ESRF Double Crystal Monochromator - Lookup Tables

Dehaeze Thomas

December 8, 2021

Contents

Several Lookup Tables (LUT) are used for the DCM in order to compensate for **repeatable** errors.

- Section 1: the stepper motors are calibrated using interferometers.
- Section 2: the Attocube periodic non-linearities are calibrated using piezoelectric actuators.

1 Stepper Motors Calibration

Fast jack coarse displacement is performed with a Stepper motor and a ball screw mechanism. Such positioning system has some repeatable motion errors than can be calibrated using a measurement system having less errors.

For the DCM, this can be done using the interferometers.

1.1 Schematic

In order to measure the errors induced by the fast jacks, we have to make some scans, and measure simultaneously:

- The wanted fast jack position: signal/step sent by the IcePAP
- The actual (measured) position

The experimental setup to perform this is shown in Figure 1.1.

The procedure is the following:

- A scan on the Bragg angle θ is generated from Bliss
- Reference paths $[r_{u_r},\ r_{u_h},\ r_d]$ are sent to the IcePAP
- Initially, the LUT inside the IcePAP is not changing the reference path
- The IcePAP generates some steps $[u_{u_r},\ u_{u_h},\ u_d]$ that are sent to the fast jacks
- The motion of the crystals $[d_z,\ r_y,\ r_x]$ is measured with the interferometers and computed in the Speedgoat
- Finally, the corresponding motion $[d_{u_r},\ r_{u_h},\ r_d]$ of the fast jack is computed afterwards in BLISS

The measured motion of the fast jacks $[d_{u_r},\ r_{u_h},\ r_d]$ can be compared with the IcePAP steps $[u_{u_r},\ u_{u_h},\ u_d]$ in order to create the LUT inside the IcePAP.

1.2 Patterns in the Fast Jack motion errors

In order to understand what should be the "sampling distance" for the lookup table of the stepper motor, we have to analyze the displacement errors induced by the stepper motor.

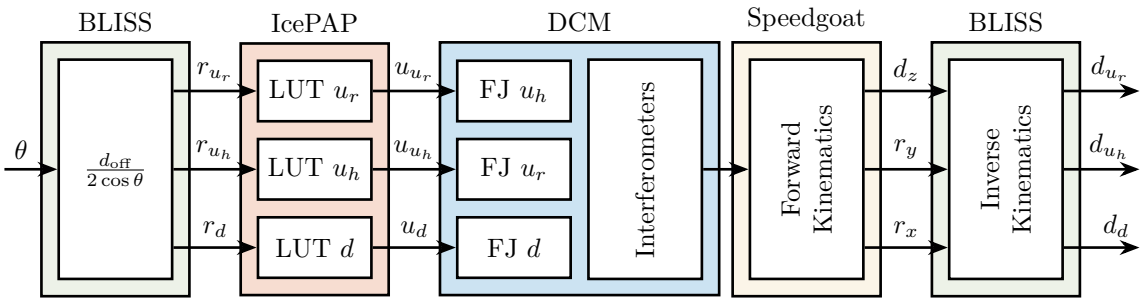


Figure 1.1: Block diagram of the experiment to create the Lookup Table

Let's load the measurements of one bragg angle scan without any LUT.

Matlab

```
%% Load Data of the new LUT method
ol_bragg = (pi/180)*1e-5*double(h5read('Qutools_test_0001.h5','/33.1/instrument/trajmot/data')); % Bragg angle [rad]
ol_dzw   = 10.5e-3./(2*cos(ol_bragg)); % Wanted distance between crystals [m]

ol_dz    = 1e-9*double(h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_dz_filter/data'));  % Dz distance between
↪  crystals [m]
ol_dry   = 1e-9*double(h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_dry_filter/data')); % Ry [rad]
ol_drx   = 1e-9*double(h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_drx_filter/data')); % Rx [rad]

ol_t = 1e-6*double(h5read('Qutools_test_0001.h5','/33.1/instrument/time/data')); % Time [s]

ol_ddz   = ol_fj-ol_dz; % Distance Error between crystals [m]
```

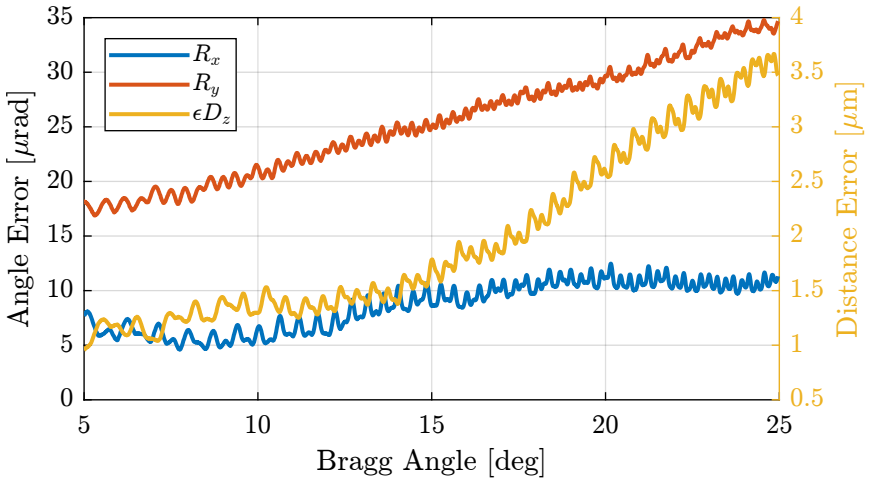


Figure 1.2: Orientation and Distance error of the Crystal measured by the interferometers

Now let's convert the errors from the frame of the crystal to the frame of the fast jacks (inverse kinematics problem) using the Jacobian matrix.

Matlab

```
%% Compute Fast Jack position errors
% Jacobian matrix for Fast Jacks and 111 crystal
J_a_111 = [1,  0.14, -0.1525
           1,  0.14,  0.0675
           1, -0.14,  0.0425];

ol_de_111 = [ol_ddz'; ol_dry'; ol_drx'];
```

```
% Fast Jack position errors
ol_de_fj = J_a_111*ol_de_111;

ol_fj_ur = ol_de_fj(1,:);
ol_fj_uh = ol_de_fj(2,:);
ol_fj_d  = ol_de_fj(3,:);
```

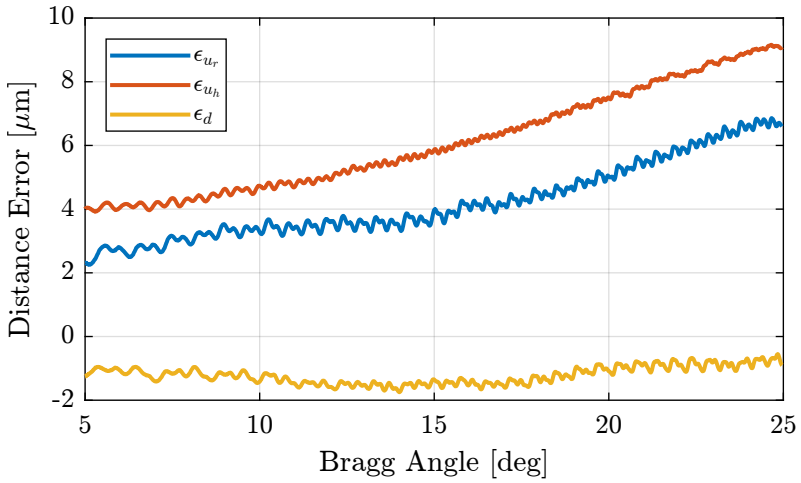


Figure 1.3: Estimated motion errors of the fast jacks during the scan

Let's now identify this pattern as a function of the fast-jack position.

As we want to done frequency Fourier transform, we need to have uniform sampling along the fast jack position. To do so, the function `resample` is used.

Matlab
```
Xs = 0.1e-6; % Sampling Distance [m]

%% Re-sampled data with uniform spacing [m]
ol_fj_ur_u = resample(ol_fj_ur, ol_dzw, 1/Xs);
ol_fj_uh_u = resample(ol_fj_uh, ol_dzw, 1/Xs);
ol_fj_d_u  = resample(ol_fj_d,  ol_dzw, 1/Xs);

ol_fj_u = Xs*[1:length(ol_fj_ur_u)]; % Sampled Jack Position
```

The result is shown in Figure 1.4.

Let's now perform a Power Spectral Analysis of the measured displacement errors of the Fast Jack.

Matlab
```
% Hanning Windows with 250um width
win = hanning(floor(400e-6/Xs));

% Power Spectral Density [m2/(1/m)]
[S_fj_ur, f] = pwelch(ol_fj_ur_u-mean(ol_fj_ur_u), win, 0, [], 1/Xs);
[S_fj_uh, ~] = pwelch(ol_fj_uh_u-mean(ol_fj_uh_u), win, 0, [], 1/Xs);
[S_fj_d,  ~] = pwelch(ol_fj_d_u -mean(ol_fj_d_u ),  win, 0, [], 1/Xs);
```

As shown in Figure 1.5, we can see a fundamental "reciprocal length" of $5 \cdot 10^4\,[1/m]$ and its harmonics. This corresponds to a length of $\frac{1}{5 \cdot 10^4} = 20\,[\mu m]$.

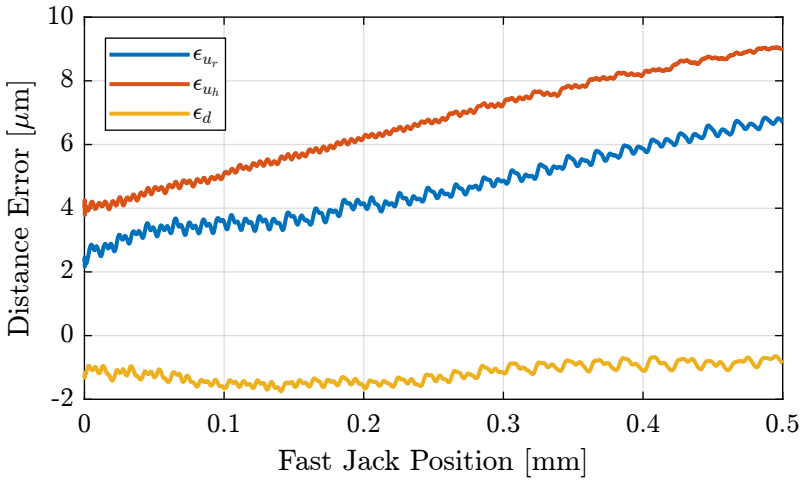


Figure 1.4: Position error of fast jacks as a function of the fast jack motion

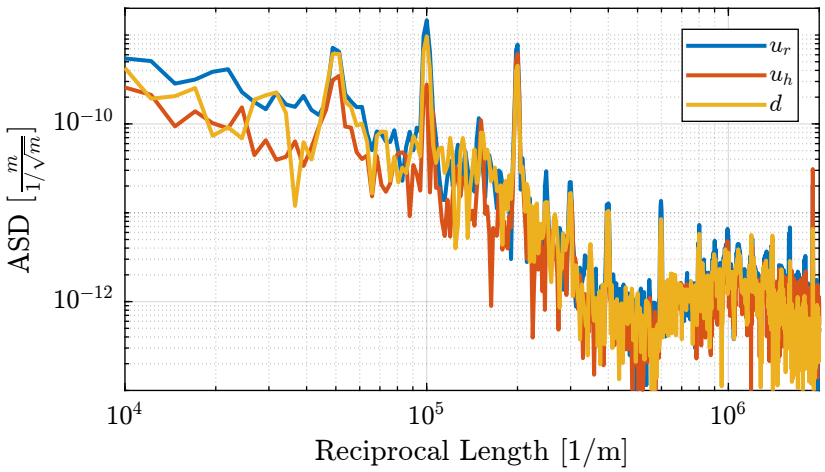


Figure 1.5: Spectral content of the error as a function of the reciprocal length

Instead of looking at that as a function of the reciprocal length, we can look at it as a function of the spectral distance (Figure 1.6).

We see that the errors have a pattern with "spectral distances" equal to $5\,[\mu m]$, $10\,[\mu m]$, $20\,[\mu m]$ and smaller harmonics.

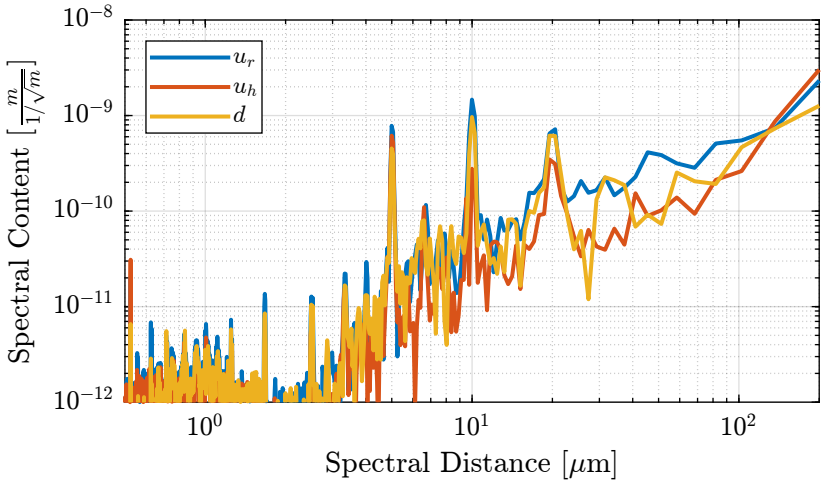


Figure 1.6: Spectral content of the error as a function of the spectral distance

Let's try to understand these results. One turn of the stepper motor corresponds to a vertical motion of 1mm. The stepper motor has 50 pairs of poles, therefore one pair of pole corresponds to a motion of $20\,[\mu m]$ which is the fundamental "spectral distance" we observe.

Matlab
```
CPS_ur = flip(-cumtrapz(flip(f), flip(S_fj_ur)));
CPS_uh = flip(-cumtrapz(flip(f), flip(S_fj_uh)));
CPS_d  = flip(-cumtrapz(flip(f), flip(S_fj_d)));
```

From Figure 1.7, we can see that if the motion errors with a period of $5\,[\mu m]$ and $10\,[\mu m]$ can be dealt with the lookup table, this will reduce a lot the positioning errors of the fast jack.

Matlab
```
%% Cumulative Spectrum
figure;
hold on;
plot(1e6./f, sqrt(CPS_ur), 'DisplayName', '$u_r$');
plot(1e6./f, sqrt(CPS_uh), 'DisplayName', '$u_j$');
plot(1e6./f, sqrt(CPS_d),  'DisplayName', '$d$');
hold off;
set(gca, 'xscale', 'log'); set(gca, 'yscale', 'log');
xlabel('Spectral Distance [$\mu m$]'); ylabel('Cumulative Spectrum [$m$]')
xlim([1, 500]); ylim([1e-9, 1e-5]);
legend('location', 'northwest');
```

1.3 Experimental Data - Current Method

The current used method is an iterative one.

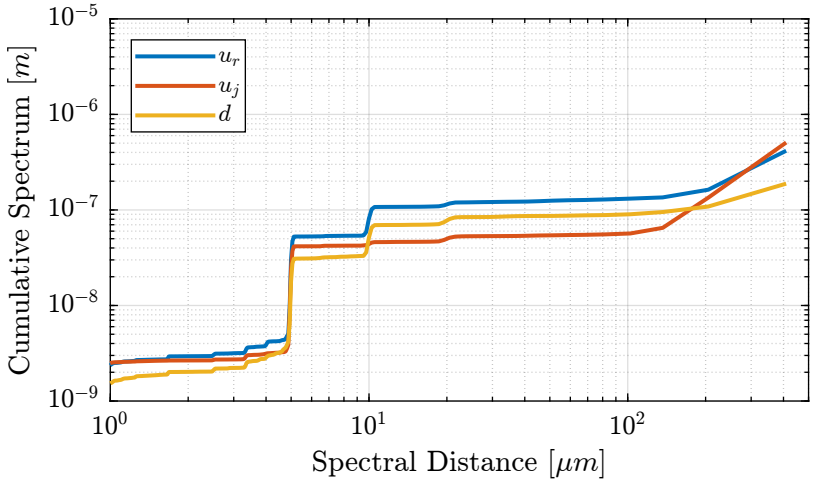


Figure 1.7: Cumulative spectrum from small spectral distances to large spectral distances

Matlab

```
%% Load Experimental Data
ol_bragg    = double(h5read('first_beam_0001.h5','/31.1/instrument/trajmot/data'));
ol_drx      = h5read('first_beam_0001.h5','/31.1/instrument/xtal_111_drx_filter/data');

lut_1_bragg = double(h5read('first_beam_0001.h5','/32.1/instrument/trajmot/data'));
lut_1_drx   = h5read('first_beam_0001.h5','/32.1/instrument/xtal_111_drx_filter/data');

lut_2_bragg = double(h5read('first_beam_0001.h5','/33.1/instrument/trajmot/data'));
lut_2_drx   = h5read('first_beam_0001.h5','/33.1/instrument/xtal_111_drx_filter/data');

lut_3_bragg = double(h5read('first_beam_0001.h5','/34.1/instrument/trajmot/data'));
lut_3_drx   = h5read('first_beam_0001.h5','/34.1/instrument/xtal_111_drx_filter/data');

lut_4_bragg = double(h5read('first_beam_0001.h5','/36.1/instrument/trajmot/data'));
lut_4_drx   = h5read('first_beam_0001.h5','/36.1/instrument/xtal_111_drx_filter/data');
```

The relative orientation of the two `111` mirrors in the x directions are compared in Figure 1.8 for several iterations. We can see that after the first iteration, the orientation error has an opposite sign as for the case without LUT.

1.4 Simulation

In this section, we suppose that we are in the frame of one fast jack (all transformations are already done), and we wish to create a LUT for one fast jack.

Let's say with make a Bragg angle scan between 10deg and 60deg during 100s.

Matlab

```
Fs = 10e3; % Sample Frequency [Hz]
t = 0:1/Fs:10; % Time vector [s]
theta = linspace(10, 40, length(t)); % Bragg Angle [deg]
```

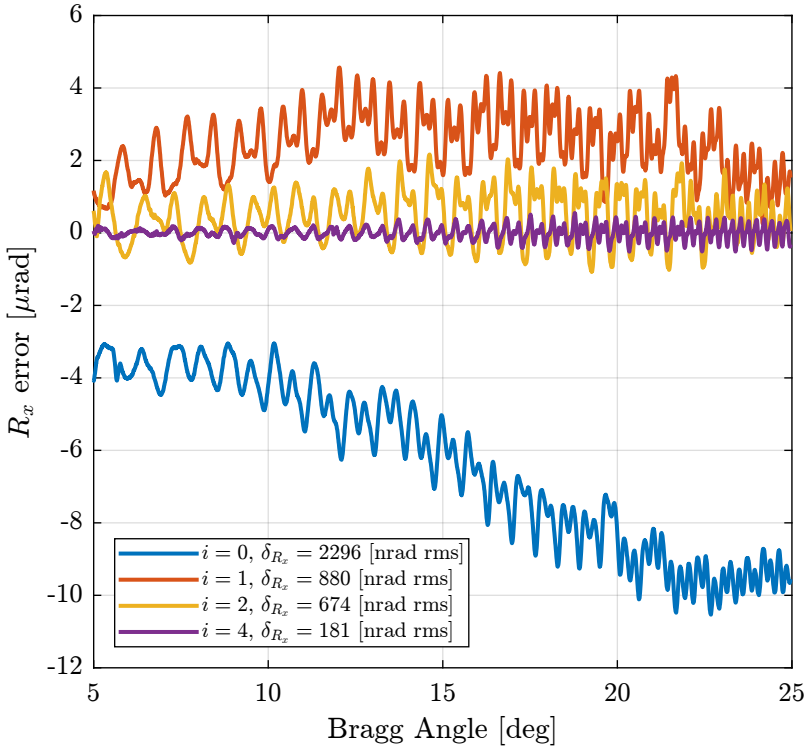


Figure 1.8: R_x error with the current LUT method

The IcePAP steps are following the theoretical formula:

$$d_z = \frac{d_{\text{off}}}{2\cos\theta} \tag{1.1}$$

with θ the bragg angle and $d_{\text{off}} = 10\,mm$.

The motion to follow is then:

```Matlab
perfect_motion = 10e-3./(2*cos(theta*pi/180)); % Perfect motion [m]
```

And the IcePAP is generated those steps:

```Matlab
icepap_steps = perfect_motion; % IcePAP steps measured by Speedgoat [m]
```

Then, we are measuring the motion of the Fast Jack using the Interferometer. The motion error is larger than in reality to be angle to see it more easily.

```Matlab
motion_error = 100e-6*sin(2*pi*perfect_motion/1e-3); % Error motion [m]

measured_motion = perfect_motion + motion_error; % Measured motion of the Fast Jack [m]
```

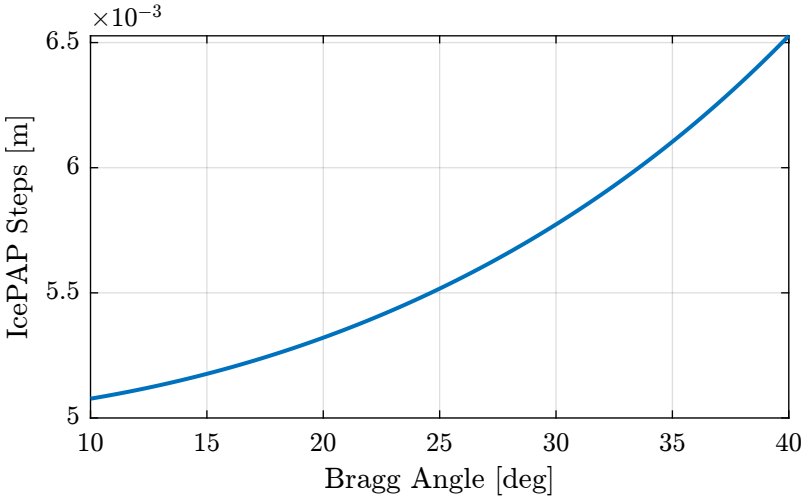


Figure 1.9: IcePAP Steps as a function of the Bragg Angle

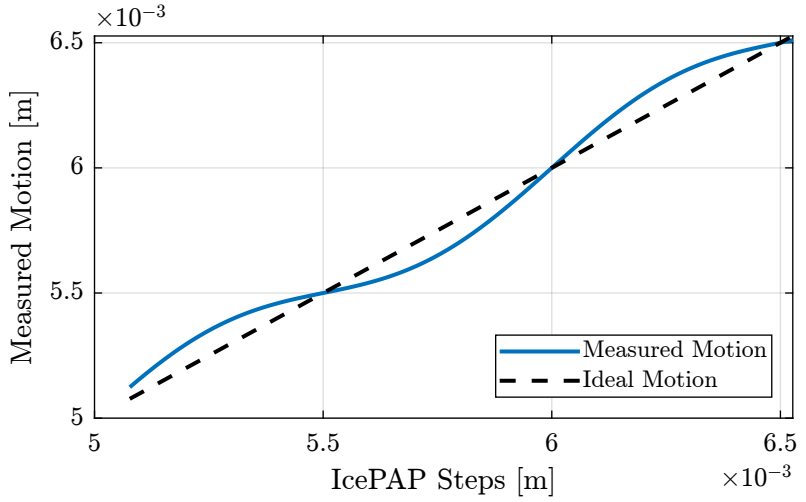


Figure 1.10: Measured motion as a function of the IcePAP Steps

Let's now compute the lookup table. For each micrometer of the IcePAP step, another step is associated that correspond to a position closer to the wanted position.

Matlab
```
%% Get range for the LUT
% We correct only in the range of tested/measured motion
lut_range = round(1e6*min(icepap_steps)):round(1e6*max(icepap_steps)); % IcePAP steps [um]

%% Initialize the LUT
lut = zeros(size(lut_range));

%% For each um in this range
for i = 1:length(lut_range)
    % Get points indices where the measured motion is closed to the wanted one
    close_points = measured_motion > 1e-6*lut_range(i) - 500e-9 & measured_motion < 1e-6*lut_range(i) + 500e-9;
    % Get the corresponding closest IcePAP step
    lut(i) = round(1e6*mean(icepap_steps(close_points))); % [um]
end
```

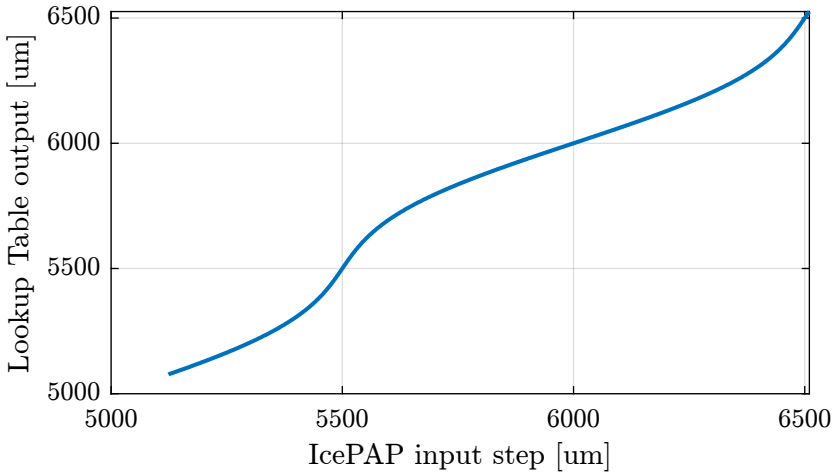


Figure 1.11: Generated Lookup Table

The current LUT implementation is the following:

Matlab
```
motion_error_lut = zeros(size(lut_range));
for i = 1:length(lut_range)
    % Get points indices where the icepap step is close to the wanted one
    close_points = icepap_steps > 1e-6*lut_range(i) - 500e-9 & icepap_steps < 1e-6*lut_range(i) + 500e-9;
    % Get the corresponding motion error
    motion_error_lut(i) = lut_range(i) + (lut_range(i) - round(1e6*mean(measured_motion(close_points)))); % [um]
end
```

Let's compare the two Lookup Table in Figure 1.12.

If we plot the "corrected steps" for all steps for both methods, we clearly see the difference (Figure 1.13).

Let's now implement both LUT to see which implementation is correct.

Matlab
```
motion_new = zeros(size(icepap_steps_output_new));
motion_old = zeros(size(icepap_steps_output_old));
```

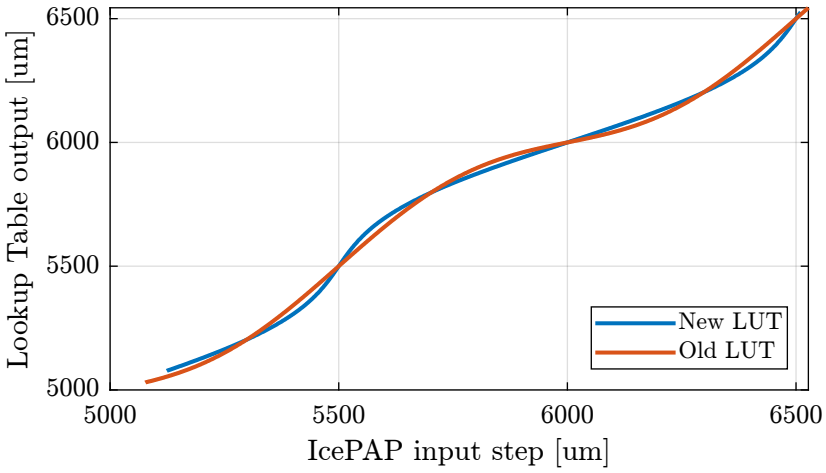


Figure 1.12: Comparison of the two lookup tables

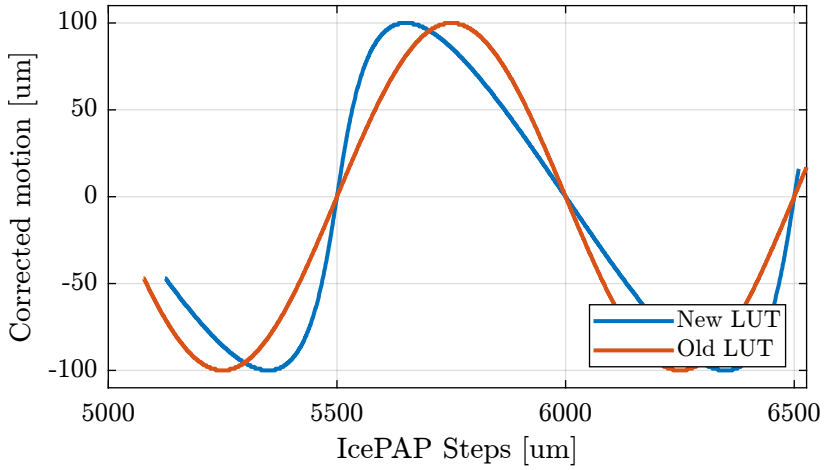


Figure 1.13: LUT correction and motion error as a function of the IcePAP steps

```
for i = 1:length(icepap_steps_output_new)
    [~, i_step] = min(abs(icepap_steps_output_new(i) - 1e6*icepap_steps));
    motion_new(i) = measured_motion(i_step);

    [~, i_step] = min(abs(icepap_steps_output_old(i) - 1e6*icepap_steps));
    motion_old(i) = measured_motion(i_step);
end
```

The output motion with both LUT are shown in Figure 1.14. It is confirmed that the new LUT is the correct one. Also, it is interesting to note that the old LUT gives an output motion that is above the ideal one, as was seen during the experiments.

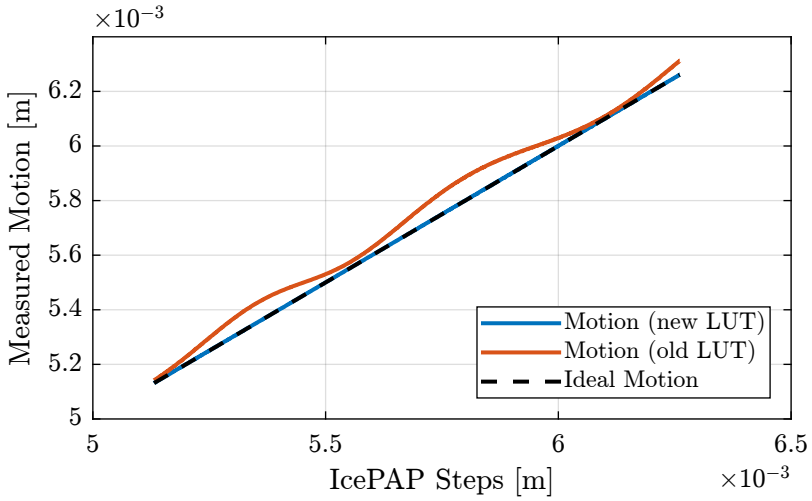


Figure 1.14: Comparison of the obtained motion with new and old LUT

1.5 Experimental Data - Proposed method

The new proposed method has been implemented and tested.

The result is shown in Figure 1.15. After only one iteration, the result is close to the previous method.

Matlab
```
%% Load Data of the new LUT method
ol_new_bragg = double(h5read('Qutools_test_0001.h5','/33.1/instrument/trajmot/data'));
ol_new_drx   = h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_drx_filter/data');

lut_new_bragg = double(h5read('Qutools_test_0001.h5','/34.1/instrument/trajmot/data'));
lut_new_drx   = h5read('Qutools_test_0001.h5','/34.1/instrument/xtal_111_drx_filter/data');
```

If we zoom on the 20deg to 25deg bragg angles, we can see that the new method has much less “periodic errors” as compared to the previous one which shows some patterns.

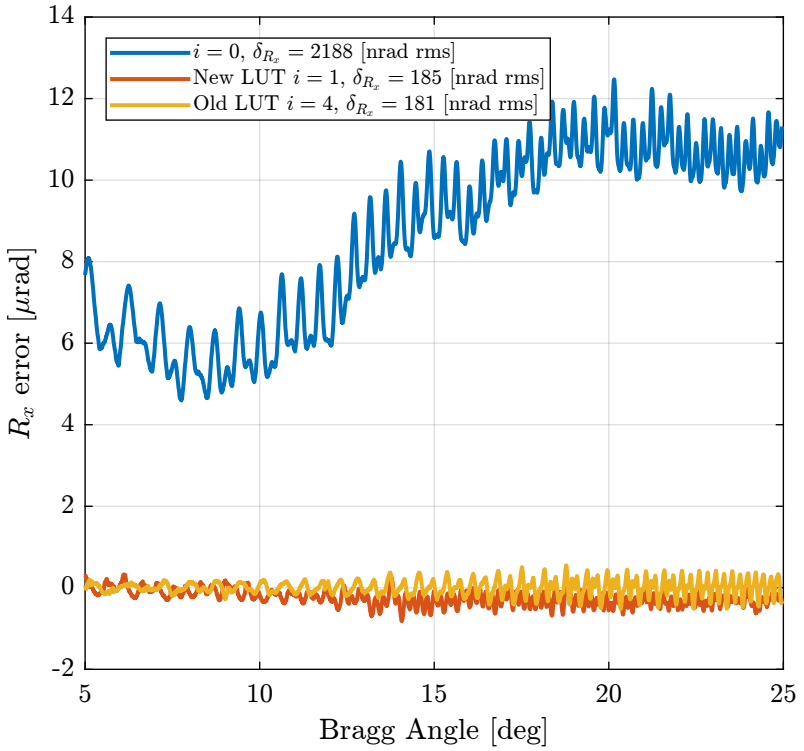
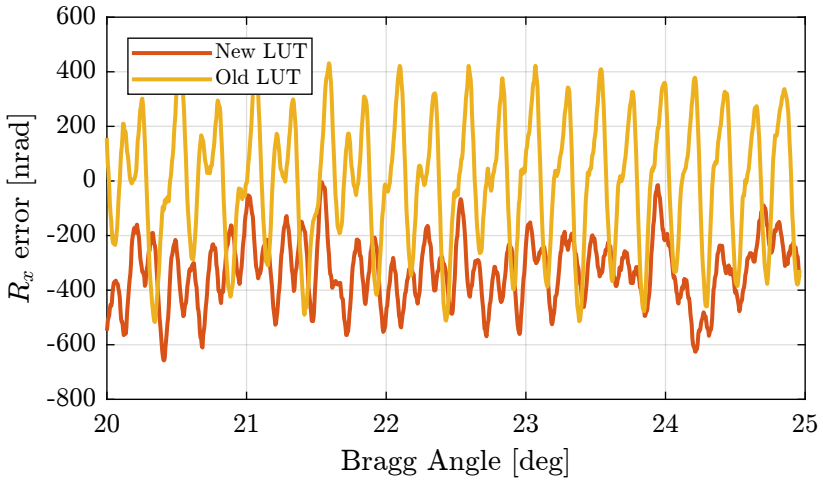


Figure 1.15: Comparison of the R_x error for the current LUT method and the proposed one



2 Attocube Calibration

The idea is to calibrate the periodic non-linearity of the interferometers, a known displacement must be imposed and the interferometer output compared to this displacement. This should be performed over several periods in order to characterize the error.

We here suppose that we are already in the frame of the Attocube (the fast-jack displacements are converted to Attocube displacement using the transformation matrices). We also suppose that we are at a certain Bragg angle, and that the stepper motors are not moving: only the piezoelectric actuators are used.

The setup is schematically with the block diagram in Figure 2.1. The signals are:

- u: Actuator Signal (position where we wish to go)
- d: Disturbances affecting the signal
- y: Displacement of the crystal
- y_g: Measurement of the crystal motion by the strain gauge with some noise n_g
- y_a: Measurement of the crystal motion by the interferometer with some noise n_a

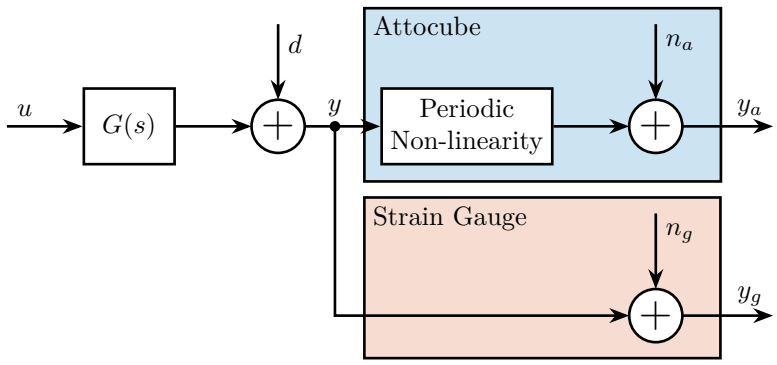


Figure 2.1: Block Diagram schematic of the setup used to measure the periodic non-linearity of the Attocube

The problem is to estimate the periodic non-linearity of the Attocube from the imperfect measurments y_a and y_g.

Then a Lookup Table (LUT) is build.

The wavelength of the Attocube is 1530nm, therefore the non-linearity has a period of 765nm. The amplitude of the non-linearity can vary from one unit to the other (and maybe from one experimental condition to the other). It is typically between 5nm peak to peak and 20nm peak to peak.

2.1 Simulations

We have some constrains on the way the motion is imposed and measured:

- We want the frequency content of the imposed motion to be at low frequency in order not to induce vibrations of the structure. We have to make sure the forces applied by the piezoelectric actuator only moves the crystal and not the fast jack below. Therefore, we have to move much slower than the first resonance frequency in the system.

- As both y_a and y_g should have rather small noise, we have to filter them with low pass filters. The cut-off frequency of the low pass filter should be high as compared to the motion (to not induce any distortion) but still reducing sufficiently the noise. Let's say we want the noise to be less than 1nm (6σ).

Suppose we have the power spectral density (PSD) of both n_a and n_g.

- ☐ Take the PSD of the Attocube

- ☐ Take the PSD of the strain gauge

- ☐ Using 2nd order low pass filter, estimate the required low pass filter cut-off frequency to have sufficiently low noise