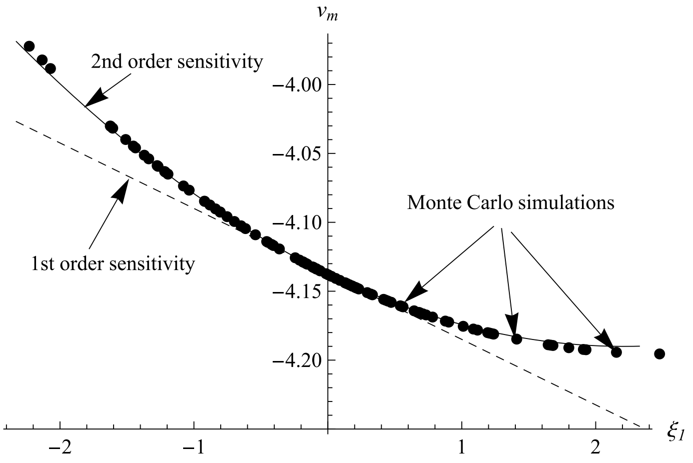


Fig. 9 Deflection in the middle of cladding, obtained with sensitivity analysis of different orders and MC simulations

$$h(X, \boldsymbol{\xi}) = \bar{h}_{wave} + \sum_{k=1}^{M} \sqrt{\lambda_k} f_k(X)\, \xi_k \tag{37}$$

The corresponding first- and second-order shape design velocity fields are then

$$\frac{\partial Y}{\partial \xi_k} = \sqrt{\lambda_k} f_k(X)\, \sin \frac{n\pi X}{L}, \quad \frac{\partial^2 Y}{\partial \xi_k \partial \xi_l} = 0 \tag{38}$$

where $\bar{h}_{wave}$ is the expected value of amplitude.

Stochastic field of wave amplitude change is represented via the first four terms of K-L expansion, thus $\boldsymbol{\xi} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$. In Fig. 9, the vertical displacement v_m in the middle of the cladding is calculated in dependence of ξ_1, while other random variables are taken at their mean value ($\xi_2 = \xi_3 = \xi_4 = 0$). The results of the first- and second-order sensitivity analysis are compared with those obtained by 100 Monte Carlo (MC) simulations. It can be seen that the second-order sensitivity analysis suits almost exactly the direct evaluation of the response, for approximately two standard deviations from the mean value.

In Table 2, the calculated mean value, standard deviation and CPU time are compared for statistics of the response obtained by the first-order sensitivity analysis, the second-order sensitivity analysis, finite difference approximation of the second-order sensitivities, and Monte Carlo simulations. In this comparison, all four random variables ($\xi_1, \xi_2, \xi_3, \xi_4$) are considered. In MC simulations, the range of random variables was limited to the interval between 0.001 and 0.999 quantile, due to physically acceptable results. The results justify the use of the second-order sensitivity analysis instead of the analysis of the first order, since the second-order results fit the results

Table 2 Mean value and standard deviation of vertical displacement v_m (in cm) and total CPU time, needed for calculation

	Mean (v_m)	Standard deviation (v_m)	Total CPU time
First-order sensitivity analysis	−4.1374 cm	0.0555 cm	0.25 s
Second-order sensitivity analysis	−4.1263 cm	0.0577 cm	0.39 s
Second-order finite difference	−4.1263 cm	0.0577 cm	46.58 s
10^3 MC simulations	−4.1262 cm	0.0588 cm	271 s
4×10^4 MC simulations	−4.1264 cm	0.0578 cm	10084 s

considerably better. As can be seen, the exact second-order sensitivity analysis is considerably more efficient in comparison with all other methods for comparable results.

4.2 Asymptotic Numerical Methods

At present, in solid mechanics and nonlinear structural mechanics there exists no iterative method that can be applied to all different problem areas in an efficient and robust way. Additionally, for highly nonlinear problems the solution of time-independent problems cannot, in general, be achieved in one step. More efficient procedures can be derived when the resulting system of equations can be naturally parametrized in a way that for some given value of parameter the solution is trivial. The system of equations $\mathbf{R}(\mathbf{p}) = \mathbf{0}$ will be parametrized for the following considerations in the form:

$$\mathbf{R}(\mathbf{p}, \lambda) = \mathbf{0}, \tag{39}$$

where λ is parameter, and solved using the standard Newton–Raphson method. With the introduction of parameter λ, the final solution is achieved in n_{step} incremental steps with associated solution vectors $\mathbf{p}_0, \ldots \mathbf{p}_{n_{\text{step}}}$. As an example, problems in solid mechanics and nonlinear structural mechanics subjected to quasi-static proportional load are frequently parametrized by introducing the loading parameter λ as follows:

$$\mathbf{R}(\mathbf{p}, \lambda) = \mathbf{R}^{\text{int}}(\mathbf{p}) - \mathbf{F}(\lambda) = 0, \quad \mathbf{F} = \lambda\, \mathbf{F}^{\text{ref}} \tag{40}$$

where $\mathbf{R}^{\text{int}}$ denotes the contribution of internal forces to the nodal force vector and $\mathbf{F}^{\text{ref}}$ is the reference load vector associated with the pattern of the applied nodal forces.

Within the asymptotic numerical method approach (see e.g., Nezamabadi et al. (2011)), a more efficient load stepping scheme is derived by expansion of the response with respect to parameter of the problem (load level λ). Thus, sensitivity parameter of the problem is $\boldsymbol{\phi} = \{\lambda\}$ and the response is approximated as

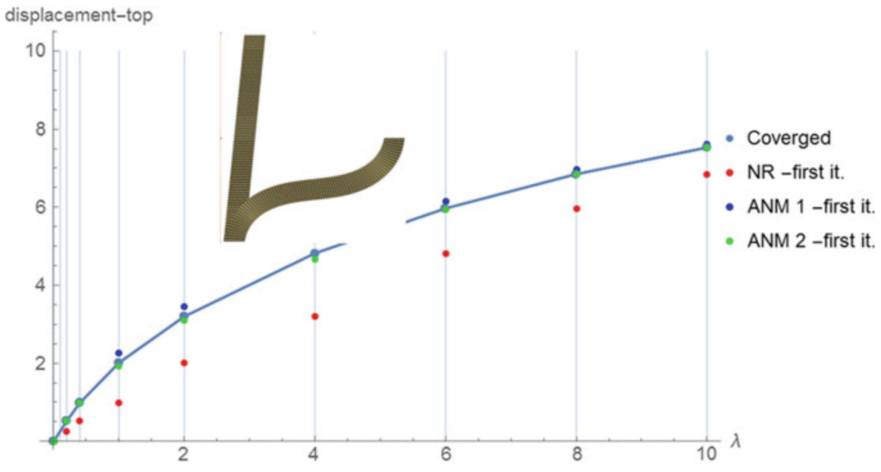


Fig. 10 Bending of column using asymptotic numerical methods

$$\mathbf{p}(\lambda) = \mathbf{p}_0 + \frac{\partial \mathbf{p}}{\partial \lambda}\delta\lambda + \frac{1}{2}\frac{\partial^2 \mathbf{p}}{\partial \lambda^2}\delta\lambda^2 \dots \tag{41}$$

The corresponding first- and second-order natural boundary condition design velocity fields are then

$$\frac{\partial \mathbf{F}}{\partial \lambda} = \mathbf{F}^{\text{ref}}, \quad \frac{\partial^2 \mathbf{F}}{\partial \lambda^2} = \mathbf{0}. \tag{42}$$

Due to the fact that within the asymptotic numerical methods, we deal with only one sensitivity parameter, also the sensitivities of the order higher than two can be obtained in a reasonable computational time (see e.g. Nezamabadi et al. (2011)).

A numerical example of bending of column modeled by two-dimensional finite strain elements is presented in Fig. 10. The final load is achieved in 8 load steps. For each load step, the converged solution is depicted together with the converged solution from the previous load step (the usual initial guess for the standard Newton–Raphson method), the first-order ANM approximation and the second-order ANM approximation. It can be seen that in this case second-order ANM approximation gives almost an exact solution . By using even higher orders one can skip Newton iterations altogether (see e.g., Nezamabadi et al. (2011)). However, this can also change dramatically, for example, with more dense meshes and non-monotonic response.

4.3 Optimization

Optimization problems were one of the first problems, where sensitivity analysis was used to improve numerical efficiency of optimization algorithms. Depending on the available order of sensitivity analysis, constrained or unconstrained problem and the form of objective function, the optimization can benefit from sensitivity analysis in several ways. Just to name some:

- the first-order sensitivity analysis is essential for the gradient-based optimization algorithms for the solution of constrained or unconstrained optimization problems,
- with the second-order sensitivity analysis, an unconstrained optimization problem can be solved using quadratically convergent NEWTON- - RAPHSON type algorithms,
- the first- or second-order sensitivity analysis of objective function can be used to form response surface leading to sequential linear or quadratic programing algorithms.

The last possibility is especially useful when the evaluation of the objective function is very costly (e.g., requires full nonlinear analysis of the global FE problem) and in the case of multiple constraints. An example will be given here where the sensitivity analysis is used to solve the problem of worst imperfection of structures in means of ultimate limit states(Kristanic and Korelc (2008)) using sequential linear programming approach. It is well known that geometrical, structural, material, and load imperfections play a crucial role in the load-carrying behavior, especially of thin-walled structures. The idea to find such a combination of imperfections that will cause the structure to fail at the lowest possible load is as old as the ascertainment of the crucial role of imperfections itself. The review of different approaches accompanied with an impact on modern design procedures of engineering structures can be found in Schmidt (2000). When analyzing structures discretized with finite elements, it turns out that the choice of the shape and size of initial imperfections have a major influence on the response of the structure and its limit state.

With the use of direct and sensitivity analysis combined with optimization, it is possible to determine the most unfavorable combination of chosen shapes representing the initial imperfection, which leads to the least possible ultimate load. Within the optimization algorithm, the objective function is constructed by means of a fully nonlinear direct and first-order sensitivity analysis. The method is not limited to small imperfections or a linear fundamental path based on Koiters asymptotical theory (Koiter (1945)) and also allows the imposition of technological constraints on the shape of the imperfection, thus making it possible to avoid unrealistically low ultimate loads. When carefully constructed, the objective function and constraints remain linear, enabling the use of numerically efficient and readily available sequential linear programming algorithms.

Let $\mathbf{X}_p$ be a coordinate of the nodes of the perfect geometry, $\mathbf{X} = \mathbf{X}_p + \bar{\mathbf{X}}$ coordinates of the imperfect geometry, where imperfection $\bar{\mathbf{X}}$ is approximated as linear combination of M base shapes $\mathbf{\Gamma}_i$ and corresponding weights α_i (43).

$$\mathbf{X} = \mathbf{X}_p + \bar{\mathbf{X}} = \mathbf{X}_p + \sum_{i=1}^{M} \alpha_i \mathbf{\Gamma}_i \tag{43}$$

Base shapes can be chosen arbitrarily. The most convenient set of shapes is the set of buckling modes that can be extended by eigenshapes of tangent matrix, empirically known as worst shapes or deformation shapes. The response of the imperfect, materially and geometrically nonlinear structure is defined by its response curve $\mathbf{u}(\lambda)$, where λ is the load level as defined for proportional loading by (40). Let λ_l be the ultimate load factor. A limit state of a structure is generally defined with the limit point of the equilibrium path. In real, imperfect structures, this criterion proves unreliable because of the possible exceeding of permissible tolerances of displacements or deformations before reaching the limit point. The goal is to determine such coefficients α_i that the ultimate limit load factor λ_l of the structure would be minimal. Therefore, a minimization problem (44) for the limit load factor can be defined, where the imposition of technological constraints requires that the maximal amplitude of the imperfection has to be equal to or smaller than the amplitude of the prescribed equivalent geometrical imperfections e_0.

$$\begin{aligned} &\min_{\alpha_i} \lambda_l \\ &||\bar{\mathbf{X}}||_\infty \leqslant e_0 \end{aligned} \tag{44}$$

Solution of the nonlinear optimization problem (44) requires full nonlinear analysis (direct and, depending on optimization algorithm, also sensitivity analysis) of the structure at every iteration of optimization algorithm. Because of the enormous computational time required, this approach is not feasible at this time. The fully nonlinear problem (44) is simplified by expansion of the limit state load factor of the imperfect structure to a Taylor series around the imperfect geometry. The limit load factor $\lambda_l(\bar{\mathbf{X}}(\alpha_i))$ is then for k^{th} global iteration of the sequential nonlinear optimization algorithm written as

$$\lambda_l \approx \lambda_l^{k-1} + \sum_{i=1}^{M} \left.\frac{\partial \lambda_l}{\partial \alpha_i}\right|_{\alpha_i^{k-1}} \Delta\alpha_i^k \tag{45}$$

where coefficients of the series expansion $\partial\lambda_l/\partial\alpha_i|_{\alpha_i^{k-1}}$ are obtained by the first-order shape sensitivity analysis. Sensitivity parameters of the problem are weights α_i and the the corresponding shape design velocity field is obtained by the differentiation of (43) with respect to sensitivity parameters

$$\frac{\partial \mathbf{X}}{\partial \alpha_i} = \mathbf{\Gamma}_i. \tag{46}$$

Function (45) is a linear function. However, the constraint in (44) is a highly nonlinear function. A set of linear constraints for the maximal amplitude of the total imperfection vector $|\bar{X}_{l,m}| = |\sum_i \alpha_i \Gamma_{l,m}| \leqslant e_0; \forall l, m$, where $\bar{X}_{l,m}$ and $\Gamma_{l,m}$ are the m^{th} component of the imperfection and base shape vector in l^{th} node, can be defined instead. The result is numerically highly efficient sequential linear programming problem. For each global iteration of sequential linear programming algorithm only one fully nonlinear limit state analysis together with the shape sensitivity analysis

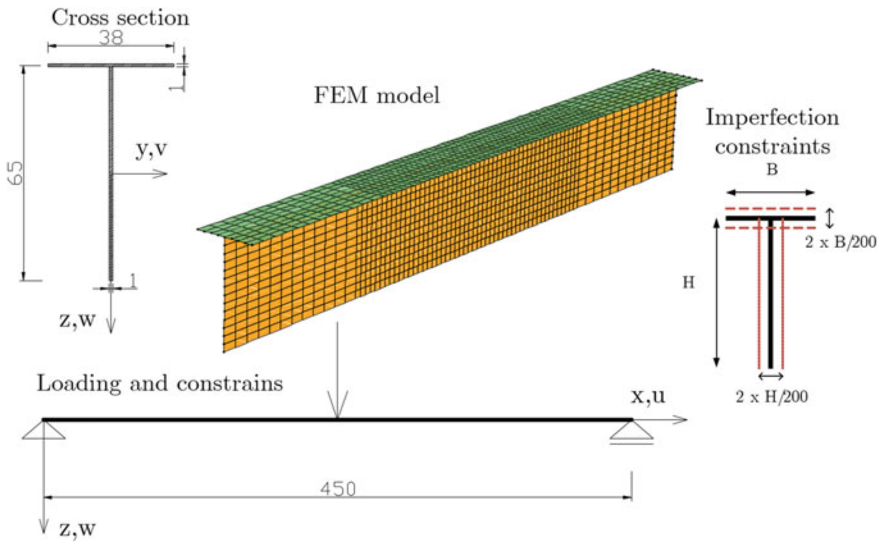


Fig. 11 Example of a T cross-sectional thin-walled beam example

has to be performed. Computational cost for the solution of the corresponding linear programming problem is in fact negligible.

The example presented refers to the ultimate load calculation of a simply supported thin-walled beam with a T cross section, loaded with a concentrate force at the mid-length. The geometrical details and loads are presented in Fig. 11. The thin-walled girders in this section were modeled by elastoplastic four node shell elements based on finite rotations, six-parameter shell theory combined with assumed natural strain formulation and two enhanced strain modes for improved performance. Within the optimization problem, it was necessary to define 3150 constraint equations for the maximal initial imperfection amplitude perpendicular to the web and 2025 constraint equations for the maximal imperfection amplitude perpendicular to the flange. The structure is analyzed considering the shape base consisting of buckling modes. In Fig. 12, the calculated limit load of the T-beam with increasing number of base shapes is shown. The results show a clear convergence of the calculated limit load.

Convergence of the global iterative optimization process of finding the most unfavorable imperfection by considering 52 base shapes ($M = 52$) is presented in Fig. 13. The most unfavorable initial imperfection is achieved within engineering tolerances in the 4^{th} global iteration of the sequential linear programming algorithm.

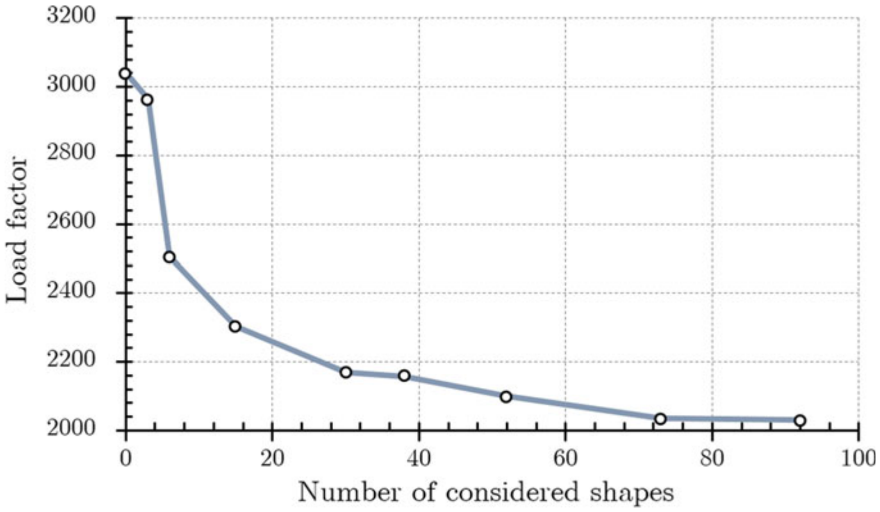


Fig. 12 Convergence of the ultimate limit load with the number of base shapes

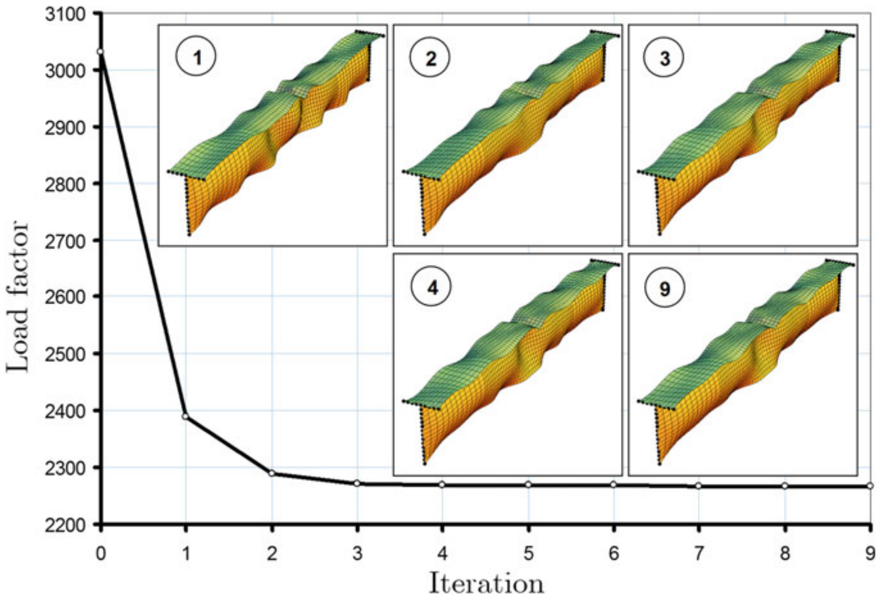


Fig. 13 Convergence of the global iterative optimization process of finding the most unfavorable imperfection shape of a T cross section

4.4 Multi-scale Modeling

The use of different kinds of multi-scale methods is limited by specifications of the problem to be solved. Standard two-level finite element homogenization approach FE^2 is appropriate for problems with weakly coupled scales. If the difference between two scales is finite, the FE^2 multi-scale approach fails, then some sort of domain decomposition method can be applied. Within the sensitivity analysis based multi-scale computational environment, various types of multi-scale approaches can be freely mixed. The automation of the FE^2 methods requires the first-order sensitivity analysis with respect to prescribed essential boundary conditions, and the automation of domain decomposition methods requires the second-order sensitivity analysis with respect to prescribed essential boundary conditions. Thus, finite element code that supports the first- and second-order sensitivity analyses enable unification and automation of various multi-scale approaches for an arbitrary nonlinear, time-dependent, coupled problem (e.g., general finite strain plasticity). More details can be found in Korelc and Zupan (2018).

5 Conclusions

The paper describes a hybrid symbolic-numerical approach to the automation of primal and sensitivity analyses of computational models formulated and solved by finite element method. A hybrid symbolic-numerical approach that combines a general computer algebra system, an automatic differentiation technique, and an automatic code generation with the general-purpose finite element environment is proposed as an appropriate method.

Additional to the solution of primal problem, efficient computational algorithms often require also the solution of sensitivity problem. Thus, any proposed method of automation should address the automation of primal as well as sensitivity analysis. ADB notation together with automatic differentiation and automatic code generation enables automatic derivation of element-level subroutines for the evaluation of analytically exact pseudo-load vectors, while the global sensitivity problem remains independent of element formulation. Consequently, once we have the individual element codes that support primal and sensitivity analyses for all input parameters of the individual finite elements, the only unanswered question remains "What is the velocity field of the problem?". Sensitivity analysis based stochastic analysis and asymptotic numerical methods were given as examples of identification and definition of design velocity fields. It is important to notice that no additional functionality or coding is needed for the implementation of these examples, apart from knowing the design velocity field of the problem.

Acknowledgements The financial support for this work was obtained from the Slovenian Research Agency within the research group P2-0158.

References

Bischof, Ch., Hovland, P., & Norris, B. (2002). Implementation of automatic differentiation tools. In C. Norris, & Jr. J. B. Fenwick (Eds.), *Proceedings of the ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation*. New York: ACM Press.

Choi, K. K., & Kim, N. H. (2005a). *Structural sensitivity analysis and optimization 1*. New York: Linear systems. Springer Science+Business Media.

Choi, K. K., & Kim, N. H. (2005b). *Structural sensitivity analysis and optimization 2*. New York: Nonlinear systems and applications. Springer Science+Business Media.

Eyheramendy, D., & Zimmermann, Th. (2000). Object-oriented symbolic derivation and automatic programming of finite elements in mechanics. *Engineering with Computers*, *15*(1), 12–36.

Ghanem, R., & Spanos, P. D. (2003). *Stochastic finite elements: A spectral approach*. New York: Dover.

Griewank, A. (2000). *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. Philadelphia: SIAM.

Hudobivnik, B., & Korelc, J. (2016). Closed-form representation of matrix functions in the formulation of nonlinear material models. *Finite Elements in Analysis and Design*, *111*, 19–32.

Keulen, F., Haftka, R. T., & Kim, N. H. (2005). Review of options for structural design sensitivity analysis. part 1: Linear systems. *Computer Methods in Applied Mechanics and Engineering, 194*, 3213–3243.

Kleiber, M., Antunez H., Hien, T. H., & Kowalczyk, P. (1997). *Parameter sensitivity in nonlinear mechanics*. John Wiley and Sons.

Koiter, W. T. (1945). *The stability of elastic equilibrium.* Stanford University.

Korelc, J. (1997). Automatic generation of finite-element code by simultaneous optimization of expressions. *Theoretical Computer Science*, *187*, 231–248.

Korelc, J. (2002). Multi-language and multi-environment generation of nonlinear finite element codes. *Engineering with Computers*, *18*, 312–327.

Korelc, J. (2009). Automation of primal and sensitivity analysis of transient coupled problems. *Computational Mechanics*, *44*, 631–649.

Korelc, J. (2018). *AceFEM and AceGen user manuals* (6.902 edition). http://symech.fgg.uni-lj.si/.

Korelc, J., & Stupkiewicz, S. (2014). Closed-form matrix exponential and its application in finite-strain plasticity. *International Journal for Numerical Methods in Engineering*, *98*, 960–987. https://doi.org/10.1002/nme.4653.

Korelc, J., & Wriggers, P. (2016). *Automation of finite element methods*. Switzerland: Springer.

Korelc, J.,& Zupan, N. (2018). Unified approach to sensitivity analysis based automation of multiscale modelling. In J. SORIC, P. WRIGGERS, & O. ALLIX (Eds.), *Multiscale modeling of heterogeneous structures, (Lecture notes in applied and computational mechanics)*. Berlin: Springer International Publishing AG.

Kristanic, N., & Korelc, J. (2008). Optimization method for the determination of the most unfavorable imperfection of structures. *Computational Mechanics*, *42*, 859–872.

Logg, A., Mardal, K. A., & Wells, G. N. (2012). *Automated solution of differential equations by finite element method*. Berlin Heidelberg: Springer.

Melink, T., & Korelc, J. (2014). Stability of karhunen- love expansion for the simulation of gaussian stochastic fields using galerkin scheme. *Probabilistic Engineering Mechanics*, *37*, 7–15.

Nezamabadi, S., Zahrouni H., & Yvonnet, J. (2011). Solving hyperelastic material problems by asymptotic numerical method. *Computational Mechanics, 47*, 77–92.

Schmidt, H. (2000). Stability of steel shell structures - general report. *Journal of Constructional Steel Research*, *55*, 159–181.

Solinc, U., & Korelc, J. (2015). A simple way to improved formulation of fe2 analysis. *Computational Mechanics*, *56*, 905–915.