Force Measurement

ELEC-C1310 Laboratory Exercise in Automation and Control Engineering

1. Introduction

Force is a quantity capable of changing the size, shape, or motion of an object. It is a vector quantity and therefore, it has both direction and magnitude.

The forces acting within and between parts of a mechanism are fundamental to the safety, assembly, and use of any piece of equipment. The measurement of those forces is vital for understanding and monitoring the activity, which mechanism is designed to undertake. To make reliable and accurate measurement of force, it is necessary to use appropriate measurement techniques.

In this laboratory exercise, you will take measurements to determine the material properties of different specimens. You will take experiments to measure the contact force and estimate the modulus of elasticity (Young's modulus) of bulk materials using contact mechanics theory.

1.1. Classical theory of elastic contact

Contact mechanics is the study of the deformation of solids that touch each other at one or more points. The contact theory between elastic bodies determines the contact areas, indentation depths and interaction force between bodies and surfaces of simple geometries. One example of the contact theory is the case of non-adhesive elastic contact between a sphere (here a non-deformable sphere) and an elastic surface as illustrated in Figure 1.

When a sphere with radius R indents a surface to depth d, the area of a spherical cap between the sphere and the surface has base has radius $\boldsymbol{a}$:

$$a = \sqrt{Rd} \tag{1}$$

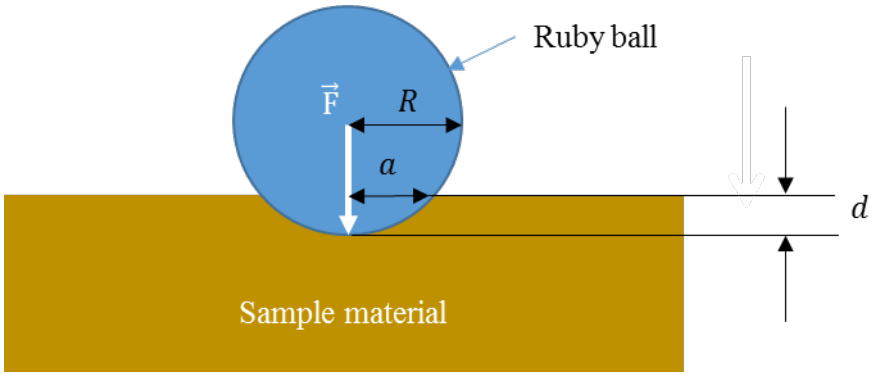


Figure 1. Non-adhesive elastic contact between a non-deformable sphere and an elastic surface.

According to Hertzian theory of non-adhesive elastic contact, the relationship between the applied force (F) and the displacement (d) is:

$$F = \frac{4}{3} E^* R^{1/2} d^{3/2} \quad (2)$$

where E^* is the **effective elastic modulus** defined as:

$$\frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2} \quad (3)$$

E_1, E_2 are the **elastic moduli** and ν_1, ν_2 are the Poisson's ratio of the plane material and the spherical body, respectively.

Note! *Effective elastic modulus* is different from *elastic modulus (* Young's modulus).

1.2. Elastic modulus (Young's modulus) and Poisson's ratio

Elastic modulus (Young's modulus) is a mechanical property of solid materials that measures the stiffness of the material. It defines the relationship between stress (force per unit area) and strain (proportional deformation) in a material.

Young's modulus E is calculated by dividing the tensile stress σ by the tensile strain ε:

$$E = \frac{\sigma}{\varepsilon} = \frac{F/A}{\Delta L/L_0} = \frac{FL_0}{A\Delta L} \quad (4)$$

where F is the exerted force, A is the cross-section of the area that the force is applied over, ΔL is the change in length of the body, and L_0 is the original length of the body. Figure 2 illustrates the case of tensile force, where the force is applied along the axis of deformation.

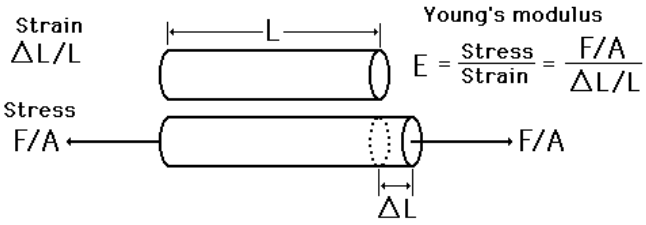


Figure *2*. Definition of Young's modulus[1].

Poisson's ratio is a mechanical property of solid materials. It determines how material expands in directions perpendicular to the direction of compression. Most materials have Poisson's ratio values ranging between 0 and 0.5. Poisson's ratio value for incompressible materials is 0.5.

To have a better understanding of Poisson's effect, consider a cube with side length of L, subjected to tension along the x-axis, as illustrated in Figure 3. The green cube is original unstrained and the red cube denotes the expansion in the x direction by ΔL due to tension, and contraction in the y and z directions by $\Delta L'$.

[1] Adapted from: http://www.markedbyteachers.com/gcse/maths/investigate-the-elastic-properties-of-a-strip-of-metal-hacksaw-blade-and-use-the-results-to-determine-the-value-of-young-s-modulus-of-the-metal.html

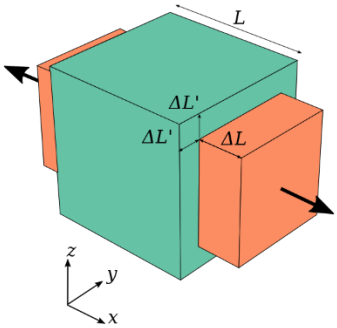


Figure *3*. Definition of Poisson's ratio[2].

1.3. Stress Relaxation

Stress relaxation is the gradual reduction of stress with time at constant strain. Shortly after the strain is applied to a polymer, the stress tends to decrease in response to the same amount of strain. This is referred to as stress relaxation.

1.4. Laboratory Measurements

In this laboratory experiment, you are measuring the force to determine the Young's modulus of elastic materials and observing stress relaxation phenomena. In the experiment, a ruby ball with the diameter of 3 mm indents the surface of an elastic material as illustrated in Figure 1. Ruby is one of the hardest known materials. Since the elastic modulus of ruby ball is very high, the second term on the right hand side of the Equation 3 is negligible:

$$\frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} \tag{5}$$

The Poisson's ratio for incompressible materials is 0.5, and in this laboratory work we assume all sample materials to be incompressible.

To calculate the Young's modulus of the sample material, this experiment involves measuring the force by using a force sensor and recording the displacement (indentation) by controlled movement of a precision stage that holds the sample. Inserting the measured values to Equations 2 and 3 yields the Young's modulus of the elastic sample material.

Observation of the stress relaxation phenomena will be conducted by measuring the force several times with certain delay in between measurements for the same value of indentation.

2. Hardware and Software

2.1. Hardware overview

Figures 4 and 5 show the setup diagram for this laboratory exercise. The material sample to be characterized is mounted on a sample holder. The force sensor is fixed at a certain position above the sample. The sample holder is attached to a micro-translation stage so that it can move vertically (Z-axis) in order to indent the sample by the tip of force sensor. The micro-translation stage is attached to a manual stage (XY-axis) via aluminum adapter, thus the sample can be moved laterally in XY-direction for fine alignment relative to the tip of force sensor. The force sensor measures the force

[2] Adapted from: https://en.wikipedia.org/wiki/Poisson%27s_ratio

exerted by indentation of the material. Output signal of force sensor is amplified by an amplifier to be in the same range as input voltage range of the DAQ (Data Acquisition device). Amplified signal is then digitized by the DAQ, and digitized signal is read by a PC. A motor controller controls the movement of the micro-translation stage according to the commands sent from the PC to the motor controller.

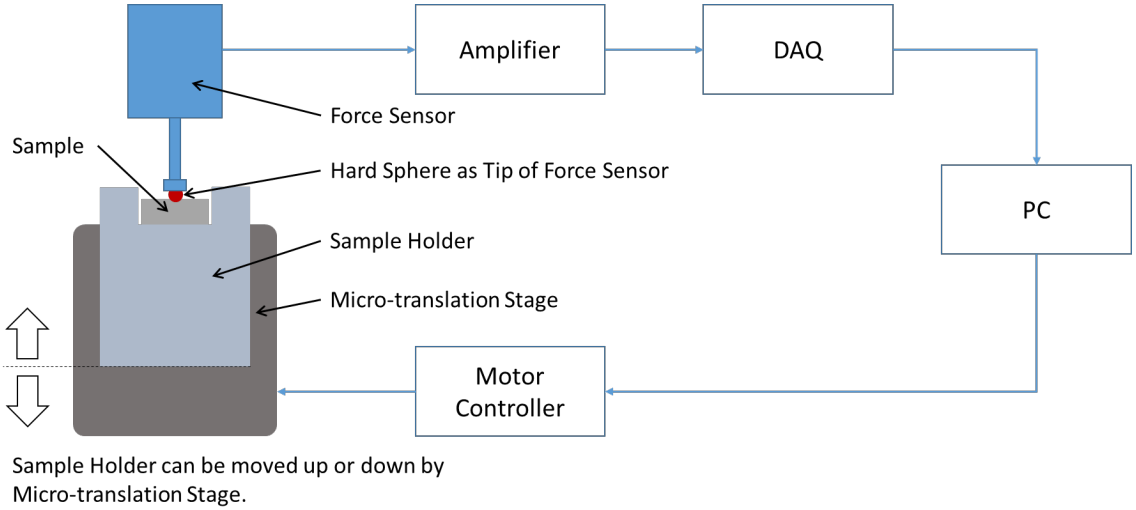


Figure 4. Experimental setup diagram.

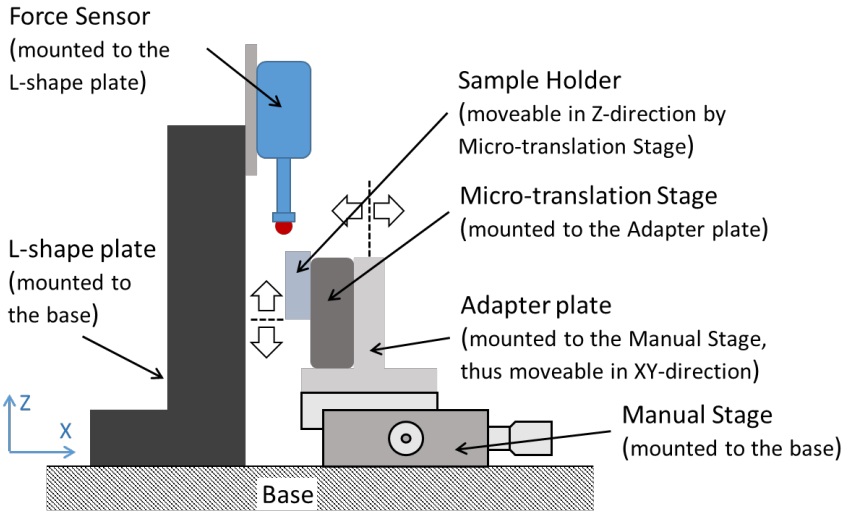


Figure 5. Experimental setup diagram, side view.

Description of the hardware:

1. Motorized Micro-translation Stage (Z-axis), **M-112.1DG** (*Physik Instrumente*)

   The micro-translation stage has translation range up to 25 mm with minimum 0.05 µm incremental motion and maximum motion velocity of 1.5 mm/s. It has integrated rotary encoder as the position sensor. It has a load capacity of 20 N and is able to withstand up to 10 N lateral force. In this laboratory exercise setup the micro-translation stage is placed vertically, thus the force exerted by the measurement system (sample holder, sample, and force sensor) must be lower than the maximum lateral force. We will call this as "micro-translation stage" or simply as "stage".

   Detailed specification can be found in "Technical Data" section of "Motorized Micro-translation Stage - M110_Precision_Positioning_Stage.pdf".

2. Motor Controller, **C-863** (*Physik Instrumente*)

   The controller is used to control the motion of the micro-translation stage based on commands sent from the PC, e.g. move the stage to certain position, set the speed, get the current position of the stage, etc. The controller communicates with PC via USB or RS-232.

   Detailed specification can be found in "Specifications" section of "Motor Controller - PI_Datasheet_C-863_20150120.pdf".

3. Manual Stage (XY-axis), **M-460A-XY** (*Newport*)

   The manual stage is used for fine alignment of sample holder in lateral direction (XY-axis). The two adjustment screws can be turned to move the sample holder in X and Y directions.

4. Low Range Force Sensor, **UF1 25g** (*LCM Systems*)

   The sensor is a resistivity based load cell, having a measurement range from 0 g to 25 g with 0.05 g resolution (see Note). This force sensor has a Wheatstone bridge, and it must be connected to excitation voltage source (2 connectors for supply voltage and ground). The sensor output is a differential voltage (2 connectors).

   Note: the force is expressed in grams, or specifically gram-force, which is equivalent to a mass of one gram multiplied by the standard acceleration due to gravity on Earth (9.80665 m/s^2).

   Detailed specification can be found in "Force Sensor - UF1 Low Range Isometric Force Sensor.pdf".

5. Analogue Strain Gauge Amplifier, **SGA/A** (*LCM Systems*)

   The amplifier has two functions: (1) to supply the excitation voltage to the force sensor, and (2) to amplify the signal from the force sensor to be in the same range as the DAQ input. It has internal switches to adjust the required amplification, level of excitation voltage, as well as type of output (e.g. voltage or current, and its range).

   Detailed specification can be found in "Chapter 9" of "Amplifier - SGA Manual.pdf".

6. Data Acquisition (DAQ) device, **USB6003** (*National Instrument*)

   The DAQ has digital input/output and analogue input/output. For this laboratory exercise, digital input will be used to read the signal from the sensor after amplification. This DAQ has internal ADC (analogue to digital converter) to convert analog input into digital data, thus the digital device (e.g. PC) can read the data. Maximum sample rate is 100 kS/s (kilo-samples per second), i.e. the DAQ is able to read and convert the analogue input 100000 times in 1 second, or every 10 µs. The ADC has 16-bit resolution, thus if the input range is set to -10V to +10V, it is able to detect the voltage change as small as:

$$\frac{(\text{Max Input Range}) - (\text{Min Input Range})}{2^{(\text{ADC bit resolution})} - 1} = \frac{(+10\text{ V}) - (-10\text{ V})}{2^{16} - 1} \approx 0.305\text{ mV}$$

   The DAQ communicates with PC via USB.

   Detailed specification can be found in "Analog Input" section of "DAQ - NI USB-6003 Specifications.pdf".

7. PC

   The PC runs MATLAB to perform reading from DAQ and send command to Motor Controller.

2.2. Force sensor calibration

Since this laboratory exercise requires a ruby ball to be attached at the force sensor tip, the sensor has to be calibrated to account for the added mass. This calibration was done by attaching various reference masses to the tip, and observing the amplified sensor output for each. The calibration result is shown in Figure 6, and the linear fit function to convert from *voltage* of amplified sensor output to *gram* (or *gram-force*) of exerted force is:

$$F = A \cdot v + B \tag{6}$$

F is exerted force, in gram.

v is amplified sensor output, in Volt.

A = 3.5949 g/V

B = -3.1 g

Increasing F value (positive) represents the case where the force sensor is pulled, and decreasing F value (negative) represents the case where the force sensor is pushed.

When calculating forces in the laboratory exercise, use the calibration values given above.

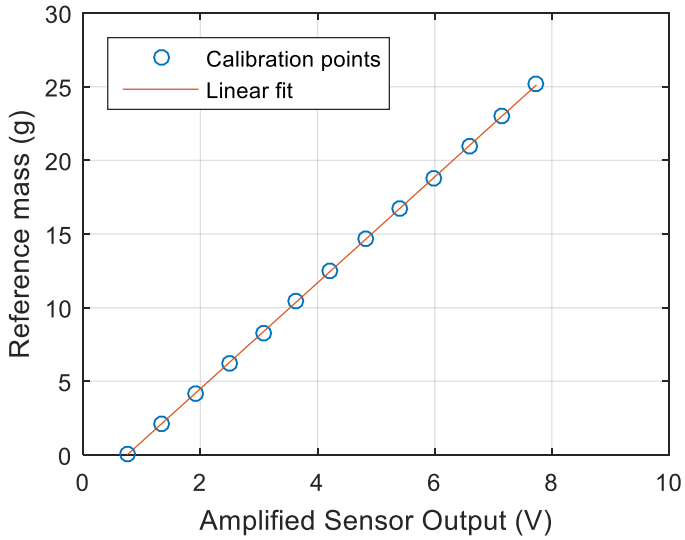


Figure 6. Sensor calibration result.

2.3. System capability

From the hardware's specification, we can estimate the system capability in terms of *range* and *resolution* of force, and range of indentation depth. *Range* represents the span of value (minimum to maximum) of a parameter, and *resolution* represents smallest change in value of a parameter.

Range of force: From the specification of the force sensor, the range of force is 0 g to 25 g for both direction (thus actually -25 g to +25 g). Let's estimate whether our system is able to detect such range.

In the laboratory exercise we will only use push force, i.e. the force range is 0 g to -25 g. By using the sensor calibration result, we can estimate the range of expected amplified sensor output, and shown in the following. Therefore, we can set the DAQ input range to +/- 10 V.

	Minimum	***Maximum***
Push force	0 g	-25 g
Amplified sensor output	0.86 V	-6.08 V

Resolution of force: From the specification of the force sensor, the resolution of force is 0.05 g; which represents the smallest force change detectable by the force sensor. Let's estimate whether our system is able to detect such smallest exerted force change.

In our setup, we need to ensure that the DAQ is able to detect the voltage change smaller than the smallest change in amplified sensor output (due to smallest change in exerted force). We can consider the case where a force F_1 is first exerted on the sensor, and then a force $F_2 = F_1 + \Delta F$, with ΔF equivalent to sensor resolution.

$$\begin{aligned} F_1: \quad & F_1 = A \cdot v_1 + B \\ F_2: \quad & F_1 + \Delta F = A \cdot v_2 + B \\ \text{Force different:} \quad & \Delta F = A \cdot (v_2 - v_1) \end{aligned}$$

So the change of 0.05 g in force value (ΔF) is equivalent to change of 13.9 mV in amplified sensor output ($v_2 - v_1$ = 0.05 g / (3.5949 g/V)). Since smallest signal change that is detectable by the DAQ is about 0.3 mV, our system will be able to detect the force change of 0.05 g.

Range of indentation depth: Once the material sample has been brought into contact with the force sensor tip, we estimate the maximum allowable indentation depth to avoid exceeding the force sensor range.

We can use the Hertzian theory to estimate the maximum indentation depth for a given maximum force.

$$d_{max} = \left(\frac{3}{4}\frac{F_{max}}{E^* R^{1/2}}\right)^{2/3}$$

Therefore, the maximum indentation depth depends on the material properties. For rubber with Young's modulus of 1 MPa and Poisson's ratio of 0.50, its effective Young's modulus is 1.33 MPa and the maximum indentation depth can be made before exerting force more than 25 g to the force sensor is:

$$d_{max} = \left(\frac{3}{4}\frac{25\ g\ \times \frac{1}{1000}\frac{g}{kg}\ \times 9.80665\ m/s^2}{1.33 \times 10^6\ Pa\ \times\ (1.5\ mm\ \times \frac{1}{1000}\frac{m}{mm})^{1/2}}\right)^{2/3} = 233.46\ \mu m$$

Softer material (small Young's modulus) has larger maximum indentation depth, while harder material (larger Young's modulus) has smaller maximum indentation depth.

2.4. Software

The hardware-software interfacing diagram is shown in Figure 7. MATLAB will be used as main user interface. PI-863.dll (dynamic-link library) contains functions to interact with the micro-translation stage, and Data Acquisition Toolbox will be used to interact with the DAQ device.

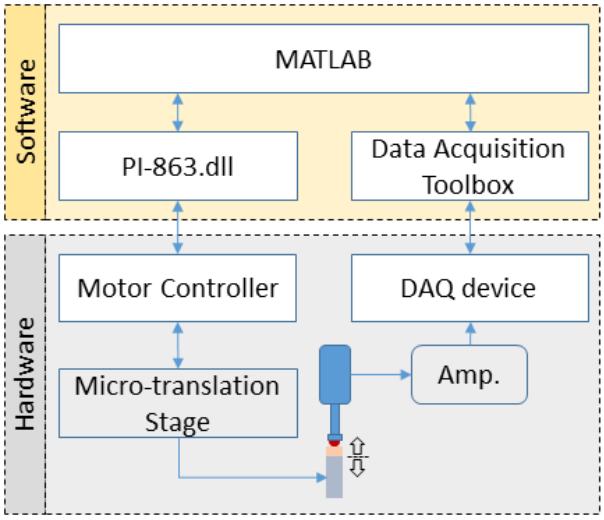


Figure 7. Hardware-software interfacing diagram.

Interfacing with micro-translation stage via motor controller:

PI-863.dll library provides the functions to connect to the stage, get the position of the stage, set properties of stage (e.g. name, velocity, displacement range, etc.), and various motion commands. The steps to **initialize** the micro-translation stage is as the following.

1. Load the PI-863.dll into MATLAB workspace, hence we can call the functions within PI-863.dll library from MATLAB.

```
lib_PI = NET.addAssembly('C:\Users\MNRLab\Desktop\STUDENTS\lib\PI-863.dll');
```

2. Call PI-863 function to connect to the stage, the function will search for any connected stage(s) to the PC. Once found then assign the name to the stage.

```
PI.PI_C863.Connect % Search and connect
PI.PI_C863.AssignStage('X', 'M-112.1DG', 1) % Assign the name of 'X'
```

3. Reference the stage to at least one of its translation range limits. *Negative Limit* represent the minimum range, e.g. position 0 mm, and *Positive Limit* represent the maximum range, e.g. 25 mm for M-112.1DG Stage.

```
PI.PI_C863.GotoNegativeLimitSwitch('X'); % Ask Stage "X" to go to its Negative Limit
PI.PI_C863.WaitReferencing('X');
```

The following are examples on how to use various motion commands, please refer to the "Appendix 1" for the detail description of each commands.

1. Set motion velocity.

```
PI.PI_C863.SetVelocity('X', 1.0); % Set velocity to 1.0 mm/s
```

2. Move the stage to absolute position.

```
PI.PI_C863.Move('X',10.00); % Move to position 10 mm
```

3. Move the stage by a relative displacement.

```
PI.PI_C863.MoveRelative('X',0.2); % This will move the stage by 0.20 mm
```

Note that the commands is **non-blocking**, in which the next command will be executed even though the previous command is not completed yet (i.e. still running in the background). For example if we run the script containing these commands:

```
PI.PI_C863.Move('X',15.00);
PI.PI_C863.GetPosition('X') % Display the current position
```

`GetPosition()` command will be executed immediately after `Move()` command even though the stage has not reach the position 15 mm. Therefore, in this case `GetPosition()` command will return the current position of the stage, which may not be the final target.

```
start_pos = PI.PI_C863.GetPosition('X'); % Get start position
PI.PI_C863.SetVelocity('X', 0.75); % Set velocity to 0.75 mm/s
PI.PI_C863.Move('X',15.00); % Move to new position
WaitMillisecond(((15.00-start_pos)/0.75)*1000); % Wait (in millisecond)
PI.PI_C863.GetPosition('X') % Display the new position
```

Note that MATLAB function `pause()` can only create delay with resolution of 1 second, so we use `WaitMillisecond()` which has milli-second resolution.

Interfacing with DAQ device:

MATLAB Data Acquisition Toolbox provides various functions to work with DAQ device. In this laboratory exercise, we need to know how to connect to DAQ device, how to configure input channel, and how to perform data acquisition. Please refer to "Appendix 1" for the detail description of each commands.

1. Connect and configure the DAQ device.

   MATLAB configure and control the DAQ device using a *session* object, so we need to create a session.

   ```
   s = daq.createSession('ni'); % Create session object
   ```

   Use the following command to get the device ID.

   ```
   devices = daq.getDevices; % Get the info of connected device
   devID = devices.ID; % Get the ID of connected device
   ```

   Add measurement channel. In this laboratory setup, the output of the amplifier (**SGA/A**) is connected to **Analog Input 0** as differential voltage input.

   ```
   channelInput = 0;  % Analog Input 0
   addAnalogInputChannel(s, devID, channelInput, 'Voltage'); % As voltage measurement
   ```

   Now we can set the measurement setting, e.g. sampling rate and acquisition duration.

   ```
   s.Rate = 100000; % Set sampling rate to 100 kS/s (maximum for USB6003)
   s.DurationInSeconds = 10; % In second
   ```

2. Read sensor value.

   ```
   curr_reading = inputSingleScan(s);
   ```

3. Perform continuous data acquisition.

   We can perform data acquisition either in the *foreground* (as blocking command) or in the *background* (as non-blocking command). In this laboratory instruction, we will not discuss the continuous data acquisition in the background since require usage of *events* and *listeners*. However, we will provide you with the script to perform continuous background acquisition during the laboratory exercise.

   Continuous data acquisition in the foreground can be performed using the following command.

```
data = startForeground(s); % Start foreground acquisition
```

   `startForeground(s)` command will run for a specified duration as set in `s.DurationInSeconds` and at specified sampling rate as set in `s.Rate`. For example, if we set the duration as 5 second and sampling rate as 100000 (i.e. 100 kS/s), then the `data` will contain 500000 data. The next command will not be executed (blocked) until `startForeground(s)` has completed.

3. Measurement Methods

Obtaining the force-displacement curve (in this case, displacement is indentation of material sample) can be performed in various ways. In this laboratory exercise, we will use the following methods.

3.1. Continuous Motion

In this method, once the material sample has been brought into contact with the tip of the sensor, start the measurement by moving the stage to the position slightly smaller than maximum indentation depth of provided material sample (see section "System Capability" on how to calculate such maximum value). Since the command is non-blocking, we can acquire data in the meantime. The current indentation depth is estimated from the elapsed time and velocity of the stage. The method is illustrated in Figure 8. Termination condition could be based on current position or current force value.

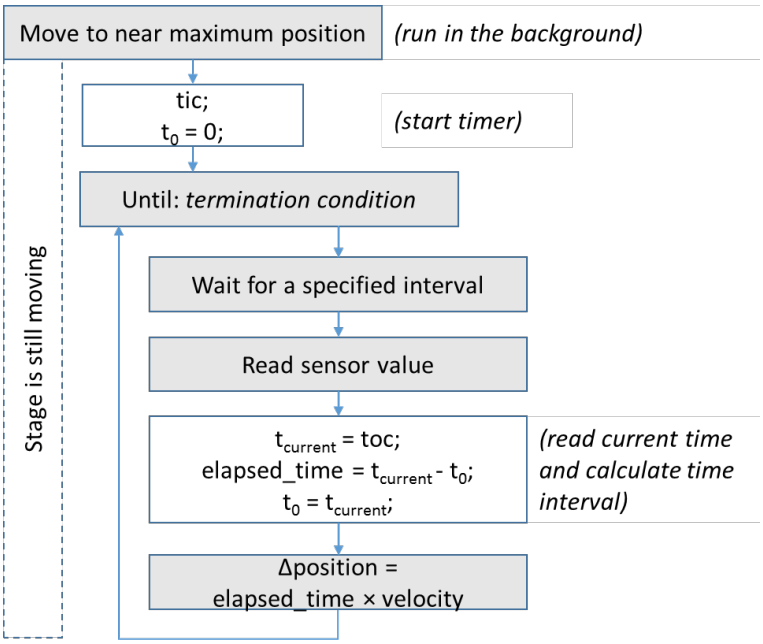


Figure 8. Continuous motion method.

3.2. Steps Motion

In this method, once the material sample has been brought into contact with the tip of the sensor, start the measurement by moving the stage one step, wait until the motion is completed, read sensor value, then repeat until termination condition is met. Termination condition could be based on either current position or current force value. At each step, you need to read sensor values several times with pauses in between. Think why this is needed and what can you observe with such measurement?

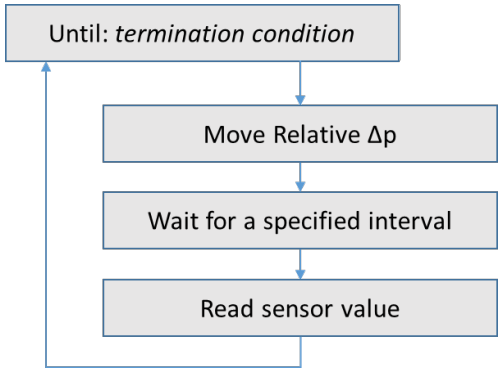


Figure 9. Steps motion method.

Prehearing assignment

Answer the following questions and submit *24 hours* before prehearing session!

1. Find from literature the value of Young's modulus and Poisson's ratio for the materials:
    a. Ethylene Vinyl Acetate (EVA),
    b. PDMS,
    c. Ecoflex soft elastomer,
    d. Ruby ball.
2. Describe the stress relaxation phenomenon in terms of force and indentation.
3. Explain general working principle of resistivity-based load cell.
4. The force sensor UF1 is internally arranged in form of Wheatstone bridge, what is the proper DAQ input mode: DIFF (differential) or SRE (single reference ended)?
5. Get familiar with measurement setup description and be ready to show components of the real setup during prehearing.
6. Write the MATLAB script to implement two measurement methods:
    a. Continuous motion; (filename: Script_Continuous.m)
    b. Steps motion (filename: Script_Steps.m).

    (Please use the template script provided on the MyCourses page, under "Microforce measurement" folder → "Matlab templates").

Laboratory session

Due to COVID-related restrictions, the lab session will be arranged remotely. The whole group will participate via Zoom. The instructor will assist with hardware, while students will be able to control the system remotely with their Matlab codes. Students will also observe how system operates through the camera. After the session, instructor will send the data obtained during measurements for analysis and report.

Prepare and fix your scripts as instructed in prehearing session! You will use them to operate the system. Upload them into dedicated folder in MyCourses *12 h before your lab session*– instructor will download them beforehand.

Part 1: Familiarize with the instrument and commands, setup the system and test your codes

In order to avoid the sample holder accidentally hitting the sensor tip, move the sample holder away horizontally for part 1 (instructor will assist with that).

I. Setup the system.

Run Script01_Setup.m provided by the instructor.

Familiarize with the following stage commands (in Script01_Setup.m):

a) Load the library and connect to the stage

```
% Load the library
lib_PI                                                                    =
NET.addAssembly('C:\Users\MNRLab\Desktop\STUDENTS\lib\PI-
863.dll');


% Connect and assign the name for stage
PI.PI_C863.Connect
PI.PI_C863.AssignStage('X', 'M-112.1DG', 1)
```

b) Reference the stage.

```
% Reference the stage to negative limit
PI.PI_C863.GotoNegativeLimitSwitch('X');
PI.PI_C863.WaitReferencing('X');
```

c) Play around with these functions to understand their behavior.

```
% Set velocity to 0.1 mm/s (maximum is 1.5 mm/s)
PI.PI_C863.SetVelocity('X', 0.1);
```

```
PI.PI_C863.GetPosition('X')
PI.PI_C863.MoveRelative('X',0.5);
PI.PI_C863.Move('X',10.0);
```

Familiarize with the data acquisition commands (in Script01_Setup.m):

d) Configure the DAQ

```
devices = daq.getDevices; % Get the list of available devices

% Set the properties of DAQ device
devID = devices.ID;    % Get the device ID
channelInput = 0;     % Which channel input the sensor is connected
to
samplingRate = 100000;% Valid range from 0.1 to 100000.0 scans/sec

% Create session and assign input
s = daq.createSession('ni');
addAnalogInputChannel(s, devID, channelInput, 'Voltage'); % Set
as voltage measurement
s.Rate = samplingRate;

% Set acquisition duration
s.DurationInSeconds = 10; % In second
```

e) Read measurement data

```
% Read sensor value
curr_reading = inputSingleScan(s);

% Read measurement data for a specified duration as set in
DurationInSeconds
[data,time] = s.startForeground; % This is a blocking command
```

II. Test your codes

IMPORTANT: Update parameters in your code. All parameters have been defined for you and added to your code. You need to select correct ones depending on the sample using commenting/uncommenting.

Run your codes without sample and at secure stage position to see if they work/give any error.

Part 2: Perform the measurements

Sample loading:

You will test 3 samples in this lab. Instructor will change samples for you.

1) Move the sample holder away horizontally so that the stage is NOT under the sensor tip.
2) Place sample to the sample holder. DO NOT TOUCH THE SENSOR TIP! It may disturb measurements or even destroy the sensor. This step will be performed by instructor.
3) Move back the sample holder such that the sensor tip is around the center of sample top surface.

Sample specific setup:

If the sample was changed, the setup described in part 1 must be repeated with sample-specific stage settings. Instructor will assist you in that.

I. "Script_Continuous.m" experiment: Measure the force-displacement curve using the method of continuous motion

1) Execute your "Script_Continuous.m" (Note: make sure your values are correct for the sample!).
2) Save the measurement data. You need to change the name of the recorded file after each measurement! The file will be overwritten when you execute this code again. Use file name that helps identifying what the data is.

II. "Script_Steps.m" experiment: Measure the force-displacement curve using the method of steps motion

1) Execute your "Script_Steps.m" (Note: make sure your values are correct for the sample!).
2) Save the measurement data. You need to change the name of the recorded file after each measurement! The file will be overwritten when you execute this code again. Use file name that helps identifying what the data is.

III. Additional experiment: Measure the force-displacement curve using the method of steps motion with continuous sensor reading

1) Instructor will provide the script. You need to update material-specific parameters and start and end position of the stage.
2) Execute the provided script and observe the result plot. Does it look like what you expected?
3) Save the measurement data. You need to change the name of the recorded file after each measurement! The file will be overwritten when you execute this code again. Use file name that helps identifying what the data is.

Sample-specific data

Table 1. Sample-specific variables. These parameters have been added to your codes. You need to select correct ones for each sample.

		Ecoflex	PDMS	EVA
Stage limit (Setup step)				
Contact_start_pos		5.8	5.2	5.2
Contact_end_pos		8.5	5.5	5.36
Offset		0.1	0.1	0.1
Incremental step experiment	Velocity	0.1	0.1	0.1
	Step size	0.1	0.01	0.01
Continuous move experiment	Velocity	0.1	0.01	0.01
	Step size	0.025	0.0001	0.0001

Final Report

For writing your report, use “Lab report template” to get familiar with scientific article format.

1. Explain shortly what you have performed in the Lab.
2. Plot the force-displacement curve for all your experiment results.
3. Calculate the modulus of elasticity (Young’s modulus) of each material sample from experimental results.
4. Compare your calculated Young’s modulus results with literature. Is the derived value in range of existing data?
5. Explain the result of Experiment 4 based on stress relaxation concept. You may provide figure you obtained and explain what happens of the graph and why.

Appendix 1

Some useful PI-863 commands.

`PI.PI_C863.Connect`

Connect to the PI motion controller. This should be called before any operation.

`PI.PI_C863.AssignStage (deviceName, model, deviceNumber)`

Assign a stage to the system, and give it a friendly name. The range of motion can be later changed using SetRange().

Remark: You can find the model of the stages on the hardware. If you are not sure which stage is connected to which axis, you can check it with the PIMikroMove software.

`deviceName`	The name that you wouldlike to refer in your code, e.g. X, Y, Gripper, etc
`model`	Model of the stage, e.g. M-404.8PD, M-111.1DG, M-122.2DD.
`deviceNumber`	The deviceNumber of the controller, starting from 1, 2, 3, 4, ...

`PI.PI_C863.GotoNegativeLimitSwitch (deviceName)`

Go to the negative limit switch. Note: this is the hardware limit, not the software limit. The operation will also reset the valid motion range. This operation is necessary before any Move command is used.

`deviceName`	Name of the axis

`PI.PI_C863.WaitReferencing (deviceName)`

Wait a referencing command, e.g. GotoReference, GotoNegativeLimitSwitch, GotoPositiveLimitSwitch to finish

`deviceName`	Name of the axis

`PI.PI_C863.SetVelocity (deviceName, v)`

Set the velocity of the axis

`PI.PI_C863.GetPosition (deviceName)`

Get the position of the axis

`deviceName`	Name of the axis

`PI.PI_C863.Move (deviceName, v)`

Move the stage to an absolute position

`deviceName`	Name of the stage
`v`	Target position

`PI.PI_C863.MoveRelative (deviceName, v)`

Move a stage to the target position related to its current position. Return 0 if success, 1 if out of range.

`deviceName`	Name of the stage
`v`	Target position related to current position for the stage

Some useful Data Acquisition Toolbox commands.

`daq.createSession(vendor)`

Create data acquisition session for specific vendor hardware, return a session object. Create one session per vendor and configure the properties accordingly (e.g. add channel, set sampling rate, etc).

Some useful session's properties:

`Rate`	Specify the device's sampling rate, in scans per second.
`IsContinuous`	Specify if operation continues until manually stopped.
`DurationInSeconds`	Specify duration of acquisition.

`addAnalogInputChannel(s,deviceID,channelID,measurementType)`

Add analog input channel to the specified session.

`s`	Session object.
`deviceID`	Device ID as defined by the device vendor. Use `daq.getDevices` to obtain the Device ID.
`channelID`	Location of the channel on the device.
`measurementType`	Vendor-defined measurement type (e.g. `'Voltage'`, `'Current'`, `'Accelerometer'`, etc).

`inputSingleScan(s)`

Acquire single scan from all input channels in the session.

`s`	Session object.

`startForeground(s)`

Starts operations of the session object and block MATLAB command line and other code until the session operation is completed.

`s`	Session object.

`startBackground(s)`

Starts the operation of the session object without blocking MATLAB command line and other code.

`s`	Session object.

Appendix 2: Linear Regression

Linear regression is an approach for modeling the relationship between a dependent, or response, variable *y* and one or more explanatory independent, or predictor, variables *X* (consist of $x_1, x_2, \ldots, x_n$). If only one independent variable is considered, the relation is

$$y(x) = p_1 \cdot x + p_2$$

where p_1 is the slope, and p_2 is the *y*-intercept.

We can use the MATLAB function of `polyfit(x,y,n)` (polynomial curve fitting) to perform linear regression by supplying independent variable *x*, dependent variable *y*, and setting *n* (polynomial order) to 1, i.e. linear. The following provides an illustration on how to use `polyfit()` function to perform linear regression.

Consider that you have set of data as in the following table.

***x*:**	0.00	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2.00
***y*:**	-2.71	-1.29	-0.34	0.29	2.35	3.68	4.68	6.01	7.24

First, assign the data into MATLAB variable, e.g. `x` and `y`.

```
>> x = [0.00, 0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 2.00];
>> y = [-2.71, -1.29, -0.34, 0.29, 2.35, 3.68, 4.68, 6.01, 7.24];
```

Use `polyfit` to model `y` as linear function of `x`, the `polyfit` returns coefficients of p_1 and p_2.

```
>> p = polyfit(x,y,1)
p =
    5.0087   -2.7964
```

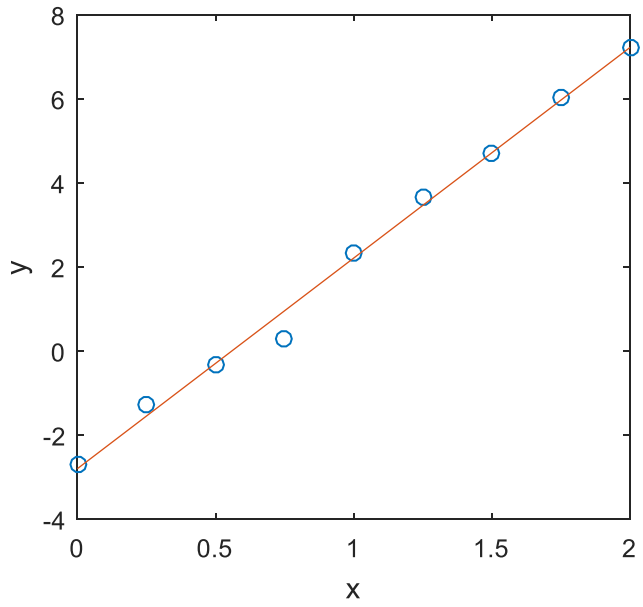
Fitted function can be calculated as:

```
>> yfit = p(1)*x + p(2);
```

Plot the results; use circle mark to represent original data and line to represent fitted data.

```
>> plot(x,y,'o'); % Plot original data using circle mark
>> hold on;       % Next plot will be plotted on previous window, not by creating new window
>> plot(x,yfit);  % Plot fitted function, default mark is line
>> hold off;      % Stop "hold on"
>> xlabel('x'); ylabel('y'); % Label the axis
```

The plot result should look like this:



In the case of modelling the force F and the indentation depth d in the Hertzian theory,

$$F = \frac{4}{3} E^* R^{\frac{1}{2}} d^{\frac{3}{2}}$$

Even though the relationship of F and d is not linear, we can instead model the relationship between F and $d^{\frac{3}{2}}$ which is linear.

$$F = \left(\frac{4}{3} E^* R^{\frac{1}{2}}\right) \cdot \left(d^{\frac{3}{2}}\right) = (slope) \cdot \left(d^{\frac{3}{2}}\right)$$

Once the relationship between F and $d^{\frac{3}{2}}$ has been modelled by using linear regression to obtain p_1 (slope) and p_2 (y-intercept), we can calculate effective elastic moduli as the following.

$$E^* = p_1 \cdot \frac{3}{4} \frac{1}{R^{\frac{1}{2}}}$$