AN296229
MCO-0001049
P0106

A31315 SINGLE DIE SENSOR ERROR FUNCTION – HOW TO USE

By Fabian Winkler
Allegro MicroSystems

INTRODUCTION

The 3D magnetic sensing devices offered by Allegro MicroSystems can be used in a wide range of applications where mechanical motion (linear or rotational) can be expressed by a changing magnetic angle. To estimate or simulate the performance for all applications of 3D devices is complicated. Which errors in the datasheet can be used? When they are valid? Can the angle error be used in the datasheet for a linear position application? What happens with the performance if the temperature range is within -40°C to +150°C or if the magnetic field is different than the test conditions specified?

For this reason, Allegro MicroSystems created the A31315 Sensor Error function in MATLAB. This error function takes into account the magnetic field and ambient temperature in order to apply appropriate sensor error and provide the user with application-specific behavior of the device. The output of the function gives a Monte Carlo based distribution of the magnetic field with added sensor errors.

SENSOR ERROR FUNCTION DATA

This error function uses characterization data from a limited number of sensors over temperature (-40°C to +150°C in 10°C steps). The lifetime drift was characterized using environmental stresses defined in an AEC-Q100 qualification. The measurements were performed separately for "XY" and "XZ" as well as for "Single Die" and "Dual Die" configurations. With this data, it is possible to calculate all the different sensor errors, which are needed to simulate the sensor performance.

SENSOR ERRORS

The following sensor error sources are considered in the function. Note that the communication interface was not taken into account.

Offset Error

The offset error is an offset of the signal and is measured in gauss. This kind of error will have the highest impact if the magnetic field itself is small. The higher the magnetic field, the less the impact of the offset error.

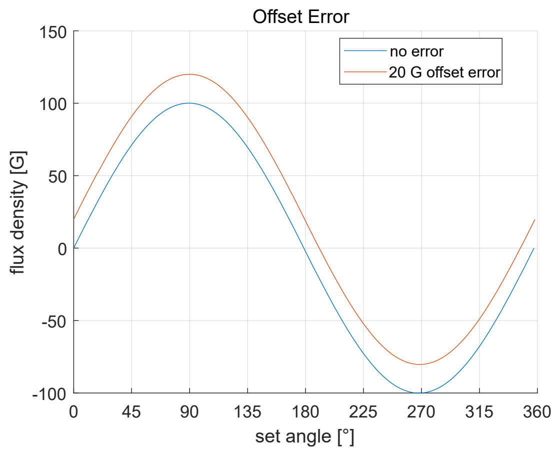

Figure 1: Example Offset Error

Sensitivity Error

The sensitivity error is a change in sensitivity compared to its intended sensitivity and is measured in percent. The field with a sensitivity error looks like it is scaled up or down. Since the sensitivity error is measured in percent, the impact on performance is not related to the amplitude of applied magnetic field.

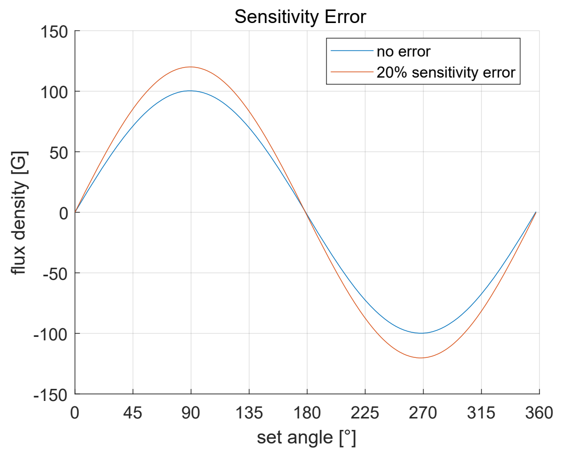

Figure 2: Example Sensitivity Error

Channel Sensitivity Mismatch Error

Sensitivity mismatch is the deviation of each channel from the average sensitivity of the two selected channels.

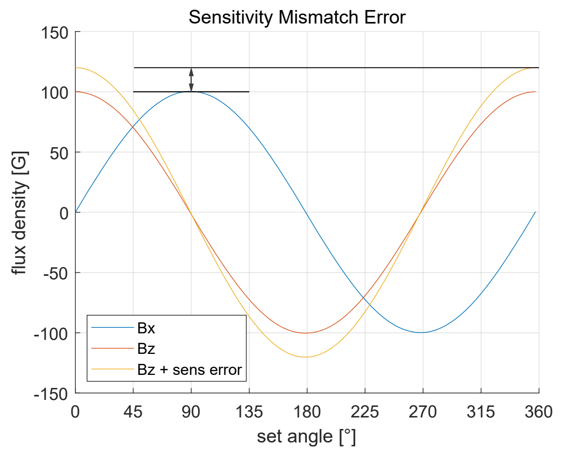

Figure 3: Example Channel Sensitivity Mismatch Error

Channel Orthogonality Error

Orthogonality error is the degree to which the two selected channels are not orthogonal. Normally, the phase difference should be 90°, but due to the orthogonality error, this is not the case.

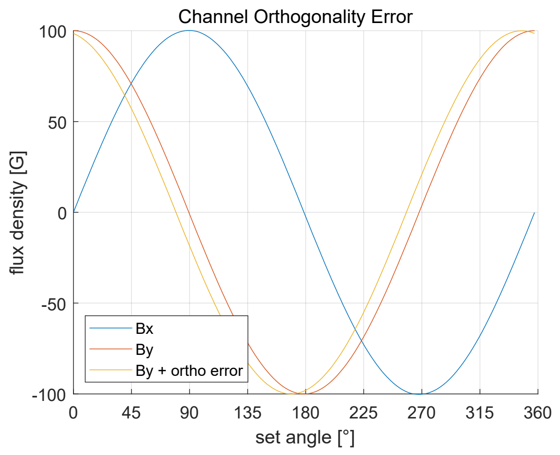

Figure 4: Example Channel Orthogonality Error

Temperature Drift

Temperature drift is in this case a change in ambient temperature from 20°C, and the temperature drift for an error source is therefore zero at 20°C. If the temperature changes from 20°C, the sensor error sources will drift relative to the applied temperature.

Temperature drift of error sources may be the sole consideration in the case where calibration is employed, e.g. linearization, at end of line. Factory calibration is normally performed at room temperature (20°C) and will eliminate (in large part) the effect of absolute individual error sources at this temperature, making temperature drift of error sources the most relevant to analyze for the application.

Temperature drift is valid for all sensor error sources explained previously.

Lifetime Drifts

At Allegro MicroSystems, the lifetime of a sensor is defined by the environmental stress tests performed during an AEC-Q100 qualification. During qualification, the sensor will experience numerous temperature cycles and many hours of operation at extreme temperatures. Sensor error sources are measured before and after environmental stress, and the difference between these measurements is the lifetime drift for each error source. This approach is taken for all sensor error sources explained previously.

Error Function Input Parameters

The sensor error function has some parameters to allow it to be used in a wide range of use cases.

function [B_Err_ChnA, B_Err_ChnB] = A31315_BwithError(cordic, B_chnA, B_chnB, Temperature, LinUsed, n_samples, errorOverLife

cordic

The parameter *cordic* selects which of the two magnetic fields should be used, either 'XY' or 'XZ'. Other orientations are currently not supported by this sensor and function. If this parameter is set to 'XZ' it means that *B_chnA* will be interpreted as channel 'X', and *B_chnB* will be interpreted as channel 'Z' of the sensor. Depending on the set value for *cordic*, different sensor errors will be used. This is because each orientation has different error source contributions.

B_chnA and B_chnB

The parameters *B_chnA* an *B_chnB* are the magnetic field values in gauss without sensor errors. The field must be given in a vector, e.g. field over rotation or movement.

In Figure 5, Bx has a size of [360 1] and has 360 values for the angle rotation from 0° to 360°. In this specific example, the maximum value in this matrix has an amplitude of ±200 gauss.

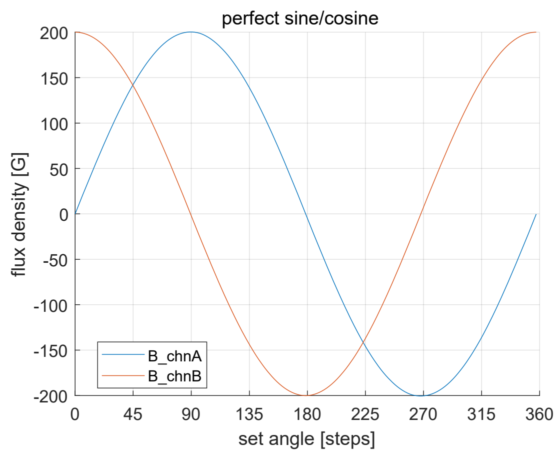

Figure 5: Example of magnetic field for a rotation

Temperature Error

The parameter Temperature is the temperature in °C. This parameter is needed since the sensor errors are temperature dependent.

For example, the offset error may be worse at around 100°C, while the sensitivity error may be worse at –40°C. The magnetic input field will not be temperature-corrected. This needs to be done before calling this function and will be shown in an example later.

LinUsed

If the output data will be used with a linearization or end-of-line calibration, the parameter *LinUsed* should be set to '1'. The function will then only consider the sensor error drift, outgoing from 20°C. The output of the function does not exactly match the performance with linearization applied and is only an approximation. The function is not doing the linearization itself.

LinUsed = false:

Error over temperature will be used

LinUsed = true:

Sensor temperature drift will be used

n_samples

The parameter *n_samples* is the number of samples which should be created for the Monte Carlo analysis. Set to '1000', the function will provide 1000 values with sensor errors. It is recommended to use a value like '1000' or higher to get a good distribution of sensor errors. These values then can be used afterwards to calculate e.g. 3-sigma values and look at the distribution of the channel or angle errors.

errorOverLife

The parameter *errorOverLife* can be set to '1' to add lifetime drift as defined by an AEC-Q100 qualification. If set to '0', only temperature errors and drifts will be considered.

ERROR FUNCTION OUTPUT VALUES

The function returns the magnetic fields with added errors: *B_Err_ChnA* and *B_Err_ChnB*. For each value of the input, the function will return *n_samples* calculated values.

In Figure 5, a Bx and Bz with a size of [360 1] is used. After using the sensor error function, the output of *B_Err_ChnA* has a size of [1000 360]. For each initial value, there are now 1000 sample values with added sensor errors. The result can be seen in Figure 6. Instead of just one line for each magnetic field, there are now 1000, providing a distribution of the sensor error performance.

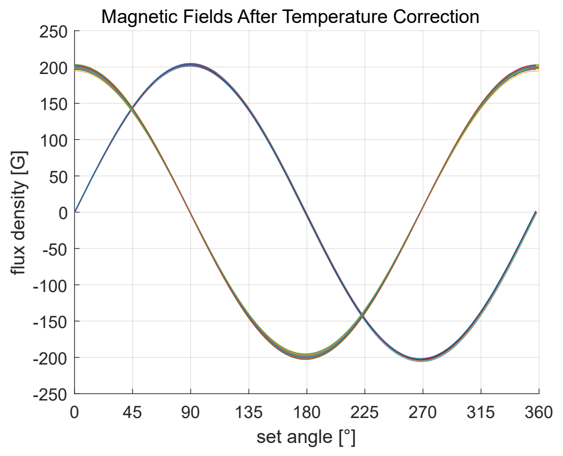

Figure 6: Example Output Of Error Function

It is important to note, that the result is **not** representative of *n_samples* individual sensors. Each calculated value is not based on the previous value/error. Therefore, the result from this function provides the distribution of errors but no single simulated sensor errors.

HOW TO USE

Procedure

1. Get your magnetic fields from a …
   A. Simulation
   B. Calculation
   C. Measurement
2. Correct the magnetic field regarding the desired temperature and magnet temperature coefficient
3. Use the sensor error function to add sensor errors to the signal
4. Process the data
   A. Calculate the angle
   B. Calculate the angle error
   C. Calculate mean/sigma lines
5. Plot the result

MATLAB Code and Explanation

This section will guide through the procedure described above

Generate Data

As an example, create a perfect sine and cosine signal for an 'XY' configuration with an amplitude of 300 G. This would be comparable to an end shaft application as shown in Figure 7.

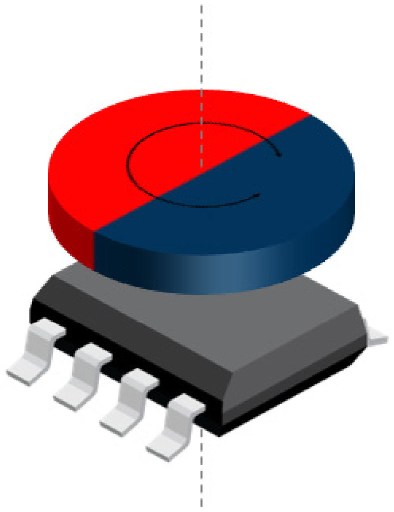
Figure 7: Example for an End-of-Shaft application

```
% create a perfect sine/cosine
% with an amplitude of 300G
Amp_G = 300; % amplitude in Gauss
steps = 360; % # of data points for rotation
B_chnX = cos(linspace(0,2*pi,steps))' .* Amp_G;
B_chnY = sin(linspace(0,2*pi,steps))' .* Amp_G;
```

The next step is to create the values for the X axis. This is nearly the same as the first step:

```
x_values = linspace(0,360,steps);
```

The result shown in Figure 8 is comparable to the following simulation parameters:

- Round magnet OD = 7 mm
- Round magnet Height = 3 mm
- Magnetization = diametral
- Br@20°C = 0.405T (Ferrite)
- Simulation temperature = 20°C
- Crystal Air Gap = 2.1 mm

(magnet surface to active Hall element)

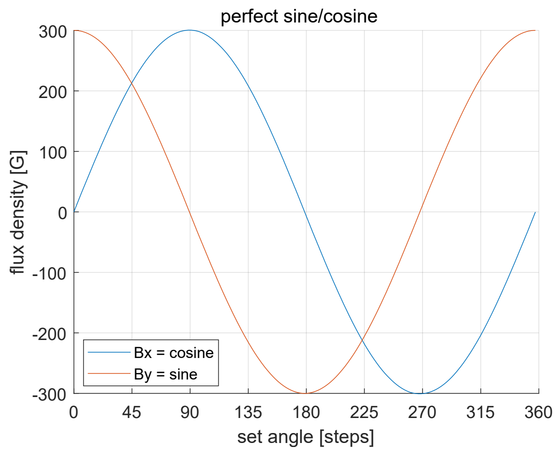

Figure 8: Calculated perfect simulation data

In a next step, the angle for the data above will be calculated (angle shown in Figure 9).

```
% calc angle from Bx/By signals
perfect_ang = mod(atan2d(B_chnY, B_chnX),360);
```

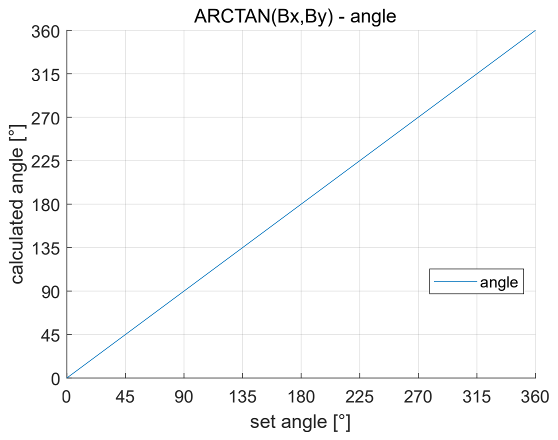

Figure 9: Calculated angle from perfect simulation data

Correct field regarding temperature

The next step would be to correct the magnetic fields regarding the temperature. For this example, define temperature range as –40°C to 150°C and assume that the magnetic field in Figure 8 is at 20°C. The magnet temperature coefficient will be –0.19%/°C in this example, which is comparable to a typical ferrite magnet. This value is related to the magnet material and can be found in the datasheet of the magnet.

```
% Temperature correction
SimTemp = 20;      % °C temperature for original data
Temp = -40:10:150; % °C temperature range in 10°C steps
magnetTc = -0.19;  % %/°C
```

```
% create empty array with size of the expected result
[Bx_overTemp, By_overTemp] = ...
deal(zeros(numel(Temp), numel(B_chnX)));

for t_idx = 1:length(Temp) % for all temperatures
  % calc the new field for the temperatue and magnet Tc
    Bx_overTemp(t_idx,:) = B_chnX .* ((100 - ...
(SimTemp-Temp(t_idx))*magnetTc)/100);
    By_overTemp(t_idx,:) = B_chnY .* ((100 - ...
(SimTemp-Temp(t_idx))*magnetTc)/100);
End
The result of the temperature correction can be seen in Figure 10.
```

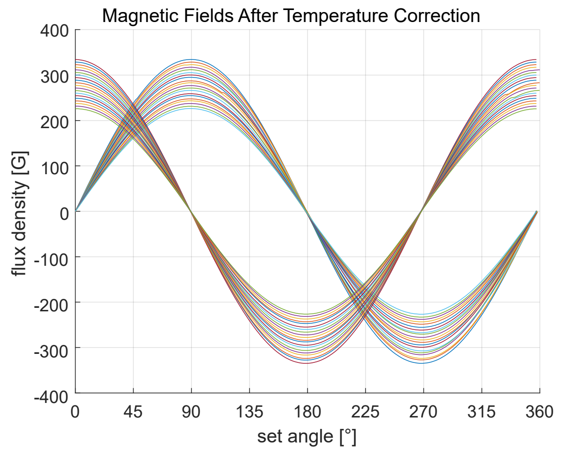

Figure 10: Temperature corrected magnetic field

Using the Error Function

The next step is actually using the A31315 error function. To do this, some missing parameters must be defined:

```
cordic = ‘XY’;     % use function for XY orientation
LinUsed = false;   % no linearization afterwards
n_samples = 1000;  % 1000 samples should be calculated
LifeError = false; % no lifetime errors
```

After defining the parameters, the function can be used for each temperature. The result will be stored in *Bx_Err* and *By_Err*.

```
% add sensor errors for each temperature
for t_idx = 1:length(Temp)
    [Bx_Err(t_idx,:,:), By_Err(t_idx,:,:)] = ...
A31315_BwithError(...
cordic, Bx_overTemp(t_idx,:), ...
By_overTemp(t_idx,:), Temp(t_idx), LinUsed, ...
n_samples, LifeError);
end
```

The function provides in this case 1000 values for each temperature and position (Figure 11). It is recommended to limit the number of lines plotted to avoid slowdowns, e.g. only plot the first 50 lines.

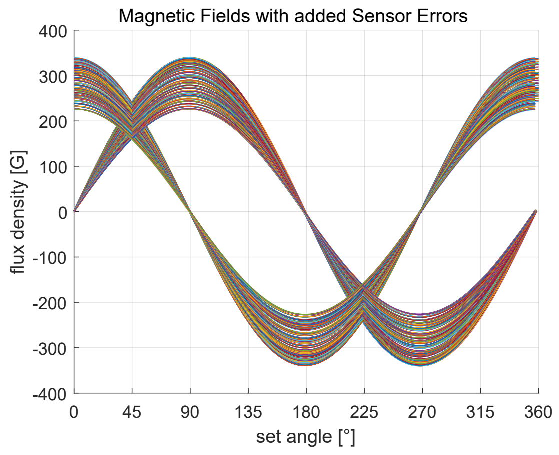

Figure 11: Temperature corrected magnetic field with added sensor errors

Calculate Angle Error

The next step is to calculate the angle error. Therefore, the angles for each temperature, position, and samples are needed. Afterwards, the initial angle without errors will be subtracted (Figure 12).

```
% calculate angle with sensor errors
Ang_wErr = mod(atan2d(By_Err, Bx_Err), 360);

% calculate angle error
for t_idx = 1:length(Temp)
    for s_idx = 1:n_samples
        Ang_Err(t_idx, s_idx,:) = ...
wrapTo180(squeeze(Ang_wErr(t_idx, s_idx,:)) ...
- perfect_ang);
    end
end
```

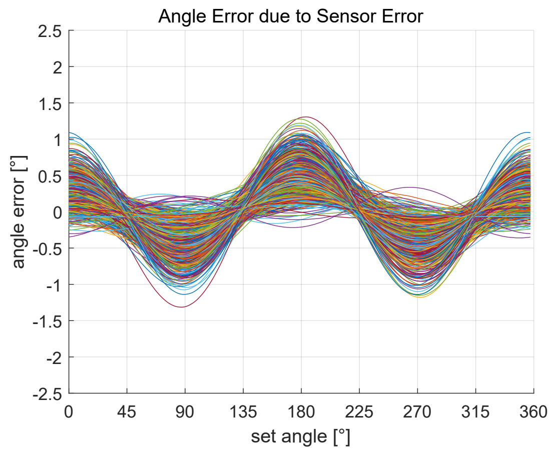

Figure 12: Angle error due to simulated sensor errors

Adding mean and sigma lines

This step is already optional, but since this is a Monte Carlo based simulation, the mean and sigma lines are a good way to check the result and overall performance. The mean and sigma values will be calculated over all temperatures and samples, but individually for each position (Figure 13).

```
% define the factor for sigma line
Sigma = 3; % e.g. 3 = 3-Sigma

% calculate mean and sigma values
Ang_Err_mean = squeeze(mean(Ang_Err, [1 2]));
Ang_Err_sigma = Sigma * squeeze(std(Ang_Err, 0, [1 2]));
```

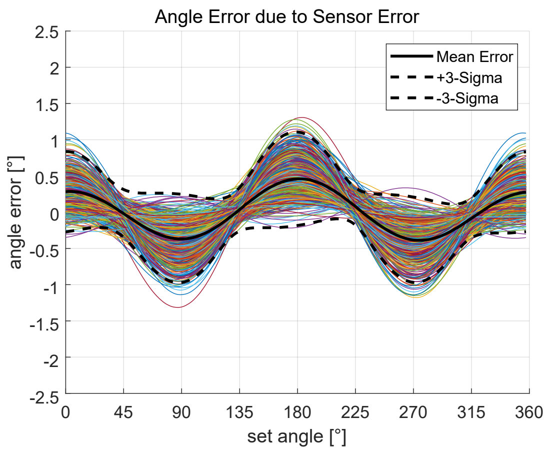

Figure 13: Angle error with calculated mean and sigma lines

Linearization

Though this document does not show the user how to linearize a sensor, the sensor error function can still be used. For that case, linearization coefficients must be calculated for the raw magnetic signals at 20°C. Then the parameter '*LinUsed*' nmust be set to 'true' for the error function. After calculating all the values, the initial calculated linearization coefficients must be applied on the result from the error function.

For information on how to perform a linearization with a microcontroller, please see the application note AN296160 on the Allegro MicroSystems website.

CONCLUSION

Datasheet specifications do not always lend themselves easily to statistical modeling or reflect the dependency of each error source across the operating temperature range. Some error sources peak in value at low temperatures and others peak at high temperature. The ultimate performance of the sensor is the superposition of various error sources, and the error function described in this document helps the user estimate how the sensor will perform in their application.

APPENDIX: FULL MATLAB LIVESCRIPT CODE (WITH FIGURES)

Create Test Data

```
close all;
clear; clc;

% create a perfect sine/cosine - can be replaced with simulation/mapping data
% with an amplitude of 300G
Amp_G = 300; % in Gauss
steps = 360; % # of data points for rotation

B_chnX = cos(linspace(0,2*pi,steps))' .* Amp_G;
B_chnY = sin(linspace(0,2*pi,steps))' .* Amp_G;
x_values = linspace(0,360,steps);

xlim_range = [0 360];
xticks_val = 0:45:360;

figure
hold on; grid on;
plot(x_values, B_chnX);
plot(x_values, B_chnY);
title('perfect sine/cosine');
xlabel('set angle [steps]');
ylabel('flux density [G]');
xlim(xlim_range); xticks(0:45:360);
legend('Bx = cosine', 'By = sine', 'location', 'best');

% calc angle from Bx/By signals
perfect_ang = mod(atan2d(B_chnY, B_chnX),360);

figure
nexttile
% plot perfect angle
hold on; grid on;
plot(perfect_ang,perfect_ang);
xlim(xlim_range); xticks(xticks_val);
ylim(xlim_range); yticks(xticks_val);
title('ARCTAN(Bx,By) - angle');
xlabel('set angle [°]');
ylabel('calculated angle [°]');
legend('angle', 'location', 'best');
```

Temperature Correction

```
% Settings Temperature correction
SimTemp = 20; % °C temperature for original data
Temp = -40:10:150; % °C temperature range in 10°C steps
magnetTc = -0.19; % %/°C

% create empty array with size of the expected result
[Bx_overTemp,By_overTemp] = deal(zeros(numel(Temp), numel(B_chnX)));

for t_idx = 1:length(Temp) % for all temperatures
  % calc the new field for the temperatue and magnet Tc
    Bx_overTemp(t_idx,:) = B_chnX .* ((100 - (SimTemp-Temp(t_idx))*magnetTc)/100);
    By_overTemp(t_idx,:) = B_chnY .* ((100 - (SimTemp-Temp(t_idx))*magnetTc)/100);
end

figure
hold on; grid on;
plot(perfect_ang, Bx_overTemp);
plot(perfect_ang, By_overTemp);
title('Magnetic Fields After Temperature Correction');
xlabel('set angle [°]');
ylabel('flux density [G]');
xlim(xlim_range); xticks(0:45:360);
legend('Bx over T', 'By over T', 'location', 'best');
```

Calculate Errors

```
cordic = ‘XY’;      % use function for XY orientation
LinUsed = false;    % no linearization will be used afterwards
n_samples = 1000;   % 1000 samples should be calculated
LifeError = false;  % no lifetime errors

% add sensor errors for each temperature
for t_idx = 1:length(Temp)
    [Bx_Err(t_idx,:,:), By_Err(t_idx,:,:)] = A31315_BwithError(cordic, Bx_overTemp(t_idx,:), By_overTemp(t_idx,:), Temp(t_idx), LinUsed, n_samples,
LifeError);
end

figure
hold on; grid on;
for t_idx = 1:length(Temp)
    plot(perfect_ang, squeeze(Bx_Err(t_idx,1:50,:)));
    plot(perfect_ang, squeeze(By_Err(t_idx,1:50,:)));
end
title(‘Magnetic Fields with added Sensor Errors’);
xlabel(‘set angle [°]’);
ylabel(‘flux density [G]’);
xlim(xlim_range); xticks(0:45:360);
legend(‘Bx with error’, ‘By with error’, ‘location’, ‘best’);
```

Calculate Angle Error

```
% calculate angle with sensor errors
Ang_wErr = mod(atan2d(By_Err, Bx_Err), 360);

% calculate angle error
for t_idx = 1:length(Temp)
    for s_idx = 1:n_samples
        Ang_Err(t_idx, s_idx,:) = wrapTo180(squeeze(Ang_wErr(t_idx, s_idx,:)) - perfect_ang);
    end
end

figure
hold on; grid on;
for t_idx = 1:length(Temp)
    plot(perfect_ang, squeeze(Ang_wErr(t_idx,1:50,:)));
end
title(‘Angles with Sensor Errors’);
xlabel(‘set angle [°]’);
ylabel(‘angle with error [°]’);
xlim(xlim_range);xticks([0:45:360]);

figure
hold on; grid on;
for t_idx = 1:length(Temp)
    plot(perfect_ang, squeeze(Ang_Err(t_idx,1:50,:)));
end
title(‘Angle Error due to Sensor Error’);
xlabel(‘set angle [°]’);
ylabel(‘angle error [°]’);
xlim(xlim_range); xticks(0:45:360);
ylim([-2.5 2.5])
```

Calculation & Adding Mean/3-Sigma Lines

```
% define the factor for sigma line
Sigma = 3; % e.g. 3 = 3-Sigma

%calculate mean and sigma values
Ang_Err_mean = squeeze(mean(Ang_Err, [1 2]));
Ang_Err_sigma = Sigma*squeeze(std(Ang_Err, 0, [1 2]));

figure
hold on; grid on;
for t_idx = 1:length(Temp)
    plot(perfect_ang, squeeze(Ang_Err(t_idx,1:50,:)));
end
h1=plot(perfect_ang, Ang_Err_mean,’-k’,’LineWidth’,2);
h2=plot(perfect_ang, Ang_Err_mean+Ang_Err_sigma,’--k’,’LineWidth’,2);
h3=plot(perfect_ang, Ang_Err_mean-Ang_Err_sigma,’--k’,’LineWidth’,2);
title(‘Angle Error due to Sensor Error’);
xlabel(‘set angle [°]’);
ylabel(‘angle error [°]’);
xlim(xlim_range); xticks(0:45:360);
ylim([-2.5 2.5])
legend([h1 h2 h3], ‘Mean Error’, [‘+’ num2str(Sigma) ‘-Sigma’], [‘-’ num2str(Sigma) ‘-Sigma’], ‘location’, ‘best’)
```

Revision History

Number	Date	Description	Responsibility
–	March 22, 2021	Initial release	Fabian Winkler