

UNIVERSITÀ DEGLI STUDI DI PAVIA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA CIVILE

**DEVELOPMENT AND RAPID PROTOTYPING OF NEW
NUMERICAL MODELS ORIENTED TO THE HONEYCOMB
SANDWICH PANELS MODELING**

Sviluppo e prototipazione rapida di nuovi modelli numerici
orientati alla modellazione di pannelli a sandwich
con struttura a nido d'ape

Supervisor:

Prof. Ferdinando Auricchio

Author:

Luca Albertario

Academic Year 2009-2010

Abstract

The present research work is mainly focused on the use of two innovative FEM software packages, created by Prof. Jože Korelc, especially designed for the development of new numerical models. Such software are the *AceGen*, which is an on-demand numerical code generator, and its sibling finite element environment *AceFem*. Since these programs employ a really innovative approach based on a symbolic formulation and automatic derivation of formulae needed in numerical procedures, a wide study of their main features is therefore presented.

The numerical models implemented using such software are oriented to develop a new phenomenological constitutive model able to well-simulate the crushing behavior of honeycomb-cored sandwich panels. Nowadays, the studies on the crashworthiness of such structures are subject of great scientific interest which involve various fields of engineering applications. For these reasons, a former detailed study on the mechanical and geometrical features of honeycomb core materials is needful.

In order to achieve the ambitious modeling results, this thesis advances a *multi-step* approach, which allows the user to define the required numerical model through the development of a logical sequence of intermediate models, by following a gradual order of complexity and precision. All the proposed models have been developed and calibrated starting from the results obtained through several tests carried out by the Department of Structural Engineering, University of Naples “Federico II”.

Acknowledgements

Voglio esprimere un ringraziamento particolare al mio relatore Prof. Ferdinando Auricchio per la fiducia che ha mostrato in me. Lavorare con lui e essere parte della sua squadra è stato davvero un onore per me. L'incoraggiamento e il suo costante sostegno mi ha permesso di completare raggiungere il mio obiettivo di tesi con successo.

Ringrazio inoltre tutte le persone che con la loro presenza e il loro appoggio morale mi hanno aiutato e sostenuto psicologicamente ad affrontare tutti questi cinque meravigliosi anni universitari.

A questo scopo il mio primo ringraziamento è indirizzato a tutta la mia famiglia che si è rivelata, come sempre lo è stato, fonte continua di sostegno morale e fiducia, anche nei momenti di difficoltà che ho dovuto affrontare. In particolare mia sorella Anna, sostegno insostituibile e spesso motivo di sfogo psicologico. Ringrazio i miei genitori e mia nonna, che hanno creduto in me e che con la loro esperienza mi hanno sempre saputo dare la giusta risposta nel momento di bisogno.

Ringrazio Isabella, che è sempre stata capace di rassicurarmi e comprendermi più di chiunque altro.

Ringrazio gli amici che mi hanno accompagnato in questa splendida avventura universitaria in particolare Paolo, Corbe, Jacopo, Giulia, Stefano, Antonio con cui ho condiviso centinaia di pause caffè, nottate di studio ma anche i momenti più divertenti.

Ringrazio per la pazienza tutta la mia compagnia di amici mi ha sempre rassicurato e ed è sempre stata motivo di gioia e serenità.

Ringrazio Marco, amico fedele e sincero, e tutta la sua famiglia. Ringrazio la mia squadra di pallacanestro che, nonostante qualche infortunio di troppo, si è rivelata uno sfogo e motivo di divertimento.

TABLES OF CONTENTS

Abstract	i
Acknowledgements	ii
TABLES OF CONTENTS	iii
LIST OF FIGURES	vii
1. Introduction	1
1.1 Overview on the traditional FEM environment.....	1
1.2 Research aim	2
1.3 Research outline	3
2. Honeycomb sandwich panel structures	5
2.1 Overview on sandwich structures features	5
2.1.1 Introduction.....	5
2.1.2 Components and internal structure.....	6
2.2 Honeycomb core features	10
2.2.1 Definition.....	10
2.2.2 Application fields	12
2.2.3 Manufacturing process	13
2.2.4 Honeycomb classification.....	15
2.3 Macro-Scale mechanical properties	18
2.3.1 Orthotropy definition.....	18
2.3.2 Honeycomb orthotropy.....	19
2.3.3 Static Test methods	21
2.3.4 Analytical models.....	25

2.4	Experimental investigation.....	28
2.4.1	Introduction.....	28
2.4.2	Material specimens.....	29
2.4.3	Static Tests.....	30
2.4.4	Dynamic tests: Drop weight test.....	33
2.4.5	Conclusions.....	35
2.5	Literature survey on virtual testing of sandwich core structures.....	36
3.	AceGen and AceFem.....	38
3.1	Introduction.....	39
3.1.1	AceGen and AceFem overview.....	39
3.1.2	Traditional FEM applications.....	40
3.1.3	Hybrid Symbolic-Numerical approach.....	41
3.2	AceGen.....	44
3.2.1	Features and advantages.....	44
3.2.2	Mathematica–AceGen combination.....	46
3.2.3	Symbolic-Numeric Interface: Auxiliary variables.....	47
3.2.4	AceGen codes and commercial FE environments.....	50
3.2.5	Standard FE procedure.....	51
3.2.6	Automatic differentiation and FEM.....	54
3.2.7	Debugging module.....	58
3.3	AceFem.....	59
3.3.1	AceFem overview.....	59
3.3.2	Comparison between Traditional commercial FEM environments and AceFem.....	61
3.3.3	AceFem internal Structure.....	63

3.3.4	General AceFem procedure	64
3.3.5	AceShare.....	65
3.3.6	Interactive Debugging	66
4.	<i>Multi-step</i> approach: Developed numerical models	67
4.1	Introduction	67
4.1.1	Overview of the proposed numerical models	68
4.1.2	Finite strain kinematical recalls	70
4.1.3	General Isoparametric concept.....	72
4.2	Hyperelastic elements	74
4.2.1	Hyperelastic model formulation	74
4.2.2	Bi-linear quadrilateral.....	76
4.2.3	Three-dimensional isoparametric element	77
4.2.4	AceGen input file: Hypersolid2D_Quadrilateral	78
4.2.5	AceGen input file: Hypersolid3D_Hexahedral.....	82
4.3	Element <i>Mini</i> implementation.....	83
4.3.1	Overview on incompressible materials	83
4.3.2	Element Mini formulation	84
4.3.3	AceGen input file: MINI_7_int_point.....	85
4.4	Three-dimensional elasto-plastic element.....	87
4.4.1	Introduction on inelastic strain: small displacement gradients	87
4.4.2	Classical plasticity	89
4.4.3	J2 classical plasticity	90
4.4.4	Proposed formulation for J ₂ plasticity.....	92
4.4.5	J2 plasticity with particular kinematic hardening rule....	94

4.4.6	AceGen input file: Element_J2_My_Hard.....	95
4.5	Honeycomb-core constitutive model proposed by <i>Mohr</i>	103
5.	AceFem: Numerical simulation and results.....	105
5.1	Introduction.....	105
5.1.1	Debug module.....	106
5.1.2	Example of AceFem input file.....	107
5.2	Numerical analysis on 2D domain.....	108
5.2.1	Hyperelastic element.....	110
5.2.2	Element Mini.....	110
5.3	Numerical analysis on 3D domain.....	111
5.3.1	Hyperelastic 3D element.....	113
5.3.2	Classical J2 plasticity element.....	114
5.3.3	J2 plasticity with particular kinematic hardening rule..	115
6.	Conclusions and future work.....	116
6.1	Developed activity.....	116
6.2	Contribution to the modeling of honeycomb core.....	118
6.3	Future work.....	119
7.	Bibliography and reference works.....	120
	Appendix A.....	123
	Appendix B.....	128

LIST OF FIGURES

Figure 2-1 Increase of stiffness due to honeycomb structure presence, maintaining constant total weight.....	8
Figure 2-2 Schematic detailed example of the honeycomb sandwich structure.....	10
Figure 2-3 Honeycomb cell structure.....	11
Figure 2-4 Micro structural geometry and his main directions.....	11
Figure 2-5 Expansion Process of Honeycomb Manufacture	13
Figure 2-6 Corrugated Process of Honeycomb Manufacture	14
Figure 2-7 cell configuration	17
Figure 2-8 Main directions characterizing the geometry of honeycomb structures	19
Figure 2-9 Definition of in- and out-of-plane loading.....	21
Figure 2-10 Schematic representation of Compressive Test [24]	22
Figure 2-11 “T” direction compressive load profile [5]	23
Figure 2-12 Schematic example of tensile plate shear methods [24] .	23
Figure 2-13 Arcan apparatus pinned (left side) and clamped (right side) configuration [5].....	24
Figure 2-14 Elastic deformation on W and L directions [Gibson L.J. and Ashby M.F. Cellular Solids [25]].....	25
Figure 2-15 Deformed honeycomb due to out-of-plane compression loading.....	26
Figure 2-16 Alexander's model: Folding of thin walls in a cylinder	26
Figure 2-17 McFarland model: ‘T’ direction crushing mechanism	27
Figure 2-18 Specimens manufacturer data-sheet	29
Figure 2-19 Experimental results of static flat-wise stabilized compression test.	30
Figure 2-20 Static indentation test of sandwich beam	31
Figure 2-21 residual dent at different displacement levels during static indentation tests.....	32

Figure 2-22	Experimental results of indentation test: (left) force-displacement curves; (right) residual dent depth at different imposed displacement values.	32
Figure 2-23	Dynamic compression tests on honeycomb structures: filtering data (left side); stress-strain curves at different strain rates (right side).....	34
Figure 2-24	Time-consuming approach: full geometrical representation of honeycomb core.....	37
Figure 3-1	Multi-language and multi-environment FE code generatio	42
Figure 4-1	Finite element interpolation:	72
Figure 4-2	Two dimensional gauss integration for quadrilateral elements	76
Figure 4-3	Isoparametric 8-node hexahedral element.....	77
Figure 4-4	Element MINI nodal characterization.....	84
Figure 4-5	27 integration point are defined (3 x 3 x 3 Gauss integration rule).....	95
Figure 5-1	Run time debugging module	106
Figure 5-2	Run time debugging example.....	107
Figure 5-3	Domain discretized using elements MINI.....	108
Figure 5-4	Deformed mesh after the numerical analysis	109
Figure 5-5	Element Hyperelastic 2D: stress-strain response.....	110
Figure 5-6	Element MINI stress-strain response	110
Figure 5-7	Domain Ω^3 a geometry and boundary conditions	111
Figure 5-8	Deformed mesh after the numerical analysis	112
Figure 5-9	Hyperelastic 3D element: out of plane strain-stress relationship.	113
Figure 5-10	elasto-perfectly plastic 3D: out of plane strain-stress relationship.....	114
Figure 5-11	original elasto-plastic 3D: out of plane strain-stress relationship.....	115

1. Introduction

1.1 Overview on the traditional FEM environment

Nowadays, the numerical simulations are well established in several engineering fields such as in automotive, aerospace, civil engineering. Considerable improvements in these fields have already been achieved by using standard features of the currently available finite element (FE) packages, where the mathematical models are described by a system of partial differential equations. Most of the existing numerical methods for solving partial differential equations can be classified into two classes: *Finite Difference Method* (FDM) and *Finite Element Method* (FEM). Unfortunately, the applicability of such numerical methods is often limited; the search for methods which can provide a general tool for arbitrary problems in mechanics of solids has a long history.

Using the traditional commercial finite element environment, such as *FEAP* (Taylor, 1990), *ABAQUS*, etc., for analyzing a variety of problems and for incorporating new elements is now already an everyday practice [20]. However large FE systems can be for the development and testing of new numerical procedures awkward.

The basic numerical tests performed on a single finite element or on a small patch of elements can be done most efficiently by using the general symbolic-numeric environments such as *Mathematica*, *Maple*, etc. Unfortunately, such symbolic-numeric environments become very inefficient if there is a larger number of elements. In order to assess element performances under real conditions the easiest way is to perform tests on sequential machines with good debugging capabilities (typically personal computers and programs written in FORTRAN or C/C++ language).

For this reasons, during the development of new numerical models, the classical approach, based on the re-coding of the element in different languages would be extremely time consuming and it is never done.

1.2 Research aim

The main purpose of this thesis is to show how the combination of the finite element environment *AceFem* with its sibling code-generator *AceGen* package, represents an ideal tool for the development of new numerical models.

*Wolfram Research*¹ is recently releasing such two innovative packages which bring to *Mathematica* a robust and optimized architecture for a rapid numerical prototyping and a finite element analysis. *Mathematica* is a well known software program largely used in several technical computing like scientific, engineering and other mathematical fields. In particular, during this research work, it has been used the *Mathematica v.7.0.0*, and the *AceGen-AceFem v.2.3*. Each package combines use of *Mathematica*'s facilities with external handling of intensive computation by compiled modules.

¹ **Wolfram Research** is one of the world's most respected software companies that produce mathematics programs. His main product is *Mathematica*.

This package of *Mathematica* applications was written and developed by Jože Korelc, professor at the Faculty of Civil and Geodetic engineering of University of Ljubljana.

In order to explore the great potential of such packages, several numerical models have been implemented with the aim of establishing a new phenomenological constitutive model able to well-simulate the crushing behavior of honeycomb-cored sandwich panels. Taking into account the great complexity of such materials, the modeling of honeycomb core materials is a really ambitious goal. After all, the studies on the crashworthiness of such structures are subject of great engineering interest. Despite of his increasing use in several industrial applications, the sandwich honeycomb materials are not still extensively investigated on how material properties and geometrical configuration can affect the hypothetical damage.

In order to achieve the aspiring modeling results, this thesis advances a *multi-step* approach, which allows the user to define the required numerical model through the development of a logical sequence of intermediate models, by following a gradual order of complexity and precision.

1.3 Research outline

Considering the above mentioned aim of this thesis, **Chapter 2** provides a needful former detailed study on the mechanical and geometrical features of honeycomb core materials. Moreover, it presented an experimental investigation on *Nomex*[®] honeycomb, carried out by the University of Naples “Federico II”.

A large overview on *AceGen* and *AceFem* software is presented in **Chapter 3**. *AceGen* is a *Mathematica* package designed for the automatic deriving of formulae needed in numerical procedures. The great advantage of this application is that different techniques are

combined in order to shorten the evolution route of any computing procedures. Moreover the results could be exported as compiled *FORTRAN* or *C* code with automated interfacing. *AceGen* is set up to talk with other numerical environments, including its sibling *AceFem*. This extremely innovative capability is due to a new hybrid symbolic-numerical approach.

AceFem is the second part of this applications package, and it consist in a innovative finite element environment offering parallel options for either manual exploration of behavior or high speed black-box computation of results. The *AceFem* package explores advantages of symbolic capabilities of Mathematica while maintaining numerical efficiency of commercial finite element environments. *AceFem* application could employ specific own elements or general codes previously created by *AceGen*.

Chapter 4 represents the core of the thesis, since all the phases of the *multi-step* approach are described. This particular multi-step process allows the user to formulate a new constitutive model, through the development of a logical sequence of intermediate constitutive models, by following a gradual order of complexity and precision. During this chapter the development of a continuum model for the honeycomb core structure is advanced.

Chapter 5 presents the results of the numerical simulation executed through the use of *AceFem* finite element environment.

2. Honeycomb sandwich panel structures

In this chapter, after a general description of the theory behind the sandwich structures, the attention is focused on the sandwich honeycomb-cored structures. In particular a presentation of the most characterizing features and a macro-scale mechanical characterization is proposed.

Later an example of experimental investigation on Nomex® core materials is proposed in order to calibrate the future numerical models. At the end a literature survey on the virtual testing of sandwich core structures is proposed.

2.1 Overview on sandwich structures features

2.1.1 Introduction

The theory behind sandwich structures is a well-known technology, whose basic concept is to use thin, dense and strong external sheets

(skins) which are bonded to a thick light-weight *core* material. Individually the components are weak but when they work together they provide an extremely stiff, strong and light-weight structure.

The use of sandwich panels is rapidly increasing for applications ranging from satellites, aircrafts, bridge construction, ships, automobiles, rail cars and many more. The main advantage of the employ of sandwich construction is the development of new structures with high performance and low weight ratios [1].

The bending behavior of a sandwich panel can be reasonably compared to an *I-beam*, but with the flanges and web extended in all directions. The skins of a sandwich panel could be correlated with the flanges of the *I-beam*, and the sandwich core is similar to the web. However, because it is a panel, there is bending strength in all planes, not only the Y-Z plane (like the I-beam) but the X-Z plane too. When a sandwich panel is bent, one skin experiences tension, and the other skin experiences compression.

The strength of a sandwich structures is the result of a combination of his main components properties. Any damage accumulated in one, or more, of the base materials (skin, core and interface) will have an overall effect on the behavior of the complete sandwich

2.1.2 Components and internal structure

The sandwich composites are based on the **sandwich theory** which describes the behavior of a beam, plate or shell by a three layers system:

- the *facing sheets (skins)*;
- the *core-to-facing bonding adhesives*.
- the *core*.

The **facing sheets** play a fundamental role since they basically withstands the in-plane and bending actions, due to his higher elastic properties. Common materials for the sandwich skins are fiber-composite or wood laminates and thin aluminum sheets.

Adhesives' role in the sandwich structures is to keep the faces and the core co-operating with each other. The adhesive between the faces and the core must be able to transfer the shear forces between the faces and the core. The adhesive must be able to carry shear and tensile stresses. During the present research paper the presence of the core-to-facing bonding adhesive isn't taking into account, since of his complex nature.

The **core** of a sandwich structure represents the most important layer, since his geometric and constitutive characteristics are strongly important in order to define the behavior of the whole structure. In fact, the core is basically required in order to keep the skins at the correct distance apart and to avoid that one face could slide over the other one when a honeycomb sandwich panel subjected to flexural loading. Such rigidities are called the *transverse* and *shear* stiffness of the core. In other words, it must be rigid and strong in direct tension and compression (perpendicular to the faces) and in shear (in the planes perpendicular to the faces).

The presence of the thick core means that the deformation characteristics of a sandwich shell are different from those of a classical laminated shell or a monolayer structure. Although the contribution from the core to the overall weight of the sandwich structure is minimal, the moment of inertia of the structure increases substantially by the separation of the face carriers from the core, thanks to the presence of the thick core.

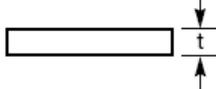
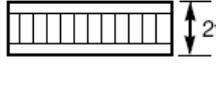
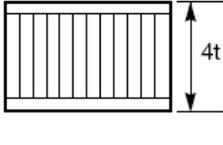
	Solid Metal Sheet	Sandwich Construction	Thicker Sandwich
			
Relative Stiffness	100	700 7 times more rigid	3700 37 times more rigid!
Relative Strength	100	350 3.5 times as strong	925 9.25 times as strong!
Relative Weight	100	103 3% increase in weight	106 6% increase in weight

Figure 2-1 Increase of stiffness due to honeycomb structure presence, maintaining constant total weight

On the other hand, due to the limited flexural rigidity of the low-density core, the influences of the local phenomena, such as local instability, load-induced thickness reduction, etc., on the global phenomena, such as stability, deformation, etc., have to be taken into account during the designing phase of sandwich structures. Hence, micromechanical models dealing with sandwich shells should place proper emphasis on the importance of the core's mechanical behavior.

A big number of core materials and core configurations have been proposed nowadays. The most commonly used sandwich core materials exploited for mechanical applications, could be divided into two main groups: *homogeneous core* (like wood cores and foam cores) and *structured cores* (like honeycombs, corrugated cores and textile cores) which offer high stiffness with less weight and they can be constructed from any material.

The choice of materials used to create these composite structures depends not only on strength and stiffness requirements, but also on process and cost considerations. Moreover, the selection of a specific core (honeycomb or foam structures) for a particular engineering

application is guided by models which describe their mechanical behavior in terms of cell geometry and deformation and failure mechanisms.

For example, the *foam* cores are preferably used when the waterproof, sound and heat insulation qualities of cores are required. Additionally, the foam cores are the least expensive among core materials and can offer some advantages in sandwich manufacture.

The *honeycomb core*, which will be described in the following chapters, possess a higher stiffness-to-weight ratio compared to foam core materials. However, the weakest point of such core is the small adhesive area of honeycomb cells with face sheets that due to manufacturing defects or in-service conditions.

The filling of honeycomb cells with foam can be considered as the enhancement of debonding resistance and ability to produce new types of sandwich cores. This concept combines the benefits of honeycomb and foam cores. The increased adhesive area of foam-filled honeycomb cells is only one of them. On the other hand, the filling leads to changes of the dynamic properties of the honeycomb sandwiches.

Sometimes, for simplicity and efficiency, the cellular honeycomb core could be idealized as a homogeneous material and its equivalent mechanical properties are used in analysis and design. Therefore, the knowledge of the equivalent transverse shear stiffness of honeycomb is very important for the analysis and design of sandwich plates.

2.2 Honeycomb core features

In this Section, as previously posted, are introduced the most characterizing features of the honeycomb-cored sandwich structures. The interest of this research work is focused on the honeycomb structures made from *Nomex*[®] 2 [8], and Aluminum honeycombs, since the employment of these kinds of structures is strongly increasing in order to replace traditional materials in highly loaded applications.

2.2.1 Definition

Honeycomb materials are described as cellular solids, materials that make use of voids to decrease mass, whilst maintaining qualities of stiffness and energy absorption. Moreover, the honeycomb structures have been received much attention in recent years because of their high strength/weight and stiffness/weight ratios, good acoustic properties, excellent heat resistance and favorable energy-absorbing capacity.

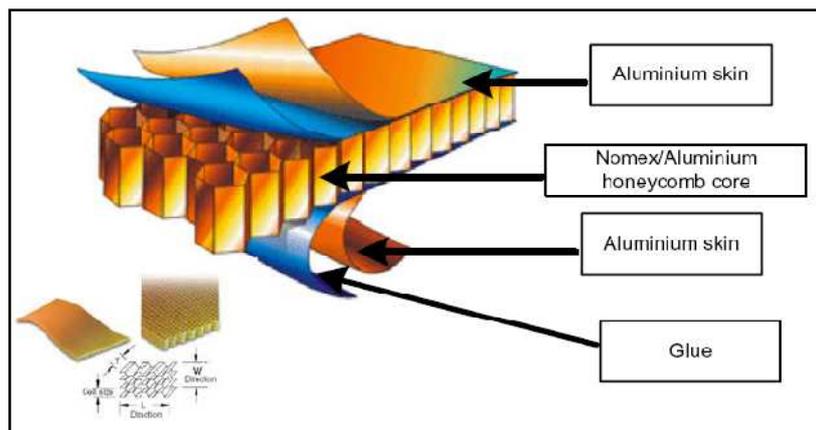


Figure 2-2 Schematic detailed example of the honeycomb sandwich structure

² **Nomex** is a registered trademark for flame resistant meta-aramid material developed in the early 1960s by DuPont.

All these features are due to the honeycomb particular cellular microstructure basically composed of a network of joined, parallel thin-walled hexagonal tubes [2]. Since it resembles the bee's honeycomb found in nature, it gets its name. Most honeycomb materials have cells which are hexagonal and we shall focus our attention on them, but triangular and rectangular cells are possible, too.

As shown below, the honeycomb hexagonal geometry can be characterized by the cell wall thickness (t), cell width (b), minor diameter of the cell (D) and height of the cells ($2H$).

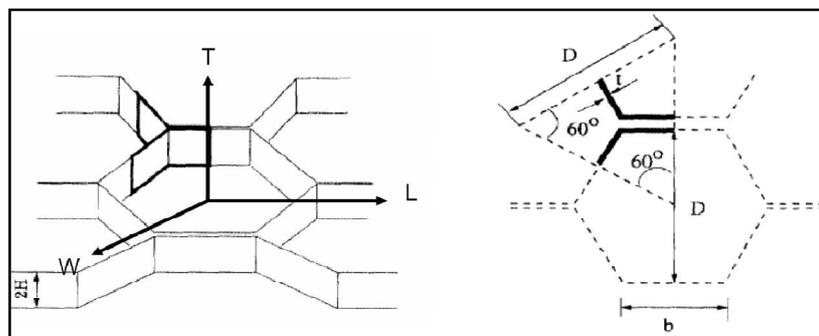


Figure 2-3 Honeycomb cell structure

The hexagonal cell arrangement leads to highly *orthotropic* material properties through three main directions commonly labeled:

- **T**, thickness or tubular direction;
- **L**, longitudinal direction (parallel to core ribbons);
- **W**, width or transverse direction (perpendicular to core ribbons).

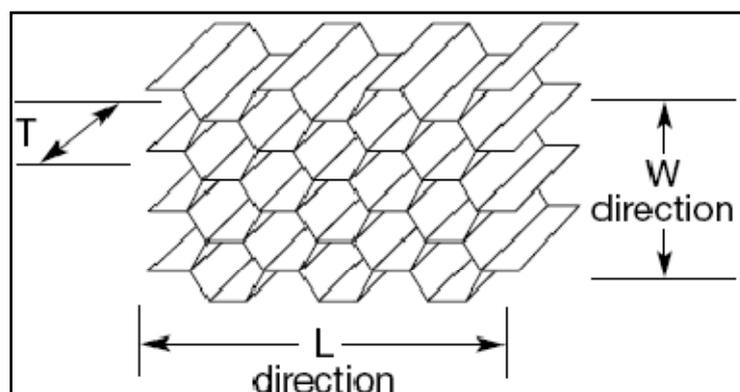


Figure 2-4 Micro structural geometry and his main directions

By varying the core and the thickness and material of the face sheet, it is possible to obtain various properties and desired performance, particularly high strength-to-weight ratio.

2.2.2 Application fields

The industrial interesting on sandwich structure is rapidly increasing, since it involves a wide application fields like satellites, aircraft, ships, automobiles, rail cars, wind energy systems, and bridge construction to mention only a few. Moreover, the development of new materials and the need for high performance and low-weight structures insure that sandwich construction will continue to be in demand. They can be easily molded from composite structures to perform the aerodynamic profile demanded by modern high performance applications.

Between a wide range of available materials, the interesting on *honeycomb core* for sandwich structure is strongly increasing since they are able to replace traditional materials in highly loaded applications. Thanks to their optimum strength-to weight characteristics, the honeycomb core composites offer engineers attractive alternatives for different applications.

Recently the applications of these sandwich materials composite in the civil engineering are increasing. In fact, they are primarily specified to civil applications, because they can be used to produce cost-effective, lightweight components of relatively complex geometries. The most appreciate applications in civil field are:

- exterior architectural curtain wall panels;
- heating, ventilation, air conditioning equipment and devices;
- acoustic attenuation;
- clean room panels;

- energy absorption protective structures.

The unique properties of honeycombs core, arising from their cellular structures, can be exploited in engineering design. Because of their low compressive strength and high deformation capacity, they are outstanding *energy absorption*; this property is exploited in packaging and protective padding of all sorts. Their low density, furthermore, makes them ideal core materials for *light-weight structural* sandwich panels used in modern structural and aerospace component. The small cell size and low volume fraction of solids in closed-cell foams make them *excellent thermal insulators* for applications ranging from coffee cups to building panels.

2.2.3 Manufacturing process

The honeycomb structures can be manufactured by two different methods:

- a) *expansion methods*;
- b) *corrugated process*;

The honeycomb fabrication process by the *expansion method* begins with the stacking of sheets of the substrate material on which adhesive node lines have been printed. The adhesive lines are then cured to form a HOBE (*Honeycomb Before Expansion*) block.

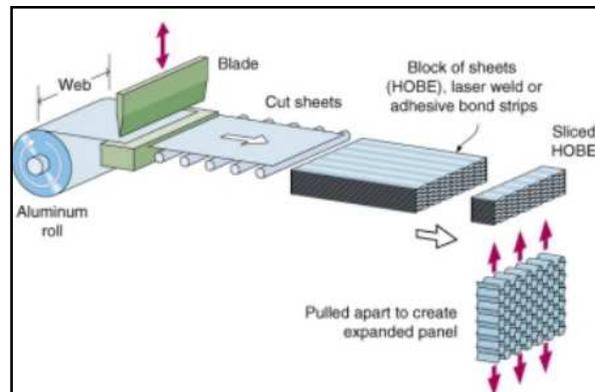


Figure 2-5 Expansion Process of Honeycomb Manufacture

The HOBE block may be expanded after curing to give an expanded block. Slices of the expanded block may then be cut to the desired T dimension. Alternately, HOBE slices can be cut from the HOBE block to the appropriate T dimension and subsequently expanded. Slices can be expanded to regular hexagons, under expanded to 6-sided diamonds, and over expanded to nearly rectangular cells. The expanded sheets are trimmed to the desired L dimension (ribbon direction) and W dimension (transverse to the ribbon).

The *corrugated process* of honeycomb manufacture is normally used to produce products in the higher density range. In this process adhesive is applied to the corrugated nodes, the corrugated sheets are stacked into blocks, the node adhesive cured, and sheets are cut from these blocks to the required core thickness.

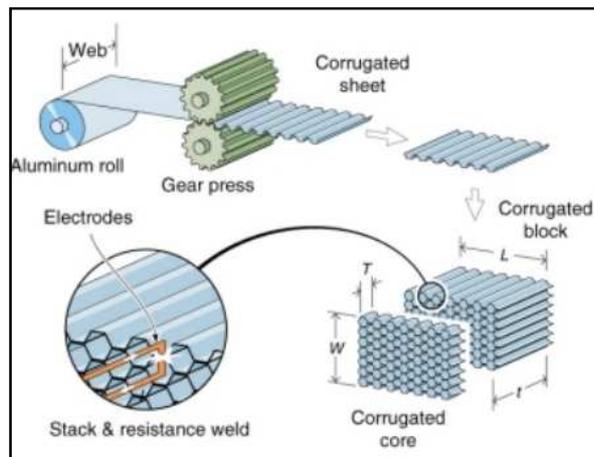


Figure 2-6 Corrugated Process of Honeycomb Manufacture

2.2.4 Honeycomb classification

The honeycomb panels are often available in a wide range of materials and cell configurations, and additional products are continually developed in response to new uses for honeycomb sandwich structures. During the initial design phase, the user must choose the most suitable honeycomb configuration to his designing purpose. There are quite a lot of useful attributes that could help the user choice. Obviously the most important requirements are related to strength and geometrical features, like the:

- *material used;*
- *strength* (compressive, impact, shear, fatigue, etc);
- *cell configuration;*
- *cell size;*
- *alloy and foil gauge (aluminum honeycomb only);*
- *density.*

However there are a lot of other features, that the user should remind, strongly connected to the customer requirements, such as the flammability retardance, the color, the thermal conductivity or the wall surface smoothness.

Through all these parameter it's possible to classify all the honeycomb panels. This *classification* is based on their material and geometry properties.

Depending on the materials used, the honeycomb core could be briefly classified on:

- Aluminum Honeycomb
- Aramid Fiber Honeycomb (Nomex®/Kevlar)
- Fiberglass
- Polyurethane
- Carbon

Honeycomb sandwich panel structures

Each of the honeycomb materials profiled above has specific behavior that is key to its specification. In general terms, some of the most beneficial properties could be summarized in the Table 2.1.

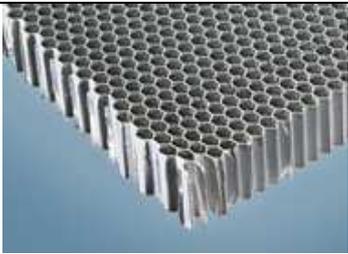
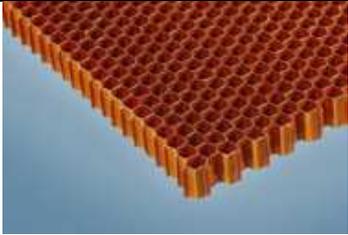
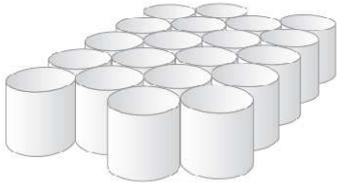
Materials	Features	
<i>Aluminum Honeycomb</i>	Relatively low cost; Best for energy absorption; Greatest strength/weight; Thinnest cell walls; Smooth cell walls; Conductive heat transfer; Electrical shielding;	
<i>Aramid Fiber Honeycomb</i>	Flammability/fire retardance; Large selection of cell sizes, Densities, and strengths; Formability and parts-making experience;	
<i>Fiberglass</i>	Multidimensional Strength of a Woven Structure; Heat formability; Isolative; Low dielectric properties;	
<i>Polyurethane</i>	Cushioning; Unaffected by moisture; Energy redirection; Fatigue-Resistant; Color Choices;	
<i>Carbon Fiber</i>	Based on a need in the space community; Dimensional stability; Performance at high temperatures; Relatively high shear modulus;	

Table 2-1 Most beneficial properties of each honeycomb materials

The main cell configurations are:

- a) *Hexagonal Core*, the standard hexagonal honeycomb is the basic and most common cellular honeycomb configuration, and is currently available in all metallic and nonmetallic materials;
- b) *OX-Core* is a hexagonal honeycomb that has been over-expanded in the W direction, providing a rectangular cell configuration that facilitates curving or forming in the L direction;
- c) *Reinforced Hexagonal Core*, reinforced honeycomb has a sheet of substrate material placed along the nodes in the ribbon direction to increase the mechanical properties;
- d) *Flex-Core*, which provides for exceptional formability in compound curvatures;
- e) *Double-Flex* is a unique large cell Flex-Core for excellent formability and high specific compression properties;
- f) *Tube-Core* is constructed of alternate sheets of flat aluminum foil and corrugated aluminum foil wrapped around a mandrel and adhesively bonded.

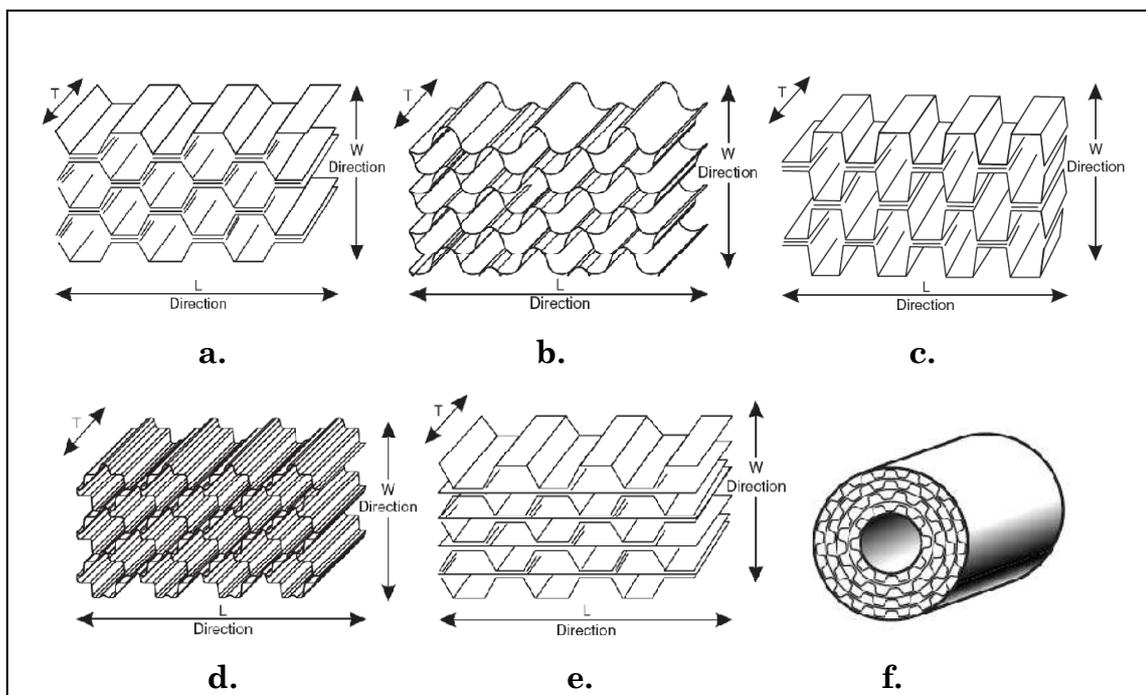


Figure 2-7 cell configuration

2.3 Macro-Scale mechanical properties

From a design or application prospective, the properties of honeycomb can be simplified in such a way as to assume a simple block of homogenous material with orthotropic properties. *Orthotropy* is an important concept because honeycomb cores are highly orthotropic materials. Therefore a brief explanation on these topics is provided.

2.3.1 Orthotropy definition

The generalized Hooke's law that relates the stresses to strains can be written as [3]:

$$\sigma_i = C_{ij} \cdot \varepsilon_j, \quad i, j = 1, \dots, 6$$

C_{ij} is the stiffness matrix and it has 36 constants. However, it can be easily shown that less than 36 of the constants are independent for elastic materials. In fact, the elastic stiffness matrix is symmetric and it has only 21 independent constants.

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_{23} \\ \tau_{31} \\ \tau_{12} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{12} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{13} & C_{23} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{14} & C_{24} & C_{34} & C_{44} & C_{45} & C_{46} \\ C_{15} & C_{25} & C_{35} & C_{45} & C_{55} & C_{56} \\ C_{16} & C_{26} & C_{36} & C_{46} & C_{56} & C_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_{23} \\ \gamma_{31} \\ \gamma_{12} \end{Bmatrix}$$

This is the most general expression for linear elasticity. An anisotropic material is the one that exhibits material properties that are directionally independent, i.e., a given material property can have different values in different directions.

Unlike anisotropic materials, orthotropic materials show symmetric material properties. If there are two orthogonal planes of material property symmetry for a material, symmetry will exist relative to a third mutually orthogonal plane. This defines an orthotropic material. The independent elastic coefficients reduce to 9 for an orthotropic material. The stress-strain relations for an orthotropic material are given by:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_{23} \\ \tau_{31} \\ \tau_{12} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{13} & C_{23} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_{23} \\ \gamma_{31} \\ \gamma_{12} \end{Bmatrix}$$

2.3.2 Honeycomb orthotropy

As posted before, the honeycomb microstructure has a strong orthotropic behavior, and his initial orthotropy axes are usually called **W**, **L** and **T**, where the T-direction is aligned with the axis of the thin-walled tubes, while the W and L denotes the in-plane directions [5].

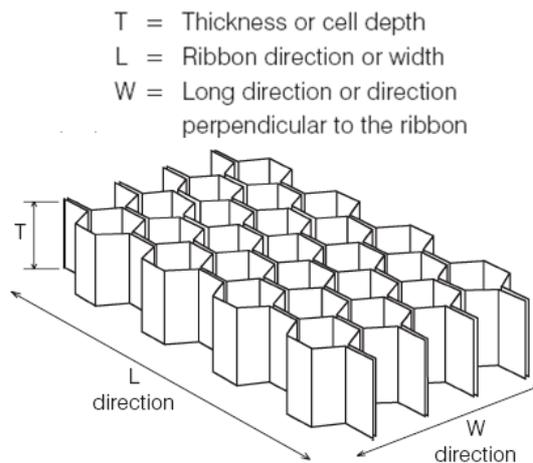


Figure 2-8 Main directions characterizing the geometry of honeycomb structures

The properties of these materials in published literature, for example the manufacturer's datasheets from *HEXCEL*³, use these directions as a reference frame.

The properties in the 'W' and 'L' directions are described as in-plane properties, whilst those in the 'T' direction are out-of-plane.

The consecutive stress-strain orthotropic relationship for honeycomb cores is shown as follow:

$$\begin{Bmatrix} \varepsilon_I \\ \varepsilon_W \\ \varepsilon_L \\ \gamma_{WL} \\ \gamma_{TL} \\ \gamma_{TW} \end{Bmatrix} = \begin{bmatrix} 1/E_I & -\nu_{WI}/E_W & -\nu_{LI}/E_L & 0 & 0 & 0 \\ -\nu_{TW}/E_T & 1/E_W & -\nu_{LW}/E_L & 0 & 0 & 0 \\ \nu_{TL}/E_T & \nu_{WL}/E_W & 1/E_L & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/G_{WL} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/G_{TL} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/G_{TW} \end{bmatrix} \begin{Bmatrix} \sigma_I \\ \sigma_W \\ \sigma_L \\ \sigma_{WL} \\ \sigma_{TL} \\ \sigma_{TW} \end{Bmatrix}$$

In order to determinate or estimate the values of the main mechanical properties needed to define the stiffness matrix C_{ij} , the designer can follow two different ways:

- *Static test methods;*
- *Analytic models.*

Experimental and analytic studies have shown that traditionally the properties in the 'T' direction are higher than those in the 'W' and 'L' directions. The definitions of in-plane and out-of-plane normal and shear loading conditions are presented in the next figure.

Both methods will be described in the following chapters.

³ *Hexcel* is a materials company that manufactures advanced composite materials and structural parts. Hexcel's primary markets are aerospace, defense, wind energy and high performance industrial markets including automotive, marine and recreation.

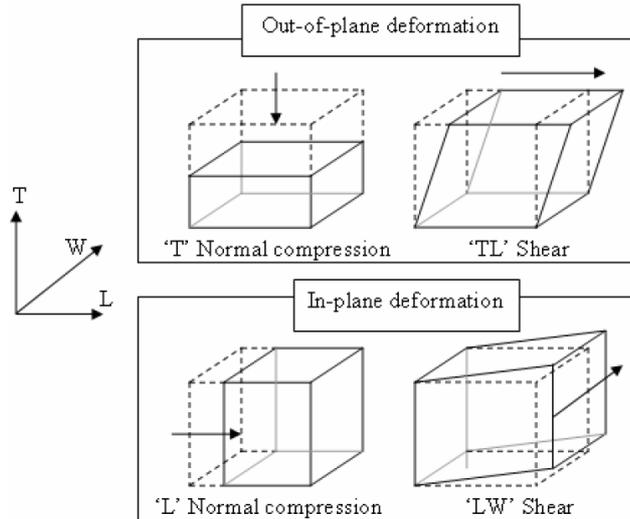


Figure 2-9 Definition of in- and out-of-plane loading

2.3.3 Static Test methods

The intrinsic complexity of the multilayered configurations and the interaction of failure modes make difficult an accurate analysis of honeycomb panels' mechanical properties. The general behavior of these structures depends on the material properties of the constituents (skins, core, and adhesive if involved), geometric dimensions, and type of loading. Moreover, the experimental methods for composite materials are more complex than for isotropic materials and require significant modifications [6]. Usually the producers' data sheets offer to the user all the main mechanical properties related to a particular honeycomb panels. These properties are the results of several test methods.

Such standard tests are executed in *static conditions*, and all the dynamic tests are not used in order to define the mechanical constants.

The testing methods to determine out-of-plane normal and shear properties of the honeycomb, with respect to the principal directions, have been standardized.

ASTM C365-03 refers to the testing method required to determine ‘T’ direction crushing properties [14]. The experimental layout, shown as follow, compresses a small sample loaded at a constant rate of 0.5mm/min and uses a mechanical displacement measuring device to determine relative displacement of the upper and lower faces.

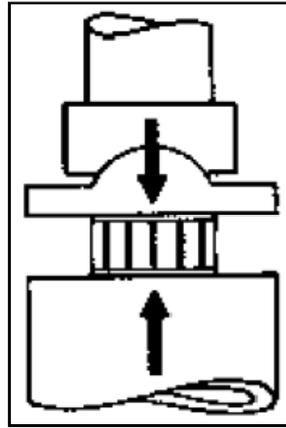


Figure 2-10 Schematic representation of Compressive Test [24]

The standard also specifies the size of the test sample depending on the diameter of the hexagonal cell [24].

The ‘T’ direction compression strength and strain for the elastic and plastic crushing domains are then established using the following equation:

$$\sigma = \frac{P}{A}$$

where P is the applied force and A is the cross-Sectional area.

When loaded in the ‘T’ direction, a general force-displacement trend is followed as shown in the following figure.

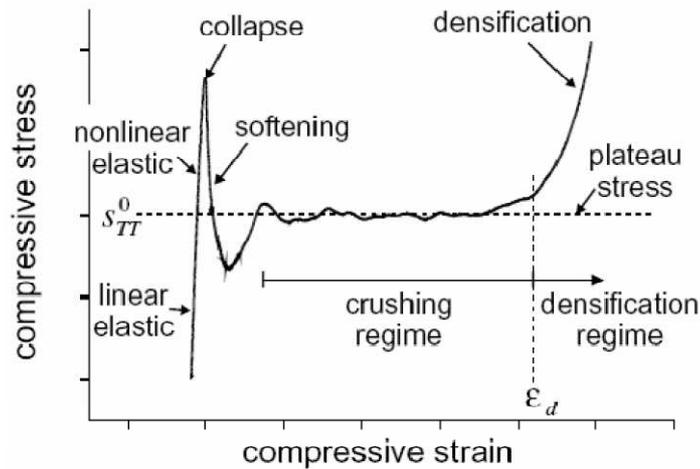


Figure 2-11 “T” direction compressive load profile [5]

This curve has an initial linear elastic phase to a peak load at which point buckling of the cell walls is initiated; thereafter, the material crushes at a lower, approximately constant, plateau load, up to a state of full compaction when the load capacity rapidly rises.

The shear properties of honeycomb are established using ASTM 273-00 [21]. There are two approaches to determine shear properties of honeycomb samples, namely, compression and tension. In each test case, the sample is bonded to the loading blocks which have the ability to move and rotate relative to each other. This creates the condition of pure shear as the thickness of the sample is allowed to change during the test. Kelsey et al. [22] have identified a number of potential difficulties with honeycomb shear testing, such as glue fillet influence on the depth of the specimen, which can influence the shear properties of the sample.

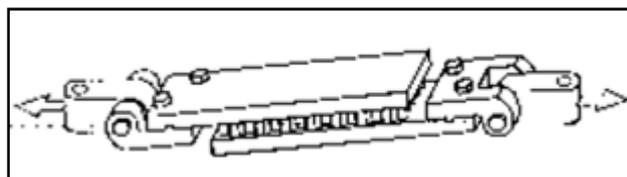


Figure 2-12 Schematic example of tensile plate shear methods [24]

During the tensile plate shear method, a lot of potential difficulties can occur such as glue fillet influence on the depth of the specimen. For this reasons the out-of-plane shear and compression test could represent an interesting and favorable test method. In particular, Mohr and Doyoyo have investigated the effects of varying loading direction with respect to the out-of-plane shear and normal properties of honeycomb. The investigation specified design modifications to the standard *Arcan* apparatus to examine the effects of multi-directional loading on cellular solids. These improvements include:

- The circular Sections must be clamped instead of pinned. This restricts the rotation of the circular Sections of the apparatus, which prevents localized buckling of the honeycomb cells.
- The clamps should be fixed in such a way that they do not move laterally relative to each other.
- A method to measure the vertical and horizontal load components is necessary.

Using these modifications, Mohr and Doyoyo [5] produced the test apparatus shown in the following figure.

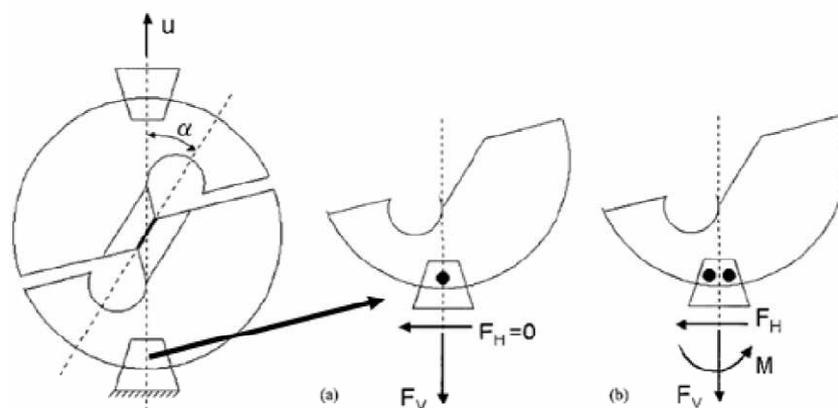


Figure 2-13 Arcan apparatus pinned (left side) and clamped (right side) configuration [5]

2.3.4 Analytical models

The properties described in the previous Section are a consequence of folding and collapse processes taking place in the cell walls. Numerous analytical models have been developed to predict the overall properties of honeycomb based on the geometry of the cells and properties of the base material. In researched studies and literature, the analysis of honeycomb is often separated into two groups, the mechanics of in-plane and out-of-plane deformation.

In-Plane Properties

Ashby and Gibson [25] and Zhang and Gibson [26] have established predictive methods to determine in-plane properties. This work reduced the complexity of the honeycomb cells to a single wall and resolving the forces and moments, so that in-plane properties can be predicted.

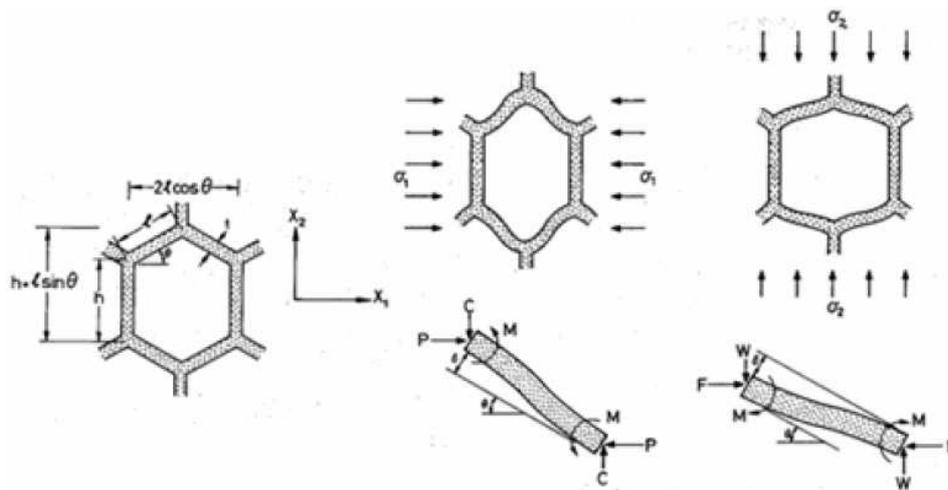


Figure 2-14 Elastic deformation on W and L directions [Gibson L.J. and Ashby M.F. Cellular Solids [25]]

Out-of-Plane Properties

The mechanics of deformation in the ‘T’ direction, or out-of-plane direction, are based on the mechanics of folding walls. Figure 2-15 shows the regular folding pattern of a honeycomb cell under out-of-plane loading.

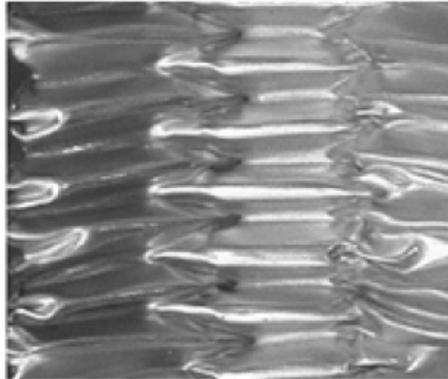


Figure 2-15 Deformed honeycomb due to out-of-plane compression loading

Early research studies to determine the folding and energy absorbing mechanisms similar to those in honeycomb cells were confined to thin-walled cylinders; one such example was conducted by Alexander [27] producing the model shown in the following figure.

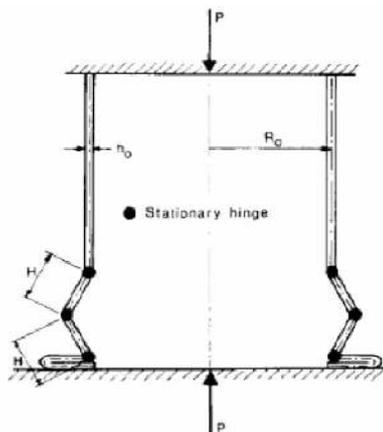


Figure 2-16 Alexander's model: Folding of thin walls in a cylinder

The Alexander model mathematically describes the crushing process for cylindrical structures; however, it did contain unrealistic

assumptions concerning the folding profile and under-predicts the crushing strength of the structure.

In a later work, McFarland [28] presented a predictive method to determine the crushing strength in the out-of-plane direction based on a hexagonal cell structure. McFarland idealized the crushing mode to determine the mechanisms of collapse. The collapse profile, as shown in the following figure, was used to relate the energy involved in the bending deformation to the mean crushing strength. This mechanism was simplified to introduce the effect of in-plane shearing during the ‘T’ direction crushing process.

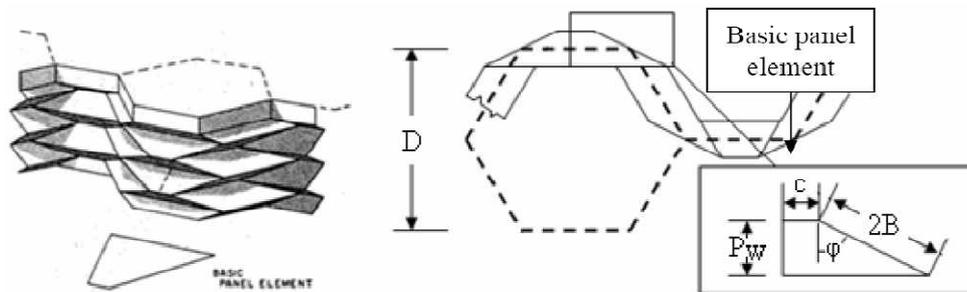


Figure 2-17 McFarland model: ‘T’ direction crushing mechanism

2.4 Experimental investigation

The experimental investigation presented in this Section has been executed by Eng. A. Zinno, D. Asprone in the Department of Structural Engineering, University of Naples “Federico II” [4]. This experimental investigations addressed the compression dynamic behavior and the impact response of the sandwich panels, obtained combining phenolic resin-based glass fiber reinforced plastics and phenolic resin-impregnated aramid paper honeycomb structure (Nomex®).

This kind of investigation was needed to define a phenomenological model for honeycomb core materials, presented in chapter 4.

2.4.1 Introduction

In Section 2.3.3 have been presented, in a generic way, the main test methods usually conducted on sandwich honeycomb structures; a wide range of possible results have been illustrated depending of the geometry and the material used.

The manufacturer data sheets usually point out a lot of information about the main mechanical properties. However, when the designer is interested in a particular mechanical property, not included in the data sheet, the designer is often forced to depend on his own tests. In particular, all the properties included in producer’s data sheet are usually obtained by using only *static test methods*, leaving out the dynamic behavior of such structures.

In this Section, after a brief description of the *Nomex*® sandwich specimen features, the main outcomes of some non-standard experiment are presented.

In particularly, the experimental investigation is addressed to the compression dynamic behavior and the impact response of the sandwich panels, obtained combining phenolic resin-based glass fiber reinforced plastics and phenolic resin-impregnated aramid paper honeycomb structure (*Nomex*®). However, a static analysis has been conducted too.

2.4.2 Material specimens

The *Nomex*® core is a 48 kg/m³ hexagonal honeycomb structure with a nominal cell size of 3.18 mm and made of phenolic resin-impregnated aramid paper. In the following data sheet paper are illustrated the main proprieties provided by manufacturer’s datasheet. All these values are the results of the typical static test methods, illustrated in the 2.3 Section.

HRH-10 Aramid Fiber/Phenolic Resin Honeycomb

Hexcel Honeycomb Designation Material - Cell Size - Density	Compressive					Plate Shear					
	Bare		Stabilized			L Direction			W Direction		
	Strength psi		Strength psi		Modulus ksi	Strength psi		Modulus ksi	Strength psi		Modulus ksi
Hexagonal	typ	min	typ	min	typ	typ	min	typ	typ	min	typ
HRH-10 – 1/16 – 3.4	195	160	205	170	20	155	125	6.0	85	65	2.9
HRH-10 – 1/8 – 1.8	105	85	115	95	8	90	75	3.8	50	40	1.5
HRH-10 – 1/8 – 3.0	290	235	325	270	20	175	155	6.5	100	85	3.5
HRH-10 – 1/8 – 4.0	520	400	575	470	28	255	225	8.6	140	115	4.7
HRH-10 – 1/8 – 5.0	700	560	770	620	37	325	275	10.2	175	150	5.4
HRH-10 – 1/8 – 6.0	1050	850	1125	925	60	385	330	13.0	200	170	6.5

Figure 2-18 Specimens manufacturer data-sheet

The skins consist of satin-weave fabric E-glass fiber reinforced phenolic resin prepregs with a cured ply thickness of 0.25 mm. The materials were laminated in an autoclave at a temperature of 135°C, a vacuum pressure of 2.5 bar and a curing time of 90 min.

2.4.3 Static Tests

Flat wise stabilized compressive tests

Out-of-plane crushing behavior of *Nomex*® honeycomb has been investigated by **flat-wise** stabilized compressive tests according to ASTM C365-05 standard. The tests were run on five 60x60x32.2mm coupons bonded between two 1 mm of phenolic skins on a MTS universal testing facility with a constant cross head velocity of 0.5 mm/min. The stress-strain relationship, like still discussed in Section 2.3, consists of three main areas:

- the ***elastic regime*** up to the stabilized compressive strength;
- the ***crushing regime*** at nearly constant plateau stress (crush strength);
- the ***densification regime***, where the cellular structure is fully compacted resulting in a steep stress increase.

Figure 2-19 shows, on the left-side the experimental results of static flat-wise stabilized compression test. On the right are presented the three main deformation stages: initial (top), crushing regime (middle) and densification (bottom).

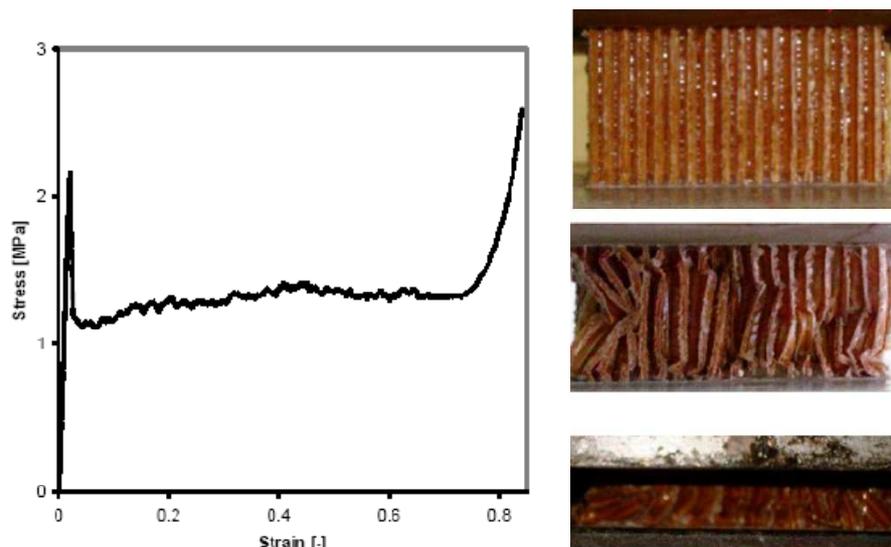


Figure 2-19 Experimental results of static flat-wise stabilized compression test.

Each of these three regimes is characterized by a particular internal deformation mechanism [10]. When the honeycomb structures are loaded at low strain rates it's possible to observe that the linear elastic behavior is due to the bending of the cell walls, while the stress plateau to the elastic buckling, the plastic yielding or brittle fracture all depend on the nature of the cell wall material. The last densification is due to the meeting of opposing cell walls once the cells have completely collapsed.

Indentation tests

In order to investigate the crushing behavior of sandwich structures exposed to localized point load, **indentation tests** were performed on 50*250mm sandwich specimens manufactured by laminating 32.2 mm of Nomex® core between two glass/phenolic skins with a thickness of 1 mm.

The sandwich beams were supported by a steel substrate, thus the overall bending on the specimen was avoided. The tests were carried out in a MTS universal testing machine under displacement control at a loading rate of 2 mm/min. The indentation load was applied through a steel cylinder (20 mm in diameter) across the whole width of the beam cross-Section as shown in Figure 2-20.

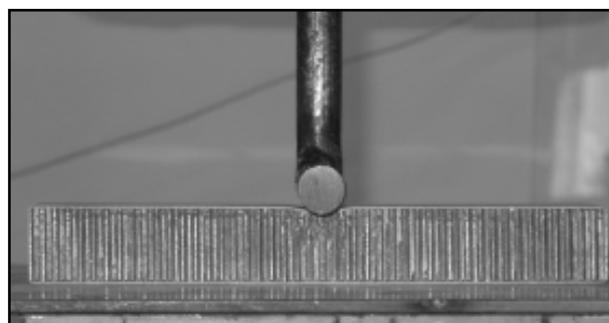


Figure 2-20 Static indentation test of sandwich beam

The test was conducted on three replicate specimens for each of five different displacement levels. Afterwards the load was released at a cross-head speed of 20 mm/min. During the unloading the face sheet flexed back but did not recover completely its undeformed shape. Thus, a residual face sheet dent remained as shown in Figure 2.21.

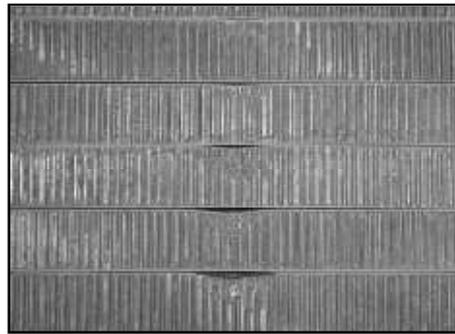


Figure 2-21 residual dent at different displacement levels during static indentation tests

During the indentation test, the load indentation curve was recorded for both loading and unloading steps. A typical load-indentation curve for a Nomex® honeycomb sandwich specimen is shown in the left side of the Figure 2-22. The curve showed a *linear* behavior under the peak load. It was believed that the emission of a noise (cracking sound) at the end of the linear domain should be associated with the onset of core crushing. After that the curve became *nonlinear* with a decrease in the stiffness.

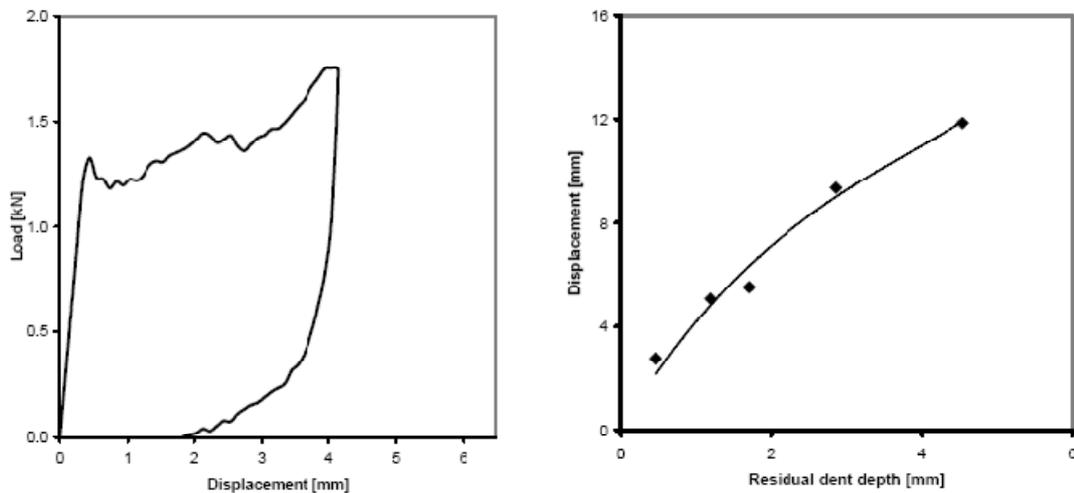


Figure 2-22 Experimental results of indentation test: (left) force-displacement curves; (right) residual dent depth at different imposed displacement values.

The nonlinear behavior was due to the progressive honeycomb crushing in the area under the indenter. On the right side of Figure 2-22 it's shown the residual dent magnitude, equal to the displacement when the load dropped to zero, as function of the imposed indentation.

2.4.4 Dynamic tests: Drop weight test

A drop weight tower was used for dynamic testing at different loading rates for both compressive tests on honeycomb specimens and impact tests on sandwich specimens.

A laser measurement system provides displacement measurements for drop weight tests. Out-of-plane compressive tests were conducted on the drop tower facility at three different strain rates (60 s^{-1} , 120 s^{-1} and 200 s^{-1}) on cylindrical specimens of 45 mm diameter and 32.2 mm and 10.5 mm of thickness. Different thicknesses were considered to achieve higher strain rates.

In order to prevent local crushing at the edge of the honeycomb structure the core was bonded to 1 mm glass/phenolic skins. A fourth order electronic *butterworth* low-pass filter was used in order to filter out superposed high frequency oscillations associated with dynamic loads, as illustrated if the left side of Figure 2-23. In this way, comparability of dynamic and static test data have been achieved. In the right side of Figure 2-23, is shown the stress-strain diagrams at the investigated strain rates. Each shown curve represents the mean data of five replicate specimens for each specimen family.

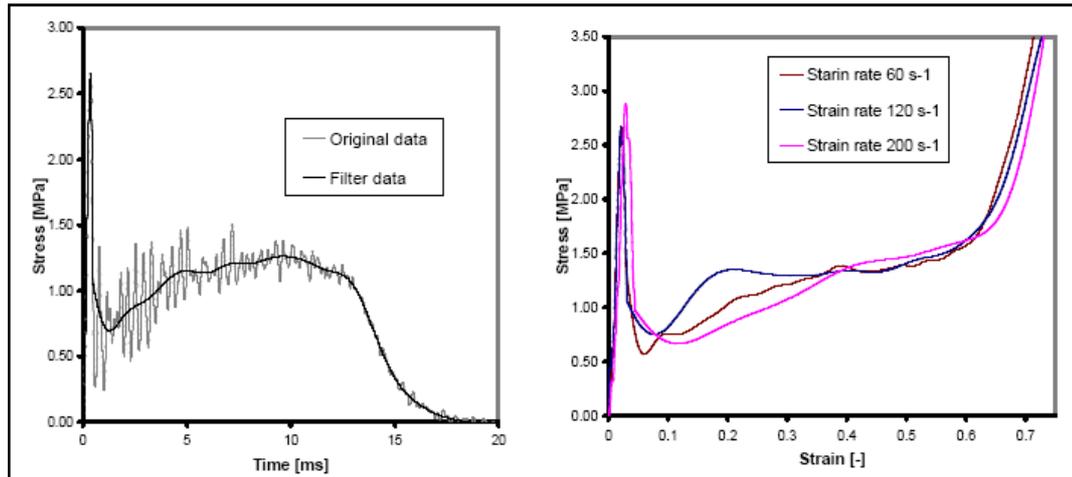


Figure 2-23 Dynamic compression tests on honeycomb structures: filtering data (left side); stress-strain curves at different strain rates (right side).

It can be seen that dynamic loading leads to a significant increase of both compressive and crush strength; in particular the compressive strength presents a DIF (Dynamic Increase Factor, ratio of the dynamic value over the static one) of 1.20 (at 200s-1), whereas the crush strength presents a DIF of 1.10 (at 200s-1). Influence of dynamical loading on the densification point has been also observed: the dynamic strain value is about 10% lower than quasi-static one. On the contrary no influence on the initial stiffness has been observed.

For aluminum honeycomb structures the DIF for crush strength was observed to be about 33% (gas gun test, 100s-1), 40% (split Hopkinson pressure bar, 800 s-1) and 50% (gas gun test, 2000 s-1) above the quasi-static value. For Nomex® honeycomb structures the increase of plateau stress is about 10-30% in the strain rate domain from 50s-1 to 300 s-1.

2.4.5 Conclusions

In the presented paper the results of an experimental program characterizing the dynamic behavior of sandwich phenolic structures are presented. In particular, compression and impact tests have been analyzed. The following main outcomes have been obtained:

- the *Nomex*® honeycomb structure presents a strain rate sensitive compressive behavior; in particular, at a strain rate of about 300 s⁻¹, the compressive strength presented a DIF of 1.20, whereas the crush strength presented a DIF of 1.10.
- the comparison between the static indentation tests and dynamic impact tests reveals that in the quasi-static regime the absorbed energy and the damage mechanisms are governed by the core behavior, whereas in the dynamic regime the global behavior is due to the contribution of the skins.
- as it was expected, in impact tests, in case of thicker skins delamination occurs at the bottom skin.

In future work, more dynamic tests could be conducted to validate the preliminary outcomes on the dynamic characterization of the investigated sandwich structures and to better understand the role of core material and the possible interaction between different modes of absorption energy at higher strain rate levels. Based on the experimental results, dynamic constitutive laws will be calibrated in order to conduct reliable finite element simulations and investigate the main mechanisms and parameters involved in the dynamic behavior. The long-range objective is to obtain a reliable numerical model to be employed in more complex dynamic virtual tests.

2.5 Literature survey on virtual testing of sandwich core structures

Despite of the increasing use in several industrial and civil applications, the honeycomb sandwich materials are not still extensively investigated on how material properties and geometrical configuration can affect the hypohetic damage. Usually, the manufacturer data sheets don't contain all the information on the material properties needed by the designer. For this reason the user is often forced to depend on testing.

In order to predict the whole mechanical behavior of these complex structures, avoiding expensive and time-consuming laboratory tests, the importance of FEM analysis is increasing. Moreover such virtual tests are useful if analytical models need to be validated when experimental data are not available for validation.

The idea of performing virtual tests on honeycomb core structures with detailed FE simulation models is not new and first came up in the end of the 1980s [11]. The reason was to obtain the in-plane elastic properties of aluminum honeycomb cores, which are not provided in honeycomb data-sheets, since those values are very low compared to the out-of-plane properties and often they are neglected.

A lot of FEM environment software offer a lot of pre-arranged elements, which could be used to facilitate the modeling of such structures, for example NASTRAN, ADINA, PAM-CRASH, ABAQUS, LS-DYNA, RADIOSS, ect.

Such software have been largely used to model the behavior of sandwich core structure, in order to calculate:

- the effective elastic properties;
- the cell wall folding mechanisms in the post-damage region;
- the influence of geometric and material imperfections;

In the most of cases, the honeycomb internal structure has been modeled representing the full geometry of the honeycomb cells, trying to represent the buckling effects of cells walls. Despite the accuracy of the results, this method involves a great loss of time in the process of geometric modeling. In addition, this approach doesn't represent the most versatile way to model such materials.

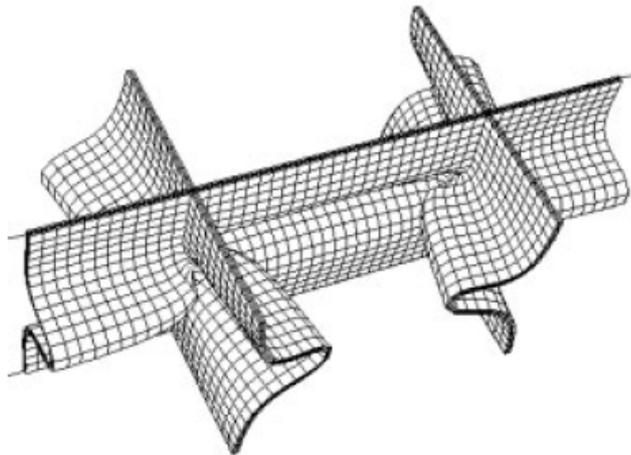


Figure 2-24 Time-consuming approach: full geometrical representation of honeycomb core [Zhenyu Xue and John W. Hutchinson, *Constitutive model for quasi-static deformation of metallic sandwich cores*]

3. AceGen and AceFem

In the present research work, a couple of innovative *Mathematica*⁴ packages, called *AceGen* and *AceFem*, have been used in order to modeling the mechanical behavior of honeycomb sandwich structures. In this chapter is presented a wide report over such packages, pointing out all their main innovative features. These FEM software packages, running in parallel with *Mathematica*, provide a robust and optimized architecture for rapid numerical prototyping and finite element analysis.

⁴ *Mathematica* is a well known Computer Algebra (CA) system largely used in several technical computing like scientific, engineering and other mathematical fields. *Mathematica* is one of the most powerful academic software for the manipulation of formulae and for performing several mathematical operations by computer. *Mathematica* (www.wolfram.com)

3.1 Introduction

3.1.1 *AceGen and AceFem overview*

The *AceFem* and *AceGen* packages has been written and developed by Jože Korelc, professor at the University of Ljubljana's Faculty of Civil and Geodetic engineering.

Each package combines use of *Mathematica*'s facilities with external handling of intensive computation by compiled modules.

The task of the *Mathematica* package ***AceGen*** is the automatic and optimized derivation of formulae needed during numerical procedures. The great advantage of this tool, stands on his new and innovative approach which exploits all the symbolic and algebraic capabilities of *Mathematica*, avoiding all the typical troubles of such symbolic software during the analysis of complex mechanical models.

By using an innovative *Symbolic-Numerical approach*, *AceGen* is able to combine different techniques, explained in the following chapters, in order to shorten the evolution route of any computing procedures. Moreover the results could be exported as compiled *FORTRAN* or *C* code with automated interfacing, exploiting all the capabilities of such languages. *AceGen* is set up to talk with other numerical environments, including its sibling *AceFem*.

AceFem is an innovative finite element environment designed to solve multi-physics and multi-field problems. The *AceFem* package explores advantages of symbolic capabilities of *Mathematica* while maintaining numerical efficiency of commercial finite element environments. *AceFem* application could employ specific own elements or general codes previously created by *AceGen*. The *AceFem* package contains a large library of finite elements (solid, thermal, contact... 2D, 3D...) including full symbolic input for most of

the elements. Additional elements can be accessed through the **AceShare** finite element file sharing system.

As presented in the following sections, the *AceFem* package exploits an *element oriented approach* which enables easy creation of customized finite element based applications in *Mathematica*.

The combination of the automatic code generation package *AceGen* and the *AceFem* package represent an ideal tool for a rapid development of new numerical models.

3.1.2 Traditional FEM applications

During the last decades, important improvements have been achieved by using standard features of the currently available finite element (FE) package like *Abaqus*⁵, *Feap*, etc.

Numerical simulations are well established in several engineering fields, and all of them use several mathematical models described by a system of differential equations. Most of existing numerical methods for solving partial differential equations could be divided into two main groups: *Finite Difference Method (FDM)* and *Finite Element Method (FEM)*. Unfortunately, the definition of a new finite element method would be time extremely consuming, since a lot of time is spent to derive characteristic quantities like gradients, Jacobean, Hessian, etc.

The use of commercial FEM environment represents a common practice to analyze a great variety of physical problems; however, the use of such large systems is not very awkward for developing and testing new numerical procedures. In fact, during the initial research phases, it's more efficient the use of the symbolic – numerical environments like *Mathematica* or *Maple*. Moreover, the

⁵ **Abaqus** is a suite of software applications for finite element analysis and computer-aided engineering, originally released in 1978.

identification of many design flow (such as poor convergence proprieties) could be easier through the use of symbolic environments.

Despite these advantages, the symbolic level of analysis becomes very inefficiently if iterative procedures have to be performed, or if a large number of elements have to be considered. In order to assess element performances under real conditions the best way is to perform tests on sequential machines with good debugging procedures (programs written in FORTRAN or C/C++).

By the classical approach, re-coding of the element in different languages would be extremely time consuming and for this reason it's never done.

3.1.3 Hybrid Symbolic-Numerical approach

The real power of the symbolic approach for the development, testing and application of new, unconventional ideas is provided by general purpose CA systems, such as *Mathematica* or *Maple*. Unfortunately traditional CA systems are very powerful when dealing with one isolated formula, but become very awkward when the code to be generated contains control structures such as *If* and *Do* constructs. For these reasons, their use is limited for problems which lead to large systems like finite element simulations. Furthermore, the use of large-scale commercial finite element environments for analyzing a variety of problems is an everyday practice of engineers. Although large FE environments often offer a possibility to incorporate user defined elements and material modes, it is time consuming to develop and test these user defined new pieces of software. Practice shows that, at the research stage of the derivation of a new numerical model, different languages and different platforms are the best means for the assessment of specific

performances and, of course, failures of the numerical model. At the end, for real industrial simulations involving complex geometries, a large commercial FE environment has to be used.

In order to avoid the above described troubles, related to the use of CA systems, the *hybrid symbolic-numerical* (HSN) approach, implemented in *AceGen*, is a way to combine different techniques inside one system. In order to meet all these demands in an optimal way, an approach is needed that would offer *multi-language* and *multi-environment* generation of numerical codes. The automatically generated code is then incorporated into the FE environment that is most suitable for the specific step of the research process. Such approach is idea implemented in *AceGen* software.

The structure of the hybrid symbolic-numerical system ***AceGen*** for multi-language and multi-environment code generation introduced by Korelc [15] is presented in Figure 3-1.

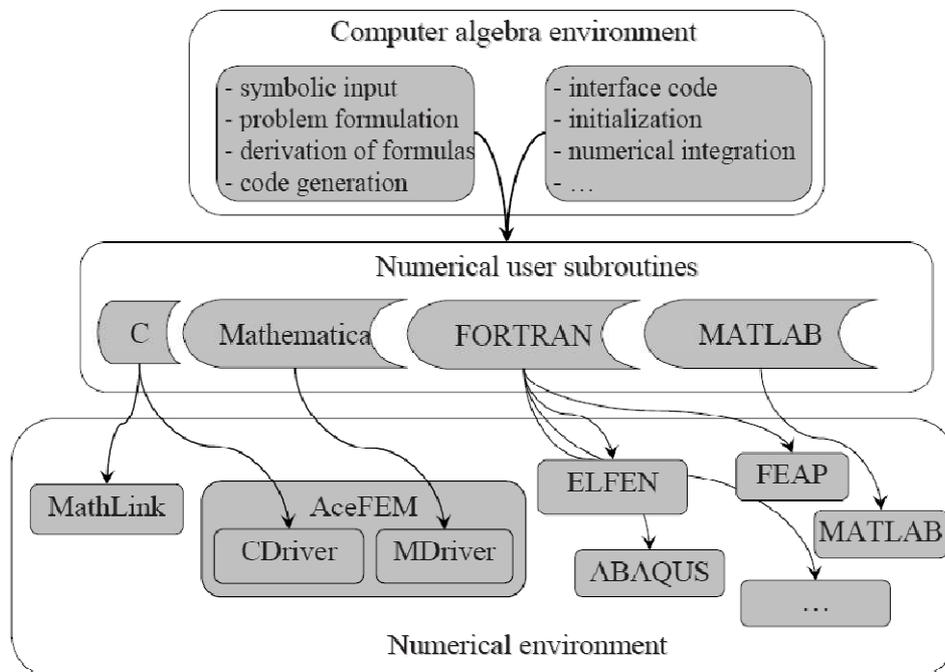


Figure 3-1 Multi-language and multi-environment FE code generation [15]

AceGen enables multi-language and multi-environment generation of non-linear finite element codes from the same symbolic description. For this important feature, the re-code phase comes practically for free. This extremely innovative capability is due to a new hybrid symbolic-numerical approach.

An innovative approach for automatic code generation is employed in *AceGen* and called *Simultaneous Stochastic Simplification* of numerical code, see Korelc [16]. This approach avoids the problem of expression swell by combining the following techniques:

- symbolic and algebraic capabilities of the general computer algebra system *Mathematica*;
- automatic differentiation techniques;
- simultaneous optimization of expressions with automatic selection and introduction of appropriate intermediate variables.

Formulae are optimized, simplified and replaced by the auxiliary variables simultaneously with the derivation of the problem. A stochastic evaluation of the formulae is applied for determining the equivalence of algebraic expressions, see e.g. Gonnet (1986), instead of the conventional pattern matching techniques. The simultaneous approach is appropriate also for problems where intermediate expressions can be subjected to an uncontrolled swell.

3.2 AceGen

3.2.1 Features and advantages

As briefly posted in the previous chapter, *AceGen* package was designed in order to approach especially hard problems, where the general strategy to efficient formulation of numerical procedures, such as the analytical sensitivity analysis of complex multi-field problems, had not yet been established [21].

His main task is to derive automatically formulae required during numerical procedures. In the case of complex numerical models, the direct use of traditional and algebraic programs like *Mathematica* or *Maple* is not possible. Despite of their undoubted capabilities, during the development phase the symbolic derivation of formulae leads to uncontrollable growth of expressions and consequently redundant operations and inefficient programs.

The innovative goal of *AceGen* is to combine different techniques inside one system in order to avoid the above mentioned problem. Thus, in *AceGen* are implemented different techniques in such a way that the implementation of arbitrary numerical procedures is made possible. Moreover, *AceGen* package doesn't combine different system; it combines different techniques inside one system in order to avoid the usual problem related to the use of traditional symbolic and algebraic programs.

In *AceGen* is implemented an automatic code generation called *Simultaneous Stochastic Simplification of numerical code* (Korel [16]). The above mentioned approach combines the general computer algebra system *Mathematica* with an automatic differentiation technique and an automatic theorem proving by examples. This innovative stochastic evaluation of the formulae is used to define the

equivalence of algebraic expression, instead of the conventional pattern matching technique.

Moreover his multi-language capabilities can be exploited for a rapid prototyping of numerical procedures in several script languages like *Mathematica*, *Matlab* as well as to create highly optimized and efficient compiled language codes in *FORTRAN* or *C*.

The most important features of *AceGen* code generator are:

- simultaneous optimization of expressions immediately after they have been derived;
- automatic differentiation technique;
- automatic selection of the appropriate intermediate variables;
- the whole program structure can be generated;
- appropriate for large problems; an intermediate expressions can be subjected to the uncontrolled swell;
- improved optimization procedures with stochastic evaluation of expressions;
- differentiation with respect to indexed variables;
- automatic interface to other numerical environments (by using *Splice* command of *Mathematica*);
- multi-language code generation (*Fortran*/*Fortran90*, *C/C++*, *Mathematica* language, *Matlab* language);
- advanced user interface;
- advanced methods for exploring and debugging of generated formulae;
- special procedures are needed for non-local operations.

3.2.2 Mathematica–AceGen combination

The *AceGen* application is written in the symbolic language of *Mathematica* and consists of about 300 functions and 20000 lines of *Mathematica*'s source code.

Typical *AceGen* function takes the expression provided by the user, either interactively in file, and returns an optimized version of the expression. Optimized version of the expression can result in a newly created auxiliary symbol, or in an original expression in parts replaced by previously created auxiliary symbols. In the first case *AceGen* stores the new expression in an internal data base. The data base contains a global vector of all expressions, information about dependencies of the symbols, labels and names of the symbols, partial derivatives, etc. The data base is a global object which maintains information during the *Mathematica* session.

AceGen exploits an innovative way of optimizing expressions in computer algebra systems, since formulae are optimized, simplified and replaced by the *auxiliary variables* simultaneously with the derivation of the problem. The optimized version is then used in further operations. If the optimization is performed simultaneously, the explicit form of the expression is obviously lost, since some parts are replaced by intermediate variables.

Since *AceGen* runs in parallel with *Mathematica* we can use all the capabilities of *Mathematica*. The major algebraic computations can be directly implemented also with the built-in *Mathematica* functions and the result optimized by *AceGen*:

- analytical differentiation;
- symbolic evaluation;
- symbolic solution to the system of linear equations;
- symbolic integration;
- symbolic solution to the system of algebraic equations.

3.2.3 Symbolic-Numeric Interface: Auxiliary variables

AceGen system is able to pass data from the main program into the automatically generated routine (and to get the results back to the main program) through the use of *external variables* (or *auxiliary variables*). So, they are used to establish the interface between the numerical environment and the automatically generated code.

All the external variables appear in a list of input/output parameters of the subroutine's declaration (as a part of expression) and when the values are assigned to the output parameters of the subroutines.

The definition of the input/output parameters is made possible through the *SMSModule* function.

The form of the external variables is prescribed and is characterized by the “\$” signs at the end of its name. The standard *AceGen* form is automatically transformed into the chosen language when the code is generated. The standard formats for external variables when they appear as part of subroutine declaration and their transformation into FORTRAN and C language declarations are as follows:

Type	<i>AceGen</i> definition
real variable	x\$\$, x\$\$\$\$
real array	x\$\$[10], x\$\$[i\$\$,"*"]
integer variable	i\$\$, i\$\$\$\$
integer array	i\$\$[10], i\$\$[i\$\$,"*"]
logical variable	l\$\$[10]

Arrays can have arbitrary number of dimensions, and the dimension can be an integer constant, an integer external variable or a "*" character constant if the dimension is unknown.

The standard formats for external variables when they appear as part of expression and their transformation into FORTRAN and C language formats is then:

The way how the auxiliary variables are labeled is crucial for the interaction between the *AceGen* and *Mathematica*. *AceGen* system can generate three types of auxiliary variables:

- real type
- integer type
- logical type

New auxiliary variables are labeled consecutively in the same order as they are created, and these labels remain fixed during the *Mathematica* session. This enables free manipulation with the expressions returned by the *AceGen* system. With *Mathematica*, user can perform various algebraic transformations on the optimized expressions independently on *AceGen*. Although auxiliary variables are named consecutively, they are not always stored in the data base in the same order. Indeed, when two expressions contain a common sub-expression, *AceGen* immediately replaces the sub-expression with a new auxiliary variable which is stored in the data base in front of the considered expressions. The internal representation of the expressions in the data base can be continuously changed and optimized.

Auxiliary variables have standardized form $\$V[i, j]$, where i is an index of auxiliary variable and j is an instance of the i^{th} auxiliary variable. The new instance of the auxiliary variable is generated whenever specific variable appears on the left hand side of equation. Variables with more than one instance are "multi-valued variables".

The input for *Mathematica* that generates new auxiliary variable is as follows:

“lhs” operator “rhs”

This structure first evaluates right-hand side expression *rhs*, creates new auxiliary variable, and assigns the new auxiliary variable to be the value of the left-hand side symbol *lhs*. From then on, *lhs* is replaced by a new auxiliary variable whenever it appears. The *rhs* expression is then stored into the *AceGen* database.

In *AceGen* there are four operators:

<i>AceGen Operators</i>	
$v \vDash \text{exp}$	A new auxiliary variable is created if <i>AceGen</i> finds out that the introduction of the new variable is necessary, otherwise $v = \text{exp}$. This is the basic form for defining new formulae. Ordinary <i>Mathematica</i> input can be converted to the <i>AceGen</i> input by replacing the Set operator ($a = b$) with the \vDash operator
$v \vdash \text{exp}$	A new auxiliary variable is created, regardless on the contents of <i>exp</i> . The primal functionality of this form is to force creation of the new auxiliary variable.
$v \vDash \text{exp}$	A new auxiliary variable is created, regardless on the contents of <i>exp</i> . The primal functionality of this form is to create variable which will appear more than once on a left-hand side of equation (multi-valued variables).
$v \dashv \text{exp}$	A new value (<i>exp</i>) is assigned to the previously created auxiliary variable <i>v</i> . At the input <i>v</i> has to be auxiliary variable created as the result of $v \vDash \text{exp}$ command. At the output there is the same variable <i>v</i> , but with the new signature (new instance of <i>v</i>).

So, the operators \vDash and \vdash are used for variables that will appear only once on the left-hand side of equation. For variables that will appear more than once on the left-hand side the operators \vDash and \dashv have to

be used. These operators are replacement for the simple assignment command in Mathematica ($lhs = rhs$).

3.2.4 *AceGen codes and commercial FE environments*

The *AceGen* package provides a collection of prearranged modules for the automatic creation of the interface between the finite element code and several finite element environments like: *Abaqus*, *Feap*, *Elfen* and obviously *AceFem*.

The Interfacing phase of the automatically generated code and FE environment is a two stage process.

At the first stage user subroutine codes are generated, and each user subroutine performs a specific task. The input/output arguments of the generated subroutines are environment and language dependent; however they should contain the same information. Due to the fundamental differences among FE environments, the required information is not readily available.

Thus, at the second stage the contents of the "*splice-file*" that contains additional environment dependent interface and supplementary routines is added to the user subroutines codes. The "*splice-file*" code ensures proper data transfer from the environment to the user subroutine and back.

Automatic interface is already available for a collection of basic tasks required in the finite element analysis. There are several possibilities in the case of need for an additional functionality. Standard user subroutines can be used as templates by giving them a new name and, if necessary, additional arguments.

The additional subroutines can be called directly from the environment or from the enhanced "*splice-file*". The additional subroutines can be generated independently just by using the

AceGen code generator and called directly from the environment or from the enhanced "splice-file".

Since the complexity of the problem description mostly appears in a symbolic input, we can keep the number of data structures that appear as arguments of the user subroutines at minimum. The structure of the data is depicted below. If the "default form" of the arguments as external AceGen variables (see Symbolic-Numeric Interface) is used, then they are automatically transformed into the form that is correct for the selected FE environment. The basic data structures are as follows:

- environment data defines a general information common to all nodes and elements;
- nodal data structure contains all the data that is associated with the node;
- element specification data structure contains information common for all elements of particular type;
- node specification data structure contains information common for all nodes of particular type;
- element data structure contains all the data that is associated with the specific element.

3.2.5 Standard FE procedure

In this chapter a brief overview over the great potentiality of the *AceGen* package are presented. In order to generate a new finite element source code the following four-step procedure has to be executed:

- 1. AceGen Initialization.*** The *AceGen* is loaded while *Mathematica* is running.

- 2. *Template Initialization.*** During this phase all the constants that are needed for proper symbolic-numeric interface are initialized. The essential template constants are basically the *element topology* and the *number of nodal degrees of freedom*. In order to support the user, *AceGen* provides a list of standard element topology, which can be divided into three main groups: 1D, 2D and 3D. *Table 3-1* summarizes and shows the main available element topologies; each one is called through a specific code. During the definition of the template constants, there are a lot of additional options, which can be easily found in the manual, which includes the possibility to define additional nodes (not included in the standard element topology), the number of spatial dimension, the number of degrees of freedom per node for all nodes and arbitrary real values per node.
- 3. *Definition of user subroutines.*** The user can define his original subroutine with the default names and arguments. Moreover, the user can exploit the pre-arranged subroutines, which are able to calculate the *tangent matrix* and *residual* for the current values of nodal and element data, the *post-processing* quantities or the sensitivity pseudo-load vector for the current sensitivity parameters.
- 4. *Code generation.*** It's the last phase of the *AceGen* procedure where the element source code is generated and written in the previously defined environment.

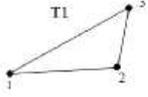
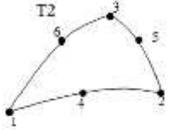
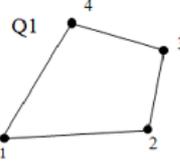
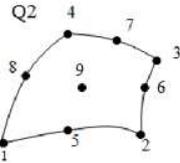
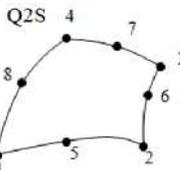
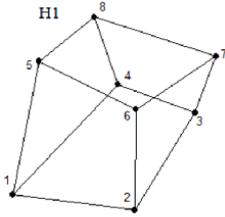
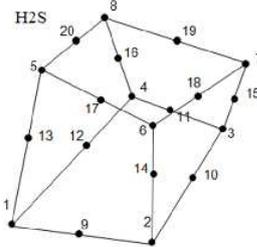
Code	Description	Node numbering
"L1"	2D curve with 2 nodes	
"LX"	2D curve with arbitrary number of nodes	
"T1"	2D Triangle with 3 nodes	
"T2"	2D Triangle with 6 nodes	
"Q1"	2D Quadrilateral with 4 nodes	
"Q2"	2D Quadrilateral with 9 nodes	
"QS"	2D Quadrilateral with 8 nodes	
"H1"	3D Hexahedron with 8 nodes	
"H2S"	3D Hexahedron with 20 nodes	

Table 3-1 Some of the most important element topology implemented in the *AceGen* package.

3.2.6 Automatic differentiation and FEM

Differentiation is an arithmetic operation that plays crucial role in the development of new numerical procedures. The procedure implemented in the *AceGen* system represents a version of automatic differentiation technique.

The tools for *automatic differentiation* (AD) were primarily developed for the evaluation of the gradient of an objective function used within the Newton-type optimization procedures where the Hessian of objective function is needed. The objective functions are often defined by a large, complex program composed of many subroutines. Thus AD tools can be applied directly within the complete FE environment, including all subroutines, to obtain the required derivatives when the evaluation of the objective function involves finite element simulations.

The AD tools have been successfully applied to get gradients of residuals defined by FE environments with several hundred thousand lines of code, see e.g. Bischof [29] et al. (2003). The AD technology can also be used for the evaluation of specific quantities that appear as part of a finite element simulation. It would be difficult and computationally inefficient to apply the AD tools within large FE systems to get e.g. the global stiffness matrix of large-scale problem directly. This is especially problematic when a fully implicit Newton type procedure is used to solve nonlinear, transient and coupled problems involving various types of elements, complicated continuation or arc-length methods and adaptive procedures.

However, one can still use automatic differentiation at the single element level to evaluate element specific quantities in an efficient way such as:

- consistent tangent stiffness matrix;
- strain and stress tensors;
- nonlinear coordinate transformations;
- residual vector.

A direct use of automatic differentiation tools for the development of nonlinear finite elements turns out to be complex and not straightforward; furthermore the numerical efficiency of the resulting codes is poor. One solution, followed in *AceGen* is to combine a general computer algebra system and the AD technology. The implementation of the automatic differentiation procedure has to fulfill specific requirements in order to develop element source codes automatically that are as efficient as manually written codes. Some basic requirements are:

- The storage of the intermediate variables is not a limitation when the differentiation in reverse mode is used at the single element level. Finite element formulations at the single element level involve a relatively small set of independent and intermediate variables.
- For the reasons of efficiency, the results of all previous applications of automatic differentiation have to be accounted for, when automatic differentiation is used several times inside the same subroutine.
- The user has to be able to employ all the capabilities of the symbolic system within the final and the intermediate results of the AD procedure.

The mathematical formalisms that are part of the traditional FE formulation are e.g. partial derivatives $\frac{\partial(\cdot)}{\partial(\cdot)}$, total derivatives $\frac{D(\cdot)}{D(\cdot)}$ or directional derivatives. They can all be represented by an AD procedure if possible exceptions are treated in a proper way. However, the result of AD procedure may not automatically correspond to any of the above mathematical formalisms. Hence let us define a “conditional derivative” by the following formalism:

$$\nabla f = \frac{\partial f(a, b(a))}{\partial(a)} \bigg|_{\frac{\partial(b)}{\partial(a)}=M}$$

where function f depends on a set of mutually independent variables \mathbf{a} and a set of mutually independent intermediate variables \mathbf{b} . The above formalism has to be viewed in an algorithmic way. It represents the automatic differentiation of function f with respect to variables \mathbf{a} . During the AD procedure, the total derivatives of intermediate variables \mathbf{b} with respect to independent variables are set to be equal to matrix \mathbf{M} . Some situations that typically appear in the formulation of finite elements are presented in Table 4-2.

Type	Formalism	Schematic <i>AceGen</i> input
A	$\Delta f = \frac{\partial f(\mathbf{a}, \mathbf{b}(\mathbf{a}))}{\partial(\mathbf{a})} \bigg _{\frac{\partial(\mathbf{b})}{\partial(\mathbf{a})}=\mathbf{M}}$	<code>a = SMSReal[a\$\$\$] b = SMSFreeze[f_b[a]] df = SMSD[f[a, b], a, "Implicit" → {b, a, M}]</code>
B	$\Delta f = \frac{\partial f(\mathbf{b})}{\partial(\mathbf{a}(\mathbf{b}))} \bigg _{\frac{\partial(\mathbf{b})}{\partial(\mathbf{a})}=\mathbf{M}}$	<code>b = SMSReal[b\$\$\$] a = SMSFreeze[f_a[b]] df = SMSD[f[b], a, "Implicit" → {b, a, M}]</code>
C	$\Delta f = \frac{\partial f(\mathbf{a}, \mathbf{b}(\mathbf{a}))}{\partial(\mathbf{a})} \bigg _{\frac{\partial(\mathbf{b})}{\partial(\bullet)}=0}$	<code>a = SMSReal[a\$\$\$] b = f_b[a] df = SMSD[f[a, b], a, "Constant" → b]</code>
D	$\frac{\partial(\bullet)}{\partial(\bullet)} \bigg _{\frac{\partial(\mathbf{b})}{\partial(\mathbf{a})}=\mathbf{M}}$	<code>a = SMSReal[a\$\$\$] b = SMSFreeze[f_b[a], "Dependency" → {a, M}] ... df_i = SMSD[f_i[a, b], a]</code>

Table 3-2 Automatic differentiation exception [22]

In case A, there exists an explicit algorithmic dependency on \mathbf{b} with respect to \mathbf{a} , hence the derivatives can be obtained in principle

automatically, without intervention by the user, simply by the chain rule. However, there also exists a profound mathematical relationship that enables evaluation of derivatives in a more efficient way. This is often the case when the evaluation of \mathbf{b} involves iterative loops, inverse matrices, etc.

Case B represents the situation when variables \mathbf{b} are independent variables and variables \mathbf{a} implicitly depend on \mathbf{b} . This implicit dependency has to be considered for the differentiation. In this case, automatic differentiation would not provide the correct result without the user intervention. A

Typical example for this situation is a differentiation that involves a transformation of coordinates. Usually the numerical integration procedures as well as interpolation functions require additional reference coordinate. An exception for automatic differentiation of type B is then introduced to properly handle differentiation involving coordinate transformations from initial X to reference coordinates ξ as follows:

$$\frac{\partial(\cdot)}{\partial X} \Rightarrow \frac{\partial(\cdot)}{\partial X} \bigg|_{\frac{\partial \xi}{\partial X} = \left[\frac{\partial X}{\partial \xi} \right]^{-1}}$$

In case C, there exists an explicit dependency between variables \mathbf{b} and \mathbf{a} that has to be neglected for differentiation. The status of the dependent variable \mathbf{b} is thus temporary. For the duration of the AD procedure, it is changed into an independent variable. The situation frequently appears in the formulation of mechanical problems where instead of the total variation some arbitrary variation of a given quantity has to be evaluated.

3.2.7 Debugging module

Since *AceGen* exploits a simultaneous optimization procedure, during the code creation phase it's possible to execute each step separately and examine intermediate results. This characteristic allows to trace the errors that might occur during *AceGen* session.

Despite this strong capability, is not always easy understanding the automatically generated formulae. For example, after using the automatic differentiation tools we have no insight in the actual structure of the derivatives. While formulae are derived automatically with *AceGen*, it tries to find the actual meaning of the auxiliary variables and assigns appropriate names.

In order to avoid such problems, it's possible to retain this information and explore the structure of the generated expressions through two main ways: the *Browser mode submenu* or the *run time debugging*.

The break points can be inserted into the source code by *SMSSetBreak* function. Break points are inserted only if the code is generated with the "Mode"→"Debug" option. In "Debug" mode the system also automatically generates file with the name "*sessionname.dbg*" where all the information necessary for the run-time debugging is stored. The data is restored from the file by the *SMSLoadSession* command. The number of break points is not limited. All the user defined break points are by default active. With the option "*Active*"→*False* the break point becomes initially inactive. The break points are also automatically generated at the end of *If..endif* and *Do...enddo* statements a additionally to the user defined break points. All automatically defined break points are by default inactive. Using the break points is also one of the ways how the automatically generated code can be debugged.

3.3 AceFem

3.3.1 AceFem overview

The *AceFem* package is a general finite element environment which exploits the advantages of symbolic capabilities offered by *Mathematica* to solve multi-physics and multi-field problems, while maintaining numerical efficiency of commercial finite element environment [22]. In combination with the automatic code generation package AceGen, the AceFem package represents an ideal tool for a rapid development of new numerical models.

The *AceFem* package is designed to solve steady-state or transient finite element and similar type problems implicitly by means of Newton-Raphson type procedures.

The idea implemented in *AceFem* is to design a FE environment where code complexity will be shifted out of the finite element environment to a symbolic module, which will provide all the necessary formulation dependent codes by automatic code generation.

The shift concerns the numerical algorithms (at the local element level) and data structures like the organization of environment and the nodal or element data.

One of the most important and original issue of *AceFem* package is that this software contains a lot of useful pre-arranged procedures, written and executed directly inside *Mathematica* and usually not numerically intensive. For example, it's easily possible to:

- Process the user's input data;

- Generate mesh;
- Control solutions procedures;
- Interactive post-processing analysis;
- Create graphic post-processing of the results.

Moreover, *AceFem* includes a numerical module which exists as *Mathematica* package as well as external program written in C language and is connected with *Mathematica* via the *MathLink* protocol. This peculiar and unique capability allows the user to solve different large-scale problems with several 100000 unknowns.

A large group of numerically intensive operations are included in the numerical module allowing the user to remarkably simplify numerical procedures. For instance, the following operations are included in the numerical module:

- Evaluation and assembly of the finite element quantities like the tangent matrix, sensitivity vectors, etc.;
- Solution of linear system of equations;
- contact search procedures.

The use of *AceFem* package lets the user to make use of advanced capabilities of *Mathematica* like: high precision arithmetic, interval arithmetic; or even symbolic evaluation of FE quantities to analyze various properties of the numerical procedures on relatively small examples.

The *AceFem* package comes with a large library of finite elements (solid, thermal, contact, 2D, 3D,) including full symbolic input for most of the elements. *AceShare* is a finite element file sharing system which allows the user to access additional elements.

The fundamental idea of *AceFem* software is to create an *Element Oriented Approach*, which enables easy creation of customized finite element based applications in *Mathematica*.

3.3.2 Comparison between Traditional commercial FEM environments and AceFem

The complexity of advanced numerical software arises from the need to describe in the most realistic way physical phenomena involved in engineering or scientific problems, using high-efficient numerical procedures.

The traditional commercial FEM systems have usually incorporated several hundred different element formulations, but the use of reusable parts of the code is obviously necessary. In fact, the element shape functions, material models, pre and post-processing procedures are often written as reusable codes. The complete structure appears inside FEM environment which is usually written in several languages like *FORTRAN* or *C*.

During the last decades, the best way to create reusable and extensible numerical software was considered the use of the *Object-Oriented Approach* or *Object-Oriented Programming (OOP)*. An object-oriented program may thus be viewed as a collection of interacting *objects*, as opposed to the conventional model, in which a program is seen as a list of tasks (subroutines) to perform. In OOP, each object is capable of receiving messages, processing data, and sending messages to other objects. Each object can be viewed as an independent "machine" with a distinct role or responsibility. The actions (or "methods") on these objects are closely associated with the object.

Usually, the non-OOP programs may be composed of just one long list of commands. More complex programs will group lists of commands into functions or subroutines each of which might perform a particular task. With designs of this sort, it is common for the program's data to be accessible from any part of the program.

By contrast, the OOP encourages the programmer to place data where it is not directly accessible by the rest of the program. Instead the data is accessed by calling specially written functions, commonly called *methods*. The *object* is a programming construct that combines data with a set of methods for accessing and managing that data.

Most of the main FEM programs (like *Abaqus*, *Ansys*, *Marc*, etc.) are still written in non-OOP, despite the undoubted success of OOP approach in different areas.

One of the reasons for this is that only the shift of complexity of data management has been performed by utilizing OOP methods, while the level of abstraction of the problem description remains the same. Through the use of the symbolic approach *AceFem* can bypass the drawback of the OOP formulation (the level of abstraction of the problem description remains the same). This is made possible since only the basic functionality is provided at the global level of the finite element environment which is manually coded, while all the codes at the local level of the finite element are automatically generated.

Based on the OOP, the *AceFem* package has been designed in an innovative way that explores advantages of a new symbolic approach based on to the design of finite element environments.

3.3.3 AceFem internal Structure

The *AceFem* structure could be divided into two main parts. The first one is a group of useful procedures, written and executed directly inside *Mathematica*, usually not numerically intensive. These procedures consist of some basic pre-processing and post-processing functions (SMTMesh, SMTShowMesh) which could help the user to visualize the geometry of the elements, the meshing, the boundary conditions, the input data phase and so on.

The second part of *AceFem* structure is the numerical module. In The *AceFem* package there are two versions of numerical module:

- **CDriver**; it is the basic version which is independent executable written in C language and is connected with *Mathematica* through a *MathLink* protocol. Since the element subroutines are linked dynamically with the *CDriver* there are as many dynamically linked library files (dll file) as is the number of different elements. The *dll* file is created automatically by the pre-arranged *SMTMakeDll* function. The user can derive or use its own user defined finite elements or it can use standard elements from the extensive library of standard elements (accessing elements from shared libraries).

- **MDriver**; it's the alternative version and it can exploit the advanced capabilities of *Mathematica*, such as high precision arithmetic, interval arithmetic, or even symbolic evaluation of FE quantities to analyze various properties of the numerical procedures. It's is completely written in *Mathematica*'s symbolic language, but it doesn't support advanced post processing, contact, etc.

3.3.4 General AceFem procedure

Basically the standard *AceFem* procedure scheme can be divided in three main phases:

1. ***Input data phase.*** During this phase is composed by different and obligatory commands. The users can load the needed element code and define the data common to all elements of the specific type. The element codes can be taken from the *AceShare* libraries or they can be previously generated by the *AceGen* code generator. Later the geometry and the mesh of the element are defined through a particular function, which define the nodes coordinates and their connectivity for topological mesh. The topological mesh is a base on which the actual finite element mesh is constructed. Finally, the essential and the natural boundary conditions (respectively, Dirichlet and Neumann b.c.) of the problem are specified by the specific commands. If sensitivity analysis is required then the user specifies the type and the values of the sensitivity parameters.

2. ***Analysis phase.*** The analysis phase can be considered as the core of the *AceFem* procedure, since in few lines of code, it is able to solve steady-state or transient finite element and related problem implicitly by means of **Newton-Raphson** procedures. The actual solution procedure is executed accordingly to the *Mathematica* input given by the user. During this phase the correctness of the previously defined input data is checked, and then compiles the element source files and creates dynamic link library files (dll file) with the user subroutines or in the case of *MDriver* reads all the element source files into *Mathematica*. The *SMTAnalysis* command transcripts input data structures into analysis data base

structures. When the analysis phase is initialized, it starts the *iterative solution procedure* using a Newton-Raphson iterative procedure. The actual solution procedure is executed accordingly to the *Mathematica* input given by the user. The solution procedure is executed accordingly to the *Mathematica* input given by the user through the *SMTConvergence*. This command is a high performance and powerful function. It checks if the convergence conditions for the iterative procedure have been satisfied. In fact, this function is defined by a tolerance, a maximum number of iterations, and a desired number of iterations. During the iterative procedure, the increment of time "*t*" or of boundary conditions multiplier "*λ*" can be constant or increasing in an adaptive way, defining through the *SMTConvergence* function, a maximal and a minimal increment.

3. ***Post-processing phase.*** The graphic post-processing of the results can be part of the analysis or done later independently of the analysis.

3.3.5 AceShare

The *AceFem* application has a built-in library including standard solid, structural, thermal and contact elements.

The ***AceShare*** system is a finite element file sharing software built in *AceFem* that makes *AceGen* generated finite element source codes available for other users to download through the Internet.

Moreover, it allows to:

- browse the online FEM libraries;
- download the finite elements from the on-line libraries;

- create a user defined library that can be posted on the internet to be used by other users of the *AceFem* system.

The AceShare system offers for each finite element included in the on-line library: the element home page with basic descriptions, the *AceGen* template for the symbolic description of the element, the element source codes for all supported finite element environments (*FEAP*, *AceFem* -MDriver, *Abaqus* ...).

3.3.6 Interactive Debugging

The procedures described in the Section of the *AceGen* debugging module (for the run-time debugging of automatically generated codes), can be used within the *AceFem* environment as well. For the interactive debugging procedures, the code has to be generated in "Debug" mode. By default compiler compiles the code generated in "Debug" mode with the compiler options for debugging. For a large scale problem this may represent a drawback.

The *AceFem* package could be exploiting for debugging and testing new finite element before it is included into the commercial finite element environment. Directly in *Mathematica* is possible to experience several tests, like:

- tests of the element eigenvalues;
- convergence of iterative procedures;
- test of objectivity.
- different forms of the patch tests;
- element distortion tests.

4. *Multi-step* approach: Developed numerical models

4.1 Introduction

The aim of the present research work is the development of a sequence of new numerical models, oriented to the honeycomb sandwich panels modeling, by using an innovative package of software for FEM analysis. In particular, the *AceGen* symbolic formulation and its sibling FEM environment, *AceFem*, have been selected to show the great potential of such packages on rapid prototyping of existing and original numerical models.

In this chapter, is presented the *multi-step* approach adopted in order to achieve the above mentioned research purpose. This particular multi-step process allows the user the formulation of a new constitutive model, through the development of a logical

sequence of intermediate constitutive models, by following a gradual order of complexity and precision.

4.1.1 Overview of the proposed numerical models

As posted before, during the present research work, the symbolic formulation of *AceGen* and the FEM environment of *AceFem* have been chosen to show the great potential of such packages on rapid prototyping of new numerical models. In particular, all the developed elements are oriented to the developing of a new phenomenological constitutive model able to well-simulate the crushing behavior of honeycomb-cored sandwich panels.

The behavior of the honeycomb core materials subjected to out-of-plane compressive loads represents a topic of strong engineering interest.

In order to achieve this ambitious purpose, a *multi-step* process is proposed. Such original multi-step method helps the user to formulate a new and complex constitutive model through the development of a logical sequence of constitutive models, by following a gradual order of complexity.

During the present research work, the model of such material doesn't require the time-consuming and complex definition of the exact geometry of honeycomb; all the constitutive models are oriented to model the honeycomb core materials, by using a continuum model.

Obviously other FEM software allows the generation of a continuum model; but by using the *AceGen* package the user is able to rapidly create a self-own element, without using an already implemented element. For this reason, the main advantage of such method is that the topic presented in this paper is studied under the versatile perspective.

Since this kind of multi-step approach requires the development of several new numerical models, an appropriate knowledge of general FEM theory and of the used software is needful.

The following table lists all the main elements created during the present thesis:

- *Hyperelastic element*
 - 2D element, quadrilateral element with four nodes;
 - 3D element, hexahedral element with eight nodes;
- *Element MINI*
 - 2D element, triangular element enriched by an inner bubble, where seven integration points are defined;
- *Elasto-plastic 3D element*
 - Traditional J2 plasticity;
 - Elasto-perfectly plastic;
 - J2 plasticity with an original hardening rule.

4.1.2 Finite strain kinematical recalls

In the finite deformation analysis a careful distinction has to be underlined between the coordinate systems that can be chosen to describe the behavior of the body whose motion is under consideration [23]. All the relevant quantities, such as density ρ , can be described in two main ways:

- *Material description*, where the variation of ρ over the body is described with respect to the original (or initial) coordinate \mathbf{X} used to label a material particle in the continuum at time $t=0$;
- *Spatial description*, where ρ is described with respect to the position in space \mathbf{x} , currently occupied by a material particle in the continuum at time t .

Let's consider a continuum solid body Ω_0 subjected to a change of his spatial configuration. The result of such new configuration is a displacement of the body which can be described with two main components: a *rigid-body displacement* and a *deformation*. The rigid-body displacement consists of a simultaneous translation and rotation of the body without changing its shape or size. Deformation implies the change in shape and/or size of the body from an initial or unreformed configuration to a current or deformed configuration.

The *displacement field* \mathbf{u} is a vector field of all displacement vectors for all particles in the body, which relates the deformed configuration with the undeformed configuration. In such a way the displacement field describes the motion of each point in the solid.

A key quantity in finite deformation analysis is the *deformation gradient* \mathbf{F} , which is involved in all equations relating quantities before deformation to corresponding quantities after deformation. The deformation gradient tensor enables the relative spatial position

of two neighboring particles after deformation to be described in terms of their relative material position before deformation. For this reason such quantity is central to the description of deformation and hence strain. Starting from the definition of the displacement gradient tensor,

$$\nabla u = \frac{\partial u_i}{\partial x_k}$$

The deformation gradient tensor can be defined as

$$\mathbf{F} = \mathbf{I} + \nabla u \quad \text{or} \quad F_{ik} = \delta_{ik} + \frac{\partial u_i}{\partial x_k}$$

Where the δ_{ik} is the Kronekor delta symbol.

The Jacobian measures the volume change produced by a deformation and it is defined as:

$$J = \text{def}(\mathbf{F}) = \det\left(\delta_{ik} + \frac{\partial u_i}{\partial x_k}\right)$$

In 1839, George Green introduced a deformation tensor known as the *right Cauchy-Green deformation tensor* or *Green's deformation tensor*, defined as:

$$\mathbf{C} = \mathbf{F}^T \mathbf{F} = \mathbf{U}^2 \quad \text{or} \quad C_{ij} = F_{kI} F_{kJ} = \frac{\partial x_k}{\partial X_I} \frac{\partial x_k}{\partial X_J}$$

Finally, the concept of *strain* is used to evaluate how much a given displacement differs locally from a rigid body displacement. One of such strains for large deformations is the Lagrangian finite strain tensor, also called the *Green-Lagrangian strain tensor* or *Green - St-Venant strain tensor*, defined as:

$$\mathbf{E} = \frac{1}{2} (\mathbf{C} - \mathbf{I})$$

The stress-strain law must then be deduced by differentiating the strain energy density. Here the strain energy density is defined in terms of \mathbf{F} , as:

$$\sigma_{ij} = \frac{1}{J} F_{ik} \frac{\partial W}{\partial F_{kj}}$$

4.1.3 General Isoparametric concept

Different approximations are made when the method of finite elements is applied to discretize weak forms of the general non-linear problems. FEM approach approximates the geometry of a body B in the initial configuration by:

$$B \approx B^h = \bigcup_{e=1}^{ne} \Omega_e$$

where the continuous body is subdivided into n_e finite elements.

Within such finite element methodology, interpolation functions have to be chosen in order to approximate the primary field variables.

Let the generic finite element e be defined by n_{node} nodes with one shape function (or interpolation function) $N_I(X)$ associated with each node I whose coordinate is x^i . Figure 4-1 illustrates a three-noded triangular element with linear shape functions.

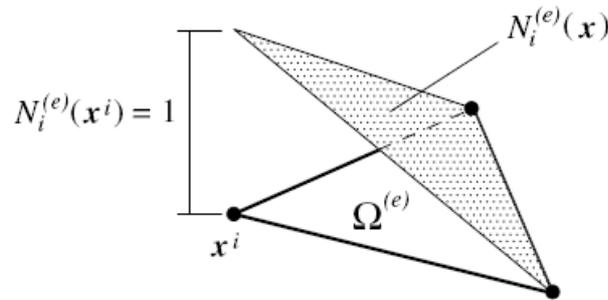


Figure 4-1 Finite element interpolation:
Example of triangular element shape function.

Each shape function $N_I(X)$ is defined so that its value is unity at node i , i.e.

$$N_I^e(x^i) = 1$$

and zero at any other node of the element:

$$N_I^e(x^i) = 0 \quad \text{for } j \neq i$$

Hence the exact solution of the mathematical model is approximated within one finite element by:

$$u_{exact}(X) \approx u_h(X) = \sum_{I=1}^n N_I(X)u_i,$$

Where the vector \mathbf{X} denotes the position vector with respect to the initial configuration in Ω_e , $N_I(\mathbf{X})$ are the shape functions which are defined in Ω_e and the unknown nodal quantities of primary variable are represented by \mathbf{u}_i .

One basic requirement for the choice of the approximation \mathbf{u}_h is the convergence of the finite element solution to the true solution of the underlying partial differential equation. Different possibilities exist to construct interpolation for geometry and variables within the finite element method. For convergence reasons, these functions have to be completed up to the approximation.

Due to its general applicability, the isoparametric concept is mainly used as interpolation scheme for many engineering problems. Isoparametric elements allow a very good and sufficiently accurate mapping of arbitrary geometries into a finite element mesh. Moreover, this concept is extremely well suited for nonlinear problems since a discretization of the spatial formulation is easily obtained. Within the isoparametric theory all the kinematical variables as well as the geometry (e.g. the geometry in the initial \mathbf{X} or spatial configuration \mathbf{x}) are interpolated by the same shape functions N_I . This can be also be expressed mathematically within one element Ω_e by

$$X_e = \sum_{I=1}^n N_I(\xi) X_I \quad \text{and} \quad x_e = \sum_{I=1}^n N_I(\xi) x_I$$

where $\xi = \{\xi, \eta, \zeta\}$ represent the natural coordinate of the local system.

In this Section the simplest two-dimensional elements are presented. In the following Sections will be presented the three-dimensional elements formulation too.

4.2 Hyperelastic elements

As following described, the first elements implemented for the present multi-step approach, are the 2D and 3D hyperelastic elements. After a brief introduction on the hyperelasticity theory, is presented an introductory review of the adopted kinematical relationships.

4.2.1 Hyperelastic model formulation

Hyperelastic constitutive laws are used to model materials that respond elastically when subjected to very large strains. They account both for nonlinear material behavior and large shape changes. The main applications of such theory are to model the rubbery behavior of a polymeric material, and to model polymeric foams that can be subjected to large reversible shape changes.

The literature on stress-strain relations for finite elasticity can be hard to follow, partly because nearly every paper uses a different notation, and partly because there are many different ways to write down the same stress-strain law. For this reason the formulation of hyperelastic model is presented.

In order to define a hyperelastic models it should be defined the stress-strain relation for the solid by specifying its strain energy density W as a function of deformation gradient tensor:

$$W=W(F) \qquad \text{Eq. 4.1}$$

This ensures that the material is perfectly elastic, and also means that we only need to work with a scalar function. The general form of the strain energy density is guided by experiment; and the

formula for strain energy density always contains material properties that can be adjusted to describe a particular material.

Formulas for stress in terms of strain are calculated by differentiating the strain energy density.

Considering Ω_0 the initial domain of the problem, the proposed formulation is defined by the hyperelastic **Neo-Hooke** type strain energy potential, where λ and μ represent the first and the second *Lame's* material constants:

$$W = \int_{\Omega_0} \left(\frac{\lambda}{2} (\text{Det}(F) - 1)^2 + \mu \left(\frac{\text{Tr}[C] - 3}{2} - \text{Log}[\text{Det}(F)] \right) \right) d\Omega_0,$$

The \mathbf{F} is the deformation gradient related to the \mathbf{u} displacement field, and \mathbf{C} is the *right Cauchy-Green tensor*.

In order to well understand these formulations all the kinematical quantities used to define the present strain energy potential are introduced in the following Section.

4.2.2 Bi-linear quadrilateral

The present hyperelastic 2D element is based on an isoparametric formulation for a bilinear quadrilateral element with four nodes.

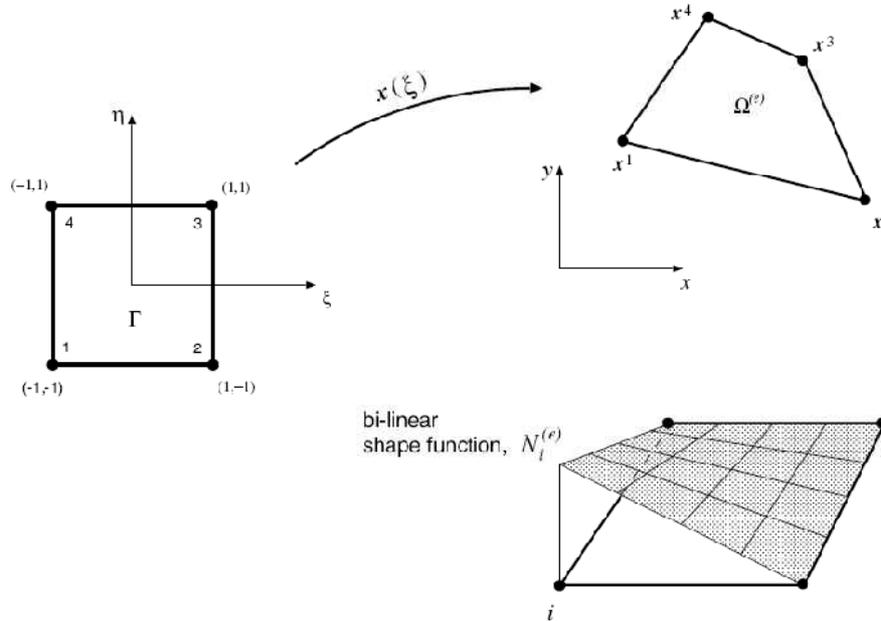


Figure 4-1 Bi-linear quadrilateral

The bi-linear quadrilateral has the following shape functions:

$$\begin{aligned}
 N_1(\xi) &= \frac{1}{4}(1 - \xi - \eta + \xi\eta) & N_2(\xi) &= \frac{1}{4}(1 - \xi + \eta - \xi\eta) \\
 N_3(\xi) &= \frac{1}{4}(1 + \xi + \eta + \xi\eta) & N_4(\xi) &= \frac{1}{4}(1 + \xi - \eta - \xi\eta)
 \end{aligned}$$

The four-point (2 × 2) Gauss quadrature is usually adopted for this element. The corresponding sampling point positions and weights can be found in *AceGen* subroutine, and they are shown in Figure 4-2.

p	ξ_p	η_p	W_p	Position of points
1	$-1/\sqrt{3}$	$-1/\sqrt{3}$	1	
2	$+1/\sqrt{3}$	$-1/\sqrt{3}$	1	
3	$-1/\sqrt{3}$	$+1/\sqrt{3}$	1	
4	$+1/\sqrt{3}$	$+1/\sqrt{3}$	1	

Figure 4-2 Two dimensional gauss integration for quadrilateral elements

4.2.3 Three-dimensional isoparametric element

Three-dimensional finite elements can have a variety of different shapes. They can be tetrahedral or hexahedral shapes but also mixtures of both types for special geometries like prism.

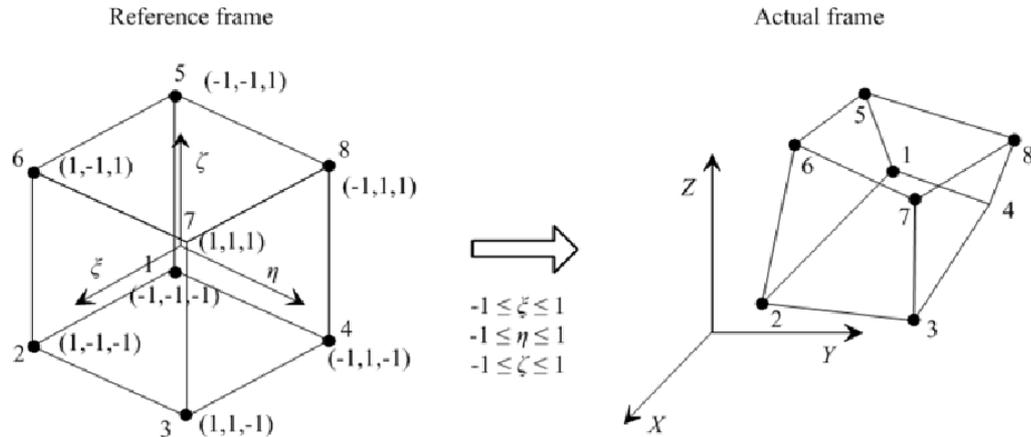


Figure 4-3 Isoparametric 8-node hexahedral element

In order to model the present hyperelastic three-dimensional element, the formulation of hexahedral element is adopted. Again the isoparametric formulation is used to be able to generate general shape functions for the discretization of arbitrary three-dimensional geometries.

The shape functions for this element are given by:

$$N_I(\xi, \eta, \zeta) = \frac{1}{2}(1 + \xi_I \xi) \frac{1}{2}(1 + \eta_I \eta) \frac{1}{2}(1 + \zeta_I \zeta)$$

Where ξ , η and ζ represent the natural coordinates of the local system.

4.2.4 AceGen input file: *Hypersolid2D_Quadrilateral*

In this Section the structure of the input for *AceGen* package is presented for the generation of a 2D four node finite element. In particular are presented all the main steps needed for the generation of the code.

- Step 1 – **Initialization**

Here the *AceGen* is initialized and the element characteristics necessary for the automatic creation of the interface between the automatically generated code and the chosen finite element environment are defined. Moreover, through the *Debug mode*, it's possible during the *AceFem* analysis the run time debugging of all the calculated quantities. In *Debug mode* the system also automatically generates file with the name "*Hypersolid-2D_Quadrilateral.dbg*" where all the information necessary for the run-time debugging is stored.

Through the *SMSStandardModule* command starts the definition of the user subroutines for the calculation of the *tangent matrix* and the *residual vector*. After that the loop over the Gauss point (**Ig**) is initialized. All the input data like the *Young modulus*, the *density*, the *Poisson ratio* and the *forces per unit mass* are defined.

```
<< "AceGen`";
SMSInitialize["Hypersolid2D_Quadrilateral",
  "Environment" → "AceFEM",
  "Mode" → "Debug"];
SMSTemplate["SMSTopology" → "Q1",
  "SMSSymmetricTangent" → True,
  "SMSGroupDataNames" → {"E -elastic modulus", "ν -poisson ratio",
    "t -thickness", "ρ0 -density", "bX -force per unit mass X",
    "bY -force per unit mass Y"},
  "SMSDefaultData" → {21 000, 0.3, 1, 1, 0, 0}];
SMSStandardModule["Tangent and residual"];
SMSDo[Ig, 1, SMSInteger[es$$["id", "NoIntPoints"]]];
```

- Step 2 – Interface to the input data of the user element subroutine and the kinematical formulation

Here the coordinates of the current integration point ξ, η , the integration point weights ω_g , the coordinates of the element nodes X_i, Y_i , the current value of the displacement u_i, v_i and the material properties of the element are taken from the supplied arguments of the subroutine. All global degrees of freedom are then collected in one vector such that the proper degree of freedom ordering is established.

Moreover, during this step are defined the shape functions N_i , the interpolation of the physical coordinates X, Y and the displacements u, v within the element, the *Jacobi matrix* of the isoparametric mapping, the displacement gradient and the *right Cauchy-Green deformation tensor*. At the end the hyperelastic **Neo-Hooke** type strain energy potential is defined as:

$$W = \frac{\lambda}{2} (\text{Det}(F) - 1)^2 + \mu \left(\frac{\text{Tr}[C] - 3}{2} - \text{Log}[\text{Det}(F)] \right)$$

```

E = {ξ, η, ζ} + Table[SMSReal[es$$["IntPoints", i, Ig]], {i, 3}];
Xh + Table[SMSReal[nd$$[i, "X", j]],
  {i, SMSNoNodes}, {j, SMSNoDimensions}];
Nh = 1/4 {(1 - ξ) (1 - η), (1 + ξ) (1 - η), (1 + ξ) (1 + η), (1 - ξ) (1 + η)};
X + SMSFreeze[Append[Nh.Xh, ζ]];
Jg = SMSD[X, E];
Jgd = Det[Jg];
uh + SMSReal[Table[nd$$[i, "at", j],
  {i, SMSNoNodes}, {j, SMSNoDimensions}]];
pe = Flatten[uh];
u = Append[Nh.uh, 0];
Dg = SMSD[u, X, "Dependency" -> {E, X, SMSInverse[Jg]}];
SMSFreeze[F, IdentityMatrix[3] + Dg, "KeepStructure" -> True];
JF = Det[F]; Cg = Transpose[F] . F;
{Em, v, tζ, ρ0, bX, bY} +
  SMSReal[Table[es$$["Data", i], {i, Length[SMSGGroupDataNames]}]];
{λ, μ} = SMSHookeToLame[Em, v]; bb = {bX, bY, 0};
W = 1/2 λ (JF - 1)^2 + μ (1/2 (Tr[Cg] - 3) - Log[JF]);

```

In the following table are well underlined the *AceGen* functions employed during this step.

<u>nd</u> \$\$[j, "X", i]	i^{th} coordinate of j^{th} node
<u>nd</u> \$\$[j, "at", i]	i^{th} d.o.f. in j^{th} node
<u>es</u> \$\$["Data", i]	i^{th} material constant
<u>es</u> \$\$["IntPoints", i, j]	i^{th} coordinate of i^{th} Gauss point
<u>es</u> \$\$["IntPoints", 4, j]	Gauss point weight in j^{th} Gauss point
<u>es</u> \$\$["id", "NoIntPoints"]	number of Gauss points

- Step 3 – **element tangent stiffness matrix and internal force vector**

Here the internal force vector Rg and the stiffness matrix Kg are evaluated for the final values at a Gauss point Ig .

The vectors, containing the quantities Kg and the Rg are exported by *SMSExport* function to the output parameters of the user element subroutine.

```
wgp = SMSReal [es$$["IntPoints", 4, Ig]] ;
SMSDo [
  Rg = Jgd t$ wgp SMSD [W - rho0 u.bb, pe, i] ;
  SMSExport [SMSResidualSign Rg, p$$[i], "AddIn" -> True] ;
  SMSDo [
    Kg = SMSD [Rg, pe, j] ;
    SMSExport [Kg, s$$[i, j], "AddIn" -> True] ;
    , {j, i, 8}] ;
    , {i, 1, 8}] ;
  SMSEndDo [] ;
```

- Step 4 – **Post-processing**

Here all the quantities for the post-processing are defined. This phase allows the user to plot the results of the future analysis. The stress-strain law must then be deduced by differentiating the strain energy density. Here the *Lagrangian finite strain tensor* and the strain energy density are defined in terms of \mathbf{F} , as:

$$E = \frac{1}{2}(C - I)$$
$$\sigma_{ij} = \frac{1}{J} F_{ik} \frac{\partial W}{\partial F_{kj}}$$

```
SMSStandardModule["Postprocessing"];
SMSDo[Ig, 1, SMSInteger[es$$["id", "NoIntPoints"]]];
ElementDefinitions[];
SMSNPostNames = {"DeformedMeshX", "DeformedMeshY", "u", "v"};
SMSExport[Table[Join[uh[[i]], uh[[i]], {i, SMSNoNodes}], npost$$];
Eg = 1/2 (Cg - IdentityMatrix[3]);
σ = (1/JF) * SMSD[W, F, "IgnoreNumbers" -> True] . Transpose[F];
SMSGPostNames = {"Exx", "Eyy", "Exy", "Sxx", "Syy", "Sxy", "Szz"};
SMSExport[Join[Extract[Eg, {{1, 1}, {2, 2}, {1, 2}}],
  Extract[σ, {{1, 1}, {2, 2}, {1, 2}, {3, 3}}]], gpost$$[Ig, #1] &];
SMSEndDo[];
```

- Step 5 – **Code generation**

This is the end of the integration loop. The element source code is generated and written in C language. Now it can be used for the FEM analysis of *AceFem*.

```
SMSWrite[];
[3] Consistency check - global
[4] Consistency check - expressions
[4] Generate source code :
Events: 14 SMSDB-0
[4] Final formatting
```

File:	Hypersolid2D_Quadrilateral.c	Size: 27231
Methods	No. Formulae	No. Leafs
SKR	118	2050
SPP	105	1560

4.2.5 AceGen input file: *Hypersolid3D_Hexahedral*

The steps are almost the same of the previous *Hypersolid-2D_Quadrilateral*. The definition of the shape functions defined in section 4.2.3 is outlined.

```
En = {{-1, -1, -1}, {1, -1, -1}, {1, 1, -1}, {-1, 1, -1},  
      {-1, -1, 1}, {1, -1, 1}, {1, 1, 1}, {-1, 1, 1}};  
NI = Table[1/8 (1 + ξ En[[i, 1]]) (1 + η En[[i, 2]]) (1 + ζ En[[i, 3]]), {i, 1, 8}];
```

4.3 Element *Mini* implementation

In this Section a study of the finite elasticity problems for incompressible materials is proposed. The formulation is based on the theory proposed by *F. Auricchio, L. Beirao da Veiga, C. Lovadina, and A. Reali* [12], and it is focused on a high constrained two-dimensional problem.

During the present research work, this two-dimensional problem is studied with *AceGen* package, in order to test the program on high constrained problems. In particular, the formulation of the bidimensional element *MINI* (*Arnold* [13]), has been implemented. In the following, just a brief theoretical overview on the incompressible materials and on *MINI* element is presented, because this paper is basically focused on the implementation of such numerical models through *AceGen* package.

4.3.1 Overview on incompressible materials

In the case of highly constrained situations, like the incompressible materials, there are several efficient FE interpolation scheme able to well represent the behavior for the case of small deformation problems, like the *standard mixed elements* (see, for instance, *Bathe, 1996, Brezzi and Fortin, 1991*) or the *enhanced strain elements* (see *Auricchio et al., 2005a, Braess, 1998*).

However, it is also well established that the extension of such schemes to the case of finite strain problems is by no way trivial.

Since a satisfactory analysis of finite element methods for finite strain problems is still missing, a theoretical approach is proposed by *Auricchio*¹³. One of the used interpolation scheme proposed to discretize the domain is the element *MINI*. In the following the formulation for the development of such element is described.

4.3.2 Element Mini formulation

The element MINI is a triangular element with piecewise linear continuous approximation for both displacements and pressure, with the displacements enriched by a cubic bubble as proposed by *Arnold, Brezzi, and Fortin* (1984). The unstable linear velocity-linear pressure Stokes element may be stabilized by the addition of a single internal velocity degree of freedom via a bubble ⁶.

Therefore, the element *MINI* could be seen as a particular triangular element, with three external nodes, defining the geometry of the element, and an internal bubble. As shown in Figure 4-4, each of the three external nodes is characterized by three degrees of freedom; the pressure p , and the two displacement components ($\mathbf{u} = \{u, v\}$). Instead, the internal bubble has just two degrees of freedom without considering the pressure. The inner bubble, stabilize the typical *Stokes* element formulation.

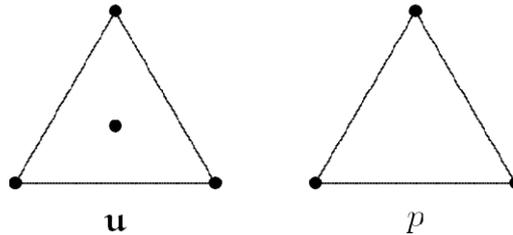


Figure 4-4 Element MINI nodal characterization

For a homogeneous neo-Hookean material we define (see for example *Bonet and Wood, 1997*) the potential energy function as:

$$\psi(\hat{\mathbf{u}}) = \frac{1}{2}\mu[I - \hat{\mathbf{C}} - d] - \mu \ln \hat{f} + \frac{\lambda}{2}\theta(\hat{f})^2 ,$$

Where $\hat{\mathbf{C}}$ is the right Cauchy-Green deformation tensor, λ and μ are positive constants, “:” represents the usual inner product for second

⁶ *Mixed finite element methods for elliptic problems, Douglas n. Arnold*

order tensor and $\hat{\mathbf{J}} = \text{Det } \hat{\mathbf{F}}$ ($\hat{\mathbf{F}}$ is the deformation gradient, see 4.2.2). Moreover, θ is a real function usually chosen as

$$\theta(J) = \ln J \quad \text{or} \quad \theta(J) = J - 1$$

Introducing the pressure-like variable (or simply pressure) $\hat{p} = \lambda\theta(\hat{J})$, the potential energy can be equivalently rewritten as function of \hat{u} and \hat{p} as:

$$\psi(\hat{u}, \hat{p}) = \frac{1}{2}\mu[I - \hat{C} - d] - \mu \ln \hat{J} + \hat{p} \theta(\hat{J}) - \frac{1}{2\lambda}\theta\hat{p}^2$$

When the domain Ω is subjected to a given load $b = b(X)$ per unit volume in the reference configuration, the total elastic energy functional reads as follows:

$$\Pi(\hat{u}, \hat{p}) = \int_{\Omega} \psi(\hat{u}, \hat{p}) - \int_{\Omega} b \cdot \hat{u}$$

4.3.3 AceGen input file: *MINI_7_int_point*

Since the element *MINI* is characterized by the mentioned above features, in the present Section, the *AceGen initialization step* is detailed. In order to create such triangular element, the already implemented standard *T1* element is used, but introducing some fundamental modifications. Therefore, a inner bubble node should be added. Moreover, an extra degree of freedom has to be added to the three external nodes, in order to take in account the pressure.

The integration loop is executed on *seven integration points* (with a 5th integration order) defined during the *SMSTemplate* function

```
<< "AceGEN`"  
SMSInitialize["Mini_7_int_point", "Environment" → "AceFEM",  
  "Mode" → "Debug"];  
SMSTemplate[  
  "SMSTopology" → "T1",  
  "SMSNoNodes" → 4,  
  "SMSDofGlobal" → {3, 3, 3, 2},  
  "SMSAdditionalNodes" → Hold[{(#1 + #2 + #3) / 3} &],  
  "SMSNodeID" → {"MIX -L", "MIX -L", "MIX -L", "D"},  
  "SMSSegments" → {{1, 2, 3}},  
  "SMSDefaultIntegrationCode" → 0,  
  "SMSNodeOrder" → {1, 2, 3, 4},  
  "SMSReferenceNodes" → {{0, 0}, {1, 0}, {0, 1}, {1/3, 1/3}},  
  "SMSSymmetricTangent" → True,  
  "SMSGroupDataNames" -> {"C2", "Qx", "Qy"}];
```

Since the number of d.o.f. per node is not the constant, the d.o.f. are divided in u_i , v_i and p_i components. Later, the shape function for the pressure (N_{ip}) and the displacement (N_i) are defined.

```
dof = SMSReal[Table[nd$$[i, "at", j], {i, SMSNoNodes},  
  {j, SMSDofGlobal[[i]]}];  
ui = {dof[[1, 1]], dof[[2, 1]], dof[[3, 1]], dof[[4, 1]]};  
vi = {dof[[1, 2]], dof[[2, 2]], dof[[3, 2]], dof[[4, 2]]};  
pi = {dof[[1, 3]], dof[[2, 3]], dof[[3, 3]]};  
ξ = SMSReal[SMSPart[ξall, IpIndex]];  
η = SMSReal[SMSPart[ηall, IpIndex]];  
wGauss = SMSReal[SMSPart[Wpall, IpIndex]];  
κ = 1 - ξ - η;  
Ni = {κ, ξ, η, κ ξ η};  
Nip = {κ, ξ, η};
```

4.4 Three-dimensional elasto-plastic element

One of the most challenging problems in computational plasticity is the development of efficient, robust, and mathematically well-founded anisotropic plasticity models capable of extending the uniaxial behavior of materials to multi-axial cyclic loading conditions.

The classical bilinear kinematically hardening-type J_2 -plasticity model is the simplest of such. Associative J_2 -plasticity is motivated by the principle of maximum plastic dissipation and the existence of potential energy functions. Possibly because of this, the currently well-known implicit return mapping schemes allow for efficient implementation of the bilinear model.

4.4.1 Introduction on inelastic strain: small displacement gradients

An **elastic** material can be defined as a material for which the stress is function only of the strain, or, equivalently in the case of no internal constraint, a material for which the strain is function only of the stress:

$$\varepsilon = \varepsilon(\sigma)$$

In particular, for an elastic material, the constitutive response is independent, for example, from the process followed in reaching the actual state [19].

Thinking of strain as the dependent variable, at a given time, the strain may in fact depend not only on the stress at the current time, but also on the stress path followed to reach the current state. Whenever this dependence is significant enough to be taken into account, the material must be regarded as **inelastic**.

Henceforth, we define a material to be inelastic if the strain is function not only of the stress, but also of other quantities, in general, indicated as *internal variables*, ξ :

$$\varepsilon = \varepsilon(\sigma, \xi)$$

The internal variables ξ can be a set of scalars, tensors, or whatever is needed to properly describe the material response. Moreover, since the internal variables ξ represent a new and extra set of variables, we need to introduce a corresponding new and extra set of equations, in general indicated as internal variable constitutive equations and represented, for example, in a quite general format through rate equations as follows:

$$\dot{\xi} = g(\sigma, \xi)$$

that is, through differential equations, that determines the rates in time (time derivatives) of such new variables.

In the realm of ***small deformations***, the material inelastic behavior is in general represented by additively decomposing the strain as follows:

$$\varepsilon = \varepsilon^e + \varepsilon^i$$

where ε^e is the elastic strain, and ε^i is the inelastic strain. The *elastic strain* can be defined as the part of the strain related to the stress through an elastic relation or as the part of the strain which is function only of the stress. The *inelastic strain* can be defined as the difference between the total strain and the elastic strain.

The inelastic strain ε^i is assumed to play the role of an internal variable; hence it could be seen as an element of ξ :

$$\xi = \{\varepsilon^i, \beta\}$$

4.4.2 Classical plasticity

Accordingly, we may assume the existence of two continuous real-valued functions, the *limit function* F and the *yield function* f . The *limit function* F distinguishes between *admissible* and *non-admissible* states, that is:

$$F \leq 0: \text{ admissible state}$$

$$F > 0: \text{ non-admissible state}$$

The *yield function* f distinguishes between *elastic* and *inelastic* states, that is:

$$f < 0: \text{ elastic state}$$

$$f \geq 0: \text{ inelastic state}$$

In plasticity theory it is also conventional to indicate the inelastic strain as $\boldsymbol{\varepsilon}^p$, instead of $\boldsymbol{\varepsilon}^i$, such that the additive strain decomposition is now rewritten as:

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p$$

The internal variables can for example assumed to be the plastic strain $\boldsymbol{\varepsilon}^p$, the back stress $\boldsymbol{\alpha}$ and the accumulated plastic strain $\overline{\boldsymbol{\varepsilon}^p}$. The back stress $\boldsymbol{\alpha}$ represents the location of the center of the yield surface, which may shift as a result of the kinematic hardening mechanism, while $\overline{\boldsymbol{\varepsilon}^p}$ is an accumulated measure of the plastic strain, often used to model an isotropic hardening mechanism.

We also assume the existence of a flow potential g , such that:

$$\dot{\boldsymbol{\varepsilon}}^p = \dot{\gamma} \frac{\partial g}{\partial \boldsymbol{\sigma}}$$

where a superpose dot indicates a time derivative. The $\dot{\gamma}$ is a non negative scalar quantity, embodying the plastic rate characteristic of the material and, therefore, it is called *plastic rate parameter*.

For the classical plasticity models $\dot{\gamma}$ is usually computed requiring the satisfaction of the consistency condition and it is often called

consistency parameter. According to the existence of an elastic region, we require that:

$$\begin{aligned}\dot{\gamma} &= 0 \quad \text{when } f < 0 \\ \dot{\gamma} &\geq 0 \quad \text{when } f \geq 0\end{aligned}$$

Since only the case of a flow rule associated with the yield function f , or briefly an associative flow rule, $g = f$ so that:

$$\dot{\mathbf{e}}^p = \dot{\gamma} \frac{\partial f}{\partial \boldsymbol{\sigma}}$$

4.4.3 J2 classical plasticity

We consider a von-Mises plasticity model with linear isotropic and kinematic hardening within a small deformation regime [18].

Splitting the stress and the strain tensor, $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$, in deviatoric and volumetric part we have:

$$\begin{aligned}\boldsymbol{\sigma} &= \mathbf{s} + p\mathbf{I} \quad \text{with} \quad p = \frac{1}{3}tr(\boldsymbol{\sigma}) \\ \boldsymbol{\varepsilon} &= \mathbf{e} + \frac{1}{3}\theta\mathbf{I} \quad \text{with} \quad \theta = tr(\boldsymbol{\varepsilon})\end{aligned}$$

where tr indicates the trace operator, while \mathbf{I} , \mathbf{s} , p , \mathbf{e} , θ and $\boldsymbol{\varepsilon}$ are, respectively, the second order identity tensor, the deviatoric and volumetric stress, the deviatoric and volumetric strain.

This model is based in the assumption that the yield function depends only on the norm of the deviatoric stress (i.e. on $\|\mathbf{s}\| = \sqrt{2J_2}$, accordingly we refer to this general class as von Mises or J2 materials). Accordingly, we have:

$$\dot{\Theta}^p = 0, \quad \text{and} \quad \dot{\boldsymbol{\varepsilon}}^p = \dot{\mathbf{e}}^p$$

Where Θ^p is the volumetric component of the plastic strain tensor, and e is the *Eulerian* strain tensor defined in terms of left *Cauchy-Green* strain tensor \mathbf{b} , as follow:

$$\mathbf{e} = \frac{1}{2}(\mathbf{I} - \mathbf{b}^{-1})$$

and consequently the \mathbf{e}^p is the *Eulerian* plastic stress tensor.

The following box summarizes the governing equations needed to define the present formulation.

$p = K\theta$	<i>Eq. 4.1</i>
$\mathbf{s} = 2G\mathbf{e}^e = 2G[\mathbf{e} - \mathbf{e}^p]$	<i>Eq. 4.2</i>
$\mathbf{\Sigma} = \mathbf{s} - \boldsymbol{\alpha}$	<i>Eq. 4.3</i>
$f = \ \mathbf{\Sigma}\ - \sigma_y(\ \dot{\mathbf{e}}^p\)$	<i>Eq. 4.4</i>
$\dot{\mathbf{e}}^p = \dot{\gamma} \frac{\partial f}{\partial \boldsymbol{\sigma}} = \dot{\gamma} \frac{\partial f}{\partial \mathbf{\Sigma}} = \dot{\gamma} \mathbf{n}$	<i>Eq. 4.5</i>
$\dot{\boldsymbol{\alpha}} = H_{kin}\dot{\mathbf{e}}^p - H_{nl}\ \dot{\mathbf{e}}^p\ \boldsymbol{\alpha}$	<i>Eq. 4.6</i>
$\dot{\gamma} \geq 0, \quad F \leq 0, \quad \dot{\gamma}F = 0$	<i>Eq. 4.7</i>

The Eq. 4.1 is the linear elastic relation between the pressure p and the volumetric part of the strain θ , K being the bulk modulus.

The Eq. 4.2 shows the relation between the deviatoric stress \mathbf{s} and the deviatoric elastic strain the \mathbf{e}^e through the shear modulus G . After the definition of the relative stress $\mathbf{\Sigma}$ (Eq. 4.3), expressed on terms of deviatoric stress strain \mathbf{s} and on the deviatoric back stress $\boldsymbol{\alpha}$, the von Mises yield function is defined for a linear isotropic hardening mechanism and a non-linear kinematic mechanism (Eq.4.4). In particular the definition of the relative stress norm and the radius of the yield surface σ_y are required. In the present paper the isotropic mechanism is defined in the simplest way, as:

$$\sigma_y = \sigma_{y,0} + H_{iso} \mathbf{e}^p$$

where $\sigma_{y,0}$ is the initial yield stress and H_{iso} is the Isotropic hardening constant, which is a material constant.

The Eq. 4.5 is the constitutive equation for the deviatoric plastic strain in the framework of associative plasticity. Note that, due to the specific form of the yield function f , the second-order tensor \mathbf{n} is:

$$\mathbf{n} = \frac{\mathbf{\Sigma}}{\|\mathbf{\Sigma}\|}$$

and it has a unit norm.

The Eq. 4.6 is the constitutive equation for the kinematic hardening mechanism, where H_{kin} and H_{nl} are material constants. This equation could be rewrite, using the Armstrong and Fredrick formulation, as:

$$\dot{\boldsymbol{\alpha}} = H_{kin}\dot{\boldsymbol{\gamma}}\mathbf{n} - H_{nl}\dot{\boldsymbol{\gamma}}\boldsymbol{\alpha}$$

The Eq. 4.7 is the *Kuhn-Tucker* conditions, which reduce the plastic problem to a constrained optimization problem.

4.4.4 Proposed formulation for J_2 plasticity

Let $\mathbf{u}_{e,n+1}$ be a vector of generalized displacement parameters of the element, \mathbf{h}_{n+1} a vector of unknowns at Gauss point level and \mathbf{h}_n is a vector of history values at Gauss point level from the previous time step. The elasto-plastic problem is defined by a *elastic strain energy function* W , a *yield condition* f and a set of *algebraic constraints* to be fulfilled at Gauss point level $\mathbf{Q}_{n+1}(\mathbf{u}_{n+1}, \mathbf{h}_{n+1}, \mathbf{h}_n)$ that have to be solved for unknowns \mathbf{h}_{n+1} when the material point is in a plastic state.

In general, the vector \mathbf{h}_{n+1} is composed of the state variables; in this case such vector contains the components of the *plastic strain vector* at time t_{n+1} , the components of the *deviatoric back stress* $\boldsymbol{\alpha}$ and the *consistency parameter* λ_{n+1} . For this reason this vector is composed by thirteen state variables. Un additional value, labeled *State*, is used like an elasto/plastic state indicator.

The set of *algebraic constraints* \mathbf{Q}_{n+1} , is composed of the corresponding set of discretized evolution equations that describe the evolution of plastic strains and hardening variables and the consistency condition $f = 0$.

Due to the fact that the evolution equations are stiff, an **implicit Euler integration** is chosen. The yield condition is evaluated for a the trial state by freezing the state variables as follows

$$\mathbf{f}_{tr} = \mathbf{f}(\mathbf{u}_{n+1}, \mathbf{h}_n)$$

The associated nonlinear equations are solved by the iterative *Newton method* using an additional iterative loop at each Gauss point. Due to the iterative loop needed to solve \mathbf{Q}_{n+1} , the variables \mathbf{h}_{n+1} depend now implicitly upon the generalized displacement parameters \mathbf{u}_{n+1} .

4.4.5 J2 plasticity with particular kinematic hardening rule

Starting from the classical formulation of J2 plasticity model, a particular hardening rule has been developed in order to represent the mechanical behavior of honeycomb core subjected to compressive loads. The abstract symbolic formulation is briefly summarized in the following table:

$$\begin{aligned} \mathbf{u} &= \{u_i N_i, v_i N_i, w_i N_i\} \\ \mathbf{D} &= \frac{\partial \mathbf{u}}{\partial X} = \frac{\partial \mathbf{u}}{\partial \xi} \frac{\partial \xi}{\partial X} \\ \boldsymbol{\varepsilon} &= \frac{1}{2} \mathbf{D} \mathbf{D}^T \\ \boldsymbol{\varepsilon}_e &= \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_p \\ W &= W(\boldsymbol{\varepsilon}_e) = \frac{\lambda}{2} \text{Tr}(\boldsymbol{\varepsilon}_e)^2 + \mu \text{Tr}(\boldsymbol{\varepsilon}_e \cdot \boldsymbol{\varepsilon}_e) \\ \boldsymbol{\sigma} &= \boldsymbol{\sigma}(\boldsymbol{\varepsilon}_e) = \frac{\partial W}{\partial \boldsymbol{\varepsilon}_e} \\ \boldsymbol{\sigma} &= \mathbf{s} + p \mathbf{I} \\ \mathbf{s} &= 2G \mathbf{e}^e = 2G[\mathbf{e} - \mathbf{e}^p] \\ \boldsymbol{\Sigma} &= \mathbf{s} - \boldsymbol{\alpha} \\ \dot{\boldsymbol{\alpha}} &= H_k \frac{\gamma}{\gamma_{limit} - \gamma} \dot{\boldsymbol{\varepsilon}}^p \\ f &= \|\boldsymbol{\Sigma}\| - \sigma_y(\|\dot{\boldsymbol{\varepsilon}}^p\|) \\ \dot{\boldsymbol{\varepsilon}}^p &= \dot{\gamma} \frac{\partial f}{\partial \boldsymbol{\sigma}} = \dot{\gamma} \frac{\partial f}{\partial \boldsymbol{\Sigma}} = \dot{\gamma} \mathbf{n} \end{aligned}$$

4.4.6 AceGen input file: Element_J2_My_Hard

In the *Appendix A* is shown the fully commented *AceGen* input file for J_2 plasticity element where a original hardening rule is defined . However, in this section is presented, after a brief overview of its main features, the structure of the input for *AceGen* package for the generation of a three-dimensional eight node finite element. In particular all the new steps needed for the generation of the code are illustrated.

- **Step 1 – Initialization**

The *spatial discretization* of domain and displacement is based on the three-dimensional isoparametric concept as presented in Section 4.3.1, for the hyperelastic hexahedral element. However, for the present element, a $3 \times 3 \times 3$ Gauss integration formulation is proposed, through the *SMSDefaultIntegrationCode* $\rightarrow 8$, which allows the definition such twenty-seven integration points (see Figure4-5).

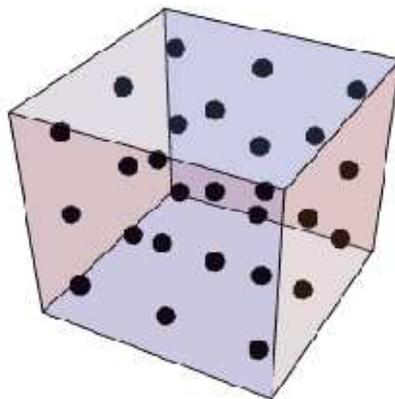


Figure 4-5 27 integration point are defined (3 x 3 x 3 Gauss integration rule)

Like the previous elements, the *AceGen* is initialized and the element characteristics necessary for the automatic derivation of formulae are defined. Through the *SMSStandardModule* command starts the definition of the user subroutines for the calculation of the

tangent matrix and the residual vector. After that the loop over the Gauss point (Ig) is initialized. All the input data like the Young modulus, the density, the Poisson ratio and the forces per unit mass and the kinematic and isotropic hardening constant (H_{Kin} and H_{Iso}) are defined.

```
<< AceGen` ;
SMSInitialize["J2_plasticity_Kin&Iso_hardening",
  "VectorLength" → 2000,
  "Environment" -> "AceFEM",
  "Mode" → "Debug"];
Ngh = 13; (* 13 state variables per integration point *)
Lgh = Ngh + 1; (*history data for the elasto/plastic state indicator*)
Lhe = Lgh es$$["id", "NoIntPoints"];
SMSTemplate["SMSTopology" → "H1", "SMSSymmetricTangent" → True,
  "SMSNoTimeStorage" → Lhe,
  (*store the components of strain tensor for postprocessing*)
  "SMSNoElementData" → 6 es$$["id", "NoIntPoints"],
  "SMSPostIterationCall" → True,
  "SMSDefaultIntegrationCode" → 8,
  "SMSGroupDataNames" -> {"E -elastic modulus", "ν -poisson ration",
    "σy0 -yield stress", "Hkin - Kinematic hardening coefficient",
    "Hiso - isotropic hardening coefficient", "ρ0 -density",
    "bX -force per unit mass X", "bY -force per unit mass Y",
    "bZ -force per unit mass Z", "λlim"}
  ];
SMSStandardModule["Tangent and residual"];
SMSDo[Ig, 1, SMSInteger[es$$["id", "NoIntPoints"]]];
```

- Step 2 – Interface to the input data of the user element subroutine and the kinematical formulation

Here the coordinates of the current integration points ξ , η , ζ the integration point weights ω_g , the coordinates of the element nodes X_i , Y_i , Z_i the current value of the displacement u_i , v_i , w_i and the material properties of the element are taken from the supplied arguments of the subroutine. All global degrees of freedom are then collected in one vector such that the proper degree of freedom

ordering is established. Moreover, during this step are defined the shape functions N_i , the interpolation of the physical coordinates X , Y , Z and the displacements u , v , w within the element. Later the *displacement gradient*, the *small strain tensor* $\boldsymbol{\varepsilon}$ is defined, and stored in the element "Data" file for the post-processing purposes.

The variable Igh , defines the location of the variables $hhgn$ at Gauss point level within the field of the element history variables ($es\$\$/hp, \dots$) for the Ig^{th} integration point.

```

E = {ξ, η, ζ} ← Table[SMSReal[es$$["IntPoints", i, Ig]], {i, 3}];
XI ← Table[SMSReal[nd$$[i, "X", j]], {i, SMSNoNodes},
           {j, SMSNoDimensions}];
En = {{-1, -1, -1}, {1, -1, -1}, {1, 1, -1}, {-1, 1, -1},
      {-1, -1, 1}, {1, -1, 1}, {1, 1, 1}, {-1, 1, 1}};
NI ← Table[1/8 (1 + ξ En[[i, 1]]) (1 + η En[[i, 2]]) (1 + ζ En[[i, 3]]), {i, 1, 8}];
X ← SMSFreeze[NI.XI];
Jg ← SMSD[X, E]; Jgd ← Det[Jg];
uI ← SMSReal[Table[nd$$[i, "at", j], {i, SMSNoNodes},
                  {j, SMSNoDimensions}]];
pe = Flatten[uI]; u ← NI.uI;
Dg ← SMSD[u, X, "Dependency" → {E, X, SMSInverse[Jg]}];
SMSFreeze[ε, 1/2 (Dg + Transpose[Dg]), "KeepStructure" → True];
SMSExport[{ε[[1, 1]], ε[[2, 2]], ε[[3, 3]], ε[[1, 2]], ε[[1, 3]], ε[[2, 3]]},
           Table[ed$$["Data", (Ig - 1) 6 + i], {i, 6}]];
Igh ← SMSInteger[(Ig - 1) Lgh];
hhgn ← Table[SMSReal[ed$$["hp", Igh + i]], {i, Lgh}];
hgn = hhgn[[1 ;; Ngh]]; state = hhgn[[Ngh + 1]];
{Em, ν, σy0, Hkin, Hiso, ρ0, bX, bY, bZ, λlim} ←
  SMSReal[Table[es$$["Data", i], {i, Length[SMSGROUPDataNames]}]];
{λ, μ} ← SMSHookeToLame[Em, ν];
bb = {bX, bY, bZ};

```

- Step 3 – Definition of the constitutive model dependent quantities.

Here the **WFQ** function is defined. It returns, the value of the input parameter *task*, the yield condition *f*, the strain energy function *W* or the evolution equations *Q* at a Gauss point. They are evaluated for the given values of the state variables. Thus, the function returns,

with respect to the supplied parameters, either trial values or the iterative values.

```

WFQ[ task_, hgt_] :=
(


|          |          |          |
|----------|----------|----------|
| hgt[[1]] | hgt[[4]] | hgt[[5]] |
| hgt[[4]] | hgt[[2]] | hgt[[6]] |
| hgt[[5]] | hgt[[6]] | hgt[[3]] |



|           |           |           |
|-----------|-----------|-----------|
| hgt[[7]]  | hgt[[10]] | hgt[[11]] |
| hgt[[10]] | hgt[[8]]  | hgt[[12]] |
| hgt[[11]] | hgt[[12]] | hgt[[9]]  |


;
  ep =  $\begin{pmatrix} \text{hgt}[[1]] & \text{hgt}[[4]] & \text{hgt}[[5]] \\ \text{hgt}[[4]] & \text{hgt}[[2]] & \text{hgt}[[6]] \\ \text{hgt}[[5]] & \text{hgt}[[6]] & \text{hgt}[[3]] \end{pmatrix}$ ;
  at =  $\begin{pmatrix} \text{hgt}[[7]] & \text{hgt}[[10]] & \text{hgt}[[11]] \\ \text{hgt}[[10]] & \text{hgt}[[8]] & \text{hgt}[[12]] \\ \text{hgt}[[11]] & \text{hgt}[[12]] & \text{hgt}[[9]] \end{pmatrix}$ ;

   $\lambda_m = \text{hgt}[[13]]$ ;



|          |          |          |
|----------|----------|----------|
| hgn[[1]] | hgn[[4]] | hgn[[5]] |
| hgn[[4]] | hgn[[2]] | hgn[[6]] |
| hgn[[5]] | hgn[[6]] | hgn[[3]] |



|           |           |           |
|-----------|-----------|-----------|
| hgn[[7]]  | hgn[[10]] | hgn[[11]] |
| hgn[[10]] | hgn[[8]]  | hgn[[12]] |
| hgn[[11]] | hgn[[12]] | hgn[[9]]  |


;
   $\epsilon_{pn} = \begin{pmatrix} \text{hgn}[[1]] & \text{hgn}[[4]] & \text{hgn}[[5]] \\ \text{hgn}[[4]] & \text{hgn}[[2]] & \text{hgn}[[6]] \\ \text{hgn}[[5]] & \text{hgn}[[6]] & \text{hgn}[[3]] \end{pmatrix}$ ;
   $\alpha_n = \begin{pmatrix} \text{hgn}[[7]] & \text{hgn}[[10]] & \text{hgn}[[11]] \\ \text{hgn}[[10]] & \text{hgn}[[8]] & \text{hgn}[[12]] \\ \text{hgn}[[11]] & \text{hgn}[[12]] & \text{hgn}[[9]] \end{pmatrix}$ ;

   $\lambda_{mn} = \text{hgn}[[13]]$ ;
  SMSFreeze[ee, e - ep, "Symmetric" -> True];
  W = Simplify[ $\lambda / 2 \text{Tr}[ee]^2 + \mu \text{Tr}[ee.ee]$ ];
  If[task == "W", Return[]];
  SMSFreeze[ $\sigma$ , SMSD[W, ee, "Symmetric" -> True], "Symmetric" -> True];
  aux2 = Hkin  $\frac{\lambda_m}{\lambda_{lim} - \lambda_m} (\epsilon_p - \epsilon_{pn})$ ;
  s =  $\sigma - 1/3 \text{IdentityMatrix}[3] \text{Tr}[\sigma]$ ;
  epDOT = ep -  $\epsilon_{pn}$ ;
  Normep = SMSqrt[Tr[ep.ep]];
   $\sigma_y = \sigma_0 + \text{Hiso} * \text{Normep}$ ;
  aux = s - aux2;
  F = SMSqrt[Total[aux aux, 2]] -  $\sigma_0$ ;
  If[task == "F", Return[]];
  R = Simplify[SMSD[F,  $\sigma$ , "Symmetric" -> True]];
  Z = ep -  $\epsilon_{pn} - (\lambda_m - \lambda_{mn}) R$ ;
  H = Hkin  $\frac{\lambda_m}{\lambda_{lim} - \lambda_m} (\text{epDOT}) - (\alpha_t - \alpha_n)$ ;
  Qg = {Z[[1, 1]], Z[[2, 2]], Z[[3, 3]], Z[[1, 2]], Z[[1, 3]], Z[[2, 3]],
  H[[1, 1]], H[[2, 2]], H[[3, 3]], H[[1, 2]], H[[1, 3]], H[[2, 3]],
  F}; If[task == "FQ", Return[]];
)

```

- Step 4 – **Elastic part**

Here, the *State* history value is used within the first global Newton iteration in order to improve the convergence radius of the global Newton iteration used to solve the nonlinear weak form of the problem at hand. The state variables are set to be the same as at the end of the previous time step.

```
WFQ["F", hgn];
iNR = SMSInteger[idata$$["Iteration"]];
SMSIF[(iNR == 1 && state == 0) || {iNR > 1 && F < 1/10^8}];
hg = hgn;
SMSEXP[Join[hgn, {0}], Table[ed$$["ht", Igh + i], {i, Lgh}]];
SMSElse[];
```

- Step 5 – **Plastic part**

For the plastic deformation ($\mathcal{F}_{tr} > 0$), the local Newton iterative loop is implemented. Independent vector of state variables hgj is first introduced. The trial value for hgj is taken to be the one at the end of the previous time step (hgn). The local system of equations \mathbf{Q} is evaluated, linearized and solved using the Newton-Raphson procedure. Consistent linearization of the global system of equations requires evaluation of the implicit dependencies among the state variables and the components of the total strain tensor ($DhgD\epsilon =$

$$\frac{h_g^{(j)}}{\partial \epsilon}). \text{ Implicit dependencies are obtained by definition } A \frac{h_g^{(j)}}{\partial \epsilon} = -\frac{\partial Q}{\partial \epsilon} \Rightarrow \frac{h_g^{(j)}}{\partial \epsilon}.$$

The *SMSVariables[ε]* function returns the true independent variables in $\boldsymbol{\epsilon}$ with the symmetry of $\boldsymbol{\epsilon}$ correctly considered.

Later Get new, independent state variables and derive strain potential for them again. In this case, the automatic differentiation

procedure will ignore the sub-iteration loop. The **type D** exception (see Section 3.2.6) is used to specify partial derivatives $\frac{\partial h_g}{\partial \epsilon}$.

The error flag is set, if the Newton iterative loop could not converge within 30 iterations. A useful function (*SMSSLUFactor*) performs full symbolic factorization of the system of linear equations and the function *SMSSLUSolve* executes a full symbolic back substitution.

```
hgj = hgn ;
SMSDo[jNR, 1, 30, 1, hgj] ;
WFQ["FQ", hgj] ;
Ag = SMSD[Qg, hgj] ;
SMSSetBreak["k"] ;
LU = SMSLUFactor[Ag] ;
Ah = SMSLUSolve[LU, -Qg] ;
hgj = hgj + Ah ;
SMSIF[Sqrt[Ah.Ah] < 1 / 10 ^ 9 ,
  DhDe = SMSLUSolve[LU, -SMSD[Qg, SMSVariables[ε], "Constant" → hgj]] ;
  SMSExport[Join[hgj, {1000 + SMSAbs[F]}],
    Table[ed$$["ht", Igh + i], {i, Lgh}]] ;
  SMSBreak[] ;
] ;
SMSIF[jNR == "29"
, SMSExport[{1, 2}, {idata$$["SubDivergence"], idata$$["ErrorStatus"]}]] ;
(*exit the sub-iterative loop if the convergence was not reached*)
SMSBreak[] ;
] ;
SMSEndDo[hgj, DhDe] ;
hg = SMSFreeze[hgj, "Dependency" → {SMSVariables[ε], DhDe}] ;
SMSEndIf[hgj] ;
```

- **Step 5 – Element tangent stiffness matrix and internal force vector**

Here the internal force vector **Rg** and the stiffness matrix **Kg** are evaluated for the final values at a Gauss point **Ig**.

The vectors, containing the quantities **Kg** and the **Rg** are exported by *SMSExport* function to the output parameters of the user element subroutine.

```

W $\mathcal{F}$ Q["W", hg];
wgp = SMSReal[es$$["IntPoints", 4, Ig]];
SMSDo [
  Rg = Jgd wgp SMSD[W -  $\rho$ 0 u.bb, pe, i, "Constant"  $\rightarrow$  hg];
  SMSExport[SMSResidualSign Rg, p$$[i], "AddIn"  $\rightarrow$  True];
  SMSDo [
    Kg = SMSD[Rg, pe, j];
    SMSExport[Kg, s$$[i, j], "AddIn"  $\rightarrow$  True];
    , {j, i, SMSNoDOFGlobal}];
    , {i, 1, SMSNoDOFGlobal}];
  SMSEndDo[]];

```

• Step 6 – Post-processing

```

SMSStandardModule["Postprocessing"];
SMSDo [Igh = SMSInteger[(Ig - 1) Lgh];
  hhg = Table[SMSReal[ed$$["ht", Igh + i]], {i, Lgh}];
  hg = hhg[[1 ;; Ngh]]; state = hhg[[Ngh + 1]]; hgn = hg;
  {Em,  $\nu$ ,  $\sigma$ 0, Hkin, Hiso,  $\rho$ 0, bX, bY, bZ,  $\lambda$ lim} =
    SMSReal[Table[es$$["Data", i], {i, Length[SMSGGroupDataNames]}]];
  { $\lambda$ ,  $\mu$ } = SMSHookeToLame[Em,  $\nu$ ];
  ei = Table[SMSReal[ed$$["Data", (Ig - 1) 6 + i]], {i, 6}];
   $\epsilon$  = 

|         |         |         |
|---------|---------|---------|
| ei[[1]] | ei[[4]] | ei[[5]] |
| ei[[4]] | ei[[2]] | ei[[6]] |
| ei[[5]] | ei[[6]] | ei[[3]] |

;
  W $\mathcal{F}$ Q[" $\mathcal{F}$ ", hg];
  SMSGPostNames = {"Sxx", "Sxy", "Sxz", "Syx", "Syy",
    "Syz", "Szx", "Szy", "Szz", "Exx", "Exy", "Exz", "Eyx",
    "Eyy", "Eyz", "Ezx", "Ezy", "Ezz", "Exxp", "Exyp",
    "Exzp", "Eyxp", "Eyyp", "Eyzp", "Ezxp", "Ezyp", "Ezzp",
    "Accumulated plastic deformation",
    "State 0-elastic 1000+f -plastic", "F"};
  SMSExport[Flatten[{ $\sigma$ ,  $\epsilon$ ,  $\epsilon$ p,  $\lambda$ m, state,  $\mathcal{F}$ }], gpost$$[Ig, #1] &];
  , {Ig, 1, SMSInteger[es$$["id", "NoIntPoints"]}];
  SMSNPostNames = {"DeformedMeshX", "DeformedMeshY",
    "DeformedMeshZ", "u", "v", "w"};
  uI = SMSReal[Table[nd$$[i, "at", j], {i, SMSNoNodes},
    {j, SMSNoDimensions}]];
  SMSExport[Table[Join[uI[[i]], uI[[i]]], {i, SMSNoNodes}], npost$$];

```

- Step 7 – **Code generation**

This is the end of the integration loop. The element source code is generated and written in *C* language. Now it can be used for the FEM analysis of *AceFem*.

```
SMSwrite[];  
[29] Consistency check - global  
[29] Consistency check - expressions  
[32] Generate source code :  
Events: 31 SMSDB-0  
[34] Final formating
```

File:	J2_plasticity_Kin&Iso_hardening.c	Size:	106 655
Methods	No.Formulae	No.Leafs	
SKR	902	15944	
SPP	84	1587	

4.5 Honeycomb-core constitutive model proposed by Mohr

The phenomenological constitutive model proposed by Mohr Doyoyo [5], was formulated for large deformations and focuses on the out-of plane behavior in the crushing and densification regime of metallic honeycomb. The central assumptions of such model are:

- The in-plane strains are small;
- The in plane stresses are small as compared to the outofplane stresses;
- The concept of plateau stress applies, i.e. the assumption of constant normal and shear stress plateaus in the crushing regime provides a satisfactory approximation of the complex stress-strain response.

The first step is the definition of the orthotropy axes labeled W , L and T , as defined in Section 2.2.1.

The yield surface is conical in the shear-normal stress space, while the direction of plastic flow was assumed to be parallel to the direction of the compressive principal stress.

The experimental observed concept of crushing envelope suggests the following three-dimensional yield surface to describe the boundary of the elastic domain under large deformation:

$$f(\sigma, s) = \frac{\sigma_{TT}}{s_{TT}} + \left[\left(\frac{\tau_{TW}}{s_{TW}} \right)^2 + \left(\frac{\tau_{TL}}{s_{TL}} \right)^2 \right]^{m/2} - 1 = 0$$

where m is the shear exponent,

The vector $s^T = \{s_{TT}, s_{TW}, s_{TL}\}$ denotes the deformation resistance. It is constant in the crushing regime, but increases in the densification regime. Significant strain hardening is characteristic for the densification regime, where cell wall contact within the

microstructure continually rises the load carrying capacity of the honeycomb. The evolution equation for the deformation resistance vector s reads as:

$$ds = d\lambda \mathbf{q}$$

where we have the initial condition

$$s(\varepsilon_{TT}^p = 0) = 0$$

and

$$\begin{cases} \mathbf{q} = 0 & \text{for } \varepsilon_{TT}^p \geq \varepsilon_d \\ \mathbf{q} = \frac{h_d}{1 + \varepsilon_{TT}^p} \frac{r_3}{s_{TT}^0} s^0 & \text{for } \varepsilon_{TT}^p < \varepsilon_d \end{cases}$$

where h_d represent the densification module and ε_d is the densification strain. The factor $\frac{h_d}{1 + \varepsilon_{TT}^p}$ is introduced based on the results from uniaxial experiments.

Such model need a model calibration which involves two steps:

1. Fit of the field surface to the experimental data (determination of s^0 and m);
2. Fit of the flow rule to the experimental data.

5. AceFem: Numerical simulation and results

5.1 Introduction

All the numerical simulations are performed by the *AceFem* package. As posted in chapter 3, *AceFem* is a general finite element environment designed to solve multi-physics and multi-field problems.

In this chapter are presented the results achieved from the numerical simulations performed on the elements previously generated by *AceGen* package. The description of such elements is presented in chapter 4.

In particular, the behavior of such generated elements when they are subjected to a compressive load is analyzed in terms strain-strength relationship.

Moreover during the analysis phase all the data needed to be checked out, could be printed during the flow of the analysis (Figure 5-2).

Step = 1	Yield func. = -0.412122	$\sigma_z = -0.72$	$ez = -0.008$	$ezzp = 0.$
T/ $\Delta T=1./1.$ $\lambda/\Delta\lambda=0.01/0.01$ $\ \Delta a\ /\ \Psi\ =5.931 \times 10^{-15}/1.49884 \times 10^{-14}$ Iter/Total=3/3 Status=0/(Convergence)				
Step = 2	Yield func. = 0.0000859729	$\sigma_z = -1.22485$	$ez = -0.016$	$ezzp = -0.00239055$
T/ $\Delta T=2./1.$ $\lambda/\Delta\lambda=0.02/0.01$ $\ \Delta a\ /\ \Psi\ =1.3503 \times 10^{-7}/2.59612 \times 10^{-14}$ Iter/Total=4/7 Status=0/(Convergence)				
Step = 3	Yield func. = 0.00125846	$\sigma_z = -1.22629$	$ez = -0.024$	$ezzp = -0.0103746$
T/ $\Delta T=3./1.$ $\lambda/\Delta\lambda=0.03/0.01$ $\ \Delta a\ /\ \Psi\ =0.0000855233/9.16002 \times 10^{-10}$ Iter/Total=3/10 Status=0/(Convergence)				
Step = 4	Yield func. = 0.00225022	$\sigma_z = -1.2275$	$ez = -0.032$	$ezzp = -0.0183611$
T/ $\Delta T=4./1.$ $\lambda/\Delta\lambda=0.04/0.01$ $\ \Delta a\ /\ \Psi\ =0.0000868457/9.59356 \times 10^{-10}$ Iter/Total=3/13 Status=0/(Convergence)				
Step = 5	Yield func. = 0.00326149	$\sigma_z = -1.22874$	$ez = -0.04$	$ezzp = -0.0263473$
T/ $\Delta T=5./1.$ $\lambda/\Delta\lambda=0.05/0.01$ $\ \Delta a\ /\ \Psi\ =0.0000881904/1.00491 \times 10^{-9}$ Iter/Total=3/16 Status=0/(Convergence)				
Step = 6	Yield func. = 0.0042933	$\sigma_z = -1.23$	$ez = -0.048$	$ezzp = -0.0343333$
T/ $\Delta T=6./1.$ $\lambda/\Delta\lambda=0.06/0.01$ $\ \Delta a\ /\ \Psi\ =0.0000895662/1.05301 \times 10^{-9}$ Iter/Total=3/19 Status=0/(Convergence)				
Step = 7	Yield func. = 0.00534628	$\sigma_z = -1.23129$	$ez = -0.056$	$ezzp = -0.042319$
T/ $\Delta T=7./1.$ $\lambda/\Delta\lambda=0.07/0.01$ $\ \Delta a\ /\ \Psi\ =0.000090974/1.10378 \times 10^{-9}$ Iter/Total=3/22 Status=0/(Convergence)				
Step = 8	Yield func. = 0.00642109	$\sigma_z = -1.23261$	$ez = -0.064$	$ezzp = -0.0503043$
T/ $\Delta T=8./1.$ $\lambda/\Delta\lambda=0.08/0.01$ $\ \Delta a\ /\ \Psi\ =0.0000924149/1.15744 \times 10^{-9}$ Iter/Total=3/25 Status=0/(Convergence)				

Figure 5-2 Run time debugging example

5.1.2 Example of AceFem input file

In this Section the *AceFem* input file, related to the numerical simulation of a static compressive tests, are briefly described

In this Section, like widely explained in Section 3.3.4, a brief description of the *AceFem* analysis procedure is presented. In order to describe such procedure, the main steps of the *AceFem* input file are presented. In *Appendix B* is reported such full commented input file. Despite their similarities, two different inputs are created in order to execute the required tests on 2D and 3D problems.

5.2 Numerical analysis on 2D domain

Let's consider a simple bidimensional domain Ω^2 which represents the cross section of the *Nomex*® honeycomb core.

Using the honeycomb coordinates system (W, T) , is defined the a rectangular material body whose reference configuration is :

$$\Omega^2 = (0, 0) \times (L, H).$$

Such coordinate system is presented in Section 2.2.1.

The executed numerical simulation is addressed to carry out a uniaxial static compressive test, under displacement control, by imposing a increasing essential (*Dirichlet*) boundary condition on the upper side of the domain.

Such condition enforce a $\Delta \mathbf{L}$ displacement of the upper edge ($T=H$) of the domain. As shown in the Figure 5-3, the bottom edge of domain is clamped only along W -direction while the left side is clamped along T -direction.

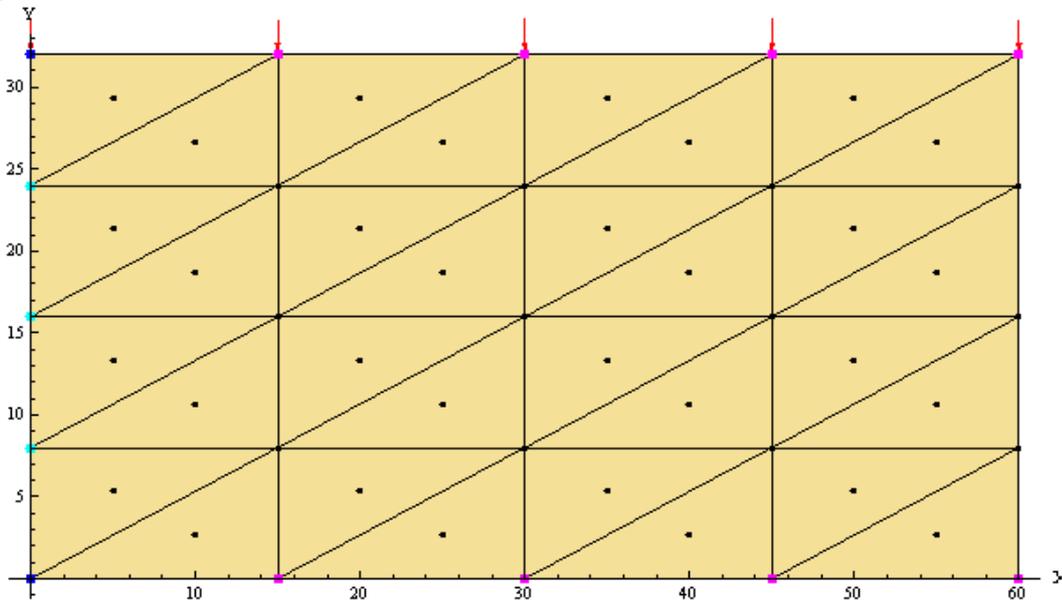


Figure 5-3 Domain discretized using elements MINI

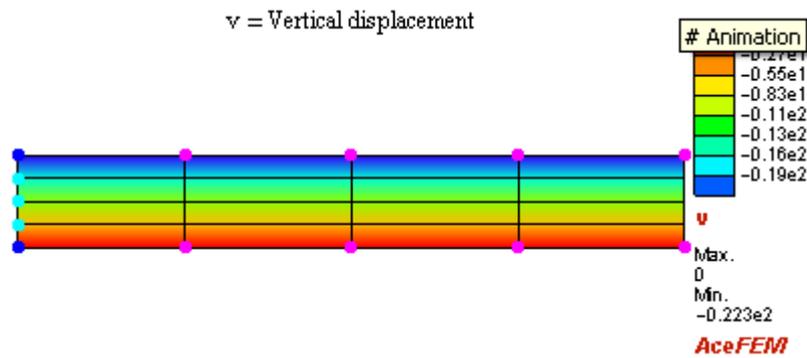


Figure 5-4 Deformed mesh after the numerical analysis

Following the AceFem procedure all the input data have to be previously defined.

Starting from the material specimens of *Nomex*® honeycombs, the input values are:

$$L = 60 \text{ mm}; \quad \rho = 1;$$

$$H = 32,2 \text{ mm}; \quad E = 90 \text{ Mpa}; \quad \Delta L = 0,7 \times H = -22,4 \text{ mm}$$

Such constants are defined and calibrated starting from the experimental investigation presented in Section 2.4.2.

5.2.1 Hyperelastic element

In this section the domain is discretized by the 2D hyperelastic elements described in section 4.2.4. The compressive behavior is analyzed in terms of his compressive strain-strength relationship.

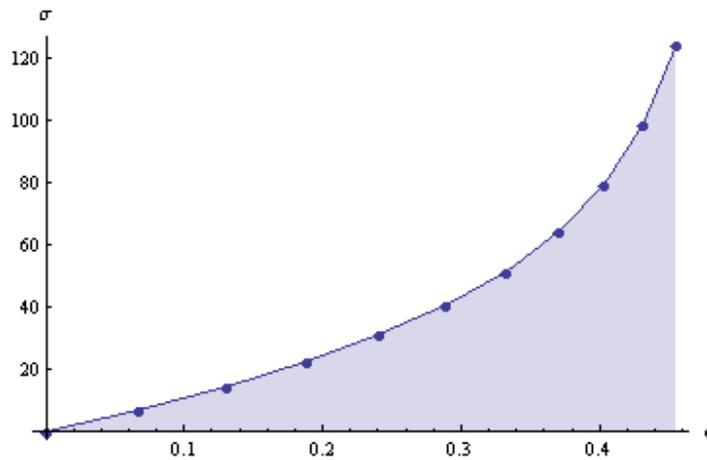


Figure 5-5 Element Hyperelastic 2D: stress-strain response

As expected, the Figure 5-5 shows the non-linear elastic behavior of this 2D hyperelastic element

5.2.2 Element Mini

In this section the domain is discretized by the 2D element *MINI* described in section 4.3. The compressive behavior is analyzed in terms of his compressive properties (strain-strength relationship).

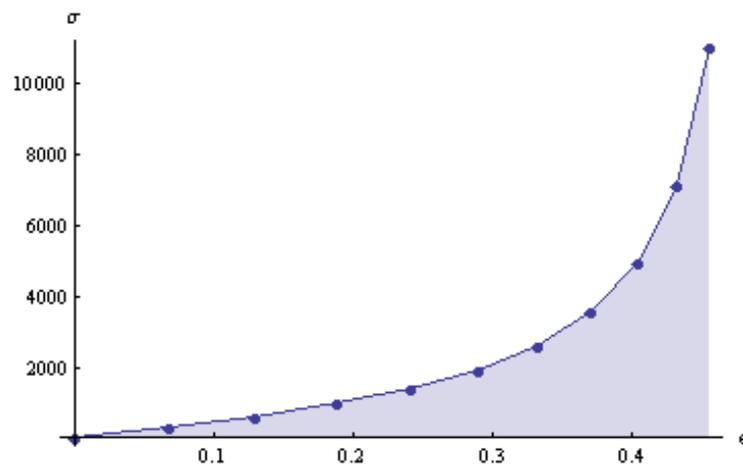


Figure 5-6 Element MINI stress-strain response

5.3 Numerical analysis on 3D domain

Let's consider a tridimensional domain Ω^3 which represents the continuum model of the *Nomex*® honeycomb core.

Using the honeycomb coordinates system (W, L, T) , is defined the a hexahedral solid element whose reference configuration is :

$$\Omega^3 = (0, 0, 0) \times (L_W, L_L, H).$$

Such coordinate system is presented in Section 2.2.1.

Like the 2D case, this 3D numerical simulation is addressed to carry out a uniaxial static compressive test, under displacement control, by imposing a increasing essential (*Dirichlet*) boundary conditions on the upper side of the domain Ω^3 . Such boundary condition is applied along T -direction, in order to reproduce the out-of-plane compressive behavior of such model. Such condition enforce a ΔL displacement of the upper side ($T=H$) of the domain.

Figure 5-7 tries to summarize the above described boundary conditions.

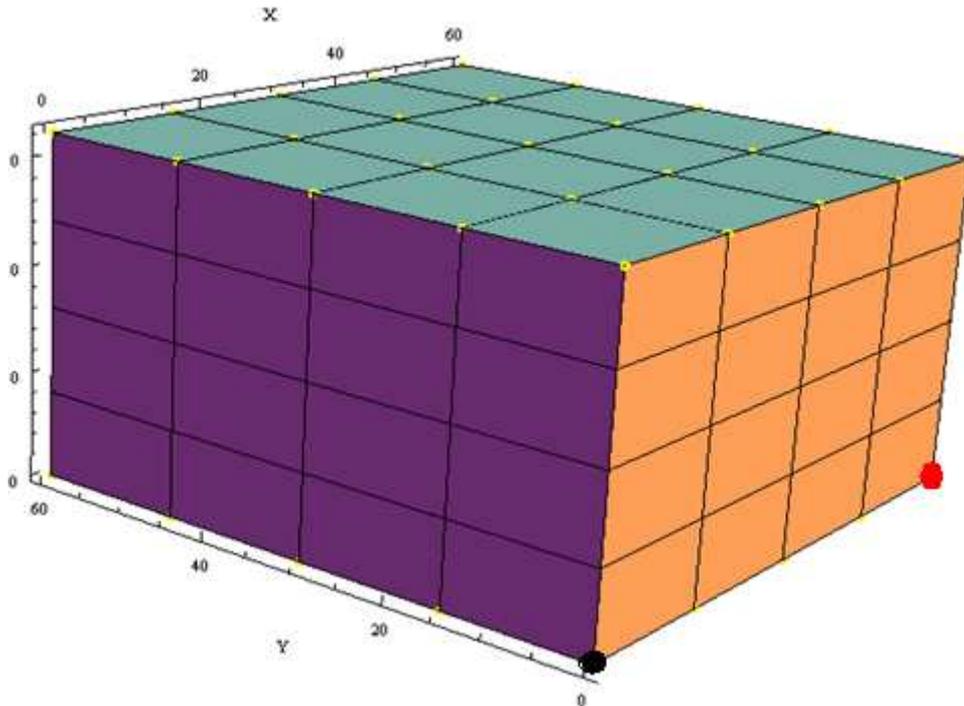


Figure 5-7 Domain Ω^3 a geometry and boundary conditions

The bottom side of domain ($T=0$) is clamped only along T direction, except the axis origin ($W=L=T=0$) where all the d.o.f. are clamped. This point is pointed out with a red dot on the previous figure. In order to avoid rotational movements around the origin of axes, a point, whose coordinates are $\mathbf{P} = \{Lw, 0, 0\}$ has been bounded along the transverse L direction. This time the point \mathbf{P} is pointed out with a black dot on the previous figure. The yellow points on the domain show the points where the Dirichlet boundary conditions are applied.

The input data are defined and calibrated starting from the experimental investigation presented in Section 2.4.2. Starting from the material specimens of *Nomex*® honeycombs, the input values are:

$$Lw = 60 \text{ mm}; \quad L_L = 60 \text{ mm}; \quad \rho = 1;$$

$$H = 32,2 \text{ mm}; \quad E = 90 \text{ Mpa}; \quad \Delta L = 0,8 \times H = -258 \text{ mm}$$

$$\sigma_{y0} = \text{Yield stress} = 1.2 \text{ Mpa}$$

$$H_{iso} = \text{isotropic hardening coefficient} = 0;$$

$$H_{kin} = \text{kinematical hardening coefficient} = 10;$$

$$\gamma_{lim} = 1$$

The elements behavior is analyzed in terms of his out-of-plane compressive strain-strength relationship.

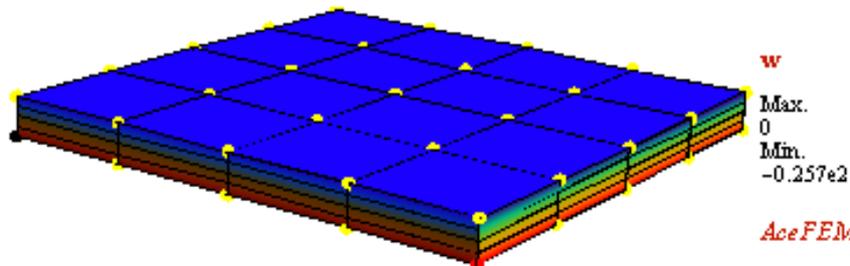


Figure 5-8 Deformed mesh after the numerical analysis

5.3.1 Hyperelastic 3D element

In this section are shown the result, in terms of out-of-plane strain-stress relationship when the domain has been discretized by the 3D hyperelastic elements described in section 4.2.5.

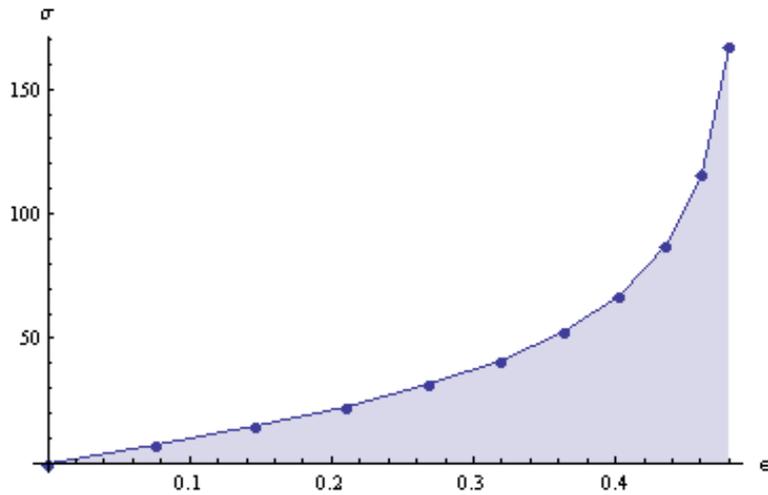


Figure 5-9 Hyperelastic 3D element: out of plane strain-stress relationship.

As expected, Figure 5-9 shows the non-linear elastic behavior of this 3D hyperelastic element. This result is very similar to the obtained result on the 2D element.

5.3.2 Classical J_2 plasticity element

In this section are shown the result, in terms of out-of-plane strain-stress relationship when the domain has been discretized by the 3D elasto-perfectly plastic elements described in section 4.4.4.

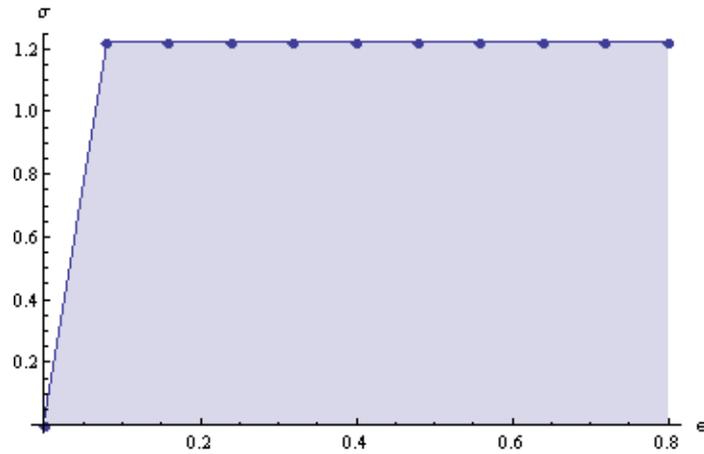


Figure 5-10 elasto-perfectly plastic 3D: out of plane strain-stress relationship

Such element represents an important step of our multi-step analysis, since the strain-stress relationship shows a crushing regime. It can be pointed out the achievement of the crush strength, which could corresponds to the constant plateau stress, obtained during the experimental tests.

However, the develop of more suitable models is required.

5.3.3 *J2* plasticity with particular kinematic hardening rule

In this section are shown the result, in terms of out-of-plane strain-stress relationship when the domain has been discretized by the 3D elasto-plastic element described in section 4.2.5.

The purpose of such model was to define a original constitutive laws able to well reproduce the strain-stress relationship described in chapter 2.3. For this reason such implemented element represents the most important step of our multi-step process. In fact this implemented element shows a stress-strain curve which is able to reproduce the main crushing phenomena of honeycomb core materials.

The following figure shows the strain-stress curve obtained during the analysis. in such curve can be pointed out the:

- the ***elastic regime*** until the compressive strength is reached;
- the ***crushing regime*** at nearly constant plateau stress (crush strength);
- the ***densification regime***, where the cellular structure should be fully compacted.

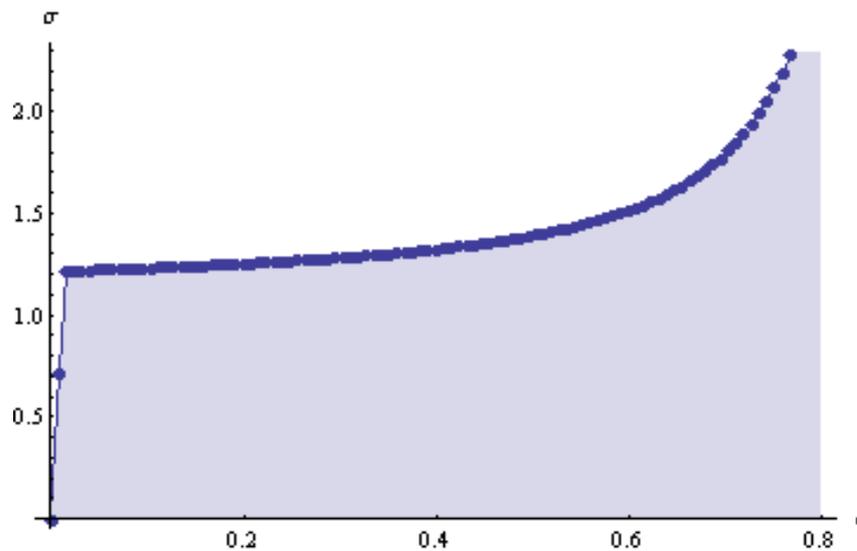


Figure 5-11 original elasto-plastic 3D: out of plane strain-stress relationship

6. Conclusions and future work

6.1 Developed activity

The activity herein presented shows the undoubted potential of the combination of *AceGen* code-generator with its sibling FE environment *AceFem*. The great advantage of this package is that in a few lines of code the user is able to generate an original numerical model. Such results is made possible by combining several techniques as the symbolic and algebraic capabilities of *Mathematica*, the automatic code generation technique and the simultaneous optimization of expression and theorem proving by a stochastic evaluation of the expression.

During the present research work several numerical models have been presented, like:

- *Hyperelastic element*
 - 2D element: quadrilateral element with four nodes;
 - 3D element: hexahedral element with eight nodes;

- *Element MINI*
 - 2D element: triangular element enriched by an inner bubble, where seven integration points are defined;
- *Elasto-plastic 3D element*
 - Traditional J2 plasticity element;
 - J2 plasticity with particular kinematic hardening rule

The generation of such models was oriented to the ambitious modeling of the crushing behavior of the honeycomb core materials, by using a *multi-step* approach. Such multi-step approach has proven a successful method to achieve the aim of the thesis. In fact, the results of the numerical simulations executed on the above described numerical models are really encouraging, since their behavior reflects the prospective and hoped proceeding. Moreover, the results of each performed steps gradually approximates the real behavior of honeycomb core material.

6.2 Contribution to the modeling of honeycomb core

About the contribution to the modeling of honeycomb core, this research work propose a way to avoid the troubles related to the traditional modeling approach of honeycomb core. Usually, as presented in Section 2.5, the honeycomb structure is modeled by using detailed models, where each cell wall is modeled by several shell elements. Such method strongly depends on the discretization of the domain. In fact, under large deformation, reliable modeling of the core behavior requires very fine meshes on the micro-structural level. For these reasons this kind of approach is really time-consuming, and often such modeling exceeds the capabilities of the state of art for high performance FE-codes.

During the present work the way to avoid such troubles is to represent the honeycomb core by using a continuum model. The use of a continuum model provides an attractive solution to this problem. Exploiting this technique, we can replace the great number of shell elements for the sandwich core by a single reduced-integration solid element, where we use a constitutive model for honeycombs to perform the stress update at integration point. Thus, the size of the computational problem is strongly reduced.

The benefit of the present model in different applications must be shown in the future.

6.3 Future work

Several directions for future research could be followed starting from the results of this thesis. In fact, starting from the presented multi-step approach, new steps could be performed in order to better describe the complex behavior of the sandwich honeycomb materials. In particular these future steps could be:

- The modeling of the two face-sheets which bound the honeycomb core;
- The modeling of the *core-to-facing bonding adhesives*. The adhesive between the faces and the core must be able to transfer the shear forces between the faces and the core.
- The numerical investigation on the behavior of sandwich panels subjected to different combination of loads (shear, bending or mixed loading);
- The strain rate effects, in particular such effects on the macroscopic level are expected at high loading velocities where the lateral inertia of the folding cell walls influences the folding mechanism of the microstructure. The applicability of plateau stresses must be carefully examined for high loading velocities.
- Tension tests, since all the models proposed do not provide a reliable prediction when tensile stresses are applied along T-direction;
- A study on deformation-induced anisotropy and evolving elastic moduli;

7. Bibliography and reference works

- [1] *A study on composite honeycomb sandwich panel structure*, Meifeng He, Wenbin Hu.
- [2] *Analysis of structural performance of sandwich plates with foam-filled aluminum hexagonal honeycomb core*, Vyacheslav N. Burlayenko a,b, Tomasz Sadowski.
- [3] *Large Deformation Simulation of Anisotropic material*, R.H.W. ten Thije, R. Akkerman, J. Huétink.
- [4] *Experimental Characterization Of Strain Rate Effects And Impact Behaviour Of Phenolic Sandwich Structures*, A. Zinno, D. Asprone, A. Prota and C.E. Bakis.
- [5] *Large plastic deformation of metallic honeycomb: orthotropic rate-independent constitutive model*, Dirk Mohr, Mulalo Doyoyo.
- [6] *Multi-scale finite-strain plasticity model for stable metallic honeycombs incorporating microstructural evolution*, Dirk Mohr.
- [7] *Nucleation and propagation of plastic collapse bands in aluminum honeycomb*, Dirk Mohr and Mulalo Doyoyo.

- [8] *Mechanical properties of Nomex material and Nomex honeycomb structure*, Choon Chiang Foo *, Gin Boay Chai, Leong Keey Seah.
- [9] *Modelling of composite sandwich structures with honeycomb core subjected to high-velocity impact*, Brenda L. Buitrago, Carlos Santiuste, Sonia Sánchez-Sáez, Enrique Barbero.
- [10] *Modelling of low-energy/low-velocity impact on Nomex honeycomb sandwich structures with metallic skins*, B. Castaniè ,C. Bouvet, Y. Aminanda, J.J. Barrau, P. Thevenet.
- [11] *Virtual testing of sandwich core structures using dynamic finite element simulations*, Sebastian Heimbs.
- [12] *Stability of Some Finite Element Methods for Finite Elasticity Problems* F. Auricchio , L. Beirao da Veiga, C. Lovadina and A. Reali.
- [13] *Mixed Finite Element Methods For Elliptic Problems*, Douglas N. Arnold.
- [14] *The out-of-plane compressive behavior of metallic honeycombs*, F. Coté, V.S. Deshpande, N.A. Fleck , A.G. Evans
- [15] *Multi-language and Multi-environment Generation of Nonlinear Finite Element Codes*, Korelc J., (2002),
- [16] *Automatic generation of finite-element code by simultaneous optimization of expressions*, Korelc, J. (1997),
- [17] *Symbolic Approach in Computational Mechanics and its Application to the Enhanced Strain Method*, Doctoral Dissertation. Korelc, J. (1996).
- [18] *Object-Oriented Non-Linear Finite Element Analysis: Application to J2 Plasticity*, Ph. Menétrey and Th. Zimmermann
- [19] *The Mathematical Theory of Plasticity*, Hill (1950).
- [20] *Non-uniform Hardening Constitutive Model for Compressible Orthotropic Materials with Application to Sandwich Plate Cores*, Zhenyu Xue¹, Ashkan Vaziri and John W. Hutchinson.
- [21] *AceGen Manual*, Korelc, J.

- [22] *AceFem Manual*, Korelc Jože.
- [23] *Computational methods for plasticity theory and applications*, Souza Neto.
- [24] *HexwebTM Honeycomb Sandwich Design Technology*
- [25] *Cellular Solids – Structure and Properties*. Gibson L.J. and Ashby.
- [26] *Buckling of Honeycombs under In-plane Biaxial Stresses*, Zhang J. and Ashby.
- [27] *An Approximate Analysis of the Collapse of Thin Cylindrical Shells under Axial Loading*, Alexander J.M.
- [28] *Hexagonal Cell Structures under Post-Buckling Axial Load*, McFarland R.K.
- [29] *Implementation of automatic differentiation tools*, Bischof C., Hovland P. and Norris B.,

Appendix A

Three Dimensional, Elasto-Plastic Element

- Here the *AceGen* and the *AceFEM* interface variables are initialized.

```

<< AceGen`;
SMSInitialize["Element_J2_My_hard",
  "VectorLength" -> 2000, "Environment" -> "AceFEM", "Mode" -> "Debug"];
Ngh = 13; (* 13 state variables per integration point *)
Lgh = Ngh + 1;
(* add an additional history data for the elasto/plastic state indicator*)
Lhe = Lgh es$$["id", "NoIntPoints"];
SMSTemplate["SMSTopology" -> "H1", "SMSSymmetricTangent" -> True,
  "SMSNoTimeStorage" -> Lhe,
  (*store the components of strain tensor for postprocessing*)
  "SMSNoElementData" -> 6 es$$["id", "NoIntPoints"],
  "SMSPostIterationCall" -> True,
  "SMSDefaultIntegrationCode" -> 8,
  "SMSGroupDataNames" ->
  {"E -elastic modulus", "v -poisson ration", "oy0 -yield stress",
   "Hkin - Kinematic hardening coefficient", "Hiso - isotropic hardening coefficient",
   "rho0 -density", "bX -force per unit mass X",
   "bY -force per unit mass Y", "bZ -force per unit mass Z", "lambda"}
];
SMSStandardModule["Tangent and residual"];
time=0 variable= 0 = {}

```

- Element is numerically integrated by one of the built-in standard numerical integration rules (see Numerical Integration). This starts the loop over the integration points, where ξ, η, ζ are coordinates of the current integration point.

```

SMSDo[Ig, 1, SMSInteger[es$$["id", "NoIntPoints"]]];
E = {xi, eta, zeta} Table[SMSReal[es$$["IntPoints", i, Ig]], {i, 3}];

(* es$$["IntPoints", i, j] =
  coordinates and weights of the numerical integration points, pag 128:
  xi = es$$["IntPoints", 1, i],
  eta = es$$["IntPoints", 2, i],
  zeta = es$$["IntPoints", 3, i],
  wi = es$$["IntPoints", 4, i].
*)

```

- Standard isoparametric interpolation of coordinates and displacements within the element domain is performed here. The $N_i = 1/8 (1 + \xi \xi_i)(1 + \eta \eta_i)(1 + \zeta \zeta_i)$ is the shape function for i -th node where $\{\xi_i, \eta_i, \zeta_i\}$ are the coordinates of the node.

```

XI = Table[SMSReal[nd$$[i, "X", j]], {i, SMSNoNodes}, {j, SMSNoDimensions}];
En = {{-1, -1, -1}, {1, -1, -1}, {1, 1, -1}, {-1, 1, -1},
      {-1, -1, 1}, {1, -1, 1}, {1, 1, 1}, {-1, 1, 1}};

```

```

NI = Table[1/8 (1 + xi En[[i, 1]]) (1 + eta En[[i, 2]]) (1 + zeta En[[i, 3]]), {i, 1, 8}];
X = SMSFreeze[NI.XI]; Jg = SMSD[X, E]; Jgd = Det[Jg];

```

- Definition of the small strain tensor ϵ :

$$\mathbf{u} = \{u_i N_i, v_i N_i, w_i N_i\}$$

$$\mathbf{D} = \frac{\partial \mathbf{u}}{\partial \mathbf{X}}$$

$$\epsilon = \frac{1}{2} (\mathbf{D} + \mathbf{D}^T)$$

```

uI = SMSReal[Table[nd$$[i, "at", j], {i, SMSNoNodes}, {j, SMSNoDimensions}]];
pe = Flatten[uI]; u = NI.uI;
Dg = SMSD[u, X, "Dependency" -> {E, X, SMSInverse[Jg]}];
(* "Dependency" = define partial derivatives of external data object (SMSEExternalF)
  with respect to given auxiliary variables (for the detailed syntax see SMSFreeze) *)

```

The *SMSFreeze* function is used here in order to enable differentiation with respect to the elements of the strain tensor later on when consistent linearization is performed ($\frac{\partial Q_g}{\partial \epsilon}$). The "KeepStructure"→True option is necessary here to preserve the symmetry of ϵ .

```
time=1 variable= 100 ≡ {η, x₂}

SMSFreeze[ε, 1 / 2 (Dg + Transpose[Dg]), "KeepStructure" → True];

SMSExport[{ε[[1, 1]], ε[[2, 2]], ε[[3, 3]], ε[[1, 2]], ε[[1, 3]], ε[[2, 3]]},
  Table[ed$$["Data", (Ig - 1) 6 + i], {i, 6}]];

Igh = SMSInteger[(Ig - 1) Lgh];
hhgn = Table[SMSReal[ed$$["hp", Igh + i]], {i, Lgh}];
hgn = hhgn[[1 ;; Ngh]]; state = hhgn[Ngh + 1];

{Em, ν, σy0, Hkin, Hiso, ρ0, bX, bY, bZ, λlim} =
  SMSReal[Table[es$$["Data", i], {i, Length[SMSGroupDataNames]}]];
{λ, μ} = SMSHookToLame[Em, ν];
bb = {bX, bY, bZ};

time=2 variable= 200 ≡ {Hiso}
```

```
WFQ["F", hgn];
iNR = SMSInteger[idata$$["Iteration"]];
SMSIf[(iNR == 1 && state == 0) || (iNR > 1 && F <  $\frac{1}{10^8}$ )];
```

- This is elastic part of the elasto-plastic formulation. The state variables are set to be the same as at the end of the previous time step.

```
hg = hgn;
SMSExport[Join[hgn, {0}], Table[ed$$["ht", Igh + i], {i, Lgh}]];

SMSElse[];
```

- This is plastic part of the elasto-plastic formulation.

Independent vector of state variables $h_g^{(j)}$ is first introduced. The trial value for $h_g^{(j)}$ is taken to be the one at the end of the previous time step ($h_{g,n}$). The local system of equations Q is evaluated, linearized and solve using the Newton-Raphson procedure. Consistent linearization of the global system of equations requires evaluation of the implicit dependencies among the state variables and the components of the total strain tensor ($DhgD\epsilon := \frac{h_g^{(j)}}{\partial \epsilon}$). Implicit dependencies are obtained by definition $A \frac{h_g^{(j)}}{\partial \epsilon} = -\frac{\partial Q}{\partial \epsilon} \implies \frac{h_g^{(j)}}{\partial \epsilon}$. The *SMSVariables[ε]* function returns the true independent variables in ϵ with the symmetry of ϵ correctly considered.

```
hgj = hgn;
SMSDo[jNR, 1, 30, 1, hgj];
WFQ["FQ", hgj];
Ag = SMSD[Qg, hgj];
SMSSetBreak["k"];
```

```
time=4 variable= 300 ≡ { $\sqrt[278]{F|F}$ }
```

Forward differentiation of 74 variables.

```
time=6 variable= 400 ≡ {Qg5;εp33, Z3+1;εp33, Z1+3;εp33}
```

```

LU = SMSLUFactor[Ag];
Δh = SMSLUSolve[LU, -Qg];
hgj ← hgj + Δh;
SMSIf[Sqrt[Δh.Δh] < 1 / 10^9,
  (*the operator =
  is necessary here because the δhε will be exported out from the loop *)
  DhDe = SMSLUSolve[LU, -SMSD[Qg, SMSVariables[ε], "Constant" → hgj]];
  (*The values of the state variables
  are stored back to the history data of the element.*)
  SMSExport[Join[hgj, {1000 + SMSAbs[ $\mathcal{F}$ ]}], Table[ed$$["ht", Igh + i], {i, Lgh}]];
  (*exit the sub-iterative loop when the local equations are satisfied*)
  SMSBreak[];
];
SMSIf[jNR == "29"
  , SMSExport[{1, 2}, {idata$$["SubDivergence"], idata$$["ErrorStatus"]}];
  (*exit the sub-iterative loop if the convergence was not reached*)
  SMSBreak[];
];
SMSEndDo[hgj, DhDe];

```

- Get new, independent state variables and derive strain potential for them again. In this case, the automatic differentiation procedure will ignore the sub-iteration loop. The type D exception is used to specify partial derivatives $\frac{\partial h_g}{\partial \epsilon}$ (Exceptions in Differentiation).

```

hg ← SMSFreeze[hgj, "Dependency" → {SMSVariables[ε], DhDe}];
SMSEndIf[hg];

```

time=19 variable= 1000 ≡ {hg₂}

- Here the Gauss point contribution to the global residual R_g and the global tangent matrix K_g are evaluated and exported to the output parameters of the user subroutine. The strain energy function is first evaluated for the final value (valid for both elastic and plastic case) of the state variables h_g .

```

W $\mathcal{F}$ Q["W", hg];
wgp ← SMSReal[es$$["IntPoints", 4, Ig]];
SMSDo[
  Rg = Jgd wgp SMSD[W - ρ0 u.bb, pe, i, "Constant" → hg];
  SMSExport[SMSResidualSign Rg, p$$[i], "AddIn" → True];
  SMSDo[
    Kg = SMSD[Rg, pe, j];
    SMSExport[Kg, s$$[i, j], "AddIn" → True];
    , {j, i, SMSNoDOFGlobal}};
    , {i, 1, SMSNoDOFGlobal}};

```

Backward Differentiation of 69 variables.

Backward Differentiation of 75 variables.

time=27 variable= 1100 ≡ { $\bar{\epsilon}_{1,2}(hg_1|Rg)$ }

- This is the end of the numerical integration loop.

```

SMSEndDo[];

```

```

SMSStandardModule["Postprocessing"];
SMSDo [
  Igh ← SMSInteger[(Ig - 1) Lgh];
  hhg ← Table[SMSReal[ed$$["ht", Igh + i]], {i, Lgh}];
  hg = hhg[[1 ;; Ngh]]; state = hhg[[Ngh + 1]]; hgn = hg;
  {Em, ν, σy0, Hkin, Hiso, ρ0, bX, bY, bZ, λlim} ←
    SMSReal[Table[es$$["Data", i], {i, Length[SMSGROUPDataNames]}]];
  {λ, μ} ← SMShookeToLame[Em, ν];
  ei ← Table[SMSReal[ed$$["Data", (Ig - 1) 6 + i]], {i, 6}];

  e = 

|         |         |         |
|---------|---------|---------|
| ei[[1]] | ei[[4]] | ei[[5]] |
| ei[[4]] | ei[[2]] | ei[[6]] |
| ei[[5]] | ei[[6]] | ei[[3]] |

;

  W $\mathcal{F}$ Q[" $\mathcal{F}$ ", hg];
  SMSGPostNames = {"Sxx", "Sxy", "Sxz", "Syx", "Syy", "Syz", "Szx", "Szy", "Szz",
    "Exx", "Exy", "Exz", "Eyx", "Eyy", "Eyz", "Ezx", "Ezy", "Ezz",
    "Exxp", "Exyp", "Exzp", "Eyxp", "Eyy $\mathcal{P}$ ", "Eyz $\mathcal{P}$ ", "Ezxp", "Ezyp", "Ezz $\mathcal{P}$ ",
    "Accumulated plastic deformation", "State 0-elastic 1000+f -plastic", "F"};
  SMSExport[Flatten[{σ, ε, ep, λm, state,  $\mathcal{F}$ }], gpost$$[Ig, #1] &];
  , {Ig, 1, SMSInteger[es$$["id", "NoIntPoints"]}];
  SMSNPostNames = {"DeformedMeshX", "DeformedMeshY", "DeformedMeshZ", "u", "v", "w"};
  uI ← SMSReal[Table[nd$$[i, "at", j], {i, SMSNoNodes}, {j, SMSNoDimensions}]];
  SMSExport[Table[Join[uI[[i]], uI[[i]], {i, SMSNoNodes}], npost$$];
  • Here the element source code is written in "ExamplesElastoPlastic3D.c" file.

  SMSWrite[];

[31] Consistency check - global
[31] Consistency check - expressions
[34] Generate source code :

Events: 31 SMSDB-0

[36] Final formatting

```

File:	Element_J2_My_hard.c	Size:	106304
Methods	No.Formulae	No.Leafs	
SKR	902	15944	
SPP	84	1587	

- The $WFQ[task_ , hgt_]$ function calculates accordingly to the task required the elastic strain energy W , the yield function \mathcal{F} and the system of local equations Q or as follows:

```

WFQ[task_, hgt_] :=
(
  ep = 

|          |          |          |
|----------|----------|----------|
| hgt[[1]] | hgt[[4]] | hgt[[5]] |
| hgt[[4]] | hgt[[2]] | hgt[[6]] |
| hgt[[5]] | hgt[[6]] | hgt[[3]] |

;

  alpha = 

|           |           |           |
|-----------|-----------|-----------|
| hgt[[7]]  | hgt[[10]] | hgt[[11]] |
| hgt[[10]] | hgt[[8]]  | hgt[[12]] |
| hgt[[11]] | hgt[[12]] | hgt[[9]]  |

;

  lambda = hgt[[13]];
  prova = lambda;

  epn = 

|          |          |          |
|----------|----------|----------|
| hgn[[1]] | hgn[[4]] | hgn[[5]] |
| hgn[[4]] | hgn[[2]] | hgn[[6]] |
| hgn[[5]] | hgn[[6]] | hgn[[3]] |

;

  alpha_n = 

|           |           |           |
|-----------|-----------|-----------|
| hgn[[7]]  | hgn[[10]] | hgn[[11]] |
| hgn[[10]] | hgn[[8]]  | hgn[[12]] |
| hgn[[11]] | hgn[[12]] | hgn[[9]]  |

;

  lambda_n = hgn[[13]];
  SMSFreeze[ee, e - ep, "Symmetric" -> True];
  W = Simplify[lambda / 2 Tr[ee]^2 + mu Tr[ee.ee]];
  If[task == "W", Return[]];
  SMSFreeze[sigma, SMSD[W, ee, "Symmetric" -> True], "Symmetric" -> True];

  aux2 = Hkin  $\frac{\lambda_m}{\lambda_{lim} - \lambda_m}$  (ep - epn);
  s = sigma - 1 / 3 IdentityMatrix[3] Tr[sigma];
  aux = s - aux2;
  epDOT = ep - epn;
  Normep = SMSsqrt[Tr[ep.ep]]; (* Norma == ||ep|| *)
  oy = oy0 + Hiso * Normep;
  F = SMSsqrt[Total[aux aux, 2]] - (oy0);

  SMSSetBreak["k"];

  If[task == "F", Return[]];
  A = Simplify[SMSD[F, sigma, "Symmetric" -> True]];

  Z = ep - epn - (lambda - lambda_n) A;
  H = Hkin  $\frac{\lambda_m}{\lambda_{lim} - \lambda_m}$  (epDOT) - (alpha - alpha_n);
  Qg = {Z[[1, 1]], Z[[2, 2]], Z[[3, 3]], Z[[1, 2]], Z[[1, 3]], Z[[2, 3]],
  H[[1, 1]], H[[2, 2]], H[[3, 3]], H[[1, 2]], H[[1, 3]], H[[2, 3]],
  F};

  If[task == "FQ", Return[]];
)

WFQ["F", hgn];
iNR = SMSInteger[idata$$["Iteration"]];
SMSIf[(iNR == 1 && state == 0) || (iNR > 1 && F <  $\frac{1}{10^8}$ )];

```

- This is elastic part of the elasto-plastic formulation. The state variables are set to be the same as at the end of the previous time step.

```

hg = hgn;
SMSExport[Join[hgn, {0}], Table[ed$$["ht", Igh + i], {i, Lgh}]];

```

Appendix B

AceFEM Session Step by Step

- This loads the *AceFEM* package, and prepares input data structures and starts input data section

```
<< "AceFem`";
(* GEOMETRY DIMENSION *)
h = 32.2;          (* mm *)
Lw = 60;          (* mm *)
LL = 60;          (* mm *)
ΔL = .8 * h;
nx = 4; ny = 4; nz = 4;    (* MESH Dimension *)

SMTInputData[];
```

SMTAddDomain[*did*, *element_code*, *data₁* → *d₁*, *data₂* → *d₂*, ...] add new user element

```
SMTAddDomain["Honeycomb", "Element_J2_My_hard", {
  "E -elastic modulus" → 90, "ν -poisson ration" → 0.3,
  "σy0 -yield stress" → 1., "Hkin - Kinematic hardening coefficient" → 10,
  "Hiso - isotropic hardening coefficient" → 0, "ρ0 -density" → 1,
  "bX -force per unit mass X" → 0, "bY -force per unit mass Y" → 0,
  "bZ -force per unit mass Z" → 0, "λlim" → 1
}];

SMTMesh["Honeycomb", "H1", {nx, ny, nz},
  {{{{0, 0, 0}, {Lw, 0, 0}}, {{0, LL, 0}, {Lw, LL, 0}}},
  {{{0, 0, h}, {Lw, 0, h}}, {{0, LL, h}, {Lw, LL, h}}}}];

SMTAddEssentialBoundary[{"Z" == 0 &, 3 → 0}]; (* support*)
SMTAddEssentialBoundary[{"Z" == 0 && "X" == 0 && "Y" == 0 &, 3 → 0, 1 → 0, 2 → 0}];
SMTAddEssentialBoundary[{"Z" == 0 && "X" == Lw && "Y" == 0 &, 2 → 0}];

SMTAddEssentialBoundary[ "Z" == h &, 3 → -ΔL] , (*prescribed displacement*)

• This checks the input data, creates data structures and starts the analysis. The SMTAnalysis also compiles the element source files and creates dynamic link library files (dll file) with the user subroutines or in the case of MDriver reads all the element source files into Mathematica.

SMTAnalysis[];
tf = 1;
λ[t_] := If[OddQ[Floor[(t+1)/2]], 1, -1] (2 Floor[(t+1)/2] - t);
Plot[λ[t], {t, 0, tf}]
```

SMTNextStep[Δt, Δλ] increment time and boundary conditions multiplier

SMTNewtonIteration[] perform one Newton iteration

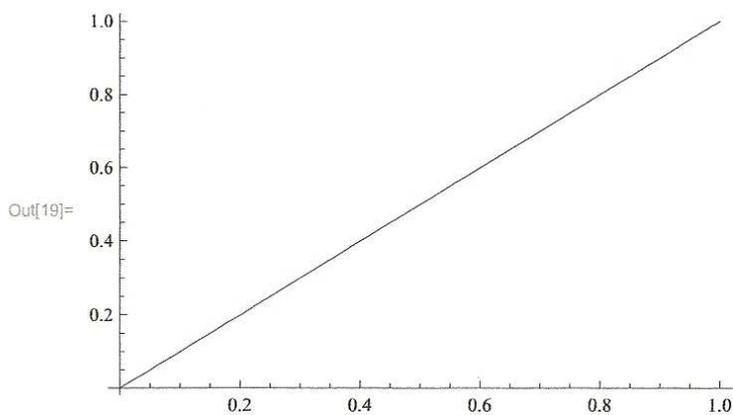
SMTShowMesh[] postprocessing

SMTConvergence[*tolerance*, *maxit*] return True if the *tolerance* is not reached and the maximum number of Newton iteration is not reached

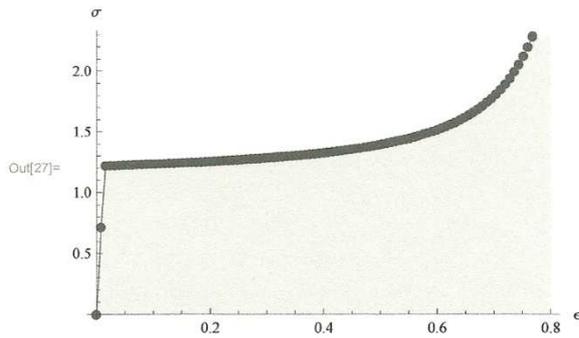
```

plotσe = {{0, 0}};
deltaT = 0.01;
Yield = {};
Do[
  SMTNextStep[1, deltaT];
  While[
    SMTConvergence[10^-3, 10], SMTNewtonIteration[];
  ];
  Print[
    "Step = ", Style[SMTData["Step"], Red], "          ",
    "Yield func. = ",
    Style[SMTPostData["F*"], {Lw/2, LL/2, h/2}], Blue], "          ",
    "σz = ", Style[SMTPostData["Szz"], {Lw/2, LL/2, h/2}], Green], "          ",
    "εz = ", Style[SMTPostData["Ezz"], {Lw/2, LL/2, h/2}], Orange], "          ",
    "εzzp = ", Style[SMTPostData["Ezpz"], {Lw/2, LL/2, h/2}], Black]
  ]
  SMTShowMesh["Field" -> "w", "DeformedMesh" -> True, "Mesh" -> True,
    "Legend" -> "MinMax", "BoundaryConditions" -> True, "Show" -> "Window"];
  SMTStatusReport[];
  σzz = SMTPostData["Szz", {Lw/2, LL/2, h/2}];
  εzz = SMTPostData["Ezz", {Lw/2, LL/2, h/2}];
  aux = {-εzz, -σzz};
  plotσe = Append[plotσe, aux];
  aux2 = SMTPostData["F *", {Lw/2, LL/2, h/2}];
  Yield = Append[Yield, aux2],
  {i, 1, tf*1/deltaT}
];

```



```
In[27]:= ListLinePlot[plotσ, AxesOrigin -> {0, 0},
  Filling -> Axis, PlotMarkers -> Automatic, AxesLabel -> {ε, σ}]
```



```
In[29]:= SMTPostData["Szz", {Lw/2, LL/2, h/2}] / SMTPostData["Ezz", {Lw/2, LL/2, h/2}]
```

Out[29]= 3.4671

```
In[30]:= SMTSimulationReport[];
```

No. of nodes	125
No. of elements	64
No. of equations	322
Data memory (KBytes)	500
Number of threads used/max	2/2
Solver memory (KBytes)	1817
No. of steps	100
No. of steps back	0
Step efficiency (%)	100.
Total no. of iterations	301
Average iterations/step	3.01
Total time (s)	79.359
Total linear solver time (s)	0.497999
Total linear solver time (%)	0.627526
Total assembly time (s)	32.998
Total assembly time (%)	41.5807
Average time/iteration (s)	0.263651
Average linear solver time (s)	0.00165448
Average K and R time (s)	0.00171294
Total Mathematica time (s)	16.953
Total Mathematica time (%)	21.3624
USER-IData	
USER-RData	

- The sparsity structure of the resulting tangent matrix can be graphically displayed by the MatrixPlot function.
- SMTData["TangentMatrix"] returns current global tangent matrix.

```
In[31]:= MatrixPlot[SMTData["TangentMatrix"]]
```

