Universidad de Cantabria

Facultad
de
Ciencias

Modelización de histéresis en materiales viscoelásticos

(Hysteresis modelling in visco-elastic materials)

Trabajo de Fin de Grado
para acceder al

GRADO EN FÍSICA

Author: Mar González Alonso
Director: Diego Ferreño Blanco
Co-director: Álvaro Rodríguez Luis
13 de noviembre de 2024

A mis padres, por mantener la ilusión por la vida y fomentar mi curiosidad y creatividad.
A Ale, por ser el empujón que tantas veces he necesitado.

Quiero agradecer a los amigos que he ido haciendo estos años en la universidad la autenticidad de vuestro ser.
Gracias, Nadir, por haber compartido estos años conmigo y hacer que los días tuvieran más luz.

Fanny, María, Emma, Nico, Joakim, Mart, Johannes, gracias por haber sido una segunda familia. Haber vivido en Noruega con vosotros avivó el interés que siempre he tenido por las energías renovables y la tecnología offshore. Este trabajo no habría tenido lugar sin vosotros.

Las últimas líneas les pertenecen a mis compañeros en el IHCantabria. Gracias por haberme acogido con tanta calidez, y en especial a Álvaro por haberme guiado durante tantos meses.

Abstract

This study focuses on the hysteresis phenomenon in viscoelastic materials, specifically focusing on the energy dissipation that occurs during loading and unloading cycles. A non-linear constitutive model incorporating a Boltzmann integral term is applied to a Dyneema chain, a high-performance synthetic polymer, to accurately represent its mechanical behavior without permanent damage. Experimental tests are specifically designed to obtain the experimental data at LADICIM, which is then used for optimizing the model's parameters. The results show that the model successfully predicts energy dissipation through hysteresis and highlights the strain rate's impact on mechanical response.

KEYWORDS: hysteresis, viscoelastic materials, non-linear constitutive model, Dyneema chain

Resumen

Este estudio trata el fenómeno de la histéresis en materiales viscoelásticos, enfocándose específicamente en la disipación de energía que ocurre durante los ciclos de carga y descarga. Se aplica un modelo constitutivo no lineal que incorpora un término integral de Boltzmann a una cadena de Dyneema, un polímero sintético de alto rendimiento, para representar con precisión su comportamiento mecánico sin causar daño permanente. Los ensayos experimentales son diseñados específicamente para obtener los datos experimentales en LADICIM, los cuales se utilizan posteriormente para optimizar los parámetros del modelo. Los resultados muestran que el modelo predice con éxito la disipación de energía a través de la histéresis y resalta el impacto de la tasa de deformación en la respuesta mecánica.

Palabras clave: histéresis, materiales viscoelásticos, modelo constitutivo no lineal, cadena de Dyneema

Contents

1 Introduction

The mechanical response of solid materials to external forces varies significantly depending on their composition and structure. No universal physical model exists that can describe the behavior of all materials under these conditions, as each material responds differently based on its internal characteristics and the type of load applied.

1.1 Motivation

Polymers play a crucial role in a variety of industries, especially in applications where maintaining structural integrity under mechanical loads is essential. One such application is in the offshore industry, where polymers are commonly used in the fabrication of mooring lines that anchor floating structures such as ships, oil platforms, and renewable energy platforms (e.g., wind and solar). In these contexts, polymers are ensured to not sustain permanent damages when receiving waves and currents forces, since it could lead to cracks and failures.

Beyond offshore applications, polymers are also used in underwater pipelines, protective coatings, and cable insulation, where their ability to withstand mechanical stresses without permanent damage is key to ensuring long-term reliability, safety, and efficiency.

Therefore, understanding and predicting their mechanical behavior under controlled conditions is of high importance, and that is why laboratory tests are carried out carefully. These tests typically consist on slow loading and unloading forces to minimize the risk of permanent damage and overheating, simulating the real-world conditions.

1.2 Objectives

The main objective of this study is validating an analytical model to accurately describe the viscoelastic behavior of a specific polymer, a Dyneema chain, under uniaxial tensile loading and unloading, for the future use of it in mooring lines, as well as designing the tests to characterize this behavior. Experimental test conditions will involve no permanent damage as well as no overheating effects since in mooring lines, water acts as a cooling agent and waves have low frequencies. With controlled loading and unloading cycles, the material´s hysteresis behavior, i.e. energy dissipation, can be analyzed.

Calibration of the parameters in the model was conducted through optimization methods. These parameters are dependent on the test´s conditions. Afterwards, validation was carried out in order to characterize the material for these specific conditions.

1.3 Document organization

To introduce the reader to the concept of hysteresis and its implications, Section 2 gives an explanation of the basic concepts (covered in the books [1] and [2]) needed to understand viscoelasticty in polymers and important phenomena related to it. Afterwards, analytical models are introduced with the aim of showing how each model describes different phenomena. In Section 3 the methodology is described, which involves the explanation of the chosen analytical model, the experimental set up, the carried out tests, as well as how

calibration and validation are carried out. Section 4 shows the experimental results and its analysis. The conclusion is found in Section 5, as well as a suggestion of future work to go further into this study. Finally, the Python code is shown in the Appendix.

2 Solid body mechanics review

The application of a force to an object is known as loading. Materials can be subjected to many different loading scenarios and its performance is dependent on the loading conditions. There are five fundamental loading conditions for uniaxial components: tension, compression, bending, shear, and torsion, shown in Fig. 1.

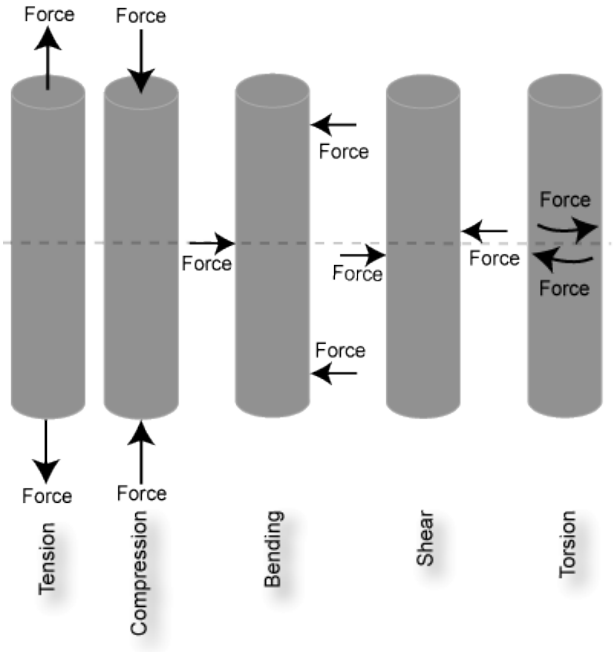


Figure 1: Different loading conditions for uniaxial components. Image from [3].

2.1 Stress

The term stress, σ, is used to express the loading in terms of force applied to a certain cross-sectional area of an object, usually measured in Pascals (Pa).

- From the perspective of loading, stress is the applied force or system of forces that produces deformation in bodies.
- From the perspective of what is happening within a material, stress is the internal distribution of forces within a body that balance and react to the loads applied to it. The stress distribution may or may not be uniform, depending on the nature of the loading condition. For example, a bar loaded in pure tension will essentially have a uniform tensile stress distribution. However, a bar loaded in bending will have a stress distribution that changes with distance perpendicular to the normal axis.

A simple example is shown in Fig. 2, where a bar is loaded axially. The stress is simply equal to the applied force divided by the bar's cross-sectional area.

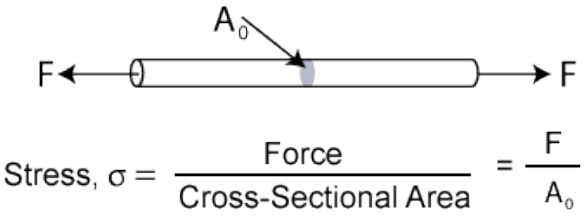


Figure 2: Simple stress. Image from [3].

2.2 Multidimensional complexity

It must be noted that the stresses in most 2-D or 3-D solids are actually more complex and need be defined more methodically.

The internal force acting on a small area of a plane can be described by three components: one normal to the plane and two parallel to the plane. The normal force component divided by the area gives the normal stress (σ), and parallel force components divided by the area give the shear stress (τ). To fully describe the state of stress at a particular point in the material, stresses acting on multiple planes that intersect at that point need to be considered. The stresses on any plane can be computed from the stresses on three orthogonal planes passing through the point. Due to the requirement of moment equilibrium, meaning the internal forces must balance without generating any net torque at any point, the final symmetric stress tensor has six stress components, which completely describe the state of stress at the point, shown in Fig. 3.

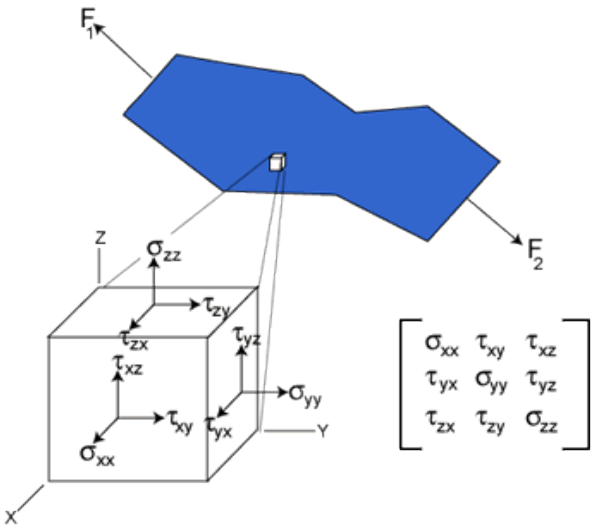


Figure 3: Complex stress. Image from [3].

Simplifying assumptions are often used to represent stress as a vector quantity for many engineering calculations and for material property determination. In this study, stresses

will be restricted to the uniaxial tension case, where there is only normal stress, and it only has one component, σ.

2.3 Strain

The following distinction is done:

- Deformation: measure of how much and object is stretched, given in units of length.
- Strain: ratio between the deformation in the direction of the applied force and the initial length of the material. This results in a unitless number, although it is often left in the unsimplified form, such as meters per meter.

Consider the example of the axially loaded bar in Fig. 4: A material sample with an initial length of L_0 is stretched in tension, that is, increases its length by a value of ΔL to the length $L_1 = L_0 + \Delta L$. Strain is the quantity ε:

$$\varepsilon = \frac{\Delta L}{L_0} = \frac{L_1 - L_0}{L_0} \tag{2.1}$$

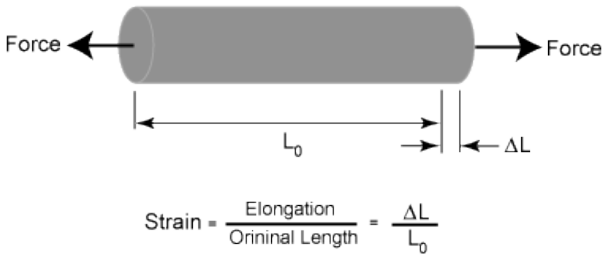


Figure 4: Strain in a bar that is being stretched in tension. Image from [3].

2.4 Mechanical properties

Mechanical properties are important in the study of deformation processes, since they predefine the characteristics of the stress-strain state of a body under certain boundary conditions. The general mathematical model for describing solid mechanical behavior contains physical equations between the stress-strain state components.

The material mechanical properties can be divided into several groups [1]:

- Strength properties: characterize the ultimate material resistance to various types of loads, such as tensile, compressive, and shear loads. Materials with high strength can withstand large forces before failing.
- Elasticity: the property of a material by which it recovers its original shape and size when the deforming force is removed. The boundary between elastic and plastic deformation is marked by the yield point. For values beyond this limit, the material undergoes permanent changes. Ductility is a property which refers to a material's

ability to undergo significant plastic deformation before rupture or failure. In simple terms, it's the material's capacity to stretch or bend without breaking. They can absorb considerable energy through deformation, making them less likely to fracture under tensile stress.

- Dynamic properties: characterize the conditions of elastic vibrations, the transmission and propagation of strain waves and stress waves in materials.
- Rheological properties: characterize how materials deform over time under constant load or varying loading conditions.

2.5 Viscoelastic behavior: creep, creep-recovery, relaxation and Mullin´s effect

Some materials exhibit a time-dependent elastic behavior, meaning that their response to loading changes over time. Even though strain is recoverable in the elastic regime, viscoelastic materials behave differently from purely elastic ones. In particular, the stress-strain relationship for loading and unloading is not identical, leading to a phenomenon called hysteresis, which is characteristic of viscoelasticity. This behavior will be described in more detail in Section 2.8, where Fig. 8 illustrates the different paths taken during loading and unloading cycles.

Viscoelastic materials can present several key phenomena that depend on time, which include the following [4]:

- Creep: the progressive increase in deformation under a constant load, at a fixed temperature and humidity, over time.
- Creep-recovery: the reduction in deformation after the removal of a constant load, typically following an exponential decay over time.
- Relaxation: the gradual decrease in stress when a material is held at a constant strain, with fixed temperature and humidity conditions, over time.

The stress-strain relationship for each of these phenomena are described in Fig. 5.

- Mullin´s effect: consists on the substantial stress softening when subjected to stretching at a constant deformation rate and occurs in a variety of materials such as polymers, rubbers and living tissues. As shown in Fig. 6, instead of following the monotonous uniaxial stretch curve when loading, since some fibers are partially broken or molecular chains are disentangled, the force required during the loading and unloading every next cycle is considerably lower. After some load cycles, the material response becomes repeatable (the number of cycles depends on the material, but it is usually after 3-5).

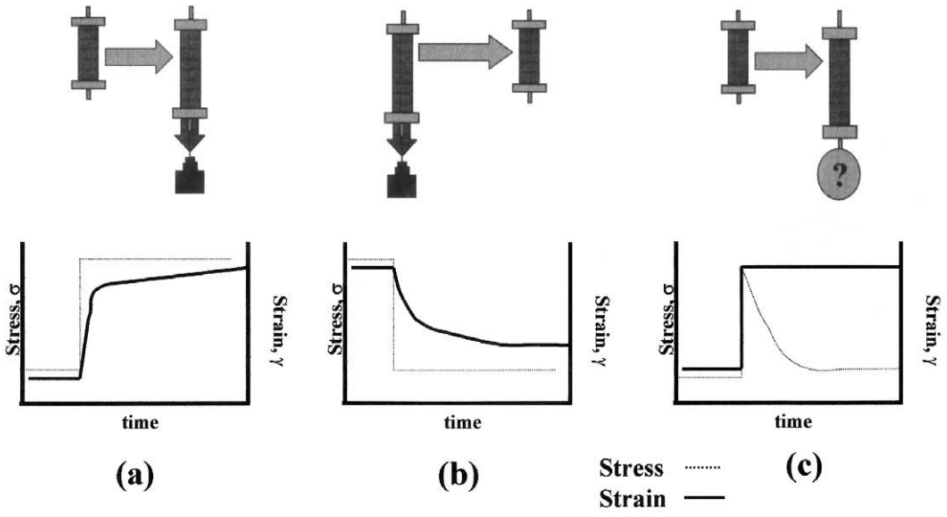


Figure 5: (a) Creep test: a load or stress is applied to a sample. (b) Recovery test: when the stress is removed and the material is allowed to recover. These two tests are often cycled. (c) Stress relaxation test: the reverse of creep. Holding a sample at a fixed length, the change in stress as a function of time or temperature is measured. Image from [5].

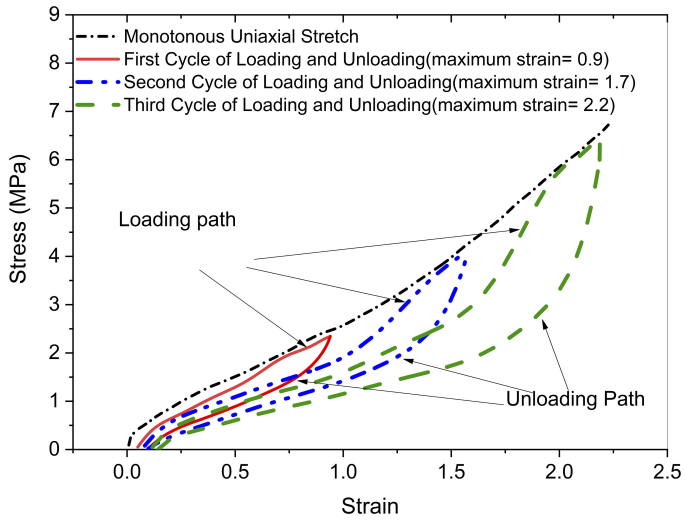


Figure 6: Schematic of the Mullin´s effect. Image from [6].

2.6 Analytical models

The way materials respond to loads can be described using simple models that represent different types of materials. Each model captures a specific behavior, such as elasticity, plasticity, or viscosity.

2.6.1 Hooke's Law

Materials exhibit elastic properties when external forces induce changes in shape without resulting in breaks or voids within the solid material. They resemble springs governed by Hooke's deformation law (Eq. 2.2), where σ is the stress, E is modulus of elasticity, and ε is the strain.

$$\sigma = E\varepsilon \tag{2.2}$$

2.6.2 Newton's Law of linear viscous deformation

Viscosity can be defined as the resistance of a fluid (liquid or gas) to a change in shape. The simplest model to describe linear viscous behaviour is the Newton's law of linear viscous deformation:

$$\sigma = \eta\frac{d\varepsilon}{dt} \quad or \quad \varepsilon(t) = \varepsilon(0) + \int_0^t \frac{\sigma(\tau)}{\eta}d\tau \tag{2.3}$$

where η is the viscosity coefficient, and $\frac{d\varepsilon}{dt}$ is the strain rate. This law shows that the faster the deformation, the greater the stress.

2.6.3 Maxwell model

The simplest model to describe linear visco-elastic behavior of solids is the Maxwell model, which illustrates the stress relaxation phenomenon. It can be represented and described by a purely viscous damper and a purely elastic spring connected in series (Fig. 7) and was originally proposed by Maxwell in 1867 to describe the dynamic behavior of gas [7]. The relationship between stress and strain in this model is:

$$\dot{\varepsilon} = \frac{\dot{\sigma}}{E} + \frac{\sigma}{\eta} \tag{2.4}$$

Alternatively, this constitutive equation can be written as:

$$\sigma(t) = E\varepsilon(t) + [\sigma(0) - E\varepsilon(0)]e^{-\frac{E}{\eta}t} - \frac{E^2}{\mu}\int_0^t e^{-\frac{E}{\eta}(t-\tau)}\varepsilon(\tau)d\tau \tag{2.5}$$

The first term in Eq. 2.5 is the elastic component of the stresses. The second term expresses the influence of initial values, which decreases over time according to the exponential function. The third term is the convolution integral, which shows the contribution of strains: more remote events have less impact at the current time than more recent events (fading memory). This model predicts that stress will decay exponentially with time in the material subjected to constant strain. However, a major limitation is that it does not predict the creep behavior present in many materials very reliably, since in this case it predicts a

linear increase in strain with time if the stress is constant, but many viscoelastic materials show a decreasing strain rate with time.

The main applications of this model are the modeling of thermoplastic polymers near their melting temperature, of fresh concrete, and of many metals near their melting point.

2.6.4 Kelvin-Voigt model

The simplest linear elastic-lag model is the Kelvin-Voigt model, which clearly explains the phenomenon of creep, but does not predict stress relaxation. It modifies the linear spring model by introducing a purely viscous damper connected in parallel (Fig. 7). The modification is done to overcome the disadvantages of the linear spring model that do not consider energy dissipation.

The parallel arrangement of the spring and dashpot causes this model to describe the creep phenomenon as there is no possibility for the spring and dashpot to expand independently. After releasing the stress, the spring would not be able to contract its length back to its original position immediately. Instead, the spring will apply a force on the dashpot and thus compressive creep under external stress will occur. After a finite time, all creep strain will be recovered. This phenomenon is called 'viscoelastic contraction' which is significant in real viscoelastic materials. [8]

The constitutive equation can be written as:

$$\sigma = E\varepsilon + \eta\dot{\varepsilon} \quad \text{or} \quad \varepsilon(t) = \varepsilon(0)e^{-\frac{E}{\eta}t} + \frac{1}{\eta}\int_0^t \sigma(\tau)e^{-\frac{E}{\eta}(t-\tau)}d\tau \tag{2.6}$$

Thus, for the Kelvin-Voigt model, the initial strain distribution fades exponentially over time, and stresses are introduced as an integral convolution with the exponential core.

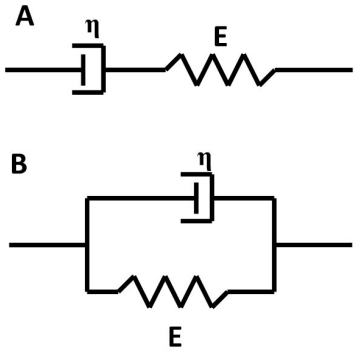


Figure 7: Spring-dashpot models for viscoelasticity. (A) The Maxwell model places the purely viscous dashpot and the purely elastic spring in series with one another. (B) The Kelvin–Voigt model consists of a purely viscous dashpot and a purely elastic spring in parallel. Image from [9].

2.7 Boltzmann integral models

One can generalize the basic Kelvin-Voigt and Maxwell models to relate the stress and strain by:

$$\varepsilon(t) = \sigma_0 J(t)H(t) \text{ where } \sigma = \sigma_0 H(t) \text{ and } \sigma(t) = \varepsilon_0 Y(t)H(t) \text{ where } \varepsilon = \varepsilon_0 H(t) \quad (2.7)$$

for functions $J(t)$ and $Y(t)$ the creep compliance function and the relaxation modulus function, respectively. The Creep compliance function is defined as the strain variations under the application of constant stress. The relaxation modulus represents the stress required to produce and maintain unit strain.
H is the Heaviside function.

$$H(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases} \quad (2.8)$$

Boltzmann generalized the above models to account for variables $\sigma(t)$ and $\varepsilon(t)$ (e.g. the results from stretch and relaxation tests) by considering a succession of infinitesimal steps $d\varepsilon(t)$ and $d\sigma(t)$ in equation 2.7, obtaining

$$d\sigma(t) = d\varepsilon(\tau)Y(t-\tau)H(t-\tau) \quad (2.9)$$

One can rewrite $d\varepsilon(\tau) = \frac{d\varepsilon(\tau)}{d\tau}d\tau$, set $t \geq \tau$ and integrate from $-\infty$ to t, obtaining

$$\sigma(t) = \int_{-\infty}^{t} Y(t-\tau)\frac{d\varepsilon(\tau)}{d\tau}d\tau \quad (2.10)$$

and similarly for ε in terms of the kernel J.
The above integral is referred to as a Boltzmann integral since it conforms with the fundamentals of superposition as enunciated by Boltzmann. [1]

The Boltzmann superposition model, also called the Maxwell solid in history integral form, is one of the most often used to account for hysteresis, based on the hypothesis that the stress is strain rate dependent through a convolution relationship:

$$\sigma(t) = E_0\varepsilon(t) + \int_{-\infty}^{t} k(t-\tau)\frac{d\varepsilon(\tau)}{d\tau}d\tau \quad (2.11)$$

where E_0 represents an instantaneous modulus of elasticity and k is the relaxation response kernel.

All viscoelastic models of this type have the following characteristic features:

- They are linear.
- Their behavior is temporarily dependent, i.e. viscous (rheonomic).
- They describe behavior with decaying memory. This means that if an event occurred long enough ago, then its effect in the present is negligible.
- With very fast or very slow processes, these models behave elastically as long as all dampers become rigid or pliable, respectively.

[1] For each structural unit, the Boltzmann superposition principle can be formulated as follows: if the effects of deformation processes $\varepsilon_i(t)$, $i = 1, 2$ are stresses $\sigma_i(t)$, $i = 1, 2$, then the superposition $\varepsilon_1(t)+\varepsilon_2(t)$ leads to stresses $\sigma_1(t) + \sigma_2(t)$.

2.8 Energy dissipation in deforming materials: hysteresis

In viscoelastic materials, the response to stress causes internal friction (force-resisting motion between the elements making up the solid material while it undergoes deformation) and molecular rearrangements, leading to the so called hysteresis. Hysteresis causes energy dissipation during cyclic loading, leading to heat buildup and potential material degradation.

Plastic deformation is another important material dissipation mechanism, particularly in ductile materials. When subjected to cyclic loading and unloading, these materials undergo permanent deformation, resulting in energy dissipation. The energy required for plastic deformation is converted into heat, leading to hysteresis loops and energy loss. Over time, the accumulation of plastic strain can lead to the initiation and propagation of cracks, ultimately causing material failure.

One of the fundamental techniques for measuring hysteresis in materials is strain-stress analysis. A typical hysteresis curve is shown in Fig. 8, where the energy lost during one loading-unloading cycle is given by the area within the loop. The shape of the loop depends on the rates of loading and unloading (unlike normal time-independent elasticity).

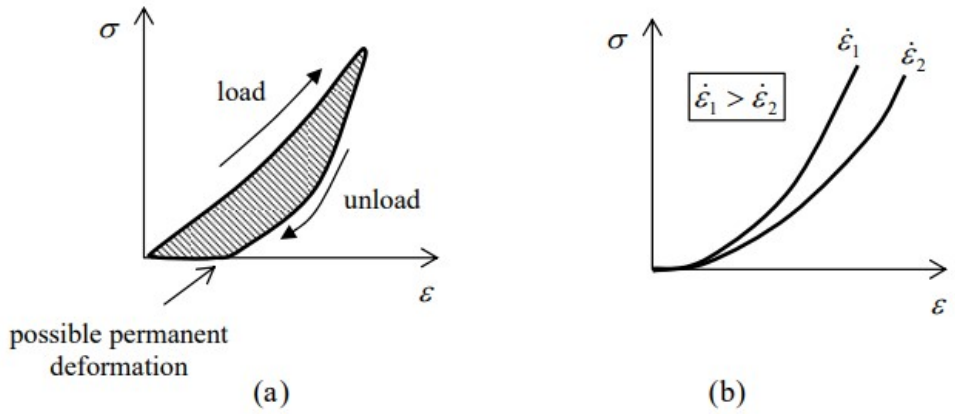


Figure 8: A schematic stress-strain curve of a viscoelastic material, showing (a) hysteresis: dissipation of energy, and (b) response to different stretching rates. Image from [10].

2.9 Factors that influence hysteresis

When dealing with hysteresis in materials, it is essential to consider the factors that influence its magnitude. The properties of the material itself, such as its microstructure and composition, greatly affect the extent of hysteresis. Additionally, external factors like temperature, loading rate, and applied stress also play a significant role.

2.10 Viscoelasticity in polymers

Polymers show both elastic and viscous behaviors due to their molecular nature. Differently from other chemical and metallic materials, polymers are long-chain molecules that interact through entanglement points. These entanglements, along with intermolecular forces, keep the polymer chains in a stable equilibrium state when no external force is applied.

When a force is applied to a polymer, some chains are forced to align in the direction of the deformation, while others resist the movement by trying to return to their equilibrium state. This resistance occurs at the entanglement points, where the chains are intertwined. During this process, bond stretching and chain straightening occur, which behave like stretching a spring. As a result, when the force is removed, the polymer chains tend to return to their original shape, which is the elastic component of the behavior.
However, viscous behavior is also present, which is primarily due to the slippage of chains relative to one another at the entanglement points. Under sustained or prolonged stress, the chains begin to move past each other, causing friction. The sliding motion, known as chain slippage, occurs without breaking the chains and leads to a time-dependent flow, characteristic of viscous materials.

This combination of recoverable elastic deformation and time-dependent viscous flow is what gives polymers their viscoelasticity. Their molecular structure, with entanglements and mobility, allows both types of deformation to occur: elastic recovery in the short term and permanent flow over longer periods. In Fig. 9, although hysteresis is shown in (3), the near-crack dissipation mechanism is also shown, for when external forces result in the breakage of chains.

2.11 Strain rate hardening in polymers

Polymers generally present higher hysteresis when subjected to rapid loading and/or unloading (faster pull and/or release in the polymer´s chains) compared to slow, steady-state loading. Theirs response to stress is highly affected by both the duration and speed of the loading process.

Many materials, especially polymers and metals, exhibit strain rate hardening, meaning they become stiffer at higher strain rates. This is because, at higher strain rates, there is less time for molecular movement (in polymers) or dislocation motion (in metals), so the material resists deformation more strongly during loading and more stored energy is released during unloading. The higher the strain rate, the faster the material responds elastically before the viscous or plastic mechanisms activate fully, resulting in steeper hysteresis loops. [11]

At lower strain rates, the material has more time to recover and relax during the unloading phase, so the hysteresis loop is broader and less steep, while higher strain rates delay relaxation, making the response more elastic-like during both loading and unloading phases.

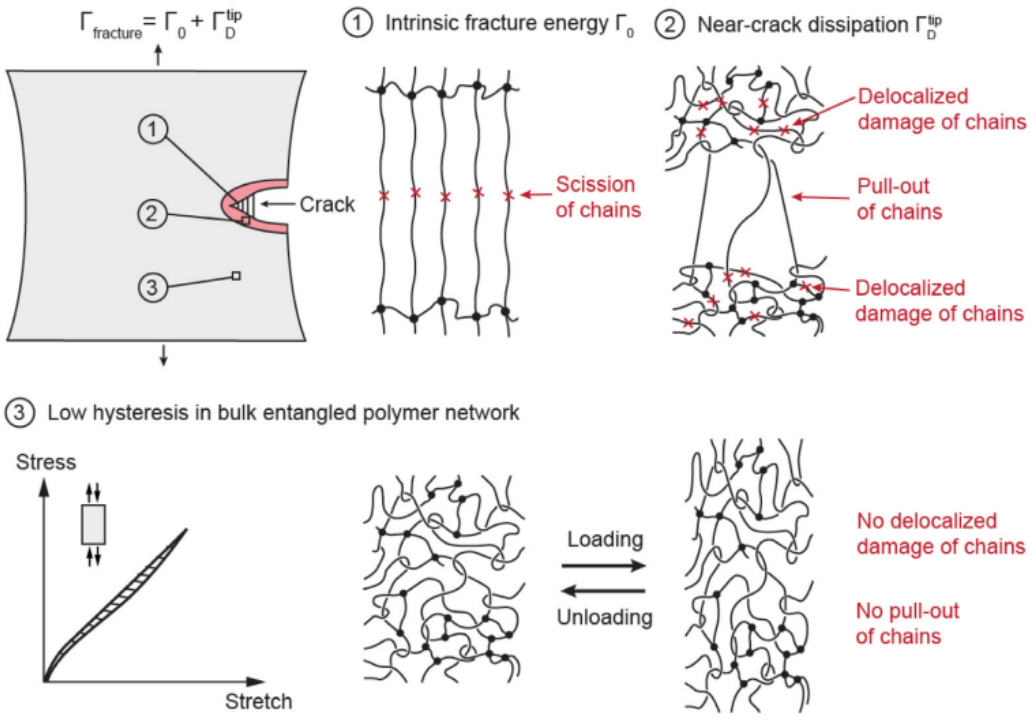


Figure 9: Schematic illustration of the near-crack dissipation mechanism for entangled polymer network. Pull-out and/or delocalized damage of chains around the crack tip can dissipate substantial energy, toughening the entangled polymer network. When an entangled polymer network is stretched before failure, the overall applied stretch may be less than the stretch in critical areas, such as the crack tip. This results in negligible localized damage and low stress–stretch hysteresis in the network as a whole. Image from [12].

For polymers and viscoelastic materials, high strain rates may cause them to behave more stiffly and elastically, but that can also mean that localized stresses are higher. If the material cannot distribute stress quickly enough, it could lead to localized failure points, like micro-cracking or brittle fractures, depending on the material's toughness and ductility.

3 Methodology

Due to the complex, non-linear, viscous behavior of the polymer under study, Dyneema, a specific model has been selected to represent its hysteresis behavior. It was ensured to be both mathematically and computationally suitable, meaning by this that the model doe not implementing complex mathematics and its computational cost is not too large.

In this section, the experiment´s design is also explained in detail, as well as the calibration, for which the required optimization is briefly discussed.

3.1 Analytical model

Since linear relationships do not describe properly the behavior shown in experiments on polymers, non-linear models are needed.

In the situation of a simple extension in a rod of length l and applied force $f(t)$ at the end $x = l$, Banks et al. [13] suggested using a Boltzmann law with a nonlinear strain functional to account for non linearities of stress and strain.

$$\sigma(t) = g_e(\varepsilon(t)) + \int_0^t Y(t-s)\frac{d}{ds}g_v(\varepsilon(s), \dot{\varepsilon}(s))ds \tag{3.1}$$

where Y is the memory kernel, and g_e and g_v are nonlinear functions accounting for the elastic and viscoelastic nonlinear responses respectively of an elastomer. This stress-strain law implies that the stress depends not only on the current strain but also on the history of the strain and the strain-rate.

Experimental observations indicate that elastomers posses different nonlinear viscoelastic responses in loading ($\dot{\varepsilon} > 0$) and unloading ($\dot{\varepsilon} < 0$), leading to the following choice for the viscoelastic response function

$$g_v(\varepsilon(s), \dot{\varepsilon}(s)) = \begin{cases} g_{vi}(\varepsilon(s)) & \text{if} \quad \dot{\varepsilon}(s) > 0 \\ g_{vd}(\varepsilon(s)) & \text{if} \quad \dot{\varepsilon}(s) < 0 \end{cases} \tag{3.2}$$

where g_{vi} and g_{vd} are required to be continuous functions.

These material´s memory significantly depends on a history of finite length. If we define the turning points t_i, $i = 0, ..., M$ as the points for which $\dot{\varepsilon} = 0$, and assume $t_k < t < t_{k+1}$, equation 3.1 leads to the final proposed equation in the article [13]:

$$\begin{aligned} \sigma(t) = g_e(\varepsilon(t)) &+ \int_0^t \dot{Y}(t-s)g_v(\varepsilon(s), \dot{\varepsilon}(s))ds \\ &+ Y(0)g_v(\varepsilon(t), \dot{\varepsilon}(t)) - Y(t)g_{vi}(\varepsilon(0)) \\ &+ \sum_{k=0}^{M} Y(t-t_k)(-1)^k[g_{vi}(\varepsilon(t_k)) - g_{vd}(\varepsilon(t_k))] \end{aligned} \tag{3.3}$$

They also proposed the following choices for elastomers:
• A third degree polynomial for the elastic component,

$$g_e(\varepsilon(t)) = \sum_{i=1}^{3} E_i \varepsilon^i(s) \tag{3.4}$$

• A third degree polynomial for the viscoelastic component,

$$g_{vi}(\varepsilon(s)) = \sum_{i=1}^{3} a_i \varepsilon^i(s) \tag{3.5}$$

$$g_{vd}(\varepsilon(s)) = \sum_{i=1}^{3} b_i \varepsilon^i(s) \tag{3.6}$$

• A decay of exponential type for the memory kernel,

$$Y(t) = e^{-Ct} \tag{3.7}$$

In the proposed model to describe hysteresis 3.3, 10 parameters, corresponding to the coefficients in Equations 3.4, 3.5, 3.6 and 3.7, define the material´s response to uniaxial tension, and need to be optimized. Explanation of the optimization method is given in the appendix in Section 3.3 Calibration and validation.

3.2 Experimental set up

Experiments have been carried out at LADICIM (Laboratorio de la División de Ciencia e Ingeniería de los Materiales) with an Instron 8803, which is a versatile servohydraulic fatigue testing system that performs static and dynamic tests on materials and components up to 500 kN (Fig. 10).

A Dyneema chain was selected as the material to study. Dyneema, also known as UHMWPE (ultra-high molecular weight polyethylene), is commonly used in the manufacture of ropes, slings, and tethers. The polyethylene used in Dyneema consists of extremely long molecular chains, which contribute to its high tensile strength: the maximum stress a material can withstand while being stretched or pulled before breaking. This material has proved to show similar behavior to elastomers, with low hysteresis, and due to its important applications, Banks' model (Eq. 3.3) is tested in this study.

Dyneema is notable for having the lowest elongation at break among synthetic fibers. While it is widely recognized for its high strength-to-weight ratio, its behavior in terms of toughness and ductility is distinct compared to traditional materials like metals or fibers such as nylon. Dyneema is extremely tough, primarily due to its molecular structure that resists crack propagation, making it ideal for applications such as body armor, ropes, and high-performance fabrics. [14] However, Dyneema is not very ductile, meaning it does not undergo significant plastic deformation before breaking. It behaves more like a brittle material, failing with minimal elongation once its strength limits are exceeded. This lack of ductility is typical of UHMWPE, which has a highly crystalline structure that adds strength and stiffness but limits its ability to stretch or bend without fracturing. Additionally, Dyneema rigging offers practical benefits for maritime use, as it is relatively easy to splice and rig. Its buoyancy (being lighter than water) and UV resistance are advantageous for applications on yachts. The primary drawback of Dyneema is its permanent elongation under high stress, known as creep. However, this issue is mitigated in the cases of interest for this study.

The theoretical free length of the Dyneema used, L_0, is 100 mm per link, giving a total length of $L_0 = 500$ mm, needed for the calculation of strain as in Eq. 2.1.
The theoretical cross-sectional area is 20×16 mm^2, and considering it is a chain, this value should be multiplied by 2, i.e. $S_0 = 640$ mm^2. However, the connection area between the links is difficult to analyze accurately. This value may not be very accurate, as the material consists of woven fibers, resulting in numerous gaps. Therefore, it is strongly more convenient to work with force (N) instead of stress (MPa).

Changing the geometry from a rope to a chain introduces additional complexity not present in Banks' experiments [15]. This change leads to increased friction at the junctions of the

chain links, which in turn generates heat. Managing this friction and the resultant heat generation presents a challenge for the model.

Figure 10: Experimental set up with the Dyneema chain sample secured at both ends, with a total of five links.

3.3 Experiment

When choosing the type of experiment to carry out, the Mullin´s effect needs to be avoided. This non-linear behaviour that shows an early damage is not described by Banks´ model. Therefore, several cycles must be performed before taking measurements.

Furthermore, temperature and humidity conditions are assumed constant in the laboratory.

In the first experimental series of tests, each test involved cycles of loading and unloading, and tests were spaced over time, with minimum-maximum applied stretching force varying between 2-20 kN, 3-30 kN, 4-40 kN, and 5-50 kN, readjusting the medium level between series of tests to 11, 16.5, 22 and 27.5 kN respectively. At the beginning of each test, the medium stretching force level is set to correspond to a fixed minimum of strain. For every series of tests, the force signal was changed from sinusoidal, to triangular, and lastly to

square, with frequencies of 10 Hz, then 1, 0.1 and 0.01 Hz. The time between tests was less than a few seconds.

When supervising the experiments, we noticed that the Dyneema chain had not been "demullinised"[2]. That is the reason why all the previous tests to demullinisation were not useful.

Afterwards, the second experimental series of tests took place, again involving cycles of loading and unloading spaced over time, with maximum-minimum applied stretching force varying between 0.5-50 kN, 0.5-40 kN, 0.5-30 kN, and 0.5-20 kN, readjusting the medium level between series of tests to 25.25, 20.25, 15.25 and 10.25 kN respectively. The force signal was sinusoidal, with frequencies of 0.01, 0.1, 1 and 10 Hz. Afterwards, with the same stretching forces, the signal used was triangular, with frequencies of 0.1 and 1 Hz.

Test	Force range (kN)	Frequency (Hz)	Force signal	Remark
1-3	2-20	1	Sin., T, Sq.	
4-6	2-20	0.1	Sin., T, Sq.	
7	50	?	?	Demullinitation. Channel´s exit not saved.
8-10	2-20	10	Sin., T, Sq.	
11-13	2-20	0.01	Sin., T, Sq.	
14-25	3-30	10, 1, 0.1, 0.01	For each frequency: Sin., T, Sq.	
26-37	4-40	10, 1, 0.1, 0.01	For each frequency: Sin., T, Sq.	
38-49	5-50	10, 1, 0.1, 0.01	For each frequency: Sin., T, Sq.	Stiffness has increased.
-				Rest
50	60	?	?	Damage. Channel´s exit not saved.
51-54	0.5-50	0.01, 0.1, 1, 10	Sin.	
55-58	0.5-40	0.01, 0.1, 1, 10	Sin.	
59-62	0.5-30	0.01, 0.1, 1, 10	Sin.	
63-66	0.5-20	0.01, 0.1, 1, 10	Sin.	
67-68	0.5-50	0.1, 1	T	
69-70	0.5-40	0.1, 1	T	
71-72	0.5-30	0.1, 1	T	
73-74	0.5-20	0.1, 1	T	

Table 1: Carried out tests. Sin. accounts for sinusoidal and Sq. for square.

The aim was to adjust a single cycle and several cycles to the model with optimization methods. Once all the optimized coefficients in the model are known in equation 3.3, the next step was to test its prediction capability for interpolation and extrapolation making the comparison with the other cycles.

The data collected are the time series, the stretching force signal, and the increment of length of the material. The error in the stretching force signal is of $\delta(F) = 1\%$, and in the increment length of the material is of $\delta(\Delta L) = 5\%$.

[2]Demullinisation is understood in this document as the cycles of loading and unloading that are given in order to make a prior damage to make the fiber adjustment in the material.

It is important to note that unlike Banks´ experiments [13], in the experiments carried out at LADICIM the strain rate was not constant. Since the frequency of the stress signal was set constant per test, this means that as the maximum stress achieved is set higher, the more rapid pull the machine needs to give in order to maintain the frequency. As no damages are occurring because of this, the model can be tested for the case of having a non constant strain rate.

Furthermore, the predictive capability of the model is tested for single spaced in time cycles, differently from what is found in Banks´experiments, where they tested the continuous and immediate time dependency of hysteresis.

3.4 Calibration and validation

Calibration was done by optimizing the ten parameters of the model (corresponding to the coefficients in equations 3.4, 3.5 and 3.6) for the chosen cycles, named in this document as training cycles. Afterwards, validation of the model was done by extrapolation and interpolation on the same type of test and same type of force signal. There is a need to have this specific criteria, and it will be better understood in the Section 4.2 Experimental results.

3.4.1 Optimization

In the proposed model to describe hysteresis by Banks et al., 3.3, optimization is needed to estimate the 10 parameters. Powell´s optimization method has given the most accurate results to describe hysteresis. It is a conjugate direction method, quadratically convergent and extremely efficient. The function being minimized need not be differentiable, and no derivatives are taken.

The chosen cost function to minimize is the RMSE (Root Mean Square Error), which heavily penalizes large errors and can be interpreted as an indication of the noise levels in the scale of standard deviations.

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{3.8}$$

where n is the total number of data points, $\hat{y}_i$ is the computed or predicted data and y_i is the experimental data . Eq. 3.8 has a linear gradient, and so fast calculations can be made for its minimization.
Initial values for the parameters are also given for local minimization methods such as Powell. Initial values are important for speed and convergence. For the elastic component, Eq. 3.4, only the linear term is estimated initially, being the slope of the cycle resulting from experimental data. The viscous terms a_1 and b_1 for loading and unloading, respectively, are set to be the ones that make hysteresis when only having the previous elastic term and for the memory kernel $C = 1$.

4 Results and analysis

4.1 Model verification

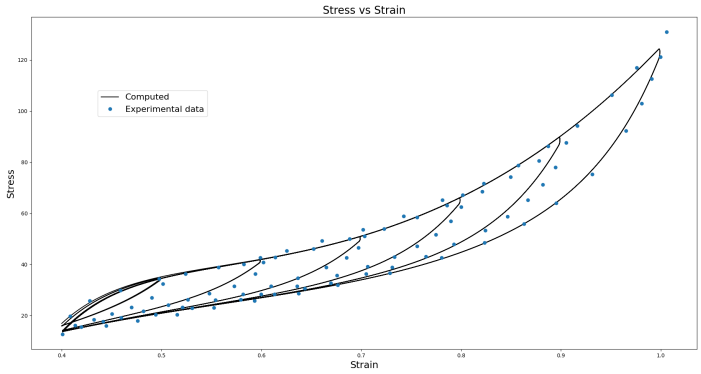
To verify that the analytical model proposed by Banks et al. [13] describes hysteresis, the first step was to implement the model in Python and replicate the results of the article with the given coefficients for Equations 3.4, 3.5, 3.6 and 3.7, shown in Table 2. The materials used are four different elastomer samples: a silica-filled silicone (Sil) material, medium-filled carbon black (CB-m) rubbers, and a highly-filled black (CB-h) rubber.

	Type I Data		Type II Data	
	Sil	CB-h	CB-h	CB-m
C_1	0.1870	0.1891	0.1501	0.3024
E_1	178.9901	282.6624	251.7778	285.3666
E_2	−479.8327	−194.4153	−192.4070	−191.1562
E_3	344.3367	121.0941	112.0081	119.4363
a_1	265.2555	705.9752	699.8914	250.8615
a_2	−143.9302	−678.9015	−578.2702	95.8064
a_3	528.3796	425.6897	469.6236	−58.1586
b_1	1403.7904	547.1567	514.1644	81.5614
b_2	−1873.0947	−853.8089	−879.8506	−170.1572
b_3	575.5831	462.1712	424.5226	117.7925

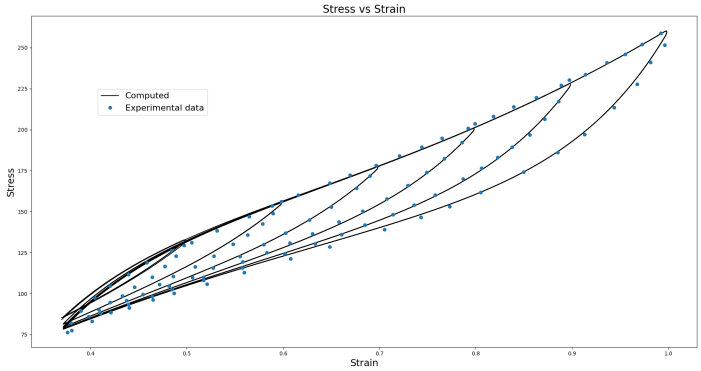
Table 2: Optimal parameters obtained in the article from inverse problems.

Two different type of tests are done. The Type I tests include a sequence of loading and unloading the sample to produce load displacement curves with decreasing maximum strain levels: three consecutive looks at 100% maximum strain, then the sequence of three loops each with maximum strain levels of 90% down to 50%. The Type II tests, the strain loops have decreasing maximum strain level as before, but instead of a fixed minimum strain level, it increases progressively: three consecutive strain loops with a minimum strain of 25% and a maximum of 100%, and continue with three consecutive loops of $30\% - 90\%$ strains, three loops of $40\% - 80\%$ strains, and finally three loops of $50\% - 70\%$ strains. At the beginning of each test, large displacements cycles are carried out to account for and remove any possible Mullin´s effects.

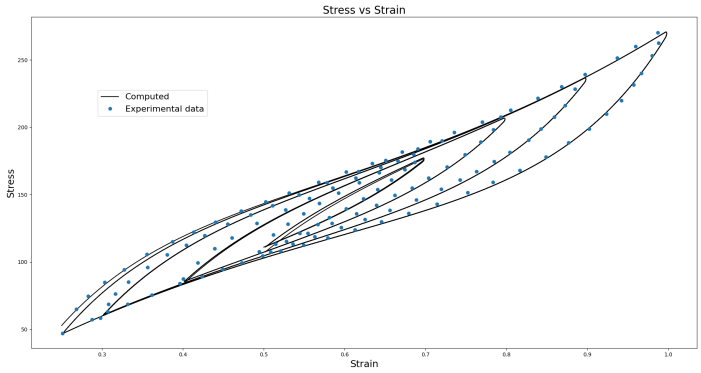
The length of the cylinders is not given in the article, and so calibration "by hand" was required to estimate the different lengths, which resulted in $l = 0.115$ m for the Type I CB-h, $l = 0.195$ m for the Type I Sil, $l = 0.11$ m for the Type II CB-h, and $l = 0.13$ m for the Type II CB-m. This value is important since it determines the strain rate $\dot{\varepsilon}(t)$, which is constant for these tests. $\dot{\varepsilon}(t) = \frac{0.127}{60} l \text{ ms}^{-1}$. The stress-strain experimental results where digitalised since the model´s prediction data points are not given in the article.



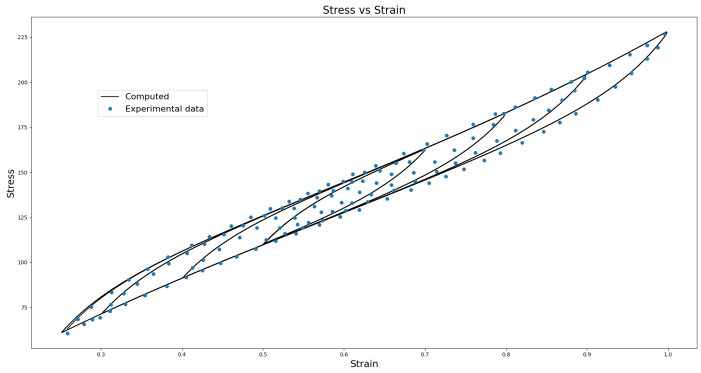
(a) Type I silica-filled silicone (Sil) data.



(b) Type I highly filled carbon (CB-h) data.



(c) Type II highly filled carbon (CB-h) data.



(d) Type II medium filled carbon (CB-m) data.

Figure 11: Stress-strain model's prediction for Type I and Type II tests.

Fig. 11 agrees with the comparisons made in Banks´ article [13], with relative errors of 4.3% for the Sil case 11a, 2.5% for the type I CBh case 11b, 1.5% for the type II CBh case 11c and 1.6% for the type II CBm case 11d .

4.2 Experimental results

4.2.1 Non constant strain rate analysis

First of all, the effect of a non constant strain rate can be analysed. For the first experimental series of tests, the sinusoidal force signal with ranges 3-30 kN has been selected for all its different frequencies.

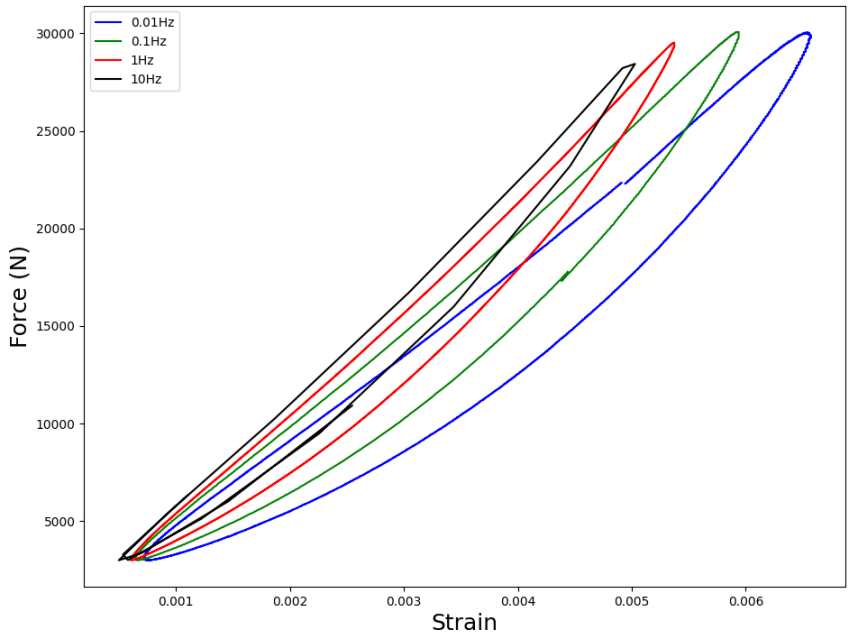


Figure 12: Hysteresis from one single cycle of the first series of tests with sinusoidal force signal with ranges 3-30 kN.

Experimental results are shown in Fig. 12, where the hardening phenomenon due to the increase of strain rate can be observed. The greater the force frequency, the greater the strain rate, and so hysteresis loops become steeper.

It can also be observed that as strain rate decreases (force frequency decreases), loops become broader because the material has more time to recover and relax during the unloading phase.

Variations of strain rate between tests are around an order of magnitude between one after the other. Bank's model has not provided satisfactory predictions for such large variations in strain rate. Using a damped sinusoidal force signal would offer a better case for study, rather than carrying out separate tests as done in this study. These improvements are left for future research.

Taking this into account, same force frequency tests can be analysed, which present small changes in strain rate.

4.2.2 First experimental series of tests: same frequency analysis

The first experimental series of tests exhibit two important issues: the square signal and the readjustment of the medium level between experiments.

The square signal is an abrupt signal that may cause a small damage in the material due to the rapid pull. In Fig. 13, it can be observed that, even though the minimum and maximum stretching forces remain the same, higher strains are achieved for the square signal. This occurs because the rapid pulling causes the material to behave more elastically.

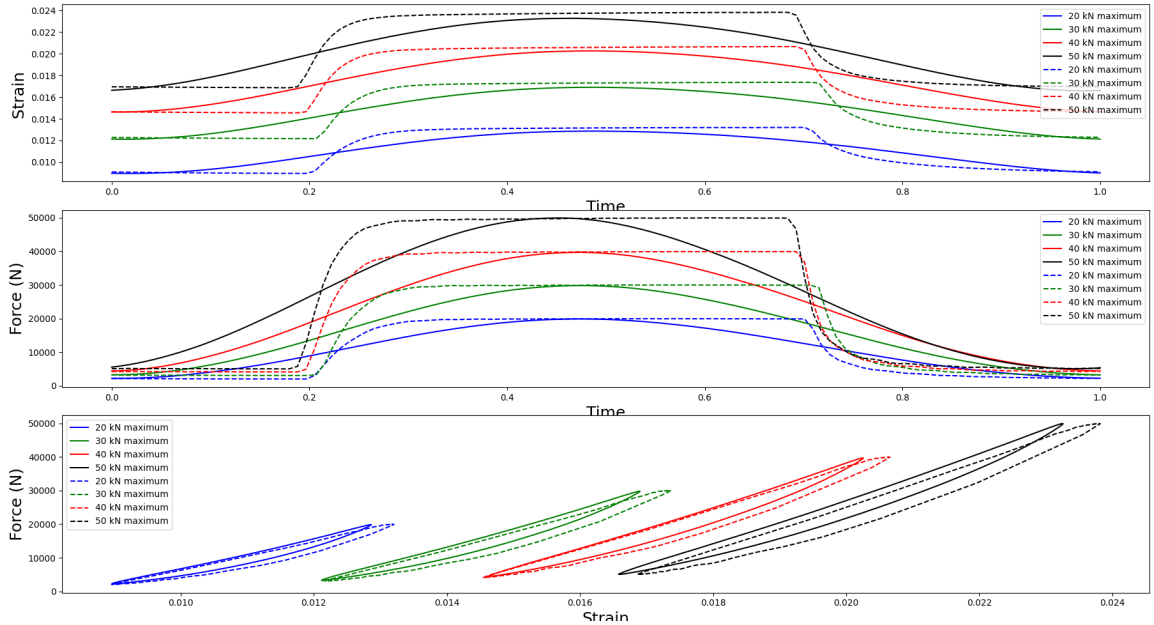


Figure 13: Raw data collected and hysteresis from one single cycle of the first series of tests with sinusoidal and square signals at 1 Hz.

The broader hysteresis loops observed for the square signal compared to the sinusoidal signal can be attributed to the nature of the loading. In the square signal, the rapid transitions between maximum and minimum forces lead to abrupt changes in strain, which generate more internal friction and energy dissipation within the material. This is due to the material's inability to relax or adapt to the sudden changes in stress. In contrast, the sinusoidal signal applies a smoother, more gradual load, allowing the material to respond more progressively and dissipate less energy during each cycle. As a result, the material shows less energy loss, and the hysteresis loops are narrower.

The broader hysteresis loops observed with square signals indicate that more energy is lost in each loading-unloading cycle. This higher energy dissipation is a sign that more energy is being absorbed by the material.

A small amount of damage can also be seen, particularly in the final cycle, where the hysteresis loop for the square signal shifts to the right. This indicates that the material chain has undergone permanent deformation by becoming slightly more elongated. Since this effect becomes noticeably stronger in the last cycle and seems to be caused by the square signal, only the sinusoidal cases have been analyzed.

It is important to note that the hysteresis loops presented in Fig. 13 are not physically accurate, as a single applied force corresponds to four significantly different strain values during both loading and unloading. This would imply that the material has undergone permanent damage. This issue arises due to the setting of the minimum strain at the medium force level, which needs to be adjusted to ensure that the hysteresis loops are physically consistent.

4.2.3 First experimental series of tests: sinusoidal same frequency force signal

Making the analysis on the sinusoidal signals, corrections were applied to the strain measurements by setting the minimum strain to the corresponding values required at force increments of 2 kN, 3 kN, 4 kN, and 5 kN across the various tests. This adjustment can be interpreted as a shift in the strain values while maintaining the force signal unchanged.

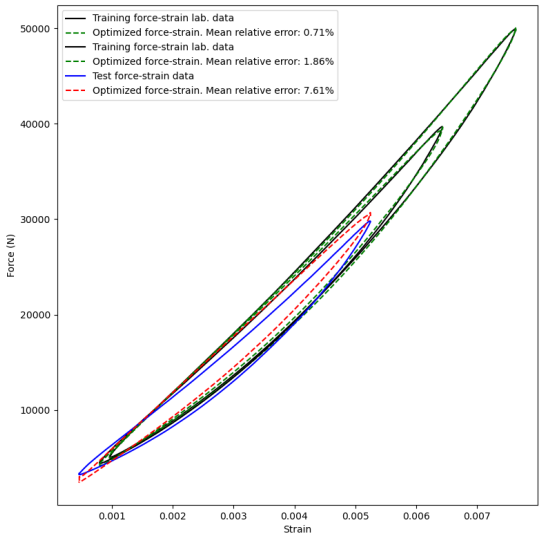
With a single training cycle or several training cycles of the same test, predictions are not made accurately. It is necessary to have training cycles of different tests. The resulting experimental hysteresis loops as well as the fits to the model and predictions are shown in Figures 14a, 14b and 15. The corresponding optimized parameters are found in Tables 3, 4 and 5 respectively.

Parameters	**Values**
Elastic coeff.	$E_1 = 5.6200 \times 10^6,\ E_2 = -1.6091 \times 10^8,\ E_3 = 3.6054 \times 10^{10}$
Visc. loading coeff.	$a_1 = 1.0911 \times 10^6,\ a_2 = 5.1866 \times 10^8,\ a_3 = -1.6994 \times 10^{10}$
Visc. unloading coeff.	$b_1 = -4.2440 \times 10^5,\ b_2 = 1.308040 \times 10^9,\ b_3 = -8.5965 \times 10^{10}$
Kernel coeff.	$C_1 = 39.2656$

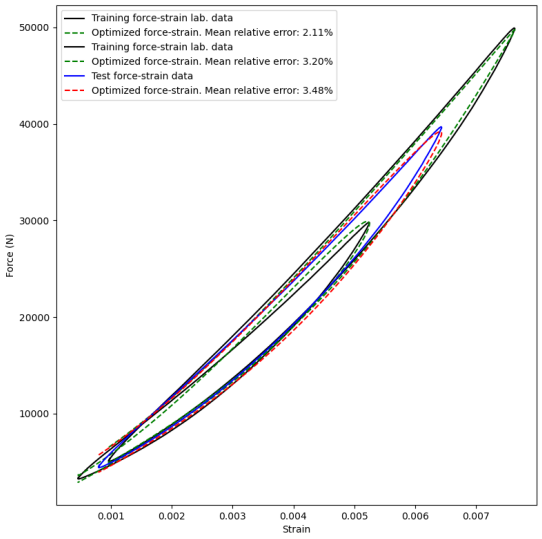
Table 3: Optimized coefficients for single training cycles of the sinusoidal force signals of the first series of tests corresponding to Fig. 14a.

Parameters	**Values**
Elastic coeff.	$E_1 = 8.4721 \times 10^6,\ E_2 = -1.7127 \times 10^9,\ E_3 = 1.7592 \times 10^{11}$
Visc. loading coeff.	$a_1 = -5.8419 \times 10^6,\ a_2 = 2.4025 \times 10^9,\ a_3 = -1.9055 \times 10^{11}$
Visc. unloading coeff.	$b_1 = -4.9505 \times 10^6,\ b_2 = 2.0606 \times 10^9,\ b_3 = -1.8155 \times 10^{11}$
Kernel coeff.	$C_1 = 2.8855$

Table 4: Optimized coefficients for single training cycles of the sinusoidal force signals of the first series of tests corresponding to Fig. 14b.



(a) Single cycle per different force range. The fittings corresponding to 5-50 kN and 4-40 kN force range have a 0.71% mean relative error and 1.86% respectively. The prediction has a 8.05% mean relative error.



(b) Single cycle per different force range. The fittings corresponding to 5-50 kN and 3-30 kN force range have a 2.85% mean relative error and 3.18% respectively. The prediction has a 5.67% mean relative error.

Figure 14: Resulting hysteresis for equally weighted cycles of the first series of tests with sinusoidal signals at 1 Hz.

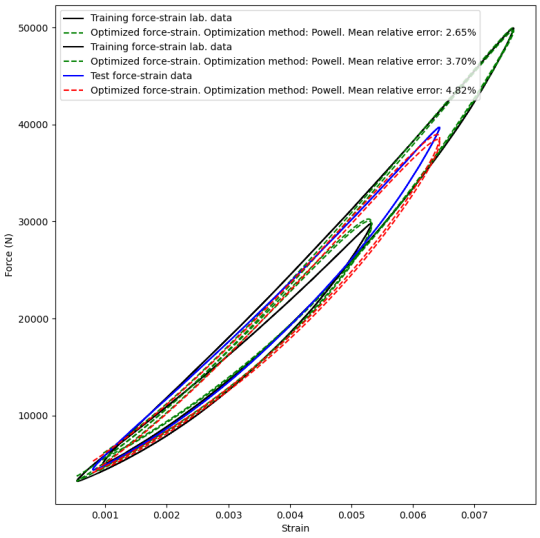


Figure 15: Resulting hysteresis for two consecutive cycles of the first series of tests with sinusoidal signals at 1 Hz. The training cycles are: two corresponding to 3-30 kN force range, and another two cycles of 5-50 kN force range. Predictions are made for two cycles of 4-40 kN range force.

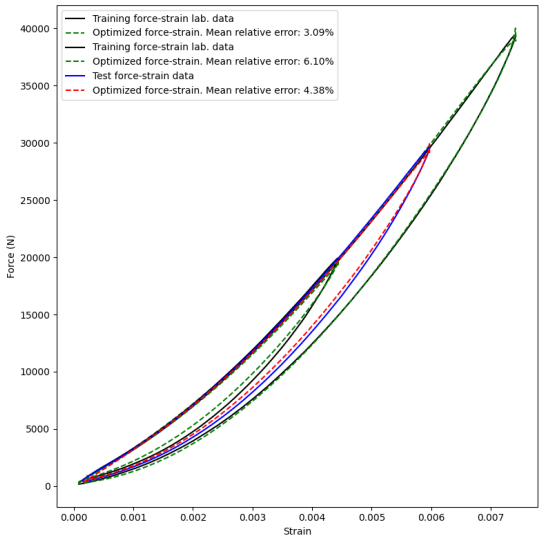
Parameters	**Values**
Elastic coeff.	$E_1 = 8.3172 \times 10^6,\ E_2 = -2.2969 \times 10^9,\ E_3 = 2.5517 \times 10^{11}$
Visc. loading coeff.	$a_1 = -5.5101 \times 10^6,\ a_2 = 3.1245 \times 10^9,\ a_3 = -2.8984 \times 10^{11}$
Visc. unloading coeff.	$b_1 = -4.9504 \times 10^6,\ b_2 = 2.4244 \times 10^9,\ b_3 = -2.3430 \times 10^{11}$
Kernel coeff.	$C_1 = 1.7010$

Table 5: Optimized coefficients for two training cycles per test corresponding to Fig. 15.

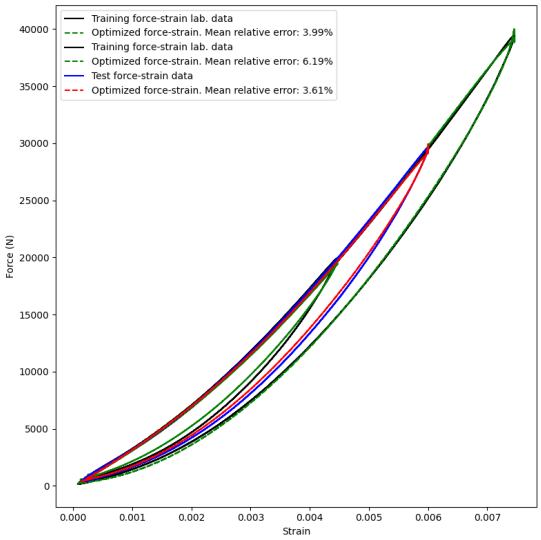
In Fig. 14a, extrapolation is tested. Even though the fits to the training cycles are excellent, the prediction to the smallest cycle is not as good as it appears slightly shifted. In Fig. 14b, the prediction made for the cycle in between the two training cycles is good, but the turning points of the cycle do not seem to be fitted that great. That is why Fig. 15 is introduced, where two training cycles per test have been used for optimization. Even though mean relative errors are similar to when using single training cycles, turning points are fitted better.

4.2.4 Second series of tests: sinusoidal same frequency force signal

By fixing the minimum applied force to 0.5 kN, similar results are found for the second series of tests with sinusoidal signals at 1 Hz. For these cases, corrections to the strain are also needed, as done previously. Results are shown in Figures 16a and 16b, where single cycles and several cycles have been used for optimization. The corresponding optimized parameters are found in Tables 6 and 7 respectively.



(a) Single cycle per different force range. The adjustments corresponding to 4-40 kN and 2-20 kN force range have a 3.09% mean relative error and 6.10% respectively. The prediction has a 4.38% mean relative error.



(b) Four cycles per different force range. The adjustments corresponding to 4-40 kN and 2-20 kN force range have a 3.99% mean relative error and 6.19% respectively. The prediction has a 3.61% mean relative error.

Figure 16: Resulting hysteresis for equally weighted cycles of the second series of tests with sinusoidal signals at 1 Hz.

Parameters	Values
Elastic coeff.	$E_1 = 2.0397 \times 10^6,\ E_2 = 6.6652 \times 10^8,\ E_3 = -3.3016 \times 10^{10}$
Visc. loading coeff.	$a_1 = 1.3782 \times 10^6,\ a_2 = -3.6305 \times 10^8,\ a_3 = 3.9758 \times 10^{10}$
Visc. unloading coeff.	$b_1 = -1.4507 \times 10^6,\ b_2 = 7.2923 \times 10^8,\ b_3 = -1.4817 \times 10^{10}$
Kernel coeff.	$C_1 = 18.3739$

Table 6: Optimized coefficients for single training cycles of the sinusoidal force signals in the second series of tests corresponding to Fig. 16a.

Parameters	Values
Elastic coeff.	$E_1 = 2.0453 \times 10^6,\ E_2 = 6.5282 \times 10^8,\ E_3 = -3.1994 \times 10^{10}$
Visc. loading coeff.	$a_1 = 1.3110 \times 10^6,\ a_2 = -3.6466 \times 10^8,\ a_3 = 4.0768 \times 10^{10}$
Visc. unloading coeff.	$b_1 = -1.4693 \times 10^6,\ b_2 = 7.6542 \times 10^8,\ b_3 = -1.8522 \times 10^{10}$
Kernel coeff.	$C_1 = 18.4492$

Table 7: Optimized coefficients for four training cycles per test of the sinusoidal force signals in the second series of tests corresponding to Fig. 16b.

Although the fittings to the cycles are slightly better when taking single training cycles, predictions are slightly worse. Having more training cycles per test does not necessarily make better adjustments nor predictions, but it does increase the optimization time.

In Fig. 16a, the prediction suggests how the hysteresis cycle would appear without damage. The small change in strain rate does not significantly alter the stiffness between smaller and larger cycles, leading to the reasonable deduction that the chain's geometry affects the hysteresis cycles in this case.

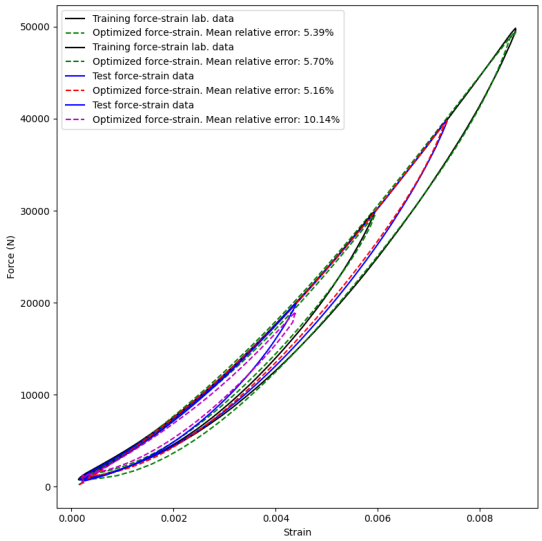
4.2.5 Second series of tests: triangular same frequency force signal

The same study is done for the triangular force signals at 1 Hz. For this case, two or three training cycles were needed, as shown in Figures 17a and 17b. The corresponding optimized parameters are found in Tables 8 and 9 respectively.

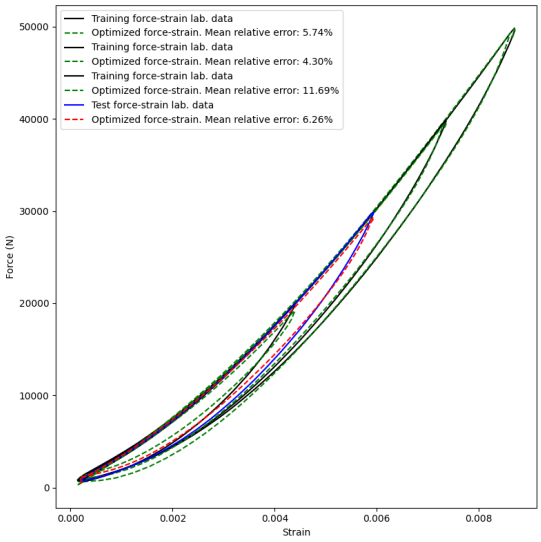
Parameters	Values
Elastic coeff.	$E_1 = 1.3042 \times 10^6,\ E_2 = 8.3440 \times 10^8,\ E_3 = -3.9266 \times 10^{10}$
Visc. loading coeff.	$a_1 = 2.4106 \times 10^6,\ a_2 = -2.3030 \times 10^8,\ a_3 = 1.2055 \times 10^{10}$
Visc. unloading coeff.	$b_1 = -2.2719 \times 10^6,\ b_2 = 5.9673 \times 10^8,\ b_3 = -1.3127 \times 10^9$
Kernel coeff.	$C_1 = 16.5187$

Table 8: Optimized coefficients for single training cycles of the triangular force signals in the second series of tests corresponding to Fig. 17a.

In Fig. 17a, the smallest cycle has the highest mean relative error: 10.14%. There are good fittings to the cycles, but not so good predictions when the prediction is made for cycles not in between the training cycles (extrapolation). For cycles in between the training cycles (interpolation), good predictions are made.



(a) Training cycles: one of 0.5-50 kN force range (mean relative error of 5.39%) and another of 0.5-30 kN force range (mean relative error of 5.70%). Predictions are made for a single cycle within the 0.5-40 kN and 0.5-20 kN force ranges.



(b) Training cycles: one of 0.5-50 kN force range (mean relative error of 5.74%), another to a 0.5-40 kN force range (mean relative error of 4.30%), and another of 0.5-20 kN force range (mean relative error of 11.69%). Predictions are made for a single cycle within the 0.5-30 kN force range.

Figure 17: Resulting hysteresis for equally weighted cycles of the second series of tests with triangular signals at 1 Hz.

Parameters	**Values**
Elastic coeff.	$E_1 = 2.0261 \times 10^6,\ E_2 = 6.1570 \times 10^8,\ E_3 = -2.4459 \times 10^{10}$
Visc. loading coeff.	$a_1 = 1.4960 \times 10^6,\ a_2 = -9.9226 \times 10^7,\ a_3 = 1.2916 \times 10^{10}$
Visc. unloading coeff.	$b_1 = -2.8991 \times 10^6,\ b_2 = 1.0472 \times 10^9,\ b_3 = -3.9156 \times 10^{10}$
Kernel coeff.	$C_1 = 19.1038$

Table 9: Optimized coefficients for single training cycles of the triangular force signals in the second series of tests corresponding to Fig. 17b.

In Fig. 17b, the prediction is good, although not as good as when training with two cycles. The fit to the smallest cycle is not as good as the others, and this is due to the equally weighted cycles when optimizing. If the weight is increased for the smallest cycle, then the fits to the other ones become much worse.

As proved previously, having more training cycles per test give similar adjustments and similar predictions, but does increase the optimization time.

4.2.6 Global analysis of the tests

It is important to observe that, from Fig. 15 to Figures 16 and 17, the strain at maximum stretching force increases, indicating greater elongation of the chain at maximum force. This phenomenon results from the breakage of small fibers due to repeated loading and unloading cycles at high stretching forces, which causes permanent damage. Consequently,

only the longer fibers remain effective when a new stretching force is applied. Therefore, the parameters optimized for the first series of tests are not suitable for the second series, necessitating a re-optimization of the parameters.

This effect is also evident in the optimized coefficients. For example, comparing Table 4 to Table 8, the parameters E_2, a_2, b_2 and E_3, a_3, b_3 decrease by an order of magnitude, indicating a slight reduction in the hysteresis cycle (i.e., the hysteresis cycle has 'fallen' and now shows less inclination).

Determining the relationship between the parameters and the fit to the hysteresis cycles is highly complex. Two different fits that are considered acceptable may have significantly different optimized coefficients. This is exemplified in the case of b_3 in Tables 8 and 9, where a difference of an order of magnitude is observed. Furthermore, it is important to consider both the sensitivity of the model and the fact that the strain error is of 5%.

5 Conclusion and future work

This study aimed to model and analyze hysteresis in viscoelastic materials, particularly focusing on the Dyneema chain, a high-performance synthetic polymer widely used in demanding mechanical applications. The study used Banks' model 3.3 for capturing nonlinear hysteresis, and applied it to a novel context involving cyclic loading and unloading in a controlled experimental setting. Key goals included evaluating the model's capacity to predict energy dissipation and analyzing the impact of varying strain rates on the mechanical response of the material. Testing conditions were specifically designed to avoid permanent damage to the Dyneema chain, ensuring the results would be applicable to real-world use cases, such as mooring lines in marine environments.

Detailed conclusions from this study are discussed.
1. Modeling accuracy of Banks' model
Banks' model has proved effective in describing the hysteresis behavior observed in the Dyneema chain, capturing the cyclic energy dissipation patterns with high accuracy. In controlled settings with relatively small variations in strain rates, the model performed particularly well, demonstrating excellent alignment between experimental data and model predictions. This suggests that Banks' model, originally developed for elastomers, can be effectively adapted to high-performance polymers like Dyneema.

2. Impact of strain rate on model performance
While Banks' model provides good fittings under small strain rate variations, it did not give satisfactory results when there were significantly different strain rates between tests.

3. Influence of multiple training cycles on prediction accuracy
An important observation in this study is that increasing the number of training cycles does not necessarily improve prediction accuracy. However, at least two different test cycles were needed to ensure good predictions, which suggests that a balance between training data volume and computational efficiency must be considered in practice. This is particularly valuable for applications requiring real-time predictions, as limiting the number of

training cycles could significantly reduce processing time without sacrificing accuracy.

4. Chain geometry and its influence on hysteresis behavior
The chain geometry of the Dyneema sample introduced certain challenges not present in standard elastomer tests, where ropes are used. The friction and interaction between chain links affected the overall hysteresis response, especially at higher forces. Although this impact was relatively minor under the test conditions, it suggests that such geometric considerations could have significant implications under more intense loading scenarios.

5. Parameter sensitivity and model optimization
The optimization of model parameters presented some challenges. Even slight changes in the parameters led to noticeable differences in the modeled hysteresis loops. This sensitivity complicates the process of establishing direct correlations between parameter values and specific test conditions, as parameter sets that produce comparable fits may vary significantly.

Taking into account the challenges found in this study, the following future work is suggested.

1. Enhanced modeling of variable strain rates
Future work should explore the application of modified force signals, such as damped sinusoidal signals, to better capture the impact of variable strain rates on hysteresis.

2. Exploring machine learning for parameter optimization
Given the sensitivity of the model, machine learning methods could refine and optimize the parameters with more accuracy as well as make better predictions, enhancing the model's adaptability to new data.

3. Investigation of geometric effects
Future studies could test how different configurations of woven or linked materials influence hysteresis. Taking into account the material geometry would enhance the accuracy and be of high relevance for predictions in real-world settings.

4. Scaling the model for real-world applications
To extend the model's applicability to operational settings, future research should include scaling experiments that simulate real-world conditions, including varied environmental factors like temperature, humidity, and long-term cyclic loading.

For the experimental test conditions of interest in this study, Banks' model has given excellent results, but this study has also addressed some of its limitations. Future research could expand the model's utility and provide a more comprehensive understanding of hysteresis in viscoelastic materials.

References

[1] M. Zhuravkov, Y. Lyu, and E. Starovoitov, *Mechanics of Solid Deformable Body*, ch. 3, 6. Springer Nature Singapore, 2023.

[2] I. M. Ward and J. Sweeney, *Mechanical Properties of Solid Polymers*, ch. 2, 3, 4, 5, 6, 11, 12. John Wiley & Sons, Ltd, 2012.

[3] I. S. U. Center for Nondestructive Evaluation, "Mechanical properties," 2021.

[4] G. Papanicolaou and S. Zaoutsos, "1 - viscoelastic constitutive modeling of creep and stress relaxation in polymers and polymer matrix composites," in *Creep and Fatigue in Polymer Matrix Composites (Second Edition)* (R. M. Guedes, ed.), Woodhead Publishing Series in Composites Science and Engineering, pp. 3–59, Woodhead Publishing, second edition ed., 2019.

[5] K. P. Menard, "Chapter 3 - rheology basics creep–recovery and stress relaxation," in *Dynamic Mechanical Analysis A Practical Introduction*, pp. 37–40, CRC Press, 2008.

[6] M. H. Pebdani, "Study mullins effect of polyurethane reinforcement with halloysite nanotube by molecular dynamics simulation," *Journal of Elastomers & Plastics*, vol. 54, no. 5, pp. 659–675, 2022.

[7] J. C. Maxwell, *On the Dynamical Theory of Gases*, pp. 197–261.

[8] J. Epaarachchi, "17 - the effect of viscoelasticity on fatigue behaviour of polymer matrix composites," in *Creep and Fatigue in Polymer Matrix Composites* (R. M. Guedes, ed.), Woodhead Publishing Series in Composites Science and Engineering, pp. 492–513, Woodhead Publishing, 2011.

[9] V. D. Gordon, M. Davis-Fields, K. Kovach, and C. A. Rodesney, "Biofilms and mechanics: a review of experimental techniques and findings," *Journal of Physics D: Applied Physics*, vol. 50, p. 223002, may 2017.

[10] P. A. Kelly, "Lecture notes: An introduction to solid mechanics. viscoelasticity," 2015.

[11] C. R. Siviour and J. L. Jordan, "High strain rate mechanics of polymers: A review," *Journal of Dynamic Behavior of Materials*, vol. 2, pp. 15–32, 2016.

[12] D. Zheng, S. Lin, J. Ni, and X. Zhao, "Fracture and fatigue of entangled and unentangled polymer networks," *Extreme Mechanics Letters*, vol. 51, p. 101608, 2022.

[13] H. T. Banks, G. A. Pintér, L. K. Potter, M. J. Gaitens, and L. C. Yanyo, "Modeling of nonlinear hysteresis in elastomers under uniaxial tension," *Journal of Intelligent Material Systems and Structures*, vol. 10, no. 2, pp. 116–134, 1999.

[14] A. Joshi, A. Mishra, and V. K. Saxena, "Impact response and energy absorption mechanisms of uhmwpe fabric and composites in ballistic applications: A comprehensive review," *Composites Part A: Applied Science and Manufacturing*, vol. 185, p. 108314, 2024.

[15] P. Pacheco, P. Kenedi, J. Jorge, and A. Paiva, "Analysis of the influence of mechanical properties on the residual stress in offshore chain links using the finite element method," *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*, vol. 3, 01 2003.

[16] H. Banks, L. Potter, and Y. Zhang, "Stress-strain laws for carbon black and silicon filled elastomers," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 4, pp. 3727–3732, 1997.

[17] M. R. V. Bertolo, M. Álison Vigilato Rodrigues, M. M. Horn, J. G. de Oliveira Filho, C. A. Marangon, M. D. Ferreira, A. M. de Guzzi Plepis, and S. Bogusz, "Chapter 7 - rheology of nanoscale polymer-based coatings," in *Polymer-Based Nanoscale Materials for Surface Coatings* (S. Thomas and J. S. George, eds.), pp. 131–149, Elsevier, 2023.

[18] L. Hai, S. qing Wang, and W. cheng Liu, "Modeling creep response for hmpe ropes by a viscoelastic damage model based on fractional derivative theory," *Ocean Engineering*, vol. 298, p. 117181, 2024.

A Appendix

A.1 Python Code

The strain rate needed was calculated with the gradient function in numpy. A low pass filter is applied since in most cases, specially for the triangular force signal, the strain rate presents high noise.

```
def filter(item, cutoff_freq):
    # Butterworth filter design
    order = 4
    # Normalize the frequency with respect to the Nyquist frequency
    nyquist_freq = 0.5 * (1 / np.mean(np.diff(time)))
    normalized_cutoff_freq = cutoff_freq / nyquist_freq
    if normalized_cutoff_freq >= 1:
        # Return the unfiltered item if cutoff_freq is too high
        item = item
    else:
        # Design the Butterworth filter
        b, a = butter(order, normalized_cutoff_freq, btype="low")
        # Apply the filter
        item = filtfilt(b, a, item)
    return item
```

The main code for implementing the Banks' model equation in 3.3 is written in Python and consists of two classes. The first class, *Strain*, receives the strain as an input signal from a file (e.g., .txt, .csv) along with the corresponding time series and strain rate signal. The second class, *Material*, calculates the stress. Material coefficients are provided via a .yaml file, while the time, strain signal, and strain rate signal are obtained from the *Strain* class.

```
import numpy as np
from pathlib import Path
import yaml
from dataclasses import dataclass, field
import pandas as pd
from scipy.signal import butter, filtfilt

@dataclass
class Strain:
    total_time: float = 0.0
    time_step: float = 0.0
    time = np.array([])
    strain_signal = np.array([])
    strain_rate_signal = np.array([])
    amplitude: float = 0.0
    period: float = 0.0
    reference: float = 0.0
    type: str = ""

    def read_df(self, df: pd.DataFrame):
        """
        To set df directly if necessary
        """
        self.type = "file"
```

```
        required_columns = ["time", "strain_signal", "strain_rate_signal"
            ]
        for col in required_columns:
            if col not in df.columns:
                raise ValueError(f"Column {col} is missing from the
                    DataFrame")

        self.time = df["time"].to_numpy()
        self.strain_signal = df["strain_signal"].to_numpy()
        self.strain_rate_signal = df["strain_rate_signal"].to_numpy()

    def read_config(self, file_path: Path):
        """
        This routine loads the object configuration parameters
        """
        with open(file_path) as file:
            dict = yaml.safe_load(file)

        if "strain_signal" not in dict.keys():
            raise ValueError("Input file badly formatted, 'strain_signal'
                is missing.")

        strain_signal_dict = dict["strain_signal"]

        if "type" in strain_signal_dict.keys():
            self.type = np.array(strain_signal_dict["type"])
        else:
            raise ValueError("Input file badly formatted, 'type' is
                missing.")

        if self.type == "cosine" or self.type == "triangular":
            if "total_time" in strain_signal_dict.keys():
                self.total_time = strain_signal_dict["total_time"]
            if "time_step" in strain_signal_dict.keys():
                self.time_step = strain_signal_dict["time_step"]
            if "amplitude" in strain_signal_dict.keys():
                self.amplitude = strain_signal_dict["amplitude"]
            if "period" in strain_signal_dict.keys():
                self.period = strain_signal_dict["period"]
            if "reference" in strain_signal_dict.keys():
                self.reference = np.array(strain_signal_dict["reference"
                    ])

            self.time = np.arange(0, self.total_time + self.time_step,
                self.time_step)
            self.strain_signal = np.array([self.compute_strain(t) for t
                in self.time])
            self.strain_rate_signal = np.array(
                [self.compute_strain_rate(t) for t in self.time]
            )

        if self.type == "file":
            if "filename" in strain_signal_dict.keys():
                filename = strain_signal_dict["filename"]
            else:
                raise ValueError("Input file badly formatted, 'filename'
```

```
                    is missing.")
            df = pd.read_csv(file_path.parent / filename)

            # Check if keys are in df
            required_columns = ["time", "strain_signal", "
                strain_rate_signal"]
            for col in required_columns:
                if col not in df.columns:
                    raise ValueError(
                        f"CSV file badly formatted, '{col}' column is
                            missing."
                    )
            # Assign   time,   strain_signal  and   strain_rate_signal
            self.read_df(df)

    def compute_strain(self, t):
        """
        Operates in a unique time point, not in an array
        """
        if self.type == "cosine":
            strain = self.reference + self.amplitude * np.cos(
                t * 2 * np.pi / self.period
            )
        elif self.type == "triangular":
            t_mod = t % self.period  # Time within the current period
            if t_mod < self.period / 2:
                strain = 4 * self.amplitude * (t_mod / self.period)  #
                    Rising edge
            else:
                strain = 4 * self.amplitude * (1 - t_mod / self.period)
                    # Falling edge

        elif self.type == "file":
            strain = np.interp(t, self.time, self.strain_signal)
        else:
            raise ValueError(f"Unsupported strain type: {self.type}")
        return strain

    def compute_strain_rate(self, t):
        """
        Operates in a unique time point, not in an array
        """
        if self.type == "cosine":
            strain_rate = (
                -self.amplitude
                * 2
                * np.pi
                / self.period
                * np.sin(t * 2 * np.pi / self.period)
            )
        elif self.type == "triangular":
            # Compute the strain rate
            t_mod = t % self.period
            if t_mod < self.period / 2:
                strain_rate = 4 * self.amplitude / self.period  #
                    Positive slope
```

```
            else:
                strain_rate = -4 * self.amplitude / self.period  #
                    Negative slope

        elif self.type == "file":
            strain_rate = np.interp(t, self.time, self.strain_rate_signal
                )
        return strain_rate

    def filter(self, item, cutoff_freq):
        """
        Filters the strain signal to make it smoother
        """
        # Butterworth filter design
        order = 4
        # Normalize the frequency with respect to the Nyquist frequency
        nyquist_freq = 0.5 * (1 / np.mean(np.diff(self.time)))
        normalized_cutoff_freq = cutoff_freq / nyquist_freq
        if normalized_cutoff_freq >= 1:
            # Return the unfiltered item if cutoff_freq is too high
            return item
        # Design the Butterworth filter
        b, a = butter(order, normalized_cutoff_freq, btype="low")
        # Apply the filter
        return filtfilt(b, a, item)

    def filter_strain(self, cutoff_freq):
        filtered_strain = self.filter(self.strain_signal, cutoff_freq)
        self.strain_signal = filtered_strain

    def filter_strain_rate(self, cutoff_freq):
        filtered_strain_rate = self.filter(self.strain_rate_signal,
            cutoff_freq)
        self.strain_rate_signal = filtered_strain_rate


@dataclass
class Material:
    elastic_stress_type: str = " "
    length: float = 0.0
    kernel_coef: float = 0.0
    elastic_coef = np.array([])
    visc_loading_coef = np.array([])
    visc_unloading_coef = np.array([])
    total_time: float = 0.0
    time_step: float = 0.0
    time = np.array([])
    strain: Strain = field(init=False, repr=False)
    stress = np.array([])

    def read_config(self, file_path: Path) -> None:
        """
        This routine loads the objetc configuration parameters
        """
        with open(file_path) as file:
            dict = yaml.safe_load(file)
```

```
        if "material" not in dict.keys():
            raise ValueError("Input file badly formatted, material is
                missing.")
        mat = dict["material"]
        if "length" in mat.keys():
            self.length = mat["length"]
        if "kernel_coef" in mat.keys():
            self.kernel_coef = mat["kernel_coef"]
        if "elastic_coef" in mat.keys():
            self.elastic_coef = np.array(mat["elastic_coef"])
        if "visc_loading_coef" in mat.keys():
            self.visc_loading_coef = np.array(mat["visc_loading_coef"])
        if "visc_unloading_coef" in mat.keys():
            self.visc_unloading_coef = np.array(mat["visc_unloading_coef"
                ])

    def set_strain(self, strain: Strain) -> None:
        self.strain = strain
        self.total_time = self.strain.total_time
        self.time_step = self.strain.time_step
        self.time = self.strain.time

    def kernel(self, t):
        """
        Proposed Kernel for the memory capability of the viscoelastic
            material
        """
        kernel = np.exp(-self.kernel_coef * t)
        return kernel

    def kernel_rate(self, t):
        kernel_rate = -self.kernel_coef * self.kernel(t)
        return kernel_rate

    def compute_stress(self):
        num_points = len(self.time)
        # Initiallize stress and strain arrays
        stress_signal = np.zeros_like(self.time)
        # Initialize arrays required for integral computation
        tau = np.array([])
        visc_resp = np.array([])
        # Strain rate zero-crossing times
        zc_times = np.array([])
        # Strain rate zero-crossing viscoelastic loading responses
        zc_visc_resp_load = np.array([])
        # Strain rate zero-crossing viscoelastic unloading responses
        zc_visc_resp_unload = np.array([])
        # Loop over all time steps
        for i in range(num_points):
            # Extract time and strain values
            tmp_time = self.time[i]
            tmp_strain = self.strain.compute_strain(tmp_time)
            tmp_strain_rate = self.strain.compute_strain_rate(tmp_time)
            # Print the current time and the total time, overwriting the
                previous line
            print(f"Time: {tmp_time:.2f}/{self.time[-1]:.2f}", end="\r")
```

```
            # Evaluate the cubic polynomial elastic stress function
            elastic_stress = sum(
                (tmp_strain ** (j + 1)) * self.elastic_coef[j] for j in
                   range(3)
            )
            ## Compute viscous stress
            # Check for strain rate zero crossing points
            if (
                (i > 1)  # To avoid errors in the initial state (for t=0)
                and (
                    (np.sign(tmp_strain_rate))
                    != np.sign(self.strain.compute_strain_rate(self.time[
                       i - 1]))
                )
                and (i < num_points)
            ):
                zc_time_tmp = self.time[i - 1] - self.strain.
                   compute_strain_rate(
                    self.time[i - 1]
                ) * self.time_step / (
                    tmp_strain_rate - self.strain.compute_strain_rate(
                       self.time[i - 1])
                )
                zc_times = np.append(zc_times, zc_time_tmp)
                zc_strain_tmp = self.strain.compute_strain(zc_time_tmp)
                visc_resp_load_tmp = sum(
                    (zc_strain_tmp ** (j + 1)) * self.visc_loading_coef[j
                       ]
                    for j in range(3)
                )
                visc_resp_unload_tmp = sum(
                    (zc_strain_tmp ** (j + 1)) * self.visc_unloading_coef
                       [j]
                    for j in range(3)
                )
                zc_visc_resp_load = np.append(zc_visc_resp_load,
                   visc_resp_load_tmp)
                zc_visc_resp_unload = np.append(
                    zc_visc_resp_unload, visc_resp_unload_tmp
                )

            if tmp_strain_rate >= 0:
                strain_poly_coef = self.visc_loading_coef
            else:
                strain_poly_coef = self.visc_unloading_coef
            visc_resp_tmp = 0.0
            for j in range(3):
                visc_resp_tmp = (
                    visc_resp_tmp + (tmp_strain ** (j + 1)) *
                       strain_poly_coef[j]
                )
            if i == 0:
                visc_resp_0 = visc_resp_tmp

            tau = np.append(tau, tmp_time)
            visc_resp = np.append(visc_resp, visc_resp_tmp)
```

```
            # Integration by parts
            kernel_rate_tmp_time_tau = self.kernel_rate(tmp_time - tau)
            integrand = kernel_rate_tmp_time_tau * visc_resp
            viscous_stress_integral = np.trapz(
                integrand, tau
            )  # Accepts change in tau for dx=0.001 to make refinement
            final_stress_integral = self.kernel(0) * visc_resp_tmp
            initial_stress_integral = self.kernel(tmp_time) * visc_resp_0
            sum_stress_integral = 0.0
            for ii, t_k in enumerate(zc_times):
                sum_stress_integral = sum_stress_integral + self.kernel(
                    tmp_time - t_k
                ) * ((-1) ** (ii)) * (zc_visc_resp_load[ii] -
                   zc_visc_resp_unload[ii])
            viscous_stress = (
                viscous_stress_integral
                + final_stress_integral
                - initial_stress_integral
                + sum_stress_integral
            )

            stress_signal[i] = elastic_stress + viscous_stress
        self.stress = stress_signal
        return self.stress

    def filter_stress(self, cutoff_freq):
        """
        Filters the strain signal to make it smoother
        """
        # Butterworth filter design
        order = 4
        # Normalize the frequency with respect to the Nyquist frequency
        nyquist_freq = 0.5 * (1 / np.mean(np.diff(self.time)))
        normalized_cutoff_freq = cutoff_freq / nyquist_freq
        unfiltered_stress = self.compute_stress()
        if normalized_cutoff_freq >= 1:
            # Return the unfiltered stress if cutoff_freq is too high
            return unfiltered_stress
        # Design the Butterworth filter
        b, a = butter(order, normalized_cutoff_freq, btype="low")

        # Apply the filter
        return filtfilt(b, a, unfiltered_stress)

    def get_work(self):
        # stress_signal = self.compute_stress()
        strain_rate_signal = self.strain.strain_rate_signal
        ind = np.where(self.time > self.strain.period)[0]
        work = np.trapz(
            self.time[ind] - self.strain.period,
            strain_rate_signal[ind] * self.stress[ind],
        )
        return work
```

Optimization is done separately to the main code, in another Python file. The code for optimization and plotting is given for the single training cycles of the triangular force

signals in the second series of tests.

```
import numpy as np
import pandas as pd
import yaml
from elastohyst.core import Material, Strain
from pathlib import Path
import matplotlib.pyplot as plt
from scipy.optimize import minimize
import scipy.stats as stats
from scipy.signal import butter, filtfilt

length = 0.5 # m
examples_path = Path(__file__).parent
filename_strain_20 = "data/T_20_1.txt"
filename_strain_30 = "data/T_30_1.txt"
filename_strain_40 = "data/T_40_1.txt"
filename_strain_50 = "data/T_50_1.txt"
filename_material = "data/ladicim_all_cycles_in_one.yaml"
file_path = examples_path / filename_material

with open(file_path) as file:
    dict = yaml.safe_load(file)
if "strain_signal" not in dict.keys():
    raise ValueError("Input file badly formatted, 'strain_signal' is
        missing.")

strain_dict = dict["strain_signal"]

if "filename" in strain_dict.keys():
    filename = strain_dict["filename"]
else:
    raise ValueError("Input file badly formatted, 'filename' is missing."
        )
file_path_format = file_path.parent / filename
# Read txt file
df_20 = pd.read_csv(
    examples_path / filename_strain_20,
    skiprows=[0, 1, 2, 3],
    header=[0],
    delimiter="\t",
    decimal=".",
)
df_30 = pd.read_csv(
    examples_path / filename_strain_30,
    skiprows=[0, 1, 2, 3],
    header=[0],
    delimiter="\t",
    decimal=".",
)
df_40 = pd.read_csv(
    examples_path / filename_strain_40,
    skiprows=[0, 1, 2, 3],
    header=[0],
    delimiter="\t",
    decimal=".",
```

```
)
df_50 = pd.read_csv(
    examples_path / filename_strain_50,
    skiprows=[0, 1, 2, 3],
    header=[0],
    delimiter="\t",
    decimal=".",
)
# Rename colums for uniformity
df_20.rename(
    columns={
        "Tiempo": "time",
        "Canal_0_FE=25.00": "stress",
        "Canal_1_FE=5.00": "strain_signal",
        "Canal_2_FE=0.50": "strain_rate_signal",
    },
    inplace=True,
)
df_30.rename(
    columns={
        "Tiempo": "time",
        "Canal_0_FE=25.00": "stress",
        "Canal_1_FE=5.00": "strain_signal",
        "Canal_2_FE=0.50": "strain_rate_signal",
    },
    inplace=True,
)
df_40.rename(
    columns={
        "Tiempo": "time",
        "Canal_0_FE=25.00": "stress",
        "Canal_1_FE=5.00": "strain_signal",
        "Canal_2_FE=0.50": "strain_rate_signal",
    },
    inplace=True,
)
df_50.rename(
    columns={
        "Tiempo": "time",
        "Canal_0_FE=25.00": "stress",
        "Canal_1_FE=5.00": "strain_signal",
        "Canal_2_FE=0.50": "strain_rate_signal",
    },
    inplace=True,
)
columns_to_drop = [
    "Canal_3_FE=1.00",
    "Deformacion",
    "Tension",
    "Canal_4_FE=1.00",
    "Canal_5_FE=1.00",
    "Canal_6_FE=1.00",
    "Canal_7_FE=1.00",
]
for col in columns_to_drop:
    df_20.drop(col, axis=1, inplace=True)
```

```
    df_30.drop(col, axis=1, inplace=True)
    df_40.drop(col, axis=1, inplace=True)
    df_50.drop(col, axis=1, inplace=True)

def filter(item, cutoff_freq):
    # Butterworth filter design
    order = 4
    # Normalize the frequency with respect to the Nyquist frequency
    nyquist_freq = 0.5 * (1 / np.mean(np.diff(time)))
    normalized_cutoff_freq = cutoff_freq / nyquist_freq
    if normalized_cutoff_freq >= 1:
        # Return the unfiltered item if cutoff_freq is too high
        item = item
    else:
        # Design the Butterworth filter
        b, a = butter(order, normalized_cutoff_freq, btype="low")
        # Apply the filter
        # Apply the filter
        item = filtfilt(b, a, item)
    return item

# Cycle selection
df_20 = df_20[(df_20["time"] >= 2.28) & (df_20["time"] <= 3.28)]
df_30 = df_30[(df_30["time"] >= 4.79) & (df_30["time"] <= 5.79)]
df_40 = df_40[(df_40["time"] >= 3.37) & (df_40["time"] <= 4.375)]
df_50 = df_50[(df_50["time"] >= 1.06) & (df_50["time"] <= 2.065)]
time = df_20["time"].to_numpy()
final_time_20 = 1
# Iitilialise time to 0
df_20.loc[:, "time"] = np.linspace(0, final_time_20, num=len(time))
df_30.loc[:, "time"] = np.linspace(0, final_time_20, num=len(time))
df_40.loc[:, "time"] = np.linspace(0, final_time_20, num=len(time))
df_50.loc[:, "time"] = np.linspace(0, final_time_20, num=len(time))
df_20.loc[:, "stress"] = (df_20["stress"]) * 1e3
df_20.loc[:, "strain_signal"] = (df_20["strain_signal"]) * 1e-3 / length
df_30.loc[:, "stress"] = (df_30["stress"]) * 1e3
df_30.loc[:, "strain_signal"] = (df_30["strain_signal"]) * 1e-3 / length
df_40.loc[:, "stress"] = (df_40["stress"]) * 1e3
df_40.loc[:, "strain_signal"] = (df_40["strain_signal"]) * 1e-3 / length
df_50.loc[:, "stress"] = (df_50["stress"]) * 1e3
df_50.loc[:, "strain_signal"] = (df_50["strain_signal"]) * 1e-3 / length

min_strain_20 = min(df_20.loc[:, "strain_signal"])
min_strain_30 = min(df_30.loc[:, "strain_signal"])
min_strain_40 = min(df_40.loc[:, "strain_signal"])
min_strain_50 = min(df_50.loc[:, "strain_signal"])

new_min_strain = (
    0.1 * 1e-3 / length
)  # This value is set by looking at the raw data and estimating the
   strain increase for 0.6 kN under loading conditions

df_20.loc[:, "strain_signal"] = (
    df_20.loc[:, "strain_signal"]
    - min_strain_20
    + new_min_strain
```

```
    + 0.01 * 1e-3 / length
)
df_30.loc[:, "strain_signal"] = (
    df_30.loc[:, "strain_signal"] - min_strain_30 + new_min_strain
)
df_40.loc[:, "strain_signal"] = (
    df_40.loc[:, "strain_signal"]
    - min_strain_40
    + new_min_strain
    - 0.02 * 1e-3 / length
)
df_50.loc[:, "strain_signal"] = (
    df_50.loc[:, "strain_signal"]
    - min_strain_50
    + new_min_strain
    - 0.03 * 1e-3 / length
)
# Set strain rate
df_20.loc[:, "strain_rate_signal"] = np.gradient(
    df_20.loc[:, "strain_signal"], df_20.loc[:, "time"]
)
non_filt_strain_rate_20 = df_20["strain_rate_signal"].copy()
df_30.loc[:, "strain_rate_signal"] = np.gradient(
    df_30.loc[:, "strain_signal"], df_30.loc[:, "time"]
)
non_filt_strain_rate_30 = df_30["strain_rate_signal"].copy()
df_40.loc[:, "strain_rate_signal"] = np.gradient(
    df_40.loc[:, "strain_signal"], df_40.loc[:, "time"]
)
non_filt_strain_rate_40 = df_40["strain_rate_signal"].copy()
df_50.loc[:, "strain_rate_signal"] = np.gradient(
    df_50.loc[:, "strain_signal"], df_50.loc[:, "time"]
)
non_filt_strain_rate_50 = df_50["strain_rate_signal"].copy()
# Filter strain rate
cutoff_freq_strain_rate = 3  # Cutoff frequency (this value depends on
    the data)
# Apply the filter
df_20.loc[:, "strain_rate_signal"] = filter(
    non_filt_strain_rate_20, cutoff_freq_strain_rate
)
df_30.loc[:, "strain_rate_signal"] = filter(
    non_filt_strain_rate_30, cutoff_freq_strain_rate
)
df_40.loc[:, "strain_rate_signal"] = filter(
    non_filt_strain_rate_40, cutoff_freq_strain_rate
)
df_50.loc[:, "strain_rate_signal"] = filter(
    non_filt_strain_rate_50, cutoff_freq_strain_rate
)

fig, axs = plt.subplots(4, 1, figsize=(10, 15))
# Plot 1: total_strain_rate vs. total_time
axs[0].plot(df_20["time"], df_20["strain_signal"], "b-")
axs[0].plot(df_30["time"], df_30["strain_signal"], "g-")
axs[0].plot(df_40["time"], df_40["strain_signal"], "k-")
```

```
axs[0].plot(df_50["time"], df_50["strain_signal"], "y-")
axs[0].set_xlabel("Time", fontsize=18)
axs[0].set_ylabel("Strain ", fontsize=18)

# Plot 2: total_stress vs. total_time
axs[1].plot(df_20["time"], df_20["stress"], "b-")
axs[1].plot(df_30["time"], df_30["stress"], "g-")
axs[1].plot(df_40["time"], df_40["stress"], "k-")
axs[1].plot(df_50["time"], df_50["stress"], "y-")
axs[1].set_xlabel("Time", fontsize=18)
axs[1].set_ylabel("Force (N)", fontsize=18)

# Plot 3: total_strain vs. total_time
axs[2].plot(df_20["strain_signal"], df_20["stress"], "b-")
axs[2].plot(df_30["strain_signal"], df_30["stress"], "g-")
axs[2].plot(df_40["strain_signal"], df_40["stress"], "k-")
axs[2].plot(df_50["strain_signal"], df_50["stress"], "y-")
axs[2].set_xlabel("Strain", fontsize=18)
axs[2].set_ylabel("Force (N)", fontsize=18)

axs[3].plot(df_20["time"], df_20["strain_rate_signal"], "b-")
axs[3].plot(
    df_20["time"], non_filt_strain_rate_20, "b--", label="non-filtered
        strain_rate"
)
axs[3].plot(df_30["time"], df_30["strain_rate_signal"], "g-")
axs[3].plot(
    df_30["time"], non_filt_strain_rate_30, "g--", label="non-filtered
        strain_rate"
)
axs[3].plot(df_40["time"], df_40["strain_rate_signal"], "k-")
axs[3].plot(
    df_40["time"], non_filt_strain_rate_40, "k--", label="non-filtered
        strain_rate"
)
axs[3].plot(df_50["time"], df_50["strain_rate_signal"], "y-")
axs[3].plot(
    df_50["time"], non_filt_strain_rate_50, "y--", label="non-filtered
        strain_rate"
)
axs[3].legend()
axs[3].set_xlabel("Time", fontsize=18)
axs[3].set_ylabel("Strain rate ", fontsize=18)
plt.tight_layout()
plt.show()

# Optimization
optimization_method = "Powell"
# Assign cycles to optimize
df_1 = df_50  # TRAINING CYCLE
df_2 = df_30  # TRAINING CYCLE
df_3 = df_40
df_4 = df_20

st_1 = Strain()
st_1.read_df(df_1)  # st has the first cycle lab data for the strain
```

```
mat = Material()
mat.read_config(file_path)
mat.set_strain(st_1)

st_2 = Strain()
st_2.read_df(df_2)  # st has the second cycle the lab data for the strain
mat = Material()
mat.read_config(file_path)
mat.set_strain(st_2)
stress_non_optimized_2 = mat.compute_stress()  # NOT OPTIMIZED

st_3 = Strain()
st_3.read_df(df_3)  # st has the third cycle lab data for the strain
st_4 = Strain()
st_4.read_df(df_4)  # st has the fourth cycle the lab data for the strain

# Set initial parameters for optimization
initial_params = np.zeros(10)
initial_params[:3] = mat.elastic_coef
initial_params[3:6] = mat.visc_loading_coef
initial_params[6:9] = mat.visc_unloading_coef
initial_params[9] = mat.kernel_coef

# Function to compute stress from Strain and Material classes
def compute_stress(strain: Strain, material: Material):
    material.set_strain(strain)
    return material.compute_stress()

# Root Mean Square Error (RMSE) as the cost function
def cost_function(params, strain: Strain, material: Material, stress_lab)
    :
    """
    Root Mean Square Error (RMSE) as the cost function
        strain: has the strain signal from the laboratory
        material: has the stress computed in elastohyst with the
            numerical model
        stress_lab: stress measured in the laboratory, to be compared
            with the computed one
    """
    # Update material parameters with the current params
    material.elastic_coef = params[:3]
    material.visc_loading_coef = params[3:6]
    material.visc_unloading_coef = params[6:9]
    material.kernel_coef = params[9]
    stress_predicted = compute_stress(strain, material)
    return np.sqrt(np.mean((stress_lab - stress_predicted) ** 2))

def cost_function_2_cycles(
    params,
    strain_1: Strain,
    strain_2: Strain,
    material: Material,
    stress_lab_1,
    stress_lab_2,
):
    """
```

```
    Root Mean Square Error (RMSE) as the cost function
        strain: has the strain signal from the laboratory
        material: has the stress computed in elastohyst with the
            numerical model
        stress_lab: stress measured in the laboratory, to be compared
            with the computed one
    """
    # Update material parameters with the current params
    material.elastic_coef = params[:3]
    material.visc_loading_coef = params[3:6]
    material.visc_unloading_coef = params[6:9]
    material.kernel_coef = params[9]
    stress_predicted_1 = compute_stress(strain_1, material)
    stress_predicted_2 = compute_stress(strain_2, material)
    error_1 = np.sqrt(np.mean((stress_lab_1 - stress_predicted_1) ** 2))
    error_2 = np.sqrt(np.mean((stress_lab_2 - stress_predicted_2) ** 2))
    return (1 / 2) * error_1 + (1 / 2) * error_2

if optimization_method == "Powell":
    # Minimize the RMSE using Powell:
    result_Powell = minimize(
        cost_function_2_cycles,
        initial_params,
        args=(st_1, st_2, mat, df_1["stress"], df_2["stress"]),
        method="Powell",
        options={"maxiter": 10000, "ftol": 1e-6},
    )
    # Optimal parameters
    optimal_params = result_Powell.x

## Predicted values using the optimal parameters
# Create a new material, the optimized one (instead of overwriting in the
    material)
mat_optimized: Material = mat
mat_optimized.elastic_coef = optimal_params[:3]
mat_optimized.visc_loading_coef = optimal_params[3:6]
mat_optimized.visc_unloading_coef = optimal_params[6:9]
mat_optimized.kernel_coef = optimal_params[9]

stress_optimized_1 = compute_stress(st_1, mat_optimized)
stress_optimized_2 = compute_stress(st_2, mat_optimized)
stress_optimized_3 = compute_stress(st_3, mat_optimized)
stress_optimized_4 = compute_stress(st_4, mat_optimized)
# Compute RMSE
rmse_value = cost_function_2_cycles(
    optimal_params, st_1, st_2, mat_optimized, df_1["stress"], df_2["
        stress"]
)
# Work after optimization UNITS ARE NOT CORRECT SINCE STRESS IS THE FORCE
optimized_work = mat_optimized.get_work()
er_rel_1 = abs(df_1["stress"] - stress_optimized_1) / abs(df_1["stress"])
    * 100
m_er_rel_1 = er_rel_1.mean()
er_rel_2 = abs(df_2["stress"] - stress_optimized_2) / abs(df_2["stress"])
    * 100
m_er_rel_2 = er_rel_2.mean()
```

```
er_rel_3 = abs(df_3["stress"] - stress_optimized_3) / abs(df_3["stress"])
    * 100
m_er_rel_3 = er_rel_3.mean()
er_rel_4 = abs(df_4["stress"] - stress_optimized_4) / abs(df_4["stress"])
    * 100
m_er_rel_4 = er_rel_4.mean()
print(f"Dissipated work after optimization: {optimized_work} J")
print(f"The RMSE value is: {rmse_value}")
print(f"Optimization method: {optimization_method}")
print(
    f"Initial parameters:\n"
    f"      Elastic coef: {initial_params[:3]}\n"
    f"      Visc. loading coef: {initial_params[3:6]}\n"
    f"      Visc. unloading coef: {initial_params[6:9]}\n"
    f"      Kernel coef: {initial_params[9]}"
)
print(
    f"Optimized parameters:\n"
    f"      Elastic coef: {optimal_params[:3]}\n"
    f"      Visc. loading coef: {optimal_params[3:6]}\n"
    f"      Visc. unloading coef: {optimal_params[6:9]}\n"
    f"      Kernel coef: {optimal_params[9]}"
)
fig2, axs2 = plt.subplots(1, 1, figsize=(8, 8))
# Plot for stress-strain
axs2.plot(
    df_1["strain_signal"], df_1["stress"], "k-", label="Training force-
        strain lab. data"
)
axs2.plot(
    st_1.strain_signal,
    stress_optimized_1,
    "g--",
    label=f"Optimized force-strain. Optimization method: {
        optimization_method}.  Mean relative error: {m_er_rel_1:.2f}%",
)
axs2.plot(
    df_2["strain_signal"], df_2["stress"], "k-", label="Training force-
        strain lab. data"
)
axs2.plot(
    st_2.strain_signal,
    stress_optimized_2,
    "g--",
    label=f"Optimized force-strain. Optimization method: {
        optimization_method}. Mean relative error: {m_er_rel_2:.2f}%",
)
axs2.plot(df_3["strain_signal"], df_3["stress"], "b-", label="Test force-
    strain data")
axs2.plot(
    st_3.strain_signal,
    stress_optimized_3,
    "r--",
    label=f"Optimized force-strain. Optimization method: {
        optimization_method}. Mean relative error: {m_er_rel_3:.2f}%",
)
```

```
axs2.plot(df_4["strain_signal"], df_4["stress"], "b-", label="Test force-
    strain data")
axs2.plot(
    st_4.strain_signal,
    stress_optimized_4,
    "m--",
    label=f"Optimized force-strain. Optimization method: {
        optimization_method}. Mean relative error: {m_er_rel_4:.2f}%",
)
axs2.set_xlabel("Strain")
axs2.set_ylabel("Force (N)")
axs2.legend()
# Adjust layout
plt.tight_layout()
# Show plot
plt.show()
```

List of Figures

List of Tables