EXPLORATIVE STUDY FOR STOCHASTIC FAILURE ANALYSIS OF A ROUGHENED BI-MATERIAL INTERFACE: IMPLEMENTATION OF THE SIZE SENSITIVITY BASED PERTURBATION METHOD

A Thesis
Presented to
The Academic Faculty

by

Kotaro Fukasaku

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Mechanical Engineering in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
August 2011

EXPLORATIVE STUDY FOR STOCHASTIC FAILURE ANALYSIS OF A ROUGHENED BI-MATERIAL INTERFACE: IMPLEMENTATION OF THE SIZE SENSITIVITY BASED PERTURBATION METHOD

Approved by:

Professor Mohammed Cherkaoui, Advisor
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Professor Tudor Balan
M2P Department: Mechanics, Materials, Processes
LPMM Laboratory of Physics and Mechanics of Materials: Computational Mechanics Research team
Arts et Métiers ParisTech, France

Dr. Olaf van der Sluis
Department of Process Technology
Philips Applied Technologies – Philips Research,
Eindhoven University of Technology,
The Netherlands

Dr. Laurent Capolungo
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: May 20, 2011

To my mother who never wavered in her support,

to my father who let me get up to here,

and to the fervors that were born and cheered me throughout this thesis.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my principal advisor, Pr. Mohammed Cherkaoui, for giving me the opportunity to discover research, especially applied research, and to work on the present thesis in an exceptional environment, the High Tech Campus Eindhoven which enjoys high international reputation. I would like to express my deepest gratitude to my principal supervisor, Dr. Olaf van der Sluis, for the many hours he spent in training and mentoring me; also, for his patience, guidance and passion for research. I will not forget my second supervisor, Sander Noijen, for his constant support, his pragmatism, great sense of humor, and his willingness to always make me ask the right questions. I feel I will never be able to repay the positive influence they made on my future.

I am grateful to my second advisor, Pr. Tudor Balan, who was always in a good mood and at disposition to carry me through this thesis. Many thanks to Dr. Laurent Capolungo for his particular attention by being part of the Graduate committee.

Special thanks to Pr. Miguel Gutiérrez, who was willing to share his wisdom and wide knowledge in looking after this thesis. I really appreciated his time and assistance.

Thanks also to Philips Applied Technologies, especially the group leader, Dr. Ger Janssen, for letting me work in one of his teams, and to office colleagues, Roy Engeler, Alexandru Opran, Dr. Peter Timmermans, Dr. Sergei Shulepov, Dr. Ruud Voncker, Dr. Bruno Frackowiak and Geneviève Martin, who helped me solve daily issues, physical and metaphysical questions in good mood.

Deserving thanks also are to Josyane Roschitz and Glenda Johnson for their prompt care regarding all necessary coordination and for letting me have a really nice stay in the three countries to which this thesis took me.

My thanks go out to my fellow graduate students Jean-Baptiste Bouquet and Lucas Lallemant, who walked this road with me, to groups such as Philips interns, French Lunch, and to special people with whom we had really great and unforgettable time.

Last but not least, I wish to thank my parents Dr. Yukiko and Dr. Kiichiro Fukasaku, and my girlfriend Juliette Conrath, who cheered me unfailingly and supported me to perform my best.

Contents

List of Tables

List of Figures

SUMMARY

In our age in which the use of electronic devices is expanding all over the world, their reliability and miniaturization have become very crucial. The thesis is based on the study of one of the most frequent failure mechanisms in semiconductor packages, the delamination of interface or the separation of two bonded materials, in order to improve their adhesion and *a fortiori* the reliability of microelectronic devices. It focuses on the metal (-oxide) / polymer interfaces because they cover 95% of all existing interfaces.

Since several years, research activities at mesoscopic scale (1-10μm) have proved that the more roughened the surface of the interface, i.e., presenting sharp asperities, the better the adhesion between these two materials. Because roughness exhibits extremely complex shapes, it is difficult to find a description that can be used for reliability analysis of interfaces. In order to investigate quantitatively the effect of roughness variation on adhesion properties, studies have been carried out involving analytical fracture mechanics; then numerical studies were conducted with Finite Element Analysis. Both were done in a deterministic way by assuming an ideal profile which is repeated periodically.

With the development of statistical and stochastic roughness representation on the one hand, and with the emergence of probabilistic fracture mechanics on the other, the present work adds a stochastic framework to the previous studies. In fact, one of the Stochastic Finite Element Methods, the Perturbation method is chosen for implementation, because it can investigate the effect of the geometric variations on the mechanical response such as displacement field. In addition, it can carry out at once what traditional Finite Element Analysis does with numerous simulations which require changing geometric parameters each time.

This method is developed analytically, then numerically by implementing a module in a Finite Element package MSc. Marc/Mentat. In order to get acquainted and to validate the implementation, the Perturbation method is applied analytically and numerically to the 3

point bending test on a beam problem, because the input of the Perturbation method in terms of roughness parameters is still being studied. The capabilities and limitations of the implementation are outlined.

Finally, recommendations for using the implementation and for furture work on roughness representation are discussed.

Chapter I

INTRODUCTION

The present thesis is a part of the NanoInterface Project (see Appendix A) and deals with one of the most frequent failure mechanisms in semiconductor packages, the delamination of interface or the separation of two bonded materials (see Figure 1). Studying the delamination of interface of the two materials under consideration must give solutions for improving their adhesion and *a fortiori* the reliability of the semiconductor packaging. Moreover, the work focuses especially on the metal (-oxide) / polymer interfaces, because they cover 95% of interfaces in microelectronic devices.

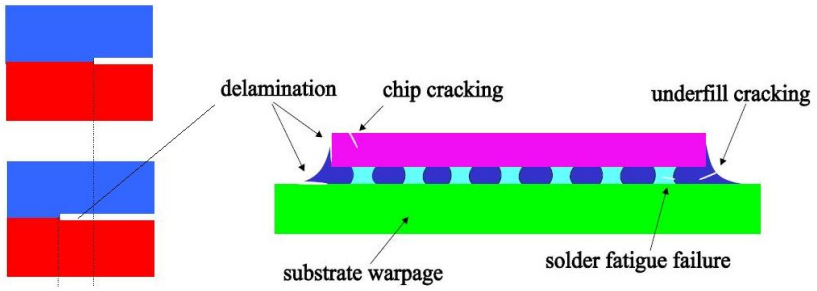


Figure 1: Scheme showing types of failures which occur in semiconductor packaging and a zoom on the delamination [Qu, 2004].

The following interactions play a role in adhesion, or cohesion of materials and interfaces:

- chemical interactions which consist of the primary atomic bonds, i.e., covalent, ionic or metallic bonds. These kinds of interactions result in interface toughness of $1-4\ J/m^2$,
- physical interactions which encompass the secondary atomic bonds such as Coulomb forces or Lipschitz van der Waals forces. These interactions result in interface toughness of $0.1-0.3\ J/m^2$,
- mechanical interlocking which takes place between material surfaces and even inside the materials (see below for explanation) due to geometric effects at mesoscopic scale

$(1 - 10\mu m)$ (see Figure 2). This type of interaction results in interface toughness of $40 - 50\ J/m^2$ or more, which underlines its dominant role in interface bonding [Noijen et al., 2009].

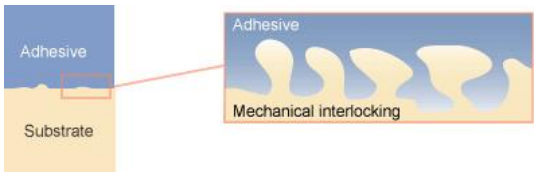


Figure 2: Mechanical interlocking [Specialchem, 2011].

The geometry of the material surfaces is known as roughness with a typical dimension of $1 - 10\mu m$. As the project deals with metal(-oxide)/polymer interface, it can be pointed out that the interface roughness refers to that of the metal surface. The polymer is liquid at the beginning of the processing and solidifies during cure, taking form complementary to the metal roughness at the interface.

At mesoscale, debonding interfaces involves two types of failure modes (or paths): the adhesive (or interfacial) and the cohesive (or bulk) cracks. This can be understood by recalling common daily experience when one removes adhesive tape from an object. Residuals and glue are often left on the object especially when the adhesion strength is high. Transposing this image at mesoscale, and based on the fact that since the cohesive strength and toughness of metal(-oxide)/polymer interface are smaller than that of the polymers which are much smaller than the strength and toughness of metals, adhesion is higher when failures occur in the polymer and not at the interface. The purpose is to increase the proportion of cohesive cracks in the polymer compared to the interfacial adhesive cracks. This can be achieved when the surface of interface is rough, i.e., when sharp asperities are presented. In fact, when the surface is smooth, most failures are interfacial, whereas harshly roughened surface yields cohesive failures and better adhesion. Intense research activities since several years have proved this phenomenon [Kim et al., 2010, Lee and Qu, 2003, 2004, Noijen et al., 2009, Yao and Qu, 2002]. Briefly, the more roughened the interfacial surface, the better the adhesion between these two materials.

One issue rises at this point. Because roughness shows extremely complex shapes, it is difficult to find a description that can be used for reliability analysis of interfaces. One can think of meshing entirely the roughness through Finite Element Analysis (FEA), but, it is obvious that this will be neither effective nor efficient.

Studies based on analytical and numerical fracture mechanics have been carried out to quantify the effect of roughness on adhesion by assuming an ideal profile which is repeated periodically [Noijen et al., 2009, Yao and Qu, 2002] (see Figure 3).

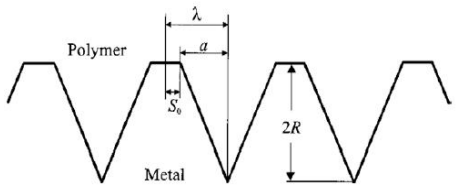


Figure 3: Ideal Profile from [Yao and Qu, 2002].

With the development of statistical and stochastic roughness representation, and with the emergence of probabilistic fracture mechanics, the present thesis adds a stochastic framework to previous studies. Stochastic Finite Element Methods (SFEM) which are seen as an extension of the classical deterministic Finite Element Analysis to stochastic dimension must be a good way to cope with our theme, since it offers tools which can combine stochastic modeling of surface roughness and the Finite Element Analysis. The first method, Monte Carlo simulations can assess the variability of mechanical response using statistics by performing Finite Element Analysis. However, it requires numerous simulations and parameters have to be changed each time. Second, the Perturbation method can evaluate the variability of the response, in particular the two first moments of the response, from those of the input through only one simulation. Spectral Stochastic Finite Element Method is specialized in studying the influence of material properties on mechnical response. Finally, the Reliability method can estimate the probability of rare and undesirable event such as failure (see Chapter 2, Appendix B and [Gutiérrez and Krenk, 2004, Stefanou, 2009, Sudret and Der Kiureghian, 2000] for details).

Because the research is mainly dedicated to a better understanding of the influence of interface roughness, the Perturbation method has been chosen. As the motivation is to

investigate quantitatively the effect of roughness variation on adhesion properties by predicting the chance (probability) of the occurrence of cohesive and adhesive failures, the aim of this thesis is to implement the Perturbation method in order to investigate the effect of geometric variations on the displacement field. Once the variability of the displacement field is assessed, the variability of the adhesion properties can be evaluated by failure mechanics. Then, the probability of the occurence of failure modes can be estimate by Reliability method. In addition, as any Stochastic Finite Element Method requires Variable sensitivity computation. The variable can be a material or geometric parameter. As the geometric variations are our main interest, the size is chosen as variable. That is why the Size sensitivity based Perturbation method is implemented as an explorative study for stochastic failure analysis of a roughened bi-material interface. This mesoscale analysis has to be translated to macroscale adhesion properties, as underlined above. This work, called homogenization, has been carried out independently within the project [Lallemant, 2011].

In Chapter 2, the Perturbation method is derived analytically and applied to the so-called 3 point bending test on a beam problem in order to get acquainted with the method and because the input of the Perturbation method in terms of roughness parameters is still being studied. Chapter 3 shows the implementation of the Size sensitivity based Perturbation method in a commercial Finite Element package, MSc. Marc/Mentat which is the software used in the project and is specialized in performing non-linear analysis (see [MSC.Software, 2011] for more details). The implementation is verified by performing the 3 point bending test on a beam problem. Chapter 4 is devoted to drawing up recommendations on how the roughness could be modeled by showing a preliminary analysis of the roughness of the foil surface of the sample studied. From the roughness measurement data, the work uses several softwares such as Matlab, and a well known package for industries, MountainsMaps [DigitalSurf, 2011], to see what it currently offers in terms of analysis. Last but not least, it deals with the capabilities and limitations of the implementation to roughness study. The conclusion introduces another SFEM, the Reliability method, which can be useful for this research.

Chapter II

ANALYTIC STOCHASTIC ANALYSIS

The present Chapter aims to show the usefulness of the stochastic framework for mechanical problems and to introduce the Stochastic Finite Elements Methods (SFEM). It especially focuses on one of the methods, the Perturbation method, and explains why it is chosen for this thesis in section 2.1. In section 2.2, this method is developed analytically. A typical mechanical problem, the 3 point bending test on a beam is chosen as the application of this method, which is shown in section 2.3.

2.1 Introduction to stochastic analysis

The influence of inherent uncertainties on system behavior has led the scientific community to recognize the importance of the stochastic approach to mechanical problems [Stefanou, 2009]. Uncertainties are involved in the evaluation of any types of loading, material or geometric properties, and are characterized by means of statistics and probability theory. One possible way of solving such problems is to formulate a stochastic differential equation to describe the problem. However, finding out the solution requires tremendous work without having the guarantee of even getting a response [Gutiérrez and Krenk, 2004].

An interesting alternative is to take advantage of the rapid development of computing technology in the mechanical field for use in this stochastic framework. Combining them has created the field of computational stochastic mechanics. It has been recognized that the computational methods permit the analysis and design of complex and large-scale engineering systems and are widely used by the engineering and scientific communities. The archetypal example is the Finite Element Method (FEM). The Stochastic Finite Element Methods (SFEM) consist of an extension of the classical deterministic FE approach to the stochastic framework. In fact, the analysis is performed by deterministic algorithm where inputs and finite elements are characterized by statistical properties, and the response is a stochastic field. The goal is to determine the variation in the displacement field (response),

which depends on a stochastically distributed parameter, the roughness height (input). This can be done by solving the well known global system of the equilibrium equation used in FEM with stochastic properties which is:

$$\tilde{\boldsymbol{K}}(\tilde{\boldsymbol{s}})\tilde{\boldsymbol{u}}^s(\tilde{\boldsymbol{s}}) = \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{s}}), \tag{1}$$

where $\tilde{\boldsymbol{u}}^s$ is the vector containing the values of displacements at the nodes – the superscript ^{s} denotes that the term is scaled, $\tilde{\boldsymbol{K}}$ is the stochastic system stiffness matrix and $\tilde{\boldsymbol{f}}$ is the stochastic loading. $\tilde{\boldsymbol{K}}$ and $\tilde{\boldsymbol{f}}$ depend on stochastic material or geometric properties $\tilde{\boldsymbol{s}}$. Bold capital letters refer to matrices, bold letters refer to vectors and tildes denote stochastic variables.

SFEM offers two general types of analysis:

- Uncertainty analysis which consists in assessing how selected characteristics of the input field such as the mean value and standard deviation, influence those of a response field like the displacement field. This type of analysis can be performed by the following three methods:

  - Monte Carlo simulation (MCS): this method consists in performing a set of simulations where the input parameters are changed in each of them in order to get a set of responses. The response variability of the system is calculated using simple statistical relationships. For instance, the first two moments which are the mean value and the standard deviation of the displacement field can be assessed. The accuracy of the estimation depends on the number of simulations i.e., the number of responses. A small number of samples, like 50, permits only a rough approximation of the first two moments of the response, while a larger sample size, like 500, enables estimating the cumulative distribution function (CDF) of the response. Besides this 'direct' MCS, several variants such as the fast MCS, importance sampling, subset simulation and line sampling exist. Design of Experiments (DoE) and Response Surface Analysis (RSA) are based on

MCS. To sum up, MCS is obviously the simplest method for calculating the response variability in the framework of SFEM and can afford numerous variants and analysis, but also the method that is the most computationally inefficient and time-consuming.

  - Perturbation method: this method calculates the two first moments (mean and standard deviation) of the response from those of the input variables based on the Taylor series expansion. This method is more thoroughly explained in section 2.2.

  - Spectral formulation known as Spectral Stochastic Finite Element Method (SSFEM): this method is an extension of the deterministic finite element method for the solution of boundary value problems with stochastic material properties. It expresses the response as a serie developed in orthogonal polynomials which is known as the Hermite polynomial or polynomial chaos expansion.

- Reliability analysis which provides an accurate approximation of the probability distribution of the response field from that of the input field.

  - Monte Carlo simulation (MCS): (see above).

  - Reliability method: this method is typically used in estimating the probability of a rare and undesirable event such as failure. It consists in converting the original stochastic input variables of the problem into independent normal variables, then using geometric arguments to identify the most likely modes of 'failure' and the associated probabilities. Then, the probability density is a function of the distance from the 'center', representing the expected value.

  For more details, see Appendix B and [Gutiérrez and Krenk, 2004, Stefanou, 2009, Sudret and Der Kiureghian, 2000].

All these methods require the evaluation of the influence of the stochastic input variables, representing stochastic fields such as material properties or geometric parameters, on the response. This is done by means of the assessment of the derivatives of the response with

respect to the stochastic input variables which is performed as Variable sensitivity computation.

Because this research is mainly dedicated to a better understanding of the influence of the variation of interface roughness, the implementation of the Perturbation method is chosen. Indeed, this method can investigate the effect of geometric variations on the mechanical response. Plus, it can perform at the same time the simulations of different roughness profiles which the traditional FEA does with numerous simulations which require changing material and geometric parameters each time (i.e., MCS). In addition, as the geometric variations are our main interest, the Size variable sensitivity computation will be implemented in particular. Chapter 3 shows the implementation of the Size sensitivity computation based Perturbation method.

The Perturbation method will be performed on a simple mechanical problem to validate the implementation. The 3 point bending test on a beam problem has been chosen for that purpose. This well known mechanical problem will be solved in an analytical and numerical way to get acquainted with the Perturbation method.

2.2 Analytic Perturbation method

The present section shows the analytic method to evaluate the two first moments of the response from those of the input variable in a general case.

In theory, several ways exist to determine the first two moments i.e., mean value and standard deviation of $\tilde{y}$, denoted μ_y and σ_y respectively, where $\tilde{y}$ takes the form $\tilde{y} = f(\tilde{x})$. The first two moments of $\tilde{y}$ are expressed as those of $\tilde{x}$ denoted μ_x and σ_x respectively.

Approximations of the mean value and the standard deviation of $\tilde{y}$ can be obtained by utilizing Taylor series expansion about the mean of $\tilde{x}$ up to the second order (the higher terms are negligible compared to the first two orders):

$$\tilde{y} = f(\mu_x) + (\tilde{x} - \mu_x)f'(\mu_x) + \frac{1}{2}(\tilde{x} - \mu_x)^2 f''(\mu_x) + \cdots . \tag{2}$$

where f' and f'' are the first and the second derivatives of f with respect to $\tilde{x}$.

Some statistical definitions and notations are introduced as follows:

- the expectation of $\tilde{y}$, the mean value, is denoted $E(\tilde{y}) = \mu_y$,

- the variance of $\tilde{y}$ is defined as the square of the standard deviation, i.e., $V(\tilde{y}) = \sigma_y^2 = V[(\tilde{x} - \mu_x)^2]$.

From that, the expectation of the response is expressed as:

$$E(\tilde{y}) = E[f(\mu_x)] + E[(\tilde{x} - \mu_x)f'(\mu_x)] + \frac{1}{2}E[(\tilde{x} - \mu_x)^2)f''(\mu_x)] + \cdots . \tag{3}$$

By the definitions and properties of the expectation and the variance, the expectation of $\tilde{y}$ reads:

$$E(\tilde{y}) \backsimeq f[E(\tilde{x})] + \frac{1}{2}f''[E(\tilde{x})]V(\tilde{x}), \quad \text{i.e., } \mu_y \backsimeq f[\mu_x] + \frac{1}{2}f''(\mu_x)\sigma_x^2. \tag{4}$$

Note that the first order term vanishes because of the linearity of the expectation.
In the same way, variance is assessed from Taylor series expansion up to the first order (the second order term is negligible compared to the first order term):

$$V(\tilde{y}) = V[f(\mu_x)] + V[(\tilde{x} - \mu_x)f'(\mu_x)] + \cdots . \tag{5}$$

By the properties of variance, (5) becomes:

$$V(\tilde{y}) \backsimeq f'[(E(\tilde{x})]^2 V(\tilde{x}), \quad \text{i.e., } \sigma_y \backsimeq f'(\mu_x)\sigma_x. \tag{6}$$

See [Hines et al., 2009] for more details.

Then, the analytic Perturbation method is put forward: it gives the mean value and the standard deviation of the response from those of the input even if their relationship is complicated.

2.3 3 point bending test on a beam

2.3.1 Derivation of the deflection

The typical 3 point bending test on a beam problem has been chosen to get acquainted with the Perturbation method analytically and numerically. A horizontal beam with a rectangular section A, clamped in x and y directions in the left extremity, in y direction in the right extremity, is subject to the load P at middle (see Figure 4).

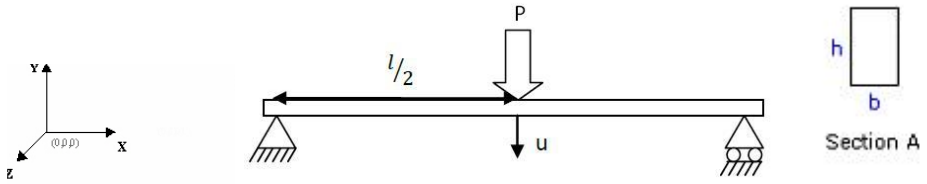


Figure 4: 3 point bending test on a beam problem.

In this Figure, l the beam length, h the height, b the thickness, P the load, E the Young modulus, μ the Poisson ratio, and ρ the density. The study focuses on the influence of the beam length l on the displacement field in y direction denoted as u_y, because their relationship is not linear (see below). First, the deflection of the beam is briefly presented; then, the analytic derivation and numerical application of the Perturbation method are performed in sections 2.3.2 and 2.3.3.

Based on the Euler-Bernoulli beam theory, the expression and the distribution of the reaction force, the moment through z axis, strain, stress and the displacement fields are given as function of the loading and the beam parameters. The distribution of the reaction force T, the moment Mf through z axis and the displacement through y direction u_y are given and drawn in Figure 43.

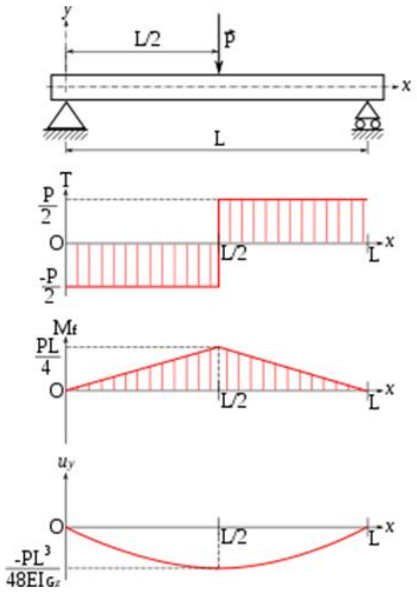


Figure 5: Distributions of the reaction force T, moment Mf and the displacement through y, u_y, of the 3 point bending test on a beam problem [Dau, 2005].

From that, the deflection, δ, can be derived, which is the displacement u_y, at the middle of the beam, $l/2$, as follows:

$$\delta = u_y(x = \frac{l}{2}) = \frac{Pl^3}{48EI_{Gz}} \qquad with\ I_{Gz} = \frac{bh^3}{12}, \tag{7}$$

where I_{Gz} is the area moment of inertia for the rectangular cross section. See Appendix C for the derivation of the deflection.

As the focus is on the influence of the beam length l on deflection δ, the equation reads:

$$\delta = al^3 \qquad with\ a = \frac{P}{48EI_{Gz}}. \tag{8}$$

The deflection δ is a function of the beam length l to the power of 3. As the relationship is not linear, assessing the first two moments of the deflection δ is not straightforward from those of the beam length l by simple stastistics. That is why using the Perturbation method is interesting at this point. Sections 2.3.2 and 2.3.3 show that.

2.3.2 Analytic application

Based on (8), the beam length l is the input variable and the deflection δ is the response. Therefore, let the beam length l be a stochastic variable with its two first moments known, and let the other parameters remain deterministic. The response becomes a stochastic variable and is expressed as:

$$\tilde{\delta} = a\tilde{l}^3 \qquad with\ a = \frac{P}{48EIz}. \tag{9}$$

The purpose of this section is to determine the mean and the variance of the stochastic deflection $\tilde{\delta}$ from those of the stochastic length $\tilde{l}$ knowing that their relationship is not straightforward.

The analytic Perturbation method developed in section 2.2 is introduced as follows:

$$\tilde{\delta} = a\tilde{l}^3 \quad with\ a = \frac{P}{48EIz} \qquad \text{takes the form } \tilde{y} = a\tilde{x}^3 \quad with\ \tilde{x} = \tilde{l}\ and\ \tilde{y} = \tilde{\delta}, \tag{10}$$

i.e., the relationship between the response and the input is:

$$f(\tilde{x}) = a\tilde{x}^3, \quad f'(\tilde{x}) = 3a\tilde{x}^2,\ and\ f''(\tilde{x}) = 6a\tilde{x}. \tag{11}$$

By applying (4) and (5) of the Perturbation method, the mean and the standard deviation of stochastic deflection $\tilde{\delta}$ are evaluated from those of the stochastic beam length $\tilde{l}$ such that:

$$\mu_\delta \backsimeq a\mu_l(\mu_l^2 + 3\sigma_l^2) \quad and \quad \sigma_\delta \backsimeq 3a\mu_l^2\sigma_l, \tag{12}$$

where μ_l and σ_l are the mean and the standard deviation of stochastic beam length $\tilde{l}$, μ_δ and σ_δ are those of the stochastic deflection $\tilde{\delta}$.

2.3.3 Numerical application

Regarding the numerical application of the analytic Perturbation method, taking the example of an animal hanging on a wooden beam with a rectangle shaped section leads to the following values: $E = 10000MPa$, $\nu = 0.4$, $b = 50mm$, $h = 50mm$, $P = 1500N$. The parameter a gathering all the deterministic parameters can be calculated such that:

$$\tilde{\delta} = a\tilde{l}^3 \qquad with\ a = \frac{P}{48EI_{Gz}} = 6e^{-9}\ mm^{-2}. \tag{13}$$

The numerical application is performed with the beam length $\tilde{l}$ as Gaussian distributed, and then as uniformly distributed. Both are varying from $1m = 1000mm$ to $7m = 7000mm$. This means that when a set of beams is considered, it is characterized by the beam length distribution. For instance, an uniformly distributed set of beams means that the set consists of the same number of beams with a given length, i.e., there are as many $1m$ beams as $2m$ beam or $3m$ beams and so on.

According to (13), the deflection δ for a beam length l of $4000mm$ is $\delta = 384mm$.

Let the length of the beam $\tilde{l}$ be considered as Gaussian distributed such that its mean is $4m = 4000mm$ and its standard deviation is $0.4m = 400mm$, i.e., statistically speaking $\mu_l = 4000mm, \sigma_L = 400mm$.

Note that the standard deviation σ_L is equal to 10% of the value of the expectation for usual experimental simulations.

The application gives:

$$\mu_\delta \backsimeq 396mm,\ and\ \sigma_\delta \backsimeq 115mm. \tag{14}$$

Let the stochastic beam length $\tilde{l}$ be uniformly distributed such that a=1m=1000mm and b = 7m=7000mm; then, $\mu_l = \frac{a+b}{2} = 4000mm$ and $V(\tilde{l}) = \frac{(b-a)^2}{12} = 3e^6mm^2$ i.e., $\sigma_l = 1732mm$.

The application gives:

$$\mu_\delta \backsimeq 600mm, \ and \ \sigma_\delta \backsimeq 499mm. \tag{15}$$

Note that this is also the solution for a Gaussian distribution with $\mu_l = 4000$ and $\sigma_l = 1732mm$.

Compared to the deflection of $384mm$ for a $4000m$ beam, $389mm$ means that it is the mean value of deflections calculated from (13) for each beam of a set characterized by a mean of $4000mm$ and a standard deviation of $400mm$. For any set of beams, the traditional method or MCS would be to calculate the deflection for each beam, then obtaining the expectation and its variance, but this is inefficient. The Perturbation method does that at once. This shows the added value of the Perturbation method and *a fortiori* the Stochastic Finite Element Methods over the traditional Finite Element Method. Chapter 3 deals with the implementation of this Perturbation Method on a commercial Finite Element package, MSc. Marc/Mentat, used in the project.

Chapter III

STOCHASTIC FINITE ELEMENT ANALYSIS

This Chapter shows the numerical implementation of the Perturbation method by means of the Finite Element Methods. As presented in section 2.1, this method requires Size sensitivity computation. The entire implementation is integrated in the commercial Finite Element package used in the project, MSc. Marc/Mentat, through the user-defined element subroutine (see [MSC.Software, 2011] for more details). Section 3.1 explains and shows the equations which govern the implementation of the Size sensitivity based Perturbation method through user subroutine. Section 3.2 presents the implementation itself. Last, the implementation is applied to the one element tensile test and the 3 point bending test problem in order to verify its consistency with the analytic Perturbation method presented in section 3.3.

3.1 Equations of Size sensitivity computation and Perturbation method

The development of the present implementation is based on the Size sensitivity computation for uniform scaling, non-linear and fracture problem shown in [Gutierrez and De Borst, 2003], the Variable sensitivity computation, Perturbation method of Stochastic Finite Element Methods [Gutiérrez and Krenk, 2004], and the theory of the Finite Element Method [Zienkiewicz and Taylor, 2005].

Modeling roughness requires studying the size-influence in three dimensions, i.e., in the three directions (x, y and z). Models are built in 2D plane strain condition at the first stage of the project. So, the present scaling are in two directions, x and y, whether they are independent or depedent on each other. Therefore, the uniform Size sensitivity computation for the non-linear and fracture problem presented in [Gutierrez and De Borst, 2003] has been adapted to the so called non-uniform scaling. Actually, a 'general' case is computed here.

Sensitivity computation examines the influence of the stochastic input variables, representing stochastic fields such as material properties or geometric parameters, on the response. This is done by assessing the derivatives of the response with respect to the stochastic input variables. Three techniques exist for the computation of basic Variable sensitivity, the Direct Differentiation Method (DDM), the Adjoint System Method (ASM) and the Finite Difference Method (FDM). The DDM is chosen, because of its applicability to geometrically and physically nonlinear problems for the evaluation of material parameters and shape sensitivities [Gutiérrez and Krenk, 2004]. Here, sensitivity in terms of size is of interest, because the influence of the interface roughness shape is the focus. The DDM is applied to Size sensitivity computation.

Based on (1), the general equilibrium equation adapted to both size sensitivity computation and Perturbation method, is expressed as follows:

$$\boldsymbol{K}(\hat{\boldsymbol{s}})\boldsymbol{u}^s(\hat{\boldsymbol{s}}) = \boldsymbol{f}(\hat{\boldsymbol{s}}), \tag{16}$$

where $\hat{\boldsymbol{s}}$ represents the scaling of the system as an input, $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{f}$ the internal force vector and $\boldsymbol{u}^s$ is the response which has to be determined – the superscript ^{s} denotes that the term is scaled.

When sensitivity computation is performed, $\hat{\boldsymbol{s}}$ denotes the scaling factors s_x and s_y of the system in x and y directions respectively. While Perturbation method is performed, $\hat{\boldsymbol{s}}$ supplies the mean value, the standard deviation of the scaling in x and y directions, and the correlation coefficient between them. These are denoted μ_{s_x}, σ_{s_x}, μ_{s_y}, σ_{s_y} and $\rho_{s_x - s_y}$ respectively. The correlation coefficient informs on the degree of dependence between the scaling in x and in y directions. Indeed, $\hat{\boldsymbol{s}}$ is deterministic for sensitivity computation, and becomes stochastic for Perturbation method. This is the root of the relationship between sensitivity computation and Perturbation method.

In the same way as presented in section 2.2, the response, represented by the displacement field, is expressed by the second order Taylor expansion at the mean of the scaling

$\boldsymbol{\mu}_s = (s_x, s_y)$:

$$\boldsymbol{u}(\boldsymbol{s}) = \boldsymbol{u}(\boldsymbol{\mu}_s) + \frac{\partial \boldsymbol{u}}{\partial s_i}(\boldsymbol{\mu}_s)(s_i - \mu_{s_i}) + \frac{1}{2}\frac{\partial^2 \boldsymbol{u}}{\partial s_i \partial s_j}(\boldsymbol{\mu}_s)(s_i - \mu_{s_i})(s_j - \mu_{s_j}) + \cdots \quad \text{for } i \in \{x, y\} \tag{17}$$

The mean and the standard deviation of the displacement are expressed as follows:

$$\boldsymbol{\mu}_{\boldsymbol{u}} \approx \boldsymbol{u}(\mu_{s_x}, \mu_{s_y}) + \frac{1}{2}\Sigma_{s_x,s_x}\frac{\partial^2 \boldsymbol{u}}{\partial s_x^2}(\mu_{s_x}, \mu_{s_y}) + \frac{1}{2}\Sigma_{s_x,s_y}\frac{\partial^2 \boldsymbol{u}}{\partial s_x \partial s_y}(\mu_{s_x}, \mu_{s_y}) + \frac{1}{2}\Sigma_{s_x,s_y}\frac{\partial^2 \boldsymbol{u}}{\partial s_y \partial s_x}(\mu_{s_x}, \mu_{s_y}) + \frac{1}{2}\Sigma_{s_y,s_y}\frac{\partial^2 \boldsymbol{u}}{\partial s_y^2}(\mu_{s_x}, \mu_{s_y}), \tag{18}$$

$$\boldsymbol{\sigma}_{\boldsymbol{u}}^2 \approx \frac{\partial \boldsymbol{u}}{\partial s_x}(\mu_{s_x}, \mu_{s_y})\boldsymbol{\Sigma}_{s_x,s_x}\frac{\partial \boldsymbol{u}}{\partial s_x}(\mu_{s_x}, \mu_{s_y})^T + \frac{\partial \boldsymbol{u}}{\partial s_x}(\mu_{s_x}, \mu_{s_y})\boldsymbol{\Sigma}_{s_x,s_y}\frac{\partial \boldsymbol{u}}{\partial s_y}(\mu_{s_x}, \mu_{s_y})^T + \frac{\partial \boldsymbol{u}}{\partial s_y}(\mu_{s_x}, \mu_{s_y})\boldsymbol{\Sigma}_{s_x,s_y}\frac{\partial \boldsymbol{u}}{\partial s_x}(\mu_{s_x}, \mu_{s_y})^T + \frac{\partial \boldsymbol{u}}{\partial s_y}(\mu_{s_x}, \mu_{s_y})\boldsymbol{\Sigma}_{s_x,s_y}\frac{\partial \boldsymbol{u}}{\partial s_y}(\mu_{s_x}, \mu_{s_y})^T, \tag{19}$$

where Σ_{s_i,s_j} for $(i,j) \in \{(x,y),(y,x)\}$ are components of the covariance matrix $\boldsymbol{\Sigma}_{s_x,s_y}$ and ^{T} the transpose operator.

The following underlined derivatives:

- $\underline{\frac{\partial u_x}{\partial s_x}}, \underline{\frac{\partial u_x}{\partial s_y}}, \underline{\frac{\partial u_y}{\partial s_x}}, \underline{\frac{\partial u_y}{\partial s_y}}$,
- $\underline{\frac{\partial^2 u_x}{\partial s_x^2}}, \underline{\frac{\partial^2 u_x}{\partial s_y^2}}, \underline{\frac{\partial^2 u_x}{\partial s_x \partial s_y}}, \underline{\frac{\partial^2 u_y}{\partial s_y \partial s_x}}, \underline{\frac{\partial^2 u_y}{\partial s_x^2}}, \underline{\frac{\partial^2 u_y}{\partial s_y^2}}, \underline{\frac{\partial^2 u_y}{\partial s_x \partial s_y}}$ and $\underline{\frac{\partial^2 u_y}{\partial s_y \partial s_x}}$,

(20)

are equivalent to the derivatives of the function f of section 2.2, f' and f''. It is the aim of the Size sensitivity computation to obtain them.

The method for computing the Size sensitivity consists first of incorporating the scale factor into the discretized equilibrium equations, then formulating the derivatives of the displacement with respect to the scale factor. This is the aim of the sections 3.1.1, 3.1.2 and 3.1.3.

3.1.1 Boundary value problem

Based on the reference solid Ω^r under plane strain assumption, a scaled solid Ω^s is introduced by means of the so-called scale factor matrix denoted by $\boldsymbol{s}$:

$$\Omega^s = \left\{ \boldsymbol{Y}^s \in \mathbb{R}^2 \mid \boldsymbol{Y}^s = \boldsymbol{s}\boldsymbol{X}^r \right\} \tag{21}$$

with $\boldsymbol{Y}^s = \begin{pmatrix} x^s \\ y^s \end{pmatrix}$, $\boldsymbol{s} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \in \mathbb{R}^2 \times \mathbb{R}^2$ and $\boldsymbol{X}^r = \begin{pmatrix} x^r \\ y^r \end{pmatrix} \in \Omega^r$ which is equivalent to

$$\begin{pmatrix} x^s \\ y^s \end{pmatrix} = \begin{pmatrix} s_x x^r \\ s_y y^r \end{pmatrix}.$$

Note that the scaling factors s_x and s_y are not additional degrees of freedom but parameters. In the absence of body forces, the equations of the boundary value problem are as follows:

$$\nabla \cdot \boldsymbol{\sigma}^s = \mathbf{0} \qquad \text{in } \Omega^{(s)}, \tag{22a}$$

$$\boldsymbol{u}^s = \boldsymbol{s}^{\tau} \overline{\boldsymbol{u}}^r \qquad \text{on } \partial\Omega_1^{(s)}, \tag{22b}$$

$$\boldsymbol{\sigma}^s \cdot \boldsymbol{n} = {}^{\tau}\overline{\boldsymbol{\sigma}} \qquad \text{on } \partial\Omega_2^{(s)}, \tag{22c}$$

where $\partial\Omega_1^{(s)} \cup \partial\Omega_2^{(s)} = \partial\Omega^{(s)}$, $\boldsymbol{\sigma}^s$ is the stress tensor, $\boldsymbol{u}^s$ is the displacement field, ${}^{\tau}\overline{\boldsymbol{u}}^r$ are the prescribed boundary displacements in the reference solid, ($\boldsymbol{u}^r$ is the displacement field in the reference solid Ω^r), $\boldsymbol{n}$ is the outward normal vector to $\partial\Omega^{(s)}$, and ${}^{\tau}\overline{\boldsymbol{\sigma}}$ is the prescribed boundary loading.

Since prescribed boundary conditions in displacements and in loading are scaled, the stress field does not depend on scaling:

$$\frac{\partial \boldsymbol{\sigma}^s}{\partial s_x} = \frac{\partial \boldsymbol{\sigma}^s}{\partial s_y} = \mathbf{0}, \tag{23}$$

whereas the displacement does, so that:

$$\mathbf{u}^s = \boldsymbol{s}\mathbf{u}^r. \tag{24}$$

In addition, the strain field does not depend on the scaling, because:

$$\frac{\partial \boldsymbol{\sigma}^s}{\partial s_i} = \mathbf{D}\frac{\partial \boldsymbol{\epsilon}^s}{\partial s_i} \Rightarrow \frac{\partial \boldsymbol{\epsilon}^s}{\partial s_i} = \mathbf{0} \quad \text{for } i \in \{x, y\}, \tag{25}$$

where $\boldsymbol{D}$ is the stiffness matrix. fFor more details see 31.

3.1.2 Finite element formulation

The Finite element formulation implies a finite element discretization $\{\Omega_e^s\}$ of the scaled solid Ω^s and the isoparametric domain Ω^0. In addition, the formulation has to be based on

a specific element. The so-called linear, four-node, isoparametric, plane strain, quadrilateral element (see Figure 6) is used on numerical models of the interface delamination studied in the framework of the project.

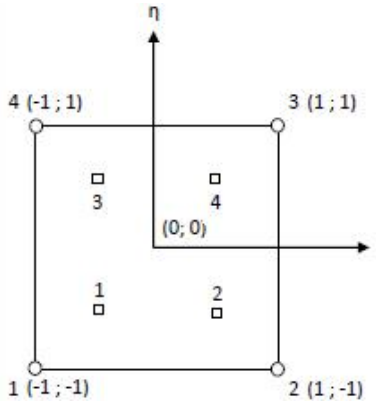


Figure 6: Linear, four-node, quadrilateral element.

Generally, the models of mechanical problems we are involved in, often rely on this element. This element is noted on Marc/Mentat as 11.

The element is a rectangular linear element which belongs to either Lagrangian or Serendipity family. As we have 4 nodes with 2 degrees of freedom per node (horizontal and vertical displacements, denoted as u_x and u_y respectively), the shape function takes the form:

$$u = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy. \tag{26}$$

Hence, the variable transformation can be expressed as:

$$\begin{aligned} \boldsymbol{Y}^s &= \sum_i \boldsymbol{Y}_i^s N_i(\xi,\eta) = \sum_i \boldsymbol{s}\boldsymbol{X}_i^r N_i(\xi,\eta) \\ \Leftrightarrow \begin{pmatrix} x^s \\ y^s \end{pmatrix} &= \sum_i \begin{pmatrix} s_x x_i^r \\ s_y y_i^r \end{pmatrix} N_i(\xi,\eta) \Leftrightarrow \begin{cases} x^s = \sum_i s_x x_i^r N_i(\xi,\eta) \\ y^s = \sum_i s_y y_i^r N_i(\xi,\eta) \end{cases} \end{aligned} \tag{27}$$

where $\boldsymbol{X}_i^r \in \Omega_e$, $\boldsymbol{Y}_i^s \in \Omega_e^s$, both of which are global coordinates, and N_i the shape function, ξ and η isoparametric coordinates. See Figure 7 which depicts variable transformations.

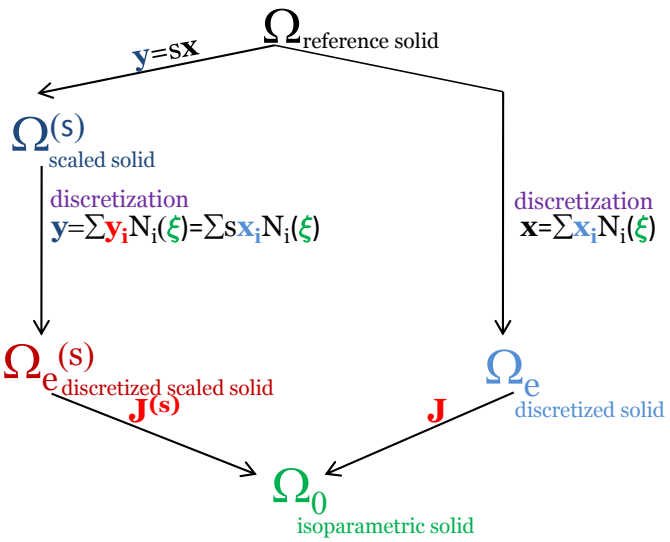


Figure 7: Size sensitivity finite element formulation for non-uniform scaling.

In this Figure, J^r and J^s are Jacobians of the isoparametric transformations.

Based on nodal coordinates, the shape function and its derivatives are expressed in the isoparametric domain such that:

- $$N_i(\xi,\eta) = \frac{1}{4}(1+\xi\xi_i)(1+\eta\eta_i)$$

- $$\begin{cases} \dfrac{\partial N_i}{\partial \xi}(\xi,\eta) &= \dfrac{1}{4}\xi_i(1+\eta\eta_i) \\ \\ \dfrac{\partial N_i}{\partial \eta}(\xi,\eta) &= \dfrac{1}{4}\eta_i(1+\xi\xi_i). \end{cases} \tag{28}$$

Note that the shape function and its derivatives do not depend on the scaling, because they belong to the isoparametric domain.

Concerning the Jacobian, the Jacobian matrix is expressed as follows:

$$\mathbf{J}^s = \begin{pmatrix} j_{11} & j_{12} \\ j_{21} & j_{22} \end{pmatrix} = \begin{pmatrix} s_x\dfrac{\partial x}{\partial \xi} & s_y\dfrac{\partial y}{\partial \xi} \\ s_x\dfrac{\partial x}{\partial \eta} & s_y\dfrac{\partial y}{\partial \eta} \end{pmatrix} = \begin{pmatrix} \displaystyle\sum_i \frac{\partial N_i}{\partial \xi}s_x x_i & \displaystyle\sum_i \frac{\partial N_i}{\partial \xi}s_y y_i \\ \displaystyle\sum_i \frac{\partial N_i}{\partial \eta}s_x x_i & \displaystyle\sum_i \frac{\partial N_i}{\partial \eta}s_y y_i \end{pmatrix}.$$

The Jacobian is the determinant of the above matrix:

$$J^s = |\boldsymbol{J}^s| = j_{11}j_{22} - j_{12}j_{21} = s_x s_y J^r. \tag{29}$$

Regarding the strain displacement matrix denoted as $\boldsymbol{B}_e^s$, the derivatives of the shape function according to the global coordinate system read:

$$\begin{pmatrix} N_{i,x}^s \\ \\ N_{i,y}^s \end{pmatrix} = \begin{pmatrix} \dfrac{\partial N_i^s}{\partial x} \\ \\ \dfrac{\partial N_i^s}{\partial y} \end{pmatrix} = \frac{1}{J^s} \begin{pmatrix} s_y \dfrac{\partial y}{\partial \eta} & -s_y \dfrac{\partial y}{\partial \xi} \\ \\ -s_x \dfrac{\partial x}{\partial \eta} & s_x \dfrac{\partial x}{\partial \xi} \end{pmatrix} \begin{pmatrix} \dfrac{\partial N_i}{\partial \xi} \\ \\ \dfrac{\partial N_i}{\partial \eta} \end{pmatrix}.$$

Hence,

$$\begin{cases} \dfrac{\partial N_i^s}{\partial x} = \dfrac{1}{s_x} \dfrac{\partial N_i}{\partial x} \\ \\ \dfrac{\partial N_i^s}{\partial y} = \dfrac{1}{s_y} \dfrac{\partial N_i}{\partial y}. \end{cases} \tag{30}$$

As the solid is a homogeneous, isotropic, elastic medium in plane strain condition, Hooke's law can be applied such that:

$$\boldsymbol{\sigma}^s = \boldsymbol{D_e} \boldsymbol{\epsilon}^s, \tag{31}$$

$$\text{where } \boldsymbol{D_e} = \begin{pmatrix} D_{11} & D_{12} & 0 \\ D_{12} & D_{11} & 0 \\ 0 & 0 & D_{33} \end{pmatrix} \text{ which does not depend on the scaling,} \tag{32}$$

$$\text{with } D_{11} = E \frac{(1-\nu)}{(1+\nu)(1-2\nu)}, \tag{33}$$

$$D_{12} = \frac{\nu D_{11}}{(1-\nu)} = \frac{\nu E}{(1+\nu)(1-2\nu)}, \tag{34}$$

$$D_{33} = \frac{D_{11} - D_{12}}{2} = \frac{E}{2(1-\nu)}. \tag{35}$$

The element stiffness matrix $\boldsymbol{K}_e^s$ is expressed by $\boldsymbol{B}_e^s$ and $\boldsymbol{D_e}$ such that:

$$\boldsymbol{K}_e^s = \int_{V^e} \boldsymbol{B}_e^{sT} \boldsymbol{D}_e \boldsymbol{B}_e^s dV = \int_{V_e} \hat{\boldsymbol{K}}_e^s dV \tag{36}$$

with components of $\hat{\boldsymbol{K}}_e^s$:

$$\hat{\boldsymbol{k}}_{e\,ij}^s = \begin{pmatrix} N_{i,x}^s D_{11} N_{j,x}^s + N_{i,y}^s D_{33} N_{j,y}^s & N_{i,x}^s D_{12} N_{j,y}^s + N_{i,y}^s D_{33} N_{j,x}^s \\ N_{i,y}^s D_{12} N_{j,x}^s + N_{i,x}^s D_{33} N_{j,y}^s & N_{i,y}^s D_{11} N_{j,y}^s + N_{i,x}^s D_{33} N_{j,x}^s \end{pmatrix}. \tag{37}$$

The strain, stress, and internal force matrices, $\boldsymbol{\epsilon}^s$, $\boldsymbol{\sigma}^s$ and $\boldsymbol{f}_e^s$ are expressed as follows:

$$\begin{aligned} \boldsymbol{\epsilon}^s &= \boldsymbol{B}_e^s \boldsymbol{u}^s \qquad \text{with } \boldsymbol{u}^s \text{ as the nodal displacement vector,} \\ \boldsymbol{\sigma}^s &= \boldsymbol{D}_e \boldsymbol{\epsilon}^s = \boldsymbol{D}_e \boldsymbol{B}_e^s \boldsymbol{u}^s, \\ \boldsymbol{f_{e\,i}^s} &= -\int_{V_e} \boldsymbol{B}_e^{sT} \boldsymbol{\sigma}^s dV. \end{aligned} \tag{38}$$

The integration of the element stiffness and the internal force matrices is treated by means of Gaussian integration method, which consists in adding weighted integrands at several points, as (39) and Table 1 show:

$$\int_{-1}^{1}\int_{-1}^{1} g(\xi,\eta)d\xi d\eta \approx \sum_{i=1}^{4}\sum_{j=1}^{4} g(\xi_i,\eta_j) w_i w_j. \tag{39}$$

Table 1: Four integration points in the rectangle.

		location of the integration points		
n_i	point	ξ	η	w_i
4	1	-0.57735	-0.57735	1
	2	0.57735	-0.57735	1
	3	-0.57735	0.57735	1
	4	0.57735	0.57735	1

Finally, the following equilibrium equation at element level can be computed:

$$\boldsymbol{K}_e^s(\boldsymbol{s})\boldsymbol{u}_e^s(\boldsymbol{s}) = \boldsymbol{f}_e^s(\boldsymbol{s}) \Leftrightarrow \boldsymbol{K}_e^s(s_x,s_y)\boldsymbol{u}_e^s(s_x,s_y) = \boldsymbol{f}_e^s(s_x,s_y). \tag{40}$$

After matrix assembly and calculation, the displacement $\boldsymbol{u}^s$, strain $\boldsymbol{\epsilon}^s$ and stress $\boldsymbol{\sigma}^s$ fields are obtained.

3.1.3 The scale factor sensitivity

In this section, the sensitivity of system response, the displacement, with respect to scale factors are studied, i.e., the derivatives $\frac{\partial u_x}{\partial s_x}$, $\frac{\partial u_x}{\partial s_y}$, $\frac{\partial u_y}{\partial s_x}$, $\frac{\partial u_y}{\partial s_y}$, $\frac{\partial^2 u_x}{\partial s_x^2}$, $\frac{\partial^2 u_x}{\partial s_y^2}$, $\frac{\partial^2 u_x}{\partial s_x \partial s_y}$, $\frac{\partial^2 u_y}{\partial s_y \partial s_x}$, $\frac{\partial^2 u_y}{\partial s_x^2}$, $\frac{\partial^2 u_y}{\partial s_y^2}$, $\frac{\partial^2 u_y}{\partial s_x \partial s_y}$ and $\frac{\partial^2 u_y}{\partial s_y \partial s_x}$. To get them, the equilibrium equation at the element level 40 is derived with respect to the scale factors s_x and s_y by means of direct differentiation method (DDM) as follows,

$$1^{st} \text{ order } : \quad \frac{\partial \boldsymbol{K}_e^s}{\partial s_i}\boldsymbol{u}_e^s + \boldsymbol{K}_e^s \frac{\partial \boldsymbol{u}_e^s}{\partial s_i} = \frac{\partial \boldsymbol{f}_e^s}{\partial s_i} \quad \text{for } i \in \{x, y\}, \tag{41}$$

$$2^{nd} \text{ order } : \begin{cases} \dfrac{\partial^2 \boldsymbol{K}_e^s}{\partial s_i^2}\boldsymbol{u}_e^s + 2\dfrac{\partial \boldsymbol{K}_e^s}{\partial s_i}\dfrac{\partial \boldsymbol{u}_e^s}{\partial s_i} + \boldsymbol{K}_e^s \dfrac{\partial^2 \boldsymbol{u}_e^s}{\partial s_i^2} = \dfrac{\partial^2 \boldsymbol{f}_e^s}{\partial s_i^2} & \text{for } i \in \{x, y\}, \\ \dfrac{\partial^2 \boldsymbol{K}_e^s}{\partial s_i s_j}\boldsymbol{u}_e^s + \dfrac{\partial \boldsymbol{K}_e^s}{\partial s_i}\dfrac{\partial \boldsymbol{u}_e^s}{\partial s_j} + \dfrac{\partial \boldsymbol{K}_e^s}{\partial s_j}\dfrac{\partial \boldsymbol{u}_e^s}{\partial s_i} + \boldsymbol{K}_e^s \dfrac{\partial^2 \boldsymbol{u}_e^s}{\partial s_i s_j} = \dfrac{\partial^2 \boldsymbol{f}_e^s}{\partial s_i s_j} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } (i, j) \in \{(x, y), (y, x)\} \end{cases} \tag{42}$$

where $\boldsymbol{K}_e^s$ remains the same as the one in 40; $\frac{\partial \boldsymbol{u}_e^s}{\partial s_i}$, $\frac{\partial^2 \boldsymbol{u}_e^s}{\partial s_i^2}$ and $\frac{\partial^2 \boldsymbol{u}_e^s}{\partial s_i s_j}$ are unknown; $\frac{\partial \boldsymbol{K}_e^s}{\partial s_i}$, $\frac{\partial \boldsymbol{f}_e^s}{\partial s_i}$, $\frac{\partial^2 \boldsymbol{K}_e^s}{\partial s_i^2}$, $\frac{\partial^2 \boldsymbol{f}_e^s}{\partial s_i^2}$, $\frac{\partial^2 \boldsymbol{K}_e^s}{\partial s_i s_j}$ and $\frac{\partial^2 \boldsymbol{f}_e^s}{\partial s_i s_j}$ are expressed in Appendix D.1; and $\boldsymbol{u}_e^s$ is the displacement calculated in (40).

These equations can be integrated into the following system and solved at once:

$$\begin{pmatrix} \boldsymbol{K}_e^s & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \frac{\partial \boldsymbol{K}_e^s}{\partial s_x} & \boldsymbol{K}_e^s & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \frac{\partial \boldsymbol{K}_e^s}{\partial s_y} & \boldsymbol{0} & \boldsymbol{K}_e^s & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \frac{\partial^2 \boldsymbol{K}_e^s}{\partial s_x^2} & 2\frac{\partial \boldsymbol{K}_e^s}{\partial s_x} & \boldsymbol{0} & \boldsymbol{K}_e^s & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \frac{\partial^2 \boldsymbol{K}_e^s}{\partial s_y^2} & \boldsymbol{0} & 2\frac{\partial \boldsymbol{K}_e^s}{\partial s_y} & \boldsymbol{0} & \boldsymbol{K}_e^s & \boldsymbol{0} & \boldsymbol{0} \\ \frac{\partial^2 \boldsymbol{K}_e^s}{\partial s_x s_y} & \frac{\partial \boldsymbol{K}_e^s}{\partial s_y} & \frac{\partial \boldsymbol{K}_e^s}{\partial s_x} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{K}_e^s & \boldsymbol{0} \\ \frac{\partial^2 \boldsymbol{K}_e^s}{\partial s_y s_x} & \frac{\partial \boldsymbol{K}_e^s}{\partial s_y} & \frac{\partial \boldsymbol{K}_e^s}{\partial s_x} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{K}_e^s \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_e^s \\ \frac{\partial \boldsymbol{u}_e^s}{\partial s_x} \\ \frac{\partial \boldsymbol{u}_e^s}{\partial s_y} \\ \frac{\partial^2 \boldsymbol{u}_e^s}{\partial s_x^2} \\ \frac{\partial^2 \boldsymbol{u}_e^s}{\partial s_y^2} \\ \frac{\partial^2 \boldsymbol{u}_e^s}{\partial s_x s_y} \\ \frac{\partial^2 \boldsymbol{u}_e^s}{\partial s_y s_x} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_e^s \\ \frac{\partial \boldsymbol{f}_e^s}{\partial s_x} \\ \frac{\partial \boldsymbol{f}_e^s}{\partial s_y} \\ \frac{\partial^2 \boldsymbol{f}_e^s}{\partial s_x^2} \\ \frac{\partial^2 \boldsymbol{f}_e^s}{\partial s_y^2} \\ \frac{\partial^2 \boldsymbol{f}_e^s}{\partial s_x s_y} \\ \frac{\partial^2 \boldsymbol{f}_e^s}{\partial s_y s_x} \end{pmatrix}. \tag{43}$$

Each component of the stiffness system matrix $\frac{\partial \boldsymbol{K}_e^s}{\partial s_i}$, $\frac{\partial^2 \boldsymbol{K}_e^s}{\partial s_i^2}$ and $\frac{\partial^2 \boldsymbol{K}_e^s}{\partial s_i s_j}$ as well as those of the internal force vector $\frac{\partial \boldsymbol{f}_e^s}{\partial s_i}$,$\frac{\partial^2 \boldsymbol{f}_e^s}{\partial s_i^2}$ and $\frac{\partial^2 \boldsymbol{f}_e^s}{\partial s_i s_j}$ for $(i, j) \in \{(x, y), (y, x)\}$ are derived and presented in Appendix D.1. An important of aspect of singularity of the element stiffness matrix K_e^s,

a fortiori its derivatives and the entire above system comming from body motions, which have to be suppressed by prescriptions in displacement in order to make the system regular such that the calculation yields an unique solution, is also discussed in Appendix D.2.

(43) and equations in Appendix D.1 of the Size sensitivity computation, and (18) and (19) of the Perturbation method have been derived for the implementation of the Size sentivity based Perturbation method. Section 3.2 presents the implementation.

3.2 Presentation and utilization of the Size sensitivity based Perturbation method user-defined element 11 subroutine

From the equations in section 3.1, and the pattern of user defined element subroutine available in Marc/Mentat [MSC.Software, 2011], the so-called Size sensitivity based Perturbation method user-defined element 11 is implemented. Figure 8 describes the overall presentation of the implementation. It allows performing:

- simulations with the linear, four-node, isoparametric, plane strain, quadrilateral element itself,
- Size sensitivity computation in which the scaling can be either uniform or non uniform. Input is composed of scale factors in x and y directions, denoted as s_x and s_y respectively. and the output of reaction force, displacement, stress, strain and derivatives of the displacement fields,
- Perturbation method where the Size sensitivity computation is run with the mean, the variance and the coefficient of correlation of the scale factors as input, denoted as $\boldsymbol{\mu_s}$, $\boldsymbol{\sigma_s}$, $\rho_{s_x - s_y}$ respectively. Mean $\boldsymbol{\mu_u}$, standard deviation $\boldsymbol{\sigma_u}$ of the displacement field as output are calculated by post-processing from the output of the Size sensitivity computation.

  See .

The code and the utilization of the implementation is thoroughly shown and explained in Appendix E).

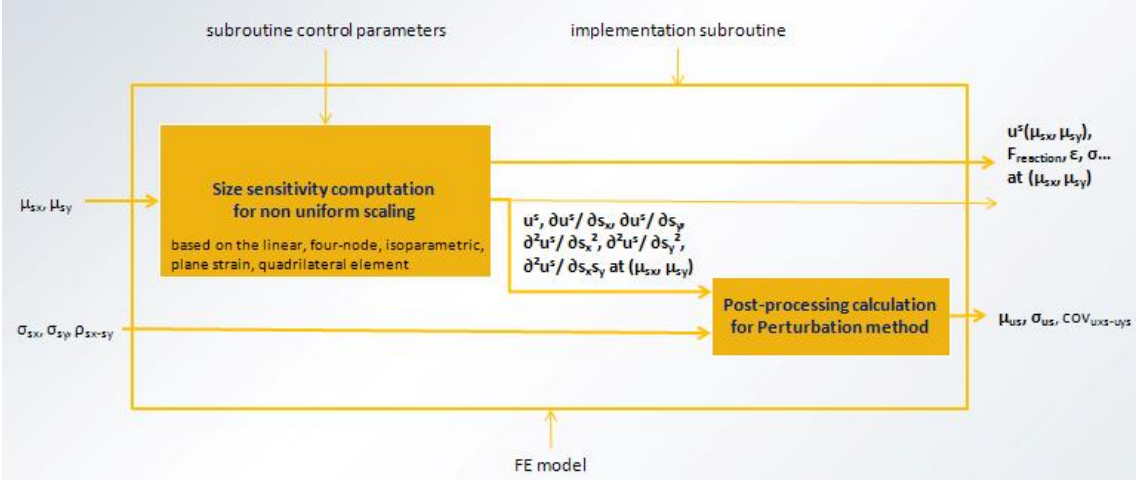


Figure 8: Overview of the implementation.

To sum up, the implementation gives the variability of the displacement field $\boldsymbol{u}$ with its mean $\boldsymbol{\mu_u}$ and standard deviation $\boldsymbol{\sigma_u}$ at each node besides what the generic linear, four-node, isoparametric, plane strain, quadrilateral element can offer such as reaction force, displacement, stress, strain fields. Section 3.3 presents simulations with this implementation, in particular it aims to show the validation process of the implementation through benchmarks, which is an important aspect.

3.3 Benchmarks for validation

Simulations have been performed to check each step of the code and validate the entire implementation. The one element tensile test has been at the base of the verification process. Its analytic derivation and results are presented and compared to the numerical results from simulations. Last, the implementation is applied to the 3 point bending test to verify that the numerical results are the same as the ones derived analytically in Chapter 2.

3.3.1 One element tensile test

From the beginning to the validation of the user element, the one element plane strain tensile test in one direction, y direction, is performed. Figure 9 shows the model with boundary conditions, which can be viewed on the Mentat interface.

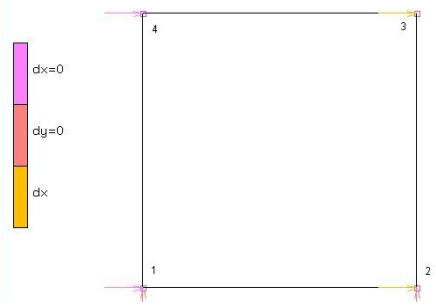


Figure 9: Model of the tensile test on one element.

Node 1 is clamped, node 2 is clamped in y direction and pulled in x direction in the amount of $d_x = \bar{u_x} = 0.01$, node 3 is pulled in x direction in the amount of $d_x = \bar{u_x} = 0.01$, and node 4 is clamped in x direction. The material and geometric properties are the same as the ones for the 3 point bending test on a beam problem, i.e., $E = 10000$, $\nu = 0.4$ and $thickness = 50$.

The aim of the implementation is to get the right displacement, the one analytically calculated which is the same as the one performed with the Marc/Mentat element 11. For that, the stiffness matrix has been checked first, then the internal force vector and the displacement, as well as the strain and stress fields has been verified. Then, the Size sensitivity computation and the Perturbation method are verified by comparing the derivatives and the first two moments of the displacement derived analytically.

3.3.1.1 Analytic derivation

This problem is derived analytically, in order to get the expected results and understand them in detail.

From the boundary conditions and Hooke's law, the strain and stress fields are expressed as follows:

- Regarding the displacement, strain and stress fields,
  - boundary condition: $\bar{u_x} = 0.01$ and $\epsilon_{xx} = \frac{\bar{u_x}}{s_x} = \frac{0.01}{s_x}$,
  - $\epsilon_{yy} = \frac{\bar{u_y}}{s_y} = \frac{-0.00667}{s_x}$,
  - plane strain problem: $\epsilon_{zz} = 0$,

- pure tensile test: $\epsilon_{xy} = 0$,
- Hooke's law: $\sigma_{xx} = \frac{E\epsilon_{xx}}{1-\nu^2} = \frac{E\bar{u_x}}{s_x(1-\nu^2)} = \frac{119.0476}{s_x}$,
- boundary condition: $\sigma_{yy} = 0$,
- Hooke's law: $\sigma_{zz} = \frac{E\nu\epsilon_{xx}}{1-\nu^2} = \frac{E\nu\bar{u_x}}{s_x(1-\nu^2)} = \frac{47.619}{s_x}$,
- pure tensile test: $\sigma_{xy} = 0$.

Note that the strain and the stress fields are homogeneous in the element due to the essence of the problem.

- Concerning the Size sensitivity computation and the Perturbation method:
  - as node 1 is clamped, its nodal displacement, derivatives and the first two moments of the displacement are null,
  - for nodes 2 and 3, the displacement in x direction is equal to the prescription $u_{x,2} = \bar{u_x} = 0.01$, the mean is $\mu_{u_x} = \bar{u_x} = 0.01$, the standard derivation is null because the prescription is a constant independent of the scaling,

    as node 2 is clamped in y direction, its nodal y displacement, derivatives and the first two moments of the displacement are null,
  - for nodes 3 and 4 in y direction, the displacement is $u_y = -0.00667\frac{s_y}{s_x}$, and the derivatives are:
    * $\frac{\partial u_y}{s_x} = -u_y\frac{s_y}{s_x^2} = \frac{-0.00667 s_y}{s_x^2}$,
    * $\frac{\partial u_y}{s_y} = \frac{u_y}{s_x} = \frac{-0.00667}{s_y}$,
    * $\frac{\partial^2 u_y}{s_x^2} = 2u_y\frac{s_y}{s_x^3} = \frac{0.01334 s_y}{s_x^3}$,
    * $\frac{\partial^2 u_y}{s_x^2} = \frac{-u_y}{s_x^2} = \frac{0.00667}{s_x^2}$,

    from the expression of the displacement, the mean is expressed as:

$$\mu_{u_y} = -0.00667\frac{\mu_{s_y}}{\mu_{s_x}}, \tag{44}$$

    and the standard deviation is equal to:

$$\sigma_{u_y} = 0.00667\sqrt{\frac{\mu_{s_y}^2}{\mu_{s_x}^2} + \sigma_{s_x}^2\frac{\mu_{s_y}^2}{\mu_{s_x}^4} - 2\rho\frac{\mu_{s_y}}{\mu_{s_x}^3}\sigma_{s_x}\sigma_{s_y}} \approx 0, \tag{45}$$

– node 4 is clamped in x direction, its nodal x displacement, derivatives and the first two moments of the displacement are null.

3.3.1.2 Numerical simulations

The deformed shape in Figure 10 with the total displacement field is obtained:

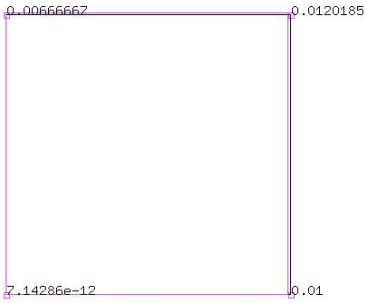


Figure 10: Deformed shape of the tensile test on one element.

The implementation has been verified at each step of the calculations for the 1D tensile test. The displacement, strain and stress fields are the same in analytic derivations or simulation with the generic Marc/Mentat element or with the implementation. In addition, all the derivatives of the displacement with respect to the scaling factors and the mean and standard deviation of the displacement are the same in either analytic derivations or simulation with the implementation.

3.3.2 3 point bending test on a beam

The 3 point bending test on a beam as shown in Figure 4 was performed (see Figure 11).

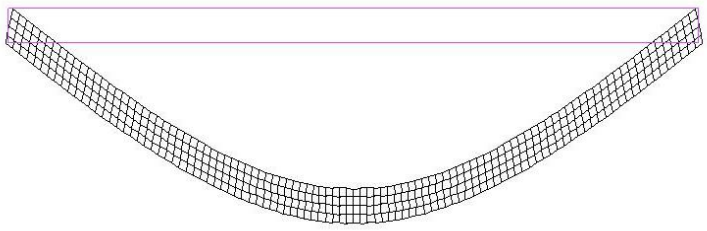


Figure 11: Numerical simulation of the 3 point bending test on a beam problem.

The subroutine and the generic element 11 lead to the same displacement field. The deflection is 384mm.

One point can be raised here. The analytical derivation involves the Euler Bernoulli beam theory whereas the numerical simulation involves plane strain condition. Results can be comparable if the stiffness takes into account the factor $\frac{1}{1-\nu^2}$ because the plane strain condition is in two dimensions so it does not allow the material to move in the third direction, which makes the system stiffer. To sum up, $E_{analytical} = \frac{E_{numerical}}{1-\nu^2}$.

In addition, the result depends on the number of elements regarding meshing. Indeed, the deflection comes up to the same value as the one derived analytically with Euler Bernoulli beam theory (384mm) with 25600 elements. This is called mesh convergence.

In the same way, the assumed strain option which allows enhancing the bending state of simulations with the linear quadrilateral element can be added. This makes plane strain simulations reflect the same bending state of the 3 point bending test on a beam modelized with the Euler Bernoulli beam theory with less elements – 1600 elements.

This is also confirmed by using the eight-node, isoparametric, plane strain, quadrilateral element which is adapted to bending behavior with a much lesser number of elements – 400 elements.

Finally, the implementation of the Size sensitivity based Perturbation method user defined element 11 is verified by means of the previously analytically derived 3 point bending test on beam.

Recall that the analytical Perturbation method yields a mean of deflection of 396 mm and a deviation of 115mm from the mean input of 4000mm and the deviation of 400mm, given that the deflection is 384mm with a 4m beam.

Regarding numerical simulations, as mentionned above, the assumed strain enhancement is not in the implementation, therefore the value of deflection is reached only by increasing the number of elements in the model to 25600 elements. With that, the numerical Perturbation method using the implementation for a model with 1000m beam and the same height, thickness and material properties, and with $\mu_x = 4 \; and \; \sigma_x = 0.4$, leads to the same

value of deflection and its first two moments.

This proves the consistency of the implementation of the Perturbation method.

3.4 Computational cost of the implementation

Last, the computational cost, which an important aspect regarding the implementation, is presented here. Time of computation and storage are compared. As the one element tensile test consists of only one element, it does not lead to significant values for comparison. The 3 point bending test on a beam problem yields to meaningful comparative values. Two types of simulations are chosen:

1. 4m beam which consists of the generic Marc/Mentat elements 11,

2. 1 beam in which the implementation is applied to all elements with the following parameters: $\mu_{s_x} = 4$, $\mu_{s_y} = 1$, $\sigma_{s_x} = 0.4$, $\sigma_{s_y} = 0$, $\rho_{s_x - s_y} = 0$, which leads to the same simulation of the first one and gives in addition the variability of the displacement field with mean $\boldsymbol{\mu_u}$ and standard deviation $\boldsymbol{\sigma_u}$.

Table 2: Computational cost comparative table of the implementation based on the 3 point bending test on a beam problem.

Simulations	Number of elements	Number of increments	Computation time (in s)	Storage (in MB)
4m beam with generic	6400	1	2.15	104
Marc/Mentat element 11	25600	1	7.73	18
1m beam with implementation				
with $\mu_{s_x} = 4$, $\mu_{s_y} = 1$,	6400	1	19.83	717
$\sigma_{s_x} = 0.4$, $\sigma_{s_y} = 0$, $\rho_{s_x - s_y} = 0$	25600	1	148.8	3313

The Table 2 shows that for less than 10000 element models, the computation time and the storage of simulations with the implementation is about 10 times bigger than simulations without it. However, the computation time is no more than several minutes, and it is even counted in seconds for more than 20000 element models, which means that simualtions are still instantaneous.

As a conclusion, the Size sensitivity based Perturbation method has been implemented through user defined element subroutine and recall that it allows performing:

- the linear, four-node, isoparametric, plane strain, quadrilateral element itself,
- the Size sensitivity computation in which scaling can be either uniform or non uniform so that the input is composed of scale factors in x and y directions and the output of reaction force, displacement, stress, strain and derivatives of the displacement fields,
- the Perturbation method where Size sensitivity computation is run with the mean, the variance and the coefficient of correlation of the scale factors as input. Mean and standard deviation of the displacement field as output are calculated by post-processing of the output of the Size sensitivity computation.

Chapter IV

APPLICATION TO ROUGHNESS MODELS

As the size sensitivity based Perturbation method has been implemented, the present Chapter aims to draw up some recommendations for its application to roughness models. To a large extent, in the framework of the project, the development at mesoscopic scale of a stochastic model of surface roughness of the metal(-oxide)/polymer interface which predicts the chance of different failure modes (cohesive or adhesive) during its delamination, needs some input data reflecting roughness parameters at the first stage.

The notion of roughness is discussed first. Engineers are used to the idea that materials have intrinsic bulk properties such as density, elastic modulus, etc. Similarly, surfaces representing material boundaries have intrinsic properties. Surface roughness is one of them. It may be thought that this notion seems more difficult to define, because it depends on the scale of observation. Because roughness shows extremely complex shapes, it is difficult to find a description that can be used for reliability analysis of interfaces. Figure 12 illustrates this issue.

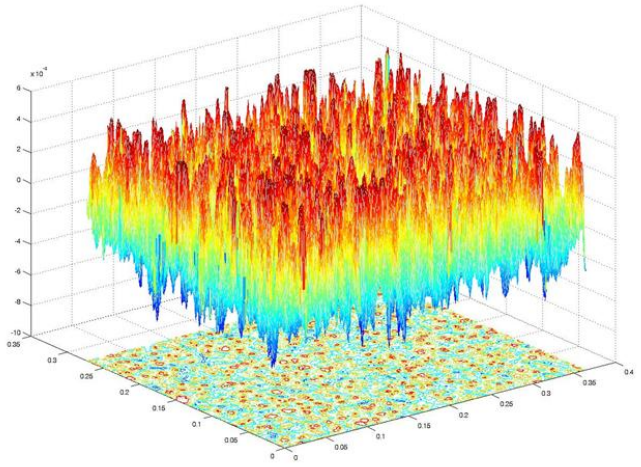


Figure 12: Surface Roughness.

In fact, meshing roughness in a Finite Element Analysis is almost impossible, and the computation will be inefficient or even ineffective. This chapter reexamines the meaning of roughness and develops some models of roughness representation.

Roughness can be characterized by a measure of the texture of a surface. From the idea of a perfectly flat surface, the roughness of an actual surface quantifies the vertical deviations, called imperfections or errors. If these deviations are large, the surface is considered as rough; if they are small, the surface is smooth.

Concerning this project which deals with the metal(-oxide)/polymer interface, the interface roughness refers to the surface roughness of the metal, because the polymer is liquid at the beginning and solidifies at cure, taking form complementary to the metal roughness at the interface.

Several ways to develop roughness models can be identified:

- the deterministic method which gives height amplitudes parameters,
- the statistical method which can be considered as an extension of the previous one dealing with average values and distribution representations of roughness,
- the stochastic models which study the height variation or slope and is really useful in distinguishing surfaces with similar types of roughness and which considers roughness as a random process and field.

  See [Thomas, 1999] for more details.

The present Chapter consists of:

- gathering roughness data: this task takes advantage of the technical process carried out previously in the NanoInterface project by Philips [PhilipsAppliedTechnologies, 2009], especially, it takes advantage of the experimental investigation of failure paths at roughened copper/epoxy interfaces,
- 2D analysis on a trace which is a cross section of the surface, 3D analysis on the entire surface roughness (plots, height distribution) with Matlab, and the comparison between them with some Gaussian distribution,

- using slightly a third party software, MountainsMaps [DigitalSurf, 2011], which is specialized in roughness analysis to see which types of parameters are currently used by institutions.

4.1 Preliminary analysis: roughness measurement and model

4.1.1 Presentation of the studied sample

High quality electro-deposited copper foils called TWS have been selected due to their noticeable surface roughness (see Appendix F). From these, 4 types of copper foils have been chosen according to the following characteristics:

- Thickness: 35 or 70μm (see Figures 13 and 14),
- Roughness treatment of each side: TWS copper foils have a smooth side and a rough one, whereas TWS-TWS copper foils are rough on both sides.

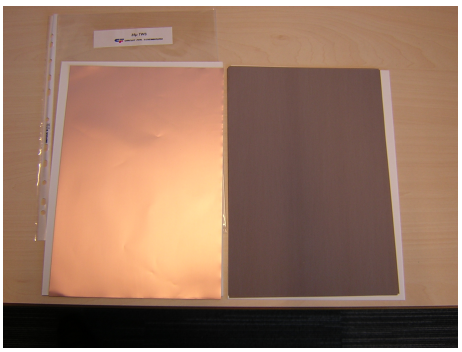


Figure 13: TWS 35μm copper foil.

Figure 14: TWS 70μm copper foil.

4.1.2 Measurements

For each foil side presented previously, the white light interferometry examination (Appendix G) has been performed, which gives the height value z(x,y) for the entire surface roughness. Then, the surface map and the main roughness parameters are deduced (see Figure 15 which presents the roughness information of the TWS-TWS 70μm copper foil #2 side 1).

Table 3 shows the main roughness parameter values of all of the examined TWS-TWS copper foils.

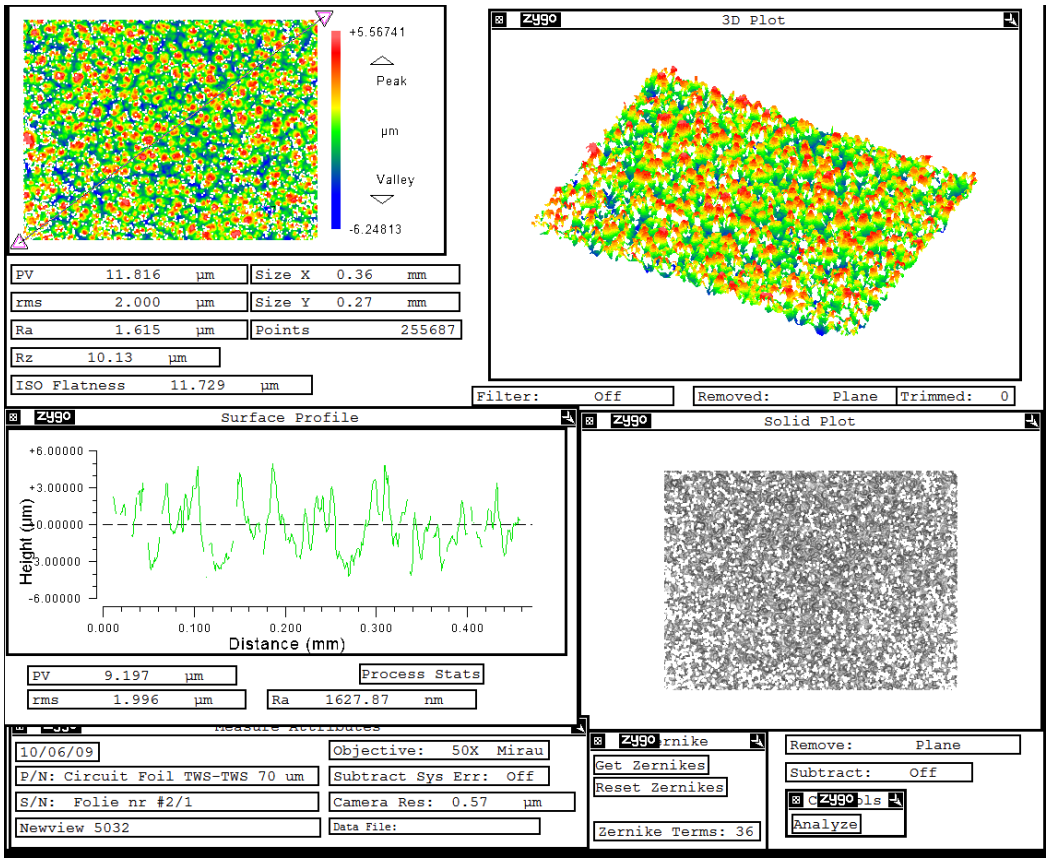


Figure 15: White Light Interferometry measurement on TWS-TWS 70μm copper foil #2 side 1 [PhilipsAppliedTechnologies, 2009].

Note that:

- PV = Maximum height of the profile Rt, which is the easiest manipulation and measure, one could perform it by picking the highest and lowest points of the profile/surface and calculating the distance between them.
- Ra = arithmetic average of absolute values, i.e., $R_a = \frac{1}{L_x L_y} \int_0^{L_x} \int_0^{L_y} |z(x,y)| dxdy$,
- Rz = average distance between the 5 highest peaks and the 5 deepest valleys.

The TWS-TWS 70μm copper foil #2 side 1 is chosen as illustration for the following analysis.

Table 3: Overview of TWS-TWS copper foils roughness parameters (values in μm).

foil thickness	foil#	side	Ra	Rt	Rz
35μm	1	1	0.476	6.905	6.6
		2	1.375	10.729	10.11
	2	1	0.584	6.858	6.61
		2	1.499	11.206	10.33
	3	1	0.537	6.686	6.16
		2	1.824	14.396	10.65
	4	1	0.505	6.565	6.04
		2	1.917	14.269	11.58
	5	1	0.497	6.179	5.8
		2	1.681	12.474	11.65
70μm	1	1	1.592	11.759	10.08
		2	0.466	6.652	6.24
	2	1	1.615	11.816	10.13
		2	0.506	7.298	6.74
	3	1	1.594	12.302	11.02
		2	0.62	7.982	7.37
	4	1	1.7	12.073	9.31
		2	0.558	6.903	6.63
	5	1	1.547	11.158	10.15
		2	0.602	7.944	7.49

4.1.3 2D roughness analysis on foil trace

4.1.3.1 Roughness Profile of the diagonal trace

The measurement consists of the height values of the diagonal trace (cross section of one of the diagonals of the foil shown in the first map on Figure 15) of the TWS-TWS 70μm copper foil #2 side 1 on y coordinate with respect to x coordinate with a constant measurement interval called pitch. The first analysis is to plot (Figure 16) and determine the pitch, which is called roughness profile.

Note that the pitch is a constant calculated by x(i+1)-x(i).

From that, the following roughness parameters have been obtained:

- Amplitude parameter: Rt=9.197μm,
- Average parameters: Ra=1.629μm, Rms=1.996μm, where Rms is the root mean square roughness such that $Rms = \sqrt{\frac{1}{L_x L_y} \int_0^{L_x} \int_0^{L_y} |z(x,y)|^2 dxdy}$

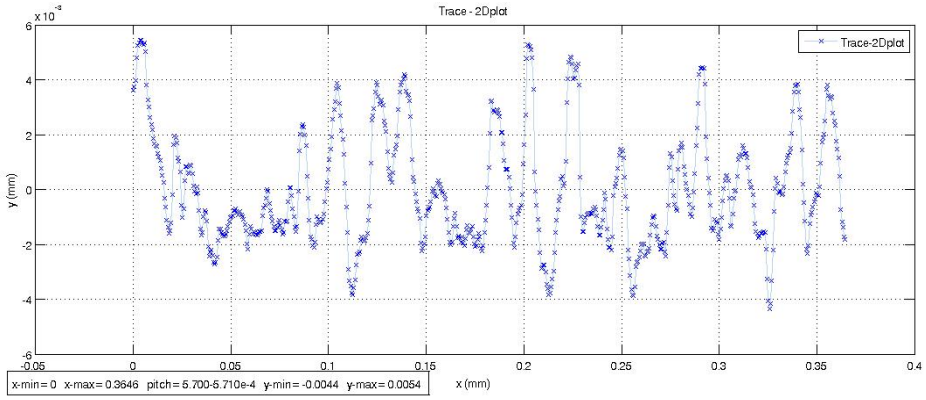


Figure 16: Diagonal trace of the studied sample.

4.1.3.2 Height distribution

The roughness profile from the trace plot can be drawn in such a way that the x coordinate and the y coordinate are inversed so that the number of height points encountered for each given level of height is underlined (see Figure 17).

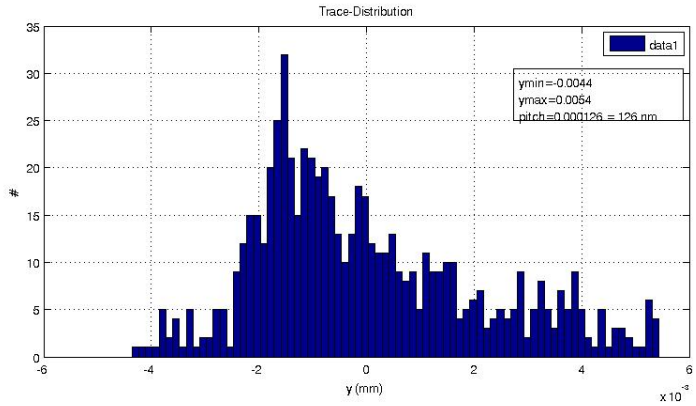


Figure 17: Height distribution of the diagonal trace of the studied sample.

The number of bins influences the distribution. It has been shown that 10 – 20 bins are sufficient to obtain the roughness information [Thomas, 1999]. Let the number of bins be reduced so that peak (curve) fitting can be performed easily to see which known distributions describe the height distribution the best (see Figure 18). It is obvious that the current case is not far from being a normal distribution. It can be seen that the Dagum (4P) suits better than before the Lognormal and normal ones.

Note that the Dagum (4P) distribution fits the best with k=79.507, α=72.257, β=0.11328,

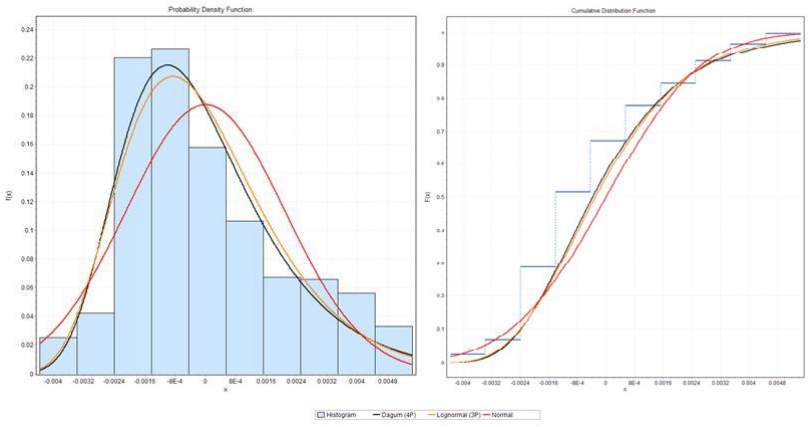


Figure 18: Peak fitting of the diagonal trace height distribution of the studied sample.

γ=-0.12133. Then, there is the Lognormal (3P) one with deviation of σ=0.29578, mean of μ=5.0111, γ=-0.00896. And, the Normal distribution suits with deviation of σ=0.00208 and mean of $\mu \approx 0$.

4.1.3.3 Stochastic descriptions

Physically speaking, up to here, roughness has been depicted only coarsely in terms of amplitude, but it is of utmost importance to know how quickly the height changes with position i.e., how to describe slopes (see Figure 19). This refers mainly to the utilization of random process and field which describe roughness stochastically. Stochastic modeling of surface roughness, surface height, slope, topography wavelengths, etc. have to be considered as a random variable from probability distribution and are modeled by height distribution functions, density functions, power spectrum, and auto-correlation functions.

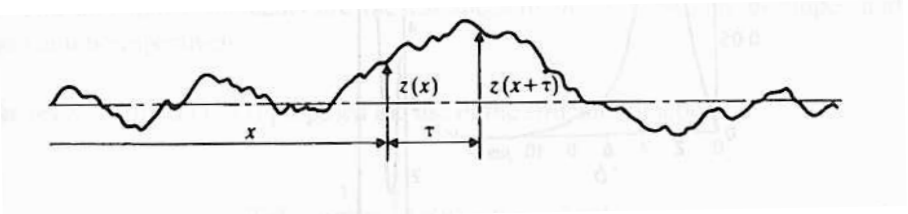


Figure 19: Basic description of fluctuation [Thomas, 1999].

Describing slopes, called also fluctuations corresponds to a large extent in depicting surface texture by distribution of those fluctuations. This is done by investigating the correlation between pairs of points which vary.

Core roughness Rk, peak height Rpk and valley depth Rvk are obtained as Figure 20 suggests. Core roughness *Rk* represents the central and the flattest portion. It is calculated as the distance between the two intersections of the straight template covering 40% of the total bearing area curve with zero and maximal values of the cumulative distribution. Peak height *Rpk* and valley depth *Rvk* are defined as the area above and below the core roughness *Rk* respectively.

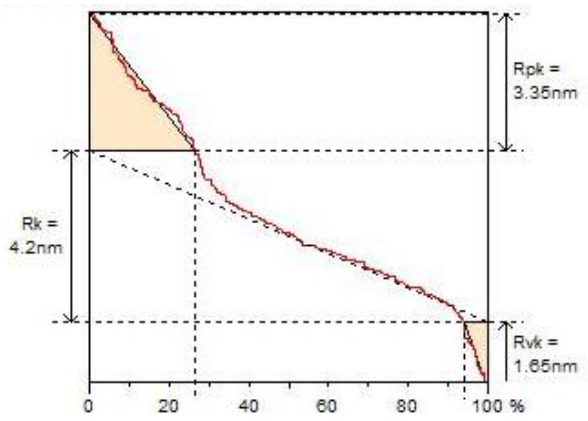


Figure 20: Core roughness, peak height and valley depth of the studied sample [DigitalSurf, 2011].

In the same way, the spectrum of the trace is performed (see Figure 21).

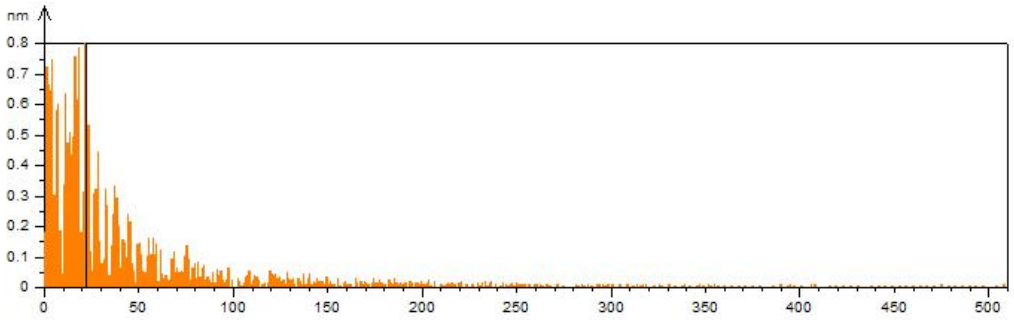


Figure 21: Spectrum of the diagonal trace of the studied sample [DigitalSurf, 2011].

The power spectrum is an expression of the fluctuation in frequency space. In detail, random fluctuation is grasped as the consolidation of multiple wave forms of different wavelengths and amplitudes in frequency space. It is derived from Fourier transformation expressing the squared average of fluctuations as a function of spatial coordinate x, which can be seen

as the power distribution:

$$P(f_x) = \lim_{T\to\infty} \frac{1}{T^2} | \int_{-T/2}^{T/2} y(x) e^{-\imath 2\pi (f_x x)} dx|^2, \tag{46}$$

where the assessment length varies from $-T/2$ to $T/2$, f_x is the reciprocal of the wavelength in x direction and y the height.

Small values of the horizontal axis in Figure 21 correspond to small wavelengths or high fluctuations whereas higher values reflect the curvatures of the profile.

Some typical roughness information taken from a trace has been presented. Section 4.1.4 shows the analysis of the surface roughness. A comparison between them will be discussed in the conclusion.

4.1.4 3D roughness analysis on foil surface

4.1.4.1 Surface roughness profile

Here are the 3D and 2D plots (see Figures 22 and 23) showing the roughness of the entire surface of TWS-TWS 70μm copper foil #2 side 1.

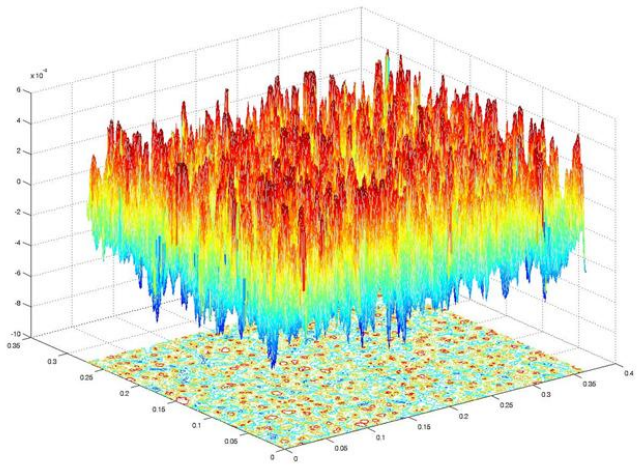


Figure 22: Surface Roughness in 3D of the studied sample.

From that, the following roughness parameters have been obtained:

- Amplitude parameters: St=11.5μm, Sp=5.51μm, Sv=5.96μm, Sz=11μm,
- Average parameters: Sa=1.54μm, Sq=1.91μm,
- Higher moments: $S_{Skewness}$=0.423, $S_{Kurtosis}$=2.79.

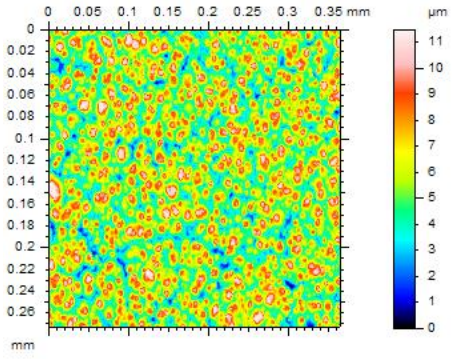


Figure 23: Surface Roughness in 2D of the studied sample.

4.1.4.2 Height distribution

The height distribution and the peak fitting can be performed. Here the distribution can be approximated to a normal distribution, then the Lognormal as well as the Wakeby distribution (see Figures 24 and 25).

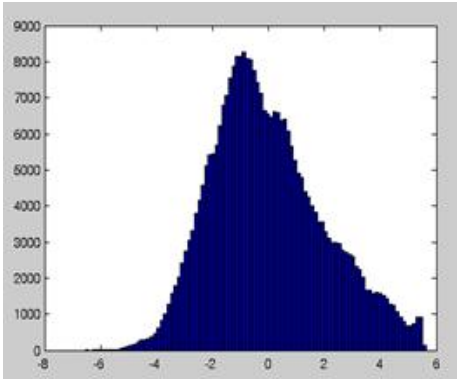


Figure 24: Surface roughness distribution of the studied sample.

Note that the Wakeby fits the best with α=0.02411, β=12.962, γ=0.00325, δ=-0.35304 and ξ=-0.00413. The Lognormal (3P) one suits with deviation σ=0.17988, mean of μ=-4.5102 and γ=-0.01118. And the normal one fits with deviation of σ=0.00202, mean of $\mu \approx 0$.

4.1.4.3 Stochastic descriptions

The core roughness Rk, the peak height Rpk and the valley depth Rvk can be obtained as Figure 26 suggests.

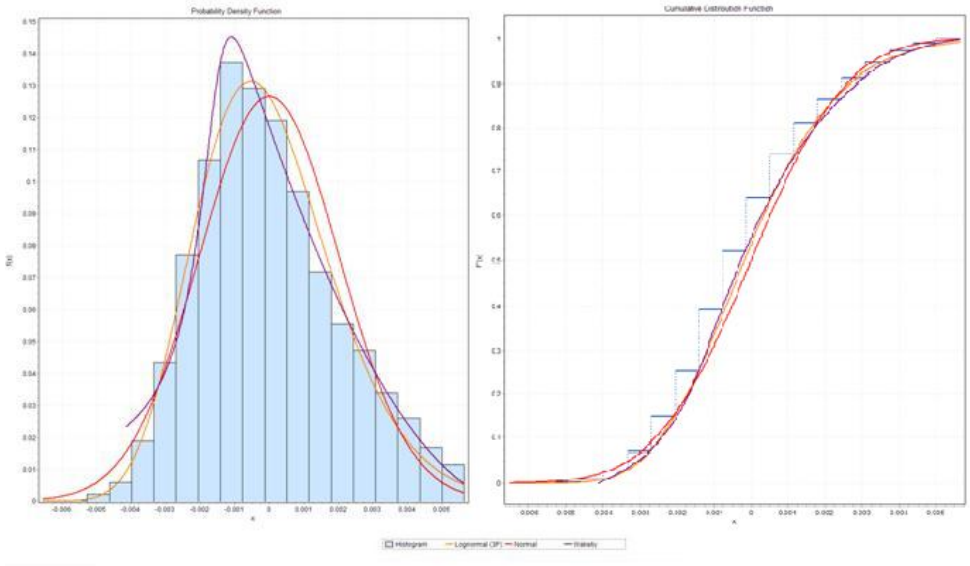


Figure 25: Peak fitting of the surface roughness distribution of the studied sample.

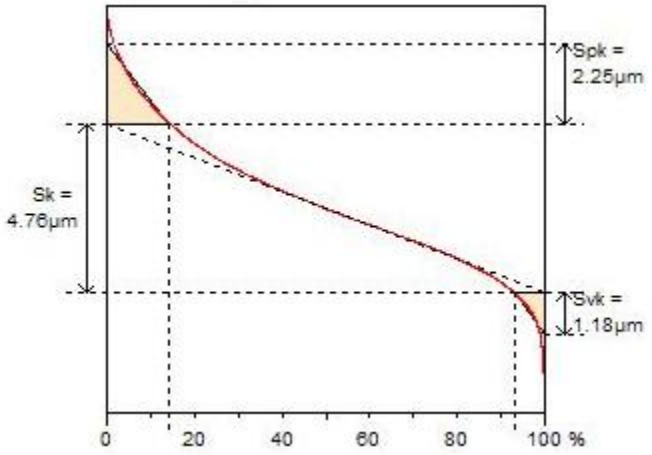


Figure 26: Core roughness, peak height, valley depth of the studied sample [DigitalSurf, 2011].

This section is a preliminary application of the surface roughness representation. Deterministic models such as amplitude parameters, and statistical ones, such as average parameters, core roughness, peak height and valley depth were obtained. Some stochastic representations like power spectrum were obtained.

It can be underlined that, even if at first glance, both the diagonal trace extended to any trace, and the surface have the same roughness information in terms of amplitude and average parameters, their height distribution is different. The trace distribution tends to be a Dagum one, while the surface distribution looks as a Gaussian one. This is confirmed by

core roughness, peak height and valley depth parameters. Any trace can catch the amplitude of the surface roughness, whereas the surface roughness fluctuations are not perceived in the same way as in a trace. Hence, traces are not representative of the surface roughness. In conclusion, maximum height R_z, average roughness R_a, average wavelength λ_a (see [Thomas, 1999, Yao and Qu, 2002]), moments from the height distributions and the core roughness describe roughness easily and straightforwardly and are means of comparing roughnesses.

This work can be continued in order to get more rigorous and thorough representations and to define in a rigorous way the stochastic variables which depict the surface roughness.

4.2 Application of the Size sensitivity based Perturbation method to Finite Element roughness model

Typical roughness model used in the project which is taken from [Lallemant, 2011] (see Figure 27).

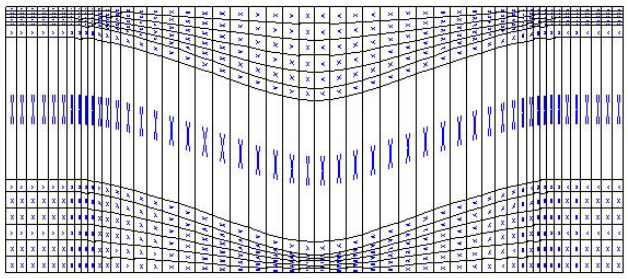


Figure 27: Numerical simulation of the delamination of a roughened copper/metal interface [Lallemant, 2011].

Briefly, this model comes from the constitution of a unit cell which can be understood as a pattern of the ideal profile from [Yao and Qu, 2002] (see Figure 3). It consists of the upper polymer layer and the lower metal (copper for example) layer, both discretized with linear, four-node, isoparametric, plane strain, quadrilateral elements, and a layer of cohesive zone element modeling the adhesion between them by the traction separation law. More explicitly, when the upper and the lower layers are pulled out through prescribed displacement at the top left side, the upper layer is delaminated from the lower one, and the adhesion properties are stocked in the cohesive zone element drawing the traction separation

law curve (see Figure 28). The main adhesion properties can be determined from this curve:

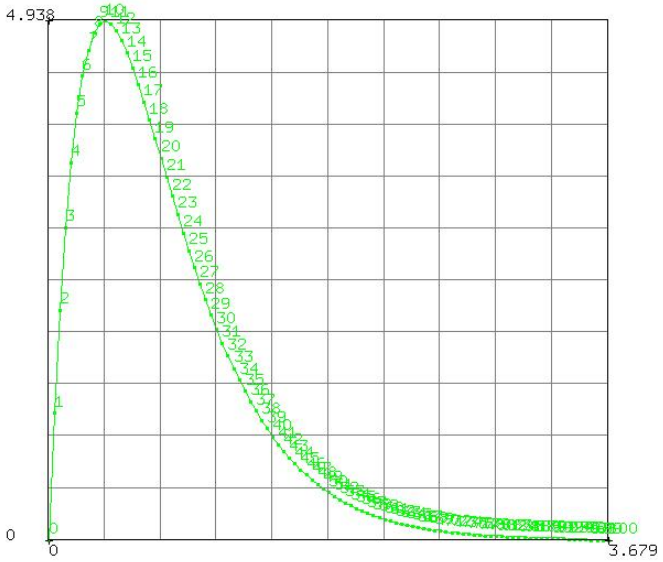


Figure 28: Traction separation law curve.

- the maximum of the curve representing the adhesion strength of the interface,
- the area below the curve representing the toughness G of the adhesion,
- the 'openness' of the curve showing the ductility or the brittleness.

The aim is to increase the adhesion strength and toughness of the interface.

The main question that can posed at this point is how the Size sensitivity based Perturbation method implementation and especially the scaling of Size sensitivity computation influences the Finite Element roughness modeling? Also, does the scaling take place globally or locally? The scaling on elements which are in the 'right' direction, i.e., x and y directions, like all applications done up to here (one element tensile test, 3 point bending test on a beam) is obviously scaled in x and y directions. However, one might wonder how an element which is neither placed in x nor y direction, is treated with this Size sensitivity computation.

One element tensile test in which the element is in oblique is considered as in Figure 29.

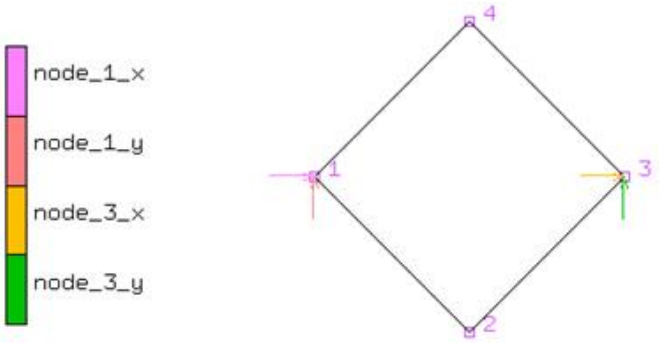


Figure 29: 1D tensile test with one element 11 in oblique.

The influence of a scaling in the vertical direction is studied.

In this Figure, the prescribed displacement in x direction in node 3 is 0.01 and others are fixed.

This model will run with and without a scaling factor of 3 in y direction. The results are gathered and compared in Table 4.

Table 4: Comparison of the displacement field of a scaled and a non scaled 1D oblique one element tensile test.

Displacement field	$s_y = 1$	$s_y = 3$
Node 1 x	0	0
Node 1 y	0	0
Node 2 x	0.05	0.05
Node 2 y	0.00333	0.001
Node 3 x	0	0
Node 3 y	0.01	0.01
Node 4 x	0.05	0.05
Node 4 y	-0.00333	-0.001

This means that whichever the orientation of the element, scaling affects the displacement field such that the element is scaled only in the direction considered without taking into account the direction of elements. Moreover, even if the element is scaled locally at the element level, the scaling can be considered in a global way, i.e., if scaling of 2 in the vertical direction (at the element level) is chosen, the vertical amplitude (R_a) of the roughness will be twice higher so that the roughness will be sharper.

From the simple roughness model based on the previous one with the average roughness and wavelength as descriptor parameters, Figure 30 illustrates this purpose:

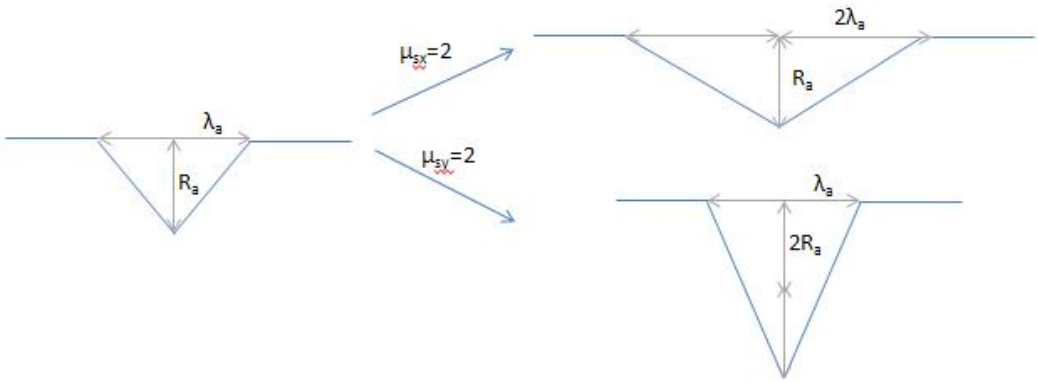


Figure 30: Influence of the scaling on the roughness modeling.

Scaling in x direction increases the average wavelength such that the roughness becomes smoother, whereas scaling in y direction makes the average roughness higher so that it becomes sharper.

Moreover, the study of the variability of the displacement field at the interface roughness is wanted (see Figure 31). This is the main purpose of the Size sensitivity based Perturbation method.

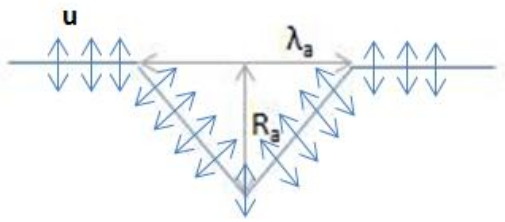


Figure 31: Varibility of displacement field at the roughness.

Therefore, the following numerical simuation is carried out (see Figure 32). The variability of displacement field in y direction is studied by pulling the roughness pattern in x direction (see Figure 33). The left extremity is clamped while the right extremity is pulled of the amount of $dx = 5\%\ of\ the\ roughness\ pattern$. The bottom part and the top right node

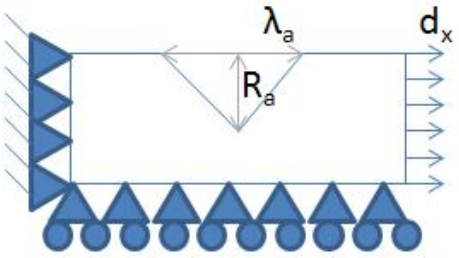


Figure 32: Scheme of the numerical simulation of the varibility of displacement field at the roughness.

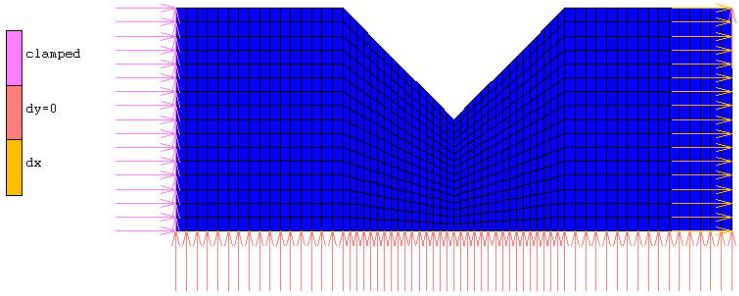


Figure 33: Numerical simulation of the varibility of displacement field at the roughness.

are clamped in y direction. The opening of the roughness is $\frac{\pi}{4}$. And, $\mu_{s_x} = 1$, $\mu_{s_y} = 1$, $\sigma_{s_x} = 0.1$, $\sigma_{s_y} = 0.1$ and $\rho_{s_x - s_y} = 0$.

The implementation gives the variability of the displacement field at the interface roughness. Figure 34 shows the contour bands drawing the evolution of the mean of y displacement.

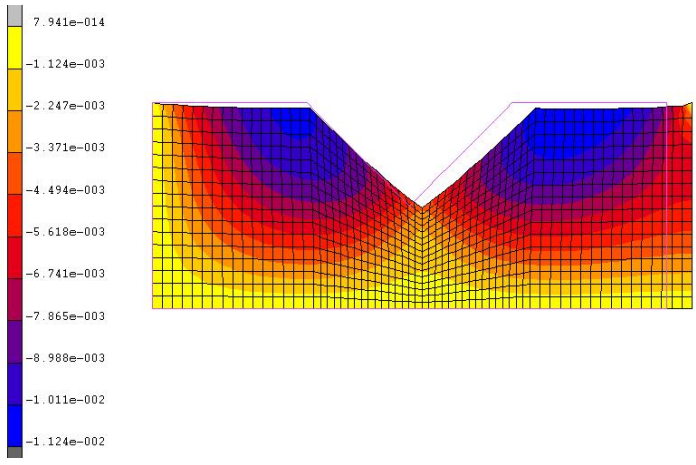


Figure 34: Contour bands drawing the mean of the y displacement.

The Size sensitivity based Perturbation can evaluate the variability of the displacement

field.

The variability of adhesion properties are based on the variability of the displacement field. The implementation can be integrated to the metal(-oxide)/polymer roughened interface adhesion properties calculation based on numerical fracture mechanics simulations, with for instance cohesive zone element, presented briefly above and in [Noijen et al., 2009]. Therefore, the variability of adhesion properties such as strength and toughness can be obtained.

To sum up, the Size sensitivity based Perturbation method implementation:

- requires as input, the mean, standard deviation and the coefficient of correlation of the scaling in x and y directions,
- gives as output, the displacement, strain, stress fields (plus other outputs available in the post processing of Marc/Mentat with linear, four-node, isoparametric, plane strain, quadrilateral element), and the first two moments of the displacements fields at each node.

Lastly, a limitation seen here is that the model with the scaling is not drawn on the Mentat interface, since the Size sensitivity computation is within the element calculation as the name suggests.

The variation on the strain and stress fields can be evaluated from the current output of the implementation.

In addition this Size sensitivity based Perturbation method can be applied to other elements such as quadratic quadrilateral element, etc.

Chapter V

CONCLUSION AND RECOMMENDATIONS

The Size sensitivity based Perturbation method of the Stochastic Finite Element Methods has been implemented to ameliorate the development of Finite Element roughness models. The Perturbation method is chosen, because it can investigate the effect of geometric variations on the mechanical response, and it can carry out at once what traditional Finite Element Analysis does with numerous simulations that require changing geometric parameters each time. This was computed on an user defined element reproducing the linear, four-node, isoparametric, plane strain, quadrilateral element, because roughness models use mainly this element, and it works as a module on the FE software. It allows performing:

- linear, four-node, isoparametric, plane strain, quadrilateral element itself,
- Size sensitivity computation in which the scaling can be either uniform or non uniform so that the input is composed of scale factors in x and y directions and the output composed of reaction force, displacement, stress, strain and derivatives of the displacement fields,
- Perturbation method where the Size sensitivity computation is run with the mean, the variance and the coefficient of correlation of the scale factors as input. The mean, the standard deviation of the displacement field as output are calculated by post-processing of the output of the Size sensitivity computation.

The Perturbation method has been derived analytically and applied to the so-called 3 point bending test on a beam problem as reference to verify that the implementation leads to the same result. Moreover, a preliminary analysis of roughness modeling which has been shown, motivates giving recommendations to define a better statistical and stochastic roughness representation.

As a recommendation, the implementation of the Reliability method can be carried out to complete the investigation of predicting the chance of different failure modes (cohesive or adhesive) during the delamination process. Indeed this method is typically used to estimate the probability of a rare and undesirable event such as failure.

Based on linear elastic fracture mechanics from [Noijen et al., 2009], the following limit state functions can be defined: $g_1 = \frac{G_{i,c}}{G_{p,c}} - \frac{G_i}{G_p}$ and $g_2 = G_i - G_{i,c}$ where G_i and G_p are the crack tip energy release rates along the interface and in the polymer, respectively, and $G_{i,c}$ and $G_{p,c}$ are the respective interface toughness and polymer toughness. If $g_1 > 0$ and $g_2 > 0$, the interfacial crack goes into the polymer, i.e., crack kinking occurs, which means that the failure mode changes from adhesive to cohesive. This phenomenon has to be reached most of the time and the Reliability method can assess its probability by approximation, from some roughness parameters such as the average roughness $z_1 = R_a$, the average wavelength $z_2 = \lambda_a$ as inputs.

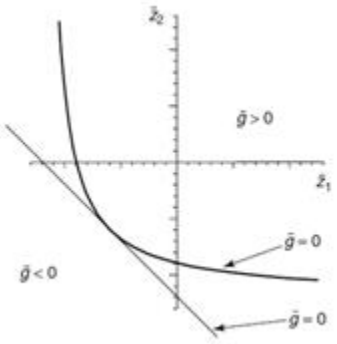


Figure 35: The domain for Reliability method with linear approximation [Gutiérrez and Krenk, 2004].

This completes the research, and the implementation of the Perturbation method shows the way to add a stochastic framework in the Finite Element model of surface roughness on the metal oxide polymer interface in order to predict the chance of different failure modes (cohesive or adhesive) during delamination. The roughness can be modeled such that the adhesion properties of metal(-oxide)/polymer interface is enhanced by the mechanical interlocking phenomenon so that semiconductors can become more reliable.

Appendix A

NANOINTERFACE PROJECT

The present thesis is a part of the NanoInterface Project [PhilipsAppliedTechnologies, 2009], partially funded by the seventh framework program of the European Commission, FP7, which aims to develop a new powerful, quantitative, knowledge-based, multi-scale modeling technique in order to develop new micro- and nano-electronic products with tailored properties and which are more reliable, so that trial-and-error-based reliability testing for those new technologies can also be considerably reduced. Ten partners shown in Figure 36 coming from industry, research laboratories and educational institutions are involved.

The project relies on the direct coupling of molecular and continuum models, and involves multi-disciplinary approach, multi-scale modeling methods (see Figure 37), and new micro- and nano-scale measurement techniques.

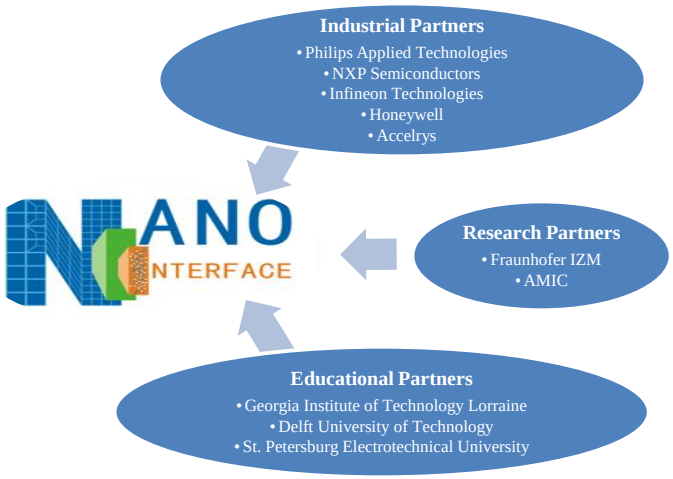


Figure 36: NanoInterface partners.

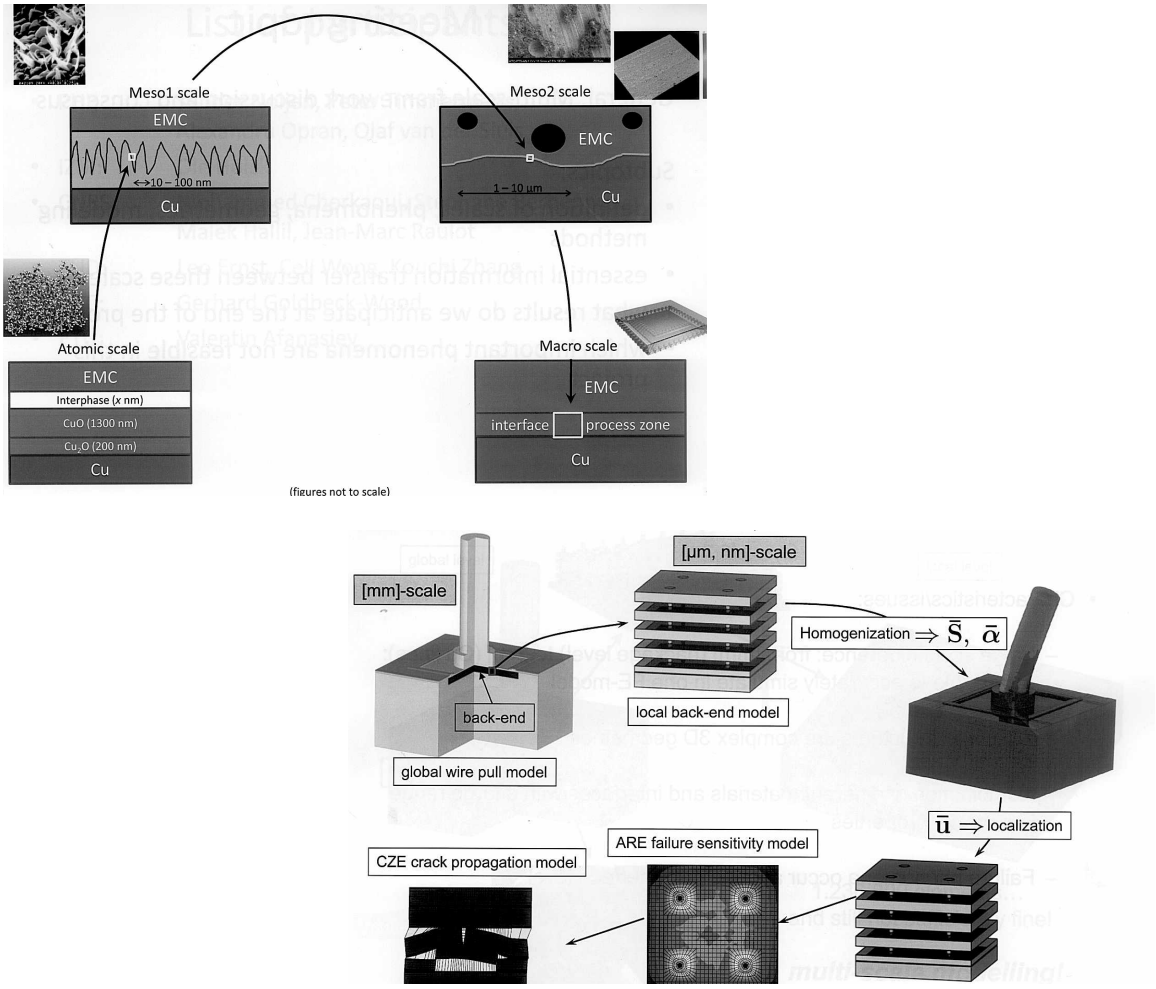


Figure 37: Introduction to multi-scale modeling [Van der Sluis, 2006b].

Appendix B

STOCHASTIC FINITE ELEMENT METHODS

This Chapter aims to explain the basics of the most interesting SFEM in the framework of the project: Monte Carlo Simulations, Perturbation method and the Reliability methods. The general formulation of each method will be esoteric. So, each method is carried out in the same generic mechanical problem – 1D tensile test in order to understand its principle.

B.1 1D tensile test

The 1D tensile test consists of a homogeneous, isotropic, linear elastic bar with stiffness k subjected to an axial load q:

Figure 38: 1D tensile test [Gutiérrez and Krenk, 2004].

The problem is governed by:

$$ku = q \leftrightarrow u = \frac{q}{k}, \tag{47}$$

where **u** is the vector containing the values of displacements at the nodes, **k** is the stiffness matrix, and **q** is the loading.

The goal is to find out the statistical properties (mean and standard deviation) of the random displacement $\tilde{u}$ from those of the random stiffness, $\tilde{k}$, i.e.:

$$\tilde{u} = \frac{q}{\tilde{k}}. \tag{48}$$

The mean and the standard deviation of the displacement are:

$$\mu_u = \mu_{1/k} q \quad and \quad \sigma_u = \sigma_{1/k} q. \tag{49}$$

As the FEA suggests, the bar is discretized into n identical elements with $\tilde{k}_i$ stiffness, as

Figure 39 shows. The assembled stiffness becomes:

Figure 39: Discretized 1D tensile test [Gutiérrez and Krenk, 2004].

$$\frac{1}{\tilde{k}} = \frac{1}{\tilde{k}_1} + \cdots + \frac{1}{\tilde{k}_n}. \tag{50}$$

As all elements are identical, the mean and the standard deviation of the displacement are:

$$\mu_{1/k} = n\mu_{1/k_1} \quad and \quad \sigma^2_{1/k} = \sigma^2_{1/k_1}[n\frac{1+\rho}{1-\rho} - 2\rho\frac{1-\rho^n}{(1-\rho)^2}], \tag{51}$$

where $-1 < \rho < 1$ is the correlation coefficient between each element. If $\rho = 0$ the property of individual elements are independent, if not they are correlated. If ρ tends to 1, elements are fully correlated and $\sigma_{1/k} = n\sigma_{1/k_1}$.

As the first two moments of the assembled stiffness, those of the displacement are known.

B.2 Monte Carlo simulation (MCS)

Monte Carlo simulation consists in performing NSIM simulations as the traditional FEA does but with stochastic system matrix. MCS is obviously the simplest method for treating the response variability calculation in the framework of SFEM. This method leads to a set of the response vector. Based on this set, the response variability of the system is calculated using simple relationships of statistics. If we take the example of the nodal displacement $\tilde{u}_i$, its mean and standard deviation are:

$$\mu_{u_i} = \frac{1}{NSIM}\sum_{j=1}^{NSIM} u_i(j) \quad and \quad \sigma_{u_i} = \sqrt{\frac{1}{NSIM-1}[\sum_{j=1}^{NSIM} u_i^2(j) - NSIME^2(u_i)]}. \tag{52}$$

The accuracy of these estimations depends on the number of samples NSIM. Moreover, the estimation of standard deviation is inversely proportional to $\sqrt{NSIM}$. A small number of samples, like 50, permits only a rough approximation of the mean value and standard deviation of the response. While, a larger sample size, like 500, enables to estimate the Cumulative Distribution Function of the response.

Besides this direct MCS, several variants MCS, such as the fast MCS exist [Gutiérrez and Krenk, 2004, Stefanou, 2009]. This method is closely linked with the reliability method which will be treated below. Methods such as importance sampling, subset simulation and line sampling are both variants of MCS and are based on the reliability method.

B.3 Perturbation Method

The perturbation method aims to calculate the two first moments (mean and standard deviation) of the response from the moments of the input variable. This is based on the Taylor series expansion at mean of the input, as follows:

$$u(k) = u(\mu_k) + \partial_k u(\mu_k)(k - \mu_k) + \frac{1}{2}\partial^2_{kk} u(\mu_k)(k - \mu_k)^2 + \cdots \tag{53}$$

The response is written as follows in 1D case:

$$\mu_u \approx u(\mu_k) + \frac{1}{2}\partial^2_{kk} u(\mu_k)\sigma^2_k \qquad and \qquad \sigma_u \approx \partial_k u(\mu_k)\sigma_k. \tag{54}$$

This method is applied to the 1D tensile test. The difficult part here is to determine the first and the second derivative of stiffness. The way to get them is to differentiate the equation 47 with respect to the stiffness:

$$u + k\partial_k u = 0. \tag{55}$$

Substituting for $k = \mu_k$ and solving reads:

$$\partial_k u(\mu_k) = -\frac{q}{\mu^2_k}. \tag{56}$$

Differentiating again, i.e., to the second order, the equation 47 permits to obtain the second derivative of the stiffness:

$$2\partial_k u + k\partial^2_{kk} u = 0. \tag{57}$$

Hence,

$$\partial^2_{kk} u(\mu_k) = \frac{2q}{\mu^3_k}. \tag{58}$$

By the way, the zero order term is provided by $k = \mu_k$ directly:

$$u(\mu_k) = -\frac{q}{\mu_k}. \tag{59}$$

Finally, we get a measure of the statistical variability of the response $\tilde{u}$ by its mean and standard deviation:

$$\mu_u \approx \frac{q}{\mu_k}[1 + (\frac{\sigma_k}{\mu_k})^2] \quad and \quad \sigma_u = \frac{q\sigma_k}{\mu_k^2}. \tag{60}$$

B.4 Spectral Finite Element Method

The spectral finite element method is an extension of the deterministic finite element method for the solution of boundary value problems with random material properties. This method is in stand by along the study because it is more dedicated to the material sensitivity. However, the reader is referred to [Gutiérrez and Krenk, 2004, Stefanou, 2009, Sudret and Der Kiureghian, 2000] for details. However, note that the method expresses the response as a series development in orthogonal polynomials, which is known as the Hermite polynomial, or polynomial chaos expansion.

B.5 Reliability Method

This method is typically used to estimate the probability of a rare and undesirable event such as failure. This probability can only be obtained in an approximate sense.

The most well known and general version is based on converting the original stochastic variables of the problem into independent normal variables and then using geometric arguments to identify the most likely modes of failure and the associated probabilities. An adequacy by construction between the geometric space and the normal distribution of converted input is carried out. Then the probability density is only a function of the distance from the center, representing the expected value.

This method is illustrated with the 1D tensile test discretized in 2 elements. Two stiffnesses take place in the analysis. The generic definition is presented in the following.

The random stiffness $\tilde{k}$ is assumed to follow a normal distribution with expectation μ_k and standard deviation σ_k, and is expressed in terms of a standard normal variable $\tilde{z}$ such that:

$$\tilde{k} = \mu_k + \sigma_k \tilde{z}. \tag{61}$$

The displacement must not exceed the threshold u_0 which is equal to q/k_0. This value is converted in the normalized variable $\tilde{z}$ as:

$$z_0 = \frac{k_0}{\sigma_k} - \frac{\sigma_k}{\mu_k}. \tag{62}$$

The so-called limit state function can be introduced such that:

$$\tilde{g} = u_0 - \tilde{u} = u_0 - q(\frac{1}{(\mu_k + \sigma_k)\tilde{z}_1} + \frac{1}{(\mu_k + \sigma_k)\tilde{z}_2}) \tag{63}$$

This function marks the limit between the 'safe' domain, $\tilde{g} > 0$, from the 'unsafe' one, $\tilde{g} < 0$, as the following Figure shows.

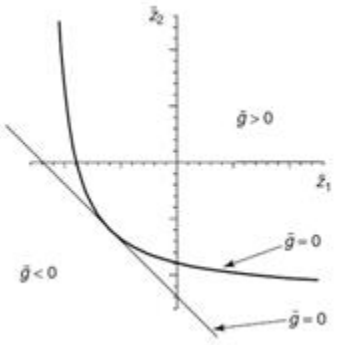


Figure 40: The domain for reliability method with the linear approximation [Gutiérrez and Krenk, 2004].

The probability that $\tilde{u}$ exceeds u_0 is given by:

$$Pr(\tilde{u} > u_0) = Pr(\tilde{g} < 0) = \int_{g<0} \phi_2(z_1, z_2, 0) dz_1 dz_2 \tag{64}$$

where ϕ_2 represents the bivariate standard normal probability density function with the correlation coefficient as the third argument which is equal to 0 here because $\tilde{z}_1$ and $\tilde{z}_2$ are independent.

But the curve $\tilde{g}$ is far from being easy to find out. The essence of the geometric reliability method is to approximate the boundary g = 0 by a simpler curve such as a line (First Order Reliability Method) or a quadratic curve (Second Order Reliability Method).

The linear approximation of g is expressed as:

$$\bar{g} = \alpha_1 z_1 + \alpha_2 z_2 + \beta = \alpha^T \mathbf{z} + \beta \tag{65}$$

where α is a unit normal vector and β is the distance from the line to the center of the coordinate system (see Figure 40).

The approximation point is chosen such that the line $\bar{g} = 0$ is tangent to the limit state $g = 0$ at the point closest to the origin. So, the best linear approximation to the probability is:

$$\int_{g<0} \phi_2(z_1, z_2, 0) dz_1 dz_2 = \Phi(-\beta), \tag{66}$$

where β is called design point or β-point of the reliability index. It is also observed that α represents the outward normal vector to the set $g < 0$ at the approximation point, which consequently has coordinates $z = \alpha$. Here the solution is explicitly:

$$\beta = \sqrt{2}|\frac{2q}{\sigma u_0} - 1| \quad and \quad \alpha = [\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}]^T. \tag{67}$$

This solution is illustrated in Figure 41. The method is easily generalized to n elements, but in that case, the correlation between the stiffness of the individual elements must be taken into account.

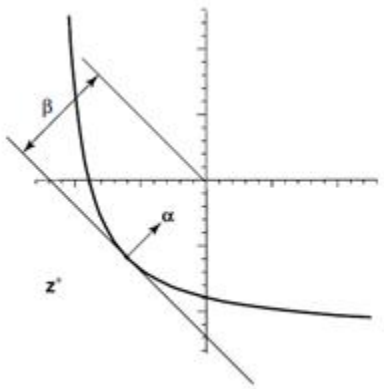


Figure 41: Geometric interpretation of the design point [Gutiérrez and Krenk, 2004].

The principles of Monte Carlo Simulations, Perturbation method and Reliability method have been presented by means of application to a generic mechanical problem – 1D tensile test. The modeling of the uncertainty characterizing input using the theory of stochastic functions – processes/fields was not dealt here even if it is a huge part of the SFEM. However, the general idea and the aim of each method are intended to be understood.

Appendix C

DEFLECTION OF THE 3 POINT BENDING TEST ON A BEAM

Figure 42 presents the 3 point bending test on a beam problem.

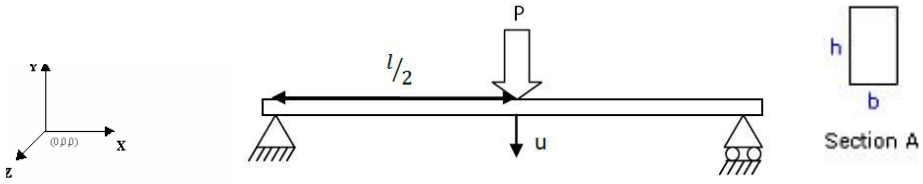


Figure 42: 3 point bending test on a beam problem.

This section is devoted to expressing the deflection.

C.1 Geometric property

The second moment through z, I_{Gz}, of the rectangular cross section is expressed as:

$$I_{Gz} = \iint_A y^2 dS = \int_{-b/2}^{b/2} dz \int_{-h/2}^{h/2} \tag{68}$$

Hence,

$$I_{Gz} = \frac{bh^3}{12}. \tag{69}$$

C.2 Boundary condition

The beam is subject to the following external forces:

$$\boldsymbol{F_{ext \to beam}} = \frac{P}{2}\boldsymbol{e_y}\delta_{x=0} - P\boldsymbol{e_y}\delta_{x=l/2} + \frac{P}{2}\boldsymbol{e_y}\delta_{x=l}. \tag{70}$$

C.3 Tensor of cohesion

The equilibrum equation leads to the equivalence of the tensor of cohesion and tensor of external force such that:

- if $l/2 < x < l$,

  the reaction force $\boldsymbol{T} = \frac{P}{2}\boldsymbol{e_y}$,

  the moment $\boldsymbol{Mf} = (l - x)\frac{P}{2}\boldsymbol{e_z}$,

- if $0 < x < l/2$,

  the reaction force $\boldsymbol{T} = -\frac{P}{2}\boldsymbol{e_y}$,

  the moment $\boldsymbol{Mf} = x\frac{P}{2}\boldsymbol{e_z}$.

C.4 Displacement field

The Euler Bernoulli beam theory states that:

$$\frac{\partial^2 u_y}{\partial x^2} = \frac{Mf}{EI_{Gz}}. \tag{71}$$

Hence, the dispalcement in y direction reads:

$$u_y(x) = \frac{Px}{4EI_{Gz}}(\frac{x^2}{3} - \frac{l^2}{4}). \tag{72}$$

Then, the deflection is expressed as:

$$\delta = |u_y(l/2)| = \frac{Pl^3}{48EI_{Gz}}. \tag{73}$$

The following Figure sums up the distribution of the reaction force T, moment Mf and the displacement through y, u_y.

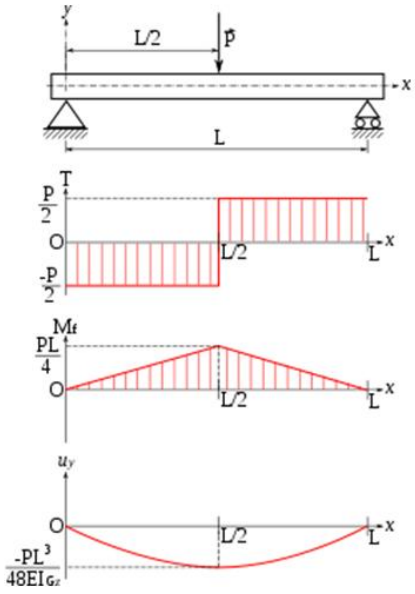


Figure 43: Distributions of the reaction force T, moment Mf and the displacement through y, u_y, of the 3 point bending test on a beam problem [Dau, 2005].

Appendix D

ON THE EQUILIBRUM SYSTEM

D.1 Derivation of the components of stiffness system matrix and the internal force vector

The derivation of $\frac{\partial \boldsymbol{K}_e^s}{\partial s_i}$, $\frac{\partial^2 \boldsymbol{K}_e^s}{\partial s_i^2}$ and $\frac{\partial^2 \boldsymbol{K}_e^s}{\partial s_i s_j}$ for $(i,j) \in \{(x,y),(y,x)\}$ of the equilibrum system 43 are presented here.

From section 3.1.2, components of the stiffness matrix, $\boldsymbol{K}_e^s$, can be expressed as:

$$\boldsymbol{k}_{e\,ij}^s = \sum_{ij} \omega_i \omega_j J^s \begin{pmatrix} N_{i,x}^s D_{11} N_{j,x}^s + N_{i,y}^s D_{33} N_{j,y}^s & N_{i,x}^s D_{12} N_{j,y}^s + N_{i,y}^s D_{33} N_{j,x}^s \\ N_{i,y}^s D_{12} N_{j,x}^s + N_{i,x}^s D_{33} N_{j,y}^s & N_{i,y}^s D_{11} N_{j,y}^s + N_{i,x}^s D_{33} N_{j,x}^s \end{pmatrix}. \tag{74}$$

By expanding terms with s_x and s_y:

$$\boldsymbol{k}_{e\,ij}^s = \sum_{ij} \omega_i \omega_j J \begin{pmatrix} \frac{s_y}{s_x} N_{i,x} D_{11} N_{j,x} + \frac{s_x}{s_y} N_{i,y} D_{33} N_{j,y} & N_{i,x} D_{12} N_{j,y} + N_{i,y} D_{33} N_{j,x} \\ N_{i,y} D_{12} N_{j,x} + N_{i,x} D_{33} N_{j,y} & \frac{s_x}{s_y} N_{i,y} D_{11} N_{j,y} + \frac{s_y}{s_x} N_{i,x} D_{33} N_{j,x} \end{pmatrix}. \tag{75}$$

Hence, the first and second order derivatives of the stiffness matrix are expressed as follows:

$$\frac{\partial \boldsymbol{k}_{e\,ij}^s}{\partial s_x} = \sum_{ij} \omega_i \omega_j J \begin{pmatrix} \frac{-s_y}{s_x^2} N_{i,x} D_{11} N_{j,x} + \frac{1}{s_y} N_{i,y} D_{33} N_{j,y} & 0 \\ 0 & \frac{1}{s_y} N_{i,y} D_{11} N_{j,y} + \frac{-s_y}{s_x^2} N_{i,x} D_{33} N_{j,x} \end{pmatrix}, \tag{76}$$

$$\frac{\partial \boldsymbol{k}_{e\,ij}^s}{\partial s_y} = \sum_{ij} \omega_i \omega_j J \begin{pmatrix} \frac{1}{s_x} N_{i,x} D_{11} N_{j,x} + \frac{-s_x}{s_y^2} N_{i,y} D_{33} N_{j,y} & 0 \\ 0 & \frac{-s_x}{s_y^2} N_{i,y} D_{11} N_{j,y} + \frac{1}{s_x} N_{i,x} D_{33} N_{j,x} \end{pmatrix}, \tag{77}$$

$$\frac{\partial^2 \boldsymbol{k}_{e\,ij}^s}{\partial s_x^2} = \sum_{ij} \omega_i \omega_j J \frac{2 s_y}{s_x^3} \begin{pmatrix} N_{i,x} D_{11} N_{j,x} & 0 \\ 0 & N_{i,x} D_{33} N_{j,x} \end{pmatrix}, \tag{78}$$

$$\frac{\partial^2 \boldsymbol{k}_{e\,ij}^s}{\partial s_y^2} = \sum_{ij} \omega_i \omega_j J \frac{2 s_x}{s_y^3} \begin{pmatrix} N_{i,y} D_{33} N_{j,y} & 0 \\ 0 & N_{i,y} D_{11} N_{j,y} \end{pmatrix}, \tag{79}$$

$$\frac{\partial^2 \boldsymbol{k}_{e\,ij}^s}{\partial s_x s_y} = \sum_{ij} \omega_i \omega_j J \begin{pmatrix} \frac{-1}{s_x^2} N_{i,x} D_{11} N_{j,x} + \frac{-1}{s_y^2} N_{i,y} D_{33} N_{j,y} & 0 \\ 0 & \frac{-1}{s_y^2} N_{i,y} D_{11} N_{j,y} + \frac{-1}{s_x^2} N_{i,x} D_{33} N_{j,x} \end{pmatrix}, \tag{80}$$

$$\frac{\partial^2 \boldsymbol{k}^s_{e\,ij}}{\partial s_y s_x} = \sum_{ij} \omega_i \omega_j J \begin{pmatrix} \frac{-1}{s_x^2} N_{i,x} D_{11} N_{j,x} + \frac{-1}{s_y^2} N_{i,y} D_{33} N_{j,y} & 0 \\ 0 & \frac{-1}{s_y^2} N_{i,y} D_{11} N_{j,y} + \frac{-1}{s_x^2} N_{i,x} D_{33} N_{j,x} \end{pmatrix}. \tag{81}$$

Note that:

$$\frac{\partial^2 \boldsymbol{k}^s_{e\,ij}}{\partial s_x s_y} = \frac{\partial^2 \boldsymbol{k}^s_{e\,ij}}{\partial s_y s_x}. \tag{82}$$

At each integration point, the strain, stress and internal force vectors are calculated. Regarding the strain, an incremental strain vector is calculated then summed to get the total strain vector. For clarity, here is the derivation:

$$\begin{aligned} \boldsymbol{\epsilon}^s_{inc} &= \boldsymbol{B}^s_e \boldsymbol{u}^s_e \qquad \text{with } \boldsymbol{u}^s_e \text{ as the nodal displacement vector} \\ &= \sum_i \begin{pmatrix} N^s_{i,x} & 0 \\ 0 & N^s_{i,y} \\ N^s_{i,y} & N^s_{i,x} \end{pmatrix} \begin{pmatrix} u^s_{x,i} \\ u^s_{y,i} \end{pmatrix} \qquad \text{with } u^s_{j,i} \text{ as the i nodal displacement} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{in the j direction} \\ &= \sum_i \begin{pmatrix} \frac{1}{s_x} N_{i,x} u^s_{x,i} \\ \frac{1}{s_y} N_{i,y} u^s_{y,i} \\ \frac{1}{s_y} N_{i,y} u^s_{x,i} + \frac{1}{s_x} N^s_{i,x} u^s_{y,i} \end{pmatrix}. \end{aligned} \tag{83}$$

Then,

$$\boldsymbol{\epsilon}^s = \sum_{inc} \boldsymbol{\epsilon}^s_{inc}. \tag{84}$$

Concerning the stress:

$$\begin{aligned} \boldsymbol{\sigma}^s &= \boldsymbol{D}_e \boldsymbol{\epsilon}^s (= \boldsymbol{D}_e \boldsymbol{B}^s_e \boldsymbol{u}^s_e) \\ &= \sum_i \begin{pmatrix} \frac{D_{11}}{s_x} N_{i,x} u^s_{x,i} + \frac{D_{12}}{s_y} N_{i,y} u^s_{y,i} \\ \frac{D_{12}}{s_x} N_{i,x} u^s_{x,i} + \frac{D_{11}}{s_y} N_{i,y} u^s_{y,i} \\ \nu(\sigma^s_{11} + \sigma^s_{22}) \\ D_{33}(\frac{1}{s_y} N_{i,y} u^s_{x,i} + \frac{1}{s_x} N^s_{i,x} u^s_{y,i}) \end{pmatrix}. \end{aligned} \tag{85}$$

In the same way $\frac{\partial \boldsymbol{f}_e^s}{\partial s_i}, \frac{\partial^2 \boldsymbol{f}_e^s}{\partial s_i^2}$ and $\frac{\partial^2 \boldsymbol{f}_e^s}{\partial s_i s_j}$ for $(i,j) \in \{(x,y),(y,x)\}$ can be calculated:

$$
\begin{aligned}
\boldsymbol{f}^s_{e\,i} &= -\int_{V_e} \boldsymbol{B}_e^{sT} \boldsymbol{\sigma}^s dV \\
&= -\int_{V_e} \sum_i \begin{pmatrix} N^s_{i,x} & 0 & N^s_{i,y} \\ 0 & N^s_{i,y} & N^s_{i,x} \end{pmatrix} \begin{pmatrix} \sigma^s_{11} \\ \sigma^s_{22} \\ \sigma^s_{12} \end{pmatrix} dV \\
&= -\sum_i \sum_{jk} \omega_j \omega_k J^s \begin{pmatrix} N^s_{i,x}\sigma^s_{11} + N^s_{i,y}\sigma^s_{12} \\ N^s_{i,y}\sigma^s_{22} + N^s_{i,x}\sigma^s_{12} \end{pmatrix} \\
&= -\sum_i \sum_{jk} \omega_j \omega_k J \begin{pmatrix} N_{i,x}(\frac{s_y}{s_x} D_{11} N_{i,x} u^s_{x,i} + D_{12} N_{i,y} u^s_{y,i}) \\ N_{i,y}(D_{12} N_{i,x} u^s_{x,i} + \frac{s_x}{s_y} D_{11} N_{i,y} u^s_{y,i}) \\ \qquad + N_{i,y} D_{33}(\frac{s_x}{s_y} N_{i,y} u^s_{x,i} + N_{i,x} u^s_{y,i}) \\ \qquad + N_{i,x} D_{33}(N_{i,y} u^s_{x,i} + \frac{s_y}{s_x} N_{i,x} u^s_{y,i}) \end{pmatrix}.
\end{aligned} \tag{86}
$$

The first and second derivatives of the internal force vector are expressed as follows:

$$
\frac{\partial \boldsymbol{f}^s_{e\,i}}{\partial s_x} = -\sum_i \sum_{jk} \omega_j \omega_k J \begin{pmatrix} N_{i,x}(\frac{-s_y}{s_x^2} D_{11} N_{i,x} u^s_{x,i}) + N_{i,y} D_{33}(\frac{1}{s_y} N_{i,y} u^s_{x,i}) \\ N_{i,y}(\frac{1}{s_y} D_{11} N_{i,y} u^s_{y,i}) + N_{i,x} D_{33}(\frac{-s_y}{s_x^2} N_{i,x} u^s_{y,i}) \end{pmatrix}, \tag{87}
$$

$$
\frac{\partial \boldsymbol{f}^s_{e\,i}}{\partial s_y} = -\sum_i \sum_{jk} \omega_j \omega_k J \begin{pmatrix} N_{i,x}(\frac{1}{s_x} D_{11} N_{i,x} u^s_{x,i}) + N_{i,y} D_{33}(\frac{-s_x}{s_y^2} N_{i,y} u^s_{x,i}) \\ N_{i,y}(\frac{-s_x}{s_y^2} D_{11} N_{i,y} u^s_{y,i}) + N_{i,x} D_{33}(\frac{1}{s_x} N_{i,x} u^s_{y,i}) \end{pmatrix}, \tag{88}
$$

$$
\frac{\partial^2 \boldsymbol{f}^s_{e\,i}}{\partial s_x^2} = -\sum_i \sum_{jk} \omega_j \omega_k J \frac{2 s_y}{s_x^3} \begin{pmatrix} N_{i,x}(D_{11} N_{i,x} u^s_{x,i}) \\ N_{i,x} D_{33}(N_{i,x} u^s_{y,i}) \end{pmatrix}, \tag{89}
$$

$$
\frac{\partial^2 \boldsymbol{f}^s_{e\,i}}{\partial s_y^2} = -\sum_i \sum_{jk} \omega_j \omega_k J \frac{2 s_x}{s_y^3} \begin{pmatrix} N_{i,y} D_{33}(N_{i,y} u^s_{x,i}) \\ N_{i,y}(D_{11} N_{i,y} u^s_{y,i}) \end{pmatrix}, \tag{90}
$$

$$
\frac{\partial^2 \boldsymbol{f}^s_{e\,i}}{\partial s_x s_y} = -\sum_i \sum_{jk} \omega_j \omega_k J \begin{pmatrix} N_{i,x}(\frac{-1}{s_x^2} D_{11} N_{i,x} u^s_{x,i}) + N_{i,y} D_{33}(\frac{-1}{s_y^2} N_{i,y} u^s_{x,i}) \\ N_{i,y}(\frac{-1}{s_y^2} D_{11} N_{i,y} u^s_{y,i}) + N_{i,x} D_{33}(\frac{-1}{s_x^2} N_{i,x} u^s_{y,i}) \end{pmatrix}, \tag{91}
$$

$$
\frac{\partial^2 \boldsymbol{f}^s_{e\,i}}{\partial s_y s_x} = -\sum_i \sum_{jk} \omega_j \omega_k J \begin{pmatrix} N_{i,x}(\frac{-1}{s_x^2} D_{11} N_{i,x} u^s_{x,i}) + N_{i,y} D_{33}(\frac{-1}{s_y^2} N_{i,y} u^s_{x,i}) \\ N_{i,y}(\frac{-1}{s_y^2} D_{11} N_{i,y} u^s_{y,i}) + N_{i,x} D_{33}(\frac{-1}{s_x^2} N_{i,x} u^s_{y,i}) \end{pmatrix}. \tag{92}
$$

Note that:

$$\frac{\partial^2 \boldsymbol{f}^s_{ei}}{\partial s_x s_y} = \frac{\partial^2 \boldsymbol{f}^s_{ei}}{\partial s_y s_x}. \tag{93}$$

D.2 Singularity of the element stiffness matrix

At increment 0, iteration 0, the simulation runs with no prescription which allows to get the element stiffness matrix. The generic user element, the element 11 has the same stiffness, see Figure 44:

```
 0.41666656953125005E+06  0.22321428571428565E+06 -0.32738085524553567E+06
 0.22321428571428565E+06  0.41666656953124999E+06 -0.13392857142857139E+06
-0.32738085524553567E+06 -0.13392857142857139E+06  0.41666656953125005E+06
 0.13392857142857139E+06  0.11904771618303568E+06 -0.22321428571428565E+06
-0.20833343046874998E+06 -0.22321428571428568E+06  0.11904771618303568E+06
-0.22321428571428565E+06 -0.20833343046874998E+06  0.13392857142857142E+06
 0.11904771618303568E+06  0.13392857142857139E+06 -0.20833343046874998E+06
-0.13392857142857142E+06 -0.32738085524553567E+06  0.22321428571428565E+06

 0.13392857142857139E+06 -0.20833343046874998E+06 -0.22321428571428565E+06
 0.11904771618303568E+06 -0.22321428571428568E+06 -0.20833343046874998E+06
-0.22321428571428565E+06  0.11904771618303568E+06  0.13392857142857142E+06
 0.41666656953124999E+06 -0.13392857142857139E+06 -0.32738085524553567E+06
-0.13392857142857139E+06  0.41666656953124999E+06  0.22321428571428568E+06
-0.32738085524553567E+06  0.22321428571428568E+06  0.41666656953124999E+06
 0.22321428571428568E+06 -0.32738085524553567E+06 -0.13392857142857139E+06
-0.20833343046874998E+06  0.13392857142857139E+06  0.11904771618303568E+06

                                                    0.11904771618303568E+06 -0.13392857142857142E+06
                                                    0.13392857142857139E+06 -0.32738085524553567E+06
                                                   -0.20833343046874998E+06  0.22321428571428565E+06
                                                    0.22321428571428568E+06 -0.20833343046874998E+06
                                                   -0.32738085524553567E+06  0.13392857142857139E+06
                                                   -0.13392857142857139E+06  0.11904771618303568E+06
                                                    0.41666656953124999E+06 -0.22321428571428568E+06
                                                   -0.22321428571428568E+06  0.41666656953125005E+06
```

Figure 44: Element stiffness matrix.

Recall that any element stiffness matrix is a square matrix and has to be singular because of body motions. The body motions in a planar problem are the displacements in two directions plus one rotation. Hence, the determinant of the stiffness matrix is null, and its rank is equal to its dimension minus three (of the body motion). At least three boundary

conditions in displacement need to be prescribed to prevent those body motions and make the system regular and solvable, which will give an unique solution.

NB: Computationally, the significant digit is important here. If it is not sufficient, the singularity of the matrix cannot be proved even if it is. If 5 significant digits are used as the usual print does (fortran format: e13.5), the determinant of the matrix is a very large number, and its rank is equal to its dimension. 17 significant digits (e25.17) are needed to get the determinant null and the rank at 5 as Figure 44 shows.

This singularity of the element stiffness matrix is applied to its derivatives so that derivatives of the prescribed displacements with respect to the scaling factors are necessary in order to make the whole system 43 regular.

Appendix E

THE SIZE SENSITIVITY BASED PERTURBATION METHOD USER-DEFINED ELEMENT 11 SUBROUTINES CODE

E.1 Presentation

From the equations of the section 3.1, and the pattern of user defined element subroutine available in Marc/Mentat, the so-called Size sensitivity based Perturbation method user-defined element 11 is implemented.

Recall that the user defined element subroutine [MSC.Software, 2011], called USELEM coded in Fortran 77 [Fortran, 2011], allows the user in particular to calculate his own finite element stiffness and can be used as interface with other numerical techniques which are Size sensitivity computation and Perturbation method in our case. This user subroutine is called a multiple number of times. The calls and the user's requirements are defined by iflag as follows:

- iflag=1 calculation returns the equivalent nodal loads (F) given distributed surface or body loads which is not needed in our case,
- iflag=2 calculation returns the element tangent stiffness matrix (K). For an elastic analysis, this is the usual stiffness,
- iflag=3 returns the mass matrix (M) for a dynamic analysis which is not needed in our case,
- iflag=4 calculates the incremental strains (DE), generalized stresses (GSIGS) and the internal force (R),
- iflag=5 outputs Size sensitivity and Perturbation method results in our case.

Here is a brief description of the coding step showing how the implementation has been done:

- Variable types statements, and inputs and outputs signification,
- Declaration of variables' dimension,
- Including common blocks which allows getting information with the entire program such as material and geometrical data, and calculation advancements such as the number of cycles and iterations,
- Declaration of internal variables (type and dimension),
- Checking inputs such as material, geometric data, and calculation advancements,
- Iflag independent and iflag dependent initialization according to the following calculation process,
- Calculation which consists of:
  - iflag independent calculation where the shape functions, their derivatives, the B matrix, the Jacobian and the material are computed,
  - iflag dependent calculation where
    - iflag 2 stiffness matrix and internal force vector are calculated,
    - iflag 4 incremental and total strains, stresses and internal force vectors are assessed.
    - iflag 5 output which is used for calculating the first two moments of the displacement field for the Perturbation method.

The implementation of the Size sensitivity based Perturbation method allows to perform:

- the generic linear, fournode, isoparametric, plane strain, quadrilateral Marc/Mentat element 11,
- the Size effect analysis on either x or y direction,
- the Perturbation analysis of displacement field.

It is implemented in such modular design that it can be incorporated in any model involving the element 11 with slight modifications and runs on Marc/Mentat 2008 r1 (2008r1i4) and 2010.1.0 (2010r1i8) at least.

It consists of:

- M data module containing dynamic arrays storing results for the postprocessing,
- USDATA user subroutine to read the scale factor input of the model,
- USELEM userdefined element subroutine performing the size sensitivity based perturbation method on the base of the userdefined element 11 (linear, fournode, isoparametric, plane strain, quadrilateral Marc/Mentat element 11),
- UFCONN user subroutine of MSc. Marc to add/modify element connectivity,
- UPSTNO user subroutine to postprocess nodal displacements, derivatives, means, standard deviations and covariance,
- cb_size_pm_uselem_11_v2011_03 common block storing the scale factors read from the USDATA subroutine.

NB: All write statements are in comment c, uncomment them if prints are needed.

E.2 Utilization

1. Put the following 3 files in the same directory:
   - The model file: model1.mud,
   - The subroutine file of the Size sensitivity based Perturbation method implementation: size_pm_uselem_11_v2011_03.f,
   - The common block file: cb_size_pm_uselem_11_v2011_03.com.
2. Make the FE model as usual with element 11 without the prescribed force boundary conditions. Indeed, the present deals with point load boundary condition. The reader is referred to the Marc/Mentat manual Vol. C (see [MSC.Software, 2011]) for parameterizing other types of loadings such as surface force, thermal loading, etc., and for using the trick described in 6., point load statement.

3. In the model file, specify the utilization of the size_pm_uselem_11_v2011_03.f in the run job, user subroutine file option.

4. In the run job, solver option, check in addition the non-symmetric solution.

   For the following, red statements are the command which has to be written in the model file and ' ' has to be replaced by value (or name).

5. Create a set called all_elements containing all elements 11 of the model.

6. In the additional input text file, parameter section option, specify

   - ELEMENTS,11
   - USER,11,14,4,4,4,2,4,3,1,5,0,,31
   - ALIAS,1,11,11

   in order to run correctly the USELEM such that it sets 11 as the element number, 14 degrees of freedom per node, 4 nodes per element, 2 coordinates per node, 4 integration points, plain strain mechanical model as element class and 4node quadrilateral (linear) topology class.

7. In the additional input text file, model definition option, specify

   - UFCONN
   - all_elements

   which calls the UFCONN subroutine for all elements 11.

   - USDATA,5
   - 'μ_{s_x}', 'μ_{s_y}', 'σ_{s_x}', 'σ_{s_y}', 'ρ'

   where μ_{s_x} is the mean of the scaling factor in x direction s_x, μ_{s_y} is the mean of s_y, σ_{s_x} is the standard deviation of s_x, σ_{s_y} is the standard deviation of s_y, rho is the correlation function between s_x and s_y ($1 \leq \rho \leq 1$).

   If the user wants to perform the Size effect analysis, replace μ_{s_x} by s_x value, μ_{s_y} by

s_y , and leave the other input parameters null.

this calls the USDATA subroutine to read the input.

- POINT LOAD
- 1,0,0,0,0,0,'name of the prescribed point load x'
- 'f_x','f_y',0,0,0,0,0,0,C
- 0,0,0,0,0,0
- 0,0,0,0,0,0,0,0,C
- 0,0,0,0,0,0
- 2
- 'list of applied nodes'

where f_x and f_y are the component of the force in x and y direction respectively.

which defined the point load type traction boundary condition.

This has to be done for each type of point load.

Any prescribed load point has to be defined in this way and cannot be done through the Mentat interface, because the number of degree of freedom is 14. Indeed, the C statement allows to define the degree of freedom beyond the 6 usual ones available in the Mentat interface.

Other types of loading such as surface force, thermal loading can be parameterized, referring to the Marc/Mentat manual Vol. C (see [MSC.Software, 2011]) by using this method if necessary.

- FIXED DISP
-
- 1,0,0,0,1,0, 'prescribed displacement name x'dx
- 0,0,0,0,0,0
- 0,0,0,0,0,0
- 3,5,7,9,11,13

- 2
- 'list of applied nodes'

which prescribed the s_x derivatives of the displacement for nodes with fixed (both null and nonnull) prescribed displacement boundary condition.
Then add immediately,

- 1,0,0,0,1,0, 'prescribed displacement name x'dy
- 0,0,0,0,0,0
- 0,0,0,0,0,0
- 4,6,8,10,12,14
- 2
- 'list of applied nodes'

which prescribed the s_y derivatives of the displacement for nodes with fixed (both null and nonnull) prescribed displacement boundary condition.

- LOADCASE
- 'total number of boundary condition'
- 'name of the prescribed point load 1'
- 'name of the prescribed point load 2'
- ...
- 'prescribed displacement name 1'
- 'prescribed displacement name 1'dx
- 'prescribed displacement name 1'dy
- 'prescribed displacement name 2'
- 'prescribed displacement name 2'dx
- 'prescribed displacement name 2'dy
- ...

which sets up the loadcase with all the boundary conditions.

Attention: Since the number of prescriptions are increased with the prescriptions of derivatives of the displacement for nodes with fixed (both null and nonnull) prescribed displacement boundary condition, all of them have to be taken account into the loadcase.

8. In the run job results option, select the user nodal quantity 1 to 19 (User Sub UPSTNO).

Now, the model can be run!

E.3 Results and Postprocessing

In the postprocessing, the user nodal quantity refers as follows,

1. is the displacement in x direction,
2. is the displacement in y direction,
3. is the first order derivative of the x displacement wrt s_x,
4. is the first order derivative of the y displacement wrt s_x,
5. is the first order derivative of the x displacement wrt s_y,
6. is the first order derivative of the y displacement wrt s_y,
7. is the second derivative of the x displacement wrt s_x^2,
8. is the second derivative of the y displacement wrt s_x^2,
9. is the second derivative of the x displacement wrt s_y^2 ,
10. is the second derivative of the y displacement wrt s_y^2,
11. is the second derivative of the x displacement wrt $s_x s_y$,
12. is the second derivative of the y displacement wrt $s_x s_y$,
13. is the second derivative of the x displacement wrt $s_y s_x$, which is equal to 11,

14. is the second derivative of the y displacement wrt $s_y s_x$, which is equal to 12,

15. is the mean of the x displacement,

16. is the mean of the y displacement,

17. is the standard deviation of the x displacement,

18. is the standard deviation of the y displacement,

19. is the covariance of the x and y displacement.

To sum up the Size sensitivity based Perturbation method implementation:

- requires as input, the mean, standard deviation and the coefficient of correlation of the scaling in x and y direction,
- gives as output, the displacement, strain, stress fields (plus other outputs available in the post-processing of Marc/Mentat with linear, fournode, isoparametric, plane strain, quadrilateral element), and the first two moments of the displacements fields at each node.

E.4 Remarks

The limitation which can be seen is that the model with the scaling is not drawn on the Mentat interface, since the Size sensitivity computation is within the element calculation. In addition, the method can be implemented in other elements such as quadratic quadrilateral element and so on.
Last, it is possible to perform a multi-increment calculation such as constant step.

E.5 Codes

Here is the size_pm_uselem_11_v2011_03.f code:

```
c-----------------------------------------------------------------------c
c Size sensitivity based perturbation method user-defined element 11   c
c**********************************************************************c
c Running on Marc/Mentat 2008 r1 (2008r1-i4) and 2010.1.0 (2010r1i8)   c
c at least.                                                            c
c Set of module, common block and user subroutines which allows to     c
c perform:                                                             c
c - the generic linear, four-node, isoparametric, plane strain,        c
c   arbitrary quadrilateral Marc/Mentat element 11,                    c
c - the size sensitivity computation on x and y direction (non-uniform c
c   scaling),                                                           c
c - the perturbation method of stochastic finite element methods based c
c   on the above size sensitivity computation.                         c
c                                                                      c
c Contents:                                                            c
c - M_data module containing dynamic arrays storing results for        c
c   the post-processing,                                               c
c - USDATA user subroutine to read the scale factor input of the model,c
c - USELEM user-defined element subroutine performing the size         c
c   sensitivity based perturbation method user-defined element 11      c
c   (linear, four-node, isoparametric, plane strain, arbitrary         c
c   quadrilateral Marc/Mentat element 11),                             c
c - UFCONN user subroutine of MSC-Marc to add/modify element           c
c   connectivity,                                                      c
c - UPSTNO user subroutine to post-process nodal displacements, and    c
c   its derivatives, mean, variance and covariance with respect to the c
c   scale factors,                                                     c
c - cb_size_pm_uselem_11_v2011_03 common block storing the scale       c
c   factors read from the USDATA subroutine.                           c
c NB: Write statements are in comment c, uncomment them if prints are  c
c     needed.                                                          c
c**********************************************************************c
c-----------------------------------------------------------------------c
c M_data module for the post-processing                                c
c**********************************************************************c
c Contents:                                                            c
c - elecon(l,m): general node number of local node l of element m,     c
c - result(ndeg,numnp): array where displacement field, its            c
c   derivatives wrt the scaling uc(ndeg,nnode), and its first two      c
c   moment field meanu(2,nnode), dev(2,nnode), and cov(nnode) are      c
c   stored.                                                            c
c**********************************************************************c
      module M_data
C
      implicit none
C
c     Dynamic arrays
      integer, dimension(:,:), allocatable :: elecon
* elecon(l,m): Node number of local node l of element m
      real*8, dimension(:,:), allocatable :: result
* result(i,j): i->ndeg, j->numnp
C
      end module M_data
c-----------------------------------------------------------------------c
c USDATA user subroutine to read scale factor input of the model       c
c**********************************************************************c
c Running with the additional input file text - model definition       c
c section option command:                                              c
c USDATA,5                                                             c
c sx or mean of sx, sy or mean of sy, dev of sx, dev of sy, cov of sx  c
c and sy                                                               c
```

```
c NB: Detailed description in the above Content,                      c
c     In general, dev is not more than 10% of the mean.              c
c**********************************************************************
c Content:                                                            c
c - scaling(5): vector of 5 components storing scale factors sx and   c
c   sy and/or the mean, the variance dev, the correlation coefficient c
c   cov of the element length on x and y direction via usdata         c
c   subroutine and additional input file text - model definition      c
c   section option command.                                          c
c**********************************************************************
c Input and output variables explanation:
c kin  = input unit
c kou  = output unit
c ic   = 1 : pre-reader
c      = 2 : real reader
c**********************************************************************
      subroutine usdata(kin,kou,ic)
C
      include '../common/implicit'
      include '../common/concom'
C
      include 'cb_size_pm_uselem_11_v2011_03.com'
C
c ** Start of generated type statements **
      integer ic, kin, kou
c ** End of generated type statements **
C     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      read(KIN,*)scaling
c      write(0,*)'USDATA-------------------------------------------------'
c      write(0,'(''mu_x ='',e13.5,2x,''mu_y ='',e13.5,2x,''dev_x ='',
c     1e13.5,2x,''dev_y ='',e13.5,2x,''rho ='',e13.5)')
c     2(scaling(i),i=1,5)
C
      return
      end
c-----------------------------------------------------------------------c
c USELEM user-defined element subroutine performing the size          c
c sensitivity based perturbation method user-defined element 11       c
c (linear, four-node, isoparametric, plane strain, arbitrary          c
c quadrilateral Marc/Mentat element 11)                               c
c**********************************************************************
c Input and output variables explanation:                             c
c     m             user element number
c     xk            stiffness matrix
c     xm            mass matrix
c     nnode         number of nodes per element
c     ndeg          maximum number of degrees of freedom per node
c     f             externally applied equivalent nodal loads
c     r             internal forces
c     jtype         user element type (negative)
c     dispt         total nodal displacements of this element
c     disp          incremental nodal displacements of this element
c     ndi           number of direct components of stress
c     nshear        number of shear components of stress
c here named ipasss in order to avoid the confusion with the ipass
c belonging to the common block concom
c     ipass         flag to indicate which pass for coupled analysis
c     ipass=1       stress analysis pass
c     ipass=2       heat transfer pass
c     ipass=3       fluid pass
c     ipass=4       joule heating pass
c     ipass=5       pore pressure pass
c     ipass=6       electrostatic pass
c     ipass=7       magnetostatic pass
c     ipass=8       electromagnetic pass
c     nstats        number of state variables
```

```
c     ngenel        number of generalized strains
c     intel         number of integration points
c     coord         nodal original coordinates
c     ncrd          maximum number of coordinates per node
c
c     iflag         indicates what is to be returned by the user
c     iflag=1       called by opress
c                   user returns f
c     iflag=2       called by oasemb
c                   user returns xk,r
c     iflag=3       called by oasmas
c                   user returns xm
c     iflag=4       called by ogetst
c                   user returns r,gsigs,de,etota,sigxx
c     iflag=5       called by scimp
c                   user prints results
C
c     idss          size of element stiffness matrix (ndeg*nnode)
c     t             state variables
c     dt            increment of state variables
c     etota         total strain
c     gsigs         generalized stresses
c     de            increment of strain
c     geom          geometric parameters
c     jgeom         table ids for  geometric parameter
c     sigxx         stresses (equal to gsigs for continuums)
c     nstrmu        number of stresses per integration point
*****************************************************************
      subroutine uselem(m,xk,xm,nnode,ndegg,f,r,
     * jtype,dispt,disp,ndi,nshear,ipasss,nstats,ngenel,
     * intel,coord,ncrdd,iflag,idsss,t,dt,etota,gsigs,de,
     * geom,jgeom,sigxx,nstrmu)
C
      use M_data
C
      include '../common/implicit'
C
c include cb_size_pm_uselem_11_v2011_03.com common block for getting
c the scale factors
      include 'cb_size_pm_uselem_11_v2011_03.com'
C
c include the commonblock matdat for material data
c for young modulus et(1) and poisson's ratio xu(1)
      include 'matdat'
C
c include the common block concom  and dimen for obtaining
c the increment number inc, iteration number ncycle and number of
c nodal points numnp
      include 'concom'
      include '../common/dimen'
C
c ** Start of generated type statements **
      real*8 coord, de, disp, dispt, dt, etota, f, geom, gsigs
      integer idsss, iflag, intel, ipasss, jgeom, jtype, m, ncrdd, ndegg
      integer ndi, ngenel, nnode, nshear, nstats, nstrmu
      real*8 r, sigxx, t, xk, xm
c ** End of generated type statements **
C
      dimension xk(idss,idss),xm(idss,idss),dispt(ndeg,*),disp(ndeg,*)
      dimension t(nstats,*),dt(nstats,*),coord(ncrd,*)
      dimension etota(ngenel,*),gsigs(ngenel,*),de(ngenel,*)
      dimension f(ndeg,*),r(ndeg,*),sigxx(nstrmu,*),geom(*),jgeom(*)
C
c internal variables
      integer allsta
      real*8 shp(nnode),dshp(2,nnode),xii(nnode),etai(nnode)
```

```
      real*8 uc(ndeg,nnode)
      real*8 xj(nnode,nnode),xjs(nnode,nnode),dj,djs
      real*8 b(3,2,nnode),dshpg(2,nnode)
      real*8 bs(3,2,nnode), dshpgs(2,nnode)
      real*8 d(3)
      real*8 wg(intel)
      real*8 gxi(intel),geta(intel)
201   real*8 meanu(2,nnode),dev(2,nnode),cov(nnode)
      if (.not.allocated(result)) then
        allocate(result(ndeg+5,numnp),stat=allsta)
      endif
C     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c Checking inputs                                                       c
C     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c       write(0,*) 'USELEM Checking inputs----------------------------'
C
c       write(0,*) 'Program*****************************************'
211   c       write(0,'(''inc ='',i5,2x,''ncycle ='',i5,2x,''ipass ='',i5,2x,
c      *''iflag ='',i5,2x,''m ='',i5)') inc,ncycle,ipasss,iflag,m
C
c       write(0,*) 'Parameters**************************************'
c       write(0,'(''nnode ='',i5,2x,''ndeg ='',i5,2x,
c      1''jtype ='',i5,2x,''ndi ='',i5,2x,''nshear ='',i5,2x,
c      2''nstats ='',i5,2x,''ngenel ='',i5,2x,''intel ='',i2,2x,
c      3''ncrd ='',i5,2x,''idss ='',i5,2x,''nstrmu ='',i5,2x,''numnp ='',
c      4i5)')nnode,ndeg,jtype,ndi,nshear,nstats,ngenel,intel,ncrd,idss,
c      5nstrmu,numnp
221   C
c       write(0,'(''E ='',e13.5,2x,''nu ='',f2.1,2x,''thickness ='',
c      *f8.2)') et(1),xu(1),geom(1)
C
c       write(0,*) 'Vectors*****************************************'
c       write(0,*) 'coord'
c       do inode=1,nnode
c       write(0,'(''node ='',i5,2x,''coord ='', 2e13.5)') inode,
c      *(coord(incrd,inode),incrd=1,ncrd)
c       enddo
231   C
c       write(0,*) 'dispt'
c       do inode=1,nnode
c       write(0,'(''node ='',i5,2x,''dispt ='', 14e13.5)') inode,
c      *(dispt(indeg,inode),indeg=1,ndeg)
c       enddo
C
c       write(0,*) 'disp'
c       do inode=1,nnode
c       write(0,'(''node ='',i5,2x,''disp ='', 14e13.5)') inode,
241   c      *(disp(indeg,inode),indeg=1,ndeg)
c       enddo
C     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c Initialization                                                        c
C     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c       write(0,*) 'Initialization------------------------------------'
* note: this initialization is for each element,
* so for all intpoi within the element
c ***************** iflag independent initialization ******************
c       write(0,*) 'iflag independent initialization**********'
251   C
      xii(1)=-1d0
      xii(2)=1d0
      xii(3)=1d0
      xii(nnode)=-1d0
      etai(1)=-1d0
```

```
      etai(2)=-1d0
      etai(3)=1d0
      etai(nnode)=1d0
C
      do i = 1,nnode
         shp(i) =  0d0
         dshp(1,i) = 0d0
         dshp(2,i) = 0d0
      enddo
C
      substitution = 0d0
C
      do i = 1,nnode
         dshpg(1,i) = 0d0
         dshpg(2,i) = 0d0
         dshpgs(1,i) = 0d0
         dshpgs(2,i) = 0d0
      enddo
C
      do i=1,ndeg
         do j=1,nnode
            uc(i,j) = 0d0
         enddo
      enddo
C
      dj=0d0
C
      do 100 i = 1,3
      do 100 j = 1,2
      do 100 k = 1,nnode
      b(i,j,k) = 0d0
      bs(i,j,k) = 0d0
 100  continue
C
      do i=1,3
      d(i)=0d0
      enddo
C
      wg(1) = 1d0
      wg(2) = 1d0
      wg(3) = 1d0
      wg(4) = 1d0
c ******************** iflag dependent initialization *****************
c      write(0,*) 'iflag dependent initialization***********'
C
      if (iflag.eq.2) then
      do i = 1,idss
        do j = 1,idss
           xk(i,j) = 0d0
        enddo
      enddo
C
      do i = 1,ndeg
        do j = 1,nnode
           r(i,j) = 0d0
        enddo
      enddo
C
      elseif (iflag.eq.4) then
      do i = 1,ndeg
        do j = 1,nnode
           r(i,j) = 0d0
        enddo
      enddo
C
```

```
      do i=1,ngenel
        do j=1,intel
          etota(i,j)=0d0
        enddo
      enddo
C
      do i=1,ngenel
        do j=1,intel
          gsigs(i,j)=0d0
        enddo
      enddo
C
      do i=1,ngenel
        do j=1,intel
          de(i,j)=0d0
        enddo
      enddo
C
      do i=1,nstrmu
        do j=1,intel
          sigxx(i,j)=0d0
        enddo
      enddo
C
      elseif (iflag.eq.5) then
      do i=1,nnode
        meanu(1,i)=0d0
        meanu(2,i)=0d0
        dev(1,i)=0d0
        dev(2,i)=0d0
        cov(i)=0d0
      enddo
C
      endif
C     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c Calculations                                                         c
C     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c      write(0,*) 'Calculation-------------------------------------------'
c ******************** iflag-independent calculations *****************
c      write(0,*) 'iflag independent calculations***********'
C
      gxi(1)=-0.57735d0
      gxi(2)=0.57735d0
      gxi(3)=-0.57735d0
      gxi(4)=0.57735d0
      geta(1)=-0.57735d0
      geta(2)=-0.57735d0
      geta(3)=0.57735d0
      geta(4)=0.57735d0
C
c     start integration point loop
      do intpoi=1,intel
c      write(0,'(''*****************intpoint ='',I2)') intpoi
C
c ******************* code for shape function
C
      xi=gxi(intpoi)
      eta=geta(intpoi)
C
      do i = 1,nnode
        shp(i) = 0.25d0*(1d0+xi*xii(i))*(1d0+eta*etai(i))
        dshp(1,i) = 0.25d0*xii(i)*(1d0+eta*etai(i))
        dshp(2,i) = 0.25d0*etai(i)*(1d0+xi*xii(i))
      enddo
```

```
c       write(0,*) 'Shape function ='
c       write(0,'(e13.5,2x,e13.5,2x,e13.5,2x,e13.5)')
c      *(shp(inode),inode=1,nnode)
c       write(0,*) 'Derivative iso shape function ='
c       do i=1,2
391 c       write(0,'(e13.5,2x,e13.5,2x,e13.5,2x,e13.5)') dshp(i,1),dshp(i,2),
c      1dshp(i,3),dshp(i,nnode)
c       enddo
c ********************** code for jacobian
c ******* displacement uc
        if (ncycle.eq.0) then
           do i=1,ndeg
              do j=1,nnode
                 uc(i,j)=dispt(i,j)
              enddo
401        enddo
        elseif (ncycle.ne.0.AND.iflag.ne.5) then
           do i=1,ndeg
              do j=1,nnode
                 uc(i,j)=dispt(i,j)+disp(i,j)
              enddo
           enddo
        elseif (ncycle.ne.0.AND.iflag.eq.5) then
           do i=1,ndeg
              do j=1,nnode
411              uc(i,j)=dispt(i,j)
              enddo
           enddo
        endif
c       write(0,*) 'Current displacement uc'
c       do inode=1,nnode
c       write(0,'(''node ='',i5,2x,''uc ='', 14e13.5)') inode,
c      *(uc(indeg,inode),indeg=1,ndeg)
c       enddo
c******* Scaled and non scaled jacobian
421     do i = 1,2
          do j = 1,2
            xj(i,j) = 0d0
            xjs(i,j) = 0d0
            do k = 1,nnode
              xj(i,j) = xj(i,j)+(coord(i,k))*dshp(j,k)
              xjs(i,j) = xjs(i,j)+(scaling(j)*coord(i,k))*dshp(j,k)
            enddo
          enddo
        enddo
431     dj = xj(1,1)*xj(2,2) - xj(1,2)*xj(2,1)
        djs = xjs(1,1)*xjs(2,2) - xjs(1,2)*xjs(2,1)
c       write(0,*) 'Jacobian array ='
c       do i=1,2
c       write(0,'(e13.5,2x,e13.5)') xj(i,1),xj(i,2)
c       enddo
c       write(0,'(''The jacobiam ='',e13.5)') dj
c       write(0,*) 'Scaled Jacobian array ='
c       do i=1,2
c       write(0,'(e13.5,2x,e13.5)') xjs(i,1),xjs(i,2)
441 c       enddo
c       write(0,'(''The scaled jacobiam ='',e13.5)') djs
c ******* Scaled B matrix
        do 110 i = 1,nnode
          substitution = (xj(2,2)*dshp(1,i) - xj(2,1)*dshp(2,i))/dj
          dshpg(2,i) = (-xj(1,2)*dshp(1,i) + xj(1,1)*dshp(2,i))/dj
          b(2,2,i)= dshpg(2,i)
          b(3,1,i)= dshpg(2,i)
          dshpg(1,i) = substitution
```

```
            b(1,1,i)= dshpg(1,i)
            b(3,2,i)= dshpg(1,i)
            substitutions = (xjs(2,2)*dshp(1,i) - xjs(2,1)*dshp(2,i))/djs
            dshpgs(2,i) = (-xjs(1,2)*dshp(1,i) + xjs(1,1)*dshp(2,i))/djs
            bs(2,2,i)= dshpgs(2,i)
            bs(3,1,i)= dshpgs(2,i)
            dshpgs(1,i) = substitutions
            bs(1,1,i)= dshpgs(1,i)
            bs(3,2,i)= dshpgs(1,i)
     110   continue
c       write(0,*) 'Scaled B matrix ='
c       do j=1,nnode
c         do i=1,3
c           write(0,'(''node ='',i2,2x,e13.5,2x,e13.5)')
c      1j,bs(i,1,j),bs(i,2,j)
c         enddo
c       enddo
c ******* code for determining material moduli
c D(1), D(2) and D(3) in plane strain condition from E=et(1), Nu=xu(1)**
      d(1)=et(1)*(1d0-xu(1))/((1d0+xu(1))*(1d0-2d0*xu(1)))
      d(2)=xu(1)*d(1)/(1d0-xu(1))
      d(3)=(d(1)-d(2))/2d0
c       write(0,*) 'd(1) =', d(1)
c       write(0,*) 'd(2) =', d(2)
c       write(0,*) 'd(3) =', d(3)
c ******************** iflag-dependent calculations ********************
c       write(0,*) 'iflag dependent calculations************'
C
          if (iflag.eq.1.and.intpoi.eq.1) then
C
              jtype=-jtype
C
* use standard marc implementation for retrieving load factor f from
* opress
C
          elseif (iflag.eq.2) then
C
c *************** code for scaled stiffness matrix iflag 2
      j1=1
        do 120 j=1,nnode
        i1=1
          do 130 i=1,j
* size-sensitititied matrix
            xk(i1,j1) = xk(i1,j1)+dshpgs(1,i)*d(1)*dshpgs(1,j)*djs
     1*wg(intpoi)*geom(1)+dshpgs(2,i)*d(3)*dshpgs(2,j)*djs*wg(intpoi)
     2*geom(1)
            xk(i1,j1+1) = xk(i1,j1+1)+dshpgs(1,i)*d(2)*dshpgs(2,j)*djs
     1*wg(intpoi)*geom(1)+dshpgs(2,i)*d(3)*dshpgs(1,j)*djs*wg(intpoi)
     2*geom(1)
            xk(i1+1,j1) = xk(i1+1,j1)+dshpgs(1,i)*d(3)*dshpgs(2,j)*djs
     1*wg(intpoi)*geom(1)+dshpgs(2,i)*d(2)*dshpgs(1,j)*djs*wg(intpoi)
     2*geom(1)
            xk(i1+1,j1+1) = xk(i1+1,j1+1)+dshpgs(2,i)*d(1)*dshpgs(2,j)
     1*djs*wg(intpoi)*geom(1)+dshpgs(1,i)*d(3)*dshpgs(1,j)*djs
     2*wg(intpoi)*geom(1)
* 1st order x derivative
            xk(i1+2,j1) = xk(i1+2,j1)+(-scaling(2)/(scaling(1)**2d0))
     1*dshpg(1,i)*d(1)*dshpg(1,j)*dj*wg(intpoi)*geom(1)+(1d0/scaling(2))
     2*dshpg(2,i)*d(3)*dshpg(2,j)*dj*wg(intpoi)*geom(1)
            xk(i1+2,j1+1) = xk(i1+2,j1+1)
            xk(i1+3,j1) = xk(i1+3,j1)
            xk(i1+3,j1+1) = xk(i1+3,j1+1)+(1d0/scaling(2))*dshpg(2,i)
     1*d(1)*dshpg(2,j)*dj*wg(intpoi)*geom(1)
```

```
     2+(-scaling(2)/(scaling(1)**2d0))*dshpg(1,i)*d(3)*dshpg(1,j)*dj
     3*wg(intpoi)*geom(1)
             xk(i1+2,j1+2) = xk(i1,j1)
             xk(i1+2,j1+3) = xk(i1,j1+1)
             xk(i1+3,j1+2) = xk(i1+1,j1)
             xk(i1+3,j1+3) = xk(i1+1,j1+1)
   * 1st order y derivative
             xk(i1+4,j1) = xk(i1+4,j1)+(1d0/scaling(1))*dshpg(1,i)*d(1)
521     1*dshpg(1,j)*dj*wg(intpoi)*geom(1)+(-scaling(1)/(scaling(2)**2d0))
     2*dshpg(2,i)*d(3)*dshpg(2,j)*dj*wg(intpoi)*geom(1)
             xk(i1+4,j1+1) = xk(i1+4,j1+1)
             xk(i1+5,j1) = xk(i1+5,j1)
             xk(i1+5,j1+1) = xk(i1+5,j1+1)
     1+(-scaling(1)/(scaling(2)**2d0))*dshpg(2,i)*d(1)*dshpg(2,j)*dj
     2*wg(intpoi)*geom(1)+(1d0/scaling(1))*dshpg(1,i)*d(3)*dshpg(1,j)
     3*dj*wg(intpoi)*geom(1)
             xk(i1+4,j1+4) = xk(i1,j1)
             xk(i1+4,j1+5) = xk(i1,j1+1)
531          xk(i1+5,j1+4) = xk(i1+1,j1)
             xk(i1+5,j1+5) = xk(i1+1,j1+1)
   * 2nd order x derivative
             xk(i1+6,j1) = xk(i1+6,j1)+(2d0*scaling(2)/(scaling(1)**3d0))
     1*dshpg(1,i)*d(1)*dshpg(1,j)*dj*wg(intpoi)*geom(1)
             xk(i1+6,j1+1) = xk(i1+6,j1+1)
             xk(i1+7,j1) = xk(i1+7,j1)
             xk(i1+7,j1+1) = xk(i1+7,j1+1)
     1+(2d0*scaling(2)/(scaling(1)**3d0))*dshpg(1,i)*d(3)*dshpg(1,j)*dj
     2*wg(intpoi)*geom(1)
541          xk(i1+6,j1+2) = xk(i1+6,j1+2)+2
     1*((-scaling(2)/(scaling(1)**2d0))*dshpg(1,i)*d(1)*dshpg(1,j)*dj
     2*wg(intpoi)*geom(1)+(1d0/scaling(2))*dshpg(2,i)*d(3)*dshpg(2,j)*dj
     3*wg(intpoi)*geom(1))
             xk(i1+6,j1+3) = xk(i1+6,j1+3)
             xk(i1+7,j1+2) = xk(i1+7,j1+2)
             xk(i1+7,j1+3) = xk(i1+7,j1+3)
     1+2d0*((1d0/scaling(2))*dshpg(2,i)*d(1)*dshpg(2,j)*dj*wg(intpoi)
     2*geom(1)+(-scaling(2)/(scaling(1)**2d0))*dshpg(1,i)*d(3)
     3*dshpg(1,j)*dj*wg(intpoi)*geom(1))
551          xk(i1+6,j1+6) = xk(i1,j1)
             xk(i1+6,j1+7) = xk(i1,j1+1)
             xk(i1+7,j1+6) = xk(i1+1,j1)
             xk(i1+7,j1+7) = xk(i1+1,j1+1)
   * 2nd order y derivative
             xk(i1+8,j1) = xk(i1+8,j1)+(2d0*scaling(1)/(scaling(2)**3d0))
     1*dshpg(2,i)*d(3)*dshpg(2,j)*dj*wg(intpoi)*geom(1)
             xk(i1+8,j1+1) = xk(i1+8,j1+1)
             xk(i1+9,j1) = xk(i1+9,j1)
             xk(i1+9,j1+1) = xk(i1+9,j1+1)
561     1+(2d0*scaling(1)/(scaling(2)**3d0))*dshpg(2,i)*d(1)*dshpg(2,j)
     2*dj*wg(intpoi)*geom(1)
             xk(i1+8,j1+4) = xk(i1+8,j1+4)+2*((1d0/scaling(1))
     1*dshpg(1,i)*d(1)*dshpg(1,j)*dj*wg(intpoi)*geom(1)
     2+(-scaling(1)/(scaling(2)**2d0))*dshpg(2,i)*d(3)*dshpg(2,j)*dj
     3*wg(intpoi)*geom(1))
             xk(i1+8,j1+5) = xk(i1+8,j1+5)
             xk(i1+9,j1+4) = xk(i1+9,j1+4)
             xk(i1+9,j1+5) = xk(i1+9,j1+5)+2d0
     1*((-scaling(1)/(scaling(2)**2d0))*dshpg(2,i)*d(1)*dshpg(2,j)*dj
571     2*wg(intpoi)*geom(1)+(1d0/scaling(1))*dshpg(1,i)*d(3)*dshpg(1,j)*dj
     3*wg(intpoi)*geom(1))
             xk(i1+8,j1+8) = xk(i1,j1)
```

```
              xk(i1+8,j1+9) = xk(i1,j1+1)
              xk(i1+9,j1+8) = xk(i1+1,j1)
              xk(i1+9,j1+9) = xk(i1+1,j1+1)
    * 2nd order x y derivative
              xk(i1+10,j1) = xk(i1+10,j1)+(-1d0/(scaling(1)**2d0))
         1*dshpg(1,i)*dshpg(1,j)*d(1)*dj*wg(intpoi)*geom(1)
         2+(-1d0/(scaling(2)**2d0))*dshpg(2,i)*d(3)*dshpg(2,j)*dj*wg(intpoi)
581      3*geom(1)
              xk(i1+10,j1+1) = xk(i1+10,j1+1)
              xk(i1+11,j1) = xk(i1+11,j1)
              xk(i1+11,j1+1) = xk(i1+11,j1+1)+(-1d0/(scaling(2)**2d0))
         1*dshpg(2,i)*d(1)*dshpg(2,j)*dj*wg(intpoi)*geom(1)
         2+(-1d0/(scaling(1)**2))*dshpg(1,i)*d(3)*dshpg(1,j)*dj*wg(intpoi)
         3*geom(1)
              xk(i1+10,j1+2) = xk(i1+10,j1+2)+(1d0/scaling(1))*dshpg(1,i)
         1*d(1)*dshpg(1,j)*dj*wg(intpoi)*geom(1)
         2+(-scaling(1)/(scaling(2)**2d0))*dshpg(2,i)*d(3)*dshpg(2,j)*dj
591      3*wg(intpoi)*geom(1)
              xk(i1+10,j1+3) = xk(i1+10,j1+3)
              xk(i1+11,j1+2) = xk(i1+11,j1+2)
              xk(i1+11,j1+3) = xk(i1+11,j1+3)
         1+(-scaling(1)/(scaling(2)**2))*dshpg(2,i)*d(1)*dshpg(2,j)*dj
         2*wg(intpoi)*geom(1)+(1d0/scaling(1))*dshpg(1,i)*d(3)*dshpg(1,j)*dj
         3*wg(intpoi)*geom(1)
              xk(i1+10,j1+4) = xk(i1+10,j1+4)
         1+(-scaling(2)/(scaling(1)**2d0))*dshpg(1,i)*d(1)*dshpg(1,j)*dj
         2*wg(intpoi)*geom(1)+(1d0/scaling(2))*dshpg(2,i)*d(3)*dshpg(2,j)*dj
601      3*wg(intpoi)*geom(1)
              xk(i1+10,j1+5) = xk(i1+10,j1+5)
              xk(i1+11,j1+4) = xk(i1+11,j1+4)
              xk(i1+11,j1+5) = xk(i1+11,j1+5)+(1d0/scaling(2))*dshpg(2,i)
         1*d(1)*dshpg(2,j)*dj*wg(intpoi)*geom(1)
         2+(-scaling(2)/(scaling(1)**2d0))*dshpg(1,i)*d(3)*dshpg(1,j)*dj
         3*wg(intpoi)*geom(1)
              xk(i1+10,j1+10) = xk(i1,j1)
              xk(i1+10,j1+11) = xk(i1,j1+1)
              xk(i1+11,j1+10) = xk(i1+1,j1)
611           xk(i1+11,j1+11) = xk(i1+1,j1+1)
    * 2nd order y x derivative
              xk(i1+12,j1) = xk(i1+12,j1)+(-1d0/(scaling(1)**2d0))
         1*dshpg(1,i)*dshpg(1,j)*d(1)*dj*wg(intpoi)*geom(1)
         2+(-1d0/(scaling(2)**2d0))*dshpg(2,i)*d(3)*dshpg(2,j)*dj*wg(intpoi)
         3*geom(1)
              xk(i1+12,j1+1) = xk(i1+12,j1+1)
              xk(i1+13,j1) = xk(i1+13,j1)
              xk(i1+13,j1+1) = xk(i1+13,j1+1)+(-1d0/(scaling(2)**2d0))
         1*dshpg(2,i)*d(1)*dshpg(2,j)*dj*wg(intpoi)*geom(1)
621      2+(-1d0/(scaling(1)**2d0))*dshpg(1,i)*d(3)*dshpg(1,j)*dj*wg(intpoi)
         3*geom(1)
              xk(i1+12,j1+2) = xk(i1+12,j1+2)+(1d0/scaling(1))*dshpg(1,i)
         1*d(1)*dshpg(1,j)*dj*wg(intpoi)*geom(1)
         2+(-scaling(1)/(scaling(2)**2d0))*dshpg(2,i)*d(3)*dshpg(2,j)*dj
         3*wg(intpoi)*geom(1)
              xk(i1+12,j1+3) = xk(i1+12,j1+3)
              xk(i1+13,j1+2) = xk(i1+13,j1+2)
              xk(i1+13,j1+3) = xk(i1+13,j1+3)
         1+(-scaling(1)/(scaling(2)**2))*dshpg(2,i)*d(1)*dshpg(2,j)*dj
631      2*wg(intpoi)*geom(1)+(1d0/scaling(1))*dshpg(1,i)*d(3)*dshpg(1,j)*dj
         3*wg(intpoi)*geom(1)
              xk(i1+12,j1+4) = xk(i1+12,j1+4)
         1+(-scaling(2)/(scaling(1)**2d0))*dshpg(1,i)*d(1)*dshpg(1,j)*dj
```

```
     2*wg(intpoi)*geom(1)+(1d0/scaling(2))*dshpg(2,i)*d(3)*dshpg(2,j)*dj
     3*wg(intpoi)*geom(1)
            xk(i1+12,j1+5) = xk(i1+12,j1+5)
            xk(i1+13,j1+4) = xk(i1+13,j1+4)
            xk(i1+13,j1+5) = xk(i1+13,j1+5)+(1d0/scaling(2))*dshpg(2,i)
     1*d(1)*dshpg(2,j)*dj*wg(intpoi)*geom(1)
641  2+(-scaling(2)/(scaling(1)**2d0))*dshpg(1,i)*d(3)*dshpg(1,j)*dj
     3*wg(intpoi)*geom(1)
            xk(i1+12,j1+12) = xk(i1,j1)
            xk(i1+12,j1+13) = xk(i1,j1+1)
            xk(i1+13,j1+12) = xk(i1+1,j1)
            xk(i1+13,j1+13) = xk(i1+1,j1+1)
            i1=i1+ndeg
 130       continue
           j1=j1+ndeg
 120     continue
651  C
      do i=15,16
         do j=1,2
            xk(i,j) = xk(j,i)
            xk(i+2,j) = xk(j+2,i)
            xk(i+2,j+2) = xk(j+2,i+2)
            xk(i+4,j) = xk(j+4,i)
            xk(i+4,j+4) = xk(j+4,i+4)
            xk(i+6,j) = xk(j+6,i)
            xk(i+6,j+2) = xk(j+6,i+2)
661         xk(i+6,j+6) = xk(j+6,i+6)
            xk(i+8,j) = xk(j+8,i)
            xk(i+8,j+4) = xk(j+8,i+4)
            xk(i+8,j+8) = xk(j+8,i+8)
            xk(i+10,j) = xk(j+10,i)
            xk(i+10,j+2) = xk(j+10,i+2)
            xk(i+10,j+4) = xk(j+10,i+4)
            xk(i+10,j+10) = xk(j+10,i+10)
            xk(i+12,j) = xk(j+12,i)
            xk(i+12,j+2) = xk(j+12,i+2)
671         xk(i+12,j+4) = xk(j+12,i+4)
            xk(i+12,j+12) = xk(j+12,i+12)
         enddo
      enddo
      do i=29,30
         do j=1,2
            xk(i,j) = xk(j,i)
            xk(i+2,j) = xk(j+2,i)
            xk(i+2,j+2) = xk(j+2,i+2)
            xk(i+4,j) = xk(j+4,i)
681         xk(i+4,j+4) = xk(j+4,i+4)
            xk(i+6,j) = xk(j+6,i)
            xk(i+6,j+2) = xk(j+6,i+2)
            xk(i+6,j+6) = xk(j+6,i+6)
            xk(i+8,j) = xk(j+8,i)
            xk(i+8,j+4) = xk(j+8,i+4)
            xk(i+8,j+8) = xk(j+8,i+8)
            xk(i+10,j) = xk(j+10,i)
            xk(i+10,j+2) = xk(j+10,i+2)
            xk(i+10,j+4) = xk(j+10,i+4)
691         xk(i+10,j+10) = xk(j+10,i+10)
            xk(i+12,j) = xk(j+12,i)
            xk(i+12,j+2) = xk(j+12,i+2)
            xk(i+12,j+4) = xk(j+12,i+4)
            xk(i+12,j+12) = xk(j+12,i+12)
         enddo
```

```
      enddo
      do i=29,30
         do j=15,16
            xk(i,j) = xk(j,i)
            xk(i+2,j) = xk(j+2,i)
            xk(i+2,j+2) = xk(j+2,i+2)
            xk(i+4,j) = xk(j+4,i)
            xk(i+4,j+4) = xk(j+4,i+4)
            xk(i+6,j) = xk(j+6,i)
            xk(i+6,j+2) = xk(j+6,i+2)
            xk(i+6,j+6) = xk(j+6,i+6)
            xk(i+8,j) = xk(j+8,i)
            xk(i+8,j+4) = xk(j+8,i+4)
            xk(i+8,j+8) = xk(j+8,i+8)
            xk(i+10,j) = xk(j+10,i)
            xk(i+10,j+2) = xk(j+10,i+2)
            xk(i+10,j+4) = xk(j+10,i+4)
            xk(i+10,j+10) = xk(j+10,i+10)
            xk(i+12,j) = xk(j+12,i)
            xk(i+12,j+2) = xk(j+12,i+2)
            xk(i+12,j+4) = xk(j+12,i+4)
            xk(i+12,j+12) = xk(j+12,i+12)
         enddo
      enddo
      do i=43,44
         do j=1,2
            xk(i,j) = xk(j,i)
            xk(i+2,j) = xk(j+2,i)
            xk(i+2,j+2) = xk(j+2,i+2)
            xk(i+4,j) = xk(j+4,i)
            xk(i+4,j+4) = xk(j+4,i+4)
            xk(i+6,j) = xk(j+6,i)
            xk(i+6,j+2) = xk(j+6,i+2)
            xk(i+6,j+6) = xk(j+6,i+6)
            xk(i+8,j) = xk(j+8,i)
            xk(i+8,j+4) = xk(j+8,i+4)
            xk(i+8,j+8) = xk(j+8,i+8)
            xk(i+10,j) = xk(j+10,i)
            xk(i+10,j+2) = xk(j+10,i+2)
            xk(i+10,j+4) = xk(j+10,i+4)
            xk(i+10,j+10) = xk(j+10,i+10)
            xk(i+12,j) = xk(j+12,i)
            xk(i+12,j+2) = xk(j+12,i+2)
            xk(i+12,j+4) = xk(j+12,i+4)
            xk(i+12,j+12) = xk(j+12,i+12)
         enddo
      enddo
      do i=43,44
         do j=15,16
            xk(i,j) = xk(j,i)
            xk(i+2,j) = xk(j+2,i)
            xk(i+2,j+2) = xk(j+2,i+2)
            xk(i+4,j) = xk(j+4,i)
            xk(i+4,j+4) = xk(j+4,i+4)
            xk(i+6,j) = xk(j+6,i)
            xk(i+6,j+2) = xk(j+6,i+2)
            xk(i+6,j+6) = xk(j+6,i+6)
            xk(i+8,j) = xk(j+8,i)
            xk(i+8,j+4) = xk(j+8,i+4)
            xk(i+8,j+8) = xk(j+8,i+8)
            xk(i+10,j) = xk(j+10,i)
            xk(i+10,j+2) = xk(j+10,i+2)
```

```
                xk(i+10,j+4) = xk(j+10,i+4)
                xk(i+10,j+10) = xk(j+10,i+10)
761             xk(i+12,j) = xk(j+12,i)
                xk(i+12,j+2) = xk(j+12,i+2)
                xk(i+12,j+4) = xk(j+12,i+4)
                xk(i+12,j+12) = xk(j+12,i+12)
             enddo
          enddo
          do i=43,44
             do j=29,30
                xk(i,j) = xk(j,i)
                xk(i+2,j) = xk(j+2,i)
771             xk(i+2,j+2) = xk(j+2,i+2)
                xk(i+4,j) = xk(j+4,i)
                xk(i+4,j+4) = xk(j+4,i+4)
                xk(i+6,j) = xk(j+6,i)
                xk(i+6,j+2) = xk(j+6,i+2)
                xk(i+6,j+6) = xk(j+6,i+6)
                xk(i+8,j) = xk(j+8,i)
                xk(i+8,j+4) = xk(j+8,i+4)
                xk(i+8,j+8) = xk(j+8,i+8)
                xk(i+10,j) = xk(j+10,i)
781             xk(i+10,j+2) = xk(j+10,i+2)
                xk(i+10,j+4) = xk(j+10,i+4)
                xk(i+10,j+10) = xk(j+10,i+10)
                xk(i+12,j) = xk(j+12,i)
                xk(i+12,j+2) = xk(j+12,i+2)
                xk(i+12,j+4) = xk(j+12,i+4)
                xk(i+12,j+12) = xk(j+12,i+12)
             enddo
          enddo
C
791 c       write(0,*) 'Scaled stiffness K matrix iflag 2 ='
c       do i=1,idss
c       write(0,'(56e25.17)') (xk(i,iidss),iidss=1,idss)
c       enddo
C
c ********************** code r iflag 2
          do 150 i=1,nnode
              de(1,intpoi)= uc(1,i)*dshpgs(1,i)
              de(2,intpoi)= uc(2,i)*dshpgs(2,i)
              de(4,intpoi)= uc(1,i)*dshpgs(2,i)+uc(2,i)*dshpgs(1,i)
801           etota(1,intpoi)= etota(1,intpoi)+de(1,intpoi)
              etota(2,intpoi)= etota(2,intpoi)+de(2,intpoi)
              etota(4,intpoi)= etota(4,intpoi)+de(4,intpoi)
 150      continue
              gsigs(1,intpoi)=d(1)*etota(1,intpoi)+d(2)*etota(2,intpoi)
              gsigs(2,intpoi)=d(2)*etota(1,intpoi)+d(1)*etota(2,intpoi)
              gsigs(3,intpoi)=xu(1)*(gsigs(1,intpoi)+gsigs(2,intpoi))
              gsigs(4,intpoi)=d(3)*etota(4,intpoi)
              sigxx(1,intpoi)=gsigs(1,intpoi)
              sigxx(2,intpoi)=gsigs(2,intpoi)
811           sigxx(3,intpoi)=gsigs(3,intpoi)
              sigxx(4,intpoi)=gsigs(4,intpoi)
          do 160 i=1,nnode
* size-sentivititied matrix
              r(1,i)=r(1,i)+(dshpgs(1,i)*gsigs(1,intpoi)+dshpgs(2,i)
     1*gsigs(4,intpoi))*djs*wg(intpoi)*geom(1)
              r(2,i)=r(2,i)+(dshpgs(2,i)*gsigs(2,intpoi)+dshpgs(1,i)
     1*gsigs(4,intpoi))*djs*wg(intpoi)*geom(1)
* 1st order x derivative
              r(3,i)=r(3,i)+(dshpg(1,i)*(-scaling(2)/(scaling(1)**2d0))*d(1)
```

```
821         1*dshpg(1,i)*uc(1,i)+dshpg(2,i)*d(3)*(1d0/scaling(2))*dshpg(2,i)
            2*uc(1,i))*dj*wg(intpoi)*geom(1)
                  r(4,i)=r(4,i)+(dshpg(2,i)*(1d0/scaling(2))*d(1)*dshpg(2,i)
            1*uc(2,i)+dshpg(1,i)*d(3)*(-scaling(2)/(scaling(1)**2d0))
            2*dshpg(1,i)*uc(2,i))*dj*wg(intpoi)*geom(1)
      * 1st order y derivative
                  r(5,i)=r(5,i)+(dshpg(1,i)*(1d0/scaling(1))*d(1)*dshpg(1,i)
            1*uc(1,i)+dshpg(2,i)*d(3)*(-scaling(1)/(scaling(2)**2d0))
            2*dshpg(2,i)*uc(1,i))*dj*wg(intpoi)*geom(1)
                  r(6,i)=r(6,i)+(dshpg(2,i)*(-scaling(1)/(scaling(2)**2d0))*d(1)
831         1*dshpg(2,i)*uc(2,i)+dshpg(1,i)*d(3)*(1d0/scaling(1))*dshpg(1,i)
            2*uc(2,i))*dj*wg(intpoi)*geom(1)
      * 2nd order x derivative
                  r(7,i)=r(7,i)+(2d0*scaling(2)/(scaling(1)**3d0))*dshpg(1,i)
            1*d(1)*dshpg(1,i)*uc(1,i)*dj*wg(intpoi)*geom(1)
                  r(8,i)=r(8,i)+(2d0*scaling(2)/(scaling(1)**3d0))*dshpg(1,i)
            1*d(3)*dshpg(1,i)*uc(2,i)*dj*wg(intpoi)*geom(1)
      * 2nd order y derivative
                  r(9,i)=r(9,i)+(2d0*scaling(1)/(scaling(2)**3d0))*dshpg(2,i)
            1*d(3)*dshpg(2,i)*uc(1,i)*dj*wg(intpoi)*geom(1)
841               r(10,i)=r(10,i)+(2d0*scaling(1)/(scaling(2)**3d0))*dshpg(2,i)
            1*d(1)*dshpg(2,i)*uc(2,i)*dj*wg(intpoi)*geom(1)
      * 2nd order x y derivative
                  r(11,i)=r(11,i)+(dshpg(1,i)*(-1d0/(scaling(1)**2d0))*d(1)
            1*dshpg(1,i)*uc(1,i)+dshpg(2,i)*d(3)*(-1d0/(scaling(2)**2d0))
            2*dshpg(2,i)*uc(1,i))*dj*wg(intpoi)*geom(1)
                  r(12,i)=r(12,i)+(dshpg(2,i)*(-1d0/(scaling(2)**2d0))*d(1)
            1*dshpg(2,i)*uc(2,i)+dshpg(1,i)*d(3)*(-1d0/(scaling(1)**2d0))
            2*dshpg(1,i)*uc(2,i))*dj*wg(intpoi)*geom(1)
      * 2nd order y x derivative
851               r(13,i)=r(13,i)+(dshpg(1,i)*(-1d0/(scaling(1)**2d0))*d(1)
            1*dshpg(1,i)*uc(1,i)+dshpg(2,i)*d(3)*(-1d0/(scaling(2)**2d0))
            2*dshpg(2,i)*uc(1,i))*dj*wg(intpoi)*geom(1)
                  r(14,i)=r(14,i)+(dshpg(2,i)*(-1d0/(scaling(2)**2d0))*d(1)
            1*dshpg(2,i)*uc(2,i)+dshpg(1,i)*d(3)*(-1d0/(scaling(1)**2d0))
            2*dshpg(1,i)*uc(2,i))*dj*wg(intpoi)*geom(1)
       160  continue
      C
      c      write(0,*) 'Scaled internal forces iflag 2 ='
      c      do i=1,ndeg
861   c       write(0,'(e13.5,2x,e13.5,2x,e13.5,2x,e13.5)') r(i,1),r(i,2),
      c     *r(i,3),r(i,nnode)
      c      enddo
      C
               elseif (iflag.eq.3.and.intpoi.eq.1) then
      C
                     jtype=-jtype
      C
               elseif (iflag.eq.4) then
      C
871   c ************************** code r iflag 4
            do 170 i=1,nnode
               de(1,intpoi)= uc(1,i)*dshpgs(1,i)
               de(2,intpoi)= uc(2,i)*dshpgs(2,i)
               de(4,intpoi)= uc(1,i)*dshpgs(2,i)+uc(2,i)*dshpgs(1,i)
               etota(1,intpoi)= etota(1,intpoi)+de(1,intpoi)
               etota(2,intpoi)= etota(2,intpoi)+de(2,intpoi)
               etota(4,intpoi)= etota(4,intpoi)+de(4,intpoi)
       170  continue
               gsigs(1,intpoi)=d(1)*etota(1,intpoi)+d(2)*etota(2,intpoi)
881            gsigs(2,intpoi)=d(2)*etota(1,intpoi)+d(1)*etota(2,intpoi)
               gsigs(3,intpoi)=xu(1)*(gsigs(1,intpoi)+gsigs(2,intpoi))
```

```
            gsigs(4,intpoi)=d(3)*etota(4,intpoi)
            sigxx(1,intpoi)=gsigs(1,intpoi)
            sigxx(2,intpoi)=gsigs(2,intpoi)
            sigxx(3,intpoi)=gsigs(3,intpoi)
            sigxx(4,intpoi)=gsigs(4,intpoi)
         do 180 i=1,nnode
     * size-sentivititied matrix
            r(1,i)=r(1,i)+(dshpgs(1,i)*gsigs(1,intpoi)+dshpgs(2,i)
891        1*gsigs(4,intpoi))*djs*wg(intpoi)*geom(1)
            r(2,i)=r(2,i)+(dshpgs(2,i)*gsigs(2,intpoi)+dshpgs(1,i)
        1*gsigs(4,intpoi))*djs*wg(intpoi)*geom(1)
     * 1st order x derivative
            r(3,i)=r(3,i)+(dshpg(1,i)*(-scaling(2)/(scaling(1)**2d0))*d(1)
        1*dshpg(1,i)*uc(1,i)+dshpg(2,i)*d(3)*(1d0/scaling(2))*dshpg(2,i)
        2*uc(1,i))*dj*wg(intpoi)*geom(1)
            r(4,i)=r(4,i)+(dshpg(2,i)*(1d0/scaling(2))*d(1)*dshpg(2,i)
        1*uc(2,i)+dshpg(1,i)*d(3)*(-scaling(2)/(scaling(1)**2d0))
        2*dshpg(1,i)*uc(2,i))*dj*wg(intpoi)*geom(1)
901  * 1st order y derivative
            r(5,i)=r(5,i)+(dshpg(1,i)*(1d0/scaling(1))*d(1)*dshpg(1,i)
        1*uc(1,i)+dshpg(2,i)*d(3)*(-scaling(1)/(scaling(2)**2d0))
        2*dshpg(2,i)*uc(1,i))*dj*wg(intpoi)*geom(1)
            r(6,i)=r(6,i)+(dshpg(2,i)*(-scaling(1)/(scaling(2)**2d0))*d(1)
        1*dshpg(2,i)*uc(2,i)+dshpg(1,i)*d(3)*(1d0/scaling(1))*dshpg(1,i)
        2*uc(2,i))*dj*wg(intpoi)*geom(1)
     * 2nd order x derivative
            r(7,i)=r(7,i)+(2d0*scaling(2)/(scaling(1)**3d0))*dshpg(1,i)
        1*d(1)*dshpg(1,i)*uc(1,i)*dj*wg(intpoi)*geom(1)
911         r(8,i)=r(8,i)+(2d0*scaling(2)/(scaling(1)**3d0))*dshpg(1,i)
        1*d(3)*dshpg(1,i)*uc(2,i)*dj*wg(intpoi)*geom(1)
     * 2nd order y derivative
            r(9,i)=r(9,i)+(2d0*scaling(1)/(scaling(2)**3d0))*dshpg(2,i)
        1*d(3)*dshpg(2,i)*uc(1,i)*dj*wg(intpoi)*geom(1)
            r(10,i)=r(10,i)+(2d0*scaling(1)/(scaling(2)**3d0))*dshpg(2,i)
        1*d(1)*dshpg(2,i)*uc(2,i)*dj*wg(intpoi)*geom(1)
     * 2nd order x y derivative
            r(11,i)=r(11,i)+(dshpg(1,i)*(-1d0/(scaling(1)**2d0))*d(1)
        1*dshpg(1,i)*uc(1,i)+dshpg(2,i)*d(3)*(-1d0/(scaling(2)**2d0))
921     2*dshpg(2,i)*uc(1,i))*dj*wg(intpoi)*geom(1)
            r(12,i)=r(12,i)+(dshpg(2,i)*(-1d0/(scaling(2)**2d0))*d(1)
        1*dshpg(2,i)*uc(2,i)+dshpg(1,i)*d(3)*(-1d0/(scaling(1)**2d0))
        2*dshpg(1,i)*uc(2,i))*dj*wg(intpoi)*geom(1)
     * 2nd order y x derivative
            r(13,i)=r(13,i)+(dshpg(1,i)*(-1d0/(scaling(1)**2d0))*d(1)
        1*dshpg(1,i)*uc(1,i)+dshpg(2,i)*d(3)*(-1d0/(scaling(2)**2d0))
        2*dshpg(2,i)*uc(1,i))*dj*wg(intpoi)*geom(1)
            r(14,i)=r(14,i)+(dshpg(2,i)*(-1d0/(scaling(2)**2d0))*d(1)
        1*dshpg(2,i)*uc(2,i)+dshpg(1,i)*d(3)*(-1d0/(scaling(1)**2d0))
931     2*dshpg(1,i)*uc(2,i))*dj*wg(intpoi)*geom(1)
  180  continue
     C
     c       write(0,*) 'Scaled internal forces iflag 4 ='
     c       do i=1,ndeg
     c        write(0,'(e13.5,2x,e13.5,2x,e13.5,2x,e13.5)') r(i,1),r(i,2),
     c      *r(i,3),r(i,nnode)
     c       enddo
     C
     c       write(0,*) 'etota iflag 4 ='
941  c       do i=1,ngenel
     c       write(0,'(e13.5,2x,e13.5,2x,e13.5,2x,e13.5)') etota(i,1),
     c      1etota(i,2),etota(i,3),etota(i,intel)
     c       enddo
```

```
C
c       write(0,*) 'gsigs iflag 4 ='
c       do i=1,ngenel
c       write(0,'(e13.5,2x,e13.5,2x,e13.5,2x,e13.5)') gsigs(i,1),
c      1gsigs(i,2),gsigs(i,3),gsigs(i,intel)
c       enddo
C
c       write(0,*) 'de iflag 4 ='
c       do i=1,ngenel
c       write(0,'(e13.5,2x,e13.5,2x,e13.5,2x,e13.5)') de(i,1),de(i,2),
c      1de(i,3),de(i,intel)
c       enddo
C
c       write(0,*) 'sigxx iflag 4 ='
c       do i=1,nstrmu
c       write(0,'(e13.5,2x,e13.5,2x,e13.5,2x,e13.5)') sigxx(i,1),
c      1sigxx(i,2),sigxx(i,3),sigxx(i,intel)
c       enddo
C
          elseif (iflag.eq.5.and.intpoi.eq.4) then
C
        do i = 1,nnode
           meanu(1,i) = uc(1,i)+0.5d0*(scaling(3)**2d0)*uc(7,i)
     1+0.5*(scaling(3)*scaling(4)*scaling(5))*uc(11,i)
     2+0.5*(scaling(3)*scaling(4)*scaling(5))*uc(13,i)
     3+0.5*(scaling(4)**2d0)*uc(9,i)
           meanu(2,i) = uc(2,i)+0.5d0*(scaling(3)**2d0)*uc(8,i)
     1+0.5*(scaling(3)*scaling(4)*scaling(5))*uc(12,i)
     2+0.5*(scaling(3)*scaling(4)*scaling(5))*uc(14,i)
     3+0.5*(scaling(4)**2d0)*uc(10,i)
           dev(1,i) = sqrt((scaling(3)**2d0)*(uc(3,i)**2d0)
     1+2*(scaling(3)*scaling(4)*scaling(5))*uc(3,i)*uc(5,i)
     2+(scaling(4)**2d0)*uc(5,i)**2d0)
           dev(2,i) = sqrt((scaling(3)**2d0)*(uc(4,i)**2d0)
     1+2*(scaling(3)*scaling(4)*scaling(5))*uc(4,i)*uc(6,i)
     2+(scaling(4)**2d0)*uc(6,i)**2d0)
           cov(i) = (scaling(3)**2d0)*uc(3,i)*uc(4,i)
     1+(scaling(3)*scaling(4)*scaling(5))*uc(3,i)*uc(6,i)
     2+(scaling(3)*scaling(4)*scaling(5))*uc(4,i)*uc(5,i)
     3+(scaling(4)**2d0)*uc(5,i)*uc(6,i)
        enddo
C
c       write(0,*) 'Perturbation method'
c       do inode=1,nnode
c       write(0,'(''node ='',i5,2x,''mean dispt x ='',e13.5,2x,
c      1''mean dispt y ='',e13.5,2x,''dev x ='',e13.5,2x,''dev y ='',
c      2e13.5,2x,''cov dispt x dispt y =', e13.5)') inode,meanu(1,inode),
c      3meanu(2,inode),dev(1,inode),dev(2,inode),cov(inode)
c       enddo
C
c       write(0,*)'start'
        do l=1,nnode
c          write(0,'(''node ='',I3)')l
c          write(0,'(''elecon('',I2,'','',I3,'')'')')l,m
           k=elecon(l,m)
           do i=1,ndeg
c             write(0,'(''i ='',I2,''k ='',I3)')i,k
              result(i,k)=uc(i,l)
           enddo
           do i=1,2
c             write(0,'(''i+14 ='',I2,''k ='',I3)')i+14,k
              result(i+14,k)=meanu(i,l)
c             write(0,'(''i+16 ='',I2,''k ='',I3)')i+16,k
```

```
                result(i+16,k)=dev(i,l)
              enddo
c             write(0,'(''i ='',I2,''k ='',I3)')19,k
1011          result(19,k)=cov(l)
            enddo
C
c           write(0,*)'All results'
c           do l=1,nnode
c              k=elecon(1,m)
c              write(0,'(''k ='',i5,2x,''result ='',19e13.5)') k,
c           *(result(i,k),i=1,19)
c            enddo
C
1021                jtype=-jtype
C
              endif
C
            enddo
c end the integration point loop
C
            return
            end
c-----------------------------------------------------------------------c
1031  c UFCONN user subroutine of MSC-Marc to add/modify element          c
      c connectivity                                                      c
      ************************************************************************
      c Running with the additional input file text - model definition    c
      c section option command:                                           c
      c UFCONN                                                            c
      c all_elements                                                      c
      c NB: given that all_elements are defined as a set containing all the  c
      c elements of the model                                             c
      ************************************************************************
1041  c Input and output variables explanation:
      c     j       element number
      c     itype   element type
      c     lm      connectivity
      c     nnodmx  maximun number of nodes per element
      ************************************************************************
            subroutine ufconn(j,itype,lm,nnodmxx)
C
            use M_data
C
1051        implicit real*8 (a-h,o-z)
C
      c ** Start of generated type statements **
            integer itype,j,lm,nnodmxx
      c ** End of generated type statements **
            dimension lm(*)
C
            include '../common/dimen'
C
            integer   allsta
1061  C    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      c      write(0,*)'UFCONN-------------------------------------------------'
      c      write(0,'('' number of elements='',i5,
      c     *     ''number of nodes='',i5)') numel,numnp
      C
            if (.not.allocated(elecon).and.numel.gt.0) then
               allocate(elecon(nnodmxx,numel),stat=allsta)
      c         write(0,'('' size elecon='',2i5)') nnodmxx,numel
            endif
C
1071  c     store nodes of element j
            if (allocated(elecon)) then
```

```
            elecon(1:nnodmx,j)=lm(1:nnodmx)
c           write(0,'('' element='',i3,'' nodes='',4i5)') j,elecon(1:4,j)
          endif
C
          return
          end
c-----------------------------------------------------------------------c
c UPSTNO user subroutine to post-process nodal displacements, and      c
c its derivatives, mean, variance and covariance with respect to the   c
c scale factors                                                        c
************************************************************************
c Running with selecting the following available nodal quantities in   c
c the job results:                                                     c
c - displacement                                                       c
c (- rotation                                                          c
c - external force                                                     c
c - external moment)                                                   c
c - reaction force                                                     c
c - reaction moment                                                    c
c - user nodal quantity 1 to 19 (User Sub UPSTNO) given that           c
c    1 is the displacement in x direction = above displacement         c
c    2 is the displacement in y direction = above displacement         c
c    3 is the first order derivative of the x displacement wrt sx      c
c    4 is the first order derivative of the y displacement wrt sx      c
c    5 is the first order derivative of the x displacement wrt sy      c
c    6 is the first order derivative of the y displacement wrt sy      c
c    7 is the second derivative of the x displacement wrt sx^2         c
c    8 is the second derivative of the y displacement wrt sx^2         c
c    9 is the second derivative of the x displacement wrt sy^2         c
c   10 is the second derivative of the y displacement wrt sy^2         c
c   11 is the second derivative of the x displacement wrt sx,sy        c
c   12 is the second derivative of the y displacement wrt sx,sy        c
c   13 is the second derivative of the x displacement wrt sy,sx = 11   c
c   14 is the second derivative of the y displacement wrt sy,sx = 12   c
c   15 is the mean of the x displacement                               c
c   16 is the mean of the y displacement                               c
c   17 is the deviation of the x displacement                          c
c   18 is the deviation of the y displacement                          c
c   19 is the covariance of the x and y displacement                   c
************************************************************************
          subroutine upstno(nqcode,nodeid,valno,nqncomp,nqtype,
         *nqaver,nqcomptype,nqdatatype,nqcompname)
C
          use M_data
C
          implicit real*8 (a-h,o-z)
C
          dimension valno(*)
          character*24 nqcompname(*)
C     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
          nqncomp=1
C
          valno(1)=result(-nqcode,nodeid)
c          write(0,*)'VALNO-----------------------------------------------',
c          write(0,'(''nqcode ='',i5,2x,''nodeid ='',i5,2x,''valno ='',e13.5)
c         *')nqcode,nodeid,valno(1)
C
          nqtype=0
* indicate that valno represents a scalar
          return
          end
c-----------------------------------------------------------------------c
```

Here is the cb_size_pm_uselem_11_v2011_03.com code:

```
c-----------------------------------------------------------------------c
```

```
c cb_size_pm_uselem_11_v2011_03 common block for Size sensitivity      c
c based perturbation method user-defined element 11 set of module,    c
c common block (present) and user subroutines gathered in             c
c size_pm_uselem_11_v2011_03.f subroutine file                        c
c*********************************************************************
c Content:                                                            c
c - scaling(5) : vector of 5 components storing scale factors         c
c   and/or the mean, the variance, the correlation coefficient of the c
c   element length on x and y direction via usdata subroutine and     c
c   additional input file text -  model definition section option     c
c   command.                                                          c
c*********************************************************************
C
c ** Start of generated type statements **
      real*8 scaling(5)
c ** End of generated type statements **
C
      common/cb_size_pm_uselem_11_v2011_03/scaling
c---------------------------------------------------------------------c
```

Appendix F

TWS-TWS

Here is the data sheet of the studied copper foil used in the NanoInterface project.

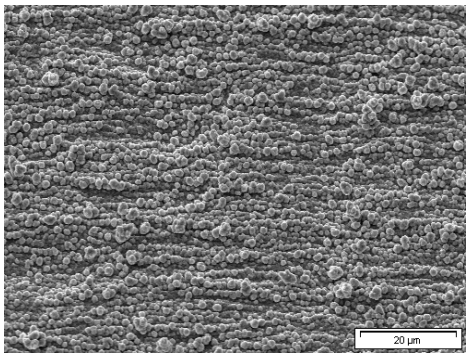
TWS-TWS

Technical Characteristics

TWS-TWS is an advanced double-side treated electro-deposited copper foil designated for use on high performance substrate. The additional bonding treatment applied to the shiny side of the Grade 3 base foil provides "ready-to-use" laminate products for inner layer PCB fabrication.

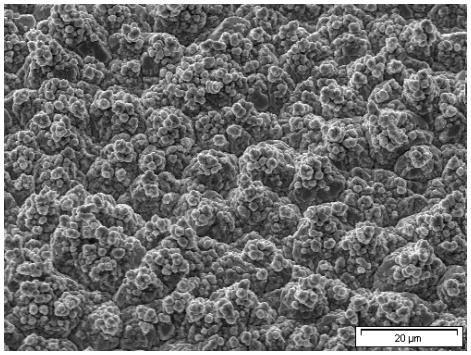
Laminates manufactured with these foils are used to produce inner layer PCB's with high inner layer-to-layer bond strengths on high performance resin systems without the necessity of wet chemical oxide or alternative processing.

Typical substrates would include FR-4 and high T_g epoxy resins, BT blends, cyanate esters, polyimides and advanced thermoplastics.



Treated shiny side

Treated matte side



Typical average properties

TWS-TWS						
MEASURED PARAMETERS	UNITS	PRODUCT GAUGE			IPC	
Nominal Thickness	µm oz.	18 1/2	35 1	70 2	Specification IPC-4562A	Test Method IPC-TM-650
Area Weight (± 5 %)	oz/ft² g/m² g/254 in²	0.52 159 26.1	0.95 290 47.5	1.92 585 95.9	(a)1.2.5, table 1-1 (b)3.4.4 (c)4.6.3	2.2.12
Treated Shiny Side Roughness (Rz)	µm µ.inch	< 5.1 < 201			3.4.5	2.2.17
Treated Matte Side Roughness (Rz)	µm µ.inch	6 - 8 236 - 315	7 - 10 276 - 394	9 - 12 354 - 472	3.4.5	2.2.17
Tensile Strength Transverse at RT	MPa k.Lb/in²	> 276 > 40			3.5.1	2.4.18
Tensile Strength Transverse at 180 °C	MPa k.Lb/in²	> 138 > 20			3.5.1	2.4.18
Elongation Transverse at RT	%	> 6	> 10	> 15	3.5.3	2.4.18
Elongation Transverse at 180 °C	%	> 3			3.5.3	2.4.18
Peel Strength (RT) polyimide [1] *Treated Shiny Side*	N/mm Lb/in	> 0.53 > 3.0	> 0.6 > 3.4	> 0.7 > 4.0	3.5.4	2.4.8
Peel Strength (RT) polyimide[1] *Treated Matte Side*	N/mm Lb/in	> 0.8 > 4.6	> 1.14 > 6.5	> 1.4 > 8.0	3.5.4	2.4.8
High Temp. Tarnish Resistance	-	120 min @ 180 °C in air: pass				
Solderability	-	Complies with IPC specification			3.6.3	2.4.12

[1] Laminate construction with thickness >= 0.5 mm

Advanced Product Features

- Consistent high inner layer-to-layer adhesion as a consequence of the well defined electrochemically applied bonding treatment on the foil's shiny side - eliminating delamination during solder shock at final assembly.
- Elimination of "Pink Ring" lamination failures during PTH processing.
- Improved yield on thin core laminates due to the elimination of the handling damage from multi-stage wet chemical processing.
- Elimination of the capital costs for multi-stage wet chemical oxide or alternative processing.
- Freedom from the variability of wet chemical processing and the associated process, disposal, pollution and environmental costs.
- High temperature elongation - [HTE-Type E / Grade 3] {IPC-4562A / 1.2.4.1} prevents "barrel cracking" failures in multi-layer PCB's.
- Thermally stable microstructure - stable mechanical properties unaffected by thermal excursion from lamination or post laminate baking cycles - which could degrade laminate dimensional stability, warp & twist, and drilling characteristics (nail heading).
- The product meets or exceeds all of the requirements of IPC-4562A when tested on typical epoxy and multifunctional prepregs, in accordance with IPC test methods, including high temperature peel strength, solder shock and accelerated ageing.

Notes

- Double Treated copper foil (TWS-TWS) is designed for use on high performance resin systems and "regular" FR-4 / glass epoxy systems.
- Products can be supplied in both roll and sheeted formats.
- Roll product is available in widths of 150 mm (~ 5.9") to 1360 mm (~ 53.5") {product up to 1430 mm (~ 56") wide product is available on special request}.
- Product is supplied on sturdy cardboard cores with an ID of ~ 80 mm (3 1/8"). Alternative core sizes and materials are available on request.
- Please visit our website (www.circuitfoil.com) for regular updates.

Circuit Foil Luxembourg
POB 9
L – 9501 Wiltz
G.D. of Luxembourg
Phone: +(352) 95 75 51 1
Fax: +(352) 95 75 51 249
E-mail: office@circuitfoil.com
Internet: www.circuitfoil.com

Circuit Foil America
625, rue du Luxembourg
Granby, J2J 2S9
Canada
Phone: +(1) 450 770 8558
Fax: +(1) 450 405 4622
E-mail: cfa@circuitfoil.com

Circuit Foil Trading Inc. (USA)
115, East Glenside Ave./Suite 12
Glenside, PA 19038
USA
Phone: +(1) 215 887 7255
Fax: +(1) 215 887 6911
E-mail: cftinc@circuitfoil.com

Circuit Foil Asia Pacific Co. Ltd.
(Zhangjiagang Free Trade Zone)
Block A, 8 Guangdong Road
Free Trade Zone Zhangjiagang
Jiangsu Province
215 634 People's Republic of China
Phone: +(86) 512 58 32 21 82 80 03
Fax: +(852) 512 58 32 21 82 81
E-mail: cfapzjg@circuitfoil.com

Circuit Foil Asia Pacific Ltd. (HK)
Hong Kong Head Office
Rm. 706, Well Fung Industr. Centre
68 Ta Chuen Ping Street, Kwai Chung
N.T. Hong Kong
Phone: +(852) 24 23 97 56
Fax: +(852) 24 23 70 92
E-mail: cfap@circuitfoil.com

ArcelorMittal

Appendix G

WHITE LIGHT INTERFEROMETRY

Briefly, a White Light Interferometry Scanner is a device for measuring the physical geometrical characteristics of an object by capturing intensity data at a series of positions along the vertical axis, which determines where the surface is located by using the shape of the white-light interferogram, the localized phase of the interferogram, or a combination of both shape and phase. It consists of the superposition of fringes generated by multiple wavelengths, obtaining peak fringe contrast as a function of scan position, that is, the red portion of the object beam interferes with the red portion of the reference beam, the blue interferes with the blue, and so forth. An imaging interferometer is vertically scanned to vary the optical path difference. During this process, a series of interference patterns are formed at each pixel in the instrument's field of view. This results in an interference function, with interference varying as a function of optical path difference. The data are stored digitally and processed in a variety of ways. The Fourier analysis method is used to convert intensity data to the spatial frequency domain, allowing production of an extremely accurate surface map. In addition, there are cross-correlation methods, and analysis in the spatial domain [Wyant, 2011].

References

François Dau. Modèle de poutres: théorie et pratique. January 2005.

DigitalSurf. Mountainsmaps. http://www.digitalsurf.fr/en/index.html, April 2011.

Fortran. Fortran 77. http://www.fortran.com, April 2011.

FrictionCenter. Surfaces and Contact Mechanics, April 2011.

Kotaro Fukasaku. Assignment: Develop, at mesoscopic scale, a stochastic model of surface roughness on the metal oxide polymer interface which predicts the chance of different failure modes (cohesive or adhesive) during its delamination. Febuary 2010.

MA Gutierrez and R. De Borst. Simulation of size-effect behaviour through sensitivity analyses. *Engineering Fracture Mechanics*, 70(16):2269–2279, 2003.

M.A. Gutiérrez and S. Krenk. Stochastic finite element methods. 2004.

Terje Haukaas. Finite element reliability using matlab. http://www.ce.berkeley.edu/projects/ferum/, April 2011.

W.W. Hines, D.C. Montgomery, and D.M.G.C.M. Borror. *Probability and statistics in engineering*. Wiley-India, 2009.

W.S. Kim, I.H. Yun, J.J. Lee, and H.T. Jung. Evaluation of mechanical interlock effect on adhesion strength of polymer-metal interfaces using micro-patterned surface topography. *International Journal of Adhesion and Adhesives*, 2010.

Lucas Lallemant. Numerical homogenization of a rough bi-material interface. May 2011.

H.Y. Lee and J. Qu. Microstructure, adhesion strength and failure path at a polymer/roughened metal interface. *Journal of Adhesion Science and Technology*, 17(2):195–215, 2003.

H.Y. Lee and J. Qu. Dimple-type failures in a polymer/roughened metal system. *Journal of Adhesion Science and Technology*, 18(10):1153–1172, 2004.

MSC.Software. Marc and mentat – advanced nonlinear simulation. http://www.mscsoftware.com/Products/CAE-Tools/Marc-And-Mentat.aspx, April 2011.

SPM Noijen, O. Van der Sluis, PHM Timmermans, and GQ Zhang. Numerical prediction of failure paths at a roughened metal/polymer interface. *Microelectronics reliability*, 49 (9-11):1315–1318, 2009.

PhilipsAppliedTechnologies. Nanointerface project, August 2009.

Jianmin Qu. Lecture: Mechanics of interfaces in microelectronic packaging. 2004.

Reno Rossetti. Semiconductor technologies for power management, June 2005.

Rubert. Roughness parameters. http://www.rubert.co.uk/Ra.htm, April 2011.

Sanyo. Lineup of many and varied packages including ultra-small wlp packages – small package eeprom (2k to 128k). http://semicon.sanyo.com/en/memory/topics/small-eeprom.php, June 2007.

Specialchem. Adhesion theory - Mechanical interlocking. http://www.specialchem4adhesives.com/resources/adhesionguide/index.aspx?id=theory4, April 2011.

G. Stefanou. The stochastic finite element method: past, present and future. *Computer Methods in Applied Mechanics and Engineering*, 198(9-12):1031–1051, 2009.

B. Sudret and A. Der Kiureghian. *Stochastic finite element methods and reliability: a state-of-the-art report*. Dept. of Civil and Environmental Engineering, University of California, 2000.

J. Sugimura. Stochastic Modeling of Surface Roughness. *Japanese Journal of Tribology*, 43 (11):1367–1375, 1998.

T.R. Thomas. *Rough surfaces*. Imperial College Press London, 1999.

Olaf Van der Sluis. Lecture: Fracture mechanics. May 2006a.

Olaf Van der Sluis. Lecture: Multi-scale method. May 2006b.

Olaf Van der Sluis. Nanointerface minutes wp2 meeting. November 2009.

James C. Wyant. White light interferometry. http://www.optics.arizona.edu/jcwyant, April 2011.

C. Xuesheng, Y. Qin, and R. Balendra. Development of a statistical parameter-based surface model for the simulation of variation of surface roughness with contact pressure. *Journal of Materials Processing Technology*, 145(2):247–255, 2004.

Q. Yao and J. Qu. Interfacial versus cohesive failure on polymer-metal interfaces in electronic packagingeffects of interface roughness. *Journal of Electronic Packaging*, 124:127, 2002.

O.C. Zienkiewicz and R.L. Taylor. *The finite element method for solid and structural mechanics*. Butterworth-Heinemann, 2005. ISBN 0750663219.