Interferometric Mirror Tracking System

DESY Summer Student Programme, 2021

Rodion Zaytsev
Moscow Institute of Physics and Technology, Russia

Supervisor
Dr. Mikhail Lyubomirskiy

September 9, 2021

Abstract

The project consisted in making a mirror tracking system based on interferometrically measured positions of the total external reflective mirror displacement. A piezo actuator drives the mirror with a frequency of approximately 500 hertz. The optical interferometer collects the positions with megahertz frame rate. The data is collected while the mirror is being moved and analyzed afterward.

Contents

1 Introduction

X-rays were discovered by Wilhelm Röntgen in 1895. He noticed that the radiation from the Geisler tube penetrated some materials, such as wood, but not others, such as metal. Taking a picture of his hand, he could see a shadow of his bones. The importance of this discovery was quickly recognized, in particular because it opened new horizons for the medical science. Apart from the absorption properties, it turned out that the diffraction properties of the X-rays can be used to investigate the crystal structures and thus determine the structure of the molecules. Nowadays X-rays are regularly used determine the structure of proteins. The main source for the X-rays since 1970s is synchrotron radiation from the storage rings, which are more brilliant then the early sources by an order of 10^{12} and has boosted the innovation in the X-ray science significantly. Recently X-ray lasers have been introduced and are likely to be the next generation source of X-ray radiation, and will no doubt bring new scientific discoveries. [1] Numerous techniques have been developed for X-ray imaging. Computer axial tomography consists in measuring intensity distribution of an object from different angles. A.M. Cormack suggested a method to reconstruct the image of an object from the intensity distributions [2]. Practically, the method commonly used in medical tomography to collect the intensity distributions at different angles is to fix the object and rotate the source and the detector around it. There are other imaging methods

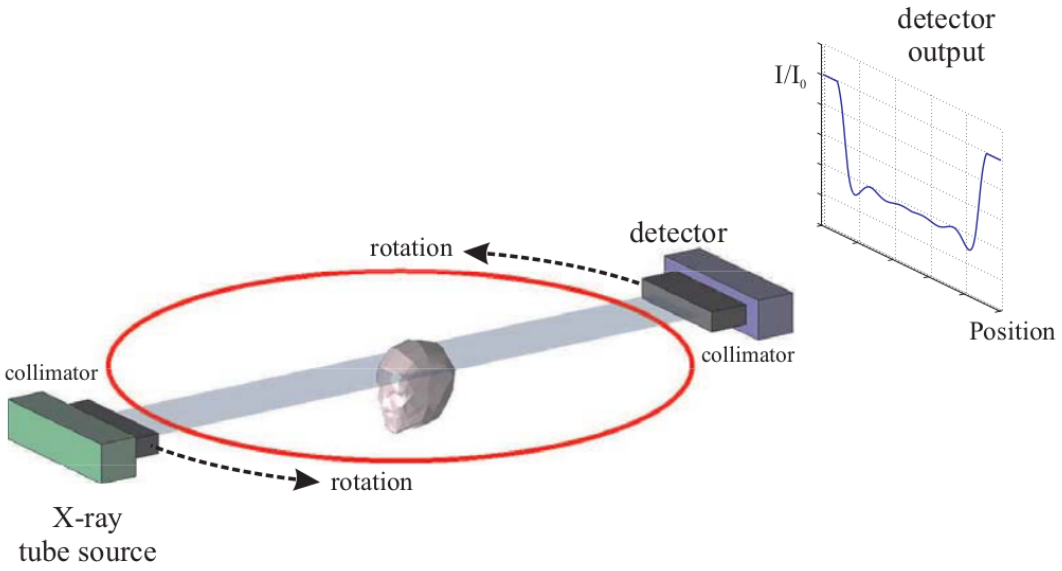


Figure 1: CT scanning scheme [1]

based on diffraction and refraction properties of the X-ray depending on the material. The problem of taking the measurement from different angles, however, is present in all the different techniques. Apart from the method described above when the detector and the source are moved around the object, it is, of course also possible to move the object itself. There are, however problems with these technique. They require moving a significant mass and are therefore inherently slow, besides which there are stability issues. These problems can be solved by introducing a mirror in the way of the beam. It is much easier to move the mirror, and it will provide the possibility to scan the object from different angles quickly

and precisely, as the mirror can be driven by a piezoelectronic actuator which allows for rapid and high precision movements. In the next section the details of the proposed method will be described.

2 Main calculations

Our main goal is to be able to determine the angle at which an X-ray beam hitting the mirror is reflected. A possible way to do this is to set up an interferometer which will measure the mirror displacement. In order to relate the measurement of the interferometer to the angle of the reflected beam, we have to calculate the position of the interferometer relative to the axis of rotation. First, let us determine the connection between the angle of reflection and the angle of rotation of the mirror. Let $\vec{n}$ be the unit normal to the mirror.

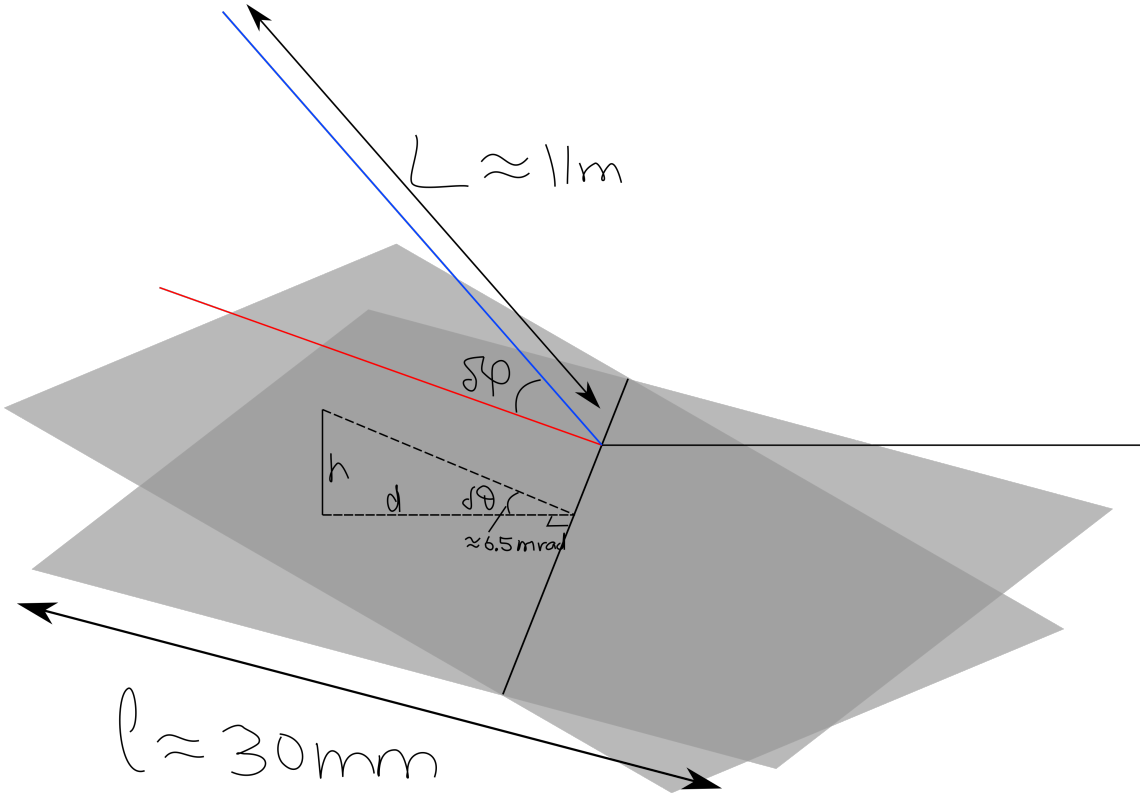


Figure 2: Geometric diagram

Then if $\vec{b}$ is the incident beam, the reflected beam $\vec{b}'$ is given by

$$\vec{b}' = \vec{b} - 2(\vec{b}, \vec{n})\vec{n} \tag{1}$$

If we measure all the angles in radians and use the small angle approximation, we have

$$|\delta\vec{b}'| \approx \delta\varphi, \ |\delta\vec{n}| \approx \delta\theta, \ h = d \cdot \delta\theta \tag{2}$$

Here the relative error is of the order $\delta\theta^2$. Notice that due to geometry $|\delta\varphi| \leq 2|\delta\theta|$, which is why it makes sense to give an upper bound only in terms of $\delta\theta$. We now relate relate $|\delta\vec{b}'|$ and $|\delta\vec{n}|$.

$$\delta\vec{b}' \approx -2(\vec{b}, \delta\vec{n})\vec{n} - 2(\vec{b}, \vec{n})\delta\vec{n} \tag{3}$$

This equation has 2-nd order accuracy, so the relative error will be approximately $\delta\theta$. In the small angle approximation, $\delta\vec{n} \perp \vec{n}$, so taking the norm, we get with a relative accuracy of order $\delta\theta$

$$|\delta\vec{b}'|^2 \approx 4(\vec{b}, \delta\vec{n})^2 + 4(\vec{b}, \vec{n})^2|\delta\vec{n}|^2 \tag{4}$$

It is clear that the right side of equation (4) quadratically depends on $\delta\theta$, whereas the left side quadratically depends on h. We thus have a linear dependence

$$C = \frac{\delta\varphi}{h}$$

with relative accuracy of order $\delta\theta$. Having calculated this constant enables us to determine the angle of the beam from the interferometer data

$$\delta\varphi = Ch$$

The angle range in the experiment is $\delta\theta \sim 10^{-3}$ radians, so the relative error will not exceed 0.1%, which, is sufficient for this experiment (because the experimental error when measuring the angle $\delta\varphi$ will be of the order 1% due to finite pixel size of the X-ray detector)

3 Experimental data

In this section an example of experimental data will be given and the program will be applied to process it. The program code is provided in the appendix.

3.1 Interferometer data

The following data was collected by the interferometer while the mirror was driven by a piezo actuator with a 100mV voltage amplitude sinusoidal signal. Python code was used

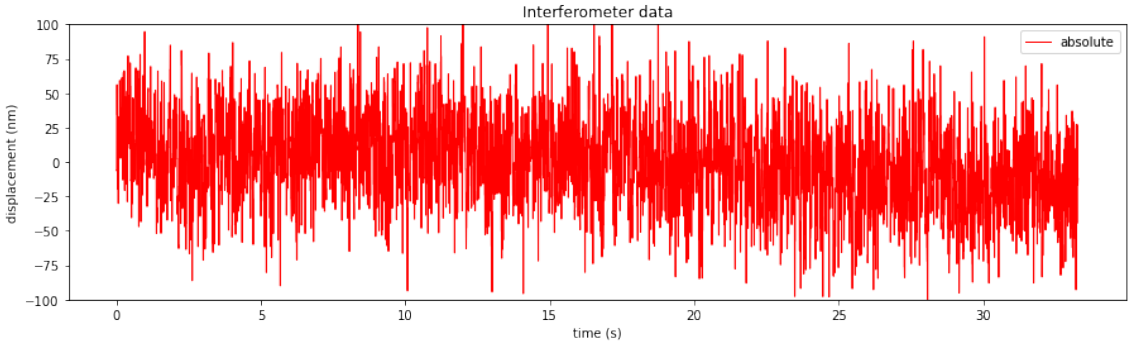


Figure 3: Data from the interferometer

to process the data and determine intervals according to the amplitude and frequency. The amplitude can be calculated as the maximum over an interval. Once the amplitude is determined, it is possible to calculate the frequency by counting the number of nodes and dividing the time interval by that number.

3.2 Data from the X-ray detector

The X-ray detector consists of 75μm pixels which measure the intensity of the falling beam. A sample of data is given below

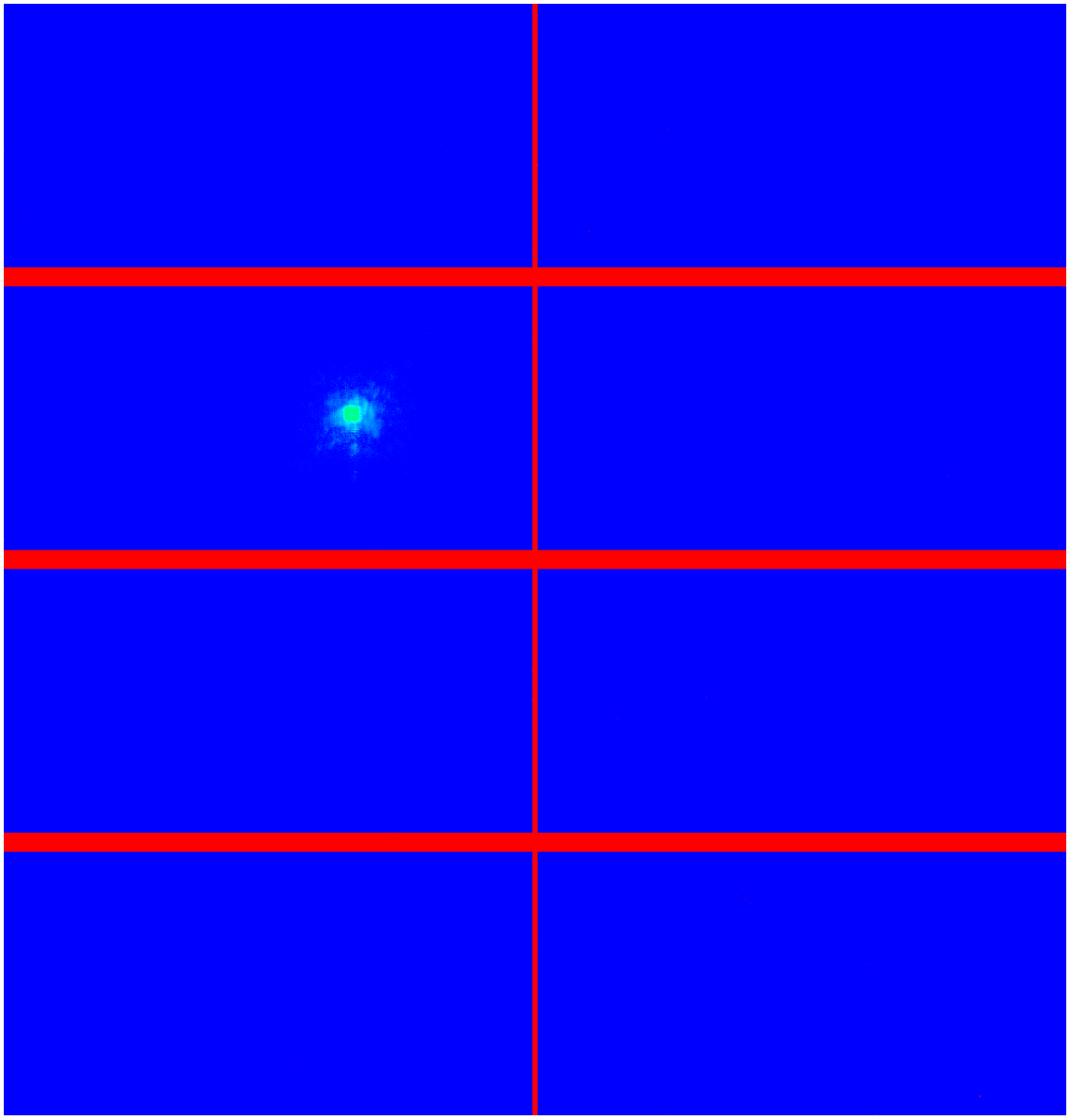


Figure 4: X-ray detector data

A python script calculates the position of the center of the beam. Technically, it calculates the center of mass (here intensity plays the role of mass).

3.3 Program workflow

The diagram below schematically illustrates the workflow of the program.

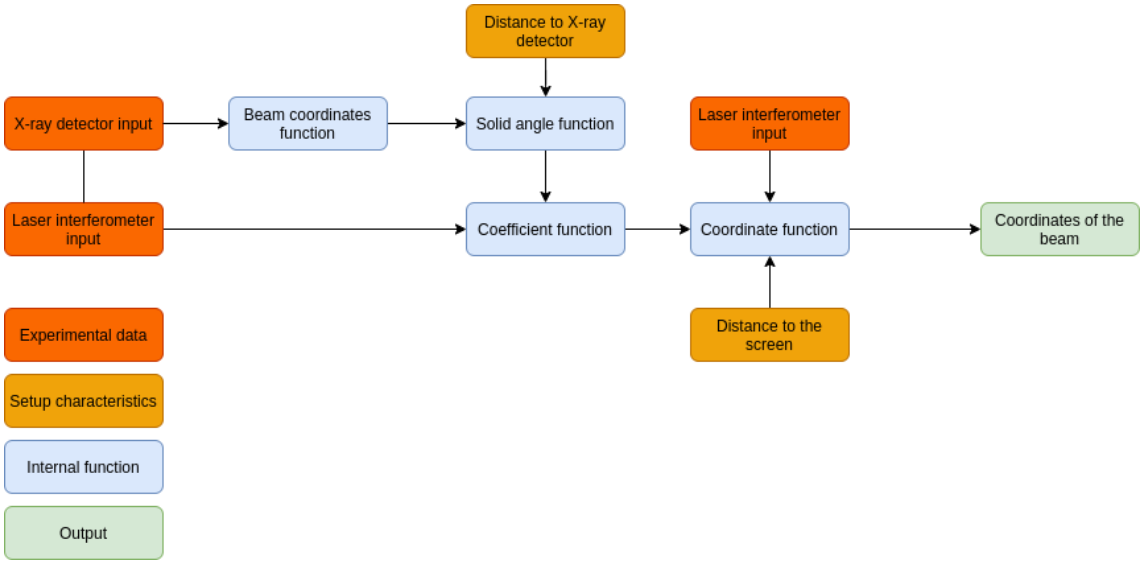


Figure 5: Program workflow diagram

Firstly the data from the interferometer and the X-ray detector is supplied to calculate the coefficient. Here is a sample output of the program along with the input data.

```
Point 1 (m): [0.10249992 0.05083273]
Point 2 (m): [0.1024572  0.05082699]
angle (rad): [-3.88347571e-06 -5.21507817e-07]
laser displacement (m): 1.724622e-08
distance to the detector (m): 11
coefficients (rad/m): [-225.1783702   -30.23896351]
```

Figure 6: Calculating the coefficient

Now that the coefficient is calculated, the program can be given the displacement measurement from the interferometer to calculate the coordinates of the beam.

```
laser displacement (m): 4.156207e-08
distance to screen (m): 0.5
coordinates of the beam (m): [-4.67943959e-06 -6.28396959e-07]
```

Figure 7: Calculating the coordinates

4 Conclusion

In this project the problem of controlling the angle of the falling X-ray beam onto the sample quickly and precisely was considered. A solution which consisted in introducing a mirror into the beam's way was proposed. Experimental data was analysed and the proposed solution was implemented as a script which processes the experimental data and provides the solution.

Acknowledgements

I want to thank the DESY Summer Student Programme organization for giving me the opportunity to work at one of the world's leading research centers. I would like to thank my supervisor Dr. Mikhail Lyubomirskiy for his enthusiastic and inspiring guidance in the project, for helping me master difficult topics and answering with great patience all the questions that arose throughout the research process. I would also like to thank Dr. Christian Schroer for the opportunity to make a presentation of this work in his seminar.

5 Appendix: program code

In [41]:

```
# libraries needed to extract data and plot it
import h5py
import hdf5plugin
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

mask = np.array(Image.open('mask.tif')) # mask used to remove shadows
file = h5py.File('eiger4m_01/scan_00032_master.h5','r')
x_pixel_size = file['entry']['instrument']['detector']['x_pixel_size']
y_pixel_size = file['entry']['instrument']['detector']['y_pixel_size']
```

In [42]:

```
# function that returns the coordinate of the beam in pixel units
def beam_coordinates(data: np.array)->float:
    c = np.zeros(2) # center of mass
    N = 0
    for depth in range(data.shape[0]):
        for i in range(data.shape[1]):
            for j in range(data.shape[2]):
                  if data[depth, i,j] * mask[i,j] != 0:
                       c[0] += i * data[depth,i,j]
                       c[1] += j * data[depth,i,j]
                       N += data[depth, i,j]
    return c/N

#sample = np.array(dset1)
#print(beam_coordinates(sample))

# function that calculates the solid angle based on the distance to the X-ray detector and two points there

def solid_angle(p1, p2, L):
    return (p2 - p1)/L
```

In [43]:

```
# function that returns the coefficients such that solid angle = c*h
def coeffs(angle, height):
    return angle / height

# function that calculates the coordinates on a screen L meters away, where c is the coefficient
def get_coordinates(c, h, L):
    return L*h*c
```

In [64]:
```
laser = np.array(h5py.File('data/test_000010.h5')['entry']['instrument']['detector']['data']['Axis2']['positio
h = (laser[1000] - laser[0])*1e-14
L = 11
```

In [65]:
```
p1 = beam_coordinates(np.array(file['entry']['data']['data_000001'])[0:1,:,:])
p2 = beam_coordinates(np.array(file['entry']['data']['data_000023']))
p1 *= x_pixel_size
p2 *= y_pixel_size
angle = solid_angle(p1, p2, 11)
c = coeffs(angle, h)
```

In [70]:
```
print('Point 1 (m):', p1)
print('Point 2 (m):', p2)
print('angle (rad):', angle)
print('laser displacement (m):', h)
print('distance to the detector (m):', L)
print('coefficients (rad/m):', c)
```

```
Point 1 (m): [0.10249992 0.05083273]
Point 2 (m): [0.1024572  0.05082699]
angle (rad): [-3.88347571e-06 -5.21507817e-07]
laser displacement (m): 1.724622e-08
distance to the detector (m): 11
coefficients (rad/m): [-225.1783702   -30.23896351]
```

In [72]:
```
dh = (laser[3000] - laser[0])*1e-14
d = 0.5
print('laser displacement (m):', dh)
print('distance to screen (m):', d)
print('coordinates of the beam (m):', get_coordinates(c, dh, d))
```

```
laser displacement (m): 4.156207e-08
distance to screen (m): 0.5
coordinates of the beam (m): [-4.67943959e-06 -6.28396959e-07]
```

We are going to plot some measurements by the interferometer when different signals are supplied to it.

In [1]:
```
# libraries needed to extract data and plot it
import h5py
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
# the following code splits the measurements by amplitude and plots it


# period of oscillation - frequency is 10 Hz
T = 1/10



def amplitude(a, start, end):
    return np.max(a[start:end]) - np.min(a[start:end])

def batch_size(sampling_frequency, frequency):
        return int(sampling_frequency / frequency)+1

def next_batch(a, start, bs, eps=0.5):
    A = amplitude(a, start, start+bs)
    start = start + bs
    while start < len(a) and abs(amplitude(a, start, start+bs)/A - 1) < eps:
        A = amplitude(a, start, start+bs)
        start += bs
    return start

def chunk(a, bs):
    chunks = []
    start = 0
    while start < len(a):
        end = next_batch(a, start, bs)
        chunks.append(a[start:end])
        start = end
    return chunks

def normalise(chunk):
    mean = np.mean(chunk)
    for i in range(len(chunk)):
        chunk[i] -= mean
```

```
# calculates the frequency of a normalised chunk of data by counting the zeros
def frequency(c, sampling_frequency):
    oscillations = 0
    for i in range(len(c) - 1):
        if c[i] * c[i+1] < 0:
            oscillations += 1
    return oscillations * sampling_frequency / len(c)

def is_max(a, t, r):
    if a[t] == np.max(a[max(t-r,0):t+r]):
        return True
    return False
def is_min(a, t, r):
    if a[t] == np.min(a[max(t-r, 0):t+r]):
        return True
    return False

def next_extremum(a, start, r=2):
    while (start < len(a)) and(not is_max(a, start, r)) and (not is_min(a, start, r)):
        start += 1
    return start
def average_amplitude(a):
    start = 0
    end=len(a)
    l = r = next_extremum(a, start) # current min and max indices
    r = next_extremum(a,r+1)
    A = 0
    cnt = 0
    while r < end - 1:
        A += abs(a[r] - a[l])
        cnt += 1
        l = r
        r = next_extremum(a, r+1)
    A /= cnt
    # calculate deviation
    s = 0
    l = r = next_extremum(a, start)
    r = next_extremum(a,r)
    while r < end:
        A_cur = abs(a[r] - a[l])
        s += (A_cur - A)**2
        l = r
        r = next_extremum(a, r+1)
    return A, np.sqrt(s/cnt)
```

```
# plots np.array of data
def plot(a):
    t = np.linspace(0, len(a)/sampling_frequency, len(a))
    plt.figure(figsize=(15,4))
    plt.title('Interferometer data')
    plt.xlabel('time (s)')
    plt.ylabel('displacement (nm)')
    plt.ylim(-100,100)
    ax = plt.gca()
    ax.autoscale_view()
    plt.plot(t, a ,'-', color='red',label="absolute", linewidth=1)
    plt.legend()
    plt.show()
```

In [3]:

```
file = h5py.File('test_000010.h5')
sampling_frequency = file['entry']['instrument']['detector']['sampling_frequency'][0]
print('sampling frequency:', sampling_frequency)
data = file['entry']['instrument']['detector']
a = np.array(data['data']['Axis2']['position_absolute'])
normalise(a)
f = frequency(a, sampling_frequency)
#r = data['data']['Axis2']['position_relative']
bs = batch_size(sampling_frequency, f)
chunks = chunk(a, 1000)
good = chunks[1::2]
c0 = good[0]
A0, s0 = average_amplitude(c0[len(c0)//4 : -len(c0)//4])
A = []
s = []
for c in good:
    A_c, s_c = average_amplitude(c[len(c)//4 : -len(c)//4])
    A.append(A_c/A0)
    s.append(s_c/s0)
```

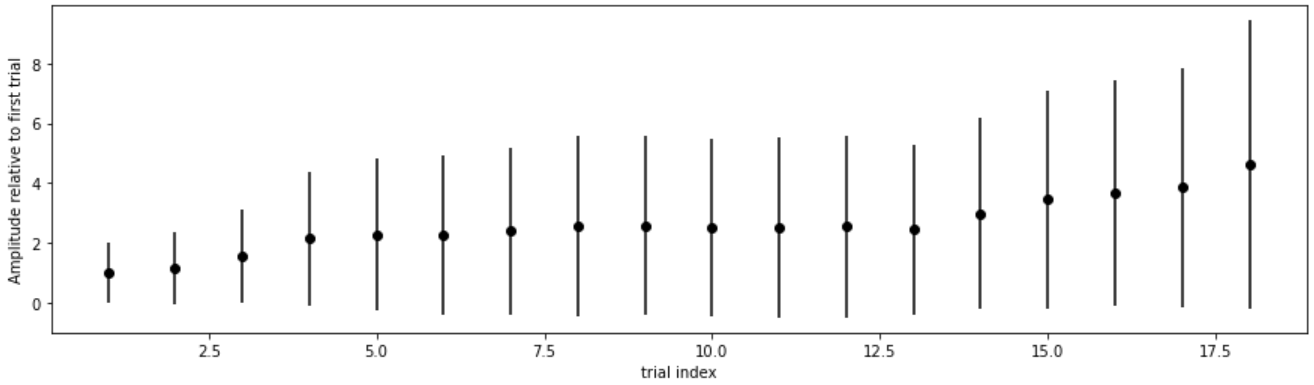
```
sampling frequency: 781.25
```

In [10]:

```
plt.figure(figsize=(15,4))
plt.xlabel('trial index')
plt.ylabel('Amplitude relative to first trial')
plt.errorbar(list(range(1, len(A)+1)), A, s, color='black', linestyle='None', marker='o')
```

Out[10]:

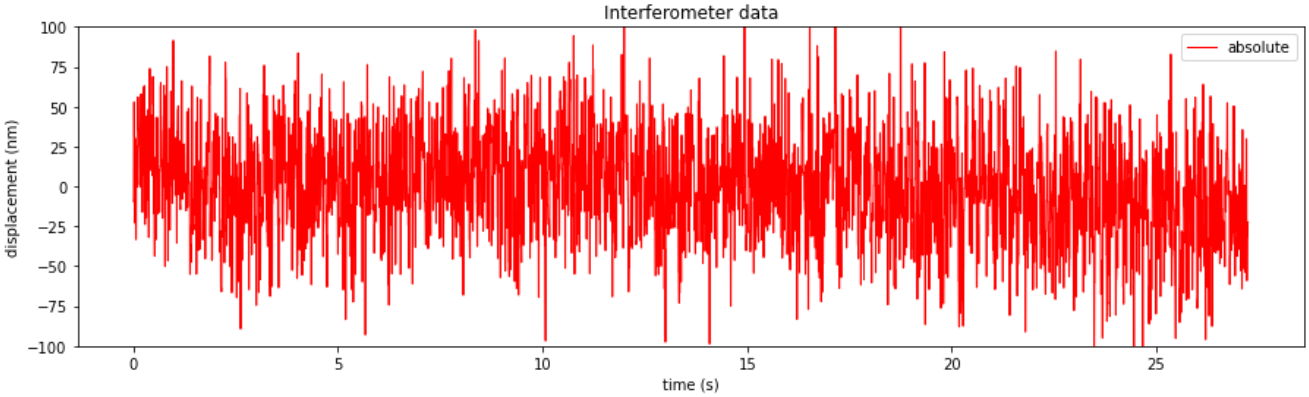
```
<ErrorbarContainer object of 3 artists>
```



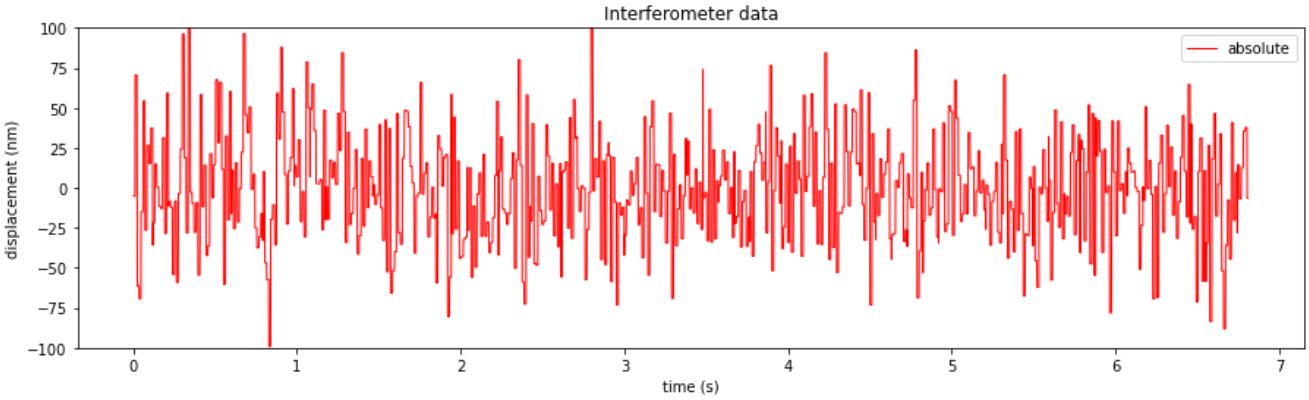
In [16]:

```
# the second measurement
file = h5py.File('test_000010.h5')
sampling_frequency = file['entry']['instrument']['detector']['sampling_frequency'][0]
print('sampling frequency:', sampling_frequency)
data = file['entry']['instrument']['detector']
a = np.array(data['data']['Axis2']['position_absolute'])
normalise(a)
f = frequency(a, sampling_frequency)
bs = batch_size(sampling_frequency, f)
chunks = chunk(np.array(a), 10*bs)
for c in chunks:
    normalise(c)
    print("frequency:", frequency(c, sampling_frequency))
    plot(c*1e-5)
```

```
sampling frequency: 781.25
frequency: 43.32119360902256
```




```
frequency: 48.7546992481203
```




```
frequency: 20.46358629130967
```

References

[1] Jens Als-Nielsen, Des McMorrow, *Elements of Modern X-ray Physics, Second Edition*, (John Wiley and Sons, 2011)

[2] A. M. Cormack *Representation of a Function by Its Line Integrals, with Some Radiological Applications*, Journal of Applied Physics **34**, 2772 (1963)