Identification of material properties and phase distribution of heterogeneous materials through data-driven computational methods : Towards an enhaced constitutive space

Gabriel Valdés-Alonzo

THESE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES
ET L'UNIVERSITAT POLITÈCNICA DE CATALUNYA

ECOLE DOCTORALE N° 602
Sciences pour l'Ingénieur
Spécialité : Mécanique des Solides, des Matériaux, des Structures et des Surfaces

Par
Gabriel VALDÉS-ALONZO

Identification of material properties and phase distribution of heterogeneous materials through data-driven computational methods

Towards an enhanced constitutive space

Thèse présentée et soutenue à l'École Centrale de Nantes le 3 juin 2022
Unité de recherche : UMR 6183, Institut de Recherche en Génie Civil et Mécanique (GeM)

Rapporteurs avant soutenance :

Iciar ALFARO	Professeure, Université de Saragosse
Robin BOUCLIER	Maître de conférences HDR, INSA Toulouse

Composition du Jury :

Président :	Amine AMMAR	Professeur des universités, ENSAM Angers
Examinatrice :	Stefanie REESE	Professeure, RWTH Aachen Université
Dir. de thèse :	Christophe BINETRUY	Professeur des universités, École Centrale de Nantes
Dir. de thèse :	Alberto GARCÍA-GONZÁLEZ	Maître assistant, Universitat Politècnica de Catalunya, Barcelone
Co-encadrant :	Adrien LEYGUE	Chargé de recherche CNRS, École Centrale de Nantes
Invité :	Nahiene HAMILA	Professeur, ENI Brest

This project is part of the Marie Sklodowska-Curie ITN-EJD ProTechTion funded by the European Union Horizon 2020 research and innovation program with grant number 764636.

Acknowledgements

I would like to start this section by thanking the members of the jury for the time and effort that they have put into this thesis. The two reviewers, Iciar Alfaro and Robin Bouclier have accepted to read, correct and evaluate this document, sacrificing their time in doing so. Also to the other members: Amine Ammar for accepting to be the president and be present during the defense, Stefanie Reese for her invaluable input in the topic, and Nahiene Hamila, for his curiosity and availability to take part in this.

Next, I need to thank the people that made all of this possible. I am forever grateful to my advisors, Christophe Binetruy and Adrien Leygue for giving me the opportunity to work with them and putting their trust in an unknown student. This work would not be the same without all the discussions, meeting and conversations that we had during these years. The knowledge that you have is incredible and I have learned so much from you both.

A special mention needs to be made to my third advisor, Berto. I am immensely grateful for all your help, not only in the academic part but also in the personal side. There was no reason for you to make the sacrifice to be part of this project, but you did it anyway, and you brought such a different approach that you made this full experience something special and unique. I never expected to continue working with you after the masters, but life brought us together anyway, so I hope this is also not the last time.

On the less academic side, many thanks to everyone in ECN who was part of this almost 4 years. No matter which building you come from, you were part of the nice environment that made it a joy to come and work every day. On a more personal note, I need to thank all the friends made in the way, but specially the three people that marked my life here in Nantes: Gabi, Smriti and Alexia, because without you three everything would have been so different. Thank you for all the support, reasoning and annoyance that you have brought into my life. Heaven knows nothing would be the same without you. Also, I want to make a special mention to you Steffi: thank you for everything you did and brought all these years.

Finalmente, pero no menos importante, la parte chilena. No hay nadie más importante que nadie que yo pueda mencionar que no sean mis padres, ya que sin ellos no estaría ni cerca de estar donde estoy hoy. Gracias por apoyarme toda una vida, por esforzarse por mi educación, por animarme a ser más y por apoya todos mis proyectos y caprichos. No muchos padres son capaces de hacer tanto por sus hijos, por lo que me siento afortunado de tenerlos. También hacer un pequeño reconocimiento a los pocos amigos que me quedan allá, que a pesar de la distancia aún me apoyan y me quieren. Gracias Val, Rebe, Club de Toby, etc. Finalmente a ti Rocío, que fuiste parte del último cachito, gracias por tu cariño y paciencia.

Probablemente la última iteración.
Dedicado a la familia,
a los amigos que estuvieron y no,
y eternamente a Sasha Grey.

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

Acronyms

DDI	Data-Driven Identification
DDCM	Data-Driven Computational Mechanics
CA	Correspondence Analysis
PCA	Principal Component Analysis
kPCA	Kernel Principal Component Analysis
DIC	Digital Image Correlation
BVP	Boundary Value Problem
FEMU	Finite Element Model Updating
CEGM	Constitutive Equation Gap Method
EGM	Equilibrium Gap Method
RGM	Reciprocity Gap Method
WYPIWYG	What You Prescribe Is What You Get
NN	Neural Network
GMT	Glass Mat Thermoplastic

Symbols

DDI related	
$\boldsymbol{\sigma}$	Stress
$\boldsymbol{\epsilon}$	Strain
$\mathbb{C}$	Pseudo-stiffness / Energy norm term
N^*	Number of material states
N^e	Number of elements
N^n	Number of nodes
N^X	Number of snapshots
N^s	Number of previous strain steps
$\mathrm{i}e^X$	State mapping
r^*	Number of mechanical states per material state
$\|\|\square\|\|^2_{\mathbb{C}}$	Energy norm in phase space
$\square_e$	Relative to element e, mechanical state
$\square^*$	Relative to material state
$\square^X$	Relative to snapshot X
$\square^*_{\mathrm{i}}$	Relative to material state i
$\square^X_e$	Relative to element e in snapshot X

Error related	
e_2	Relative 2-norm error
$e_{\mathbb{C}}$	Relative $\mathbb{C}$-norm error
$e^{\square}$	Relative error of quantity $\square$
Chapter 1	
$\boldsymbol{S}$	Piola-Kirchhoff stress tensor
$\boldsymbol{E}$	Green-Lagrange strain tensor
$\boldsymbol{P}$	First Piola-Kirchhoff stress
$\boldsymbol{F}$	Deformation gradient
Ψ	Helmholtz free-energy function
E	Young's modulus
ν	Poisson ratio
Chapter 3	
$\boldsymbol{N}$	Contingency table
$\boldsymbol{P}$	Corresponding matrix
$\boldsymbol{S}$	Standard residual matrix of $\boldsymbol{P}$
$\boldsymbol{r}, \boldsymbol{c}$	Row and Column masses (vector)
$\boldsymbol{D}_r, \boldsymbol{D}_c$	Diagonal Row and Column masses (matrix)
$\boldsymbol{U}, \boldsymbol{V}$	Singular vectors of $\boldsymbol{S}$
$\boldsymbol{D}_\alpha$	Singular values of $\boldsymbol{S}$
$\boldsymbol{\Phi}, \boldsymbol{\Gamma}$	Standard coordinates of Rows and Columns
$\boldsymbol{F}, \boldsymbol{G}$	Principal coordinates of Rows and Columns
λ_k	Principal inertias of $\boldsymbol{S}$
Chapter 5	
μ	Mean
s	Standard deviation
Z	Normalized distribution
$\bar{\square}$	Averaged quantity
$\boldsymbol{X}$	Data matrix ordered in rows
$\boldsymbol{C}$	Covariance matrix of data ($\boldsymbol{X}\boldsymbol{X}^T$)
$\boldsymbol{Q}$	Gram matrix of data ($\boldsymbol{X}^T\boldsymbol{X}$)
$\boldsymbol{U}$	Singular vectors of $\boldsymbol{C}$
$\boldsymbol{V}$	Singular vectors of $\boldsymbol{Q}$
$\boldsymbol{\Lambda}$	Matrix of singular values of $\boldsymbol{C}$
$\boldsymbol{Z}$	Matrix of principal components ($\boldsymbol{U}^T\boldsymbol{X}$)
$\hat{\square}$	Relative to kernel space
$\boldsymbol{K}$	Kernel matrix
$\tilde{\square}$	Nyström approximated quantities

Chapter 1

General introduction

1.1 Constitutive relations

In the field of computational mechanics, the goal is to solve problems that reflect everyday phenomena. This is done by solving equations that represent the physical conditions to which samples are subjected. These physical principles are universal and they apply to all materials equally, but they do not give any insight into the specific response to different conditions such as loads and temperature changes.

1.1.1 Definition

Constitutive relations [1] are specifically targeted to specify the mechanical and thermal properties of any particular material, based on their internal constitution. They describe relationships among kinematic, material and thermal equations to allow for the formulation of well-posed problems in continuum mechanics. They aim to physically describe idealized material models that serve as a reference for representing real materials. Constitutive relations are complex in nature since they attempt to define internal properties that are not known. Some materials, such as linear elastic materials, have simple definitions that we can model with one parameter, however, many others do not follow a straightforward behavior and different techniques need to be used to obtain an estimation. Another point to consider is that in general, materials do not keep the same behavior forever. Things like aging, temperature or high loads can alter the behavior that is observed in normal conditions, meaning that we would need to find a different model for a different circumstance. In that sense, it is better to think of constitutive relations as representing particular behaviors rather than particular materials.

The definition of the constitutive relation is dependent on the material studied, and typically should define relationship between relevant quantities. Common examples of this relations mathematically defined are

- Small deformations ($\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon})$): in the cases where the deformations that the material is subjected are small, we have that the constitutive relation can be defined in terms of the infinitesimal strain $\boldsymbol{\epsilon}$ and stress $\boldsymbol{\sigma}$.
- Large deformations ($\boldsymbol{S} = \boldsymbol{S}(\boldsymbol{E})$): if the deformation is large enough, the deformations of the material are defined in terms of the *Green-Lagrange strain* tensor $\boldsymbol{E}$ and the respective second Piola-Kirchhoff stress tensor $\boldsymbol{S}$.

- Hyperelastic materials ($\boldsymbol{P} = \boldsymbol{P}(\boldsymbol{F})$): for hyperelastic materials, the constitutive relation is derived from energy equations, where it is more convenient to work in terms of a deformation gradient $\boldsymbol{F}$ and its respective first Piola-Kirchhoff stress $\boldsymbol{P}$.

- Linear viscoelasticity ($\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon}, t)$): in linear viscoelastic materials we have a similar dependence of stresses on strains, with the difference that viscoelastic materials retain a *memory* of previous steps, so the relation is not instantaneous as in linear elastic materials, but rather dependent on the full history of strains.

There are many more examples for different kind of materials which not necessarily can be defined by mathematical expression. Ideally, if we have a mathematical expression, we want to use the constitutive relation to solve a *Boundary Value Problem* (BVP) that represents the situation that we are studying. To pose a problem correctly we need to combine physical principles such as

- conservation of mass, if we have transient systems;

- equations of motion, for all moving bodies;

- energy equations, for cases where we need to account for temperature and other energy sources;

with an ad-hoc constitutive relation and adequate boundary conditions that restrict the problem. Constitutive relations are an essential part of the problem and the way they are defined or obtained should not be taken lightly, which is the reason why so much work and research is put into the identification of material properties.

1.1.2 Identification of constitutive relations

Constitutive relations aim to recreate the behavior of a material, most of the time through a mathematical equation. There are different ways in which these relations can be obtained:

1. Derived from first principles: First principles are propositions that cannot be deduced by another proposition or assumption. In material science, working with first principles means that our deduction process starts at the level of established science, without relying on empirical models or parameter fitting. In practice, this means that, if we want to understand the behavior of a particular material we rely on the physical principles associated to the microstructure of the material, that then can be used to understand the behavior on a macro scale. Materials like rubber [2, 3], colloidal suspensions [4], some polymers [5] and perfect gases [6] can have their constitutive relations derived completely in a mathematical way, without the need of fitting parameters.

2. Based or estimated from physical principles: If we have the knowledge of how certain physical phenomena happen inside of a material, we can use these principles to obtain an estimation of how the constitutive relation might be described. In this case we do not necessarily know exactly what happens, but we can approximate through the use of parameters that will act as a simplification of the more complicated real behavior that is not explicitly formulated. An example of this are hyperelastic

materials, where we can derive the constitutive relation $\boldsymbol{P} = \boldsymbol{\sigma}(\boldsymbol{F})$ from a postulated *Helmholtz free-energy function* Ψ dependent on $\boldsymbol{F}$ [7]. Another example are linear elastic materials, which are arguably one of the easiest models to work with. In the simplest of cases, with one parameter we can define the relationship between strain (ϵ) and stress (σ). This equation is also derive from energy principles [8] that are later simplified.

3. Fitted structures in experimental data: In cases where we do not have any knowledge of the internal interactions of the material or they are too complicated to be posed mathematically, we can rely on the last method which aims to estimate the constitutive relations through the use of external tools. In here, we have tools such as *Neural Networks* [9, 10], *Data-Driven* methods [11], *Manifold Learning* [12], *What-You-Prescribe-Is-What-You-Get* methods [13, 14], among others. The idea of all these methods is to find a way around the complex relations between strains and stresses in a material to reach to a solution, without necessarily knowing the process in between. All these techniques rely on different principles in combination with some physical constraints that achieve the same goal with as less human input as possible.

As can be seen, the more we go down this list, the least we know about the material itself, so we have a need for more parameters to define the structure that will represent the constitutive relation. We have endless possibilities for defining a constitutive relation, but to be an adequate one it should be reasonably in agreement with experimental observations of the material. This expectation, in turn, can be problematic. In the search of an adequate fit of our data, we inevitable introduce a bias in our modeling, by making certain assumptions or choosing one model over the other. Even if the model works, a biased fit can give us a wrong idea of the material, so choosing a model is a delicate process that is inherently a source of error for any problem.

1.1.3 Parameter estimation

To identify the properties of the materials to be used in BVP, we need to estimate the values of the parameters that define the constitutive relations. Depending on the cases, estimations can be obtained through testing, where we subject samples of particular materials to certain conditions, in certain ranges of interest, that might represent how the material would behave when it is solicited in a common setting. It is important to remember that constitutive relations do not necessarily represent the entirety of the material behavior, but sometimes they represent only certain parts of it, which is why it is important that the testing is performed in ranges that would suit our needs.

One of the big problems is to understand what the parameter represents. In certain cases we have that parameters might be associated to physical properties of the material, such as stiffness or relaxation times. In this cases we can have an idea of the possible values just by observing how the material is responding to certain stimuli. Some other parameters come as byproducts of certain equations and in those cases it is more difficult to have a grasp on their proper meaning, and by consequence which values they should take.

If we, for example, take a homogeneous material which is defined through its strain and stress relation, we aim to measure values of the strain through the deformation of a sample, as well as values of stress through the forces that we apply to create this

deformation, and with both these values we can estimate the parameter that relates them. Some of the tests that are used for this purpose are

- Uniaxial traction/compression: in this test, a deformation is applied in a sample in only one direction. Given the known quantities such as forces, deformations and geometry of the sample, a set of data points can be obtained that can be used to estimate strain and stresses in the sample.
- Biaxial traction/compression: similar to the previous case, but the test is performed in two directions simultaneously. It is targeted mostly to materials that present orthotropic behavior.
- Bending: if a material will be used mostly in bending settings (such as beams in structures), it is of interest to know how the material behaves. Different variations exists, such as the one or two point bending, the latter having the particular property of having a constant shear in between the force points. With the values measured, a bending module can be obtained to simplify calculations, but also information of tensile or compressive properties can be obtained if there is access to more accurate measuring.

In more complicated settings, performing only testing might not be sufficient to obtain a value for the parameters, so we need to resort to more sophisticated ideas. For example, in most of the cases, a proper value of the stress field cannot be calculated since they represent internal interactions that cannot be observed. The common approach to estimate parameters in these cases is trying to minimize a distance between what is measured in a sample and its simulations, which should provide optimal parameters for different models. In more modern settings, the availability of imaging techniques has improved the data collection, allowing for measurements of full fields in the sample rather than just point-wise approximation. Some popular methods in this vein are [15]:

- Finite Element Model Updating (FEMU): iterative computation of finite elements simulations while varying parameters in order to obtain the closest results to the field measurements performed in the sample.
- Constitutive Equation Gap Method (CEGM): minimization of a functional of a constitutive equation gap which provides the identified values of constitutive parameters.
- Virtual Fields Method (VFM): based in the virtual work principle, which by applying well-chosen virtual fields leads to the identification of the parameters required. In the case of elastic materials it gives explicit identification of the formulae.
- Equilibrium Gap Method (EGM): based on the discretization of equilibrium equations, it is useful for obtaining heterogeneous elastic fields, as well as damage by minimizing the gap on the equilibrium.
- Reciprocity Gap Method (RGM): based on the Maxwell-Betti reciprocity theorem and adjoint fields, it works by minimizing the reciprocity gap for any adjoint field.

Different techniques exist for field measurement of properties, but one of the most common and the one that will be treated in this document is *Digital Image Correlation* (DIC) [16], in which a camera is used to measure the displacement field of a sample. Using a speckle pattern to paint the sample, the algorithm is capable of tracking the movement of the pixels, which can be translated into deformations and subsequently into strains. The method is explained in detail in Appendix A.

Finally, a last approach that can be taken is to perform simulations of microstructure plus some homogenization technique that allows us to obtain an average behavior of the material in its macrostructure. As expected, the fact that we are extending an assumption that comes from the micro-world into a more continuous setting comes with their own problems that might make more difficult the finding of the desired parameters. It is also worth noting that all these proposed methods can be combined between them in the quest of optimal parameter fitting, with the complexity and accuracy that comes along with it.

1.2 Data-based approaches

If the constitutive relation of a certain material is known, we can introduce this parameter into our system of equations and obtain a solution for our BVP. If we do not have these information, we have seen that we can fit a relationship, but we can pose the question: could we be sure that the relation provided is accurate? When working with classical materials we have many well documented databases with information that we rely on, but with newer technologies and materials appearing, data is not always available or simply unobtainable to the confidence level that we require.

1.2.1 State of the art

The exponential rise of computer power as well as storage capacities have made it easier than ever to perform analysis in big amounts of data. Techniques that allow for consolidation of information are capable of processing more, which in turn produces more reliable databases that allows us to perform better model fitting. This already works as an improvement to a classical approach, since the definition of the models becomes more accurate with more data. However, one of the bigger advantages from this improvement is the surge of the so called *data-driven methods*.

As the name implies, data-driven techniques are based purely on the direct use of data. As mentioned before, defining a material model is not easy and will unavoidable lead to biases, which will be reflected in the results obtained when these models are used. When using a data-driven approach in computational mechanics, ideally the definition of a material model is replaced by the data itself, which is directly used for the computation of displacements, strains and stresses in a given sample.

The previous statement raises the question of what exactly we can consider data. In the general sense of the word is information collected about a material through testing, as mentioned earlier, but there are nuances to be consider. When testing a material there are many factors that can be analyzed, so a careful look should be taken. Even for the same sample, applying different tests can provide different answers, so using this methods does not only consider the information itself, but also the application that we will give to the material, which will be complemented with the properly obtained dataset. The point to retain is the fact that data can take many forms, from a cloud of

unidimensional points to incomplete measurements in a sample that might not tell the full picture we are looking for. Even when avoiding the material models, data-driven methods do not provide an answer if we do not know what to look for.

One of the most important facts to consider then is how the data will be collected and interpreted. Some of the most popular techniques for collection where outlined in Section 1.1.3. However, the information obtained needs to be preprocessed before it can be effectively used, and the way this is done depends on the case analyzed. If we study elastic materials, a simple cloud of points of strain and stresses might be enough to understand the behavior of the sample, but in the case of a viscoelastic material the same cloud of points becomes an unintelligible mess, since we are not considering the effects of a complete new dimension: time. In this sense, we need to be careful with *data*: information by itself is not always enough and it needs to be complemented with knowledge of what it is being done.

Some Data-Driven methods proposed are listed here:

- Data-Driven Computational Mechanics: DDCM [17] was proposed as an alternative formulation of the Finite Elements Method for elastic materials, in where instead of considering a continuous constitutive law interpolated from the results of testing, the data itself is used to find the most optimal states for each element in the mesh. It effectively works by considering a discrete constitutive relation where the results are obtained by minimizing distances in the phase space.

- Data-Driven Identification: DDI [11] is based on DDCM, but instead it solves the inverse problem. It starts with the postulate that measurements of strains and stresses are unobtainable since stresses cannot be computed. The algorithm then solves for both the stresses in the sample as well as as a database of *material states*, which are delocalized strains and stresses that represent the behavior of the material.

- What-You-Prescribe-Is-What-You-Get: WYPIWYG [13, 14] methods work in a similar fashion as the previous methods, where the shapes of energy functions are not provided but obtained by solving equilibrium equations with experimental data.

- Neural Networks: [18] NN are a set of techniques that were designed in an attempt to recreate the way the brain makes connections. Through the use of combination of *neurons*, they are capable of describing complex phenomena by combination of paths. NN can adjust themselves to describe the subjacent relation between strains and stresses, provided they are trained with an extended and adequate dataset.

- Manifold Learning: In a similar vein, in manifold learning [12] techniques are used in big datasets to extract the underlying constitutive laws which can be presented in high dimensional spaces. The response of the material is now described by a yield surface that is extracted from the dataset, without the need of assuming certain behaviors.

From all these methods, in this work we will focus mainly on the first two. DDCM and DDI are complementing algorithms which propose an unbiased definition of the material behavior by only relying in the data itself rather than what can be extracted from them.

1.2.2 Advantages and limitations

The idea of proposing a new methodology for doing something that already works should always be presented with the advantages that it brings. As can be understood from above, the main point is to avoid making assumptions about the materials that are being used. It is expected that by using a pure data formulation to solve the problems we are not introducing any bias, since the data that it is measured is as exact as it can be (limited to the precision of the machines). In this way we avoid the *human* side of the definition of the material, which are the models that we have defined to make sense of the material behavior.

The field of data-driven computational mechanics is fairly new, so the prospects for the methodology that will be introduced here are big. In the last five years there has been an steady increase of the amount of research produced on them, and the methodology that was originally introduced for elastic materials has since been adapted to other types. Progress is also being made in complementary fields such as imaging techniques and faster computational techniques, so it is also expected that the accuracy of the methods will have room to improve.

All that has been said of course is not free of some limitations. The measurement of data is not perfect and there will always be problems such as noise and dispersion that will inevitably affect the results that data-driven methods can provide. The fact that we do not dispose of a continuous material law as in the case of the classical methods limits the solution to individual points in space, that do not necessarily represent reality, but rather the closest approximation. It is because of this that, in these early stages, data-driven methods might not be as reliable as the classical techniques, and also the reason why we need immense amount of data to trust the accuracy of the methods. However, the field is still young and improvements are definitely being made which will allows us to avoid many of these problems in the long run.

1.3 Discussion

In this introductory chapter, an overview of the constitutive relations, as well as the state of the art on how to address them, is given. Constitutive relations take many forms depending on the case and sometimes parameters are needed to characterize the link between strains and stresses. Because of this, data-driven methods have emerged as an alternative to classical approaches for constitutive laws estimations, since we have the advantage of using pure raw data without much human interaction.

In the current days, new modifications to the original DDCM algorithm are being proposed to extend the functionality in more complex cases. A variation of DDCM based on maximizing entropy [19] was proposed for noisy datasets. In other works, the use of DDCM on finite strain has been addressed in [20, 21], as well as the generation of data for its use [22]. An extension to inelastic materials has also been proposed [23], while some advances are being made in the field of fracture [24] and rate-dependent fracture [25]. In the identification front there are also many lines developing. DDI as such has already been applied with success to hyperelastic materials [26, 27]. For both elasticity and inelasticity, a manifold learning approach has been proposed [12] that waives the need of a constitutive model by proposing a manifold that represent the yield condition of a material. In *What-you-prescribe-is-what-you-get* the target is to obtain functions that define material properties through experimental data, with

success in the case of isotropic [13] and orthotropic [14] hyperelasticity. Finally, there are also some works in the vein of optimizing the algorithm. Efficiency in DDCM has been targeted in [28], while some enhancements such as tensor voting [29] have also been used successfully.

In this work we are going to focus on DDI mostly, but we want to depart from the original formulation for elastic materials and go to more complex behavior. Particularly, we put the focus on heterogeneous materials, specially in samples composed of a matrix and inclusions. Initially we have a combination of two materials that are elastic in nature, but in the general picture this is a non-elastic problem since now we have that for a particular deformed state we have two possible behaviors for the stress. This heterogeneity will be addressed and analyzed with statistical techniques. Also, in a more traditional sense, we want to address these non-elastic cases by analyzing more complex materials. We will also focus on linear viscoelastic materials, whose stress dependence in strain history rather than instantaneous strains makes it a good candidate for the use of extended formulations of the current algorithm.

As it is, the current work will be structured as follows:

- Chapter 2 will be a more detail introduction to DDCM and DDI, where the formulation will be presented and examples will be shown to address the functionality of the methods.
- Chapter 3 involves the use of DDI in heterogeneous samples and the use of a statistical technique called *Correspondence Analysis* (CA) for the separation of both behaviors, as well as the identification of the location of the inclusions in the matrix.
- Chapter 4 will show an extended formulation of DDI that will be used to address the time dependence on the stress estimation in linear viscoelastic samples. The new formulation will be presented and contrasted against the elastic formulation, to show how the modifications improve stress estimations in this case.
- Chapter 5 will see the introduction of dimension reduction techniques similar to CA, to see if we can improve the behavior and convergence of the current methods, given the computational requirements of DDI.
- Chapter 6 addresses the use of the DDI method in a real-life setting, by analyzing mixed composite samples through DIC. DDI is applied using the strain field obtained to estimate the stiffness of the material, which is contrasted to the values that can be estimated through classical means.
- Chapter 7 will act as a conclusion for the full work, where an overview of what was done is shown and where we see how does the outlook of the project looks in the future, as well as things that could be improved for better understanding of the results.

Chapter 2

Data-driven approaches in mechanics

In the previous chapter we have introduced briefly the concept of constitutive relations and the different ways in which we can estimate their value to be used in different problems. In this chapter we go more in depth about how these relations are used in different methods. Particularly, we want to focus on the Finite Elements Method, both in its traditional definition as well as the data-based approaches, to establish the trend of what will be displayed in this work.

2.1 Classical approach in Computational Mechanics

The field of computational mechanics is vast and different methodologies exist to solve problems. In this section we will focus on recalling the basic concepts for *Boundary Value Problems* (BVP), as well as their solution through the *Finite Elements Method* (FEM), which is one of the most popular techniques. FEM provides a discretization of the domain where the BVP is defined, and provides a solution typically through the use of constitutive relations. FEM is not the only method that can be used, but it is chosen due to similarities to the DDCM method that will be introduced later this chapter. The intention is to draw a parallel between both approaches, and how the solution is handled.

2.1.1 Boundary Value Problems

A Boundary Value Problem is posed as a differential equation in pair with a set of constraints that are aptly called *boundary conditions* [30]. Solutions to the problem require that the equation is solved inside of the domain while satisfying the constraints. In computational mechanics, a BVP is posed as as a combination of different physical phenomena, mainly the laws of motion and principles of kinematics.

Kinematic principles and concepts

If we have a rigid body in motion, it is subjected to both translation and deformation [7]. The variation between the reference configuration (i.e., undeformed configuration) of the body and the current one (i.e., deformed configuration) is tracked by a vector field known as a *displacement field* $\boldsymbol{u}(\boldsymbol{x}, t)$, meaning that $\boldsymbol{u}$ relates the position of a point $\boldsymbol{x}$ between both states.

Since the displacement at each point of the solid is not constant, we have that there is a rate of variation of deformation in space, which is what is known as the *deformation gradient* $\boldsymbol{\nabla u}$. This gradient is what is known as a *tensor*, which is a mathematical entity that describes a multilinear relationship between algebraic objects. In the case of kinematics, we deal with second-order tensors, which transform a vector into another one. More practically, the gradient is defined as a second-order tensor since it holds the values of all the directional derivatives of the displacement field at a particular point.

A concept that is important for our BVP is the *strain tensor*, which is an adimensional measure of the deformation of the material, and it is directly related to the displacements through what is known the *compatibility condition*. If we assume that the deformations are small enough to be infinitesimal, we can get rid of cross derivative multiplications terms and have a simplified definition of the tensor, which can be expressed in terms of the deformation gradient as

$$\begin{aligned} \boldsymbol{\epsilon} &= \frac{1}{2}\left(\boldsymbol{\nabla u} + (\boldsymbol{\nabla u})^T\right), \\ \epsilon_{ij} &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \end{aligned} \tag{2.1}$$

where the first equation is in tensor form and the second is written in index notation for a simpler understanding. The purpose of Equation 2.1 is to establish how strains are obtained, given that deformation is an external phenomenon that we can see and measure, as opposed to the strain which is internal.

Stress

Every deformation in a material is originated by forces, which in turn provoke an internal reaction inside of the material. The intensity of these reactions inside the body or on the surface is what we call *stress* [1]. The forces applied to the body can be defined as *body forces*, which are the ones that apply to the volume of the body while the forces that are acting only on the boundary of the solid are called *surface forces*. Stress ($\boldsymbol{\sigma}$) is also defined as a second order tensor in which each point is represented by a measure in each direction. If we consider each point as an infinitely small cube, each one of the components of the tensor represents the stresses in each surface, as seen in Figure 2.1. The projected stress coming out of a surface is what is known as a traction $\boldsymbol{t}$.

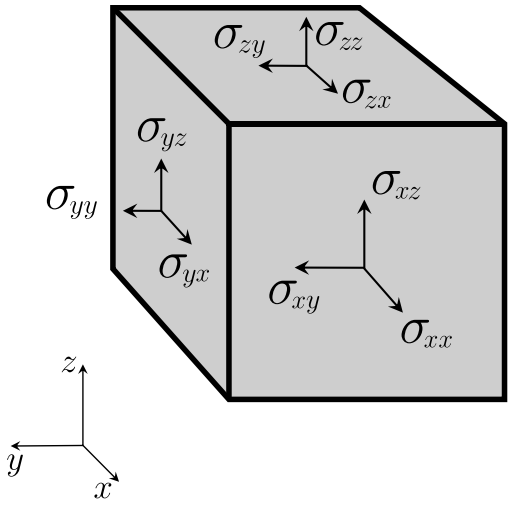


Figure 2.1: Representation of stress tensor in an infinitesimal cube. In matrix representation of the second order tensor, normal components correspond to the diagonal elements, while shear components populate the rest of the matrix.

Balance of momentum

In mechanics there are a group of physical principles that work as constraints for our BVP. They are universal for all kinds of problems, but since we are focusing only in solid mechanics, we only introduce the relevant one for our case, which corresponds to the balance of momentum. The derivation of this balance can be seen in most of the books dedicated to continuum mechanics, but in a general sense states that all bodies should respect Newton's second law of motion, which in tensor form is expressed as

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \boldsymbol{F} = \rho \ddot{\boldsymbol{u}}. \tag{2.2}$$

Equation 2.2 balances internal ($\boldsymbol{\sigma}$) and external ($\boldsymbol{F}$) forces on the left with inertial forces coming from the acceleration on the body (where ρ is the density of the material and $\ddot{\boldsymbol{u}}$ is the acceleration field or the double time derivative of the displacement field). One generalization that is common, and that will be used through this document, is to assume that the body does not move (i.e. is static), or it does in very slow settings (quasi-static), which basically allows to disregard the right side of the equation.

Full BVP problem

Having defined all the parts, the BVP is expressed fully in here. For a rigid solid body Ω with boundary $\partial\Omega$ (as seen in Figure 2.2) subjected to body forces $\boldsymbol{b}$ and surface tractions $\boldsymbol{t}$ in a quasi-static setting, the problem to solve can be posed as

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \boldsymbol{b} = 0, \tag{2.3a}$$
$$\boldsymbol{\nabla} \cdot \boldsymbol{n} - \boldsymbol{t} = 0, \tag{2.3b}$$

which is constrained by the boundary conditions $\tilde{\boldsymbol{u}}$, which are the initial prescribed displacements in the domain Ω, and $\tilde{\boldsymbol{t}}$, which are the initial prescribed forces (tractions) in the boundary $\partial\Omega$. This problem is valid for any material as long as the deformations remain small. However, since we are solving for stress, we need to find a way to relate the displacements to it. Equation 2.1 provides a relationship between displacements and strains, but we also have that stresses and strains are related to each other by equations of the form

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\epsilon}). \tag{2.4}$$

These are what we already introduced as constitutive relations, which are dependent on each material.

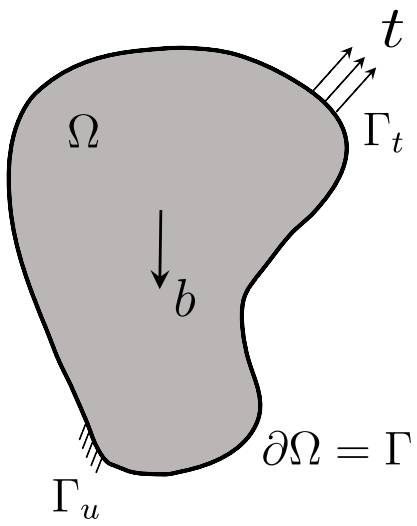


Figure 2.2: Boundary value problem for a rigid solid. Imposed boundary conditions $\tilde{\boldsymbol{u}}$ are applied to the section of the boundary Γ_u, while surface tractions are applied in Γ_t. Body forces $\boldsymbol{b}$ are distributed over the mass of the solid.

Constitutive relations

Constitutive relations vary in definition. Some might be simple equations, while others might be of a very complex form that might be impossible to describe mathematically. To illustrate the principles for these equations, we will present the Generalized Hooke's law, which is the basic constitutive equation for linear elastic materials. Mathematically, this law is written as

$$\begin{aligned} \boldsymbol{\sigma} &= \mathbb{C}\boldsymbol{\epsilon}, \\ \sigma_{ij} &= C_{ijkl}\epsilon_{kl}, \end{aligned} \tag{2.5}$$

with $\boldsymbol{\epsilon}$ being the strain and $\boldsymbol{\sigma}$ the stress, both second order tensors. In here, $\mathbb{C}$ is a fourth order tensor that relates both quantities, which for elasticity is called the stiffness tensor. Given its dimension, $\mathbb{C}$ has 81 components that need to be defined to characterize the material studied. However, this can be simplified greatly. We know

that both $\boldsymbol{\sigma}$ and $\boldsymbol{\epsilon}$ are symmetrical tensors to satisfy angular equations of motion, which can be expressed in tensor notation as

$$\begin{aligned} \sigma_{ij} &= \sigma_{ji}, \\ \epsilon_{kl} &= \epsilon_{lk}, \end{aligned} \tag{2.6}$$

which leads to the stiffness tensor to have *minor symmetries*, namely

$$\begin{aligned} C_{ijkl} &= C_{jikl}, \\ C_{ijkl} &= C_{ijlk}, \end{aligned} \tag{2.7}$$

reducing the total number of components to 36. Since the tensor also has major symmetry, i.e.,

$$C_{ijkl} = C_{klij}, \tag{2.8}$$

we can now reduce the amount of components to 21. This is the most general case for anisotropic linear elastic materials, where the constitutive law can be expressed in a matrix form:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} C_{1111} & C_{1122} & C_{1133} & C_{1123} & C_{1113} & C_{1112} \\ & C_{2222} & C_{2233} & C_{2223} & C_{2213} & C_{2212} \\ & & C_{3333} & C_{3323} & C_{3313} & C_{3312} \\ & & & C_{2323} & C_{2313} & C_{2312} \\ & \text{sym} & & & C_{1313} & C_{1312} \\ & & & & & C_{1212} \end{bmatrix} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\epsilon_{23} \\ 2\epsilon_{13} \\ 2\epsilon_{12} \end{bmatrix}. \tag{2.9}$$

Further simplifications can be done in order to keep reducing the components, from which two are shown here, since they will directly be used in later examples. The first and simplest ones is the case for trusses, where all the elements are only under axial strains and stresses, meaning that we only need to define one component. This is typically written as

$$\sigma = E\epsilon, \tag{2.10}$$

where E is most commonly known as the *Young's Modulus* of the material. The second case that we consider is the isotropic plane stress case, where we consider flat samples that are assumed to have no stress in their normal direction, meaning that Equation 2.12 can be expressed as

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{bmatrix}, \tag{2.11}$$

with $\gamma_{xy} = 2\epsilon_{xy}$ and $\epsilon_{zz} = -\frac{\nu}{E}(\sigma_{xx} + \sigma_{yy})$. In here ν is known as the *Poisson's ratio*,

which is a constant that depends on the material itself. E again represents the Young's Modulus, which is also an intrinsic parameter of the material.

Simplifying notation

Since tensor computations are not practical, we define here what is known as Voigt notation (or engineering notation), which is just a simplification of the indices of Equation 2.9, but also properly allows to perform calculations in matrix form, which can be handled by a computer. The previous matrix is rewritten as

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ & & C_{33} & C_{34} & C_{35} & C_{36} \\ & & & C_{44} & C_{45} & C_{46} \\ & \text{sym} & & & C_{55} & C_{56} \\ & & & & & C_{66} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{bmatrix}, \tag{2.12}$$

which can now be used directly. The compatibility equation can also be expressed in simple terms. Considering a 3-dimensional deformation and a 6-dimensional strain, we can rewrite Equation 2.1 as

$$\boldsymbol{\epsilon} = \mathbb{B}\boldsymbol{u}, \tag{2.13}$$

with

$$\mathbb{B}^T = \begin{bmatrix} \frac{\partial}{\partial x_1} & 0 & 0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_3} \\ 0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_3} & 0 \\ 0 & 0 & \frac{\partial}{\partial x_3} & 0 & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} \end{bmatrix}. \tag{2.14}$$

$\mathbb{B}$ acts as an operator on $\boldsymbol{u}$ similar to the gradient. In here, the crossed components of $\boldsymbol{\epsilon}$ are expressed as $\gamma_{ij} = \epsilon_{ij} + \epsilon_{ji} = 2\epsilon_{ij}$, to account for the missing 1/2 term. In a similar fashion, $\mathbb{B}$ can be used in its transposed version to perform the opposite transformation, acting as a divergence operator.

2.1.2 Finite Elements Method

FEM is a popular method used for solving differential equations. It works by dividing the domain in which the problem is posed (Ω) into smaller parts (or finite elements, Ω_e) where the problem can be solved. Its derivation is a well documented process in many sources, so a short version based on the derivation detailed in [31] is shown here. By using variational principles, a weak form of the equation of motion can be obtained in each element, expressed as

$$\sum_e \left[\int_{\Omega_e} \delta\boldsymbol{u}^T \rho \ddot{\boldsymbol{u}} d\Omega + \int_{\Omega_e} \delta(\mathbb{B}\boldsymbol{u})^T \boldsymbol{\sigma} d\Omega - \int_{\Omega_e} \delta\boldsymbol{u}^T \boldsymbol{b} d\Omega - \int_{\Gamma_{t_e}} \delta\boldsymbol{u}^T \boldsymbol{t} d\Gamma \right] = 0, \tag{2.15}$$

where the expression is summed over all elements e and boundaries et, and $\delta\boldsymbol{u}$ are the virtual displacements. The force term $\boldsymbol{F}$ is divided into two: $\boldsymbol{b}$, corresponding to body forces and $\boldsymbol{t}$, representing boundary forces. A popular approach to solve this problem is to use the Galerkin method [31], where approximations are taken for $\boldsymbol{u}$ and $\delta\boldsymbol{u}$ with the form

$$\boldsymbol{u}(\boldsymbol{x},t) \approx \hat{\boldsymbol{u}} = \sum_b N_b(\boldsymbol{x})\tilde{\boldsymbol{u}}(t). \tag{2.16}$$

The functions N_b are known as shape functions, whose choice depends on the type of element chosen to mesh the domain, as well as the accuracy expected from them. For all the examples in this document, linear functions are used. When using this approximation we limit the integration to be performed only on the shape functions, given that the nodal displacements $\tilde{\boldsymbol{u}}$ are not dependent of spatial variables. This allows to rewrite Equation 2.15 as

$$\sum_e \delta\tilde{\boldsymbol{u}}^T \left[\int_{\Omega_e} \boldsymbol{N}^T \rho \boldsymbol{N} \ddot{\tilde{\boldsymbol{u}}} d\Omega + \int_{\Omega_e} \boldsymbol{B}^T \boldsymbol{\sigma} d\Omega - \int_{\Omega_e} \boldsymbol{N}^T \boldsymbol{b} d\Omega - \int_{\Gamma_{t_e}} \boldsymbol{N}^T \boldsymbol{t} d\Gamma \right] = 0, \tag{2.17}$$

which in matrix form yields

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{P}(\boldsymbol{\sigma}) = \boldsymbol{f}, \tag{2.18}$$

with

$$\boldsymbol{M} = \sum_e \int_{\Omega_e} \boldsymbol{N}^T \rho \boldsymbol{N} d\Omega, \tag{2.19a}$$

$$\boldsymbol{P} = \sum_e \int_{\Omega_e} \boldsymbol{B}^T \boldsymbol{\sigma} d\Omega, \tag{2.19b}$$

$$\boldsymbol{f} = \sum_e \left[\int_{\Omega_e} \boldsymbol{N}^T \boldsymbol{b} d\Omega + \int_{\Gamma_{t_e}} \boldsymbol{N}^T \boldsymbol{t} d\Gamma \right]. \tag{2.19c}$$

In this notation we drop the tilde from $\boldsymbol{u}$ since from now on we will always refer to nodal displacements. We also change $\mathbb{B}$ to $\boldsymbol{B} = \mathbb{B}\boldsymbol{N}$, which is the application of the gradient operator on the shape functions rather than being directly applied in the displacement field. In Equation 2.19, matrix $\boldsymbol{M}$ correspond to the mass matrix, used mainly in problems involving dynamics; vector $\boldsymbol{f}$ represents the external forces and $\boldsymbol{P}(\boldsymbol{\sigma})$ is the stress divergence. In the case of linear elastic materials, we know that we can express $\boldsymbol{\sigma}$ as

$$\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\epsilon}, \tag{2.20}$$

where $\boldsymbol{D}$ is the generalized constitutive relation, which changes the term $\boldsymbol{P}$ to

$$\boldsymbol{P}(\boldsymbol{\sigma}) = \sum_e \int_{\Omega_e} \delta \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{\epsilon} \, d\Omega \quad = \sum_e \int_{\Omega_e} \delta \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} \boldsymbol{u} \, d\Omega = \boldsymbol{K} \boldsymbol{u}, \tag{2.21}$$

which is the already known stiffness matrix. In the case of FEM, since the domain is divided in many elements, $\boldsymbol{D}$ does not necessarily stay constant between elements.

Considering the quasi-static case, the expression in Equation 2.18 is simplified to

$$\boldsymbol{K} \boldsymbol{u} = \boldsymbol{f}, \tag{2.22}$$

which is the main equation use in FEM for linear materials.

2.2 Data-Driven Computational Mechanics

In 2016, Kirchdoerfer and Ortiz [17] introduced a new paradigm to address the problems with the way constitutive laws are handled. Their particular focus was in the bias that is introduced every time observational data is used to calibrate an empirical material model, given the imperfect knowledge of the proper functional form of the underlying material law.

The approach proposed by them relies on databases, not as a way to obtain an insight on the material behavior, but as a tool that directly provides a solution to the mechanical problem. The computation is performed using the observed data in combination with adequate constraints and conservation laws, allowing to bypass completely the need for material modelling.

For a finite elements problem, each element defined in the mesh is represented by a mechanical state, which can be considered as a point in a *phase space*, which is the space defined by the strains and stresses. The position of each mechanical state is restricted by compatibility and equilibrium constraints, as well as the material model used. Since we are replacing this model with a database of observations, DDCM can be defined as a minimization problem. This means that the algorithm aims to assign to each mechanical state a point from the database that is the closest to satisfying the constraints of the problem. This can be seen as the equivalent of reducing a distance function in the phase space, subjected to the mentioned compatibility and equilibrium conditions. While describing the algorithm, the definition is limited to use only elastic materials, which will give us a phase space composed only by the strains and stresses.

When talking about distances between two points, we refer to a norm defined in the required space. Since the phase space has different units in each of its axes, the norm has to be defined in a way that compatibilizes them. The choice taken by the authors is the use of an energetic norm, defined for a two dimensional phase space (trusses) as

$$||(\epsilon_e, \sigma_e)||_C^2 = \frac{1}{2}\left(C\epsilon_e^2 + \frac{1}{C}\sigma_e^2\right). \tag{2.23}$$

In here, C is a parameter with units $[FL^{-2}]$ that works as a weight for defining the distance, meaning that higher values of C would considered a distance defined mostly by the strain values, while lower C would shift this definition towards the stresses. C

needs to be defined before running the algorithm. This norm can be generalized for higher dimensional forms of elasticity:

$$||(\boldsymbol{\epsilon}_e, \boldsymbol{\sigma}_e)||^2_{\mathbb{C}} = \frac{1}{2}\left(\boldsymbol{\epsilon}_e : \mathbb{C} : \boldsymbol{\epsilon}_e + \boldsymbol{\sigma}_e : \mathbb{C}^{-1} : \boldsymbol{\sigma}_e\right), \tag{2.24}$$

with $\mathbb{C}$ always taking the dimensionality of the constitutive tensor associated to the phase space. It is worth mentioning that, despite the similarities, $\mathbb{C}$ does not represent an actual material behavior and its definition comes from a purely numerical standpoint.

For each element e in the mesh we have a defined mechanical state $(\boldsymbol{\epsilon}_e, \boldsymbol{\sigma}_e)$, whose distance to a material state $(\boldsymbol{\epsilon}^*_{\mathrm{ie}}, \boldsymbol{\sigma}^*_{\mathrm{ie}})$ can be expressed as

$$F_e = ||(\boldsymbol{\epsilon}_e - \boldsymbol{\epsilon}^*_{\mathrm{ie}}, \boldsymbol{\sigma}_e - \boldsymbol{\sigma}^*_{\mathrm{ie}})||^2_{\mathbb{C}}, \tag{2.25}$$

where ie is an index that refers to a point i in the material database associated to the element e. As a global problem, the algorithm seeks to minimize this distance for all the elements in the mesh, namely

$$(\boldsymbol{\epsilon}_e, \boldsymbol{\sigma}_e, \mathrm{ie}) = \arg\min_{\boldsymbol{\epsilon}_e, \boldsymbol{\sigma}_e, \mathrm{ie}} \sum_e w_e ||(\boldsymbol{\epsilon}_e - \boldsymbol{\epsilon}^*_{\mathrm{ie}}, \boldsymbol{\sigma}_e - \boldsymbol{\sigma}^*_{\mathrm{ie}})||^2_{\mathbb{C}}. \tag{2.26}$$

The global minimization is weighted by the values of w_e that correspond to the integration weights of each element. This problem is subjected to the constraints

$$\boldsymbol{\epsilon}_e = \sum_j \boldsymbol{B}_{ej} \boldsymbol{u}_j, \tag{2.27a}$$

$$\boldsymbol{f}_j = \sum_e w_e \boldsymbol{B}_{ej} \boldsymbol{\sigma}_e. \tag{2.27b}$$

Equation 2.27a represent the conditions of compatibility of strains and Equation 2.27b is the equation for mechanical equilibrium. $\boldsymbol{B}_{ej}$ is the standard connectivity matrix from FEM, relating the elements e to the nodes j of the mesh. $\boldsymbol{f}_j$ and $\boldsymbol{u}_j$ are the nodal forces and displacements, respectively.

To solve the problem, the constraints from Equation 2.27 are enforced in Equation 2.26. The compatibility constraint is applied by expressing the strains in terms of displacements. Equilibrium is applied with the use of Lagrange multipliers $(\boldsymbol{\eta}_j)$, which yields to

$$\begin{aligned}\delta \Bigg(\sum_e w_e \Bigg|\Bigg| \Bigg(\sum_j \boldsymbol{B}_{ej} \boldsymbol{u}_j - \quad & \boldsymbol{\epsilon}^*_{\mathrm{ie}}, \boldsymbol{\sigma}_e - \boldsymbol{\sigma}^*_{\mathrm{ie}} \Bigg) \Bigg|\Bigg|^2_{\mathbb{C}} \\ & - \sum_j \left(\sum_e w_e \boldsymbol{B}_{ej} \boldsymbol{\sigma}_e - \boldsymbol{f}_j \right) \boldsymbol{\eta}_j \Bigg) = 0.\end{aligned} \tag{2.28}$$

From Equation 2.28 we take the variation of the problem according to parameters that

are not fixed to obtain a system of equations to solve:

$$\delta \boldsymbol{u}_j \to \sum_e w_e \boldsymbol{B}_{ej}^T \mathbb{C} \left(\sum_k \boldsymbol{B}_{ek} \boldsymbol{u}_k - \boldsymbol{\epsilon}_{\text{ie}}^* \right) = 0, \tag{2.29a}$$

$$\delta \boldsymbol{\sigma}_e \to \mathbb{C}^{-1} (\boldsymbol{\sigma}_e - \boldsymbol{\sigma}_{\text{ie}}^*) - \sum_j \boldsymbol{B}_{ej} \boldsymbol{\eta}_j = 0, \tag{2.29b}$$

$$\delta \boldsymbol{\eta}_j \to \sum_e w_e \boldsymbol{B}_{ej}^T \boldsymbol{\sigma}_e - \boldsymbol{f}_j = 0. \tag{2.29c}$$

The linear system defined in Equation 2.29 provides solutions for nodal displacements, mechanical stresses and the corresponding Lagrange multipliers. Combining these equations, we obtain a simpler system of equations that can be solved in matrix form, i.e.,

$$\sum_j \left(\sum_e w_e \boldsymbol{B}_{ek}^T \mathbb{C} \boldsymbol{B}_{ej} \right) \boldsymbol{u}_j = \sum_e w_e \boldsymbol{B}_{ek}^T \mathbb{C} \boldsymbol{\epsilon}_e^*, \tag{2.30a}$$

$$\underbrace{\sum_j \left(\sum_e w_e \boldsymbol{B}_{ek}^T \mathbb{C} \boldsymbol{B}_{ej} \right)}_{\boldsymbol{K}} \boldsymbol{\eta}_j = \boldsymbol{f}_k - \sum_e w_e \boldsymbol{B}_{ek}^T \boldsymbol{\sigma}_e^*. \tag{2.30b}$$

From Equation 2.30 the similarities between the finite elements method and the DDCM procedure can be seen. The matrix $\boldsymbol{K}$ defined on the left side resembles the definition of the stiffness matrix for elastic materials. However, since we have no *a priori* knowledge of the material behavior, the constitutive tensor is changed by a numerical constant that we have used to define the energetic norm. The right side of the equations changes in order to account for the constraints we have set earlier without relying in the mathematical relation between strains and stresses.

The full procedure for solving the DDCM problem is outlined in Algorithm 2.1. The algorithm returns the mechanical states $(\boldsymbol{\epsilon}_e, \boldsymbol{\sigma}_e)$ that minimize the total distance to the material database, which is coded in the indices ie. As it is discussed in [11], the initialization of the algorithm plays a role in the convergence of the algorithm. Kirchdoerfer and Ortiz decided for a randomized initialization of the material states, while in [22] Platzer chose an approach based in the *k-means* algorithm, whose purpose is to find clusters of points in the space. By initializing the algorithm choosing more appropriate points there is a reduction of iteration steps.

The convergence criterion for stopping the algorithm can also be discussed. One common approach that was chosen in both [17] and [11] checks how much the material states vary in each step. If the material states $(\boldsymbol{\epsilon}_{\text{ie}}^{*(k)}, \boldsymbol{\sigma}_{\text{ie}}^{*(k)})$ are similar to $(\boldsymbol{\epsilon}_{\text{ie}}^{*(k+1)}, \boldsymbol{\sigma}_{\text{ie}}^{*(k+1)})$ up to a certain tolerance we can consider that the algorithm has already converged.

2.3 Data-Driven Identification

DDI [11] was introduced as a method that identifies the mechanical response of materials without having to rely on a constitutive model on a similar vein to DDCM, since this

Algorithm 2.1: Summarized DDCM procedure.

Output: $\boldsymbol{\epsilon}_e$, $\boldsymbol{\sigma}_e$, ie
Input: $\mathbb{C}$, $\boldsymbol{\epsilon}^*_{\text{ie}}$, $\boldsymbol{\sigma}^*_{\text{ie}}$, $\boldsymbol{f}_j$, $\boldsymbol{B}_{ej}$

DDCM procedure

Set $k \to 0$.

Initialization: for each element e, pick $(\boldsymbol{\epsilon}^{*(k=0)}_{\text{ie}}, \boldsymbol{\sigma}^{*(k=0)}_{\text{ie}})$ for first calculations.

while *convergence criterion* $<$ *tolerance* **do**

Solve

$$\sum_j \left(\sum_e w_e \boldsymbol{B}^T_{ek} \mathbb{C} \boldsymbol{B}_{ej} \right) \boldsymbol{u}^{(k)}_j = \sum_e w_e \boldsymbol{B}^T_{ek} \mathbb{C} \boldsymbol{\epsilon}^{*(k)}_e,$$

$$\sum_j \left(\sum_e w_e \boldsymbol{B}^T_{ek} \mathbb{C} \boldsymbol{B}_{ej} \right) \boldsymbol{\eta}^{(k)}_j = \boldsymbol{f}_k - \sum_e w_e \boldsymbol{B}^T_{ek} \boldsymbol{\sigma}^{*(k)}_e.$$

to obtain $\boldsymbol{u}^{(k)}_j$ and $\boldsymbol{\eta}^{(k)}_j$.

for *each element* e **do**

Solve

$$\boldsymbol{\epsilon}^{(k)}_e = \sum_j \boldsymbol{B}_{ej} \boldsymbol{u}^{(k)}_j$$

$$\mathbb{C}^{-1}(\boldsymbol{\sigma}^{(k)}_e - \boldsymbol{\sigma}^{*(k)}_{\text{ie}}) - \sum_j \boldsymbol{B}_{ej} \boldsymbol{\eta}^{(k)}_j = 0,$$

to update $(\boldsymbol{\epsilon}^{(k)}_e, \boldsymbol{\sigma}^{(k)}_e)$.

for *each mechanical state* $(\boldsymbol{\epsilon}^{(k)}_e, \boldsymbol{\sigma}^{(k)}_e)$ **do**

Find new closest material point $(\boldsymbol{\epsilon}^{*(k+1)}_{\text{ie}}, \boldsymbol{\sigma}^{*(k+1)}_{\text{ie}})$.

Set $k \to k+1$.

method is derived from it. Assuming that displacement and strain fields are available, DDI solves an inverse problem that identifies also the stress fields and the material database. Given the displacement field, measured for instance by DIC, a collection of strains can be obtained from analyzing multiple *snapshots* of a sample subjected to different loading conditions. DDI provides estimations to the stresses associated to these deformations, which we also call here *mechanical stresses*. The algorithm aims to find the most suitable answer in the phase space, which is the one formed by the strain and stresses. This is achieved by adjusting *material states*. The concept in DDI is similar to DDCM, but not the same. In DDI, material states are strain-stress couples that characterize the behavior of the material, but they are not known initially. The numerical advantage of material states is that they are not constrained, so they can be set in a way that allows to simultaneously find balanced mechanical stress fields. The mechanical stresses are as close as possible to each one of these material points, with the distance defined by an energetic norm, which in here we take equivalently as the one defined in Equation 2.24. The phase space definition in DDI is the same as DDCM.

In practice, the method is also defined as a minimization problem. The biggest difference from DDCM is that we know from the beginning the displacement fields (or mechanical strains in DDCM) of the samples we are analyzing, which is the main input for our problem. We consider a large database of measurements performed in samples of the same material. Each sample is tested in different loading conditions, which we reference by the index X. In theory, each snapshot X can be a different sample, so we are not limited to test one particular piece, as long as they have the same behavior. However, in reality it is more practical, both from a testing and algorithmic point of view, to keep the samples constant during the process.

When the samples are analyzed, for each snapshot X we have the following properties known:

- applied nodal forces ($\boldsymbol{f}_j^X$), which typically are known from the test performed in each sample;
- nodal displacements ($\boldsymbol{u}_j^X$), obtained through imaging techniques;
- and geometry and connectivity of the mesh ($\boldsymbol{B}_{ej}^X$), that in combination with the nodal displacements allows us to define the mechanical strains as $\boldsymbol{\epsilon}_e^X = \sum_j \boldsymbol{B}_{ej}^X \boldsymbol{u}_j^X$.

The geometry of the sample and therefore the connectivity of the mesh are arbitrary definitions that we can control, which is advantageous since we can track at any moment the location of the mechanical states according to their numeration. For the nodal forces, even if we know the forces used for the test, we cannot know how they are applied in the border nodes. A practical approach used by the authors and reproduced here is to consider a *clamping force* on the affected border, which is the force known from testing. The force is applied in a group of nodes which is introduced in the system as a constraint. In this way, we avoid defining arbitrary nodal forces and we focus in a global approach. Same as the DDCM algorithm, there are parameters that we need to set for the algorithm. $\mathbb{C}$ is again defined as a tensor that defines the energetic norm for measuring distances. We also include the parameter N^*, which is the amount of material states points that will sample the material behavior.

DDI seeks to compute N^* material states $(\boldsymbol{\epsilon}_i^*, \boldsymbol{\sigma}_i^*)$, common to all mechanical states, thus independent from X. These points are defined such that they allow all snapshots X to obtain balanced mechanical stresses, $\boldsymbol{\sigma}_e^X$. For each one of the mechanical states $(\boldsymbol{\epsilon}_e^X, \boldsymbol{\sigma}_e^X)$, the closest material state $(\boldsymbol{\epsilon}_{\mathrm{i}e^X}^*, \boldsymbol{\sigma}_{\mathrm{i}e^X}^*)$ is assigned. $\mathrm{i}e^X$ correspond to indices that establish the relationship between material and mechanical states. We call these indices *state mapping*, and it can be understood as a function $\mathrm{i}(e, X)$, where for each element e in a snapshot X we obtain the material point i. The importance of the state mapping will be seen later, where it will also be explained in more detail.

This problem can be represented mathematically in a similar fashion to DDCM as

$$\left(\boldsymbol{\sigma}_e^X, \boldsymbol{\epsilon}_i^*, \boldsymbol{\sigma}_i^*, \mathrm{i}e^X\right) = \arg \min_{\boldsymbol{\sigma}_e^X, \boldsymbol{\epsilon}_i^*, \boldsymbol{\sigma}_i^*, \mathrm{i}e^X} \sum_X \sum_e w_e^X ||(\boldsymbol{\epsilon}_e^X - \boldsymbol{\epsilon}_{\mathrm{i}e^X}^*, \boldsymbol{\sigma}_e^X - \boldsymbol{\sigma}_{\mathrm{i}e^X}^*)||_{\mathbb{C}}^2, \quad (2.31)$$

constrained by

$$\sum_e w_e^X \boldsymbol{B}_{ej}^X \boldsymbol{\sigma}_e^X = \boldsymbol{f}_j^X \quad \forall X, j. \tag{2.32}$$

In both Equation 2.31 and 2.32 w_e^X are the integration weights for each element in each snapshot. Equation 2.32 represents the equilibrium constraint that we have also seen in DDCM. Since $\boldsymbol{\epsilon}_e^X$ are not an unknown anymore we do not have to apply the compatibility requirements and we use the mechanical strains directly in our calculations.

To solve the problem, the approach followed is similar to the one seen in Section 2.2. Enforcing Equation 2.32 by the use of Lagrange multipliers, we study perform the minimization by studying the variation of the system with respect the non fixed variables, i.e.

$$\begin{aligned}\delta \Bigg(\sum_X \sum_e w_e^X \left|\left| (\boldsymbol{\epsilon}_e^X - \quad \boldsymbol{\epsilon}_{\mathrm{ie}^X}^*, \boldsymbol{\sigma}_e^X - \boldsymbol{\sigma}_{\mathrm{ie}^X}^*) \right|\right|_{\mathbb{C}}^2 \\ - \sum_j \left(\sum_e w_e^X \boldsymbol{B}_{ej}^X \boldsymbol{\sigma}_e^X - \boldsymbol{f}_j^X \right) \boldsymbol{\eta}_j^X \Bigg) = 0.\end{aligned} \tag{2.33}$$

The variations of Equation 2.28 give the system of equations to solve:

$$\delta \boldsymbol{\epsilon}_i^* \rightarrow \sum_X \sum_{e|\mathrm{ie}^X = i} w_e^X \mathbb{C} : (\boldsymbol{\epsilon}_e^X - \boldsymbol{\epsilon}_{\mathrm{ie}^X}^*) = 0 \quad \forall i; \tag{2.34a}$$

$$\delta \boldsymbol{\sigma}_i^* \rightarrow \sum_X \sum_{e|\mathrm{ie}^X = i} w_e^X \mathbb{C}^{-1} : (\boldsymbol{\sigma}_e^X - \boldsymbol{\sigma}_{\mathrm{ie}^X}^*) = 0 \quad \forall i; \tag{2.34b}$$

$$\delta \boldsymbol{\sigma}_e^X \rightarrow w_e^X \mathbb{C}^{-1} : (\boldsymbol{\sigma}_e^X - \boldsymbol{\sigma}_{\mathrm{ie}^X}^*) - \sum_j w_e^X \boldsymbol{B}_{ej}^X \boldsymbol{\eta}_j^X = 0 \quad \forall e, X; \tag{2.34c}$$

$$\delta \boldsymbol{\eta}_j^X \rightarrow \sum_e w_e^X \boldsymbol{B}_{ej}^{X^T} \boldsymbol{\sigma}_e^X - \boldsymbol{f}_j^X = 0 \quad \forall j, X. \tag{2.34d}$$

In these equation, the notation $\sum_{e|\mathrm{ie}^X=i}$ refers to a sum of all elements e for each snapshot X that satisfy the condition $\mathrm{ie}^X = i$. In simpler words, it means that the sum is performed in all the mechanical states associated to material point i. Combining Equation 2.34b to Equation 2.34d we obtain the system defined by

$$\underbrace{\sum_k \sum_e w_e^X \boldsymbol{B}_{ej}^{X^T} : \mathbb{C} : \boldsymbol{B}_{ek}^X \boldsymbol{\eta}_k^X}_{\boldsymbol{K}^X} + \underbrace{\sum_e w_e^X \boldsymbol{B}_{ej}^{X^T} \boldsymbol{\sigma}_{\mathrm{ie}^X}^*}_{\boldsymbol{S}^X} = \boldsymbol{f}_j^X \quad \forall j, X; \tag{2.35a}$$

$$\sum_{e|\mathrm{ie}^X=i} \sum_X \sum_j w_e^X \boldsymbol{B}_{ej}^X \boldsymbol{\eta}_j^X = 0 \quad \forall i. \tag{2.35b}$$

The meaning of these equations is straightforward. For each loading case X we have that the imbalance originated from applying the forces $\boldsymbol{f}_j^X$ and stresses $\boldsymbol{\sigma}^*_{\mathrm{i}e^X}$ is balanced by virtual displacements η, in a similar way as it would be done in FEM. From the second equation we have that the weighted average of strains coming from virtual displacements is zero. In matrix form, the system can be expressed as

$$\begin{bmatrix} \boldsymbol{K}^1 & & & & \boldsymbol{S}^1 \\ & \boldsymbol{K}^2 & & & \boldsymbol{S}^2 \\ & & \ddots & & \vdots \\ & & & \boldsymbol{K}^{N^X} & \boldsymbol{S}^{N^X} \\ \boldsymbol{S}^1 & \boldsymbol{S}^2 & \dots & \boldsymbol{S}^{N^X} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}^1 \\ \boldsymbol{\eta}^2 \\ \vdots \\ \boldsymbol{\eta}^{N^X} \\ \boldsymbol{\sigma}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^1 \\ \boldsymbol{f}^2 \\ \vdots \\ \boldsymbol{f}^{N^X} \\ \boldsymbol{0} \end{bmatrix}, \tag{2.36}$$

where the outputs of the algorithm are:

- the mechanical stresses ($\boldsymbol{\sigma}_e^X$), which are the local stresses related to the measured strains, i.e., the estimation of the real stresses in each of the snapshots;
- the material states ($\boldsymbol{\epsilon}_i^*, \boldsymbol{\sigma}_i^*$), that sample the material behavior acting as the centroid of clusters of mechanical states;
- and the state mapping ($\mathrm{i}e^X$). In here, $(\boldsymbol{\epsilon}^*, \boldsymbol{\sigma}^*)_{\mathrm{i}e^X}$ are the material states in $(\boldsymbol{\epsilon}_i^*, \boldsymbol{\sigma}_i^*)$ closest to a mechanical state $(\boldsymbol{\epsilon}_e^X, \boldsymbol{\sigma}_e^X)$, where $\boldsymbol{\epsilon}_e^X$ is measured and $\boldsymbol{\sigma}_e^X$ is estimated. $\mathrm{i}e^X$ can be interpreted as a binary adjacency matrix of a bipartite graph between mechanical states and material states. In this interpretation of $\mathrm{i}e^X$ as a matrix, rows represent the mechanical states while columns represent the material states. In mathematical form, $\mathrm{i}e^X$ can be interpreted as a function $\mathrm{i}e^X = \mathrm{i}(e, X)$ defined as

$$\mathrm{i}e^X = \arg\min_{\mathrm{i}} ||(\boldsymbol{\epsilon}_e^X - \boldsymbol{\epsilon}_{\mathrm{i}}^*, \boldsymbol{\sigma}_e^X - \boldsymbol{\sigma}_{\mathrm{i}}^*)||_{\mathbb{C}}^2. \tag{2.37}$$

Solving the algorithm with Equation 2.36 can get expensive really fast due to the size of the involved matrices. Because of this, in practice we solve the DDI problem using an alternated minimization scheme, where we update iteratively the values of $\boldsymbol{\sigma}_e^X$ and $\boldsymbol{\sigma}^*_{\mathrm{i}e^X}$ until we reach convergence to a desired tolerance. This procedure for solving DDI is detailed in Algorithm 2.2. In here the same considerations about initialization and convergence are carried from the DDCM algorithm. In all of the examples that will be shown in this document, we consider a k-means initialization of the state mapping, as well as a convergence criterion that compares the variation of the material states from step to step.

Finally, a small discussion need to be made about the parameter $\mathbb{C}$. In many cases during this work, a value of $\mathbb{C} = \infty$ is considered. This is due to what has been observed in [26], that shows that increasing the value of $\mathbb{C}$ effectively increases the accuracy of the stress estimations. To consider a high value of this parameter means that we are effectively removing the stress from the norm defined in Equation 2.24, so the pairing performed in the algorithm now is exclusively dependent on the strains. In the algorithm this is applied by restricting the iterations to just one step, meaning that we perform an adjustment on the valued for the material stresses, but the material strain

Algorithm 2.2: Summarized description of DDI.

Output: $\boldsymbol{\epsilon}_{\mathrm{i}}^{*}$, $\boldsymbol{\sigma}_{\mathrm{i}}^{*}$, $\boldsymbol{\sigma}_{e}^{X}$, ie^{X}
Input: N^{*}, $\mathbb{C}$, $\boldsymbol{u}_{j}^{X}$, $\boldsymbol{f}_{j}^{X}$, $\boldsymbol{B}_{ej}^{X}$, $\boldsymbol{\epsilon}_{e}^{X}$

DDI procedure

Set $k \to 0$.

Initialization: set $\boldsymbol{\sigma}_{e}^{X(k=0)} = 0$ for each element e in every snapshot X. Likewise, initialize $\boldsymbol{\sigma}_{\mathrm{i}}^{*(k=0)} = 0$ for each material state i.

Initialization: create state mappings ie^{X} by clustering all $(\boldsymbol{\epsilon}_{e}^{X(0)}, \boldsymbol{\sigma}_{e}^{X(0)})$ into N^{*} groups.

while *convergence criterion for* $\boldsymbol{\epsilon}_{i}^{*} < tolerance$ **do**

while *convergence criterion for* $\boldsymbol{\sigma}_{i}^{*} < tolerance$ **do**

for $X = 1, \ldots, N^{X}$ **do**

Solve equations

$$\sum_{k}\sum_{e} w_{e}^{X}\boldsymbol{B}_{ej}^{X^{T}} : \mathbb{C} : \boldsymbol{B}_{ek}^{X} \cdot \boldsymbol{\eta}_{k}^{X(k)} - \sum_{e} w_{e}^{X}\boldsymbol{B}_{ej}^{X^{T}}\boldsymbol{\sigma}_{\mathrm{ie}^{X}}^{*(k)} = \boldsymbol{f}_{j}^{X} \quad \forall j, X;$$

$$\boldsymbol{\sigma}_{e}^{X(k+1)} = \boldsymbol{\sigma}_{\mathrm{ie}^{X}}^{*(k)} + \sum_{j} \mathbb{C} : \boldsymbol{B}_{ej}^{X} \cdot \boldsymbol{\eta}_{j}^{X(k)} \quad \forall e, X.$$

with current values of $\boldsymbol{\sigma}_{\mathrm{i}}^{*(k)}$ to obtain updated values of $\boldsymbol{\sigma}_{e}^{X(k+1)}$.

Solve

$$\sum_{X}\sum_{e|\mathrm{ie}^{X}=i} w_{e}^{X}\mathbb{C} : (\boldsymbol{\sigma}_{e}^{X(k+1)} - \boldsymbol{\sigma}_{\mathrm{ie}^{X}}^{*(k+1)}) = 0 \quad \forall i$$

with obtained $\boldsymbol{\sigma}_{e}^{X(k+1)}$ to update $\boldsymbol{\sigma}_{i}^{*(k+1)}$.

for $X = 1, \ldots, N^{X}$ **do**

Solve

$$\sum_{X}\sum_{e|\mathrm{ie}^{X}=i} w_{e}^{X}\mathbb{C} : (\boldsymbol{\epsilon}_{e}^{X} - \boldsymbol{\epsilon}_{\mathrm{ie}^{X}}^{*(k+1)}) = 0 \quad \forall i$$

to update $\boldsymbol{\epsilon}_{i}^{*(k+1)}$.

Update state mappings ie^{X} using Equation 2.37.

Set $k \to k + 1$.

remain the same as the initial pairing. It is because of this that a good initialization is needed. In [11] it has been seen that a proper initialization of the problem leads to faster solutions, however, if we decide to run the case of $\mathbb{C} = \infty$, this initial pairing is essential also for the accuracy of the solution. It is because of this that we rely on unsupervised clustering techniques to perform this step. During this work we rely particularly in the k-means method [32], which is explained in detail in Appendix B.

As a way to show the similarities between both methods, Table 2.1 shows side by side which variables of are known before, and which ones are estimated by the algorithm.

Table 2.1: Comparison between DDI and DDCM variables.

Variable		DDCM	DDI
Mechanical strain	$\boldsymbol{\epsilon}_e$	unknown	known
Mechanical stress	$\boldsymbol{\sigma}_e$	unknown	unknown
Material strain	$\boldsymbol{\epsilon}_i^*$	known	unknown
Material stress	$\boldsymbol{\sigma}_i^*$	known	unknown
Geometry	$\boldsymbol{B}_{ej}$	known	known
Nodal forces	$\boldsymbol{f})j$	known	known
Pseudo-stiffness	$\mathbb{C}$	chosen	chosen
Amount of material points	N^*	given	chosen

2.4 Examples

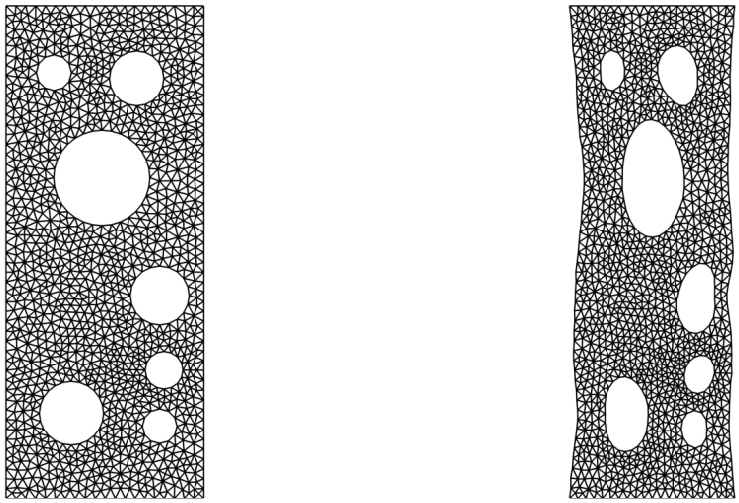
As a way to demonstrate the use of both DDI and DDCM, the following section will be dedicated to show some examples of how they work and how they complement each other. First, we simulate testing performed in lab samples made of trusses to obtain the behavior using DDI, to then solve a different problem using DDCM with DDI as a database for it, which we compare with FEM results to test the accuracy of the method. The same is done after with samples defined in the plane stress setting, to have an overview of how the methods behave in a non-scalar case.

2.4.1 Examples for trusses

DDI has been proposed as an alternative to find the constitutive equations of materials through testing. To keep the problems more realistic, we perform this synthetic examples to recreate the experiences on an experimental setting.

DDI in trusses

For the experiment, a test in a membrane is proposed. The membrane, as seen in Figure 2.3a, is composed of one material with holes in it to avoid having homogeneity in the stress fields. The simulation is performed by clamping both upper and lower ends of the sample. The lower end is fixed, while the upper end is allowed to move as a block. In this setting, a vertical stretch is applied in 3 steps. The deformation is shown from Figure 2.3b to 2.9b.



(a) Undeformed mesh of the membrane. **(b)** Traction deformation for the membrane.

Figure 2.3: Mesh and deformed state for the DDI example.

The mesh is defined as a truss structure, where every bar in the mesh is defined as a nonlinear elastic bar with a constitutive equation of the form

$$\sigma = K_1\epsilon + K_3\epsilon^3, \tag{2.38}$$

where the parameters are set to $K_1 = 1$ and $K_3 = 7.5$.

As it was mention in Section 2.3, for running the algorithm we need the database of strains, as well as the conditions that were applied when they were obtained. Normally, this would be done using the DIC technique, but since this is a numerically generated example, the values of strains are obtained by solving a FEM problem in each step of deformation. This gives us also the advantage of providing a set of stresses, which DDI can only estimate, allowing us to compare both results and have a proper grasp of the accuracy of the predictions.

As a final step, it is necessary to define the values for the parameters of the algorithm. Since our deformation is only applied in three steps, the amount of mechanical states is low, so we choose a lower ratio of mechanical states to material points, to have a bigger database. In this case, we set that we will have in average 20 mechanical states per material point. For the case of the parameter C, it has been seen that increasing its value leads to lower errors, a trend that is kept as the number increases. For this reason, we decide to set $C = \infty$, which in practice means that we will let the algorithm run for only one iteration with $C = 1$.

The results given by DDI can be seen in Figure 2.4. The algorithm estimates the values of stresses for the given strains by setting a material point in the phase space. The material states, that are used as guides for minimizing the distance between points are placed on the center of the clusters of points. Since for homogeneous trusses the algorithm works well, a zoomed window is provided in order to see how the material states behave as the centroid of the different clusters, which are aligned with the underlying constitutive curve used to generate the data.

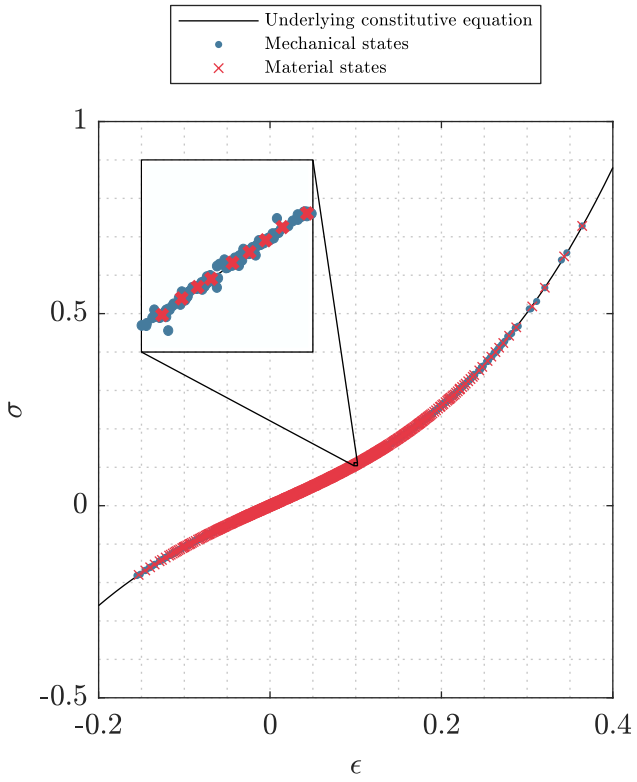
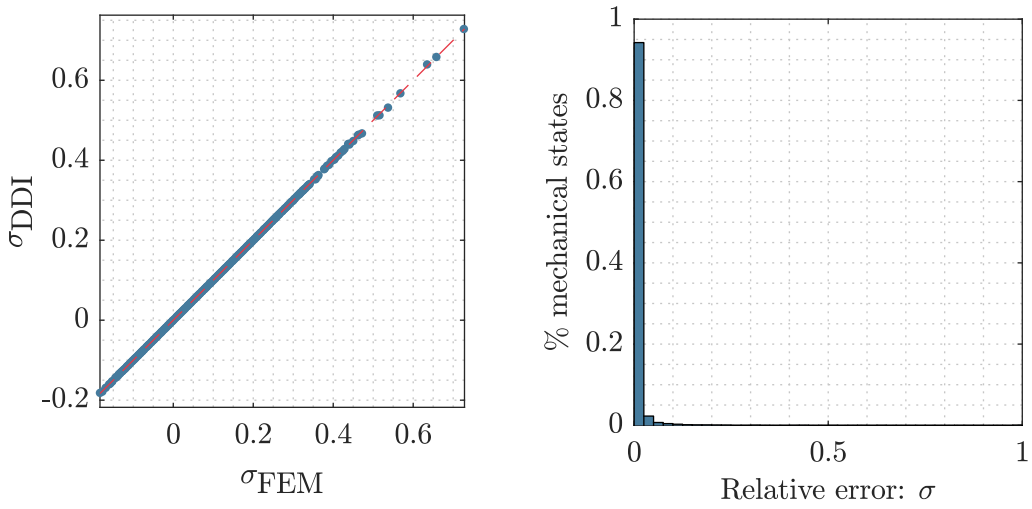


Figure 2.4: Phase space for the DDI truss example.

In general the results look good, but it is necessary to quantify the error with respect to the FEM solution. One way of doing this is computing the overall relative error between DDI and FEM, which is computed with the formula

$$e_2 = \frac{||\boldsymbol{\sigma}_{FEM} - \boldsymbol{\sigma}_{DDI}||_2}{||\boldsymbol{\sigma}_{FEM}||_2}, \tag{2.39}$$

which computes a distance between all the points obtained by DDI and FEM, relative to the real solution. In the case of this example, the overall error is $e_2 = 0.0026$, meaning that there is a 0.26% difference between the real solution of the problem and the estimated one. This can be appreciated better when both stresses are plotted against each other, as done in Figure 2.5a. In this plot, every point who stays close to the diagonal is accurately represented, while the dispersion indicates error. As can be seen, the points are mostly aligned to the diagonal because of the accuracy of the estimations. Finally, another way of visualizing the error is with the use of the histogram seen in Figure 2.5b. In this graphic, the x-axis represents the error range for all mechanical states, while the y-axis represents the percentage of points that are in a particular error range. In this sense, if we have a histogram that is shifting to the left it means that we have a good estimation of the values of stresses. In the case of Figure 2.5b we see that close to 95% of all the mechanical stresses are estimated to a relative error of 0.02 or less, which reaffirms our good results.



(a) Diagonal error plot.

(b) Histogram error plot.

Figure 2.5: Different error representations for the DDI truss example.

All this three displays of the error have their own advantages. Computing e_2 allows us to have a quick sense of how the overall system is behaving, which in turn allows us to optimize the algorithm a bit better or to catch any kind of general error during the calculation. Same for the diagonal plot, it gives a general sense of the results but also allows us to see in which part of the phase space our problems are. These plots tend to drift away from the main diagonal in the extreme values of stress, which happens typically due to the fact that there are less mechanical points due to the over-representation of the low stress cases. Finally, the histogram gives us an idea of how is the distribution of errors in our calculations. If we have a flat curve we can expect that some tweaking of the algorithm can improve our results, however, if there is a shift to the right we can assume that something is wrong in the definition of the problem. Histograms are more useful when comparing different instances of DDI, since they give a very good visual representation of the improvement.

Example of DDCM for a truss

The example shown for DDI is purely academic in nature, since samples are tested to obtain results but no problem is being solved, we focus solely on obtaining information about the material. In the following example, we use this information obtained to solve a problem that is closer to a real case.

To show the capabilities of DDCM in trusses, we propose a structure as seen in Figure 2.6. The frame has completely restricted movement on the base nodes to simulate an anchoring point, while the top of the structure is submitted to a uniform downwards force, which could represent any kind of ceiling structure. We also added in one of the side a lateral load on the top as a simplification of possible movements caused by wind or earthquakes.

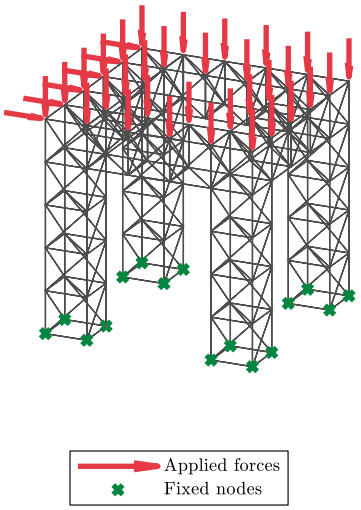


Figure 2.6: Structure to be solved with DDCM, including loads and boundary conditions.

The truss elements have one integration point and the material is not defined, since it is provided by the DDI example studied previously. The procedure to follow outlined in Algorithm 2.1 is similar to FEM, since the loads and boundary conditions are defined in the same way, while the stiffness matrix is now built using the values of C. As opposed to the case of DDI, we cannot use higher values of C expecting better results, since DDCM is very sensitive to the choice of this parameter. For this case, we have chosen a value of $C = 2$ which has proven to give the most accurate results.

The solution of the example obtained with DDCM is shown in Figure 2.7. In here, the deformed structure is imposed over the original to show the displacement, which is in accordance to the application of the forces and restrictions.

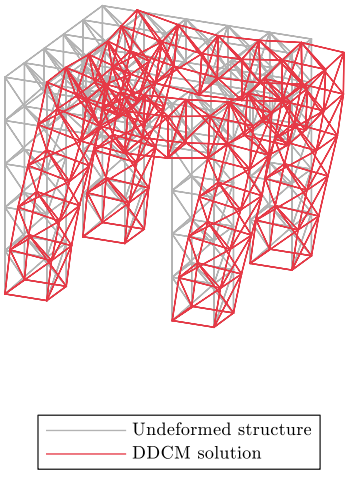
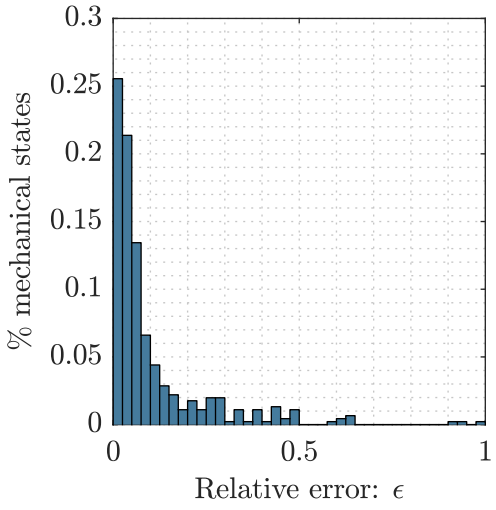


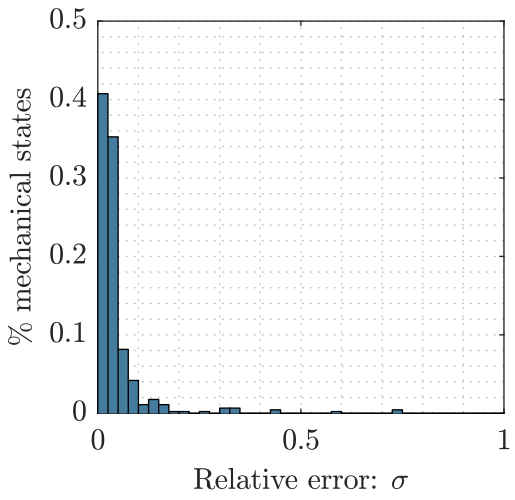
Figure 2.7: Solution for the truss example with DDCM.

To measure the accuracy of the predictions, we follow the same procedure as in the examples for DDI. For this, we now assume that the bars have the same property as the sample from the previous section (defined in Equation 2.38) and we obtain an FEM solution of the problem, to then compare the displacement of each node, as

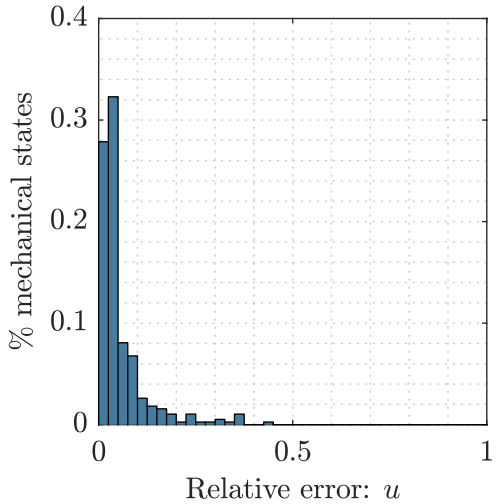
well as the strains and stresses of each bar. Using Equation 3.4, we obtain that the overall error of the displacement in the solution is $e_2^u = 0.0261$, while for the strains is $e_2^\epsilon = 0.0396$ and the stresses is $e_2^\sigma = 0.0511$. In general, predictions are very good in this case. Any divergence with the theoretical solution can be explained by the fact that we are allowing the system only a limited amount of states to be, rather than having a continuous behavior for the phase space. This is one of the reasons why for this method it is very necessary to provide a high amount of data points, since a more complete database makes up for the discontinuities. The reason why displacement has the lowest error is due to the fact that it is a direct calculation from ϵ_e^*, while the strain ϵ_e is computed using these displacements, propagating the error. The same can be said about stresses, which depend on both σ_e^* and η. The histograms with the distribution of the error are shown in Figure 2.8. As can be seen, for both strains and displacements around 50% of all the predictions have a relative error lower than 0.05, while for the stresses this is around 70%.



(a) Histogram error plot for strains.



(b) Histogram error plot for stresses.



(c) Histogram error plot for displacements.

Figure 2.8: Error histograms for the truss case in DDCM.

2.4.2 Examples for plane stress

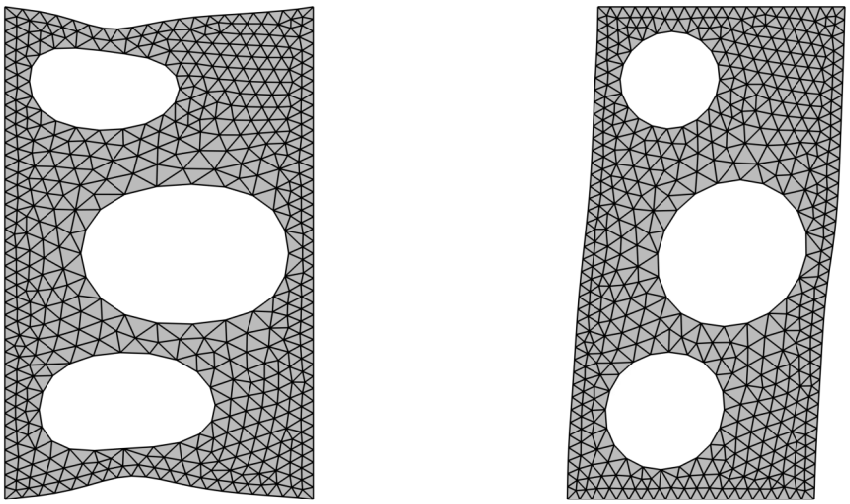
Both examples shown in Section 2.4.1 show that for the scalar case the pairing of DDI and DDCM work fine for solving problems in a different way. However, the case for a high-dimension setting such as plane stress can become a bit more problematic, since the database obtained by DDI needs to be in similar ranges as the problem that will be solved in DDCM, which is a more challenging task in a six-dimensional case.

DDI in plane stress

An example of DDI run in plane stress is also shown here to demonstrate the adaptability of the algorithm. The process is done in a similar fashion to the truss example, choosing the same values for the parameters N^* and $\mathbb{C}$. In the case of $\mathbb{C} = \infty$, since we are now dealing with tensors rather than scalar values, we choose $\mathbb{C}$ as the identity matrix, while limiting the run of the algorithm to just one iteration.

In plane stress, we consider triangles elements, all of them with the same material properties. We consider linear elastic elements, with the elasticity tensor defined as Equation 2.11, with a Young's modulus of $E = 5$ and a Poisson ratio of $\nu = 0.3$. In all these cases the units for all properties are left undefined for simplicity, but they are all chosen to be compatible.

To make the database more diverse, we simulate different tests in the sample. Since we are considering linear elements, we can expand easily the amount of snapshots, so we consider now 20 steps of deformation for the stretching. We also include a compression simulation in the sample, as well as shear, to avoid problems with the other components of the strain and stress tensor. These deformations are shown in Figure 2.9.



(a) Traction deformation for the membrane.

(b) Shear deformation for the membrane.

Figure 2.9: Mesh and deformation of the membrane for DDI in plane stress.

The results obtained for the phase space can be seen in Figure 2.10. The accuracy of DDI is generally lower for higher dimensional problems, so we can see here the dispersion of the values around the underlying constitutive equation. This dispersion is more pronounced in the shear component of the stress, where it is typically more difficult to obtain a proper estimation.

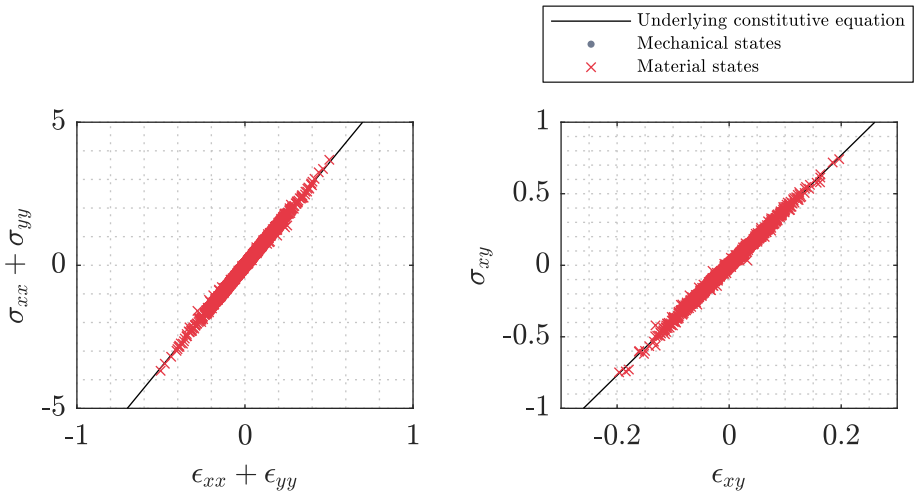
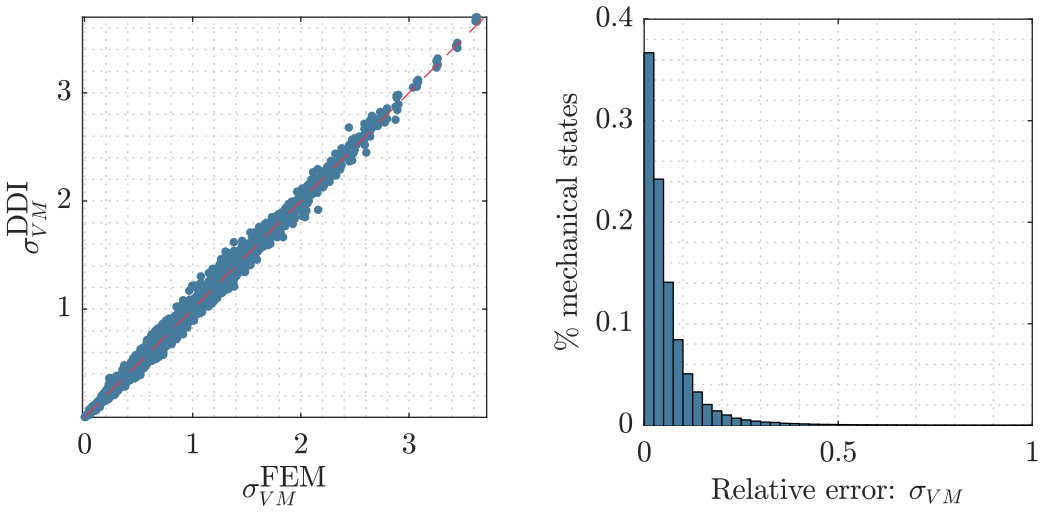


Figure 2.10: Phase space for the DDI example in plane stress.

To estimate the error we use the same procedure as before, however, in order to consider all the components of the stress tensor, we compute all the errors based on the Von Mises stress, which is defined as

$$\sigma_{VM} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 + 3\sigma_{xy}^2 - \sigma_{xx}\sigma_{yy}}. \tag{2.40}$$

The diagonal plot and the histogram for errors is shown in Figure 2.11. As can be seen, there is some dispersion around the diagonal line, but in general we have a tendency of points staying close to it. One disadvantage of using the Von Mises stress is the fact that we do not have negative values, so we cannot find exactly the part of the phase space in which we have a bigger error. For the histogram we can see that we have a good tendency of the curve to the left, but compared to the truss case the curve is much lower. This time only around 36% of all the mechanical states have an error of 0.02 or lower, which is a typical amount for plane stress cases.



(a) Diagonal error plot.

(b) Histogram error plot.

Figure 2.11: Error representations for DDI in the plane stress example.

Using Equation 2.39 for the Von Mises stress we obtain an error $e_2^{\sigma,VM} = 0.0361$, which is low. Even though we might have some points misrepresented mostly on the higher stresses, we still have a good overall estimation of the system.

Example of DDCM for plane stress

Providing an example for plane stress is just an extension of what has been done for the trusses. However, to prove the universality of the algorithm, for this example different elements will be used. The DDI samples provided for plane stress in the previous section were made out of linear elastic triangular elements with one integration point. For the DDCM example, quadrilateral elements are used. Since DDI provides values for stresses and strains, it should not be an issue for the algorithm to change the shape or order of the elements used. Same as in the truss case, we avoid defining material properties for the elements.

For the plane stress case, a mesh representing a beam is used, as seen in Figure 2.12. The rectangular beam is completely restricted of movement on the nodes on the left, while downward displacements are applied in the nodes of the right boundary, which is restrained in the horizontal direction. The solution obtained by the algorithm is shown in Figure 2.13, where we can see that the displacement of the nodes is according to the conditions applied to the system.

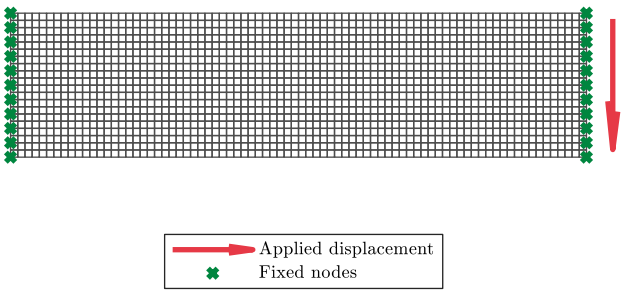


Figure 2.12: Boundary conditions and applied loads for DDCM example in plane stress.

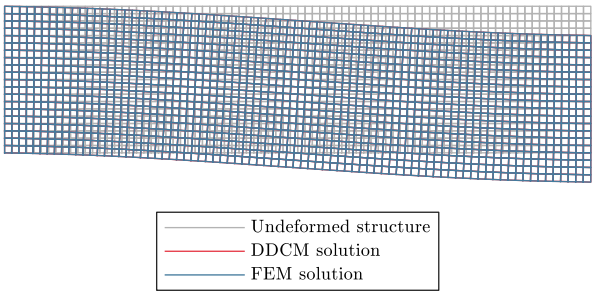
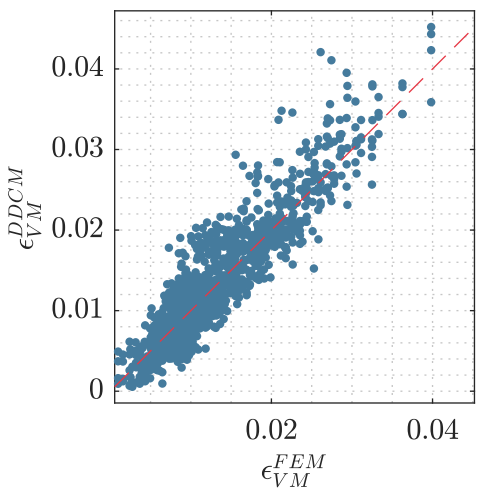
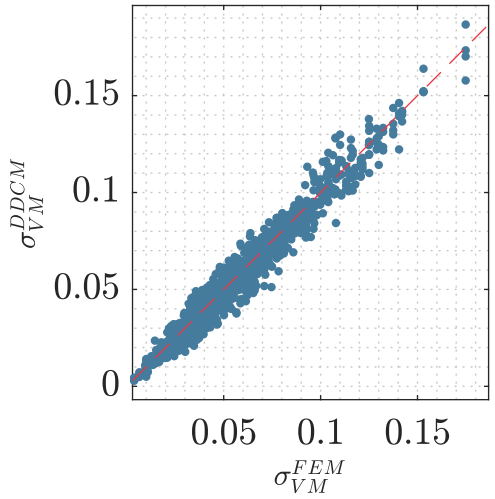


Figure 2.13: Solution for the DDCM example in plane stress.

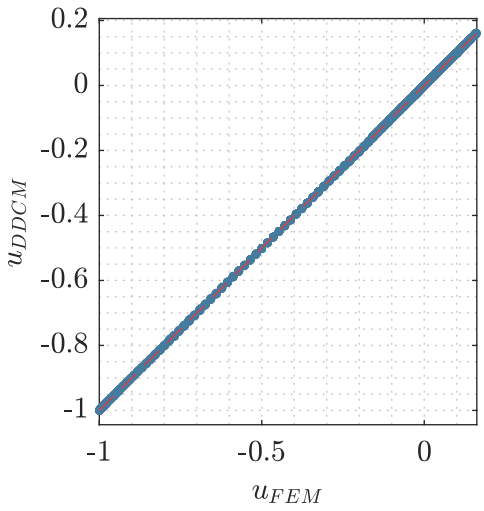
We can provide a similar error analysis to the truss case. Same as in the DDI, instead of computing errors for each component of the strain and stress tensors, we compare the Von Mises values to combine all the behaviors. If we compute the 2-norm relative error we obtain for the displacements a value of $e_2^u = 0.0033$, while for the strains a value of $e_2^{\epsilon,VM} = 0.1993$ is obtained. In the case of the stresses, the total estimated relative error is $e_2^{\sigma,VM} = 0.1228$. We can see that the computed displacement of the mesh is very close to the values obtained with FEM. In this example the stress is better predicted than the strain, which could be explained by the use of a higher value of $\mathbb{C}$, which tends to skew the results towards the stresses. In this example we have chosen $\mathbb{C} = 5\mathbb{I}_{3\times 3}$, to be in a similar value to the original elasticity tensor, although this choice is not necessarily the most optimal. This can be seen better in the diagonal plots of Figure 2.14, where we can see that the stresses, although disperse, have a tendency to follow the line, while the others are over or underestimated.



(a) Diagonal error plot for strains.



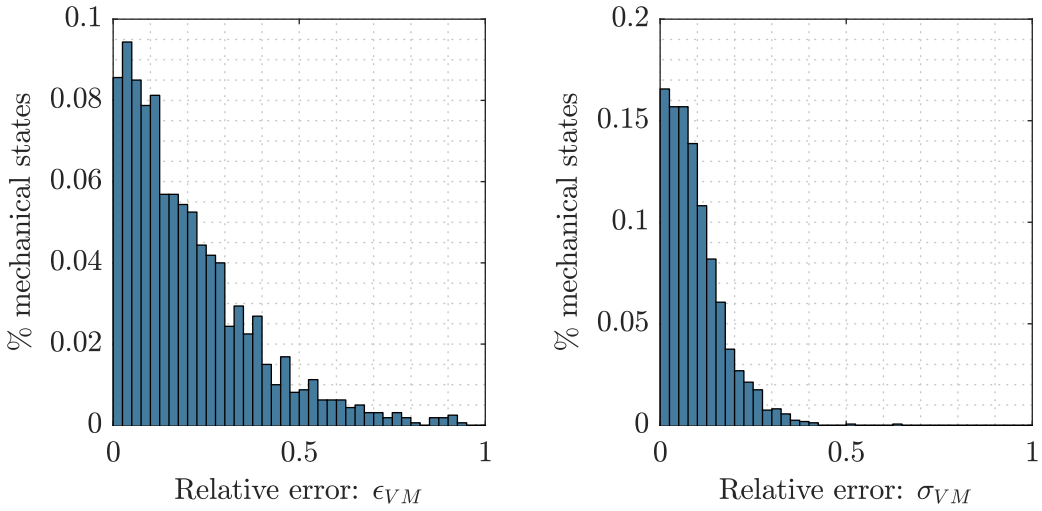
(b) Diagonal error plot for stresses.



(c) Diagonal error plot for displacements.

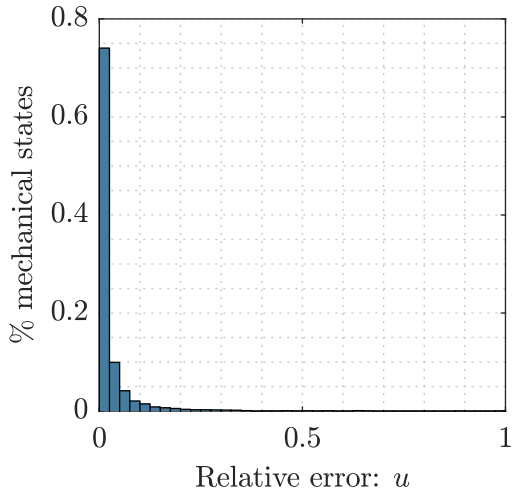
Figure 2.14: Diagonal error plots for the plane stress case in DDCM.

We can see in the histogram plots of Figure 2.15 that all quantities tend to follow the expected shape of the histograms tending to the left, specially when checking displacements, which correlates with the total error of the system.



(a) Histogram error plot for strains.

(b) Histogram error plot for stresses.



(c) Histogram error plot for displacements.

Figure 2.15: Histogram error plots for the plane stress case in DDCM.

2.5 Discussion

In this chapter, a little introduction into the world of data-driven method is given. It starts by recalling the basics of data gathering in the field of material science, as well as the basic physical principles for mechanics. It also provides an overview on computational mechanics and a short summary of how the Finite Elements Method works, to then explain in detail how both Data-Driven Computational Mechanics and Data-Driven Identification are implemented, complemented with examples that show the behavior and how they complement each other.

DDI and DDCM are two analogous algorithms based on the same principle of finding a solution through big data schemes. DDI was derived from DDCM by changing the approach, where now we aim to find the stresses in the samples studied, while at the same time we obtain a database of values that represents the material. While DDCM is convenient to avoid the definition of the constitutive model, in practice it might be complicated to even have access to strain-stress pairs, since for most materials we do

not know how stress relates to the displacements in that we measure. This is also the reason why both methods complement each other, since, as shown in the examples before, we can use DDI to obtain the information we require from the material without proposing a constitutive law, but at the same time we have enough to solve actual problems with DDCM.

Both data-driven algorithm share common traits with the FEM method. They are basically iterative schemes to minimize a problem that tends to a physically allowed solution, as a replacement for the direct calculation that is performed with finite elements. This has the consequence that data-driven schemes will never be able to provide a fully accurate solution like FEM does, but depending on the amount of data available and a good choice of parameters, this can be overcome to a great extent. The big novelty of the data-driven methods is that we do not need to make assumptions on the material before choosing a constitutive model, which sometimes is a necessity when we have behaviors that might resemble others. Relying purely on physical principles frees us from the arbitrarity of a chosen model, which can induce errors even when using exact methods as FEM.

Chapter 3

Data-driven identification for samples with two or more phases

In the previous chapter the focus was put on recalling the classical approaches for solving mechanical problems, as well as introducing the newer techniques and how they compare. In this section, we extend on the formulation on DDI to account for samples with heterogeneities, particularly samples created as a matrix with inclusions of stiffer materials, a setting commonly seen in composites[1].

3.1 Motivation

In the last years there has been an increase of the amount of new materials appearing, largely due to the improvement in manufacturing techniques, which allows for more precise and intricate design. In organic composites, fibers have been successfully used to reinforce polymers to achieve enhanced mechanical properties. For the developing of more sustainable solutions, there is an increasing interest in tailored composite materials to meet multiple properties and functionalities. A new generation of technologies have been recently developed, such as the *Automated Fiber Placement* [34], *Tailored Fiber Placement* [35], and *Additive Manufacturing* [36] to name a few, which allow more complex design concepts to be made. Composites with continuously evolving microstructure can be manufactured to get optimal functional performance requirements that vary with location within the part. For instance, curvilinear continuous fiber reinforced polymers can be produced. As these technologies are relatively young, there is no computational model yet to predict the microstructure in large engineered parts. These deficiencies are a limitation for the industry, where the aim is to take advantage of this new class of composites for the design and manufacture of new structural components with optimized mechanical properties. In this sense, a fundamental understanding of mechanical properties derived from the constitutive interactions between sub-components, their distribution, gradients and patterned structures under mechanical loadings, are needed to perform these optimizations. In all these cases, there is a need to develop an alternative approach to identify spatially evolving material proper-

[1] This chapter is taken from: G. Valdés-Alonzo, C. Binetruy, B. Eck, A. García-González and A. Leygue. *Phase distribution and properties identification of heterogeneous materials: A data-driven approach*, Computer Methods in Applied Mechanics and Engineering, vol. 390, p. 114354, 2022 [33].

ties.

3.2 Preliminary considerations

As was shown in Chapter 2, DDI gives us the freedom to identify the mechanical stress in samples from full field displacement measurements. So far, DDI has mainly been used in samples made of homogeneous materials where elastic behavior was assumed.

A first, naive approach to the problem is to apply DDI directly into an heterogeneous case. In this chapter, in order to study the behavior of the algorithm, the focus will be put in one particular sample with a set of parameters that will be changed according to the different analysis performed.

3.2.1 General properties of the sample

The sample used in this chapter is a rectangular membrane. A *plane stress* assumption is made, meaning that the perpendicular component of the stress is considered to be negligible. Even though the strain in the normal axis is not zero, we ignore its value through the rest of this document, given that it doesn't provide useful information.

The materials to be used for both the matrix and the inclusions in the sample are linear elastic with different values of Young's moduli depending on the case to study. For simplicity, the samples are isotropic, so the strain-stress tensorial relation is expressed as

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} \frac{E}{1-\nu^2} & \frac{E\nu}{1-\nu^2} & 0 \\ \frac{E\nu}{1-\nu^2} & \frac{E}{1-\nu^2} & 0 \\ 0 & 0 & \frac{E}{1+\nu} \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{xy} \end{bmatrix}, \tag{3.1}$$

where E is the Young's moduli and ν is the Poisson's ratio. In all the examples shown, ν is considered to be 0.3. To study the cases, the elastic moduli are not specified directly, but instead they are referred as a ratio between the stiffnesses, defined as

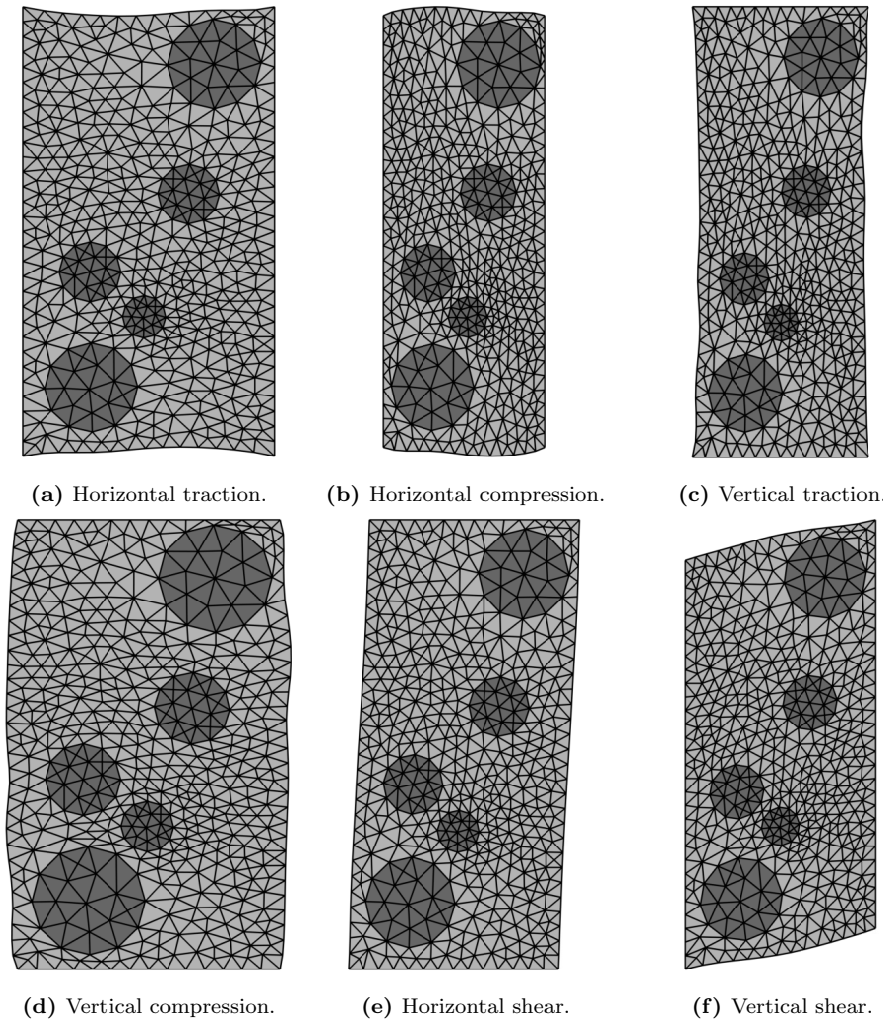
$$r_E = \frac{E_i}{E_m}. \tag{3.2}$$

where E_m is the stiffness of the matrix and E_i for the inclusions. The matrix material is modeled to always have an elastic modulus of $E_m = 1$, allowing us to only change the stiffness of the inclusions.

We define a mesh of linear triangle elements with one integration point. The choice of elements comes from practicality, since the code used for generating the data was already written with triangle elements in mind. However, as seen in the examples of Chapter 2, the choice of shape of the elements plays no role in the results obtained from DDI. Furthermore, using higher order elements might provide a more localized estimation of properties, but it is something that will not be addressed in the course of this work.

For the example in which DDI will be applied, the sample is modeled as a matrix material with circular inclusions with stiffness $E_i = 10$. To generate a rich input, we apply traction, shear and compression loads in both axes as illustrated in Figure 3.1. The mesh has $N^n = 527$ nodes and $N^e = 964$ elements and we consider $N^X = 80$

loading cases, or snapshots obtained by applying imposed displacements on the sample in linear increments. This gives a total of $80 \times 964 = 77120$ mechanical points. The size of this input data is actually low with respect to other published applications of DDI either on synthetic [11, 37] or experimental [27] data. As a result one can expect that the predictions will be of poorer quality. This choice is purposely made to expose the added value of the proposed approach applied to limited data.



(a) Horizontal traction. **(b)** Horizontal compression. **(c)** Vertical traction.

(d) Vertical compression. **(e)** Horizontal shear. **(f)** Vertical shear.

Figure 3.1: Deformations applied to the mesh.

3.2.2 Choice of parameters

A parametric study is performed to see how the algorithm behaves under different values of both $\mathbb{C}$ and N^*. As shown in [27], the value of N^* is expected to have an optimum. If the amount of material states is too low, the clusters of mechanical points are too big and they will not represent the stresses correctly. However, if the amount is too high, there is a risk of over-representation that will also hinder the estimation.

With the sample defined in Section 3.2.1, we run the algorithm for different amount of material states, also considering different values of the tensor $\mathbb{C}$, which for simplicity, is defined as a proportional to the unit tensor. We could expect to have a better estimation if $\mathbb{C}$ would be closer to the original elasticity tensor, but that might carry

some complexity in the problem given the need to define two parameters inside of it. Also, it is important to note that even if $\mathbb{C}$ is similar in form to the elasticity tensor, it does not represent any physical property of the material, but it is rather a numerical constant used by the algorithm to minimize the distances in the phase space, so its form is free as long as it is dimensionally correct. For studying N^*, we define a ratio r^* defined as

$$r^* = \frac{N^e \cdot N^X}{N^*}. \tag{3.3}$$

Finally, we compute the 2-norm relative error e_2, using as reference the FEM solution for the sample. Mathematically this is defined as

$$e_2 = \frac{||\boldsymbol{\sigma}_e^{DDI} - \boldsymbol{\sigma}_e^{FEM}||}{||\boldsymbol{\sigma}_e^{FEM}||}. \tag{3.4}$$

In Figure 3.2, the results are in accordance with the ones reported in [27]. The optimal ratio is between 20 and 100 depending on the value of $\mathbb{C}$, meaning that in average, each material point should gather that amount of mechanical states. The optimal ratio is in the same order of magnitude than the one obtained for an homogeneous material case.

Unlike what has been observed for single material samples [26] it is not optimal to consider higher values for $\mathbb{C}$. In [26], DDI was applied to a hyperelastic single material specimen where stress is obviously uniquely determined by strain. It is therefore not surprising that high $\mathbb{C}$ values, which promote a strain-based clustering of the mechanical states, are beneficial to the stress estimation. In the present case, as the specimen involves two material phases with different stiffness, the same strain can correspond to different stress values. It is therefore reasonable to assume that both strain and stress have to be accounted for in the clustering of mechanical states around material states. In this case, the value of $\mathbb{C} = 5E_m\ \mathbf{1}_{3\times3}$ gives the lowest error for estimating the local stresses in the sample. The specific value $\mathbb{C} = 1.3963E_m\ \mathbf{1}_{3\times3}$ corresponds to the homogenized stiffness of the sample in the y-direction.

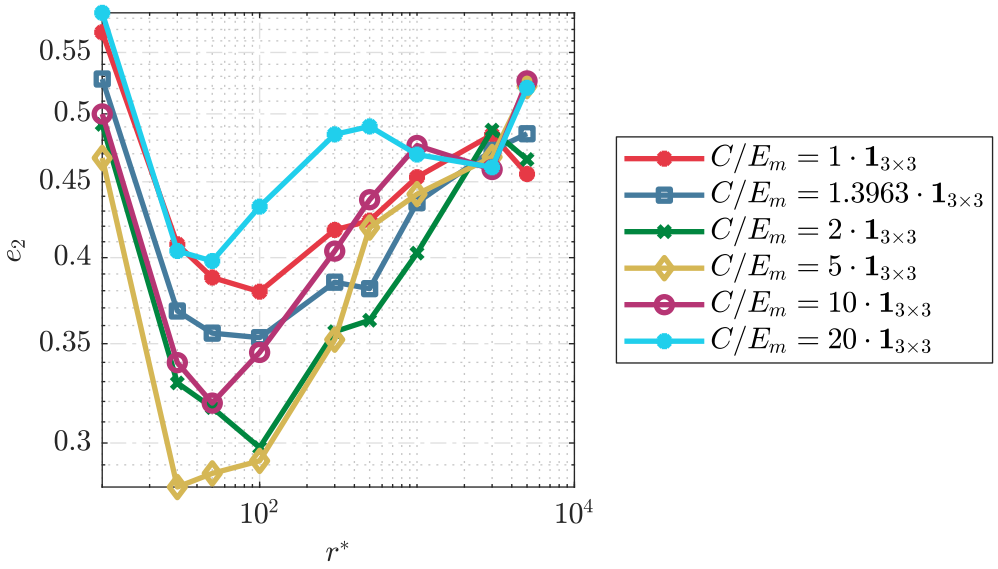
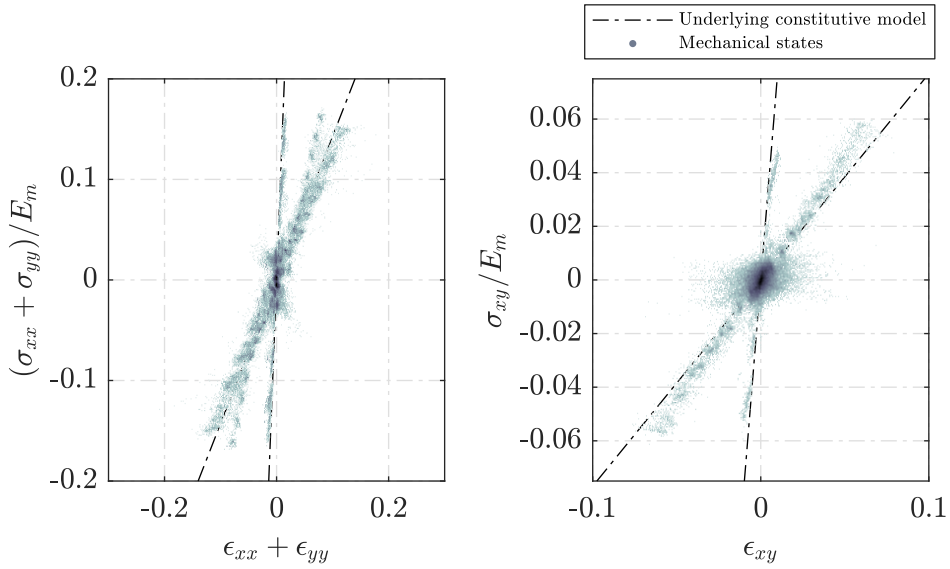


Figure 3.2: Relative error of estimated stresses against amount of material points.

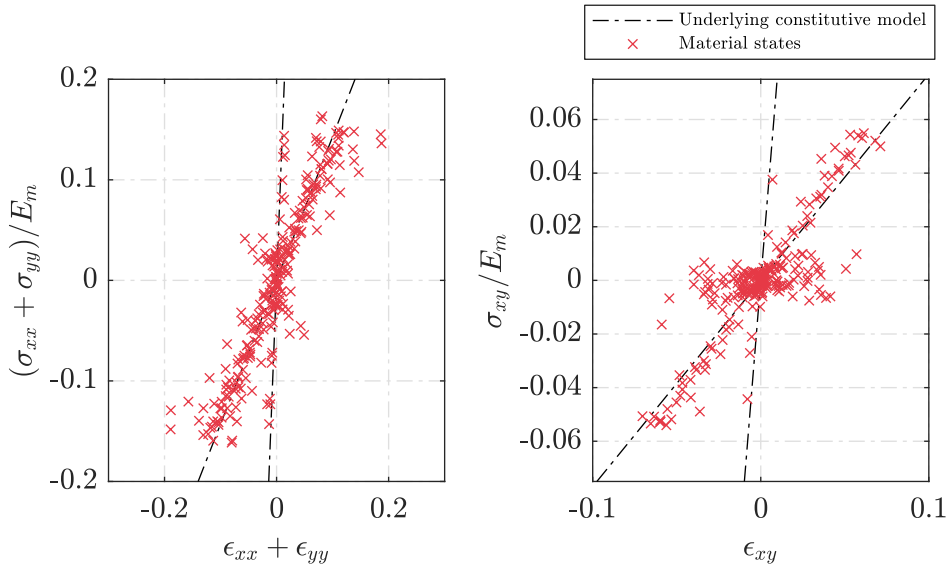
3.3 Application of DDI to heterogeneous samples

We focus on the case with $\mathbb{C} = 1.3963 E_m \ \mathbf{1}_{3\times3}$ and $r^* = 300$ to further analyze DDI predictions. This choice is made since this value represents the homogenization of the sample, meaning an estimation obtained with the force applied and the total stretch. Figure 3.3 shows the results for both mechanical and material states, together with the underlying constitutive laws used to generate the data. The behavior is similar to the one observed in the previous studies with one material. The mechanical states tend to gather around the expected values of stress-strain in clusters that have one of the material states as a centroid. There is a relative loss of accuracy when we go to the extreme values of strain, which can be attributed to the lower amount of mechanical states involving higher deformations.

The results show a clear trend, where both mechanical and material states tend to group around the two suggested constitutive equations. Since there are two material behaviors present in the data, the algorithm places the material states in a way that they sample simultaneously both materials, estimating also the stresses clustered around them. However, determining which behavior corresponds to each material is not straightforward. The fact that there is more than one behavior present represented through a unique set of material points, means that a mechanical state can be associated with any of them. This problem is certainly more pronounced when strains tend to zero, given the high amount of mechanical points close to the intersection of the behaviors of the materials. In Figure 3.3b we see that, due to the dispersion of material points close to the origin, it is difficult to associate them to any of the two material phases in a simple manner.



(a) Mechanical states and their underlying constitutive relations.



(b) Material states and their underlying constitutive relations.

Figure 3.3: Results for DDI with two material phases.

Convergence was obtained after 23 iterations performed in 62.8075 seconds in a laptop computer with a 2.2GHz Intel Core i7-8750H processor and 32GB of RAM. The amount of iterations and time consumed are highly variable and strongly depend on the quality of the initialization of the algorithm, especially the ie^X pairing. As shown by Kanno [38] in the case of DDCM, the DDI problem is a mixed integer quadratic programming problem. The minimization may fall into a local minimum and leads to a sub-optimal solution. The algorithm may also stagnate around a local minimum before "escaping" and finding a more optimal solution. These are factors that need to be considered while assessing the computation time.

3.4 Separating behaviors

From the previous results we have seen that DDI indeed samples the 2 different material behaviors, but does not separate them. However, we can see that, since material points tend to gather around the underlying constitutive equations, we have mechanical states that are associated to only one of the behaviors, particularly at high strains. This serves as the basis to establish a methodology for separating the phases in the sample.

3.4.1 Heuristic approach

In Figure 3.4, a simplified version of representative output DDI consisting of six elements over two loading increments (12 mechanical states) and five material states is illustrated. The results are represented in a simplified two-dimensional strain-stress space where the underlying material behaviors of the matrix and inclusions phases are depicted as dashed lines. The mechanical state, *i.e.* the strain (measured) and the stress (identified) values within each element are represented using triangle symbols ($\triangle_1$) where the subscript indicates the element number within the mesh. As DDI is performed over several loading increments (snapshots) each element appears several times, once for each increment. The material states of the identified database are represented as stars ($\star_a$) where the letter subscript is used to number each state. The values of both strain and stress of each material state are DDI outputs. Additionally, the color of the symbol of each mechanical state ($\triangle$) matches the color of the material state ($\star$) that is the closest according to the norm defined in Equation 2.24. This information is provided in the DDI output in the $\mathrm{i}e^X$ state mapping. For example in Figure 3.4 we have $\mathrm{i}2^1 = \mathrm{i}(2,1) = d$ and $\mathrm{i}2^2 = \mathrm{i}(2,2) = e$ as the triangle $\triangle_2$ is closest to $\star_d$ in one loading step (assumed here to be the first) and closest to $\star_e$ in another. The cells of the material states are delimited using dotted lines.

From this representation it is visually straightforward to postulate that elements ($\triangle_4, \triangle_5$) (resp. elements ($\triangle_2, \triangle_6$)) are likely to belong to the matrix (resp. inclusion) phase and that material states ($\star_b, \star_e$) (resp. material states $\star_d, \star_e$)) represent the material behavior of the matrix (resp. inclusion) phase. For elements ($\triangle_1, \triangle_3$) it is also possible to make an educated guess as to which phase they belong but it is impossible for material state ($\star_a$).

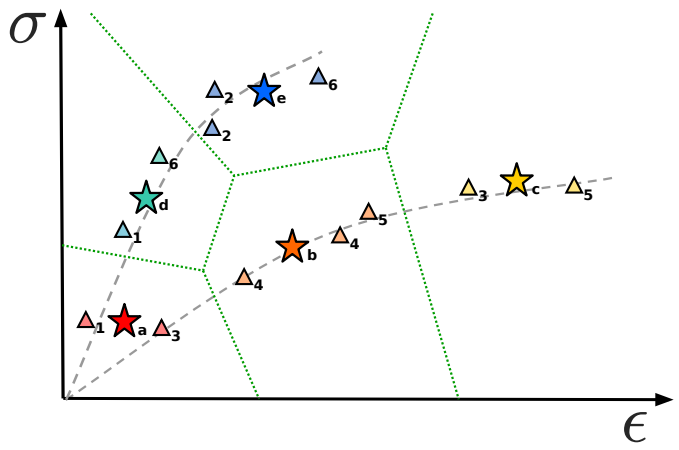


Figure 3.4: Simplified representation of DDI. For a mesh with 6 elements and two loading states, the 12 triangles (△) represent the mechanical states and the 5 stars (☆) the material points.

The previous analysis is merely heuristic and can only be performed in small cases of limited dimensionality involving only scalar strain and stress. For a more rigorous identification of the different materials we introduce the following clustering problem: we seek a clustering of the elements and of the material states in respectively two groups, such that the first (resp. second) cluster of elements mostly correspond through $\mathrm{i}e^X$ to the first (resp. second) cluster of material states. To solve this problem, we introduce the element-material frequency table $\boldsymbol{N}$. This table has as many rows as there are elements in the mesh (N^e) and as many columns as there are material states in the database (N^*). Its entries simply count the number of times each element is associated to a particular material state and it can be computed in a straightforward manner from $\mathrm{i}e^X$. The frequency table corresponding to the case depicted in Figure 3.4 is shown in Figure 3.5.

△ \ ☆	a	b	c	d	e
1	1			1	
2					2
3	1		1		
4		2			
5		1	1		
6				1	1

Figure 3.5: Element vs. material state frequency table $\boldsymbol{N}$.

3.4.2 Correspondence Analysis

The study of the correspondence between different categories, here element number against material state number, can be achieved using *Correspondence Analysis* (CA)

[39]. Examples of its use can be found in [40] for social sciences and [41] for ecology, where it is applied to visualize the information in a different way. Specifically, CA is a multivariate statistical technique used for summarizing matrices of non-negative data in a graphical form [42]. It differs from other similar methods like *Principal Component Analysis* (PCA) because of its application in categorical data, instead of continuous one. CA studies the relation between the rows and columns of a frequency table by considering them as points in a space. The technique aims to measure the distances between these points and then projecting them in a low-dimensional space where similar categories are clustered. The procedure for CA presented in [39] is outlined here. We start by considering the element-material state frequency table N, which is the sum over all loading steps of all state mappings $\mathrm{i}e^X$, i.e.,

$$\boldsymbol{N} = \sum_{X=1}^{N^X} \mathrm{i}e^X. \tag{3.5}$$

$\boldsymbol{N}$ is a matrix of dimensions $N^e \times N^*$. Each matrix $\mathrm{i}e^X$ tracks to which material state a mechanical state is associated in a particular loading case X. The sum of them counts how many times a specific element of the mesh has been paired to each material state. This table of frequencies is the basis for any type of CA. Matrix $\boldsymbol{N}$ is normalized as

$$\boldsymbol{P} = \frac{1}{n}\boldsymbol{N}, \tag{3.6}$$

where $n = \sum_i \sum_j N_{ij}$. This matrix is called the *correspondence matrix*, and is the base for the analysis. From here, we obtain the row and column masses,

$$\begin{aligned} r_i &= \textstyle\sum_j p_{ij}, \\ c_j &= \textstyle\sum_i p_{ij}. \end{aligned} \tag{3.7}$$

$\boldsymbol{r}$ is a column vector of length N^e, collecting the sum of each row. Similarly, $\boldsymbol{c}$ is a row vector of length N^* with the sum of each column. From here, we can define $\boldsymbol{D}_r$ and $\boldsymbol{D}_c$ as the diagonal matrices with the terms of $\boldsymbol{r}$ and $\boldsymbol{c}$. We use them to compute the matrix of standard residuals $\boldsymbol{S}$ as

$$\boldsymbol{S} = \boldsymbol{D}_r^{-1/2}(\boldsymbol{P} - \boldsymbol{r}\boldsymbol{c})\boldsymbol{D}_c^{-1/2}. \tag{3.8}$$

The matrix of standard residuals corresponds to the square root of the inertia of the data. This means that S effectively measures the χ-square distance of each point to their centroids, weighted by the inverse of the expected average profile [43]. A *Singular Value Decomposition* (SVD) is performed in the matrix $\boldsymbol{S}$:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{D}_\alpha\boldsymbol{V}^T. \tag{3.9}$$

The SVD technique decomposes the matrix into 3 new matrices: The matrix $\boldsymbol{U}$, of dimension $N^e \times N^e$, where each column is called a *singular vector*; $\boldsymbol{V}$, of dimension $N^* \times N^*$ and analogous to $\boldsymbol{U}$; and the diagonal matrix $\boldsymbol{D}_\alpha$, with dimensions $N^e \times N^*$,

in which each term is called *singular value*. These matrices are used to compute the principal components $\boldsymbol{F}$ and $\boldsymbol{G}$:

$$\begin{aligned} \boldsymbol{F} &= \boldsymbol{D}_r^{-1/2}\boldsymbol{U}\boldsymbol{D}_\alpha, \\ \boldsymbol{G} &= \boldsymbol{D}_c^{-1/2}\boldsymbol{V}\boldsymbol{D}_\alpha^T. \end{aligned} \tag{3.10}$$

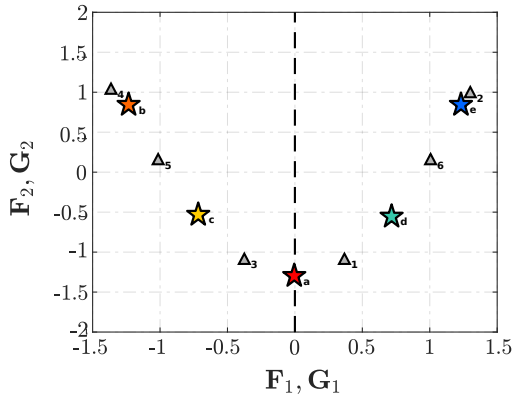
Optionally, the principal inertias of the data can be computed as

$$\lambda_k = \alpha_k^2, \tag{3.11}$$

where α_k is each one of the diagonal elements of $\boldsymbol{D}_\alpha$. Appendix D includes some notes and further explanation of the mathematical aspects of the algorithm, which are technically out of the scope of this work, since we only use it as a tool for a particular purpose.

The results of CA can be understood as a parallel to the concept of eigenvalues and vectors of a matrix. The principal coordinates $\boldsymbol{F}$ and $\boldsymbol{G}$ correspond to values associated with the rows (mesh elements) and columns (material states) of the correspondence matrix, respectively. In a similar fashion to the eigenvectors, the columns of the principal coordinates create a space that we can represent graphically. The principal inertias estimate the relevance of each principal coordinate, similar to the eigenvalues of a matrix. In this space, the data is rearranged and presented in clusters that can be interpreted as the different phases of the material. Depending on the complexity of the problem and the quality of the data, the amount of dimensions used to perform the analysis can vary. In most of the cases a 2D representation of the principal coordinates is enough, but as it will be shown in Section 3.5, more dimensions can solve a problem that is not visually accessible in 2D.

For the example presented earlier in this section, the results can be seen in Figure 3.6a. In here we see that the groups that we formed with our heuristic approach are respected by appearing on opposite sides of the space. Furthermore, elements ($\triangle_1, \triangle_3$) are assigned to different phases, without the need for a guess. However, in the case of ($\star_a$) we have the ambiguity since the elements associated to it are equally represented in each material phase.



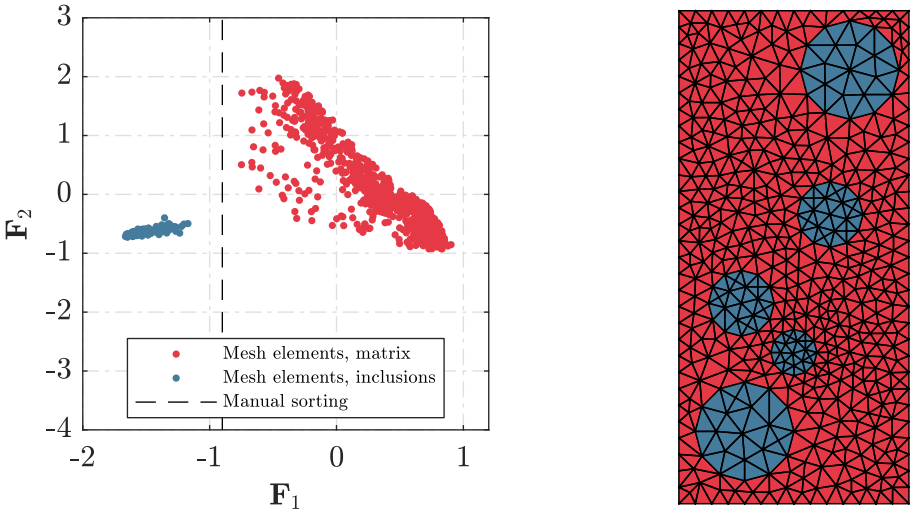
(a) Schematic of CA space.

△ \ ☆	b	c	a	d	e
4	2				
5	1	1			
3		1	1		
1			1	1	
2					2
6				1	1

(b) Sorted contingency table $\boldsymbol{N}$.

Figure 3.6: Results of the heuristic approach.

We now apply this analysis to the example presented in Section 3.3. The result of plotting the first two columns of $\boldsymbol{F}$ is shown in Figure 3.7a. The correspondence matrix clearly appears as a set of clusters in this new space, which represent categories in the ie^X pairing. We separate them assuming that these categories actually correspond to the different material behaviors, and identify the two clusters with the matrix and inclusion phases. Locating these clusters back in the mesh gives the result shown in Figure 3.7b. From this, we see that applying CA to the ie^X pairing *a posteriori* can indeed separate the two materials that DDI identified blindly.



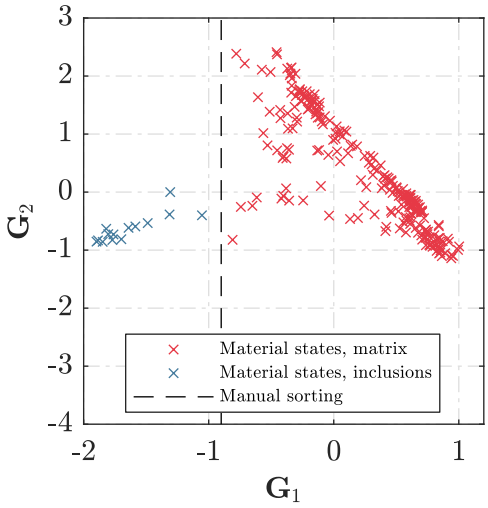
(a) CA plot for mesh elements.

(b) Mesh elements sorted.

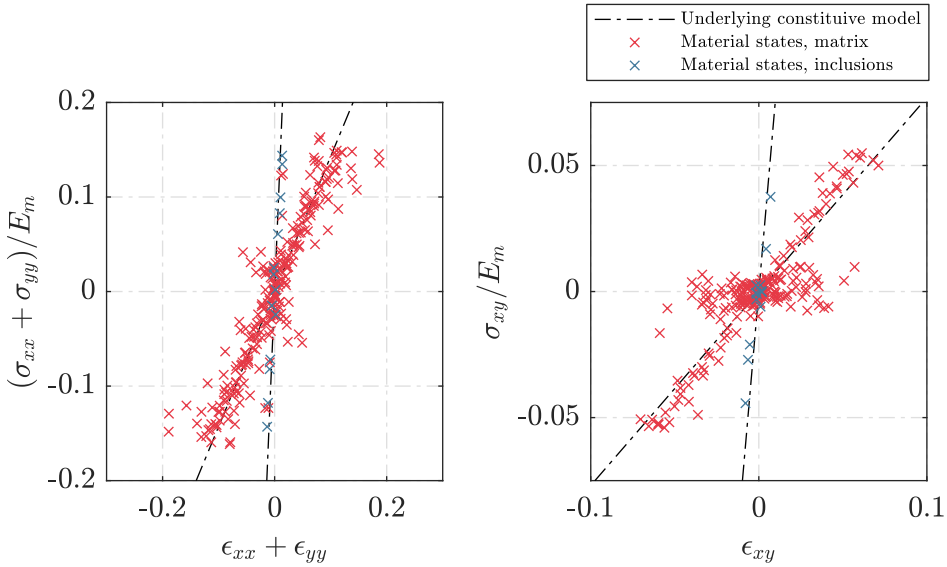
Figure 3.7: Mesh elements: Separation process.

To perform the separation on the material states in the space $(\boldsymbol{G}_1, \boldsymbol{G}_2)$ we use the same separation as in the $(\boldsymbol{F}_1, \boldsymbol{F}_2)$ space. Following the same procedure we can obtain the material states that can be represented in the (ϵ, σ) space, displaying the two material behaviors, as seen in Figure 3.8b. The criterion for separation is taken as the same in the $\boldsymbol{F}$ space because both spaces are equivalent representations of the

same original matrix. However, performing the separation in the $\boldsymbol{F}$ space is easier since the clusters have a physical meaning, i.e. the location in the sample. In the $\boldsymbol{G}$ space the separation is more challenging, since it can be difficult to establish what represents each behavior in the phase space.



(a) CA plot for material states.



(b) Material states separated in phase space.

Figure 3.8: Material states: Separation process.

3.5 Ranges for the heterogeneous DDI algorithm

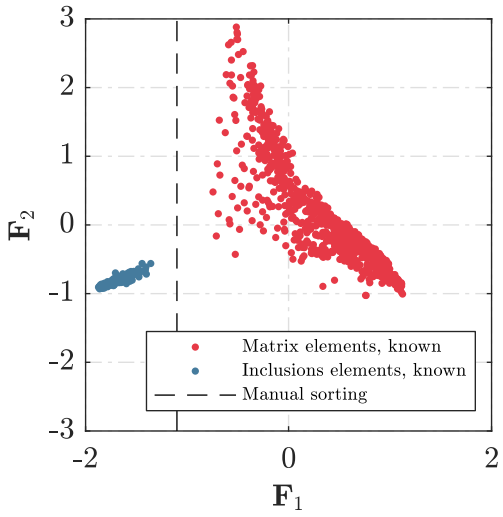
The previous section showed that DDI combined with CA is able to provide an estimate of the stress field in the sample, its material phases distribution and the mechanical response of each phase. To show the capabilities of the proposed method, we test this approach in cases for lower inclusion/matrix stiffness ratios and input data with limited kinematics.

3.5.1 Minimum stiffness ratio for inclusions

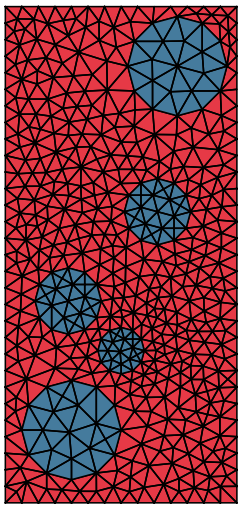
An important point to analyze is the sensitivity of the algorithm with respect to the stiffness ratio of the two-phase heterogeneous material. Since the estimation of stress tends to cluster around the unknown surfaces that define the materials responses, noise can at some point prevent the identification of both materials if the surfaces are too close. To analyze this effect, we use the same sample as before and we set the number of material states to a value close to the optimal one found in the previous section ($r^* = 50$). The value of $\mathbb{C}$ is fixed again from the homogenized stiffness of the sample to unify all the cases under the same testing conditions. The stiffness value is tested, starting from the highest ratio and then decreasing it to find the limit from which separation through CA cannot be found. In general, we are not interested to test cases with a higher stiffness ratio, since CA works well when there is a clear separation of behaviors. The only case where CA would fail for higher ratios is when inclusions become too stiff that the sensitivity of the camera used for DIC is not capable to detect the movement, which is a problem not related to the method, but to the physical setting of the experiment.

Results for different stiffness ratios are summarized in Figure 3.9. On the figures of the left, we show with colors the exact phases of the material, with the arbitrary separation marked by the dotted line. On the right, the colors of the mesh are plotted according to the separation. We can see that with the decrease of the stiffness ratio, the identification of clusters from the CA analysis becomes more difficult. For $r = 10$, the separation can be performed by only using the first principal component $\boldsymbol{F}_1$. When the ratio decreases, it is necessary to extract more information. With a stiffness of $r = 3.5$, the behaviors are still separable using two principal coordinates as in Figure 3.9a. For a ratio of 3.125, we still can perform a separation visually if we include the third principal component in the analysis, but we observe some misplaced elements (Figure 3.9f). Plotting the principal coordinates for lower ratios show results similar to those observed in Figure 3.10a, for which separation is impossible using visual techniques. It is expected that using more principal coordinates could help in these cases, but the impossibility of analyzing this case visually makes it impractical. In these cases, we have to turn to unsupervised clustering methods.

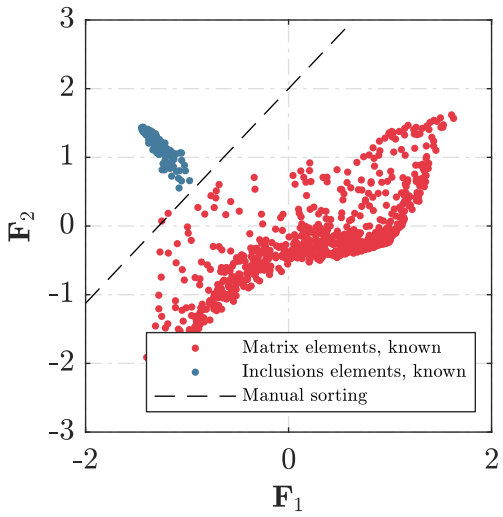
Among all the clustering techniques, *k-means* [32] is a popular choice that allows to partition a sample space in k groups. We choose this technique due to its easy access in the Matlab suite, and the details of how the algorithm is implemented are detailed in Appendix B. Using k-means with a *k-means++* initialization [44] for finding groups in higher dimensions ($n = 4, 5$) allows to obtain satisfactory results. As the quality of k-means clustering strongly depends on the initialization step, it is expected that more robust clustering methods might lead to better and highly reproducible results. In Figure 3.10 we report the phases identified for a stiffness ratio $r = 2$ using k-means clustering on the first five principal coordinates $\boldsymbol{F}_1$ to $\boldsymbol{F}_5$. Phase separation is satisfactory, even if some elements belonging to the matrix phase are assigned to the inclusion. A possible path for improving the results could be to simultaneous cluster mesh elements in the $\boldsymbol{F}$ space, as well as in physical space to prevent the prediction of physically isolated inclusion elements.



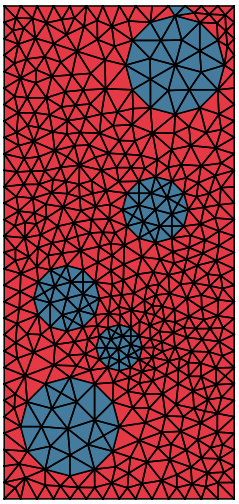
(a) Separation for high stiffness ratio ($r = 10$).



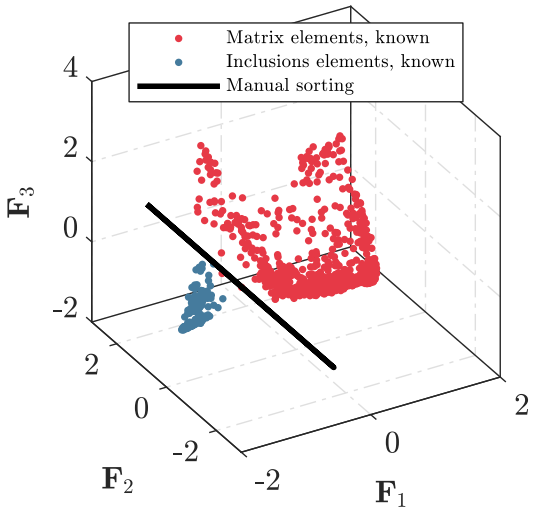
(b) Sorted mesh for $r = 10$.



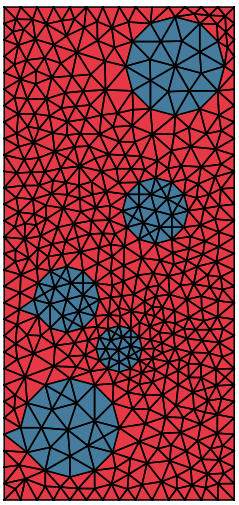
(c) Separation at $r = 3.5$.



(d) Sorted mesh for $r = 3.5$.

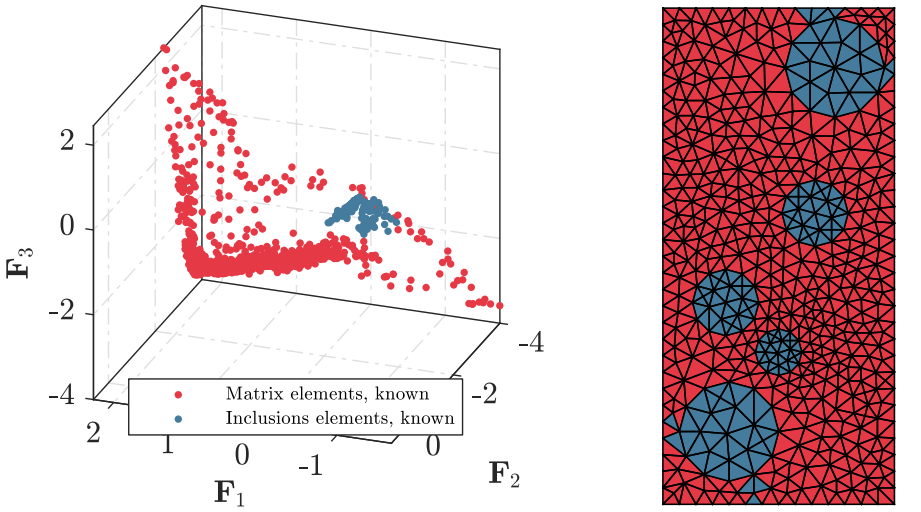


(e) Separation at $r = 3.125$.



(f) Sorted mesh for $r = 3.125$.

Figure 3.9: Separation for different stiffness ratios.



(a) Elements corresponding to each material.

(b) Sorted mesh for a ratio of $r = 2$.

Figure 3.10: Identification of stiffness ratio $r = 2$ using 5 modes and k-means clustering.

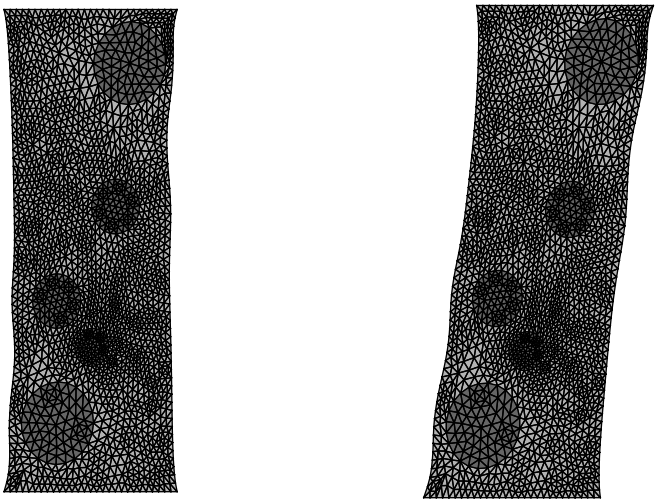
3.5.2 Identification with poorer inputs

So far, we have considered academic benchmark problems, for which we had a collection of kinematically rich strains. However, this ideal situation is rarely encountered in real-life testing. Samples must be tested consistently, ideally in one stage to not alter the results due to external factors. In this sense, using equipment such as an hexapod [26], allows to load samples along multiple axes without changing the experimental setup and keeping the same reference configuration. Even then, results might not be as rich as the previously used synthetic data. To prove the robustness of the proposed method, in this section we analyze a sample considering loadings that are representative of what is achievable using an hexapod.

The procedure is as follows: we stretch the sample vertically in 40 steps, similar to a tensile test. When the maximum strain is reached, the upper clamp is displaced horizontally to add shear stress in the sample, again in 40 steps. Snapshots are collected regularly during the whole procedure. Since some loading conditions are omitted (i.e. compression testing) with respect to previous scenarios, we can expect the identified behavior to be more limited. The stiffness ratio is also lowered to 5.

For this case, we use the same sample defined in Figure 3.1 but with a finer mesh. This refined mesh consists of $N^n = 2017$ nodes and $N^e = 3856$ elements, giving a total of $3856 \times 80 = 308480$ mechanical points. This will help us increase the amount of data and hopefully compensate the loss of information due to the poorer kinematics. The parameter $\mathbb{C}$ is kept similar as the previous case, considering the homogenized stiffness of the sample. We set a ratio for the material points of $r^* = 600$, which is not the optimal but it allows the results to be more accessible visually.

Figure 3.11 shows the deformation of the sample in both stages.

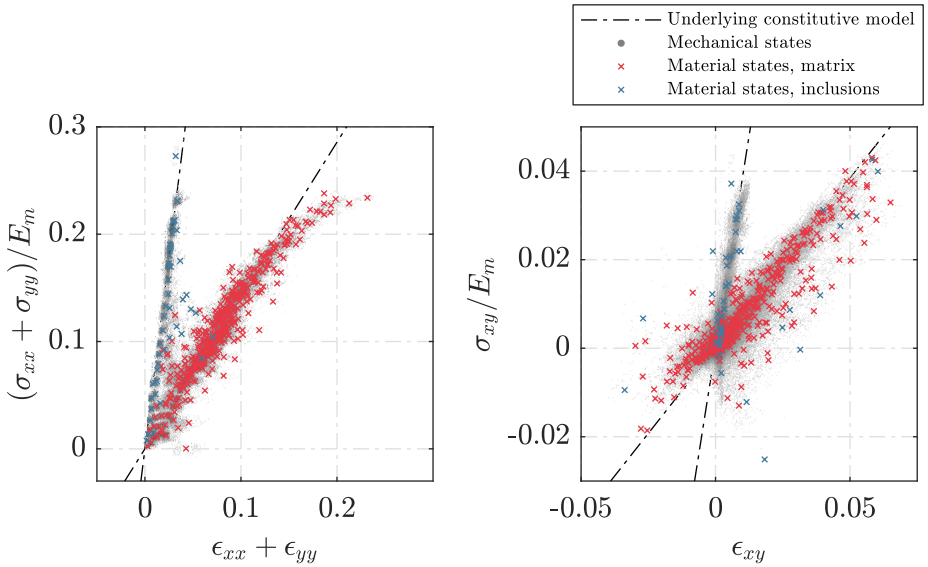


(a) Traction deformation.

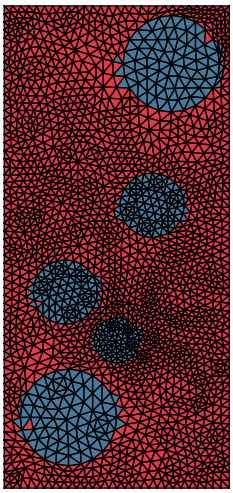
(b) Horizontal shear deformation.

Figure 3.11: Deformation stages of the sample.

Applying the algorithm and performing the correspondence analysis yields the results presented in Figure 3.12. CA performs well in finding the position of the inclusions, where only some elements are assigned wrongly close to interfaces. In Figure 3.12a the strain-stress representation shows that the isotropic response of both phases is better captured than shear. We observe a much larger scatter for both mechanical and material states. The different material behaviors are poorly sampled close to the origin, and the shear behavior of the inclusion phase is poorly predicted, with some material states far from the actual material behavior. This can be expected from this limited loading scenario. However, despite the observed scatter, there is a significant correlation with the underlying material laws represented by the dash-dotted lines.



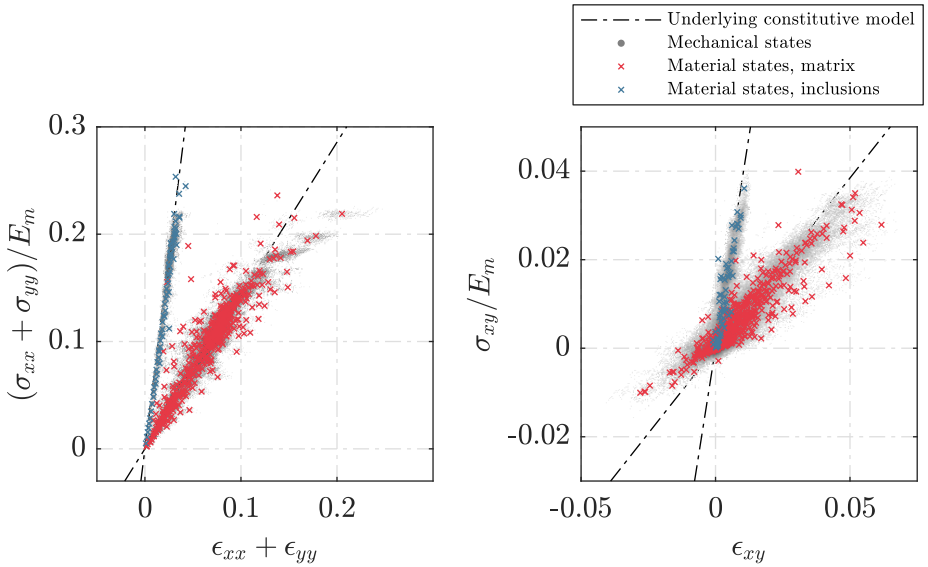
(a) Phase space of the solution.



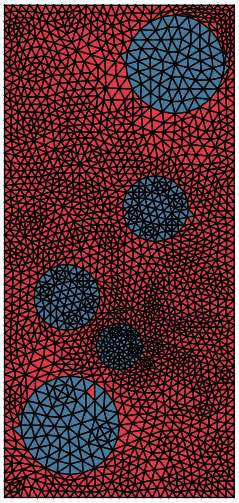
(b) Mesh of the sample with assigned elements.

Figure 3.12: Sample results.

To improve these predictions, we propose to use the output of CA to better initialize the DDI analysis in a second run. Through the ie^X variable, the optimization problem solved by DDI is indeed of combinatorial nature and the algorithm in the majority of cases converges to a local minimum. Therefore, in the second run of the DDI, ie^X is initialized in a way that it respects the matrix and previously identified inclusion phases. Only mechanical states that are identified by CA as belonging to the same phase can be associated to a specific material state through ie^X. This assumption is only made for initializing the algorithm, and is then relaxed to use the normal DDI as proposed in Chapter 2. CA is performed again after this second run. The results are presented in Figure 3.13.



(a) Phase space of the updated solution.



(b) Updated mesh.

Figure 3.13: Updated results after second run.

In Figure 3.13b, we observe that the phase distribution is much better predicted and most of mesh elements are now correctly associated to the correct phase. In the strain-stress space represented in Figure 3.13a, the scatter is visibly less pronounced. Moreover, the material states provide a finer description of the materials behavior at small strain, both for the isotropic and deviatoric part of the stress. Table 3.1 displays a quantitative comparison of prediction errors in both stages, all error indicators improve.

Table 3.1: Error of estimation, comparison of both iterations of the algorithm.

Error	First iteration	Second iteration
$\|\|\boldsymbol{\sigma}^{FEM}-\boldsymbol{\sigma}^{DDI}\|\|_{\mathbb{C}}/\|\|\boldsymbol{\sigma}^{FEM}\|\|_{\mathbb{C}}$	0.0125	0.0085
$\|\|\boldsymbol{\sigma}^{FEM}-\boldsymbol{\sigma}^{DDI}\|\|_2/\|\|\boldsymbol{\sigma}^{FEM}\|\|_2$	0.1682	0.1384
Amount of wrong elements in CA	50 (1.3%)	11 (0.29%)

We can also visualize the distribution with the use of the histogram plots. In Figure 3.14 the histograms of both the first and second iteration are plotted together to show the change. As can be seen, for the trace component of the stresses we have that the curve slightly moves to the left, which means that we have an overall improvement even though the maximum error value is increase. However, the improvement is much more pronounced in the shear component of the stress, which tends to be more complicated to estimate accurately.

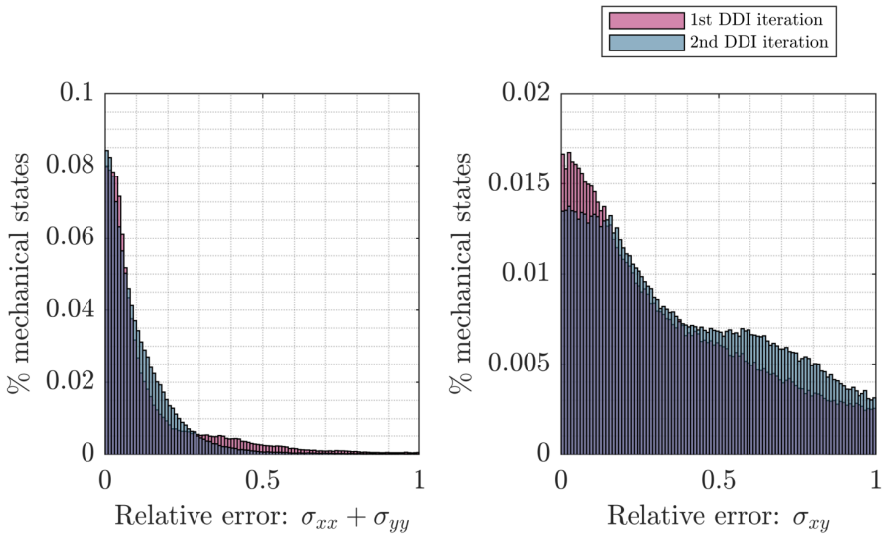


Figure 3.14: Histogram error plot for both iterations of DDI.

The improvement provided by the better initialization of ie^X can be appreciated in the correspondence analysis as well. In Figure 3.15, the clusters of points associated with the different behaviors are not clearly separated in the first CA run. Clustering based on the first principal component is not obvious. On the second run, the two sets become more apparent.

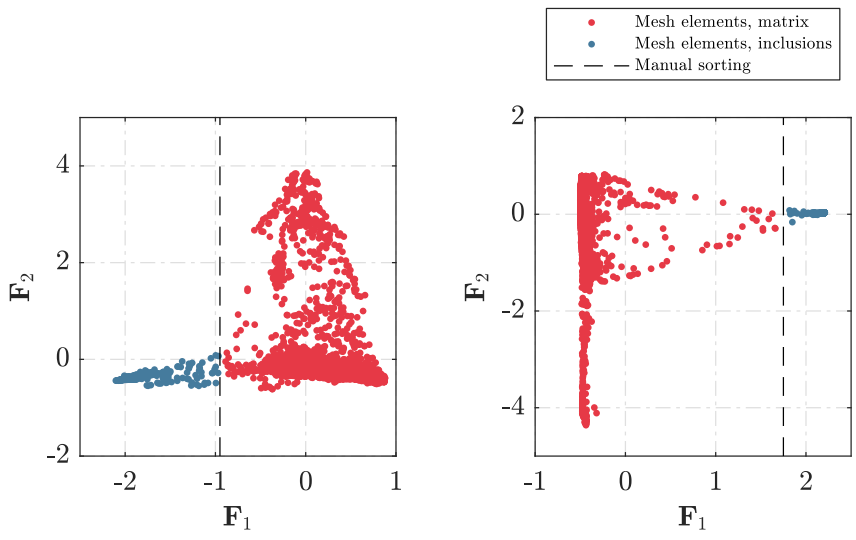


Figure 3.15: Comparison of CA in both cases.

3.6 Discussion

By complementing DDI with CA we were able to identify stress fields in synthetically manufactured samples composed of a matrix with stiffer inclusions. CA results in a different visualization of the data that allowed us to predict the location of these inclusions, which in turn can inform about the underlying behavior of each phase. As a consequence, the behavior of both the matrix and inclusions can be simultaneously identified.

The sensitivity of the method was tested by running test cases with different stiffness ratios between matrix and inclusion. It has been shown that the technique can work at low contrast limit, and when results tend to degrade, we can increase the dimensionality of CA to enhance sensitivity with very few added computations. We also observed that for samples with low richness of strains we can iterate between CA and DDI to improve stress estimation. CA provides an estimate of the position of the different phases, which can be used to properly initialize a second DDI analysis for improved predictions. Furthermore, the proposed approach is not necessarily limited to two-phase samples. The theory of CA allows the identification of more phases by generating of more clusters in the transformed phase space. However, in this setting we would expect a loss of accuracy, given the saturation of information in the strain-stress space. Finally, as the distinction between linear and non-linear elasticity is irrelevant for DDI, the present results are not limited by the choice of linear behavior to generate the synthetic input data.

One shortcoming of the current approach is that DDI does not provide accurate stress estimates when the stiffness contrast r_E is lower than 1. In the case of softer inclusions, the softer phase is kinematically very constrained and experiences strain levels very similar to the stiffer one which yields a very homogeneous strain field, containing very little usable data for the DDI. Modifications to the original algorithm might be devised to make use of the slight inhomogeneities in the strain field. However, we believe that one way of overcoming this problem is using a normalized version of the algorithm that automatically adjusts the value of the DDI $\mathbb{C}$ parameter, which is explored in detail in Chapter 5.

Chapter 4

Data-driven identification in linear viscoelasticity

In previous chapters we have stated the algorithm of DDI and its use in elastic cases. However, elastic behavior is only a subset of all the behaviors that can be found, while others tend to increase the complexity of the general stress identification. It is because of this that we want to explore further the use of the DDI algorithm in settings that differ from classical elasticity.

The main motivation for the study of linear viscoelasticity with DDI is simplicity. Viscoelasticity encompass a range of materials that vary in behavior according to the models selected. However, linear viscoelasticity is a phenomenon that is well understood and that can be kept simple by taking the proper approach. In this sense, linear viscoelasticity represents a smooth transition towards more complex material behavior.

One of the advantages are the similarities with the elastic case. Linear viscoelastic materials can behave like linear elastic materials when they are subjected to deformation, but if they are kept in the deformed position the stresses relax and decrease. In fact, if we consider a viscoelastic solid sample in which the relaxation is slow enough, we just revert back to the linear elastic case. This relaxation phenomenon is dependent on the material itself and it can be estimated if we consider the stress as a function of strain history, rather than just a particular strain in time. We aim in this chapter to tackle this particular point.

4.1 Linear viscoelasticity

Viscoelasticity is a term that refers to all materials that simultaneously present both elastic and viscous properties [45]. For elastic materials, Robert Hooke introduced the idea of spring-like materials, where the tension in the springs is directly proportional to the extension that it is subjected. Similarly for fluids materials, Isaac Newton proposed the idea that in a shear flow, the resistance on a fluid is dependent on the speed at which the flow is being disturbed. Both effects can be respectively written as

$$F = K \cdot \delta, \tag{4.1a}$$

$$F = \eta \cdot \dot{\gamma}, \tag{4.1b}$$

with K being the elastic constant of the spring and δ its deformation, while η represents the *viscosity* of the fluid and $\dot{\gamma}$ the rate of deformation. When using simplified models to represent the material behavior, springs are used to represent elasticity, while dashpots are used for viscosity. This graphic representation can be seen in Figure 4.1.

(a) Representation of an elastic element. **(b)** Representation of a viscous element.

Figure 4.1: Representation of behaviors in a material model.

One of the particular properties of viscoelastic materials is that the stresses in the sample are dependent on the strain history, rather than an instantaneous strains. It is because of this that the analysis performed are done in a prescribed time interval where the effects of the previous deformations will be considered.

For these kind of materials an analytical solution exists, although it might not be explicit, so mathematical schemes need to be used to obtain the solution. Furthermore, some models require the use of derivatives of both strains and stresses to obtain results. We have already discuss the difficulty of estimating stresses and how they are a source of error, a phenomenon that gets aggravated when derivatives are considered. In general and to abide by the principle of linearity, the constitutive law of a viscoelastic material is defined in a general way as [46]

$$\sigma(t) = \boldsymbol{\mathcal{E}}\{\epsilon(\tau)\}|_{\tau=t_0}^{\tau=t}, \tag{4.2}$$

where $\boldsymbol{\mathcal{E}}$ is a linear functional that relates the strains in the sample $\epsilon(t)$ to their respective values of stress $\sigma(t)$ in an specific time interval between $\tau = t_0$ and $\tau = t$. In here t represents the time of interest for the analysis in the material. It is important to mention that this functional definition is generalized: if we want to address elastic materials we can take an instantaneous value for $\tau = t$ and revert back to classical definition. If we perform a test that is force-driven rather than deformation-driven, the opposite relation can be defined as

$$\epsilon(t) = \boldsymbol{\mathcal{D}}\{\sigma(\tau)\}|_{\tau=t_0}^{\tau=t}, \tag{4.3}$$

where constants are defined in a similar way.

4.1.1 Viscoelastic models

In linear viscoelasticity, since both Hooke's and Newton's laws are linear relationships, we can apply the superposition principle and also obtain a linear relationship for these kind of materials. The naive idea would be to consider a behavior as a sum of both elastic and viscous behavior, which we obtain by summing the forces defined by Equation 4.1, which yields

$$\begin{aligned} \sigma(t) &= \sigma_e(t) + \sigma_v(t), \\ &= K\epsilon(t) + \eta\dot{\epsilon}(t), \end{aligned} \tag{4.4}$$

where σ_e and σ_v are the elastic and viscous part of the stress respectively and the notation of the dot represents a time derivative. In here we change forces and displacements for stresses and strains to refer to the internal phenomena in the material. In that sense, K and η do not represent the same quantities from Equation 4.1 but they serve similar purposes. From now on and to be consistent with the literature, we change K for E, which is the most common letter used to represent the linear constant between strains and stresses (i.e. the Young's modulus).

The representation defined in Equation 4.4 is one of the simplest viscoelastic models and it is typically known as the Kelvin-Voigt model. One of the advantages of this model is that it provides analytical solutions for certain cases. If a constant stress σ_0 is considered starting in $t = 0$, the differential equation in Equation 4.4 can be solved, which gives the strain

$$\epsilon(t) = \frac{\sigma_0}{E}\left[1 - \exp\left(-\frac{t}{\tau_K}\right)\right], \tag{4.5}$$

with $\tau_K = \eta/E$ being a ratio of time units that controls the growth rate of strain, also known as *retardation time* of the material. In Figure 4.2 the plot for the dimensionless quantity $\epsilon E/\bar{\sigma}$ is shown, which displays one of the main characteristics of viscoelastic materials: given a constant stress the increase of the strain will decrease exponentially up to an equilibrium point that is equivalent to the value that would be obtained if the material was purely elastic. How fast this decrease is experienced is governed by the parameter τ_K, hence its name.

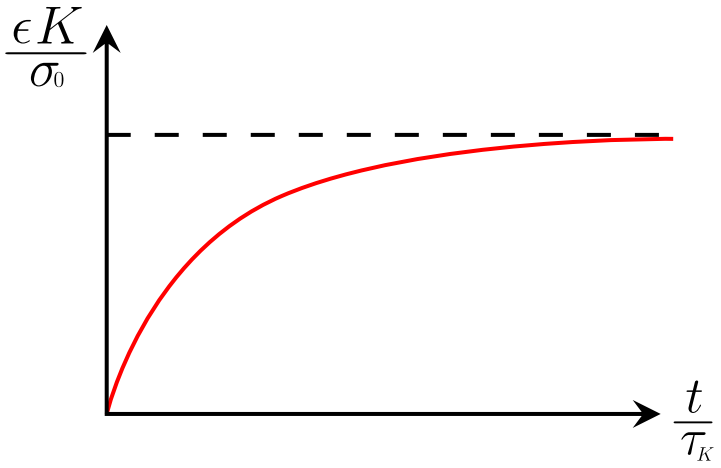


Figure 4.2: Stress over time for the Kelvin model. As can be observed, the increase of strain in time is a negative exponential curve that tends to a constant value. When one relaxation time has passed the strain is about 62% of the strain that we would have if the material was linear elastic.

Using springs and dashpots to represent material behaviors, the combination of elements is performed in a similar way to a system of springs in mechanics. A combination of elements in series implies that the stress remains constant in the branch, while the strains are obtained as a sum of the strains of each component. The opposite case happens when elements are connected in parallel: strains are equal in both branches while the total stress are a sum of the effects in each branch. Like this, it is easy to explain the Kelvin-Voigt model as a connection in parallel of both a spring and a dashpot since the stress is the sum of the effects in each of them. This can be seen in Figure 4.3.

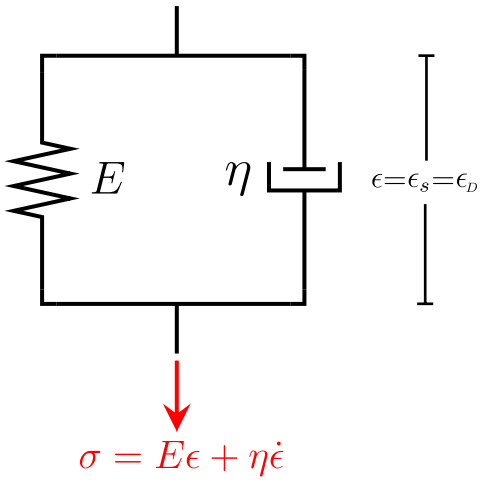


Figure 4.3: Representation of the Kelvin model. The strain of each branch is the same, while the stress is a combination of both behaviors.

There are many simplified models that are popular, but in this work we will present two more. If we decide instead to connect the spring and dashpot in series, we obtain a model that analyzes the problem from a strain-driven perspective. If we consider that

the strain rate of a sample is a sum of both elastic and viscous behaviors, we have

$$\begin{aligned}\dot{\epsilon}(t) &= \dot{\epsilon}_e(t) + \dot{\epsilon}_v(t), \\ &= \frac{\dot{\sigma}(t)}{E} + \frac{\sigma(t)}{\eta},\end{aligned} \tag{4.6}$$

which can be rearranged as

$$\sigma(t) + \frac{\eta}{E}\dot{\sigma}(t) = \eta\dot{\epsilon}(t). \tag{4.7}$$

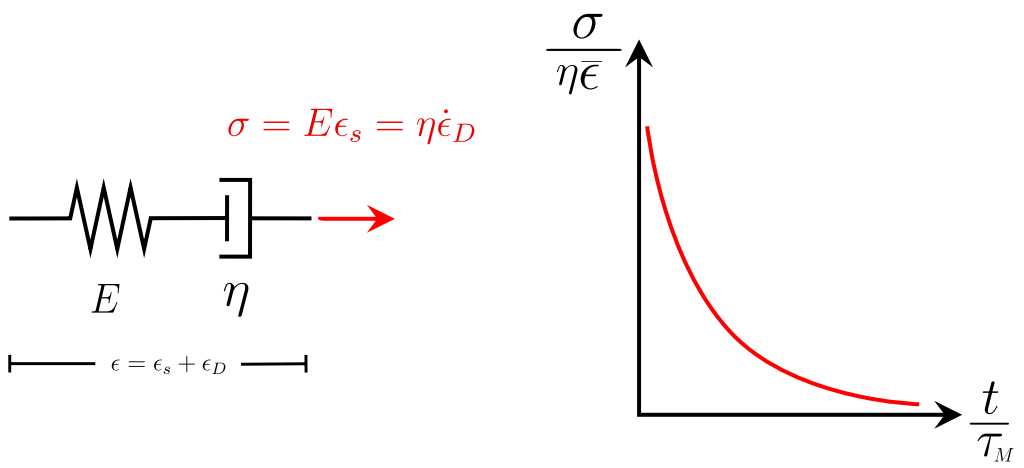
In this model we have the ratio η/E appearing again in the expression, which we can define as τ_M. In an analogous case to the Kelvin-Voigt model, taking a constant strain rate ϵ_0 starting from $t = 0$ allows to obtain an analytical solution of the stress, namely

$$\sigma(t) = \eta\epsilon_0 \left[1 - \exp\left(-\frac{t}{\tau_M}\right)\right], \tag{4.8}$$

which shows that in this model, when the strain rate is kept constant in time the stress grow similarly to the Kelvin-Voigt model. However, if the constant strain rate is removed at $t = 0$, the solution becomes

$$\sigma(t) = \eta\epsilon_0 \cdot \exp\left(-\frac{t}{\tau_M}\right), \tag{4.9}$$

meaning that if the material is left undisturbed the stress will decrease exponentially until the samples relaxes completely. A zero strain rate does not mean that the strains are inexistent, but that they are constant in time, meaning that the material is relaxing in a deformed state. This is graphically represented in Figure 4.4b.



(a) Representation of the Maxwell model

(b) Stress relaxation on the Maxwell model.

Figure 4.4: Maxwell model of viscoelasticity. Same as for the Kelvin-Voigt model, the decrease of the stress has an exponential nature governed by τ_M. The strain in this model is a sum of the effects from the elastic and the viscous part.

Maxwell model is used to represent fluids, since the relaxation of stresses in the deformed state explains the property of fluids of adapting to the shape of the container where it is hold. It is worth noting that the definition of τ_M is identical to τ_K, however, their effect have different approaches, so they are defined separately. In the Maxwell model, τ is now known as the *relaxation time*, since it refers to the scale of time in which the material will reduce the stresses until it reaches its relaxed state.

The final model that we will mention is the *Generalized Maxwell model*, also known as Maxwell-Weichart model [47]. The idea behind it is that materials in real life are more complicated that what has been introduced here and some behaviors can be modelled better if different relaxation times are considered. It is defined as a collection in parallel of n Maxwell branches with different values for each spring and dashpot. One extra brach considered is just a spring that represent the elastic behavior in the material leaving a total of $n + 1$ branches. Its representation is shown in Figure 4.5.

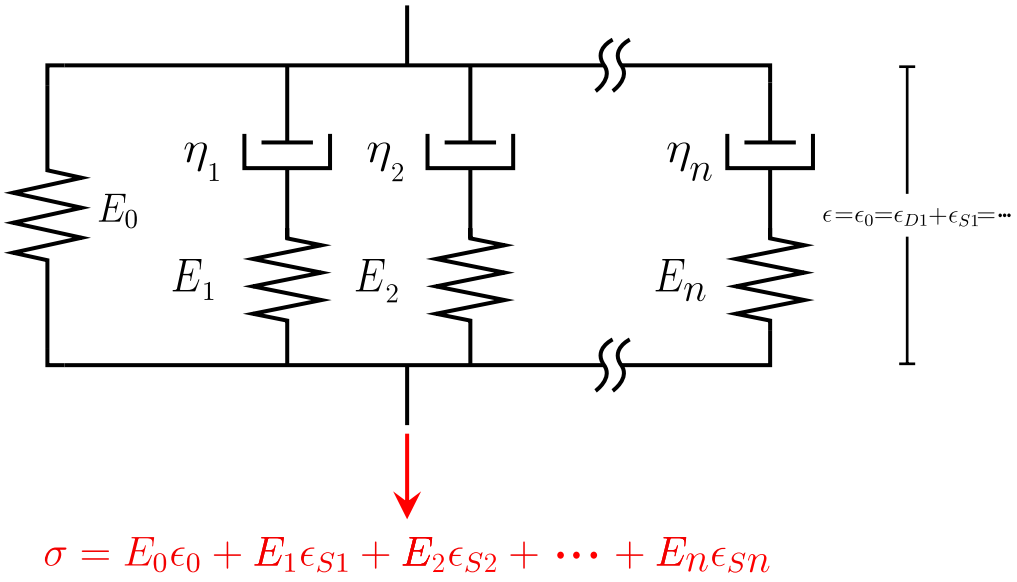


Figure 4.5: Generalized Maxwell model. The strain in each branch is the same and equivalent to the elastic strain from the first branch. Every other branch has a strain value that is a combination of its own elastic and viscous behavior. The stress is just a sum of the stresses of all the springs.

In this work, all the examples will be applied to the case where $n = 1$. This simplified model is known as the *Standard linear solid model* or *Zener model*, shown in Figure 4.6. The reason to use this model is because it is the simplest model that include all the phenomena associated to the previous models shown. It also is one of the simplest representation for viscoelastic solids.

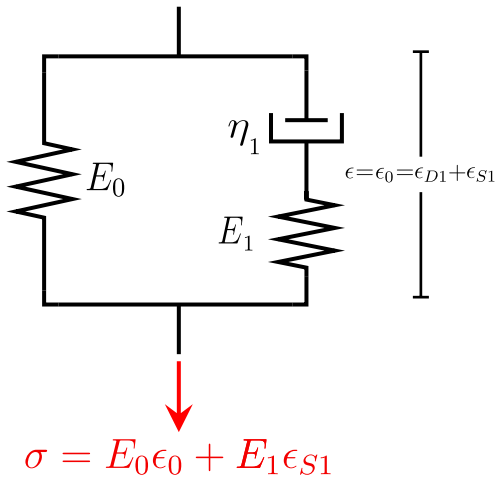


Figure 4.6: Representation of the Zener model.

The Zener model has the following properties:

- The total stress is a sum of the stresses of each branch, i.e., $\sigma = \sigma_{S0} + \sigma_M$.
- The stress of the spring and dashpot in the Maxwell branch are equal ($\sigma_{S1} = \sigma_{D1}$).

- The strain on both branches is the same, so the strain is equal to the elastic strain ($\epsilon = \epsilon_M = \epsilon_{S0}$).
- The strain on the Maxwell branch is the sum of the elements, i.e., $\epsilon_M = \epsilon_{S1} + \epsilon_D$.

We can apply this properties to obtain an equation that can be solved. We start by taking the total strain and stress of the system:

$$\sigma = \sigma_{S0} + \sigma_M, \tag{4.10a}$$
$$\dot{\epsilon} = \dot{\epsilon}_{S1} + \dot{\epsilon}_D, \tag{4.10b}$$

where in Equation 4.10b we consider that the total strain is equivalent to the strain in the Maxwell branch and the relationship holds after applying the time derivatives. Since the stresses of the spring and dashpot in the Maxwell branch are the same, we can define

$$\dot{\epsilon}_D = \frac{E_1}{\eta}\epsilon_{S1}, \tag{4.11}$$

which we can plug into Equation 4.10b. If we rename $\epsilon_{S1} \equiv q$ and remember the definition of relaxation time as $\tau = \eta/E_1$, we can write the expression as

$$\dot{\epsilon}(t) = \dot{q}(t) + \frac{1}{\tau}q(t). \tag{4.12}$$

The expression in Equation 4.12 is useful since the total strain is defined in terms of the partial strains in the Maxwell branch. Considering also that the stress in a spring can be described by Equation 4.1a and that the stress in each branch can be defined only by their respective springs, we can rewrite Equation 4.10a as

$$\sigma(t) = E_0\epsilon(t) + E_1 q(t), \tag{4.13}$$

where we take advantage of the definition of $q(t)$. Both Equation 4.12 and Equation 4.13 can be used in numerical schemes to obtain the solution with techniques like FEM, and is indeed how the synthetic data for this chapter is generated. The procedure will be explained in detail in Section 4.2.4.

A more general relation for the Zener model can be obtained if we rewrite Equation 4.10b as

$$\dot{\epsilon} = \frac{\dot{\sigma}_{S1}}{E_1} + \frac{\sigma_D}{\eta}. \tag{4.14}$$

Considering that $\sigma_{S1} = \sigma_D = \sigma_M$, we can express Equation 4.10a and its derivative as

$$\begin{aligned} \sigma_M &= \sigma - E_0\epsilon, \\ \dot{\sigma}_M &= \dot{\sigma} - E_0\dot{\epsilon}, \end{aligned} \tag{4.15}$$

which can be directly plugged in Equation 4.14 to obtain the general differential equation for the model:

$$(E_0 + E_1)\ \dot{\epsilon}(t) = \frac{1}{\tau}\left[\tau\dot{\sigma}(t) + \sigma(t) - E_0\epsilon(t)\right]. \tag{4.16}$$

This equation does not have much use for this work other than verification of the implementation, but it is left as it is relevant to the subject.

Differential representation

Until now, we have used simplified models based on internal variables of the material to represent the constitutive relations of some of the viscoelastic materials. However, these representations are limited to each particular configuration of the system of springs and dashpots used, so we need to find expressions for each one of them. In this sense, we need to define a general relation of strain and stress. This is achieved by defining a general differential equation of the form [45]

$$\begin{aligned}\left(1 + \alpha_1\frac{\partial}{\partial t} + \alpha_2\frac{\partial^2}{\partial t^2} + \cdots + \alpha_n\frac{\partial^n}{\partial t^n}\right)\sigma(t) \\ = \left(\beta_0 + \beta_1\frac{\partial}{\partial t} + \beta_2\frac{\partial^2}{\partial t^2} + \cdots + \beta_m\frac{\partial^m}{\partial t^m}\right)\epsilon(t),\end{aligned} \tag{4.17}$$

where $n = m$ or $n = m - 1$. Up until now, we have define scalar models that represent the behavior of one generic element. Equation 4.17 is a generalization of the constitutive relation of viscoelastic materials, meaning that no matter the dimensionality of the problem, the equation still stands. In our case, this equation is valid for both the scalar case (trusses), as well as a generalized 3D problem.

All the cases shown earlier can be derived from Equation 4.17. For example, if every parameter in Equation 4.17 is taken as zero with the exception of β_0, we just revert back to the case of linear elasticity (i.e., $\sigma = \beta_0\epsilon$). The same can be performed if we leave now only β_1 as the non-zero parameter, which gives the relation for newtonian fluids (i.e., $\sigma = \beta_1\frac{\partial\epsilon}{\partial t}$). Both of these equations are equivalents to the ones described in Equation 4.1a and 4.1b respectively.

From the presented viscoelastic formulations, the Kelvin-Voigt model can be retrieved if we set $\beta_0 = E$ and $\beta_1 = \eta$, with every other term equaling zero. In a similar vein, the Maxwell representation comes from setting $\beta_1 = \eta$ and $\alpha_1 = \tau_M$, while the Zener model becomes more complicated, but can be obtained by rearranging terms in Equation 4.16. More complex models such as the *Jeffrey's model* [48] can be used by involving second derivatives, but those models are out of the scope of this work.

Integral representation

Every representation of viscoelastic materials that we have presented depends on knowing certain parameters that allows us to describe the behavior. The models using internal variables are simple to use but the lack of analytical solutions for specific cases can be a problem. In a same way, the differential formulation allows us to obtain a more general representation of all viscoelastic behaviors, but we give up some simplicity.

We can abstract even more this generalization, by writing the relation of stresses against strain history as a time integral of a modulus multiplied by the strain rate applied on the material. Since all the history of the strains is relevant, the integration domain goes from negative infinity to the current time, t. This equation can be written as

$$\sigma(t) = \int_{-\infty}^{t} \boldsymbol{D}(t-\tau)\dot{\epsilon}(\tau)\ d\tau, \tag{4.18}$$

where $\boldsymbol{D}$ is what is known as the relaxation modulus, it depends on each material and its expression is to be identified.

The use of this equations is not an optimal way of solving the problems, since they just work as a generic definition for a full range of phenomena encased into the viscoelasticity term. If we have access the expression of the relaxation modulus, we can obtain a very accurate solution for the viscoelastic problem, but in most of the cases these moduli are unknown or impossible to represent. It is because of this that some simplifications are made to understand better the behavior and to be able to model the materials. In this work we stick to samples defined with simplified models, since its simplicity and parametrization allows us to study better how different ranges of values will behave when testing DDI.

4.1.2 Viscoelastic response for oscillatory movement

A popular way to test viscoelastic models is through small oscillations. As it is known, oscillatory movement is defined by functions with sines and cosines. To derive the formulation, a cosine strain is defined to be implemented into the equations for the previous models. Considering an oscillatory shear strain of the form

$$\gamma(t) = \gamma_0 \cos(\omega t), \tag{4.19}$$

the time derivative of this oscillatory strain is

$$\dot{\gamma}(t) = -\omega\gamma_0 \sin(\omega t). \tag{4.20}$$

This expression for the strain and strain rate can be substituted in any of the equations for the models presented before in order to obtain a solution. In the case of the Kelvin-Voigt model, the solution is rather trivial. Applying both γ and $\dot{\gamma}$ in Equation 4.4 we obtain

$$\begin{aligned} \sigma(t) &= E\gamma(t) + \eta\dot{\gamma}(t) \\ &= E\gamma_0 \cos(\omega t) - \eta\omega\gamma_0 \sin(\omega t). \end{aligned} \tag{4.21}$$

If we group the terms associated to the sine and cosine separately, we can obtain a relation of the form

$$\sigma(t) = \gamma_0 G'(\omega)\cos(\omega t) + \gamma_0 G''(\omega)\sin(\omega t), \tag{4.22}$$

and comparing both Equation 4.21 and Equation 4.22 we can then define $G'(\omega)$ and $G''(\omega)$ as

$$G'(\omega) = E, \tag{4.23a}$$
$$G''(\omega) = -\eta\omega. \tag{4.23b}$$

In literature, G' is usually called the *storage modulus* and G'' the *loss modulus*. The storage modulus is the part of the stress that remains *in phase* with the strain (due to both being cosine functions) and it represents the elastic component of the material behavior. On the other hand, the loss modulus deals with the part that is out of phase and it can be interpreted as the part of the stress that it is being relaxed in time. Both moduli correspond to *Fourier transform* [49] coefficients of the viscoelastic modulus $\boldsymbol{D}(t)$, so they can be used to estimate the behavior of the material. The moduli are dependent on the frequency of oscillation, meaning that certain frequencies might trigger a higher viscoelastic response than others.

Applying the same methodology to the Maxwell model, a linear differential equation is obtained with the form

$$\sigma(t) + \tau\dot{\sigma}(t) = -\eta\omega\gamma_0 \sin(\omega t), \tag{4.24}$$

whose solution is

$$\sigma(t) = \frac{\eta\omega^2\tau}{\omega^2\tau^2 + 1}\gamma_0 \cos(\omega t) + \frac{\eta\omega}{\omega^2\tau^2 + 1}\gamma_0 \sin(\omega t), \tag{4.25}$$

yielding the moduli

$$G'(\omega) = \frac{\eta\omega^2\tau}{\omega^2\tau^2 + 1}, \tag{4.26a}$$
$$G''(\omega) = \frac{\eta\omega}{\omega^2\tau^2 + 1}. \tag{4.26b}$$

Finally, for the Zener model the same procedure is followed, giving the expressions for the moduli as

$$G'(\omega) = E_0 + \frac{E_1\omega^2\tau^2}{\omega^2\tau^2 + 1}, \tag{4.27a}$$
$$G''(\omega) = \frac{E_1\omega\tau}{\omega^2\tau^2 + 1}. \tag{4.27b}$$

A useful ratio that will be used here is the coefficient $\tan\delta$, which is defined as

$$\tan\delta = \frac{G''(\omega)}{G'(\omega)}. \tag{4.28}$$

The representation of $\tan\delta$ is convenient, since it shows a ratio of how viscoelastic the material will behave at a certain frequency, since it compares the loss modulus over the storage one. The plots for the moduli, as well as $\tan\delta$, as a function of ωt space for the Zener model are shown in Figure 4.7, where it can be seen where the functions have a higher response. When generating data for the study of the DDI algorithm, the values for the G moduli are going to be considered to determine how viscoelastic the sample should react.

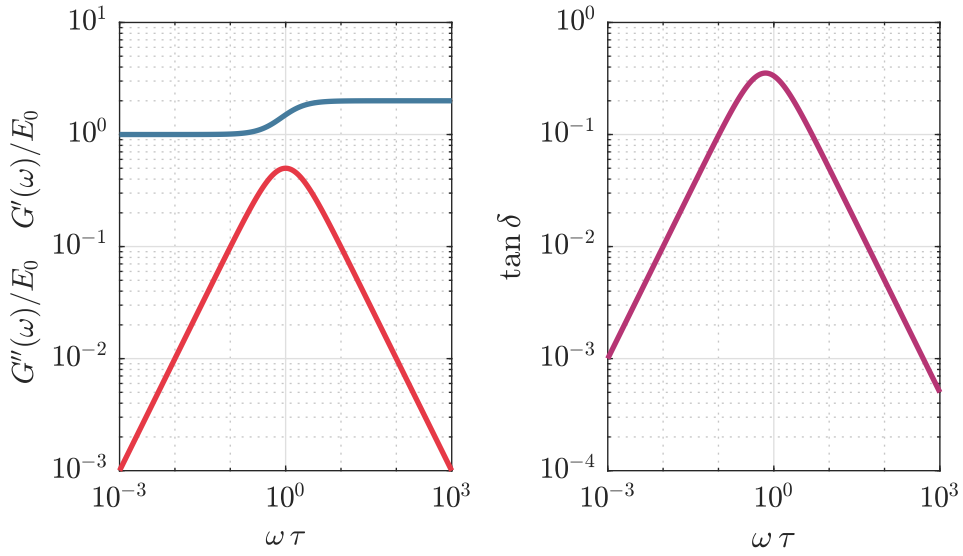


Figure 4.7: Plots for viscoelastic moduli of the Zener model. In here, the blue curve represents the storage modulus and the red one the loss modulus. In this particular example we can see that the storage modulus ranges between 1 and 2, while the loss modulus tends to zero on the extremes of the plot, which are related to very fast or very slow oscillations. The curve on the right represents the ratio between the moduli, determining which zones of the $\omega\tau$ space are more or less viscoelastic.

4.2 Data generation and analysis

For the study of DDI in linear viscoelastic samples, we need to define the samples that we will use, as well as the properties associated to them. These properties are defined by certain parameters that we are interested in and that we will modify, in a similar way as it has been done in Chapter 3. All the data for this chapter will be synthetically generated by solving different cases through the FEM method.

4.2.1 Model selection

The material considered will be a linear viscoelastic material based on the Zener model, as defined in Equation 4.13 with $q_1(t)$ taken as

$$q_1(t) = \int_{-\infty}^{t} \exp\left[\frac{-(t-t')}{\tau_1}\right] \dot{\epsilon}(t)\; dt', \tag{4.29}$$

and shown in Figure 4.6. The elastic branch of the model is set to one to standardize the testing, so we would always define the stiffness of the maxwell branch in terms of a

ratio. The same convention is applied to the relaxation time, since setting $\tau = 1$ allows us to modify the experiments around the sample by making them longer or shorter, instead of defining different samples with different values of it. We do this since we would like to define a methodology that could be applied in real life, where it is easier to adapt the parameters of the machine rather than the ones in a sample.

When using a 3D formulation, we consider that the material will be isotropic, with a Poisson ratio taken as $\nu = 0.3$, to keep it consistent with previous chapters. This ratio is applied for both springs of the Zener model.

To study the behavior of the algorithm in different setting, we change the values of the parameters that define the Zener model. The main parameters that will be researched are:

- Viscoelasticity ratio (E_v): In the Zener model there are two moduli representing the springs in the elastic branch and in the Maxwell branch, as seen in Figure 4.6. It is clear that removing each one of the branches will yield either an elastic material or a pure Maxwell viscoelastic one. In order to study how this affects the algorithm, the ratio E_v is proposed, which is defined as

$$E_v = \frac{E_1}{E_0}. \tag{4.30}$$

  Since E_0 is the unity, what it is effectively controlled when changing the parameter is the stiffness of the spring in the Maxwell. For studying the effect, the parameter is varied from zero to infinity, which yields three main possibilities:

  - $E_v \to 0$: If the value of E_1 tends to zero the maxwell branch has no effect in the model, meaning that the material effectively behaves as a normal elastic material.
  - $E_v \to \infty$: If the value of E_1 is much bigger than E_0 the effect of the purely elastic branch is reduced and can be neglected, meaning that we deal with a material that is Maxwell viscoelastic.
  - $E_v \approx 1$: If both stiffness moduli are similar, then the effect is a combination of Kelvin-Voigt and Maxwell model that was intended in the definition of the Zener model.

  In the 3D cases, where the stiffness modulus is defined by a tensor, the ratio only affects the value of the Young's modulus constant and not the tensor itself. In this sense, the ratio is not defined as a matrix multiplication but just as a comparison of the E values.

- Time scale for experimentation (D_v): As explained in [45], it can be expected that every material in the universe can behave as a viscoelastic material, if the amount of time it is given to deform under a load is enough. Viscoelastic behavior is highly dependent in the velocity in which the solicitations are applied in the material, which comes directly from the definition of the constitutive law in Equation 4.18. In that sense, the parameter D_v is defined as a relation between the relaxation

time and the time that the experimentation takes to be performed. It is defined as

$$D_v = \tau\dot{\epsilon}. \tag{4.31}$$

Similarly to E_v, with the relaxation time conveniently set to the unity we only need to effectively change the length of the experimentation by speeding up or slowing down the deformation, i.e., modifying the strain rate, which is a measurable property. Three cases can be also be described here:

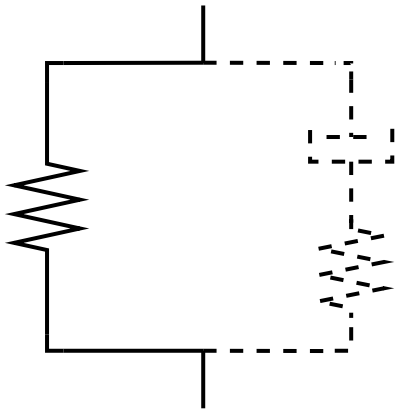
- $D_v \to 0$: If the strain rate goes to zero, meaning a very slow deformation in the sample, the dashpot in the model will have enough time to develop the viscous behavior, so we end up having a viscoelastic behavior.
- $D_v \to \infty$: Having the strain rate going to infinity means that the experiment will be instantaneous, which translates to the viscoelastic behavior being suppressed. In this case we should again revert to the normal linear elastic case.
- $D_v \approx 1$: If the strain rate is in the same order as the relaxation time we should see a moderate viscoelastic behavior.

As can be expected, a constant strain rate is difficult to define for most cases. In this case, the parameter D_v is mostly reserved for using in force profiles such as extension test, where the strain rate is usually kept constant. For other similar cases such as the creep test, where the force profile is constant but not the strain rate, an alternative definition for the parameter can be defined as

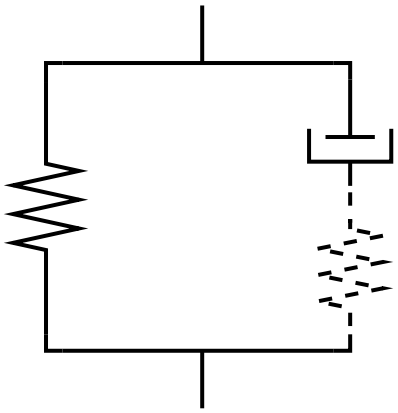
$$D_v = \frac{\tau}{t_{exp}}, \tag{4.32}$$

where t_{exp} represents the length of the experiment, which can be defined a priori. In general, D_v is defined in combination with E_v. Considering different values for stiffnesses and strain rates creates different cases that can be analyzed, which are presented in Figure 4.8.

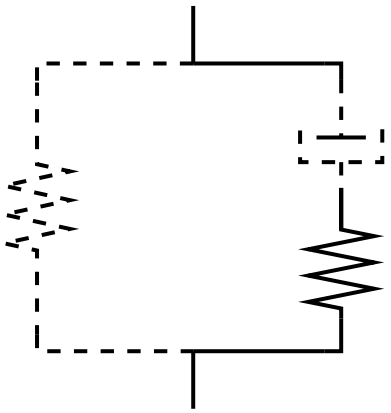
- Frequency of oscillation (ωt): In the case of oscillatory motion the effects of time scale cannot be defined easily in the same fashion as D_v, since questions such as how to define an oscillatory strain rate in a constant fashion arise. In this case, the definition of viscoelasticity for small oscillations becomes handy, since a similar analysis can be performed. As was mentioned in Section 4.1.2, in the case of oscillatory motion a storage (G') and a loss (G'') modulus can be defined as a function of ω, where the first one represents the elastic part and the second the loss of stress through relaxation. The idea is to choose different values of the spectrum that will yields more elastic or more viscoelastic behaviors and compare the performance.



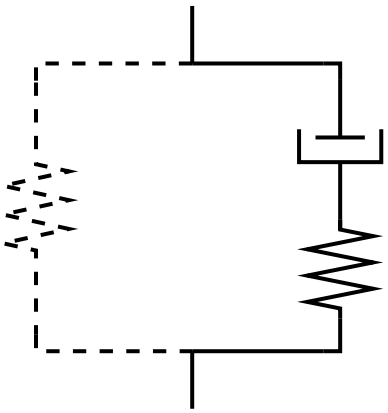
(a) Case for $E_v \to 0$ and $D_v \to \infty$.



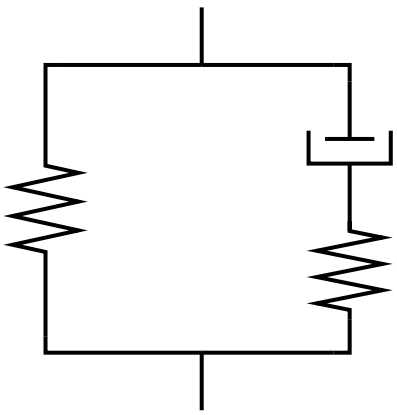
(b) Case for $E_v \to 0$ and $D_v \to 0$.



(c) Case for $E_v \to \infty$ and $D_v \to \infty$.



(d) Case for $E_v \to \infty$ and $D_v \to 0$.



(e) Case for $E_v \approx 1$ and $D_v \approx 1$.

Figure 4.8: Equivalent models for the different parametrizations of the Zener model. Whenever we consider $E_v \to 0$ it is equivalent to ignoring the spring in the Maxwell branch, while $E_v \to \infty$ represents the opposite case, where the elastic branch is ignored. In an analogous case, $D_v \to \infty$ suppresses the effect of the dashpot due to how slow the displacement is applied.

4.2.2 Sample description

All the cases that will be studied here are performed in the same sample mesh. The mesh, shown in Figure 4.9, follows the same pattern as the examples shown in previous chapters. We consider a rectangular sample made out of a single isotropic material with holes in it to avoid generating homogeneous strains through the sample. The type of elements considered vary between tests, but is either linear triangles with one integration point in their centers or linear bar elements formed by the edges of the triangles. Some examples will be presented as trusses due to its simplicity, since its scalar representation allows us to see better the relation between strain and stress, although it is implied that the results will work accordingly in the generalized 3D case.

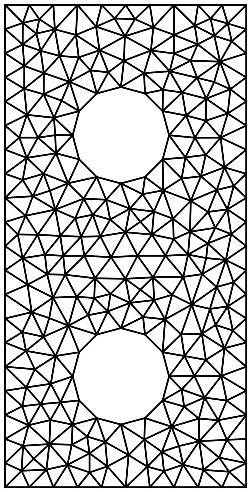


Figure 4.9: Mesh used for testing viscoelastic formulation of DDI.

In all the cases the samples are fixed in both the lower and upper boundaries. A vertical deformation profile is applied, always to the top boundary of the sample. This deformation profile is applied as a block to all the nodes, meaning that the nodes on the top boundary are also constrained horizontally, as they would be in a real test performed with a clamp. The lower remains fixed during all the testing. The force profile used depends on the case studied and they will be defined for each example.

A summary of the properties of the mesh is shown in Table 4.1.

Table 4.1: Properties of the mesh used for viscoelasticity.

Property	Truss case	General 3D case
# of nodes	270	270
# of elements	729	540

4.2.3 Testing setting

From the theory that has been presented, we know that viscoelastic behavior has a high dependence on the solicitation applied to the sample, particularly associated to the time scale of the problem and the time that the material has to recover from this changes. It is because of this that the deformation profiles are chosen to represent feasible cases

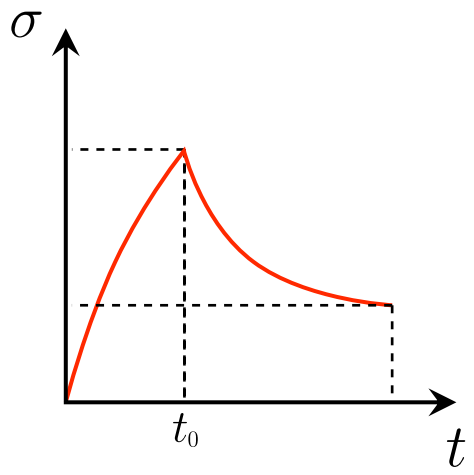
in real testing that are also standard tests performed in this kind of materials. In this chapter, the following four profiles will be used in different examples:

- Extension-Relaxation test: The sample is subjected to a constant strain for a certain amount of seconds and then the deformation is left fixed to observe the relaxation effect of viscoelastic materials. This is done by subjecting the mesh to a vertical deformation on the upper edge of the sample, while keeping the lower end fixed. The expected behavior is shown in Figure 4.10a: the constant increase of strain produces an increase of stress that stabilizes after a time related to the parameter τ. When the strain is kept constant, the stress decreases exponentially until it reaches a residual level dictated by the stiffness of the spring in the Maxwell branch. For this force profile, we consider t_{exp} the time that it takes to reach the maximum strain, just before it is set constant.

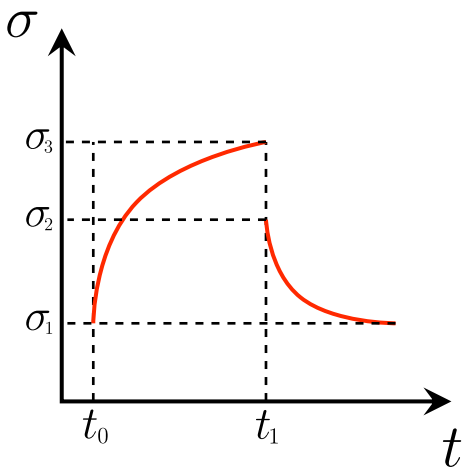
- Creep test: Similar to the previous case, it is now a constant stress that is applied to the sample, achieved by applying a constant force in the upper end of the mesh while keeping the bottom clamped. The force is applied from a time $t_0 \neq 0$ until an arbitrary time t_1, where the sample is released. Before t_1 the strain in the sample increases logarithmically, while the inverse effect happens in the relaxation phase. This is depicted in Figure 4.10b. In here, t_{exp} is the time during which the force is applied, encompassed between t_0 and t_1. The creep test is shown here due to its importance in the field of viscoelasticity, but it will not be applied to any example shown in this chapter due to the difficulty of parametrization.

- Oscillatory motion: It allows to obtain information of the storage and loss moduli of the material according to what was shown in Section 4.1.2. For these tests, a cosine wave function is applied in the upper edge while the lower side is clamped. The oscillation is applied in a pre-stressed setting, meaning that even considering the oscillation of the upper edge, the membrane will never be in compression. Another version of this test is performed considering a wave-like function that is defined as a sum of two cosines. This is done to avoid repetition in the strains of the sample, which could affect the accuracy of the DDI algorithm by repetition of mechanical points. The schematics for both cases are seen in Figure 4.10c. If we consider a constant sinusoidal function, we can define t_{exp} as the time it takes to perform one oscillation, however, for combinations of sines this definition loses meaning, and a study based on the oscillatory formulation is preferred.

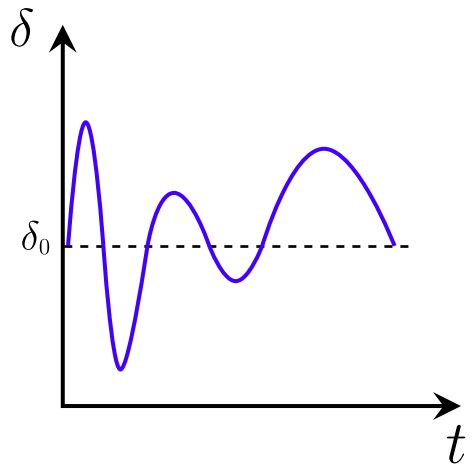
- Cubic cosine function: The motion prescribed is the same setting as for the oscillatory motion, but instead of considering a cosine wave function, its cube is taken. This is specifically done to have discontinuities in the strain rate, so the effect can be studied in DDI. The schematic is shown in Figure 4.10d. In this case the same rules of oscillatory motion apply when defining t_{exp}.



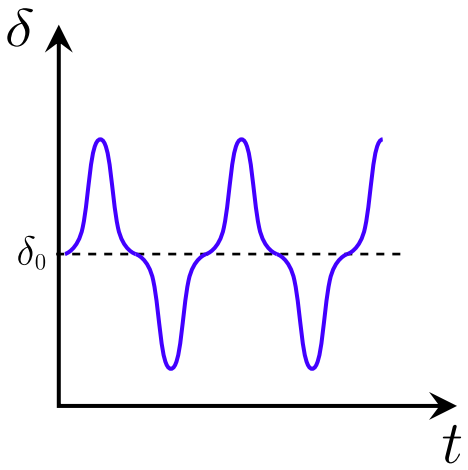
(a) Relaxation test, stress profile.



(b) Creep test, stress profile.



(c) Oscillatory test, deformation profile.



(d) Cubic oscillatory function, deformation profile.

Figure 4.10: Force profiles for viscoelastic tests.

4.2.4 Finite elements procedure for linear viscoelasticity

For testing the new methodology of DDI we need to have access to a database of strains in a similar fashion as the previous chapters. Because of this is that the same approach of synthetically generated data through the finite elements method was implemented. Since the FEM formulation introduced in Section 2.1 was directed at linear elastic materials, we redefine it here with linear viscoelasticity in mind.

As mentioned in Section 4.1.1, we work with the Zener model. The implementation of a numerical scheme for this particular model is covered thoroughly in [50], so the main points of the procedure will be outlined here.

The starting point for the numerical scheme is Equation 4.12. The treatment of the time derivatives is done using a forward Euler scheme, which renders the equation as

$$\frac{\epsilon_{n+1} - \epsilon_n}{\Delta t} = \frac{1}{\tau} q_n + \frac{q_{n+1} - q_n}{\Delta t}. \tag{4.33}$$

Reordering the equation we obtain the explicit relation for q_{n+1}

$$q_{n+1} = \left(1 - \frac{\Delta t}{\tau}\right) q_n + \epsilon_{n+1} - \epsilon_n, \tag{4.34}$$

which works if we assume that we know each value of ϵ_i and q_i until timestep n. A generalization of this can be done using the midpoint rule [50], which renders the relation as

$$\left(1 + \frac{\gamma \Delta t}{\tau}\right) q_{n+1} = \left(1 - \frac{(1-\gamma)\Delta t}{\tau}\right) q_n + \epsilon_{n+1} - \epsilon_n, \tag{4.35}$$

where γ is a parameter that has a value between 0 and 1 and allows to shift between a forward or backward Euler scheme. Equation 4.34 is equivalent to Equation 4.35 when $\gamma = 0$ is taken. In general, 0 will be considered to obtain the solutions due to stability of the algorithm.

The generalized equation for FEM was introduced in Equation 2.18. In the case of viscoelasticity we need to define the stress differently, and we do it Equation 4.13, which adapted to the numerical scheme reads as

$$\sigma_{n+1} = E_0 \epsilon_{n+1} + E_1 q_{n+1}, \tag{4.36}$$

which plugged in Equation 2.19b translates into

$$\boldsymbol{P}(\sigma) = \sum_e \int_{\Omega_e} \delta \boldsymbol{B}^T \left[\boldsymbol{D}_0 \boldsymbol{\epsilon}_{n+1} + \boldsymbol{D}_1 \boldsymbol{q}_{n+1}\right] d\Omega. \tag{4.37}$$

In here, the notation is changed for the more generalized version of the algorithm, with the strain and stress not being necessarily scalar, and replacing the stiffness of the springs for the more generalize tensorial moduli, $\boldsymbol{D}$. Equation 4.37 can be expanded by replacing $\boldsymbol{q}_{n+1}$ with Equation 4.35, as well as remembering the compatibility condition of $\boldsymbol{\epsilon} = \boldsymbol{B}\boldsymbol{u}$. This yields to

$$\boldsymbol{P}(\sigma) = \int \boldsymbol{B}^T \left[\boldsymbol{D}_0 \boldsymbol{B} \boldsymbol{u}_{n+1} + \boldsymbol{D}_1 \underbrace{\frac{1}{1 + \frac{\gamma \Delta t}{\tau}}}_{\mu_t} \left(\overbrace{\left(1 - \frac{(1-\gamma)\Delta t}{\tau}\right)}^{\alpha_t} \boldsymbol{q}_n + \boldsymbol{B}\boldsymbol{u}_{n+1} - \boldsymbol{B}\boldsymbol{u}_n \right) \right]. \tag{4.38}$$

For Equation 4.38 the formalities of the integral have been left out because of space concerns, but it is noted that the integration is performed in the domain Ω_e and it is a summation of integrals over all the elements in the mesh.

The idea behind Equation 4.38 is to organize the terms into a linear system of equations in the same manner as it has been done in Equation 2.18, that will allow to obtain the values for the displacement at each timestep. Reordering the terms and

plugging it into Equation 2.18:

$$\underbrace{\int \boldsymbol{B}^T(\boldsymbol{D}_0 + \mu_t \boldsymbol{D}_1)\boldsymbol{B}}_{\boldsymbol{K}_T} \boldsymbol{u}_{n+1} = \boldsymbol{f}_{n+1} - \mu_t \alpha_t \underbrace{\int \boldsymbol{B}^T \boldsymbol{D}_1}_{\boldsymbol{S}_1} \boldsymbol{q}_n + \mu_t \underbrace{\int \boldsymbol{B}^T \boldsymbol{D}_1 \boldsymbol{B}}_{K_1} \boldsymbol{u}_n. \quad (4.39)$$

Using the naming conventions indicated, Equation 4.39 can be finally written as

$$\boldsymbol{K}_T \boldsymbol{u}_{n+1} = \boldsymbol{f}_{n+1} - \mu_t \alpha_t \boldsymbol{S}_1 \boldsymbol{q}_n + \mu_t \boldsymbol{K}_1 \boldsymbol{u}_n, \quad (4.40)$$

where the new values of displacement are computed based on the values of the previous timestep. The values of $\boldsymbol{\epsilon}_{n+1}$ can be obtain by using the compatibility condition, while the values of $\boldsymbol{q}_{n+1}$ and $\boldsymbol{\sigma}_{n+1}$ are obtained from Equation 4.35 and Equation 4.36.

As a final note on the method for this particular model, it is worth noting that this approach can be easily extended to a generalized Maxwell model with n branches due to the superposition principle. In this case Equation 4.40 looks like

$$\boldsymbol{K}_T \boldsymbol{u}_{n+1} = \boldsymbol{f}_{n+1} - \sum_i \left(\mu_i \alpha_i \boldsymbol{S}_i \boldsymbol{q}_n^{(i)} + \mu_i \boldsymbol{K}_i \boldsymbol{u}_n \right), \quad (4.41)$$

with $\boldsymbol{K}_T = \boldsymbol{K}_0 + \sum_i \mu_i \boldsymbol{K}_i$ an every other term including i being the values of the respective branches. It is then necessary to consider n different values of relaxation times, as well as n different computations of $\boldsymbol{q}^{(i)}$ for each branch.

4.3 DDI algorithm for linear viscoelasticity

If we analyze the FEM formulation for the Zener viscoelastic model, we can see that the DDI formulation is potentially deficient for this kind of materials, due to the way in which the data is processed. In the original formulation for elastic materials we have always considered the input data as a collection of strain fields associated to one particular deformed state of the sample. Each one of these snapshots is instantaneous and independent from one another, which works fine when we deal with elasticity. However, we can see from the formulation of Section 4.2.4 that the stresses of each deformed states are dependent on the previous timesteps, which is what has been established from the theory of viscoelastic behavior. In this setting we need to rethink the concept of snapshots and how it is used in the DDI algorithm, since now we have to consider a time relationship between them.

4.3.1 Modifications to DDI algorithm

Whenever data is collected, a temporal progression needs to be kept in order to be able to associate each instantaneous snapshot with its previous one. In that regard, snapshots now do not represent just a collection of strains, but also a progression for all the elements in the mesh (i.e. a strain history). Taking this into consideration, we propose a new formulation in which different mechanical states represent a progression in time of the strain and stress of each element in the sample.

With this in mind, we need to modify the algorithm to fit this new strategy. If we revisit Equation 2.33, we can notice that the definition of the DDI problem is quite generic. We attempt to minimize a distance between mechanical and material points subjected to equilibrium conditions. However, the way in which Equation 2.33 is written it is particularly defined for elasticity: when we minimize the distance we do by seeking points in the phase space, which are defined by the instantaneous strain and stresses that belong to the elements in all the samples. This approach is expected given that for elastic materials, $\sigma = f(\epsilon)$.

Looking at the definition for stresses that were proposed in Section 4.1.1, we now that the stress for a Zener model can be of the form $\sigma = \mathcal{F}(t, \epsilon, q)$ according to Equation 4.13; or of the form $\sigma = \mathcal{F}(t, \epsilon, \dot{\epsilon})$ when defined as an integral according to Equation 4.18. The first definition for stress is beneficial, since it defines the viscoelastic strain in a similar fashion as the elastic stress. Each value of stress σ_t in time is defined by a pair (ϵ_t, q_t), with t being some timestep. Time history is not directly addressed, but in the way the problem is posed, the information of previous steps is contained in $q(t)$. In fact, when performing a separation of the stress, the part defined by $\epsilon(t)$ remains constant, while $q(t)$ accounts for the relaxation effects in the material. The problem with this formulation comes from the fact that there is no way to measure the value of q at any given time since it is an internal variable, opposite to ϵ which can be captured by imaging techniques.

If we revisit again Equation 4.18, we have an equation that depends on the continuous full time history, however, data collection is discrete. If we perform a numerical scheme for this formulation we obtain

$$\begin{aligned} \sigma(t) &= \int_0^t \boldsymbol{D}(t-\tau)\dot{\epsilon}(\tau)\, d\tau \\ \sigma_t &= \Delta t \sum_{i=1}^{t} \boldsymbol{D}_{t-i}\dot{\epsilon}_i \\ \sigma_t &= \sum_{i=1}^{t} \boldsymbol{D}_{n-i}(\epsilon_i - \epsilon_{i-1}). \end{aligned} \tag{4.42}$$

For this expression we have considered that the integration is performed at time $\tau = 0$ to be able to perform the summation, which is the case in most of the experiments. By inspecting the expression obtained, we can see that sigma can be presented as a function of previous strains in order to account for the time, meaning

$$\sigma_t = \mathcal{F}(\epsilon_t, \epsilon_t - 1, \ldots, \epsilon_0). \tag{4.43}$$

This becomes the basis of how viscoelasticity will be treated in this chapter. We use a mixed formulation based on internal variables to generate the data with FEM, with the integral formulation used to define the phase space in DDI. In this setting, the phase space is now an $n+2$ dimensional space where for each point we have a full time history associated to it, with n being the amount of timesteps of the experiment

performed. The $\mathbb{C}$-norm that we use to measure the distances is now defined as

$$||([\boldsymbol{\epsilon}_e^t, \boldsymbol{\epsilon}_e^{t-1}, \ldots, \boldsymbol{\epsilon}_e^0], \boldsymbol{\sigma}_e^t)||_{\mathbb{C}}^2 = \frac{1}{2}\left([\boldsymbol{\epsilon}_e^t, \boldsymbol{\epsilon}_e^{t-1}, \ \ldots, \boldsymbol{\epsilon}_e^0] : \mathbb{C}_\epsilon : [\boldsymbol{\epsilon}_e^t, \boldsymbol{\epsilon}_e^{t-1}, \ldots, \boldsymbol{\epsilon}_e^0] + \boldsymbol{\sigma}_e^t : \mathbb{C}_\sigma^{-1} : \boldsymbol{\sigma}_e^t\right), \tag{4.44}$$

which now updates the formulation of the DDI problem to

$$\left(\boldsymbol{\sigma}_e^t, \boldsymbol{\epsilon}_i^*, \ \boldsymbol{\sigma}_i^*, \mathrm{ie}^X\right) = \arg \min_{\boldsymbol{\sigma}_e^t, \boldsymbol{\epsilon}_i^*, \boldsymbol{\sigma}_i^*, \mathrm{ie}^X} \sum_t \sum_e w_e^t ||([\boldsymbol{\epsilon}_e^t, \boldsymbol{\epsilon}_e^{t-1}, \ldots, \boldsymbol{\epsilon}_e^0] - \boldsymbol{\epsilon}_{\mathrm{ie}^X}^*, \boldsymbol{\sigma}_e^t - \boldsymbol{\sigma}_{\mathrm{ie}^X}^*)||_{\mathbb{C}}^2. \tag{4.45}$$

In this renewed formulation the definition of $\mathbb{C}$ and $\boldsymbol{\epsilon}_{\mathrm{ie}^X}^*$ is changed to be consistent with the dimensions of the different matrices. Since the stress remains the same, we have that, in practice $\mathbb{C}_\sigma = \mathbb{C}$. However, in our new phase space we do not use the strain but rather the strain history, that for algorithmic purposes we define as a matrix of horizontally concatenated strains, which means that $\mathbb{C}_\epsilon$ has to have adequate dimension. In the simplest of cases, $\mathbb{C}_\epsilon$ can be defined as a block diagonal matrix with $\mathbb{C}$ as each one of the main blocks, repeated $n+1$ times, e.g.,

$$\mathbb{C}_\epsilon = \underbrace{\begin{bmatrix} \mathbb{C} & & & \\ & \mathbb{C} & & \\ & & \ddots & \\ & & & \mathbb{C} \end{bmatrix}}_{(n+1)\times(n+1)}, \tag{4.46}$$

but the definition could vary if needed (for example, considering decreasing or increasing values of $\mathbb{C}$). We have decided to use this formulation since it relies on setting just one value for $\mathbb{C}$, and different schemes will not be studied in this work. In a similar way, $\boldsymbol{\epsilon}_{\mathrm{ie}^X}^*$ are points of $n+1$ dimensions that are consistent with the formulation of Equation 4.45.

The solution for this updated DDI algorithm is performed exactly as shown in Algorithm 2.2. Except from the change of dimension, all other aspects of the algorithm remain the same, and the procedure only needs to account for the new dimensionality of the problem, which is done by changing $\mathbb{C}$ for $\mathbb{C}_\epsilon$ in Equation 2.34a.

The main concern with this formulation comes from the fact that the strain history size increases on each timestep, which means that different mechanical states will have incompatible sizes. The solution for this is addressed by *padding* the strain history for earlier timesteps. In this way, every point has the same dimension, but when there are no previous timesteps, the leftover spaces are filled with zeros. This big amount of information for each element in each timestep can grow fast, which has a consequence of reducing the efficiency of the algorithm, making it computationally expensive. As a secondary effect we also have the curse of dimensionality [51, 52], which will be dealt in more detail in Chapter 5, but it basically refers to the effect that increasing the dimension of the problem has, mainly reducing accuracy due to the spacing of data.

4.3.2 Parametrization

As a final step, to study the new formulation of DDI we need to see how the different parameters of the algorithm affect the solutions obtained, in a similar fashion as it was studied in Chapter 3. As explained in Section 2.3, we have the two main parameters of DDI, $\mathbb{C}$ and N^*. We also introduce a new parameter related to the strain history. Since we mentioned that increasing the phase space is expensive in terms of computation, we would like to reduce the sizes of the matrices. The simplest way to do this is to reduce the size of the strain history, with the corresponding loss of accuracy that this implies. For this, we introduce the parameter n_t, which defines the size of the strain history consider (or equivalently, the amount of previous steps). Particularly for the Zener model, we know that the relaxation modulus is exponentially decreasing, meaning that each step further away from the point of interest is less relevant in the calculation of the stress. It is because of this that we can afford to reduce the amount of steps used in the calculation.

To study the algorithm in the different viscoelastic behaviors, the following parameters are analyzed:

- Amount of previous timesteps (n_t): Given the way in which the integration is approximated by the discrete timestep, it is expected that accuracy should increase if we consider a bigger phase space, simply because the integration will be more accurate. However, since DDI works with a clustering scheme, it can also realistically be expected that adding more dimensions to the problem will return less accurate solutions. This should be seen in the results as finding an optimal amount of steps that will yield the best solution.

- Amount of timesteps per relaxation time (h): From the previous point, it is realistic to expect that the amount of timesteps needed for an optimal solution is not the same for every case and it will be highly dependent on the settings of the experiment. One way to standardize the analysis is to consider the ratio h which is defined as

$$h = \frac{n_t}{\tau}, \tag{4.47}$$

  and provides an idea of how much of the strain history is being considered with respect to the relaxation time of the material.

- Energetic norm tensor ($\mathbb{C}$): Same as in the linear elastic case, it is expected that the $\mathbb{C}$ tensor will have an effect in the accuracy of the solution. The effect will be analyzed and discussed, but in the majority of the cases, $\mathbb{C}$ will be consider as infinite, which is equivalent to perform the clustering only considering the distance between strains and not stresses. Algorithmically, this is achieved by limiting the outer iterations of DDI to just one cycle. This is done in accordance to the results by Dalémat in [26], where it was seen that for single material samples the accuracy of DDI tends to stabilize at a lower point when bigger values of $\mathbb{C}$ are considered. This choice is made also to have simpler results: studying many parameters together provides endless combinations, so in the cases where $\mathbb{C}$ is not explicitly analyzed then it is not considered.

Finally, a small mention of the parameter r^*. The effect this value has already been studied by Dalémat in [26, 27], and it has also been address in Chapter 3, with similar results. Because of this, we will not refer to its effects in this chapter. A value in the range of $r^* = 100 - 300$ will be chosen, since it should provide accurate solutions without slowing down the computations or cluttering the images obtained. The effects of the variation of this parameter will not be addressed in this work, although similar results to [26, 27] can be expected due to the mathematical nature of this parameter.

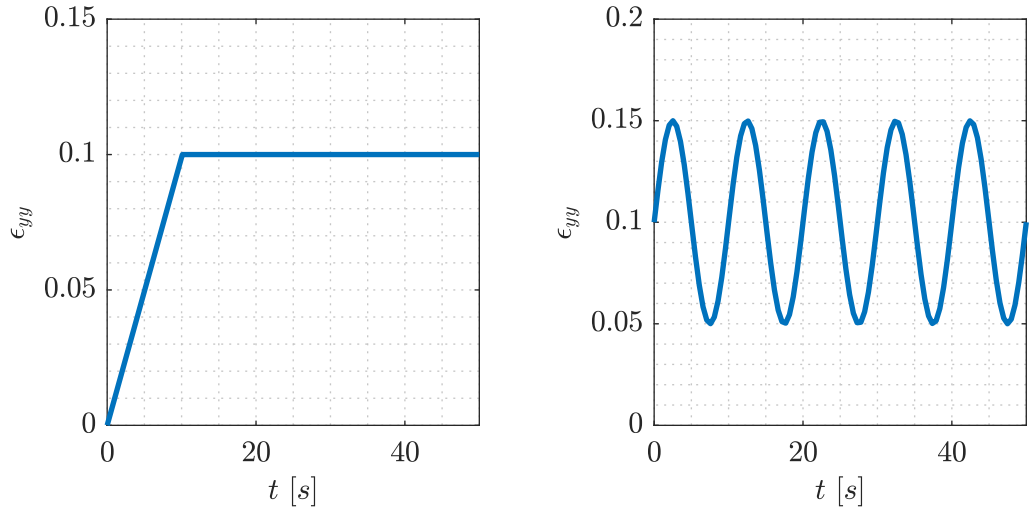
4.4 Analysis of the updated algorithm

To test the efficiency of the modifications to DDI we propose a set of examples designed to study the differences between this new formulation and the original one. The tests are performed taking into consideration the parametrizations mentioned in the previous section, as well as the most common force profiles presented in Figure 4.10.

4.4.1 Comparison between DDI and updated version for a simple case

A first small example is performed just to show how the algorithm works. We consider the sample in the truss case, With a parameter $E_v = 10$, meaning that the stiffness of the viscoelastic branch of the model is ten times higher than the elastic one. This should provide a sample with a rather viscoelastic behavior. The parameter for the time scale is set as $D_v = 1$, which means that the experiment should last around the same amount as the relaxation time, so there is enough time to develop the viscoelastic behavior in the sample.

The displacement profiles applied in the sample are arguably the two most common tests for viscoelasticity: first, a relaxation test is performed, in which the sample is stretched 10% of its length for 1 experiment time ($t_{exp} = 1$ in this case) and then is kept in position until 5 times the experiment time is reached. The second case is an oscillatory test. The sample is initially prestressed by stretching it 10% of its length, assumed to be long enough before the test so the sample has already relaxed. After this, the sample is stretched and compressed 5% of the length around this initial point, meaning that the sample is oscillating between a deformation between 5 to 15%. We consider that one oscillation is performed in one experiment time, so the test is run until 5 experiment times are passed. The displacement profiles are shown in detail in Figure 4.11.

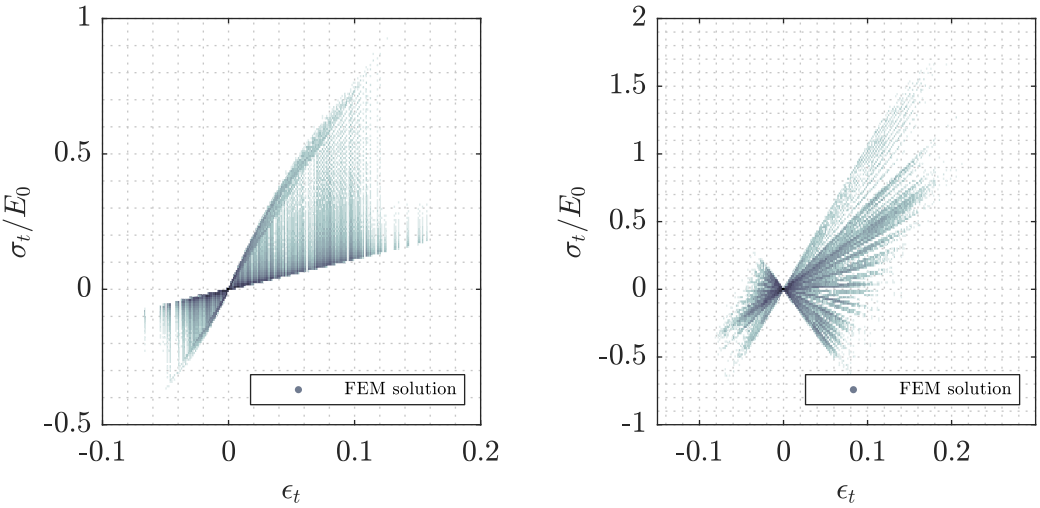


(a) Displacement profile of relaxation test.

(b) Displacement profile of oscillatory test.

Figure 4.11: Displacement profiles for DDI test.

For both tests, the problem is solved using FEM to obtain the strains that are needed in DDI. The advantage of this is that we also have the real solution for the stresses that we can use to compare against the estimations of DDI. If all strains ϵ_t and stresses σ_t (with t representing each timestep in the simulation) are plotted together, we can obtain the $\epsilon_t - \sigma_t$ phase space shown in Figure 4.12. We can see that without a distinction of time progression there is not much use for this kind of graphic, since there is no way that any property of the material can be understood from here. However, when compared to the same results obtained by DDI it gives us an idea of the behavior of the algorithm and the accuracy of the estimations.



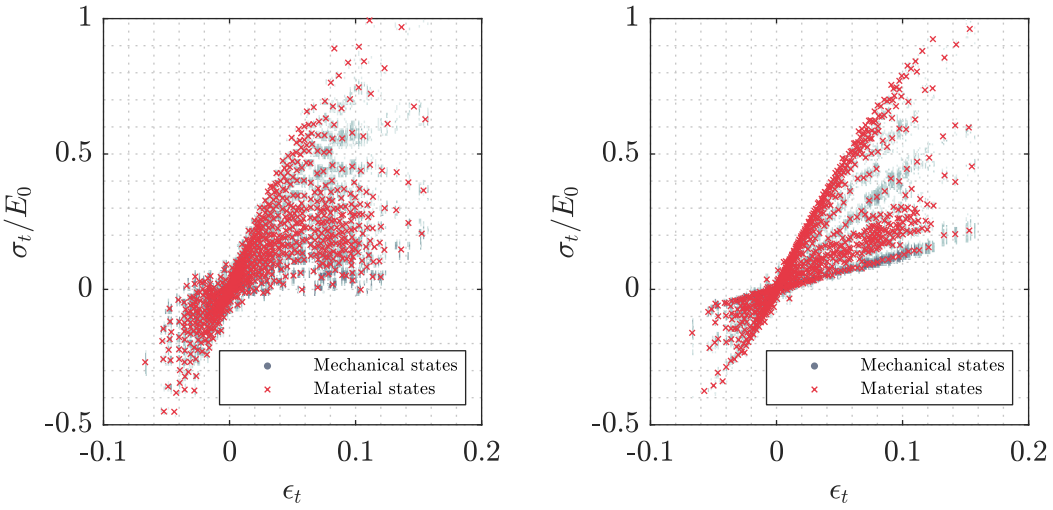
(a) ϵ_t-σ_t space for relaxation test.

(b) ϵ_t-σ_t space for oscillatory test.

Figure 4.12: Strain and stresses for both deformation profiles.

For the relaxation test, DDI is performed in two ways. The first one is using the normal DDI, considering a phase space formed by instantaneous ϵ_t and σ_t, as we have done previously in elastic materials. The second approach is the modified version,

where we assume some sort of temporal connection between snapshots. In this case, we consider a phase space formed by ϵ_t, ϵ_{t-1} and σ_t. The results for both attempts are shown in Figure 4.13. It is clear by the dispersion of the mechanical states that using the modified formulation for DDI visibly improves the estimations of stresses, giving a clearer visualization of the $\epsilon_t - \sigma_t$ space. Since stresses are defined by a full strain history, considering even just one step of past strains helps to predict better the behavior of the stresses. In these images, the material states are left as a reference, but for the time being they serve no purpose other than stabilizing the algorithm.



(a) ϵ_t-σ_t space for original DDI.

(b) ϵ_t-σ_t space for modified DDI.

Figure 4.13: Strain and stresses for both DDI algorithms in the relaxation test.

This can be appreciated better when we visualize the stress over time. Taking a random element in the mesh, we plot the estimated stresses over time and we compare it to the original solution provided by FEM, which is shown in Figure 4.14. In here we see that the results obtained are quite noisy given the nature of the test, but however, we can observe that the modified version of DDI provides a curve that remains closer to the real solution. Since strains are kept constant over time for most of the test, we can also see that DDI tends to *converge* to a final value of stress which cannot be corrected, since we are basically feeding the algorithm the same strain state over and over.

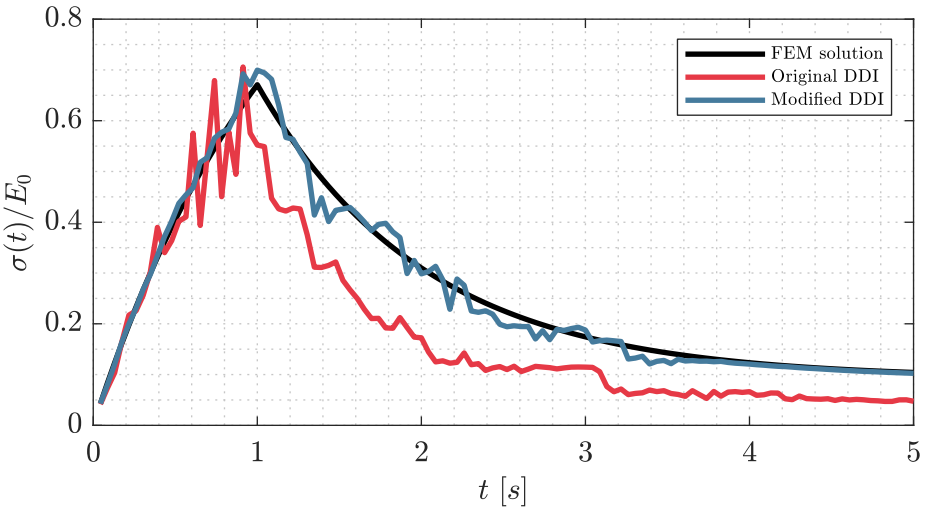
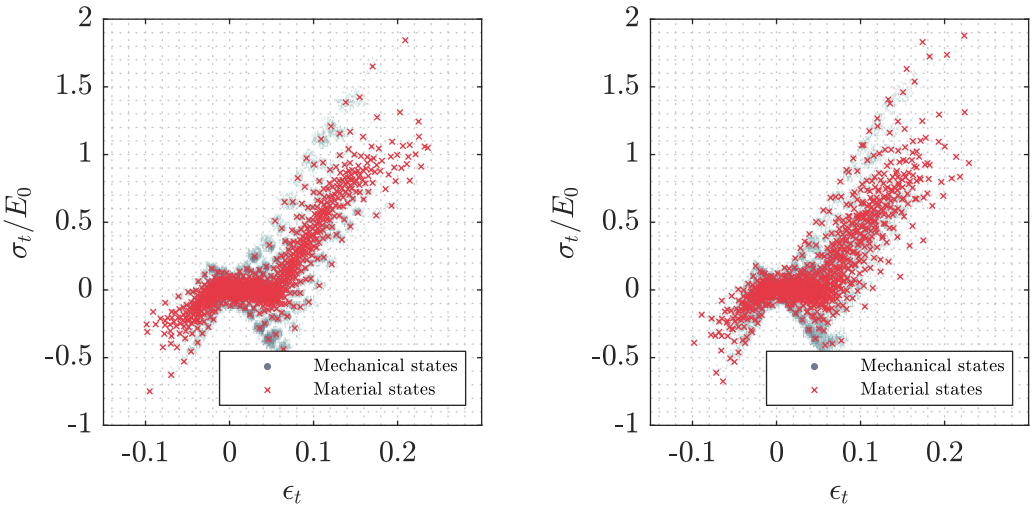


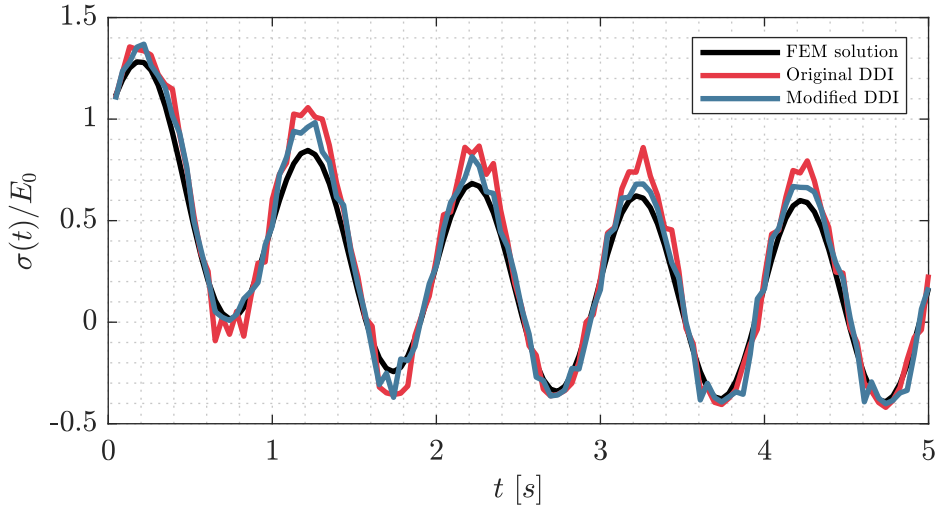
Figure 4.14: Stress over time for an element of the mesh in the relaxation test.

We can repeat the same exercise for the oscillatory test, with the results shown in Figure 4.15. From here we can see a similar trend as the relaxation test, specially when plotting stress over time. However, one thing to notice is the fact that the improvement seems to be less pronounced than the relaxation case. One theory that could explain this behavior is that, since we are using an oscillatory strain profile, the strains are repeated many times, which provides DDI with a better database to estimate the stresses. We have already seen in previous chapters that the most important factor for data-driven methods is the availability of a good database of values on which to base the estimations.



(a) ϵ_t-σ_t space for original DDI.

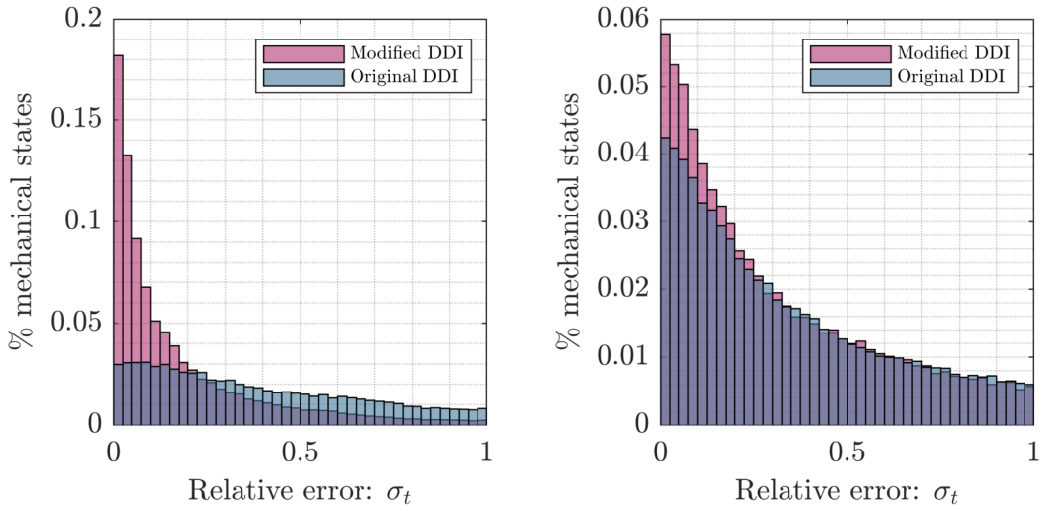
(b) ϵ_t-σ_t space for modified DDI.



(c) Stress over time for an element of the mesh.

Figure 4.15: Results of DDI estimations for oscillatory test.

To compare the accuracy of the two DDI algorithm we go back to the histogram plots introduced in previous chapters. In Figure 4.16 the error distribution of the estimated mechanical stresses is displayed for both algorithms together in both tests. It is clear from this figure that the relaxation case benefits greatly from a modified formulation of DDI. For the oscillatory case we have an improvement, but not a big one. Given the increased computational time that the modified version of DDI can incur if a bigger phase space is used, this is a point that needs to be carefully considered.



(a) Histogram error plot for relaxation test.

(b) Histogram error plot for oscillatory test.

Figure 4.16: Histogram error plots for both cases.

4.4.2 Parameter study and limit cases

The previous example sheds a very good light on the new formulation of DDI, but it consist only on one case that was specifically selected to display these improvement. To test the overall performance of the extended formulation, a parametric study is performed to study where are the advantages of the new DDI and where it falls short.

As before, the stiffness of the elastic branch as well as the relaxation time are fixed and we perform different tests by varying the stiffness of the viscoelastic branch and the length of the experiment time. The displacement profile used for these tests is a cubic oscillatory function, which is applied in a similar fashion: the sample is prestressed by stretching it 10% of its length and when it is relaxed, a cubic sine function ($\sin^3$) is applied to keep the sample stretching in between 5 to 15% of its length. The time of the experiment coincides with one oscillation, but its length varies according to the parameter D_v. The use of a cubic sine is made to avoid having continuous derivatives, which can have a reinforcing effect on the data provided for the DDI algorithm.

Analysis of E_v, D_v and n_t

As it was shown in Figure 4.8, when we talk about the parametrization of the Zener model we cannot make a separation between the stiffnesses of the spring or the time scale of the problem, since they are all closely related to give the properties to the material. Giving a higher stiffness to the viscoelastic branch does not guarantee a viscoelastic behavior since we could have a test that is too fast for the viscoelasticity to develop, and the opposite can also be true. Because of this, the parameters will be studied together.

For all our cases we consider three different values for both E_v and D_v. For E_v, we take values of 0.1 to represent a highly elastic behavior, 10 for a highly viscoelastic behavior and 1 for a balanced case. The same values are considered for D_v, where 0.1 gives us a slow test that will be able to develop viscoelasticity, 10 is a fast case that will make the behavior be closer to elasticity and 1 is the moderate case. Of all these combinations, we will show here the cases for $E_v = 0.1$ and $D_v = 1$, which will

behave as an elastic case; $E_v = D_v = 1$, which is a mild viscoelastic case; and $E_v = 10$ with $D_v = 0.1$ and $D_v = 10$, which should behave as a viscoelastic and an elastic case, respectively.

Finally, to test the accuracy of the modified algorithm we perform DDI considering a different amount of previous steps (n_t). All the examples here are discretized by taking 20 timesteps per experiment time, meaning that all the simulations run for 100 steps. An unmodified DDI is always run ($n_t = 0$) to be compared with the modified versions. The modified algorithms are consider by taking 1, 5, 10 and 20 previous strain steps, which correspond to different intervals based on the experiment time.

- First, we check the elastic case of $E_v = 0.1$ and $D_v = 1$. Considering a low value of E_v basically ignores the contribution of the viscoelastic branch completely, so in this case the value chosen for D_v is irrelevant for viscoelasticity purposes. Performing the DDI on the tested sample gives us the different estimation for the mechanical stresses, whose accuracy is analyzed through the histogram plot from Figure 4.17. Although it has a different look, this plot represents the same information that we have seen in previous histograms, with the difference that now bars are changed for points representing their value. This is done to have a cleaner view of the different methods, since we are now comparing five different cases together. In here we can see that the accuracy of DDI is not very much affected by the amount of steps considered, but there is a slight decrease as more previous steps are added. We attribute this to the fact that elastic materials do not have a need for strain histories since all stresses are instantaneous. Adding a history increases the complexity of the problem without providing any benefit to the calculations, which in turn is detrimental for the accuracy of the method.

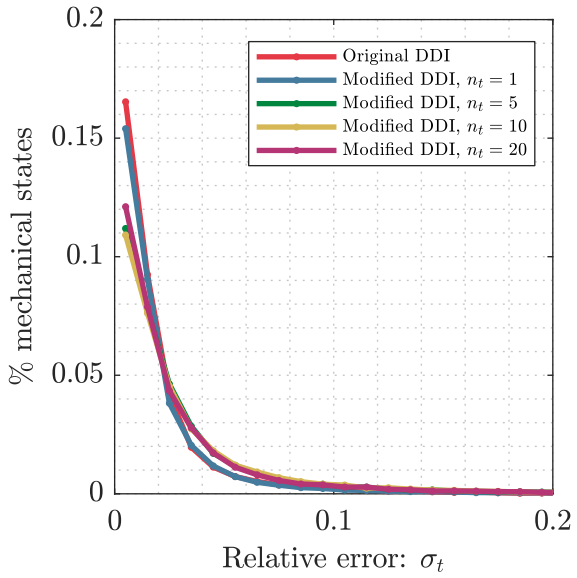


Figure 4.17: Histogram plot for elastic case with $E_v = 0.1$ and $D_v = 1$.

This behavior is not limited to the case where $D_v = 1$. If we do an error analysis of the full system considering $E_v = 0.1$ for different values of D_v we can see that the trend is similar for all values. In general considering a classical DDI approach is better for the estimations than the modified version, which is represented in

Figure 4.18, however, no matter what value we take for D_v the accuracy will be similar, since we are disregarding the viscoelastic part of the problem.

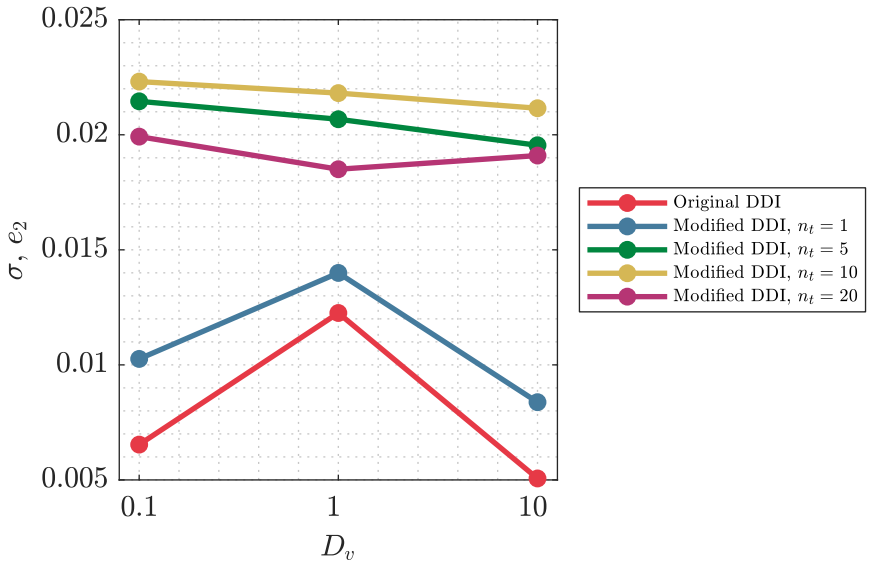


Figure 4.18: Error analysis for $E_v = 0.1$.

- For the second case, we want to analyze the other extreme, where we consider a heavy influence of the viscoelastic branch. In this case, we consider that $E_v = 10$, while we have two possibilities with respect to the time scale. First, we choose $D_v = 10$, which means that the experiment time will be to short with respect to the relaxation time. This effectively makes the model behave as an elastic case, since there is no time to develop the relaxation of the stresses. The second possibility is to take $D_v = 0.1$, which makes the test slow enough, allowing us to observe the viscoelastic behavior.

  For the first case with $D_v = 10$, the results from DDI are shown in Figure 4.19. In here we see a correlation with the previous example from Figure 4.17, since there is not much variation in the estimations provided by the different DDI algorithms. There is a small improvement in the case with $n_t = 20$ which can be attributed to the fact that the sample is viscoelastic, it has only not being allowed to develop this behavior yet.

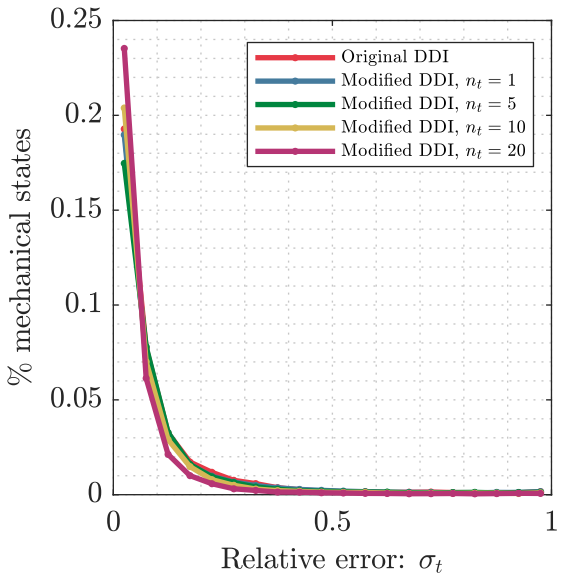


Figure 4.19: Histogram plot for $E_v = 10$ and $D_v = 10$.

For the case with $D_v = 0.1$, the results are shown in Figure 4.20. In here, where the relaxation time is much smaller than the scale of the experiment we can clearly see the improvement of considering a bigger phase space. In fact, the classical formulation of DDI has a very low accuracy even compared with the case of $n_t = 1$, since this algorithm is not adapted for cases outside of the elastic range.

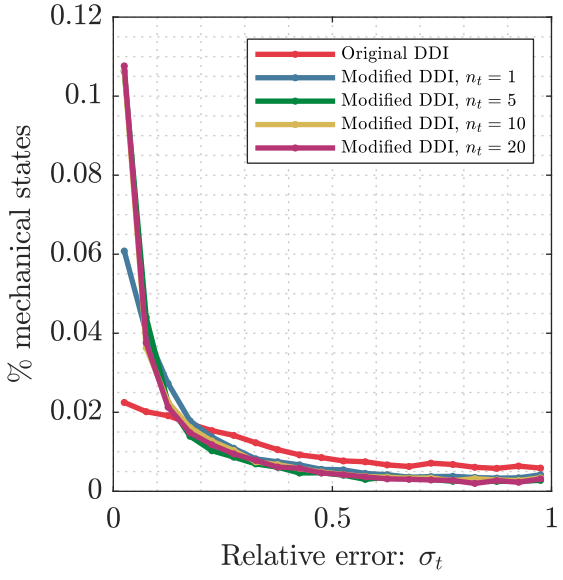


Figure 4.20: Histogram plot for $E_v = 10$ and $D_v = 0.1$.

We can have another look at the methods by plotting the stress versus time for one element of the mesh. As can be seen in Figure 4.21, the original DDI has troubles identifying the peaks of the oscillations due to the sudden changes in strain. For

the modified DDI, we can clearly see how with an increase of dimensionality the estimations get better, and closer to the curve that was obtained using FEM. Even though the start is not very accurate, the case with $n_t = 20$ stabilizes later in time.

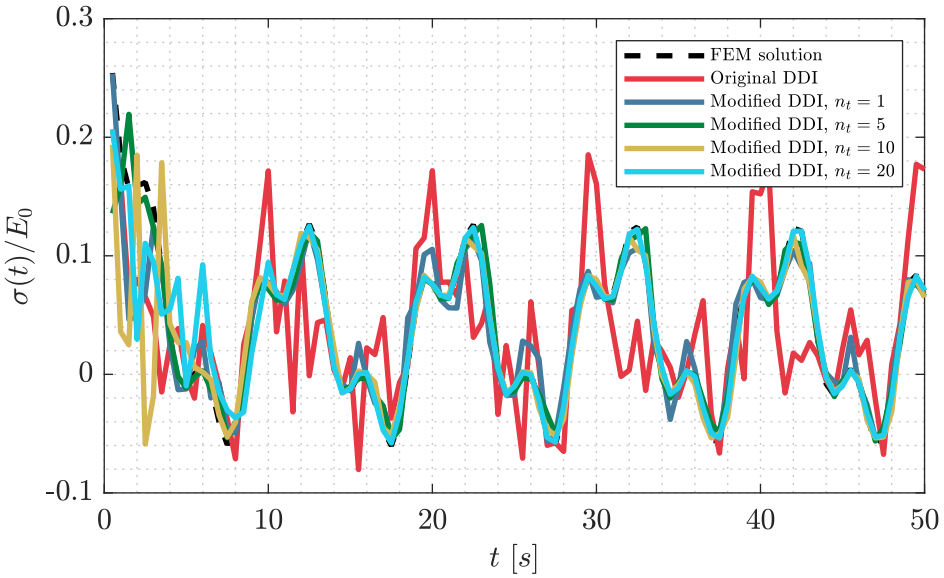


Figure 4.21: Stress versus time for $E_v = 10$ and $D_v = 0.1$.

We can do the same exercise as before and analyze the error of the system for different values of D_v. In Figure 4.22 we clearly see that the classical DDI formulation is much less accurate compared to the others, but when we approach the case of $D_v = 10$ (pseudo-elastic case) the differences become much less important and classical DDI becomes more convenient again.

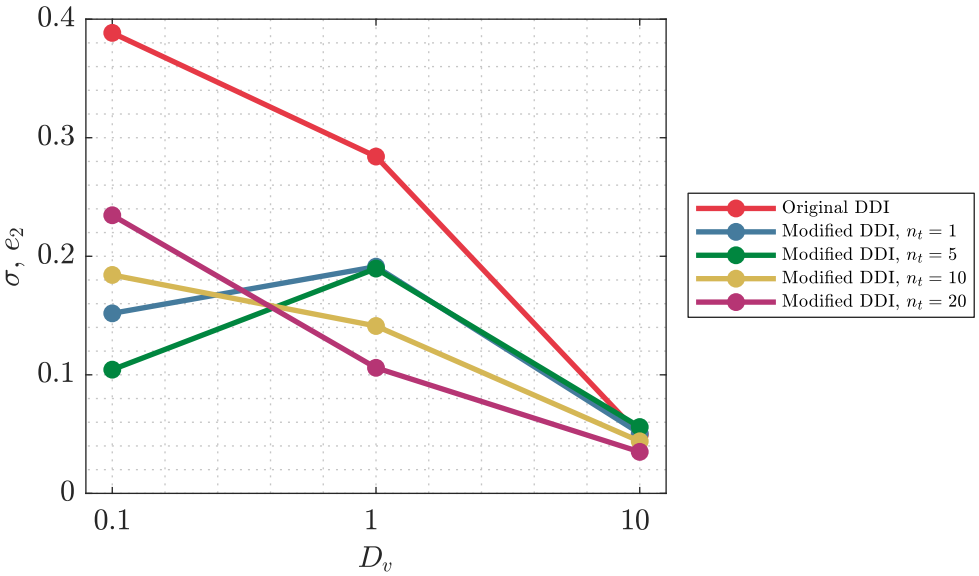


Figure 4.22: Error analysis for $E_v = 10$.

- Finally, we show the case with $E_v = 1$ and $D_v = 1$, which is an average viscoelastic case. In this scenario, we should be able to observe a behavior of DDI in between the cases exposed before. In Figure 4.23 we see that the trend is to have an improvement the more we add previous steps in the phase space, as it was predicted.

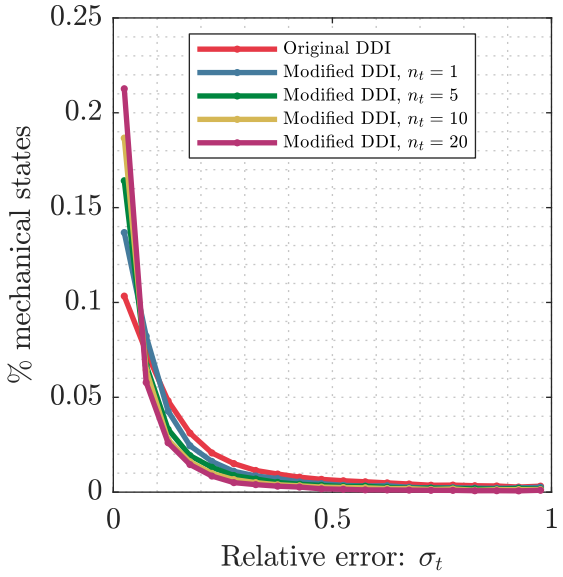


Figure 4.23: Histogram plot for $E_v = 1$ and $D_v = 1$.

From all the test that we have run here, we can see that the trend of the modified DDI algorithm is to improve results when the viscoelastic effect is pronounced. If we are dealing with elastic or almost-elastic cases, then the classical formulation is the better choice, for both accuracy and computational time. With respect to the amount of previous steps to consider, we can see that in general it is good to have more information, since we have improvements in accuracy. However, two things need to be considered: adding more steps increases the dimensionality of the problem which in turn increases the demand for both memory and computation time, which is something that needs to be acknowledged in bigger cases. The other thing is that there is a point in which adding more steps seems to be less beneficial or outright detrimental for the results. This effect can be attributed to different causes, which will be analyzed later, but it is important to keep in mind that more dimensions is not necessarily the appropriate approach.

Analysis of $\mathbb{C}$

In previous works related to DDI, there is always a discussion centered around the selection of the parameter $\mathbb{C}$. As discussed in Chapter 2, when we have elastic samples with one material taking a value of $\mathbb{C} \to \infty$ tends to produce the most accurate estimations. However, when looking at the results from Chapter 3 we see that we might have an optimal value for this parameter. In most cases we decide to consider a value of $\mathbb{C}$ based on heuristic approaches, such as homogenization values or approximations of the expected Young's modulus of the material. In this section we want to explore the effect that $\mathbb{C}$ have on the different cases that we have analyzed.

For testing this effect, we will consider three of the cases shown in the previous subsection: one quasi-elastic case, one mild viscoelastic and one fully viscoelastic. The tests are performed with an oscillatory displacement in the same truss sample. For each case we consider a value $\mathbb{C} = C_h$, with C_h being a constant defined by the properties of the material model as

$$C_h = \left(1 + \left(\frac{1}{1 + \frac{\Delta t}{\tau}}\right)\right) E_v, \tag{4.48}$$

where $\Delta t = t_{exp}/20$. The time of the experiment is the time that it takes for one oscillation to take place, and it is dependent on the parameter D_v. This choice of value for $\mathbb{C}$ is based on the idea of an equivalent value of a Young's modulus based on the properties of both branches of the Zener model. For each test case DDI will be run with values of $0.1C_h$, C_h and $10C_h$, to analyze the behavior of the system with low and high values of $\mathbb{C}$.

The first test case is performed in a sample with $E_v = 0.1$ and $D_v = 1$, which represent a quasi-elastic case as seen previously. DDI is performed considering 0, 1, 5, 10, 20, 40 and 60 previous steps and the error of the full system is obtained, for which the results are visible in Figure 4.24. In general we see the trend that has been explained before: increasing the value of $\mathbb{C}$ tends to improve the accuracy of DDI, which was expected for the elastic case with one material. However, the improvement is not very pronounced. Even more, for cases with low n_t the changes can be attributed more to the randomness of the method rather than a real improvement.

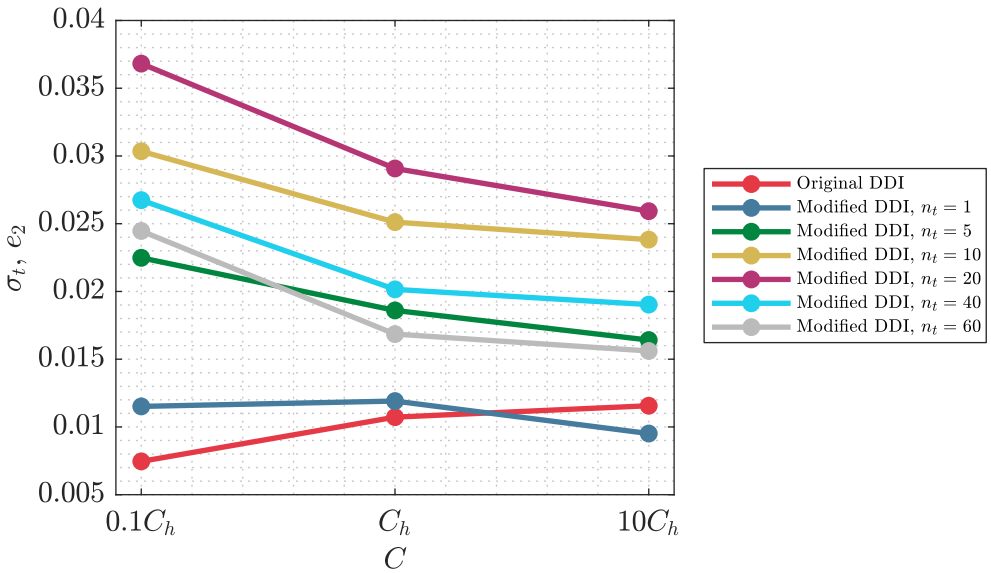
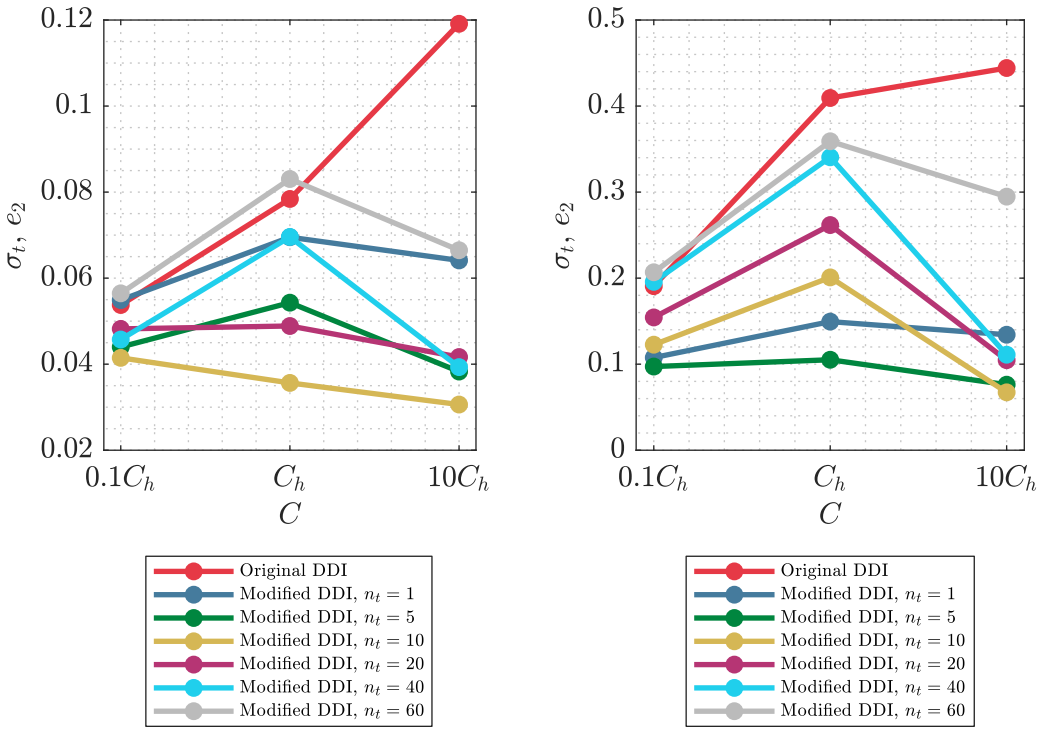


Figure 4.24: Error curves for elastic case with $E_v = 0.1$ and $D_v = 1$.

The second test is performed with $E_v = 1$ and $D_v = 1$ considering the same DDI cases. As can be seen in Figure 4.25a, there is not a concrete trend on the effect of $\mathbb{C}$. For the classical formulation of DDI we have an increase of the error with the increase of $\mathbb{C}$, while for the others the accuracy tends to stay in the same range. This is also observed in the case with $E_v = 10$ and $D_v = 0.1$ for lower n_t, but for higher ones we

can see that the effect of $\mathbb{C}$ value is not very relevant. This is reported in Figure 4.25b.



(a) Error curves for mild viscoelasticity.

(b) Error curves for viscoelasticity.

Figure 4.25: Viscoelastic test cases.

From what we have seen, there is an influence of the parameter $\mathbb{C}$ in the algorithm, but its contribution seems to be more complicated and with small benefits. In a general sense, it is more convenient to choose the value of the parameter still using the heuristic approach. We have seen that the value of $\mathbb{C}$ used in Equation 4.48 gives a good result while using the algorithm and it is a fairly easy value to select. We have also run tests considering $\mathbb{C} = \infty$, which provides a good accuracy and it also has the advantage of performing shorter computations.

4.4.3 Analysis for the oscillatory formulation

In Section 4.4.2 we have studied the effects of the different configurations of the Zener model and how DDI behaves, however, as it was mention in Section 4.1.2, there is a more convenient definition of viscoelasticity when we analyze oscillatory cases. In this section, we aim to check if the behavior that has been observed by defining the pair of parameters $E_v - D_v$ can be recreated by changing the latter for the frequency of oscillation, ω. It is worth mentioning that, in the way we have defined the experiment time, both definitions are analogous. However, an oscillatory formulation allows us to choose better what frequency we want to use in our test, as well as allowing us to use combinations of sines and cosines that would complicate the definition of an experiment time.

For this section we analyze a case with $E_v = 10$, so we have a viscoelastic behavior that can be influenced by the frequencies selected. We consider a plane stress assumption with linear triangles and we set the discretization as 20 steps. In this case we do

not discretize the experiment time, but rather the relaxation time. When we analyze the algorithm we will focus on the vertical component of the stress tensor. Since all the tests performed include variations of a vertical load, the y component of the tensor is accurately represented, while the x and shear components tend to be more unreliable due to the lack of data.

Three cases are chosen to study the oscillatory formulation. All tests will be considered as a sum of two sine functions applied in a prestressed sample. In Figure 4.26 we have the plots for the storage and loss moduli, as well as the plot of $\tan\delta$ ratio for this particular example. In both plots we also show the chosen frequencies for the three cases are marked. Test 1 is performed in the middle part of the frequency range, where the loss modulus is the highest, meaning that the behavior is the most viscoelastic. In here we should observe the improvement of the modified algorithm. To contrast, tests 2 and 3 are performed on the extremes of the range in order to observe a more moderate viscoelastic range, closer to the elasticity case. In both cases this happens because the frequency of oscillation is either to fast or too slow, so the viscoelastic behavior of the material is not able to develop completely.

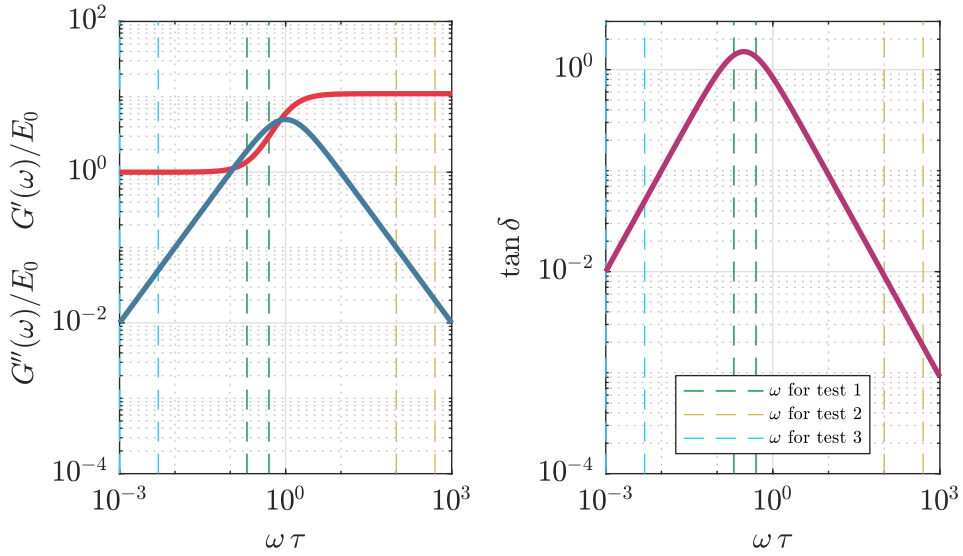
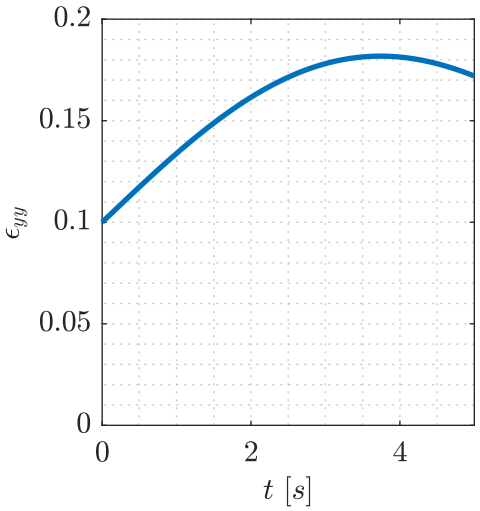
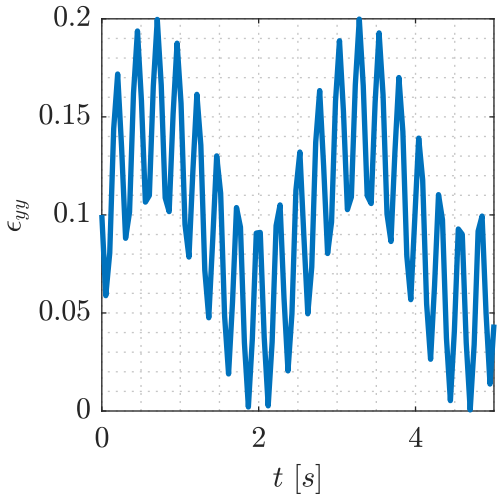


Figure 4.26: Loss and storage moduli for the Zener model for $E_0 = 1$ and $E_1 = 10$.

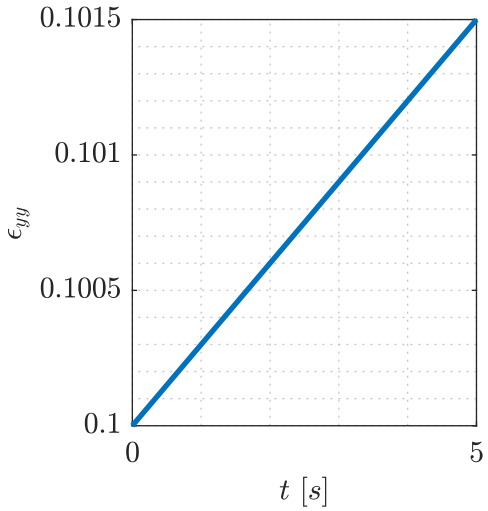
The three displacement profiles applied to the sample can be seen in Figure 4.27.



(a) Test 1, viscoelastic.



(b) Test 2, elastic.



(c) Test 3, elastic.

Figure 4.27: Displacement profiles for selected frequencies.

The results for test 1 show the expected behavior. In Figure 4.28 we can see that there is an improvement of the stress estimations when we consider more steps, which is in line with previous results. The overall accuracy of all the methods is lower than before given that in general DDI has poorer performances for higher dimensional cases, such as plane stress, something that has been seen in previous works. In the modified algorithm this problem is accentuated, but we can see that the same postulates made on the scalar case hold in here.

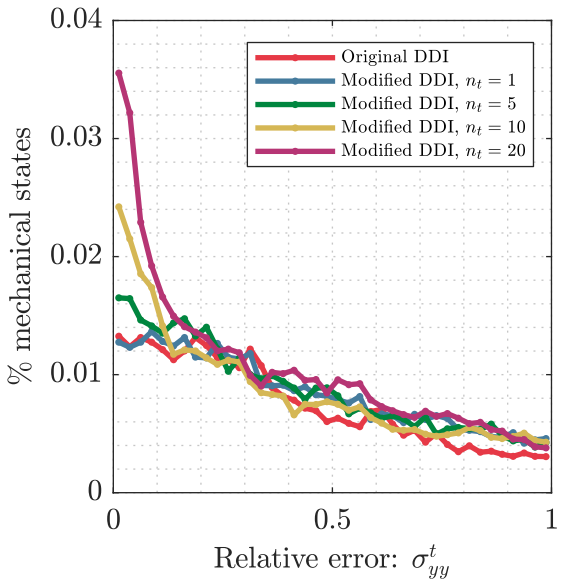
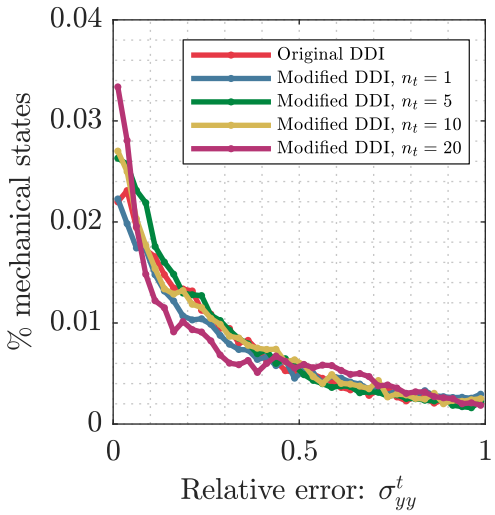
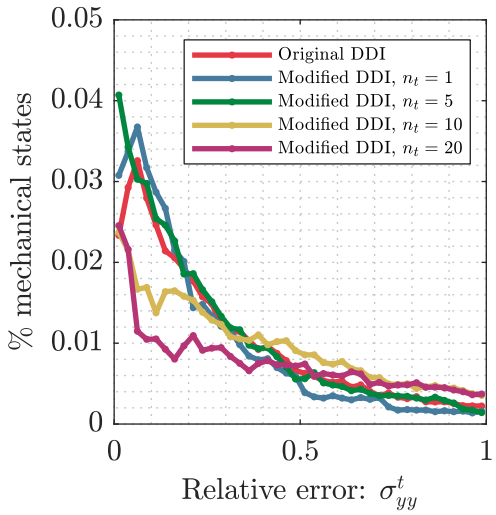


Figure 4.28: Histogram plot for viscoelastic oscillation.

For test 2 and 3 we notice that considering more dimensions in the problem is not beneficial, as it was expected. In Figure 4.29a we see no noticeable improvement from the modified algorithms, while in Figure 4.29b we actually see a decrease in accuracy for the higher dimensional cases, since the deformation is small enough to be precise.



(a) Histogram plot for test 2.



(b) Histogram plot for test 3.

Figure 4.29: Histogram plot for elastic oscillation.

The oscillatory formulation for viscoelasticity is a convenient way to study the problem when we have complex functions describing the displacements. We see that the use of the storage and loss moduli representation allows us to understand the behavior of the material a priori, which can give us an idea of which DDI formulation we should use.

4.4.4 Effect of time discretization

From the previous sections we have established that increasing the dimensionality of the problem helps obtain a better accuracy for the viscoelastic cases, but it is not a constant improvement and there is a limit on how many previous steps are beneficial before the accuracy drops again. In this section we expect to find what is that particular limit and how it is affected by the time discretization chosen for the problem.

Figure 4.30 shows an example of this phenomenon for an oscillatory test. There is a progressive improvement of the accuracy of DDI when more steps are considered, until a peak is reached at $n_t = 10$. If we continue increasing the amount of steps we notice that the error starts increasing again, where having 40 extra steps is almost equally accurate as only having five.

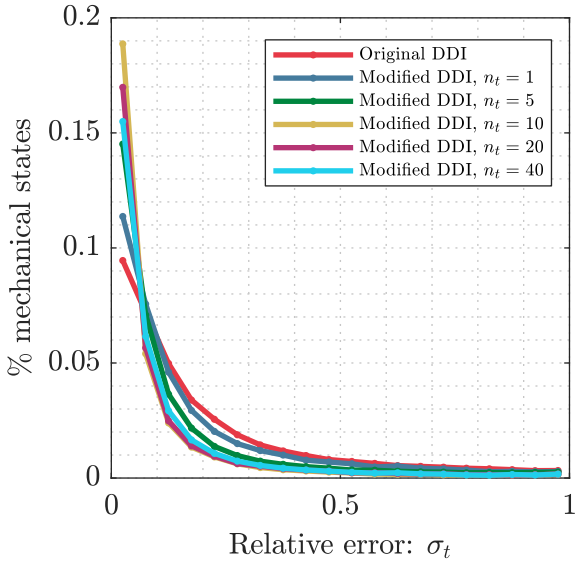
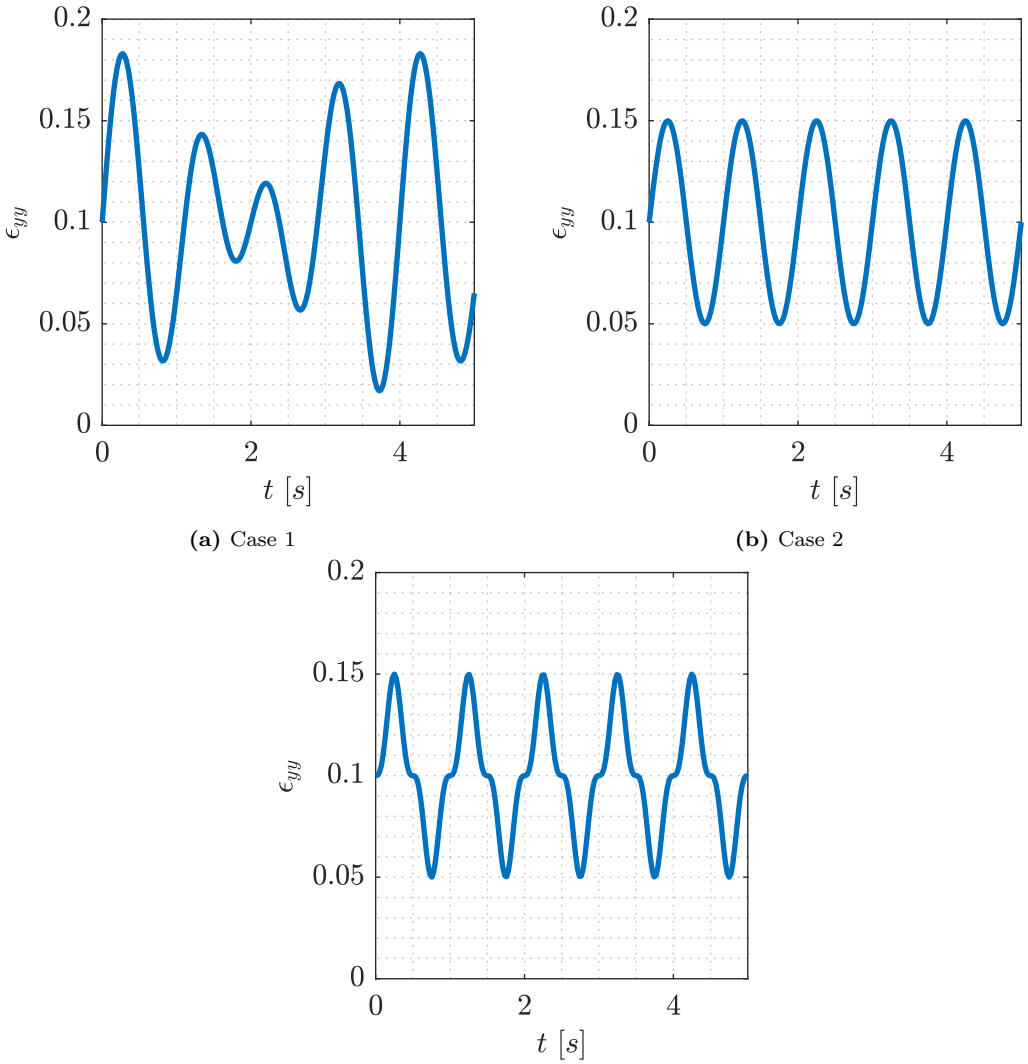


Figure 4.30: Example for an oscillatory test.

On this basis we want to study what is the optimal amount of steps that we should consider for performing DDI. We also explore the possibility that the amount of steps to consider is not a fixed quantity, but mostly related to the relaxation time of the material.

To test this we run three different cases in our sample mesh, considering bar elements. We consider $E_v = D_v = 1$ in order to have a moderate viscoelastic case and we take different deformation profiles based on oscillatory movement, as shown in Figure 4.31. For each one of the test cases the strains are obtained using three different time discretizations, considering $\Delta t = \tau/10$, $\tau/20$ and $\tau/40$. Having different intervals means that more or less steps are needed to cover the strain history associated to a full relaxation time, which could translate in the need of less steps to obtain same levels of accuracy.



(a) Case 1

(b) Case 2

(c) Case 3

Figure 4.31: Displacement profiles used.

For the first case, the results are shown in Figure 4.32. DDI is run with different amounts of previous steps for the different time discretizations, and the full error of the system is plotted for every single case. As it can be seen, there is an optimal point for each curve, where the error is minimized. Considering too many steps is detrimental due to the overload of information, but there are other causes to be considered. DDI is dependent on finding nearest neighbors (which is performed when we minimize the distances on the phase space). When the dimensionality increases too much the curse of dimensionality makes it difficult to perform kNN searches.

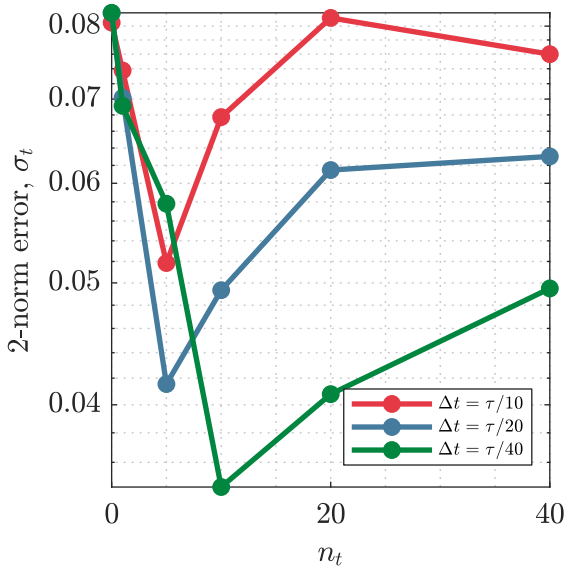


Figure 4.32: Error curves for case 1.

We can see that the optimal amount of steps to consider is typically lower than the amount of steps that cover a full relaxation time. If we scale the results based on the discretization, we can obtain the error plotted against relaxation times, which is what is shown in Figure 4.33. In here we see that the optimal point is actually fixed and corresponds to $\tau/4$. For the case of $\Delta t = \tau/10$ we have the optimal point at $\tau/2$, which is different due to the fact that the case of $n_t = 2.5$ is not available.

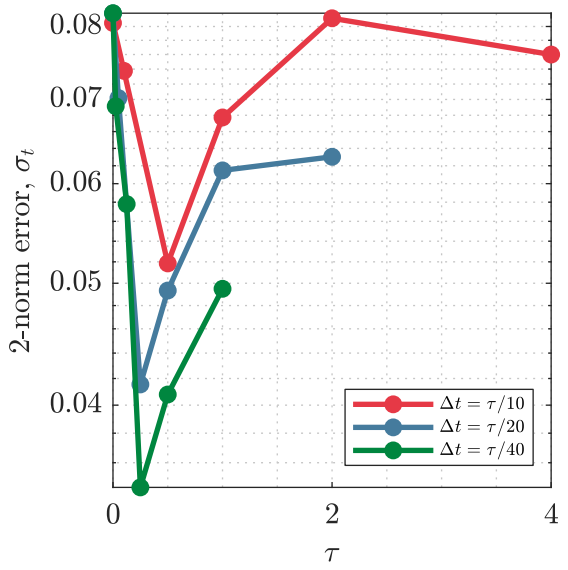
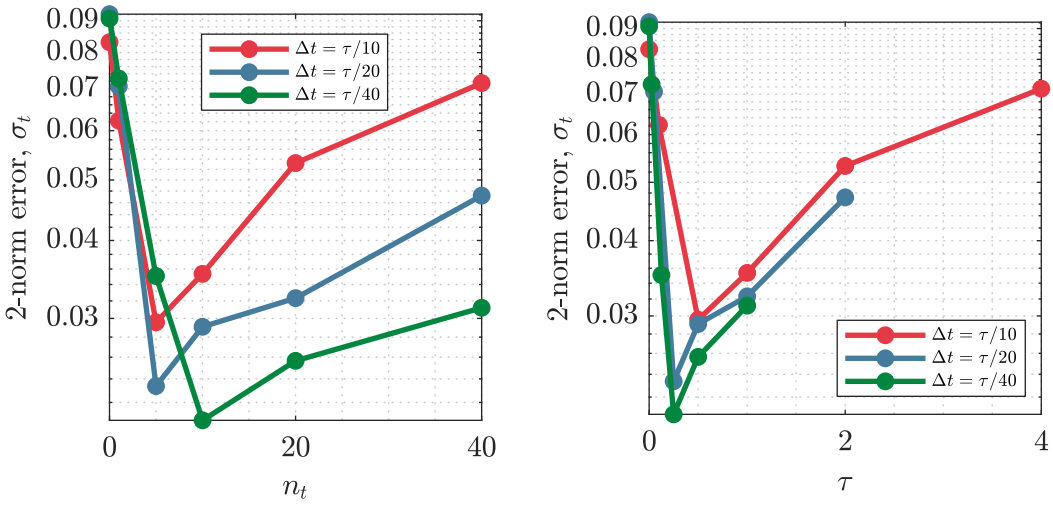


Figure 4.33: Error curves for case 1, scaled.

This results are not limited to the first case. In Figure 4.34 we show the results for the second test where we can see similar results, having the same optimal points as case 1.

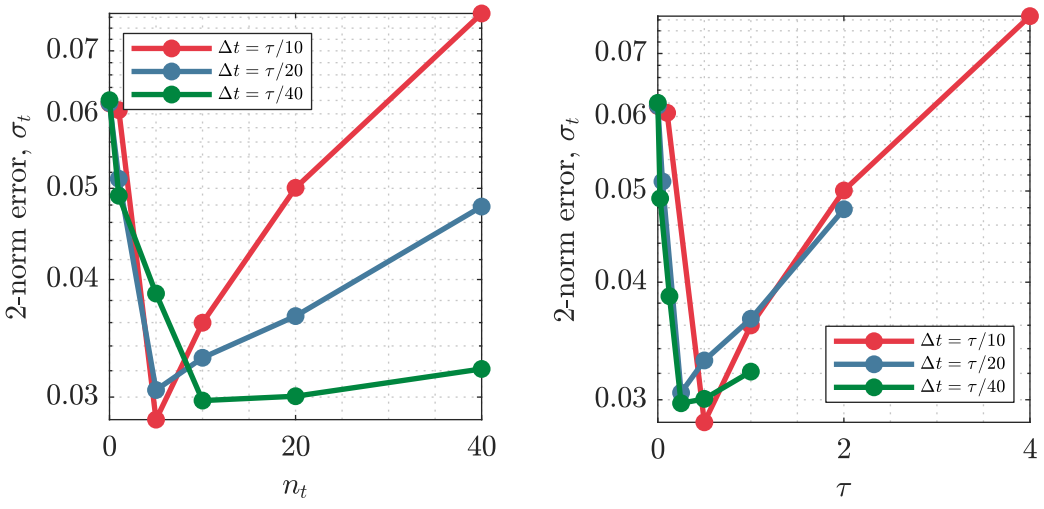


(a) Error curves for case 2.

(b) Error curves for case 1, scaled.

Figure 4.34: Case 2.

Finally we show case 3 in Figure 4.34 to confirm the same results. These results gives us a good insight at how the modified algorithm works, since we can state now that the amount of steps that we need depends mostly on the properties of the material rather than the experiment around it, so we can adapt the parameters if we need to reduce the amount of computations. However, it is worth mentioning that these experiments are run in one particular mesh, and it can be expected that the optimal point differs if we consider different samples.



(a) Error curves for case 3.

(b) Error curves for case 3, scaled.

Figure 4.35: Case 3.

4.5 Discussion

During the course of this chapter, we have attempted to tackle the challenge of using DDI for more complex types of materials. In particular, we have settled on the appli-

cation for linear viscoelasticity, given that is one of the simplest material models that can be analyzed outside of the elasticity range. The approach that we have taken is to modify our DDI algorithm to account for the extra information that viscoelasticity needs. In particular, we have decided to extend the definition of the phase space as a way to account for the strain history that defines the way that viscoelastic stress behaves.

The first results obtained show that there is an improvement in the accuracy of the results when we use the modified formulation. This is further tested by running different cases with different parameters to understand better if this improvement is general or if it limited to certain cases. The results show that the modified DDI algorithm works better when we deal with cases associated to viscoelasticity. For cases where the viscoelastic effect is too small, the strain history becomes irrelevant and we tend to revert to an elastic case, where the classical approach of instantaneous strain-stress pairs for DDI is much more accurate. Another important results that we have seen is that the amount of previous steps that need to be considered in the modified formulation is highly dependent on the relaxation time of the material, rather than external factors such as the forces used. This allows us to wage if we want to have a more accurate solution with less mechanical points or if we would like to run a longer simulation with more steps in between.

One of the main issues with this new formulation is the role that material states play. Material states, mathematically, have always played the role of stabilization of the algorithm. They are there to assist with the minimization of the distances in the phase space, but in elastic materials they also work as a database of points that samples the behavior of the material. In viscoelasticity, this is not the case. Since the new formulation includes some sort of temporal connection between the different snapshots, material points are now clustering points independent of time, which does not allows us to define any kind of constitutive curve. This issue, however, needs to be studied further, since there might be possibilities to treat the data in a way that might allows us to obtain information out of them.

Another issue that we have run into is the plane stress formulation. From the oscillatory formulation case we see that the algorithm works and behaves accordingly to the results obtained in trusses. However, accuracy for the plane stress case has always been low, and the for the viscoelastic cases this is more pronounced. One possible explanation might be that we have only performed uniaxial tests, where we know that results might not be accurate due to the under-representation of certain directions. The other problem associated with it is the fact that just by using plane stress we immediately increase the dimensions of the problem, so considering more previous steps to improve the accuracy quickly becomes prohibitively when we are studying samples with a high amount of mechanical points. This is one of the motivations for the analysis that is performed in Chapter 5: we seek to use data analysis techniques that can help us improve both accuracy and computational time through the reduction of the dimensionality of the problems.

In the current state, the modified algorithm of DDI cannot be proposed as a way of identifying material properties. Nonetheless, we have verified that it can be properly used in order to obtain the estimation of the stresses on a sample at every timestep. In this way, it might be possible to analyze the properties of a material by running DDI, for example in a relaxation test, where we can denoise the results and fit curves on the relaxation curves in order to estimate the relaxation time of the material. These

approaches are not as clean as the ones proposed in [11] or in Chapter 3, but they are definitely a step forward on the issue of DDI and viscoelasticity.

Chapter 5

Alternative approaches to the data-driven algorithm

In the previous chapters we have focused on different adaptations of the DDI algorithm aiming to solve problems for different kinds of materials. However, these approaches do not diverge very far from the original formulation and are more of extensions to solve new problems. In this chapter, the emphasis will be put in modifications to the core of the algorithm and combinations with other techniques that could improve the performance of DDI. Furthermore, such techniques will be also developed as a post-processing tool for the results.

5.1 Motivation

DDI depends on data that we obtain from testing to avoid having to do arbitrary choices. Nevertheless, this does not save us from the need of defining certain parameters. In particular, parameter $\mathbb{C}$ is complicated in nature. In previous works, the parameter $\mathbb{C}$ has been defined in several ways: Some have referred to it as a numerical constant that defines a penalty function (as seen in [17]), while in other approaches (such as [11]) it is treated as a value that allows to define a distance norm. It has also been called a pseudo-stiffness due to the way it replaces the real one in the equations, even though its nature has nothing to do with this value. All of these references make it difficult to give a proper definition of the parameter, and they create confusion when the method is introduced to people who are not familiar with these techniques. Another problem related to $\mathbb{C}$ is its effect on the accuracy of the problem. A common approach in the works shown in this document was to take a value of $\mathbb{C}$ in the order of magnitude of the underlying stiffness. In [11] it was seen that lowering the value gave more accurate predictions. In the work of Dalémat [26] however, it is seen that for elastic samples, accuracy increases when $\mathbb{C}$ tends to infinity. This also clashes with the results seen in Chapter 3, where it is shown that $\mathbb{C}$ has an optimal value for the tested samples.

This sensitivity of the parameter $\mathbb{C}$ is not observed in the other parameter N^*: it has been consistently seen that the parameter has an optimal value that can be found, given that taking too many or too few points causes problems in the clustering of the material states. It is because of all these issues that it would be desirable to avoid using $\mathbb{C}$, to reduce the influence of those choices.

Another aspect to consider for modifying the process is to improve both the speed and accuracy of the solutions. In Chapter 4 we saw that increasing the dimensionality of the problem leads to a slowing of the computation. Dimensionality reduction techniques can eliminate redundant information, which in turn allows for smaller problems that can be handled faster. It is also expected that removing the influence of the parameter $\mathbb{C}$ would allow for a more optimal estimation of stresses, that is only dependent on the amount of clusters selected.

5.2 Modified DDI algorithms

The value of $\mathbb{C}$ is used in the definition of the norm to compute the distance between points. The reason behind is the compatibilization of units from all the dimensions that define the phase space. For modifying the algorithm we address this issue in particular. A very naive approach to avoid the difference of units will be to perform a normalization in each one of the coordinates.

5.2.1 Normalization in DDI

Normalization of units

In statistics, normalization is a common technique that allows the comparison of different datasets. This is achieved by adjusting the scale of each dataset to remove the differences that restrict the analysis, setting a common ground. There are many ways of normalizing data, but in this formulation we limit ourselves to the simplest one, which is the one based on the standard score. If we have a dataset X, assuming that it follows a normal probability distribution, then it is defined by two parameters: μ and s, which correspond to the mean (or center) and the standard deviation (or spread) of the samples in it. In mathematical notation, this distribution is expressed as [53]

$$X \sim N(\mu, s^2). \tag{5.1}$$

We can adjust the data into a standard normal distribution Z with the expression

$$Z = \frac{X - \mu}{s}, \tag{5.2}$$

which gives a centered distribution ($\mu = 0$) with the standard deviation set to one. An important property of this distribution is the fact that it is unitless. Working with data without units is convenient since now the distance in the phase space can be defined using an euclidean norm, which eliminates the need to use the parameter $\mathbb{C}$.

Application in the algorithm

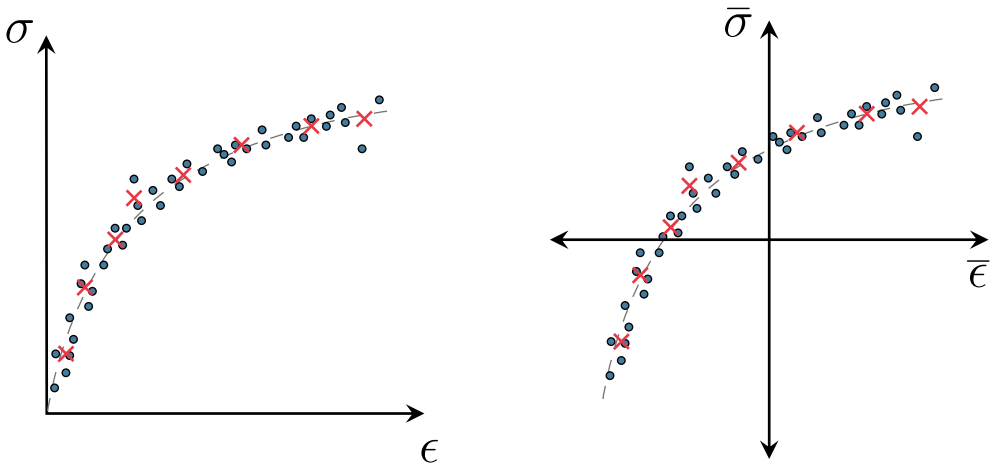
When applying these ideas into DDI, it should not be forgotten that, even when talking about normalized datasets, the problem to be solved is physical in nature. Even if the input data is normalized, the computation of mechanical stresses is constrained by physical terms, such as the applied forces. All these quantities lie on the phase space, so whenever a normalization is performed, we are effectively transforming the data from the original space into a new one, which we will call the *normalized space*. Since these

constraints are not necessarily valid in the this new space, we want to limit its use to the clustering phase of the algorithm, so the process will include a constant transform between the phase and normalized space.

The normalization of the mechanical states is performed such that

$$\begin{aligned} \bar{\boldsymbol{\epsilon}} &= \frac{\boldsymbol{\epsilon} - \mu_\epsilon}{s_\epsilon}, \\ \bar{\boldsymbol{\sigma}} &= \frac{\boldsymbol{\sigma} - \mu_\sigma}{s_\sigma}, \end{aligned} \tag{5.3}$$

which are just a rewritting of Equation 5.2. In here, normalization is performed in each one of the components of the strain and stress tensors independently. This means that an independence between the components is assumed to exist in the normalized space. This independent scaling of each dimension accounts for some of the difference between the results obtained with the original DDI method and the normalized variant.



(a) Phase space **(b)** Normalized space

Figure 5.1: Effect of the normalization on the phase space for a scalar case. The data is centered in the origin with a unit standard deviation.

Having the values of $\bar{\boldsymbol{\epsilon}}$ and $\bar{\boldsymbol{\sigma}}$ allows us to obtain a state mapping in the normalized space by minimizing the euclidean distances between normalized mechanical states $(\bar{\boldsymbol{\epsilon}}_e^X, \bar{\boldsymbol{\sigma}}_e^X)$ and normalized material points $(\bar{\boldsymbol{\epsilon}}^*_{\mathrm{ie}X}, \bar{\boldsymbol{\sigma}}^*_{\mathrm{ie}X})$. This is performed by using the k-means algorithm in order to find N^* different clusters. This approach mirrors the original DDCM and DDI methodology

As mentioned before, the normalized quantities are limited only to the clustering phase of the algorithm. The search for updated mechanical states, which are constrained, has to be performed in the original phase space. For this, the inverse transformation has to be performed in the normalized material states, i.e.,

$$\begin{aligned} \boldsymbol{\epsilon}^*_{\mathrm{ie}X} &= \mu_\epsilon + s_\epsilon \bar{\boldsymbol{\epsilon}}^*_{\mathrm{ie}X}, \\ \boldsymbol{\sigma}^*_{\mathrm{ie}X} &= \mu_\sigma + s_\sigma \bar{\boldsymbol{\sigma}}^*_{\mathrm{ie}X}. \end{aligned} \tag{5.4}$$

By using the generated state mapping we can track the points that are clustered with

each material point, obtain a local mean and standard deviation of the cluster, and then perform the inverse process. In here, for the sake of simplicity, the values used for μ and s are the same as those obtained for the mechanical states. This means that we are treating each one of the material states as added data points in the set that, when placed, have no effect in both the mean and the standard deviation of the full sample group. Finally, since now we have the material states in the original phase space we can proceed with the update of the mechanical states using the original Equation 2.34c.

This process is repeated iteratively until we have reached our desired convergence. For this, we check that in each step the variation of the values for the material states, and we stop the process when this variation is small enough. The main procedure for this modified version of DDI is outlined in Algorithm 5.1.

5.2.2 PCA as a generalization of the normalized algorithm

The high dimension of the data used in DDI is the most important bottleneck when running the algorithm. All of the data-driven methods presented in this document rely on the search of nearest neighbors that minimize the objective function. It is known that increasing the dimensionality of problems has a negative effect on the efficiency of kNN searches [51] and clustering techniques [52]. Measuring distances becomes difficult, since in spaces with higher dimensions points tend to become sparser and the differences between pairs of samples decrease. This phenomenon is called *the curse of dimensionality*, and it affects many domains where data needs to be processed.

To improve the performance of the algorithms presented, we focus on reducing the size of the data in our problems. We address this issue with the use dimensionality reduction techniques. One of the simplest and most popular techniques is the *Principal Component Analysis* (PCA), which we will include in our DDI formulation.

Principal Component Analysis

PCA [54] is a multivariate statistical technique that has an extended use in many disciplines. The method analyzes a dataset consisting of n samples of dimension d to extract information of the relation between the variables. The output of PCA are known as *principal components*, which are a set of orthogonal vectors that best fit the data. This means that each vector represents a line that minimizes the average squared distance from all the points to the line, which is equivalent to the axis that has the lowest covariance.

PCA is performed by grouping the dataset in one matrix, which we call $\boldsymbol{X}$. Each column corresponds to one observation $\boldsymbol{x}_i$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, so naturally, $\boldsymbol{X}$ has dimensions $d \times n$. The rank of $\boldsymbol{X}$ is L, which is equal or lower than the smallest dimension of the matrix. In a mathematical form, we call each element of the matrix as X_{ij}, referring to a sample j in the dimension i.

Before computing the principal components, the data needs to be preprocessed. Since PCA measures the principal vectors from the origin, having non-centered data results in a first principal vector that does not represent the data, since it is actually a vector pointing from the origin to the data cloud. Figure 5.2 explains this effect in a graphical form. Because of this, it is always recommended that the data is centered in the columns, meaning that each dimension of the problem is shifted to the origin. Additionally, if the units of each dimension are different, it is necessary to normalize

Algorithm 5.1: DDI procedure with normalized mechanical states.

Output: $\boldsymbol{\epsilon}_{\mathrm{i}}^{*}$, $\boldsymbol{\sigma}_{\mathrm{i}}^{*}$, $\boldsymbol{\sigma}_{e}^{X}$, ie^{X}
Input: N^{*}, $\boldsymbol{u}_{j}^{X}$, $\boldsymbol{f}_{j}^{X}$, $\boldsymbol{B}_{ej}^{X}$, $\boldsymbol{\epsilon}_{e}^{X}$

Normalized DDI procedure

Set $k \rightarrow 0$

Initialization: set $\boldsymbol{\sigma}_{e}^{X(k=0)} = 0$ for each element e in every snapshot X. Likewise, initialize $\boldsymbol{\sigma}_{\mathrm{i}}^{*(k=0)} = 0$ for each material state i.

Initialization: for each element e in each snapshot X, normalize the mechanical strains ($\boldsymbol{\epsilon}_{e}^{X}$) and stresses ($\boldsymbol{\sigma}_{e}^{X(k=0)}$) with Equation 5.3. Retain the values of $\mu^{(k=0)}$ and $s^{(k=0)}$ for all quantities.

Initialization: Obtain ($\bar{\boldsymbol{\epsilon}}_{\mathrm{ie}^{X}}^{*(k=0)}, \bar{\boldsymbol{\sigma}}_{\mathrm{ie}^{X}}^{*(k=0)}$) by clustering all ($\bar{\boldsymbol{\epsilon}}_{e}^{X}, \bar{\boldsymbol{\sigma}}_{e}^{X(k=0)}$) into N^{*} groups.

while *convergence criterion for* $\boldsymbol{\epsilon}_{i}^{*}$ *and* $\boldsymbol{\sigma}_{i}^{*} <$ *tolerance* **do**

Obtain the not normalized material states ($\boldsymbol{\epsilon}_{\mathrm{ie}^{X}}^{*(k)}, \boldsymbol{\sigma}_{\mathrm{ie}^{X}}^{*(k)}$) using Equation 5.4.

for $X = 1, \ldots, N^{X}$ **do**

Solve equations

$$\sum_{k}\sum_{e} w_{e}^{X}\boldsymbol{B}_{ej}^{X^{T}} : \mathbb{C} : \boldsymbol{B}_{ek}^{X} \cdot \boldsymbol{\eta}_{k}^{X(k)} - \sum_{e} w_{e}^{X}\boldsymbol{B}_{ej}^{X^{T}}\boldsymbol{\sigma}_{\mathrm{ie}^{X}}^{*(k)} = \boldsymbol{f}_{j}^{X} \quad \forall j, X;$$

$$\boldsymbol{\sigma}_{e}^{X(k+1)} = \boldsymbol{\sigma}_{\mathrm{ie}^{X}}^{*(k)} + \sum_{j} \mathbb{C} : \boldsymbol{B}_{ej}^{X} \cdot \boldsymbol{\eta}_{j}^{X(k)} \quad \forall e, X.$$

with current values of $\boldsymbol{\sigma}_{\mathrm{i}}^{*(k)}$ to obtain updated values of $\boldsymbol{\sigma}_{e}^{X(k+1)}$.

Compute

$$\mu_{\sigma}^{(k+1)} = \frac{1}{N^{e}N^{X}}\sum_{e}\sum_{X}\boldsymbol{\sigma}_{e}^{X(k+1)}$$

$$s_{\sigma}^{(k+1)} = \sqrt{\frac{1}{N^{e}N^{X}}\sum_{e}\sum_{X}(\boldsymbol{\sigma}_{e}^{X(k+1)} - \mu_{\sigma}^{(k+1)})^{2}}$$

Normalize the $\boldsymbol{\sigma}_{e}^{X(k+1)}$

$$\bar{\boldsymbol{\sigma}}_{e}^{X(k+1)} = \frac{\boldsymbol{\sigma}_{e}^{X(k+1)} - \mu_{\sigma}^{(k+1)}}{s_{\sigma}^{(k+1)}}$$

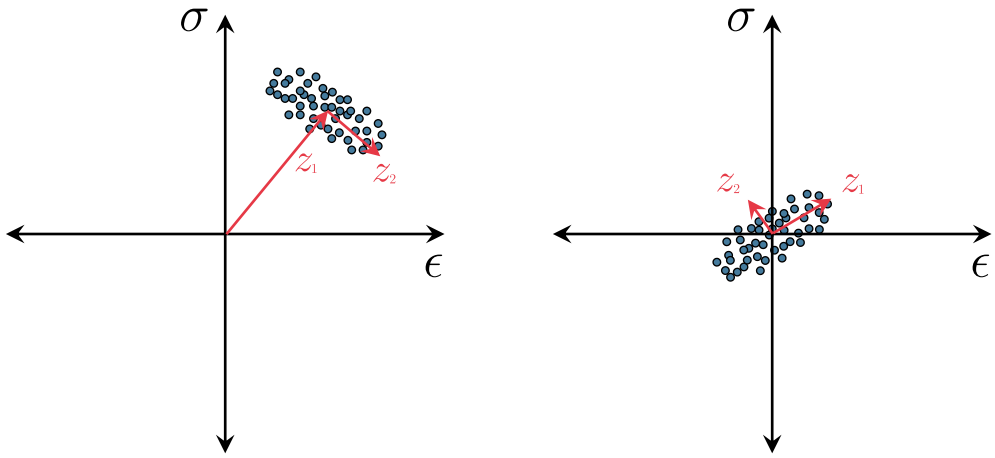
Perform a k-means clustering on all ($\bar{\boldsymbol{\epsilon}}_{e}^{X}, \bar{\boldsymbol{\sigma}}_{e}^{X(k+1)}$) to update the material states ($\bar{\boldsymbol{\epsilon}}_{\mathrm{ie}^{X}}^{*(k+1)}, \bar{\boldsymbol{\sigma}}_{\mathrm{ie}^{X}}^{*(k+1)}$).

Set $k \rightarrow k+1$

the columns by their norm:

$$\tilde{\boldsymbol{X}}_i = \frac{\boldsymbol{X}_i - \mu_i}{||\boldsymbol{X}_i||}, \tag{5.5}$$

with $\boldsymbol{X}_i$ being a row vector and μ_i its mean. By normalizing each column in this way we obtain a correlation matrix when the multiplication $\boldsymbol{C} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T$ is performed. This is the basis for PCA analysis.



(a) Principal components of uncentered data. **(b)** Principal components of centered data.

Figure 5.2: Difference of principal components for centered and uncentered data. The shift of the data cloud far from the origin creates a first principal component that does not represent correctly the relation between data points.

Matrix $\boldsymbol{C}$ is positive semi-definite of dimension $d \times d$. Performing an eigendecomposition we can separate $\boldsymbol{C}$ into two matrices $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$, both in $\mathbb{R}^{n\times n}$ according to the equation

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}. \tag{5.6}$$

In this expression, each column of $\boldsymbol{U}$ is an eigenvector of $\boldsymbol{C}$, while $\boldsymbol{\Lambda}$ is a diagonal matrix with the corresponding eigenvalues from highest to lowest value. The principal components of $\boldsymbol{X}$ are the columns of the matrix $\boldsymbol{Z}$ that can be computed as

$$\boldsymbol{Z} = \boldsymbol{U}^{-1}\boldsymbol{X}. \tag{5.7}$$

Matrix $\boldsymbol{U}$, as a matrix of eigenvectors, represents an orthonormal basis. Performing the multiplication in Equation 5.7 is equivalent to perform a rotation of the data $\boldsymbol{X}$ in the direction of the principal axes of $\boldsymbol{U}$, which are the axes with the lowest covariance.

PCA can be used for reducing the dimension of the problem when some of the principal components are eliminated. The trace of the matrix $\boldsymbol{C}$ is equivalent to the total variance of the samples. This quantity is also equivalent to the trace of $\boldsymbol{\Lambda}$ or the total sum of all the eigenvalues. If the eigenvalues decrease in size fast, this means that

only some of the total amount of dimensions in the problem hold most of the variance. If we set a tolerance ε, we can approximate the total variance of the dataset with only $k < n$ dimensions, such that

$$\sum^{k} \lambda_i \geq (1 - \varepsilon) \sum^{n} \lambda_i. \tag{5.8}$$

In this case, we can omit the last $d - k$ columns of the matrix $\boldsymbol{U}$ to create a reduced set of principal components $\boldsymbol{Z}^r \in \mathbb{R}^{d \times k}$, consisting of only the first k columns of $\boldsymbol{Z}$. This is important for the backward mapping of the data. Inverting the expression in Equation 5.7 gives the reconstitution of the original data matrix $\boldsymbol{X}$. If we use the reduced principal components, then we have

$$\boldsymbol{X}^r = \boldsymbol{U}^{r\,T} \boldsymbol{Z}^r, \tag{5.9}$$

where we take advantage of the orthonormal nature of $\boldsymbol{U}$, since $\boldsymbol{U}^{-1} = \boldsymbol{U}^T$. In this expression, $\boldsymbol{X}^r$ represents a reduced version of the dataset $\boldsymbol{X}$, which is an approximation in where some of the information is removed. The error of the approximation reduces with the tolerance, meaning that the more dimensions we consider, the closer both matrices are.

PCA method through SVD

Singular Value Decomposition (SVD) is a technique which works as a generalization of eigendecomposition for non-square matrices. For any matrix $\boldsymbol{M}$ of size $m \times p$, its decomposition is expressed as

$$\boldsymbol{M} = \boldsymbol{F}\boldsymbol{\Sigma}\boldsymbol{V}^*, \tag{5.10}$$

where $\boldsymbol{F} \in \mathbb{R}^{m \times m}$ and $\boldsymbol{V} \in \mathbb{R}^{p \times p}$ are matrices whose columns are known as singular vectors; and $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times p}$ is a matrix formed by the singular values of $\boldsymbol{M}$ in its main diagonal. Both $\boldsymbol{F}$ and $\boldsymbol{V}$ are orthonormal bases. In Equation 5.10, $\boldsymbol{V}^*$ stands for the conjugate transpose. If the matrix $\boldsymbol{M}$ is real, $\boldsymbol{V}^*$ can be written simply as $\boldsymbol{V}^T$. A full explanation of the SVD method can be found in Appendix C.

Performing the SVD in the data matrix $\boldsymbol{X}$ yields the decomposition

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{Q}^T, \tag{5.11}$$

where the matrix $\boldsymbol{U}$ is equivalent to the one obtained from the eigendecomposition of $\boldsymbol{X}\boldsymbol{X}^T$, and $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}\boldsymbol{\Sigma}^T$, with each element $\lambda_i = \sigma_i^2$. This comes from performing the multiplication of the decomposed matrix and taking advantage of the orthonormal

properties of $\boldsymbol{Q}$, i.e., $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$. This results in

$$\begin{aligned}\boldsymbol{X}\boldsymbol{X}^T &= (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{Q}^T)(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{Q}^T)^T \\ &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{\Sigma}^T\boldsymbol{U}^T \\ &= \boldsymbol{U}(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T)\boldsymbol{U}^T \\ &= \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T,\end{aligned} \tag{5.12}$$

which is equivalent to Equation 5.6. It is worth mentioning that the opposite relation also holds, meaning that $\boldsymbol{Q}$ is equivalent to eigenvectors of the matrix $\boldsymbol{G} = \boldsymbol{X}^T\boldsymbol{X}$, whose eigenvalues $\tilde{\boldsymbol{\Lambda}}$ can be defined as $\tilde{\boldsymbol{\Lambda}} = \boldsymbol{\Sigma}^T\boldsymbol{\Sigma}$.

Generally, PCA is always performed using SVD. The computation of the matrix $\boldsymbol{X}\boldsymbol{X}^T$ can become expensive if the amount of dimensions d is big enough. Even though the same principal components can be obtained from performing the opposite multiplication (i.e., $\boldsymbol{X}^T\boldsymbol{X}$), SVD allows us to have both sets of vectors from just one computation. This advantage becomes more important later in the kernel formulation, where we do not necessarily have access to the opposite multiplication. Another advantage from using SVD comes from its implementation. Every mathematical software includes functions to directly perform SVD in an optimized way, which simplifies and speeds up the method. Finally, another consideration is stability. It has been seen [55] that performing PCA through the SVD of $\boldsymbol{X}$ avoids errors coming from the eigendecomposition, as the precision can be worsened when multiplications of small numbers are made.

5.2.3 kPCA for non-linear manifolds

PCA provides a useful tool that allows to visualize the data in a more optimal space, which most of the times has the advantage of being reduced. However, there are some limitations. PCA is a linear technique, meaning that the method is designed to study the relations between the variables when they lie in a linear manifold. This works fine when working with simple cases like elasticity, but it falls short when the manifolds are not linear.

The problem can be assessed with the use of non-linear PCA [56]. The idea is to find a function that *untangles* the manifold, i.e. finding a function that projects the data to a higher dimension where the underlying structure is linear. The problem with this approach is that such functions might be difficult or even impossible to find. The non-linearity of the problem also might lead to non-unique solutions, as opposed to the original PCA.

One way to circumvent this difficulty is through the use of kernels, which are specific functions designed to skip through the definition of the projecting function and provide a shortcut to the reduced solution that it is being sought.

Kernel functions and PCA

The basis of *kernel Principal Component Analysis* (kPCA) is derived from the non-linear variant. Assuming that the data is mapped nonlinearly from its original base $\mathbb{R}^d$

to a high dimensional *feature space* $\mathcal{F}$, we can express this relation as [57]

$$\Phi : \boldsymbol{x} \in \mathbb{R}^d \to \Phi(\boldsymbol{x}) = \hat{\boldsymbol{x}} \in \mathcal{F} \subseteq \mathbb{R}^d, \tag{5.13}$$

where $\mathcal{F}$ is the space where a linear PCA can be performed if we choose the function Φ correctly. Assuming that the projected data is centered, i.e. $\sum_k^d \Phi(x_k) = 0$, we perform the same multiplication as before for the correlation matrix, namely

$$\begin{aligned} \hat{\boldsymbol{C}} &= \hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^T \\ &= \sum_{i=1}^{n} \Phi(\boldsymbol{x}_i)(\Phi(\boldsymbol{x}_i))^T. \end{aligned} \tag{5.14}$$

As for the linear case, the eigenvalues and vectors of matrix $\hat{\boldsymbol{C}}$ have to be found with the expression $\lambda \boldsymbol{V} = \hat{\boldsymbol{C}}\boldsymbol{V}$. Since all vectors $\boldsymbol{V}$ lie in the span of $\Phi(\boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x}_n)$, the system can be equivalently written as [58]

$$\lambda \Phi(\boldsymbol{x}_i) \cdot \boldsymbol{V} = \Phi(\boldsymbol{x}_i) \cdot \hat{\boldsymbol{C}}\boldsymbol{V}, \quad \forall k = 1, \ldots, n, \tag{5.15}$$

where there are coefficients $\alpha_1, \ldots, \alpha_n$ such that

$$\boldsymbol{V} = \sum_{i=1}^{n} \alpha_i \Phi(\boldsymbol{x}_i). \tag{5.16}$$

Substituting everything in Equation 5.15 and simplifying, we arrive at the expression

$$\lambda \boldsymbol{\alpha} = \boldsymbol{K} \tag{5.17}$$

where

$$K_{ij} = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j) \tag{5.18}$$

is a matrix of inner products between the projections of the dataset. The solutions α_k of Equation 5.17 are normalized by forcing the corresponding vector $\boldsymbol{v}_k$ to be normalized, i.e., having a norm of 1, which makes the base $\boldsymbol{V}$ orthonormal.

As in the linear case, to extract the principal components we must perform the rotation on the dataset by multiplying against the k highest eigenvectors (according to the tolerance set). Applying this to one point $\boldsymbol{x}_j$, we obtain

$$\boldsymbol{V}^k \cdot \Phi(\boldsymbol{x}_j) = \sum_{i}^{n} \alpha_i^k (\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)). \tag{5.19}$$

From here it can be seen that neither during the eigendecomposition of the problem in Equation 5.15 nor during the computation of the principal components in Equation 5.19

the explicit value of the function Φ is required, since we only need inner products between sample projections. Thus, we can use functions, called *kernel functions*, to approximate this unknown relations. In mathematical terms, this translates to

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) \approx \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{y}). \tag{5.20}$$

One definition that is important in kPCA is the term of *Gram matrix*. This matrix is defined as the matrix of inner products, meaning that each element ij represents the dot product between samples $\Phi(\boldsymbol{x}_i)$ and $\Phi(\boldsymbol{x}_j)$. If this matrix is approximated with the kernel functions it is often referenced as a *kernel matrix* $\boldsymbol{K}$, where each term is defined by $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. One of the conditions that the kernel function κ needs to follow in order to be a valid choice is that it needs to be defined in such a way that allows matrix $\boldsymbol{K}$ to be positive semi-definite, which means that each eigenvalue will be positive or zero. As was mentioned before, this condition is also fulfilled in the linear case given how matrix $\boldsymbol{C}$ is defined, so it is an imposed condition when projecting the data into higher spaces.

The selection of the kernel to use is a sensitive choice that mainly depends on the problem. Every kernel used gives results, but some provide a better insight on the data than others. In this case, there are a set of kernels that are normally used due to their simplicity of implementation and good performance. In this document we focus specifically on *gaussian kernels* [59], which are defined in a way similar to gaussian functions [59], namely

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{y}||^2}{b^2}\right), \tag{5.21}$$

where b is the parameter to define. Usually, b is taken as the mean of the distances between samples, so it is in the order of magnitude of the data, similar to the standard deviation. This kernel is also referred sometimes as a radial kernel. Other famous choice that is commonly used is the *polynomial kernel* [60], which is defined as an inner product raised to some power that needs to be defined. In a general sense, they can be written as

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + c)^d, \tag{5.22}$$

where c is a constant value that can be used to shift the functions, but typically is left as 0; and d which should be chosen depending on the distribution of the data. When we consider an unshifted polynomial kernel of order 1 (i.e., $c = 0$ and $d = 1$), we just revert back to the original case of linear PCA.

One final consideration is the centering of the kernel matrix. In linear PCA we start by taking centered samples according to Equation 5.5. In the kPCA case we start with the same centered samples, but there is no certainty that the samples will be centered in the feature space. Because of this, the centering of the kernel matrix is performed in the feature space before performing the eigendecomposition. Considering that the covariance matrix in the feature space can be expressed as $\hat{\boldsymbol{C}} = \hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^T$ (where $\hat{\boldsymbol{X}}$ is

unknown), we can center each dimension of the data as

$$\bar{\hat{\boldsymbol{x}}}_j = \hat{\boldsymbol{x}}_j - \frac{1}{n}\sum_i^n \hat{\boldsymbol{x}}_i, \tag{5.23}$$

meaning that we are centering the columns of $\hat{\boldsymbol{X}}$. In matrix form this is expressed as

$$\bar{\boldsymbol{X}} = \boldsymbol{X}\left(\mathbb{I}_n - \frac{1}{n}\mathbf{1}_n\right), \tag{5.24}$$

where the subindex n means that the matrix is of size $n \times n$. Performing the multiplication we obtain

$$\begin{aligned}\bar{\hat{\boldsymbol{C}}} = \bar{\hat{\boldsymbol{X}}}\bar{\hat{\boldsymbol{X}}}^T &= \hat{\boldsymbol{X}}\left(\mathbb{I}_n - \frac{1}{n}\mathbf{1}_n\right)\left(\mathbb{I}_n - \frac{1}{n}\mathbf{1}_n\right)^T\hat{\boldsymbol{X}}^T \\ &= \left(\hat{\boldsymbol{X}} - \frac{1}{n}\hat{\boldsymbol{X}}\mathbf{1}_n\right)\left(\hat{\boldsymbol{X}}^T - \frac{1}{n}\mathbf{1}_n\hat{\boldsymbol{X}}^T\right) \\ &= \hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^T - \frac{1}{n}(\hat{\boldsymbol{X}}\mathbf{1}_n)\hat{\boldsymbol{X}}^T - \frac{1}{n}\hat{\boldsymbol{X}}(\mathbf{1}_n\hat{\boldsymbol{X}}^T) - \frac{1}{n^2}(\hat{\boldsymbol{X}}\mathbf{1}_n)(\mathbf{1}_n\hat{\boldsymbol{X}}^T),\end{aligned} \tag{5.25}$$

which is equivalent to the elemental expression

$$\bar{\hat{C}}_{ij} = \hat{\boldsymbol{x}}_i\hat{\boldsymbol{x}}_j{}^T - \frac{1}{n}\sum_{k=1}^n \hat{\boldsymbol{x}}_k\hat{\boldsymbol{x}}_j{}^T - \frac{1}{n}\sum_{q=1}^n \hat{\boldsymbol{x}}_i\hat{\boldsymbol{x}}_q{}^T - \frac{1}{n^2}\sum_{k=1}^n\sum_{q=1}^n \hat{\boldsymbol{x}}_k\hat{\boldsymbol{x}}_q{}^T. \tag{5.26}$$

In general, performing kPCA is straightforward. The procedure is the same as the one shown in Algorithm 5.2, with the added step of computing the kernel matrix first. However, as was mentioned before in Section 5.2.2, the computation of the multiplication $C = XX^T$ is expensive when the amount of samples is too big. Even more, the size of the kernel matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ can become too big to store in memory. In linear PCA we got around this problem performing an SVD decomposition. In the non-linear case, since we do not know the untangling function Φ, this is impossible. Given that for performing a kPCA we only need the eigenvalues and vectors, an approximation method is proposed to estimate these quantities from reduced subsets of the original dataset.

Nystrom approximation

When performing kernel-based problems, the computational complexity required to find the solution is in the order of $\mathcal{O}(n^3)$ [61], which is the reason why the problem becomes unaffordable. The proposed solution, called the *Nyström method*, works by approximating the kernel matrix $\boldsymbol{K}$ with a reduced-rank matrix $\tilde{\boldsymbol{K}}$. This allows us to reduce the complexity of the problem to the order of $\mathcal{O}(m^2 n)$, where m is the rank of the reduced matrix. The Nyström method [62] was originally presented as a numerical way of solving integrals, similar to a Gaussian quadrature. Lately, with the rise in popularity of kernel-based methods it has made its way into the field to address the

size problems.

When considering a kernel $\kappa(\boldsymbol{x}, \boldsymbol{y})$ in a generalized form [61]

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{y}), \tag{5.27}$$

with λ_i being the eigenvalues arranged in a decreasing form and ϕ_i the respective eigenfunctions of the kernel κ, in a way that we have

$$\int \kappa(\boldsymbol{y}, \boldsymbol{x}) \phi_i(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} = \lambda_i \phi_i(\boldsymbol{y}), \tag{5.28}$$

where $p(\boldsymbol{x})$ is the probability density of $\boldsymbol{x}$. If the probability distribution is replaced by a set of distributed samples $\boldsymbol{x}_k$, Equation 5.28 can be approximated as

$$\frac{1}{q} \sum_{k=1}^{q} \kappa(\boldsymbol{y}, \boldsymbol{x}_k) \phi_i(\boldsymbol{x}_k) \approx \lambda_i \phi_i(\boldsymbol{y}). \tag{5.29}$$

The eigenvectors are p-orthogonal, meaning that $\int \phi_i(\boldsymbol{x}) \phi_j(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} = \delta_{ij}$. This is equivalent to a constraint in the discrete problem of the form $\frac{1}{q} \sum_{k=1}^{q} \phi_i(\boldsymbol{x}_k) \phi_j(\boldsymbol{x}_k) \approx \delta_{ij}$. In matricial form, Equation 5.29 can be expressed as

$$\boldsymbol{K}^{(q)} \boldsymbol{U}^{(q)} = \boldsymbol{U}^{(q)} \boldsymbol{\Lambda}^{(q)}, \tag{5.30}$$

considering $\boldsymbol{K}^{(q)}$ as the kernel matrix of a dataset of q samples, called *landmark points*, $\boldsymbol{U}^{(q)}$ are the orthonormal eigenvectors of $\boldsymbol{K}^{(q)}$ and $\boldsymbol{\Lambda}^{(q)}$ are the decreasing eigenvalues as a diagonal matrix. If the q samples are a subset of the original dataset, taking $\boldsymbol{x}_j$ as $\boldsymbol{y}$ in Equation 5.29 and matching it to Equation 5.30, we obtain

$$\begin{aligned} \phi_i(\boldsymbol{x}_j) &\approx \sqrt{q} \boldsymbol{U}_{ji}^{(q)}, \\ \lambda_i &\approx \frac{\lambda_i^{(q)}}{q} \end{aligned} \tag{5.31}$$

which can be plugged again into Equation 5.29 to get

$$\phi_i(\boldsymbol{y}) \approx \frac{\sqrt{q}}{\lambda_i^{(q)}} \sum_{k=1}^{q} \kappa(\boldsymbol{y}, \boldsymbol{x}_k) \boldsymbol{U}_{ki}^{(q)} = \frac{\sqrt{q}}{\lambda_i^{(q)}} \boldsymbol{\kappa}_y \cdot \boldsymbol{u}_i^{(q)}, \tag{5.32}$$

where $\boldsymbol{\kappa}_y$ accounts for the vector $[\kappa(\boldsymbol{x}_1, \boldsymbol{y}), \ldots, \kappa(\boldsymbol{x}_q, \boldsymbol{y})]^T$ and $\boldsymbol{u}_i^{(q)}$ is the i-th column of $\boldsymbol{U}^{(q)}$. Equation 5.31 and Equation 5.32 is what is referred as the Nyström approxi-

mation of the i-th eigenfunction. In matricial form they are respectively

$$\boldsymbol{\Lambda} = \frac{1}{q}\boldsymbol{\Lambda}^{(q)}, \tag{5.33a}$$

$$\boldsymbol{U} = \sqrt{q}\boldsymbol{K}_{q,y}\boldsymbol{U}^{(q)}{\boldsymbol{\Lambda}^{(q)}}^{-1}. \tag{5.33b}$$

When performing the kPCA analysis, what we look for, i.e. the principal components, are basically just combinations of the eigenvalues and vectors of the original kernel matrix. The Nyström technique shown above allows us to directly compute an approximation of both eigenvalues and vectors without the need of explicitly creating the original matrix $\boldsymbol{K}$. If we consider the kernel matrix $\tilde{\boldsymbol{K}} \in \mathbb{R}^{m\times m}$ as the kernel matrix computed with a subset of m points (as seen in Figure 5.3), it can be eigendecomposed as always with Equation 5.6 into

$$\boldsymbol{K} \approx \tilde{\boldsymbol{K}} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{U}}, \tag{5.34}$$

with both $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{\Lambda}}$ being the eigenvectors and values of the reduced matrix. By matching Equation 5.33 against the matrices from Equation 5.34, we reach

$$\boldsymbol{\Lambda} \approx \frac{n}{m}\tilde{\boldsymbol{\Lambda}}, \tag{5.35a}$$

$$\boldsymbol{U} \approx \sqrt{\frac{m}{n}}\boldsymbol{K}_{nm}\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Lambda}}^{-1}, \tag{5.35b}$$

where $\boldsymbol{K}_{mn} = \kappa(\boldsymbol{x}_m, \boldsymbol{x}_n)$ is the portion of the kernel matrix that relates the subset of m landmark points against all points in the dataset.

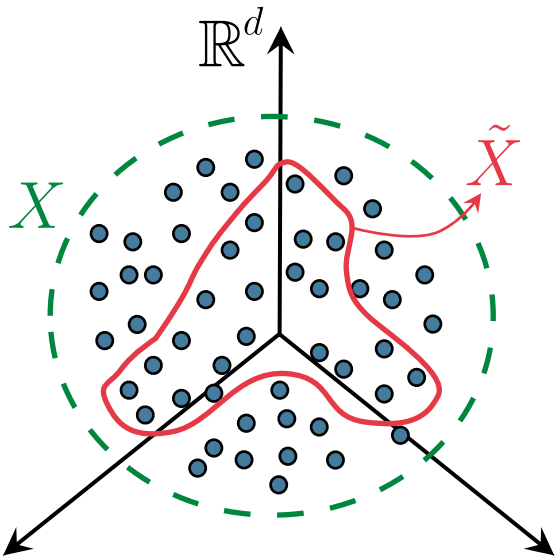


Figure 5.3: Diagram of the idea behind the Nyström approximation. A subset $\tilde{\boldsymbol{X}}$ of points is chosen from the full dataset $\boldsymbol{X}$ to perform the matrix calculations. The subset of points has to represent the properties of the full dataset, so they need to be chosen properly to avoid over-representing certain features.

As can be seen, if we consider $m << n$ the computation complexity of the problem reduces drastically, since the main problem is reduces from a complexity of $\mathcal{O}(n^3)$ to $\mathcal{O}(m^2 n)$, while the accuracy of the solution does not decrease dramatically with a low amount of points [61].

The choice of the points to perform the approximation is approached in many articles. Since the data selected should be representative of the full dataset, we need them to be properly distributed in the space to avoid a shift in the principal components due to variations of the mean values and their standard deviation. A common approach was to select the points randomly [61]: if a certain amount of points is taken we can expect them to be equally distributed without a bias. However, these points do not have to necessarily belong to the dataset. Proposed by Zhang and Kwok [63], the idea of clustered Nyström proposes another solution for this sampling. As its name indicates, the method works by using grouping techniques that allows to find the centroids of subgroups of points in the dataset. In their work, they use particularly the k-means technique [32] to identify them. The main idea is to approximate the kernel matrix through the use of the k points obtained by k-means. It is known that the Nyström approximation of $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is exact if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the subset of landmark points, but this only happen with a reduced amount of samples. Because of this, it is proposed that the landmark points are actually the centroids of m disjoint clusters in the dataset, since each point would represent an average of the cluster. This approach allows to avoid the selection of the landmark points and gives a relatively accurate solution, since the total error for the approximation of the kernel matrix is bounded [63].

The clustered approach is of interest for our particular case, since we do not have the need to perform an extra step for clustering. In the DDI case, we can take advantage of the material points, which already represent the centroids of N^* clusters of mechanical points, which are the ones we have been using in our PCA approach to DDI.

5.2.4 Application of principal components in DDI

The use of both PCA and kPCA methods in the DDI algorithm follow similar considerations as the normalized case. We need to be careful of the fact that the mechanical states can only be computed in the phase space, so we will be projecting again back and forth between different spaces.

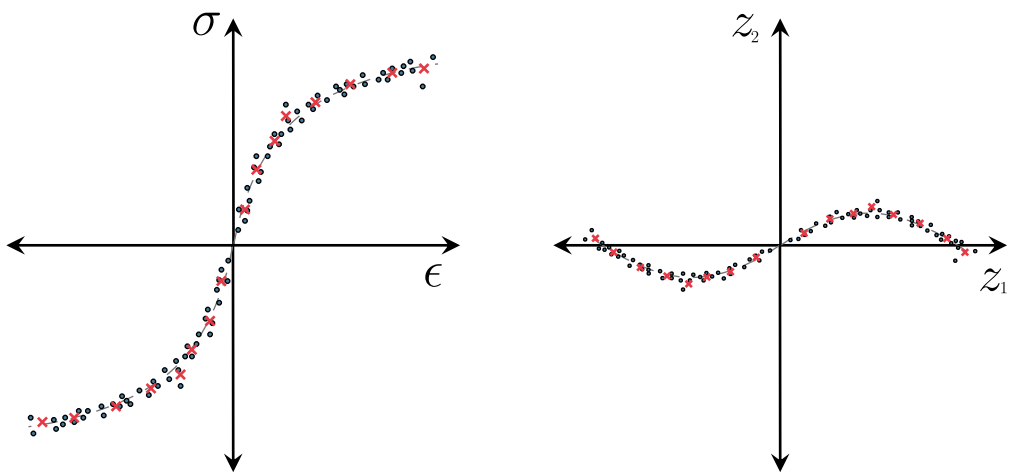
To perform this variant of DDI we start by normalizing the mechanical states according to Equation 5.5. As in the normalized case, we have to perform this operation in each row of the data matrix, meaning that each component of the strain and stress tensor are considered independently. As an example, we consider the plane stress case, where $\boldsymbol{\epsilon} = [\epsilon_{xx}\ \epsilon_{yy}\ \epsilon_{xy}]^T$ and $\boldsymbol{\sigma} = [\sigma_{xx}\ \sigma_{yy}\ \sigma_{xy}]^T$. In this case, each sample $\boldsymbol{x}_i$ can be written as

$$\boldsymbol{x}_i = [(\epsilon_{xx})_e^X\ (\epsilon_{yy})_e^X\ (\epsilon_{xy})_e^X\ (\sigma_{xx})_e^X\ (\sigma_{yy})_e^X\ (\sigma_{xy})_e^X]^T, \tag{5.36}$$

this time as a column vector, where e represents each element and X each snapshot from where the data point was taken. This gives us a data matrix $\boldsymbol{X}$ of size $6 \times (N^e \times n^X)$ of the form

$$\boldsymbol{X} = \begin{bmatrix} \epsilon_{xx}^1 & \epsilon_{yy}^1 & \epsilon_{xy}^1 & \sigma_{xx}^1 & \sigma_{yy}^1 & \sigma_{xy}^1 \\ \epsilon_{xx}^2 & \epsilon_{yy}^2 & \epsilon_{xy}^2 & \sigma_{xx}^2 & \sigma_{yy}^2 & \sigma_{xy}^2 \\ \vdots & & & & & \vdots \\ \epsilon_{xx}^{N^s} & \epsilon_{yy}^{N^s} & \epsilon_{xy}^{N^s} & \sigma_{xx}^{N^s} & \sigma_{yy}^{N^s} & \sigma_{xy}^{N^s} \end{bmatrix}. \tag{5.37}$$

In the case of linear PCA, after normalizing each column, an SVD is performed to obtain the decomposition of the matrix according to Equation 5.11. These matrices are used to compute the principal components $\boldsymbol{Z}$ using Equation 5.7. A graphical display of how this transformation affects a dataset for a truss case is shown in Figure 5.4. The case for the trusses is shown given that its 2D representation has a better visualization, but the same effect applies to any number of dimensions without loss of generality.



(a) Dataset in phase space

(b) Dataset in projected space

Figure 5.4: Transformation of the dataset when PCA is applied. As it can be seen, the projected space is just a rotation of the data to fit its principal axes, where the covariance is the lowest.

In the case of kPCA, before performing the SVD we obtain the kernel matrix with our chosen kernel according to Equation 5.21 and 5.22. Once the matrix is obtained it is centered in the feature space with Equation 5.25. This centered matrix is either subjected to the SVD procedure to obtain the principal components in a similar fashion to the linear case or subjected to the Nyström procedure to obtain the approximated components.

Now the process continues as in the normalized case. In this feature space we can find the state mapping by minimizing the distances between each projected mechanical state (i.e., $\boldsymbol{z}_i$) and the projected material states ($\boldsymbol{z}^*_{\text{ie}X}$) using again a nearest neighbors search or a k-means clustering. The main difference here is that in this part of the process we can perform the dimensionality reduction. If the first principal components carry most of the information from the data matrix $\boldsymbol{X}$, we can omit the last ones and perform the clustering algorithm in a set with smaller dimensions, which improves the speed of the minimization.

After the projected material points are found, the inverse process is performed to obtain the material points using Equation 5.9 in the linear case. In here we know that the material states do not belong to the original dataset, but since $\boldsymbol{U}$ correspond to an orthonormal base the transformation still holds. These reconstituted material states are normalized, so the normalization is removed performing the opposite relation as was done in Equation 5.4, namely

$$(\boldsymbol{x}^*_{\text{ie}X})_d = \mu_{\boldsymbol{x}_d} + ||(\boldsymbol{x})_d||(\boldsymbol{z}^*_{\text{ie}X})_d, \tag{5.38}$$

where d corresponds to each one of the dimensions of each sample $\boldsymbol{x}$.

In the kernel version, the issue of the backwards projection from the feature space needs to be addressed. Since the projection function is not known, there is not a proper way to track back the material states, considering that they do not have a pre-image in the phase space. A proper approach is to consider that the pre-image $\boldsymbol{x}^*$ of a point $\boldsymbol{z}^*$ in the feature space can be obtained as a weighted average of the points $\boldsymbol{x}_j$ [64]. In this

regard, the idea is that, the closer the points $\boldsymbol{z}_j$ are to the new point $\boldsymbol{z}^*$ in the feature space, the closer the points $\boldsymbol{x}_j$ will also be to $\boldsymbol{x}^*$ in the phase space. For simplicity, in this document we will take advantage of the clustering performed and we will consider the material point in the phase space as an average of the mechanical points associated to it in the feature space.

Now, we proceed in a similar fashion as the normalized case: the mechanical states are updated with Equation 2.34c and redo each iteration until the desired tolerance is reached. The full procedure is outlined in Algorithm 5.2.

5.3 Comparison of all presented methods

The presented methods in Section 5.2 introduce a complexity in the computation of DDI, which can affect both accuracy and efficiency. In the present section we aim to analyze the performance of all the new methods introduced and compare them against the original DDI algorithm used in previous chapters to see the advantages and disadvantages of the new formulation.

5.3.1 Analysis of performance

As mentioned above, our main focus is to check how the accuracy of the stress estimations compares between the different methods, as well as how long does it take to converge to the same criteria in each one. It might be worth to have a bit worse accuracy if the algorithm behaves much faster or vice-versa, but not if results stay in similar ranges.

Since the DDI algorithm and its derivatives are random in nature, we perform repetitions of the same problem in order to obtain an average and dispersion of the results. With this we want to study in particular four points in the algorithm: accuracy, initialization time, convergence time and accuracy. These points will be studied in the cases we have seen earlier, trusses for the scalar problem, plane stress for a more conventional analysis and viscoelasticity to assess the dimensionality issue.

Truss case

The aim of this analysis is not to test extreme cases, but to study the behavior of the methods in a common and simple setting. For the case of the truss, we will work with an heterogeneous sample in a similar fashion to the ones studied in Chapter 3.

The sample will be a rectangular membrane, shown in Figure 5.5. It is embedded with stiffer inclusions in a ratio of stiffnesses of $r_E = 10$ as defined in Equation 3.2. The matrix is modeled with stiffness $E_m = 1$ to follow the same procedure as the previous chapter. The sample is meshed with triangles, where each edge of the triangle is considered as a bar element with one integration point. In this case, the mesh is composed of $N^n = 904$ nodes and $N^e = 2629$ elements. We use the same loading cases shown in Figure 3.1 but we consider the force applied in 20 steps, so we have $N^X = 160$ snapshots. This amounts to a total of 420640 mechanical points, which are clustered with $N^* = 1400$ material points, meaning that there is a ratio $r^* = 300$ between mechanical and material states. For the constant $\mathbb{C}$ we choose the unity, so we can let the algorithm converge.

Algorithm 5.2: DDI procedure with PCA.

Output: $\boldsymbol{\epsilon}_{\mathrm{i}}^{*}$, $\boldsymbol{\sigma}_{\mathrm{i}}^{*}$, $\boldsymbol{\sigma}_{e}^{X}$, ie^{X}
Input: N^{*}, $\boldsymbol{u}_{j}^{X}$, $\boldsymbol{f}_{j}^{X}$, $\boldsymbol{B}_{ej}^{X}$, $\boldsymbol{\epsilon}_{e}^{X}$

PCA DDI procedure

Set $k \rightarrow 0$.

Initialization: set $\boldsymbol{\sigma}_{e}^{X(k=0)} = 0$ for each element e in every snapshot X. Likewise, initialize $\boldsymbol{\sigma}_{\mathrm{i}}^{*(k=0)} = 0$ for each material state i.

Initialization: data matrix $\boldsymbol{X}^{(k=0)}$ is formed by stacking each mechanical state $(\boldsymbol{\epsilon}_{e}^{X\,(k=0)}, \boldsymbol{\sigma}_{e}^{X\,(k=0)})$ from element e and snapshot X in columns as shown in Equation 5.37.

Initialization: perform the normalization of each row of $\boldsymbol{X}^{(k=0)}$ with Equation 5.5.

Initialization: perform an SVD of this normalized matrix $\tilde{\boldsymbol{X}}^{(k=0)}$ to obtain the principal components $\boldsymbol{Z}$, as in Equation 5.11.

Perform a k-means clustering on the feature space formed by $\boldsymbol{Z}^{(k=0)}$ to obtain the projected material states $\boldsymbol{z}_{i}^{*(k=0)}$ and the state mapping ie^{X}.

while *convergence criterion for* $\boldsymbol{\epsilon}_{i}^{*}$ *and* $\boldsymbol{\sigma}_{i}^{*} <$ *tolerance* **do**

Project back from the feature space $\boldsymbol{Z}^{(k)}$ using Equation 5.9 to obtain the material states $(\boldsymbol{\epsilon}_{\mathrm{i}}^{*(k)}, \boldsymbol{\sigma}_{\mathrm{i}}^{*(k)})$.

for $X = 1, \ldots, N^{X}$ **do**

Solve equations

$$\sum_{k}\sum_{e} w_{e}^{X}\boldsymbol{B}_{ej}^{X\,T} : \mathbb{C} : \boldsymbol{B}_{ek}^{X} \cdot \boldsymbol{\eta}_{k}^{X(k)} - \sum_{e} w_{e}^{X}\boldsymbol{B}_{ej}^{X\,T}\boldsymbol{\sigma}_{\mathrm{ie}^{X}}^{*(k)} = \boldsymbol{f}_{j}^{X} \quad \forall j, X;$$

$$\boldsymbol{\sigma}_{e}^{X(k+1)} = \boldsymbol{\sigma}_{\mathrm{ie}^{X}}^{*(k)} + \sum_{j} \mathbb{C} : \boldsymbol{B}_{ej}^{X} \cdot \boldsymbol{\eta}_{j}^{X(k)} \quad \forall e, X.$$

with current values of $\boldsymbol{\sigma}_{\mathrm{i}}^{*(k)}$ to obtain updated values of $\boldsymbol{\sigma}_{e}^{X(k+1)}$.

Form data matrix $\tilde{\boldsymbol{X}}^{(k+1)}$ by stacking the updated mechanical states and normalizing the columns.

Perform an SVD on $\tilde{\boldsymbol{X}}^{(k+1)}$ to obtain the principal components $\boldsymbol{Z}^{(k+1)}$.

Perform a k-means clustering on $\boldsymbol{Z}^{(k+1)}$ to obtain the projected material states $\boldsymbol{z}_{i}^{*(k+1)}$ and the new state mapping ie^{X}.

Set $k \rightarrow k+1$

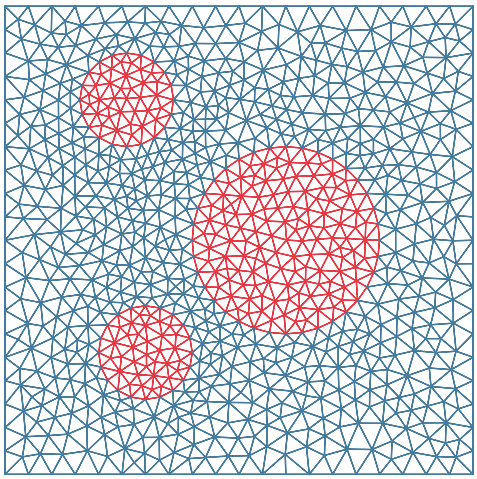
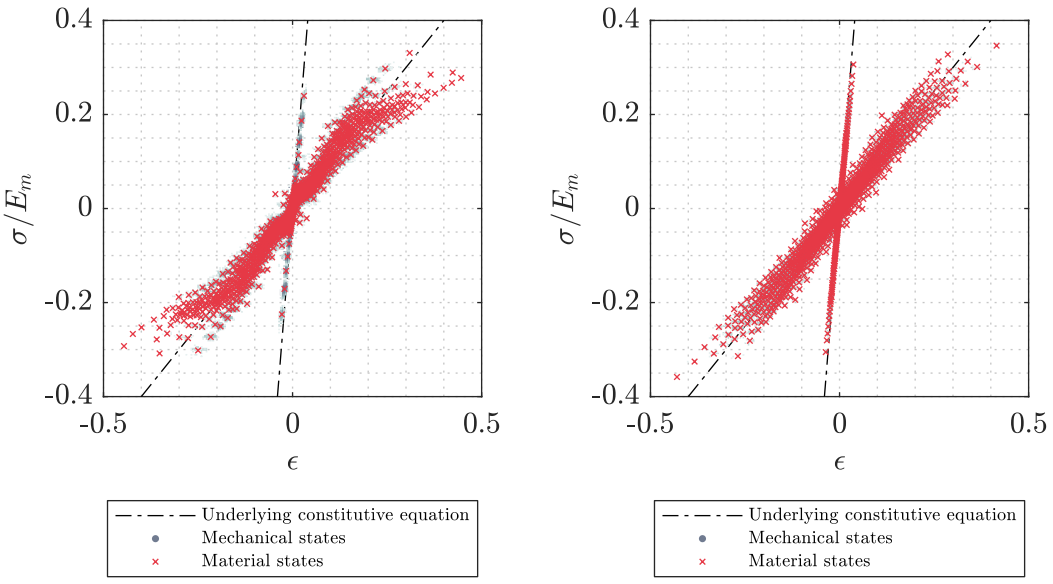


Figure 5.5: Heterogeneous mesh for truss example.

If we run the original DDI algorithm, we obtain the results shown in Figure 5.6a. This is contrasted against the results from the normalized algorithm in Figure 5.6b. As can be seen from the images, using a normalized approach for DDI tends to move the mechanical states closer to the reference lines given by the underlying constitutive model used for generating the data. We can see this also on the material states, which now represent more accurately the specified behavior.



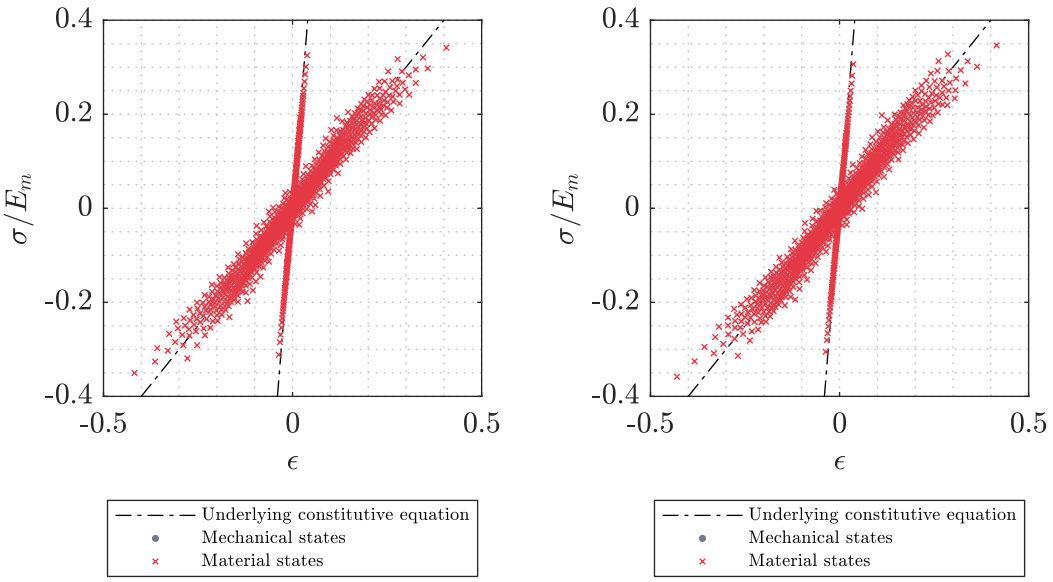
(a) Phase space results for original DDI.

(b) Phase space results for normalized DDI.

Figure 5.6: Phase space results for different DDI algorithms.

The results for the PCA algorithm are shown in Figure 5.7a. Next to it we repeat the results from the normalized algorithm from Figure 5.6b. This is done to show the similarity of the results, given the procedure performed. Both results tend to show similar results due to the way they are formulated. As was explained before, PCA requires the normalization of the columns that are used in the algorithm since we are

dealing with different units. This in fact makes the PCA algorithm behave in the same way as the normalized one, only with the exception that the data is rotated around the origin to fit the axes with smallest covariance. This case is only valid as long as we are not reducing dimensions when using PCA, which is something that is not done in the truss case since we only consider two.



(a) Phase space results for PCA DDI.

(b) Phase space results for normalized DDI.

Figure 5.7: Phase space results for different DDI algorithms.

Some of the aspects of the algorithm are compared in Table 5.1. We can clearly notice that both modified algorithms are not efficient in comparison to normal DDI. We have an increased initialization time due to the transformations and clustering that is required in the beginning of a new run. Convergence time is also high, but if we check against the number of iterations we can see that the performance of all three algorithms are not so far apart. In fact, we can see that the modified algorithms tend to have troubles converging to the desired solution, but in general they have a similar performance. The biggest advantage from using this new formulations comes from the accuracy of the estimations. We can see an improvement of both algorithms with respect to the original DDI, despite the increased computation.

Table 5.1: Performance comparison for truss case.

	Original DDI	Normalized DDI	PCA DDI
Initialization time (s)	1.519 ± 0.350	11.844 ± 1.507	11.135 ± 0.157
Convergence time (s)	67.144 ± 2.601	280.788 ± 44.149	261.083 ± 24.016
Number of iterations	136.0 ± 0.000	492.0 ± 20.575	493.3 ± 21.187
Time per iteration (s)	0.494 ± 0.019	0.570 ± 0.079	0.529 ± 0.037
2-norm error	0.268 ± 0.000	0.172 ± 0.005	0.176 ± 0.007

This error can be captured better if we use the histogram plots. In here we notice that both curves for the new algorithms are very similar, and that they predict much better the mechanical states.

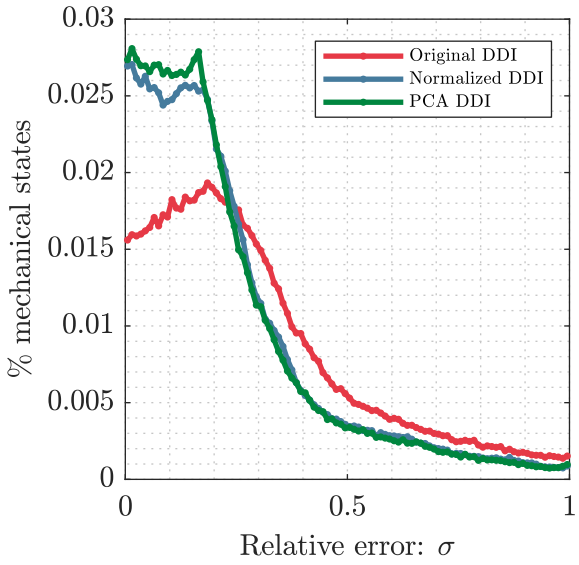


Figure 5.8: Error histogram comparing algorithms.

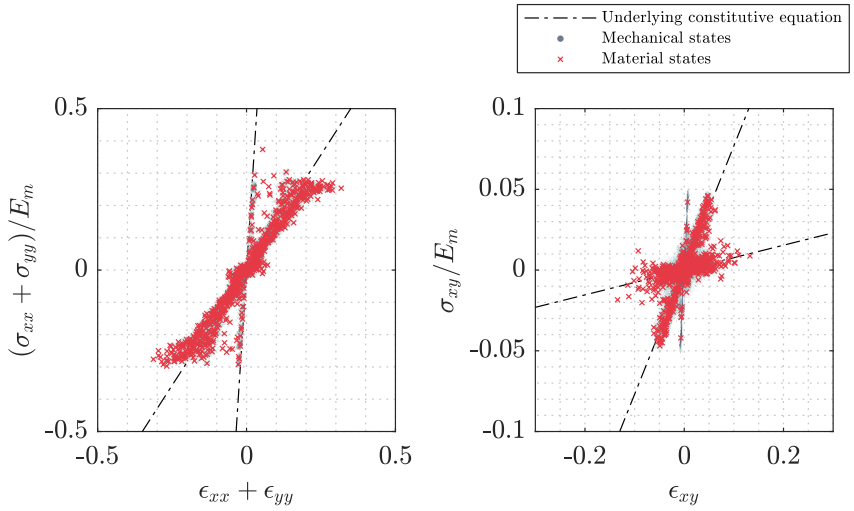
During the process of analyzing the algorithms we have decided not to consider a kernel approach for DDI. PCA has the advantage that the performance of the SVD is a fast process, however, the Nyström approximation, while it improves performance dramatically in kPCA, it is still too inefficiently applied in our algorithm to provide a useful result. Kernel methods will however be used in Section 5.4, since we do not need to address the computation time or refer to recursive schemes.

Plane Stress case

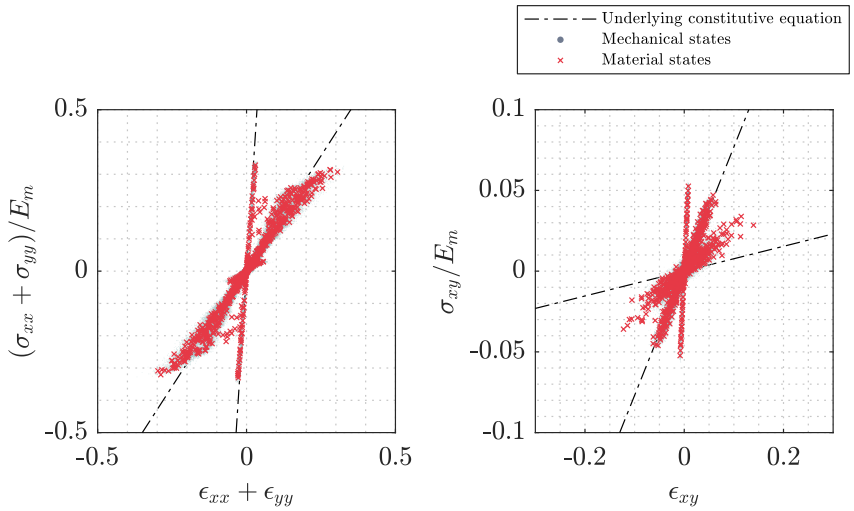
For the plane stress case we work with the same data as for the truss. The mesh is the same one from Figure 5.5, but now we consider the triangles as the elements with one integration point. This keeps the amount of nodes equal at $N^n = 904$, but changes the number of elements to $N^e = 1726$, leaving us with 276160 mechanical points and $N^* = 920$ material states. The rest of the parameters is left the same, only adapting $\mathbb{C} = \mathbb{I}_{3\times 3}$ to keep the dimensions accordingly.

The difference from both the classical DDI approach and the normalized algorithm are more pronounced in this case. In Figure 5.9 we can see that using the normalized version of DDI tends to sharpen the results in the trace component of the tensor, so we have less dispersion of the points around the underlying behavior. The same effect happens with the shear component, but in here we see that the values of the matrix phase of the sample differ from the expected curve. In both cases we have points that gather on a vertical line at the origin, which can be attributed to errors in the DDI process of the data. These points could be corrected with a more adequate choice of the parameter $\mathbb{C}$.

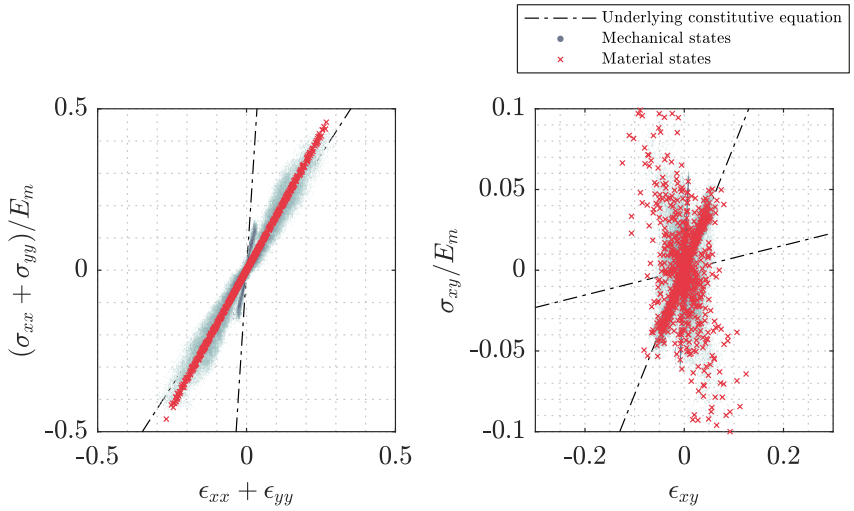
The case of the PCA DDI with reduced dimensions is shown as a reference, since the algorithm clearly does not produce proper results. This is also reflected on the histogram plot in Figure 5.10, where we can see that the error is similar to the original DDI, but when visualized in the phase space we have a distribution of the points that is unclear and does not fit properly the underlying behaviors. This can be attributed to the fact that we are only considering 5 out of the 6 principal components that are available. In the case of plane stress where each one of the six dimensions are relevant,



(a) Phase space results for original DDI.



(b) Phase space results for normalized DDI.



(c) Phase space results for PCA DDI.

Figure 5.9: Phase space results for different DDI algorithms.

removing one of them becomes problematic. Adding the results of performing the PCA algorithm considering all the dimensions serves as a contrast for this effect: in here the effectiveness of the algorithm is increased and we have similar results to what we observe when using the normalized version of the algorithm, similar to what we saw on the truss case.

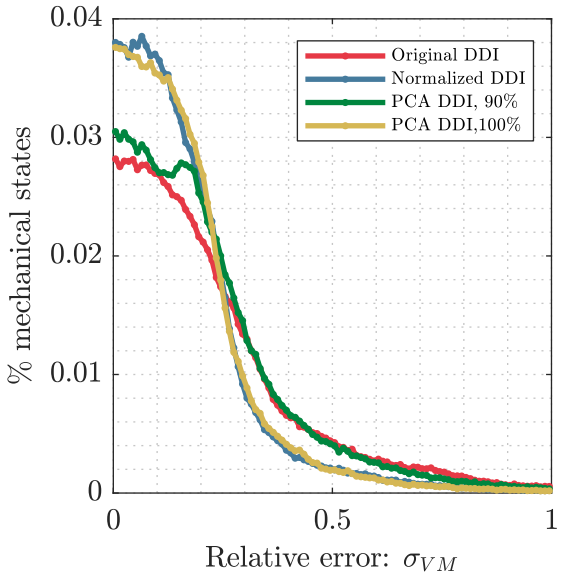


Figure 5.10: Error histogram comparing algorithms.

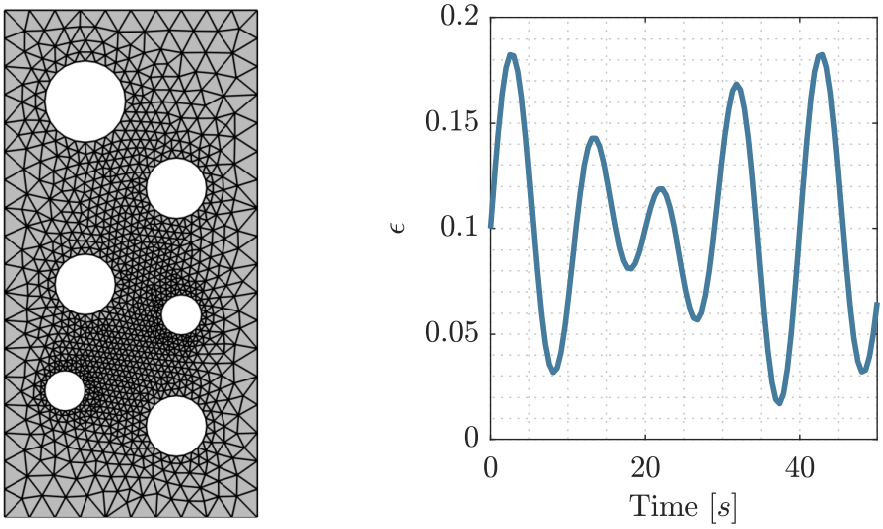
Checking the same parameters as for the truss case, the results are shown in Table 5.2. We can notice that for both convergence time and iterations the PCA algorithm tends to have a big dispersion, which explains in part the bad results. Checking the total error of the system we see that, even when the distribution of the error for the mechanical states is similar between PCA and the original DDI, the performance of the PCA algorithm with trimmed dimensions is worse, but when all dimensions are taken the values are in similar ranges to the normalized procedure.

Table 5.2: Performance comparison for plane stress case.

	Original DDI	Normalized DDI
Initialization time (s)	1.6063 ± 0.2126	13.4863 ± 0.7958
Convergence time (s)	54.5952 ± 1.1921	140.0424 ± 5.7927
Number of iterations	31.0 ± 0.0000	288.0 ± 33.8001
Time per iteration (s)	1.7611 ± 0.0385	0.4907 ± 0.0444
2-norm error	0.2913 ± 0.0000	0.2165 ± 0.0014
	PCA DDI (90%)	PCA DDI (100%)
Initialization time (s)	9.9650 ± 0.4661	11.3745 ± 0.8591
Convergence time (s)	36.3866 ± 35.4598	134.6251 ± 18.0822
Number of iterations	41.8 ± 35.2824	288.3 ± 41.0259
Time per iteration (s)	3.2995 ± 3.2494	0.4678 ± 0.0215
2-norm error	0.3961 ± 0.0890	0.2205 ± 0.0035

Viscoelasticity case

For the viscoelastic case, since we do not work with heterogeneous sample, we use a different mesh, shown in Figure 5.11a. We take a rectangular membrane with holes to avoid homogenization and we take triangle elements with one integration point, having $N^n = 1203$ nodes and $N^e = 2216$ elements. The deformation profile applied in the upper border is a combination of sines as seen in Figure 5.11b, which is computed in 100 steps. this gives an amount of $N^* = 740$ material points for 221600 mechanical states. As in Chapter 4, we define the material with a Zener model with the stiffness of the elastic branch as $E_0 = 1$ and a relaxation time of $\lambda = 1$. The viscoelastic branch of the model takes a stiffness of $E_1 = 10$, which represents a highly viscoelastic case. The speed of the experiment is slow, with $D_v = 0.1$, for a better visualization of the viscoelastic behavior. As in the plane stress case, we consider $\mathbb{C}$ as an identity matrix.

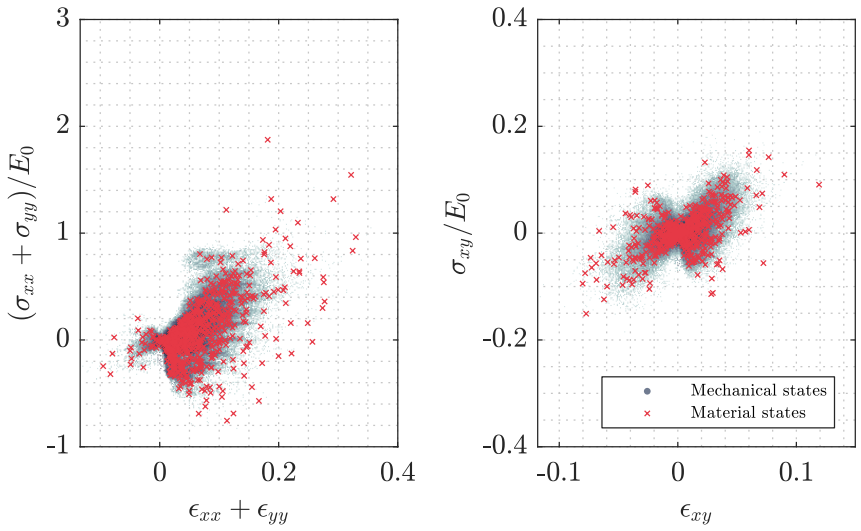


(a) Mesh for viscoelastic case.

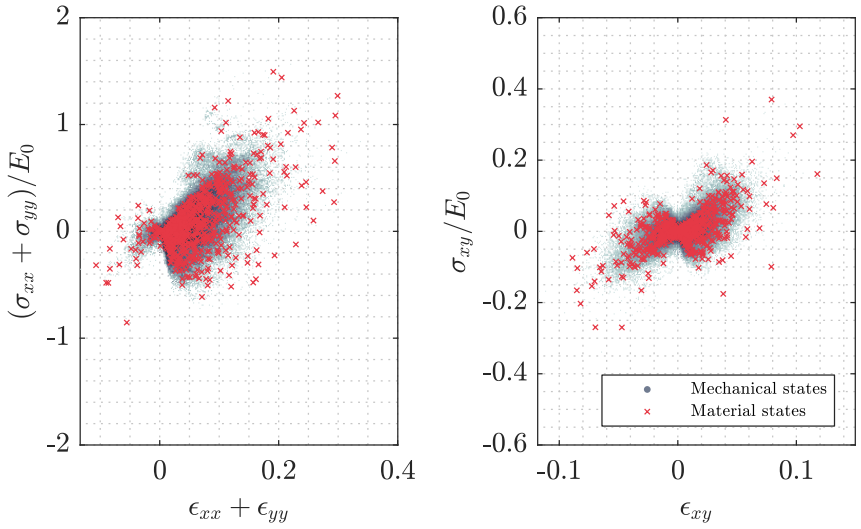
(b) Deformation for viscoelastic case.

Figure 5.11: Mesh and deformation profile for viscoelastic case.

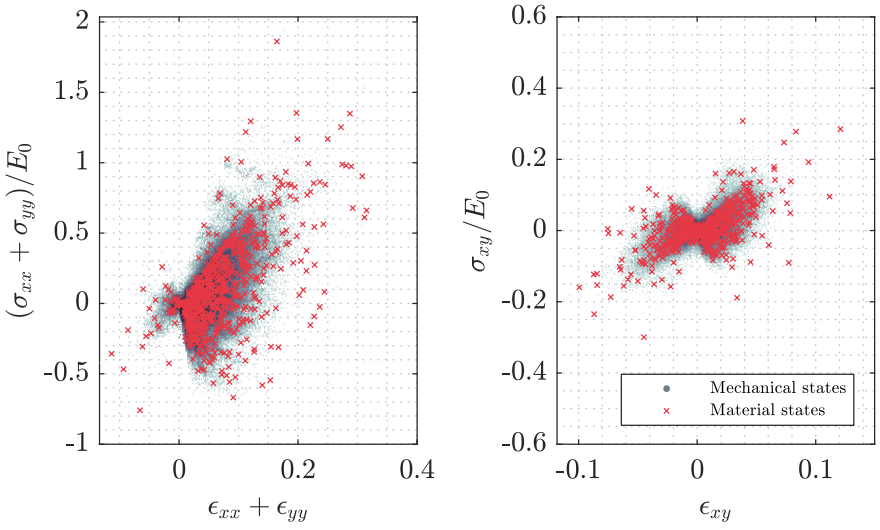
In the phase space plots we show only the cases for DDI when considering 20 previous timesteps in Figure 5.12. We can notice that both in the trace and shear component of the phase space the modified algorithms reduce the dispersion of the points, which is more noticeable in the PCA case.



(a) Phase space results for original DDI.



(b) Phase space results for normalized DDI.



(c) Phase space results for PCA DDI.

Figure 5.12: Phase space results for different DDI algorithms.

If we plot all the errors for the three algorithms and the five different amount of previous steps, we can have a generalized view of the problem. Figure 5.13 shows that the original DDI particularly behaves badly when considering lower amount of previous steps, which is the expected behavior, in line with the results seen in Chapter 4. However, we can see that, for the modified algorithms, the amount of previous steps that we use to estimate the mechanical stresses does not have a very big effect, since the results tend to stay in a similar range of error. In particular, PCA algorithm behaves better than the normalized one, given that we are now able to reduce the dimensionality of the problem without losing much information.

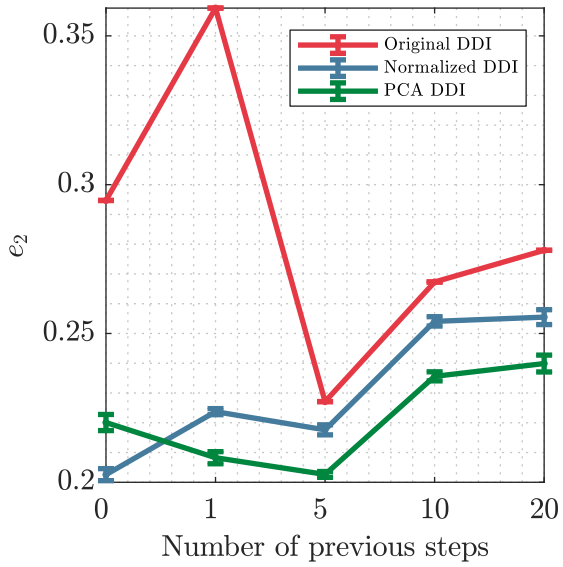


Figure 5.13: Error plot comparing algorithms.

In Table 5.3 we can see that the time per iteration for the PCA algorithm is much faster than the normalized version, and it is mostly in par with the original DDI. Even though PCA is a less efficient algorithm, we can see that reducing the dimensionality is beneficial for the clustering problem.

Table 5.3: Performance comparison for viscoelastic case.

	Original DDI	Normalized DDI	PCA DDI
0 previous steps			
Initialization time (s)	1.105 ± 0.060	14.121 ± 0.610	7.281 ± 0.100
Convergence time (s)	26.996 ± 0.135	61.741 ± 20.131	49.834 ± 5.038
Number of iterations	38.0 ± 0.000	182.8 ± 112.447	95.0 ± 17.448
Time per iteration (s)	0.710 ± 0.004	0.394 ± 0.109	0.540 ± 0.046
2-norm error	0.295 ± 0.000	0.203 ± 0.002	0.220 ± 0.003
1 previous step			
Initialization time (s)	1.306 ± 0.065	19.296 ± 0.203	8.896 ± 0.220
Convergence time (s)	23.581 ± 0.101	107.647 ± 34.865	45.004 ± 3.993
Number of iterations	30.0 ± 0.000	284.8 ± 150.872	95.0 ± 16.138
Time per iteration (s)	0.786 ± 0.003	0.429 ± 0.113	0.481 ± 0.055
2-norm error	0.359 ± 0.000	0.224 ± 0.001	0.208 ± 0.002
5 previous steps			
Initialization time (s)	1.664 ± 0.052	50.632 ± 0.989	10.076 ± 0.209
Convergence time (s)	21.033 ± 0.077	189.106 ± 52.727	30.928 ± 2.638
Number of iterations	22.0 ± 0.000	157.6 ± 105.729	34.8 ± 4.290
Time per iteration (s)	0.956 ± 0.004	1.430 ± 0.393	0.893 ± 0.042
2-norm error	0.227 ± 0.000	0.218 ± 0.002	0.203 ± 0.001
10 previous steps			
Initialization time (s)	2.258 ± 0.053	117.944 ± 1.241	12.908 ± 0.459
Convergence time (s)	32.161 ± 0.153	310.491 ± 16.859	41.203 ± 1.755
Number of iterations	27.0 ± 0.000	69.9 ± 2.331	31.2 ± 1.932
Time per iteration (s)	1.191 ± 0.006	4.443 ± 0.221	1.323 ± 0.063
2-norm error	0.267 ± 0.000	0.254 ± 0.002	0.236 ± 0.002
20 previous steps			
Initialization time (s)	3.166 ± 0.041	244.212 ± 1.750	20.394 ± 0.384
Convergence time (s)	39.915 ± 0.129	598.064 ± 58.218	83.568 ± 4.256
Number of iterations	23.0 ± 0.000	74.9 ± 12.600	37.2 ± 1.989
Time per iteration (s)	1.735 ± 0.006	8.122 ± 1.154	2.248 ± 0.089
2-norm error	0.278 ± 0.000	0.256 ± 0.003	0.240 ± 0.003

5.3.2 Practical example: softer inclusions

One of the issues that we have addressed earlier in this work is the case where the stiffness of the inclusions is lower than the matrix. The fact that the boundary of the inclusion is subjected to the displacement of the stiffer surrounding material limits the deformation that the inclusion would observe compared to the case it were free, which in turn might hide the real value of the stresses. If the camera we are using for the imaging technique is not good enough, we could mistakenly assume that the sample is homogeneous or that the inclusions are harder than what they really are. In this section we want to see if using the alternative definitions for DDI can helps us identify better the phase and behavior for these inclusions.

One of the interests for studying this case is localized failure in a sample. A softer inclusion is used to represent how a certain section of the material that has already failed behaves. If the material has completely failed, we can assume that the inclusion will have zero stiffness and the deformation of that section is infinite, but confined to

the boundaries of the inclusion. If the failure is partial we can have a reduced value for stiffness with similar effects. In the example analyzed here we assume that the inclusions are areas with the stiffness reduced to 50% of the original stiffness $E_m = 1$, which would also trigger the problems we have seen in the correspondence analysis method in Section 3.5.1.

We consider the same mesh from Section 3.5.2 in plane stress, as seen in Figure 3.12b. Since this problem is more complicated for DDI, we profit from having a richer database. Because of this, we apply deformations to the mesh in a similar fashion as seen in Figure 3.1 rather than the more realistic setting that was used originally in the mesh. The matrix of the sample is considered to have a Young's modulus of $E_m = 1$ and a Poisson's ratio of $\nu = 0.3$, while the inclusions are taken with $E_i = 0.5$. The mesh considers $N^n = 941$ nodes and $N^e = 1792$ elements. Considering $N^X = 80$ snapshots, the total number of mechanical states amounts to 143360 points. We take $\mathbb{C}$ as the identity, while we consider a ratio $r^* = 300$, which gives 480 material states.

First, we start by analyzing the histogram plots to see the error distribution of the mechanical states for the different algorithms, as shown in Figure 5.14. We can notice straight away that the PCA algorithm with removed dimensions is not very useful in this case, as observed in the plane stress case of sectionSection 5.3.1, however, PCA with all dimensions seems to improve on the results in the same way as the normalized algorithm, which correlates with the previous results.

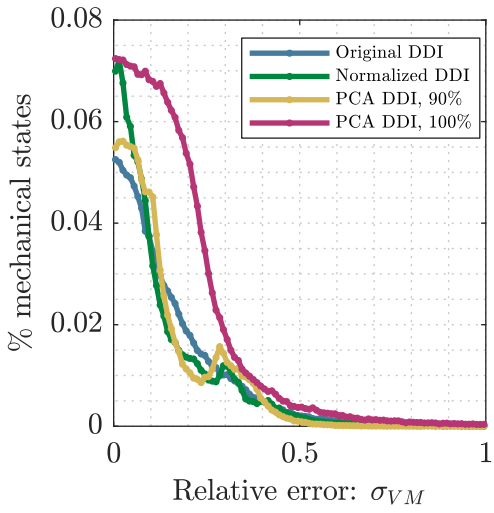
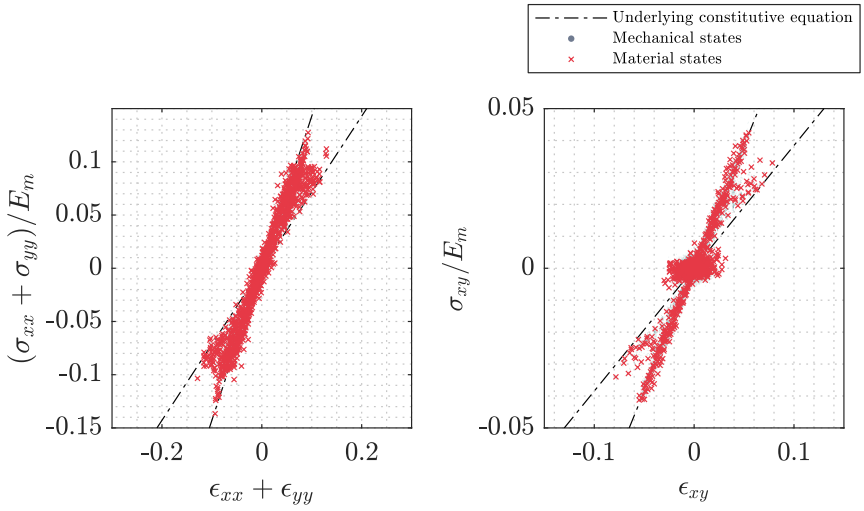
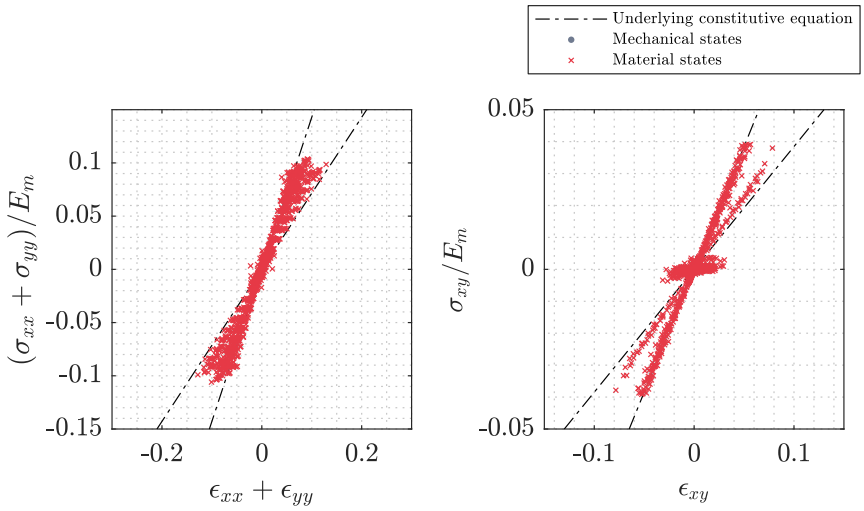


Figure 5.14: Error histogram comparing algorithms.

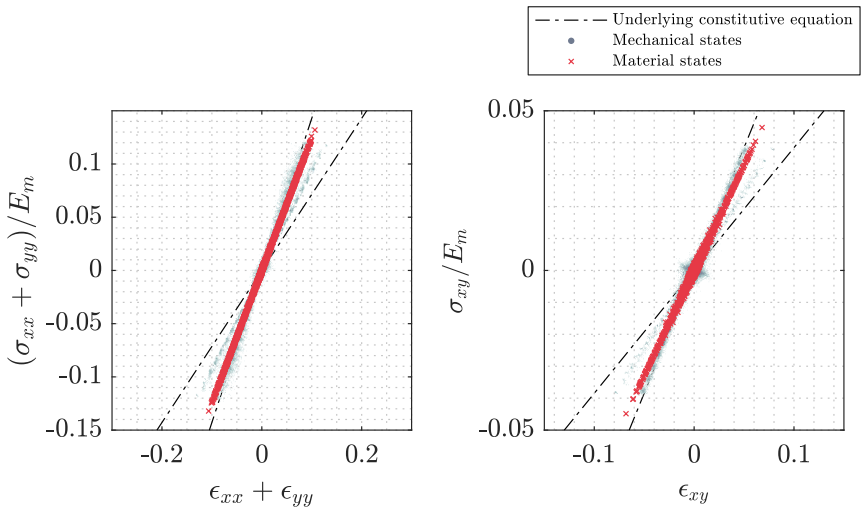
If we compare the phase spaces for both cases, in Figure 5.15a we can observe two different behaviors without a clear separation in the trace component. The mechanical points associated to the matrix elements gather around the expected underlying behavior, but as we expected it, the mechanical states associated to the inclusions are not estimated correctly. In Figure 5.15b we see that the normalized algorithm helps to clean a bit the results, particularly in the shear component of the phase space, where we now are able to see two well separated curves. Finally, Figure 5.15c shows that the PCA algorithm with trimmed dimensions does not recover a separated behavior. As in the case from Section 5.3.1, considering a PCA algorithm with all the dimensions is equivalent to performing the normalized DDI, so the results are not displayed due to their similarity.



(a) Phase space results for original DDI.



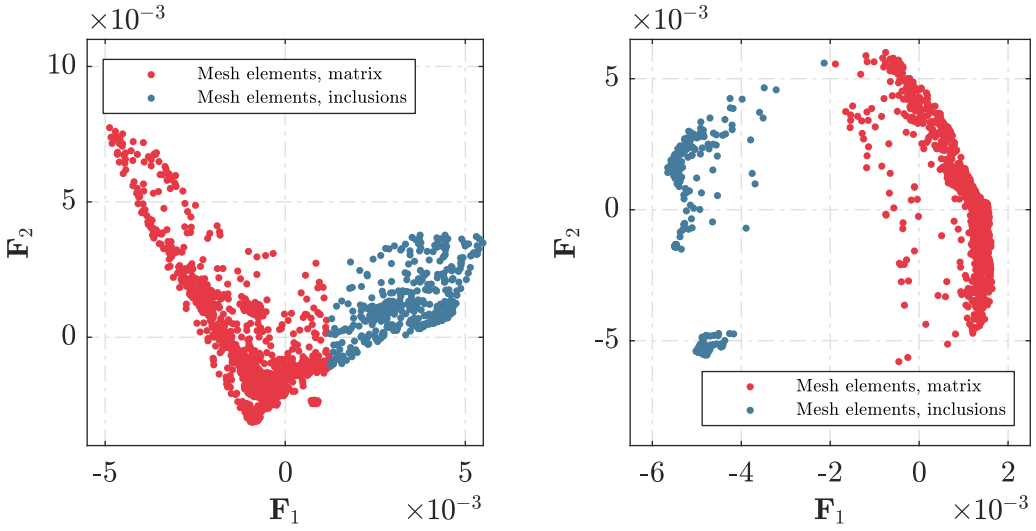
(b) Phase space results for normalized DDI.



(c) Phase space results for PCA DDI.

Figure 5.15: Phase space results for different DDI algorithms.

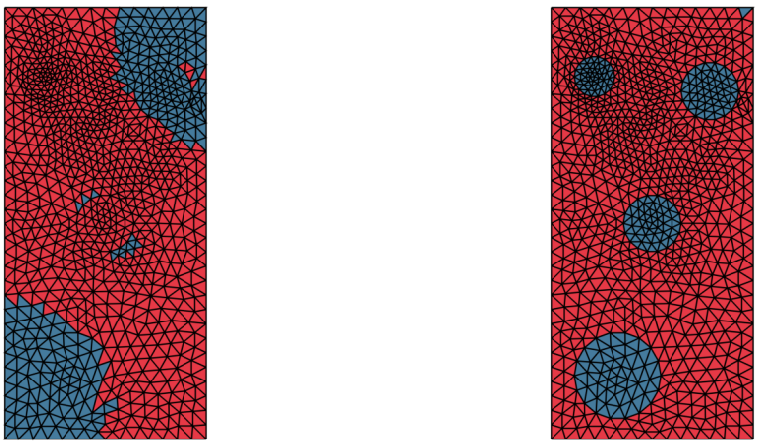
The improved results with the normalized algorithm are also visible in the correspondence analysis. Figure 5.16 has the comparison of the results for CA for both algorithms, used to perform the separation of the mesh elements. In here, the normalized algorithm, as well as the PCA with all its dimensions (which is identical to the normalized results so they are omitted), provides a clear cut of which elements belong to each phase of the material, while in the original DDI we have one single cluster of points with no discernible separation. This is a direct consequence from the improvement at the shear component of the normalized DDI. The results from the CA can be transported to the mesh, where we can see very well in Figure 5.17 that with the normalized algorithm we are capable of locating all four inclusions in the mesh, while the original DDI fails to see any of the inclusions.



(a) CA for original DDI.

(b) CA for normalized DDI.

Figure 5.16: CA comparison for original DDI and normalized DDI algorithms.



(a) Phase location with original DDI.

(b) Phase location with normalized DDI.

Figure 5.17: Separated meshes with different algorithms.

The algorithms performance comparison is detailed in Table 5.4. With respect to

time efficiency and convergence we see similar issues as in previous tests. What is interesting to see is that both of the modified algorithms (normalized and PCA) have a faster iteration time than the original DDI. With respect to the error, we see again that PCA tends to produce worse results in cases with plane stress due to the removal of important components, but we see that the normalized algorithm improves the overall solution.

Table 5.4: Performance comparison for the soft inclusions case.

	Original DDI	Normalized DDI
Initialization time (s)	0.7496	6.2300
Convergence time (s)	60.2180	67.9326
Number of iterations	20	500
Time per iteration (s)	3.0109	0.1359
Von Mises error	0.1672	0.1481
	PCA DDI (90%)	PCA DDI (100%)
Initialization time (s)	4.4945	5.1165
Convergence time (s)	26.1698	54.5085
Number of iterations	253	500
Time per iteration (s)	0.1034	0.1090
Von Mises error	0.1508	0.1494

5.4 Statistical techniques as post-processing tools

Dimensionality reduction techniques have been used extensively as post-processing tools. As aforementioned, considering a reduced amount of principal components to recompose the original data matrix removes redundant information, and thus reducing the noise. We have taken advantage of this property to improve the clustering performed in each iteration of DDI, but the same can be done after results are obtained.

The application of the PCA procedure for post-processing is basically the same as for the algorithm as shown in Section 5.2.4, but instead of applying the transformation for performing the clustering, we apply it on the converged result to try to reduce the error in the final estimation.

5.4.1 Examples

The results that we obtain from techniques such as PCA are highly dependent on what we choose as input data. In the cases that we show here, we want to improve the estimations for the stresses, so we consider as input data the same mechanical states ordered in columns for each snapshot, as done in the examples of Section 5.3. One of the issues with this approach is that, by updating the values of mechanical states, the material points are not necessary the centroids of the data anymore. An alternative approach where each material point corresponds to an extra column in the data matrix can be taken to also include the values of material states, so we will consider it and compare the performance. However, a problem that might arise from this definition of the data is that even if we are considering two different group of points with the same units, the fact that they have a different origin can disrupt the condition of mechanical equilibrium in the results, rendering the results to be physically unfeasible.

To test the post-processing technique, we take as a reference the results from Section 3.5.2. We choose these results particularly because they were intentionally computed in a poor setting, so the results are already noisy and less accurate. For applying the PCA filter, we consider three different ways to organize the data, so we expect to have different results in each one:

1. The first way is to organize the columns of the data matrix $\boldsymbol{X}$ as each one of the snapshots of the problem, which is the same approach that we have taken in previous examples and yields a matrix of size $3N^e \times N^X$. Since the filter is expected to move some of the points around, after applying the filter we proceed to do a new k-means clustering to rebalance the material states.

2. The second option is to organize the columns of the input data as mechanical states, which would not take into account how the stresses are organized in the sample, since there is no correlation between two columns. This yields a matrix $\boldsymbol{X}$ of size $3 \times N^e \cdot N^X$. After applying the filter we need to perform again a k-means clustering to rebalance the material states.

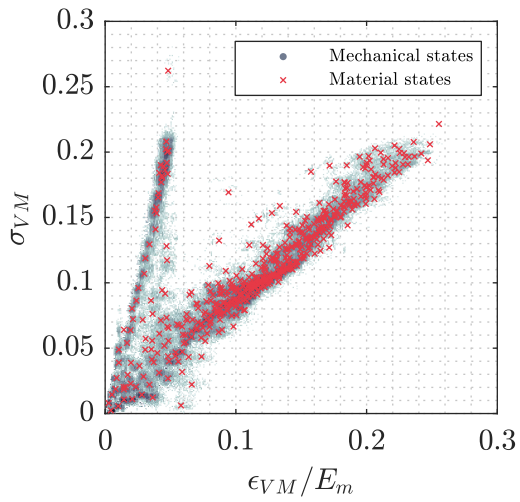
3. The last way is an extension of the second, but we add also the material states as columns in the matrix, which expands the size of $\boldsymbol{X}$ to $3 \times (N^e \cdot N^X + N^*)$. In this case, since the material states are filtered too it is not necessary to perform a clustering after.

The phase spaces for the Von Mises strains and stresses are shown in Figure 5.18. As can be seen, if we apply the filter using case 1 we have the most visible results. The fact that we have in each column a full set of points means that there is a correlation, since each row of the full matrix will represent a particular component of the stress tensor for a specific element. This ensures that the underlying linear behavior that must have been predicted by DDI is enforced during the filtering, resulting in the straight lines we see in Figure 5.18b. For the other two cases we cannot see this correlation, since we are only considering all the points independently, even if we decide to include the material states. It is because of this that the results do not seem to change as much.

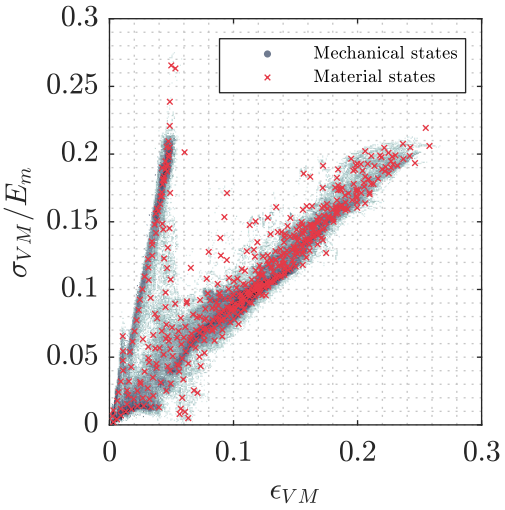
We include also in Figure 5.18 the results for kPCA, particularly for the case 1. In here, we consider a kPCA computed with and without the Nyström approximation, to compare the differences between them. As can be seen, the dispersion with the kPCA filters are lower than the ones obtained with the linear PCA. Between both versions of kPCA, except for the distribution of the points, there seems to be no differences in the quality of the results.

These results can be further analyzed if we check the results from CA. When we consider cases 2 and 3, we can see that CA obtains very similar results, only mirroring the dimensions. However, when we use the PCA as proposed in case 1 we see a proper separation of both phases. In here the separation of CA is performed automatically in the middle, but it can be improved by making some considerations. However, it is very clear that the filtered version provides an easier alternative for visualizing the CA space. This effect is better appreciated with the use of kPCA, where the points in the projected space tend to separate more into two clear groups associated with each phase, which is something similar to what we saw in Section 3.5.2 when a second run for the DDI was applied.

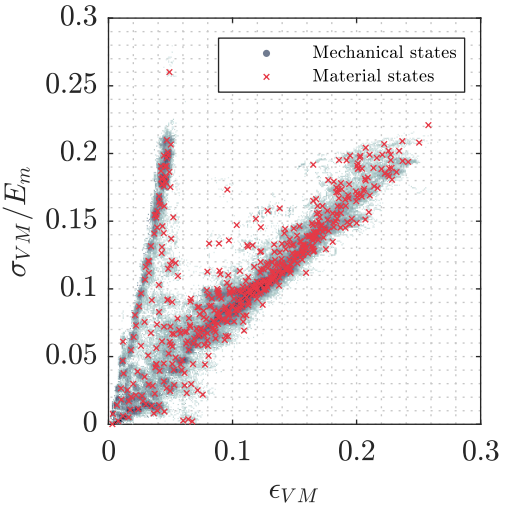
For the error analysis, Figure 5.20, we see that in general filtering the results provides a slight improvement of the estimations, which is more pronounced for the case



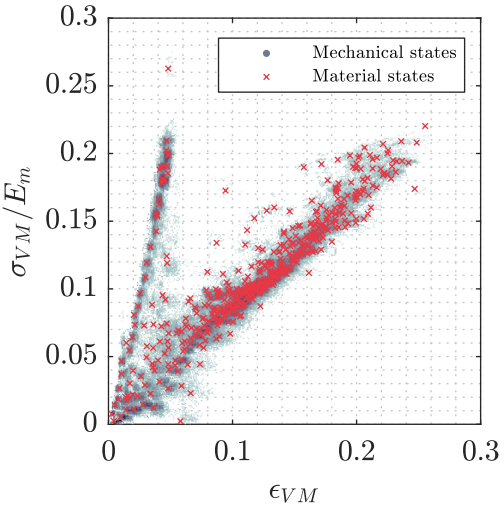
(a) Original results.



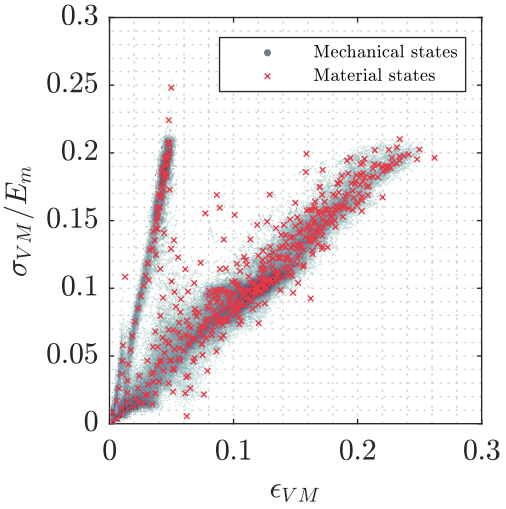
(b) Results filtered by snapshot.



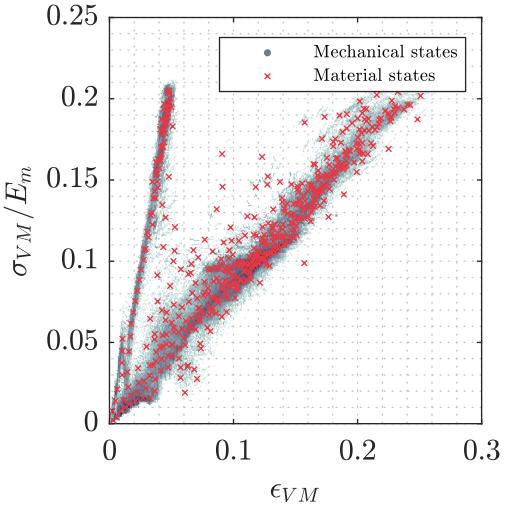
(c) Results filtered by mechanical points.



(d) Results filtered by mechanical and material points.

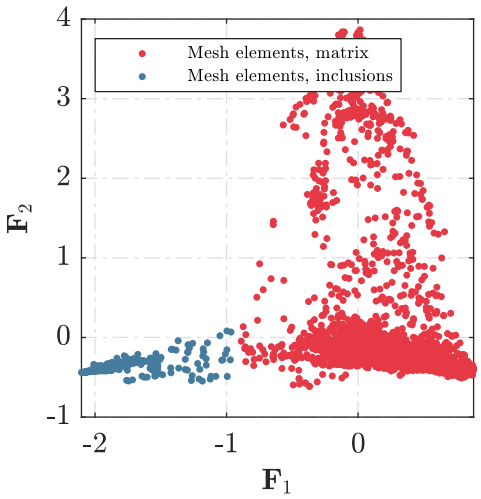


(e) Results filtered by snapshot, kPCA.

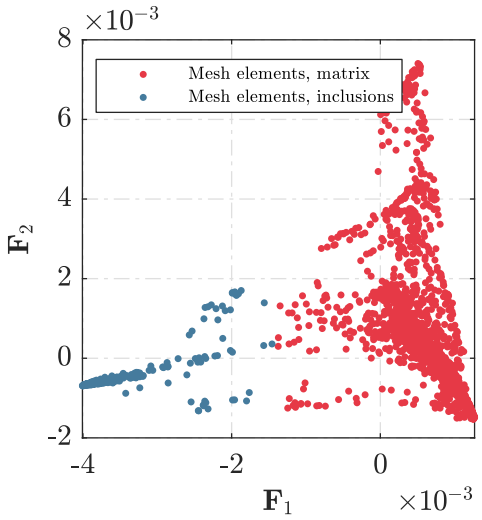


(f) Results filtered by snapshot, kPCA Nyström.

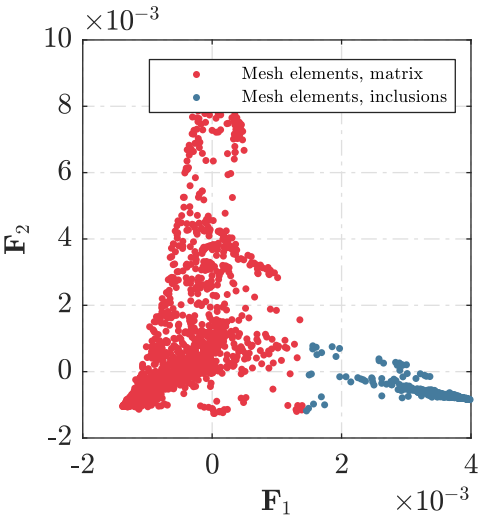
Figure 5.18: Phase space comparison for filtered DDI.



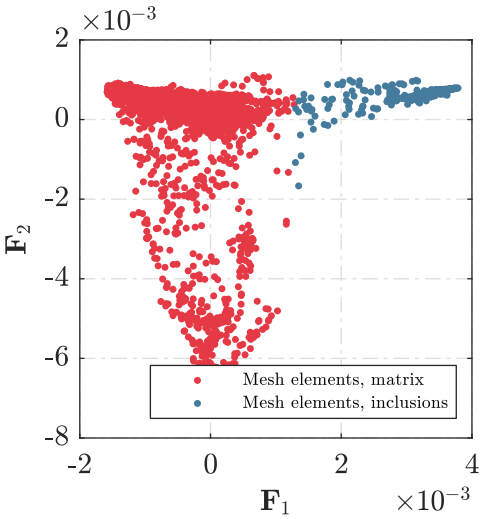
(a) Original results.



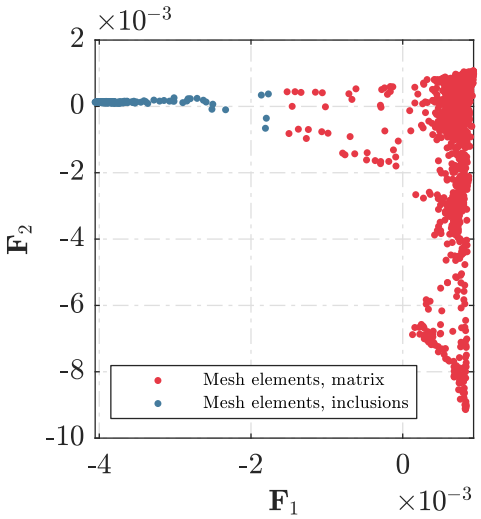
(b) Results filtered by snapshot.



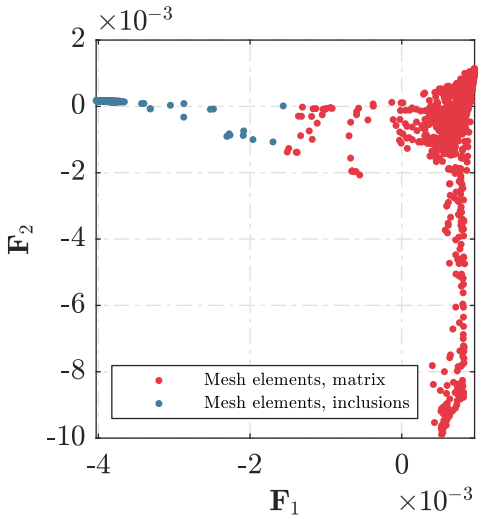
(c) Results filtered by mechanical points.



(d) Results filtered by mechanical and material points.



(e) Results filtered by snapshot, kPCA.



(f) Results filtered by snapshot, kPCA Nyström.

Figure 5.19: CA comparison for filtered DDI, mesh elements.

where we take the snapshots as a basis. Since for the other cases we chose the points as a basis, even if we add the material states they do not provide a difference, so the error curves for both cases are exactly the same. For the kPCA filter, we have that using the non-approximated method yields worse results than any other filter used. Several factors could be considered for this, including the way matrices are generated when performing the kPCA.

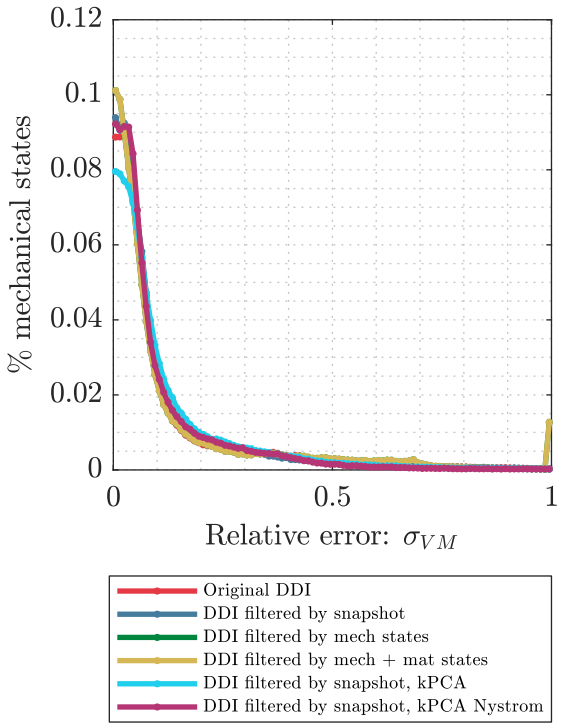


Figure 5.20: Error histogram comparing different cases.

In fact, if we compare the performance of the different versions of PCA for case 1 in Table 5.5, we can see that the implementation of kPCA without approximation is slow, with the Nyström approximation helping to reduce the computation time almost a third. Linear PCA is the most efficient one, since PCA relies only in one SVD computation that is directly performed on the data. with respect to accuracy, we can see that the non-approximated kPCA worsens the results, while the other two provide an improvement, correlating with the results seen in Figure 5.20. For computing the error, the relationship shown in Equation 2.39 for e_2 is used, comparing the values for the estimated Von Mises stresses against the original FEM solution.

Table 5.5: Comparison of performance of the different PCA filters.

	Unfiltered	Case 1, PCA	Case 1, kPCA	Case 1, kPCA Nyström
Time (s)	-	0.2180	185.4665	64.7519
Error (%)	0.1414	0.1344	0.1583	0.1102

As we mentioned in the beginning we have to keep an eye on the fact that the filtering might disrupt the equilibrium condition that we impose in the DDI algorithm. To verify this, for each one of the cases we compute the divergence of the stresses for each

snapshot to check if they remain at zero. This is shown in Figure 5.21, where we can see that basically there is no disruption of the equilibrium condition. The oscillations seen, specially in case 1, are due to the fact that we are dealing with numbers too small for the computer precision. In any case, we can also consider that, since we are dealing with full snapshots rather than just points, it makes sense that this type of filtering could introduce the biggest imbalance in the problem.

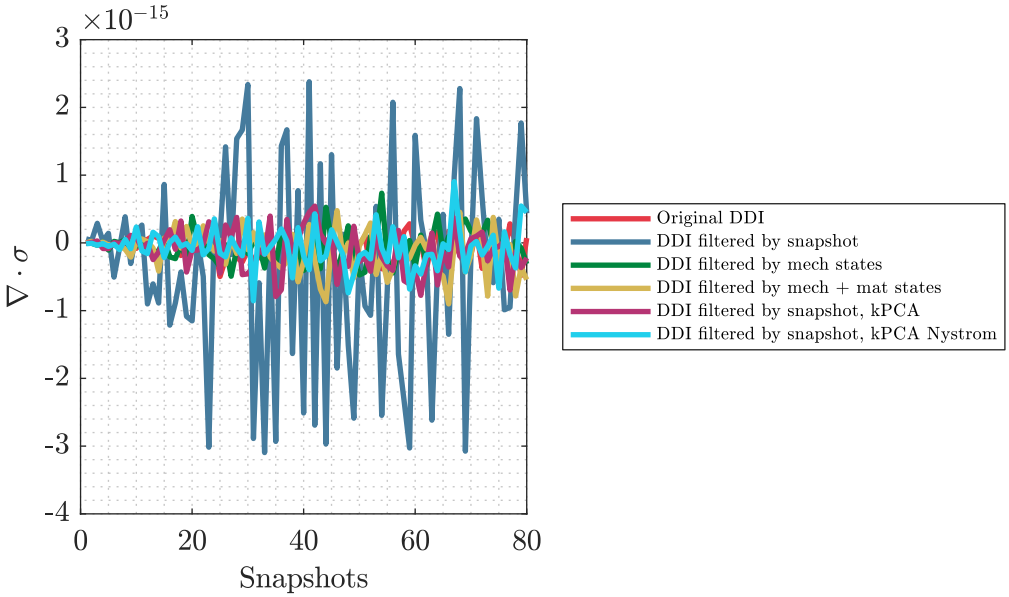


Figure 5.21: Divergence of stresses for each snapshot.

Finally, Figure 5.22 shows how the error progresses depending on the amount of principal components considered in the filter. As expected for PCA, since there is one dominant principal component, all the others can be interpreted as noise, so the error should progressively increase until the value that it has in the original algorithm. When the principal components tend to be not relevant at the end we have a plateau and the error stops increasing, but we can see that considering just one mode is enough to have an improvement. Interestingly enough, for both cases with kPCA we see that adding more components to the computation tend to decrease the error, although the plateau is reached fast. We believe that this effect is related to the fact that for kPCA we do not compute directly the SVD on the data as it is performed in the linear case, but in a kernel matrix that approximates the subjacent non-linear manifold, so adding more components increases the accuracy by adding more information to this approximation.

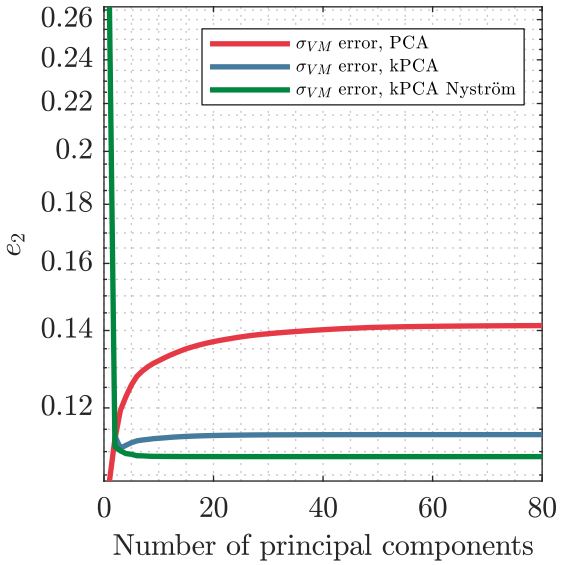


Figure 5.22: Error as a function of principal components.

5.5 Discussion

In this chapter we explored the use of dimensionality reduction techniques to try to reduce the computational load of the algorithm and have an improvement in speed, as well as accuracy. A normalized version of DDI was also introduced as an alternative to avoid having to define the parameter $\mathbb{C}$.

Testing the new algorithms in similar settings to the ones seen in previous chapters we have seen that the modified formulations tend to provide a better estimation of the mechanical stresses in most of the cases. However, this advantage is heavily restricted by the fact that these algorithms tend to run much slower than the original version of DDI. Performing a deeper analysis of the time consumption of all algorithm we noticed that the issue lies mainly in the convergence: the newer algorithms tend to drift much more before reaching a desired convergence tolerance. We saw that in some cases, even when performing several repetitions of the same problem, the new algorithms always converged to a local minimum rather than the global optimum. In this regard, some improvements could be implemented to fix both accuracy and speed. For these algorithms we have relied on the use of k-means for clustering, which can be time costly for big sets, due to the need of computing distances to each point. Perhaps a kNN search scheme could be considered, similarly to the way it is being handled in the original DDI, to obtain a speed increase. For the case of accuracy, the implementation of the kPCA algorithm could be beneficial, since the availability of different kernels means that we have access to different sets of results for the same initial problem, where some could be more accurate than others.

In previous chapters we addressed the problem with softer inclusions and DDI. We compared the performance of this problem with the newer algorithms and we saw that particularly the normalized formulation provided a slight increase on the accuracy of estimations and a speedup in the problem, although we retain the problem with convergence. The big advantage comes in the correspondence analysis, where the normalized formulation was capable of identifying both phases of the material properly, as opposed to the original DDI. It is possible that this problem can never be addressed fully by

data-driven schemes, given the nature of imaging techniques, but the results obtained here propose interesting options for future study.

Furthermore, we use PCA dimensionality reduction technique as post-processing filters for the obtained results. We see that by choosing the input data appropriately we can increase the accuracy of the estimations without losing the physical admissibility of the stresses. We have seen that applying the filters allowed us to have a cleaner separation of phases with the application of correspondence analysis, even when the original results are not optimal.

The combination of DDI with dimensionality reduction is a useful tool that was born with our DDI implementation for viscoelasticity in mind, but it can be applied to different cases as shown in this chapter. In here we dedicated ourselves to academic examples to show the capabilities, but we believe that it could have potential to be applied in other settings, specially considering more complex material behaviors.

Chapter 6

Application of DDI to hybrid polymer composite samples

The previous chapters have been devoted to the development of a new data-driven approach applied to synthetic heterogeneous materials. The main focus of this work has been to analyze the behavior and extend DDI algorithm and its variants. However, the end goal is to be able to apply this methodology as a viable alternative for material identification or stress estimation. The objective of this chapter is to apply the proposed method, as it has been presented, to real samples of polymer composite materials, to establish the constitutive properties as well as the different phases present in them. The scale at which heterogeneities appear in fiber-reinforced composites is generally too small to be detected by traditional DIC methods. Micro-fluctuations smooth out at the scale of the measurement. In fact, another family of composites used in the automotive industry is of interest here. These are hybrid composite structures combining semi-structural composites with non-oriented fibers and structural composites with continuous and oriented fibers. The mechanical property contrast is measurable at a scale compatible with the resolution of DIC methods.

6.1 Experimental setting

6.1.1 Samples

In this chapter, ten samples are analyzed. They are all cut out from a spare wheel tray designed and produced by Faurecia.

Material

The part is designed with a *Glass Mat Thermoplastic* material [65], reinforced in some areas with a woven fiber composite [66]. Both materials are made of the same polypropylene.

- Glass mat reinforced thermoplastics (GMT) are materials available as prepreg sheets which are heated and molded in compression. The glass fiber reinforcement is a needled mat of randomly oriented fibers. The fiber content in these samples is 30% by weight, which corresponds to 13% by volume. The fibers are impregnated with a polypropylene (PP) matrix.

- The second material is a woven prepreg reinforced with a twill 2/2 glass fabric. It is manufactured at Faurecia by the film stacking process with PP sheets to get a fiber content of 60% by weight, equivalent to 34.5% by volume. The final prepreg is made of three plies of woven fabric oriented in the same direction.

In our particular case, we do not have access to the properties of the materials that we are testing. This is done intentionally to be able to test the DDI capabilities, which is the reason why only some estimated values are given for both materials.

The part to be studied is shown in Figure 6.1. It can be noticed from the coloration of the part that there is an homogeneous outer region which corresponds to the GMT composite. In the center there is a rugged texture to the light, which is the part where the woven prepreg is located. The rugged surface visible in the bottom of the picture in Figure 6.2 indicates that the part has not been properly consolidated, which was confirmed by Faurecia. The interface between both composites can be appreciated better in Figure 6.2. In here, the characteristic woven pattern is much more noticeable.

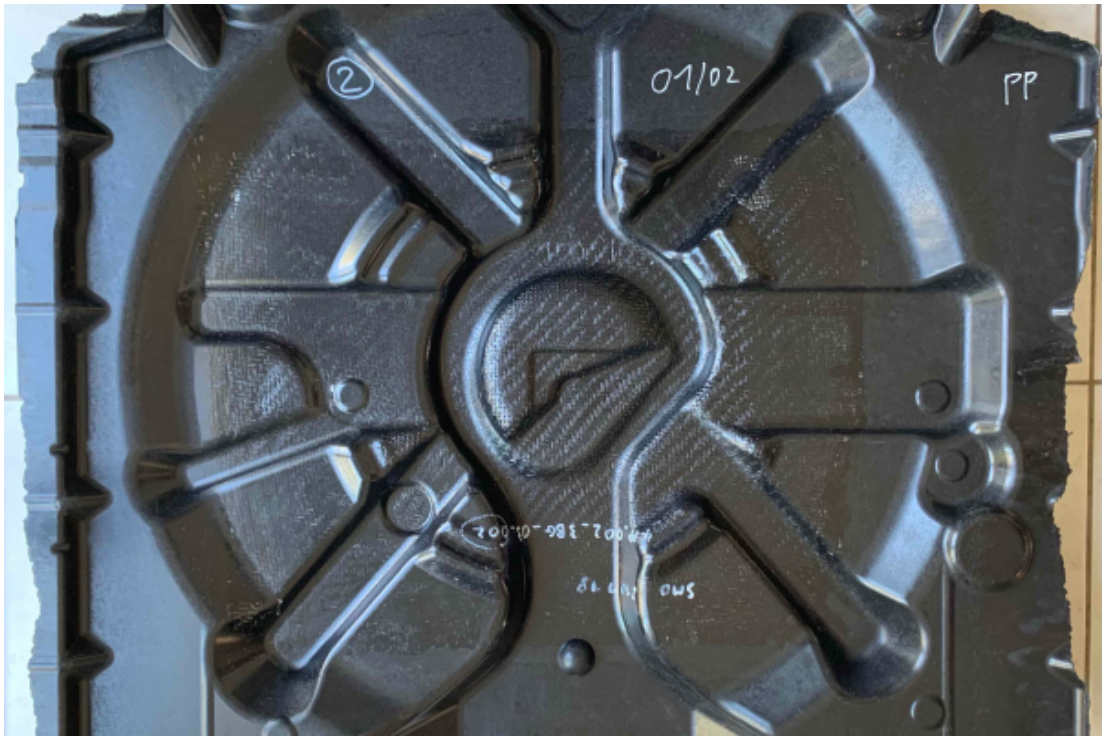


Figure 6.1: Spare wheel tray from where the samples for testing were obtained.

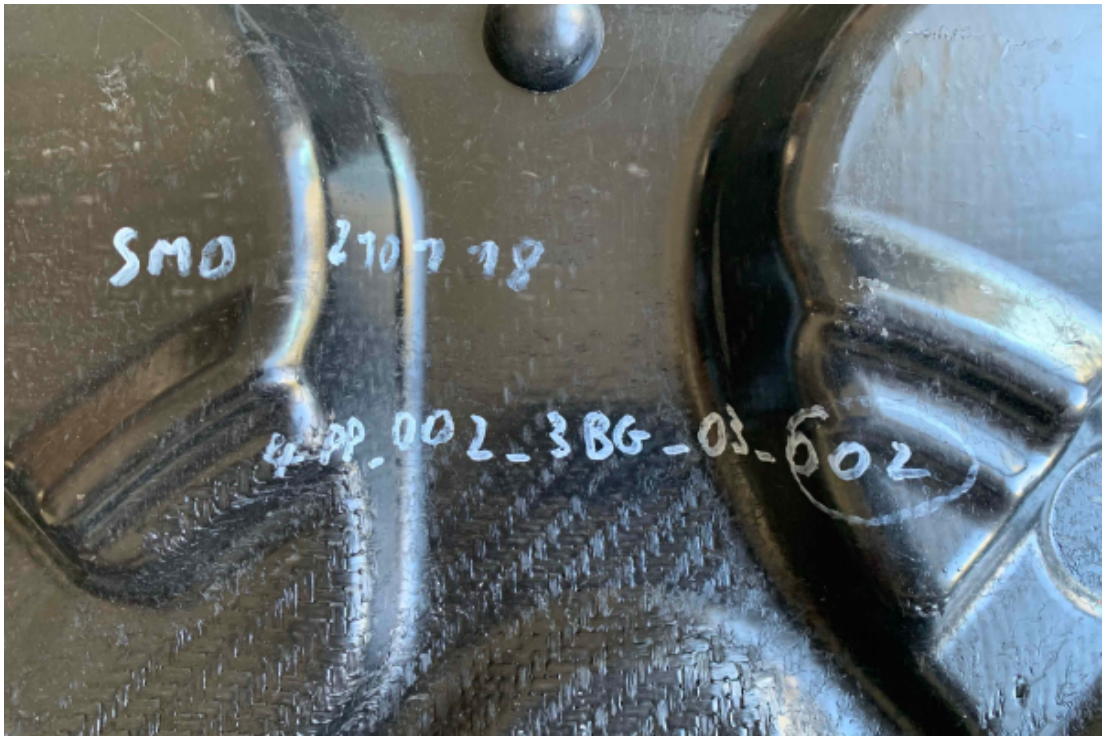


Figure 6.2: Zoom on the interface between both composites. The top smooth surface corresponds to the GMT, while the woven prepreg is in the bottom. The rugged surface of the woven prepreg zone is clearly visible.

Geometry of the samples

In order to test the properties of the materials of this part, small samples are cut out. They were cut from the flat areas of the part into rectangular coupons. Figure 6.3 shows the areas from where the different samples where obtained. Sample F is not visible in the picture, which was obtained from the lower part.

Figure 6.3: Placement of the different samples cut out from the part.

Given the location of the different samples, most of them include a mixture of both composites. Sample E is the only one fully reinforced with the woven fabric, so it is used as a reference for that material. Regarding the GMT, both samples F and G are the only ones entirely composed of this material.

Another point to be considered is that samples A, F and G were obtained from the bigger flat areas of the part, so these samples are wider than the other four. For this reason, these samples are split in two to keep the form factor equal for all the ten samples. All the cut samples are shown in Figure 6.4.

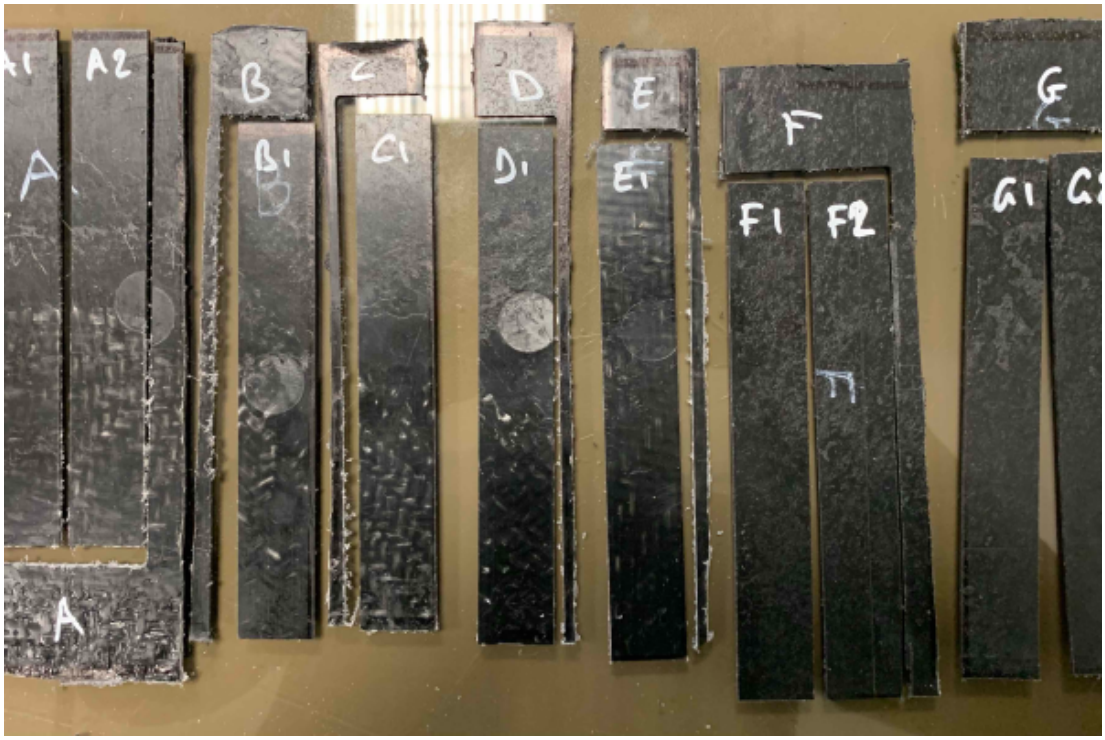


Figure 6.4: Cut samples to be tested. In this image the material composition of each sample can be appreciated in the material patterns.

To prepare the samples for the use of DIC, a speckle pattern is painted on one of their sides, in an area of interest, as shown in Figure 6.5. It can be noticed that the marks of ejectors (circular print) are still visible through the pattern, which could interfere with the proper correlation of images.

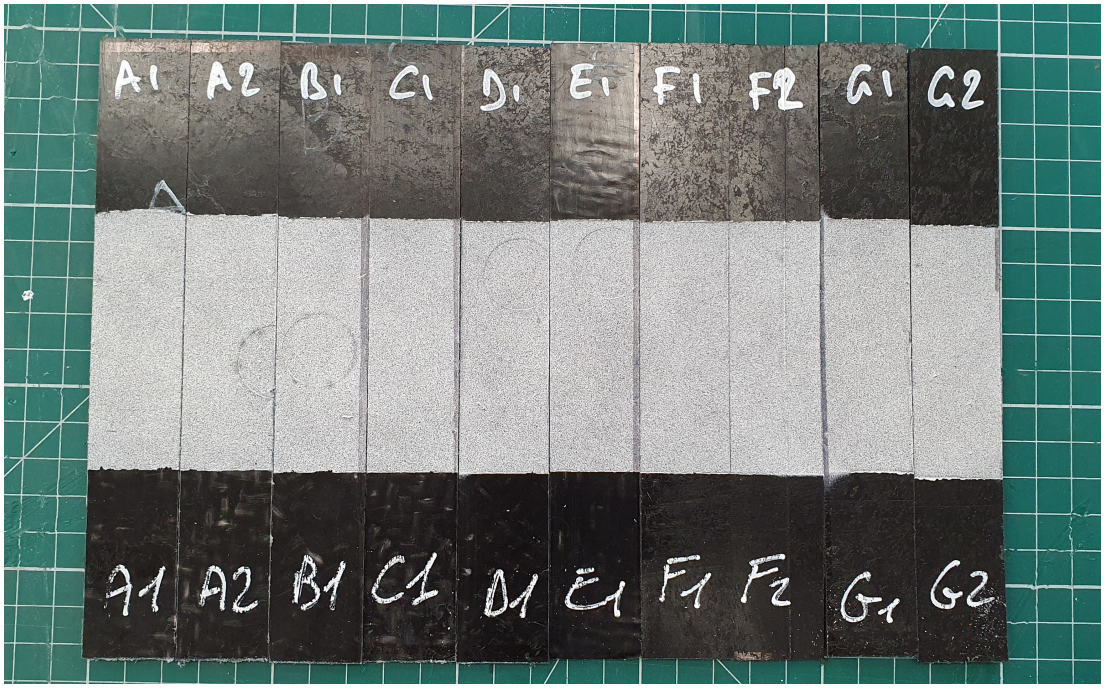


Figure 6.5: Samples painted with the speckle pattern.

6.1.2 Testing

The testing was performed in the mechanical testing laboratory at ECN on the prepared samples. The test performed was a quasi-static tensile test using an electro-mechanical universal testing machine Instron 5584 (shown in Figure 6.6), that has a maximum capacity of 100 kN in tension. The machine is controlled using the software Bluhill developed by the same company.

Figure 6.6: Instron 5584 used for testing.

For the DIC procedure, a 29 megapixel camera (seen in Figure 6.7) is used to track the movement of the sample during the tensile test. The images that are collected at certain intervals are matched with the current value of the voltage being applied in the machine, which then is correlated with a certain value of a force applied at the upper border of the sample. The obtained images are later processed using the dedicated software VIC2D (Kilonewton), specialized in DIC for flat samples.

(a) (b)

Figure 6.7: Camera used in DIC.

The samples were mounted in the tensile grips and stretched until they reach a failure state. Information about the deformation and the force applied are registered at different timesteps, which are collected in tables. These values are then used to create the force-deformation curves that were applied in the sample, which serve as a basis for the DDI analysis that will be performed later.

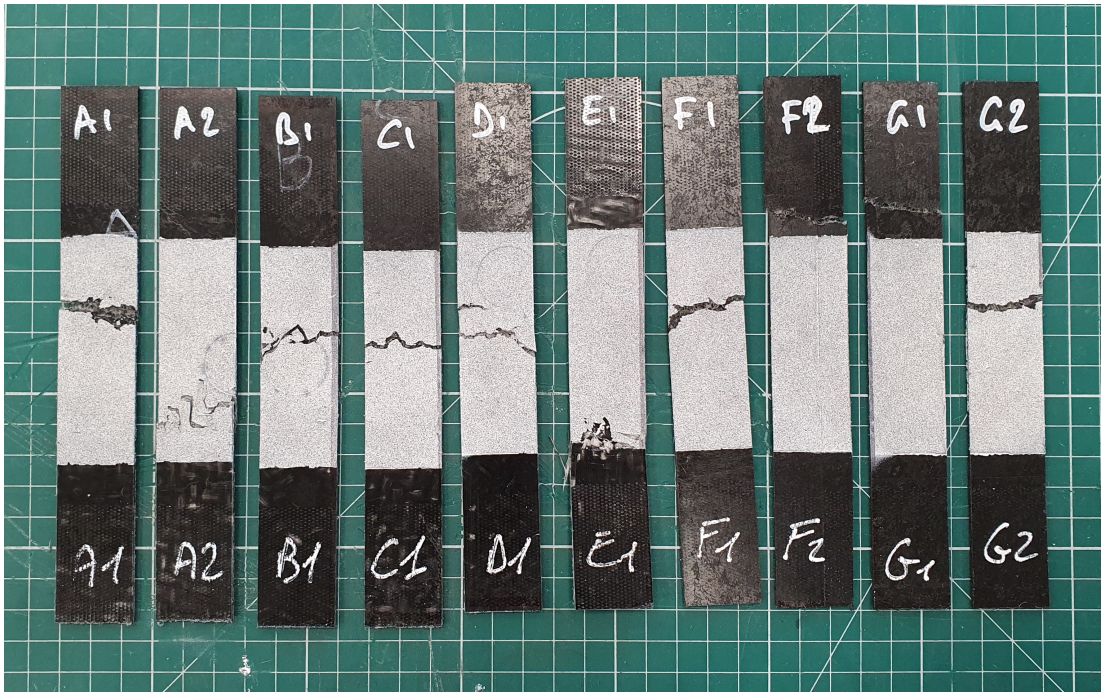


Figure 6.8: Samples broken after reaching their failure point.

6.2 Data processing

Once the test are performed, a big amount of raw data is produced. This data, consisting mostly on the measurements obtained by the machine and the parameters used, as well as the images obtained by the camera, need to be processed and adapted into information that can be used by our algorithms.

6.2.1 Preprocessing

The necessary information for the application of DDI comes from two sources. First, we have the raw images that are taken at certain intervals, that track the deformation of the sample over a certain period of time. The second source is the machine itself, which is stretching the sample and it is tracking both the displacement of the crosshead as well as the voltage that is being applied. This voltage can be translated to the net force that is applied to the sample.

First, we dedicate ourselves to the processing of the images. For this, we use the software VIC2D, which uses correlation algorithms to provide non-contact, full-field, two-dimensional displacement and strain data for mechanical testing on planar specimens. In-plane displacements are measured at every pixel subset within the area of interest, and full-field strain is computed with many tensor options. The software works by loading the set of images that were captured by the camera during the testing process. Initially, we consider the first one as a reference undeformed state. In the software we choose an area of interest where the DIC will be performed. This needs to be chosen carefully, since results will be affected by this selection. Normally, we would limit ourselves to the painted area of the sample, which should not extend to close to the clamping points since stress distribution can be more unpredictable closer to the area where the forces are being applied.

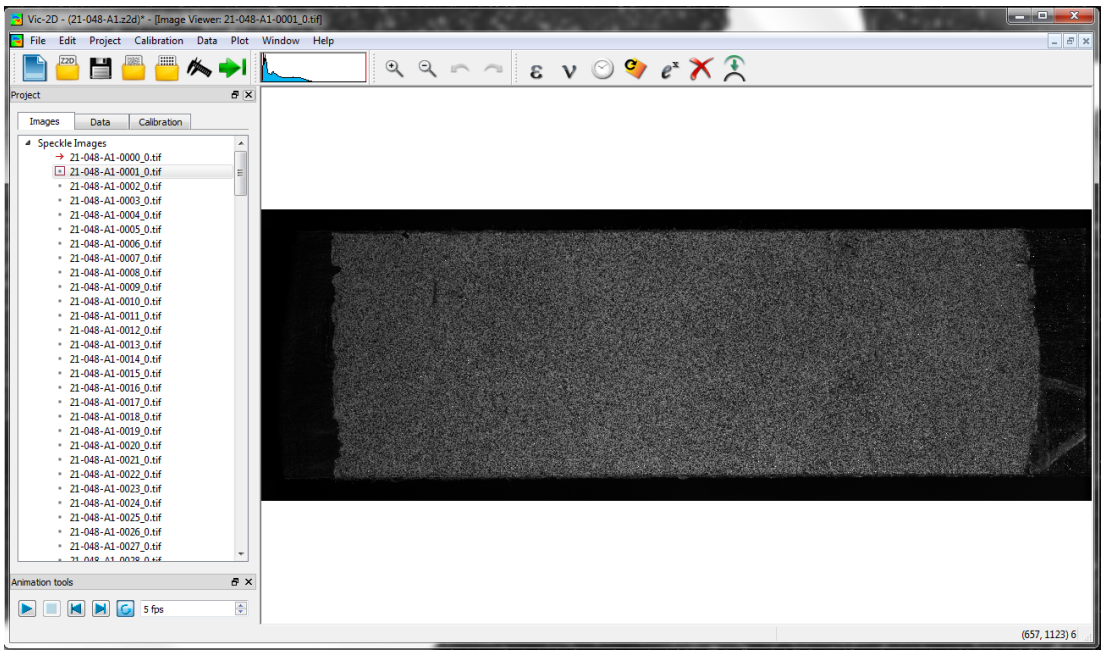


Figure 6.9: Image of sample being processed in VIC2D.

With the interest area sorted, we need to fix some parameters for the software to

work. Mainly, we need to define the size of the subsets for the image correlation to track deformation. We want to choose a value that is small enough to produce some accurate results, but large enough so that the computation of the deformation fields will not take too long. In this work, we settled for a subset size of 25×25 pixels. Appendix A explains in detail how the DIC algorithm works and what is the use of the subsets. We also need to choose which type of results we want to obtain from the analysis. For DDI, we are only interested in the values of the strain tensor, namely ϵ_{xx}, ϵ_{yy} and ϵ_{xy}. Other results that are provided but not used were the principal strains ϵ_1, ϵ_2 and γ as well as the parameter σ that tracks the certainty of the image correlation.

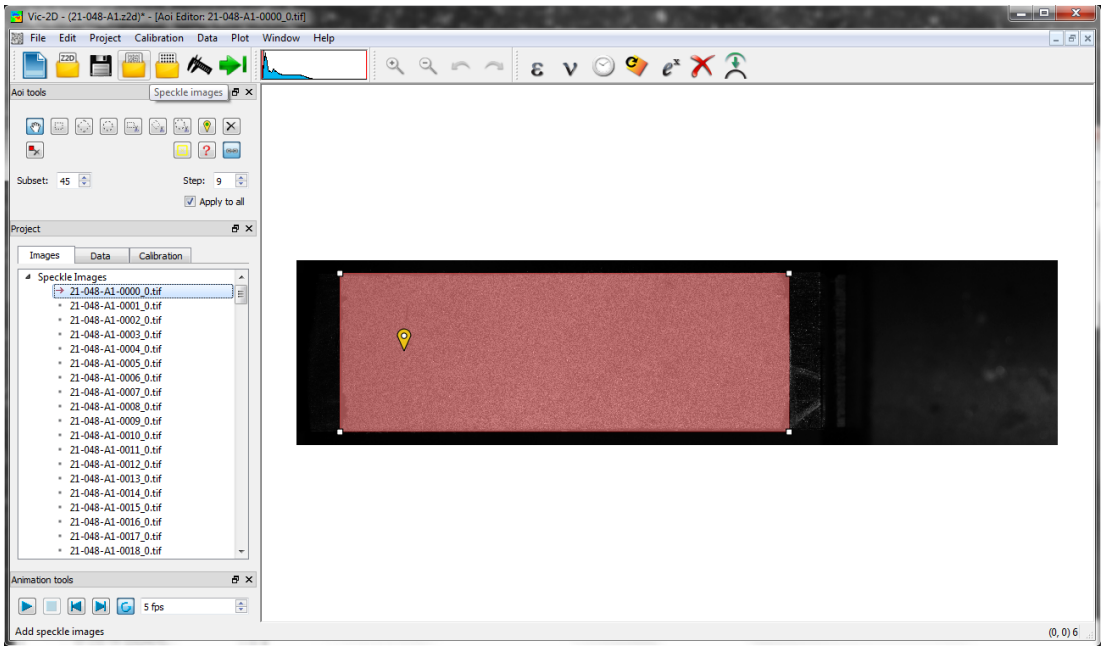


Figure 6.10: Image of parameters and area selection.

With all set, the software is run, comparing each step to the reference image to obtain a deformation and strain field for each image. These results can be plotted over the original image as shown in Figure 6.11, which allows us to have a *live* visualization of the different fields.

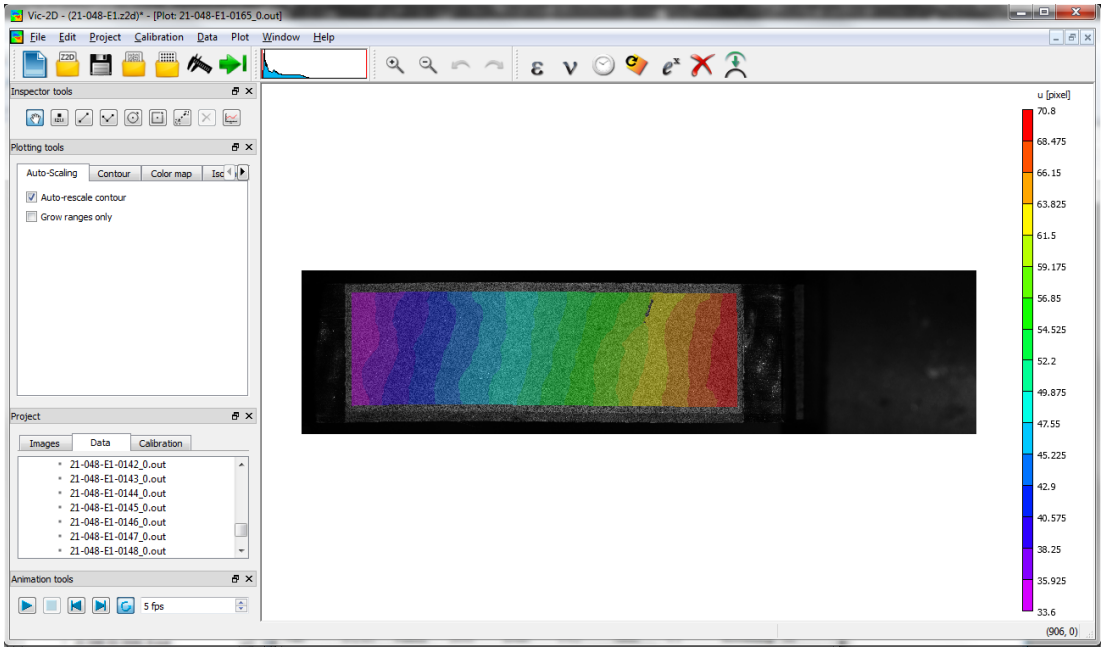


Figure 6.11: Displacement field.

Since the software is not able to tell when a sample is fully broken, it still provides results for all the snapshots taken after ultimate failure. Those cases are sorted manually and removed from the input data since they have no use for the estimation of stresses. One important thing to mention is that in all this process, the deformation fields are measured in pixels rather than metric units. This is no problem for the strain fields, but to work with appropriate units from now on we need to define the scale. Since the software tracks the boundary of interest, it is important that it is well defined and measured, so in every sample we select it as close as possible to the painted area, whose dimensions are known. From this, we can estimate the deformation in metric units to be used in following analysis, but we have to be clear that there is an inherent source of error from this procedure.

With the images already processed, we have access to the deformation and strain fields in the sample. The last data needed are the forces applied to the samples, which is the information we obtain from the testing machine. Since we cannot have information about internal nodal forces, or even the distribution of forces on the upper and lower ends of the sample, we need to make some assumptions. The first one is that, since we have a sample without any holes, crack or imperfections, we can expect that all the internal forces are balanced, and that the only disturbances are being applied in the boundaries where the grips are placed, specifically the upper and lower ends that are being pulled. The second assumption made is that we have a perfect boundary where to apply the force. It is most likely that there are some imperfections on the boundary that might make the force distribution not uniform, so we force the boundary to be a clean border and we apply a net force on the full side as a block, which is the same approach as the one we have used in the synthetic examples in previous chapters.

Finally, the system pairs each snapshot with the measurements that the machine performed at the moment the image was collected. The information provided is measured in volts, so a conversion needs to be made to obtain the values in millimeters and Newtons that we will use. For all samples, we have that the output channel in charge

of the force has a ratio of 10 kN/V, except in sample A2 where the ratio is taken as 5 kN/V. On a similar way, the output channel for the deformation has an equivalence of 5 mm/V, while for the sample A2 this is reduced to 2 mm/V.

6.2.2 Application of DDI to samples

From the software and the machine output we have all the required data to run DDI, now we need to adapt the data for the algorithm. The first step is to generate a mesh based on the fields obtained. DIC gives a point-wise deformation organized in a grid. These points are taken as the nodes for the mesh that will be generated. The mesh is created with square elements with one integration point. For obtaining the elemental strain rather than the nodal ones an average of the values is considered, taking into account how many elements each node is neighboring. A special consideration needs to be taken with the points that are not being captured by the DIC since these points have missing information, rather than being a hole in the mesh. This difference is important, since a hole will have a net force of zero on its nodes, due to being a boundary. When the image cannot capture a point there are some strains and stresses in those points, meaning that we risk having a zone with a residual force that affects the balance in the sample. The approach taken to overcome this problem is taken as it was done in [26], where a zero net force is imposed in the boundary nodes around the hole to make sure that we keep the condition as it is. A diagram for this can be seen in Figure 6.12.

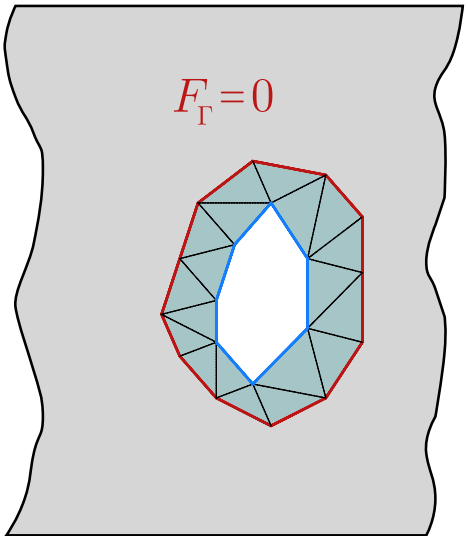


Figure 6.12: Schematics for imposed zero force on boundaries where information is missing. The blue border encircles the nodes that could not be captured by DIC, while the green elements are the neighboring elements that will be deleted from the computation. The net zero force boundary condition is imposed on the red border.

The mesh is considered to be perfectly clamped in both left and right boundaries, with a force applied to the group of nodes of the right boundary, as shown in Figure 6.13 for sample D1, including the holes for missing information.

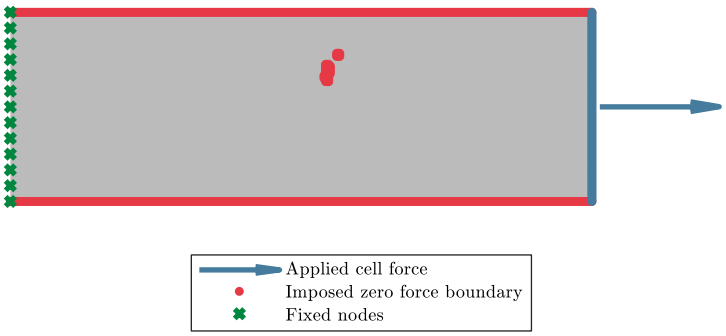


Figure 6.13: Example of the mesh for sample D1, with boundary forces and conditions. The cell force is applied in block on the right side of the sample instead of the individual nodes, since the distribution is unknown.

For the DDI parameters, we have decided to take a different approach to that of the previous chapters. A value of $r^* = 1000$ is chosen for the amount N^* of material states, given the size of the problem. For the value of $\mathbb{C}$ we choose infinity, due to the benefit in estimations, but also given the reduction in computation time. The amount of nodes and elements vary from sample to sample, but in general we have a number of nodes of about $N^n \approx 40000$, with an approximate amount of $N^e \approx 40000$ elements. The amount of snapshots N^X is dependent on how fast fracture was achieved during testing, but it ranges between 90 and 170 steps. The exact quantities are reported in Table 6.1.

Table 6.1: DDI parameters for each sample.

Sample	N^n	N^e	N^X	N^*
A1	43920	43435	170	7384
A2	41511	40826	170	6940
B1	41664	41181	150	6177
C1	41181	40700	90	3663
D1	44754	44247	99	4380
E1	41758	41252	145	5980
F1	45262	44770	145	6492
F2	45262	44770	145	6492
G1	40504	40018	145	5803
G2	41724	41245	145	5981

With respect to the time-dependent DDI algorithm, we have set that the solution should be computed considering $h = 1$ and $h = 5$ previous steps. We are not certain on the amount of steps that would provide the most efficient solution, but with this amount of previous steps we should see a trend of divergence between the different DDI algorithms.

6.3 Results

The target of analyzing these samples is twofold. Our main goal is to have an estimation on the properties of the material, which means that we are interested in the mechanical stresses, as well as the material points. Since we do not have any reference solution,

the analysis is done in a more qualitative way, meaning that we will look more into how the results look and how they relate to classical estimations of the materials. In that sense, our second goal is also to test our different approaches of DDI and see how they relate to an unknown material. In this case, we will run our original formulation of DDI, as well as the updated formulation for strain-rate dependent materials. Since we cannot estimate the error of the formulation, we aim to compare how much they diverge from each other and with that information in hand try to understand which one would probably suit better.

The traditional approach for studying these types of materials is to perform tensile tests and obtain a strain-stress curve that can be fitted with a polynomial, where a tangent Young's modulus can be obtained, typically close to the origin. Since we expect the results not to fall exactly in one curve like an elastic case, some provisions are made. Once mechanical states are computed, we can trace time curves for strain and stresses, which would be associated to each element in the mesh. Since we would be dealing with more than 40000 curves, we choose to take an average of the curves to represent the full material. In this way, we can perform a polynomial fit in the obtained average strain-stress curve, which will allow us to have an estimated value of the Young's modulus. This approach might not be the most accurate, but it will allow us to see if the values are close to estimated ranges for the studied composites.

6.3.1 All samples, engineering Young's moduli

Before using DDI, a simple approach that can be performed to obtain an idea of how the material looks is to use the classical engineering method for obtaining a value of the Young's modulus. For idealized uniaxial tension tests performed for example in bars, we can obtain the displacement of one of the ends of the bar, as well as the force that it is applied at each step. If we track as well the section of the bar at each step, we can obtain an estimation of both the stress and strain, which are then used to obtain the value of the Young's moduli. For the sake of simplicity, one can assume that the material behavior is completely linear and only obtain the values of strain and stress for the last step before failure. For the samples used, the force-deformation plot is shown in Figure 6.14. In here we can see that the curves are not linear, but they can be approximated to a good degree. The only exception we have is sample A1, which seems to harden at high deformations. This might not be the intended behavior of the material, so we take the results with care.

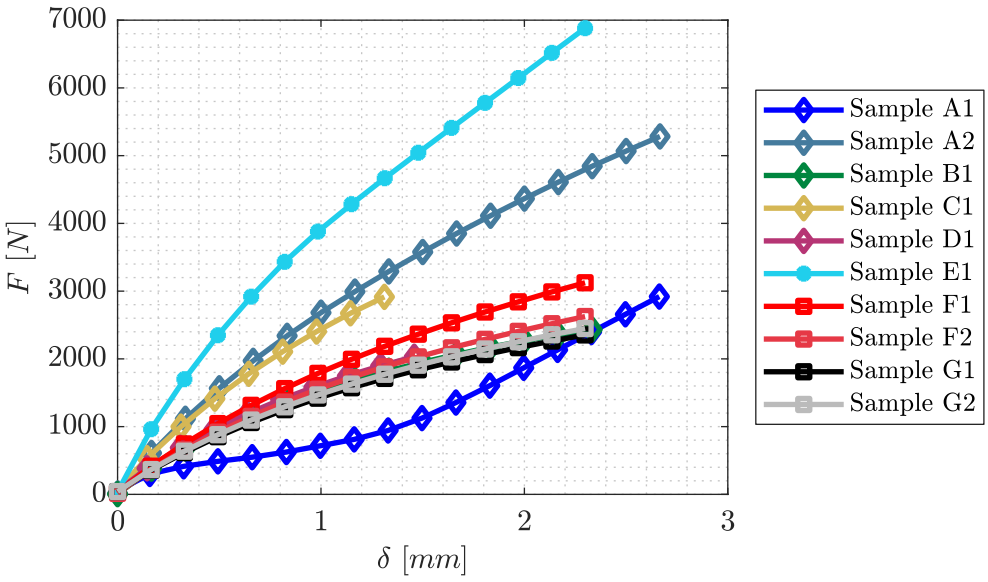


Figure 6.14: Force-deformation curves for all samples. Diamonds represent mixed composites samples, stars represent woven composite and squares are used for GMT composites.

This analysis can also be applied to our samples. Since we are working with flat samples, the section is replaced for the width (W) of the sample before the failure point. In a similar fashion, we also measure the stretched value of the length (L) of the sample, to estimate the deformation. Table 6.2 outlines the physical characteristics of all the samples analyzed, as well as the measured deformation (δ) and forces (F) before failure, the estimated value for the Young's moduli (E) and the estimated Poisson's ratio (ν) of each of them.

Table 6.2: Composite samples: characteristics and results.

Sample	L (mm)	W (mm)	δ (mm)	F (kN)	E (GPa)	ν
A1	2640.8	24.92	2.8133	3.1550	3.750	0.0759
A2	2606.4	24.77	2.8167	5.4665	7.368	0.2154
B1	2675.0	24.63	2.4832	2.4885	2.923	0.3617
C1	2631.2	24.77	1.4615	3.0855	6.753	0.3709
D1	2663.0	24.78	1.6051	2.1277	4.104	0.2751
E1	2665.0	24.93	2.3633	7.0282	12.013	0.0407
F1	2662.4	24.76	2.3635	3.1764	4.420	0.1912
F2	2651.7	24.92	2.3633	2.6662	4.342	0.1830
G1	2607.0	24.75	2.3634	2.3874	4.038	0.1776
G2	2624.5	24.83	2.3634	2.4810	3.590	0.1508

Samples A to D correspond to the uncontrolled mix of both types of composites, which can explain the dispersion in the values of their Young's moduli and Poisson's ratio. However, samples F and G are both composed of the GMT composite, who has more homogeneous properties. In these cases, the values for the Young's moduli appear to be close to each other and in line with theoretical values proposed for the sample. In the case of sample E, the value of the Young's modulus seems to be adequate to what it is expected from the woven composite, although the value for the Poisson ratio

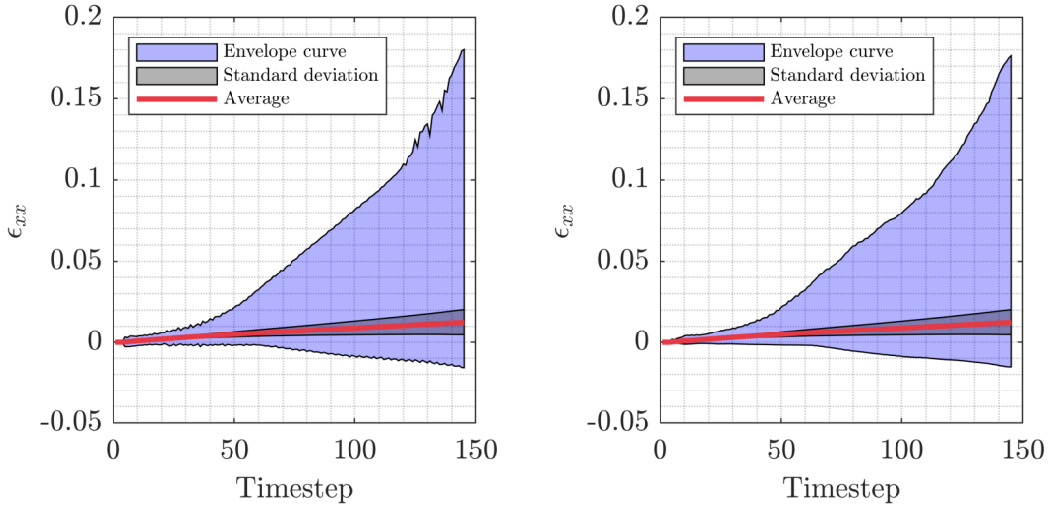
seems to be low. The visible lack of consolidation in the woven composite leads to a significant amount of voids in the material. Voids in polymers, for instance in polymer foam, are known to decrease the Poisson ration of the bulk material.

The values that have been obtained here are reference values. Since we cannot verify the the results from DDI against FEM solutions as it has been done in previous chapters, we will use the values obtained here to estimate the *error* of the DDI algorithm.

6.3.2 Woven composite, sample E1

Since we have mostly heterogeneous samples, we will focus our analysis first on the homogeneous ones, to have a reference to work with. Ideally we would like to identify first the isolated behavior and then move to the more complex case with combined materials. Given that we are doing simple tensile tests, we will limit ourselves to show only the results in the longitudinal component of the deformation (in our case, the x component of the tensors), since the other directions are poor due to the lack of testing.

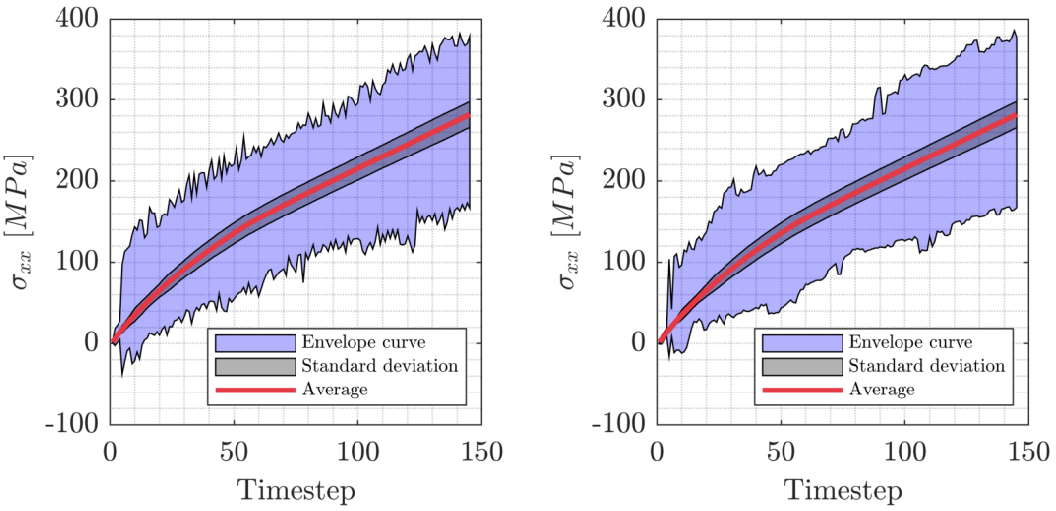
We start by analyzing sample E1, which is an homogeneous woven sample. In a first step, we compare the results for the classical DDI approach for both filtered and unfiltered data, to then also compare the results for classical DDI against the strain-rate versions of the method. Figure 6.15 shows the envelope for all the strain-time curves associated to the elements in the mesh, showing the average value and the error for each timestep for both filtered and unfiltered data. Given the way that DDI works, these values of strain are common for all the different runs involving sample E1. It can be seen that filtering the data produces a much smoother envelope, which is a sign that the individual curves are a lot less oscillating when the data is treated. Nonetheless, the average curve is not greatly affected, because of its average nature.



(a) Unfiltered. **(b)** Filtered.

Figure 6.15: Strain vs. time, sample E1.

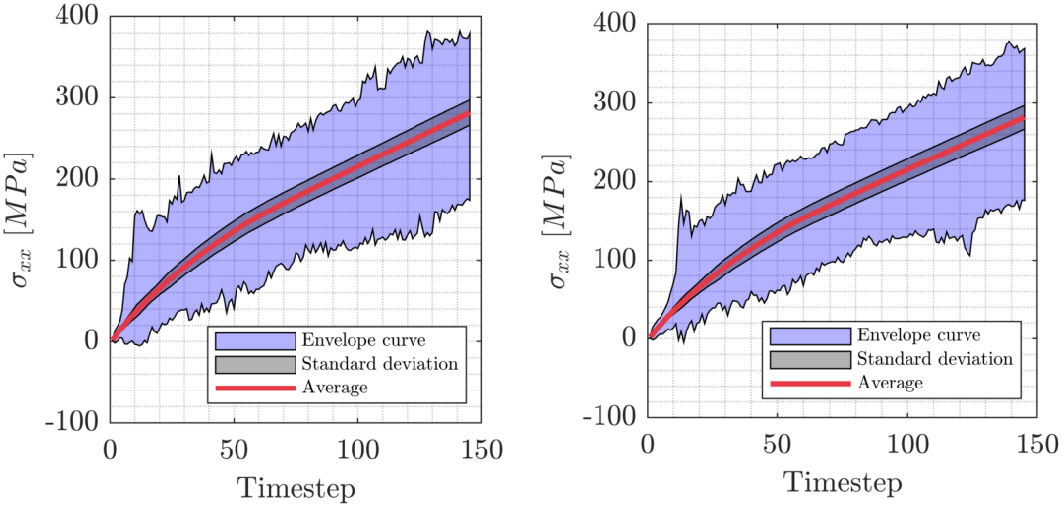
After running the classical formulation of DDI with both the filtered and unfiltered data, the obtained results for the estimated stresses are shown in Figure 6.16. In here we expected a more pronounced difference, since DDI would be run with cleaner data, however, the same pattern is noticeable. We tend to have a smoother envelope for the filtered results, but the average curve and its error tend to remain the same.



(a) Unfiltered. **(b)** Filtered.

Figure 6.16: Stress vs. time, sample E1.

The same analysis is performed for the stress considering the different versions of DDI. Figure 6.17 shows the same unfiltered curves of Figure 6.16a when the algorithm is run considering $h = 1$ and $h = 5$ previous steps. We can see some variations of the envelopes, but nothing significant. Even more, in the case with 5 previous steps we can see some added noise to the solution, which could be an indicator that the strain-rate method is not the most appropriate to study these samples.



(a) DDI with $h = 1$. **(b)** DDI with $h = 5$.

Figure 6.17: Stress vs. time, sample E1.

Since we have seen that filtering the data and performing a higher dimensional DDI are not very productive for this material, we focus on the unfiltered data with the original formulation of DDI. Figure 6.18 shows the phase space for sample E1, as well as the average curve obtained by plotting together the average strain and stress. Here we have a curve almost linear in nature, which we take as the reference for estimating the Young's modulus of the material.

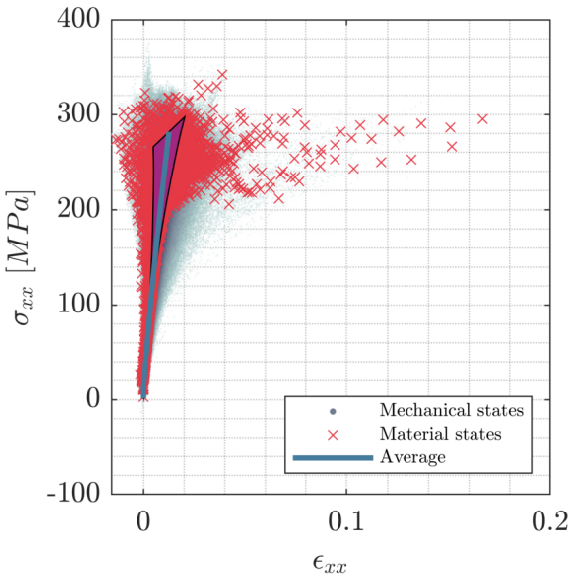


Figure 6.18: Phase space, sample E1.

We consider three different method for obtaining the modulus. The first and most simple one is to get tangent values of the modulus by computing the slope between two consecutive points, namely

$$E_t = \frac{\sigma_{i+1} - \sigma_i}{\epsilon_{i+1} - \epsilon_i}. \tag{6.1}$$

A second, less reliable option is to consider a secant value between each point and the origin of the space. This can be written as

$$E_s = \frac{\sigma_i}{\epsilon_i}. \tag{6.2}$$

We consider this case to have more information and to track how much it might vary. For both cases, since we have an almost vertical increase of the curve in the beginning due to the adjustment of the mechanical points, we do not consider the value of the Young's modulus close to the origin. We instead discard the first five steps for both and take the next one, when the curve tends to have a more stable derivative.

The last approach to obtain the value of the modulus is to perform a polynomial fitting. Even though we would be assuming the material to be relatively linear once the waviness of the weave is eliminated by stretching the fibers, we perform a third degree polynomial fit to accurately reproduce the slight curvature of the average curve. After the polynomial is fitted, the derivative is found and the instant values are taken for each point to compute a fitted modulus, E_f. Table 6.3 shows the estimated values for the three approaches.

Table 6.3: Values of Young's moduli for sample E1.

Sample	E_t (GPa)	E_s (GPa)	E_f (GPa)
E1	28.189	25.789	27.068

As it can be seen the values obtained for the Young's modulus with DDI overestimate the ones obtained by traditional means. Since we are considering a case in plane stress, the values obtained from fitting the curve are already adjusted using the Poisson ratio ν that was estimated in Section 6.3.1. In Table 6.4 the relative error with respect to the values obtained in Section 6.3.1 are shown. This is calculated using the equation

$$e^i = \frac{|E_{eng} - E_i|}{E_{eng}}, \tag{6.3}$$

where E_{eng} is the Engineering Young's modulus estimated in Section 6.3.1 and E_i is each of the different moduli calculated in this section and reported in Table 6.3.

For this woven composite, the theoretical Young's modulus is expected to be in the range of 9 GPa, while the value that we have estimated directly from the images is around 12 GPa, both much lower than the values that we have predicted with DDI. We can see that the estimations reported in Table 6.3 are in the same order of magnitude, but they are more than 100% off from the expected (theoretical) and measured values. Many causes can be attributed, particularly the error introduced by DDI, as well as propagation of other errors associated to the methodology.

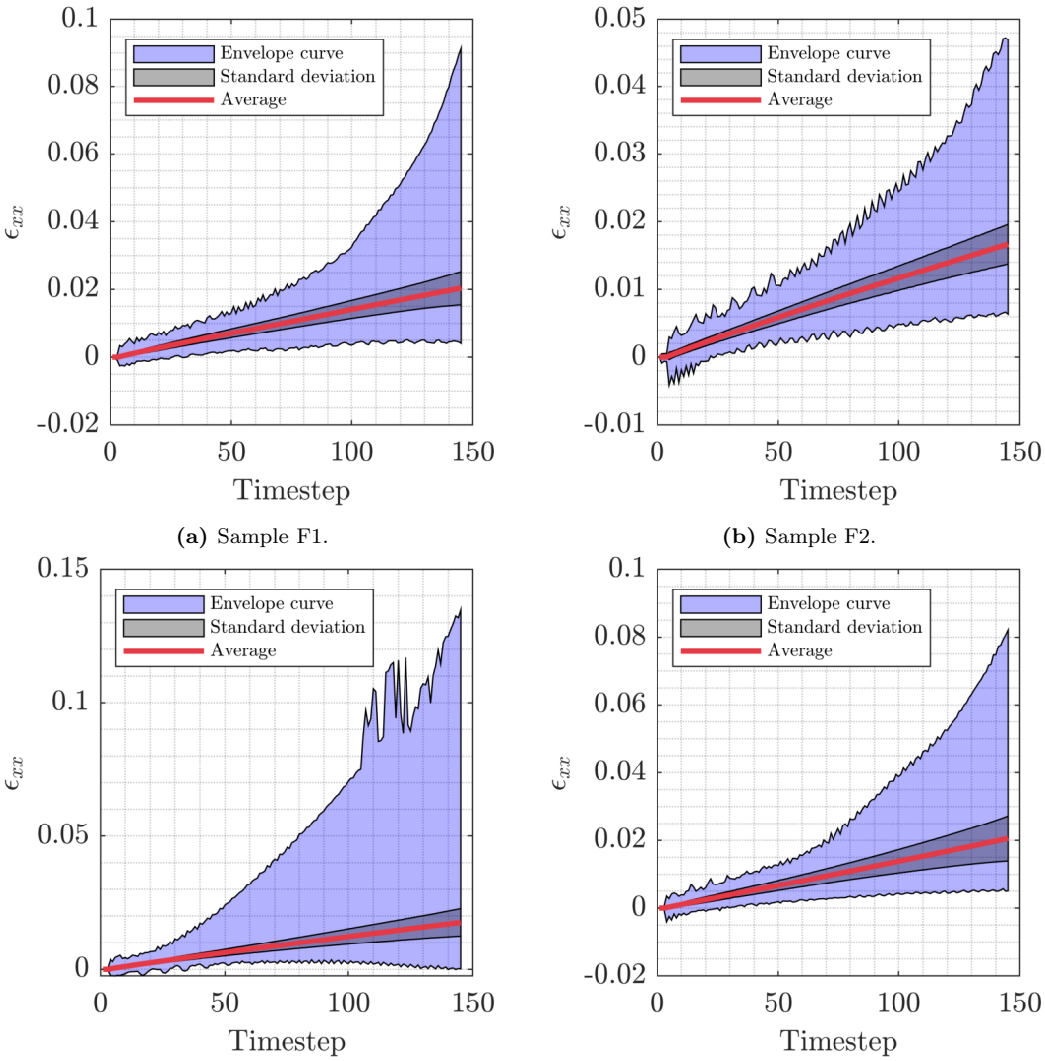
Table 6.4: Relative error of Young's moduli's estimation for sample E1.

Sample	e^t	e^s	e^f
E1	1.343	1.143	1.250

6.3.3 GMT composite, samples F and G

For the samples F and G, which are taken from GMT areas of the part, we perform a similar analysis. In these samples we will not consider filtered data nor more complicated formulations of DDI, since they didn't provide any advantage in the previous case. We also expect results to be a bit more accurate here, since the GMT material is isotropic, so we will not have issues related to the orientation of the stresses in the sample.

Figure 6.19 shows the strain vs. time curve for all four samples F1, F2, G1 and G2. We can observe that in general the envelope curves are quite smooth, hinting at a good quality of the measurements of strains obtain through DIC. For both samples G1 and G2 we have that the maximum strain achieved is higher than for samples F1 and F2. The fact that fracture took longer in one set of samples might indicate different localized resistance for the part. The flow during compression molding of GMT components may occur in axisymmetric or unidirectional manner, leading to fiber orientation distributions primarily perpendicular and longitudinal to the flow direction, respectively. Significant differences in mechanical properties will result from the flow history. In addition, layers closest to the mold surface adheres due to the rapid cooling, then the upper layers will experience more shear, hence more fiber alignment, than the inner layers.



(a) Sample F1.

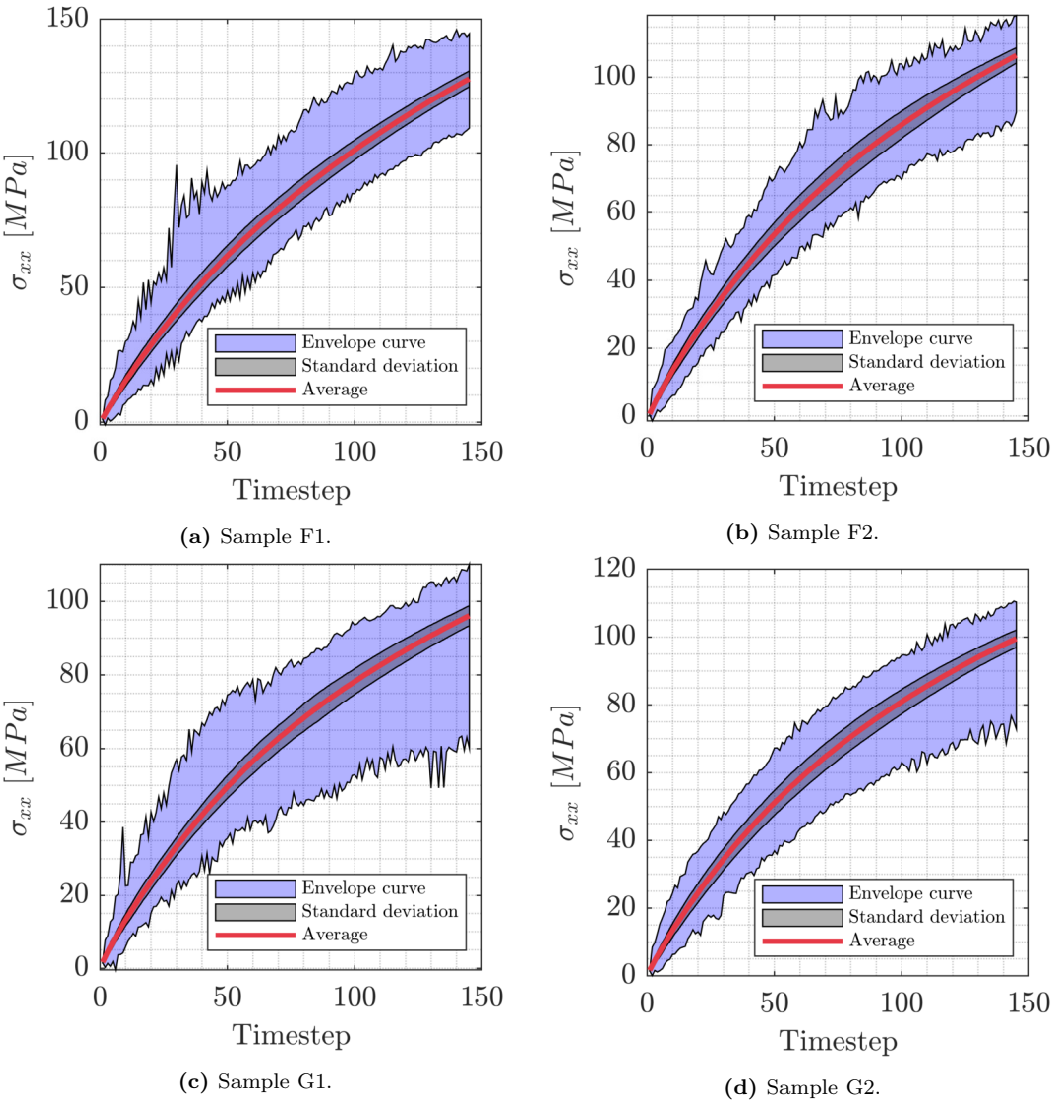
(b) Sample F2.

(c) Sample G1.

(d) Sample G2.

Figure 6.19: Strain vs. time.

Figure 6.20 shows now the stress vs. time for the same four samples. We can see that there is a correlation between the two samples obtained from the same area. Sample F1 shows a bit higher maximum stress compared to samples F1, G1 and G2, but all the values stay in very similar ranges. This correlates with the behavior observed on the strain curves, where both sets of samples behave in similar ways.



(a) Sample F1.

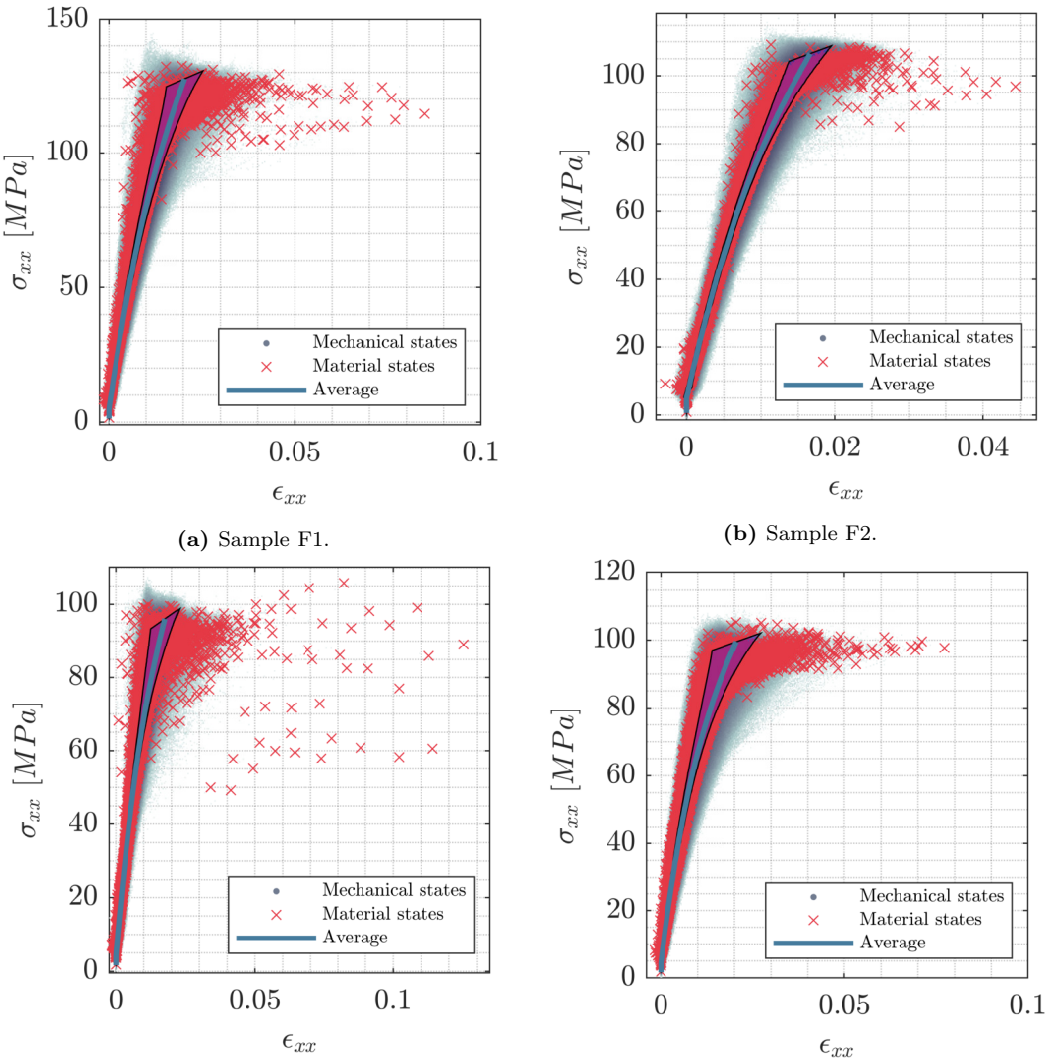
(b) Sample F2.

(c) Sample G1.

(d) Sample G2.

Figure 6.20: Stress vs. time.

Finally, phase spaces for all four samples are shown in Figure 6.21. We see that samples G1 and G2 have a bigger spread than the others, while F2 tends to have the best results. A visual analysis of the phase spaces shows similarities in the results, but it is difficult to assess it properly without performing some numerical analysis.



(a) Sample F1.

(b) Sample F2.

(c) Sample G1.

(d) Sample G2.

Figure 6.21: Phase space, samples F and G.

Table 6.5 gathers the results of the estimations for the Young's moduli, obtained using the same procedure as before. Using similar composites as reference, we expect a value of the modulus to be in the order of $E = 4.5\ GPa$, which is close to the values obtained in Section 6.3.1. We can see that compared to the woven composite, the values of moduli are closer to what is expected, but we are still far from what we should see.

Table 6.5: Values of Young's moduli for samples F and G.

Sample	$E_t\ (GPa)$	$E_s\ (GPa)$	$E_f\ (GPa)$
F1	8.318	7.950	9.073
F2	9.390	8.779	9.367
G1	7.889	7.724	8.312
G2	8.632	8.803	8.180

Similar to what we have done in Section 6.3.2, the relative error for each of the values shown are computed using Equation 6.3 and shown in Table 6.6. One of the

reasons why the results are slightly better than the woven composite case could be due to isotropy, which simplifies the estimations by removing the effects of the orientation of the fibers. However, the results shown are still not optimal, specially considering that the values from the engineering Young's modulus are closer to the theory.

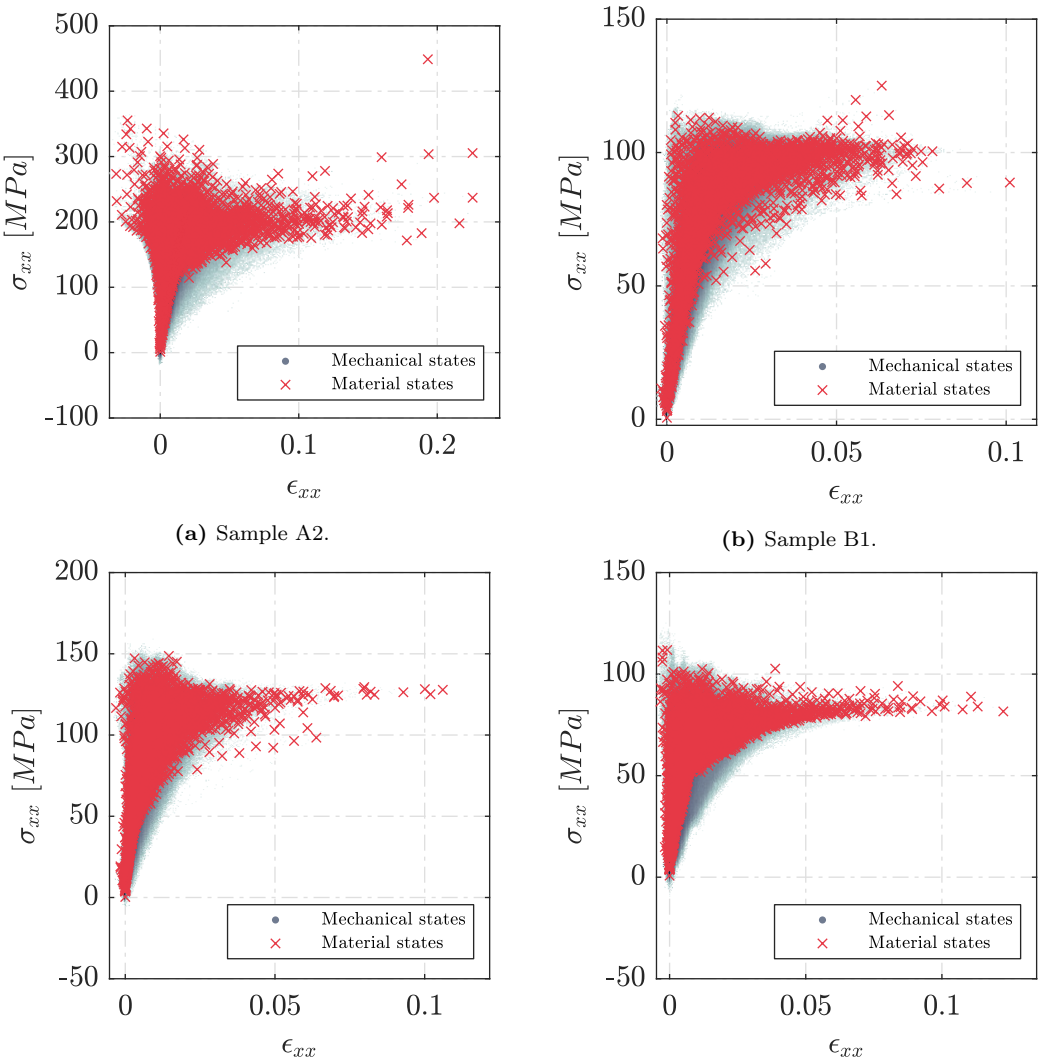
Table 6.6: Relative error of Young's moduli's estimation for samples F and G.

Sample	e^t	e^s	e^f
F1	0.813	0.733	0.978
F2	1.090	0.954	1.085
G1	0.892	0.853	0.994
G2	1.350	1.396	1.227

6.3.4 Heterogeneous composite samples

We seek to perform the same analysis for the rest of the samples composed of mixed composites, adding a correspondence analysis to find a separation that would allow us to find the different phases of the material. However, it is important to mention that if the data is as disperse as the one seen in the homogeneous samples, performing CA would return inadequate results.

To visualize the problem, we show the phase space for four samples in Figure 6.22. As we can see, a visual separation of the behavior of both materials seems impossible, since there is a smooth gradient between the stiffer responses and the softer ones.



(a) Sample A2.

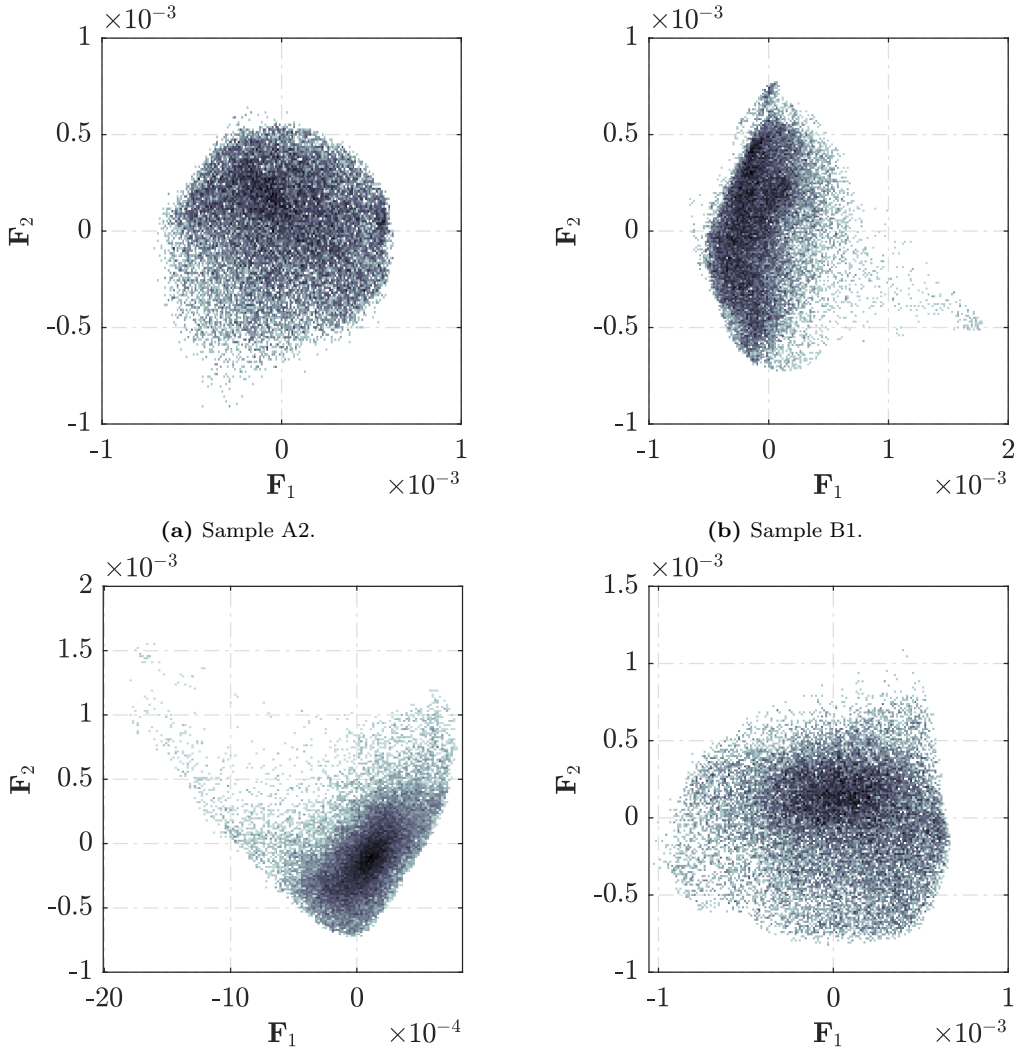
(b) Sample B1.

(c) Sample C1.

(d) Sample D1.

Figure 6.22: Phase space.

Despite the visual results, we perform CA in all samples. The $\boldsymbol{F}$ space representing the mesh elements is plotted for the same four samples in Figure 6.23 with a density plot. As can be seen, the only sample that might provide some sort of separation is the sample C1, but considering that the composition of the sample is approximately half of each material, the amount of points tending to the left of the graph seems to be more dispersion of the data rather than a cluster. For all the other samples no trend can be seen.

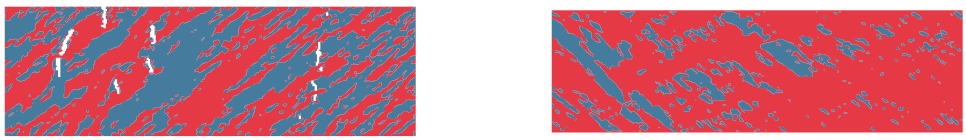


(a) Sample A2. **(b)** Sample B1.

(c) Sample C1. **(d)** Sample D1.

Figure 6.23: CA space for mesh elements.

If we do a separation for the samples A1 and C1 as an example, we obtain the results shown in Figure 6.24, where we can see that no phases have been correctly identified, but instead we tend to see stripes crossed across the sample. This phenomenon is a common result when we have an unsuccessful separation.



(a) Sample A2. **(b)** Sample C1.

Figure 6.24: Sorted meshes.

6.4 Discussion

In this chapter we applied DDI to real hybrid polymer composite samples taken from an industrial part. Up to this point the work has been highly theoretical and we have relied on synthetically generated data, which due to the nature of FEM, act as a *perfect* input for DDI. The hybrid structure is made of two polymer composites with a theoretical stiffness contrast of a factor of two. The samples were submitted to a tensile test until failure while snapshots were taken in order to perform DIC. After the strains are obtained and the force information is retrieved from the machine, different versions of DDI are performed to the samples to estimate the properties of the material and to see if a separation is possible.

The results obtained from the experiments are mixed. When performing DDI in samples who have only one material present we were able to estimate a value for the Young's modulus by making some assumptions about the behavior of the material and by averaging the behavior of all the elements in the mesh. We saw that, for the woven composite, results were in the same order of magnitude but off a theoretical value that was obtained from classical testing. However, when studying the GMT sample, values were closer to the theoretical results. There are many reasons to consider when analyzing the error in the method. Since we departed from the theoretical case, we know that every step in the retrieval of data is subjected to errors. In this case, the way in which DIC is performed can affect the quality of the results. Experimenting with different subset sizes could allow us to find better values that might produce better results. The same effect could be obtained by testing different conditions for DDI. Another important reason is the assumptions made over the material. DDI is supposed to provide information without the need of a guess, but in reality we still needed to estimate a value of the Poisson's ratio to have access to the Young's moduli. We also assumed that the material would behave somewhat elastically (at least during the extension phase), which may not be necessarily appropriate for this case.

During the course of the analysis we focused our analysis in the mechanical states, rather than the material ones, which is the opposite approach as the one seen in Chapter 3. The reason for this is the fact that with mechanical states we could track a history that would allow us to find a material curve for each individual element. In any case, the same analysis could have been performed with the material states if we would have done some regression, such as least squares, to find a polynomial fit. Results should not vary much from what we have already obtained.

With respect to the correspondence analysis, we can see that it has not provided the results we expected, and again we can put the blame in the assumptions made on the material. We have seen that both the mechanical and material states are very disperse in the phase space, which provides too much noise for CA to find a proper separation of behaviors. Something that has also been observed is the fact that CA tends not to work very well when the difference of behavior is present in a gradient, due to the fact that material curves would not be discernible. We can expect that in these samples the interface between both composites is not a hard edge, but a smooth transition between both stiffnesses, due to the flow of the GMT into the woven prepreg.

Another factor to consider is of a technical nature. Due to the manufacturing process, some areas in the material will have different properties, specially the areas with changes of geometry. The samples used here were taken from flat surfaces to avoid this problem, but we still can see that some of them have present the injection points

where the part was formed. Details like this can create small areas with a concentration of tensions that might affect the estimation of DDI. It can also cause some difficulties during the process of image collection. It could be better for future cases to sand down this areas in order to have a smoother surface to perform the analysis. Another problem could come from the part itself. The part used for the test comes from a batch of defective pieces, so the expected values from the sample can vary because of the way they were manufactured. Finally, we need to consider that the data obtained is in itself deficient because the testing performed is uniaxial, which promotes uniform strain fields. It has been seen in synthetic cases that when we have samples in this situation, the results are not properly estimated. In previous chapters we have worked in samples with holes in them precisely to avoid this effect, but in a composite sample it is not advised to cut a hole, since we disturb the structure of the fibers and we immediately lose the properties we are trying to measure in the first place. If we want the values of the Young's modulus to be better estimated, we need to perform more thorough testing in samples, either by performing complementary tests (e.g. shear or biaxial testing) or developing samples that promote this non-uniform behavior, which is also beneficial for the possibility of orthotropy in the material.

As a final remark, we would like to point out that the experimental part and the analysis of the samples were performed by different people. It could be beneficial for the testing that all the process is done by one person, since some things can be left out between the steps of the analysis, which of course increases the chance of having errors later.

DDI is a good tool that provides a nice solution for the problem of unknown properties. In previous works we have seen it successfully applied to samples of different materials, but in this case we cannot say that this was a successful experience. In general, when we have homogeneous samples the results tend to be more manageable, but we clearly need better opportunities to study heterogeneous samples. Ideally we would like to start analyzing known elastic materials first and then move on to more complicated cases such as composites. Faurecia initially had provided samples made of aluminum welded together to test this methodology, but problems during the image collection rendered the samples not fit for analysis. It is possible that these samples could have provided a better setup for the study of heterogeneous DDI. In an ideal future, it would be beneficial for DDI to be able to perform the measurement of the displacement fields on the full part, rather than just cut areas. Composites rely on a structure that it is difficult to modify once the part is manufactured, so any sample cut from the original piece will not necessarily represent the reality of the behavior. On the other hand, mechanical loadings applied to a formed part can generate a richer deformation state than that of a simple tensile specimen, which is advantageous for the DDI approach. If the technology was available to compute a full displacement field, we would expect the results from DDI to be sharper, but also better targeted for the use that the company wants to give to the particular designed part.

Chapter 7

Conclusion

In this work, we have explored in depth the field of *Data-Driven Identification* [11], in which the constitutive relationship of an elastic material is obtained through measurements of the strain in samples, generally performed using imaging techniques. The method also allows us to estimate the stresses associated to the deformations in the sample, which is a quantity that is usually difficult to compute without having previous knowledge of the material behavior.

During the course of this document, we have had as an objective extending the application of DDI, departing from elasticity towards more complex material behaviors. It is because of this that after recalling the relevant topics and state of the art in Chapter 1, we introduced briefly DDI in Chapter 2. DDI is based on an algorithm introduced by Kirchdoerfer and Ortiz called *Data-Driven Computational Mechanics* [17], in which we use pairs of strain-stress measurements previously obtained in samples to replace the constitutive law used in techniques such as the *Finite Element Method*.

In Chapter 3, we apply the DDI methodology to heterogeneous samples conformed by a linear elastic matrix and inclusions, in an attempt to identify both material behaviors simultaneously. This case is taken due to the possibility of having more than one value of stress for any given strain, which is a behavior that contrasts with the unique solutions that are obtained in elastic materials. In here we implemented a post-processing technique to obtain separated results and to be able to identify both phases and behaviors of the mixed samples. As a natural progression, in Chapter 4 we go deeper into non-elasticity, by testing the DDI algorithm in linear viscoelastic models. The dependence of viscoelastic stresses on the history of deformations in a sample poses a difficulty for the method, since until now we have worked only on instantaneous strain-stress pairs. We tackled this problem by proposing an extension of the phase space, where we choose consecutive strain states to be associated to a stress state, allowing us to include the time dependence into the method.

Chapter 5 is a departure of the trend of previous chapters. Instead of testing DDI on new materials, we take a look at the way the algorithm is defined and propose some changes in it to try to optimize the efficiency and accuracy of the method. In particular, we use statistical techniques to treat the data during the algorithm, both to clean the noise in the calculations or to reduce the dimensionality of the problem. In the same vein, we use these techniques also as post-processing tools, which gave us cleaner and more accurate results, specially improving the performance of the Correspondence Analysis for the separation of phases.

Finally, Chapter 6 is dedicated to an experimental verification of the DDI methodology. With samples provided by the industrial partner of this project, we perform *Digital Image Correlation* on mixed polymer composite samples to obtain strain fields that we could then use to perform DDI and obtain an estimation of the behavior of the parameters. Given the way the tests were performed, results were mixed, but they were helpful in shedding light in ways that we can improve the handling of samples for future testing of the DDI method.

After summarizing the contents of the project, we can outline the main contributions:

- We have successfully modified the DDI algorithm to include the time dependence phenomena of linear viscoelastic materials. By extending the phase space we have consistently obtained better results that are closer to what we have obtained by simulating different samples with FEM, specially in cases where the viscoelastic effects are more pronounced, where the original formulation of DDI tends to fall short.

- We have also managed to introduce statistical analysis into DDI as a solution for specific problems. In this work, we have particularly shown that:

  - Correspondence Analysis is a good tool for performing the separation of phases in heterogeneous behaviors. By analyzing the mechanical and material states as categorical data, we can perform statistical analysis that allows us to recover certain trends in the data. CA provides us with an alternative visualization of the data where we can find clusters representing the different behaviors in a sample, allowing us to retrieve simultaneously the constitutive relation of both phases, as well as the location of the phases in the original mesh.
  - Introducing normalization in the DDI procedure improves the results at the expense of efficiency on the calculations. However, particularly in the case of softer inclusions, it has proven to be a successful trick to identify the phases of the sample, which until now it was a problem that has always been present in DDI. Normalizing the algorithm also has the advantage of ridding us from the use of the parameter $\mathbb{C}$, which simplifies the application of the algorithm.
  - Principal Component Analysis allows us to reduce the dimensionality of the problem, as well as providing us with a different visualization of the data in a similar fashion to Correspondence Analysis. The reduction of the problems has proven useful, specially in the case of viscoelasticity where we have seen that the extension of the phase space can rapidly grow to levels where the computation becomes slow and inconvenient. We have obtained consistent results when using a reduced dimensionality formulation in the case of viscoelasticity, which produces more accurate results than other versions of DDI, due to the removal of noise as a side effect from PCA.
  - We have also seen that the filtering properties of PCA are useful for post-processing the results of DDI. When using these methods after running DDI we have slightly reduced the error of the estimations. However, we have an improved performance of the Correspondence Analysis, where the clusters are much more apparent and a separation of phases becomes easier.

With all this in hand, we can also have some perspective on the future of the field and possible continuation for this work.

- Since CA has provided good results for heterogeneous samples, it is natural to think that it could be applied in other cases as well. In that sense, it could be interesting to see if the methodology could be applied in orthotropic samples, which would allows us to see the different behaviors inside the same material.
- We have also seen that the extension of the phase space is a valid option that provides a good improvement over the original DDI for cases that are not elastic. This approach is not exclusive and it has been applied effectively in other type of materials (see [23] for inelasticity, for example), so it might be possible to keep pushing for more complex behavior, specially inside of the viscoelasticity field.
- We can also find better ways to integrate data science into the DDI formulation. PCA is a relatively simple technique that provided improvements to our results. In this sense, more complex techniques could be studied and implemented into our methodology to provide new variations that can improve the visualization of the problem and the results obtained.
- Finally, every day there are improvements in every field, including the ones concerning the recollection of data. In this sense, we can realistically expect to have better input to use in the data-driven methods and move beyond the two-dimensional cases. Improvements in 3D imaging, the use of sensors inside of samples, better cameras, etc.; will provide new visualizations that will push the boundaries of what it is possible with DDI and DDCM with exciting prospects.

Appendix A

Digital Image Correlation

Many experimental settings in solid mechanics rely in having access to a displacement field of the sample. Although different techniques can be used for this purpose, DIC stands out due to its simplicity and accessibility, since the setup needed to obtain results depends only in a good camera and proper lighting conditions.

Image detection and gray levels

The basic procedure for performing DIC is to detect images of the surface of a sample. Normally a digital camera is used to obtain grayscale pictures of different states of deformation and then a software is used to compare the different images, tracking the movement of different points in the picture. To be able to track this displacement, the image is divided in subsets, areas of a certain amount of pixels that are tracked. Subsets work under the assumption that the gray level of each subset is unique and kept constant during the full deformation, so the software matches these areas. An example of how this works is shown in Figure A.1.

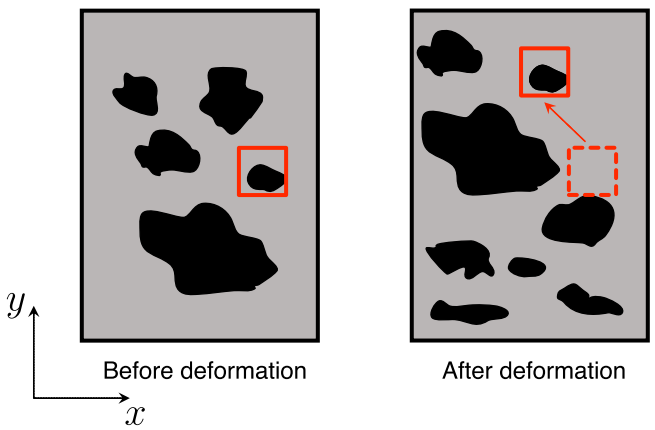


Figure A.1: Example of how the subset is tracked in the sample. The subset is represented by the red square, whose movement is observed going from the original position (dashed line) over to the updated position after deformation (solid line).

Due to the need of having some visible and unique characteristic for the tracking, typically an artificial randomized pattern is applied to the surface of the sample. The random speckle pattern that has been already seen in Chapter 6 allows to have a non-uniform distribution of gray intensities along the sample, avoiding problems on the tracking due to regularity.

To perform the tracking of subsets, the intensity levels of each pixel are defined as a function $F(x, y)$ that represents the gray level value measured from 0 to 255, with 0 being black and 255 being white. Since this function is discrete due to the nature of a pixel image, an interpolation is performed to obtain a continuous representation of the gray level in the full image, which poses the advantage of being able to track subpixel deformations.

Estimating deformation in the sample

To deal with deformation in the images, we use the kinematic theory of deformation. Assuming that the displacement gradients are constant through the subset, we can express the deformed coordinates (x^*, y^*) as [67]

$$\begin{aligned} x^* &= x + u_x + u_{x,x}\Delta x + u_{x,y}\Delta y, \\ y^* &= y + u_y + u_{y,x}\Delta x + u_{y,y}\Delta y, \end{aligned} \tag{A.1}$$

where Δx and Δy represent the distance from the center of a subset to the point (x, y) in the respective coordinates, and u represents the deformation of the original point. In here we use tensorial notation for the derivatives, where $u_{i,j} = \frac{\partial x_i}{\partial x_j}$.

If we consider a subset centered at a point P at the reference, when it moves it goes to a new point P^* and it deforms. The values of gray levels at positions P and P^* can be written as [68]

$$\begin{aligned} F(P) &= F(x, y), \\ F^*(P^*) &= F^*(x + u_x, y + u_y). \end{aligned} \tag{A.2}$$

Similarly for a point Q at a position $(x + \Delta x, y + \Delta y)$ before deformation, the gray level function for for this point in its original and deformed state can be written as

$$\begin{aligned} F(Q) &= F(x + \Delta x, y + \Delta y), \\ F^*(Q^*) &= F^*(x + v_x + \Delta x, y + v_y + \Delta y), \end{aligned} \tag{A.3}$$

where v is the deformation of point Q.

If we assume that the subset taken is small enough that straight lines remain straight after the deformation, as well as assuming that the gray levels remain the same during the full process (i.e., $F(Q) = F^*(Q^*)$), we can define the position of point Q^* as

$$\begin{aligned} (x^{**}, y^{**}) &= (x^* + \Delta x^*, y^* + \Delta y^*) \\ &= (x + u_x + \Delta x^*, y + u_y + \Delta y^*) \\ &= (x + u_x + u_{x,x}\Delta x + u_{x,y}\Delta y + \Delta x, y + u_y + u_{y,x}\Delta x + u_{y,y}\Delta y + \Delta y), \end{aligned} \tag{A.4}$$

and using Equation A.1, we can rewrite Equation A.3 as

$$F^*(Q^*) = F(x + u_x + u_{x,x}\Delta x + u_{x,y}\Delta y + \Delta x, y + u_y + u_{y,x}\Delta x + u_{y,y}\Delta y + \Delta y), \quad \text{(A.5)}$$

meaning that, if we know the displacement of the center point P and the gradients of deformation, then any point Q^* can be calculated. In the same vein, if we assume the values for u and its derivatives, we can compute estimates of P^* and the point Q^* in a small subset around. This is the basis of the image correlation analysis.

To obtain accurate estimations of the deformation and its derivatives, subsets are compared. We start by taking an undeformed subset centered at P, which is interpolated to obtain the continuous function F. Then, values for the deformation u are taken while keeping the other parameters at zero, meaning that we assume that only a translation took place. The values of (u_x, u_y) that give the best comparison between two images are taken as an approximate center for the deformed position. The most common method to compare two subsets is through the use of a cross-correlation coefficient C computed as

$$C(u_x, u_y, u_{x,x}, u_{x,y}, u_{y,x}, u_{y,y}) = \frac{\int_{M^*} F(x,y)F^*(x^*,y^*)\ dA}{\sqrt{\int_M F(x,y)^2\ dA \int_{M^*} F^*(x^*,y^*)^2\ dA}}, \quad \text{(A.6)}$$

where M represents a subset in the original state and M^* a deformed one. The values that maximize C are the local deformations for the subset, so the problem is a maximization one. This is typically solved by obtaining first the values for u until we have our approximate new center, then we iterate for different values of the gradient until we settle. This process can be repeated with smaller ranges of deformation centered around the new position to minimize the error of the deformations. Once the values of u and the derivatives are obtained, the strain fields can be computed with the finite strain theory, i.e.,

$$\epsilon_{xx} = u_{x,x} - \frac{1}{2}(u_{x,x}^2 + u_{y,x}^2), \quad \text{(A.7a)}$$

$$\epsilon_{yy} = u_{y,y} - \frac{1}{2}(u_{x,y}^2 + u_{y,y}^2), \quad \text{(A.7b)}$$

$$\epsilon_{xy} = \frac{1}{2}(u_{x,y} + u_{y,x}) - \frac{1}{2}(u_{x,x}u_{x,y} + u_{y,x}u_{y,y}). \quad \text{(A.7c)}$$

Appendix B

Unsupervised sorting: k-means algorithm

K-means [32] is one of the most popular methods for partitioning and clustering sets of data, given the speed and simplicity in which it achieves results. The main idea of the algorithm is that, for a set of dataset of n measurements in $\mathbb{R}^d$, we seek to find k *points* that reduce the distance between each measurement and their assigned centroid. One of the drawbacks of the method is the accuracy of it, as well as the uniqueness of the solution. Depending on the distribution of the measurements, k-means may provide bad clustering since it only takes into account distances rather than how this distances are related. Also, the results are heavily dependent on where the initial seeds for the centroids are placed. If the seeds are chosen randomly, there is the possibility of centroids being close to each other or misrepresenting certain sections of the dataset.

The algorithm for k-means is outlined in Algorithm B.1. This variant is performed with a k-means++ initialization [44], which seeds the initial centroids based on a probability distribution rather than a uniform random pick. In here, $D(\boldsymbol{x})$ is a function that describes the shortest distance from a measurement $\boldsymbol{x}$ to a closest center already chosen, $\boldsymbol{z}$.

Algorithm B.1: K-means algorithm with k-means++ initialization.

Input: Dataset $\mathcal{X} \subset \mathbb{R}^d$, number of clusters k
Output: Subsets $\mathcal{X}_k \subset \mathcal{X}$, centroids $\boldsymbol{z}_k \in \mathbb{R}^d$

K-means procedure

Pick centroid $\boldsymbol{z}_1$ uniformly at random from $\mathcal{X}$.

Pick each centroid $\boldsymbol{z}_i$ for $i = 2, \ldots, k$, choosing $\boldsymbol{x} \in \mathcal{X}$ with probability $\frac{D(\boldsymbol{x})^2}{\sum_{\boldsymbol{x} \in \mathcal{X}} D(\boldsymbol{x})^2}$

while $\boldsymbol{z}_k$ *keeps changing* **do**

For each $i = 1, 2, \ldots, k$ define $\mathcal{X}_i$ as the subset of measurements that are closer to $\boldsymbol{z}_i$.

For each $i = 1, 2, \ldots, k$ define $\boldsymbol{z}_i$ as the center of mass of all the points in $\mathcal{X}_i$, i.e.,

$$\boldsymbol{z}_i = \frac{1}{|\mathcal{X}_i|} \sum_{\boldsymbol{x} \in \mathcal{X}_i} \boldsymbol{x}.$$

The total squared distance of all the measurements to their centroids is a monotonically decreasing function, so the algorithm ensures that there will not be two equal configurations in any two steps. Also, since there is a limited amount of possible clusters (k^n), we know that the process will always reach to a solution.

Appendix C

Singular Value Decomposition

Singular Value Decomposition (SVD, [69]) is a factorization performed on a matrix. It is a generalization of the eigendecomposition oriented towards rectangular matrices. Eigendecomposition factorizes a square matrix into two, one orthogonal and one diagonal matrix. SVD provides a similar output, considering two orthogonal matrices and one diagonal, analogous to the square case.

If we consider any rectangular matrix $\boldsymbol{M}$ of size $m \times n$ with rank $r \leq \min(m, n)$, the SVD of $\boldsymbol{M}$ can be expressed as

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{*}, \tag{C.1}$$

where $\boldsymbol{U}$ is an $m \times m$ matrix whose columns represent the singular vectors (an analogous to the eigenvectors of a square matrix) of $\boldsymbol{M}$ associated to the rows, $\boldsymbol{\Sigma}$ is a diagonal matrix of size $m \times n$ with the singular values (analogous to the eigenvalues of a square matrix) ordered in decreasing size; and $\boldsymbol{V}^{*}$ is a matrix of size $n \times n$ with the rows containing the singular vectors associated to the columns of $\boldsymbol{M}$. The relation between SVD and eigendecomposition can be seen in the definition of $\boldsymbol{U}$ and $\boldsymbol{V}$, since $\boldsymbol{U}$ correspond to the orthonormal eigenvectors of the square matrix $\boldsymbol{M}\boldsymbol{M}^{T}$, while $\boldsymbol{V}$ is the equivalent for matrix $\boldsymbol{M}^{T}\boldsymbol{M}$. In fact, if $\boldsymbol{M}$ is a positive semi-definite square matrix, SVD and eigendecomposition are equal. Figure C.1 shows a graphical representation of this decomposition and its relations.

The particularity of the decomposition lies in the fact that each element of the matrix $\boldsymbol{\Sigma}$ is associated to a pair of eigenvectors from matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. These values can be used to obtain an approximation $\boldsymbol{M}^{(k)}$ of lower rank of the original matrix, by considering the k singular vectors associated to the k biggest singular values, and discarding the lower ones. The original matrix can be reconstructed as

$$\boldsymbol{M}^{(k)} = \boldsymbol{U}^{(k)}\boldsymbol{\Sigma}^{(k)}\boldsymbol{V}^{*(k)}, \tag{C.2}$$

where $\boldsymbol{U}^{(k)}$ is the reduced matrix of size $m \times k$ formed by the first k columns of $\boldsymbol{U}$, and a similar truncation is taken for $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$. This is the main formulation in which dimensionality reduction techniques are based on.

Another property of the SVD is that the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are each one a set of orthonormal vectors, which means that they represent a basis on its own. Because

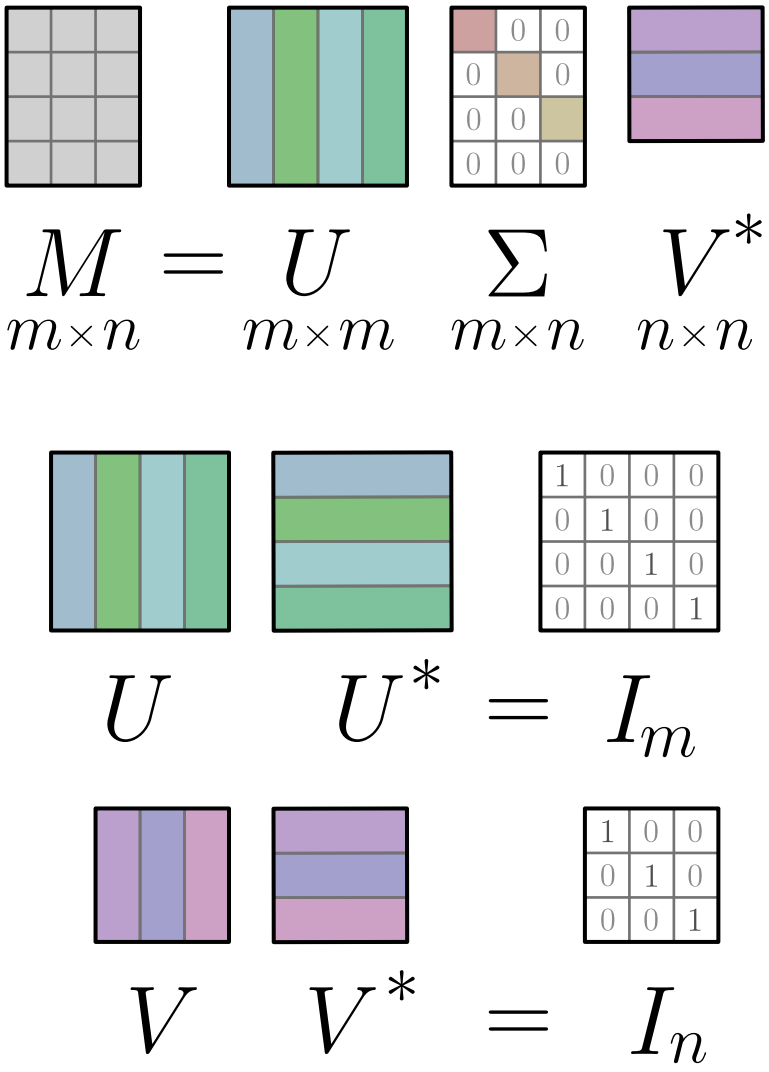


Figure C.1: Schematics of the SVD.

of this, the space created by both the columns of $\boldsymbol{U}$ or $\boldsymbol{V}$ represent projections of the data in different spaces, which gives alternative visualizations and serves as the basis for techniques such as the Principal Component Analysis (PCA, [54]) and Correspondence Analysis (CA, [39]), which are featured in this work.

Appendix D

Correspondence Analysis

Correspondence Analysis (CA) is a multivariate technique that transforms a matrix of nonnegative data, usually known as a *contingency table*, into a graphical display in which the rows and columns of this table are depicted as points [42]. It is analogous to the Principal Component Analysis (PCA) with the exception that it is tailored to categorical rather than continuous data. In its simplest form, it is used in two-dimensional tables, but its extension, known as *Multiple Correspondence Analysis* (MCA, [70]), is the generalization for higher-dimensional data.

Procedure

The general procedure for performing CA is outlined here as presented in [39]. The application to DDI is already outlined from Equation 3.5 to 3.11 in more detail, so a simpler version is presented in Algorithm D.1. In here, I and J correspond to the dimensions of the table $\boldsymbol{N}$, while $\mathbf{1}$ is a column vector full of ones of an appropriate dimension for the multiplication.

In all the coordinates computed in Equation D.6 and D.7, the rows of the matrices $\boldsymbol{\Phi}/\boldsymbol{F}$ and $\boldsymbol{\Gamma}/\boldsymbol{G}$ correspond to each one of the columns or rows of the table $\boldsymbol{N}$ respectively, while the columns are known as the principal axes or dimensions. Since the process includes performing an SVD, it means that each one of these coordinates are weighted values of the vectors that reduce the covariance of the matrix $\boldsymbol{S}$.

The reason why in CA we perform the SVD in the matrix $\boldsymbol{S}$ rather than directly applying it to the data $\boldsymbol{P}$ as it would be done in the PCA is related to the measured distance between points. In PCA, we work with a covariance between all points based on an euclidean distance. The matrix of standard residuals $\boldsymbol{S}$, however, can be defined as the weighted average of the χ-squared distances [53] between rows and columns to their centroids. The choice of χ-squared distances is logical because each distance can now be interpreted geometrically as a deviation of each row or column profile from the origin, which is also known as the *inertia*. In this sense, we can understand that the distance of each point from the origin, for each one of the points in the principal or standard components, represents how different they are from an homogeneous case, which is the reason why we focus on obtaining clusters of points in this projected space.

Algorithm D.1: General procedure for Correspondence Analysis.

Input: Contingency table, $\boldsymbol{N}$

Output: Standard coordinates, $\boldsymbol{\Phi}$,$\boldsymbol{\Gamma}$. Principal coordinates, $\boldsymbol{F}$,$\boldsymbol{G}$. Principal inertias, λ_k.

Pre-processing of the contingency table

Calculation of corresponding matrix ($\boldsymbol{P}$):

$$\boldsymbol{P} = \frac{1}{n}\boldsymbol{N} \tag{D.1}$$

where n is the sum of all elements in $\boldsymbol{N}$.

Obtain row (r_i) and column masses (c_j):

$$\begin{aligned} r_i &= \sum_{j=1}^{J} p_{ij} \quad c_j = \sum_{i=1}^{I} p_{ij} \\ \boldsymbol{r} &= \boldsymbol{P}\mathbf{1} \qquad \boldsymbol{c} = \boldsymbol{P}^T\mathbf{1} \end{aligned} \tag{D.2}$$

Form diagonal row ($\boldsymbol{D}_r$) and column masses ($\boldsymbol{D}_c$):

$$\boldsymbol{D}_r = \mathrm{diag}(\boldsymbol{r}) \quad \text{and} \quad \boldsymbol{D}_c = \mathrm{diag}(\boldsymbol{c}) \tag{D.3}$$

Correspondence Analysis procedure

Calculation of the matrix of standard residual ($\boldsymbol{S}$):

$$\boldsymbol{S} = \boldsymbol{D}_r^{-1/2}(\boldsymbol{P} - \boldsymbol{r}\boldsymbol{c}^T)\boldsymbol{D}_c^{-1/2} \tag{D.4}$$

Perform SVD calculation of $\boldsymbol{S}$:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{D}_\alpha\boldsymbol{V}^T \tag{D.5}$$

Calculation of standard coordinates of rows ($\boldsymbol{\Phi}$) and columns ($\boldsymbol{\Gamma}$):

$$\begin{aligned} \boldsymbol{\Phi} &= \boldsymbol{D}_r^{-1/2}\boldsymbol{U} \\ \boldsymbol{\Gamma} &= \boldsymbol{D}_c^{-1/2}\boldsymbol{V} \end{aligned} \tag{D.6}$$

Calculation of principal coordinates of rows ($\boldsymbol{F}$) and columns ($\boldsymbol{G}$):

$$\begin{aligned} \boldsymbol{F} &= \boldsymbol{\Phi}\boldsymbol{D}_\alpha \\ \boldsymbol{G} &= \boldsymbol{\Gamma}\boldsymbol{D}_\alpha \end{aligned} \tag{D.7}$$

Calculation of principal inertias (λ_k):

$$\lambda_k = \alpha_k^2, \quad k = 1, 2, \ldots, K \text{ where } K = \min(I-1, J-1) \tag{D.8}$$

In here, α_k are the diagonal terms of matrix $\boldsymbol{D}_\alpha$.

A total inertia of the problem can be computed as

$$t = \sum_{i=1}^{I} \sum_{j=1}^{J} \left[\frac{(p_{ij} - r_i c_j)^2}{r_i c_j} \right], \tag{D.9}$$

which would be seen as a measure of the heterogeneity of the data.

The results that we obtain from the CA are presented in a K-dimensional space, where $K = \min(I-1, J-1)$, however, since we are using the SVD, we can apply similar procedures as in the PCA. In this sense, we know that we can recreate a matrix $\boldsymbol{S}^{(m)}$, which is formed by taking the m first columns of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, and the biggest m α_k inertias. This matrix $\boldsymbol{S}^{(m)}$ is a low-rank approximation of the original matrix $\boldsymbol{S}$, meaning that CA can provide a low-dimensional categorization of the distribution of the data. In the case of CA, each one of the inertias represent how heterogeneous the data is in that dimension, which is the reason why we want to visualize the information produced by the highest components. If the data tends to be more homogeneous, we need to use more components to visualize the clusters since the χ-distances tend to be closer, which gives more components having relevant information, something that we also have seen when using the PCA method.

Bibliography

[1] George Thomas Mase and George E. Mase. *Continuum Mechanics for Engineers.* CRC Press, Boca Raton, Fla, 2nd ed edition, 1999.

[2] G. D. Spathis. Polyurethane elastomers studied by the Mooney–Rivlin equation for rubbers. *Journal of Applied Polymer Science*, 43(3):613–620, 1991.

[3] Olivier A. Bauchau and Changkuan Ju. First-Principles-Based Modeling of Elastomeric Dampers for Multibody Systems. In *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 41–49. American Society of Mechanical Engineers Digital Collection, July 2010.

[4] J. M. Brader, M. E. Cates, and M. Fuchs. A First-Principles Constitutive Equation for Suspension Rheology. *Physical Review Letters*, 101(13):138301, September 2008.

[5] Vinithra Venugopal, Hao Zhang, Robert Northcutt, and Vishnu Baba Sundaresan. A thermodynamic chemomechanical constitutive model for conducting polymers. *Sensors and Actuators B: Chemical*, 201:293–299, October 2014.

[6] Gianpietro Elvio Cossali and Simona Tonini. Introduction to Constitutive Equations. In Gianpietro Elvio Cossali and Simona Tonini, editors, *Drop Heating and Evaporation: Analytical Solutions in Curvilinear Coordinate Systems*, Mathematical Engineering, pages 207–224. Springer International Publishing, Cham, 2021.

[7] Gerhard A. Holzapfel. *Non-Linear Solid Mechanics, a Continuum Approach for Engineering.* John Wiley & Sons, 2000.

[8] William S. Slaughter. *The Linearized Theory of Elasticity.* Birkhäuser Boston, Boston, 2002.

[9] Sungmoon Jung and Jamshid Ghaboussi. Neural network constitutive model for rate-dependent materials. *Computers & Structures*, 84(15):955–963, June 2006.

[10] S. Freitag, W. Graf, and M. Kaliske. A material description based on recurrent neural networks for fuzzy data and its application within the finite element method. *Computers & Structures*, 124:29–37, August 2013.

[11] Adrien Leygue, Michel Coret, Julien Réthoré, Laurent Stainier, and Erwan Verron. Data-based derivation of material response. *Computer Methods in Applied Mechanics and Engineering*, 331:184–196, 2018.

[12] Rubén Ibañez, Emmanuelle Abisset-Chavanne, Jose Vicente Aguado, David Gonzalez, Elías Cueto, and Francisco Chinesta. A Manifold Learning Approach to Data-Driven Computational Elasticity and Inelasticity. *Archives of Computational Methods in Engineering*, 25(1):47–57, 2018.

[13] José Crespo, Marcos Latorre, and Francisco Javier Montáns. WYPIWYG hyperelasticity for isotropic, compressible materials. *Computational Mechanics*, 59(1):73–92, 2017.

[14] Marcos Latorre and Francisco Javier Montáns. What-You-Prescribe-Is-What-You-Get orthotropic hyperelasticity. *Computational Mechanics*, 53(6):1279–1298, 2014.

[15] Stéphane Avril, Marc Bonnet, Anne-Sophie Bretelle, Michel Grédiac, François Hild, Patrick Ienny, Félix Latourte, Didier Lémosse, Stéphane Pagano, Emmanuel Pagnacco, and Fabrice Pierron. Overview of Identification Methods of Mechanical Parameters Based on Full-field Measurements. *Experimental Mechanics*, 48(4):381, 2008.

[16] Michael A. Sutton, Jean-José Orteu, and Hubert W. Schreier. *Image Correlation for Shape, Motion and Deformation Measurements: Basic Concepts, Theory and Applications*. Springer Science & Business Media, New York, first edition, 2009.

[17] T. Kirchdoerfer and M. Ortiz. Data-driven computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 304:81–101, 2016.

[18] J. Ghaboussi, J. H. Garrett, and X. Wu. Knowledge-Based Modeling of Material Behavior with Neural Networks. *Journal of Engineering Mechanics*, 117(1):132–153, January 1991.

[19] T. Kirchdoerfer and M. Ortiz. Data Driven Computing with noisy material data sets. *Computer Methods in Applied Mechanics and Engineering*, 326:622–641, November 2017.

[20] Auriane Platzer, Adrien Leygue, Laurent Stainier, and Michael Ortiz. Finite element solver for data-driven finite strain elasticity. *Computer Methods in Applied Mechanics and Engineering*, 379:113756, June 2021.

[21] Sergio Conti, Stefan Müller, and Michael Ortiz. Data-Driven Finite Elasticity. *Archive for Rational Mechanics and Analysis*, 237(1):1–33, 2020.

[22] Auriane Platzer. *Mécanique Numérique En Grandes Transformations Pilotée Par Les Données: De La Génération de Données Sur Mesure à Une Stratégie Adaptative de Calcul Multiéchelle*. Theses, École centrale de Nantes, December 2020.

[23] Robert Eggersmann, Trenton Kirchdoerfer, Stefanie Reese, Laurent Stainier, and Michael Ortiz. Model-Free Data-Driven inelasticity. *Computer Methods in Applied Mechanics and Engineering*, 350:81–99, 2019.

[24] Pietro Carrara, Laura De Lorenzis, Laurent Stainier, and Michael Ortiz. Data-driven fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 372:113390, 2020.

[25] Pietro Carrara, Michael Ortiz, and Laura De Lorenzis. Data-Driven Rate-Dependent Fracture Mechanics, 2021.

[26] Marie Dalémat. *Une Expérimentation Réussie Pour l'identification de La Réponse Mécanique sans Loi de Comportement : Approche Data-Driven Appliquée Aux Membranes Élastomères*. Theses, École centrale de Nantes, December 2019.

[27] Marie Dalémat, Michel Coret, Adrien Leygue, and Erwan Verron. Measuring stress field without constitutive equation. *Mechanics of Materials*, 136:103087, 2019.

[28] Robert Eggersmann, Laurent Stainier, Michael Ortiz, and Stefanie Reese. Efficient Data Structures for Model-free Data-Driven Computational Mechanics, 2020.

[29] Robert Eggersmann, Laurent Stainier, Michael Ortiz, and Stefanie Reese. Model-free data-driven computational mechanics enhanced by tensor voting. *Computer Methods in Applied Mechanics and Engineering*, 373:113499, 2021.

[30] Daniel Zwillinger. *Handbook of Differential Equations*. Academic Press, third edition, 1997.

[31] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, Oxford, seventh edition, 2013.

[32] James B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability: Theory of Statistics*, volume 1, pages 281–297, Berkeley, December 1965. University of California Press.

[33] Gabriel Valdés-Alonzo, Christophe Binetruy, Benedikt Eck, Alberto García-González, and Adrien Leygue. Phase distribution and properties identification of heterogeneous materials: A data-driven approach. *Computer Methods in Applied Mechanics and Engineering*, 390:114354, February 2022.

[34] Jolie Frketic, Tarik Dickens, and Subramanian Ramakrishnan. Automated manufacturing and processing of fiber-reinforced polymer (FRP) composites: An additive review of contemporary and modern techniques for advanced materials manufacturing. *Additive Manufacturing*, 14:69–86, 2017.

[35] Jessy Simon, Nahiene Hamila, Christophe Binetruy, and Sébastien Comas-Cardona. A First Step Towards the Numerical Simulation of the Forming of flat TFP Preforms. *Procedia Manufacturing*, 47:126–128, 2020.

[36] Xin Wang, Man Jiang, Zuowan Zhou, Jihua Gou, and David Hui. 3D printing of polymer matrix composites: A review and prospective. *Composites Part B: Engineering*, 110:442–458, 2017.

[37] Laurent Stainier, Adrien Leygue, and Michael Ortiz. Model-free data-driven methods in mechanics: Material data identification and solvers. *Computational Mechanics*, 64(2):381–393, 2019.

[38] Yoshihiro Kanno. Mixed-integer programming formulation of a data-driven solver in computational elasticity. *Optimization Letters*, 13(7):1505–1514, 2019.

[39] Michael Greenacre. *Correspondence Analysis in Practice*. Chapman and Hall/CRC, Boca Raton, third edition, 2016.

[40] Ruben P. König. Changing social categories in a changing society: Studying trends with correspondence analysis. *Quality & Quantity*, 44(3):409–425, 2010.

[41] Katarina Zabret and Mojca Šraj. Evaluating the Influence of Rain Event Characteristics on Rainfall Interception by Urban Trees Using Multiple Correspondence Analysis. *Water*, 11(12), 2019.

[42] Michael Greenacre and Trevor Hastie. The Geometric Interpretation of Correspondence Analysis. *Journal of the American Statistical Association*, 82(398):437–447, 1987.

[43] Michael J. Greenacre. Clustering the rows and columns of a contingency table. *Journal of Classification*, 5(1):39–51, March 1988.

[44] David Arthur and Sergei Vassilvitskii. K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, New Orleans, January 2007. Society for Industrial and Applied Mathematics.

[45] Howard A. Barnes, J. F. Hutton, and Kenneth Walters. *An Introduction to Rheology*. Elsevier, Amsterdam, 1989.

[46] Severino P. C. Marques and Guillermo J. Creus. *Computational Viscoelasticity*. SpringerBriefs in Applied Sciences and Technology. Springer, Heidelberg, 2012.

[47] David Roylance. *Engineering Viscoelasticity*. MIT, Cambridge, October 2001.

[48] D. Y. Song and Ti Qian Jiang. Study on the constitutive equation with fractional derivative for the viscoelastic fluids - Modified Jeffreys model and its application. *Rheologica Acta*, 37(5):512–517, November 1998.

[49] R. M. Christensen. *Theory of Viscoelasticity: An Introduction*. Academic Press, New York, 2nd ed edition, 1982.

[50] O.C. Zienkiewicz, R.L. Taylor, and D.D. Fox. *The Finite Element Method for Solid and Structural Mechanics*. Elsevier, Oxford, seventh edition, 2014.

[51] R. B. Marimont and M. B. Shapiro. Nearest Neighbour Searches and the Curse of Dimensionality. *IMA Journal of Applied Mathematics*, 24(1):59–70, 1979.

[52] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data. *Journal of Machine Learning Research*, 11:2487–2531, 2010.

[53] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, New York, first edition, 2004.

[54] Hervé Abdi and Lynne J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459, 2010.

[55] Peter Läuchli. Jordan-Elimination und Ausgleichung nach kleinsten Quadraten. *Numerische Mathematik*, 3(1):226–240, December 1961.

[56] Yuichi Mori, Masahiro Kuroda, and Naomichi Makino. *Nonlinear Principal Component Analysis and Its Applications*. SpringerBriefs in Statistics. Springer Singapore, Singapore, 2016.

[57] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge university press, New York, first edition, 2004.

[58] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks — ICANN'97*, pages 583–588, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[59] Moo Chung. The Gaussian Kernel, 2007.

[60] S. Ali and K.A. Smith. Automatic parameter selection for polynomial kernel. In *Proceedings Fifth IEEE Workshop on Mobile Computing Systems and Applications*, pages 243–249, October 2003.

[61] Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, pages 682–688. MIT Press, 2001.

[62] E. J. Nyström. über Die Praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben. *Acta Mathematica*, 54(none):185–204, 1930.

[63] Kai Zhang and James T. Kwok. Clustered Nyström Method for Large Scale Manifold Learning and Dimension Reduction. *IEEE Transactions on Neural Networks*, 21(10):1576–1587, October 2010.

[64] Alberto García-González, Antonio Huerta, Sergio Zlotnik, and Pedro Díez. A kernel Principal Component Analysis (kPCA) digest with a new backward mapping (pre-image reconstruction) strategy. *arXiv:2001.01958 [cs, math]*, January 2021.

[65] Vijay K. Stokes. Random glass mat reinforced thermoplastic composites. Part I: Phenomenology of tensile modulus variations. *Polymer Composites*, 11(1):32–44, February 1990.

[66] N.R.L. Pearce, F.J. Guild, and J. Summerscales. An investigation into the effects of fabric architecture on the processing and properties of fibre reinforced composites produced by resin transfer moulding. *Composites Part A: Applied Science and Manufacturing*, 29(1-2):19–27, January 1998.

[67] Satoru Yoneyama. Basic principle of digital image correlation for in-plane displacement and strain measurement. *Advanced Composite Materials*, 25(2):105–123, March 2016.

[68] T. C. Chu, W. F. Ranson, and M. A. Sutton. Applications of digital image correlation techniques to experimental mechanics. *Experimental Mechanics*, 25(3):232–244, September 1985.

[69] Hervé Abdi. Singular Value Decomposition (SVD) and Generalized Singular Value Decomposition (GSVD). *Encyclopedia of measurements and statistics*, pages 907–912, 2007.

[70] Michael J. Greenacre. Interpreting multiple correspondence analysis. *Applied Stochastic Models and Data Analysis*, 7(2):195–210, June 1991.

Titre : Identification des propriétés des matériaux et de la distribution des phases des matériaux hétérogènes par des méthodes data-driven : Vers un espace constitutif amélioré

Mots clés : Méthodes data-driven, Corrélation d'images numériques, Analyse de correspondances, Analyse en composantes principales, Composites

Résumé : L'identification des relations constitutives des matériaux est une tâche essentielle pour comprendre leur comportement. Les méthodes classiques sont efficaces pour comprendre ces relations, mais l'introduction de modèles peut conduire à des formulations biaisées. En plus, il n'est pas possible de formaliser toutes les relations constitutives par des expressions mathématiques ou il peut y avoir des paramètres difficilement identifiables par des techniques courantes.
L'identification pilotée par les données (DDI), développée par Leygue et al. (2018), est un algorithme dans lequel la relation constitutive des matériaux élastiques est définie par une base de données de points matériels qui sont calculés en fonction des champs de déformation mesurés, des forces appliquées et de la géométrie connue des échantillons du matériau. L'algorithme estime simultanément les champs de contraintes associés aux déformations mesurées dans les échantillons.
Dans cette thèse, nous étendons l'algorithme DDI pour couvrir des comportements de matériaux plus complexes. Dans un premier temps, la méthode est appliquée à des échantillons hétérogènes, où un post-traitement est effectué avec l'analyse des correspondances pour séparer les différentes phases de l'échantillon et identifier leur comportement individuel. Ensuite, la DDI a également été appliquée à des matériaux viscoélastiques linéaires, où une approche étendue de l'espace de phase est utilisée pour tenir compte de la dépendance temporelle du comportement. Enfin, différentes variantes de l'algorithme sont envisagées en combinant la DDI avec différentes techniques statistiques telles que l'analyse en composantes principales, dans une recherche de rapidité et de précision des prédictions par réduction de la dimensionnalité. Parallèlement, la méthode est testée sur des échantillons composites hétérogènes et comparée aux résultats obtenus par les méthodes classiques.

Title: Identification of material properties and phase distribution of heterogeneous materials through data-driven computational methods: Towards an enhanced constitutive space

Keywords: Data-driven methods, Digital image correlation, Correspondence Analysis, Principal Component Analysis, Composites

Abstract: Identifying the constitutive relations of materials is an essential task to understand their behavior. Classical methods like testing can be effective in understanding these relationships, but introducing models can lead to biased formulations and errors. Furthermore, not all constitutive relations can be determined directly by mathematical expressions or there might be parameters that we cannot obtain easily through common techniques.
Data-Driven Identification (DDI), developed by Leygue et al. (2018), is an algorithm in which the constitutive relation of elastic materials is defined by a database of material points that need to be computed based on measured strain fields, applied forces and known geometry of tested samples of the material. The algorithm simultaneously estimates the corresponding values of the stress fields that emerge due to the deformations measured in the samples.
In this thesis, we focus on departing from elasticity to cover more complex material behaviors with the DDI algorithm. In a first step, the method is applied to heterogeneous samples, where a post-process is performed with Correspondence Analysis to separate the different phases in a sample and identify their separated behavior. Then, DDI was also applied to linear viscoelastic materials, where an extended phase-space approach is used to account for the time dependence of the behavior. Finally, different variations of the algorithm are considered by combining DDI with different statistical techniques such as the Principal Component Analysis, in a search for speed and accuracy of the predictions through dimensionality reduction. Parallel to this, the method is tested in heterogeneous composite samples and compared to expected results obtained by classical methods.