Pontificia Universidad Católica de Chile
Escuela de Ingeniería
ICE/IBM2020 Introducción a la Biomecánica

Tarea 4

Iván Vergara Lam
01 de junio del 2023
Tiempo dedicado: 42 horas

Problema 1. Carga de archivos

i) Para la carga de archivos `.csv`, se utilizó la librería `pandas` en Python. De esta manera, para cargar cada uno de los archivos como **dataframe** se empleó el siguiente código.

```
import pandas as pd

# Paths
harmonic_path = join('data', 'harmonic.csv')
quasistatic_path = join('data', 'quasistatic.csv')

# Import data
harmonic = pd.read_csv(harmonic_path)
quasistatic = pd.read_csv(quasistatic_path)
```

ii) Se debe considerar que las primeras 5 líneas de cada uno de los archivos `.csv` no aportan datos, la primera línea describe la ubicación donde se guardaron los datos, las líneas 2 y 3 están vacías, la línea 4 corresponde al título de cada columna y la línea 5 corresponde a la unidad de medida. En consecuencia, para no tener que editar cada archivo directamente, se optó por importar los archivos como **dataframe** sin considerar estas líneas. Sin embargo, se dejó la línea 4 como título de cada columna, para hacer esto se empleó el *kwarg* `skiprows` al interior de la función para leer el archivo. Además, para eliminar los espacios alrededor de los títulos de cada columna, se utilizó la función `rename`. De esta manera, el importe de archivos se realizó según el siguiente código.

```
import pandas as pd

# Paths
harmonic_path = join('data', 'harmonic.csv')
quasistatic_path = join('data', 'quasistatic.csv')

# Import data
harmonic = pd.read_csv(harmonic_path, skiprows=[0, 1, 2, 4])
quasistatic = pd.read_csv(quasistatic_path, skiprows=[0, 1, 2, 4])

# Remove spaces from column names
quasistatic.rename(columns=lambda x: x.strip(), inplace=True)
harmonic.rename(columns=lambda x: x.strip(), inplace=True)
```

Finalmente, para evaluar el importe de los datos se revisó la cabeza de cada uno de los **dataframe**, para lo que se utilizó la función `head`, resultando en lo siguiente:

	Elapsed Time	Disp	Load 3
0	0.00	2.187	-4.49
1	0.01	2.188	-4.43
2	0.02	2.192	-4.36
3	0.03	2.197	-4.18
4	0.04	2.202	-3.89

Tabla 1: Cabeza de los datos caso armónico

	Elapsed Time	Disp	Load 3
0	0.00	1.064	-7.61
1	0.01	1.065	-7.53
2	0.02	1.069	-7.50
3	0.03	1.074	-7.26
4	0.04	1.079	-6.67

Tabla 2: Cabeza de los datos caso cuasiestático

Es importante notar que el tiempo se encuentra en segundos [s], el desplazamiento en milímetros [mm] y la carga en gramos [g]. Para simplificar el trabajo con los datos, se definieron series en unidades del S.I., para ello, se utilizaron las siguientes relaciones para obtener la Tensión (1) y la Deformación (2).

$$\sigma = \frac{F}{A} = \frac{\text{Carga} \cdot 10^{-3} \cdot g}{A} \tag{1}$$

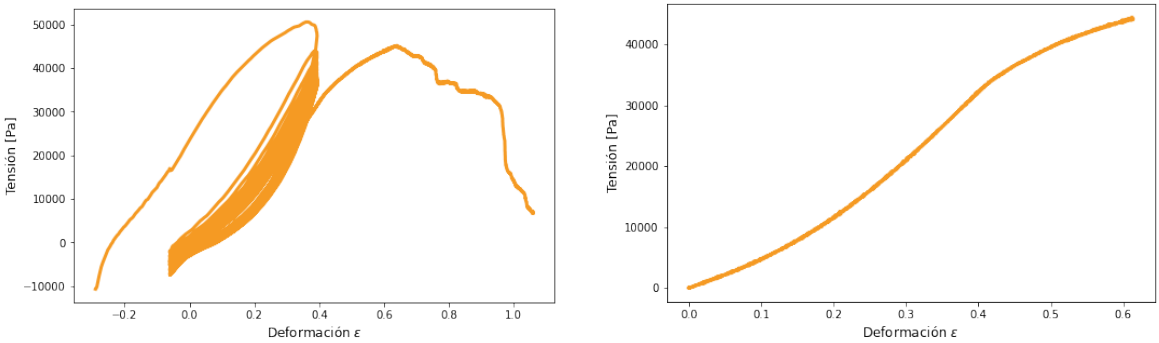
Donde $g = 9{,}81 \frac{\text{m}}{\text{s}^2}$ corresponde a la aceleración de gravedad y $A = 7 \cdot 10^{-6}$ m^2 el área del tejido sometido al ensayo. Por lo tanto, la unidad de medida σ es Pascal [Pa].

$$\epsilon = \frac{\Delta L}{L_0} = \frac{L_f - L_0}{L_0} \tag{2}$$

Donde $L_0 = 2{,}2$ mm es el largo al inicio del ensayo, ΔL el desplazamiento al que se somete la muestra y L_f el largo de la muestra al momento de la medición.

iii) Para el archivo `quasistatic.csv` se realizaron dos gráficos de tensión vs. deformación. Primero, considerando todos los datos y el segundo solo considerando los datos relevantes, los que se presentan en las Figuras 1a y 1b, respectivamente.

Para determinar los datos relevantes, primero se elaboró un gráfico de deformación vs. tiempo, lo que permitió inspeccionar visualmente el comportamiento durante el ensayo. Luego, se utilizó la ventana posterior a los ciclos de preacondicionamiento del material para realizar un gráfico de tensión vs. deformación en el rango elástico del material. Sin embargo, se notó que, en el último tramo de los datos, el material no se comportaba según lo esperado, debido a la aparición de no linealidades. Lo anterior se debe a la ruptura del tejido durante el ensayo, se optó por descartar la sección de los datos donde se presenta la ruptura del tejido, ya que no resulta ser de interés para este análisis. Finalmente, la ventana temporal escogida corresponde a t = [38.70 s, 66.70 s].



(a) Gráfico con todos los datos

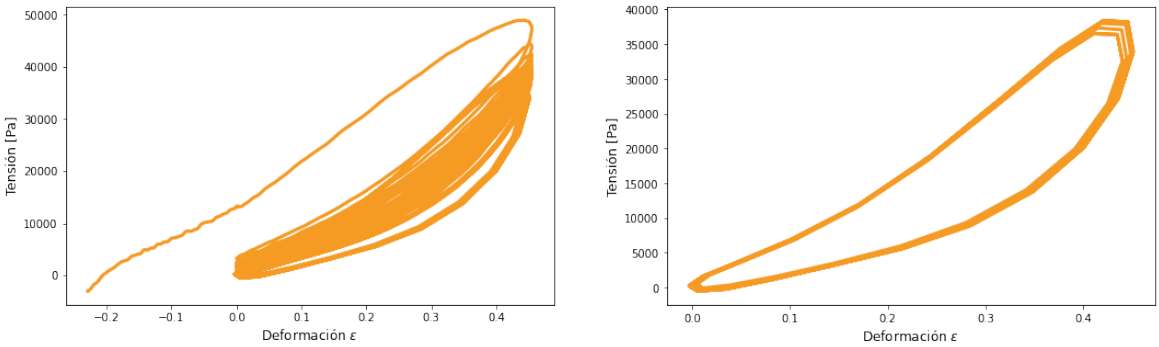
(b) Gráfico con los datos relevantes

Figura 1: Gráficos Tensión vs. Deformación caso cuasiestático

Además, se normalizaron los datos, de tal manera que, en el inicio del experimento, la deformación del material sea $\epsilon = 0$ y la tensión sobre la muestra corresponda a $\sigma = 0$.

iv) De manera análoga, para el archivo `harmonic.csv` se realizaron dos gráficos de tensión vs. deformación. Primero, considerando todos los datos y el segundo solo considerando los datos relevantes, los que se presentan en las Figuras 2a y 2b, respectivamente.

Al igual que en el caso cuasiestático, para determinar los datos relevantes primero se elaboró un gráfico de deformación vs. tiempo, lo que permitió inspeccionar visualmente el comportamiento durante el ensayo. Posteriormente, se utilizó la ventana correspondiente al último comportamiento cíclico del ensayo, sin considerar la etapa de preacondicionamiento, para realizar un gráfico de tensión vs. deformación. Finalmente, la ventana para el ensayo armónico corresponde a t = [39.35 s, 41.35 s], lo que implica que en esa ventana existan 10 ciclos con periodo T = 0.20 s.



(a) Gráfico con todos los datos

(b) Gráfico con los datos relevantes

Figura 2: Gráficos Tensión vs. Deformación caso armónico

Utilizando el mismo procedimiento que en el caso cuasiestático, se normalizaron los datos, de tal manera que, en el inicio del experimento, la deformación del material sea $\epsilon = 0$ y la tensión sobre la muestra corresponda a $\sigma = 0$.

Problema 2. Estudio de datos caso cuasiestático

i) Para un material Neohookeano incompresible, el tensor de Cauchy queda descrito por la expresión (3).

$$\boldsymbol{\sigma}^{nh} = -p\boldsymbol{I} + 2C_1\boldsymbol{B} \tag{3}$$

Donde p corresponde a la presión, $\boldsymbol{I}$ al tensor identidad, $C_1 = \frac{\mu}{2}$ a una constante del material y $\boldsymbol{B} = \boldsymbol{F}\boldsymbol{F}^T$ al tensor izquierdo de Cauchy-Green.

Considerando que el material no presenta deformaciones angulares, se puede definir el tensor gradiente de deformaciones $\boldsymbol{F}$ y el tensor izquierdo de Cauchy-Green $\boldsymbol{B}$ como:

$$[\boldsymbol{F}] = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda_T & 0 \\ 0 & 0 & \lambda_T \end{pmatrix} \qquad [\boldsymbol{B}] = \begin{pmatrix} \lambda^2 & 0 & 0 \\ 0 & \lambda_T^2 & 0 \\ 0 & 0 & \lambda_T^2 \end{pmatrix} \tag{4}$$

Donde $\lambda = 1 + \epsilon$ corresponde al estiramiento en la dirección principal $\hat{x}$, mientras que λ_T describe el estiramiento en las direcciones $\hat{y}$ y $\hat{z}$.

Además, al considerar las condiciones de superficie libre, solo existe tensión en una dirección, es decir, $\sigma_{22} = \sigma_{33} = 0$. Por lo tanto, el tensor de tensiones se puede escribir como:

$$[\boldsymbol{\sigma}^{nh}] = \begin{pmatrix} \sigma_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{5}$$

Como se trata de un material Neohookeano incompresible, se cumple que $J = \mathrm{Det}(\boldsymbol{F}) = 1$. Por lo tanto, a partir del determinante se puede determinar que λ_T es una función del estiramiento λ, en particular:

$$\begin{aligned} \mathrm{Det}(\boldsymbol{F}) &= 1 \\ \lambda\lambda_T^2 &= 1 \\ \lambda_T &= \sqrt{\frac{1}{\lambda}} \end{aligned} \tag{6}$$

De esta manera, las expresiones descritas en (4), se pueden reescribir como:

$$[\boldsymbol{F}] = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \sqrt{\frac{1}{\lambda}} & 0 \\ 0 & 0 & \sqrt{\frac{1}{\lambda}} \end{pmatrix} \qquad [\boldsymbol{B}] = \begin{pmatrix} \lambda^2 & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 \\ 0 & 0 & \frac{1}{\lambda} \end{pmatrix} \tag{7}$$

A partir de lo anterior, es posible despejar la presión p en función de las otras variables del problema. Para ello, se puede escribir la igualdad descrita por la expresión (8).

$$\begin{aligned} \boldsymbol{\sigma}^{nh} &= -p\boldsymbol{I} + 2C_1\boldsymbol{B} \\ \sigma_{22}^{nh} &= -p + 2C_1\lambda_T^2 \\ \sigma_{22}^{nh} &= -p + 2C_1\frac{1}{\lambda} \\ 0 &= -p + 2C_1\frac{1}{\lambda} \\ p &= \frac{2C_1}{\lambda} \end{aligned} \tag{8}$$

Luego, se puede utilizar la expresión para la presión encontrada en (8) para hallar la tensión en la dirección principal $\hat{x}$ en función de las variables C_1 y λ. De esta manera, se tiene:

$$\begin{aligned}
\boldsymbol{\sigma}^{nh}(\lambda) &= -p\boldsymbol{I} + 2C_1\boldsymbol{B} \\
\sigma_{11}^{nh}(\lambda) &= -p + 2C_1\lambda^2 \\
\sigma_{11}^{nh}(\lambda) &= -\frac{2C_1}{\lambda} + 2C_1\lambda^2 \\
\sigma_{11}^{nh}(\lambda) &= 2C_1\left(\lambda^2 - \frac{1}{\lambda}\right) \\
\sigma_{11}^{nh}(\lambda) &= \mu\left(\lambda^2 - \frac{1}{\lambda}\right)
\end{aligned} \tag{9}$$

ii) Existen modelos viscoelásticos que permiten relacionar tensiones con deformaciones, se estudiarán los modelos de sólido de Voight y de sólido lineal estándar, considerando la deformación como una función lineal del tiempo. En el caso de un sólido de Voight, la tensión en función de la deformación queda descrita por la expresión (19).

$$\sigma^v(\epsilon) = \mu\dot{\epsilon} + E\epsilon \tag{10}$$

Al escribir la deformación ϵ en términos del estiramiento λ utilizando la relación descrita en el apartado i), se puede representar la tensión como una función del estiramiento, de esta manera:

$$\begin{aligned}
\sigma^v(\lambda) &= \mu\dot{\overline{(\lambda - 1)}} + E(\lambda - 1) \\
\sigma^v(\lambda) &= \mu\dot{\lambda} + E(\lambda - 1)
\end{aligned} \tag{11}$$

Donde μ y E son parámetros del material, mientras que $\dot{\lambda} = \dot{\epsilon} = \frac{d\epsilon}{dt}$ corresponde a la pendiente de la recta deformación vs. tiempo calculada en el apartado iii).

Para el caso del sólido lineal estándar, se utiliza la relación entre tensión y deformación.

$$\dot{\sigma} + \frac{1}{\tau_\epsilon}\sigma = \frac{E_\infty}{\tau_\epsilon}\left(\epsilon + \tau_\sigma\dot{\epsilon}\right) \tag{12}$$

Donde

- $E_\infty = E_1$ es el **módulo elástico de régimen**
- $\tau_\epsilon = \frac{\mu}{E_2}$ es la **constante de tiempo de relajación de tensión**
- $\tau_\sigma = \mu\frac{E_1+E_2}{E_1E_2}$ es la **constante de tiempo de creep**

La EDO anterior (12) se puede reescribir como:

$$\begin{aligned}
\frac{d\sigma}{dt} + \frac{1}{\tau_\epsilon}\sigma &= \frac{E_\infty}{\tau_\epsilon}\left(\epsilon + \tau_\sigma\frac{d\epsilon}{dt}\right) \\
\frac{d\sigma}{d\epsilon}\frac{d\epsilon}{dt} + \frac{1}{\tau_\epsilon}\sigma &= \frac{E_\infty}{\tau_\epsilon}\left(\epsilon + \tau_\sigma\frac{d\epsilon}{dt}\right)
\end{aligned} \tag{13}$$

Donde $\frac{d\epsilon}{dt}$ corresponde a la pendiente de la recta deformación vs. tiempo calculada en el apartado iii). Se utilizará m en su lugar por simplicidad.

$$\begin{aligned}
\frac{d\sigma}{d\epsilon}m + \frac{1}{\tau_\epsilon}\sigma &= \frac{E_\infty}{\tau_\epsilon}\left(\epsilon + \tau_\sigma m\right) \\
\frac{d\sigma}{d\epsilon} + \frac{1}{\tau_\epsilon m}\sigma &= \frac{E_\infty}{\tau_\epsilon m}\left(\epsilon + \tau_\sigma m\right)
\end{aligned} \tag{14}$$

Para resolver la EDO (14), se puede utilizar el factor integrante, lo que resulta en:

$$\begin{aligned}\frac{d\sigma}{d\epsilon}e^{\epsilon/\tau_\epsilon m}+\frac{e^{\epsilon/\tau_\epsilon m}}{\tau_\epsilon m}\sigma &= e^{\epsilon/\tau_\epsilon m}\frac{E_\infty}{\tau_\epsilon m}\left(\epsilon+\tau_\sigma m\right)\\ \left(\sigma e^{\epsilon/\tau_\epsilon m}\right)' &= e^{\epsilon/\tau_\epsilon m}\frac{E_\infty}{\tau_\epsilon m}\left(\epsilon+\tau_\sigma m\right)\end{aligned} \tag{15}$$

Al integrar la expresión (15) considerando $\epsilon(0)=0,\ \sigma(0)=0$, lo anterior resulta en:

$$\begin{aligned}\sigma e^{\epsilon/\tau_\epsilon m} &= \int_0^\epsilon e^{\epsilon^*/\tau_\epsilon m}\frac{E_\infty}{\tau_\epsilon m}\left(\epsilon^*+\tau_\sigma m\right)d\epsilon^*\\ \sigma e^{\epsilon/\tau_\epsilon m} &= E_\infty\left(e^{\epsilon/\tau_\epsilon m}(\epsilon+\tau_\sigma m-\tau_\epsilon m)+\tau_\epsilon m-\tau_\sigma m\right)\\ \sigma^{sls}(\epsilon) &= E_\infty e^{-\epsilon/\tau_\epsilon m}\left(e^{\epsilon/\tau_\epsilon m}(\epsilon+\tau_\sigma m-\tau_\epsilon m)+\tau_\epsilon m-\tau_\sigma m\right)\\ \sigma^{sls}(\epsilon) &= E_\infty\left(\epsilon+\tau_\sigma m-\tau_\epsilon m+e^{-\epsilon/\tau_\epsilon m}(\tau_\epsilon m-\tau_\sigma m)\right)\end{aligned} \tag{16}$$

Al igual que anteriormente, se escribe la deformación ϵ en términos del estiramiento λ utilizando la relación descrita en el apartado i), por lo que la expresión (16) queda descrita por (17).

$$\sigma^{sls}(\lambda)=E_\infty\left((\lambda-1)+\tau_\sigma m-\tau_\epsilon m+e^{-(\lambda-1)/\tau_\epsilon m}(\tau_\epsilon m-\tau_\sigma m)\right) \tag{17}$$

iii) En primer lugar, se ajustó el modelo Neohookeano incompresible, para ello, se definió la expresión (9) como función en Python, a la que posteriormente se le hizo un ajuste de mínimos cuadrados utilizando la función `scipy.optimize.leastq` para determinar la constante μ.

De esta manera, la expresión para la tensión en la dirección principal $\hat{x}$ corresponde a:

$$\sigma^{nh}=23908{,}8\left(\lambda^2-\frac{1}{\lambda}\right) \tag{18}$$

Finalmente, se puede visualizar gráficamente el ajuste del modelo Neohookeano incompresible a los datos de estiramiento λ en la Figura 3.

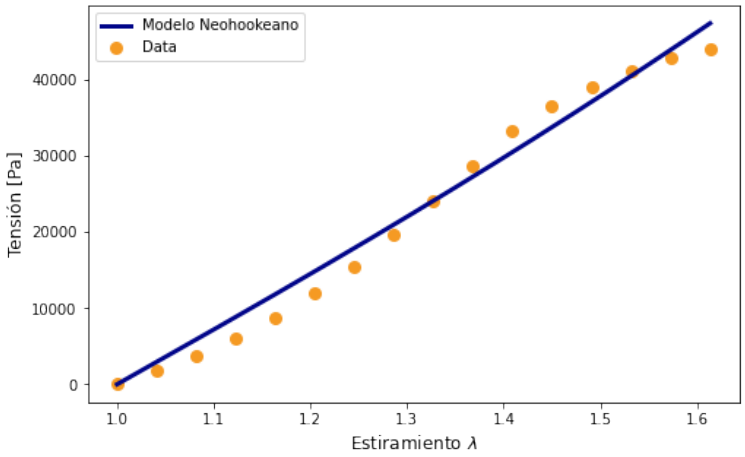


Figura 3: Ajuste del modelo Neohookeano incompresible a los datos de estiramiento

Para el modelo de sólido de Voight, en primer lugar es necesario calcular la derivada temporal de la deformación $\dot{\epsilon}$, que puede ser obtenida fácilmente al aproximar los datos de deformación vs. tiempo

como una recta y extraer la pendiente $m = 0{,}9192$. Posteriormente, se reemplazó el valor obtenido en la expresión (11). Repitiendo el procedimiento utilizado anteriormente, se empleó el método de mínimos cuadrados para determinar las constantes μ y E. De esta manera, la expresión final corresponde a:

$$\sigma^v = -1381{,}81 \cdot 2{,}27 + 82965{,}52\ (\lambda - 1) \tag{19}$$

El gráfico del ajuste del modelo de sólido de Voight a los datos se presenta en la Figura 4.

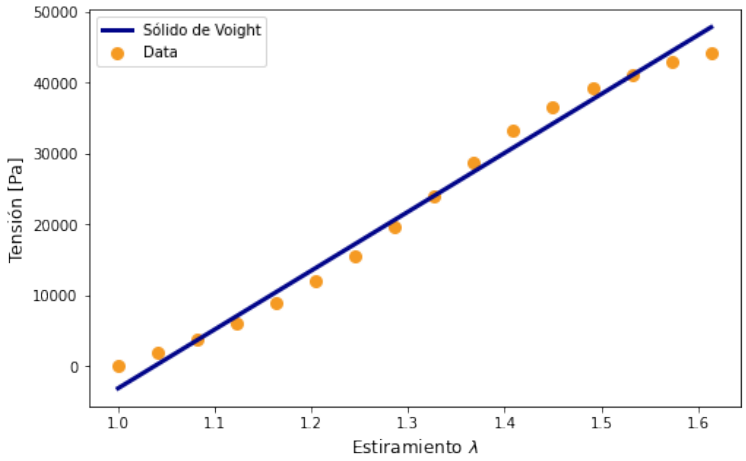


Figura 4: Ajuste del modelo de sólido de Voight a los datos de estiramiento

En cuanto al sólido lineal estándar, se utiliza la expresión (17), donde se reemplaza $m = \frac{d\epsilon}{dt} = \frac{d\lambda}{dt}$ por el mismo valor utilizado para el modelo de sólido de Voight. Posteriormente, se utiliza el método de mínimos cuadrados para determinar las constantes μ, E_1 y E_2, con las que se pueden determinar los parámetros E_∞, τ_ϵ y τ_σ. Por último, la expresión obtenida para el modelo de sólido lineal estándar luego de ajustar los datos del ensayo, corresponde a:

$$\sigma^{sls}(\lambda) = 70788{,}6 \left((\lambda - 1) + 2{,}3 \cdot (5453{,}8 - 5127{,}9) + e^{-(\lambda-1)/5127{,}9 \cdot 2{,}3} \cdot 2{,}3 \cdot (5127{,}9 - 5453{,}8) \right) \tag{20}$$

El gráfico para el modelo de sólido lineal estándar se presenta en la Figura 5.

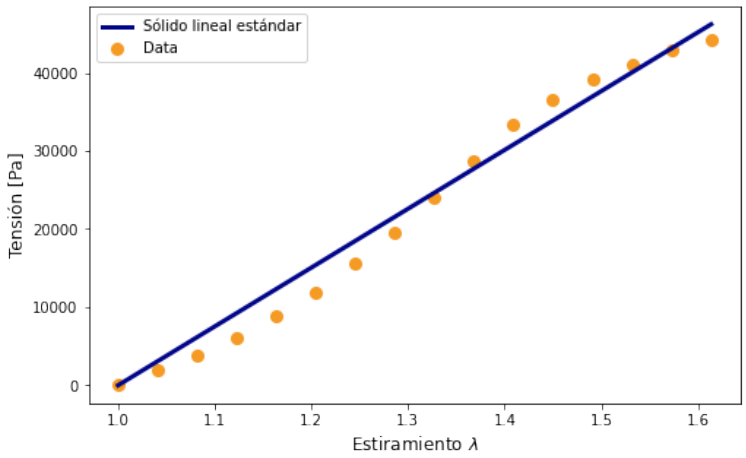


Figura 5: Ajuste del modelo de sólido lineal estándar a los datos de estiramiento

Para comprobar que el procedimiento realizado había sido correcto, se asumió la deformación como una función del tiempo $\epsilon(t)$, lo que implica que la tensión también se pueda escribir como una función del tiempo $\sigma(t)$. Esta última función es descrita por la expresión (21),

$$\sigma(t) = \epsilon(t) E_\infty \left\{ 1 - \left(1 - \frac{\tau_\sigma}{\tau_\epsilon} \right) e^{-t/\tau_\epsilon} \right\} \tag{21}$$

La función fue ajustada utilizando el método de mínimos cuadrados para encontrar las constantes, lo que resultó en el mismo gráfico que el de la Figura 5, por lo que se concluyó que el desarrollo de la EDO, utilizando los supuestos y los cambios en los diferenciales, expresados en la igualdad (13), había sido correcto.

Los parámetros obtenidos mediante el ajuste de mínimos cuadrados para cada uno de los 3 modelos viscoelásticos se presentan en la Tabla 3.

	Neohookeano incompresible	Sólido de Voight	Sólido lineal estándar
μ	23908.8455	-1381.8076	23070762.7584
E	—	82965.5169	—
E_∞	—	—	70788.6219
τ_ϵ	—	—	5127.8813
τ_σ	—	—	5453.7919

Tabla 3: Parámetros obtenidos utilizando el método de mínimos cuadrados

iv) Para comparar el ajuste realizado a los datos, en primer lugar se graficaron las 3 curvas obtenidas para poder diferenciarlas visualmente. La Figura 6 muestra el ajuste de los 3 modelos a los datos del ensayo cuasiestático.

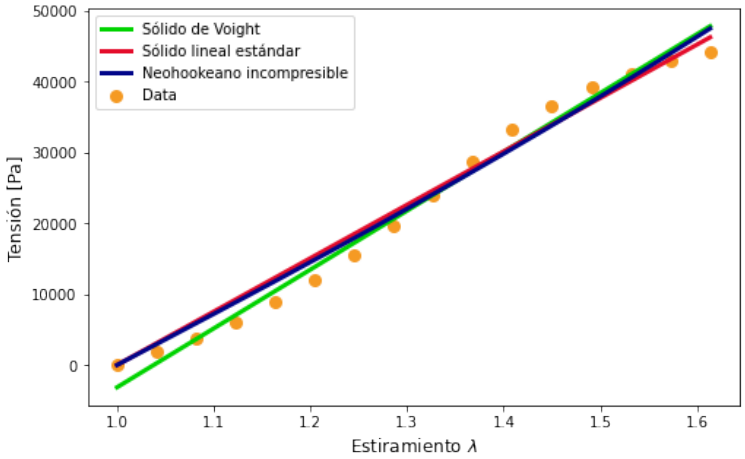


Figura 6: Ajuste de los 3 modelos a los datos de estiramiento

A partir de lo observado en la Figura 6 se puede apreciar a simple vista que el modelo que peor se ajusta a los datos corresponde al sólido de Voight, dado que modela la tensión como una función afín, cuyo coeficiente de posición corresponde a un múltiplo de la tasa de cambio de la deformación, por lo que la condición de $\epsilon(0) = \sigma(0) = 0$ está sujeta a ese término. Además, depende linealmente de la deformación, por lo que pequeñas perturbaciones podrían generar no linealidades, invalidando este supuesto.

En cuanto a los modelos de sólido neohookeano incompresible y de sólido lineal estándar, ambos tienen un ajuste similar a los datos, pero difieren en la concavidad de la curva. Sin embargo, se evaluaron distintos parámetros para inicializar el método de mínimos cuadrados, con lo cual se encontraron, entre otras cosas, curvas del modelo de sólido lineal estándar cóncavas, no convexas como en el resultado final que fue representado en el apartado iii).

Lo anterior se debe a que la función `scipy.optimize.leastq` busca el mínimo local más cercano, el que no siempre coincide con el mínimo absoluto de una función. De esta manera, el mínimo local que se encuentra depende directamente de los parámetros con los que se inicialice el ajuste. Lo anterior puede generar grandes variaciones en funciones que presentan no linealidades, como es el caso del sólido lineal estándar.

Tomando en cuenta las funciones obtenidas, se puede concluir que el modelo de solido lineal estándar logró el mejor ajuste a los datos del ensayo de estiramiento cuasiestático, lo que permite concluir que el supuesto de que el tejido es unidimensional, es válido para este caso si se describe el material según este modelo viscoelástico.

Problema 3. Estudio de datos caso armónico

i) Se quiere realizar un estudio del comportamiento armónico, por lo que resulta conveniente utilizar funciones complejas, definidas por:

$$\epsilon^*(t) = \epsilon_0 e^{i\omega t} = \epsilon_0(\cos\omega t + i\sin\omega t) \qquad \sigma^*(t) = \sigma_0 e^{i(\omega t+\delta)} = \sigma_0(\cos(\omega t+\delta) + i\sin(\omega t+\delta)) \tag{22}$$

Luego, para calcular el módulo de pérdida es necesario recordar que, bajo excitación armónica, el módulo complejo E^* se define como:

$$E^*(\omega) = \frac{\sigma^*}{\epsilon^*} = E'(\omega) + iE''(\omega) \tag{23}$$

Donde E' es el módulo de almacenamiento y E'' el módulo de pérdida.

Recordando que la relación en un sólido de Voight se describe por la expresión (10), se tiene:

$$\begin{aligned} \sigma^* &= \mu\dot{\epsilon}^* + E\epsilon^* \\ \sigma^* &= i\omega\mu\epsilon^* + E\epsilon^* \\ \sigma^* &= \epsilon^*(i\omega\mu + E) \\ E^* = \frac{\sigma^*}{\epsilon^*} &= i\omega\mu + E \end{aligned} \tag{24}$$

A partir de la expresión (24), es fácil notar que el módulo de almacenamiento corresponde a $E' = E$, mientras que el módulo de pérdida es $E'' = \omega\mu$.

En el caso de un sólido lineal estándar, la ecuación que relaciona la tensión y la deformación está descrita por la expresión

$$\begin{aligned} \sigma^* + \tau_\epsilon\dot{\sigma}^* &= E_\infty(\epsilon^* + \tau_\sigma\dot{\epsilon}^*) \\ \sigma^* + i\omega\tau_\epsilon\sigma^* &= E_\infty(\epsilon^* + i\omega\tau_\sigma\epsilon^*) \\ \sigma^*(1 + i\omega\tau_\epsilon) &= E_\infty\epsilon^*(1 + i\omega\tau_\sigma) \\ E^* = \frac{\sigma^*}{\epsilon^*} &= E_\infty\frac{1 + i\omega\tau_\sigma}{1 + i\omega\tau_\epsilon} \end{aligned} \tag{25}$$

Multiplicando está expresión (25) por el conjugado del denominador, es decir, $\frac{1-i\omega\tau_\epsilon}{1-i\omega\tau_\epsilon}$, se obtiene:

$$\begin{aligned} E^* &= E_\infty\frac{1 - i\omega\tau_\epsilon + i\omega\tau_\sigma + \omega^2\tau_\epsilon\tau_\sigma}{1 + \omega^2\tau_\epsilon^2} \\ E^* &= E_\infty\frac{1 + \omega^2\tau_\epsilon\tau_\sigma + i\omega(\tau_\sigma - \tau_\epsilon)}{1 + \omega^2\tau_\epsilon^2} \end{aligned} \tag{26}$$

A partir de la expresión (26), se puede notar que, en el modelo de sólido lineal estándar, el módulo de almacenamiento corresponde a $E' = E_\infty\frac{1+\omega^2\tau_\epsilon\tau_\sigma}{1+\omega^2\tau_\epsilon^2}$, mientras que el módulo de pérdida está dado por $E'' = E_\infty\frac{\omega(\tau_\sigma-\tau_\epsilon)}{1+\omega^2\tau_\epsilon^2}$.

ii) Para el análisis del caso armónico, se escogió el primer ciclo de excitación, comprendido en la ventana temporal t = [39.35 s, 39.55 s]. En primer lugar, se elaboró el gráfico de tensión vs. deformación para este ciclo, mostrado en la Figura 7.

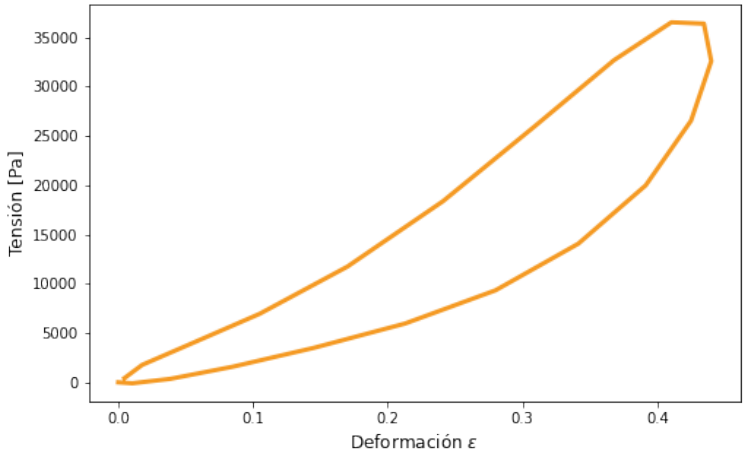


Figura 7: Gráfico de Tensión vs. Deformación para un ciclo de excitación armónica

Posteriormente, para aproximar la energía disipada durante el ciclo se utiliza la expresión (27).

$$W = \oint \sigma d\epsilon \tag{27}$$

Se quiere determinar la energía disipada a partir de un set de datos, por lo que se utiliza el método de Simpson para aproximar la integral numéricamente, el cual subdivide la función usando polinomios cuadráticos, lo que genera curvas en vez de rectas en las zonas superiores de los trapecios. Para implementar este método en Python se utilizó la función `scipy.integrate.simpson`, que recibe como argumentos los datos de tensión y de deformación.

El resultado de la integral corresponde a: $W = -4{,}1305 \cdot 10^3 \ \frac{\text{J}}{\text{m}^3}$.

iii) La energía por unidad de volumen disipada durante un ciclo de carga puede ser calculada a partir del módulo de pérdida, según la expresión (28).

$$W = \epsilon_0^2 \pi E''(\omega) \tag{28}$$

Donde $\epsilon_0 = 0{,}22$ corresponde a la máxima amplitud de la deformación y $E''(\omega)$ al módulo de pérdida. De igual manera, $\omega = 10\pi$ corresponde a la frecuencia de oscilación.

Para cada uno de los modelos, se pueden reemplazar los parámetros obtenidos en el Problema 2 y las expresiones para el módulo de pérdida en el apartado i) en la expresión (28), para encontrar la predicción de la energía disipada.

En el caso del sólido de Voight, se tiene:

$$W = 0{,}22^2 \pi \omega \mu = -6{,}6007 \cdot 10^3 \ \frac{\text{J}}{\text{m}^3} \tag{29}$$

En cuanto al sólido lineal estándar, se tiene:

$$W = 0{,}22^2 \pi E_\infty \frac{\omega(\tau_\sigma - \tau_\epsilon)}{1 + \omega^2 \tau_\epsilon^2} = 4{,}2465 \cdot 10^{-3} \ \frac{\text{J}}{\text{m}^3} \tag{30}$$

A partir de los resultados, es fácil notar que el modelo que genera la mejor predicción corresponde al sólido de Voight. Lo anterior puede tener origen en la concavidad de la curva de cada modelo ajustado, debido a que se pueden generar alteraciones en la segunda mitad del ciclo, afectando la histéresis, y así, la energía disipada.

iv) Similar a lo realizado en el Problema 2, se ajustaron los datos de tensión al modelo armónico, utilizando la expresión de carga cíclica (31).

$$\sigma(t) = \sigma_0 \cos(\omega t + \delta) \tag{31}$$

Donde $\sigma_0 = 19220{,}5928$ corresponde a la máxima amplitud de la tensión durante el ensayo armónico, $\omega = 10\pi$ a la frecuencia de oscilación y δ a la fase.

Mediante la realización de un ajuste de mínimos cuadrados, se obtuvo un valor numérico para δ, lo que permite reescribir la tensión como:

$$\sigma(t) = 19220{,}5928 \cos(10\pi t - 1{,}9934) \tag{32}$$

La función obtenida se presenta junto a los datos del ensayo de estiramiento armónico en la Figura 8.

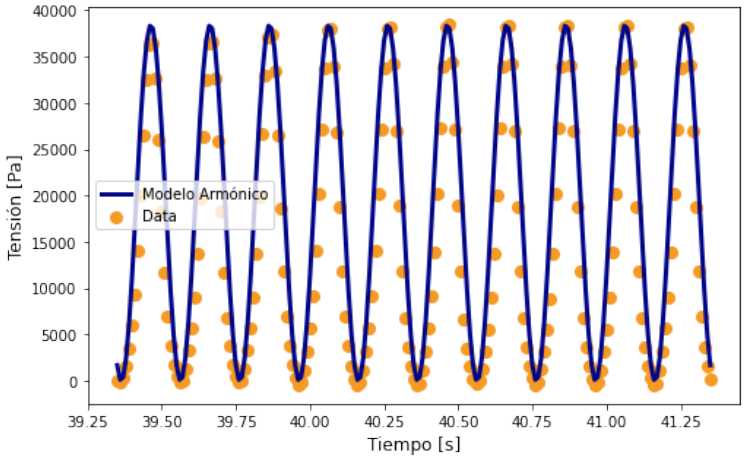


Figura 8: Ajuste del modelo armónico a los datos

Anexo: Códigos

```
#!/usr/bin/env python
# coding: utf-8

# # Tarea 4 - Introducción a la Biomecánica
# ### Pregunta 1

# Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from os.path import join
from scipy import integrate
from decimal import Decimal
import scipy.optimize as sp

# Paths
harmonic_path = join('data', 'harmonic.csv')
quasistatic_path = join('data', 'quasistatic.csv')

# Data windows
cycle_h = {             # Data windows for the harmonic experiment
    '0': 3935,          # Start of the experiment
    'f': 4136,          # End of the experiment
    '1': 3956           # End of the first oscillation
}
cycle_q = {             # Data windows for the quasistatic experiment
    '0': 3870,          # Start of the experiment
    'f': 6571           # End of the experiment
}

# Constants
g = 9.81                # Gravity
A = 7 * 10 ** (-6)      # Area of the tissue
L_0 = 2.2               # Initial length of the tissue

# Graphs
naranjo = '#F59A23'
azul = '#010589'
rojo = '#E40C2B'
verde = '#00D300'
fig_size = (8, 5)

# Import data
harmonic = pd.read_csv(harmonic_path, skiprows=[0, 1, 2, 4])
quasistatic = pd.read_csv(quasistatic_path, skiprows=[0, 1, 2, 4])
quasistatic.head()

# Remove spaces from column names
quasistatic.rename(columns=lambda x: x.strip(), inplace=True)
harmonic.rename(columns=lambda x: x.strip(), inplace=True)

# Check data
quasistatic.loc[:, 'Elapsed Time']

# Data columns
time_q = quasistatic.loc[:, 'Elapsed Time']
disp_q = quasistatic.loc[:, 'Disp']
load_q = quasistatic.loc[:, 'Load 3']
load_q = load_q - load_q[cycle_q['0']]          # Normalized load

str_q = (load_q * g / 1000) / A                 # Stress
def_q = (disp_q - disp_q[cycle_q['0']]) / L_0   # Normalized deformation

time_h = harmonic.loc[:, 'Elapsed Time']
disp_h = harmonic.loc[:, 'Disp']
load_h = harmonic.loc[:, 'Load 3']
load_h = load_h - load_h[cycle_h['0']]          # Normalized load

str_h = (load_h * g / 1000) / A                 # Stress
def_h = (disp_h - disp_h[cycle_h['0']]) / L_0   # Normalized deformation

# Plot quasistatic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    time_q,
    disp_q,
    color=naranjo,
    linewidth=3
)
plt.xlabel('Time [s]', fontsize=12)
plt.ylabel('Displacement [mm]', fontsize=12)
# plt.title('Quasistatic (Displacement vs. Time)', fontsize=14)
plt.show()

# Plot quasistatic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    time_q[cycle_q['0']:cycle_q['f']],
    disp_q[cycle_q['0']:cycle_q['f']],
    color=naranjo,
    linewidth=3
)
plt.xlabel('Time [s]', fontsize=12)
plt.ylabel('Displacement [mm]', fontsize=12)
# plt.title('Quasistatic (Displacement vs. Time)', fontsize=14)
plt.show()

# Plot quasistatic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    disp_q,
    load_q,
    color=naranjo,
    linewidth=3
)
# End of the experiment
'''
plt.plot(
    [disp_q[cycle_q['f']] for _ in range(cycle_q['0'], cycle_q['f'])],
    load_q[cycle_q['0']:cycle_q['f']],
    color=naranjo,
    linewidth=3
)
'''
plt.xlabel('Displacement [mm]', fontsize=12)
plt.ylabel('Load [g]', fontsize=12)
# plt.title('Quasistatic (Load vs. Displacement)', fontsize=14)
plt.show()

# Plot quasistatic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    disp_q[cycle_q['0']:cycle_q['f']],
    load_q[cycle_q['0']:cycle_q['f']],
    color=naranjo,
    linewidth=3
)
plt.xlabel('Displacement [mm]', fontsize=12)
plt.ylabel('Load [g]', fontsize=12)
# plt.title('Quasistatic (Load vs. Displacement)', fontsize=14)
plt.show()

# Plot quasistatic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    def_q,
    str_q,
    color=naranjo,
    linewidth=3
)
plt.xlabel(r'Deformación $\epsilon$', fontsize=12)
plt.ylabel('Tensión [Pa]', fontsize=12)
# plt.title('Quasistatic (Load vs. Displacement)', fontsize=14)
plt.show()

# Plot quasistatic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    def_q[cycle_q['0']:cycle_q['f']],
    str_q[cycle_q['0']:cycle_q['f']],
    color=naranjo,
    linewidth=3
)
plt.xlabel(r'Deformación $\epsilon$', fontsize=12)
plt.ylabel('Tensión [Pa]', fontsize=12)
# plt.title('Quasistatic (Load vs. Displacement)', fontsize=14)
plt.show()


# #### Caso Armónico

# Plot harmonic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    time_h,
    disp_h,
    color=naranjo,
    linewidth=3
)
plt.xlabel('Time [s]', fontsize=12)
plt.ylabel('Displacement [mm]', fontsize=12)
# plt.title('Harmonic (Displacement vs. Time)', fontsize=14)
plt.show()

# Plot harmonic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    time_h[cycle_h['0']:cycle_h['f']],
    disp_h[cycle_h['0']:cycle_h['f']],
    color=naranjo,
    linewidth=3
)
plt.xlabel('Time [s]', fontsize=12)
plt.ylabel('Displacement [mm]', fontsize=12)
# plt.title('Harmonic (Displacement vs. Time)', fontsize=14)
plt.show()

# Plot harmonic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    disp_h,
    load_h,
    color=naranjo,
    linewidth=3
)
plt.xlabel('Displacement [mm]', fontsize=12)
plt.ylabel('Load [g]', fontsize=12)
# plt.title('Harmonic (Load vs. Displacement)', fontsize=14)
plt.show()

# Plot harmonic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    disp_h[cycle_h['0']:cycle_h['f']],
    load_h[cycle_h['0']:cycle_h['f']],
    color=naranjo,
    linewidth=3
)
plt.xlabel('Displacement [mm]', fontsize=12)
plt.ylabel('Load [g]', fontsize=12)
# plt.title('Harmonic (Load vs. Displacement)', fontsize=14)
plt.show()
```

```
# Plot harmonic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    def_h,
    str_h,
    color=naranjo,
    linewidth=3
)
plt.xlabel(r'Deformación␣$\epsilon$', fontsize=12)
plt.ylabel('Tensión␣[Pa]', fontsize=12)
plt.show()

# Plot harmonic data
fig = plt.figure(figsize=fig_size)
plt.plot(
    def_h[cycle_h['0']:cycle_h['f']],
    str_h[cycle_h['0']:cycle_h['f']],
    color=naranjo,
    linewidth=3
)
plt.xlabel(r'Deformación␣$\epsilon$', fontsize=12)
plt.ylabel('Tensión␣[Pa]', fontsize=12)
plt.show()


# ### Pregunta 2

# Stretch as a function of deformation
stretch_q =  def_q + 1

# #### Sólido Neohookeano incompresible

# Neo-Hookean model
def neo_hookean(c, lam):
    return c * (lam ** 2 - 1 / lam)

# Neo-Hookean model error
c_0 = 286

def error_nh(c, x, y):
    return abs(y - neo_hookean(c, x))

# Neo-Hookean model fit
nh_c_fit, nh_cov = sp.leastsq(
    func=error_nh,
    x0=c_0,
    args=(
        stretch_q[cycle_q['0']:cycle_q['f']],
        str_q[cycle_q['0']:cycle_q['f']]
    )
)
mu_nh = nh_c_fit
mu_nh

# Plot Neo-Hookean model fit for quasistatic data
fig = plt.figure(figsize=fig_size)
plt.scatter(
    stretch_q[cycle_q['0']:cycle_q['f']:180],
    str_q[cycle_q['0']:cycle_q['f']:180],
    color=naranjo,
    linewidth=3,
    label='Data'
)
plt.plot(
    stretch_q[cycle_q['0']:cycle_q['f']],
    neo_hookean(nh_c_fit, stretch_q[cycle_q['0']:cycle_q['f']]),
    color=azul,
    linewidth=3,
    label='Modelo␣Neohookeano'
)
plt.legend()
plt.xlabel(r'Estiramiento␣$\lambda$', fontsize=12)
plt.ylabel('Tensión␣[Pa]', fontsize=12)
plt.show()


# #### Sólido de Voight

# Plot deformation as a function of time
fig = plt.figure(figsize=fig_size)
plt.plot(
    time_q[cycle_q['0']:cycle_q['f']],
    def_q[cycle_q['0']:cycle_q['f']],
    color=naranjo,
    linewidth=3
)
plt.xlabel('Tiempo␣[s]', fontsize=12)
plt.ylabel(r'Deformación␣$\epsilon$', fontsize=12)
plt.show()

# Slope of the deformation vs. time curve
global m
m = (def_q[cycle_q['f']] - def_q[cycle_q['0']]) / \
    (time_q[cycle_q['f']] - time_q[cycle_q['0']]) * 1e2
    #(cycle_q['f'] - cycle_q['0'])
print('%.4E' % Decimal(m))

# Voight model
def voight(c, lam):
    mu, E_v = c
    return mu * m + (lam - 1) * E_v

# Voight model error
c_1 = [1, 1]

def error_v(c, x, y):
    return abs(y - voight(c, x))

# Voight model fit
v_c_fit, v_cov_c = sp.leastsq(
    func=error_v,
    x0=c_1,
    args=(
        stretch_q[cycle_q['0']:cycle_q['f']],
        str_q[cycle_q['0']:cycle_q['f']]
    )
)
mu_v, E_v = v_c_fit
mu_v, E_v

# Plot Voight model fit for quasistatic data
fig = plt.figure(figsize=fig_size)
plt.scatter(
    stretch_q[cycle_q['0']:cycle_q['f']:180],
    str_q[cycle_q['0']:cycle_q['f']:180],
    color=naranjo,
    linewidth=3,
    label='Data'
)
plt.plot(
    stretch_q[cycle_q['0']:cycle_q['f']],
    voight(v_c_fit, stretch_q[cycle_q['0']:cycle_q['f']]),
    color=azul,
    linewidth=3,
    label='Sólido␣de␣Voight'
)
plt.legend()
plt.xlabel(r'Estiramiento␣$\lambda$', fontsize=12)
plt.ylabel('Tensión␣[Pa]', fontsize=12)
plt.show()


# #### Sólido lineal estándar

# Linear Standard solid model
def lineal_standard(c, lam):
    mu, E_1, E_2 = c
    tau_e = mu / E_2
    tau_s = mu * (E_1 + E_2) / (E_1 * E_2)
    return E_1 * ((lam - 1) + m * (tau_s - tau_e) + \
                  np.exp(-(lam - 1) / (tau_e * m)) * \
                  m * (tau_e - tau_s))

# Linear Standard solid model error
c_2 = [1e6, 2.1e6, 3.1e6]       # Best fit parameters
# c_2 = [1, 2.1, 3.1]             # Best fit graph

def error_sls(c, x, y):
    return abs(y - lineal_standard(c, x))

# Linear Standard solid model fit
sls_c_fit, sls_cov_c = sp.leastsq(
    func=error_sls,
    x0=c_2,
    args=(
        stretch_q[cycle_q['0']:cycle_q['f']],
        str_q[cycle_q['0']:cycle_q['f']],
    )
)
mu_sls, E_1, E_2 = sls_c_fit
tau_e = mu_sls / E_2
tau_s = mu_sls * (E_1 + E_2) / (E_1 * E_2)
print(mu_sls, tau_e, tau_s, E_1)

# Plot Linear Standard solid model fit for quasistatic data
fig = plt.figure(figsize=fig_size)
plt.scatter(
    stretch_q[cycle_q['0']:cycle_q['f']:180],
    str_q[cycle_q['0']:cycle_q['f']:180],
    color=naranjo,
    linewidth=3,
    label='Data'
)
plt.plot(
    stretch_q[cycle_q['0']:cycle_q['f']],
    lineal_standard(sls_c_fit, stretch_q[cycle_q['0']:cycle_q['f']]),
    color=azul,
    linewidth=3,
    label='Sólido␣lineal␣estándar'
)
plt.legend()
plt.xlabel(r'Estiramiento␣$\lambda$', fontsize=12)
plt.ylabel('Tensión␣[Pa]', fontsize=12)
plt.show()

# Linear Standard solid model
# Using the stress as a function of time
'''
def lineal_standard(c, t):
    mu, E_1, E_2 = c
    tau_e = mu / E_2
    tau_s = mu * (E_1 + E_2) / (E_1 * E_2)
    lambda_2 = stretch_q[round(t * 100)]
    return (lambda_2 - 1) * E_1 * (1 - (1 - tau_s / tau_e) * np.exp(-t / tau_e))

c_2 = [2, 2, 2]

def error_sls(c, x, y):
    return abs(y - lineal_standard(c, x))

sls_c_fit, sls_cov_c = sp.leastsq(
    func=error_sls,
    x0=c_2,
    args=(
        time_q[cycle_q['0']:cycle_q['f']],
```

```
        str_q[cycle_q['0']:cycle_q['f']],
    )
)
mu_sls, E_1, E_2 = sls_c_fit
tau_e = mu_sls / E_2
tau_s = mu_sls * (E_1 + E_2) / (E_1 * E_2)
print(mu_sls, tau_e, tau_s)

fig = plt.figure(figsize=fig_size)
plt.scatter(
    stretch_q[cycle_q['0']:cycle_q['f']:180],
    str_q[cycle_q['0']:cycle_q['f']:180],
    color=naranjo,
    linewidth=3,
    label='Data'
)
plt.plot(
    stretch_q[cycle_q['0']:cycle_q['f']],
    lineal_standard(sls_c_fit, time_q[cycle_q['0']:cycle_q['f']]),
    color=azul,
    linewidth=3,
    label='Sólido lineal estándar'
)
plt.legend()
plt.xlabel(r'Estiramiento $\lambda$', fontsize=12)
plt.ylabel('Tensión [Pa]', fontsize=12)
plt.show()
'''
pass


# #### Comparación de modelos

# Plot all models for quasistatic data
fig = plt.figure(figsize=fig_size)
plt.scatter(
    stretch_q[cycle_q['0']:cycle_q['f']:180],
    str_q[cycle_q['0']:cycle_q['f']:180],
    color=naranjo,
    linewidth=3,
    label='Data'
)
plt.plot(
    stretch_q[cycle_q['0']:cycle_q['f']],
    voight(v_c_fit, stretch_q[cycle_q['0']:cycle_q['f']]),
    color=verde,
    linewidth=3,
    label='Sólido de Voight'
)
plt.plot(
    stretch_q[cycle_q['0']:cycle_q['f']],
    lineal_standard(sls_c_fit, stretch_q[cycle_q['0']:cycle_q['f']]),
    color=rojo,
    linewidth=3,
    label='Sólido lineal estándar'
)
plt.plot(
    stretch_q[cycle_q['0']:cycle_q['f']],
    neo_hookean(nh_c_fit, stretch_q[cycle_q['0']:cycle_q['f']]),
    color=azul,
    linewidth=3,
    label='Neohookeano incompresible'
)
plt.legend()
plt.xlabel(r'Estiramiento $\lambda$', fontsize=12)
plt.ylabel('Tensión [Pa]', fontsize=12)
plt.show()


# ### Pregunta 3

# Plot first oscillation stress vs. deformation
fig = plt.figure(figsize=fig_size)
plt.plot(
    def_h[cycle_h['0']:cycle_h['1']],
    str_h[cycle_h['0']:cycle_h['1']],
    color=naranjo,
    linewidth=3
)
plt.xlabel(r'Deformación $\epsilon$', fontsize=12)
plt.ylabel('Tensión [Pa]', fontsize=12)
plt.show()

# Integrate using Simpson's rule
# https://en.wikipedia.org/wiki/Simpson%27s_rule
# https://stackoverflow.com/questions/17602076/how-do-i-integrate-two-1-d-data-arrays-in-python

e_loss = integrate.simpson(
      str_h[cycle_h['0']:cycle_h['1']],
      def_h[cycle_h['0']:cycle_h['1']],
      axis=-1,
      even='avg'
   )
print(f'Energía disipada:', '%.4E' % Decimal(e_loss))

# Plot first oscillation deformation vs. time
fig = plt.figure(figsize=fig_size)
plt.plot(
    time_h[cycle_h['0']:cycle_h['1']],
    def_h[cycle_h['0']:cycle_h['1']],
    color=naranjo,
    linewidth=3
)
plt.xlabel('Tiempo [s]', fontsize=12)
plt.ylabel(r'Deformación $\epsilon$', fontsize=12)
plt.show()

# Define constants
# sigma_0: stress amplitude
# epsilon_0: deformation amplitude
# omega: angular frequency

global sigma_0, epsilon_0, omega

sigma_0 = max(str_h[cycle_h['0']:cycle_h['f']]) / 2
epsilon_0 = max(def_h[cycle_h['0']:cycle_h['1']]) / 2
omega = 10 * np.pi

# Energy dissipated using the Voight model
epsilon_0 ** 2 * np.pi * mu_v * omega

# Energy dissipated using the linear standard model
epsilon_0 ** 2 * np.pi * E_1 * omega * \
    (tau_s - tau_e) / (1 + omega ** 2 * tau_e ** 2)


# ### Bonus

# Plot first oscillation stress vs. time
fig = plt.figure(figsize=fig_size)
plt.plot(
    time_h[cycle_h['0']:cycle_h['1']],
    str_h[cycle_h['0']:cycle_h['1']],
    color=naranjo,
    linewidth=3
)
plt.xlabel('Tiempo [s]', fontsize=12)
plt.ylabel('Tensión [Pa]', fontsize=12)
plt.show()

# Harmonic deformation model
def harmonic_model(c, t):
    delta = c
    return sigma_0 * (1 + np.cos(omega * t + delta))

# Harmonic deformation model error
c_h = 1

def error_h(c, x, y):
    return abs(y - harmonic_model(c, x))

# Harmonic deformation model fit
h_c_fit, h_cov = sp.leastsq(
    func=error_h,
    x0=c_h,
    args=(
        time_h[cycle_h['0']:cycle_h['f']],
        str_h[cycle_h['0']:cycle_h['f']]
    )
)
delta_h = h_c_fit

# Plot harmonic model fit for harmonic data
fig = plt.figure(figsize=fig_size)
plt.scatter(
    time_h[cycle_h['0']:cycle_h['f']],
    str_h[cycle_h['0']:cycle_h['f']],
    color=naranjo,
    linewidth=3,
    label='Data'
)
plt.plot(
    time_h[cycle_h['0']:cycle_h['f']],
    harmonic_model(h_c_fit, time_h[cycle_h['0']:cycle_h['f']]),
    color=azul,
    linewidth=3,
    label='Modelo Armónico'
)
plt.legend()
plt.xlabel('Tiempo [s]', fontsize=12)
plt.ylabel('Tensión [Pa]', fontsize=12)
plt.show()

delta_h


# #### Hecho con :heart: por Iván Vergara Lam
```