Laboratory 4 Regression and Interpolation

Objective / Introduction

The objective is to become familiar with regression and interpolation by applying the capabilities of MATLAB® to practical problems. The exercises will build upon previous laboratory experience.

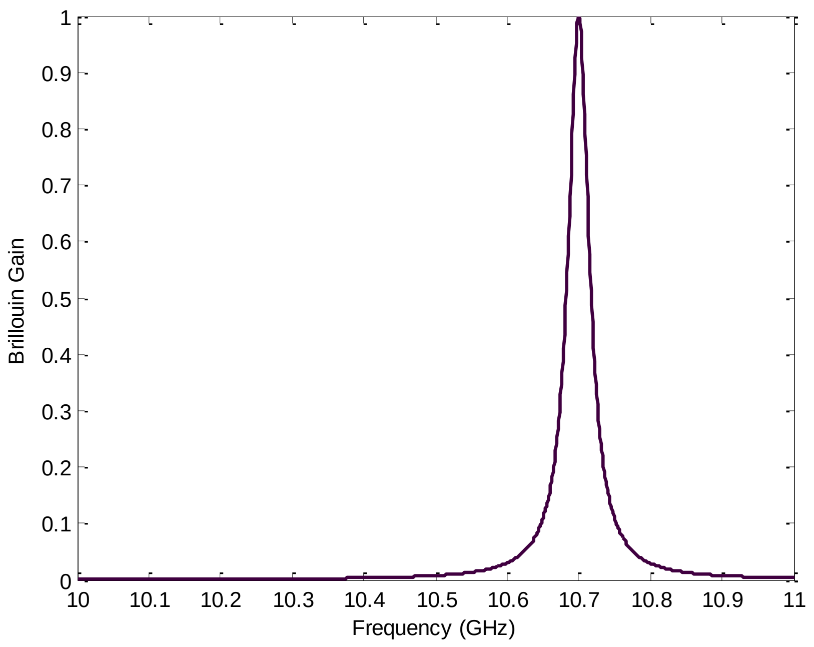
Part I Regression

The counter propagating laser light waves used in laboratory 1 form the basis of a strain and temperature sensor. The interaction of the light waves is governed by the following equation (Brillouin gain equation)

$$Bg(f) = \frac{A}{1 + 4\left(\frac{f - fb}{\Delta f}\right)^2}$$

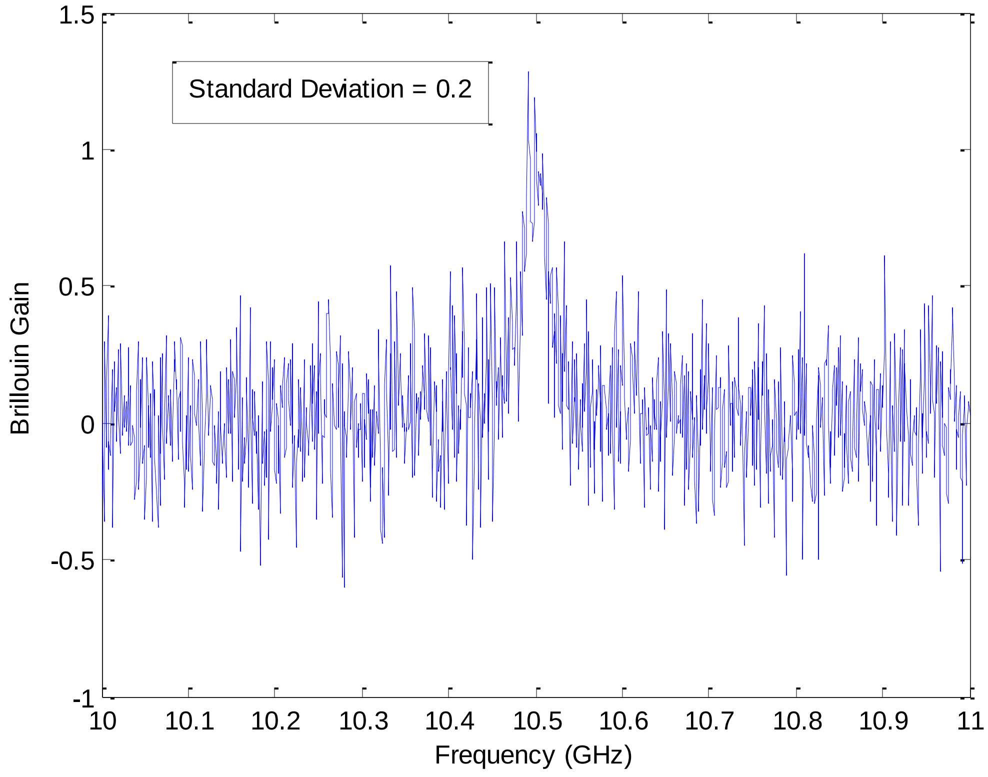
where Bg(f) is the Brillouin gain in the optical fibre and is a function of frequency,
A is the amplitude of the gain,
f is the beat frequency of the two lasers (Hz),
fb is the Brillouin frequency (Hz) which is a function of temperature and strain, and
Δf is the width of the gain profile (Hz).

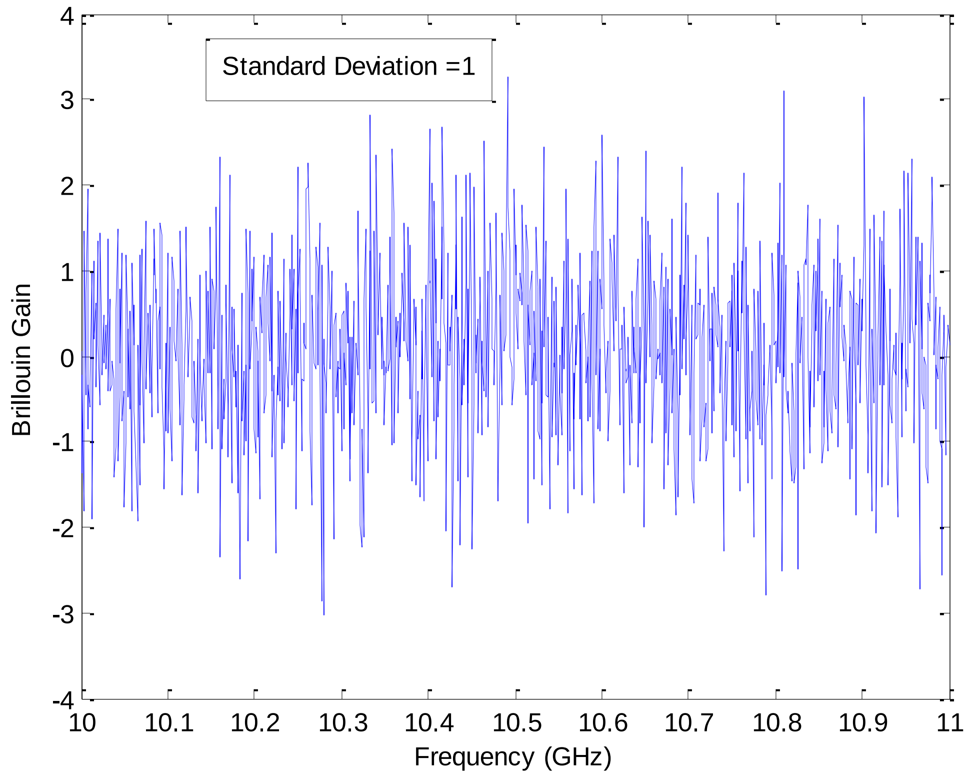
Typical values for fb and Δf are 10.7 GHz and 35 MHz respectively.
A plot the Brillouin gain profile follows.



However, this is the ideal case and in actual instruments we need to find this Lorentzian curve in very noisy data. As a strain gage or thermometer the objective is to measure the central position of the peak noting that the peak will shift upwards in frequency with increasing temperature at a rate of about 1 MHz/°C or a 20 με strain increase will cause the same 1 MHz frequency increase.

MATLAB® also has the built in capability to generate random noise through the rand and randn functions. We will use the randn function to generate a normally distributed vector of noise to contaminate our ideal data. Note that randn produces noise with zero mean, a standard deviation of one, and a variance of one. To change the mean just add the desired mean to the output and the standard deviation is set by multiplication of the output by the desired standard deviation. Shown below are plots with progressively more noise.


Standard Deviation = 0.2
1.5
1
0.5
0
-0.5
-1
Brillouin Gain
10
10.1
10.2
10.3
10.4
10.5
10.6
10.7
10.8
10.9
11
Frequency (GHz)


Standard Deviation =1
4
3
2
1
0
-1
-2
-3
-4
Brillouin Gain
10
10.1
10.2
10.3
10.4
10.5
10.6
10.7
10.8
10.9
11
Frequency (GHz)

The following MATLAB® code will produce the noisy data then use an anonymous function along with fminsearch to fit the ideal curve to the noisy data and ultimately find the center of the curve and temperature.

```
% Set up the initial conditions
f=10e9:1e6:11e9; % Investigate frequencies between 10 GHz and 11 GHz
fb=10.5e9; % Set the Brillouin frequency at 10.5 GHz
delf=35e6; % The gain profile has a 35 MHz width
amp=1; % Set the initial amplitude to 1

% Solve for and plot the Brillouin gain profile both with and without noise
bg=amp*(1+4*((f-fb)./delf).^2).^-1; % Calculate the Brillouin gain over the frequency range
lnoise=randn(size(f)); %Calculate the normally distributed noise
measured=0.1*lnoise+bg; % Add noise of standard deviation 0.1 to the gain profile
plot(f,measured,'r',f,bg,'g') % Plot the clean and the noisy data on the same graph

% Add more noise
lnoise=randn(size(f));
measured=0.2*lnoise+bg;
plot(f,measured,'r',f,bg,'g')

% Set up an anonymous function and search for a minimum
% Note that the function brillouin2 returns the sum of the squared error between the
%measured (noisy) data and the ideal curve.  In this case x(1) corresponds to the
%amplitude of the Brillouin gain, x(2) corresponds to the Brillouin frequency scaled by a
%factor of 10GHz, and x(3) corresponds to the width of the profile and is scaled by a
%factor of 10MHz.
brillouin2=@(x) sum((measured-x(1)*(1+4*((f-x(2)*1e10)./(x(3)*1e7)).^2).^-1).^2);
[x,fval,exitflag,output]=fminsearch(brillouin2,[1,1.05,3.5])
```

Case A

Investigate the level of noise that can be tolerated before the temperature error reaches 2°C. To achieve this run the above model numerous times and with each run use new noise. Record the standard deviation of the noise along with the frequency error. Plot this data and fit a line through it using the built in tools of the MATLAB figure window “tools” “basic fitting”. Identify the error limit on the graph.

Here is some sample data for noise at a standard deviation of 0.1 and 0.2. The first column is the amplitude of the “best fit” Brillouin Gain equation, the second column is the Brillouin frequency scaled by 10 GHz, and the last column is the width of the gain profile scaled by 10 MHz. Each row represents a new test case with unique noise. It is the centre column that is of interest as it contains the temperature information.

Noise 0.1

```
1.0224    1.0500    3.3733
0.9653    1.0500    3.6532
0.9569    1.0500    3.5971
```

Doubled Noise (0.2)

```
1.0084    1.0499    3.5272
0.9936    1.0500    3.3856
0.9456    1.0499    3.4792
```

Part II Interpolation

Bézier Curves and Postscript Fonts

The Bézier spline allows the user to control the slope at the knots. The smoothness at the knots is less than that of cubic splines but these splines are more suitable for instances where corners and abrupt changes are needed. Bézier developed the idea while working for the Renault automobile company. Four points are used to define one segment of the Bézier spline. The first and last points are the ends while the middle two are control points. Thus the curve begins at the first point and tangentially heads towards the first control point. The curve ends on the last point reaching it on a tangent line from the last control point. The ability to represent smooth curves and abrupt transitions in compact form has made the Bézier spline popular for drawing fonts.

The following array consists of 16 lines, each line is one Bézier spline consisting of four x-y pairs. The first pair is the start point, followed by the first control point, followed by the second control point, followed by the end point.

```
s=[
237    620    237    620    237    120    237    120
237    120    237     35    226     24    143     19
143     19    143     19    143      0    143      0
143      0    143      0    435      0    435      0
435      0    435      0    435     19    435     19
435     19    353     23    339     36    339    109
339    109    339    108    330    620    339    620
339    620    339    620    393    620    393    620
393    620    507    620    529    602    552    492
552    492    552    492    576    492    576    492
576   492   576   492   570   662   570   662
570   662   570   662     6   662     6   662
  6   662     6   662     0   492     0   492
  0   492     0   492    24   492   24    492
 24   492    48   602    71   620   183   620
183   620   183   620   237   620   237   620];
```

The Bézier curve is defined by
Endpoints (x_1,y_1), (x_4,y_4) and
control points (x_2,y_2), (x_3,y_3).

$b_x = 3(x_2 - x_1)$
$c_x = 3(x_3 - x_2) - b_x$
$d_x = x_4 - x_1 - b_x - c_x$
$b_y = 3(y_2 - y_1)$
$c_y = 3(y_3 - y_2) - b_y$
$d_y = y_4 - y_1 - b_y - c_y$

The curve is specified for $0 \leq t \leq 1$ as
$x(t) = x_1 + b_x t + c_x t^2 + d_x t^3$
$y(t) = y_1 + b_y t + c_y t^2 + d_y t^3$

The following MATLAB code implements the above equations.

```
t=0:0.01:1;
for n=1:16
x1=s(n,1);y1=s(n,2);x2=s(n,3);y2=s(n,4);x3=s(n,5);y3=s(n,6);x4=s(n,7);y4=s(n,8);
bx=3*(x2-x1);
cx=3*(x3-x2)-bx;
dx=x4-x1-bx-cx;
by=3*(y2-y1);
cy=3*(y3-y2)-by;
dy=y4-y1-by-cy;
x=x1+bx*t+cx*t.^2+dx*t.^3;
y=y1+by*t+cy*t.^2+dy*t.^3;
hold on
plot(x,y)
end
```

What is the character represented by the above array?
What is the effect of s=s*0.2;?
What is the effect of s=s+100;?

Create or find (indicate your source) and array to produce another character.
Plot this character and submit it with the lab.