PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
Escuela de Ingeniería
ICE/IBM2020 Introducción a la Biomecánica

Tarea 5

Iván Vergara Lam
13 de junio del 2023
Tiempo dedicado: 14 horas

Problema 1. Traccionamiento del ACL

El artículo de Peña et al. (2006) entrega un modelo constitutivo para el ligamento cruzado anterior (ACL) y sus parámetros ajustados. Los datos fueron utilizados para estudiar las deformaciones en una simulación computacional del modelo. En particular, se modificó el parámetro *bulk modulus* según la información entregada por el modelo, siendo definido como $\kappa = 146{,}41$ MPa. Posteriormente, se aplicó una condición de borde de empotramiento a la superficie inferior del ligamento, y un desplazamiento prescrito de 3.6 mm a 69 nodos en la parte superior.

a) En primer lugar, se generó un gráfico de fuerza vs. desplazamiento a partir de la simulación realizada. Para ello, se integraron las tensiones en los nodos en los que se impuso el desplazamiento mediante la función `Integrate` del *software* FEBio. El resultado se muestra en la Figura 1.

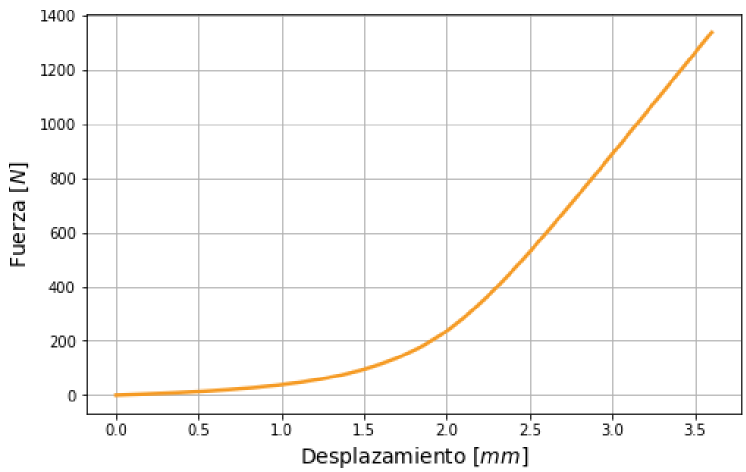

Figura 1: Gráfico de Fuerza vs. Desplazamiento

b) De igual manera, se generó el gráfico del campo de tensiones y el campo vectorial de direcciones principales. Los resultados se muestran a continuación.

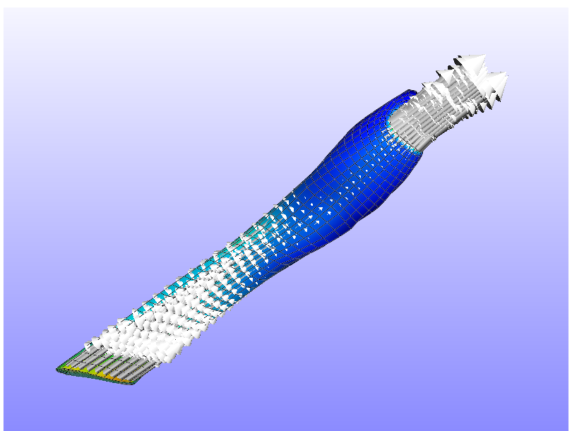

Figura 2: Campo de tensiones y campo vectorial en dirección principal 1

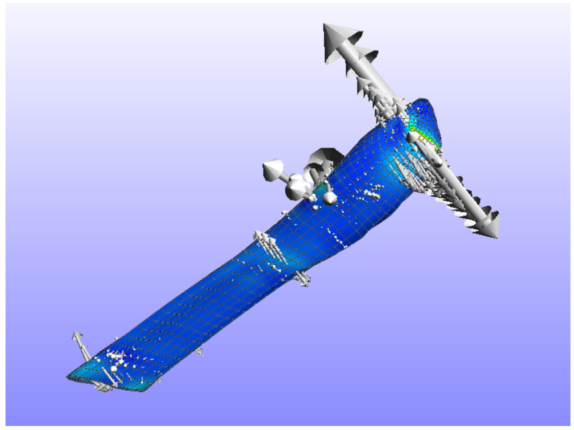

Figura 3: Campo de tensiones y campo vectorial en dirección principal 2

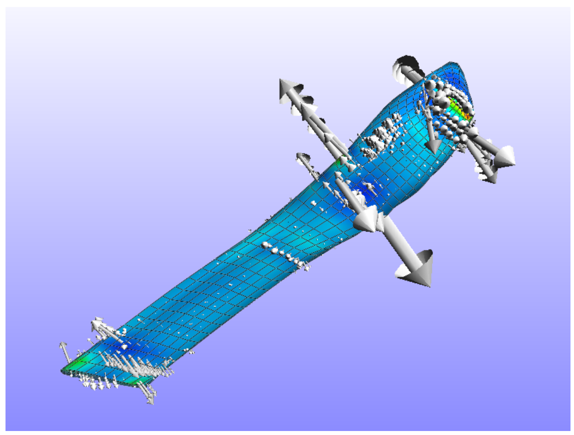

Figura 4: Campo de tensiones y campo vectorial en dirección principal 3

c) Similar al apartado anterior, se generó el gráfico del campo de deformación lagreangeana y el campo vectorial de direcciones principales asociados. Los resultados se muestran a continuación.

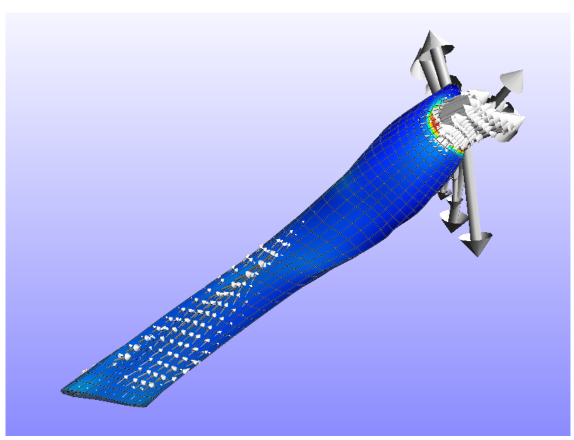

Figura 5: Deformación lagreangeana y campo vectorial en dirección principal 1

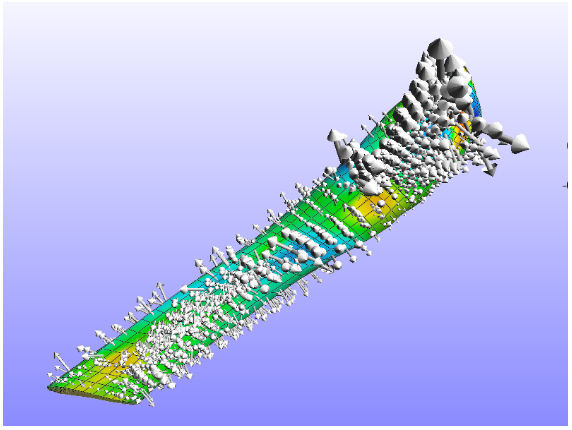

Figura 6: Deformación lagreangeana y campo vectorial en dirección principal 2

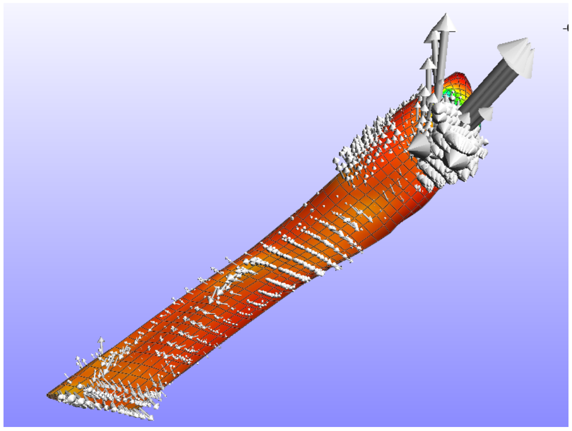

Figura 7: Deformación lagreangeana y campo vectorial en dirección principal 3

d) Por último, se elaboro una animación que muestra cómo el tejido se deforma a medida que se aumenta el desplazamiento vertical. El resultado se puede visualizar en el repositorio de GitHub.

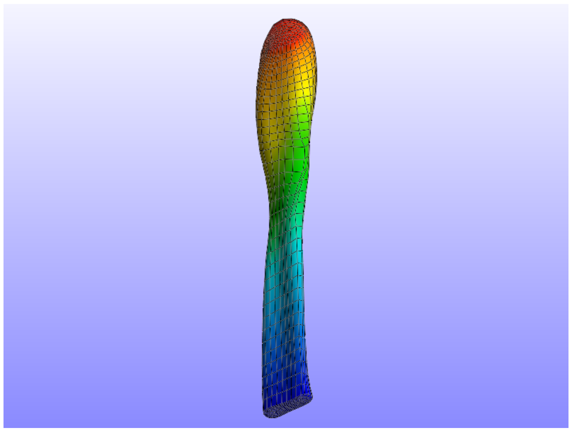

Figura 8: Máxima deformación del tejido en la animación.

Problema 2. Simulación biomecánica de ventilador mecánico

La ecuación de movimiento que simula el comportamiento de un pulmón conectado a un ventilador mecánico corresponde a la expresión (1).

$$P_{\mathrm{MV}}(t) - P_{\mathrm{PEEP}} = \frac{V(t)}{C_{\mathrm{rs}}} + R_{\mathrm{aw}}\dot{V}(t) \tag{1}$$

Donde V el volumen pulmonar, $\dot{V} = \frac{\mathrm{d}V}{\mathrm{d}t}$ es el flujo, P_{MV} (2) es la presión aplicada por el ventilador mecánico, P_{PEEP} es la presión positiva al fin de expiración que ejerce el ventilador, C_{rs} es la *compliance* pulmonar y R_{aw} es la resistencia de la vía aérea.

$$P_{\mathrm{MV}}(t) = \begin{cases} P_{\mathrm{peak}} & \text{si} \quad \text{mód} \frac{1}{RR} < IT \\ P_{\mathrm{PEEP}} & \text{e.o.c} \end{cases} \tag{2}$$

Donde RR es la frecuencia respiratoria, IT es el tiempo inspiratorio y P_{PEEP} es la presión *peak* inspiratoria

i) Se busca simular el caso de 5 ciclos respiratorios de ventilaciónn mecánica de un paciente bajo el modo presión control. Para lo anterior, se debe resolver numéricamente la ecuación (1) considerando los valores de los parámetros del modelo expuestos en la Tabla 1.

Parámetro	Valor en pulmones sanos
C_{rs}	0.5 L/cmH_2O
R_{aw}	2 cmH_2Os/L
P_{PEEP}	5 cmH_2O
P_{peak}	10 cmH_2O
IT	3 s
RR	10 bpm

Tabla 1: Parámetros de respiración sana

Utilizando los parámetros del modelo mencionados y la función `numpy.odeint` en Python se obtuvo un *array* para los valores del voltaje. El resultado obtenido se presenta en la Figura 9, que presenta las señales de P_{MV}, V y $\dot{V}$.

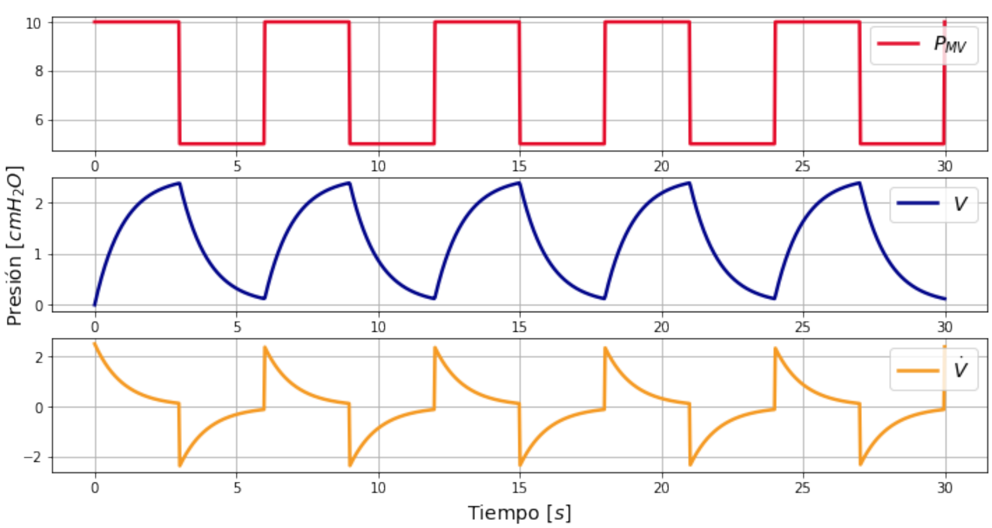

Figura 9: 5 ciclos respiratorios de un paciente bajo el modo de presión control

ii) Se busca repetir el procedimiento realizado anteriormente, pero considerando los siguientes valores asociados al síndrome de dificultad respiratoria aguda (ARDS) y enfermedad pulmonar obstructiva crónica (COPD) expuestos en la Tabla 2.

Parámetro	Valor en ARDS	Valor en COPD
C_{rs}	0.35 L/cmH_2O	0.65 L/cmH_2O
R_{aw}	1.2 cmH_2Os/L	2.5 cmH_2Os/L
P_{PEEP}	5 cmH_2O	5 cmH_2O
P_{peak}	10 cmH_2O	10 cmH_2O
IT	3 s	3 s
RR	10 bpm	10 bpm

Tabla 2: Parámetros de respiración en condición patológica

Al igual que en el primer caso, se obutvo *array* para los valores del voltaje. El resultado obtenido para los casos de ARDS y COPD se presentan en las Figuras 10 y 11, respectivamente. Ambos presentan las señales de P_{MV}, V y $\dot{V}$.

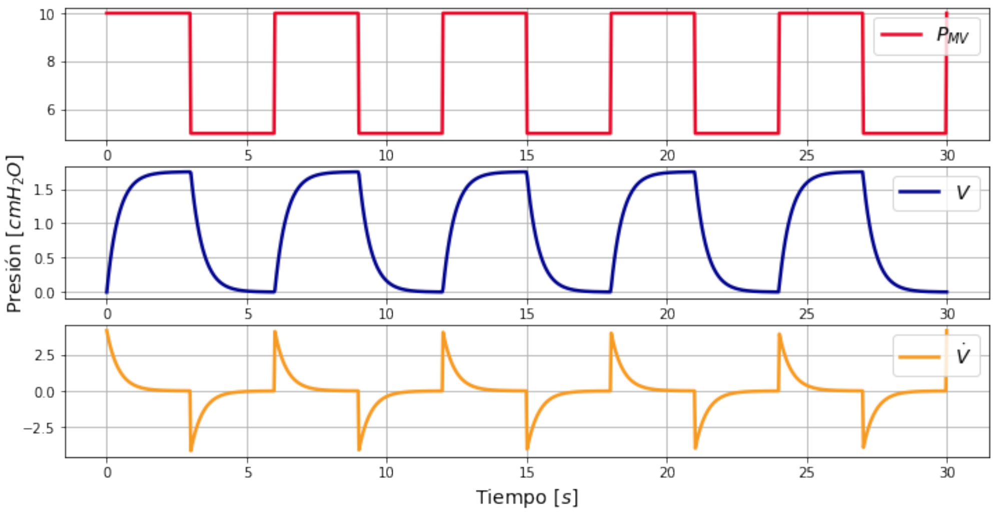

Figura 10: 5 ciclos respiratorios de un paciente con ARDS

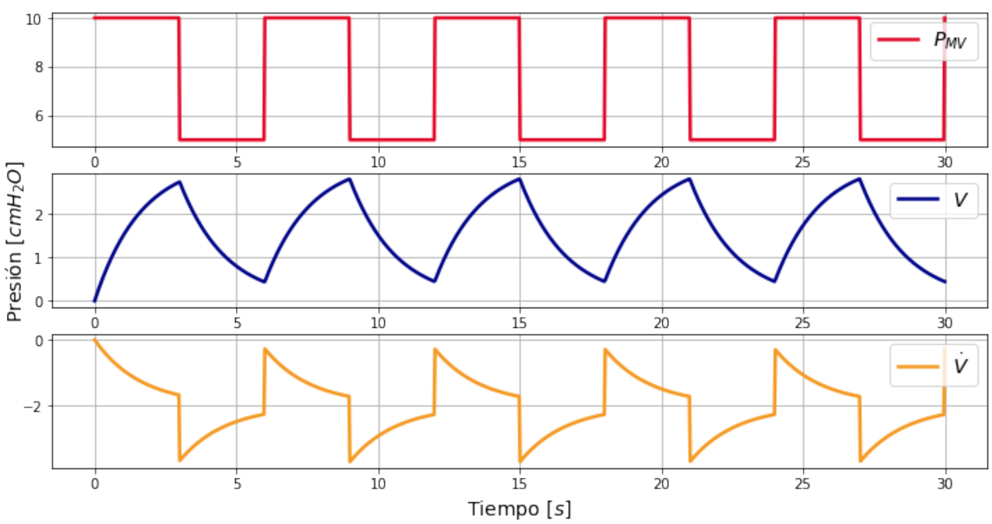

Figura 11: 5 ciclos respiratorios de un paciente con COPD

iii) Para cada caso (sano, ARDS y COPD) se calcularon los indicadores clínicos volumen minuto V_{m}, volumen corriente o tidal V_{t} y constante de tiempo $RC = R_{\mathrm{aw}}C_{\mathrm{rs}}$. Los resultados se muestran en la Tabla 3.

	Sano	ARDS	COPD
V_{t} [L]	2.381	1.749	2.806
V_{m} [L]	23.811	17.486	28.0612
RC [s]	1.0	0.42	1.625

Tabla 3: Indicadores clínicos

Se puede evidenciar como pacientes con síndrome de dificultad respiratoria aguda (ARDS) poseen respiraciones más cortas, pero con volúmenes inhalados menores. En cuanto a los pacientes con enfermedad pulmonar obstructiva crónica (COPD) presenta respiraciones más prolongadas, pero con volúmenes inhalados mayores.

Referencias

Peña, E., Calvo, B., Martínez, M., & Doblaré, M. (2006). A three-dimensional finite element analysis of the combined behavior of ligaments and menisci in the healthy human knee joint. *Journal of Biomechanics*, *39*(9), 1686-1701. https://doi.org/10.1016/j.jbiomech.2005.04.030

Anexo: Códigos

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

# # Tarea 5 - Introducción a la Biomecánica

import scipy.integrate as sp
import numpy as np
import matplotlib.pyplot as plt
from scipy import integrate

# Graphs
naranjo = '#F59A23'
azul = '#010589'
rojo = '#E40C2B'
verde = '#00D300'
grosor = 2.5
fig_size = (12, 6)
fig_size_2 = (8, 5)


# ### Pregunta 1

# Arrays
disp = (
    0,
    0.36,
    0.522481522481,
    0.678044678044,
    0.826985826985,
    0.969585969585,
    1.10611,
    1.23683,
    1.36198,
    1.48181,
    1.59653,
    1.72322,
    1.8613,
    1.99938,
    2.13159,
    2.27491,
    2.41213,
    2.54935,
    2.69744,
    2.88211,
    3.09704,
    3.31198,
    3.55592,
    3.6
)

stress = (
    0,
    8.5815,
    13.7758,
    19.84,
    27.0188,
    35.6318,
    46.0235,
    58.5594,
    73.6726,
    91.7096,
    113.097,
    142.435,
    183.017,
    235.603,
    299.175,
    380.474,
    467.603,
    561.25,
    666.528,
    801.191,
    960.496,
    1121.19,
    1303.88,
    1336.85
)

# Plot Fuerza vs. Desplazamiento
plt.figure(figsize=fig_size_2)
plt.plot(
    disp,
    stress,
    label='Datos experimentales',
    color=naranjo,
    linewidth=grosor
)
plt.xlabel(r'Desplazamiento $[mm]$', fontsize=14)
plt.ylabel(r'Fuerza $[N]$', fontsize=14)
plt.grid()
plt.show()


# ### Pregunta 2

# #### Ítem I

# Parameters
global C_rs, R_aw, P_peep, P_peak, IT, RR
C_rs = 0.5
R_aw = 2
P_peep = 5
P_peak = 10
IT = 3
RR = 10 / 60

# Square wave
def p_mv(t):
    if t % (1 / RR) < IT:
        return P_peak
    return P_peep

# Derivada control
def v_dot(v, t):
    return (p_mv(t) - P_peep - v / C_rs) / R_aw

# Volumen control
t = np.linspace(0, 5 / RR , 1000)
v_0 = 0
v_sol = sp.odeint(v_dot, v_0, t)

# Plot control
fig, ax = plt.subplots(3, 1, figsize=fig_size)
ax[0].plot(
    t, [p_mv(t[i]) for i in range(len(t))],
    label=r'$P_{MV}$',
    linewidth=grosor,
    color=rojo
)
ax[1].plot(
    t, v_sol,
    label=r'$V$',
    linewidth=grosor,
    color=azul
)
ax[2].plot(
    t, [v_dot(v_sol[i], t[i]) for i in range(len(t))],
    label=r'$\dot{V}$',
    linewidth=grosor,
    color=naranjo
)

ax[2].set_xlabel(r'Tiempo $[s]$', fontsize=14)
ax[1].set_ylabel(r'Presión $[cmH_2O]$', fontsize=14)

ax[0].legend(loc='upper right', fontsize=14)
ax[1].legend(loc='upper right', fontsize=14)
ax[2].legend(loc='upper right', fontsize=14)

ax[0].grid()
ax[1].grid()
ax[2].grid()

plt.show()


# #### Ítem II

# Parameters
global C_rs_a, R_aw_a, P_peep_a, P_peak_a, IT_a, RR_a
global C_rs_c, R_aw_c, P_peep_c, P_peak_c, IT_c, RR_c

C_rs_a = 0.35
R_aw_a = 1.2
P_peep_a = 5
P_peak_a = 10
IT_a = 3
RR_a = 10 / 60

C_rs_c = 0.65
R_aw_c = 2.5
P_peep_c = 5
P_peak_c = 10
IT_c = 3
RR_c = 10 / 60

# Square wave
def p_mv_a(t):
    if t % (1 / RR_a) < IT_a:
        return P_peak_a
    return P_peep_a

def p_mv_c(t):
    if t % (1 / RR_c) < IT_c:
        return P_peak_c
    return P_peep_c

# Definición de derivadas ARDS y COPD
def v_dot_a(v, t):
    return (p_mv_a(t) - P_peep_a - v / C_rs_a) / R_aw_a

def v_dot_c(v, t):
    return (p_mv_c(t) - P_peep_c - v / C_rs_c) / R_aw_c

# Volumen ARDS y COPD
t_a = np.linspace(0, 5 / RR_a , 1000)
v_0_a = 0
v_sol_a = sp.odeint(v_dot_a, v_0_a, t_a)

t_c = np.linspace(0, 5 / RR_c , 1000)
v_0_c = 0
v_sol_c = sp.odeint(v_dot_c, v_0_c, t_c)

# Plot ARDS
fig, ax = plt.subplots(3, 1, figsize=fig_size)
ax[0].plot(
    t_a, [p_mv_a(t_a[i]) for i in range(len(t_a))],
    label=r'$P_{MV}$',
    linewidth=grosor,
    color=rojo
)
ax[1].plot(
    t_a, v_sol_a,
```

```
    label=r'$V$',
    linewidth=grosor,
    color=azul
)
ax[2].plot(
    t_a, [v_dot_a(v_sol_a[i], t_a[i]) for i in range(len(t_a))],
    label=r'$\dot{V}$',
    linewidth=grosor,
    color=naranjo
)

ax[2].set_xlabel(r'Tiempo $[s]$', fontsize=14)
ax[1].set_ylabel(r'Presión $[cmH_2O]$', fontsize=14)

ax[0].legend(loc='upper right', fontsize=14)
ax[1].legend(loc='upper right', fontsize=14)
ax[2].legend(loc='upper right', fontsize=14)

ax[0].grid()
ax[1].grid()
ax[2].grid()

plt.show()

# Plot COPD
fig, ax = plt.subplots(3, 1, figsize=fig_size)
ax[0].plot(
    t_c, [p_mv_c(t_c[i]) for i in range(len(t_c))],
    label=r'$P_{MV}$',
    linewidth=grosor,
    color=rojo
)
ax[1].plot(
    t_c, v_sol_c,
    label=r'$V$',
    linewidth=grosor,
    color=azul
)
ax[2].plot(
    t_c, [v_dot_c(v_sol_c[i], t_c[i]) for i in range(len(t_c))] - \
        max([v_dot_c(v_sol_c[i], t_c[i]) for i in range(len(t_c))]),
    label=r'$\dot{V}$',
    linewidth=grosor,
    color=naranjo
)

ax[2].set_xlabel(r'Tiempo $[s]$', fontsize=14)
ax[1].set_ylabel(r'Presión $[cmH_2O]$', fontsize=14)

ax[0].legend(loc='upper right', fontsize=14)
ax[1].legend(loc='upper right', fontsize=14)
ax[2].legend(loc='upper right', fontsize=14)

ax[0].grid()
ax[1].grid()
ax[2].grid()

plt.show()


# #### Ítem III

# Calculo de indicadores clínicos
# Volumen tidal
v_t = max(v_sol) - min(v_sol)
v_t_a = max(v_sol_a) - min(v_sol_a)
v_t_c = max(v_sol_c) - min(v_sol_c)

# Volumen minuto
v_m = v_t * 10
v_m_a = v_t_a * 10
v_m_c = v_t_c * 10

# Constante de tiempo
rc = R_aw * C_rs
rc_a = R_aw_a * C_rs_a
rc_c = R_aw_c * C_rs_c

print(v_t, v_t_a, v_t_c)
print(v_m, v_m_a, v_m_c)
print(rc, rc_a, rc_c)
```