Practical work: ultrasonic rheology of human cells

D. K. Dysthe, T. Combriat

November 8, 2022

1 Background

The SOE instrument uses ultrasounds (US) to generate oscillatory flows on biological cultures (adherent cells). Because of this flow surrounding them, the cells in the culture will *deform*. For a given stress (τ) the amount of deformation (strain) γ is inversely proportional to their *shear modulus* G:

$$G = \frac{\tau}{\gamma} \tag{1}$$

As they are subjected to a (quasi) sinusoidal regime[1], the induced deformation occurs at the same frequency f_0 than the driving force: the flow. Digital image correlation (DIC) allows us to measure the displacements of each cells at each time step, leading to time-dependent velocity fields alongside the x and y axis: $u_{x/y}(x, y, t)$. As we know the frequency at which the biological material should respond, and using Fourrier transforms, one can recover the amplitude and the phase of the cells' oscillations at this frequency (a process called an homodyne detection):

$$u_{x/y}(x, y, t) \underbrace{\rightarrow}_{FFT} \tilde{u}_{x/y}(x, y, f = f_0). \tag{2}$$

Note that each element of these homodyne displacement fields $\tilde{u}_{x/y}$ are *complex numbers*, they carry information about both the amplitude $|\tilde{u}_{x/y}|$ and the phase $\phi(\tilde{u}_{x/y})$ of the oscillations.

1.1 Homodyne strains

Homodyne strains (cyclic deformations of the material), can be calculated just like the classical way:

$$\tilde{\gamma}_p = \frac{\partial \tilde{u}_x}{\partial x} + \frac{\partial \tilde{u}_y}{\partial y} \tag{3}$$

$$\tilde{\gamma}_s = \frac{\partial \tilde{u}_x}{\partial y} + \frac{\partial \tilde{u}_y}{\partial x} \tag{4}$$

with $\tilde{\gamma}_p$ the homodyne pure strain and $\tilde{\gamma}_s$ the homodyne simple strain.

1.2 Estimating the stress

In order to recover the shear modulus of the probed material, one also needs to estimate the stress at which the cells are subjected to (see Eq. 1). We make the hypothesis that the stress is proportional to the surrounding flow. This flow can be estimated by tracking small particles (tracers) that are seeded in the liquid, a process called particles tracking velocimetry (PTV).

We make the further hypothesis that the flow is homogeneous on the view field, hence for the PTV we recover the time dependent flow field alongside both direction $\delta_{x/y}(t)$. Using an homodyne detection, the component of the flow which is oscillating at the input frequency f_0 can be recovered: $\tilde{\delta}_{x/y}(f = f_0)$. Note that $\tilde{\delta}_x(f = f_0)$ and $\tilde{\delta}_y(f = f_0)$ are complex numbers.

This flow field can be used to normalise the displacement of the probed material. Let us define the normalised homodyne displacement fields:

$$\tilde{u}^r_{x/y} = \frac{\tilde{u}_{x/y}}{\tilde{\delta}_{x/y}} \tag{5}$$

[1] and considering that their response is linear

1.3 Normalised homodyne strains and shear modulus

Finally, one can estimate the shear modulus of the shear material by using these normalised fields (which already contain the stress) to calculate the homodyne strains:

$$\tilde{\gamma}_p^r = \frac{\partial \tilde{u}_x^r}{\partial x} + \frac{\partial \tilde{u}_y^r}{\partial y} \propto \frac{1}{\tilde{G}_p} \tag{6}$$

$$\tilde{\gamma}_s^r = \frac{\partial \tilde{u}_x^r}{\partial y} + \frac{\partial \tilde{u}_y^r}{\partial x} \propto \frac{1}{\tilde{G}_s} \tag{7}$$

With $\tilde{G}_p$ and $\tilde{G}_s$ the shear modulus extracted from the pure and simple strains, respectively. Note that these numbers are also complex.

2 Data analysis

The goal of this practical is to measure the rigidity of two types of cells and to compare them:

- Do all parts of the cells display the same mechanical properties?
- Do the two types of cells have different mechanical properties?

2.1 Pre-requisites

The analysis will be done with `Python3`. Alongside a working and recent installation of Python3, the following packages are also needed:

- `pickle` [2]
- `matplotlib`
- `numpy`
- `math`
- `scipy`
- `os`
- `skimage`

To test that your installation is complete the following code should run with no error in python:

```
import numpy as np
import matplotlib.pyplot as plt
import math as math
import os as os
import pickle
from scipy.ndimage import gaussian_filter
from matplotlib.patches import Ellipse
import skimage.transform
mic_per_pix = 600./1864 ## The scale factor of images
```

2.2 Loading of the data

Data are given as `pickle` files. Pickles are a way to store python objects in permanent storage as binary files. These objects can be reloaded in another Python script, even on another computer.

The first and lengthy part of the analysis (DIC and PTV) have already been performed on the data we acquired. The results are the following pickles:

- `fft_X.pic`
- `fft_Y.pic`

[2] `https://docs.python.org/3/library/pickle.html`

Both of them are 2D numpy, complex arrays, and they contain the normalised homodyne displacement fields: $\tilde{u}^r_{x/y}$.

They can be loaded by, for example:

```
file = open(filepath,"rb")
fft_X = pickle.load(file)
file.close()
```

A good test to see if everything has gone well is to display for instance the amplitude of the homodyne displacements over a phase contrast image took during the experiment.

```
plt.figure(figsize=(10,10))
ax = plt.subplot(121)
plt.imshow(np.abs(fft_X))
plt.imshow(PC_img, cmap="gist_gray",alpha=0.5)
plt.colorbar()
plt.subplot(122,sharex=ax,sharey=ax)
plt.imshow(np.abs(fft_Y))
plt.imshow(PC_img, cmap="gist_gray",alpha=0.5)
plt.colorbar()
plt.show()
```

2.3 Normalised strains computation

Using Eqs. 6 and 7, one can now compute the pure and simple homodynes strains. Note: `np.diff` can be useful for that, but there are a lot of different way to do it.

```
dux_dx = np.diff(fft_X,axis=1,append=0)
duy_dx = np.diff(fft_Y,axis=1,append=0)
dux_dy = np.diff(fft_X,axis=0,append=0)
duy_dy = np.diff(fft_Y,axis=0,append=0)
dux_dx_p_duy_dy = dux_dx + duy_dy
dux_dy_p_duy_dx = dux_dy + duy_dx

# Plot
plt.figure(figsize=(20,10))
ax = plt.subplot(121)
plt.imshow(gaussian_filter(np.abs(dux_dx_p_duy_dy),7),vmax =0.05)
plt.colorbar()
plt.imshow(PC_img, cmap="gist_gray",alpha=0.5)
plt.subplot(122,sharex=ax,sharey=ax)
plt.imshow(gaussian_filter(np.abs(dux_dy_p_duy_dx),7),vmax = 0.05)
plt.colorbar()
plt.imshow(PC_img, cmap="gist_gray",alpha=0.5)
plt.show()
```

You will the "jumpy" aspects of the obtained map. This is a size effect of the DIC algorithm: the algorithm uses square blocks to correlate parts of the images. Because of this, neighboring pixels are not completely independent, in particular, some pixels will have the nearly exact same $\tilde{u}^r_{x/y}$ than their neighbors, leading to problems in the differentiation process. This can be solved by applying a gaussian filter (blur) on the results with a standard deviation of the size of these "non-independent" blocks. Using the 'scipy' gaussian filter, a value of 7 works well. Note that it also means that we are oversampling the data, which can lead to statistical errors in the end (we will come back to that in 2.6).

2.3.1 If you are curious

The fact that some pixels are not independent can be easily seen by performing a 2D FFT on the amplitudes images, as this transformation tells us at which *scale* the information lies. If you have the time, this is interesting to do.

```
plt.figure()
fft = np.fft.fftshift(np.fft.fft2(fft_X))
ax = plt.gca()
plt.imshow(np.log(np.abs(fft)))
circle = Ellipse((len(fft_X[0])/2, len(fft_X)/2), len(fft_X[0])/7*2, len(fft_X)
    /7*2,facecolor="None",edgecolor="r",alpha = 0.2, ls = "-.")
ax.add_patch(circle)
plt.show()
```

2.4 Shear modulus maps computation

Using Eqs. 6 and 7, one can now compute the relative shear modulus, both from the pure and the simple strains. Note that part of the field contains only zeros because the two cameras fields do not overlap completely. This will produce some NaNs (not a number) error during this step. They can be handled with `np.nan_to_num`.

Because of the phases jumps, the produce fields will be even more jumpy than the previous ones, this can be handled with a gaussian blur as described previously. Also, some negative numbers are present in the fields because of that. As a first good approximation these can be handled by taking only the absolute value of the fields.

```
pure_shear_modulus = np.nan_to_num(1/dux_dx_p_duy_dy,posinf=0,neginf=0)
simple_shear_modulus = np.nan_to_num(1/dux_dy_p_duy_dx,posinf=0,neginf=0)

# Plot
plt.figure(figsize=(20,10))
ax = plt.subplot(121)
plt.imshow(gaussian_filter(np.abs(pure_shear_modulus.real),7),vmax =300)
plt.colorbar()
plt.imshow(PC_img, cmap="gist_gray",alpha=0.5)
plt.subplot(122,sharex=ax,sharey=ax)
plt.imshow(gaussian_filter(np.abs(pure_shear_modulus.imag),7),vmax =300)
plt.colorbar()
plt.imshow(PC_img, cmap="gist_gray",alpha=0.5)
plt.show()
```

The real and imaginary parts of these fields should be proportional to the elastic and the viscous components of the shear modulus respectively (G' and G''). The absolute value can be seen as a measure of the rigidity ($|G|$). We will consider only the latter here.

You can plot these fields on top of an image of the cells, is there

2.5 Cells identification

The GFP and RFP fluorescent signals can be used to differentiate the cells of the two different types.

These can be overlayed with the one the field produced before, for instance the rigidity in order to have a visual overview of mechanical properties and cell type at the same type. A `contour` plot can be used to avoid collision of several colorbars.

```
plt.figure(figsize=(10,10))
plt.imshow(gaussian_filter(np.abs(pure_shear_modulus),7),vmax =800)
plt.colorbar()
plt.imshow(PC_img, cmap="gist_gray",alpha=0.5)
plt.contour(GFP,cmap="Greens")
plt.contour(RFP,cmap="Reds")
plt.show()
```

2.6 Mechanotyping of the two different cell type

It is now possible to extract the distributions of shear modulus for the two cell types, using both the fluorescent signals as masks and the fields previously computed. The use of *comprehensive lists* can be useful for that. Note that:

- as the fluorescent images do not have exactly the same FOV than the recovered moduli, only areas where these two information overlap should be considered
- as mentioned in 2.3, the mechanical information we have now is *oversampled*: not every point is independent from its neighbors. These non-independent pixels do not bring any information but will contribute to statistical significance by falsely boosting it. This can be alleviated by considering only the mean of non-independent pixels (undersampling). This process is also known as *binning* in image analysis. We know that pixels in a box of 7×7 pixels are not independent so the fields need to be binned to this amount. The `skimage.transform.rescale` function can be used for this purpose[3].

[3]take care that this function discard the imaginary part of complex images, there are other solutions also

```
binned_GFP = skimage.transform.rescale(GFP,1./7,preserve_range=True)
binned_RFP = skimage.transform.rescale(RFP,1./7,preserve_range=True)
binned_pure_shear_modulus_abs = skimage.transform.rescale(np.abs(pure_shear_modulus),1./7,preserve_range=True)
binned_pure_shear_modulus_real = skimage.transform.rescale(np.abs(pure_shear_modulus.real),1./7,preserve_range=True)
binned_pure_shear_modulus_imag = skimage.transform.rescale(np.abs(pure_shear_modulus.imag),1./7,preserve_range=True)
binned_simple_shear_modulus_abs = skimage.transform.rescale(np.abs(simple_shear_modulus),1./7,preserve_range=True)
binned_simple_shear_modulus_real = skimage.transform.rescale(np.abs(simple_shear_modulus.real),1./7,preserve_range=True)
binned_simple_shear_modulus_imag = skimage.transform.rescale(np.abs(simple_shear_modulus.imag),1./7,preserve_range=True)

rigidity_GFP = binned_pure_shear_modulus_abs[np.logical_and(binned_GFP==1,binned_pure_shear_modulus!=0)]
rigidity_RFP = binned_pure_shear_modulus_abs[np.logical_and(binned_RFP==1,binned_pure_shear_modulus!=0)]

plt.figure()
plt.boxplot([rigidity_GFP,rigidity_RFP],labels=["GFP","RFP"])
```

By using statistical tests[4] what can we say about:

- the moduli extracted from the pure and simple components of the strains?
- the mechanotypes (absolute) of the two types of cells?
- can we link some mechanical properties of the cells, or part of them, to their biological properties?

[4] if you use a t-test, please make sure that all assumptions of this test are satisfied