

MCE380: Measurements and Instrumentation Laboratory

Basic Regression Techniques for Instrument Calibration

1 Introduction

As seen in class, the core of a measurement system is the *transduction principle* responsible for converting the *measurand* to *sensor* output, as seen in Fig. 1. For example, a mechanical pressure gage transforms pressure (the measurand) into angular deflection of the needle (the sensor output). Since we are interested in reading the pressure and not the angular deflection, the sensor output must be interpreted in terms of pressure. This is the role of the scale. The relationship between the sensor output and the true measurand is known as the *calibration curve*, and it is the basis for constructing a scale.

When we build an instrument and use it for the first time, we don't know the exact relationship between measurand and sensor output. At best, we have an approximate idea of the shape of the calibration curve from basic physical principles, but imperfections and deviations from ideality must be captured by performed calibration.

To *calibrate* an instrument is to apply a series of known measurands and record the resulting sensor output. The true values of the measurands are determined by a separate instrument having high accuracy. Once these data have been recorded, we can use it to determine measurand from sensor output by interpolation or regression techniques.

Another kind of calibration is frequently done between measurand and readout. This time, we apply known measurands and record the readouts instead of the sensor outputs. This assumes that a scale already exists for the instrument. This is done to correct for errors arising from instrument aging and environmental conditions.

1.1 Example

Certain temperature sensor for which the sensor output is a voltage was calibrated. Part of the data is shown in Table 1. The complete set of data is found at

http://academic.csuohio.edu/richter_h/courses/mce380/tempcal.mat

To plot the data in Matlab, load the dataset using `load tempcal`. Then plot the data

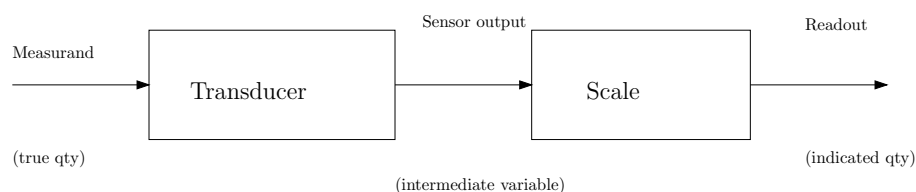


Figure 1: Basic elements of a measurement system

True Temp (° C)	Voltage
0	1.8127
2	3.2968
4	4.5119
6	5.5067
...	...
...	...
20	8.6466

Table 1: Calibration Data for Temperature Sensor

using markers to indicate actual data points: `plot(TrueTemp,Volts,'*r-')`.

In Scilab, use `loadmatfile tempcal`. Only Matlab V5 and V6 mat files can be loaded. The file was saved in V6 format, so there is not problem. The plot command is identical to the Matlab one.

As you can see, the sensor is nonlinear. The sensitivity (the slope of the curve) changes from point to point. Suppose we need to find the sensitivity at 10° C. We can either plot the curve and determine it using a graphical tangent, or we can approximate the derivative by the *secant* at neighboring points. We will do this in class, demonstrating the use of basic Matlab commands.

2 Using Linear Regression

Although instruments may exhibit a nonlinear calibration curve, it is often possible to fit a straight line approximation within a small range. As an example, we fit a straight-line approximation to the temperature sensor curve near 10° C. Specifically, we choose the range between 8° C and 12° C.

We are looking for a function of the form

$$y = mx + b$$

where m is the sensitivity, y is the sensor output and x is the true measurand. We do this in Matlab using the `polyfit` function, which fits a polynomial of any order to a set of data. In this case, we choose order 1 (straight line):

```
>>coeff=polyfit(TrueTemp(5:7),Volts(5:7),1);
```

The variable `coeff` will contain m and b . Suppose we want to evaluate y at $x = 10.1$. We type

```
>>polyval(coeff,10.1);
```

Finally, let's plot y against x in the range 8° C to 12° C, with increments of 0.1 and superimpose the plot to the actual data:

```
>>xcal=[8:0.1:12];
>>yca=polyval(coeff,x);
>>plot(TrueTemp,Volts,'*r',xcal,yca,'b');
>>legend('Data','Line Fit')
>>xlabel('Temperature, C')
>>ylabel('Voltage')
>>title('Data Points and Linear Approximation at 10 C')
```

In Scilab, we can perform linear regressions in a simple way using `reglin`:

```
-->coeff=polyfit(TrueTemp(5:7),Volts(5:7),1);
```

Note: the ordering of m and b is the opposite as in Matlab!
To evaluate the y at $x = 0.1$ we perform the following steps:

```
-->xcal=[8:0.1:12];  
-->yca1=m*xcal+b;  
-->plot(TrueTemp,Volts,'*-r',xcal,yca1,'b');  
-->legend('Data','Line Fit')  
-->xlabel('Temperature, C')  
-->ylabel('Voltage')  
-->title('Data Points and Linear Approximation at 10 C')
```

3 Linear Regression for Nonlinear Data

Often, we wish to obtain a function fit over a wider range, where the data is no longer linear. Many transduction principles obey logarithmic/exponential laws. It is possible to use the linear regression technique to obtain a data fit.

With experience, the nature of the curve (logarithmic/exponential, quadratic, cubic, etc.) can be guessed by visual inspection. Once a candidate functional form has been identified, the data is transformed accordingly and plotted again.

For example, if we take the logarithm of $10 - V$ and plot it against temperature, we obtain a linear relationship. Upon performing linear regression using T and $10 - V$, we obtain a straight line equation

$$y = mx + b$$

where $x = T$ and $y = \ln(10 - V)$. This implies that our data fit will be of the form

$$\ln(10 - V) = mT + b$$