

# Soft Robotics Materials Database and experimental tensile test data fitting tutorial



**Abstract:** This tutorial will provide an overview of the open source Soft Robotics Materials Database and its web App designed to aid material selection for soft Robotics application. It will also include an introduction to modelling constitutive models for hyperelastic materials from experimental tensile test data. Attendees will be able to practice experimental data fitting using a non linear optimization technique by coding and testing their own Python script under a Jupyter notebook.



Constitutive Models    Materials Comparison    Setup & Characterisation    About    GitHub



## Constitutive Models

Material: RTV615

VENDOR MATERIAL INFO    DOWNLOAD RAW DATA

PARAMETER	INFO
Specimen Thickness (mm)	3
Specimen Width (mm)	6
Specimen Length (mm)	33
Specimen Cross-section (cm <sup>2</sup> )	0.18
Standard	ASTM D412
Type of test	Pull-to-failure
Speed (mm/min)	450
Machine	Instron 5569
Load Cell	1 kN - Instron 2525-806
Grippers	Instron 2710-010
Strain Measurement	G10ba1
Strain calculation	Derived from the machine crosshe
Lab	Univ. Savoie Mont Blanc - SYMME
Material supplier	Homentive
Material	RTV615

Constitutive model: Ogden

Order: 3

Principal True Cauchy Stress

$$\sigma = \sum_{p=1}^n 2\mu_p \left[ \lambda^{(\alpha_p-1)} - \lambda^{-\left(\frac{1}{2}\alpha_p+1\right)} \right]$$

Model selection: Manual/Auto

Data type: True/Engineering

**FIT DATA**

Ogden parameters: (on  $\epsilon$  True data range [0.00', '1.08'])

$\mu_1$	$\mu_2$	$\mu_3$	$\alpha_1$	$\alpha_2$	$\alpha_3$
1.6236	-2.7927	1.2668	4.6671	4.8841	5.8798

AIC: -777.4

Selected strain  $\epsilon$ : [0.00', '1.08']

The data is fitted on the selected strain range. Adjust the range accordingly to your application before clicking on the 'Fit data' button to obtain a better accuracy of the model.

```
[1]: # Pandas and table
import pandas as pd
# Optimization
from scipy.optimize import least_squares
import numpy as np

[*]: #####
# OPTIMIZATION
#####

def OgdenModel(trueStrain, parameters, order):
    """Ogden hyperelastic model (incompressible material under uniaxial tension)
    Uses true strain and true stress data"""

    # parameter is a 1D array: [mu0,mu1,...,mun,alpha0,alpha1,...,alphan]
    muVec = parameters.reshape(2, order)[0]
    alphaVec = parameters.reshape(2, order)[1]
    lambda = np.exp(trueStrain)
    # broadcasting method to speed up computation
    lambda = lambda[np.newaxis, :]
    muVec = muVec[:,order, np.newaxis]
    alphaVec = alphaVec[:,order, np.newaxis]

    trueStress = np.sum(2*muVec*(lambda**(alphaVec - 1) - lambda**(-(1/2)*alphaVec + 1)), axis=0)
    return trueStress

# cost function to calculate the residuals. The fitting function holds the parameter values.
def objectiveFun_Callback(parameters, exp_strain, exp_stress):
    theo_stress = OgdenModel(exp_strain, parameters, order)
    residuals = theo_stress - exp_stress
    return residuals
```

