



FEDERAL UNIVERSITY OF SANTA CATARINA
CENTER OF TECHNOLOGY
GRADUATE PROGRAM IN MECHANICAL ENGINEERING

Vitor Takashi Endo

Hyper-dual sensitivity analysis: a second-order evaluation in structural problems

Florianópolis
2020

Vitor Takashi Endo

Hyper-dual sensitivity analysis: a second-order evaluation in structural problems

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Engineering in the Graduate Program in Mechanical Engineering of Federal University of Santa Catarina.

Supervisor: Prof. Eduardo Alberto Fancello, D.Sc.

Co-supervisor: Prof. Pablo Andrés Muñoz-Rojas, Dr.Eng.

Florianópolis

2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Endo, Vitor Takashi

Hyper-dual sensitivity analysis : a second-order
evaluation in structural problems / Vitor Takashi Endo ;
orientador, Eduardo Alberto Fancello, coorientador, Pablo
Andrés Muñoz-Rojas, 2020.

113 p.

Tese (doutorado) - Universidade Federal de Santa
Catarina, Centro Tecnológico, Programa de Pós-Graduação em
Engenharia Mecânica, Florianópolis, 2020.

Inclui referências.

1. Engenharia Mecânica. 2. Análise de sensibilidade. 3.
Números hiperduais. 4. Diferenciação automática. 5.
Hessiana diagonal. I. Fancello, Eduardo Alberto. II. Muñoz
Rojas, Pablo Andrés. III. Universidade Federal de Santa
Catarina. Programa de Pós-Graduação em Engenharia Mecânica.
IV. Título.

Vitor Takashi Endo

Hyper-dual sensitivity analysis: a second-order evaluation in structural problems

O presente trabalho em nível de Doutorado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Marco Lucio Bittencourt, Dr.Eng.
Unicamp

Prof. Rogério José Marczak, Dr.Eng.
UFRGS

Prof. Paulo de Tarso Rocha de Mendonça, Ph.D.
UFSC

Prof. Eduardo Lenz Cardoso, Dr.Eng.
UDESC

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Doutor em Engenharia Mecânica.

Prof. Paulo de Tarso R. Mendonça, Ph.D.
Coordenação do Programa de Pós-Graduação

Prof. Eduardo Alberto Fancello, D.Sc.
Orientador

Florianópolis, 2020.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my supervisors, Prof. Eduardo Alberto Fancello and Prof. Pablo Andrés Muñoz-Rojas, for sharing their extensive knowledge and experience.

Many thanks to the examiners, Prof. Marco Lucio Bittencourt, Prof. Rogério José Marczak, Prof. Paulo de Tarso Rocha de Mendonça, and Prof. Eduardo Lenz Cardoso, for their careful reading of the manuscript and their helpful comments to improve the presentation of this work.

I am deeply indebted to the members of the Department of Mobility Engineering of Federal University of Santa Catarina for supporting me in this project, especially Prof. Silvia Lopes de Sena Taglialha, Prof. Maurício de Campos Porath, Prof. André Luís Condino Fajarra, Prof. Lucas Weihmann, Prof. Luis Fernando Peres Calil, Prof. Catia Regina Silva de Carvalho Pinto, Prof. Renata Cavion, and Prof. Alexandro Garro Brito.

I am extremely grateful to my colleagues Prof. Thiago Pontin Tancredi and Prof. Marcos Alves Rabelo, who kindly took care of my students throughout this course.

I would like to extend my sincere thanks to Prof. José Carlos de Carvalho Pereira for providing valuable advice and encouragement.

Thanks also to my colleagues Thiago André Carniel, Victor Campos, André Kühn, Diego Fernandes Rodrigues, and Paulo Bastos de Castro for their practical suggestions and feedback on this study.

I would like to acknowledge the assistance of Prof. Romulo Alberto Castillo Cardenas and Prof. Leonardo Moreto Elias during our math studies.

I am also grateful to Prof. Wyllian Bezerra da Silva, Prof. Pablo Andretta Jaskowiak, and Prof. Renato Oba for their helpful guidance on \LaTeX and computer coding.

Thanks should also go to my friends and bandmates, Prof. Luiz Eduardo Bueno Minioli, Prof. Helton da Silva Gaspar, Prof. Adriano Fagali de Souza, Prof. James Schipmann Eger, and Prof. Diogo Lôndero da Silva.

This work would not have been possible without the support and nurturing of my fiancée Josiane Giese.

Special thanks to my parents, Elena Tiekko Sato Endo and Armando Haruo Endo, who provide encouragement and patience throughout all projects.

「風が立つ、生きようと試みなければならない。」
ポール・ヴァレリー

RESUMO

Neste trabalho relata-se o emprego dos números hiperduais como uma ferramenta de diferenciação automática num contexto de análise de sensibilidade. O procedimento hiperdual permite a obtenção de resultados exatos, tanto para a primeira como para a segunda derivada; apesar desta importante característica, sua aplicação no âmbito de sensibilidade em problemas estruturais não havia sido observada. Para tanto, tendo em vista a ocorrência de grandezas tensoriais na formulação de elementos finitos, foi necessário o desenvolvimento das operações aritméticas pertinentes à conversão hiperdual de uma função tensorial. Este avanço permite, por exemplo, o cômputo automático das derivadas da matriz de rigidez em relação a um vetor contendo as variáveis de projeto. Nesse contexto, enfatiza-se que as operações hiperduais contemplam informações no nível do elemento, com o intuito de restringir o tamanho dos tensores envolvidos e, assim, mitigar o seu impacto no esforço computacional. Além de uma descrição considerando estruturas de comportamento não linear, foram avaliadas particularizações assumindo hipóteses de linearidade e focando nos termos da diagonal principal da matriz Hessiana. Esta proposta simplificadora, também aplicada às operações hiperduais, reduz sobremaneira o custo computacional, tornando-se mais viável em termos de aplicação em problemas de engenharia. Salienta-se que o método descrito para análise de sensibilidade é geral, sendo capaz de acomodar uma variedade de formulações ou configurações de problemas. Esta associação com uma ferramenta de diferenciação automática permite que a técnica atinja interessante nível de generalidade. Desse modo, este estudo fundamentalmente atua no sentido de facilitar o emprego de algoritmos de segunda ordem para a solução de problemas de projeto mecânico considerando ferramentas de otimização. Ademais, uma vez que esta ferramenta de diferenciação pode ser utilizada em outros ramos do conhecimento, além da mecânica dos sólidos computacional, são apresentados fragmentos relevantes do código, facilitando aplicações e aprimoramentos futuros.

Palavras-chave: Análise de sensibilidade. Diferenciação automática. Hessiana diagonal. Método de elementos finitos. Números hiperduais. Otimização estrutural.

RESUMO EXPANDIDO

Introdução

A otimização estrutural tem sido reportada como uma importante ferramenta para o desenvolvimento de produtos. Por meio de uma avaliação guiada envolvendo protótipos virtuais, torna-se possível a obtenção de produtos de elevado desempenho. Neste contexto, muitos algoritmos utilizam informações de derivadas para a determinação da direção de busca. Esta informação, também definida como análise de sensibilidade, pode ser representada pela derivada da função objetivo em relação às variáveis de projeto.

Em aplicações de engenharia, aproximações numéricas são comumente empregadas por propiciarem generalidade ao código computacional, possibilitando o uso da ferramenta em diferentes ramos. Todavia, em muitos casos, erros podem ser introduzidos, o que pode afetar o processo de convergência do problema. Diante disso, estudos considerando abordagens alternativas buscam um aprimoramento no cálculo desta importante resposta, sobretudo quando informações de segunda ordem são requeridas.

Objetivos

Este estudo tem como objetivo a descrição de um método para obtenção da sensibilidade de segunda ordem. Nesta abordagem, ressalta-se o caráter automático para o cômputo das derivadas envolvendo expressões de elementos finitos. Para tanto, tornou-se necessário o desenvolvimento de uma técnica de diferenciação baseada nos números hiperduais.

Métodos

Neste trabalho foram desenvolvidas duas formulações hiperduais: uma voltada ao cálculo de derivada de funções tensoriais e uma variante que foca na obtenção dos termos da diagonal principal das derivadas de segunda ordem. Tais aprimoramentos exigiram uma nova definição matemática, assim como adaptações de regras aritméticas, necessárias para a conversão hiperdual de uma dada função. Ademais, foi considerada a sobrecarga de operador como técnica de implementação computacional, o que permite a manutenção da sintaxe do código.

A partir de expressões gerais de sensibilidade, foram identificados os termos contendo as informações de derivada que seriam obtidos mediante o procedimento proposto. Observou-se que este estudo de sensibilidade requer o cômputo de derivadas de expressões tensoriais, como a matriz de rigidez tangente e a força interna, em relação a um vetor contendo as variáveis de projeto. Assim, justificam-se os desenvolvimentos acerca dos números hiperduais para aplicações em análise de sensibilidade.

Destaca-se que, com o intuito de mitigar o impacto do esforço computacional, o método proposto considera que as conversões ocorrem no nível do elemento. Deste modo, restringe-se sobremaneira o tamanho dos tensores envolvidos no procedimento. A contribuição de cada elemento é então inserida na expressão global por meio de um operador montagem, que contempla os mapeamentos dos graus de liberdade e as variáveis de projeto.

Com o intuito de facilitar o emprego deste método de sensibilidade em problemas de engenharia, também foram colocadas hipóteses simplificadoras para identificar um uso apropriado em conjunto com os números hiperduais. Dentre as particularizações avaliadas, destaca-se uma abordagem na qual se obtém os termos da diagonal principal da matriz Hessiana, considerando o uso do método adjunto em estruturas de comportamento linear.

Resultados e discussões

O esquema hiperdual possibilitou a obtenção de resultados exatos para as derivadas de primeira e segunda ordem, independentemente do valor adotado para a perturbação. Ademais, permitiu a avaliação de funções tensoriais, recorrentes nas formulações de elementos finitos, com a opção de cálculo completo ou parcial das derivadas de segunda ordem. Ressalta-se também o caráter transversal deste tópico, tendo em vista que o cômputo de derivadas é presente em vários ramos da ciência.

Num estudo envolvendo a aplicação do procedimento hiperdual em modelos hiperelásticos, constatou-se um interessante comportamento em termos de seu custo computacional. Embora uma conversão hiperdual envolva numerosas operações internas, seu impacto no esforço computacional é atenuado com o aumento do número de elementos, já que a etapa de solução do sistema de equações tende a dominar o tempo de processamento da análise. Além disso, todo o comportamento do material foi descrito apenas com a implementação da função da energia de deformação; a tensão e o módulo tangente são obtidos automaticamente. Portanto, este esquema pode constituir uma conveniente ferramenta para o desenvolvimento de novos modelos constitutivos.

Foram descritas as bases teóricas de um método automático e exato para cálculo de sensibilidade de segunda ordem. Embora o procedimento hiperdual disponibilize adequadamente as informações de derivadas, observou-se que, considerando o método direto e estruturas não lineares, este cálculo de sensibilidade é bastante dispendioso, o que pode dificultar o seu uso prático. Por outro lado, ao optarmos pelos termos da diagonal principal com o uso do método adjunto em estruturas lineares, foi possível notar uma importante redução nos esforços computacionais. Ainda assim, este estudo endereça faixas importantes de aplicações, já que muitos problemas são dominados por estes termos diagonais. O esquema proposto para análise de sensibilidade é geral, sendo capaz de se adequar a uma ampla gama de formulações de elementos finitos e problemas de projeto mecânico.

Conclusões

Os desenvolvimentos descritos contribuem para o emprego de algoritmos de otimização de segunda ordem em problemas de projeto mecânico. Tais aprimoramentos de técnicas da mecânica computacional permitem importantes avanços no desenvolvimento de produtos e inovação tecnológica.

Palavras-chave: Análise de sensibilidade. Diferenciação automática. Hessiana diagonal. Método de elementos finitos. Números hiperduais. Otimização estrutural.

ABSTRACT

We report the use of hyper-dual numbers to obtain accurate derivatives in a second-order design sensitivity study. For this purpose, we develop the arithmetic operations that define the conversion of a tensor-valued function into hyper-dual numbers. Hence, by using hyper-dual numbers as the differentiation tool, we evaluate an element-level approach to evaluate the derivatives of the internal force and the tangent stiffness. Besides describing the general expressions considering structures with nonlinear behavior, we also present a particularized version focusing on the diagonal terms of the Hessian in linear structures. With modest modifications, this diagonal sensitivity scheme is significantly lighter in terms of computational costs, which facilitates its application in engineering problems. The proposed method shows interesting generality aspects due to the implementation strategy with the operator overloading technique; in this case, the sensitivity scheme can adapt itself to a variety of finite element formulations or problem settings. As this differentiation scheme using hyper-dual numbers also represents a black-box tool for general purposes, we supply the computer implementation written in Fortran. The implications of this study are primarily related to design optimization, as the hyper-dual sensitivity scheme promotes the use of second-order optimization algorithms, which may allow better convergence rates to solve intricate problems in engineering applications.

Keywords: Automatic Differentiation. Diagonal Hessian. Finite Element Method. Hyper-dual Numbers. Sensitivity Analysis. Structural Optimization.

LIST OF FIGURES

Figure 1 – Stress vs. stretch for each conjugate pair. $\text{Error}_{\max} = 10^{-15}$	41
Figure 2 – Tangent modulus vs. stretch for each conjugate pair. $\text{Error}_{\max} = 10^{-13}$	41
Figure 3 – Fortran code describing Ogden’s material model.	42
Figure 4 – Deformed shape of Model A (Neighborhood level: 1).	43
Figure 5 – Deformed shape of Model B (Neighborhood level: 2).	43
Figure 6 – Deformed shape of Model C (Neighborhood level: 3).	43
Figure 7 – Model size and overall processing time, models organized in groups with the same number of degrees of freedom.	46
Figure 8 – Model size and overall processing time, models organized in groups with the same neighborhood level.	47
Figure 9 – Relative processing time using the analytical expressions.	48
Figure 10 – Relative processing time using the hyper-dual procedure.	48
Figure 11 – Normalized computing time of intermediate operations.	49
Figure 12 – An algorithm to describe the hyper-dual sensitivity analysis in a finite element code.	55
Figure 13 – Arithmetic rules of tensor hyper-dual numbers (part 1).	95
Figure 14 – Arithmetic rules of tensor hyper-dual numbers (part 2).	96
Figure 15 – Minimal working example: evaluation of $f(x, y) = 5x^2y^3$ at $x_0 = 5$ and $y_0 = 2$	97
Figure 16 – Hyper-dual conversion of the stiffness matrix.	98
Figure 17 – Implementing the operator delta to obtain the diagonal terms.	98
Figure 18 – Arithmetic rules of diagonal hyper-dual numbers (part 1).	99
Figure 19 – Arithmetic rules of diagonal hyper-dual numbers (part 2).	100
Figure 20 – Diagonal hyper-dual conversion of the stiffness matrix.	101
Figure 21 – Truss structure with two elements.	106

LIST OF TABLES

Table 1 – Fundamental properties of hyper-dual numbers.	21
Table 2 – Derivative information obtained from the hyper-dual form.	23
Table 3 – Derivative information obtained from the diagonal hyper-dual form.	32
Table 4 – Product of two scalar hyper-dual variables: original vs. diagonal.	33
Table 5 – Measures of strain, stress and tangent modulus.	39
Table 6 – Material parameters.	40
Table 7 – Description of the numerical models in terms of model size and input parameters (DOF – degrees of freedom, nel – number of elements, A_0 – area, and F_y – vertical load).	44
Table 8 – Relationship among derivative methods, perturbation and number of iterations.	45
Table 9 – Required derivative information in nonlinear cases.	52
Table 10 – Required derivative information in linear cases.	60
Table 11 – Internal force expressions: \mathbf{q}_l	102
Table 12 – Tangent stiffness expressions: \mathbf{k}_{T1}	104
Table 13 – Tangent stiffness expressions: \mathbf{k}_{T2l}	105

CONTENTS

1	INTRODUCTION	14
1.1	KNOWLEDGE GAP	16
1.2	OBJECTIVES	16
1.3	DOCUMENT OVERVIEW	17
1.3.1	Highlights	17
1.3.2	Scientific papers	18
1.3.3	Manuscript submission	18
2	TENSOR HYPER-DUAL NUMBERS	19
2.1	NUMERICAL PROCEDURES FOR THE DERIVATIVE CALCULATION	19
2.2	TENSOR HYPER-DUAL FORMULATION	22
2.2.1	Mathematical operations	23
2.2.2	Hyper-dual conversion	27
2.2.3	Example	28
2.2.4	Computer implementation	31
2.3	DIAGONAL HYPER-DUAL FORMULATION	31
2.3.1	Adaptation of the arithmetics properties	32
2.3.2	Example	34
3	HYPER-DUAL HYPERELASTICITY	37
3.1	OGDEN'S HYPERELASTIC MODEL	38
3.2	HYPER-DUAL REPRESENTATION	39
3.2.1	Analytical validation	39
3.2.2	Material subroutine	40
3.3	A STUDY ON THE PROCESSING TIME	42
3.3.1	Effects of the perturbation value	44
3.3.2	Hyper-dual performance in constitutive models	45
3.3.2.1	Overall effort	46
3.3.2.2	Intermediate effort	47
4	HYPER-DUAL SENSITIVITY ANALYSIS	50
4.1	THEORETICAL BACKGROUND	50
4.1.1	Typical simplifications	51
4.1.2	Required derivative terms	52
4.2	SEMI-ANALYTICAL METHOD IN NONLINEAR PROBLEMS	53
4.2.1	Hyper-dual form of internal force and tangent stiffness	53
4.2.2	Sensitivity analysis algorithm	54
4.3	DIAGONAL PARTICULARIZATION IN LINEAR PROBLEMS	56
4.3.1	Direct method	56
4.3.2	Adjoint method	58

4.3.3	Required derivative terms	59
4.3.4	Element-level design sensitivity analysis	60
4.3.4.1	Hyper-dual form of stiffness matrix and external force	60
4.3.4.2	Extracting the derivatives from the hyper-dual variable	61
4.3.4.3	Global assembly	61
5	DISCUSSION	62
5.1	TENSOR HYPER-DUAL NUMBERS	62
5.2	HYPER-DUAL HYPERELASTICITY	64
5.3	HYPER-DUAL SENSITIVITY ANALYSIS	65
6	CONCLUSION	69
	REFERENCES	70
	APPENDIX A – HYPER-DUAL NUMBERS	87
A.1	HYPER-DUAL CONVERSION	87
A.1.1	Scalar function of a scalar argument	87
A.1.2	Scalar function of a vector argument	89
A.1.2.1	Full Hessian calculation	89
A.1.2.2	Mixed second-order derivative terms	90
A.1.2.3	Diagonal terms of the Hessian	91
A.2	COMPUTER IMPLEMENTATION	92
A.2.1	Tensor hyper-dual numbers	92
A.2.2	An operator to extract the diagonal terms	93
A.2.3	Diagonal hyper-dual numbers	94
	APPENDIX B – NONLINEAR TRUSS FINITE ELEMENT	102
B.1	INTERNAL FORCE	102
B.2	TANGENT STIFFNESS MATRIX	103
	APPENDIX C – DISPLACEMENT SENSITIVITY ANALYSIS	106
C.1	CASE STUDY	106
C.2	HYPER-DUAL PROCEDURE	107
C.3	GLOBAL FINITE DIFFERENCE METHOD	112

1 INTRODUCTION

Numerical methods have been used in the industry to develop innovative engineering products. As the virtual prototypes are properly tested using CAE¹ tools, significant improvements have been observed in terms of performance, reliability, and development costs. The state of art technology related to product development involves a combination of optimization algorithms and multi-physics simulation. Concerning the field of computational solid mechanics, recent achievements allow the simulation of intricate phenomena, including the modeling of new materials, part interactions, and so forth. Therefore, studies involving such numerical methods, particularly the FEM² for structural analyses, still represent an important research topic.

Aiming for disruptive advances in engineering products, many optimization techniques have been applied as a relevant mechanical design tool. Essentially, design optimization studies identify the best configuration of variables by extremizing an objective function subject to a set of constraints. Structural optimization methods, frequently categorized into size, shape, and topology optimization (BENDSØE; SIGMUND, 2003), allow the production of innovative and cost-effective structures based on the evaluation of guided trial designs. Thus, researchers have been reporting the relevance of optimization tools to solve intricate engineering problems, contributing to the life quality of modern societies.

In this context, many optimization algorithms and reliability studies depend on the derivative information defined as design sensitivity analysis (CHO; JUNG, 2003; XIAO et al., 2011), which evaluates how the variables affect the system response (TORTORELLI; MICHALERIS, 1994; CHOI; KIM, 2005a, 2005b; KLEIBER, 1997; HAFTKA; GÜRDAL, 1991). For instance, some gradient-based algorithms, such as the steepest descent method, rely on the first-order derivative information to identify the search direction. As an attempt to improve the search direction in intricate optimization problems, we can consider second-order algorithms, which may provide better results in terms of rate of convergence (ARORA, J., 2016). Primarily, Newton's method uses the Hessian of the function and shows a quadratic rate of convergence within a certain radius of the minimum (CHANG, 2015).

J. S. Arora and Q. Wang (2005) reviewed relevant optimization algorithms in structural optimization, including second-order formulations. Horowitz and Afonso (2002) stated that the SQP³ algorithm is specifically well suited for structural optimization with highly nonlinear functions. Holzleitner and Mahmoud (1999) evaluated shape optimization using SQP, whereas Rong et al. (2013) and Albert A. Groenwold and Etman (2010) studied topology optimization problems. In this regard, Rojas-Labanda and Stolpe (2016) remarked that the Hessian allowed a reduction in terms of the number of iterations and the value of the objective function.

Although desirable in second-order algorithms, the full Hessian information is in general

¹ CAE – Computer Aided Engineering.

² FEM – Finite Element Method.

³ SQP – Sequential Quadratic Programming

difficult to handle; the mathematical complexity, accuracy problems, and computational costs hinder the practical usage of such methods in large models (CHANG, 2015). Thus, many studies reported the use of Hessian approximations based on the calculation of the diagonal terms (FLEURY, 1989; JAWED; MORRIS, 1984; XIA; LIU, P., 1987). Etman et al. (2012) developed a sequential convex programming algorithm based on diagonal quadratic subproblems. Albert Groenwold et al. (2007) presented an incomplete series expansion, including the description of the diagonal approximation. Focusing on structural design, Li and Khandelwal (2015) and Albert A. Groenwold and Etman (2010) evaluated a diagonal quadratic approximation in topology optimization.

The computer algorithms would primarily require analytical expressions concerning the derivatives of the objective function with respect to the design variables. However, by definition, the analytical representation is focused on a very specific case, lacking generality. To overcome such limitation, a general approach using numerical approximations is frequently observed, since it can consistently adjust itself according to a given problem. This approach typically involves a trade-off between accuracy and efficiency. To illustrate, assuming the use of the finite difference method to obtain the derivative approximations, it is well established that this strategy might suffer from accuracy issues that could limit the rate of convergence. Recently, Giraldo-Londoño et al. (2020) and Andrei (2020) obtained approximations of the diagonal of the Hessian using the BFGS⁴ algorithm and the finite difference method, respectively. Therefore, in this context concerning design optimization, it is of interest to develop an adaptable procedure to accurately perform the derivative calculation, especially if we consider a second-order evaluation.

To address these drawbacks, we studied a recent differentiation method based on hyper-dual numbers. As described by Jeffrey Alan Fike (2013), this scheme represents a tool to obtain both first and second-order derivatives. In contrast to the conventional finite differences using real numbers, this methodology shows negligible errors and it is not affected by perturbation values. As also discussed by Neuenhofen (2018), the hyper-dual procedure comprises an automatic differentiation tool when implemented via operator overloading. Essentially, a function evaluation is decomposed in a sequence of elementary sub-expressions that describe the derivative information. Therefore, this scheme is a pertinent tool for many computational applications, as it is relatively easy to be implemented in a computer code (CHIVERS; SLEIGHTHOLME, 2011). Recently, Vigliotti and Auricchio (2020) argued that automatic differentiation methods are relevant in computational solid mechanics.

Few works discuss the hyper-dual scheme alongside numerical methods for engineering applications. Cohen and Shoham (2015, 2017) evaluated multi-body kinematic expressions of a robot manipulator using hyper-dual numbers. Brake et al. (2016) studied uncertainty quantification by developing parameterized reduced order models. Some authors (KIRAN; KHANDELWAL, 2015; TANAKA et al., 2015, 2016; ENDO et al., 2017) evaluated hyper-dual numbers to obtain stress and tangent modulus in hyperelastic material models. Fujikawa et al.

⁴ BFGS – Broyden-Fletcher-Goldfarb-Shanno.

(2016) calculated internal force vectors and stiffness matrices by differentiating the strain energy function using a hyper-dual scheme.

1.1 KNOWLEDGE GAP

To our knowledge, no previous research has investigated the use of hyper-dual numbers for the second-order sensitivity analysis in structural problems. Moreover, a hyper-dual sensitivity study focusing on the diagonal terms of the Hessian matrix was neither reported. We only found studies using the complex-step method, which is focused on the first-order evaluation (BANKS et al., 2015; KIRAN et al., 2017). Our proposal considers a semi-analytical approach in which the derivative calculations are confined at element level; Jin et al. (2010), Geovane Augusto Haveroth et al. (2015) and Geovane A. Haveroth and Muñoz-Rojas (2016) studied the complex-step counterpart.

As for the differentiation tool, we could not find documents regarding the arithmetic operations of tensor hyper-dual variables with vector argument; the mathematical definition and computational implementation of these rules comprise an important requirement for this study since many finite element expressions are represented by tensor variables.

1.2 OBJECTIVES

The objective of this study is to evaluate hyper-dual numbers as a numerical derivative tool in a design sensitivity context. We aim to evaluate the expressions of a second-order sensitivity analysis, in which we calculate the derivatives of the internal force and the tangent stiffness at the element-level using the hyper-dual approach. Essentially, this scheme contributes to the use of second-order algorithms in structural optimizations.

As an incremental extension of the work of Jeffrey Alan Fike (2013), we aim to present the arithmetic operations of the scalar, vector, and tensor hyper-dual numbers. As it gathers together analytical accuracy and numerical generality, this derivative scheme might represent a useful black-box tool in many applications.

Additionally, we present a diagonal particularization of the sensitivity expressions in which the derivatives are calculated using a new hyper-dual variant. This approach provides an interesting balance among accuracy, generality, and computational costs.

It is important to highlight that since we describe a general sensitivity procedure, it can be easily adapted to different finite element formulations or structural optimization problems. As a result, this study could bring important benefits for the structural optimization procedures and ultimately for the development of new products.

1.3 DOCUMENT OVERVIEW

In Chapter 2, we present the hyper-dual procedure as a general derivative tool to evaluate tensor-valued functions of a vector argument. We describe the arithmetics involving the conversion of a given function into the hyper-dual form. Chapter 3 illustrates the use of hyper-dual numbers in constitutive models, particularly to calculate stress and tangent modulus in hyperelasticity. Also, we show a study on the processing time, in which we assess the effects of this derivative method and the model size on the computational effort.

To explore the applications of the hyper-dual scheme, we develop a general method comprising the element-level sensitivity analysis in Chapter 4. We introduce the role of this differentiation scheme to evaluate the tensor-valued functions found in the sensitivity expressions considering nonlinear structures. Additionally, we particularize the method in order to focus on the diagonal terms of the Hessian.

In Chapter 5, we discuss the findings of the aforementioned chapters. Finally, we state the conclusions of this research in Chapter 6.

Additionally, we provide complementary contents regarding hyper-dual numbers in Appendix A, including detailed examples of the hyper-dual conversion and their corresponding computer codes. To illustrate the hyper-dual sensitivity method, we show a case study in Appendix C, in which we evaluate the second-order displacement sensitivity of a truss structure. In this context, we review the expressions of the nonlinear truss formulation in Appendix B.

1.3.1 Highlights

We summarize the highlights of this study as follows:

- Tensor hyper-dual numbers (Chapter 2)
 - The tensor hyper-dual scheme provides accurate second-order derivative calculations of scalar, vector, and tensor-valued functions.
 - With modest modifications, a variant formulation focuses on the diagonal terms of the Hessian, reducing the computational effort.
 - The implementation considering the operator overloading defines the hyper-dual procedure as a forward mode automatic differentiation tool.
- Hyper-dual hyperelasticity (Chapter 3)
 - The hyper-dual form of the strain energy function provides exact values of stress and tangent modulus.
 - Using the hyper-dual scheme, the processing time of the material subroutine increased by a factor of 4, compared to the analytical reference.
 - Considering large models, the overall performance of the subroutine with the hyper-dual scheme is equivalent to the analytical expressions.

- Hyper-dual sensitivity analysis (Chapter 4)
 - The hyper-dual sensitivity method automatically returns exact Hessian information, useful in optimization studies.
 - Assuming a nonlinear structure, it involves the hyper-dual conversion of the internal force and the tangent stiffness of a given element.
 - The diagonal hyper-dual formulation is convenient to obtain the diagonal Hessian in linear structures.
 - The hyper-dual sensitivity is a general method that can adapt itself to various finite element formulations and design problems.

1.3.2 Scientific papers

These related works were recommended for publication:

- ENDO, Vitor T.; FANCELLO, Eduardo A.; MUÑOZ-ROJAS, Pablo A. **A first implementation of geometrically nonlinear hyperelastic truss elements using hyper-dual numbers.** In: 6th International Symposium on Solid Mechanics (MECSOL 2017). Joinville, Brazil: ABCM, 2017
- ENDO, Vitor T.; FANCELLO, Eduardo A.; MUÑOZ-ROJAS, Pablo A. **A study on the computational effort of hyper-dual numbers to evaluate derivatives in geometrically nonlinear hyperelastic trusses.** *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 2020.

1.3.3 Manuscript submission

We submitted the contents of this research, still under review, as follows:

- ENDO, Vitor T.; FANCELLO, Eduardo A.; MUÑOZ-ROJAS, Pablo A. **Hyper-dual sensitivity analysis: an automatic second-order evaluation in nonlinear problems.**
- ENDO, Vitor T.; FANCELLO, Eduardo A.; MUÑOZ-ROJAS, Pablo A. **Second-order design sensitivity analysis using diagonal hyper-dual numbers.**

2 TENSOR HYPER-DUAL NUMBERS

In this chapter, we present the theoretical framework regarding the use of hyper-dual numbers as an automatic differentiation tool. In Section 2.1, we review numerical methods for the derivative calculation of scalar-valued functions. In Section 2.2, we present the definition of tensor hyper-dual numbers, allowing a second-order evaluation of tensor-valued functions. In addition, we present a diagonal particularization of tensor hyper-dual numbers in Section 2.3. In this formulation, aiming for a reduction in terms of the computational costs, we intentionally neglect the calculation of the off-diagonal terms.

2.1 NUMERICAL PROCEDURES FOR THE DERIVATIVE CALCULATION

Although analytical derivative expressions represent the epitome of accuracy, this method intrinsically lacks adaptability. To overcome such an issue, numerical techniques are often applied in engineering problems by providing approximate derivative calculations. In this context, many schemes are derived from the Taylor series expansion indicated by

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \dots, \quad (1)$$

wherein the summation terms contain the derivatives represented as $f'(x)$, $f''(x)$, and $f'''(x)$. Note that if a finite number of summation terms is considered, truncation errors are introduced in the results.

In the finite difference method, the first-order derivative is obtained considering a real perturbation h and neglecting higher-order terms. In this case, $\mathcal{O}_{FD}(h^2)$ represents the omitted terms in the power series. Therefore, using forward finite differences, we can calculate the derivative as given by

$$f'_{FD}(x) \approx \frac{f(x+h) - f(x)}{h}. \quad (2)$$

It should be noted that round-off errors occur during the subtraction operation of similar values, which is related to arithmetic operations with a finite number of digits (BURDEN; FAIRES, 2010). Besides truncation error, it is well-established that this method is severely affected by the perturbation values.

Alternatively, the complex variable method improves the derivative calculation, as this scheme applies the perturbation in the imaginary axis denoted by i . Using the complex number property $i^2 = -1$, the Taylor expansion becomes

$$f(x+ih) = f(x) + ihf'(x) - \frac{h^2}{2!}f''(x) - \frac{ih^3}{3!}f'''(x) + \dots. \quad (3)$$

Noting that the operator \Im takes only the imaginary part of the complex function, we obtain the derivative using

$$f'_{CS}(x) \approx \frac{\Im[f(x+ih)]}{h}, \quad (4)$$

which neglects higher-order terms, such that $\mathcal{O}_{CS}(h^3)$. Although this procedure still suffers from truncation errors, it enables very small perturbation values and does not suffer from round-off errors related to the subtraction operation. Moreover, by comparing $\mathcal{O}_{FD}(h^2)$ with $\mathcal{O}_{CS}(h^3)$, we can find that the complex-step method provides a better approximation. Geovane A. Haveroth and Muñoz-Rojas (2016) investigated the use of complex variables to perform sensitivity analysis and reported perturbation factors as low as 10^{-300} .

In this context, it is important to highlight that Lantoine et al. (2012) described the multicomplex arithmetic to obtain partial derivatives of any order. By using multicomplex numbers, a multi-dimensional generalization of complex numbers, it is possible to retrieve higher-order information exact to machine precision.

Particularly, as reviewed by Behr et al. (2019), dual numbers consider another definition for the non-real part represented by ϵ , a nilpotent number. Using the property $\epsilon^2 = 0$, the Taylor series expansion is given by

$$f(x + h\epsilon) = f(x) + \epsilon h f'(x). \quad (5)$$

In this particular case, it should be noted that the expansion is exact and does not neglect any term. Yu and Blair (2013) found that this procedure is even more accurate and efficient than the complex-step automatic differentiation. The first-order derivative using the dual number scheme is obtained as

$$f'(x) = \frac{\Im[f(x + h\epsilon)]}{h}, \quad (6)$$

which is free from both truncation and round-off errors. Nevertheless, this method is limited to calculate first-order derivatives.

Jeffrey Alan Fike (2013) studied hyper-dual numbers as a derivative tool by considering ϵ_1 , ϵ_2 , and $\epsilon_1\epsilon_2$ as non-real parts. This scheme allows the evaluation of first and second-order derivatives (FIKE, J. A.; ALONSO, 2011; FIKE, J. A. et al., 2011). Particularly, using binomial theorem statements and the properties indicated in Tab. 1, the Taylor series expansion becomes

$$f(x + h(\epsilon_1 + \epsilon_2)) = f(x) + h\epsilon_1 f'(x) + h\epsilon_2 f'(x) + h^2\epsilon_1\epsilon_2 f''(x). \quad (7)$$

Similarly as observed in the case of dual numbers, it is noteworthy that there are no truncation errors in the final expression.

As shown in Eq. 7, a hyper-dual variable consists of four parts, being one real and three terms corresponding to ϵ_1 , ϵ_2 , and $\epsilon_1\epsilon_2$. The real term represents the function itself and the non-real terms comprise the derivative information; axes ϵ_1 and ϵ_2 contain the first-order derivatives, whereas axis $\epsilon_1\epsilon_2$ contains the second-order derivative information. In short, once a given function is properly converted into hyper-dual numbers, we can extract the derivatives using the mathematical operators \Im_{ϵ_1} , \Im_{ϵ_2} , and $\Im_{\epsilon_1\epsilon_2}$, as indicated by

$$f'(x) = \frac{\Im_{\epsilon_1}[f(x + h(\epsilon_1 + \epsilon_2))]}{h} = \frac{\Im_{\epsilon_2}[f(x + h(\epsilon_1 + \epsilon_2))]}{h} \quad \text{and} \quad (8)$$

$$f''(x) = \frac{\Im_{\epsilon_1\epsilon_2}[f(x + h(\epsilon_1 + \epsilon_2))]}{h^2}. \quad (9)$$

Moreover, inasmuch as this method is insensitive to the perturbation value (FIKE, Jeffrey Alan, 2013; TANAKA et al., 2015), we can consider $h = 1$ for the sake of simplicity.

Table 1 – Fundamental properties of hyper-dual numbers.

$\epsilon_1^2 = 0$	$\epsilon_2^2 = 0$	$\epsilon_1\epsilon_2 = \epsilon_2\epsilon_1$
Source – Jeffrey Alan Fike (2013).		

In fact, the hyper-dual scheme can be considered as an automatic differentiation tool, which comprises the techniques regarding the application of the chain rule in a computer program. Also denoted as algorithmic differentiation, this strategy commonly evaluates the sequence of arithmetic operations to provide the derivatives of arbitrary functions. As cited by Atılım Günes Baydin et al. (2017), automatic differentiation is one of the computational methods for the derivative calculation, as summarized by:

1. Manually working out derivatives and coding them;
2. Numerical differentiation using finite difference approximations;
3. Symbolic differentiation using expression manipulation in computer algebra systems;
4. Automatic differentiation;

Dedicated contents and discussions regarding the latter topic can be obtained in the works of Gay (2005), C. H. Bischof and H. M. Bücker (2000) and Bischof et al. (2002); Naumann (2011) also brings a comprehensive description of algorithmic differentiation. Furthermore, we find an important source of information on Autodiff website. Ruffwind (2016) states the following:

"The important realization that leads to automatic differentiation is the fact that even the biggest and most complicated program must be built from a small set of primitive operations such as addition, multiplication, or trigonometric functions."

Particularly, we can categorize the hyper-dual scheme into the forward mode of algorithmic differentiation. In this case, the chain rule occurs from inside to outside i.e. we fix the independent variable and evaluate a sequence of operations to obtain the function and its derivative (NEIDINGER, 2010). Conversely, the reverse mode evaluates the derivatives of dependent variables with respect to intermediate variables and propagates the chain rule backward; in this case, we start from the output variables. Nørgaard et al. (2017), Ruffwind (2016), Leal et al. (2018) and Griewank and Walther (2008) presented introductory examples to explain these modes.

According to Griewank and Walther (2008), the reverse mode is particularly efficient in problems containing a single objective function depending upon millions of inputs, for instance. This a common case in machine learning applications. Nevertheless, Dunham (2017) stated

that the choice of the ideal method is ultimately problem-dependent and a matter of user preference. Thus a thorough investigation is necessary when introducing new methods and applications.

Automatic differentiation has been reported in many applications (KONTAK et al., 2020; MEHMOOD; OCHS, 2020; MARGOSSIAN, 2019; MANZYUK et al., 2019; PEÑUÑURI et al., 2020). Dilgen et al. (2018) discussed the application of automatic differentiation to calculate exact sensitivities of turbulent flow topology optimization problems; this tool allowed easy accommodation of new physics and turbulence closure models with minimal implementation effort. In a study concerning automatic differentiation in topology optimization (NØRGAARD et al., 2017), the researchers stated that this technique automates the calculation by software, allowing developers to focus on the derivative solution, rather than the derivation process.

The hyper-dual procedure is a general and accurate derivative tool to evaluate both first and second-order derivatives. Due to these distinguished qualities, this scheme represents an important tool in engineering applications. We propose to extend the hyper-dual formulation to tensor-valued functions, allowing the use of this differentiation method in a plethora of scientific fields.

2.2 TENSOR HYPER-DUAL FORMULATION

We can consider the hyper-dual procedure to automatically calculate the derivatives of tensor-valued functions, which are commonly found in many numerical methods, such as finite elements. This differentiation method provides full Hessian calculation, which includes the mixed second-order derivative terms. Let us consider $w \in \mathbb{R}$, $\mathbf{q} \in \mathbb{R}^n$, and $\mathbf{K} \in \mathbb{R}^{n \times n}$ as scalar, vector, and second-order tensor valued functions of a vector argument $\mathbf{s} \in \mathbb{R}^N$, respectively.

The corresponding hyper-dual representation is presented as follows¹:

Scalar function (example: strain energy potential)

$$w = w_0 + \mathbf{w}_1\epsilon_1 + \mathbf{w}_2\epsilon_2 + \mathbf{w}_3\epsilon_1\epsilon_2. \quad (10)$$

Vector function (example: internal force vector)

$$\mathbf{q} = \mathbf{q}_0 + \mathbf{q}_1\epsilon_1 + \mathbf{q}_2\epsilon_2 + \mathbf{q}_3\epsilon_1\epsilon_2. \quad (11)$$

Tensor function (example: tangent stiffness matrix)

$$\mathbf{K} = \mathbf{K}_0 + \mathbf{K}_1\epsilon_1 + \mathbf{K}_2\epsilon_2 + \mathbf{K}_3\epsilon_1\epsilon_2. \quad (12)$$

¹ We presented the scalar, vector, and tensor hyper-dual numbers using lowercase, boldface lowercase and boldface uppercase sans serif fonts, respectively.

Tab. 2 summarizes the definition of each term and the derivative information that is obtained from the hyper-dual representation. Assuming the application of hyper-dual numbers in structural mechanics, the indexes are represented by $i = 1 \dots n$, $j = 1 \dots n$, $k = 1 \dots N$, $l = 1 \dots N$, where n is the number of degrees of freedom and N is the number of design variables. These indexes are consistent with the expected tensor dimensions related to the derivative information; for instance, the second-order derivative of the second-order tensor $[K_0]_{ij}$ is the fourth order tensor represented by $[K_3]_{ijkl}$.

Table 2 – Derivative information obtained from the hyper-dual form.

Eq.	Hyper-dual	Real part \Re	\Im_{ϵ_1}	Non-real parts \Im_{ϵ_2}	$\Im_{\epsilon_1 \epsilon_2}$	
10	Scalar	w	$w_0 \in \mathbb{R}$ w_0	$\mathbf{w}_1 \in \mathbb{R}^N$ $[w_1]_k = \frac{\partial w}{\partial s_k}$	$\mathbf{w}_2 \in \mathbb{R}^N$ $[w_2]_l = \frac{\partial w}{\partial s_l}$	$\mathbf{w}_3 \in \mathbb{R}^{N \times N}$ $[w_3]_{kl} = \frac{\partial^2 w}{\partial s_l \partial s_k}$
11	Vector	\mathbf{q}	$\mathbf{q}_0 \in \mathbb{R}^n$ $[q_0]_i$	$\mathbf{q}_1 \in \mathbb{R}^n \times \mathbb{R}^N$ $[q_1]_{ik} = \frac{\partial q_i}{\partial s_k}$	$\mathbf{q}_2 \in \mathbb{R}^n \times \mathbb{R}^N$ $[q_2]_{il} = \frac{\partial q_i}{\partial s_l}$	$\mathbf{q}_3 \in \mathbb{R}^n \times \mathbb{R}^{N \times N}$ $[q_3]_{ikl} = \frac{\partial^2 q_i}{\partial s_l \partial s_k}$
12	Tensor	\mathbf{K}	$\mathbf{K}_0 \in \mathbb{R}^{n \times n}$ $[K_0]_{ij}$	$\mathbf{K}_1 \in \mathbb{R}^{n \times n} \times \mathbb{R}^N$ $[K_1]_{ijk} = \frac{\partial K_{ij}}{\partial s_k}$	$\mathbf{K}_2 \in \mathbb{R}^{n \times n} \times \mathbb{R}^N$ $[K_2]_{ijl} = \frac{\partial K_{ij}}{\partial s_l}$	$\mathbf{K}_3 \in \mathbb{R}^{n \times n} \times \mathbb{R}^{N \times N}$ $[K_3]_{ijkl} = \frac{\partial^2 K_{ij}}{\partial s_k \partial s_l}$

Source – Own author.

2.2.1 Mathematical operations

To obtain its hyper-dual form, we must apply a sequence of operators that describe a given function. Hence, the hyper-dual conversion requires the mathematical definition of many arithmetic operations, including addition, subtraction, multiplication, and so on. In this work, we describe the arithmetic operations involving scalar, vector and tensor hyper-dual numbers, as defined in Eqs. 10, 11 and 12, respectively.

We found comprehensive documentation regarding the properties of scalar hyper-dual numbers (FIKE, Jeffrey Alan, 2013); however, we could not find a corresponding work considering vector or tensor hyper-dual variables. Thus, we define a set of rules concerning tensor hyper-dual numbers in this section based on an adaptation of the operators of scalar hyper-dual numbers. Additionally, unlike other related works, we present the expressions in index notation, which is a convenient format for the computer implementation of higher-order tensors.

We conduct the addition operation by considering that both arguments present a consistent tensor order. Thus, for each hyper-dual variable, this operator is presented by:

	\mathfrak{R}	$\mathfrak{S}_{\epsilon_1}$	$\mathfrak{S}_{\epsilon_2}$	$\mathfrak{S}_{\epsilon_1\epsilon_2}$	
$\mathbf{a} + \mathbf{b}$	$a_0 + b_0$	$[a_1]_k + [b_1]_k$	$[a_2]_l + [b_2]_l$	$[a_3]_{kl} + [b_3]_{kl}$	(13)
$\mathbf{q} + \mathbf{f}$	$[q_0]_i + [f_0]_i$	$[q_1]_{ik} + [f_1]_{ik}$	$[q_2]_{il} + [f_2]_{il}$	$[q_3]_{ikl} + [f_3]_{ikl}$	(14)
$\mathbf{K} + \mathbf{T}$	$[K_0]_{ij} + [T_0]_{ij}$	$[K_1]_{ijk} + [T_1]_{ijk}$	$[K_2]_{ijl} + [T_2]_{ijl}$	$[K_3]_{ijkl} + [T_3]_{ijkl}$	(15)

As for the mathematical notation, let us consider \mathbf{a} and \mathbf{b} as scalar hyper-dual variables. The addition operation concerning these variables (Eq. 13) is represented by

$$\mathbf{a} + \mathbf{b} = \overbrace{(a_0 + b_0)}^{\mathfrak{R}[\mathbf{a}+\mathbf{b}]} + \overbrace{(\mathbf{a}_1 + \mathbf{b}_1)}^{\mathfrak{S}_{\epsilon_1}[\mathbf{a}+\mathbf{b}]} \epsilon_1 + \overbrace{(\mathbf{a}_2 + \mathbf{b}_2)}^{\mathfrak{S}_{\epsilon_2}[\mathbf{a}+\mathbf{b}]} \epsilon_2 + \overbrace{(\mathbf{a}_3 + \mathbf{b}_3)}^{\mathfrak{S}_{\epsilon_1\epsilon_2}[\mathbf{a}+\mathbf{b}]} \epsilon_1\epsilon_2, \quad (16)$$

where \mathfrak{R} , $\mathfrak{S}_{\epsilon_1}$, $\mathfrak{S}_{\epsilon_2}$, and $\mathfrak{S}_{\epsilon_1\epsilon_2}$, denotes the extraction operation of a specific hyper-dual part. Alternatively, we can represent the calculation of each term as

$$\begin{aligned} \mathfrak{R}[\mathbf{a} + \mathbf{b}] &= a_0 + b_0, & \mathfrak{S}_{\epsilon_1}[\mathbf{a} + \mathbf{b}]_k &= [a_1]_k + [b_1]_k, \\ \mathfrak{S}_{\epsilon_2}[\mathbf{a} + \mathbf{b}]_l &= [a_2]_l + [b_2]_l, & \text{and} & \mathfrak{S}_{\epsilon_1\epsilon_2}[\mathbf{a} + \mathbf{b}]_{kl} &= [a_3]_{kl} + [b_3]_{kl}. \end{aligned} \quad (17)$$

Thus, it should be noted that each column indicated in Eq. 13 describes a certain part of the hyper-dual variable. In addition, we highlight that the results are consistent with the arguments in terms of tensor order.

The subtraction operation works analogously, i.e. the operator is applied on each hyper-dual term individually:

	\mathfrak{R}	$\mathfrak{S}_{\epsilon_1}$	$\mathfrak{S}_{\epsilon_2}$	$\mathfrak{S}_{\epsilon_1\epsilon_2}$	
$\mathbf{a} - \mathbf{b}$	$a_0 - b_0$	$[a_1]_k - [b_1]_k$	$[a_2]_l - [b_2]_l$	$[a_3]_{kl} - [b_3]_{kl}$	(18)
$\mathbf{q} - \mathbf{f}$	$[q_0]_i - [f_0]_i$	$[q_1]_{ik} - [f_1]_{ik}$	$[q_2]_{il} - [f_2]_{il}$	$[q_3]_{ikl} - [f_3]_{ikl}$	(19)
$\mathbf{K} - \mathbf{T}$	$[K_0]_{ij} - [T_0]_{ij}$	$[K_1]_{ijk} - [T_1]_{ijk}$	$[K_2]_{ijl} - [T_2]_{ijl}$	$[K_3]_{ijkl} - [T_3]_{ijkl}$	(20)

As an adaptation of the scalar hyper-dual case (FIKE, Jeffrey Alan, 2013), the mathematical operator for function evaluations concerning hyper-dual variables is conducted in a general form as:

	\mathfrak{R}	$\mathfrak{S}_{\epsilon_1}$	$\mathfrak{S}_{\epsilon_2}$	$\mathfrak{S}_{\epsilon_1\epsilon_2}$	
$f(\mathbf{a})$	$f(a_0)$	$f'(a_0) [a_1]_k$	$f'(a_0) [a_2]_l$	$f'(a_0) [a_3]_{kl} + f''(a_0) [a_1]_k [a_2]_l$	(21)

Based on this general definition, we can describe the corresponding hyper-dual operations related to power, inverse, and square root, as given by:

	\Re	\Im_{ϵ_1}	\Im_{ϵ_2}	$\Im_{\epsilon_1\epsilon_2}$	
\mathbf{a}^n	a_0^n	$na_0^{n-1} [a_1]_k$	$na_0^{n-1} [a_2]_l$	$na_0^{n-1} [a_3]_{kl} + n(n-1)a_0^{n-2} [a_1]_k [a_2]_l$	(22)

$\frac{1}{\mathbf{a}}$	$\frac{1}{a_0}$	$-\frac{1}{a_0^2} [a_1]_k$	$-\frac{1}{a_0^2} [a_2]_l$	$-\frac{1}{a_0^2} [a_3]_{kl} + \frac{2}{a_0^3} [a_1]_k [a_2]_l$	(23)
------------------------	-----------------	----------------------------	----------------------------	---	------

$\sqrt{\mathbf{a}}$	$\sqrt{a_0}$	$\frac{1}{2\sqrt{a_0}} [a_1]_k$	$\frac{1}{2\sqrt{a_0}} [a_2]_l$	$\frac{1}{2\sqrt{a_0}} [a_3]_{kl} - \frac{1}{4(\sqrt{a_0})^3} [a_1]_k [a_2]_l$	(24)
---------------------	--------------	---------------------------------	---------------------------------	--	------

It should be noted that inverse and square root operators are in fact extensions of the power operator. Additionally, we can use Eq. 21 to obtain the expressions describing trigonometric operations, for instance.

Some product operation possibilities involving scalar and vector hyper-dual numbers are summarized as:

	\Re	\Im_{ϵ_1}	\Im_{ϵ_2}	$\Im_{\epsilon_1\epsilon_2}$	
\mathbf{ab}	a_0b_0	$a_0 [b_1]_k$ $+b_0 [a_1]_k$	$a_0 [b_2]_l$ $+b_0 [a_2]_l$	$a_0 [b_3]_{kl} + [a_1]_k [b_2]_l$ $+ [b_1]_k [a_2]_l + b_0 [a_3]_{kl}$	(25)

\mathbf{aq}	$a_0 [q_0]_i$	$a_0 [q_1]_{ik}$ $+ [q_0]_i [a_1]_k$	$a_0 [q_2]_{il}$ $+ [q_0]_i [a_2]_l$	$a_0 [q_3]_{ikl} + [a_1]_k [q_2]_{il}$ $+ [q_1]_{ik} [a_2]_l + [q_0]_i [a_3]_{kl}$	(26)
---------------	---------------	---	---	---	------

$\mathbf{q} \cdot \mathbf{f}$	$[q_0]_i [f_0]_i$	$[q_0]_i [f_1]_{ik}$ $+ [f_0]_i [q_1]_{ik}$	$[q_0]_i [f_2]_{il}$ $+ [f_0]_i [q_2]_{il}$	$[q_0]_i [f_3]_{ikl} + [q_1]_{ik} [f_2]_{il}$ $+ [f_1]_{ik} [q_2]_{il} + [f_0]_i [q_3]_{ikl}$	(27)
-------------------------------	-------------------	--	--	--	------

$\mathbf{q} \otimes \mathbf{f}$	$[q_0]_i [f_0]_j$	$[q_0]_i [f_1]_{jk}$ $+ [f_0]_j [q_1]_{ik}$	$[q_0]_i [f_2]_{jl}$ $+ [f_0]_j [q_2]_{il}$	$[q_0]_i [f_3]_{jkl} + [q_1]_{ik} [f_2]_{jl}$ $+ [f_1]_{jk} [q_2]_{il} + [f_0]_j [q_3]_{ikl}$	(28)
---------------------------------	-------------------	--	--	--	------

These expressions illustrate the higher computational costs associated with hyper-dual numbers when compared to an ordinary real operation. For instance, as observed in Eq. 25, a single product operation considering hyper-dual variables involves 9 multiplications and 5 additions. This effort becomes more evident when higher-order tensors take place, as observed in the operations regarding tensor hyper-dual numbers indicated by:

	\Re	\Im_{ϵ_1}	\Im_{ϵ_2}	$\Im_{\epsilon_1\epsilon_2}$	
aK	$a_0 [K_0]_{ij}$	$a_0 [K_1]_{ijk}$ + $[K_0]_{ij} [a_1]_k$	$a_0 [K_2]_{ijl}$ + $[K_0]_{ij} [a_2]_l$	$a_0 [K_3]_{ijkl}$ + $[a_1]_k [K_2]_{ijl}$ + $[K_1]_{ijk} [a_2]_l$ + $[K_0]_{ij} [a_3]_{kl}$	(29)

Ku	$[K_0]_{ij} [u_0]_j$	$[K_0]_{ij} [u_1]_{jk}$ + $[u_0]_j [K_1]_{ijk}$	$[K_0]_{ij} [u_2]_{jl}$ + $[u_0]_j [K_2]_{ijl}$	$[K_0]_{ij} [u_3]_{jkl}$ + $[K_1]_{ijk} [u_2]_{jl}$ + $[u_1]_{jk} [K_2]_{ijl}$ + $[u_0]_j [K_3]_{ijkl}$	(30)
-----------	----------------------	--	--	--	------

KT	$[K_0]_{ij} [T_0]_{jm}$	$[K_0]_{ij} [T_1]_{jmk}$ + $[T_0]_{jm} [K_1]_{ijk}$	$[K_0]_{ij} [T_2]_{jml}$ + $[T_0]_{jm} [K_2]_{ijl}$	$[K_0]_{ij} [T_3]_{jmkl}$ + $[K_1]_{ijk} [T_2]_{jml}$ + $[T_1]_{jmk} [K_2]_{ijl}$ + $[T_0]_{jm} [K_3]_{ijkl}$	(31)
-----------	-------------------------	--	--	--	------

Additionally, the transpose operator is given by:

	\Re	\Im_{ϵ_1}	\Im_{ϵ_2}	$\Im_{\epsilon_1\epsilon_2}$	
K^T	$[K_0]_{ji}$	$[K_1]_{jik}$	$[K_2]_{jil}$	$[K_3]_{jikl}$	(32)

Particularly, as for the addition and the subtraction operations, if the arguments of these mathematical operations are real and hyper-dual numbers, only the real term is affected. Considering the hyper-dual representation of real variables², these operations are defined as:

	\Re	\Im_{ϵ_1}	\Im_{ϵ_2}	$\Im_{\epsilon_1\epsilon_2}$	
a + c	$a_0 + c_0$	$[a_1]_k$	$[a_2]_l$	$[a_3]_{kl}$	(34)

q + c	$[q_0]_i + [c_0]_i$	$[q_1]_{ik}$	$[q_2]_{il}$	$[q_3]_{ikl}$	(35)
--------------	---------------------	--------------	--------------	---------------	------

K + C	$[K_0]_{ij} + [C_0]_{ij}$	$[K_1]_{ijk}$	$[K_2]_{ijl}$	$[K_3]_{ijkl}$	(36)
--------------	---------------------------	---------------	---------------	----------------	------

a - c	$a_0 - c_0$	$[a_1]_k$	$[a_2]_l$	$[a_3]_{kl}$	(37)
--------------	-------------	-----------	-----------	--------------	------

q - c	$[q_0]_i - [c_0]_i$	$[q_1]_{ik}$	$[q_2]_{il}$	$[q_3]_{ikl}$	(38)
--------------	---------------------	--------------	--------------	---------------	------

K - C	$[K_0]_{ij} - [C_0]_{ij}$	$[K_1]_{ijk}$	$[K_2]_{ijl}$	$[K_3]_{ijkl}$	(39)
--------------	---------------------------	---------------	---------------	----------------	------

² For each function type, the hyper-dual representation of real variables is given by

$$c = c_0 + \mathbf{0}\epsilon_1 + \mathbf{0}\epsilon_2 + \mathbf{0}\epsilon_1\epsilon_2, \quad \mathbf{c} = [c_0]_i + \mathbf{0}\epsilon_1 + \mathbf{0}\epsilon_2 + \mathbf{0}\epsilon_1\epsilon_2 \quad \text{and} \quad \mathbf{C} = [C_0]_{ij} + \mathbf{0}\epsilon_1 + \mathbf{0}\epsilon_2 + \mathbf{0}\epsilon_1\epsilon_2. \quad (33)$$

Additionally, the product operations regarding a hyper-dual and real numbers can be presented as particular cases of the previously mentioned expressions. Therefore, it can be shown that the real number operates over all hyper-dual terms, as summarized by:

$$\begin{array}{cccccc} & \mathfrak{R} & \mathfrak{S}_{\epsilon_1} & \mathfrak{S}_{\epsilon_2} & \mathfrak{S}_{\epsilon_1\epsilon_2} & \\ \hline \mathbf{ca} & c_0 a_0 & c_0 [a_1]_k & c_0 [a_2]_l & c_0 [a_3]_{kl} & (40) \end{array}$$

$$\mathbf{cq} \quad c_0 [q_0]_i \quad c_0 [q_1]_{ik} \quad c_0 [q_2]_{il} \quad c_0 [q_3]_{ikl} \quad (41)$$

$$\mathbf{cK} \quad c_0 [K_0]_{ij} \quad c_0 [K_1]_{ijk} \quad c_0 [K_2]_{ijl} \quad c_0 [K_3]_{ijkl} \quad (42)$$

$$\mathbf{ca} \quad [c_0]_i a_0 \quad [c_0]_i [a_1]_k \quad [c_0]_i [a_2]_l \quad [c_0]_i [a_3]_{kl} \quad (43)$$

$$\mathbf{c} \cdot \mathbf{q} \quad [c_0]_i [q_0]_i \quad [c_0]_i [q_1]_{ik} \quad [c_0]_i [q_2]_{il} \quad [c_0]_i [q_3]_{ikl} \quad (44)$$

$$\mathbf{c} \otimes \mathbf{f} \quad [c_0]_i [f_0]_j \quad [c_0]_i [f_1]_{jk} \quad [c_0]_i [f_2]_{jl} \quad [c_0]_i [f_3]_{jkl} \quad (45)$$

$$\mathbf{cK} \quad [c_0]_i [K_0]_{ij} \quad [c_0]_i [K_1]_{ijk} \quad [c_0]_i [K_2]_{ijl} \quad [c_0]_i [K_3]_{ijkl} \quad (46)$$

$$\mathbf{Ca} \quad [C_0]_{ij} a_0 \quad [C_0]_{ij} [a_1]_k \quad [C_0]_{ij} [a_2]_l \quad [C_0]_{ij} [a_3]_{kl} \quad (47)$$

$$\mathbf{Cu} \quad [C_0]_{ij} [u_0]_j \quad [C_0]_{ij} [u_1]_{jk} \quad [C_0]_{ij} [u_2]_{jl} \quad [C_0]_{ij} [u_3]_{jkl} \quad (48)$$

$$\mathbf{CT} \quad [C_0]_{ij} [T_0]_{jm} \quad [C_0]_{ij} [T_1]_{jmk} \quad [C_0]_{ij} [T_2]_{jml} \quad [C_0]_{ij} [T_3]_{jmkl} \quad (49)$$

It is important to highlight that the aforementioned operations consider the summation convention, which occurs when an index variable is found twice in a single term (MALVERN, 1969). We presented the arithmetic operations in index notation form to facilitate further computational implementation. This representation ensures the identification of the indexes related to the loop operation in the code.

2.2.2 Hyper-dual conversion

In summary, using the logic of the forward mode of algorithmic differentiation, the hyper-dual conversion requires the following procedures:

- Step 1. Define the design variables as a hyper-dual number with unit perturbation in the axes ϵ_1 and ϵ_2 .
- Step 2. Apply the hyper-dual operators to obtain the corresponding representation of the original function.
- Step 3. Extract the non-real parts to obtain the derivative information.

As an introductory material, we explain the steps regarding the hyper-dual conversion using a scalar-valued function in Section A.1.2. We show how to follow the conversion steps and apply the aforementioned arithmetics to obtain the gradient and the Hessian matrix. Thus, that section comprises a preparatory content for the evaluation of tensor-valued functions using hyper-dual numbers.

2.2.3 Example

In this section, we illustrate the hyper-dual representation of a tensor-valued function of a vector argument. We present the procedures to convert the stiffness matrix of a truss element into hyper-dual numbers. It is important to highlight that the example presented in this section is deliberately simplistic and should not be considered as a practical application of automatic differentiation. We aim to illustrate the hyper-dual conversion steps introduced in Section 2.2.2. Thus, we can clearly exemplify the use of the expressions indicated in Section 2.2.1, which describe the arithmetic operations of hyper-dual numbers.

Considering a linear elastic case, the stiffness matrix \mathbf{K} , written in the local coordinate system, can be obtained using the following expression:

$$\mathbf{K} = \mathbf{B}^\top E\mathbf{B}AL = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad (50)$$

where E , A and L are scalar variables that represent the elastic modulus of the material, the cross-sectional area and the length of an element, respectively; \mathbf{B} is the strain-displacement matrix, presented in this case as a row vector indicated by

$$\mathbf{B} = \left\{ -1/L \quad 1/L \right\}. \quad (51)$$

Let us consider that one requires both first and second-order derivatives of the stiffness matrix with respect to the design variables. The latter is organized as a vector indicated by

$$\mathbf{s} = \left\{ \begin{matrix} A \\ L \end{matrix} \right\}. \quad (52)$$

Hence, the expected derivative results are represented by

$$\frac{\partial \mathbf{K}}{\partial \mathbf{s}} = \left[\begin{array}{cc|cc} \frac{\partial K_{11}}{\partial A} & \frac{\partial K_{12}}{\partial A} & \frac{\partial K_{11}}{\partial L} & \frac{\partial K_{12}}{\partial L} \\ \frac{\partial K_{21}}{\partial A} & \frac{\partial K_{22}}{\partial A} & \frac{\partial K_{21}}{\partial L} & \frac{\partial K_{22}}{\partial L} \end{array} \right] \quad \text{and} \quad \frac{\partial^2 \mathbf{K}}{\partial \mathbf{s}^2} = \left[\begin{array}{cc|cc} \frac{\partial^2 K_{11}}{\partial A \partial A} & \frac{\partial^2 K_{12}}{\partial A \partial A} & \frac{\partial^2 K_{11}}{\partial L \partial A} & \frac{\partial^2 K_{12}}{\partial L \partial A} \\ \frac{\partial^2 K_{21}}{\partial A \partial A} & \frac{\partial^2 K_{22}}{\partial A \partial A} & \frac{\partial^2 K_{21}}{\partial L \partial A} & \frac{\partial^2 K_{22}}{\partial L \partial A} \\ \frac{\partial^2 K_{11}}{\partial A \partial L} & \frac{\partial^2 K_{12}}{\partial A \partial L} & \frac{\partial^2 K_{11}}{\partial L \partial L} & \frac{\partial^2 K_{12}}{\partial L \partial L} \\ \frac{\partial^2 K_{21}}{\partial A \partial L} & \frac{\partial^2 K_{22}}{\partial A \partial L} & \frac{\partial^2 K_{21}}{\partial L \partial L} & \frac{\partial^2 K_{22}}{\partial L \partial L} \end{array} \right]. \quad (53)$$

It should be noted that the derivative information refers to higher-order tensors; to facilitate their presentation, in this work we consider a matrix representation. As for the first-order derivative, we can divide the matrix into two blocks, each one referring to the derivative terms with respect to a design variable. The matrix representation of the second-order derivative can be organized in terms of quadrants; the second and the fourth quadrant represent the diagonal terms, whereas the first and the third quadrant represent the mixed derivative terms.

To obtain the required derivative information, we propose the use of hyper-dual numbers. For this purpose, we describe how to obtain the hyper-dual form of the stiffness matrix indicated in Eq. 50, which can be written as

$$\mathbf{K} = eal(\mathbf{b} \otimes \mathbf{b}), \quad (54)$$

where \mathbf{e} , \mathbf{a} , \mathbf{l} and \mathbf{b} are the hyper-dual form of E , A , L and \mathbf{B} , respectively. The latter is a vector-valued function, whereas the other ones are scalar-valued functions.

As discussed in Section 2.2.2, a hyper-dual conversion typically requires 3 main steps. The first one refers to the definition of the design variables as scalar hyper-dual numbers. Thus, we define the hyper-dual variables \mathbf{a} and \mathbf{l} as

$$\mathbf{a} = A + \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \epsilon_1 \epsilon_2 \quad \text{and} \quad (55)$$

$$\mathbf{l} = L + \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \epsilon_1 \epsilon_2. \quad (56)$$

It should be noted that a unit perturbation was applied in axes ϵ_1 and ϵ_2 .

The second step is related to the manipulation of variables to obtain the hyper-dual form indicated in Eq. 54. Using Eq. 25, which describes the product of scalar hyper-dual numbers, we obtain

$$\mathbf{al} = AL + \begin{Bmatrix} L \\ A \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} L \\ A \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \epsilon_1 \epsilon_2. \quad (57)$$

Similarly, the product of a real number³ and a scalar hyper-dual number results in

$$\mathbf{eal} = EAL + \begin{Bmatrix} EL \\ EA \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} EL \\ EA \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & E \\ E & 0 \end{bmatrix} \epsilon_1 \epsilon_2. \quad (59)$$

To obtain the hyper-dual form of the strain-displacement matrix, some preliminary procedures are required. Using Eq. 23, we calculate the inverse of the element length as

$$\frac{1}{\mathbf{l}} = \frac{1}{L} + \begin{Bmatrix} 0 \\ -1/L^2 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 0 \\ -1/L^2 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 \\ 0 & 2/L^3 \end{bmatrix} \epsilon_1 \epsilon_2. \quad (60)$$

As defined in Eq. 26, we obtain the hyper-dual form of the strain-displacement by applying the product of a scalar hyper-dual and a real vector⁴

$$\mathbf{b} = \begin{Bmatrix} -1/L \\ 1/L \end{Bmatrix} + \begin{bmatrix} 0 & 1/L^2 \\ 0 & -1/L^2 \end{bmatrix} \epsilon_1 + \begin{bmatrix} 0 & 1/L^2 \\ 0 & -1/L^2 \end{bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 & 0 & -2/L^3 \\ 0 & 0 & 0 & 2/L^3 \end{bmatrix} \epsilon_1 \epsilon_2. \quad (62)$$

³ The hyper-dual form of the real variable E is given by

$$\mathbf{e} = E + \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \epsilon_1 \epsilon_2. \quad (58)$$

⁴ In this case, we defined the intermediate hyper-dual variable \mathbf{t} , such that $\mathbf{b} = \frac{1}{\mathbf{l}} \mathbf{t}$.

$$\mathbf{t} = \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \epsilon_1 + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \epsilon_1 \epsilon_2. \quad (61)$$

It should be noted that the hyper-dual term $\epsilon_1\epsilon_2$ is a third-order tensor represented by a matrix. As we apply the tensor product of two vector variables according to Eq. 28, we obtain a tensor hyper-dual variable, as given by

$$\mathbf{b} \otimes \mathbf{b} = \begin{bmatrix} 1/L^2 & -1/L^2 \\ -1/L^2 & 1/L^2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & -2/L^3 & 2/L^3 \\ 0 & 0 & 2/L^3 & -2/L^3 \end{bmatrix} \epsilon_1 + \begin{bmatrix} 0 & 0 & -2/L^3 & 2/L^3 \\ 0 & 0 & 2/L^3 & -2/L^3 \end{bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 6/L^4 & -6/L^4 \\ 0 & 0 & -6/L^4 & 6/L^4 \end{bmatrix} \epsilon_1\epsilon_2. \quad (63)$$

We complete the hyper-dual conversion of the stiffness matrix using Eq. 29; the result of the multiplication of Eqs. 59 and 63 is

$$\mathbf{K} = \begin{bmatrix} EA/L & -EA/L \\ -EA/L & EA/L \end{bmatrix} + \begin{bmatrix} E/L & -E/L & -EA/L^2 & EA/L^2 \\ -E/L & E/L & EA/L^2 & -EA/L^2 \end{bmatrix} \epsilon_1 + \begin{bmatrix} E/L & -E/L & -EA/L^2 & EA/L^2 \\ -E/L & E/L & EA/L^2 & -EA/L^2 \end{bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 & -E/L^2 & E/L^2 \\ 0 & 0 & E/L^2 & -E/L^2 \\ -E/L^2 & E/L^2 & 2EA/L^3 & -2EA/L^3 \\ E/L^2 & -E/L^2 & -2EA/L^3 & 2EA/L^3 \end{bmatrix} \epsilon_1\epsilon_2. \quad (64)$$

The last step involves extracting specific terms from the hyper-dual variable to obtain the desired derivative information. For this purpose, we use the operator \mathfrak{S} , as indicated by

$$\mathfrak{S}_{\epsilon_1} [\mathbf{K}] = \begin{bmatrix} E/L & -E/L & -EA/L^2 & EA/L^2 \\ -E/L & E/L & EA/L^2 & -EA/L^2 \end{bmatrix} \quad \text{and} \quad (65)$$

$$\mathfrak{S}_{\epsilon_1\epsilon_2} [\mathbf{K}] = \begin{bmatrix} 0 & 0 & -E/L^2 & E/L^2 \\ 0 & 0 & E/L^2 & -E/L^2 \\ -E/L^2 & E/L^2 & 2EA/L^3 & -2EA/L^3 \\ E/L^2 & -E/L^2 & -2EA/L^3 & 2EA/L^3 \end{bmatrix}. \quad (66)$$

The derivative results, extracted from the hyper-dual representation, are consistent with the analytical predictions. Thus, we state that the hyper-dual scheme allows a proper evaluation of first and second-order derivatives of tensor-valued functions. It is important to highlight that the use of diagonal hyper-dual numbers in design sensitivity analysis is not restricted or intended to truss elements. We bring this example to facilitate the understanding of the transformation process of a second-order tensor-valued function into hyper-dual numbers. Since it shows a limited number of degrees of freedom, the matrix representation of higher-order tensors is favorable.

2.2.4 Computer implementation

Following the study of Tanaka et al. (2015) and Yu and Blair (2013), we implemented the hyper-dual operators covered in Section 2.2.1 using the operator overloading technique. In this work, in addition to scalar hyper-dual numbers, we also define vector and tensor hyper-dual variables. This implementation scheme allows the use of the same syntax for different cases depending on the arguments and outputs of each arithmetic operation. Therefore, this strategy reduces significant modifications of the original real-valued code when compared to function or subroutine calls. We can find additional information concerning the operator overloading in the literature (CHIVERS; SLEIGHTHOLME, 2011; PRESS, 1992).

Initially, the hyper-dual variables are defined in a `Fortran` module as a new data type with a proper definition for scalar, vector and tensor variables, as presented in Eqs. 10, 11 and 12, respectively. Functions and subroutines describing the mathematical arithmetics of each operator are introduced according to the expressions indicated in Section 2.2.1. Finally, an interface syntax is assigned to establish the mathematical symbol operator in `Fortran`. It is important to highlight that the subroutine inputs and outputs must be consistent with the mathematical definition in terms of variable type, dimension, and size.

Once the mathematical operators that describe these arithmetics are implemented, we can easily obtain the hyper-dual form of a given function. Then, the gradient and the Hessian matrix are automatically obtained by extracting particular hyper-dual terms, as described earlier. Thus, we present the `Fortran` module containing the arithmetics of tensor hyper-dual numbers in Appendix A.2.1. We also show the code referring to the example presented in Section 2.2.3.

2.3 DIAGONAL HYPER-DUAL FORMULATION

In addition to the tensor hyper-dual formulation described in Section 2.2, we present in this section a modified version that is focused on the calculation of the diagonal terms of the second-order derivatives. By intentionally neglecting the off-diagonal terms, we reduce the number of internal operations of the arithmetic operators; moreover, we simplify the storage of the second-order derivatives.

We define the diagonal hyper-dual variables⁵ as

Scalar diagonal hyper-dual

$$\tilde{w} = w_0 + \mathbf{w}_1\epsilon_1 + \mathbf{w}_2\epsilon_2 + \mathbf{w}_3\epsilon_1\epsilon_2. \quad (67)$$

Vector diagonal hyper-dual

$$\tilde{\mathbf{q}} = \mathbf{q}_0 + \mathbf{q}_1\epsilon_1 + \mathbf{q}_2\epsilon_2 + \mathbf{q}_3\epsilon_1\epsilon_2. \quad (68)$$

Tensor diagonal hyper-dual

$$\tilde{\mathbf{K}} = \mathbf{K}_0 + \mathbf{K}_1\epsilon_1 + \mathbf{K}_2\epsilon_2 + \mathbf{K}_3\epsilon_1\epsilon_2. \quad (69)$$

⁵ We present the diagonal hyper-dual variables using sans serif font and tilde symbol.

For each hyper-dual variable presented in Eqs. 67, 68, and 69, we summarize a description of the terms in Tab. 3. In this particular case, we introduce another index related to the design variables, as given by $b = 1 \dots N$.

Table 3 – Derivative information obtained from the diagonal hyper-dual form.

Eq.	Diagonal hyper-dual	Real part \Re	Non-real parts		
			\Im_{ϵ_1}	\Im_{ϵ_2}	$\Im_{\epsilon_1 \epsilon_2}$
67	Scalar \tilde{w}	$w_0 \in \mathbb{R}$ w_0	$\mathbf{w}_1 \in \mathbb{R}^N$ $[w_1]_k = \frac{\partial w}{\partial s_k}$	$\mathbf{w}_2 \in \mathbb{R}^N$ $[w_2]_l = \frac{\partial w}{\partial s_l}$	$\mathbf{w}_3 \in \mathbb{R}^N$ $[w_3]_b = \frac{\partial^2 w}{\partial s_b^2}$
68	Vector $\tilde{\mathbf{q}}$	$\mathbf{q}_0 \in \mathbb{R}^n$ $[q_0]_i$	$\mathbf{q}_1 \in \mathbb{R}^n \times \mathbb{R}^N$ $[q_1]_{ik} = \frac{\partial q_i}{\partial s_k}$	$\mathbf{q}_2 \in \mathbb{R}^n \times \mathbb{R}^N$ $[q_2]_{il} = \frac{\partial q_i}{\partial s_l}$	$\mathbf{q}_3 \in \mathbb{R}^n \times \mathbb{R}^N$ $[q_3]_{ib} = \frac{\partial^2 q_i}{\partial s_b^2}$
69	Tensor $\tilde{\mathbf{K}}$	$\mathbf{K}_0 \in \mathbb{R}^{n \times n}$ $[K_0]_{ij}$	$\mathbf{K}_1 \in \mathbb{R}^{n \times n} \times \mathbb{R}^N$ $[K_1]_{ijk} = \frac{\partial K_{ij}}{\partial s_k}$	$\mathbf{K}_2 \in \mathbb{R}^{n \times n} \times \mathbb{R}^N$ $[K_2]_{ijl} = \frac{\partial K_{ij}}{\partial s_l}$	$\mathbf{K}_3 \in \mathbb{R}^{n \times n} \times \mathbb{R}^N$ $[K_3]_{ijb} = \frac{\partial^2 K_{ij}}{\partial s_b^2}$

Source – Own author.

To illustrate the extraction of the hyper-dual part containing the diagonal terms of the second-order derivatives, we show the following expressions:

$$\Im_{\epsilon_1 \epsilon_2} [\tilde{w}] = \mathbf{w}_3, \quad \text{where} \quad [w_3]_b = \frac{\partial^2 w}{\partial s_b^2}; \quad (70)$$

$$\Im_{\epsilon_1 \epsilon_2} [\tilde{\mathbf{q}}] = \mathbf{q}_3, \quad \text{where} \quad [q_3]_{ib} = \frac{\partial^2 q_i}{\partial s_b^2}; \quad \text{and} \quad (71)$$

$$\Im_{\epsilon_1 \epsilon_2} [\tilde{\mathbf{K}}] = \mathbf{K}_3, \quad \text{where} \quad [K_3]_{ijb} = \frac{\partial^2 K_{ij}}{\partial s_b^2}. \quad (72)$$

When compared to the original formulation, in this diagonal particularization we lower the tensor order concerning the axis that contains the second-order derivative information. For instance, the second-order derivatives of K_{ij} are represented by $\frac{\partial^2 K_{ij}}{\partial s_b^2}$, instead of $\frac{\partial^2 K_{ij}}{\partial s_l \partial s_k}$. This derivative information is now organized as a third-order tensor, where each term is represented by $[K_3]_{ijb}$. Thus, for each function type, all non-real terms are represented by the same tensor order, which is an important achievement to reduce the computational effort.

Besides, we highlight that the diagonal formulation preserves the hyper-dual terms ϵ_1 and ϵ_2 , which still represent the first-order derivatives of the function. Interestingly, to convert a function into the diagonal hyper-dual form, we follow the same general principles in terms of conversion steps and arithmetics; only a few modifications are required.

2.3.1 Adaptation of the arithmetics properties

In this section, we present a modified version of the tensor hyper-dual arithmetics. We introduce the operator $\delta_{k l b}$, presented in Section A.2.2, into the original hyper-dual rules

presented in Section 2.2.1. Since the modifications are confined only in the non-real term $\epsilon_1\epsilon_2$, we can state that this new formulation keeps the essence of tensor hyper-dual numbers.

To illustrate this adaptation process, let us consider the expression that defines the product of two scalar hyper-dual variables⁶. In Tab. 4, we show the original and the diagonal version of this operator for a quick comparison. Each row of the table represents a particular hyper-dual term, including the real and the non-real parts. We highlight the modifications made in axis $\epsilon_1\epsilon_2$, which includes the operator δ_{klb} . As noted, when compared to the original expression, the diagonal form does not require many changes.

Table 4 – Product of two scalar hyper-dual variables: original vs. diagonal.

	Hyper-dual number: ab	Diagonal hyper-dual number: $\tilde{a}\tilde{b}$
\Re	a_0b_0	a_0b_0
\Im_{ϵ_1}	$a_0[b_1]_k + b_0[a_1]_k$	$a_0[b_1]_k + b_0[a_1]_k$
\Im_{ϵ_2}	$a_0[b_2]_l + b_0[a_2]_l$	$a_0[b_2]_l + b_0[a_2]_l$
$\Im_{\epsilon_1\epsilon_2}$	$a_0[b_3]_{kl} + [a_1]_k[b_2]_l$ $+ [b_1]_k[a_2]_l + b_0[a_3]_{kl}$	$a_0[b_3]_b + [a_1]_k[b_2]_l \delta_{klb}$ $+ [b_1]_k[a_2]_l \delta_{klb} + b_0[a_3]_b$

Source – Own author.

We generalize this procedure to obtain a modified form of tensor hyper-dual operators, which comprise the arithmetics of diagonal hyper-dual numbers. We present a collection of expressions regarding the arithmetics of diagonal hyper-dual numbers, as given by:

⁶ The scalar diagonal hyper-dual variables \tilde{a} and \tilde{b} are given by

$$\tilde{a} = a_0 + \mathbf{a}_1\epsilon_1 + \mathbf{a}_2\epsilon_2 + \mathbf{a}_3\epsilon_1\epsilon_2 \quad \text{and} \quad \tilde{b} = b_0 + \mathbf{b}_1\epsilon_1 + \mathbf{b}_2\epsilon_2 + \mathbf{b}_3\epsilon_1\epsilon_2, \quad (73)$$

where the terms of vectors \mathbf{a}_1 , \mathbf{a}_2 , \mathbf{a}_3 , \mathbf{b}_1 , \mathbf{b}_2 , and \mathbf{b}_3 are presented in index notation by $[a_1]_k$, $[a_2]_l$, $[a_3]_b$, $[b_1]_k$, $[b_2]_l$, and $[b_3]_b$, respectively.

	\mathfrak{R}	$\mathfrak{S}_{\epsilon_1}$	$\mathfrak{S}_{\epsilon_2}$	$\mathfrak{S}_{\epsilon_1\epsilon_2}$	
$\tilde{\mathbf{a}} + \tilde{\mathbf{b}}$	$a_0 + b_0$	$[a_1]_k + [b_1]_k$	$[a_2]_l + [b_2]_l$	$[a_3]_b + [b_3]_b$	(74)
$\tilde{\mathbf{a}}^n$	a_0^n	$na_0^{n-1} [a_1]_k$	$na_0^{n-1} [a_2]_l$	$na_0^{n-1} [a_3]_b$	
				$+n(n-1)a_0^{n-2} [a_1]_k [a_2]_l \delta_{klb}$	(75)
$\tilde{\mathbf{a}} \tilde{\mathbf{b}}$	$a_0 b_0$	$a_0 [b_1]_k$	$a_0 [b_2]_l$	$a_0 [b_3]_b + [a_1]_k [b_2]_l \delta_{klb}$	
		$+b_0 [a_1]_k$	$+b_0 [a_2]_l$	$+ [b_1]_k [a_2]_l \delta_{klb} + b_0 [a_3]_b$	(76)
$\tilde{\mathbf{a}} \tilde{\mathbf{q}}$	$a_0 [q_0]_i$	$a_0 [q_1]_{ik}$	$a_0 [q_2]_{il}$	$a_0 [q_3]_{ib} + [a_1]_k [q_2]_{il} \delta_{klb}$	
		$+ [q_0]_i [a_1]_k$	$+ [q_0]_i [a_2]_l$	$+ [q_1]_{ik} [a_2]_l \delta_{klb} + [q_0]_i [a_3]_b$	(77)
$\tilde{\mathbf{a}} \tilde{\mathbf{K}}$	$a_0 [K_0]_{ij}$	$a_0 [K_1]_{ijk}$	$a_0 [K_2]_{ijl}$	$a_0 [K_3]_{ijb} + [a_1]_k [K_2]_{ijl} \delta_{klb}$	
		$+ [K_0]_{ij} [a_1]_k$	$+ [K_0]_{ij} [a_2]_l$	$+ [K_1]_{ijk} [a_2]_l \delta_{klb} + [K_0]_{ij} [a_3]_b$	(78)
$\tilde{\mathbf{q}} \otimes \tilde{\mathbf{f}}$	$[q_0]_i [f_0]_j$	$[q_0]_i [f_1]_{jk}$	$[q_0]_i [f_2]_{jl}$	$[q_0]_i [f_3]_{jb} + [q_1]_{ik} [f_2]_{jl} \delta_{klb}$	
		$+ [f_0]_j [q_1]_{ik}$	$+ [f_0]_j [q_2]_{il}$	$+ [f_1]_{jk} [q_2]_{il} \delta_{klb} + [f_0]_j [q_3]_{ib}$	(79)

The arithmetics involving the diagonal variant are quite similar to the original expressions of tensor hyper-dual numbers. With small modifications, we could define a new set of arithmetic properties, providing a base material for computer implementation. For the sake of conciseness, we focused on the operations to write the diagonal hyper-dual form of the stiffness matrix of a truss element, which is presented in Section 2.3.2 as an example.

2.3.2 Example

In this section, we show how to apply the arithmetics of the diagonal hyper-dual formulation. We focus on the calculation of the diagonal terms of the second-order derivative information based on a diagonal hyper-dual representation of the stiffness matrix of a truss element, which is given by

$$\tilde{\mathbf{K}} = \tilde{\mathbf{e}} \tilde{\mathbf{a}} \tilde{\mathbf{l}} (\tilde{\mathbf{b}} \otimes \tilde{\mathbf{b}}). \quad (80)$$

We aim to calculate the derivatives with respect to a vector containing the following design variables: cross-sectional area A and elastic modulus E .

As the initial step, we define the design variables A and L as diagonal hyper-dual numbers with unit perturbation in axes ϵ_1 and ϵ_2 , as given by

$$\tilde{\mathbf{a}} = A + \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \epsilon_1 \epsilon_2 \quad \text{and} \quad (81)$$

$$\tilde{\mathbf{l}} = L + \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \epsilon_1 \epsilon_2. \quad (82)$$

It should be noted that the diagonal form of hyper-dual numbers assumes the same tensor order for all non-real terms.

To proceed with the conversion process, we apply the product operation (Eq. 76) to obtain

$$\tilde{\mathbf{a}}\tilde{\mathbf{l}} = AL + \begin{Bmatrix} L \\ A \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} L \\ A \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \epsilon_1\epsilon_2 \quad \text{and} \quad (83)$$

$$\tilde{\mathbf{e}}\tilde{\mathbf{a}}\tilde{\mathbf{l}} = EAL + \begin{Bmatrix} EL \\ EA \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} EL \\ EA \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \epsilon_1\epsilon_2. \quad (84)$$

Then, we obtain the strain-displacement matrix using the expressions indicated in Eqs. 75 and 77, as given by

$$\frac{1}{\tilde{\mathbf{l}}} = \frac{1}{L} + \begin{Bmatrix} 0 \\ -1/L^2 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 0 \\ -1/L^2 \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 0 \\ 2/L^3 \end{Bmatrix} \epsilon_1\epsilon_2 \quad \text{and} \quad (85)$$

$$\tilde{\mathbf{b}} = \begin{Bmatrix} -1/L \\ 1/L \end{Bmatrix} + \begin{bmatrix} 0 & 1/L^2 \\ 0 & -1/L^2 \end{bmatrix} \epsilon_1 + \begin{bmatrix} 0 & 1/L^2 \\ 0 & -1/L^2 \end{bmatrix} \epsilon_2 + \begin{bmatrix} 0 & -2/L^3 \\ 0 & 2/L^3 \end{bmatrix} \epsilon_1\epsilon_2. \quad (86)$$

Based on the definition of tensor product (Eq. 79), we obtain

$$\begin{aligned} \tilde{\mathbf{b}} \otimes \tilde{\mathbf{b}} = & \begin{bmatrix} 1/L^2 & -1/L^2 \\ -1/L^2 & 1/L^2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & -2/L^3 & 2/L^3 \\ 0 & 0 & 2/L^3 & -2/L^3 \end{bmatrix} \epsilon_1 + \begin{bmatrix} 0 & 0 & -2/L^3 & 2/L^3 \\ 0 & 0 & 2/L^3 & -2/L^3 \end{bmatrix} \epsilon_2 \\ & + \begin{bmatrix} 0 & 0 & 6/L^4 & -6/L^4 \\ 0 & 0 & -6/L^4 & 6/L^4 \end{bmatrix} \epsilon_1\epsilon_2. \end{aligned} \quad (87)$$

We complete the diagonal hyper-dual conversion of the stiffness matrix using the product operator indicated in Eq. 78, as given by

$$\begin{aligned} \tilde{\mathbf{K}} = & \begin{bmatrix} EA/L & -EA/L \\ -EA/L & EA/L \end{bmatrix} + \begin{bmatrix} E/L & -E/L & -EA/L^2 & EA/L^2 \\ -E/L & E/L & EA/L^2 & -EA/L^2 \end{bmatrix} \epsilon_1 \\ & + \begin{bmatrix} E/L & -E/L & -EA/L^2 & EA/L^2 \\ -E/L & E/L & EA/L^2 & -EA/L^2 \end{bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 & 2EA/L^3 & -2EA/L^3 \\ 0 & 0 & -2EA/L^3 & 2EA/L^3 \end{bmatrix} \epsilon_1\epsilon_2. \end{aligned} \quad (88)$$

Finally, we extract the derivatives from the hyper-dual variable using the operator \mathfrak{S} , as illustrated by

$$\mathfrak{S}_{\epsilon_1} [\tilde{\mathbf{K}}] = \frac{d\mathbf{K}}{ds} = \begin{bmatrix} \frac{\partial K_{11}}{\partial A} & \frac{\partial K_{12}}{\partial A} & \frac{\partial K_{11}}{\partial L} & \frac{\partial K_{12}}{\partial L} \\ \frac{\partial K_{21}}{\partial A} & \frac{\partial K_{22}}{\partial A} & \frac{\partial K_{21}}{\partial L} & \frac{\partial K_{22}}{\partial L} \end{bmatrix} = \begin{bmatrix} E/L & -E/L & -EA/L^2 & EA/L^2 \\ -E/L & E/L & EA/L^2 & -EA/L^2 \end{bmatrix} \quad \text{and} \quad (89)$$

$$\mathfrak{S}_{\epsilon_1\epsilon_2} [\tilde{\mathbf{K}}] = D \left[\frac{d^2\mathbf{K}}{ds^2} \right] = \begin{bmatrix} \frac{\partial^2 K_{11}}{\partial A^2} & \frac{\partial^2 K_{12}}{\partial A^2} & \frac{\partial^2 K_{11}}{\partial L^2} & \frac{\partial^2 K_{12}}{\partial L^2} \\ \frac{\partial^2 K_{21}}{\partial A^2} & \frac{\partial^2 K_{22}}{\partial A^2} & \frac{\partial^2 K_{21}}{\partial L^2} & \frac{\partial^2 K_{22}}{\partial L^2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 2EA/L^3 & -2EA/L^3 \\ 0 & 0 & -2EA/L^3 & 2EA/L^3 \end{bmatrix}. \quad (90)$$

As shown in this example, the diagonal hyper-dual formulation provides the exact calculation of the first-order information and the diagonal terms of the second-order derivative; i.e., this derivative result is consistent with the analytical predictions.

In addition, we show the computer implementation and the corresponding code concerning this example in Section A.2.3. Since we use the operator overloading technique (CHIVERS; SLEIGHTHOLME, 2011; NEUENHOFEN, 2018), this program represents a black-box tool that provides the diagonal terms of the second-order derivatives.

As a result, by using this hyper-dual variant, we can significantly mitigate the computational effort in terms of storage and operations, while keeping accurate derivative calculations. This particularization is especially convenient in circumstances in which the full Hessian is not required.

3 HYPER-DUAL HYPERELASTICITY

As each hyperelastic material model is defined by a different potential, the corresponding first and second-order derivatives have to be obtained individually for each law, a procedure that might demand cumbersome calculations if performed analytically. A possibility to alleviate this burden is to employ numerical methods, as suggested by Yuxiang Wang and Gerling (2016). This process is critical if we consider that the quadratic convergence of the iterative finite element solution only occurs if the derivatives are accurately taken.

In this context, the traditional finite differences are typically inefficient and inaccurate but their complex step variant has been shown to provide accurate results for first-order derivatives (KIRAN; KHANDELWAL, 2014). In order to obtain the second-order information, Jeffrey Alan Fike (2013) studied the hyper-dual numbers and observed negligible errors, as this method is neither affected by subtraction nor truncation errors. More recently, Tanaka et al. (2015), Kiran and Khandelwal (2015) and Fujikawa et al. (2020) reported the successful use of hyper-dual numbers to calculate stress and tangent modulus from a strain energy potential in the context of an anisotropic hyperelastic material in a 3D continuum setting; Tanaka et al. (2016) also evaluated the application of hyper-dual numbers for the modeling of dissipative behavior of materials. Fohrmeister et al. (2018) studied thermomechanically coupled gradient-enhanced elastoplasticity using hyper-dual numbers.

As a contribution to this subject, we evaluated the hyper-dual numbers in the well-known Ogden hyperelastic law. This constitutive model allows straightforward derivative calculations, which are useful to establish an analytical reference; moreover, we can compare and validate our numerical results with commercial finite element codes, as this material model is frequently included in their libraries. In addition, as a potential reference for further studies, we also included the expressions regarding stress and tangent modulus considering different energetically conjugate pairs.

In addition to calculation accuracy and generality, one aspect that deserves our attention in terms of the engineering applications of the hyper-dual numbers is the amount of operations to properly convert a given expression. In this regard, the hyper-dual arithmetics are between 4 and 14 times as expensive as the real function evaluation (FIKE, Jeffrey Alan, 2013). However, when compared to the analytical expressions or other numerical methods, how does the hyper-dual scheme perform in terms of processing time?

This chapter comprises a preliminary study on hyperelastic material models, wherein the derivatives are obtained using the hyper-dual scheme. In Section 3.1, we review Ogden's model assuming an incompressible hypothesis; additionally, we summarize the analytical expressions for stresses and tangent moduli, considering different energetically conjugate pairs. We discuss the hyper-dual conversion of the potential function in Section 3.2. The analytical validation of the hyper-dual scheme in this constitutive model context is presented in Section 3.2.1. The aforementioned sections comprise the theoretical framework for this study. In Section 3.3,

we describe the structural problem and the numerical models for this study concerning the processing time.

3.1 OGDEN'S HYPERELASTIC MODEL

Constitutive equations establish the mechanical behavior of materials, which is commonly exemplified by the Hooke's law for linear elastic materials. Hyperelastic models are especially suited to represent nonlinear elastic materials subjected to large strains. Assuming the material incompressibility hypothesis with the principal stretches $\lambda_2 = \lambda_3 = \lambda_1^{-\frac{1}{2}}$, Ogden's hyperelasticity model for truss elements is given by

$$W(\lambda_1) = \sum_{j=1}^M \frac{\gamma_j}{\beta_j} \left(\lambda_1^{\beta_j} + 2\lambda_1^{\frac{-\beta_j}{2}} - 3 \right), \quad (91)$$

wherein the material coefficients γ_j and β_j are frequently obtained using parameter identification and curve-fitting procedures (STUMPF; MARCZAK, Rogério José, 2016). In addition, j and M are the parameter index and the upper bound of the summation, respectively.

As noted in Eq. 91, the strain energy is represented by a scalar function of a scalar argument, which simplifies further derivative calculations to obtain stresses and tangent moduli. Considering the rotated engineering definition as a reference, these variables are defined by

$$\sigma^{RE} = \frac{dW}{d\lambda_1} \quad \text{and} \quad E_T^{RE} = \frac{d^2W}{d\lambda_1^2}. \quad (92)$$

Moreover, Tab. 5 summarizes the mathematical expressions to obtain the measures of strain, stress and tangent modulus considering different energetically conjugate pairs¹. It should be noted that it is possible to write them as a function of the rotated engineering terms, which are described by the analytical expressions

$$\begin{aligned} \sigma^{RE} &= \sum_{j=1}^M \gamma_j \left(\lambda_1^{\beta_j-1} - \lambda_1^{\frac{-\beta_j}{2}-1} \right) \quad \text{and} \\ E_T^{RE} &= \sum_{j=1}^M \gamma_j \left[(\beta_j - 1) \lambda_1^{\beta_j-2} + \left(\frac{\beta_j}{2} + 1 \right) \lambda_1^{\frac{-\beta_j}{2}-2} \right]. \end{aligned} \quad (93)$$

¹ The superscripts related to the measures of strain, stress and tangent modulus are described as Rotated Engineering (^{RE}), Logarithmic (^L), Green-Lagrange (^{GL}), Cauchy (^C), and Second Piola-Kirchhoff (^{2PK}).

Table 5 – Measures of strain, stress and tangent modulus.

Strain	Stress	Tangent modulus
$\varepsilon^{RE} = \lambda_1 - 1$	$\sigma^{RE} = \frac{dW}{d\lambda_1}$	$E_T^{RE} = \frac{d\sigma^{RE}}{d\lambda_1}$
$\varepsilon^L = \ln(\lambda_1)$	$\sigma^C = \lambda_1 \sigma^{RE}$	$E_T^C = \lambda_1^2 E_T^{RE} + \lambda_1 \sigma^{RE}$
$\varepsilon^{GL} = \frac{1}{2}(\lambda_1^2 - 1)$	$\sigma^{2PK} = \lambda_1^{-1} \sigma^{RE}$	$E_T^{2PK} = \lambda_1^{-2} E_T^{RE} - \lambda_1^{-3} \sigma^{RE}$

Source – Own author.

3.2 HYPER-DUAL REPRESENTATION

From Eqs. 92 and 7, we can see that the hyper-dual representation of the potential function includes relevant information related to the mechanical behavior, as it simultaneously outputs the respective rotated engineering stress and the tangent modulus. The hyper-dual form of the strain energy function is mathematically described by

$$W(\lambda_1 + \varepsilon_1 + \varepsilon_2) = W(\lambda_1) + \sigma^{RE}(\lambda_1)\varepsilon_1 + \sigma^{RE}(\lambda_1)\varepsilon_2 + E_T^{RE}(\lambda_1)\varepsilon_1\varepsilon_2. \quad (94)$$

Thus, as shown in Eq. 9, we can extract the derivative information in a straightforward manner, i.e.

$$\begin{aligned} \sigma^{RE}(\lambda_1) &= \mathfrak{S}_{\varepsilon_1}[W(\lambda_1 + \varepsilon_1 + \varepsilon_2)] = \mathfrak{S}_{\varepsilon_2}[W(\lambda_1 + \varepsilon_1 + \varepsilon_2)] \quad \text{and} \\ E_T^{RE}(\lambda_1) &= \mathfrak{S}_{\varepsilon_1\varepsilon_2}[W(\lambda_1 + \varepsilon_1 + \varepsilon_2)]. \end{aligned} \quad (95)$$

Additionally, once σ^{RE} and E_T^{RE} are known, we can easily obtain the stresses and tangent moduli for the other conjugate pairs, as previously shown in Tab. 5.

It is important to highlight that handling hyper-dual numbers requires mathematical operations with properly defined arithmetic properties (FIKE, Jeffrey Alan, 2013). In order to illustrate these operations, we present a detailed example in Section A.1.1. Moreover, we coded such properties in Fortran language program using the operator overloading scheme, which provides interesting generality aspects. Huck et al. (2016) suggested this approach to easily add new features, such as algorithmic differentiation, to an existing software. In Section 3.2.2, we illustrate the computer implementation of a material subroutine using the hyper-dual format.

3.2.1 Analytical validation

To validate this numerical procedure, we compared analytical and hyper-dual evaluations of the stresses and the tangent moduli of Ogden's model. We present these topics in Sections 3.1 and 3.2, respectively. For this study, we considered the coefficients indicated in Tab. 6, which describe a compliant incompressible material.

According to Ogden et al. (2004), it is possible to estimate the initial shear modulus

Table 6 – Material parameters.

j	γ_j	β_j
1	0.77817	2.7971
2	-0.011229	-2.7188
3	1.269×10^{-7}	10.505
4	16.169	0.33382

Source – Ogden et al. (2004).

G_0 and the initial Young modulus E_0 using the expressions

$$G_0 = \left(\frac{1}{2}\right) \sum_{j=1}^M \gamma_j \beta_j \quad \text{and} \quad E_0 = 2G_0(1 + \nu_0) = 3G_0. \quad (96)$$

Based on this information, we can check the results regarding the tangent modulus in the undeformed situation, i.e. when $\lambda_1 = 1$. It should be noted that the expressions indicated in Eq. 96 are only valid considering the material incompressibility hypothesis. Assuming small deformations, it is well-established that the initial Poisson ratio ν_0 defines the material compressibility.

Hence, by applying the material parameters shown in Tab. 6 into Eq. 93, the material behavior is graphically presented in terms of stress and tangent modulus in Figs. 1 and 2, respectively. The comparison between the analytical and the numerical results showed a good agreement for all outputs; the information regarding the error² is also presented in each figure's caption. Since the output variables are stored with a limited number of digits, the observed errors can be related to floating-point arithmetics³. As for the energetically conjugate pairs, the curves show similar values only when the stretch values approach unity, as expected; we highlighted this detail in Figs. 1 and 2. Additionally, we confirmed the initial stiffness predictions indicated by Eq. 96.

3.2.2 Material subroutine

In this section, we illustrate the Fortran code concerning the material subroutine. Assuming the use of the operator overloading technique for the hyper-dual implementation, we can virtually preserve the code syntax. In order to illustrate this statement, we show the Fortran codes side-by-side in Fig. 3 for a quick comparison.

As noted, the ordinary procedure requires the material information regarding stress and tangent modulus, along with the parameters that describe a particular material. In the

² The maximum error is defined as

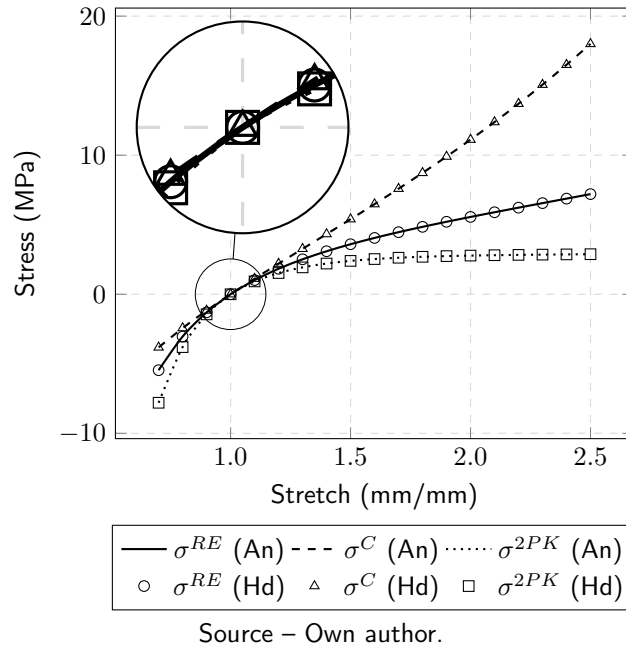
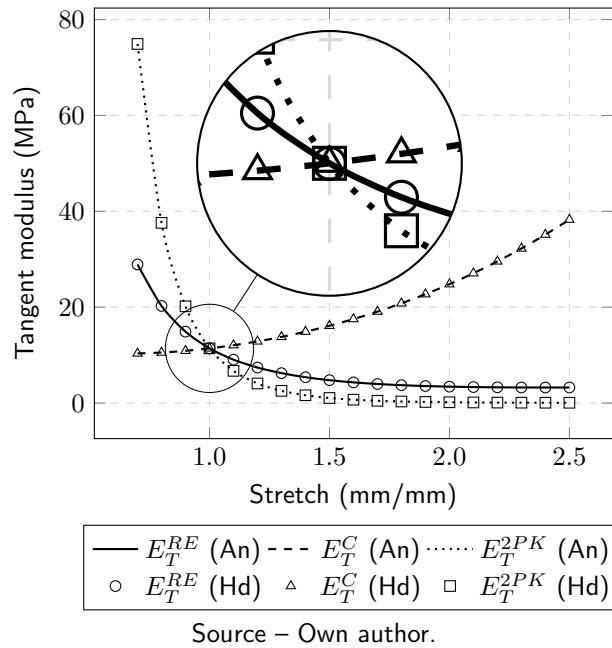
$$\text{Error}_{\max} = |\text{An} - \text{Hd}|, \quad (97)$$

where An and Hd represent the analytical and hyper-dual outputs, respectively.

³ Hanselman and Littlefield (2011) exemplify this type of error in Matlab with the expression

$$1 - 0.2 - 0.2 - 0.2 - 0.2 - 0.2,$$

as it results in a residuum of 10^{-17} , instead of exactly null.

Figure 1 – Stress vs. stretch for each conjugate pair. $\text{Error}_{\max} = 10^{-15}$.Figure 2 – Tangent modulus vs. stretch for each conjugate pair. $\text{Error}_{\max} = 10^{-13}$.

hyper-dual subroutine, only the strain energy function must be introduced, wherein we can observe the same code syntax. It should be noted that this scheme requires the application of a perturbation value on the stretch variable, as the derivatives are calculated with respect to such a variable.

Figure 3 – Fortran code describing Ogden's material model.

Material subroutine with the analytical expressions.

```

1  subroutine ogden_an(stretch , sigma , E)
2
3  real(8) , intent(in) :: stretch
4  real(8) , intent(out) :: sigma , E
5  real(8) , allocatable , dimension(:) ::
   ↪ gamma , beta
6  integer :: j , M
7
8
9
10
11  ! material parameters
12  M = 4; allocate(gamma(M) , beta(M))
13  gamma(1)=0.77817;    beta(1)=2.7971
14  gamma(2)=-0.011229; beta(2)=-2.7188
15  gamma(3)=1.269e-7;  beta(3)=10.505
16  gamma(4)=16.169;    beta(4)=0.33382
17  ! material model
18  sigma = 0; E = 0
19
20  do j = 1,M
21  sigma = sigma + gamma(j) * ( stretch
   ↪ *(beta(j)-1) - stretch**(-
   ↪ beta(j)/2 -1) )
22  E = E + gamma(j) * ( (beta(j)-1)*
   ↪ stretch**(beta(j)-2) + (beta(j)
   ↪ )/2 + 1)*stretch**(-beta(j)/2
   ↪ -2) )
23  enddo
24
25  end subroutine ogden_an

```

Material subroutine with the hyper-dual expression.

```

1  subroutine ogden_hd(stretch , sigma , E)
2  use hyperdual_module
3  real(8) , intent(in) :: stretch
4  real(8) , intent(out) :: sigma , E
5  real(8) , allocatable , dimension(:) ::
   ↪ gamma , beta
6  integer :: j , M
7  type(hyperdual_scalar) :: stretch_hd
8  type(hyperdual_scalar) ::
   ↪ ogden_energy_hd
9  real(8) , dimension(1) :: grad
10 real(8) , dimension(1,1) :: hess
11 ! material parameters
12 M = 4; allocate(gamma(M) , beta(M))
13 gamma(1)=0.77817;    beta(1)=2.7971
14 gamma(2)=-0.011229; beta(2)=-2.7188
15 gamma(3)=1.269e-7;  beta(3)=10.505
16 gamma(4)=16.169;    beta(4)=0.33382
17 ! material model
18 grad = 0; hess = 0; ogden_energy_hd =
   ↪ hyperdual_scalar(0 , grad , grad ,
   ↪ hess)
19 grad = 1; stretch_hd =
   ↪ hyperdual_scalar(stretch , grad ,
   ↪ grad , hess)
20 do j = 1,M
21 ogden_energy_hd = ogden_energy_hd + (
   ↪ ( stretch_hd ** beta(j) ) +
   ↪ ( stretch_hd ** (-beta(j)/2 )
   ↪ * 2d0 - 3d0 ) * (gamma(j)/beta
   ↪ (j) ) )
22 enddo
23 sigma = ogden_energy_hd%a_1(1)
24 E = ogden_energy_hd%a_3(1,1)
25 end subroutine ogden_hd

```

Source – Own author.

3.3 A STUDY ON THE PROCESSING TIME

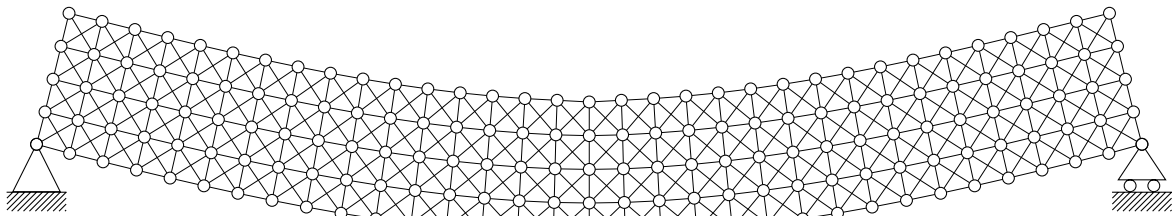
In this section, we present a comparative study regarding the effects of the derivative method on the processing time of the structural analysis. We evaluated the hyper-dual and the central finite differences as numerical derivative methods in a constitutive model context. As a reference, we also considered the use of analytical expressions for the stress and the tangent modulus calculation.

We introduced the hyper-dual approach in the continuum-like nonlinear truss formulation presented by Muñoz-Rojas (n.d.); a concise review regarding this topic is presented in Appendix B. We implemented the contents of this section in Fortran language program considering an adaptation of the skyline matrix storage proposed by Dhatt and Touzot (1984) and the symmetric bandsolver indicated by Lo (1992). It is important to highlight that we validated this computer implementation considering the material model calculations with the hyper-dual scheme with the commercial software Abaqus.

In this numerical study, we proposed an evaluation of a beam deflection model, whose initial length and height are 800 mm and 100 mm, respectively. Our model assumes that this structure is made of several hyperelastic truss elements with the material parameters indicated in Tab. 6. In terms of load and boundary conditions, the cases are categorized as center-loaded simply-supported beams.

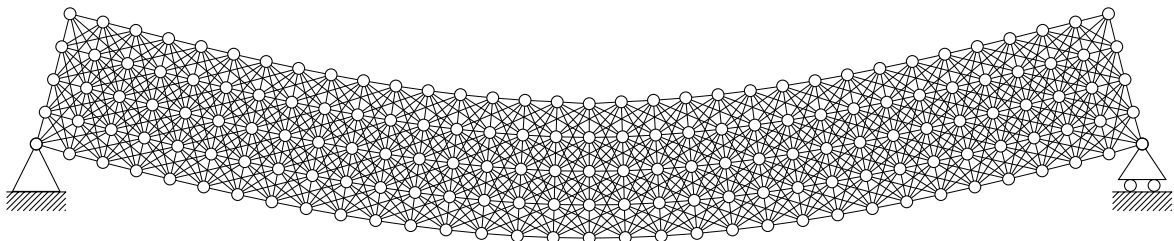
For this study, we controlled the model size using a specific mesh generator (ANDRADE et al., 2004). A particular configuration setting denoted by neighborhood level defines the creation of an element based on the proximity of two nodes. Using the values 1, 2, and 3 for this parameter, Figs. 4, 5 and 6 illustrate the deformed shape of the Models A, B, and C, respectively. As noted, the described models show an increasing number of elements, while the number of nodes is fixed.

Figure 4 – Deformed shape of Model A (Neighborhood level: 1).



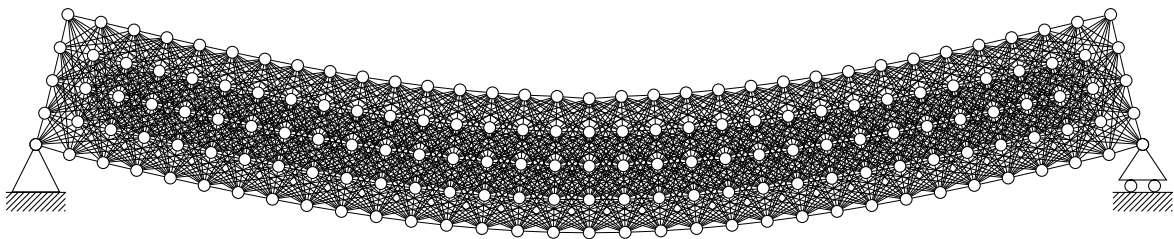
Source – Own author.

Figure 5 – Deformed shape of Model B (Neighborhood level: 2).



Source – Own author.

Figure 6 – Deformed shape of Model C (Neighborhood level: 3).



Source – Own author.

This mesh strategy involving one-dimensional elements was an attempt to detach the influence of the model size in terms of degrees of freedom and elements on the processing

time. Thus, it could provide a distinctive discussion regarding the computational effort of the hyper-dual procedure in a structural analysis context. We built the numerical models indicated in Tab. 7, which includes the information concerning model size, geometry, and loading. Aiming for a similar result regarding the deformed shape and stress levels among the models, we adjusted the latter simulation input parameters for each model. Consequently, as this condition led to an equivalent convergence difficulty, all models required the same number of Newton-Raphson iterations, considering the analytical baseline for the constitutive model.

In terms of hardware specifications, we obtained the finite element solution using a desktop computer with Intel® Core™ i7-3770K processor and Gfortran compiler 7.4.0 running on Ubuntu 18.04. We collected the processing time of each structural analysis using the intrinsic CPU_TIME subroutine. For each case presented in Tab. 7, the structural simulation was repeated 5 times.

Table 7 – Description of the numerical models in terms of model size and input parameters (DOF – degrees of freedom, nel – number of elements, A_0 – area, and F_y – vertical load).

Model	Group	Neighborhood	DOF	nel	A_0 (mm ²)	F_y (N)
A	1	1	495	548	4	-2
B	1	2	495	988	2	-2
C	1	3	495	1660	2	-4
D	2	1	2145	2634	4	-4
E	2	2	2145	5046	2	-5
F	2	3	2145	9434	2	-10
G	3	1	3783	4716	2	-2
H	3	2	3783	9108	2	-5
I	3	3	3783	17252	2	-12
J	4	1	10143	12980	4	-6
K	4	2	10143	25420	2	-8
L	4	3	10143	49228	2	-20
M	5	1	25443	33056	2	-5
N	5	2	25443	65248	1	-6
O	5	3	25443	127912	1	-11

Source – Own author.

In this section, we present the results concerning the use of hyper-dual numbers in a material subroutine. Section 3.3.1 comprises a preliminary evaluation to discuss the effects of the perturbation values on the convergence behavior. In Section 3.3.2, we present the results regarding the effects of the model size on the processing time considering the use of the hyper-dual procedure in the material model.

3.3.1 Effects of the perturbation value

As we aim to discuss a numerical alternative for derivative calculations in constitutive models, a comparison with ordinary methods becomes relevant. We initially evaluated the

effects of the perturbation value on the convergence behavior considering the finite difference approximation. In this regard, we evaluated the first group of models indicated in Tab. 7, which considers a growing number of elements and a fixed quantity of degrees of freedom (Figs. 4, 5 and 6).

We assumed the perturbation values h for the numerical derivative methods, as summarized in Tab. 8. As a result, the simulation convergence for each method is described by the number of iterations k . Due to the accurate derivative calculation, we found that the hyper-dual scheme (Hd) followed the analytical (An) convergence behavior for all models. Although we had assumed unit perturbation, the accuracy of this numerical method is not affected by such parameter (FIKE, Jeffrey Alan, 2013).

Table 8 – Relationship among derivative methods, perturbation and number of iterations.

Derivative method		Model A		Model B		Model C	
		h	k	h	k	h	k
An	Analytic	–	7	–	7	–	7
Hd	Hyper-dual	1	7	1	7	1	7
Fd1	Finite differences	0.02	7	0.02	7	0.02	7
Fd2	Finite differences	0.084	14	0.058	19	0.065	26

Source – Own author.

Initially, considering the central finite differences, we carefully had chosen perturbation values that could preserve the original rate of convergence, as indicated by the derivative method (Fd1) in Tab. 8. As an illustration of the typical perturbation dependency of this method, we included another set of results indicated by (Fd2). In this case, the number of iterations increased significantly, as a consequence of the inaccurate derivative information. Moreover, it is important to highlight that the structural analysis diverged for a multitude of perturbation values.

The finite difference results (Fd1) and (Fd2) exemplified the effect of the perturbation value on the analysis behavior. Therefore, recalling that an ideal perturbation value can not be assumed a priori, we demonstrated that this numerical method shows limited predictability in terms of analysis convergence and processing time. In this context, the hyper-dual scheme provided a stable analysis, with identical convergence behavior when compared to the analytical derivative method. Therefore, we could observe the negligible impact of the hyper-dual error shown in Figs. 1 and 2 on the structural analysis.

3.3.2 Hyper-dual performance in constitutive models

In this section, we extend the discussion regarding the use of hyper-dual numbers and the processing time by considering all the numerical models presented in Tab. 7. It is important to highlight that all simulations required the same number of iterations to converge.

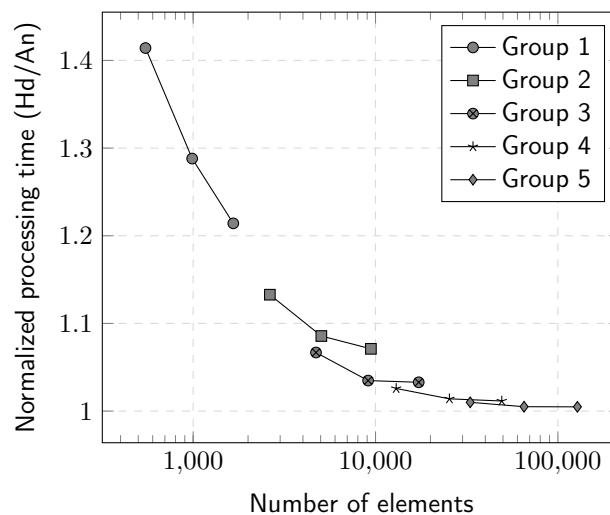
We evaluate the overall performance of the finite element analysis in Section 3.3.2.1. The computational time related to the intermediate procedures is presented in Section 3.3.2.2.

3.3.2.1 Overall effort

Initially, we present the results regarding the overall processing time of the analysis. For each model, based on the collected computing times, we considered the average values for further data treatment. Thus, we calculated the normalized computational effort as the ratio between the processing times using the hyper-dual scheme and the analytical expressions.

Fig. 7 shows the relationship between the normalized processing time and the model size in terms of the number of elements. We grouped the models with the same number of degrees of freedom according to Tab. 7 and noticed a decreasing pattern in the results. As the number of elements increases, the normalized processing time tends to unity, i.e. the hyper-dual and the analytical performances gradually become similar. The data represented by Group 5 achieved stabilization in terms of the normalized computing time and model size. Thus, this particular set of results suggests the model size from which the hyper-dual procedure becomes highly comparable with the analytical one.

Figure 7 – Model size and overall processing time, models organized in groups with the same number of degrees of freedom.



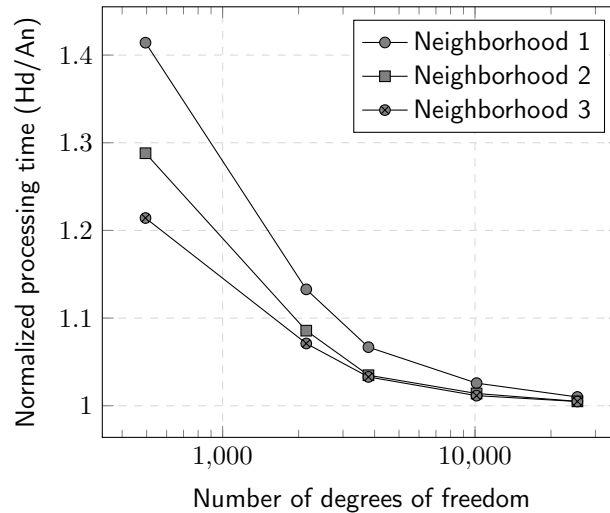
Source – Own author.

Thus, although our study focused on truss structures, we confirmed the statements of Tanaka et al. (2015) regarding the overall computational costs of the hyper-dual numbers in hyperelastic models; these authors had indicated the asymptotic curves in a study using solid elements. Additionally, the obtained results are corroborated by the conclusions of Fohrmeister et al. (2018) concerning the scalability of a hyper-dual implementation.

Alternatively, as shown in Fig. 8, these same results concerning the normalized processing times can be reorganized according to the neighborhood level and the number of degrees of

freedom. We noted similar values for the models with neighborhood levels 2 and 3, which showed a faster convergence to the unity value when compared to the results concerning the level 1 data.

Figure 8 – Model size and overall processing time, models organized in groups with the same neighborhood level.



Source – Own author.

3.3.2.2 Intermediate effort

In addition to the total processing time evaluation, which comprises the entire code, we also evaluated the average processing times related to the following operations: the solution of the linear equation system and the global tangent stiffness matrix calculation, which examines the element loop. As these operations occur multiple times in an iterative process, the relative processing time considered the ratio between the summation of the respective values and the total processing time.

This supplementary information is important since it clearly distinguishes the contribution of each stage in terms of the overall computational effort. Particularly, as the stiffness matrix procedure involves element calculations, it includes the material model evaluation using the hyper-dual scheme. Considering the use of the analytical expressions in the constitutive model, Fig. 9 illustrates the average values for the relative computational effort concerning the finite element analysis; similarly, Fig. 10 shows the results considering the hyper-dual approach. It should be noted that we organized the numerical models according to their size, as described in Tab. 7.

We can observe a descending pattern related to the relative global stiffness matrix calculation. In general, as expected for larger models, we could assert how the solution of the equation system dominates the computational effort along the numerical models, which occurs

Figure 9 – Relative processing time using the analytical expressions.

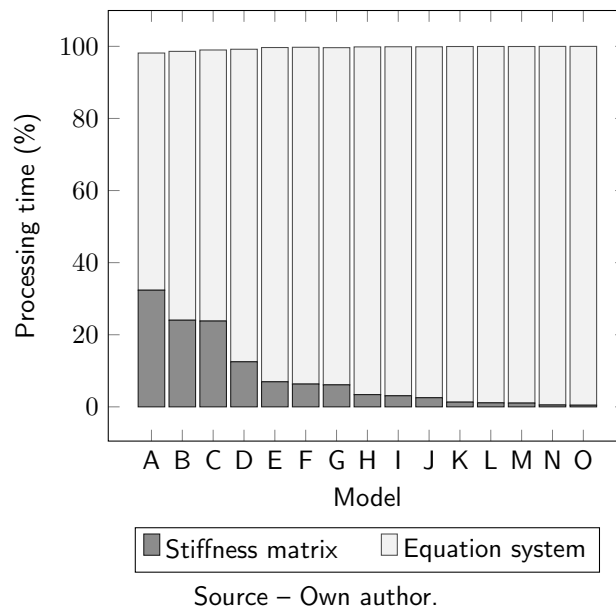
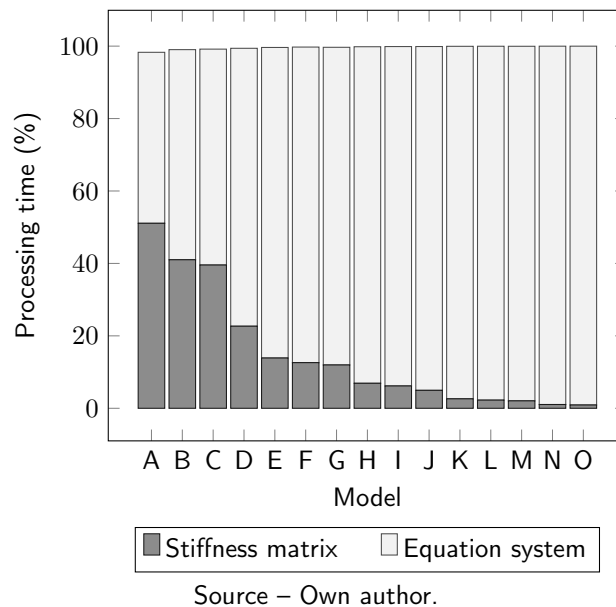


Figure 10 – Relative processing time using the hyper-dual procedure.



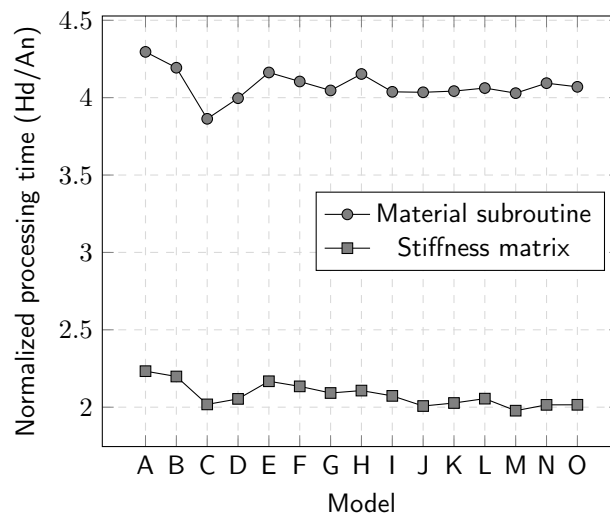
for both methods⁴. As for the hyper-dual calculation, through this comparative analysis, we noted a larger contribution related to the stiffness matrix, especially in the relatively smaller models. For instance, Model A shows the values 32% and 51% for the analytical and the hyper-dual scheme, respectively. These findings explain the results previously indicated in Fig. 7; as the computing time is prominently dominated by the solution of the equation system, the choice for a derivative method becomes less relevant in the larger models.

⁴ For some models indicated in Figs. 9 and 10, the summation of the relative times did not achieve 100% because some intermediate calculations, such as input data reading and convergence criteria evaluation, had not been particularized in this study.

We can also evaluate the relationship between the neighborhood level and effort to solve the equation system in Figs. 9 and 10. For this purpose, it is noteworthy that each group consisting of three models has the same number of degrees of freedom⁵. Although a fixed number of degrees of freedom would initially suggest a constant computational effort to solve the equation system, we found that the required processing time showed a considerable increase in terms of the relevance of the equation solution. This behavior is due to the bandwidth modifications within each group of models. Therefore, a larger matrix bandwidth imposed a significant impact on the computational effort, being more evident than the element loop for the stiffness matrix assembly.

In addition, we presented the normalized processing times related to the material subroutine and global tangent stiffness matrix in Fig. 11. For each operation, we observed an approximately constant tendency along the models. These data show how the hyper-dual costs are attenuated when more calculations are considered in the code. The material subroutine with the hyper-dual procedure was around 4 times slower than the analytical scheme; as for the global tangent stiffness calculation, the ratio decreased to 2, approximately.

Figure 11 – Normalized computing time of intermediate operations.



Source – Own author.

⁵ Based on the number of degrees of freedom (Tab. 7), we can define the groups ABC, DEF, GHI, JKL and MNO.

4 HYPER-DUAL SENSITIVITY ANALYSIS

In this chapter, we present the use of hyper-dual numbers as an automatic differentiation tool in design sensitivity analysis. Initially, in Section 4.1, we review general expressions concerning second-order sensitivity analysis (HAUG et al., 1986; TORTORELLI; MICHALERIS, 1994; CHOI; KIM, 2005a, 2005b; KLEIBER, 1997; HAFTKA; GÜRDAL, 1991). In this context, in Section 4.2, we describe a method to evaluate the element-level derivatives using the hyper-dual numbers scheme. We also present a strategy to obtain the diagonal terms of the Hessian matrix using the diagonal hyper-dual variant; this particularization involving linear structures is shown in Section 4.3.

4.1 THEORETICAL BACKGROUND

Structural optimization methods typically aim to extremize an objective function ψ subject to design constraints; an iterative process is conducted based on the manipulation of the design variables s_k . In this context, many algorithms rely on design sensitivity analysis, which aims to establish the effects of design parameters on the objective function. Design sensitivity is mathematically represented by a derivative, which can be written in index notation form¹ as

$$\frac{d\psi}{ds_k} = \frac{\partial\psi}{\partial s_k} + \frac{\partial\psi}{\partial u_j} \frac{du_j}{ds_k}, \quad (99)$$

where u_j represents the nodal displacements of a structure. The indexes are represented by $k = 1 \dots N$ and $j = 1 \dots n$, where N is the number of design variables and n refers to the number of degrees of freedom. Similarly, the second-order derivative is given by

$$\frac{d^2\psi}{ds_l ds_k} = \frac{\partial^2\psi}{\partial s_l \partial s_k} + \frac{\partial^2\psi}{\partial u_j \partial s_k} \frac{du_j}{ds_l} + \frac{\partial^2\psi}{\partial s_l \partial u_j} \frac{du_j}{ds_k} + \frac{\partial^2\psi}{\partial u_m \partial u_j} \frac{du_m}{ds_l} \frac{du_j}{ds_k} + \frac{\partial\psi}{\partial u_j} \frac{d^2u_j}{ds_l ds_k}. \quad (100)$$

The indexes of the design variables s_l and the degrees of freedom u_m are given by $l = 1 \dots N$ and $m = 1 \dots n$. It should be noted that the indexes are related to the tensor size concerning the derivative information.

In nonlinear static problems, the equilibrium condition is obtained when the solution satisfies the residuum expression \mathbf{r} defined by

$$\|\mathbf{r}(\mathbf{u}(\mathbf{s}), \mathbf{s})\| \approx 0. \quad (101)$$

¹ As we use the index notation throughout the document, we emphasize that the summation rule occurs when an index appears twice in a term (MALVERN, 1969). It occurs until we reach the total number of degrees of freedom or design variables, according to the case. For instance, an alternative form of Eq. 99 is given by

$$\frac{d\psi}{ds_k} = \frac{\partial\psi}{\partial s_k} + \sum_{j=1}^n \frac{\partial\psi}{\partial u_j} \frac{du_j}{ds_k}. \quad (98)$$

As shown, the upper bound of summation is related to the number of degrees of freedom.

As shown in Eq. 101, this magnitude depends on the nodal displacements, which in turn depend on the design variables. Therefore, by initially considering the direct method, the first-order derivative of the displacement vector with respect to the design variables can be obtained as the solution of the equation system

$$\frac{\partial r_i}{\partial u_j} \frac{du_j}{ds_k} = -\frac{\partial r_i}{\partial s_k}, \quad (102)$$

where the corresponding index of the residuum is described by $i = 1 \dots n$. Similarly, we calculate the second-order displacement sensitivity analysis using the expression

$$\frac{\partial r_i}{\partial u_j} \frac{d^2 u_j}{ds_l ds_k} = -\left(\frac{\partial^2 r_i}{\partial s_l \partial s_k} + \frac{\partial^2 r_i}{\partial u_j \partial s_k} \frac{du_j}{ds_l} + \frac{\partial^2 r_i}{\partial s_l \partial u_j} \frac{du_j}{ds_k} + \frac{\partial^2 r_i}{\partial u_m \partial u_j} \frac{du_m}{ds_l} \frac{du_j}{ds_k} \right). \quad (103)$$

As noted in Eqs. 102 and 103, the derivative calculation of the residuum expression plays a fundamental role to obtain the displacement sensitivity. Although the analytical expressions are the epitome of accuracy and computational performance, this approach lacks generality. The adaptation ability is an important aspect of numerical methods in engineering.

In this context, numerical differentiation methods have been considered to obtain such information in general cases. The finite difference method typically introduces errors during the calculations according to the perturbation value, which can significantly affect convergence. To overcome such an important issue, while keeping interesting generality qualities, we propose the hyper-dual scheme to obtain the highlighted terms in Eqs. 102 and 103.

4.1.1 Typical simplifications

The expressions related to the sensitivity analysis were derived in a general form considering nonlinear problems. To facilitate the presentation of our method, we discuss typical simplifications in specific terms, which are allowed in some practical cases.

As for the expressions concerning the objective function (Eqs. 99 and 100), we can assume that the explicit derivatives with respect to the design variables are null, as also considered by Haftka and Gürdal (1991). As observed by Haftka and Gürdal (1991) and Kleiber (1997), these derivatives are usually zero or easy to calculate analytically. Assuming the former case, the mathematical expressions regarding the derivatives of the objective function with respect to the design variables can be described by

$$\frac{d\psi}{ds_k} = \frac{\partial \psi}{\partial u_j} \frac{du_j}{ds_k} \quad \text{and} \quad (104)$$

$$\frac{d^2 \psi}{ds_l ds_k} = \frac{\partial^2 \psi}{\partial u_m \partial u_j} \frac{du_m}{ds_l} \frac{du_j}{ds_k} + \frac{\partial \psi}{\partial u_j} \frac{d^2 u_j}{ds_l ds_k}. \quad (105)$$

We can also consider a set of simplifications regarding Eqs. 102 and 103. For instance, the external force can change along the increments, as observed in follower loads, such as pressure. However, in ordinary load cases like concentrated nodal forces, there is no influence

of the displacement solution on the external force. In this case, the derivatives of the load vector with respect to displacements are null. Moreover, similar simplifications can be performed considering that the external forces are not affected due to design variable changes². Considering the definition of the tangent stiffness matrix as the derivative of the internal force with respect to the degrees of freedom, we can write

$$\frac{\partial^2 r_i}{\partial s_k \partial u_j} = \frac{\partial^2 q_i}{\partial s_k \partial u_j} - \frac{\partial^2 f_i}{\partial s_k \partial u_j} = \frac{\partial}{\partial s_k} \left(\frac{\partial q_i}{\partial u_j} \right) = \frac{\partial K_{Tij}}{\partial s_k}, \quad (107)$$

where q_i is the internal force, f_i is the external force, and K_{Tij} is the tangent stiffness matrix.

Hence, assuming that the external force is neither affected by the displacement values during the analysis nor by the design variables, the expressions describing the displacement sensitivity (Eqs. 102 and 103) are now presented as

$$K_{Tij} \frac{du_j}{ds_k} = - \frac{\partial q_i}{\partial s_k} \quad \text{and} \quad (108)$$

$$K_{Tij} \frac{d^2 u_j}{ds_l ds_k} = - \left(\frac{\partial^2 q_i}{\partial s_l \partial s_k} + \frac{\partial K_{Tij}}{\partial s_k} \frac{du_j}{ds_l} + \frac{\partial K_{Tij}}{\partial s_l} \frac{du_j}{ds_k} + \frac{\partial^2 q_i}{\partial u_m \partial u_j} \frac{du_m}{ds_l} \frac{du_j}{ds_k} \right). \quad (109)$$

4.1.2 Required derivative terms

Instead of using analytical expressions or rough approximations, we aim to calculate the highlighted derivative terms using the hyper-dual approach presented in Chapter 2. Thus, we provide an accurate and general procedure to evaluate the derivatives of tensor-valued functions, particularly the internal force vector and the tangent stiffness matrix. We summarize the required derivatives in Tab. 9.

Table 9 – Required derivative information in nonlinear cases.

Derivative	Tangent stiffness matrix	Internal force	
1st-order	$\frac{\partial K_{Tij}}{\partial s_k}$	$\frac{\partial q_i}{\partial s_k}$	–
2nd-order	–	$\frac{\partial^2 q_i}{\partial s_l \partial s_k}$	$\frac{\partial^2 q_i}{\partial u_m \partial u_j}$

Source – Own author.

² These simplifications are mathematically described by

$$\frac{\partial r_i}{\partial u_j} = \frac{\partial q_i}{\partial u_j} - \frac{\partial f_i}{\partial u_j}, \quad \frac{\partial^2 r_i}{\partial u_m \partial u_j} = \frac{\partial^2 q_i}{\partial u_m \partial u_j} - \frac{\partial^2 f_i}{\partial u_m \partial u_j}, \quad \frac{\partial r_i}{\partial s_k} = \frac{\partial q_i}{\partial s_k} - \frac{\partial f_i}{\partial s_k} \quad \text{and} \quad \frac{\partial^2 r_i}{\partial s_l \partial s_k} = \frac{\partial^2 q_i}{\partial s_l \partial s_k} - \frac{\partial^2 f_i}{\partial s_l \partial s_k}. \quad (106)$$

4.2 SEMI-ANALYTICAL METHOD IN NONLINEAR PROBLEMS

Previously, in Section 4.1, we presented the global expressions concerning the direct method of design sensitivity analysis. In this context, we identified the required terms to obtain the second-order displacement sensitivity. Now, in this section, we present an element-level method (JIN et al., 2010) to obtain the derivatives using hyper-dual numbers. Since this localized procedure confines the hyper-dual operations inside the element loop, the size of the tensors are limited to the number of degrees of freedom and design variables of each element, which is convenient in terms of the computational effort.

In summary, we convert the internal force and the tangent stiffness into hyper-dual numbers, aiming to extract the required derivative information regarding each element. We mount the global variables using an assembly operator, similarly as observed during the calculation of the global stiffness matrix in an ordinary finite element code, and eventually evaluate the sensitivity of the structure.

4.2.1 Hyper-dual form of internal force and tangent stiffness

Using the conversion steps presented in Section 2.2.2, we obtain the hyper-dual form of the internal force \mathbf{q}_s^e and the tangent stiffness matrix $\mathbf{K}_{T_s}^e$, as given by

$$\mathbf{K}_{T_s}^e = (\mathbf{K}_T)^e + \left(\frac{\partial \mathbf{K}_T}{\partial \mathbf{s}}\right)^e \epsilon_1 + \left(\frac{\partial \mathbf{K}_T}{\partial \mathbf{s}}\right)^e \epsilon_2 + \left(\frac{\partial^2 \mathbf{K}_T}{\partial \mathbf{s}^2}\right)^e \epsilon_1 \epsilon_2 \quad \text{and} \quad (110)$$

$$\mathbf{q}_s^e = (\mathbf{q})^e + \left(\frac{\partial \mathbf{q}}{\partial \mathbf{s}}\right)^e \epsilon_1 + \left(\frac{\partial \mathbf{q}}{\partial \mathbf{s}}\right)^e \epsilon_2 + \left(\frac{\partial^2 \mathbf{q}}{\partial \mathbf{s}^2}\right)^e \epsilon_1 \epsilon_2. \quad (111)$$

The superscript \mathbf{e} indicates that the expressions are related to the element information, whereas the subscript \mathbf{s} denotes that the perturbation was applied on the design variables. Then, using the mathematical operator \mathfrak{S} , the derivatives with respect to design variables are extracted from the hyper-dual variables, as indicated by

$$\left(\frac{\partial \mathbf{q}}{\partial \mathbf{s}}\right)^e = \mathfrak{S}_{\epsilon_1} [\mathbf{q}_s^e], \quad \left(\frac{\partial^2 \mathbf{q}}{\partial \mathbf{s}^2}\right)^e = \mathfrak{S}_{\epsilon_1 \epsilon_2} [\mathbf{q}_s^e] \quad \text{and} \quad \left(\frac{\partial \mathbf{K}_T}{\partial \mathbf{s}}\right)^e = \mathfrak{S}_{\epsilon_1} [\mathbf{K}_{T_s}^e]. \quad (112)$$

It is noteworthy that, despite the fact that we calculated the term $\mathfrak{S}_{\epsilon_1 \epsilon_2} [\mathbf{K}_{T_s}^e]$, we neglect this term containing the second-order derivative. As shown in Tab. 9, this information is not required.

Additionally, although the hyper-dual scheme was initially intended for the calculation of the derivatives with respect to the design variables, we can also evaluate the derivatives with respect to the degrees of freedom. Using the hyper-dual procedure, we obtain the derivatives of the internal force, as indicated by

$$\mathbf{q}_u^e = (\mathbf{q})^e + \left(\frac{\partial \mathbf{q}}{\partial \mathbf{u}}\right)^e \epsilon_1 + \left(\frac{\partial \mathbf{q}}{\partial \mathbf{u}}\right)^e \epsilon_2 + \left(\frac{\partial^2 \mathbf{q}}{\partial \mathbf{u}^2}\right)^e \epsilon_1 \epsilon_2, \quad (113)$$

where \mathbf{q}_u^e refers to the hyper-dual form of the element expression of the internal force considering the perturbation in the degrees of freedom. We extract the required derivative according to the following expression:

$$\left(\frac{\partial^2 \mathbf{q}}{\partial \mathbf{u}^2}\right)^e = \mathfrak{S}_{\epsilon_1 \epsilon_2} [\mathbf{q}_u^e]. \quad (114)$$

It should be noted that the calculation related to the terms $\mathfrak{S}_{\epsilon_1}$ and $\mathfrak{S}_{\epsilon_2}$ is redundant since it corresponds to tangent stiffness matrix³.

As the obtained derivatives refer to element-level information, we use an assembly operator⁴ to mount the global variables, as described by

$$\begin{aligned} \frac{\partial \mathbf{q}}{\partial \mathbf{s}} &= \bigwedge_{e=1}^{e_{max}} \left(\frac{\partial \mathbf{q}}{\partial \mathbf{s}}\right)^e, & \frac{\partial^2 \mathbf{q}}{\partial \mathbf{s}^2} &= \bigwedge_{e=1}^{e_{max}} \left(\frac{\partial^2 \mathbf{q}}{\partial \mathbf{s}^2}\right)^e, \\ \frac{\partial \mathbf{K}_T}{\partial \mathbf{s}} &= \bigwedge_{e=1}^{e_{max}} \left(\frac{\partial \mathbf{K}_T}{\partial \mathbf{s}}\right)^e, & \text{and} & \quad \frac{\partial^2 \mathbf{q}}{\partial \mathbf{u}^2} = \bigwedge_{e=1}^{e_{max}} \left(\frac{\partial^2 \mathbf{q}}{\partial \mathbf{u}^2}\right)^e, \end{aligned} \quad (117)$$

where e is the element number and e_{max} is the total number of elements. In this case, the assembly operator \bigwedge considers the mapping of the degrees of freedom and the design variables. Finally, as we calculate the required derivative terms indicated in Tab. 9, we can apply this global derivative information into Eqs. 108 and 109 to evaluate the displacement sensitivity analysis.

Besides design optimization, displacement sensitivity is a valuable information in a product development context, since many failure modes are closely related to structural stiffness. This information is relevant to guide engineering decisions in terms of design, manufacturing, or reliability aspects. To illustrate the use of hyper-dual numbers in this context, we present a case study in Appendix C.

4.2.2 Sensitivity analysis algorithm

To describe this proposal, we present the main procedures in Fig. 12. We illustrate how a general finite element code could be modified to accommodate the proposed method for the design sensitivity analysis. It is noteworthy that this description is valid for path independent

³ For a given element, this term represents the tangent stiffness matrix, as

$$\mathfrak{S}_{\epsilon_1} [\mathbf{q}_u^e] = \mathfrak{S}_{\epsilon_2} [\mathbf{q}_u^e] = \left(\frac{\partial \mathbf{q}}{\partial \mathbf{u}}\right)^e = (\mathbf{K}_T)^e. \quad (115)$$

⁴ The assembly operator \bigwedge is typically applied to obtain the global internal force and the global tangent stiffness matrix in a finite element code, as represented by

$$\mathbf{q} = \bigwedge_{e=1}^{e_{max}} (\mathbf{q})^e \quad \text{and} \quad \mathbf{K}_T = \bigwedge_{e=1}^{e_{max}} (\mathbf{K}_T)^e, \quad (116)$$

where \mathbf{q} and \mathbf{K}_T are global variables, e is the element number, e_{max} is the total number of elements and $(\mathbf{q})^e$ and $(\mathbf{K}_T)^e$ are element variables. The assembly operator considers the element connectivity based on the numbering of the degrees of freedom.

problems, i.e. this formulation does not account for dissipative phenomena, such as material plasticity. An analysis including internal variables can be obtained by an extension of the work proposed by Geovane A. Haveroth and Muñoz-Rojas (2016).

Figure 12 – An algorithm to describe the hyper-dual sensitivity analysis in a finite element code.

```

Require: problem input           ▷ node, element, boundary conditions, loads, material, design variables
1: procedure NONLINEAR FINITE ELEMENT ANALYSIS
2:   u = 0                               ▷ initial trial solution
3:   while |r| > tol do                   ▷ convergence criteria
4:     for  $e \rightarrow 1, e_{max}$  do                 ▷ evaluate each element
5:        $(\mathbf{q})^e, (\mathbf{K}_T)^e$                  ▷ internal force and tangent stiffness
6:        $\mathbf{q} = \bigwedge (\mathbf{q})^e, \mathbf{K}_T = \bigwedge (\mathbf{K}_T)^e$    ▷ global assembly operator
7:     end for                               ▷ global variables
8:      $r_i = q_i - f_i$                          ▷ residuum calculation
9:      $K_{Tij} \Delta u_j = -r_i$                  ▷ update the displacement for the next iteration
10:  end while                               ▷ finite element convergence
11: end procedure
12: procedure DISPLACEMENT SENSITIVITY ANALYSIS
13:  for  $e \rightarrow 1, e_{max}$  do                   ▷ evaluate each element
14:     $\mathbf{s}^e$                                ▷ hyper-dual form of design variables with unit perturbation
15:     $\mathbf{q}_s^e, \mathbf{K}_{T_s}^e$                  ▷ hyper-dual form of internal force and tangent stiffness
16:     $\left(\frac{\partial \mathbf{q}}{\partial \mathbf{s}}\right)^e = \Im_{\epsilon_1} [\mathbf{q}_s^e], \left(\frac{\partial^2 \mathbf{q}}{\partial \mathbf{s}^2}\right)^e = \Im_{\epsilon_1 \epsilon_2} [\mathbf{q}_s^e], \left(\frac{\partial \mathbf{K}_T}{\partial \mathbf{s}}\right)^e = \Im_{\epsilon_1} [\mathbf{K}_{T_s}^e]$    ▷ extract non-real parts
17:     $\frac{\partial \mathbf{q}}{\partial \mathbf{s}} = \bigwedge \left(\frac{\partial \mathbf{q}}{\partial \mathbf{s}}\right)^e, \frac{\partial^2 \mathbf{q}}{\partial \mathbf{s}^2} = \bigwedge \left(\frac{\partial^2 \mathbf{q}}{\partial \mathbf{s}^2}\right)^e, \frac{\partial \mathbf{K}_T}{\partial \mathbf{s}} = \bigwedge \left(\frac{\partial \mathbf{K}_T}{\partial \mathbf{s}}\right)^e$    ▷ global assembly operator
18:     $\mathbf{u}^e$                                ▷ hyper-dual form of displacements with unit perturbation
19:     $\mathbf{q}_u^e$                                ▷ hyper-dual form of internal force
20:     $\left(\frac{\partial^2 \mathbf{q}}{\partial \mathbf{u}^2}\right)^e = \Im_{\epsilon_1 \epsilon_2} [\mathbf{q}_u^e]$    ▷ extract non-real parts
21:     $\frac{\partial^2 \mathbf{q}}{\partial \mathbf{u}^2} = \bigwedge \left(\frac{\partial^2 \mathbf{q}}{\partial \mathbf{u}^2}\right)^e,$    ▷ global assembly operator
22:  end for                               ▷ global variables
23:   $K_{Tij} \frac{du_j}{ds_k} = -\frac{\partial q_i}{\partial s_k}$    ▷ 1st order analysis
24:   $K_{Tij} \frac{d^2 u_j}{ds_l ds_k} = -\left(\frac{\partial^2 q_i}{\partial s_l \partial s_k} + \frac{\partial K_{Tij}}{\partial s_k} \frac{du_j}{ds_l} + \frac{\partial K_{Tij}}{\partial s_l} \frac{du_j}{ds_k} + \frac{\partial^2 q_i}{\partial u_m \partial u_j} \frac{du_m}{ds_l} \frac{du_j}{ds_k}\right)$    ▷ 2nd order analysis
25: end procedure

```

Source – Own author.

The finite element method considers that the global variables regarding internal force and tangent stiffness are obtained by considering the contribution of each element. These calculations are performed separately for each element (line 5) and an assembly operator mounts global information based on the connectivity of elements and degrees of freedom (line 6). The residuum vector r_i is defined by the difference between the internal and the external force (line 8). Once the nodal displacement solution that allows the Newton-Raphson convergence is obtained (line 3), the structural analysis is completed and the sensitivity analysis begins (line 12).

As the design variables for a given element are properly identified, they are converted into scalar hyper-dual numbers with unit perturbation in ϵ_1 and ϵ_2 axes (line 14). The hyper-dual form of internal force and tangent stiffness matrix is obtained using the hyper-dual arithmetics

(line 15). The gradient and Hessian information are extracted from the hyper-dual variables using the mathematical operator \mathfrak{S} (line 16). The element contribution is mounted into the global variables using an assembly operator (line 17).

Additionally, to obtain the second-order derivative of internal force with respect to the degrees of freedom, the nodal displacement is converted into hyper-dual numbers with unit perturbation (line 18). We obtain this derivative information by extracting the non-real part (line 20) of the hyper-dual variable (line 19); the global information is mounted at line 21. These processes are repeated for all elements that describe the structure.

The nodal displacement update involves the solution of the expression indicated at line 9. This process typically demands the factorization of the global tangent stiffness matrix. This same information is necessary to obtain the first and second-order displacement sensitivity analysis, as observed at lines 23 and 24, respectively. Therefore, we can reduce the computational costs by reusing this factorized information during the sensitivity analysis. It should be emphasized that it is not necessary to convert the entire program into hyper-dual numbers, which would unnecessarily increase the computational effort. The finite element solution related to the search of nodal displacements is provided using real numbers. Moreover, it is important to highlight that the hyper-dual routine is confined inside the element loop. This strategy allows reductions in terms of computational and storage costs due to the smaller size of the tensor variables.

4.3 DIAGONAL PARTICULARIZATION IN LINEAR PROBLEMS

In this section, we examine the design sensitivity expressions considering two particularizations: sensitivity analysis considering only the diagonal terms of the Hessian matrix and linear structural behavior. We present the sensitivity expressions considering the direct and the adjoint method in Sections 4.3.1 and 4.3.2, respectively. In Section 4.3.3, we summarize the required derivative terms for a second-order analysis. These topics comprise the general context wherein our automatic differentiation method fits. In Section 4.3.4 we address the derivative calculation using the diagonal hyper-dual variant.

4.3.1 Direct method

As presented in Section 4.1, the first and second-order derivatives of the objective function with respect to the design variables are given by

$$\frac{d\psi}{ds_k} = \frac{\partial\psi}{\partial u_j} \frac{du_j}{ds_k} \quad \text{and} \quad (118)$$

$$\frac{d^2\psi}{ds_l ds_k} = \frac{\partial\psi}{\partial u_j} \underbrace{\frac{d^2 u_j}{ds_l ds_k}}_{\text{costly}} + \frac{\partial^2\psi}{\partial u_m \partial u_j} \frac{du_m}{ds_l} \frac{du_j}{ds_k}, \quad (119)$$

where ψ is the objective function, \mathbf{s} is the vector containing the design variables, \mathbf{u} is the nodal displacement vector; the indexes are represented by $k = 1 \dots N$, $l = 1 \dots N$, $j = 1 \dots n$ and $m = 1 \dots n$, where N and n are the numbers of design variables and degrees of freedom, respectively.

As indicated in Eq. 119, the second-order derivative of the displacement with respect to the design variables is a concern in terms of computational costs, as this third-order tensor requires the solution of a dense equation system. As an attempt to mitigate this effort, we can focus the calculation on the diagonal terms of the Hessian matrix. Thus, we represent the latter expression as

$$\frac{d^2\psi}{ds_b^2} = \frac{\partial\psi}{\partial u_j} \frac{d^2u_j}{ds_b^2} + \frac{\partial^2\psi}{\partial u_m \partial u_j} \frac{du_m}{ds_l} \frac{du_j}{ds_k} \delta_{klb}, \quad \text{where } \delta_{klb} = \begin{cases} 1, & \text{if } k = l = b \\ 0, & \text{otherwise} \end{cases}. \quad (120)$$

In this particular case, we introduce another index related to the design variables, as given by $b = 1 \dots N$.

Besides facilitating the calculation of the displacement sensitivity, by considering this diagonal particularization, we represent the second-order derivative of the objective function in a vector form, instead of the usual matrix representation. Hence, by neglecting the off-diagonal terms representing the mixed derivatives, the sensitivity expression indicated in Eq. 120 allows an important reduction in terms of processing and storage.

In nonlinear structural problems, we typically conduct an iterative process to identify the nodal displacement solution using finite elements. The convergence criterium can consider the evaluation of the residuum vector \mathbf{r} , represented by

$$\|\mathbf{r}(\mathbf{u}(\mathbf{s}), \mathbf{s})\| \approx 0. \quad (121)$$

Accordingly, assuming the diagonal particularization for the second-order derivative, we obtain the derivatives of the displacement vector with respect to the design variables, as given by

$$\frac{\partial r_i}{\partial u_j} \frac{du_j}{ds_k} = -\frac{\partial r_i}{\partial s_k} \quad \text{and} \quad (122)$$

$$\frac{\partial r_i}{\partial u_j} \frac{d^2u_j}{ds_b^2} = -\frac{\partial^2 r_i}{\partial s_b^2} - \underbrace{\left(\frac{\partial^2 r_i}{\partial u_j \partial s_k} \frac{du_j}{ds_l} + \frac{\partial^2 r_i}{\partial s_l \partial u_j} \frac{du_j}{ds_k} + \frac{\partial^2 r_i}{\partial u_m \partial u_j} \frac{du_m}{ds_l} \frac{du_j}{ds_k} \right)}_{\text{costly}} \delta_{klb}, \quad (123)$$

where r_i represents the i -th term of the residuum vector, such that $i = 1 \dots n$.

Even considering the diagonal particularization, the computational effort concerning the second-order sensitivity of nonlinear problems is still high. As indicated in Eq. 123, it inconveniently requires the calculation and the storage of a third-order tensor referring to the second-order derivatives of the residuum with respect to the degrees of freedom. We can interpret this information as the derivative of the tangent stiffness matrix with respect to the degrees of freedom.

It should be noted that these expressions refer to global variables, which consider all degrees of freedom and design variables of the problem. In this context, the second-order sensitivity study typically requires the storage of global third-order tensors, which might impose memory requirements in large problems, regardless of the derivative scheme.

Alternatively, one could approximate the sensitivity expressions by simply neglecting the last term of Eq. 123. As a drawback, the convergence behavior during the optimization process might be affected due to this loss of information. However, the severity of this issue depends on each structural problem. For instance, if the nonlinear behavior of a structure is not so prominent, i.e. the changes of the tangent stiffness during the load increments are subtle, this sensitivity still might be a good representation.

If we consider linear structural problems, we can accurately evaluate the second-order sensitivity focusing on the diagonal terms⁵. In this case, we avoid the aforementioned issues related to the nonlinear formulation, since we do not eliminate any term. The displacement sensitivity expressions considering linear structures are given by

$$K_{ij} \frac{du_j}{ds_k} = \frac{df_i}{ds_k} - \frac{dK_{ij}}{ds_k} u_j \quad \text{and} \quad (124)$$

$$K_{ij} \frac{d^2 u_j}{ds_b^2} = \frac{d^2 f_i}{ds_b^2} - \frac{d^2 K_{ij}}{ds_b^2} u_j - \left(\frac{dK_{ij}}{ds_k} \frac{du_j}{ds_l} + \frac{dK_{ij}}{ds_l} \frac{du_j}{ds_k} \right) \delta_{klb}, \quad (125)$$

where K_{ij} is the stiffness matrix and f_i is the external force vector. As noted in the obtained expressions, their corresponding derivatives with respect to the design variables play an important role in this sensitivity analysis. Particularly, we emphasize that the second-order derivatives concern only the diagonal terms.

4.3.2 Adjoint method

Alternatively, we can also evaluate the diagonal terms of the design sensitivity expression using the adjoint method. For the sake of conciseness, we focus on the second-order expression

⁵ The second-order displacement sensitivity including the mixed derivative terms is represented by

$$K_{ij} \frac{d^2 u_j}{ds_l ds_k} = \frac{d^2 f_i}{ds_l ds_k} - \frac{d^2 K_{ij}}{ds_l ds_k} u_j - \frac{dK_{ij}}{ds_k} \frac{du_j}{ds_l} - \frac{dK_{ij}}{ds_l} \frac{du_j}{ds_k}.$$

considering linear structural problems⁶, which is given by

$$\frac{d^2\psi}{ds_b^2} = -\omega_i \left(\frac{dK_{ij}}{ds_k} \frac{du_j}{ds_l} + \frac{dK_{ij}}{ds_l} \frac{du_j}{ds_k} \right) \delta_{klb} + \omega_i \left(\frac{d^2 f_i}{ds_b^2} - \frac{d^2 K_{ij}}{ds_b^2} u_j \right) + \frac{\partial^2 \psi}{\partial u_m \partial u_j} \frac{du_m}{ds_l} \frac{du_j}{ds_k} \delta_{klb}, \quad (129)$$

where ω_i represents the terms of the adjoint vector, obtained by solving the equation system indicated by

$$K_{ij}\omega_i = \frac{\partial \psi}{\partial u_j}. \quad (130)$$

Based on this brief concerning sensitivity analysis, we can establish a convenient strategy to reduce the computational effort of a second-order study. The significant effort involving the adjoint method is related to the solution of Eqs. 124 and 130, whereas the direct method requires the solution of Eqs. 124 and 125. Thus, based on this comparative evaluation, even considering the diagonal particularization, the adjoint method is still preferable (HAFTKA; GÜRDAL, 1991).

4.3.3 Required derivative terms

In Sections 4.3.1 and 4.3.2, we indicated the relevance of particular derivative terms in a design sensitivity analysis, either using the direct or the adjoint method. As highlighted in Eqs. 124, 125 and 129, both strategies require the same derivative information regarding the discretized structural problem, as summarized in Tab. 10.

In this context, Haftka and Gürdal (1991) highlight that the analytical derivatives of the stiffness matrix and load vectors with respect to design variables are often difficult to obtain, especially for shape design variables. Since it would provide generality, an automatic differentiation scheme could bring interesting benefits to an already existing code.

⁶ We represent the general expression considering the adjoint method as

$$\frac{d\psi}{ds_k} = \frac{\partial \psi}{\partial s_k} + \omega_i \left(\frac{df_i}{ds_k} - \frac{dK_{ij}}{ds_k} u_j \right). \quad (126)$$

To obtain the second-order expression, we evaluate the derivative of Eq. 126 with respect to the design variables, which is given by

$$\frac{d^2\psi}{ds_l ds_k} = \frac{\partial^2 \psi}{\partial s_l \partial s_k} + \frac{\partial^2 \psi}{\partial u_j \partial s_k} \frac{du_j}{ds_l} + \overbrace{\frac{d\omega_i}{ds_l}}^{\text{Eq. 128}} \overbrace{\left(\frac{df_i}{ds_k} - \frac{dK_{ij}}{ds_k} u_j \right)}^{\text{Eq. 124}} + \omega_i \left(\frac{d^2 f_i}{ds_l ds_k} - \frac{d^2 K_{ij}}{ds_l ds_k} u_j - \frac{dK_{ij}}{ds_k} \frac{du_j}{ds_l} \right), \quad (127)$$

where the derivative of the adjoint vector expression (Eq. 130) with respect to the design variables is

$$K_{ij} \frac{d\omega_i}{ds_l} = \frac{\partial^2 \psi}{\partial s_l \partial u_j} + \frac{\partial^2 \psi}{\partial u_m \partial u_j} \frac{du_m}{ds_l} - \frac{dK_{ij}}{ds_l} \omega_i. \quad (128)$$

We obtain the second-order sensitivity expression considering the adjoint method using Eqs. 128 and 124, as indicated in Eq. 127.

Table 10 – Required derivative information in linear cases.

Derivative	Stiffness matrix	External force
1st-order	$\frac{dK_{ij}}{ds_k}$	$\frac{df_i}{ds_k}$
2nd-order	$\frac{d^2K_{ij}}{ds_b^2}$	$\frac{d^2f_i}{ds_b^2}$

Source – Own author.

Therefore, we propose the use of hyper-dual numbers to obtain the derivatives of the stiffness matrix and the external force with respect to the design variables. Due to the specificity concerning the diagonal terms of the second-order derivatives, the development of a new hyper-dual variant becomes convenient.

4.3.4 Element-level design sensitivity analysis

Previously, in Sections 4.3.1 and 4.3.2, we described the diagonal form of second-order design sensitivity analysis considering linear structural problems. These expressions are global, i.e. they represent the entire structural problem. To reduce the computational effort related to the derivative calculation, we confine the hyper-dual conversion at the element level. In this section, we address the calculation of the required derivative terms discussed in Section 4.3.3 by using the diagonal hyper-dual scheme.

4.3.4.1 Hyper-dual form of stiffness matrix and external force

To obtain the derivatives indicated in Tab. 10, we convert the stiffness matrix and the external force into diagonal hyper-dual numbers; these variables are described by $\tilde{\mathbf{K}}^e$ and $\tilde{\mathbf{f}}^e$, respectively. It should be noted that these variables refer to element-level information, as represented by the superscript *e*. Considering the diagonal hyper-dual form, the non-real terms contain the required derivative information for a given element, as indicated by

$$\tilde{\mathbf{K}}^e = (\mathbf{K})^e + \left(\frac{d\mathbf{K}}{ds}\right)^e \epsilon_1 + \left(\frac{d\mathbf{K}}{ds}\right)^e \epsilon_2 + \left(D \left[\frac{d^2\mathbf{K}}{ds^2}\right]\right)^e \epsilon_1\epsilon_2 \quad \text{and} \quad (131)$$

$$\tilde{\mathbf{f}}^e = (\mathbf{f})^e + \left(\frac{d\mathbf{f}}{ds}\right)^e \epsilon_1 + \left(\frac{d\mathbf{f}}{ds}\right)^e \epsilon_2 + \left(D \left[\frac{d^2\mathbf{f}}{ds^2}\right]\right)^e \epsilon_1\epsilon_2, \quad (132)$$

where $\left(D \left[\frac{d^2\mathbf{K}}{ds^2}\right]\right)^e$ and $\left(D \left[\frac{d^2\mathbf{f}}{ds^2}\right]\right)^e$ represent the diagonal terms of the second-order derivatives of the stiffness matrix and the load vector, respectively. *D* represents a mathematical operator that extracts the diagonal terms from a tensor.

It is important to highlight that these expressions regarding the stiffness matrix and the external force are general and may comprise a variety of finite element formulations. Opportunely, the hyper-dual scheme, established as an automatic differentiation tool, provides the required generality to easily obtain the derivatives.

4.3.4.2 Extracting the derivatives from the hyper-dual variable

After the hyper-dual conversion of the element expressions, we extract the derivatives as real variables, as shown by

$$\begin{aligned} \mathfrak{S}_{\epsilon_1} [\tilde{\mathbf{K}}^e] &= \left(\frac{d\mathbf{K}}{ds} \right)^e, & \mathfrak{S}_{\epsilon_1} [\tilde{\mathbf{f}}^e] &= \left(\frac{d\mathbf{f}}{ds} \right)^e, \\ \mathfrak{S}_{\epsilon_1\epsilon_2} [\tilde{\mathbf{K}}^e] &= \left(D \left[\frac{d^2\mathbf{K}}{ds^2} \right] \right)^e, & \text{and} & \quad \mathfrak{S}_{\epsilon_1\epsilon_2} [\tilde{\mathbf{f}}^e] = \left(D \left[\frac{d^2\mathbf{f}}{ds^2} \right] \right)^e. \end{aligned} \quad (133)$$

As shown in Eq. 133, the obtained derivative information concerns a single element. In this context, it is important to highlight that the size of the tensors and the hyper-dual operations correspond to the element-level expressions, rather than global variables describing the entire structural problem. Consequently, aiming for a rational use of the resources, we apply the hyper-dual operations separately for each element, which is important to reduce the overall effort.

To illustrate the hyper-dual conversion and the derivative extraction of tensor-valued functions using a straightforward example, we present the conversion steps of the stiffness matrix of a truss element in Section 2.3.2. It should be noted that the specific conversion procedures depend on the formulation of a given element type since each function requires a certain set of arithmetic rules for its construction.

4.3.4.3 Global assembly

We evaluate the derivatives regarding all elements by considering an element loop in the computer code. Based on this element-level information, we mount the global variables using an assembly operator, similar to the function that returns the global stiffness matrix in an ordinary finite element code. The assembly operation of the global derivative information is represented by

$$\begin{aligned} \frac{d\mathbf{K}}{ds} &= \bigwedge_{e=1}^{e_{max}} \left(\frac{d\mathbf{K}}{ds} \right)^e, & \frac{d\mathbf{f}}{ds} &= \bigwedge_{e=1}^{e_{max}} \left(\frac{d\mathbf{f}}{ds} \right)^e, \\ D \left[\frac{d^2\mathbf{K}}{ds^2} \right] &= \bigwedge_{e=1}^{e_{max}} \left(D \left[\frac{d^2\mathbf{K}}{ds^2} \right] \right)^e, & \text{and} & \quad D \left[\frac{d^2\mathbf{f}}{ds^2} \right] = \bigwedge_{e=1}^{e_{max}} \left(D \left[\frac{d^2\mathbf{f}}{ds^2} \right] \right)^e, \end{aligned} \quad (134)$$

where \bigwedge represents the assembly operator to mount the global variables and e_{max} is the total number of elements.

As we complete the calculation of the required global derivative terms presented in Tab. 10, we can apply them in Eqs. 124, 125, and 129 to obtain the second-order design sensitivity. Therefore, we described the use of diagonal hyper-dual numbers as an automatic differentiation tool to accurately obtain the diagonal terms of the Hessian matrix.

5 DISCUSSION

In this section, we discuss the findings, drawbacks, limitations, and recommendations for a future research. The contents related to Tensor hyper-dual numbers (Chapter 2), Hyper-dual hyperelasticity (Chapter 3), and Hyper-dual sensitivity analysis (Chapter 4) are described in Sections 5.1, 5.2, and 5.3, respectively.

5.1 TENSOR HYPER-DUAL NUMBERS

We described the use of hyper-dual numbers as a general differentiation tool to evaluate tensor-valued functions, as an extension of the work of Jeffrey Alan Fike (2013). This automatic scheme allows accurate calculations of both first and second-order derivatives, providing important results in many scientific fields, including computational solid mechanics.

The obtained expressions for the arithmetic operators for tensor hyper-dual numbers were consistent with the conclusions stated by Cohen and Shoham (2018); these authors suggested that the algebra of hyper-dual numbers and hyper-dual vectors of higher order are described by the same basic formulae. This argument corroborates our findings, as the expressions presented in Section 2.2.1 were inspired by the arithmetics of scalar hyper-dual functions provided by Jeffrey Alan Fike (2013). In this work, we implemented the mathematical operators in a computer code and validated the results using analytical references.

We found that the mathematical representation of the arithmetic rules using the index notation form facilitated the implementation process. This format was convenient for the application of the summation rule during the tensor contractions, for instance. Additionally, the hyper-dual implementation in an existing finite element code was straightforward due to the operator overloading technique. The use of virtually the same code syntax established a black-box system for the derivative calculation. During the implementation, we noted that each hyper-dual arithmetic operator involved a large number of internal procedures, which can promote a significant increase in terms of computational costs when compared to real number operators, especially when higher-order tensors take place.

The tensor formulation facilitates the use of the hyper-dual scheme when a code is already written in terms of matrices, a common assumption in many numerical codes. Otherwise, if we consider only the formulation concerning scalar functions, it would be necessary to rewrite the main program in terms of each Cartesian component, which can require a significant effort. As we develop the hyper-dual formulations considering scalar, vector, and tensor variables, we provide a general tool that can accommodate various situations.

We also developed a hyper-dual variant that focuses on the diagonal terms of the second-order derivatives. The diagonal formulation can promote a significant reduction in terms of computational effort, as we reduce the tensor order and the number of operations related to the non-real term contained in axis $\epsilon_1\epsilon_2$. Roughly, when compared to the original tensor hyper-dual formulation, this diagonal variant provides second-order derivative information at

the cost of the first-order terms.

Future work could expand the collection of arithmetic rules of tensor hyper-dual numbers. It would contribute to a comprehensive automatic differentiation tool, allowing the hyper-dual conversion of a wider range of functions. As for the diagonal formulation, one should follow the same adaptation principles presented in this work. In terms of implementation strategies, future work could exploit the sparsity and the symmetry of the higher-order tensors as an attempt to improve the computational performance. For instance, since Hessian shows similar aspects of stiffness matrix in terms of symmetry¹ and sparsity, one could apply an analogous process to reduce the computational cost and storage of the hyper-dual terms. In this context, Dhatt et al. (2012) showed numerical procedures related to the treatment of band matrices, such as the skyline storage; Dhatt and Touzot (1984) described a Fortran code for matrix storage and treatment. Gebremedhin et al. (2009) studied an efficient implementation of sparse Hessians using automatic differentiation; Petra et al. (2018) implemented an algorithm intended for large-scale sparse Hessians in Julia.

Additionally, a study on parallel processing strategies for tensor hyper-dual numbers could bring important advances; in this context, Fike (FIKE, Jeffrey Alan, 2013) evaluated a GPU² parallel program development using CUDA³. The application of advanced programming techniques might compensate the large number of operations related to the tensor hyper-dual numbers. These coding improvements could bring practical benefits to this numerical derivative scheme, as the costly operations are the main drawback of this method.

The outcomings of this study are primarily related to computational mechanics. As stated by Vigliotti and Auricchio (2020), automatic differentiation provides a streamlined solution in solid mechanics, since it is possible to write a program without explicitly introducing the derivatives in the main code. Furthermore, we can extend the implications of this research to other scientific fields, since differentiation is a transversal topic. Basically, a general-purpose approach for the derivative calculation is useful when we do not know the objective function a priori. For instance, Atilim Gunes Baydin and Pearlmutter (2014) argued that automatic differentiation allows accurate and concise implementations in machine learning applications (BYRD et al., 2011; ERWAY et al., 2020). Since Atilim Gunes Baydin et al. (2018) and Vigliotti and Auricchio (2020) cited dual numbers as a relevant method for the derivative calculation, we assert that our study concerning hyper-dual numbers can also contribute to these important fields of science and technology.

Therefore, we can consider the achievements related to the hyper-dual scheme as a standalone topic since they comprise a general differentiation tool. In short, this scheme provides accurate first and second-order derivatives of tensor-valued functions, with the option to include the mixed second-order terms or not. To promote further applications, in Section A.2

¹ As stated by Ju et al. (1995), although tangent stiffness matrix is typically considered as a symmetric tensor; it becomes unsymmetrical in frictional contact problems.

² GPU – Graphics Processing Unit..

³ CUDA – Compute Unified Device Architecture.

we show our preliminary Fortran code describing the arithmetics of tensor hyper-dual numbers.

5.2 HYPER-DUAL HYPERELASTICITY

This study aimed to describe the use of hyper-dual numbers to obtain stresses and tangent moduli in hyperelastic truss structures. We built numerical models with an increasing number of elements and degrees of freedom for a comparative test considering different strategies for the derivative calculation. As the research focused on numerical methods, we evaluated three main characteristics: generality, accuracy, and computational costs. In this work, we evaluated the latter attribute by checking the processing time of the finite element analysis.

Assuming the hyper-dual implementation with the operator overloading technique, the strain energy function is declared with virtually the same code syntax. Therefore, this strategy provides an interesting generality property, a notable quality in engineering methods. As a consequence of the outstanding accuracy for both first and second-order derivatives, this scheme kept the quadratic rate of convergence in the vicinity of the solution. Due to the number of intermediate operations for the hyper-dual conversion, this method showed a slower performance, when compared to the respective processing time considering the analytical expressions. However, we found that this effect was severely reduced in the larger models.

As the normalized overall processing time showed an asymptotic behavior, we could indicate the model size from which the analytical and the hyper-dual methods are equivalent. The identification of this model size, as a result of our strategy to generate the samples, is important to establish a reference value for this type of numerical model. A truss model containing approximately 30000 elements and 25000 degrees of freedom requires roughly the same processing time, either using the analytical expressions or the hyper-dual procedure. Moreover, we detailed the hyper-dual costs by measuring the processing time at different levels in the program, i.e. the material subroutine and the global tangent stiffness matrix. Therefore, we could track the computational costs related to this alternative derivative scheme in a finite element code.

As for the finite difference method, assuming that the ideal perturbation value might be unattainable, we showed that inaccurate derivative information significantly affects the processing time, due to the increasing number of iterations to converge. Conversely, the hyper-dual procedure did not suffer from such an issue, indicating better predictability attributes in terms of convergence.

Thus, recalling the desired characteristics for a numerical method, we claim that the hyper-dual scheme provides an interesting tool for the development of constitutive models, especially when the derivative expressions are either unavailable or mathematically difficult to obtain. This method allows a proper evaluation of hyperelastic models by simply introducing the strain energy function, which avoids the manipulation of the functions regarding stress and tangent modulus. Besides providing exact results, the derivative calculation is automatic

and efficient in large models. These items comprise the main advantage of the hyper-dual procedure in hyperelastic models.

Additionally, one could consider the use of this scheme in commercial finite element programs that allow user-defined material models. Hence, these findings can promote interesting benefits for the development of cutting-edge material models, particularly when conditional statements occur in such constitutive expressions. In this context, we speculate that the hyper-dual procedure can provide coherent results no matter how close to a discontinuity of the primary function or its derivatives.

The development of numerical tools to evaluate hyperelastic models play an important role in the mechanical design of rubber-like materials. As new materials and constitutive models are constantly developed, we hope that this work could contribute to innovations in the field of product development with numerical simulation tools.

5.3 HYPER-DUAL SENSITIVITY ANALYSIS

In this work, we developed the mathematical expressions of the hyper-dual sensitivity analysis, which provided exact second-order information for the design optimization of nonlinear structures. The proposed algorithm implied the hyper-dual conversion of a particular fragment of the finite element code. Particularly, we confined the hyper-dual representation of the internal force and the tangent stiffness at the element level. As a numerical case study, we applied this procedure to perform a second-order displacement sensitivity analysis.

The hyper-dual sensitivity analysis offered a method with interesting generality qualities. We found that the hyper-dual implementation in an existing finite element code was straightforward since the operator overloading technique ensured a very similar syntax. In this case, once the design variables are defined, the hyper-dual conversion occurs automatically and certain non-real parts containing the derivative information are extracted. Additionally, although we had illustrated the hyper-dual procedure in a particular setting, it can be easily incorporated in a variety of formulations, including other element types and design sensitivity methods, such as adjoint and hybrid. Therefore, this strategy featured important characteristics of numerical methods: accuracy and generality.

In this strategy for nonlinear problems, we observed a drawback related to the fact that the second-order derivative of tangent stiffness with respect to the design variables would be unnecessarily calculated. An insightful solution would be related to a representation of the tangent stiffness expression as dual numbers, rather than hyper-dual numbers, since this method provides accurate first-order derivatives (YU; BLAIR, 2013). Thus, the resulting strategy is a combination of differentiation methods. In addition, regardless of the differentiation scheme, the second-order analysis requires the storage of global third-order tensors and the solution of a dense equation system, imposing important requirements in large problems.

In general, SQP codes often use the BFGS formula to calculate the Hessian of the Lagrange function (GOUVEA; ODLOAK, 1997); in this case, this approximate information

is conveniently defined as a positive definite matrix, an important characteristic in an optimization algorithm. Erleben and Andrews (2019) discussed the effects of definiteness of exact Hessian matrices in inverse kinematic problems. Recalling that the hyper-dual procedure returns analytical results, the exact Hessian may undergo negativeness issues along the optimization process. Although the hyper-dual scheme contributes to accurate derivative calculation, the BFGS method provides a robust approximate solution. Considering Newton's method as the optimization algorithm, we can address this drawback with the algorithmic strategy denoted by Marquadt modification (ARORA, J., 2016).

In order to favor second-order methods in structural optimization, we also evaluated a set of particularizations in the design sensitivity expressions. By focusing on the diagonal terms in a linear structural assumption, we found that the derivatives of the stiffness matrix and the load vector with respect to the design variables were required. We evaluated the diagonal variant of hyper-dual numbers to obtain the exact diagonal terms of the Hessian matrix, either using the direct or the adjoint method.

This diagonal particularization of the second-order sensitivity analysis provides useful information in many practical cases, especially when an optimization problem is diagonally dominant. In this case, the loss of information concerning the mixed derivative terms would not significantly affect the convergence behavior. Andrei (2020) states that diagonal dominance is a common assumption in nonlinear optimization. Thus, by providing accurate diagonal derivative calculations, the outcomings of this study can contribute to optimization studies that rely on diagonal approximations (GROENWOLD, A. et al., 2007; DUYSINX, Pierre et al., 1995, 2001; ZHANG; DUYSINX, 2003; ANDREI, 2019, 2020; VIE, 2016).

Hence, the hyper-dual sensitivity method can be used as a building block for design optimization studies. In addition to applications involving the diagonal information, one could apply the hyper-dual sensitivity procedure in current topology optimization codes (LIU, Q.; PAAVOLA, 2014; ROJAS-LABANDA; STOLPE, 2016; MORALES et al., 2011), since an automatic differentiation scheme would contribute to the generality aspects of the code. In this context, one could investigate the behavior and the performance of the optimization process using hyper-dual numbers. Another study might evaluate the hyper-dual sensitivity scheme using parallel computing; in this case, several element-level derivatives would be calculated simultaneously. Schittkowski (2015) summarized a list of optimization studies using second-order algorithms, facilitating comparative case studies.

Besides the hyper-dual scheme, we can evaluate the derivatives considering other approaches in automatic differentiation. For instance, assuming that the exact second-order derivative is required in a given problem, one could develop the analytical expression describing the first-order derivative and convert it into dual numbers. As a result of this process, the second-order derivative of the original function is obtained. Assuming this strategy, two analytical expressions must be defined beforehand: the original function and its first-order derivative; conversely, the hyper-dual scheme would require only the implementation of the original

expression, which might be convenient in general cases. Therefore, according to the case and the available data, we can define a suitable strategy to evaluate the derivatives.

In this regard, one could evaluate the computational costs of a strategy considering a combination of automatic differentiation modes when compared to the hyper-dual scheme. For instance, what if we use the standard reverse mode associated with the dual number procedure? These comparative studies could elucidate the computational costs of different approaches to automatic differentiation in design sensitivity. A discussion concerning the implementation effort of each method, e.g., the required modifications in a code, should be considered as well.

Nevertheless, we speculate that it is possible to extend the findings of our study regarding constitutive models (Section 3.3, also published as a scientific paper (ENDO et al., 2020)), to design sensitivity problems. We showed that the choice of the differentiation method would become less relevant in terms of computational costs as we increase the model size, which corroborates the results of Tanaka et al. (2015). These studies showed that the hyper-dual scheme can be equivalent to the analytical expressions in terms of both accuracy and processing time. We argue that, as we also evaluate the hyper-dual operations element-wise in design sensitivity, the solution of the equation system would dominate the overall processing time. This future work can provide quantitative discussion and assessment of the computational complexity of hyper-dual numbers in the context of structural and multidisciplinary optimization.

We speculate that the hyper-dual sensitivity method can improve the optimization process in problems with conditional statements, such as structural contact or remeshing. The issue related to problems with discontinuities has been discussed in the literature. Stupkiewicz et al. (2002) presented a direct differentiation approach for shape sensitivity analysis of large deformation frictional contact problems; these authors also included a discussion about the finite difference scheme, in which this approximation of the response could fail to converge due to the application of a perturbation. Since conditional statements generate a discontinuous behavior as either the function value itself is discontinuous or the discontinuity is in the derivatives, Martins et al. (2003) suggested the use of the complex-step method. These authors emphasized that the finite difference method provides incorrect results if the function evaluations are within the discontinuity location. Ubessi and R. J. Marczak (2018) studied the complex-step differentiation to calculate the first-order sensitivity in frictional contact problems. Wilke and Kok (2014) and Banks et al. (2015) also evaluated the complex-step method aiming to compute first-order sensitivity information for discontinuous optimization problems. According to the authors, the ordinary finite difference method is problematic when real steps are taken over a discontinuity. However, despite the accurate results, the complex-step method is focused on the first-order evaluation. Therefore, a second-order study using hyper-dual numbers could contribute to the works of Bonte et al. (2010), Deshak et al. (2017) and Ozturk et al. (2016), who evaluated optimization methods for the forging process.

Additionally, we conjecture that the proposed hyper-dual scheme for design sensitivity analysis might address a numerical pathology observed in the traditional semi-analytical evalua-

tion when the structure suffers large rigid body rotations (KEULEN et al., 2005; BLETZINGER et al., 2008; WANG, W. et al., 2015). Since the complex-step method solved the first-order sensitivity (HAVEROTH, Geovane Augusto et al., 2015), by extension the hyper-dual sensitivity scheme could provide accurate second-order information. Moreover, one could extend the formulation regarding the hyper-dual sensitivity by considering structural problems with dissipation phenomena, such as material plasticity; in this context, Geovane A. Haveroth and Muñoz-Rojas (2016) studied the complex-step counterpart.

6 CONCLUSION

In conclusion, by expanding the current hyper-dual formulation, we provided a general and accurate tool to obtain the sensitivity information in structural problems. The proposed scheme is not limited to a particular setting, being able to adjust itself to a myriad of conditions. Because of the findings of this paper, we can automatically evaluate the design sensitivity, facilitating the use of second-order algorithms to solve intricate optimization problems.

As it contributes to the development of numerical tools in structural mechanics, the proposed method can play an important role in technological innovations. Additionally, the tensor hyper-dual arithmetics can affect other scientific fields, as this tool simply aims to obtain the derivatives of a given function. Therefore, the outcomings of this work surpass applications in computational solid mechanics.

REFERENCES

AHMADVAND, Hosein; HABIBI, Alireza. A new second-order approximation method for optimum design of structures. **Australian Journal of Civil Engineering**, Taylor & Francis, v. 0, n. 0, p. 1–26, 2020. DOI: 10.1080/14488353.2020.1798039. eprint: <https://doi.org/10.1080/14488353.2020.1798039>. Available from: <https://doi.org/10.1080/14488353.2020.1798039>.

ANDRADE, Felipe Xavier Costa; WILDEMANN, Fernando; MUÑOZ-ROJAS, Pablo A. An Educational Software for the Layout Optimization of 3D Trusses. In: CILAMCE - Congresso Ibero Latino Americano sobre Métodos Computacionais em Engenharia. Recife, Brazil: [s.n.], 2004.

ANDREI, Neculai. A New Diagonal Quasi-Newton Updating Method With Scaled Forward Finite Differences Directional Derivative for Unconstrained Optimization. **Numerical Functional Analysis and Optimization**, Taylor & Francis, v. 40, n. 13, p. 1467–1488, 2019. DOI: 10.1080/01630563.2018.1552293. eprint: <https://doi.org/10.1080/01630563.2018.1552293>. Available from: <https://doi.org/10.1080/01630563.2018.1552293>.

ANDREI, Neculai. Diagonal Approximation of the Hessian by Finite Differences for Unconstrained Optimization. **Journal of Optimization Theory and Applications**, May 2020. ISSN 1573-2878. DOI: 10.1007/s10957-020-01676-z. Available from: <https://doi.org/10.1007/s10957-020-01676-z>.

ARORA, J. S.; WANG, Q. Review of formulations for structural and mechanical system optimization. **Structural and Multidisciplinary Optimization**, v. 30, n. 4, p. 251–272, Oct. 2005. ISSN 1615-1488. DOI: 10.1007/s00158-004-0509-6. Available from: <https://doi.org/10.1007/s00158-004-0509-6>.

ARORA, J.S. **Introduction to Optimum Design**. [S.l.]: Elsevier Science, 2016. ISBN 9780128009185. Available from: <https://books.google.com.br/books?id=h-9eBwAAQBAJ>.

BANKS, H.; BEKELE-MAXWELL, K.; BOCIU, L.; NOORMAN, Marcella; TILLMAN, K. The complex-step method for sensitivity analysis of non-smooth problems arising in biology. **Eurasian Journal of Mathematical and Computer Applications**, v. 3, p. 16–68, 2015.

BAYDIN, Atilim Gunes; PEARLMUTTER, Barak A. **Automatic Differentiation of Algorithms for Machine Learning**. [S.l.: s.n.], 2014. arXiv: 1404.7456 [cs.LG].

BAYDIN, Atilim Gunes; PEARLMUTTER, Barak A.; RADUL, Alexey Andreyevich; SISKIND, Jeffrey Mark. Automatic Differentiation in Machine Learning: a Survey. **Journal of Machine Learning Research**, v. 18, n. 153, p. 1–43, 2018. Available from: <http://jmlr.org/papers/v18/17-468.html>.

BAYDIN, Atılım Günes; PEARLMUTTER, Barak A.; RADUL, Alexey Andreyevich; SISKIND, Jeffrey Mark. Automatic Differentiation in Machine Learning: A Survey. **J. Mach. Learn. Res.**, JMLR.org, v. 18, n. 1, p. 5595–5637, Jan. 2017. ISSN 1532-4435.

BEHR, Nicolas; DATTOLI, Giuseppe; LATTANZI, Ambra; LICCIARDI, Silvia. Dual Numbers and Operational Umbral Methods. **Axioms**, Multidisciplinary Digital Publishing Institute, v. 8, n. 3, p. 77, 2019.

BENDSØE, Martin P.; SIGMUND, Ole. **Topology Optimization - Theory, Methods, and Applications**. Germany: Springer Verlag, 2003. ISBN 3-540-42992-1.

BISCHOF, C. H.; BÜCKER, H. M.; LANG, B. Automatic Differentiation for Computational Finance. In: KONTOGHIORGHES, E. J.; RUSTEM, B.; SIOKOS, S. (Eds.). **Computational Methods in Decision-Making, Economics and Finance**. Dordrecht: Kluwer Academic Publishers, 2002. v. 74. (Applied Optimization). chap. 15, p. 297–310. DOI: 10.1007/978-1-4757-3613-7_15.

BLETZINGER, Kai-Uwe; FIRL, Matthias; DAOUD, Fernass. Approximation of derivatives in semi-analytical structural optimization. **Computers & Structures**, v. 86, n. 13, p. 1404–1416, 2008. Structural Optimization. ISSN 0045-7949. DOI: <https://doi.org/10.1016/j.compstruc.2007.04.014>. Available from: <http://www.sciencedirect.com/science/article/pii/S0045794907001721>.

BONTE, Martijn H. A.; FOURMENT, Lionel; DO, Tientho; BOOGAARD, A. H. van den; HUÉTINK, J. Optimization of forging processes using Finite Element simulations. **Structural and Multidisciplinary Optimization**, v. 42, n. 5, p. 797–810, 2010. ISSN 1615-1488. DOI: 10.1007/s00158-010-0545-3. Available from: <https://doi.org/10.1007/s00158-010-0545-3>.

BRAKE, M.R.W.; FIKE, J.A.; TOPPING, S.D. Parameterized reduced order models from a single mesh using hyper-dual numbers. **Journal of Sound and Vibration**, v. 371,

- p. 370–392, 2016. ISSN 0022-460X. DOI:
<https://doi.org/10.1016/j.jsv.2016.02.026>. Available from:
<http://www.sciencedirect.com/science/article/pii/S0022460X16001693>.
- BURDEN, Richard L.; FAIRES, J. Douglas. **Numerical Analysis**. [S.l.]: Brooks Cole, 2010. ISBN 0538733519.
- BYRD, Richard H.; CHIN, Gillian M.; NEVEITT, Will; NOCEDAL, Jorge. On the Use of Stochastic Hessian Information in Optimization Methods for Machine Learning. **SIAM Journal on Optimization**, v. 21, n. 3, p. 977–995, 2011. DOI: 10.1137/10079923X. eprint: <https://doi.org/10.1137/10079923X>. Available from:
<https://doi.org/10.1137/10079923X>.
- C. H. BISCHOF; H. M. BÜCKER. Computing Derivatives of Computer Programs. In: GROTENDORST, J. (Ed.). **Modern Methods and Algorithms of Quantum Chemistry: Proceedings, Second Edition**. Jülich: NIC-Directors, 2000. v. 3. (NIC Series). P. 315–327. Available from:
https://juser.fz-juelich.de/record/44658/files/Band_3_Winterschule.pdf.
- CHANG, Kuang-Hua. Chapter 4 - Structural Design Sensitivity Analysis. In: CHANG, Kuang-Hua (Ed.). **Design Theory and Methods Using CAD/CAE**. Boston: Academic Press, 2015. P. 211–323. ISBN 978-0-12-398512-5. DOI:
<https://doi.org/10.1016/B978-0-12-398512-5.00004-9>. Available from:
<http://www.sciencedirect.com/science/article/pii/B9780123985125000049>.
- CHAPRA, Steven C.; CANALE, Raymond. **Numerical Methods for Engineers**. 5. ed. New York, NY, USA: McGraw-Hill, Inc., 2006. ISBN 0073101567, 9780073101569.
- CHIVERS, Ian; SLEIGHTHOLME, Jane. **Introduction to programming with Fortran**. [S.l.]: Springer, 2011.
- CHO, Seonho; JUNG, Hyun-Seung. Design sensitivity analysis and topology optimization of displacement-loaded non-linear structures. **Computer Methods in Applied Mechanics and Engineering**, v. 192, n. 22, p. 2539–2553, 2003. ISSN 0045-7825. DOI:
[https://doi.org/10.1016/S0045-7825\(03\)00274-3](https://doi.org/10.1016/S0045-7825(03)00274-3). Available from:
<http://www.sciencedirect.com/science/article/pii/S0045782503002743>.
- CHOI, K.K.; KIM, N.H. **Structural Sensitivity Analysis and Optimization 1: Linear Systems**. [S.l.]: Springer New York, 2005a. (Mechanical Engineering Series). ISBN

978-0-387-27169-9. DOI: 10.1007/b138709. Available from:
<http://dx.doi.org/10.1007/b138709>.

CHOI, K.K.; KIM, N.H. **Structural Sensitivity Analysis and Optimization 2: Nonlinear Systems and Applications**. [S.l.]: Springer New York, 2005b. (Mechanical Engineering Series). ISBN 978-0-387-27306-8. DOI: 10.1007/b138895. Available from:
<http://dx.doi.org/10.1007/b138895>.

COHEN, Avraham; SHOHAM, Moshe. Application of Hyper-Dual Numbers to Multi-Body Kinematics. v. 8, 2015.

COHEN, Avraham; SHOHAM, Moshe. Application of hyper-dual numbers to rigid bodies equations of motion. **Mechanism and Machine Theory**, v. 111, p. 76–84, 2017. ISSN 0094-114X. DOI: <https://doi.org/10.1016/j.mechmachtheory.2017.01.013>. Available from:
<http://www.sciencedirect.com/science/article/pii/S0094114X16305390>.

COHEN, Avraham; SHOHAM, Moshe. Principle of transference – An extension to hyper-dual numbers. **Mechanism and Machine Theory**, v. 125, p. 101–110, 2018. ISSN 0094-114X. DOI: <https://doi.org/10.1016/j.mechmachtheory.2017.12.007>. Available from:
<http://www.sciencedirect.com/science/article/pii/S0094114X1731385X>.

DAYNES, Stephen; FEIH, Stefanie; LU, Wen Feng; WEI, Jun. Design concepts for generating optimised lattice structures aligned with strain trajectories. **Computer Methods in Applied Mechanics and Engineering**, v. 354, p. 689–705, 2019. ISSN 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2019.05.053>. Available from:
<http://www.sciencedirect.com/science/article/pii/S0045782519303329>.

DESHAK, Vasuki Gopal; GANAPATHI, K.N.; MUSIB, Bikash; SARKAR, Malay. Optimization of Forging Process Parameters for Wheel Hub Using Numerical Simulation. **Materials Today: Proceedings**, v. 4, n. 10, p. 11107–11110, 2017. Advanced Materials, Manufacturing, Management and Thermal Science (AMMMT 2016) September 23-24, 2016. ISSN 2214-7853. DOI: <https://doi.org/10.1016/j.matpr.2017.08.073>. Available from: <http://www.sciencedirect.com/science/article/pii/S2214785317317078>.

DHATT, G.; LEFRANÇOIS, E.; TOUZOT, G. **Finite Element Method**. [S.l.]: Wiley, 2012. (ISTE). ISBN 978-1-848-21368-5.

DHATT, G.; TOUZOT, G. **The Finite Element Method Displayed**. [S.l.]: Wiley, 1984. ISBN 0471901105.

DILGEN, Cetin B.; DILGEN, Sumer B.; FUHRMAN, David R.; SIGMUND, Ole; LAZAROV, Boyan S. Topology optimization of turbulent flows. **Computer Methods in Applied Mechanics and Engineering**, v. 331, p. 363–393, 2018. ISSN 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2017.11.029>. Available from: <http://www.sciencedirect.com/science/article/pii/S0045782517307478>.

DUAN, Shengyu; XI, Li; WEN, Weibin; FANG, Daining. Mechanical performance of topology-optimized 3D lattice materials manufactured via selective laser sintering. **Composite Structures**, v. 238, p. 111985, 2020. ISSN 0263-8223. DOI: <https://doi.org/10.1016/j.compstruct.2020.111985>. Available from: <http://www.sciencedirect.com/science/article/pii/S0263822319338450>.

DUNHAM, Benjamin Z. **High-Order Automatic Differentiation of Unmodified Linear Algebra Routines via Nilpotent Matrices**. 2017. PhD thesis – University of Colorado, Boulder.

DUYSINX, Pierre; NGUYEN, Van Hien; BRUYNEEL, Michaël; FLEURY, Claude. Estimating diagonal second order terms in structural approximations with quasi-Cauchy techniques. In: 4TH WORLD CONGRESS OF STRUCTURAL and MULTIDISCIPLINARY OPTIMIZATION. PROCEEDINGS of the 4th World Congress of Structural and Multidisciplinary Optimization WCSMO4. Dalian, China: [s.n.], June 2001. Available from: <http://hdl.handle.net/2268/26092>.

DUYSINX, Pierre; ZHANG, Weihong; FLEURY, Claude; NGUYEN, Van Hien; HAUBRUGE, Sylvianne. A New Separable Approximation Scheme for Topological Problems and Optimization Problems Characterized by a Large Number of Design Variables. Goslar, Germany, p. 1–8, May 1995. Available from: <http://hdl.handle.net/2268/28259>.

ENDO, Vitor T.; FANCELLO, Eduardo A.; MUÑOZ-ROJAS, Pablo A. A study on the computational effort of hyper-dual numbers to evaluate derivatives in geometrically nonlinear hyperelastic trusses. **Journal of the Brazilian Society of Mechanical Sciences and Engineering**, (to appear), 2020. (accepted for publication).

ENDO, Vitor T.; FANCELLO, Eduardo A.; MUÑOZ-ROJAS, Pablo Andrés. A first implementation of geometrically nonlinear hyperelastic truss elements using hyper-dual

numbers. In: MECSOL. 6TH International Symposium on Solid Mechanics (MECSOL 2017). Joinville, Brazil: ABCM, 2017.

ERLEBEN, Kenny; ANDREWS, Sheldon. Solving inverse kinematics using exact Hessian matrices. **Computers & Graphics**, v. 78, p. 1–11, 2019. ISSN 0097-8493. DOI: <https://doi.org/10.1016/j.cag.2018.10.012>. Available from: <http://www.sciencedirect.com/science/article/pii/S0097849318301730>.

ERWAY, Jennifer B.; GRIFFIN, Joshua; MARCIA, Roummel F.; OMHENI, Riadh. Trust-region algorithms for training responses: machine learning methods using indefinite Hessian approximations. **Optimization Methods and Software**, Taylor & Francis, v. 35, n. 3, p. 460–487, 2020. DOI: 10.1080/10556788.2019.1624747. eprint: <https://doi.org/10.1080/10556788.2019.1624747>. Available from: <https://doi.org/10.1080/10556788.2019.1624747>.

ETMAN, L. F. P.; GROENWOLD, Albert A.; ROODA, J. E. First-order sequential convex programming using approximate diagonal QP subproblems. **Structural and Multidisciplinary Optimization**, v. 45, n. 4, p. 479–488, Apr. 2012. ISSN 1615-1488. DOI: 10.1007/s00158-011-0739-3. Available from: <https://doi.org/10.1007/s00158-011-0739-3>.

FENG, Li-Jia; XIONG, Jian; YANG, Li-Hong; YU, Guo-Cai; YANG, Wen; WU, Lin-Zhi. Shear and bending performance of new type enhanced lattice truss structures. **International Journal of Mechanical Sciences**, v. 134, p. 589–598, 2017. ISSN 0020-7403. DOI: <https://doi.org/10.1016/j.ijmecsci.2017.10.045>. Available from: <http://www.sciencedirect.com/science/article/pii/S0020740317322385>.

FIKE, J. A.; ALONSO, J. J. The Development of Hyper-Dual Numbers for Exact Second-Derivative Calculations. In: 49TH AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition. [S.l.]: American Institute of Aeronautics and Astronautics (AIAA), Jan. 2011. DOI: 10.2514/6.2011-886. Available from: <http://dx.doi.org/10.2514/6.2011-886>.

FIKE, J. A.; JONGSMA, S.; ALONSO, J. J.; WEIDE, E. van der. Optimization with Gradient and Hessian Information Calculated using Hyper-Dual Numbers. In: AIAA paper 2011-3807, 29th AIAA Applied Aerodynamics Conference. [S.l.: s.n.], 2011.

FIKE, Jeffrey Alan. **Multi-objective optimization using hyper-dual numbers**. 2013. PhD thesis – Stanford university.

FLEURY, Claude. Efficient approximation concepts using second order information.

International Journal for Numerical Methods in Engineering, v. 28, n. 9,

p. 2041–2058, 1989. DOI: 10.1002/nme.1620280905. eprint:

<https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620280905>. Available

from: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620280905>.

FOHRMEISTER, Volker; BARTELS, Alexander; MOSLER, Jörn. Variational updates for thermomechanically coupled gradient-enhanced elastoplasticity — Implementation based on hyper-dual numbers.

Computer Methods in Applied Mechanics and Engineering,

v. 339, p. 239–261, 2018. ISSN 0045-7825. DOI:

<https://doi.org/10.1016/j.cma.2018.04.047>. Available from:

<http://www.sciencedirect.com/science/article/pii/S0045782518302329>.

FUJIKAWA, Masaki; ISHIKAWA, Kiyotaka; MAKABE, Chobin; TANAKA, Masato;

SASAGAWA, Takashi; OMOTE, Ryuji. Geometrically nonlinear finite element implementation based on highly accurate 1st and 2nd numerical derivative scheme using hyper-dual numbers.

Transactions of the JSME (in Japanese), v. 82, p. 15-00454–15-00454, 834 2016. ISSN

2187-9761. DOI: 10.1299/transjsme.15-00454. Available from:

<http://doi.org/10.1299/transjsme.15-00454>.

FUJIKAWA, Masaki; TANAKA, Masato; IMOTO, Yusuke; MITSUME, Naoto;

URAMOTO, Takeo; YAMANAKA, Naoya. Formulation for an Ogden-type hyperelastic analysis with hyper dual numbers and its performance evaluation.

Transactions of the JSME (in Japanese), v. 86, n. 881, p. 19-00256-19-00256, 2020. DOI:

10.1299/transjsme.19-00256.

GAY, David M. Semiautomatic Differentiation for Efficient Gradient Computations. In: BÜCKER, H. M.; CORLISS, G.; HOVLAND, P.; NAUMANN, U.; NORRIS, B. (Eds.).

Automatic Differentiation: Applications, Theory, and Implementations. [S.l.]:

Springer, 2005. (Lecture Notes in Computational Science and Engineering). P. 147–158. DOI:

10.1007/3-540-28438-9_13.

GEBREMEDHIN, Assefaw; TARAFDAR, Arijit; POTHEN, Alex; WALTHER, Andrea. Efficient Computation of Sparse Hessians Using Coloring and Automatic Differentiation.

INFORMS Journal on Computing, v. 21, p. 209–223, May 2009. DOI: 10.1287/ijoc.1080.0286.

GIBSON, Lorna J.; ASHBY, Michael F. **Cellular Solids: Structure and Properties**. 2. ed.

[S.l.]: Cambridge University Press, 1997. (Cambridge Solid State Science Series). DOI:

10.1017/CB09781139878326.

GIRALDO-LONDOÑO, Oliver; MIRABELLA, Lucia; DALLORO, Livio; PAULINO, Glaucio H. Multi-material thermomechanical topology optimization with applications to additive manufacturing: Design of main composite part and its support structure. **Computer Methods in Applied Mechanics and Engineering**, v. 363, p. 112812, 2020. ISSN 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2019.112812>. Available from: <http://www.sciencedirect.com/science/article/pii/S0045782519307042>.

GOUVEA, M.T.; ODLOAK, D. Dealing with inconsistent quadratic programs in a SQP based algorithm. **Brazilian Journal of Chemical Engineering**, v. 14, 1997. DOI: 10.1590/S0104-66321997000100006.

GRIEWANK, Andreas; WALTHER, Andrea. **Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation**. Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008. ISBN 0898716594, 9780898716597.

GROENWOLD, Albert; ETMAN, L.; SNYMAN, J.; ROODA, J. Incomplete series expansion for function approximation. **Structural and Multidisciplinary Optimization**, v. 34, p. 21–40, May 2007. DOI: 10.1007/s00158-006-0070-6.

GROENWOLD, Albert A.; ETMAN, L. F. P. A quadratic approximation for structural topology optimization. **International Journal for Numerical Methods in Engineering**, v. 82, n. 4, p. 505–524, 2010. DOI: 10.1002/nme.2774. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2774>. Available from: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2774>.

HAFTKA, R.T.; GÜRDAL, Z. **Elements of Structural Optimization**. [S.l.]: Springer Netherlands, 1991. (Solid Mechanics and Its Applications). ISBN 9780792315056.

HANSELMAN, D.C.; LITTLEFIELD, B.L. **Mastering Matlab 8**. [S.l.]: Pearson Education, 2011. ISBN 978-0133002249.

HAUG, Edward J.; CHOI, Kyung K.; KOMKOV, Vadim. **Design Sensitivity Analysis of Structural Systems**. [S.l.]: Elsevier, Academic Press, 1986. (Mathematics in Science and Engineering 177). ISBN 978-0123329202.

HAVEROTH, Geovane A.; MUÑOZ-ROJAS, Pablo A. Complex Variable Semianalytical Method for Sensitivity Evaluation in Nonlinear Path Dependent Problems: Applications to Periodic Truss Materials. In: **Computational Modeling, Optimization and Manufacturing Simulation of Advanced Engineering Materials**. Ed. by

Pablo Andrés Muñoz-Rojas. Cham: Springer International Publishing, 2016. P. 239–270. ISBN 978-3-319-04265-7. DOI: 10.1007/978-3-319-04265-7_9. Available from: http://dx.doi.org/10.1007/978-3-319-04265-7_9.

HAVEROTH, Geovane Augusto; STAHLSCHMIDT, Joãnesson; MUÑOZ-ROJAS, Pablo A. Application of the Complex Variable Semi-analytical Method for Improved Displacement Sensitivity Evaluation in Geometrically Nonlinear Truss Problems. en. **Latin American Journal of Solids and Structures**, scielo, v. 12, p. 980–1005, 2015. ISSN 1679-7825. DOI: <http://dx.doi.org/10.1590/1679-78251911>. Available from: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1679-78252015000500980&nrm=iso.

HOLZLEITNER, L.; MAHMOUD, K.G. Structural shape optimization using msc/nastran and sequential quadratic programming. **Computers & Structures**, v. 70, n. 5, p. 487–514, 1999. ISSN 0045-7949. DOI: [https://doi.org/10.1016/S0045-7949\(98\)00179-5](https://doi.org/10.1016/S0045-7949(98)00179-5). Available from: <http://www.sciencedirect.com/science/article/pii/S0045794998001795>.

HOROWITZ, Bernardo; AFONSO, Silvana M.B. Quadratic programming solver for structural optimisation using SQP algorithm. **Advances in Engineering Software**, v. 33, n. 7, p. 669–674, 2002. Engineering Computational Technology & Computational Structures Technology. ISSN 0965-9978. DOI: [https://doi.org/10.1016/S0965-9978\(02\)00066-2](https://doi.org/10.1016/S0965-9978(02)00066-2). Available from: <http://www.sciencedirect.com/science/article/pii/S0965997802000662>.

HUCK, Alexander; UTKE, Jean; BISCHOF, Christian. Source Transformation of C++ Codes for Compatibility with Operator Overloading. **Procedia Computer Science**, v. 80, p. 1485–1496, 2016. International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA. ISSN 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2016.05.470>. Available from: <http://www.sciencedirect.com/science/article/pii/S1877050916309553>.

JAWED, Azhar H.; MORRIS, A. J. Approximate Higher-Order Sensitivities In Structural Design. **Engineering Optimization**, Taylor & Francis, v. 7, n. 2, p. 121–142, 1984. DOI: 10.1080/03052158408960634. eprint: <https://doi.org/10.1080/03052158408960634>. Available from: <https://doi.org/10.1080/03052158408960634>.

JIN, Weiya; DENNIS, Brian; WANG, Bo. Improved sensitivity analysis using a complex variable semi-analytical method. **Structural and Multidisciplinary Optimization**, v. 41, p. 433–439, Apr. 2010. DOI: 10.1007/s00158-009-0427-8.

JU, S.H.; STONE, J.J.; ROWLANDS, R.E. A new symmetric contact element stiffness matrix for frictional contact problems. **Computers & Structures**, v. 54, n. 2, p. 289–301, 1995. ISSN 0045-7949. DOI: [https://doi.org/10.1016/0045-7949\(94\)E0176-3](https://doi.org/10.1016/0045-7949(94)E0176-3). Available from: <http://www.sciencedirect.com/science/article/pii/0045794994E01763>.

KEULEN, F. van; HAFTKA, R.T.; KIM, N.H. Review of options for structural design sensitivity analysis. Part 1: Linear systems. **Computer Methods in Applied Mechanics and Engineering**, v. 194, n. 30, p. 3213–3243, 2005. Structural and Design Optimization. ISSN 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2005.02.002>. Available from: <http://www.sciencedirect.com/science/article/pii/S0045782505000459>.

KIRAN, Ravi; KHANDELWAL, Kapil. Automatic implementation of finite strain anisotropic hyperelastic models using hyper-dual numbers. **Computational Mechanics**, v. 55, n. 1, p. 229–248, Jan. 2015. ISSN 1432-0924. DOI: 10.1007/s00466-014-1094-1. Available from: <https://doi.org/10.1007/s00466-014-1094-1>.

KIRAN, Ravi; KHANDELWAL, Kapil. Complex step derivative approximation for numerical evaluation of tangent moduli. **Computers & Structures**, v. 140, p. 1–13, 2014. ISSN 0045-7949. DOI: <https://doi.org/10.1016/j.compstruc.2014.04.009>. Available from: <http://www.sciencedirect.com/science/article/pii/S0045794914001059>.

KIRAN, Ravi; LI, Lei; KHANDELWAL, Kapil. Complex Perturbation Method for Sensitivity Analysis of Nonlinear Trusses. **Journal of Structural Engineering**, v. 143, n. 1, p. 04016154, 2017. DOI: 10.1061/(ASCE)ST.1943-541X.0001619. eprint: <http://ascelibrary.org/doi/pdf/10.1061/%28ASCE%29ST.1943-541X.0001619>. Available from: <http://ascelibrary.org/doi/abs/10.1061/%28ASCE%29ST.1943-541X.0001619>.

KLEIBER, Michael. **Parameter Sensitivity in Nonlinear Mechanics: Theory and Finite Element Computations**. [S.l.]: Wiley, 1997. ISBN 9780471968542,0471968544.

KONTAK, Max; RÖHRIG-ZÖLLNER, Melven; HOFMANN, Johannes; WEISS, Felix. Automatic Differentiation in Multibody Helicopter Simulation. In: KECSKEMÉTHY, Andrés; GEU FLORES, Francisco (Eds.). **Multibody Dynamics 2019**. Cham: Springer International Publishing, 2020. P. 534–542.

LANTOINE, Gregory; RUSSELL, Ryan P.; DARGENT, Thierry. Using Multicomplex Variables for Automatic Computation of High-Order Derivatives. **ACM Trans. Math. Softw.**, Association for Computing Machinery, New York, NY, USA, v. 38, n. 3, Apr. 2012. ISSN

0098-3500. DOI: 10.1145/2168773.2168774. Available from:
<https://doi.org/10.1145/2168773.2168774>.

LEAL, Allan M. M. et al. **Autodiff, a modern, fast and expressive C++ library for automatic differentiation**. [S.l.: s.n.], 2018. <https://autodiff.github.io>. Available from: <https://autodiff.github.io>.

LI, Lei; KHANDELWAL, Kapil. An adaptive quadratic approximation for structural and topology optimization. **Computers & Structures**, v. 151, p. 130–147, 2015. ISSN 0045-7949. DOI: <https://doi.org/10.1016/j.compstruc.2015.01.013>. Available from: <http://www.sciencedirect.com/science/article/pii/S004579491500022X>.

LIU, Qimao; PAAVOLA, Juha. Sensitivity and Hessian matrix analysis of structural reliability for uniformly modulated random seismic response. **Mechanics Research Communications**, v. 62, p. 155–161, 2014. ISSN 0093-6413. DOI: <https://doi.org/10.1016/j.mechrescom.2014.10.002>. Available from: <http://www.sciencedirect.com/science/article/pii/S0093641314001311>.

LO, S.H. On bandsolver using skyline storage. **Computers & Structures**, v. 44, n. 6, p. 1187–1196, 1992. ISSN 0045-7949. DOI: [https://doi.org/10.1016/0045-7949\(92\)90362-4](https://doi.org/10.1016/0045-7949(92)90362-4). Available from: <http://www.sciencedirect.com/science/article/pii/0045794992903624>.

MALVERN, L.E. **Introduction to the mechanics of a continuous medium**. [S.l.]: Prentice-Hall, 1969. (Prentice-Hall series in engineering of the physical sciences).

MANZYUK, Oleksandr; PEARLMUTTER, Barak A.; RADUL, Alexey Andreyevich; RUSH, David R.; SISKIND, Jeffrey Mark. Perturbation confusion in forward automatic differentiation of higher-order functions. **Journal of Functional Programming**, Cambridge University Press, v. 29, e12, 2019. DOI: 10.1017/S095679681900008X.

MARGOSSIAN, Charles C. A review of automatic differentiation and its efficient implementation. **WIREs Data Mining and Knowledge Discovery**, v. 9, n. 4, e1305, 2019. DOI: <https://doi.org/10.1002/widm.1305>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1305>. Available from: <https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1305>.

MARTINS, Joaquim R. R. A.; STURDZA, Peter; ALONSO, Juan J. The Complex-step Derivative Approximation. **ACM Trans. Math. Softw.**, ACM, New York, NY, USA, v. 29,

n. 3, p. 245–262, Sept. 2003. ISSN 0098-3500. DOI: 10.1145/838250.838251. Available from: <http://doi.acm.org/10.1145/838250.838251>.

MEHMOOD, Sheheryar; OCHS, Peter. Automatic Differentiation of Some First-Order Methods in Parametric Optimization. In: CHIAPPA, Silvia; CALANDRA, Roberto (Eds.). **Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics**. Online: PMLR, Aug. 2020. (Proceedings of Machine Learning Research), p. 1584–1594. Available from: <http://proceedings.mlr.press/v108/mehmood20a.html>.

MORALES, José Luis; NOCEDAL, Jorge; WU, Yuchen. A sequential quadratic programming algorithm with an additional equality constrained phase. **IMA Journal of Numerical Analysis**, v. 32, n. 2, p. 553–579, Aug. 2011. ISSN 0272-4979. DOI: 10.1093/imanum/drq037. eprint: <https://academic.oup.com/imajna/article-pdf/32/2/553/2257336/drq037.pdf>. Available from: <https://doi.org/10.1093/imanum/drq037>.

MUÑOZ-ROJAS, Pablo A. A “continuum-like” nonlinear truss finite element description with examples in coupled damage-plasticity, viscoelasticity and hyperelasticity. To be published. [S.I.].

MUÑOZ-ROJAS, Pablo A. An introduction to nonlinear finite element analysis and linear optimization of trusses. unpublished book. [S.I.], to be published.

MUÑOZ-ROJAS, Pablo A.; CARNIEL, Thiago A.; SILVA, Emilio C. N.; ÖCHSNER, Andreas. Optimization of a Unit Periodic Cell in Lattice Block Materials Aimed at Thermo-Mechanical Applications. In: **Heat Transfer in Multi-Phase Materials**. Ed. by Andreas Öchsner and Graeme E. Murch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. P. 301–345. ISBN 978-3-642-04403-8. DOI: 10.1007/8611_2010_32. Available from: https://doi.org/10.1007/8611_2010_32.

NAUMANN, Uwe. **The Art of Differentiating Computer Programs**. [S.I.]: Society for Industrial and Applied Mathematics, 2011. DOI: 10.1137/1.9781611972078. eprint: <https://epubs.siam.org/doi/pdf/10.1137/1.9781611972078>. Available from: <https://epubs.siam.org/doi/abs/10.1137/1.9781611972078>.

NEIDINGER, Richard D. Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming. **SIAM Review**, v. 52, n. 3, p. 545–563, 2010. DOI:

10.1137/080743627. eprint: <https://doi.org/10.1137/080743627>. Available from: <https://doi.org/10.1137/080743627>.

NEUENHOFEN, Martin. Review of theory and implementation of hyper-dual numbers for first and second order automatic differentiation. **CoRR**, abs/1801.03614, 2018. arXiv: 1801.03614. Available from: <http://arxiv.org/abs/1801.03614>.

NØRGAARD, Sebastian A.; SAGEBAUM, Max; GAUGER, Nicolas R.; LAZAROV, Boyan S. Applications of automatic differentiation in topology optimization. **Structural and Multidisciplinary Optimization**, v. 56, n. 5, p. 1135–1146, Nov. 2017. ISSN 1615-1488. DOI: 10.1007/s00158-017-1708-2. Available from: <https://doi.org/10.1007/s00158-017-1708-2>.

OGDEN, R. W.; SACCOMANDI, G.; SGURA, I. Fitting hyperelastic models to experimental data. **Computational Mechanics**, v. 34, n. 6, p. 484–502, 2004. ISSN 1432-0924. DOI: 10.1007/s00466-004-0593-y. Available from: <http://dx.doi.org/10.1007/s00466-004-0593-y>.

OSANOV, Mikhail; GUEST, James K. Topology Optimization for Architected Materials Design. **Annual Review of Materials Research**, v. 46, n. 1, p. 211–233, 2016. DOI: 10.1146/annurev-matsci-070115-031826. eprint: <https://doi.org/10.1146/annurev-matsci-070115-031826>. Available from: <https://doi.org/10.1146/annurev-matsci-070115-031826>.

OZTURK, Murat; KOCAOGLAN, Sinem; SONMEZ, Fazil O. Concurrent design and process optimization of forging. **Computers & Structures**, v. 167, p. 24–36, 2016. ISSN 0045-7949. DOI: <https://doi.org/10.1016/j.compstruc.2016.01.016>. Available from: <http://www.sciencedirect.com/science/article/pii/S004579491630013X>.

PEÑUÑURI, F.; PEÓN, R.; GONZÁLEZ-SÁNCHEZ, D.; ESCALANTE SOBERANIS, M. A. Dual Numbers and Automatic Differentiation to Efficiently Compute Velocities and Accelerations. **Acta Applicandae Mathematicae**, v. 170, n. 1, p. 649–659, Dec. 2020. ISSN 1572-9036. DOI: 10.1007/s10440-020-00351-9. Available from: <https://doi.org/10.1007/s10440-020-00351-9>.

PETRA, C. G.; QIANG, F.; LUBIN, M.; HUCHETTE, J. On efficient Hessian computation using the edge pushing algorithm in Julia. **Optimization Methods and Software**, Taylor & Francis, v. 33, n. 4-6, p. 1010–1029, 2018. DOI: 10.1080/10556788.2018.1480625.

eprint: <https://doi.org/10.1080/10556788.2018.1480625>. Available from:
<https://doi.org/10.1080/10556788.2018.1480625>.

PRESS, W.H. **Numerical recipes in FORTRAN: the art of scientific computing**. [S.l.]: Cambridge Univ Pr, 1992. v. 1.

ROJAS-LABANDA, Susana; STOLPE, Mathias. An efficient second-order SQP method for structural topology optimization. **Structural and Multidisciplinary Optimization**, v. 53, n. 6, p. 1315–1333, June 2016. ISSN 1615-1488. DOI: 10.1007/s00158-015-1381-2. Available from: <https://doi.org/10.1007/s00158-015-1381-2>.

RONG, Jian Hua; TANG, Zhi Li; XIE, Yi Min; LI, Fang Yi. Topological optimization design of structures under random excitations using SQP method. **Engineering Structures**, v. 56, p. 2098–2106, 2013. ISSN 0141-0296. DOI:
<https://doi.org/10.1016/j.engstruct.2013.08.012>. Available from:
<http://www.sciencedirect.com/science/article/pii/S0141029613003799>.

RUFFWIND, Phil. **Reverse-mode automatic differentiation: a tutorial**. 2016. Available from:
<https://rufflewind.com/2016-12-30/reverse-mode-automatic-differentiation>.
Visited on: 9 Nov. 2020.

SCHITTKOWSKI, Klaus. NLPQLP: A Fortran Implementation of a Sequential Quadratic Programming Algorithm with Distributed and Non-Monotone Line Search - User's Guide, Version 4.2. Germany, July 2015. Available from: <http://www.easy-fit.de/NLPQLP.pdf>.

SILVANO, Yasmim Niehues. **Formulação de um elemento de viga simplificado hiperelástico geometricamente não linear**. 2015. S. 51. Monography (Undergraduate thesis) – State University of Santa Catarina, Joinville.

STUMPF, Felipe Tempel; MARCZAK, Rogério José. Characterization of Constitutive Parameters for Hyperelastic Models Considering the Baker-Ericksen Inequalities. In: **Computational Modeling, Optimization and Manufacturing Simulation of Advanced Engineering Materials**. Ed. by Pablo Andrés Muñoz-Rojas. Cham: Springer International Publishing, 2016. P. 375–393. ISBN 978-3-319-04265-7. DOI: 10.1007/978-3-319-04265-7_14. Available from:
http://dx.doi.org/10.1007/978-3-319-04265-7_14.

STUPKIEWICZ, Stanisław; KORELC, Jože; DUTKO, Martin; RODIČ, Tomaž. Shape sensitivity analysis of large deformation frictional contact problems. **Computer Methods in Applied Mechanics and Engineering**, v. 191, n. 33, p. 3555–3581, 2002. ISSN 0045-7825. DOI: [https://doi.org/10.1016/S0045-7825\(02\)00295-5](https://doi.org/10.1016/S0045-7825(02)00295-5). Available from: <http://www.sciencedirect.com/science/article/pii/S0045782502002955>.

TANAKA, Masato; BALZANI, Daniel; SCHRÖDER, Jörg. Implementation of incremental variational formulations based on the numerical calculation of derivatives using hyper dual numbers. **Computer Methods in Applied Mechanics and Engineering**, v. 301, p. 216–241, 2016. ISSN 0045-7825. DOI: <http://dx.doi.org/10.1016/j.cma.2015.12.010>. Available from: <http://www.sciencedirect.com/science/article/pii/S0045782515004156>.

TANAKA, Masato; SASAGAWA, Takashi; OMOTE, Ryuji; FUJIKAWA, Masaki; BALZANI, Daniel; SCHRÖDER, Jörg. A highly accurate 1st- and 2nd-order differentiation scheme for hyperelastic material models based on hyper-dual numbers. **Computer Methods in Applied Mechanics and Engineering**, v. 283, p. 22–45, 2015. ISSN 0045-7825. DOI: <http://dx.doi.org/10.1016/j.cma.2014.08.020>. Available from: <http://www.sciencedirect.com/science/article/pii/S0045782514002904>.

TORTORELLI, D. A.; MICHALERIS, P. Design sensitivity analysis: Overview and review. **Inverse Problems in Engineering**, Taylor & Francis, v. 1, n. 1, p. 71–105, 1994. DOI: 10.1080/174159794088027573. eprint: <https://doi.org/10.1080/174159794088027573>. Available from: <https://doi.org/10.1080/174159794088027573>.

TRENTIN, Robson; MUÑOZ-ROJAS, Pablo; VAZ JR, M. Parameter identification based on deep drawing experiments. In: COMPLAS X. INTERNATIONAL Conference on Computational Plasticity - Fundamentals and Applications. Barcelona, Spain: [s.n.], Sept. 2009.

UBESSI, C.J.B.; MARCZAK, R. J. Sensitivity analysis of 3D frictional contact with BEM using complex-step differentiation. en. **Latin American Journal of Solids and Structures**, scielo, v. 15, 2018. ISSN 1679-7825. Available from: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1679-78252018001000703&nrm=iso.

VAISSIER, Benjamin; PERNOT, Jean-Philippe; CHOUGRANI, Laurent; VÉRON, Philippe. Parametric design of graded truss lattice structures for enhanced thermal dissipation.

Computer-Aided Design, v. 115, p. 1–12, 2019. ISSN 0010-4485. DOI:
<https://doi.org/10.1016/j.cad.2019.05.022>. Available from:
<http://www.sciencedirect.com/science/article/pii/S0010448519301964>.

VIE, Jean-Leopold. **Second-order derivatives for shape optimization with a level-set method**. Dec. 2016. Theses – Université Paris-Est, France. Available from:
<https://hal.archives-ouvertes.fr/tel-01488770>.

VIGLIOTTI, Andrea; AURICCHIO, Ferdinando. Automatic Differentiation for Solid Mechanics. **Archives of Computational Methods in Engineering**, Jan. 2020. DOI:
10.1007/s11831-019-09396-y.

WANG, Wenjia; CLAUSEN, Peter M.; BLETZINGER, Kai-Uwe. Improved semi-analytical sensitivity analysis using a secant stiffness matrix for geometric nonlinear shape optimization. **Computers & Structures**, v. 146, p. 143–151, 2015. ISSN 0045-7949. DOI:
<https://doi.org/10.1016/j.compstruc.2014.08.008>. Available from:
<http://www.sciencedirect.com/science/article/pii/S0045794914001904>.

WANG, Yingjun; WANG, Zhenpei; XIA, Zhaohui; POH, Leong Hien. Structural Design Optimization Using Isogeometric Analysis: A Comprehensive Review. **Computer Modeling in Engineering and Sciences**, v. 117, p. 455–507, Dec. 2018. DOI:
10.31614/cmescs.2018.04603.

WANG, Yuxiang; GERLING, Gregory J. Automatic finite element implementation of hyperelastic material with a double numerical differentiation algorithm. **CoRR**, abs/1606.04987, 2016. arXiv: 1606.04987. Available from:
<http://arxiv.org/abs/1606.04987>.

WILKE, D. N.; KOK, S. Numerical sensitivity computation for discontinuous gradient-only optimization problems using the complex-step method. **Blucher Mechanical Engineering Proceedings**, v. 1, n. 1, p. 3665–3676, 2014. ISSN 2358-0828. DOI:
<http://dx.doi.org/10.5151/meceng-wccm2012-19508>. Available from:
www.proceedings.blucher.com.br/article-details/numerical-sensitivity-computation-for-discontinuous-gradient-only-optimization-problems-using-the-complex-step-method-9263.

XIA, Renwei; LIU, Peng. Structural optimization based on second-order approximations of functions and dual theory. **Computer Methods in Applied Mechanics and Engineering**, v. 65, n. 2, p. 101–114, 1987. ISSN 0045-7825. DOI:

[https://doi.org/10.1016/0045-7825\(87\)90007-7](https://doi.org/10.1016/0045-7825(87)90007-7). Available from:
<http://www.sciencedirect.com/science/article/pii/0045782587900077>.

XIAO, N.-C.; HUANG, H.-Z.; WANG, Z.; PANG, Y.; HE, L. Reliability sensitivity analysis for structural systems in interval probability form. **Structural and Multidisciplinary Optimization**, v. 44, n. 5, p. 691–705, 2011. cited By 48. DOI: 10.1007/s00158-011-0652-9. Available from:
<https://www.scopus.com/inward/record.uri?eid=2-s2.0-84855777942&doi=10.1007%2fs00158-011-0652-9&partnerID=40&md5=3410554fcbfed1291f2ddf7c46ee77d6>.

YU, Wenbin; BLAIR, Maxwell. DNAD, a simple tool for automatic differentiation of Fortran codes using dual numbers. **Computer Physics Communications**, v. 184, n. 5, p. 1446–1452, 2013. ISSN 0010-4655. DOI:
<https://doi.org/10.1016/j.cpc.2012.12.025>. Available from:
<http://www.sciencedirect.com/science/article/pii/S0010465513000027>.

ZHANG, W.H; DUYSINX, P. Dual approach using a variant perimeter constraint and efficient sub-iteration scheme for topology optimization. **Computers & Structures**, v. 81, n. 22, p. 2173–2181, 2003. ISSN 0045-7949. DOI:
[https://doi.org/10.1016/S0045-7949\(03\)00294-3](https://doi.org/10.1016/S0045-7949(03)00294-3). Available from:
<http://www.sciencedirect.com/science/article/pii/S0045794903002943>.

APPENDIX A – HYPER-DUAL NUMBERS

In this appendix, we include two important contents concerning the use of hyper-dual numbers as a tool to calculate the derivatives of a function. In Section A.1, we present a detailed description of the hyper-dual conversion of scalar-valued functions. In Section A.2, we show the computer implementation of hyper-dual numbers in Fortran using the operator overloading technique.

A.1 HYPER-DUAL CONVERSION

In this section, we present a detailed description involving the conversion of a function into the hyper-dual format. In Section A.1.1, we describe a scalar function of a scalar argument, while Section A.1.2 covers a function of a vector argument.

A.1.1 Scalar function of a scalar argument

In this section, we illustrate the hyper-dual conversion of a strain energy function, focusing on the calculation of stress and tangent modulus. In this example, we consider the hyperelastic model indicated by Eq. 91 with the bound of summation $M = 1$ and the arbitrary perturbation values h_1 and h_2 .

In this case, the hyper-dual conversion involves the application of 7 sequential operations, as described by

$$W(\lambda_1) = \frac{\gamma_1}{\beta_1} \underbrace{\left(\underbrace{\lambda_1^{\beta_1}}_{t_1} + 2 \underbrace{\lambda_1^{\frac{-\beta_1}{2}}}_{t_2} - 3 \right)}_{t_6}. \quad (135)$$

Step 1. Stretch λ_1 as a hyper-dual variable: $t_0 = \lambda_1$.

$$t_0 = \lambda_1 + h_1 \epsilon_1 + h_2 \epsilon_2 + 0 \epsilon_1 \epsilon_2.$$

Step 2. Power function: $t_1 = t_0^{(\beta_1)}$.

$$t_1 = \left[\lambda_1^{(\beta_1)} \right] + h_1 \left[\beta_1 \lambda_1^{(\beta_1-1)} \right] \epsilon_1 + h_2 \left[\beta_1 \lambda_1^{(\beta_1-1)} \right] \epsilon_2 + h_1 h_2 \left[\beta_1 (\beta_1 - 1) \lambda_1^{(\beta_1-2)} \right] \epsilon_1 \epsilon_2.$$

Step 3. Power function: $t_2 = t_0^{\left(\frac{-\beta_1}{2}\right)}$.

$$t_2 = \left[\lambda_1^{\left(\frac{-\beta_1}{2}\right)} \right] + h_1 \left[\left(\frac{-\beta_1}{2}\right) \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right] \epsilon_1 + h_2 \left[\left(\frac{-\beta_1}{2}\right) \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right] \epsilon_2 + h_1 h_2 \left[\left(\frac{\beta_1}{2}\right) \left(\frac{\beta_1}{2} + 1\right) \lambda_1^{\left(\frac{-\beta_1}{2}-2\right)} \right] \epsilon_1 \epsilon_2.$$

Step 4. Multiplication: $\mathbf{t}_3 = \mathbf{t}_2(2)$.

$$\begin{aligned} \mathbf{t}_3 = & \left[2\lambda_1^{\left(\frac{-\beta_1}{2}\right)} \right] + h_1 \left[-\beta_1 \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right] \epsilon_1 + h_2 \left[-\beta_1 \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right] \epsilon_2 \\ & + h_1 h_2 \left[\beta_1 \left(\frac{\beta_1}{2} + 1 \right) \lambda_1^{\left(\frac{-\beta_1}{2}-2\right)} \right] \epsilon_1 \epsilon_2. \end{aligned}$$

Step 5. Subtraction: $\mathbf{t}_4 = \mathbf{t}_3 - 3$.

$$\begin{aligned} \mathbf{t}_4 = & \left[2\lambda_1^{\left(\frac{-\beta_1}{2}\right)} - 3 \right] + h_1 \left[-\beta_1 \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right] \epsilon_1 + h_2 \left[-\beta_1 \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right] \epsilon_2 \\ & + h_1 h_2 \left[\beta_1 \left(\frac{\beta_1}{2} + 1 \right) \lambda_1^{\left(\frac{-\beta_1}{2}-2\right)} \right] \epsilon_1 \epsilon_2. \end{aligned}$$

Step 6. Addition: $\mathbf{t}_5 = \mathbf{t}_1 + \mathbf{t}_4$.

$$\begin{aligned} \mathbf{t}_5 = & \left[\lambda_1^{(\beta_1)} + 2\lambda_1^{\left(\frac{-\beta_1}{2}\right)} - 3 \right] + h_1 \left[\beta_1 \lambda_1^{(\beta_1-1)} - \beta_1 \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right] \epsilon_1 \\ & + h_2 \left[\beta_1 \lambda_1^{(\beta_1-1)} - \beta_1 \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right] \epsilon_2 \\ & + h_1 h_2 \left[\beta_1 (\beta_1 - 1) \lambda_1^{(\beta_1-2)} + \beta_1 \left(\frac{\beta_1}{2} + 1 \right) \lambda_1^{\left(\frac{-\beta_1}{2}-2\right)} \right] \epsilon_1 \epsilon_2. \end{aligned}$$

Step 7. Multiplication: $\mathbf{t}_6 = \mathbf{t}_5 \left(\frac{\gamma_1}{\beta_1} \right)$.

$$\begin{aligned} \mathbf{t}_6 = & \left[\frac{\gamma_1}{\beta_1} \left(\lambda_1^{(\beta_1)} + 2\lambda_1^{\left(\frac{-\beta_1}{2}\right)} - 3 \right) \right] + h_1 \left[\gamma_1 \left(\lambda_1^{(\beta_1-1)} - \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right) \right] \epsilon_1 \\ & + h_2 \left[\gamma_1 \left(\lambda_1^{(\beta_1-1)} - \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right) \right] \epsilon_2 \\ & + h_1 h_2 \left[\gamma_1 \left((\beta_1 - 1) \lambda_1^{(\beta_1-2)} + \left(\frac{\beta_1}{2} + 1 \right) \lambda_1^{\left(\frac{-\beta_1}{2}-2\right)} \right) \right] \epsilon_1 \epsilon_2. \end{aligned}$$

Finally, as indicated in Eq. 95, the results regarding strain energy, stress, and tangent modulus are obtained using particular hyper-dual terms, as indicated by

$$\begin{aligned} W(\lambda_1) &= \Re[\mathbf{t}_6] = \frac{\gamma_1}{\beta_1} \left(\lambda_1^{(\beta_1)} + 2\lambda_1^{\left(\frac{-\beta_1}{2}\right)} - 3 \right), \\ \sigma^{RE}(\lambda_1) &= \frac{\Im_{\epsilon_1}[\mathbf{t}_6]}{h_1} = \frac{h_1'}{h_1} \left[\gamma_1 \left(\lambda_1^{(\beta_1-1)} - \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right) \right] \\ &= \frac{\Im_{\epsilon_2}[\mathbf{t}_6]}{h_2} = \frac{h_2'}{h_2} \left[\gamma_1 \left(\lambda_1^{(\beta_1-1)} - \lambda_1^{\left(\frac{-\beta_1}{2}-1\right)} \right) \right] \quad \text{and} \\ E_T^{RE}(\lambda_1) &= \frac{\Im_{\epsilon_1 \epsilon_2}[\mathbf{t}_6]}{h_1 h_2} = \frac{h_1 h_2'}{h_1 h_2} \gamma_1 \left[(\beta_1 - 1) \lambda_1^{(\beta_1-2)} + \left(\frac{\beta_1}{2} + 1 \right) \lambda_1^{\left(\frac{-\beta_1}{2}-2\right)} \right]. \end{aligned} \tag{136}$$

It should be noted that the obtained expressions are identical to the analytical description of this constitutive model. Moreover, we highlight that the results were not affected by the perturbation values.

A.1.2 Scalar function of a vector argument

In this section, we describe the hyper-dual conversion of a scalar function of a vector argument containing two variables. The non-real terms $\mathfrak{S}_{\epsilon_1}$ and $\mathfrak{S}_{\epsilon_2}$ represent the gradient vector (a first-order tensor), whereas $\mathfrak{S}_{\epsilon_1\epsilon_2}$ represents the Hessian matrix (a second-order tensor). In order to illustrate this numerical derivative procedure, each step of the hyper-dual conversion is exemplified using a scalar-valued function defined by

$$f(x, y) = 5x^2y^3. \quad (137)$$

Using the same function, we explain three approaches for the derivative calculation using the hyper-dual scheme. We describe a method to obtain the full Hessian, the mixed derivative and the Hessian diagonal in Sections A.1.2.1, A.1.2.2, and A.1.2.3, respectively.

A.1.2.1 Full Hessian calculation

Initially, the real variables x and y are defined as hyper-dual numbers x and y with unit perturbation, as indicated by

$$\begin{aligned} x &= x_0 + \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \epsilon_1\epsilon_2 \quad \text{and} \\ y &= y_0 + \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \epsilon_1\epsilon_2, \end{aligned} \quad (138)$$

where x_0 and y_0 establish the real part of the functions. It should be noted that the perturbation was applied in a way to establish a variable order.

Therefore, in order to obtain the hyper-dual representation of the function presented in Eq. 137, the power operator must be applied according to Eq. 22. The application of such operator for each variable x and y results respectively in

$$\begin{aligned} x^2 &= x_0^2 + \begin{Bmatrix} 2x_0 \\ 0 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 2x_0 \\ 0 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \epsilon_1\epsilon_2 \quad \text{and} \\ y^3 &= y_0^3 + \begin{Bmatrix} 0 \\ 3y_0^2 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 0 \\ 3y_0^2 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 \\ 0 & 6y_0 \end{bmatrix} \epsilon_1\epsilon_2. \end{aligned} \quad (139)$$

Using the product operation defined in Eq. 25, we obtain

$$x^2y^3 = x_0^2y_0^3 + \begin{Bmatrix} 2x_0y_0^3 \\ 3x_0^2y_0^2 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 2x_0y_0^3 \\ 3x_0^2y_0^2 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 2y_0^3 & 6x_0y_0^2 \\ 6x_0y_0^2 & 6x_0^2y_0 \end{bmatrix} \epsilon_1\epsilon_2. \quad (140)$$

Finally, using Eq. 40, we complete the hyper-dual conversion of the function indicated in Eq. 137, as given by

$$5x^2y^3 = 5x_0^2y_0^3 + \left\{ \begin{array}{c} 10x_0y_0^3 \\ 15x_0^2y_0^2 \end{array} \right\} \epsilon_1 + \left\{ \begin{array}{c} 10x_0y_0^3 \\ 15x_0^2y_0^2 \end{array} \right\} \epsilon_2 + \left[\begin{array}{cc} 10y_0^3 & 30x_0y_0^2 \\ 30x_0y_0^2 & 30x_0^2y_0 \end{array} \right] \epsilon_1\epsilon_2. \quad (141)$$

Once the function was properly converted, a mathematical operator extracts a particular term of the hyper-dual number. The operator \Re extracts the real term, as indicated by

$$f(x_0, y_0) = \Re[5x^2y^3] = 5x_0^2y_0^3. \quad (142)$$

The first-order derivative is obtained using either the operator \Im_{ϵ_1} or \Im_{ϵ_2} , as illustrated by the expression

$$f'(x_0, y_0) = \Im_{\epsilon_1}[5x^2y^3] = \Im_{\epsilon_2}[5x^2y^3] = \left\{ \begin{array}{c} 10x_0y_0^3 \\ 15x_0^2y_0^2 \end{array} \right\}. \quad (143)$$

Finally, we obtain the second-order derivative using the operator $\Im_{\epsilon_1\epsilon_2}$, as shown by

$$f''(x_0, y_0) = \Im_{\epsilon_1\epsilon_2}[5x^2y^3] = \left[\begin{array}{cc} 10y_0^3 & 30x_0y_0^2 \\ 30x_0y_0^2 & 30x_0^2y_0 \end{array} \right]. \quad (144)$$

Based on this result, we conclude that the hyper-dual scheme allows the evaluation of first and second-order derivatives since the non-real terms \Im_{ϵ_1} and $\Im_{\epsilon_1\epsilon_2}$ return equivalent results when compared to analytical derivatives.

A.1.2.2 Mixed second-order derivative terms

Occasionally, the principal diagonal terms of the Hessian might not be required in a problem. In this case, the hyper-dual operation to convert a function can be slightly modified, as described in this section. Thus, we extend the example indicated in the previous section, aiming to describe a procedure to obtain the mixed second-order derivative term.

Differently from the previous case, in which the full Hessian is obtained, the perturbation is not applied simultaneously on axes ϵ_1 and ϵ_2 . Assuming this scheme, each axis is related to a particular design variable: the variable x is assigned to ϵ_1 and y is assigned to ϵ_2 . Thus, as the initial step for the hyper-dual conversion, we introduce the design variables with proper unit perturbation as

$$x = x_0 + 1\epsilon_1 + 0\epsilon_2 + 0\epsilon_1\epsilon_2 \quad \text{and} \quad (145)$$

$$y = y_0 + 0\epsilon_1 + 1\epsilon_2 + 0\epsilon_1\epsilon_2. \quad (146)$$

By comparing the obtained expressions with Eq. 138, we can notice a distinctive approach to apply the perturbation value. Additionally, in this example, the mathematical notation is consistent with the fact that each term is a scalar value.

The remaining stages to convert the function indicated in Eq. 137 into a hyper-dual representation are kept following the same rules for the mathematical operations. Thus, the application of the power operator described by Eq. 22 results in

$$x^2 = x_0^2 + 2x_0\epsilon_1 + 0\epsilon_2 + 0\epsilon_1\epsilon_2 \quad \text{and} \quad (147)$$

$$y^3 = y_0^3 + 0\epsilon_1 + 3y_0^2\epsilon_2 + 0\epsilon_1\epsilon_2. \quad (148)$$

Using the product operator indicated by Eq. 25, we obtain

$$x^2y^3 = x_0^2y_0^3 + 2x_0y_0^3\epsilon_1 + 3x_0^2y_0^2\epsilon_2 + 6x_0y_0^2\epsilon_1\epsilon_2. \quad (149)$$

Finally, we obtain the final hyper-dual expression as

$$5x^2y^3 = 5x_0^2y_0^3 + \underbrace{10x_0y_0^3}_{\frac{\partial f}{\partial x}}\epsilon_1 + \underbrace{15x_0^2y_0^2}_{\frac{\partial f}{\partial y}}\epsilon_2 + \underbrace{30x_0y_0^2}_{\frac{\partial^2 f}{\partial x\partial y}}\epsilon_1\epsilon_2, \quad (150)$$

where each non-real term is related to a particular derivative, as indicated.

In this section, we described how the perturbation application can affect the hyper-dual conversion and the derivative results. Interestingly, it exemplifies the generality aspects of the hyper-dual arithmetics; using the same basic mathematical rules, we can define a strategy to obtain the desired result, depending on the needs. The non-real terms of Eq. 150 include the first and the mixed second-order derivative, without the diagonal information.

A.1.2.3 Diagonal terms of the Hessian

Alternatively, we present an approach to focus on the diagonal terms of the Hessian matrix, based on the hyper-dual arithmetics presented in Section 2.3.1. To exemplify this procedure, we show the hyper-dual conversion of the function indicated in Eq. 137.

The first step involves the presentation of the design variables with unit perturbation in the hyper-dual terms ϵ_1 and ϵ_2 , as given by

$$x = x_0 + \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \epsilon_1\epsilon_2 \quad \text{and} \quad (151)$$

$$y = y_0 + \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \epsilon_1\epsilon_2, \quad (152)$$

Then, we apply the power function according to Eq. 75, which results in

$$x^2 = x_0^2 + \begin{Bmatrix} 2x_0 \\ 0 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 2x_0 \\ 0 \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 2 \\ 0 \end{Bmatrix} \epsilon_1\epsilon_2 \quad \text{and} \quad (153)$$

$$y^3 = y_0^3 + \begin{Bmatrix} 0 \\ 3y_0^2 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 0 \\ 3y_0^2 \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 0 \\ 6y_0 \end{Bmatrix} \epsilon_1\epsilon_2. \quad (154)$$

The product of two diagonal hyper-dual numbers is defined by Eq. 76; the results are given by

$$x^2y^3 = x_0^2y_0^3 + \begin{Bmatrix} 2x_0y_0^3 \\ 3x_0^2y_0^2 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 2x_0y_0^3 \\ 3x_0^2y_0^2 \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 2y_0^3 \\ 6x_0^2y_0 \end{Bmatrix} \epsilon_1\epsilon_2 \quad \text{and} \quad (155)$$

$$5x^2y^3 = 5x_0^2y_0^3 + \begin{Bmatrix} 10x_0y_0^3 \\ 15x_0^2y_0^2 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 10x_0y_0^3 \\ 15x_0^2y_0^2 \end{Bmatrix} \epsilon_2 + \begin{Bmatrix} 10y_0^3 \\ 30x_0^2y_0 \end{Bmatrix} \epsilon_1\epsilon_2. \quad (156)$$

Finally, we extract the derivative information from the hyper-dual variable using the operator \mathfrak{S} . We obtain the second-order derivative according to

$$\mathfrak{S}_{\epsilon_1\epsilon_2} [5x^2y^3] = \begin{Bmatrix} 10y_0^3 \\ 30x_0^2y_0 \end{Bmatrix} = \begin{Bmatrix} \frac{\partial^2 f}{\partial x^2} \\ \frac{\partial^2 f}{\partial y^2} \end{Bmatrix}. \quad (157)$$

As noted, this approach neglects the calculation of the off-diagonal terms of the Hessian matrix $\frac{\partial^2 f}{\partial x \partial y}$ and $\frac{\partial^2 f}{\partial y \partial x}$. We highlight that this method maintains the gradient calculation, as observed in the hyper-dual terms ϵ_1 and ϵ_2 in Eq. 156. Thus, we conclude that it is possible to manipulate the hyper-dual scheme to deliver a fragment of the Hessian matrix describing the second-order derivatives.

A.2 COMPUTER IMPLEMENTATION

In this section, we present a preliminary implementation of tensor hyper-dual numbers and the diagonal variant in Fortran. We also bring examples to explain how to use them. For the sake of conciseness, we focus on the implementation of the arithmetic rules related to these examples. We tested this code using the compiler of GNU Fortran 9.3.0. It is important to highlight that we do not claim an efficient performance in this code, as we focused on readability and clear connections with the mathematical expressions.

A.2.1 Tensor hyper-dual numbers

The hyper-dual module, indicated in Figs. 13 and 14, comprises the definition and the arithmetics of the hyper-dual variables. We covered the theoretical aspects of these topics in Tab. 2 and throughout Section 2.2.1, respectively. In addition, to facilitate the understanding, we include the equation references in the code.

In order to illustrate the use of the hyper-dual module, we evaluate the same scalar-valued function discussed in Section A.1.2. We present the corresponding Fortran code in Fig. 15. Additionally, to complement the contents of Section 2.2.3, we show in Fig. 16 the code showing the conversion of the stiffness matrix into hyper-dual numbers.

In both cases, we introduced the independent variables as scalar hyper-dual variables with unit perturbation; afterward, the function is converted using a conventional syntax for the mathematical operators. As observed, the Fortran code recognizes the variable type of arguments and automatically applies proper arithmetics, as defined previously in the hyper-dual

module. Finally, we extract the function gradient and the Hessian from the obtained hyper-dual number as real variables.

A.2.2 An operator to extract the diagonal terms

In this section, we describe a mathematical operator that extracts the diagonal terms of a tensor; this item is defined as

$$\delta_{klb} = \begin{cases} 1, & \text{if } k = l = b \\ 0, & \text{otherwise} \end{cases}. \quad (158)$$

We use this operator to obtain the vector \mathbf{d} containing the diagonal terms of the square matrix \mathbf{A} , as shown in

$$d_b = A_{kl} \delta_{klb}. \quad (159)$$

By extension, assuming that this matrix is obtained using the tensor product of vectors \mathbf{u} and \mathbf{v} , we can also describe the diagonal representation using the expression

$$d_b = u_k v_l \delta_{klb}. \quad (160)$$

The tensor product of vectors occurs in many arithmetic operators of tensor hyper-dual numbers.

The conditional statement in Eq. 158 plays an important role to reduce the number of operations of a typical tensor product of two vectors¹. Using this specific approach with the operator δ_{klb} , instead of calculating all terms and then extracting the diagonal terms, we can focus the computational effort only on the required information².

In terms of computer implementation, we argue that the direct implementation of δ_{klb} would be uneconomical. As an alternative, we present a strategy that neglects this operator and returns equivalent results. We considered the expression indicated in Eq. 160 to illustrate this scheme. We present in Fig. 17 the Fortran code that exemplifies this strategy to obtain the diagonal information of a matrix. By using this approach, we can mitigate the number of modifications in the original hyper-dual module to obtain its diagonal variant.

¹ To illustrate, let us consider that we aim to find the vector \mathbf{d} containing the diagonal terms of the square matrix \mathbf{A} , which is obtained as a result of a tensor product; these items are presented by

$$\mathbf{A} = \mathbf{u} \otimes \mathbf{v} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 \end{bmatrix} \quad \text{and} \quad \mathbf{d} = \begin{Bmatrix} u_1 v_1 \\ u_2 v_2 \\ u_3 v_3 \end{Bmatrix}. \quad (161)$$

² We calculate the vector with the diagonal terms of the matrix using the operator δ_{klb} , as given by

$$\begin{aligned} d_1 &= u_1 v_1 \delta_{111} + u_1 v_2 \delta_{121} + u_1 v_3 \delta_{131} + u_2 v_1 \delta_{211} + u_2 v_2 \delta_{221} + u_2 v_3 \delta_{231} + u_3 v_1 \delta_{311} + u_3 v_2 \delta_{321} + u_3 v_3 \delta_{331} \\ d_2 &= u_1 v_1 \delta_{112} + u_1 v_2 \delta_{122} + u_1 v_3 \delta_{132} + u_2 v_1 \delta_{212} + u_2 v_2 \delta_{222} + u_2 v_3 \delta_{232} + u_3 v_1 \delta_{312} + u_3 v_2 \delta_{322} + u_3 v_3 \delta_{332} \\ d_3 &= u_1 v_1 \delta_{113} + u_1 v_2 \delta_{123} + u_1 v_3 \delta_{133} + u_2 v_1 \delta_{213} + u_2 v_2 \delta_{223} + u_2 v_3 \delta_{233} + u_3 v_1 \delta_{313} + u_3 v_2 \delta_{323} + u_3 v_3 \delta_{333} \end{aligned} \quad (162)$$

A.2.3 Diagonal hyper-dual numbers

We present the Fortran module containing the arithmetics of diagonal hyper-dual numbers in Figs. 18 and 19. This code refers to the computer implementation of the contents presented in Tab. 3 and Section 2.3.1, which comprise the definition of diagonal hyper-dual numbers and the associated mathematical operations. The Fortran module contains the definition of the operator, which is addressed to a syntax symbol.

In Section 2.3.2, we illustrated the conversion of the stiffness matrix into the diagonal form of hyper-dual numbers. The corresponding Fortran code is shown in Fig. 20, which exemplifies the use of the module containing the arithmetic operations. The derivative results are consistent with the analytical predictions, as indicated in Eqs. 89 and 90.

Figure 13 – Arithmetic rules of tensor hyper-dual numbers (part 1).

```

hyperdual_module.f95
1  module hyperdual_module
2  implicit none; integer i,j,m,i_max,j_max,m_max,k,l,k_max,l_max
3  ! definition of hyper-dual variables
4  type hyperdual_scalar ! (Eq. 10)
5  real(8) :: a_0
6  real(8), allocatable, dimension(:) :: a_1,a_2
7  real(8), allocatable, dimension(:, :) :: a_3
8  end type hyperdual_scalar
9  type hyperdual_vector ! (Eq. 11)
10 real(8), allocatable, dimension(:) :: a_0
11 real(8), allocatable, dimension(:, :) :: a_1,a_2
12 real(8), allocatable, dimension(:, :, :) :: a_3
13 end type hyperdual_vector
14 type hyperdual_tensor ! (Eq. 12)
15 real(8), allocatable, dimension(:, :, :) :: a_0
16 real(8), allocatable, dimension(:, :, :, :) :: a_1,a_2
17 real(8), allocatable, dimension(:, :, :, :, :) :: a_3
18 end type hyperdual_tensor
19 ! operator symbol and the arithmetic operation
20 interface operator(*); module procedure mul_scalar_scalar; end interface
21 interface operator(**); module procedure mul_scalar_real; end interface
22 interface operator(**); module procedure pow_scalar_real; end interface
23 interface operator(*); module procedure mul_scalar_vector; end interface
24 interface operator(*); module procedure mul_scalar_tensor; end interface
25 interface operator(.cro.); module procedure mul_vector_vector_cross; end interface
26 contains ! subroutines and functions describing the mathematical operators
27 function mul_scalar_scalar(x,y) ! (Eq. 25)
28 type(hyperdual_scalar), intent(in) :: x,y
29 type(hyperdual_scalar) :: mul_scalar_scalar
30 real(8), allocatable, dimension(:, :, :) :: temp_tensor2; allocate(temp_tensor2(k_max,l_max
31 ↪ ↪ ))
32 do k=1,k_max; do l=1,l_max
33 temp_tensor2(k,l) = ( x%a_0 * y%a_3(k,l) ) + ( x%a_1(k) * y%a_2(l) ) &
34 + ( y%a_1(k) * x%a_2(l) ) + ( y%a_0 * x%a_3(k,l) )
35 end do; end do
36 mul_scalar_scalar%a_0 = x%a_0 * y%a_0
37 mul_scalar_scalar%a_1 = ( x%a_0 * y%a_1 ) + ( y%a_0 * x%a_1 )
38 mul_scalar_scalar%a_2 = ( x%a_0 * y%a_2 ) + ( y%a_0 * x%a_2 )
39 mul_scalar_scalar%a_3 = temp_tensor2
40 end function mul_scalar_scalar
41 function mul_scalar_real(x,y) ! (Eq. 40)
42 type(hyperdual_scalar), intent(in) :: x
43 real(8), intent(in) :: y
44 type(hyperdual_scalar) :: mul_scalar_real
45 mul_scalar_real%a_0 = x%a_0 * y; mul_scalar_real%a_1 = x%a_1 * y
46 mul_scalar_real%a_2 = x%a_2 * y; mul_scalar_real%a_3 = x%a_3 * y
47 end function mul_scalar_real
48 function pow_scalar_real(x,n) ! (Eq. 22)
49 type(hyperdual_scalar), intent(in) :: x
50 real(8), intent(in) :: n
51 type(hyperdual_scalar) :: pow_scalar_real
52 real(8), allocatable, dimension(:, :, :) :: temp_tensor2; allocate(temp_tensor2(k_max,l_max
53 ↪ ↪ ))
54 do k=1,k_max; do l=1,l_max
55 temp_tensor2(k,l) = x%a_1(k) * x%a_2(l)
56 end do; end do
57 pow_scalar_real%a_0 = x%a_0 ** n
58 pow_scalar_real%a_1 = x%a_1 * ( n * ( x%a_0 ** (n-1) ) )
59 pow_scalar_real%a_2 = x%a_2 * ( n * ( x%a_0 ** (n-1) ) )
60 pow_scalar_real%a_3 = x%a_3 * ( n * ( x%a_0 ** (n-1) ) ) &
61 + ( temp_tensor2 * ( n * (n-1) * ( x%a_0 ** (n-2) ) ) )
62 end function pow_scalar_real

```

Source – Own author.

Figure 14 – Arithmetic rules of tensor hyper-dual numbers (part 2).

```

hyperdual_module.f95
61 function mul_scalar_vector(x,y) ! (Eq. 26)
62   type(hyperdual_scalar),intent(in) :: x
63   type(hyperdual_vector),intent(in) :: y
64   type(hyperdual_vector) :: mul_scalar_vector
65   real(8),allocatable,dimension(:,:) :: temp_tensor2
66   real(8),allocatable,dimension(:,,:) :: temp_tensor3
67   allocate(temp_tensor2(i_max,k_max)); allocate(temp_tensor3(i_max,k_max,l_max))
68   do i=1,i_max; do k=1,k_max; do l=1,l_max
69     temp_tensor2(i,k) = ( x%a_0 * y%a_1(i,k) ) + ( y%a_0(i) * x%a_1(k) )
70     temp_tensor3(i,k,l) = ( x%a_0 * y%a_3(i,k,l) ) + ( x%a_1(k) * y%a_2(i,l) ) &
71       + ( y%a_1(i,k) * x%a_2(l) ) + ( y%a_0(i) * x%a_3(k,l) )
72   end do; end do; end do
73   mul_scalar_vector%a_0 = ( x%a_0 * y%a_0 ); mul_scalar_vector%a_1 = temp_tensor2
74   mul_scalar_vector%a_2 = temp_tensor2; mul_scalar_vector%a_3 = temp_tensor3
75 end function mul_scalar_vector
76 function mul_scalar_tensor(x,y) ! (Eq. 29)
77   type(hyperdual_scalar),intent(in) :: x
78   type(hyperdual_tensor),intent(in) :: y
79   type(hyperdual_tensor) :: mul_scalar_tensor
80   real(8),allocatable,dimension(:,:) :: temp_tensor2
81   real(8),allocatable,dimension(:,,:,:) :: temp_tensor3
82   real(8),allocatable,dimension(:,,:,:,:) :: temp_tensor4
83   allocate(temp_tensor2(i_max,j_max)); allocate(temp_tensor3(i_max,j_max,k_max))
84   allocate(temp_tensor4(i_max,j_max,k_max,l_max))
85   do i=1,i_max; do j=1,j_max; do k=1,k_max; do l=1,l_max
86     temp_tensor2(i,j) = ( x%a_0 * y%a_0(i,j) )
87     temp_tensor3(i,j,k) = ( x%a_0 * y%a_1(i,j,k) ) + ( y%a_0(i,j) * x%a_1(k) )
88     temp_tensor4(i,j,k,l) = ( x%a_0 * y%a_3(i,j,k,l) ) + ( x%a_1(k) * y%a_2(i,j,l) ) &
89       + ( y%a_1(i,j,k) * x%a_2(l) ) + ( y%a_0(i,j) * x%a_3(k,l) )
90   end do; end do; end do; end do
91   mul_scalar_tensor%a_0 = temp_tensor2; mul_scalar_tensor%a_1 = temp_tensor3
92   mul_scalar_tensor%a_2 = temp_tensor3; mul_scalar_tensor%a_3 = temp_tensor4
93 end function mul_scalar_tensor
94 function mul_vector_vector_cross(x,y) ! (Eq. 28)
95   type(hyperdual_vector),intent(in) :: x,y
96   type(hyperdual_tensor) :: mul_vector_vector_cross
97   real(8),allocatable,dimension(:,:) :: temp_tensor2
98   real(8),allocatable,dimension(:,,:,:) :: temp_tensor3
99   real(8),allocatable,dimension(:,,:,:,:) :: temp_tensor4
100  allocate(temp_tensor2(i_max,j_max)); allocate(temp_tensor3(i_max,j_max,k_max))
101  allocate(temp_tensor4(i_max,j_max,k_max,l_max))
102  do i=1,i_max; do j=1,j_max; do k=1,k_max; do l=1,l_max
103    temp_tensor2(i,j) = x%a_0(i) * y%a_0(j)
104    temp_tensor3(i,j,k) = ( x%a_0(i) * y%a_1(j,k) ) + ( y%a_0(j) * x%a_1(i,k) )
105    temp_tensor4(i,j,k,l) = ( x%a_0(i) * y%a_3(j,k,l) ) + ( x%a_1(i,k) * y%a_1(j,l) ) &
106      + ( y%a_1(j,k) * x%a_2(i,l) ) + ( y%a_0(j) * x%a_3(i,k,l) )
107  end do; end do; end do; end do
108  mul_vector_vector_cross%a_0=temp_tensor2; mul_vector_vector_cross%a_1 = temp_tensor3
109  mul_vector_vector_cross%a_2=temp_tensor3; mul_vector_vector_cross%a_3 = temp_tensor4
110 end function mul_vector_vector_cross
111 subroutine write_hyperdual_tensor(hd) ! display hyper-dual variable
112   type(hyperdual_tensor),intent(in) :: hd ! tensor hyper-dual number
113   write(*,*) 'real term: '; do i=1,i_max; write(*,*) ( hd%a_0(i,j), j=1,j_max ); end do
114   write(*,*) 'eps1 term: '; do k=1,k_max; write(*,*) 'page_(k)', k; do i=1,i_max;
115     write(*,*) ( hd%a_1(i,j,k), j=1,j_max ); end do; end do
116   write(*,*) 'eps2 term: '; do k=1,k_max; do l=1,l_max; write(*,*) 'page_(k,l)', k,l
117     do i=1,i_max; write(*,*) ( hd%a_3(i,j,k,l), j=1,j_max ); end do; end do; end do
118 end subroutine write_hyperdual_tensor
119 end module hyperdual_module

```

Source – Own author.

Figure 15 – Minimal working example: evaluation of $f(x, y) = 5x^2y^3$ at $x_0 = 5$ and $y_0 = 2$.

```

1  program main
2    use hyperdual_module ! Load arithmetic rules
3    use problem_index ! Load index variables
4    implicit none ! declare all variables
5    ! scalar hyperdual variables
6    type(hyperdual_scalar) :: func,x,y
7    ! point (real scalar)
8    real                    :: x0,y0
9    ! gradient (real vector)
10   real(8),allocatable,dimension(:) :: grad,func_gradient
11   ! hessian (real matrix)
12   real(8),allocatable,dimension(:,:) :: hess,func_hessian
13   ! design variable information
14   k_max = 2; l_max = k_max ! number of design variables
15   ! gradient vector size
16   allocate( grad(k_max) ); allocate( func_gradient(k_max) )
17   ! hessian matrix size
18   allocate( hess(k_max,k_max) )
19   allocate( func_hessian(k_max,k_max) )
20   hess = 0 ! initial value of hessian is zero
21   ! x is the 1st hyper-dual variable
22   grad(1) = 1 ! apply unit perturbation
23   x0 = 5 ! real value of x
24   x = hyperdual_scalar(x0, grad, grad, hess)
25   ! y is the 2nd hyper-dual variable
26   grad = 0; grad(2) = 1 ! apply unit perturbation
27   y0 = 2 ! real value of y
28   y = hyperdual_scalar(y0, grad, grad, hess)
29   ! hyper-dual function
30   func = (x**2d0)*(y**3d0)*5d0
31   call write_hyperdual_scalar(func) ! print hyper-dual
32   ! result: function gradient
33   func_gradient = func%a_1 ! extract eps1 term
34   ! result: function hessian
35   func_hessian = func%a_3 ! extract eps1eps2 term
36 end program main

```

Source – Own author.

Figure 16 – Hyper-dual conversion of the stiffness matrix.

```

1  include 'hyperdual_module.f95'
2  program main
3  use hyperdual_module
4  implicit none
5  real(8) :: area0, emod0, length0
6  type(hyperdual_scalar) :: area, length
7  type(hyperdual_vector) :: temp_vec, b
8  type(hyperdual_tensor) :: k_truss
9  real(8), allocatable, dimension(:) :: grad, temp
10 real(8), allocatable, dimension(:, :) :: hess, k_func
11 real(8), allocatable, dimension(:, :, :) :: hess2, k_grad
12 real(8), allocatable, dimension(:, :, :, :) :: k_hess
13 ! define tensor size
14 i_max = 2; j_max = i_max ! number of degrees of freedom
15 k_max = 2; l_max = k_max ! number of design variables
16 allocate( grad(k_max), temp(k_max) )
17 allocate( hess(k_max, l_max), k_func(i_max, j_max) )
18 allocate( hess2(i_max, k_max, l_max), k_grad(i_max, j_max, k_max) )
19 allocate( k_hess(i_max, j_max, k_max, l_max) )
20 ! element parameters (input values)
21 area0 = 2; length0 = 3; emod0 = 2000
22 ! step 1. design variables as hyper-dual with unit perturbation
23 hess = 0; grad = 0; grad(1) = 1
24 area = hyperdual_scalar(area0, grad, grad, hess) ! (Eq. 55)
25 grad = 0; grad(2) = 1
26 length = hyperdual_scalar(length0, grad, grad, hess) ! (Eq. 56)
27 ! step 2. hyper-dual conversion
28 temp(1) = -1; temp(2) = 1; hess2 = 0
29 temp_vec = hyperdual_vector(temp, hess, hess, hess2) ! (Eq. 61)
30 b = ( length*(-1d0) ) * temp_vec ! (Eq. 62)
31 k_truss = area * length * emod0 * (b.cro.b) ! (Eq. 64)
32 call write_hyperdual_tensor(k_truss) ! display result
33 ! step 3. extract non-real parts
34 k_func = k_truss%a_0
35 k_grad = k_truss%a_1 ! (Eq. 65)
36 k_hess = k_truss%a_3 ! (Eq. 66)
37 end program

```

Source – Own author.

Figure 17 – Implementing the operator delta to obtain the diagonal terms.

```

1  program delta_operator
2  implicit none
3  integer :: b, k, l
4  real, dimension(3) :: d, u, v
5  real, dimension(3,3,3) :: delta
6  u = (/ 1.0, 2.0, 3.0 /)
7  v = (/ 4.0, 5.0, 6.0 /)
8  d = 0 ! input values
9  ! compute the diagonal terms (option 1)
10 delta = 0; delta(1,1,1) = 1
11 delta(2,2,2) = 1; delta(3,3,3) = 1
12 do b = 1, 3; do l = 1, 3; do k = 1, 3
13 d(b) = d(b) + u(k) * v(l) * delta(k,l,b)
14 end do; end do; end do
15 write(*,*) 'diagonal_1(option_1)', d
16 ! compute the diagonal terms (option 2)
17 do b = 1, 3
18 d(b) = u(b) * v(b)
19 end do
20 write(*,*) 'diagonal_2(option_2)', d
21 end program delta_operator

```

Source – Own author.

Figure 18 – Arithmetic rules of diagonal hyper-dual numbers (part 1).

```

                                dhyperdual_module.f95
1  module hyperdual_module
2  implicit none; integer i,j,m,i_max,j_max,m_max,k,l,k_max,l_max,b,b_max
3  ! definition of diagonal hyper-dual variables
4  type hyperdual_scalar ! (Eq. 67)
5      real(8) :: a_0
6      real(8), allocatable, dimension(:) :: a_1, a_2, a_3
7  end type hyperdual_scalar
8  type hyperdual_vector ! (Eq. 68)
9      real(8), allocatable, dimension(:) :: a_0
10     real(8), allocatable, dimension(:, :) :: a_1, a_2, a_3
11 end type hyperdual_vector
12 type hyperdual_tensor ! (Eq. 69)
13     real(8), allocatable, dimension(:, :,) :: a_0
14     real(8), allocatable, dimension(:, :, :,) :: a_1, a_2, a_3
15 end type hyperdual_tensor
16 ! operator symbol and the arithmetic operation
17 interface operator(*); module procedure mul_scalar_scalar; end interface
18 interface operator(*); module procedure mul_scalar_real; end interface
19 interface operator(**); module procedure pow_scalar_real; end interface
20 interface operator(*); module procedure mul_scalar_vector; end interface
21 interface operator(*); module procedure mul_scalar_tensor; end interface
22 interface operator(.cro.); module procedure mul_vector_vector_cross; end interface
23 contains ! subroutines and functions describing the mathematical operators
24 function mul_scalar_scalar(x,y)
25     type(hyperdual_scalar), intent(in) :: x,y
26     type(hyperdual_scalar) :: mul_scalar_scalar
27     real(8), allocatable, dimension(:) :: temp_tensor2; allocate(temp_tensor2(b_max))
28     do b=1,b_max
29         temp_tensor2(b) = ( x%a_0 * y%a_3(b) ) + ( x%a_1(b) * y%a_2(b) ) &
30             + ( y%a_1(b) * x%a_2(b) ) + ( y%a_0 * x%a_3(b) )
31     end do
32     mul_scalar_scalar%a_0 = x%a_0 * y%a_0
33     mul_scalar_scalar%a_1 = ( x%a_0 * y%a_1 ) + ( y%a_0 * x%a_1 )
34     mul_scalar_scalar%a_2 = ( x%a_0 * y%a_2 ) + ( y%a_0 * x%a_2 )
35     mul_scalar_scalar%a_3 = temp_tensor2
36 end function mul_scalar_scalar
37 function mul_scalar_real(x,y)
38     type(hyperdual_scalar), intent(in) :: x
39     real(8), intent(in) :: y
40     type(hyperdual_scalar) :: mul_scalar_real
41     mul_scalar_real%a_0 = x%a_0 * y; mul_scalar_real%a_1 = x%a_1 * y
42     mul_scalar_real%a_2 = x%a_2 * y; mul_scalar_real%a_3 = x%a_3 * y
43 end function mul_scalar_real
44 function pow_scalar_real(x,n)
45     type(hyperdual_scalar), intent(in) :: x
46     real(8), intent(in) :: n
47     type(hyperdual_scalar) :: pow_scalar_real
48     real(8), allocatable, dimension(:) :: temp_tensor2; allocate(temp_tensor2(b_max))
49     do b=1,b_max
50         temp_tensor2(b) = x%a_1(b) * x%a_2(b)
51     end do
52     pow_scalar_real%a_0 = x%a_0 ** n
53     pow_scalar_real%a_1 = x%a_1 * ( n * ( x%a_0 ** (n-1) ) )
54     pow_scalar_real%a_2 = x%a_2 * ( n * ( x%a_0 ** (n-1) ) )
55     pow_scalar_real%a_3 = x%a_3 * ( n * ( x%a_0 ** (n-1) ) ) &
56         + ( temp_tensor2 * ( n * (n-1) * ( x%a_0 ** (n-2) ) ) )
57 end function pow_scalar_real

```

Source – Own author.

Figure 19 – Arithmetic rules of diagonal hyper-dual numbers (part 2).

```

                                dhyperdual_module.f95
58 function mul_scalar_vector(x,y)
59   type(hyperdual_scalar),intent(in)      :: x
60   type(hyperdual_vector),intent(in)     :: y
61   type(hyperdual_vector)                :: mul_scalar_vector
62   real(8),allocatable,dimension(:,:)    :: temp_tensor2, temp_tensor3
63   allocate(temp_tensor2(i_max,k_max)); allocate(temp_tensor3(i_max,b_max))
64   do i=1,i_max; do k=1,k_max
65     temp_tensor2(i,k) = ( x%a_0 * y%a_1(i,k) ) + ( y%a_0(i) * x%a_1(k) )
66   end do; end do
67   do i=1,i_max; do b=1,b_max
68     temp_tensor3(i,b) = ( x%a_0 * y%a_3(i,b) ) + ( x%a_1(b) * y%a_2(i,b) ) &
69                       + ( y%a_1(i,b) * x%a_2(b) ) + ( y%a_0(i) * x%a_3(b) )
70   end do; end do
71   mul_scalar_vector%a_0 = ( x%a_0 * y%a_0 )
72   mul_scalar_vector%a_1 = temp_tensor2
73   mul_scalar_vector%a_2 = temp_tensor2
74   mul_scalar_vector%a_3 = temp_tensor3
75 end function mul_scalar_vector
76 function mul_scalar_tensor(x,y)
77   type(hyperdual_scalar),intent(in)      :: x
78   type(hyperdual_tensor),intent(in)     :: y
79   type(hyperdual_tensor)                :: mul_scalar_tensor
80   real(8),allocatable,dimension(:,:)    :: temp_tensor2
81   real(8),allocatable,dimension(:,,:)   :: temp_tensor3, temp_tensor4
82   allocate(temp_tensor2(i_max,j_max)); allocate(temp_tensor3(i_max,j_max,k_max))
83   allocate(temp_tensor4(i_max,j_max,b_max))
84   do i=1,i_max; do j=1,j_max; do k=1,k_max
85     temp_tensor2(i,j) = ( x%a_0 * y%a_0(i,j) )
86     temp_tensor3(i,j,k) = ( x%a_0 * y%a_1(i,j,k) ) + ( y%a_0(i,j) * x%a_1(k) )
87   end do; end do; end do
88   do i = 1, i_max; do j = 1, j_max; do b = 1, b_max
89     temp_tensor4(i,j,b) = ( x%a_0 * y%a_3(i,j,b) ) + ( x%a_1(b) * y%a_2(i,j,b) ) &
90                       + ( y%a_1(i,j,b) * x%a_2(b) ) + ( y%a_0(i,j) * x%a_3(b) )
91   end do; end do; end do
92   mul_scalar_tensor%a_0 = temp_tensor2;   mul_scalar_tensor%a_1 = temp_tensor3
93   mul_scalar_tensor%a_2 = temp_tensor3;   mul_scalar_tensor%a_3 = temp_tensor4
94 end function mul_scalar_tensor
95 function mul_vector_vector_cross(x,y)
96   type(hyperdual_vector),intent(in)     :: x,y
97   type(hyperdual_tensor)                :: mul_vector_vector_cross
98   real(8),allocatable,dimension(:,:)    :: temp_tensor2
99   real(8),allocatable,dimension(:,,:)   :: temp_tensor3, temp_tensor4
100  allocate(temp_tensor2(i_max,j_max)); allocate(temp_tensor3(i_max,j_max,k_max))
101  allocate(temp_tensor4(i_max,j_max,b_max))
102  do i=1,i_max; do j=1,j_max; do k=1,k_max
103    temp_tensor2(i,j) = x%a_0(i) * y%a_0(j)
104    temp_tensor3(i,j,k) = ( x%a_0(i) * y%a_1(j,k) ) + ( y%a_0(j) * x%a_1(i,k) )
105  end do; end do; end do
106  do i=1,i_max; do j=1,j_max; do b=1,b_max
107    temp_tensor4(i,j,b) = ( x%a_0(i) * y%a_3(j,b) ) + ( x%a_1(i,b) * y%a_1(j,b) ) &
108                      + ( y%a_1(j,b) * x%a_2(i,b) ) + ( y%a_0(j) * x%a_3(i,b) )
109  end do; end do; end do
110  mul_vector_vector_cross%a_0 = temp_tensor2
111  mul_vector_vector_cross%a_1 = temp_tensor3
112  mul_vector_vector_cross%a_2 = temp_tensor3
113  mul_vector_vector_cross%a_3 = temp_tensor4
114 end function mul_vector_vector_cross
115 end module hyperdual_module

```

Source – Own author.

Figure 20 – Diagonal hyper-dual conversion of the stiffness matrix.

```

1  include 'dhyperdual_module.f95'
2  program main
3  use hyperdual_module
4  implicit none
5  real(8) :: area0, emod0, length0
6  type(hyperdual_scalar) :: area, length
7  type(hyperdual_vector) :: temp_vec, b_mat
8  type(hyperdual_tensor) :: k_truss
9  real(8), allocatable, dimension(:) :: grad, temp, hess
10 real(8), allocatable, dimension(:, :) :: grad2, hess2, k_func
11 real(8), allocatable, dimension(:, :, :) :: k_grad, k_hess_diag
12 ! define tensor size
13 i_max = 2; j_max = i_max ! number of degrees of freedom
14 k_max = 2; l_max = k_max; b_max = k_max ! number of design variables
15 allocate( grad(k_max), temp(k_max), hess(b_max) )
16 allocate( k_func(i_max, j_max), grad2(i_max, k_max), hess2(i_max, k_max) )
17 allocate( k_grad(i_max, j_max, k_max), k_hess_diag(i_max, j_max, b_max) )
18 ! element parameters (input values)
19 area0 = 2; length0 = 3; emod0 = 2000
20 ! step 1. design variables as hyper-dual with unit perturbation
21 hess = 0; grad = 0; grad(1) = 1;
22 area = hyperdual_scalar(area0, grad, grad, hess) ! (Eq. 81)
23 grad = 0; grad(2) = 1;
24 length = hyperdual_scalar(length0, grad, grad, hess) ! (Eq. 82)
25 ! step 2. hyper-dual conversion
26 temp(1) = -1; temp(2) = 1; hess2 = 0
27 temp_vec = hyperdual_vector(temp, grad2, grad2, hess2)
28 b_mat = ( length*(-1d0) ) * temp_vec ! (Eq. 86)
29 k_truss = area * length * emod0 * (b_mat.cro.b_mat) ! (Eq. 88)
30 call write_hyperdual_tensor(k_truss) ! display result
31 ! step 3. extract non-real parts
32 k_func = k_truss%a_0
33 k_grad = k_truss%a_1 ! (Eq. 89)
34 k_hess_diag = k_truss%a_3 ! (Eq. 90)
35 end program

```

Source – Own author.

APPENDIX B – NONLINEAR TRUSS FINITE ELEMENT

The linear FEM basically considers that the global stiffness matrix is constant along the load application. This assumption is typically verified when the material behavior is linear elastic and only small displacements and strains take place. Since hyperelastic models are typically associated to finite strains, we considered a nonlinear finite element formulation for our case studies.

In this section, we review the main expressions of the formulation presented by Muñoz-Rojas (n.d.) and coworkers (HAVEROTH, Geovane Augusto et al., 2015; SILVANO, 2015) with a special focus on the application of numerical derivatives for the hyperelastic models. We describe the expressions regarding the internal force and the stiffness matrix in B.1 and B.2, respectively.

B.1 INTERNAL FORCE

The expressions for the internal force in the local coordinate system¹ are summarized in Tab. 11, wherein the stress values can be obtained using the hyper-dual procedure described beforehand. As noted, we highlighted these terms to emphasize the hyper-dual contribution in a FEM context.

Table 11 – Internal force expressions: \mathbf{q}_l .

Total Lagrangean	Updated Lagrangean
$\mathbf{q}_l^{RE} = \int_{-1}^{+1} \mathbf{B}^{*\top}(\mathbf{X}_l) \sigma^{RE} A_0 J(\mathbf{X}_l) d\xi$	$\mathbf{q}_l^{RE} = \int_{-1}^{+1} \lambda_1 \mathbf{B}^{*\top}(\mathbf{x}_l) \sigma^{RE} A J(\mathbf{x}_l) d\xi$
$\mathbf{q}_l^C = \int_{-1}^{+1} \frac{1}{\lambda_1} \mathbf{B}^{*\top}(\mathbf{X}_l) \sigma^C A_0 J(\mathbf{X}_l) d\xi$	$\mathbf{q}_l^C = \int_{-1}^{+1} \mathbf{B}^{*\top}(\mathbf{x}_l) \sigma^C A J(\mathbf{x}_l) d\xi$
$\mathbf{q}_l^{2PK} = \int_{-1}^{+1} \lambda_1 \mathbf{B}^{*\top}(\mathbf{X}_l) \sigma^{2PK} A_0 J(\mathbf{X}_l) d\xi$	$\mathbf{q}_l^{2PK} = \int_{-1}^{+1} \lambda_1^2 \mathbf{B}^{*\top}(\mathbf{x}_l) \sigma^{2PK} A J(\mathbf{x}_l) d\xi$

Source – Muñoz-Rojas (n.d.).

The modified strain-displacement matrix in the initial configuration $\mathbf{B}^*(\mathbf{X}_l)$ and the current configuration $\mathbf{B}^*(\mathbf{x}_l)$ are represented as row vectors and given by

$$\mathbf{B}^*(\mathbf{X}_l) = \frac{1}{2J(\mathbf{X}_l)} \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{B}^*(\mathbf{x}_l) = \frac{1}{2J(\mathbf{x}_l)} \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \quad (163)$$

Additionally, A_0 and A are the initial and the current cross-sectional areas, respectively. The Jacobians considering the initial and the current positions are

$$J(\mathbf{X}_l) = \frac{X_{2l} - X_{1l}}{2} = \frac{L_0}{2} \quad \text{and} \quad J(\mathbf{x}_l) = \frac{x_{2l} - x_{1l}}{2} = \frac{L}{2}, \quad (164)$$

¹ Although the formulation is defined incrementally, the expressions considering isoparametric formulation are presented with a simplified notation, i.e. the superscript related to the increment information ($k+1$) is omitted.

where L_0 and L are the initial and the current truss lengths, respectively. Finally, according to the orientation of each truss element, we use the rotation matrix \mathbf{T} to convert this information to the global coordinate system, i.e.

$$\mathbf{q} = \mathbf{T}^\top \mathbf{q}_l. \quad (165)$$

B.2 TANGENT STIFFNESS MATRIX

In order to establish the static equilibrium, the balance of forces must be achieved considering that the residuum is null, as expressed by

$$\mathbf{r}(\mathbf{u}) = \mathbf{Q}(\mathbf{u}) - \mathbf{F}(\mathbf{u}) = \mathbf{0}, \quad (166)$$

where $\mathbf{Q}(\mathbf{u})$ and $\mathbf{F}(\mathbf{u})$ are the global internal and external force vectors. Based on a Taylor series expansion, the residuum $\mathbf{r}(\mathbf{u})$ can be linearized, as given by

$$\mathbf{r}(\mathbf{u}^{k+1}) = \mathbf{r}(\mathbf{u}^k) + (\mathbf{u}^{k+1} - \mathbf{u}^k) \frac{\partial \mathbf{r}(\mathbf{u}^k)}{\partial \mathbf{u}^k} \approx \mathbf{0}, \quad (167)$$

where the superscript k refers to the iteration number. In the case that the global external forces are not affected by the displacement values, the global tangent stiffness matrix can be defined as

$$\mathbf{K}_T = \frac{\partial \mathbf{r}(\mathbf{u}^k)}{\partial \mathbf{u}^k} = \frac{\partial \mathbf{Q}(\mathbf{u}^k)}{\partial \mathbf{u}^k} - \frac{\partial \mathbf{F}(\mathbf{u}^k)}{\partial \mathbf{u}^k}. \quad (168)$$

Thus, the displacement increment $\Delta \mathbf{u}$ is obtained as a solution of the linear equation system indicated by

$$\mathbf{K}_T(\mathbf{u}^k) \Delta \mathbf{u}^k = -\mathbf{r}(\mathbf{u}^k). \quad (169)$$

In order to evaluate the convergence of the iterative solution, it is necessary to compute the residuum values² for a certain load increment, which are defined in terms of internal and external forces with the expression

$$r = \left| \frac{\mathbf{Q} - \mathbf{F}}{\mathbf{F}} \right| < tol. \quad (170)$$

The global tangent stiffness matrix and the global internal force vector are obtained considering the contribution of the total number of truss elements nel by using the assembly operator \bigwedge , as shown by

$$\mathbf{K}_T = \frac{\partial \mathbf{Q}(\mathbf{u}^k)}{\partial \mathbf{u}^k} = \bigwedge_{e=1}^{nel} \mathbf{k}_T \quad \text{and} \quad \mathbf{Q} = \bigwedge_{e=1}^{nel} \mathbf{q}. \quad (171)$$

² In this work, the finite element solver considered the convergence tolerance $tol = 10^{-10}$. Additionally, the essential boundary conditions were imposed using the penalty method with the factor $P = 10^6$.

For each truss element, the tangent stiffness matrix set in the global coordinate system \mathbf{k}_T is obtained using Eq. 165 and is given by

$$\mathbf{k}_T = \frac{\partial \mathbf{q}(\mathbf{u})}{\partial \mathbf{u}} = \frac{\partial \left(\mathbf{T}^\top(\mathbf{u}) \mathbf{q}_l(\mathbf{u}) \right)}{\partial \mathbf{u}} = \underbrace{\frac{\partial \mathbf{T}^\top(\mathbf{u})}{\partial \mathbf{u}} \mathbf{q}_l(\mathbf{u})}_{\mathbf{k}_{T1}} + \underbrace{\mathbf{T}^\top(\mathbf{u}) \frac{\partial \mathbf{q}_l(\mathbf{u})}{\partial \mathbf{u}}}_{\mathbf{k}_{T2}}, \quad (172)$$

in which the tangent stiffness matrix is split into two terms, \mathbf{k}_{T1} and \mathbf{k}_{T2} , due to the product rule.

The main form of the term \mathbf{k}_{T1} does not change either for the total or for the updated Lagrangean descriptions; only $\tilde{\mathbf{E}}$ or $\bar{\mathbf{E}}$ must be adequately chosen according to the adopted conjugate pair, as shown in Tab. 12.

Table 12 – Tangent stiffness expressions: \mathbf{k}_{T1} .

Total Lagrangean	Updated Lagrangean
$\mathbf{k}_{T1} = \int_{-1}^{+1} \mathbf{B}^\top(\mathbf{X}_l) \tilde{\mathbf{E}} \mathbf{B}(\mathbf{X}_l) A_0 J(\mathbf{X}_l) d\xi$	$\mathbf{k}_{T1} = \int_{-1}^{+1} \mathbf{B}^\top(\mathbf{x}_l) \bar{\mathbf{E}} \mathbf{B}(\mathbf{x}_l) A J(\mathbf{x}_l) d\xi$
$\tilde{\mathbf{E}}^{RE} = \lambda_1^{-1} \sigma^{RE} \left[\mathbf{I} - \mathbf{B}(\mathbf{X}_l) \frac{\mathbf{x}\mathbf{x}^\top}{\lambda_1^2} \mathbf{B}^\top(\mathbf{X}_l) \right]$	$\bar{\mathbf{E}}^{RE} = \lambda_1 \sigma^{RE} \left[\mathbf{I} - \mathbf{B}(\mathbf{x}_l) \mathbf{x}\mathbf{x}^\top \mathbf{B}^\top(\mathbf{x}_l) \right]$
$\tilde{\mathbf{E}}^C = \lambda_1^{-2} \sigma^C \left[\mathbf{I} - \mathbf{B}(\mathbf{X}_l) \frac{\mathbf{x}\mathbf{x}^\top}{\lambda_1^2} \mathbf{B}^\top(\mathbf{X}_l) \right]$	$\bar{\mathbf{E}}^C = \sigma^C \left[\mathbf{I} - \mathbf{B}(\mathbf{x}_l) \mathbf{x}\mathbf{x}^\top \mathbf{B}^\top(\mathbf{x}_l) \right]$
$\tilde{\mathbf{E}}^{2PK} = \lambda_1^{-1} \sigma^{2PK} \left[\mathbf{I} - \mathbf{B}(\mathbf{X}_l) \frac{\mathbf{x}\mathbf{x}^\top}{\lambda_1^2} \mathbf{B}^\top(\mathbf{X}_l) \right]$	$\bar{\mathbf{E}}^{2PK} = \lambda_1 \sigma^{2PK} \left[\mathbf{I} - \mathbf{B}(\mathbf{x}_l) \mathbf{x}\mathbf{x}^\top \mathbf{B}^\top(\mathbf{x}_l) \right]$

Source – Muñoz-Rojas (n.d.).

In these equations, it should be noted that the stress values can be obtained using the hyper-dual procedure. The required strain-displacement matrices $\mathbf{B}(\mathbf{X}_l)$ and $\mathbf{B}(\mathbf{x}_l)$ are given by

$$\mathbf{B}(\mathbf{X}_l) = \frac{1}{2J(\mathbf{X}_l)} \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{B}(\mathbf{x}_l) = \frac{1}{\lambda_1} \mathbf{B}(\mathbf{X}_l). \quad (173)$$

The evaluation of the term \mathbf{k}_{T2l} of the stiffness matrix is analogous and is summarized in Tab. 13. In this case, the tangent modulus can be obtained using the hyper-dual procedure. Notice that, as also observed in the stiffness matrix term \mathbf{k}_{T1} , the main form of the term \mathbf{k}_{T2} does not change either for the total or the updated Lagrangean descriptions, and now only $\tilde{\mathbf{E}}$ or $\hat{\mathbf{E}}$ must be adequately chosen according to the adopted conjugate pair.

Like in the case of internal force, we can rotate the stiffness matrix to the global coordinate system. In this case, the term \mathbf{k}_{T2} is calculated using the expression

$$\mathbf{k}_{T2} = \mathbf{T}^\top \underbrace{\frac{\partial \mathbf{q}_l}{\partial \mathbf{u}_l}}_{\mathbf{k}_{T2l}} \mathbf{T}. \quad (174)$$

Table 13 – Tangent stiffness expressions: \mathbf{k}_{T2l} .

Total Lagrangean	Updated Lagrangean
$\mathbf{k}_{T2l} = \int_{-1}^{+1} \mathbf{B}^{*\top}(\mathbf{X}_l) \check{E} \mathbf{B}^*(\mathbf{X}_l) A_0 J(\mathbf{X}_l) d\xi$	$\mathbf{k}_{T2l} = \int_{-1}^{+1} \mathbf{B}^{*\top}(\mathbf{x}_l) \hat{E} \mathbf{B}^*(\mathbf{x}_l) A J(\mathbf{x}_l) d\xi$
$\check{E}^{RE} = E_T^{RE}$	$\hat{E}^{RE} = \lambda_1^2 E_T^{RE}$
$\check{E}^C = \lambda_1^{-2} \left(E_T^C - \sigma^C \right)$	$\hat{E}^C = E_T^C - \sigma^C$
$\check{E}^{2PK} = E_T^{2PK} + \lambda_1^{-1} \sigma^{2PK}$	$\hat{E}^{2PK} = \lambda_1^2 E_T^{2PK} + \lambda_1 \sigma^{2PK}$

Source – Muñoz-Rojas (n.d.).

Finally, the summation of \mathbf{k}_{T1} and \mathbf{k}_{T2} completes the determination of \mathbf{k}_T , as indicated in Eq. 172.

The relevance of such formulation is that it leads to expressions very similar to the ones already established for general nonlinear finite element computations. In this formulation setting, it is straightforward to adopt different energetically conjugate pairs, with minimal modifications in the numerical code. In addition, the integration of the equations is set both in the original and in the deformed rotated configurations, which are kinematically associated to the total and updated lagrangean descriptions.

APPENDIX C – DISPLACEMENT SENSITIVITY ANALYSIS

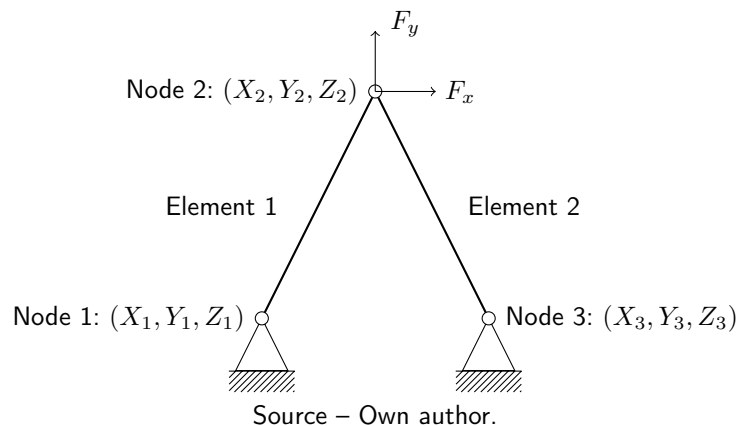
In this appendix, we present a numerical case study to illustrate the hyper-dual sensitivity analysis procedure. For the sake of clarity, we evaluate the second-order displacement sensitivity analysis of a planar truss structure described in Section C.1. The procedures to obtain the hyper-dual form of the internal force and the tangent stiffness matrix are presented in Section C.2. Besides, in Section C.3 we discuss the alternative use of central finite differences to obtain the displacement sensitivity analysis; we focus on accuracy and efficiency issues.

Although we exemplify the hyper-dual procedure in a truss element setting, we highlight that the sensitivity method described in Section 4.2 is general and can be applied in other element formulations. Even considering such a simplistic configuration, the obtained sensitivity information can be related to recent works concerning truss lattices (DUAN et al., 2020; DAYNES et al., 2019; VAISSIER et al., 2019; FENG et al., 2017). In short, as defined by Gibson and Ashby (1997), a lattice material refers to a cellular material of periodic microstructure. In this context, novel lattice geometries with optimal material distribution have been obtained using topology optimization techniques (OSANOV; GUEST, 2016; MUÑOZ-ROJAS et al., 2011). Recently, Ahmadvand and Habibi (2020) studied a second-order approximation method to evaluate the optimum design of truss structures. Therefore, the hyper-dual sensitivity method illustrated in this section might contribute to such works by providing an important instruction for the search direction.

C.1 CASE STUDY

The case study consists of two truss elements depicted in Fig. 21, both with elastic modulus E and initial cross-sectional area A_0 . Nodes 1 and 3 are fixed and a concentrated force is applied on node 2 with components F_x and F_y . The finite element convergence occurs after a few Newton-Raphson iterations; as a result, a vector with the nodal displacement is obtained.

Figure 21 – Truss structure with two elements.



As for the sensitivity analysis, we evaluate the influence of the initial position of node 2 on the deformed configuration. For this case, a vector representing the design variables is described as

$$\mathbf{s} = \begin{Bmatrix} X_2 \\ Y_2 \end{Bmatrix}. \quad (175)$$

C.2 HYPER-DUAL PROCEDURE

In this section, we describe how to perform the sensitivity analysis using the hyper-dual scheme. We meticulously indicate the arithmetic rules for the hyper-dual conversion, as defined in Section 2.2.1. It is noteworthy that this hyper-dual conversion occurs automatically in a computer program if we consider the implementation strategy indicated in Section 4.2.2. In this case, the operation rules are assigned based on the input variable type.

Following the contents presented in Section 2.2, as an initial step we declare the design variables X_2 and Y_2 as hyper-dual numbers X_2 and Y_2 with unit perturbation in the terms ϵ_1 and ϵ_2 , as presented by

$$\begin{aligned} X_2 &= X_2 + \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \epsilon_1 \epsilon_2 \quad \text{and} \\ Y_2 &= Y_2 + \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \epsilon_1 + \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \epsilon_2 + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \epsilon_1 \epsilon_2. \end{aligned} \quad (176)$$

Subsequently, we convert the expressions of internal force and tangent stiffness matrix into the hyper-dual form. For this case study concerning truss elements, we considered the geometrically nonlinear formulation proposed by Muñoz-Rojas (to be published). We illustrate the hyper-dual evaluation of the first element (Fig. 21), which starts with the hyper-dual definition of the undeformed element length as

$$L_0 = \left(\underbrace{\underbrace{(X_2 - X_1)^2}_{\text{Eq. 13}} + \underbrace{(Y_2 - Y_1)^2}_{\text{Eq. 13}}}_{\text{Eq. 22}} + (Z_2 - Z_1)^2 \right)^{\frac{1}{2}}, \quad (177)$$

$\underbrace{\hspace{10em}}_{\text{Eq. 13}}$
 $\underbrace{\hspace{10em}}_{\text{Eq. 24}}$

Notice that the capital letters (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) denote the original nodal position of nodes 1 and 2, respectively. The vector form of the current nodal position \mathbf{x} is calculated

based on the original position \mathbf{X} and the nodal displacement \mathbf{u} , as described by

$$\begin{aligned} & \overbrace{\{x_1 \ y_1 \ z_1 \ x_2 \ y_2 \ z_2\}}^{\mathbf{x}^\top} \\ &= \underbrace{\left\{ \overbrace{\{X_1 \ Y_1 \ Z_1 \ X_2 \ Y_2 \ Z_2\}}^{\mathbf{X}^\top} + \overbrace{\{u_1 \ v_1 \ w_1 \ u_2 \ v_2 \ w_2\}}^{\mathbf{u}^\top} \right\}}_{\text{Eq. 14}}. \end{aligned} \quad (178)$$

Thus, considering the updated values for each coordinate, the element length in the deformed configuration is

$$\begin{aligned} L &= \left(\underbrace{\underbrace{\underbrace{(x_2 - x_1)^2}_{\text{Eq. 35}}}_{\text{Eq. 37}}}_{\text{Eq. 22}} + \underbrace{\underbrace{\underbrace{(y_2 - y_1)^2}_{\text{Eq. 35}}}_{\text{Eq. 37}}}_{\text{Eq. 22}} + (z_2 - z_1)^2 \right)^{\frac{1}{2}}, \end{aligned} \quad (179)$$

Eq. 13
Eq. 35
Eq. 24

where (x_1, y_1, z_1) and (x_2, y_2, z_2) denotes the current nodal position of nodes 1 and 2, respectively. We omitted the references to the hyper-dual arithmetics since it works analogously to converting the undeformed element length.

Based on the element length information, we can write the hyper-dual form of Jacobian considering the undeformed position, the longitudinal element stretch and the engineering stress¹. These variables are respectively converted according to

$$J(\mathbf{X}_l) = \frac{L_0}{2} = \underbrace{L_0 \frac{1}{2}}_{\text{Eq. 40}}, \quad \lambda = \frac{L}{L_0} = L \underbrace{L_0^{-1}}_{\substack{\text{Eq. 23} \\ \text{Eq. 25}}} \quad \text{and} \quad \sigma^{RE} = E \underbrace{(\lambda - 1)}_{\substack{\text{Eq. 37} \\ \text{Eq. 40}}}. \quad (180)$$

Assuming a case with a linear shape function, the strain-displacement matrix $\mathbf{B}(\mathbf{X}_l)$ and its

¹ The superscript *RE* denotes the Rotated Engineering conjugate pair.

modified version² $\mathbf{B}^*(\mathbf{X}_l)$ are defined as

$$\mathbf{B}(\mathbf{X}_l) = \underbrace{\left(\underbrace{2J(\mathbf{X}_l)}_{\text{Eq. 40}} \right)^{-1}}_{\text{Eq. 23}} \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix} \quad \text{and}$$

$$\mathbf{B}^*(\mathbf{X}_l) = \underbrace{\left(\underbrace{2J(\mathbf{X}_l)}_{\text{Eq. 40}} \right)^{-1}}_{\text{Eq. 23}} \left\{ -1 \ 0 \ 0 \ 1 \ 0 \ 0 \right\}. \quad (181)$$

Eq. 47
Eq. 43

The internal force in the local coordinate system³ is converted into a hyper-dual representation using the mentioned product operators, as given by

$$\mathbf{q}_l^{RE} = \int_{-1}^{+1} \mathbf{B}^{*\top}(\mathbf{X}_l) \sigma^{RE} A_0 J(\mathbf{X}_l) d\xi = 2 \underbrace{\mathbf{B}^{*\top}(\mathbf{X}_l)}_{\text{Eq. 32}} \underbrace{\sigma^{RE} A_0 J(\mathbf{X}_l)}_{\text{Eq. 40}}, \quad (182)$$

Eq. 41 Eq. 25
Eq. 26

where ξ stands for the isoparametric element formulation. We rotate the internal force to the global coordinate system using the expression

$$\mathbf{q}^{RE} = \underbrace{\mathbf{T}^\top}_{\text{Eq. 32}} \underbrace{\mathbf{q}_l^{RE}}_{\text{Eq. 30}}. \quad (183)$$

where the rotation matrix \mathbf{T} relates the local and the global coordinate system; its hyper-dual

² The modified strain-displacement matrix considering the undeformed position is represented as a row vector.

³ The subscript l indicates the local coordinate system.

form is written as⁴

$$\mathbf{T} = \underbrace{L^{-1}}_{\text{Eq. 23}} \left(\underbrace{\left(\underbrace{\left(\underbrace{x_2}_{\text{Eq. 35}} \quad -x_1 \right)}_{\text{Eq. 37}} \right)}_{\text{Eq. 47}} \mathbf{T}_x + \underbrace{\left(\underbrace{y_2}_{\text{Eq. 35}} \quad -y_1 \right)}_{\text{Eq. 37}} \right)_{\text{Eq. 47}} \mathbf{T}_y + (z_2 - z_1) \mathbf{T}_z \Big). \quad (185)$$

In this formulation (MUÑOZ-ROJAS, to be published), the tangent stiffness matrix, defined as the derivative of the internal force with respect to the displacements, is represented by the contribution of two terms⁵ The first term \mathbf{k}_{T1} is defined by the expression

$$\mathbf{k}_{T1} = \int_{-1}^{+1} \mathbf{B}^\top(\mathbf{X}_l) \underbrace{\tilde{\mathbf{E}}^{RE}}_{\text{Eq. 31}} \mathbf{B}(\mathbf{X}_l) \underbrace{A_0 J(\mathbf{X}_l)}_{\text{Eq. 25}} d\xi = \underbrace{\mathbf{B}^\top(\mathbf{X}_l)}_{\text{Eq. 32}} \underbrace{\tilde{\mathbf{E}}^{RE}}_{\text{Eq. 31}} \underbrace{\mathbf{B}(\mathbf{X}_l)}_{\text{Eq. 25}} \underbrace{A_0 J(\mathbf{X}_l)}_{\text{Eq. 40}}. \quad (187)$$

The variable $\tilde{\mathbf{E}}^{RE}$ is defined by

$$\tilde{\mathbf{E}}^{RE} = \underbrace{\sigma^{RE}}_{\text{Eq. 25}} \underbrace{(\lambda_1)^{-1}}_{\text{Eq. 23}} \left[\mathbf{I} - \underbrace{\mathbf{B}(\mathbf{X}_l)}_{\text{Eq. 31}} \underbrace{\mathbf{x}\mathbf{x}^\top}_{\text{Eq. 45}} \underbrace{\lambda_1^{-2}}_{\text{Eq. 22}} \underbrace{\mathbf{B}^\top(\mathbf{X}_l)}_{\text{Eq. 29}} \right], \quad (188)$$

⁴ To write the matrix with the direction cosines, we considered the following constant values:

$$\mathbf{T}_x = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{T}_y = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{T}_z = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (184)$$

⁵ Using the product rule, we obtain

$$\mathbf{k}_T = \frac{\partial \mathbf{q}^{RE}}{\partial \mathbf{u}} = \frac{\partial (\mathbf{T}^\top \mathbf{q}_i^{RE})}{\partial \mathbf{u}} = \underbrace{\frac{\partial \mathbf{T}^\top}{\partial \mathbf{u}} \mathbf{q}_i^{RE}}_{\mathbf{k}_{T1}} + \underbrace{\mathbf{T}^\top \frac{\partial \mathbf{q}_i^{RE}}{\partial \mathbf{u}}}_{\mathbf{k}_{T2}}. \quad (186)$$

where \mathbf{I} is the identity matrix. Similarly, the term \mathbf{k}_{T2_l} is defined as

$$\mathbf{k}_{T2_l} = \int_{-1}^{+1} \mathbf{B}^{*\top}(\mathbf{X}_l) \mathbf{E} \mathbf{B}^*(\mathbf{X}_l) A_0 J(\mathbf{X}_l) d\xi = \underbrace{\mathbf{B}^{*\top}(\mathbf{X}_l)}_{\text{Eq. 26}} \underbrace{\mathbf{E} \mathbf{B}^*(\mathbf{X}_l)}_{\text{Eq. 25}} \underbrace{A_0 J(\mathbf{X}_l)}_{\text{Eq. 25}} \cdot \quad (189)$$

$$\underbrace{\hspace{10em}}_{\text{Eq. 28}} \underbrace{\hspace{10em}}_{\text{Eq. 40}}$$

$$\underbrace{\hspace{15em}}_{\text{Eq. 29}}$$

It should be noted that this term is written in the local coordinate system. We obtain the global coordinate version by using the rotation matrix \mathbf{T} , as indicated by

$$\mathbf{k}_{T2} = \underbrace{\mathbf{T}^\top}_{\text{Eq. 32}} \underbrace{\mathbf{k}_{T2_l}}_{\text{Eq. 31}} \mathbf{T}. \quad (190)$$

$$\underbrace{\hspace{10em}}_{\text{Eq. 31}}$$

Finally, as a consequence of the product rule during its mathematical demonstration (MUÑOZ-ROJAS, to be published), the tangent stiffness matrix is defined by the contribution of the terms \mathbf{k}_{T1} and \mathbf{k}_{T2} , as represented as

$$\mathbf{k}_T = \underbrace{\mathbf{k}_{T1} + \mathbf{k}_{T2}}_{\text{Eq. 15}}. \quad (191)$$

As for the described hyper-dual conversion, the obtained derivatives were taken with respect to the design variables, as a consequence of Eq. 176. Additionally, we can apply a similar approach to obtain the derivatives with respect to the degrees of freedom. This derivative information with respect to the displacements is a requirement for the second-order sensitivity analysis. In this case, we rather consider a unit perturbation applied on the degrees of freedom of a given element. These hyper-dual variables are defined according to the expressions⁶

$$\begin{aligned} u_1 &= u_1 + \left\{ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \right\}^\top \epsilon_1 + \left\{ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \right\}^\top \epsilon_2 + \mathbf{0} \epsilon_1 \epsilon_2, \\ v_1 &= v_1 + \left\{ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \right\}^\top \epsilon_1 + \left\{ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \right\}^\top \epsilon_2 + \mathbf{0} \epsilon_1 \epsilon_2, \\ w_1 &= w_1 + \left\{ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \right\}^\top \epsilon_1 + \left\{ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \right\}^\top \epsilon_2 + \mathbf{0} \epsilon_1 \epsilon_2, \\ u_2 &= u_2 + \left\{ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \right\}^\top \epsilon_1 + \left\{ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \right\}^\top \epsilon_2 + \mathbf{0} \epsilon_1 \epsilon_2, \\ v_2 &= v_2 + \left\{ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \right\}^\top \epsilon_1 + \left\{ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \right\}^\top \epsilon_2 + \mathbf{0} \epsilon_1 \epsilon_2, \quad \text{and} \\ w_2 &= w_2 + \left\{ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \right\}^\top \epsilon_1 + \left\{ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \right\}^\top \epsilon_2 + \mathbf{0} \epsilon_1 \epsilon_2. \end{aligned} \quad (192)$$

The next step involves the hyper-dual conversion of the internal force expression, according to the aforementioned arithmetics.

⁶ The null terms in $\epsilon_1 \epsilon_2$ axis represent a square matrix with size 6, which is related to the number of degrees of freedom of this truss element.

Once the hyper-dual conversion is completed, the first and second-order derivatives of internal force and tangent stiffness are obtained by extracting particular hyper-dual terms. As shown in Section 4.2.2, these calculations are performed for each truss element and, based on the element connectivity, an assembly operator mounts the global tensors. Finally, we obtain the displacement sensitivity analysis as the solution of Eqs. 108 and 109.

C.3 GLOBAL FINITE DIFFERENCE METHOD

Alternatively, we can obtain the second-order displacement sensitivity analysis using the finite difference approximation. In this section, we draw a parallel between this numerical method and the proposed scheme. We provided an initial discussion concerning the computational effort.

The global finite difference method requires the nodal displacement results considering different cases with perturbation values h_1 and h_2 in the design variables X_2 and Y_2 , respectively. Thus, the displacement solutions for the central finite differences calculations are summarized as⁷

$$\begin{aligned} \mathbf{u}^{(+h_1)} &= \mathbf{u}(X_2 + h_1, Y_2), & \mathbf{u}^{(-h_1)} &= \mathbf{u}(X_2 - h_1, Y_2), \\ \mathbf{u}^{(+h_2)} &= \mathbf{u}(X_2, Y_2 + h_2), & \mathbf{u}^{(-h_2)} &= \mathbf{u}(X_2, Y_2 - h_2), \\ \mathbf{u}^{(+h_1, +h_2)} &= \mathbf{u}(X_2 + h_1, Y_2 + h_2), & \mathbf{u}^{(-h_1, -h_2)} &= \mathbf{u}(X_2 - h_1, Y_2 - h_2), \\ & & \text{and } \mathbf{u} &= \mathbf{u}(X_2, Y_2). \end{aligned} \quad (193)$$

As noted, assuming a problem with only two design variables, the cases consist of the original unperturbed problem and six perturbed combinations.

Thus, an approximation for the first-order displacement sensitivity analysis is given by

$$\frac{\partial \mathbf{u}}{\partial X_2} \approx \frac{\mathbf{u}^{(+h_1)} - \mathbf{u}^{(-h_1)}}{2h_1} \quad \text{and} \quad (194)$$

$$\frac{\partial \mathbf{u}}{\partial Y_2} \approx \frac{\mathbf{u}^{(+h_2)} - \mathbf{u}^{(-h_2)}}{2h_2}. \quad (195)$$

Similarly, the second-order information is given by

$$\frac{\partial^2 \mathbf{u}}{\partial X_2 \partial X_2} \approx \frac{\mathbf{u}^{(+h_1)} - 2\mathbf{u} + \mathbf{u}^{(-h_1)}}{h_1^2}, \quad (196)$$

$$\frac{\partial^2 \mathbf{u}}{\partial Y_2 \partial Y_2} \approx \frac{\mathbf{u}^{(+h_2)} - 2\mathbf{u} + \mathbf{u}^{(-h_2)}}{h_2^2}, \quad \text{and} \quad (197)$$

$$\begin{aligned} \frac{\partial^2 \mathbf{u}}{\partial X_2 \partial Y_2} &= \frac{\partial^2 \mathbf{u}}{\partial Y_2 \partial X_2} \\ &\approx \frac{1}{2h_1 h_2} \left(\mathbf{u}^{(+h_1, +h_2)} + \mathbf{u}^{(-h_1, -h_2)} + 2\mathbf{u} - \mathbf{u}^{(+h_1)} - \mathbf{u}^{(+h_2)} - \mathbf{u}^{(-h_1)} - \mathbf{u}^{(-h_2)} \right). \end{aligned} \quad (198)$$

⁷ The superscripts represent the application of a perturbation value in a design variable.

In terms of the computational effort, as exemplified in the expressions shown in Eq. 193, it should be noted that the use of the finite difference method requires the solution of numerous nonlinear finite element analyses. Even considering the first-order evaluation, this method is not efficient for problems with a large number of design variables (WANG, Yingjun et al., 2018). Although it typically involves a straightforward implementation, its accuracy is highly affected by the perturbation size, especially for the second-order evaluation (CHAPRA; CANALE, 2006). This overall finite difference scheme also suffers from errors when the perturbed solution converges with a different number of iterations (TRENTIN et al., 2009). However, by considering appropriate perturbation values, we can obtain consistent derivative results.

Conversely, the hyper-dual method provides exact calculations for both first and second-order derivatives, regardless of the perturbation values (FIKE, Jeffrey Alan, 2013). Another significant advantage of the proposed scheme is related to the fact that it only requires the finite element solution of the original unperturbed problem, as depicted in Fig. 12. The benefits of this strategy would be even more evident in intricate structural problems that require several sub-steps and iterations. Therefore, the hyper-dual scheme addresses important drawbacks related to the finite difference approximation.