GE U111 HTT&TL, Lab 1:

The Speed of Sound in Air,
Acoustic Distance Measurement
& Basic Concepts in MATLAB

Contents

1 Preview: Programming & Experiments Goals

This session is used to introduce basic (MATLAB) programming concepts, including

- Variable definitions
- Arrays and array manipulations
- 2D plotting
- Script M-files

Basic (Engineering / Science) concepts, including

- Parameter estimation from noisy data
- Matching a mathematical model to data

The driving application involve estimation of the speed of sound in air [1] and acoustic distance measurement to remote objects. Specific tasks include

- Measuring the travel time between ultrasound transducers
- Estimating the speed of sound in air from measurement data
- Acoustic distance measurement

The components & equipment used in this experiment include

- Two sound wave transducers: an emitter (essentially, a small, directional loudspeaker) and a receiver (essentially, a directional microphone); a mark on the basis of each transducer assembly indicates which is the emitter and which is the receiver.
- The emitter is connected to a pre-programmed signal generator
- Both transducers are connected to a digital oscilloscope
- A measuring tape (metric), glued to the lab bench
- A clip board
- MATLAB, installed on the computer.

Experiment Outline: The emitter transducer emits a short burst of several cycles of a sound wave at a 40kHz frequency[2]. The receiver, located at a selected distance from the emitter, picks up the sound wave. The two transducers are connected to the two input channels of a digital oscilloscope. The process is automatically repeated, periodically, every 10 ms. [3]

The oscilloscope displays the recorded sound waves from both transducers. Using features of the oscilloscope display you will be able to measure the time it takes travel time of the sound wave from the emitter to the receiver. In principle, a single measurement suffices to compute an estimate of the speed of sound:

$$\text{speed of sound} = \frac{\text{travel distance}}{\text{travel time}} \tag{1}$$

or equivalently

$$\text{travel distance} = \text{speed of sound} \cdot \text{travel time} \tag{2}$$

[1] Actually, the speed of sound varies with temperature and air pressure, and the results obtained will provide an estimate for the altitude and temperature of our lab environment.

[2] The frequency of 1 Hz (one Hertz) means one cycle per second; 1 kHz (one kilo-Hertz) stands for 1000 cycles per second. Thus the sound wave used in our experiments is of 40,000 cycles per second.

[3] The notation "ms" stands for "milisecond", or 0.001 second. We shall later encounter also the "μs", standing for "microsecond", which is $1\mu s = 1 \cdot 10^{-6} = 1e-6\, sec$.

In practice, multiple measurements are needed to overcome measurement errors of both the travel time and the distance between the transducers. Moreover, due to the physical structures of the transducers, distance measurements will be biased, meaning that even without any other inaccuracies, the actual distance between transducers is different from what we can measure by a fixed amount:

$$\text{measured travel distance} = \text{speed of sound} \cdot \text{travel time} + \text{bias} \tag{3}$$

This last equation is of the general form of a linear graph

$$y = a\,x + b \tag{4}$$

You will conduct ten experiments and obtain ten values for distance / travel time pairs. You will then use a MATLAB program to optimally fit a linear graph of the form (4) to your data. The quantity "a" in that fit will be your estimate of the speed of sound, as in (3).

Having estimated the speed of sound and the distance measurement bias, acoustic distance measurement to a remote object will again use the formula (3): Now the two transducers will be placed side by side. The emitter's sound wave will travel to the target object, bounce back and be recorded by the receiver. The travel distance will thus be twice the distance to the remote object. Having measured the travel time, we can thus conclude

$$\text{distance to remote object} = 0.5\,(\text{speed of sound} \cdot \text{travel time} + \text{bias}) \tag{5}$$

The representation of the relationship between travel distance and travel time in the basic formula (3) is a mathematical model. Your final task in this lab will be find an alternative mathematical model, relating the distance to the peak amplitude of the return sound wave, instead of to the travel time.

Programming: The data recording and processing required in this experiment require an introduction to some of the essential building blocks of programming, in any programming language. We shall use this lab as an opportunity to introduce these concepts, as explained below.

2 Homework Assignment

¿From Lab 2 and on, our practice will be to go through relevant components of the text book before the lab meeting, and include assigned preparatory work by then. Since this is our first meeting, the primer component of Lab 1 is included in the work for the final lab report. In preparation of your report

- Read this entire handout carefully.
- Do the following in a Pre-Lab assignments
  - Read[4] in the course text MATLAB: An Introduction with Applications
    - Chapter 1
    - Chapter 2
    - Optional Section 4.3.1 (pp. 91-93) (We shall discuss the necessary material in class)
    - Optional: 2D Plots, pp. 119-124 (We shall discuss the necessary material in class)
  - Do the following problems (include a printout of the MATLAB command window with the solution of each problem):
    - Problems 1-5 and 1-15 on pages 28, 30
    - Problems 2-1, 2-3, 2-14 on pages 53, 54
- Following completion of the Lab, write a Post-Lab report, summarizing the experiments and your findings, as explained below.

[4]Here and throughout, preparatory reading is essential for the understanding of class material and the ability to perform tasks in class or in the lab.

3 Measuring The Speed of Sound: The Experiment

You will conduct ten experiments and record your data in your notebook. Later you will analyze these data using MATLAB.

Attention: Make sure you record all your measurements in the standard `meters`, `seconds` and `Volts` units!

1. Position the emitter so that its face will be directed along the measuring tape and the front of its base is flush with the 0 position.
2. Position the receiver at a point of your selection along the measuring tape, anywhere between 5 - 50cm, facing the emitter. (The position is measured from the front of the transducer's base.)

   The oscilloscope displays both the emitter signal (Channel 1) and the receiver signal (Channel 2). You will notice the record of the short interval of a high amplitude emitter waveform, early on (on the left), followed by an essentially (close to) zero reading and then, an area of increased amplitude waveform, more to the right. This second waveform is the record of the sound wave sensed by the receiver.
3. Read the time it takes sound to travel from the emitter to the receiver:
   (a) Press the curser bottom
   (b) Under the display window, select **t1**
   (c) Use the "**entry**" knob to bring one vertical cursor in the oscilloscope window to the point where, by your visual judgement, the emitter burst begins
   (d) Under the display window, select **t2**
   (e) Use the "**entry**" knob to bring the second vertical cursor in the oscilloscope window to the beginning of the elevation in the response, in Channel 2.
   (f) The corresponding time difference - the sought travel time - appears as "Δ**t**" in the oscilloscope display window. (Pay attention to the time units used: 1 ms = 1e-3 s; 1μs = 1e-6 s.)
4. Record the data pair, consisting of the sound travel time (in seconds!) and the distance between the fronts of the bases of the two transducers (in meters!)

   Using a calculator, estimate the speed of sound from each of the ten experiments, using the formula (1).
5. For later use, record also the peak-to-peak amplitude of the received sound wave
   (a) Press the curser bottom
   (b) Under the display window, select **v1**
   (c) Use the "**entry**" knob to bring one horizontal cursor in the oscilloscope window to the lowest peak of the receiver signal
   (d) Under the display window, select **v2**
   (e) Use the "**entry**" knob to bring the second horizontal cursor in the oscilloscope window to the highest peak of the receiver signal
   (f) The corresponding voltage difference appears as "Δ**v**" in the oscilloscope display window. (Pay attention to the voltage units used! Record the peak-to-peak value in Volts.)

4 Acoustic Distance Measurement: The Experiment

Here we shall demonstrate the principles of commercially available acoustic distance measurement devices (with diverse applications ranging from monitoring chemical process reactors to real estate), with our simple setup.

- Place the standing clipboard at the zero position of the measuring tape, with its face facing the length of the measuring tape.
- Place both the receiver and the emitter, side by side, at a selected position along the measuring tape (5 - 30 cm), facing the clipboard. (The front of their bases is used as the transducers' position, just as in the previous experiment.)
- Use the oscilloscope to measure the travel time of the sound wave from the emitter to the receiver. Record the distance and travel time in your notebook.
- Repeat the experiment 2 more times, at different distances.

In §5.6 you will compare the actual distance with a distance estimated in terms of the travel time.

5 Data Analysis With MATLAB

MATLAB basics will be learned, based on your assigned reading and class practice. §5.1 and §5.2 are included as immediate reminders. The discussion of data analysis begins in §5.3.

5.1 Creating & Saving a New M-File on a Memory Stick

MATLAB computations can be executed directly from the MATLAB command window, or by use of program files. MATLAB programs are saved as **M-files** in an accessible directory, as explained below. Since lab computers will be used by others we shall **never** save programs on the hard disk. Here and throughout, programs will be saved as M-files on a memory stick and run from the appropriate directory. In what follows we refer to the directory of the memory stick as "**E**". Make sure to check which directory is actually occupied by the memory stick and substitute the correct reference for "E". Files saved on the hard disk might be erased and lost without prior notice!

- Insert a memory clip
- In the MATLAB command window, change directory to the floppy drive

  ```
  >> cd E:\
  ```

- Verify the current directory

  ```
  >>pwd

  >>    E:\
  ```

- Under "File", select "New M-file"
- Save the new M-file under a name of your choice, with the suffix ".m" (for example SOS_Lab1.m is a good name for a program used to measure the speed of sound in Lab 1). The following rules apply to names used for both MATLAB variables and M-files:
  - the name must begin with a letter
  - the name may contain letters, numbers and the underscore sign
  - the name may not contain other characters, or a space
  - the name may not be equal to the name of a recognized MATLAB function (such as `sin`, `cos`, `log`, etc.). When in doubt, do not use a suspected name for your variables and programs!

  To run an M-file as a script program, type its name in the MATLAB workspace and hit return. Alternatively, select the run icon in the editor window.

5.2 Defining MATLAB Variables

This part is covered in the required **Lab 1** book reading and will be practiced in class. The following is are few highlights:

- Variables are used to store data values. The name of the variable is a reference to a point in the memory.
- MATLAB will not recognize a variable before it is expressly defined.

  To see that type `boom` in the command window. Assuming a variable `bum` was not previously defined, the response will be

  ```
  >>boom
  >>???  Undefined function or variable 'boom'.
  ```

- The equal sign "=" is used in MATLAB, as in most programming languages, for the **assignment** operation, acting from left to right. Thus

  ```
  >>moom=5
  ```

  assigns the value `5` to the variable `moom`. If `moom` did not previously exist, it also creates this variable in the memory. Contrary to the usual usage of "=" in mathematics, the legal operation

  ```
  >>moom=moom+1
  ```

  assigns to the existing variable `moom` a new value, comprising its old value plus one.

- By default, each value assignment command is followed by a display of the assigned value in the MATLAB command window. This is mostly undesirable, and in case of large array, will dramatically slow down you programs!

  To prevent this and as a matter of standard practice, end each assignment command with a semicolon **";"**

- Variables may be scalars (the default), taking a single number (or character) as a value. They may also be one, two or three dimensional arrays of values. Details are provided in the book (pp. 40-48).

5.3 Experimental Data Recording and Plotting

The following commands should be included in an M-file used to record and plot the data from the speed of sound experiment. Your Lab report should include the M-file and a printout of the MATLAB command window were the M-file is executed[5]

- define three row vectors of size 1×10 (remember to type a semicolon ";" at the end of each vector definition!):
  - A vector `tdistance` whose entries are the 10 distances between the two transducers in your experiments
  - A vector `ttime` whose entries are the corresponding measured travel times
  - A vector `amplitude` whose entries are the corresponding measured maximal peak-to-peak amplitudes of the receiver signal
- Create a 3×10 matrix named `sosdata` whose first row comprises the vector `tdistance`, the second row is the vector `ttime` and the third row is `amplitude`.
- Use the `disp` command to display the tabulated data in the matrix `sosdata`.
- Include a command to discrete plot of the travel distance (vertical axis) as a function of the travel time (horizontal axis), using black asterisks for data points

  ```
  plot(ttime,tdistance,'k*')
  ```

- Following the plot command, include the title "Speed of Sound Data: <your team names>", and axis titles "travel time" and "travel distance":

  ```
  >>title('Speed of Sound Data:  ...')

  >>xlabel('travel time')
  >>ylabel('travel distance')
  ```

Save and run the M-file. Your lab report will include the printed M-file and printed output of the final version of the M-file.

[5]The simplest way to print a portion of the command window is to use the mouse to highlight that section, then selecting `Print Selection` under the `File` menu.

5.4 Polynomial Curve Fitting - General

Our goal here is to fit the data in the vectors `tdistance` and `ttime` with an equation of of the form (3):

$$\texttt{tdistance} = \text{speed of sound} \cdot \texttt{ttime} + \text{bias} \tag{6}$$

Referring to this form, we are trying to estimate the values of the coefficients `a=speed of sound` and `b=bias` of a first order polynomial that best fit our recorded data. Here we shall use two MATLAB functions that are introduced only much later in the book (search the index):

$*$ The MATLAB function `polyfit` is used to fit data with a polynomial curve. We shall be interested in the spacial case of a first order (linear) polynomial of the form (4)-(6). Following is a general description of `polyfit` and the way it is used:

Given same size vectors `x=[x(1),...,x(N)]` and `y=[y(1),...,y(N)]` and a prescribed polynomial order `M`, the objective of `polyfit` is to find the polynomial

$$p(u) = \sum_{k=0}^{N} a_k u^k \tag{7}$$

that solves the approximation problem

$$\min_{a_0,\ldots,a_M} \sum_{l=1}^{N} (y(l) - p(x(l)))^2 \tag{8}$$

That is, the graph of the polynomial `p(u)` has to best approximate the graph of the data pairs `(x(k),y(k))` in the least mean squares sense. The general syntax of `polyfit` is:

```
>>p=polyfit(x,y,M)
```

where, in the notations of (7), we identify "`p`" with the coefficients' vector $p = [a_M, a_{M-1}, \ldots, a_1, a_0]$.

$*$ The MATLAB function `polyval` is used to evaluate a polynomial `v=p(u)` at a point, or list of points `x`. The syntax is

```
>>y=polyval(p,x)
```

where the interpretation of $p = [a_M, \ldots, a_1, a_0]$ is as above, and where `x` is a single variable or an array. The returned value(s) `y=p(x)` are stored in an array `y` whose size is the same size as that of `x`.

5.5 Linear Fitting of The Speed of Sound Experiment

As stated above, our goal is to fit the data pairs `(ttime,tdistance)` by a straight line, as in (6). You should therefore include in your M-file the command

```
>>est=polyfit(ttime,tdistance,1)
```

In accordance with the notations of (6), the returned vector will be the best estimate of `est=[est(1),est(2)]=[speed of sound, bias]`. If we want to see how good this estimate is, we should substitute the estimated coefficients and the entries of `ttime` in the right hand side of (3). This is done with the command

```
>>polyval(est,ttime)
```

For a visual comparison you will therefore include in your program a plot command that compares the raw data with the linear estimate

```
>>plot(ttime,tdistance,'k*',ttime,polyval(est,ttime))
```

Notice that here we allow a continuous plot of the linear approximation. Add a title (including name of team members) and axes labels commands.

To evaluate the quality of your estimate of the speed of sound, search the web for "speed of sound" and compare your estimate to what you find in the web. The site http://hyperphysics.phy-astr.gsu.edu/hbase/sound/souspe.html is one option.

5.6 Acoustic Distance Measurement

Here you will compute the estimated distance from the transducers to the clipboard in the experiments carried in §4. The estimated distance is given by Equation 5, where you have to use your estimates of the speed of sound and the bias term. That is, you have to use the MATLAB function `polyval` and the linear polynomial `est=[est(1),est(2)]=[speed of sound, bias]`. In your report compare the distance estimate with the actual distance, as recorded in the experiment.

6 An Alternative model

As an alternative to the model (3), you will seek a representation of the distance as a function of the maximal peak-to-peak amplitude of the receiver signal. We expect a reverse relation: the longer the distance, the smaller the amplitude. Therefore the sought model will be a function of the variable $x = \frac{1}{amplitude}$. You are allowed to use the `polyfit` function to test polynomial approximations of the form

$$\text{travel distance} \approx \sum_{l=0}^{M} a_l x^l \tag{9}$$

with various levels of the degree M. At this point you may rely on visual inspection to evaluate the quality of the approximation. Clearly, the higher the degree M you allow, the better the fit can be. However, we want the simplest possible formula, and will stop increasing M if there is no noticeable improvement. The findings in your report should include:

- The values of the optimal coefficients and polynomial degree
- A plot, comparing the discrete data pairs `ttime,amplitude` with the plot of the approximating function (9) that you identified. The plot must include a title (with your name) and axes labels.

Hint: To create the data vector $x = \frac{1}{amplitude}$ from the data vector `amplitude`, you use the place-by-place vector division command

```
>> x=1./amplitude;
```

If you expect the amplitude to vary with the distance according to a power law, ($amplitude = constant * distance^n$) you can try another way to fit the data: Taking the logarithm of both sides of the equation above, you get the following equation:

```
>>log (amplitude) = log (constant) + n log(distance)
```
If you then define new variables:

```
u=log(amplitude)
v=log(distance)
L=log(constant)
```

you get the equation:

```
u = L + n * v
```

This is a linear equation and you can do a least squares fit to the data to get he best fit value for n (the slope) and L (the y-intercept). Using the data you took for amplitude vs. distance, use MATLAB to create the new variables u and v. Then use `polyfit` to get the best fit values for the slope and y intercept. Finally, convert your results back to get the best fit values for `constant` and `n` in the equation above: $amplitude = constant * distance^n$.

Question (answer in your lab report):

1. How does your best fit values for constant and `n` compare with the values from your Power Law fit in Excel?
2. How do you think that Excel gets its Power Law fit?
3. How could you use polyfit and a linear straight line fit to find the best fit values of y_o and a from (x, y) data that fits the following exponential equation:

   ```
   y = y_o * exp (a *x)
   ```