Project Report on

Subaru Telescope's "LGSAO188"

FOCUS TRACKING OF THE LASER GUIDE STAR

USING THE GUIDE STAR ACQUISITION SYSTEM

PLACE

Subaru Telescope
National Astronomical
Observatory of Japan (NAOJ)
650 North A'ohoku Place
Hilo, HAWAI'I 96720,
U.S.A.

INTERNSHIP DURATION

5th April'10 - 2nd Sept'10

Submitted by

Garima SINGH
Masters Student
Astronomical & Space-based System Engineering (2009-2010)
Université Paris-Sud 11
Orsay, France
Email: garima.singh@obspm.fr, garima_singh1987@yahoo.co.in

Under the Supervision of

Yutaka HAYANO
AO Development Manager
Subaru Telescope, HAWAII
Email: hayano@naoj.org

ACKNOWLEDGEMENT

I take immense pleasure in thanking, the Dean of the **Université Paris-Sud 11**, Philippe MASSON, the Director of the **Subaru Telescope**, Hideki TAKAMI, the head of the Master **OSAE** (Outils et Systèmes de l'Astronomie et de l'Espace or Astronomical & Space-based System Engineering), Benoît MOSSER, Professpm.fr or at the Observatoire de Paris and Alain ABERGEL, Professor at the Université Paris-Sud 11, and the **LGSAO188 project leader**, Yutaka HAYANO for having permitted me to carry out the 5 months project work on Laser Guide Star Adaptive Optics system at the Subaru Telescope, Hawaii.

I would like to express my deep and sincere gratitude to my supervisor, Yutaka HAYANO, for his immense support and assistance. He taught me very patiently the conceptual, theoretical and the technical aspects of the Laser Guide Star. His wide knowledge in Adaptive Optics and his logical way of finding the solutions for achieving the desired result have been of great value for me. His understanding, encouraging and personal guidance have provided a good basis for the completion of the project.

I wish to express my deep sense of gratitude to my Internal Guide, Guy PERRIN, Researcher at the 'Laboratoire d'études spatiales et d'instrumentation en astrophysique' (LESIA), **Observatoire de Paris**, for his able guidance and useful suggestions, which helped me in completing the project work, in time.

Needless to mention that Yosuke MINOWA, **Subaru's support astronomer**, who had been a source of knowledge, support and guidance in the conduct of this project work.

Words are inadequate in offering my thanks to all the staff members of the Subaru Telescope for their encouragement and cooperation in carrying out the project work.

Finally, yet importantly, I would like to express my heartfelt thanks to my beloved parents for their blessings, my friends/classmates for their help and wishes for the successful completion of this project.

ACRONYM

Table 1: Scientific terms used in the report

ACT	Auto Collimating Telescope
ADC	Atmospheric Dispersion Corrector
AO	Adaptive Optics
APD	Avalanche PhotoDiode
AU	Acquisition Unit
BM	Beam Splitter
DM	Deformable Mirror
FOV	Field Of View
GSAU	Guide Star Acquisition Unit (1 & 2)
HOWFS	High Order WaveFront Sensor
IMR	Image Rotator
LGS	Laser Guide Star
LGSAO188	Laser Guide Star Adaptive Optics with 188 elements Curvature Wavefront Sensor
LOWFS	Low Order WaveFront Sensor
NGS	Natural Guide Star
OAP	Off-Axis Parabolic
SH/SHWFS	Shack-Hartmann/ Shack-Hartmann Wavefront Sensor
TP	Turbulent Plate
TT	Tip-Tilt
TTM	Tip-Tilt Mount
VM	Vibrating Membrane Mirror
WFS	Wavefront Sensor

ABSTRACT

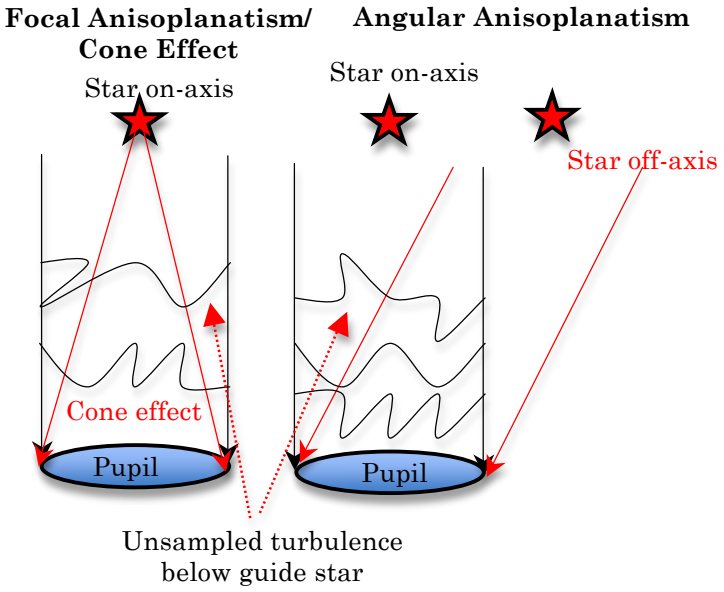
INTRODUCTION[1]

The basic requirement of an Adaptive Optics system is to find/create a bright guide star within the isoplanatic tip tilt field of the astronomical object under observation, for the real time analysis and correction of the wavefront. Finding a bright natural star near the observing object is quite random and difficult in visible but with the LGS, its possible to excite Sodium atoms at the height of 90 km and the light scattered back can be used as a bright guide source.

STATEMENT OF PROBLEM[2]

Generally the guide star is angularly displaced from the science object, so because of this displacement the wavefront sampled by it is not the same as that received from the stellar object, which arises the problem of Angular Anisoplanatism. Also the use of LGSAO is restricted by the focal anisoplanatism and the difficulty in tilt determination. Moreover a beacon projected from the ground cannot provide information on the absolute position of the object viewed. For the long exposure frames, the position of the image should be stabilized within the fraction of the spatial resolution of the science camera.

Figure 1



LGS sample a cone-shaped volume of the turbulence in the optical path of the telescope as shown in Fig 1.i), turbulence above the beacon is not measured, and that occurring below inside the cone is partially sampled that give rise to the error in wavefront known as Cone Effect/Focal Anisoplanatism.

LGS mode is suitable for measuring the high order wavefront error within an optical aperture, but it cannot be used to determine the overall tilt caused by the turbulence that actually helps in compensating the actual direction of the stellar object. As the position of the beam projected from the ground is randomly

[1] Chapter 2 of Adaptive Optics for Astronomical Telescopes by JOHN W. HARDY,
Chapter 7 of Laser Guide Star Adaptive Optics for Astronomy by N. AGEORGES & C. DAINTY

perturbed by the atmospheric turbulence but on its way back to the telescope it encounters the same amount of deflection from the turbulence, as the laser beam always appear to be on the axis no matter where the telescope is pointing. So for stabilizing the image, a separate reference source (NGS) within the isoplanatic tip tilt field of the stellar object should be considered. The unavailability of the NGS around the stellar object also restricts the use of LGSAO.

TECHNICAL DIFFICULTY

During the functioning of the LGS, the width of Na layer and the elevation (h) changes according to the position of the stellar object, which introduces the tip tilt error because the focus of the LGS changes. So while tracking the focus with a closed AO loop, this change in focus is sensed as a defocus term by the WFS. Hence, the closed loop AO system interprets this as a wavefront error and thus commands the DM to compensate for the defocus, leaving the spot unfocused on the science camera. Neither focus can be tracked nor the image dithering is possible which plays major role in spectroscopy.

REQUIREMENT[2]

The basic requirement of the LGSAO188 of the Subaru Telescope is as following-

Image dithering: It is the process of sequentially pointing the telescope to slightly different positions for imaging the background emission. There must be a system that can move the image on the science camera for the dithering to be done.

Focus tracking: A system is needed which can actually track the focus of the LGS and can feed new focus to the WFS so that it cannot interpret it as a tip tilt error.

Stabilization of the image: A separate reference source (NGS) within the isoplanatic field of the stellar object should be used for measuring the lowest order wavefront distortion component, i.e. the tip-tilt. Hence, the sky coverage obtained with laser beacons, stabilized by NGS is much greater than that of AO using NGS alone.

AIM OF THE SUBARU'S LGSAO188

The foremost goal of the Subaru Telescope and for this internship was the 'Focus Tracking'. One of the module of the LGSAO188, "The Guide Star Acquisition Unit 1 & 2' was calibrated for the tracking of the focus. There are two Acquisition units: AU1 for the HOWFS and AU2 for the LOWFS. This report explains the working of the AU1 and AU2 and divides the goal of tracking the focus in two major sequential tasks. We developed two models, first for the determination of the change in the focus because of the change in the elevation of the LGS and the second one is for defining the trajectory motion for the tracking of the focus. The installation of the LGSAO188 is currently ongoing, and we couldn't use the laser, so we used the calibration light source in LGS mode and tried to track the focus.

RESUME

INTRODUCTION

La nécessité première d'un système d'optique adaptative est de trouver / créer une étoile guide suffisamment brillante (naturelle ou artificielle) dans le champ isoplanétique de l'objet astronomique observé, pour l'analyse en temps réel et la correction du front d'onde. Trouver une étoile suffisamment brillante à proximité de l'objet observé n'est pas toujours possible. Avec l'étoile guide laser, on excite les atomes de Sodium à 90 km d'altitude et la lumière diffusée peut alors être utilisée comme une source lumineuse guide. Mais l'utilisation de l'étoile guide laser est limitée par de nombreux problèmes techniques tels que :

L'anisoplanétisme : l'étoile guide est déplacées angulairement de la source observée ; de ce fait, le front d'onde échantillonné par l'étoile guide n'est pas identique à celui reçu de la source.

L'effet de cône: l'étoile guide laser échantillonne un cône de turbulence dans le chemin optique du télescope mais la turbulence au-dessus du cône est laissée non échantillonnée.

Le problème de la détermination du Tilt: le tip-tilt causé par la turbulence ne peut pas être mesuré à l'aide de l'étoile laser. La lumière du laser emprunte le même chemin optique lors de sa propagation, à l'allé et au retour.

Indisponibilité de l'étoile guide naturelle nécessaire à l'évaluation du tip-tilt autour de l'objet stellaire.

En outre, pour les images à long temps d'exposition, la position de l'image doit être stabilisée à la fraction de la résolution spatiale de la caméra scientifique.

POINTS DURS

Lors de l'utilisation de l'étoile laser, la densité et la répartition du Sodium varient suivant la position de l'objet stellaire. Cela introduite une erreur de tip-tilt. Pendant le suivi du focus lorsque l'OA est en boucle fermée, le changement (du point le plus dense dans le couche de sodium) est ressenti comme un terme de défocalisation par le senseur du front d'onde. Par conséquent, l'OA en boucle fermée interprète cela comme une erreur de front d'onde et commande le DM pour la compenser, rendant ainsi l'objet défocalisé. Le focus sur l'objet où un tramage des images, jouant un rôle majeur dans la spectroscopie,deviennent très difficile.

EXIGENCES

Le système LGS doit être en mesure d'effectuer les fonctions suivantes :

Tramage des images: C'est un processus séquentiel dépointant le télescope vers

des positions légèrement différentes pour l'imagerie des émissions de fond du ciel. Ainsi, un système est nécessaire pour déplacer l'image sur la caméra scientifique en vue du tramage. Il est utile dans l'infrarouge proche, où le fond de ciel est brillant.

Determination du focus: un système est nécessaire pour pointer à tout moment la position de l'étoile laser afin de ne pas confondre le focus avec des erreurs de tip-tilt.

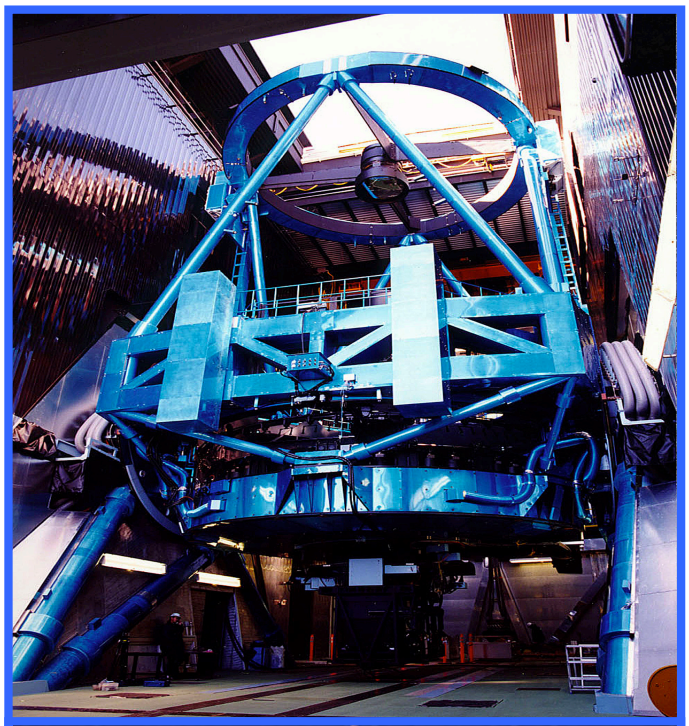
Stabilisation de l'image: Le laser projeté à partir du sol ne peut fournir des informations sur la position absolue de l'objet observé. Ainsi, pour stabiliser l'image au point focal de la caméra scientifique, une source de référence séparée qui n'est pas perturbée par l'atmosphère est nécessaire. Une étoile guide naturelle, dans le champ isoplanétique pour le tilp-tilt et le focus autour de l'objet stellaire, peut être utilisée pour mesurer ces bas ordres. La couverture du ciel obtenue avec des étoiles laser davantage distantes de l'étoile naturelle, est beaucoup plus grande que celle de l'OA avec l'étoile naturelle utilisée seule.

L'OBJECTIF DE LGSAO188 DU SUBARU

Pour surmonter le difficulté du 'determination du focus', le Telescope Subaru est en train de calibrer l'un des modules de LGSAO188, "l'unité d'Acquisition de l'étoile laser". Il y a deux unités d'acquisition: AU1 pour le HOWFS et AU2 pour le LOWFS. Ce rapport explique le fonctionnement des AU1 et AU2 et organize la determination du focus en deux grandes étapes. Nous avons développé deux modèles, l'un pour la détermination de la variation de la mise au point due au changement de l'élévation de l'étoile guide laser et le second modèle definit le mouvement de la trajectoire du 'Focus Tracking'. A ce jour, l'installation du module LGSAO188 n'est pas terminée. Il n'a donc pas été possible d'utiliser la source laser de l'OA pour tester les modèles. A la place, nous avons utilizé la source de calibration de l'OA et essayé à suivre le focus de la source de calibration.

TABLE OF CONTENTS

SUBARU TELESCOPE[2]



Primary Mirror

* Effective diameter: 8.2 m (27 ft)
* Thickness: 20 cm (7.9 inches)
* Weight: 22.8 metric tons
* Material: ULE (ultra-low thermal expansion glass)
* Focal length: 15 m (49'2")

Telescope Structure

* Mounting: Altitude-Azimuth
* Basic Optics: Ritchey-Chrétien
* Height: 22.2 m (72 ft.
* Maximum width: 27.2 m
* Weight: 555 metric tons

Foci → Four

* Primary focus -> F2.0 (with corrector lens)
* Cassegrain focus -> F12.2
* Nasmyth focus (optical) -> F12.6
* Nasmyth focus (infrared) -> F13.9
* Maximum slewing speed -> 0.5 deg/sec
* Tracking accuracy without Guiding -> 0.1"
* Best angular resolution Achieved -> 0.2" (without AO @ 2.15 μm)

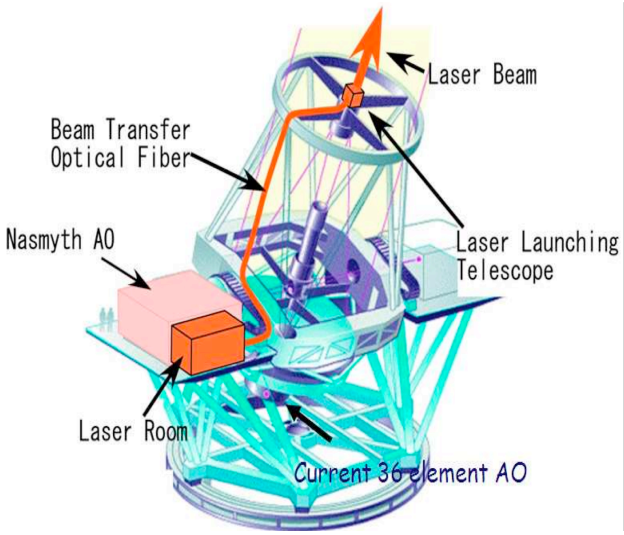
Telescope Enclosure

* Altitude -> 4139 m (13,580 ft.)
* Latitude: 19d 49m 32s N
* Longitude: 155d 28m 34s W
* Height: 43 m (141 ft.)
* Diameter at base: 40 m (131 ft.)
* Weight: 2000 metric tons

[2] Specification of the SUBARU TELESCOPE as described on its official website http://www.naoj.org/Introduction/telescope.html

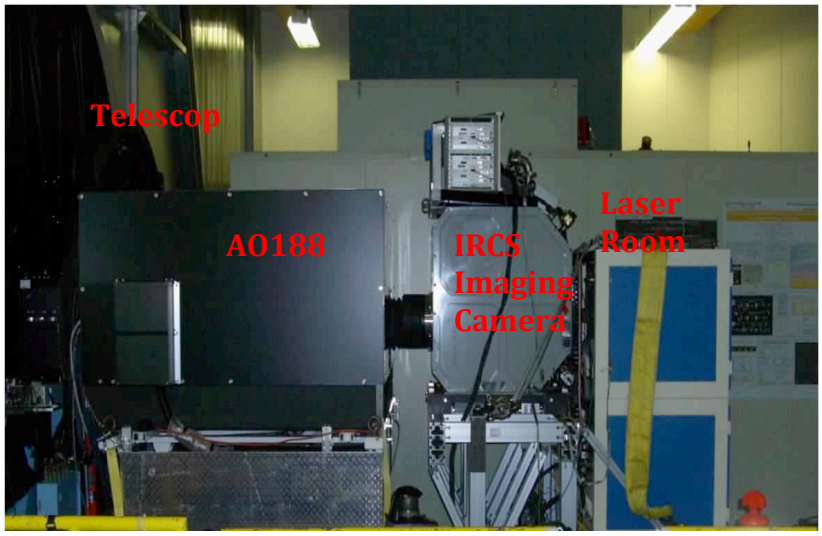
LGSAO188 of the SUBARU TELESCOPE[3]

Figure 2



AO188 is a 188-elements curvature sensor adaptive optics system that operates in both LGS and NGS mode with the FOV for the science instrument as 2 arcminute diameter. It is installed at Nasmyth platform (at f/13.9) of the Subaru telescope. It measures tip-tilt and high order terms of wavefront error using either a single NGS (2.7 arcminute FOV) or LGS/NGS (2 arcminute FOV). It has a monolithic bimorph DM having 188 element electrodes. It also has a 4-10 W solid-state sum-frequency laser to generate LGS.

Figure 3



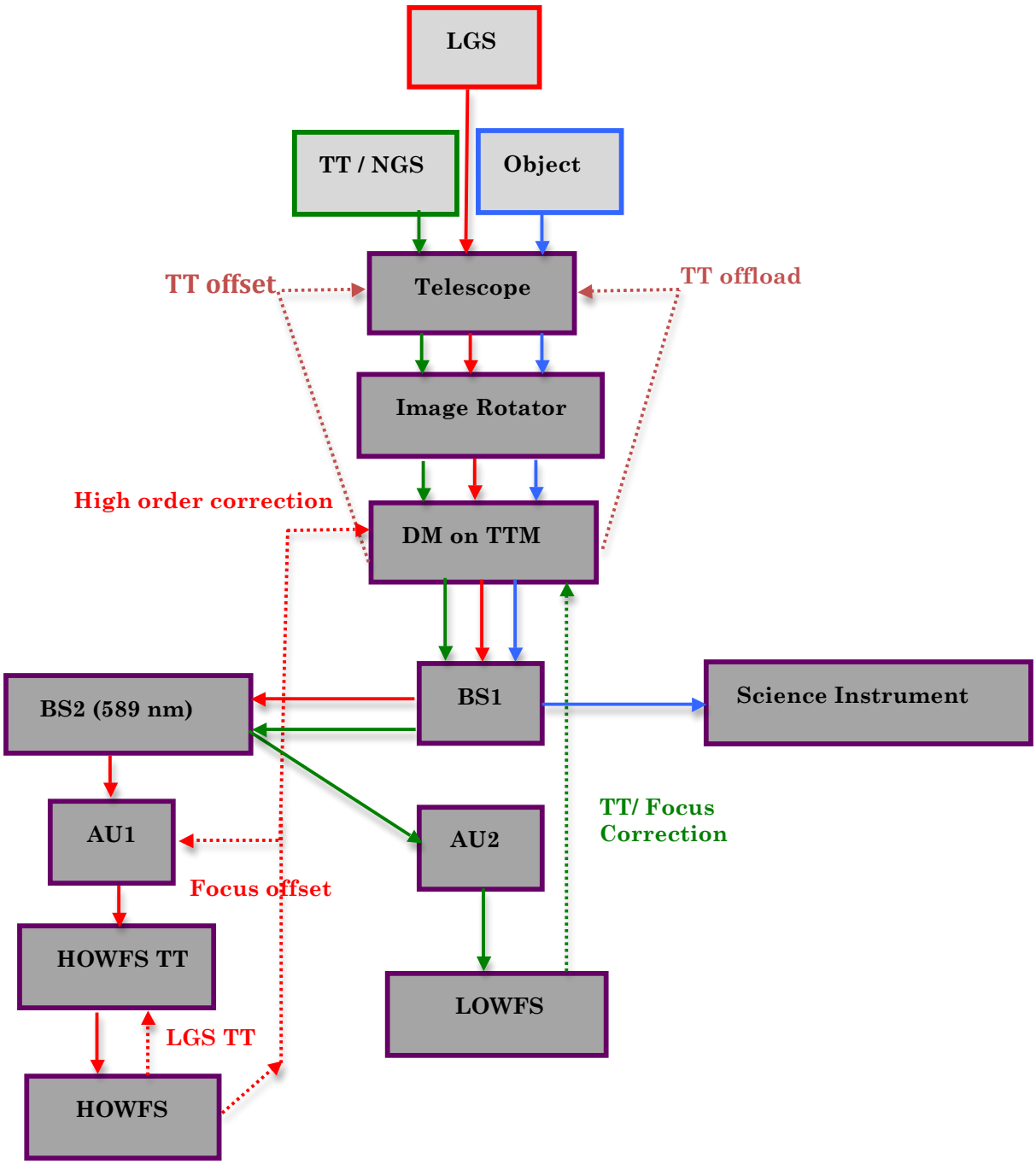
The laser-launching telescope with 50 cm aperture is installed behind the secondary mirror. It's working in NGS mode since October 2008 and has achieved K-band Strehl ratio between 60-70% using R=9.0 magnitude NGS with isoplanatic angle of 30 arcseconds. The science camera used by AO188 is the 'IRCS (Infrared Camera and Spectrograph)' with a pixel scale of 20 mas/pix, which spans the range of 0.9-5.3 μm.

[3] Performance of Subaru Adaptive optics system AO188, SPIE 2010, Adaptive Optics system II (7736-134)

Table 1: LGSAO188 result in NGS mode for R < 10 mag under seeing condition of 0.4-0.7 arcsecond

Wavelength Band	M (4.86 μm)	L (3.77 μm)	K (2.20 μm)	H (1.63 μm)	J (1.25 μm)	Z (1.03 μm)
Strehl Ratio	1.0	0.85	0.65	0.40	0.20	0.08

Figure 4: The schematic diagram of the working of LGSAO188 (LGS mode)



DESCRIPTION of the OPTICAL BENCH for LASER GUIDE STAR ADAPTIVE OPTICS (AO188) for the INFRARED NASMYTH FOCUS [4]

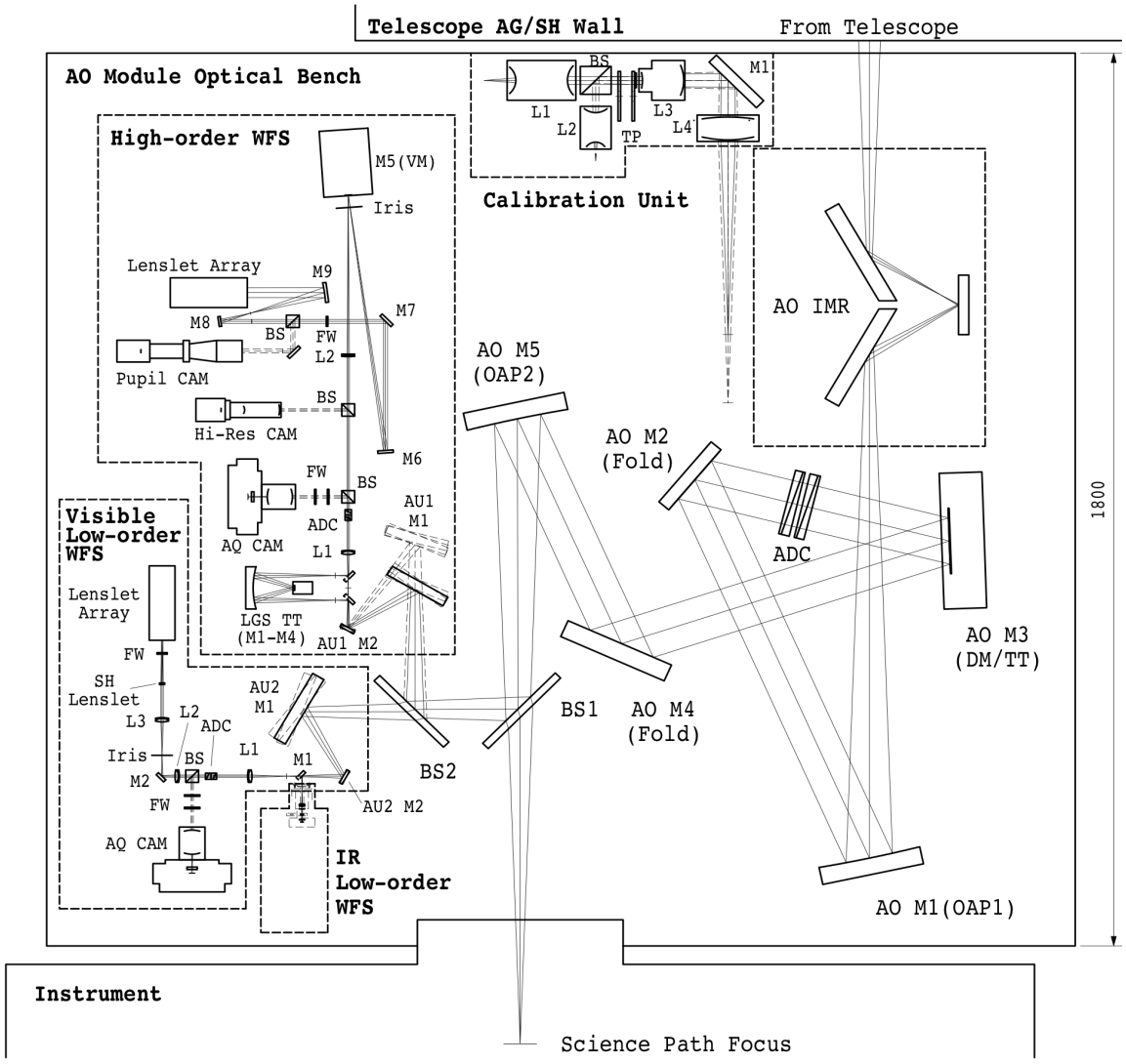
The optical bench for LGSAO188 consists of three major parts:

8.1) Science path,

8.2) Three wavefront sensors – a HOWFS, two LOWFS for visible and IR respectively, and

8.3) Calibration unit

Figure 5: Optical Layout of the AO Module (2000(W) × 1800(L) × 250(H) mm)



[4] 'Design of the Subaru laser guide star adaptive optics module' SPIE 2004 Vol. 5490

SCIENCE PATH

Table 2. Specifications for the Science path module

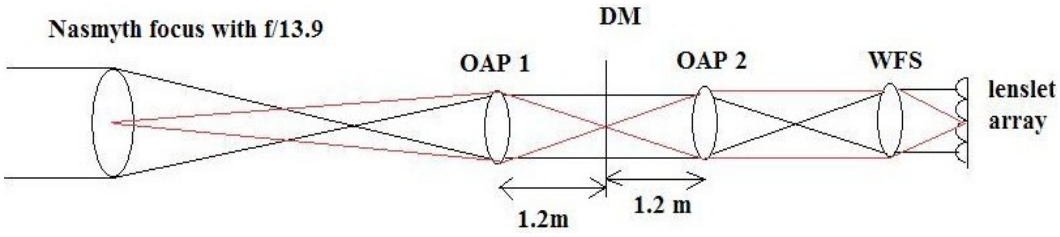
Spectral coverage	0.45-5 μm for transmission
Input & Output focal ratio	f/13.9
FOV (for science camera)	2 arcminute (in diameter)
FOV (for WFSs)	2.7 arcminute (in diameter)
DM Type	Bimorph mirror with 188 electrodes
Beam Diameter on DM	90 mm
Actuator Type of TTM for DM	Voice coils
Tip-Tilt stroke of TTM for DM	± 5 arcsecond (@ sky)
Number of BM (BM1)	4

It includes –

1) A derotator (AO IMR), a three mirror system for reorienting the field on sky from telescope towards the Nasmyth focus.

2) Two conjugated off-axis collimators OAP1 and OAP2, which are used to collimate and refocus the telescope pupil from the tertiary mirror to the DM and from DM to the BS1 respectively. Both collimators are perfect parabola having d=200mm and f=1201mm which collimates the beam with a 90 mm diameter (fig 6).

Figure: 6

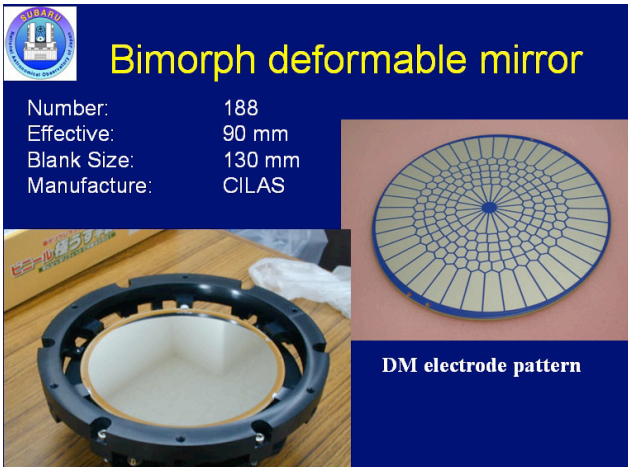


3) A bimorph type DM[5] (fig 7). Because the internal tip-tilt of DM is small, it is placed on the fast tip-tilt mount (TTM) for compensating the large tilt corresponding to the large change in the focus of the telescope. The range of the tilt angle is 5 arcsecond on the sky.

4) An ADC for compensating the chromatic dispersion created by the atmosphere. When the zenith angle is small, no correction is needed for the science instrument but for the WFS, other ADCs are inserted for correction in the visible wavelength.

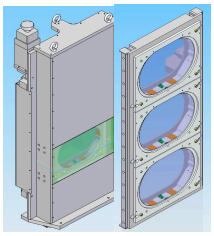
[5] TTM is build by Observatoire de Paris, France

Figure: 7



5) Two fold mirrors (AO M2 & M4), and

6) An exchangeable BS1 of CaF_2 material (with size of 200 mm × 145 mm and thickness of 15 mm) is placed to split the light between science instrument and WFSs. It is placed at a distance of 670 mm before the focus of the OAP2 and is tilted by 45 degrees to reflect the visible light. It has a slight wedge angle to avoid an astigmatism introduced by BS1 itself.



Note: Three different BS1 can be mounted-

1) Standard dichroic: 900 nm
2) Dichroic for optical instruments: 625 nm
3) IR WFS

The FOV for WFS is larger than that of the science instrument in order to extend the sky coverage of the NGS for the LOWFS (tip-tilt and defocus).

Note: At Infrared Nasmyth focus, a small coma aberration is introduced. As for the Nasmyth focus a tertiary mirror is used for the coudé focus, so it's slightly different from the Ritchey-Chrétien system because the secondary mirror is common to the Infrared Cassengrain focus and its figure is optimized for Cassengrain only and not for Nasmyth. But it's still acceptable within the FOV of the science field.

WAVEFRONT SENSOR

High Order Wavefront Sensor

It includes-

1) A fast tip-tilt mirror (LGS TT M1-M4) to stabilize the LGS image at the inside focus.

Table 3. Specifications for HOWFS

Sensor Type	Visible 188-element curvature sensor
Target Guide Star	LGS / NGS
Spectral Coverage	0.45-1.0 μm
Detectors	Photon counting APD modules
FOV from lenslet array to the optical fibers	Selectable among 4, 2 and 1 arcsecond (in diameter)
Patrolled FOV	2 arcminute (in diameter)
Main functioning	To correct high order wavefront errors only when using LGS and NGS. To correct high order as well as tip-tilt and focus errors when using NGS only.

Why we need TT mount for LGS?

Because of the tip tilt determination problem as described in the 'INTRODUCTION', So an additional tip-tilt mirror is needed to minimize the lateral shift of the pupil at the lenslet array, the feedback of which is provided to the AU1. There is some offset value for tip-tilt mirror; if it exceeds this offset value then AU1's M1 and M2 distance should be changed to adjust the offset and bring the TT back to the nominal position. In NGS mode, this LGS TT optics can be removed.

2) Relay lenses (L1 & L2) to convert the focal ratio into f/65. These are used to collimate and refocus the beam at the vibrating mirror.

3) An ADC for the NGS beam

4) A vibrating membrane mirror with a phase monitor. The collimated light from L2 is focused at it. Also there is a field stop (an iris diaphragm) located in front of it, which lowers the FOV of the HOWFS to 1 arcsecond for reducing the sky background and aliasing effect in curvature sensing. VM basically moves the position of the telescope pupil on the 188-elements fixed lenslet array back and forth. Hence its vibrating amplitude can change the distance of the pupil plane to measure the curvature.

5) Collimating mirrors (M6-M8)

6) A lenslet array coupling to APD's through optical fibers. The light corrected by each lenslet array is injected to large-core multi mode optical fibers that feed to the APD's. Index-matching oil is used between the glass surface and fiber input for avoiding the reflection.

7) An acquisition / mapping CCD camera (AQ CAM) with a 20 arcsecond FOV

8) A high-resolution camera (HI-RES CAM) with a pixel scale of a few milli-arcsecond

9) A pupil camera

Note: HI-RES CAM and the pupil camera are used for alignment and diagnostic purposes.

Visible Low Order Wavefront Sensor

Table 4. Specifications for Visible LOWFS

Sensor Type	2×2 sub-aperture Shack-Hartmann sensor
Target Guide Star	NGS
Spectral Coverage	0.45-1.0 μm
Detectors	16 photon counting APD modules
FOV from lenslet array to the optical fibers	Selectable among 4,2 and 1 arcsecond (in diameter)
Patrolled FOV	2.7 arcminute (in diameter)
Main functioning	To correct tip-tilt and focus errors of the system in LGS mode

The Wavefront tilt sensing performance and the defocus measurement capability of the visible LOWFS is optimized by Shack-Hartmann sensor. It consists of –

1) A BS2 for splitting the light into the laser and the natural light
2) Relay lenses (L1 and L2)
3) An ADC
4) A collimating lens (L3)
5) A 2×2 SH lenslet array
6) A 4×4 detector lenslet array coupling to APD's through optical fibers which are placed at the focus of 2×2 lenslet array so the number of detector lenslet per one SH sub-aperture is 2×2, and
7) An acquisition / mapping CCD camera (AQ CAM) with a 20 arcsecond FOV.

Infrared Low Order Wavefront Sensor

In near future, it can be used for the measurement of TT and slow defocus terms of the wavefront in the infrared region when no visible NGS will be available around the science target for the visible LOWFS.

Table 5. Specifications for Infrared LOWFS

Sensor Type	2×2 sub-aperture Shack-Hartmann sensor
Target Guide Star	NGS
Spectral Coverage	0.9-2.5 μm
Detectors	128×128 pixel HgCdTe array
FOV from lenslet array to the optical fibers	4 arcsecond (in diameter)
Patrolled FOV	2.7 arcminute (in diameter)

Note: Both HOWFS and LOWFS include a guide star Acquisition unit[6] named as AU1 and AU2 respectively that plays an important role for the **focus and position adjustment**

CALIBRATION UNIT

It is used to provide the light sources for the simulation, alignment, calibration, testing and the diagnostics purposes. This unit is inserted in front of the IMR.

It simulates both LGS and NGS beams, simultaneously, with and without turbulence. It can produce two turbulent layers of about 0 and 6 km altitudes and can also reproduce the anisoplanatism and the cone effect for the LGS beam.

Table 6: Specifications for the Calibration Unit

Number of Light Source	LGS: 1, NGS: 1
Spectral Coverage	LGS: 0.589 μm, NGS: 0.589-2.2 μm
Simulated FOV	LGS: 0.2 arcminute, NGS: 2 arcminute (in diameter)
Number of turbulent plates	2
Simulated altitudes of turbulent layers	~ 0, 6 km

It includes –

1) Two collimating lenses (L1 & L2) to collimate the input beam

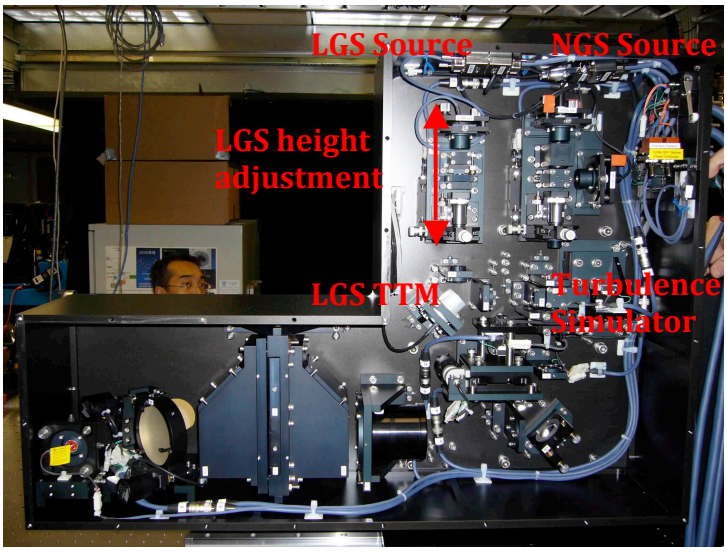
2) A BS to couple natural and laser light paths together

3) Two rotating turbulent plates

4) Re-imaging lenses (L3 & L4) to produce the f/13.9 beam on the telescope focal plane

[6] The description of the guide star acquisition unit is explained later in the same section

Figure 8

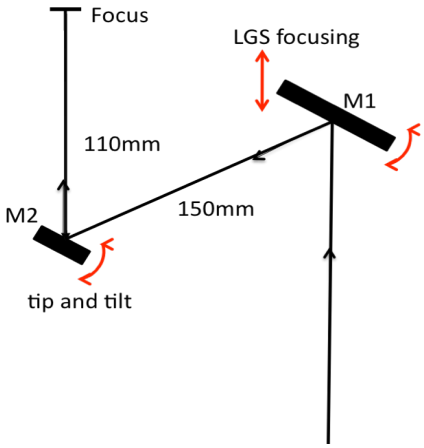


The turbulent plates are the modeled plastic plate with a deformed surface to provide a wavefront error with Kolmogorov statistics. It has a diameter of 30 mm, thickness of 2 mm and produce a beam diameter of ~ 9 mm for NGS36 but for the LGSAO188 it might need larger diameter of about 100 mm with the beam diameter of 16 mm.

APPROACH FOR THE FOCUS TRACKING THE GUIDE STAR ACQUISITION UNIT (AU1 & AU2)[7]

GSAU consists of two mirrors M1 and M2 and has 5 degrees of freedom i.e. there are four linear actuators per AU for moving the M1 and M2 in X and Y plane (M1X, M1Y, M2X and M2Y) and one linear actuator for moving the stage in order to change the distance between M1 and M2.

Figure 9: AU1 for HOWFS



Nominal position

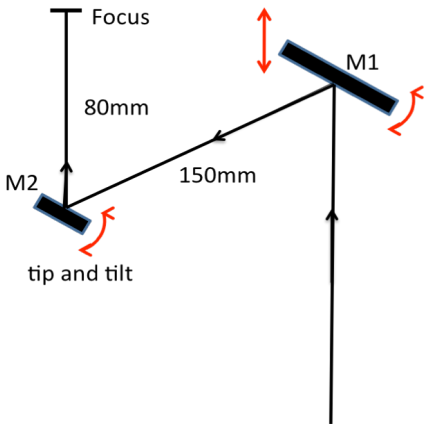
* 30 degrees tilt w.r.t the optical axis
* Distance between the two mirrors: 150 mm
* Distance from focal position and M2: 80 mm

Manipulators

* Linear actuators for tip-tilt: Newport LTA-HL
* Linear stage for LGS focusing: Newport XMS100
* Controller: Newport XPS

Figure 10: AU2 for LOWFS

[7] Taken from the presentation given by Prof Yutaka Hayano



Nominal position

* 30 degrees tilt respect to the optical axis.
* Distance between the two mirrors: 150 mm
* Distance from focal position and M2: 110 mm

Manipulators

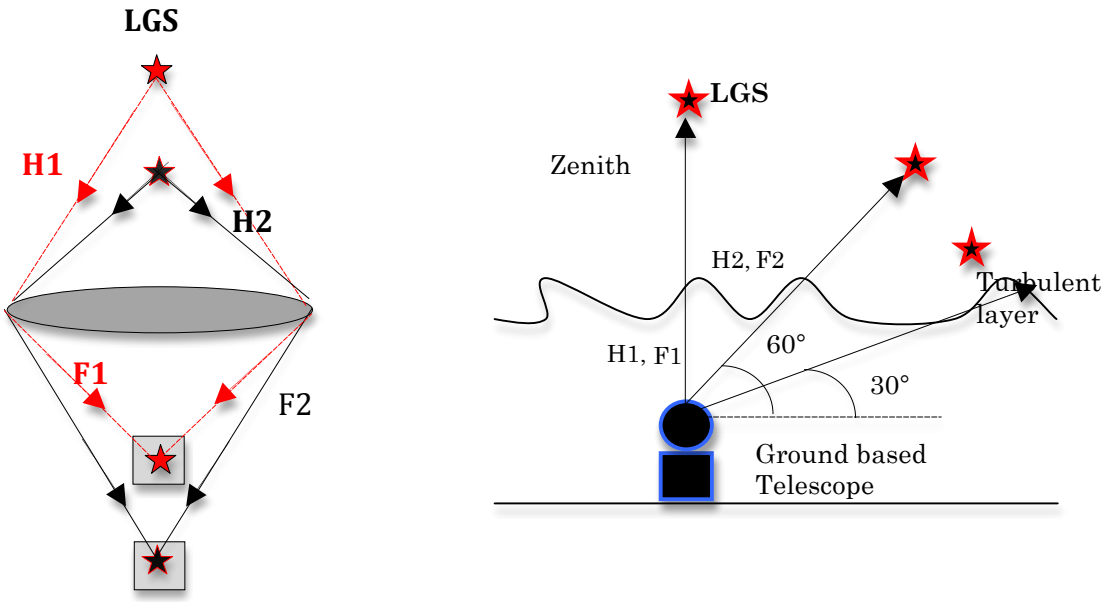
* Linear actuators for tip-tilt: Newport LTA-HL
* Linear stage for LGS focusing: Newport XMS 50 * Controller: Newport XPS

The out of focus chief ray from guide star, when arrives at the AU, the mirror M1 tilts to send the ray at the center of the mirror M2 which in turn tilts to align the chief ray within the input axis of the WFS. M1 also slides along the input axis of WFS to keep the same travel distance for rays when the mirror tilts for an off-axis field.

Why we need to tilt M1 and M2 mirrors of the AU?

There is a need of tracking the time variations in altitude of the Na layer and also the variation in the zenith angle during the telescope's movement. In the figure 11, it's clear that the focus of the LGS changes because of the finite distance between the telescope and the Na layer. It increases with decrease in the elevation and vice versa. As shown below, for the height H1>H2, the focus F2>F1

Figure 11



In figure 9 and 10, at the nominal position, if the light source is on axis and the focus is fixed then the ray will strike at the center of the M1 and M2 respectively. For the star under observation, which is located at infinity, the focus is at 110m (f_{tel}) at the WFS but for the LGS, its focus changes as the function of the elevation as shown in figure 11.

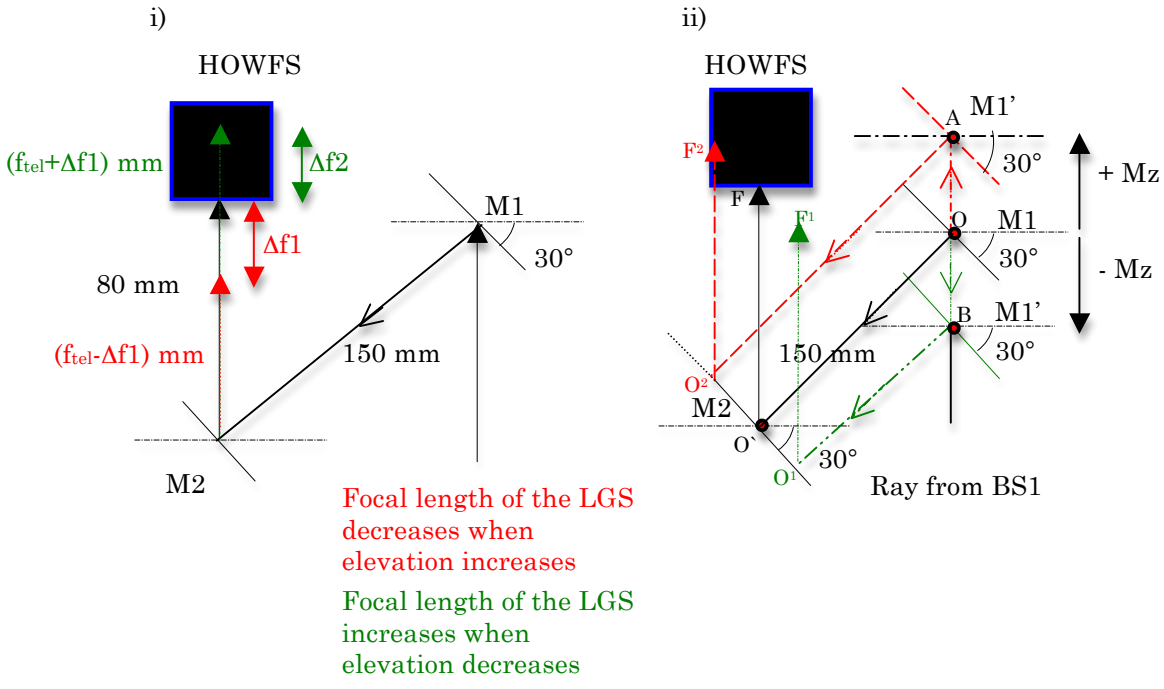
How AU works?

1. In figure 12.i), the M1 and M2 are at the nominal position i.e. at 30°. Considering the on axis light source, so when the elevation decreases, the focal length of the LGS increases (red line), and when the elevation of the LGS increases, its focal length decreases (green line). The only way to get back at the focus is to tilt the mirrors and to change the distance between M1 and M2.

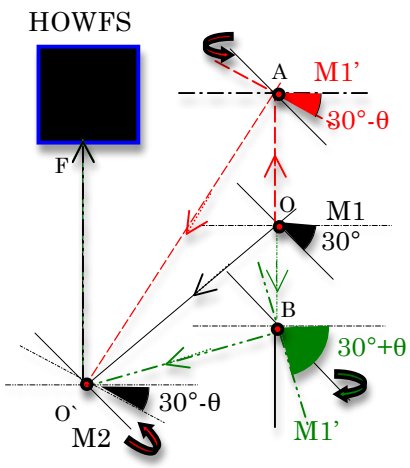
2. So for compensating the change in the focal length, the distance between M1 and M2 is changed accordingly as shown in the figure 12.ii), when elevation decreases, the position of the M1 can be increased by +Mz ($AO^2 > OO'$ i.e. distance between M1 and M2 is increased (red line)) so that the focal length OAO^2F^2 can be equal to the original focal length OO'F. Similarly, when the elevation increases, then the change in the focal length can be compensated by decreasing the distance between M1 and M2, so that the ray length OBO^1F^1 can be equal to the focal length OO'F.

3. By changing the distance between M1 and M2, the position of the focus changes as the reflected ray from M1 can't strike the center of M2 so in order to compensate the shift in the position, the mirror M1 and M2 should be tilted to bring back the focus at the optical axis of the WFS as shown in the figure 12.iii).

Figure 12



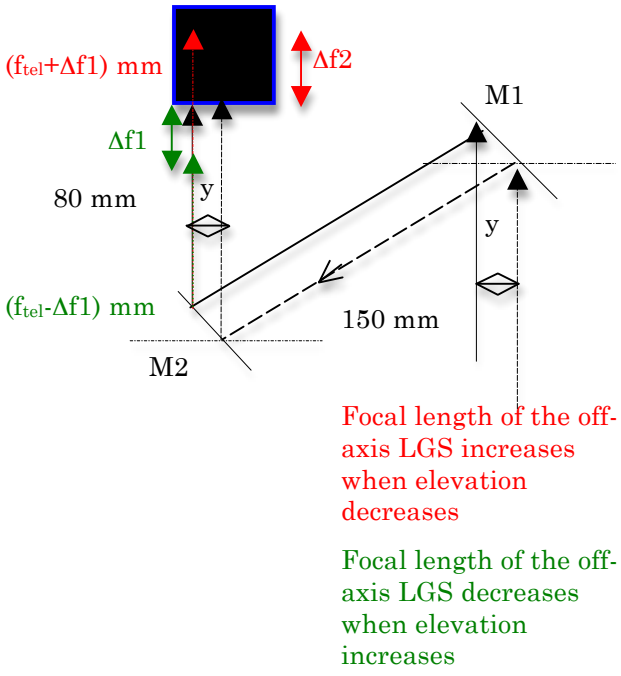
iii)



4. For the off-axis light source, as the rotational axis of the M1 and M2 is fixed, the ray doesn't strike at their center, so both the focus and the position changes as shown in figure 12.iv).

5. For compensating the shift in the position of the focus from the optical axis of the WFS, the large tilt (θ) should be provided to both M1 and M2 as shown in figure 12.v) in order to align the ray at the center of the M2.

iv)



v)

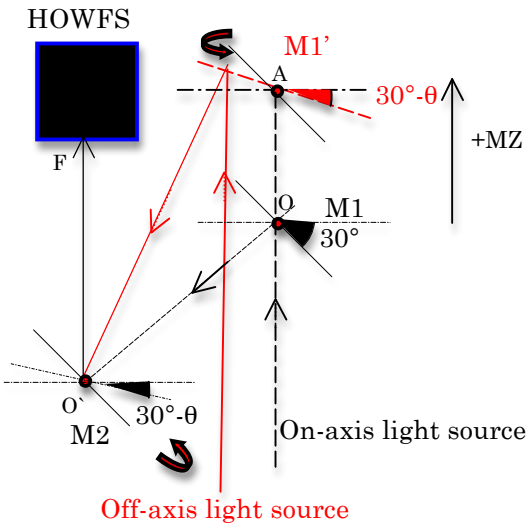


Table 7: Accuracy and Repeatability

AU1	Accuracy	Repeatability
M1 (x, y axis)	7.5 arcsecond	1.1 arcsecond (< 0.5 degree motion)
M2 (x, y axis)	28 arcsecond	4.2 arcsecond (< 2.0 degrees motion)
Linear stage	0.02 mm	0.003 mm

AU2	Accuracy	Repeatability
M1 (x, y axis)	8.5 arcsecond	1.2 arcsecond (< 0.6 degree motion)
M2 (x, y axis)	20 arcsecond	3.0 arcsecond (< 1.4 degrees motion)
Linear stage	0.021 mm	0.0032 mm

FUNCTIONAL REQUIREMENT OF THE GUIDE STAR ACQUISITION UNIT

The concept of the GSAU plays a crucial role in the focus tracking of the LGS. Its main functional requirements are described as follows-

1) AU must feed the guide star to the wavefront sensor.
2) It should track the focus of the guide star and aligns it within the optical path of the WS (especially HOWFS) and make sure that the beam is unchanged for any guide star in the FOV.
3) It should adjust the focus on the science camera.
4) It should control and move the position of the astronomical object on the science camera for performing Image and Slit dithering.

    4.1) Move AU's mirror to bring back the focus of the guide star on WFS
    4.2) WFS will detect the shift in the image and feedback the data to the DM.
    4.3) The DM tilts and moves the position of the star on the science camera

The focus correction can be split into two control loop, fast and slow i.e. the AU1 (using the LGS) can be used to measure the focus term due to the fast fluctuations of the atmosphere, while slow changes (tip-tilt and slow defocus) in the focus of the LGS can be corrected by using a AU2 (using the NGS).

OBJECTIVES OF THE INTERNSHIP

The objective of the internship was on the conceptual understanding, functioning, defining the trajectory motion for focus tracking and the testing of the both Acquisition unit, AU1 and AU2 for HOWFS and LOWFS respectively. The way of achieving the final goal of the focus tracking of the LGS consisted of two major sequential tasks which was further divided into various sequential subtasks, which are-

12.1) Backlash Measurement & Compensation

Backlash is the mechanical deficiency of the actuators because of which the focus tracking is difficult, so for understanding the backlash, measuring it for all the actuators of the GSAU at the observational floor and at the optical bench of the Subaru's LGSAO188 and also for their compensation, the different subtasks are-

1) To prepare the prototype of the AU1 at the observational floor and to understand the concept of measuring the backlash and computing its value at the different positions for the actuator X of M2.

2) After the calibration of AU1 and AU2 and before their integration on the AO optical bench, finding the backlash of all the 4 actuators of AU2 in XY plane at the observational floor.

3) After the integration of AU1 and AU2 at the AO optical bench and by using the AO calibration light source (for both LGS and NGS) and CCD, finding the backlash value of all the 8 actuators of AU1 and AU2 in XY plane and comparing the results with the experiment done at the observation floor.

4) To develop the backlash compensation algorithm and applying it to all the 8 axes/actuators of AU1 and AU2 and confirming the data again by taking the compensated backlash measurements for both AU1 and AU2 at the optical bench.

12.2) Focus Tracking

The change in the elevation of the LGS changes the focal point at the WFS, which can be compensated by changing the distance between the M1 and M2 of the AU (as described in section 9). So for understanding how the focus can be tracked by using the XPS Controller and the GSAU AU1, the subtasks are-

5) To understand the trajectory motion of the prototype (task 1) with the actuator X and Y by using Line-Arc trajectory mode of the XPS controller in XY plane.

6) By taking the AO calibration light source at the nominal position, developing the relation between the change in the focus with the changing position of the linear stage of the AU1 i.e. to find out for how much distance the 5 actuators should move (along with the backlash compensation if required) for aligning the chief ray to the optical axis of the WFS and for tracking the change in the focus.

BACKLASH MEASUREMENT & COMPENSATION

What is the Backlash?[8]

It is the mechanical deficiency in the Opto-mechanical devices that occurs when the motor driving an actuator reverses its load direction. Since there is a gap between the lifting screw and the gear, this lag, which is called backlash, is due to the small gaps between the teeth in the gears. As the gears turn in one direction, the teeth press together on one side. The gap in front of the teeth closes, and a gap opens behind. When the direction of the force-applying gear changes, it move to fill in the trailing gap and there is a brief period of rotation before the teeth mesh again.

TASK 1

To prepare the prototype of the AU1 at the observational floor and to understand the concept of measuring the backlash and computing its value at the different positions for the actuator X of M2

INTRODUCTION

It's necessary to find out the backlash of the actuator driving the GSAU in bi-direction motion for checking the angle difference in the mirror tilt due to the forward and the backward motion of the actuator at various positions. Here we developed a prototype of the AU1 considering only M2, and calculated the backlash of the actuator in X plane.

DEVICES

1) XPS controller[9]: an extremely high performance, integrated motion controller/ driver that offers high-speed communication, outstanding trajectory accuracy and powerful programming capability. It combines user-friendly web interface with advanced synchronization features to control the complex motions and trajectories. It is capable of driving up to 8 axes but for our prototype, we have used only 2 of them for the motion in X and Y plane.

2) Prototype of the HOWFS having fixed mirror M2 (fig 14). The mirror M2's tilt is not zero at the nominal position of the actuator which is 0 mm.

3) Two NEWPORT LTA-HL actuators with shift from 0 mm to 25 mm

4) An Auto collimating telescope for taking the measurements and,

5) Two reflecting mirrors (R1 and R2)

[8] http://en.wikipedia.org/wiki/Backlash

[9] User's Manual Software Tools Tutorial for XPS Controller

DESCRIPTION of the MODEL

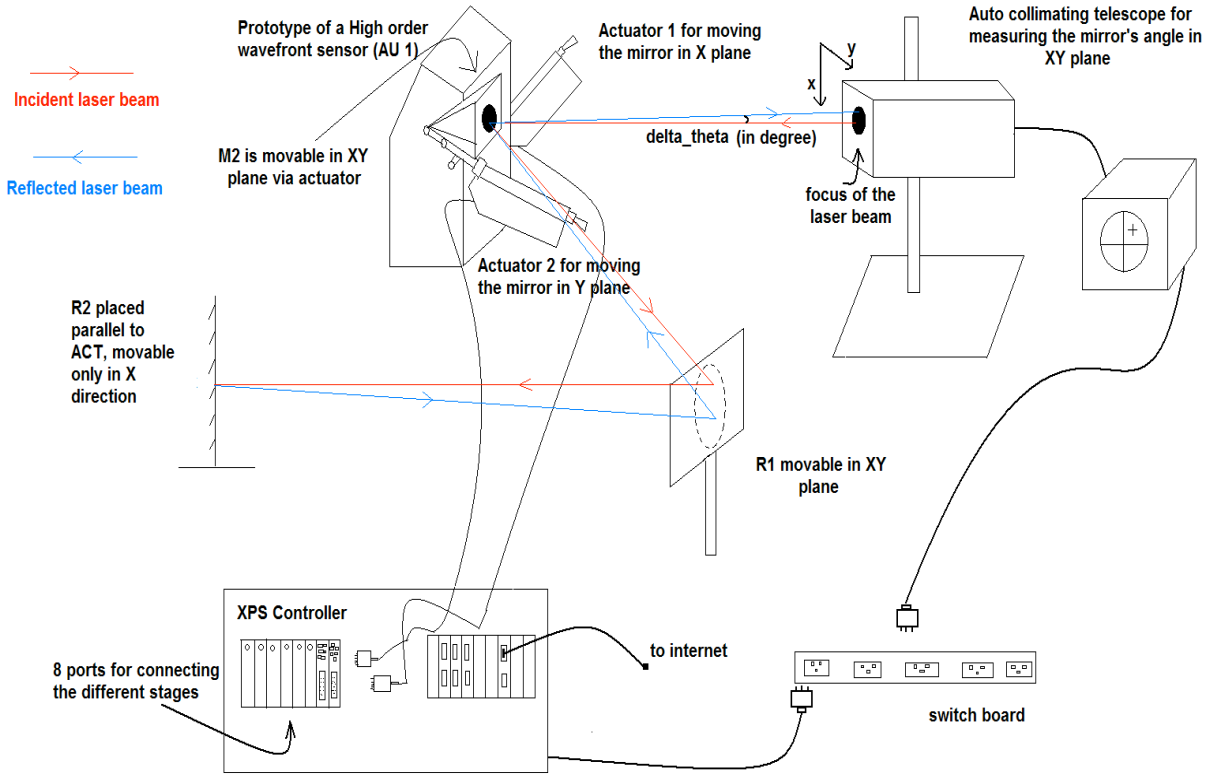
In the fig 13,

1) The mirror M2's direction in the prototype for the guide star AU1 for HOWFS is controlled by the Actuator 1 and Actuator 2 in X and Y direction respectively which are driven by the XPS controller.

2) An auto-collimating telescope (ACT) is used to measure the angle, in XY direction, between the incident and the reflected laser beam.

3) Two reflecting mirrors are used to measure the deviation for all the positions / shifts of the actuator from 0 mm to 25 mm. Reflecting mirror R2 is placed parallel to the ACT which can be moved in x direction only while mirror R1 can be moved in x and y direction both in order to track the focus of the incident laser beam while measuring the deviation for large tilt of the mirror M2 corresponding to the large shift of the actuator.

Figure 13: Prototype of the HOWFS



Note: The position of the R1 and R2 is moved manually for the tracking of the focus of the laser beam in order to take the measurements (deviation value) for different tilts of the mirror M2 corresponding to the shifts of the actuator (from 0.50 mm to 24.50 mm).

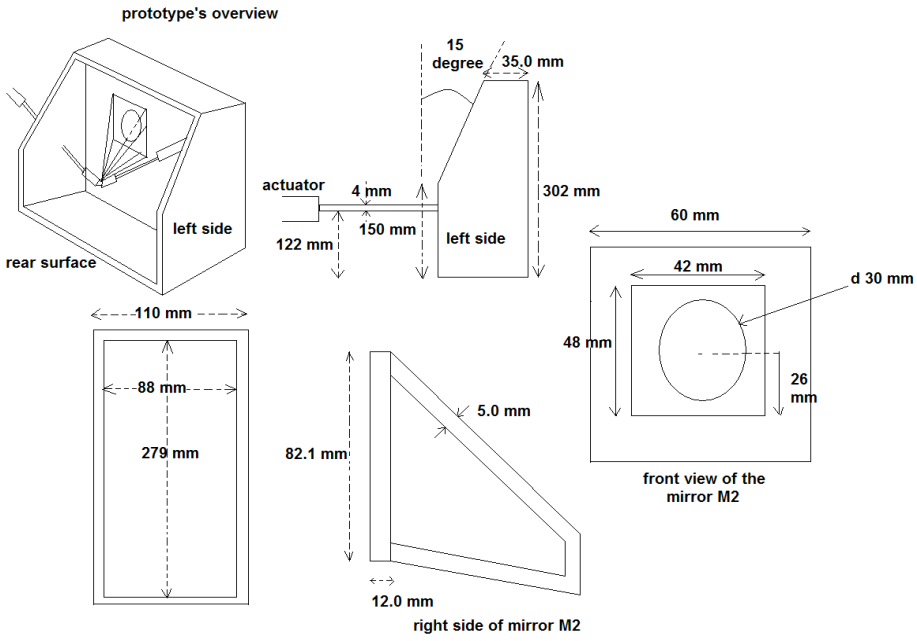
METHOD

As Backlash is defined as the difference between the forward motion and the backward motion at a particular position, so for measuring it the following steps has been done-

1) For each position ($sx^{10}(n)$, measurement points) the actuator X is moved back and forth for 10 times measuring the 10 values (will be slightly different if backlash exist) of $\Delta\theta x(n)$ ($\Delta\theta y(n)$ is similar for all the points) at the position 'n' (fig 15).

2) Then the position, sx(0) for which $\Delta\theta_x=\Delta\theta x(0)=\Delta\theta y(0)$ is calculated manually by adjusting surface 1 within surface 2 horizontally (fig 16), making them parallel to each other, and by measuring the distance between them at each edge, which should be equal.

Figure 14: The dimension of the prototype for HOWFS



3) Similarly for the actuator Y, making the surface parallel vertically, the zero position is calculated. The measurement positions, sx(n) for positioner X are then subtracted from sx(0), hence obtaining the displacement in positions. The new measurements are taken again from the ACT, for the displaced positions.

[10] Please refer to the ANNEXE for the 'List of Symbols' for the definition of the technical symbols used through out the report.

Figure 15

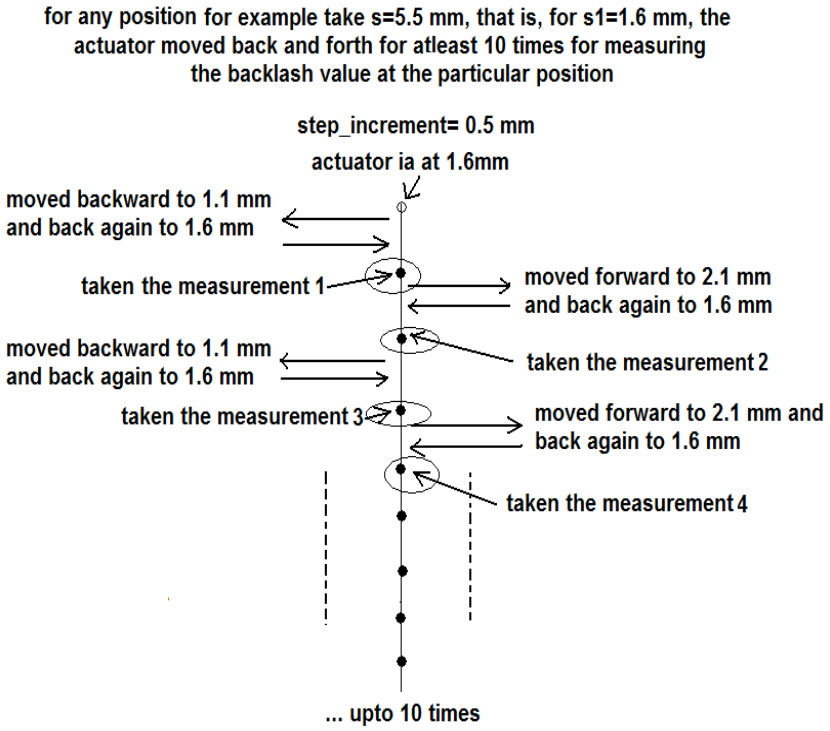
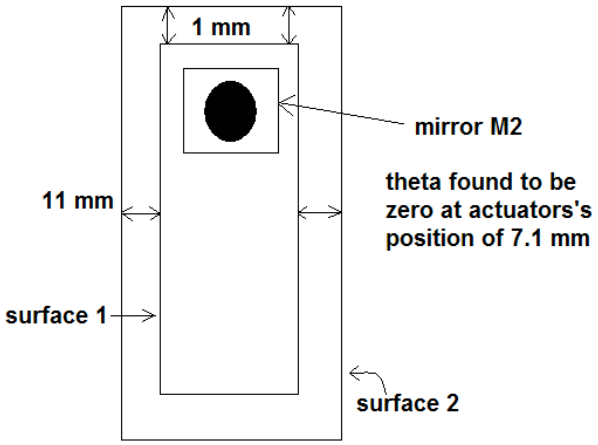


Figure 16



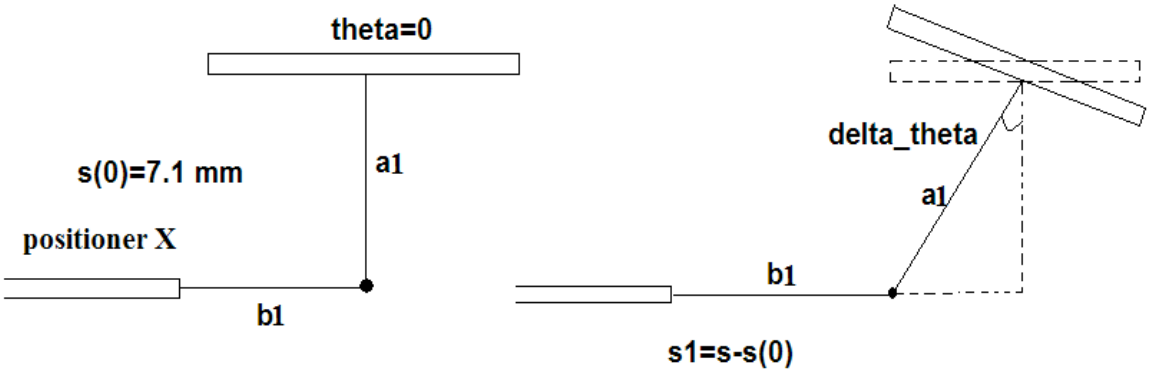
front view of the prototype

4) So, s1(n)= sx(n)-sx(0), the new displaced position of the actuator (fig 17). The $\Delta\theta_x(n)$ for every displaced position, s1(n) is measured via ACT.
As explained in fig 14, $\Delta\theta_x(n)$ can be calculated as the difference in the forward measurement and the backward measurement.
So,

$$\Delta\theta_x(n) = \sum_{n=1}^{48} \frac{\sum_{i=1}^{10} \sqrt{(\Delta\theta x_f - \Delta\theta x_b)^2 + (\Delta\theta y_f - \Delta\theta y_b)^2}}{n} \quad (1)$$

, where n=48 points with a step of 0.5 mm from 0 to 24 mm positions of the actuator

Figure 17



5) The distance from M2 and the lever for positioner X is measured as 'a1= 65 mm' (fig 18) and for positioner Y as 'a2=88.86 mm' (fig 19). And the distance from the lever to the positioner X is 'b1= 22.08 mm' and for positioner Y is 'b2=22.01 mm'.

Figure 18: Dimensions for the Positioner X

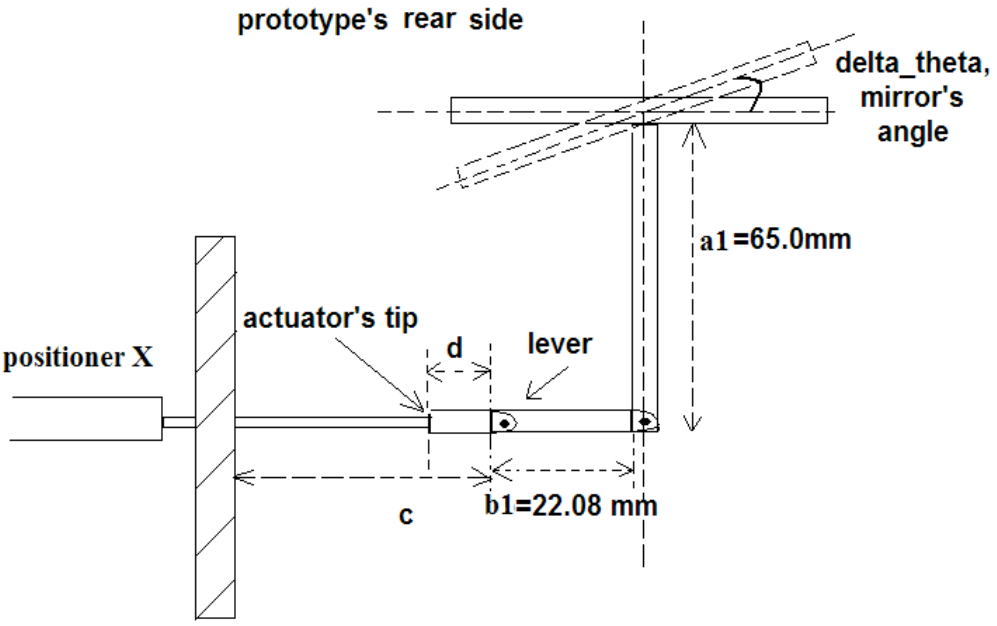
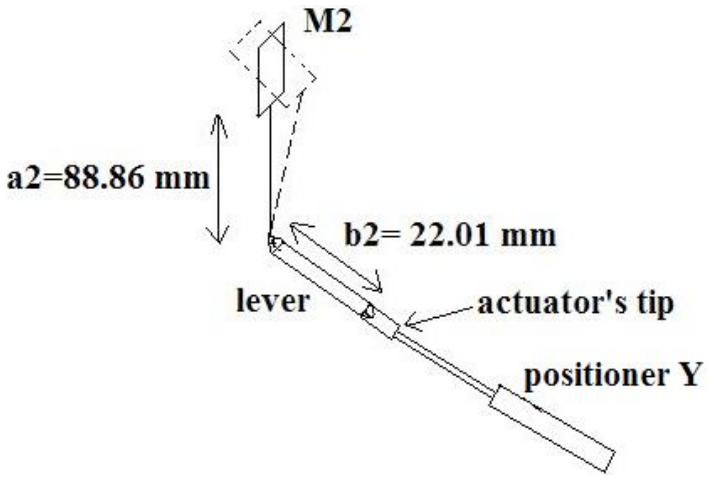


Figure 19: Dimensions for the Positioner Y



We use a model function[11], which is the relation between the positions of linear actuator (Newport LTA-HL) v.s. mirror tilt angle. (M1-x, M1-y, M2-x, M2-y)

$$shift = a1 \times \sin(theta) - \sqrt{b1^2 - [a1^2 \times (1 - \cos(theta))^2]} + b1 \quad (2)$$

As their relation is almost linear (Plot 1), so θ(n) can be approximated for the displaced position by using Newton-Raphson method for finding the root of the linear equation (see CODING 2).
Mathematically by Newton Raphson method the value of angle corresponding to the shift can be find out as-

$$\theta(0) = \frac{start\_point}{a1} \quad \text{, Initial guess assuming that } \theta(0) << 1$$

start_point= the first displaced position of s1
so, from equation (2)

$$fy(0) = a1 \times \sin(\theta(0)) - \sqrt{b1^2 - [a1^2 \times (1 - \cos(\theta(0)))^2]} + b1 - start\_point \quad (3)$$

$$fd(0) = a1 \times \cos(\theta(0)) - \frac{2 \times a1^2 \times (1 - \cos(\theta(0))) \times \sin(\theta(0))}{b1^2 - [a1^2 \times (1 - \cos(\theta(0)))]} \quad (4)$$

$$\theta(1) = \theta(0) - \frac{fy(0)}{fd(0)} \quad (5)$$

$$slope = \frac{start\_point}{\theta(0)} \quad (6)$$

[11] the model function was provided by Mr. Yosuke Minowa

$$\theta(n) = \frac{s1(n)}{slope} \tag{7}$$

$$fy(n) = a1 \times \sin(\theta(n)) - \sqrt{b1^2 - [a1^2 \times (1 - \cos(\theta(n)))^2]} + b1 - s1(n) \tag{8}$$

$$fd(n) = a1 \times \cos(\theta(n)) - \frac{2 \times a1^2 \times (1 - \cos(\theta(n))) \times \sin(\theta(n))}{b1^2 - [a1^2 \times (1 - \cos(\theta(n)))]} \tag{9}$$

$$\theta(n+1) = \theta(n) - \frac{fy(n)}{fd(n)} \tag{10}$$

So,
$\Sigma\theta(n) = \theta(n) + \Delta\theta_x(n)$
Hence,

$$s2(\Sigma\theta) = a1 \times \sin(\Sigma\theta) - \sqrt{b1^2 - a1^2 \times (1 - \cos(\Sigma\theta))^2} + b1 \tag{11}$$

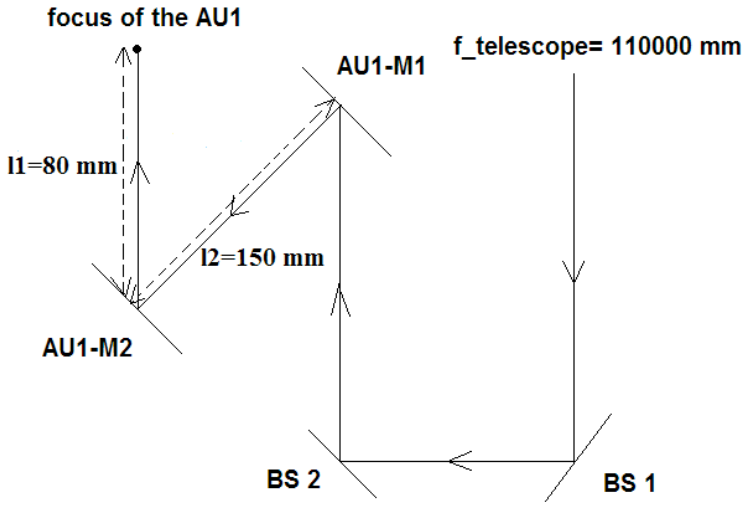
s2(n)=s`-s(0), s` being the actuator's actual position
Therefore,

Backlash value (in micrometers), $\Delta s(n) = \frac{s2(n) - s1(n)}{2}$ (12)

Also the sky backlash in arcseconds for AU1 as shown in fig 20 is calculated as-

$$sky\_backlash(n) = \frac{l2 \times \Delta s(n) \times 180 \times 3600}{f\_telescope \times \Pi} \tag{13}$$

Figure 20



IDL CODING REFERENCE

Following three procedures were developed for measuring the mirror M2's angle repeatedly via ACT by moving the actuator back and forth (from 0.50 mm to 24.50 mm) for at least 10 times for each measurement point with the step increment of 0.50 mm.

Starting point = 0.50 mm
Ending point = 24.50 mm
Actuator's zero mirror angle position, sx(0) = 7.1 mm
s1(n) = sx(n)-sx(0)
Step increment = 0.50 mm
Total number of measurement points at which the backlash is measured, n = 48

Coding 1 is for moving the actuator back and forth for at least 10 times for each measurement point and measuring the mirror's angle from ACT.

Coding 2 is for creating numerical code for measuring the approximated values of angle (θ) by Newton-Raphson method for linear equations.

Coding 3 is for reading the data (measured by ACT) and performing the calculations like finding the average, standard deviation, variance and corresponding shift (backlash) in μm after applying the formula for the conversion of backlash from degrees/radians to micrometer.

RESULT

Several simulations has been done-

1) First set of values of backlash has been taken from the setup having mirror M2 and ACT only (without R1 and R2 in fig 12), for 6 measurement points-

Start point = 5.3 mm
End point = 5.8 mm
Step increment = 0.1 mm
Measurement points under consideration, sx(n), in mm = {5.3,5.4,5.5,5.6,5.7,5.8}

Average value of backlash found to be = 5.60167 μm
Average value of standard deviation found to be = 0.308578 μm

2) Second set of values of backlash is taken with the setup shown above in the fig 12, measuring angle for all the positions of the actuator (R1 and R2 is moved manually for making the laser spot always in the FOV of the ACT).

Start point = 0.50
End point = 24.50
Step increment = 0.5
Measurement points under consideration, sx(n), in mm= {0.50,1.0,1.5,...,24.0,24.50}
Actuator's position at zero mirror's angle, sx(0) = 7.1 mm

Average value of backlash found to be = 7.58134 μm
Average value of standard deviation found to be = 0.226021 μm
Sky backlash in arc seconds found to be = 0.0369407 arcsecond

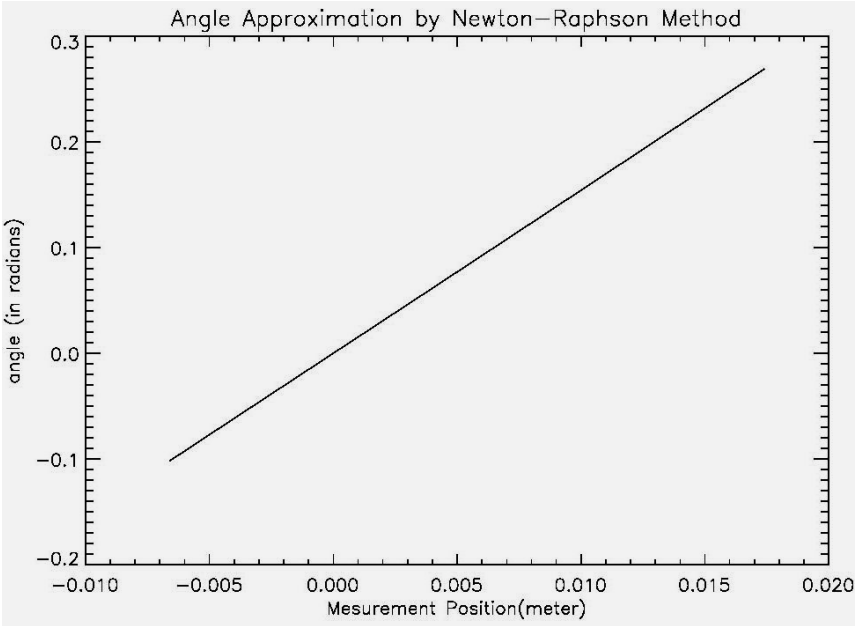
3) Third set of values of backlash is taken with the same configuration as above but without moving R1 and R2.

Start point = 5.0
End point = 5.25
Step increment = 0.05
Measurement points under consideration, s(n), in mm= {5.0,5.05,5.10,5.15,5.20,5.25}
Actuator's position at zero mirror's angle, s (0) = 7.1 mm

Average value of backlash found to be = 6.44250 μm
Average value of standard deviation found to be = 0.130745 μm

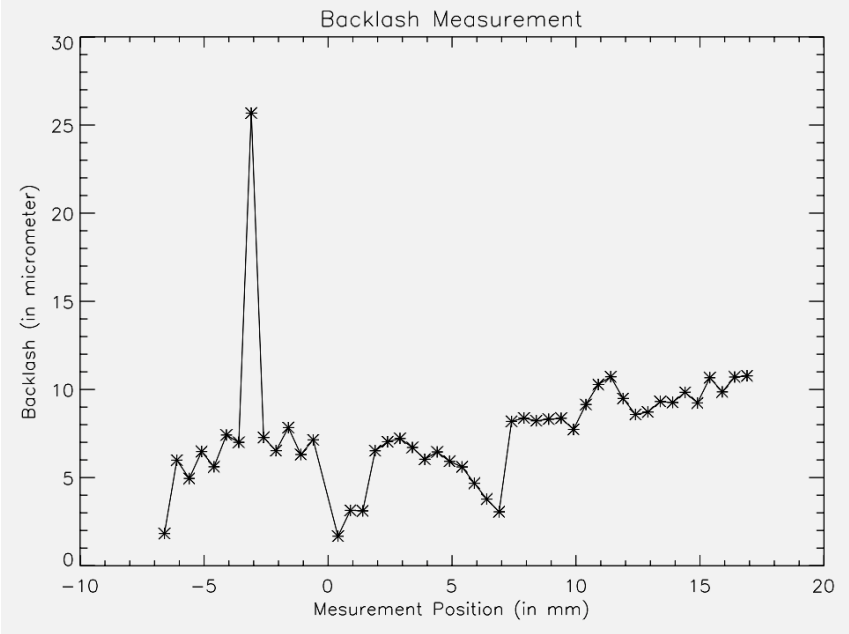
Plot 1

Plot between the measurement points (ranging from 0 mm to 25 mm) and their corresponding approximated value for θ (in radians) obtained by Newton Raphson method showing almost linear relation between the two.



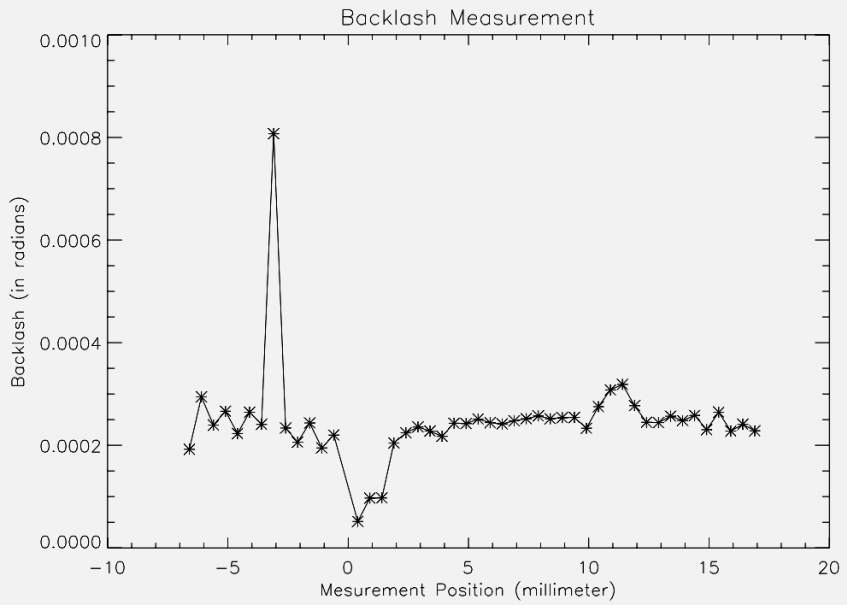
Plot 2

Plot between the measurement points (ranging from 0 mm to 25 mm) and their corresponding backlash values (in μm)



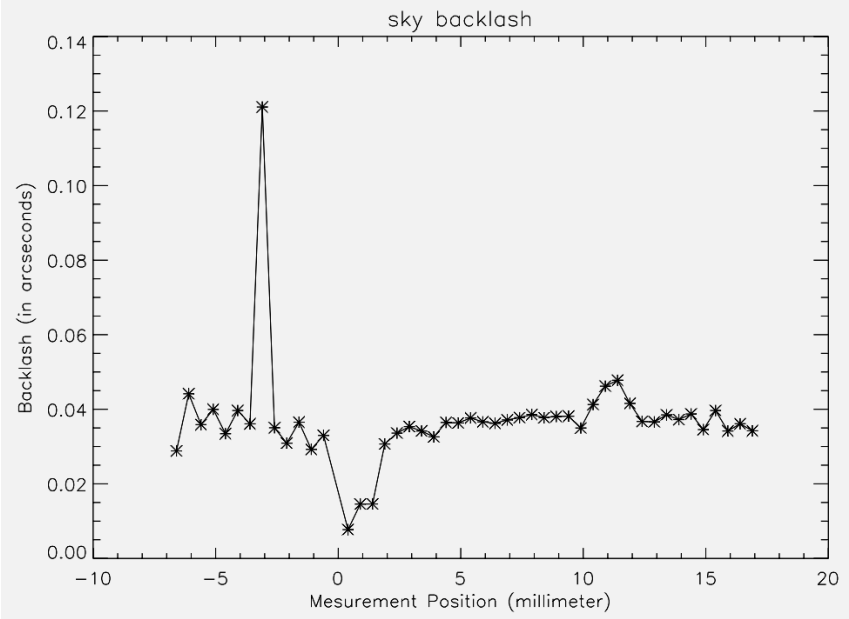
Plot 3

Plot between the measurement points (ranging from 0 mm to 25 mm) and their corresponding backlash values (in radians)



Plot 4

Plot showing the sky backlash in arcseconds



CONCLUSION FOR TASK1

For the measurement points from 0.50 mm to 24.50 mm, backlash values seems to be approximately doubled of the data which is calculated for less number of measurement points and also without moving the mirror R1 and R2. Because of the backlash there is a tilt in the mirror M2 when it is moved back and forth by the actuator, which reflects another ray out of focus of the ACT. Hence the ACT measures the backlash value two times (that is why the backlash value is divided by 2 in the equation 12). Moreover there is an error introduced while moving the mirror R1 and R2 manually for taking the measurements for all the points and keeping the focus all the time at ACT.

The backlash value found to be approximately in between 5.60167 μm to 7.58134 μm with standard deviation of 0.130745 μm to 0.308578 μm.

As all of the 10 actuators in the real system are of same type, we can consider the same backlash value for all of them.

TASK 2

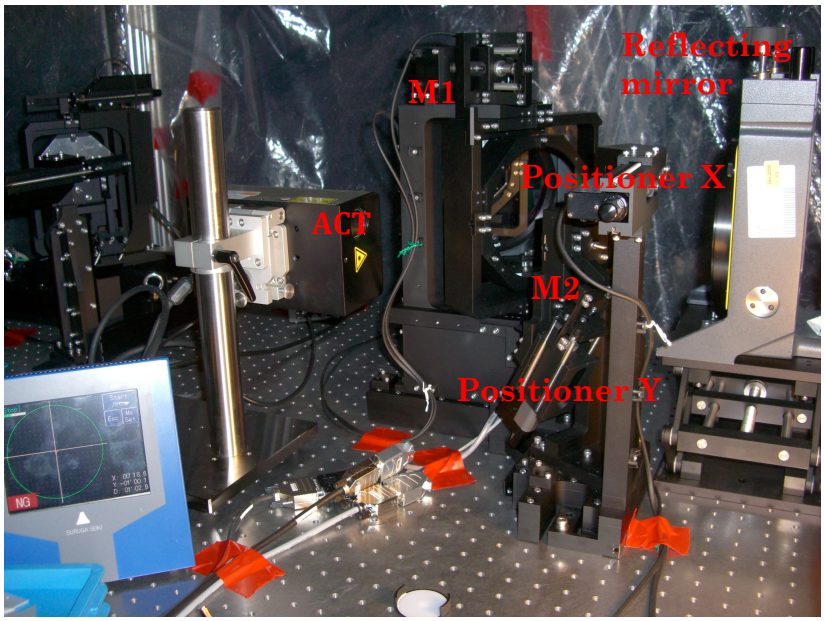
After the calibration of AU1 and AU2 and before their integration on the AO optical bench, finding the backlash of all the 4 actuators of AU2 in XY at the observational floor

INTRODUCTION

Now moving to the actual GSAU, before their installation on the LGSAO188 optical bench at the summit, first they were calibrated[12]. And after their calibration and before their integration on the optical bench, the backlash value of the 4 actuators of the AU1[13] and AU2 is calculated in order to test for the systematic error that could be introduced at the observational floor.

DEVICES

Figure 21: GSAU AU2 at the observational floor



1. Guide star acquisition unit 2 (AU2) for LOWFS
2. 4 NEWPORT LTA-HL linear actuators
3. Auto-collimating Telescope for the measurements in XY plane
4. XPS controller

METHOD

For finding the backlash value of all the actuators of the AU2 in a real system at the observational floor, the same procedure has been followed as the one used for the

[12] Calibration of the guide star acquisition unit AU1 and AU2 was done by Mr. Y. MINOWA and Miss. C. ONG

[13] The backlash measurement for AU1 at the observational floor is done by Mr. Y. MINOWA

prototype in Task 1. For all the actuators, the repeatability[14] has been tested at every position from 0 to 25 mm before calculating the backlash value.

1. Here the actuator position, sx(n)=s1(n)=AU2.s_{M1X}

2. As explained earlier (in fig 12), for tracking the focus of the LGS, the mirror M1 and M2 needs to be adjusted so as to align the ray to the optical axis of the WFS (or ACT in this case). So for the backlash measurement of M1X actuator, while moving the mirror M1 in X plane, the mirror M2 is also moved simultaneously[15] in X plane and the M1Y and M2Y actuator are kept at their nominal position so as to not to loose the focus within the FOV of the ACT. Also for the measurement of backlash for M2X actuator, M1X is moved simultaneously.
   Similarly for the M1Y, M1X and M2X actuator are kept at their nominal position and M2Y actuator is moved accordingly and vice versa for M2Y actuator.

3. For finding the backlash value and before taking the 10 measurements at every point, the repeatability is tested at each point i.e. at each point, first the actuator is moved 5 times in backward direction in order to be sure that the measurement for backlash actually starts at the desired position and then the 10 measurements were taken with back and forth motion.

4. The mode for ACT for measuring the angle is set as 'arcsecond'. So the $\Delta\theta_x(n)$ which gives the backlash value directly in arcsecond can be calculated with the help of the equation (14) as-.

$$\Delta\theta_x(n) = \sum_{n=1}^{48} \frac{\sum_{i=1}^{10} \sqrt{(\Delta\theta x_f - \Delta\theta x_b)^2 + (\Delta\theta y_f - \Delta\theta y_b)^2}}{n} \quad (14),$$

where n=48 points with a step of 0.5 mm from 0 to 24 mm positions of the actuator

IDL CODING REFERENCE

Starting point	=	0.50 mm
Ending point	=	24.50 mm
Step increment	=	0.50 mm

[14] Repeatability is the ability of the actuator to give same value of the mirror tilt when its position is moved back several times before actually moving the actuator in the forward direction. Suppose the actuator is at position 1mm, so before moving it in a forward direction the actuator will be moved back, for example to 0.5 mm and then back to 1mm several times. It removes the hysteresis of the actuator and also minimizes the backlash effect. It's the command value versus the reading of the encoder of the XPS Controller.

[15] While finding the backlash of the actuator M1X/M1Y, the corresponding amount by which the mirror M2X/M2Y should move so as not to lose the focus was provided by Mr. Yosuke Minowa.

Total number of measurement points at which the backlash is measured, n = 48

Coding 4 is for moving the actuator back and forth for at least 10 times for each measurement point and measuring the mirror's angle from ACT for AU2.s_{M1X} and AU2.s_{M2X} at the observation floor.

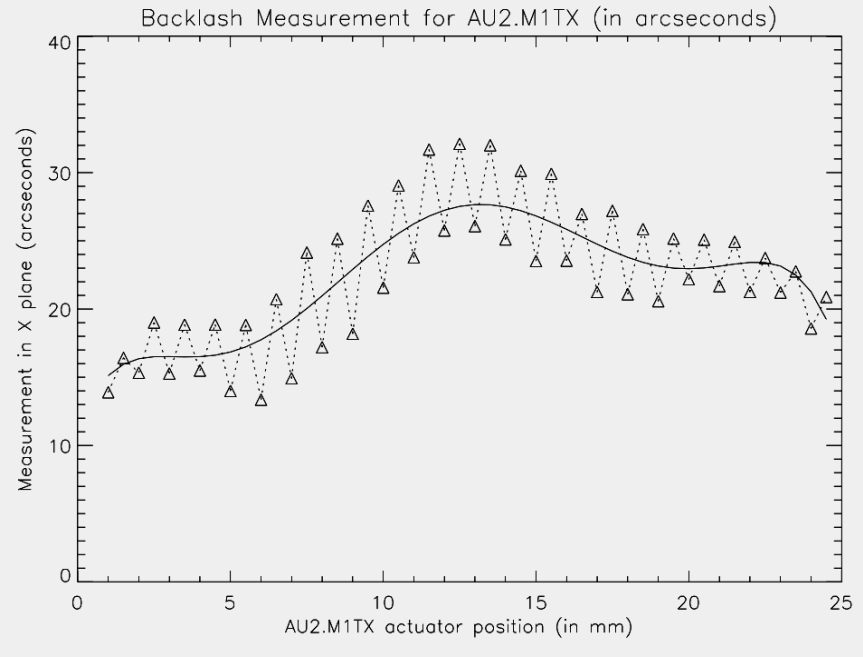
Coding 5 is for moving the actuator back and forth for at least 10 times for each measurement point and measuring the mirror's angle from ACT for AU2.s_{M1Y} and AU2.s_{M2Y}.

Coding 6 is for reading the data (measured by ACT) and performing the calculations for the measurement of backlash of the AU2's all 4 actuators.

RESULT

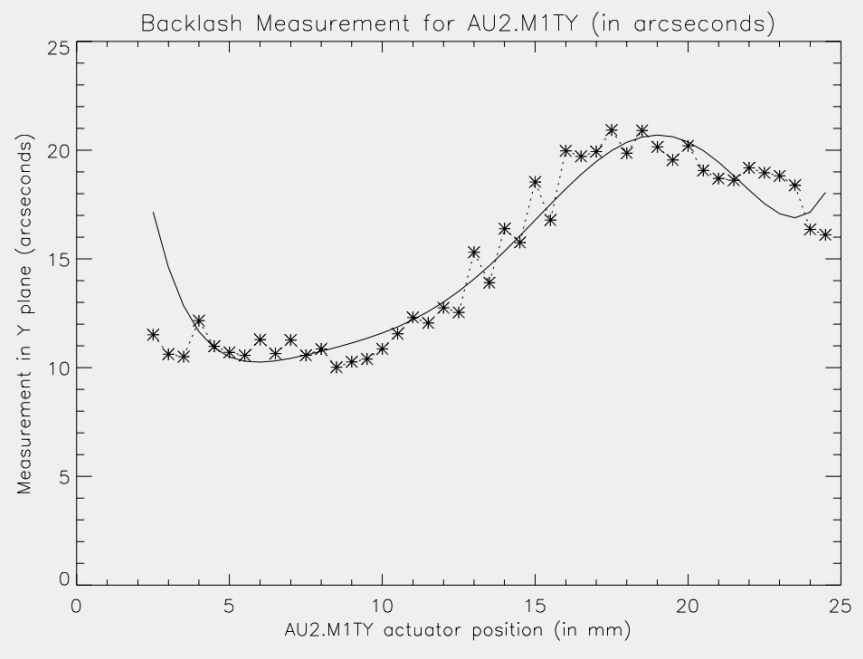
Plot 5

Plot 5 shows the backlash value in arcseconds for the positions AU2.s_{M1X}
The black dotted line with the symbol 'Δ' shows the actual backlash measurement while the thick black line shows the best fitting higher polynomial function



Plot 6

Plot 6 shows the measured backlash value (dotted line) for AU2.s_{M1Y} and its corresponding fitting model of 6th order polynomial function (thick black line)



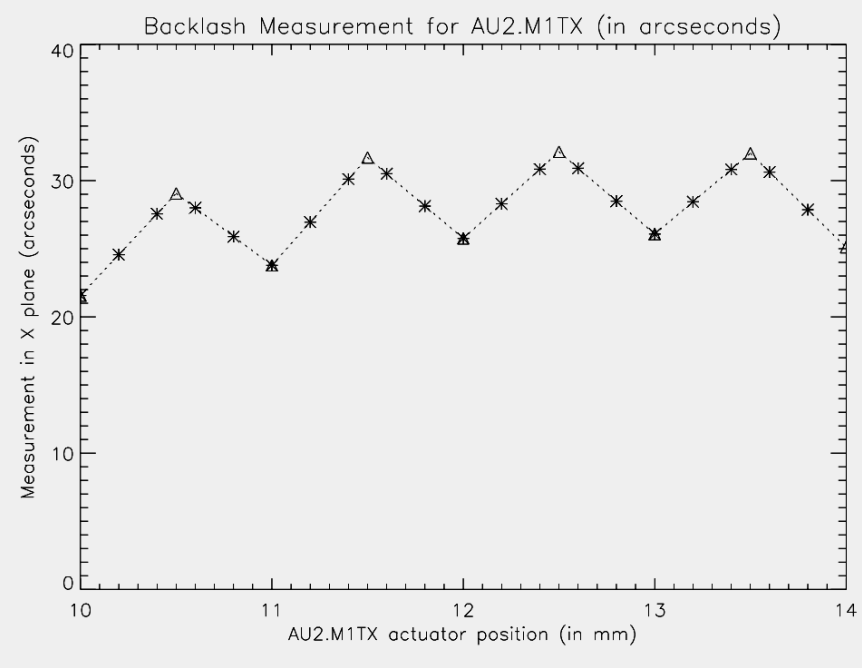
We need a model function, which can be used to find out the backlash value for any actuator (M1X, M1Y, M2X, M2Y) at any position desired by the user. From plot 5 for AU2.s_{M1X}, it's clear that the behavior of the backlash from 0.5 to 24.5 mm is oscillating. So for fitting the curve we need to find out the best-fit function comprising of oscillation fitting as well as higher order polynomial fitting, which is hard to fit.
Also it's quite difficult to fit the curve which are not Gaussian for example for M1X and M1Y, increasing the order of the polynomial only makes the curve steep at the edges.

So for simplicity, we used the interpolation method considering only 11 positions with the step of 0.1 mm around the nominal position[16] for best fitting the backlash curve, as shown in Plot 7 and Plot 8 for AU2.s_{M1X} and AU2.s_{M2X} respectively.

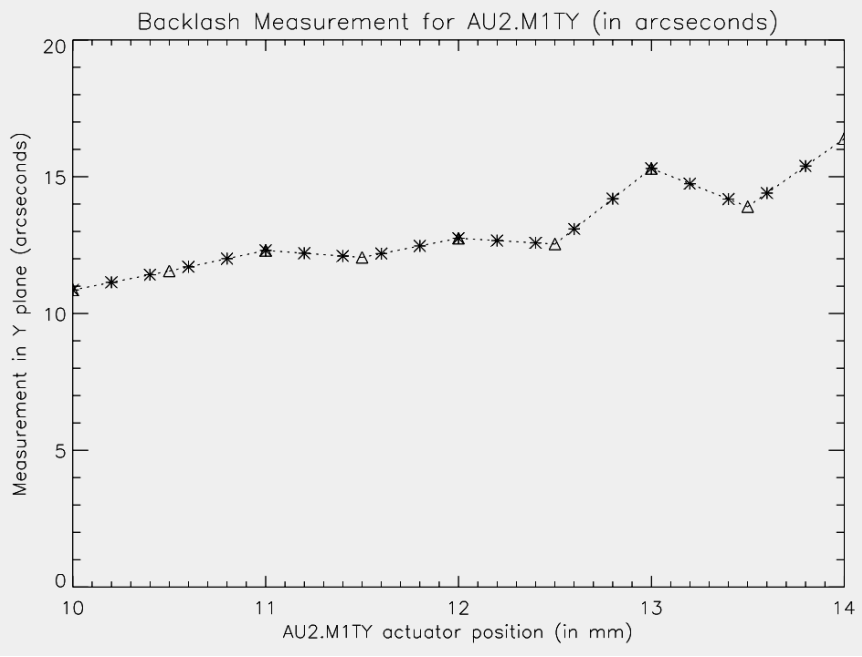
Here, the symbol 'Δ' shows the measured backlash value, and, '∗', shows the interpolated backlash value.

Plot 7

[16] The relation between the backlash and the position of the actuator can be considered as linear with such a small number of measurement points (11 points with 0.1 mm step) around the nominal position.

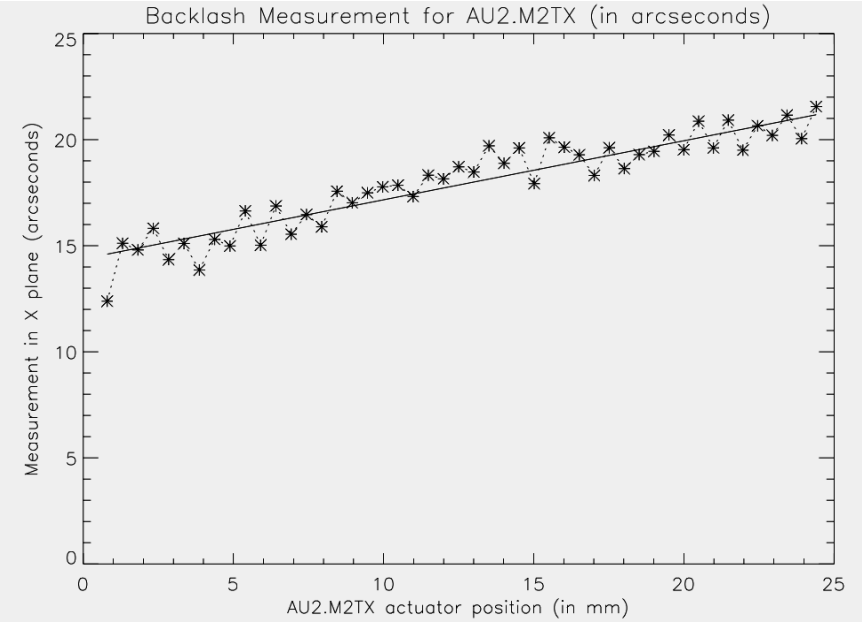


Plot 8



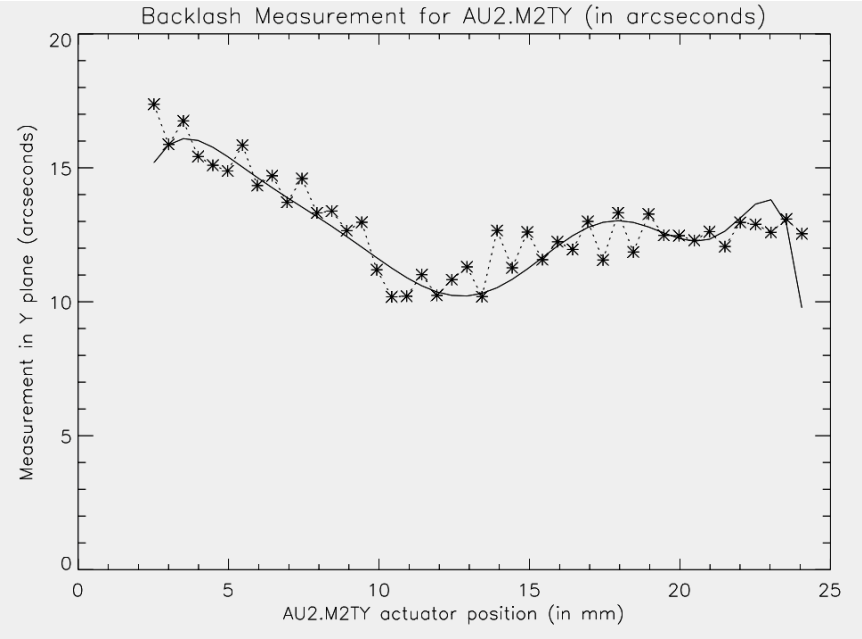
Plot 9

Plot 9 shows the measured backlash value (dotted lines with '*') for AU2.s_{M2X} and its corresponding linear fitting model (thick black line)



Plot 10

Plot 10 shows the measured backlash value (dotted lines with '*') for AU2.s_{M2Y} and its corresponding fitting model of 9th order polynomial function (thick black line)



CONCLUSION FOR TASK 2

In this task, we measured the backlash value for all the 4 actuators of AU2 at all the positions of the actuator at the observational floor, which can be compared later with the backlash value that will be calculated after the integration of AU2 at the optical bench at the summit.

For Au2.M1X, the backlash value at all the positions varies between 13-44 arcseconds.

For Au2.M1Y, the backlash value at all the positions varies between 10-22 arcseconds.

For Au2.M2X, the backlash value at all the positions varies between 12-21 arcseconds.

For Au2.M2Y, the backlash value at all the positions varies between 10-18 arcseconds.

Note: All the measurement at every position contains the ACT reading error, which is 1 arcsecond. And the reason for the variations in the backlash value at the different positions is may be because of the manufacturing of the actuators.

Also we developed the backlash measurement model for all the 4 actuators of AU2 that can be used in the future for finding out the backlash value at any random position desired by the user.

TASK 3

After the integration of AU1 and AU2 at the AO optical bench and by using the AO calibration light source (for both LGS and NGS) and CCD, finding the backlash value of all the 8 actuators of AU1 and AU2 in XY plane and comparing the results with the experiment done at the observation floor

INTRODUCTION

As the experiment of calculating the backlash for AU1 and AU2 is done at the observational floor, now after the integration of AU1 and AU2 at the optical bench of the LGSAO188, we calculated the backlash value again for AU1 and AU2 and compared them with the values measured at the observational floor.

DEVICES at the AO OPTICAL BENCH

1. Guide star acquisition unit 1 (AU1) with acquisition camera for HOWFS
2. Guide star acquisition unit 2 (AU2) with acquisition camera for LOWFS
3. 8 NEWPORT LTA-HL linear actuators
4. AO calibration light source
5. CCD camera
6. 2 XPS controller for HOWFS and LOWFS respectively

METHOD[17]

The procedure for calculating the backlash value is same as in Task 2 but the way of measuring the angle is different which is explained below.

1. At the AO optical bench of the AO188, we use AO calibration light source[18] and a CCD camera for imaging it.

Why we can't use ACT at the optical bench for AU1 and AU2?

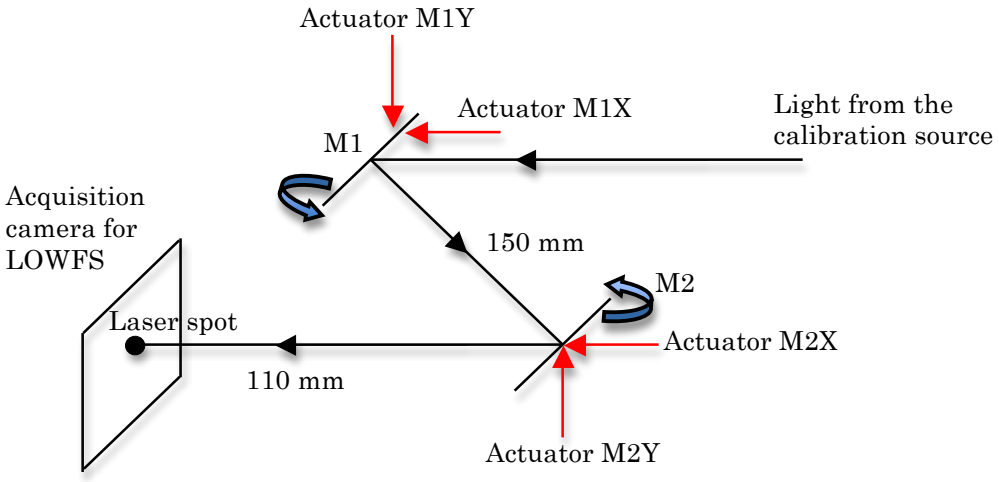
The two mirrors of AU1 are having dielectric multilayer coating as it needs high reflectivity R > 99% for covering the broader range of the wavelength (0.45-1 micron) and for AU1, its mirrors are silver coated as only R > 98 % for reflectivity is sufficient for covering the wavelength range of 0.45-2.5 micron. The ACT has a narrow light because it uses Al LED, which is not good enough to detect the reflected light because of the difference in the intensity of the different reflected beam. Calibration light source uses Laser diodes that produce broader light and the changing intensity does not affect the overall intensity of the Laser light. Moreover by using CCD we can adjust the intensity by increasing or decreasing the exposure time.

[17] The concept of calculating the backlash for AU1 and AU2 is same. Here the processing of the images for obtaining the backlash value is explained for AU2 in detail.

[18] Please see the explanation on Page 20-21

2. When the actuator moves back and forth, the mirror tilt and the spot moves on the CCD either in +ve or –ve direction depending on the direction of the actuator motion as shown in the figure below.

Figure 22



3. So the repeatability and the backlash was measured by moving the actuator back and forth and the images (.fits files) with the displaced position of spot were obtained for the 11 positions of the actuator (around nominal point). So the difference between the forward measurement and the backward measurement of the spot motion in pixels was calculated as-

$$\Delta P_x(n) = \sum_{n=1}^{48} \frac{\sum_{i=1}^{10} \sqrt{(\Delta Px_f - \Delta Px_b)^2 + (\Delta Py_f - \Delta Py_b)^2}}{n} \quad (15),$$

where n=11 points with a step of 0.5 mm from 11.5 to 12.5 mm positions of the actuator

4. Before calculating the backlash value in arcseconds from the backlash value in pixels ($\Delta P_x(n)$), first, the pixel scale (arcsecond/pixel) is calculated as follows-

   i) Move the M1X actuator of AU2 at the nominal position i.e. at 12.0 mm, move backward by 0.1 mm and again move to the nominal position, take the image and record the spot center[19] in pixel as (ΔPx_0, ΔPy_0)
   ii) Move the actuator backward by 0.1 mm from the nominal position and record the spot center as (ΔPx_b, ΔPy_b)

[19] The procedure for knowing the spot center is explained later in detail.

iii) Move the actuator forward by 0.1 mm from the nominal position and record the spot center as (ΔPx_f, ΔPy_f)
iv) Find the value of the pixel scale by the following equation-

$$pixel\_scale_{-} = \frac{\theta_{-} - \theta_0}{\sqrt{(\Delta Px_b - \Delta Px_0)^2 + (\Delta Py_b - \Delta Py_0)^2}} \quad (16)$$

$$pixel\_scale_{+} = \frac{\theta_{+} - \theta_0}{\sqrt{(\Delta Px_f - \Delta Px_0)^2 + (\Delta Py_f - \Delta Py_0)^2}} \quad (17)$$

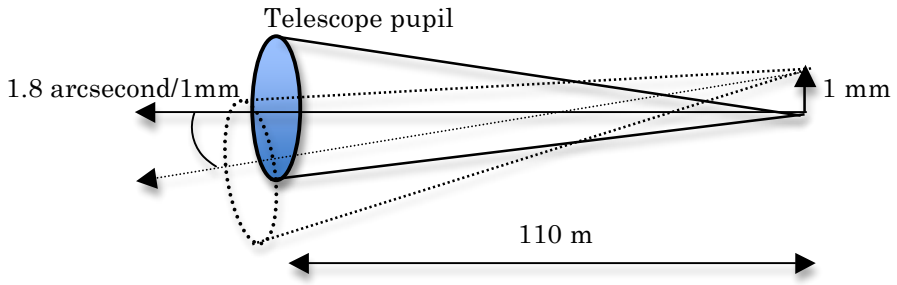
$$pixel\_scale = \frac{pixel\_scale_{-} + pixel\_scale_{+}}{2} \quad (18)$$

Note[20]: Here, θ defines the amount of the tilt in the mirror when the actuator moves (θ_0 : tilt when the actuator was at the nominal position, θ_- : tilt when the actuator moves backward, θ_+ : tilt when the actuator moves forward)

Hence, the pixel_scale for AU2.M1X found to be 4.2975 arcsecond/pixel. It means that the movement of the spot by one pixel corresponds to the tilt of the 4.2975 arcsecond on sky.

Confirmation of the pixel scale value

Figure 23

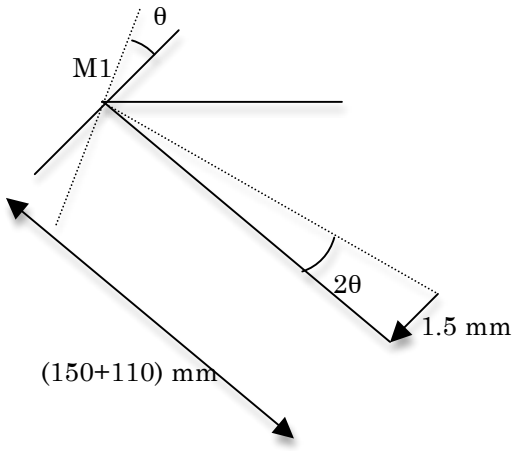


The change in the angle for 1mm move of the actuator is 1.8 arcsecond/mm ($\frac{1mm}{110m} \times \frac{180}{\Pi} \times 3600$). The specification of CCD has 20mas of tilt on sky for 1 pixel

[20] When the actuator moves (either in +ve or –ve direction) then by how much amount the mirror can be tilted was unknown till this experiment. So some raw data was provided by Hayano-san for the 11 position of the actuator with the corresponding tilt of the mirror AU2M1X. So after fitting those values in the linear equation (in reality their relation varies as the 4th order polynomial function but for small number of points (11 points with increment of 0.1 mm), it can be considered as linear) we found the slope, through which we got the value of θ for the desired position of the actuator.

movement so it means that 1 pixel change corresponds to 11μm movement of the actuator ($\frac{0.02 arc\sec ond \times 10^{-3} m}{1.8 arc\sec ond}$).
For AU2.M1X, the pixel difference (ΔPx_b - ΔPx_0) is 135 pixels for which the corresponding movement in M1X actuator is 1.5 mm ($\frac{135 pixel \times 11 \mu m}{1 pixel}$).

Figure 24



So for the actuator shift of 1.5 mm, which corresponds to the difference of 135 pixels, the angle in arcsecond can be calculated as $\theta' = \frac{1}{2} \times \frac{1.5mm}{(150+110)mm} \times \frac{180}{\Pi} \times 60 = 11'$
Hence 1 pixel will correspond to ≈ 4 arcsecond ($\frac{11' \times 60}{135} arc\sec ond$).

Similarly, the pixel scale for AU2.M2X is found to be 10.3918 arcsecond/pixel.

For finding the pixel for AU1.M1X and AU1.M2X, the same equations (16-18) are used but the relation between the actuator position and the mirror angle is known for both M1x and M2X.

For AU1M1X, the 4th order polynomial relation between the X positioner of M1 and M1's tilt[21] was-

$$\theta_{M1X} = 1.39058e-6 \times s_{M1X}{}^4 + 2.81265e-6 \times s_{M1X}{}^3 + 1.25988e-04 \times s_{M1X}{}^2 + 9.29908e-01 \times s_{M1X} - 1.1683e+01 \quad (19)$$

For AU1M2X, the 4th order polynomial relation between the X positioner of M2 and M2's tilt was-

[21] The relation between θ_{M1X} and s_{M1X} for AU1.M1X and θ_{M2X} and s_{M2X} for AU1.M2X was provided by Miss C. ONG

$$\theta_{M2X} = 1.508692e-6 \times s_{M2X}{}^{4} + 4.099467e-6 \times s_{M2X}{}^{3} + 1.24998e-04 \times s_{M2X}{}^{2} + 9.35911e-01 \times s_{M2X} - 1.17301e+01 \quad (20)$$

So in a similar manner (4th point of the method on page 45) after equating the value of θ_{M1X} and θ_{M2X} in equation 16 and 17 for AU1.M1X and AU1.M2X, the pixel scale are 4.7956 arcsecond/pixel and 13.7931 arcsecond/pixel respectively.

5. So for both AU1M1M2X and AU2M1M2X, their pixel scale value is then multiplied with their corresponding backlash value in pixels ($\Delta P_x(n)$), which gives the backlash value in arcsecond.

Note: It's clear from the figure 8 and 9 that the distance between M2 and the WFS is small for AU1 (80 mm) and large for AU2 (110 mm), hence it's obvious to have large pixel scale value for AU1 as there are less number of pixels to be covered because of the smaller distance between AU1M2 and WFS.

In the similar manner the backlash values for AU1.M1X, AU1.M2X and AU2.M2X is calculated and the results are compared.

How to read the Gaussian center of the spot?

The file '.fits' can be viewed by the astronomical imaging software 'DS9'. For finding the spot center in pixels or the FWHM of the Gaussian spot the inbuilt function 'GAUSS2DFIT' of the IDL is used in the following step-

1. Read the image by the READFITS() function of the IDL.
2. Store the pixels value of the image in a 2D array.
3. Smooth the image for removing the error pixel by using the function SMOOTH() of the IDL.
4. Find the index of the brightest pixel and crop the image around it covering at least 10 pixels at every edge around the brightest pixel. (The diameter of the Gaussian spot is around 5-7 pixel for our test)
5. Apply GAUSS2DFIT to the cropped image as it tries to best fit the Gaussian spot.
6. The result after the fitting gives the center of the FWHM of the spot.

IDL CODING REFERENCE

For AU2.M1X

Starting point = 11.50 mm
Ending point = 12.50 mm
Step increment = 0.1 mm
Total number of measurement points at which the backlash is measured, n = 11

For AU2.M2X

Starting point = 11.55 mm
Ending point = 12.55 mm
Step increment = 0.1 mm
Total number of measurement points at which the backlash is measured, n = 11

Coding 7 is for taking the backlash measurement for both AU2.M1X and AU2.M2X at the optical bench by using the acquisition camera for LOWFS and the CCD.

Coding 8 is for calculating the pixel scale, reading the Gaussian center of the laser spot from the backlash measured '.fits' file and performing the operation for finding the backlash value in arcsecond at all the 11 positions of the actuator for both AU2.M1X and AU2.M2X.

For AU1.M1X

Starting point = 6.5 mm
Ending point = 7.5 mm
Step increment = 0.1 mm
Total number of measurement points at which the backlash is measured, n = 11

For AU1.M2X

Starting point = 6.7 mm
Ending point = 7.7 mm
Step increment = 0.1 mm
Total number of measurement points at which the backlash is measured, n = 11

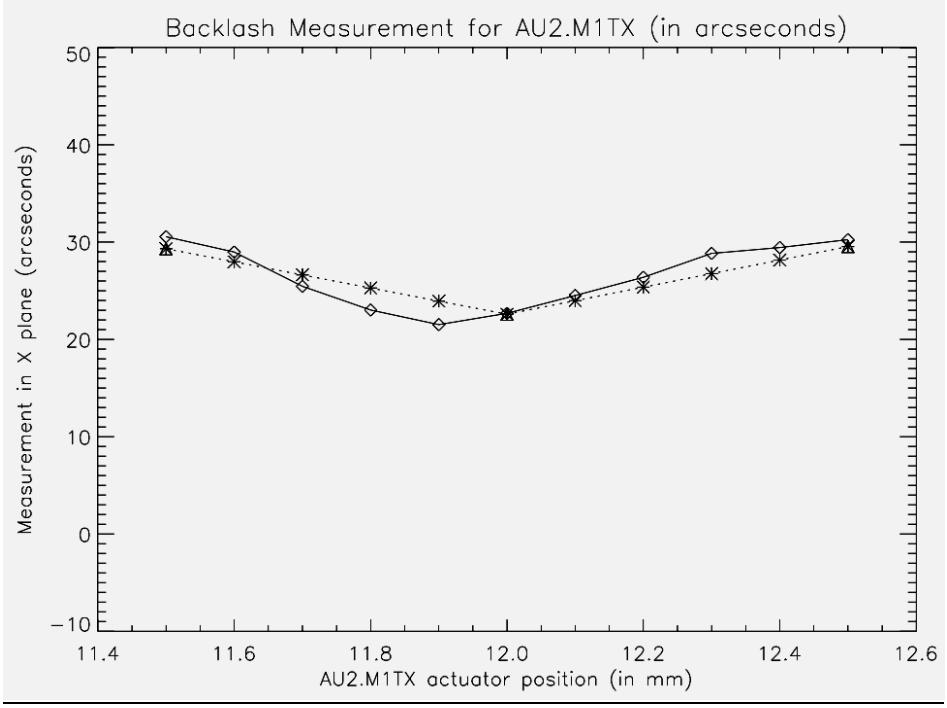
Coding 9 is for taking the backlash measurement for both AU1.M1X and AU2.M2X at the optical bench by using the acquisition camera for HOWFS and the CCD.

Coding 10 is for calculating the pixel scale, reading the Gaussian center of the laser spots from the backlash measured '.fits' file and performing the operation for finding the backlash value in arcsecond at all the 11 positions of the actuator for both AU1.M1X and AU1.M2X.

RESULTS

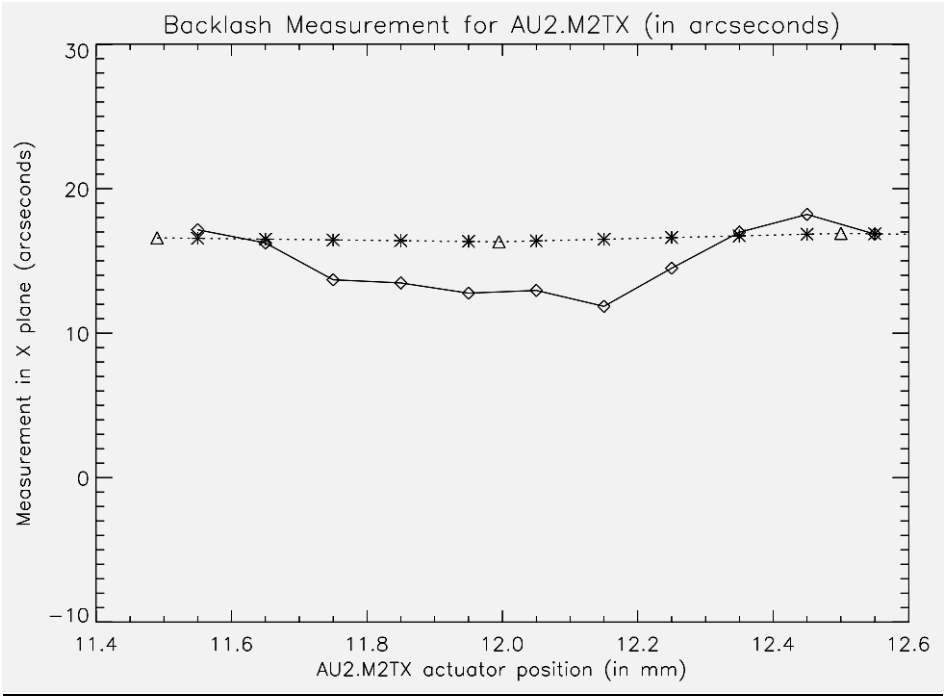
Plot 11

Plot 11 compares the backlash value for AU2.M1X measured at the observation floor (dotted line with the symbol 'Δ') and at the optical bench (thick line with the symbol '♦'). The plot with the symbol '*' shows the interpolated value of the backlash measurement done at the observation floor



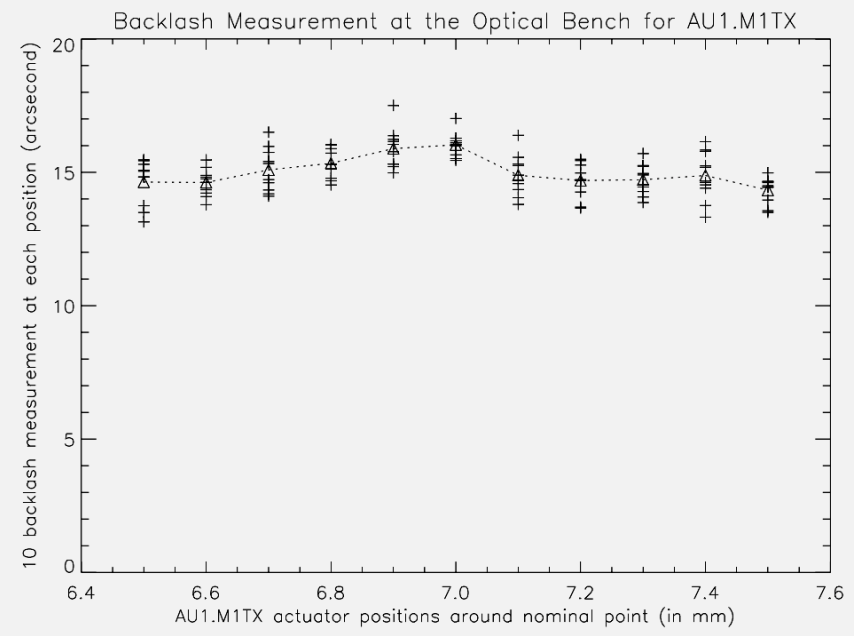
Plot 12

Plot 12 compares the backlash value for AU2.M2X measured at the observation floor (dotted line with the symbol 'Δ') and at the optical bench (thick line with the symbol '♦'). The plot with the symbol '*' shows the interpolated value of the backlash measurement done at the observation floor



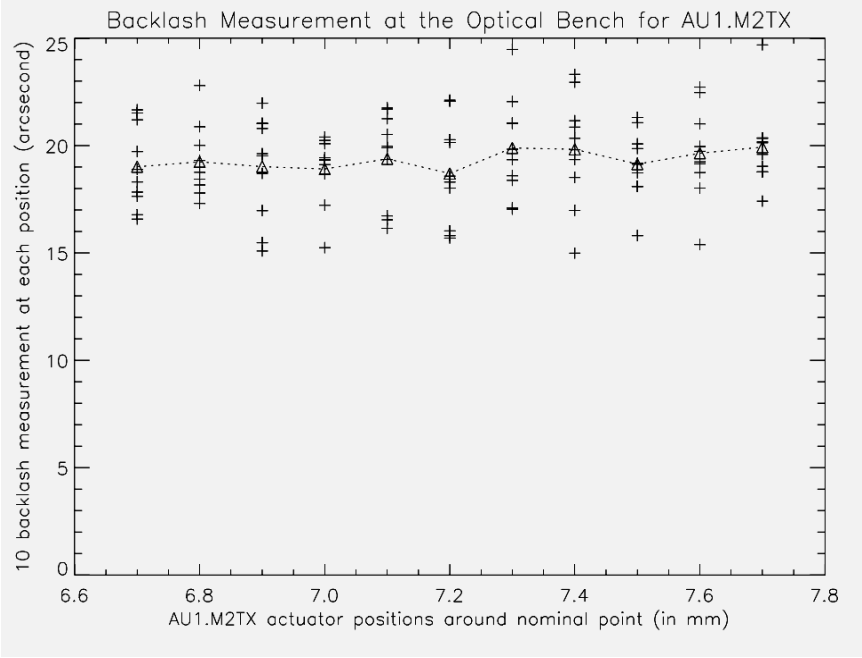
Plot 13

Plot 13 showing the 10 measurements of the backlash (symbol '+') with their average (symbol 'Δ') done at every position for AU1.M1TX in arcsecond



Plot 14

Plot 14 showing the 10 measurements of the backlash (symbol '+') with their average (symbol 'Δ') done at every position for AU1.M2TX in arcsecond



The table below shows the difference in the backlash measurement for AU2.M1X and AU2.M2X done at the observation floor and at the optical bench-

Table 8

AU2.M1X actuator position (mm)	Difference between the backlash value measured at the observation floor to that of the optical bench for AU2.M1X (arcsecond)	AU2.M2X actuator position (mm)	Difference between the backlash value measured at the observation floor to that of the optical bench for AU2.M2X (arcsecond)
11.50	-1.23450	11.55	-0.59349
11.60	-1.00888	11.65	0.26691
11.70	1.17381	11.75	2.76238
11.80	2.26162	11.85	2.92808
11.90	2.41551	11.95	3.58120
12.0	-0.08966	12.05	3.43073
12.1	-0.52971	12.15	4.64933
12.2	-1.01561	12.25	2.12281
12.3	-2.07654	12.35	-0.25096
12.4	-1.28635	12.45	-1.37276
12.5	-0.710054	12.55	0.018049

CONCLUSION FOR TASK 3

Table 8 calculates the difference in the value of the backlash at the optical bench and at the observational floor for AU2. As we have taken less number of the measurement points for the backlash calculation at the optical bench, for their comparison with the values at the observational floor, we compared them with the interpolated backlash value. We observed that the error in the backlash measurement was around 2-3 arcsecond for M1X and around 3-4 arcsecond for M2X. The reason for this difference can be described as follows-

1. **Fitting model error:** For the backlash measurement at the optical bench, we measured it for very less number of points, which can be considered as linear but at the observational floor the relation between the tilt of the mirror and the actuator position varies as the 4^{th} order polynomial function so we can't compare the linear backlash value measured at the optical bench

with the non linear interpolated backlash value at the observational floor. But from the plot 11, we see clearly that the backlash values measured at the observational floor and at the optical bench can be compared at the 1st, 6th and 11th position (dotted data in the table 8) and there we found the error of 0.5 pixel. As the pixel scale of the CCD camera is 20mas/pixel, so the error of 0.5 pixel corresponds to the error of 10mas which gives the accuracy within the specification.

2. **Setup error:** The error can be introduced because of the different setup as for the backlash measurement at the observational floor, we used the ACT (which has a measurement error of 1 arcsecond) and for the optical bench we used the CCD. So the measurement error in reading the data is one of the reason for the difference in the backlash measurement

TASK 4

To develop the backlash compensation algorithm and applying it to all the 8 axes/actuators of AU1 and AU2 and confirming the data again by taking the measurements for both AU1 and AU2[22] at the optical bench

INTRODUCTION

So far we have calculated the backlash value in arcsecond for the 4 actuators (per AU) of both AU1 and AU2 at the optical bench. If an actuator is moving in one particular direction (either forward or backward), there will be no backlash, as soon as it reverses its direction, there is a backlash (fig 25) which needs to be compensated because while tracking when the mirror of the AU moves in order to follow the focus, the actuator's motion can be forward or backward depending on the trajectory. So during the execution of the trajectory when the actuator reverses its direction, it is necessary to compensate the backlash during the real time observation. So we developed an algorithm for the compensation and applied it to all the actuators of AU1 and AU2.

DEVICES at the AO OPTICAL BENCH

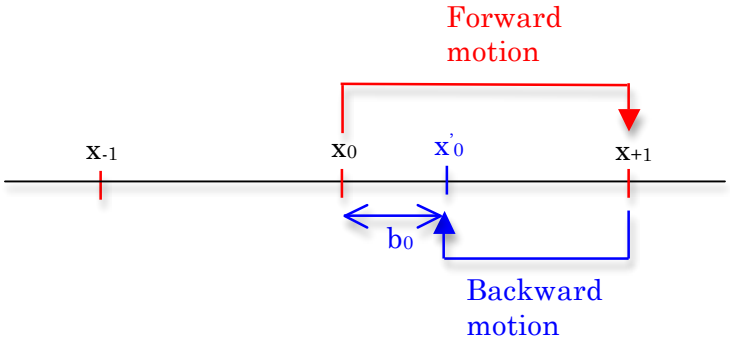
1. Guide star acquisition unit 1 (AU1) with acquisition camera for HOWFS
2. Guide star acquisition unit 2 (AU2) with acquisition camera for LOWFS
3. 8 NEWPORT LTA-HL linear actuators
4. AO calibration light source
5. CCD camera
6. 2 XPS controller for HOWFS and LOWFS respectively

METHOD

Backlash compensation algorithm

Figure 25

Suppose 'x_0' is the position of the actuator under observation, 'x_{-1}' is the position when actuator moves backward and 'x_{+1}' is the position when the actuator moves forward and 'b_0' is the backlash at 'x_0' as shown in the figure below-



[22] For the focus tracking, when the elevation changes only the X positioner of the mirror of AU needs to be tilted, the Y positioner more or less stays at the same position so for the simplicity in our experiment we will concentrate more on X positioners for both AU1 and AU2 as our final goal is the focus tracking.

Note: All the values taken here are in 'mm'.

1. When actuator moves forward i.e. from x_0 to x_{+1}, the command for the relative forward motion provided to the XPS controller is $\Delta x_f = (x_{+1} - x_0)$. The result from the XPS encoder is the actual position of the actuator, which is x_{+1} after the motion.

2. When the actuator reverses its direction i.e. when it comes back at the starting position then because of the backlash the XPS encoder value i.e. the actual position of the actuator will be x'_0 instead of x_0 after the motion.

3. So for the compensation of the backlash 'b_0' which exists at the position x_0 the relative command that should be provided to the XPS during the backward motion from x_{+1} to x_0 is $\Delta x_b = -(x_{+1} - x_0) - b_0$

4. And for moving again in the forward direction from x_0 to x_{+1}, the relative command should be $\Delta x_f = (x_{+1} - x_0) + b_0$

Calculation of the backlash value 'b_0' in terms of the actuator position

So far as we have calculated the backlash value for every actuator in 'arcsecond', for the compensation of the backlash, the corresponding value of the backlash in terms of the actuator position is necessary to calculate because the compensation algorithm can be applied only when the backlash value which to be compensated is known in 'mm'.

For AU2.M1X and AU2.M2X[23]

Let 'x' is the actuator position in mm, and θ is the corresponding mirror angle. As we have taken very small number of points for our experiment, we can consider the relation between x and θ to be linear[24]

So we can write the linear equation as-

$$\theta = (b \times x) + a \text{ , where b is the slope and a is the constant} \quad (21)$$

So for AU2.s_{M1X} (11 positions of the actuator AU2.M1X for which the backlash value is known in arcsecond), the corresponding θ_{M1} can be estimated as-

$$\theta_{M1} = (b \times AU2.s_{M1X}) + a \quad (22)$$

[23] The procedure for finding the backlash value in 'mm' is same for AU2.M1X and AU2.M2X

[24] As the relation between the mirror angle and the actuator position for AU2.M1X and AU2.M2X was not known (as already mentioned on page 51), the same value of x and θ has been taken that were used earlier for the calculation of the pixel scale for AU2.M1X and AU2.M2X.

And, backlash (degree)=backlash value calculated at the optical bench (arcsecond) × 3600

So, the corresponding backlash value to be compensated in mm is,

$$b_{AU1.sMX} = \frac{backlash(\deg ree)}{b} \quad (23)$$

For AU1.M1X

The relation between the actuator position AU1.s_{M1X} and the mirror tilt θ_{M1X} is already known from the equation 19. As this relation is a 4th polynomial relation but for our experiment we can consider their relation as linear so the linear relation from the equation 19 can be deduced as-

$$\theta_{M1X} = (9.29908e-01 \times AU1.s_{M1X}) - 1.1683e+01 \quad (24)$$

Note: For the equation 19, the nominal position is not same as the one we have considered in our experiment so the offset value should be subtracted from the equation 24.

$offset_{M1X} = (9.29908e-01 \times AU1_{M1X0}) - 1.1683e+01$, where $AU1_{M1X0}$ is the nominal position for M1's X positioner of AU1 (25)

$$\theta_{M1X} = (9.29908e-01 \times AU1.s_{M1X}) - 1.1683e+01 - offset_{M1X} \quad (26)$$

So, the slope can be found as-

$$\theta_{M1X} = (b_{M1X} \times AU1.s_{M1X}) + a_{M1X} \quad (27)$$

Hence, the corresponding backlash value to be compensated in 'mm' for AU1.M1X is,

$$b_{AU1.sM1X} = \frac{backlash(arc\sec ond) \times 3600}{b_{M1X}} \quad (28)$$

For AU1.M2X

Similarly, the linear relation between AU1.s_{M2X} and the mirror tilt θ_{M2X} from the equation 20 can be deduced as-

$$\theta_{M2X} = 9.35911e-01 \times AU1.s_{M2X} - 1.17301e+01 \quad (29)$$

$offset_{M2X} = 9.35911e-01 \times AU1_{M2X0} - 1.17301e+01$, where $AU1_{M2X0}$ is the nominal position for M2's X positioner of AU1 (30)

$$\theta_{M2X} = 9.35911e-01 \times AU1.s_{M2X} - 1.17301e+01 - offset_{M2X} \quad (31)$$

So, the slope can be found as-

$$\theta_{M2X} = (b_{M2X} \times AU1.s_{M2X}) + a_{M12} \quad (32)$$

Hence, the corresponding backlash value to be compensated in 'mm' for AU1.M2X is,

$$b_{AU1.sM2X} = \frac{backlash(arc\,second) \times 3600}{b_{M2X}} \quad (33)$$

Hence for the compensation of the backlash for AU1 and AU2's actuator in X plane, the 'corresponding backlash compensation value in 'mm' should be subtracted whenever there is a reverse motion.

IDL CODING REFERENCE

Coding 11 is to create the backlash compensation algorithm for AU1.M1X and AU1.M2X

Coding 12 is for reading the Gaussian center of the laser spots after the compensation of the backlash from '.fits' file and performing the operation for finding the residual value in arcsecond at all the 11 positions of the actuator for both AU1.M1X and AU1.M2X.

Coding 13 is to create the backlash compensation algorithm for AU2.M1X and AU2.M2X

Coding 14 is for reading the Gaussian center of the laser spots after the compensation of the backlash from '.fits' file and performing the operation for finding the residual value in arcsecond at all the 11 positions of the actuator for both AU2.M1X and AU1.M2X.

RESULTS

Table 9: Backlash for AU2.M1X at the optical bench

pixel_scale = 4.2975 arcsecond/pixel
nominal position=12.00 mm

Measurement Points	Backlash (arcsecond)	Standard Deviation (± arcsecond)	Backlash (Pixel)	Standard Deviation (± pixel)
11.50	30.5495	0.483754	7.108899	0.112570
11.60	28.9775	0.802793	6.743082	0.186811
11.70	25.4483	0.526598	5.921854	0.122540
11.80	23.0141	0.533559	5.355404	0.124160
11.90	21.5138	0.813177	5.006283	0.189227
12.0	22.6725	0.508365	5.275921	0.118297
12.1	24.5039	0.470569	5.702073	0.109502
12.2	26.3810	0.833246	6.138892	0.193897

12.3	28.8332	0.605018	6.709524	0.140788
12.4	29.4343	0.974397	6.849394	0.226743
12.5	30.2493	0.738318	7.039042	0.171807

Plot 15

Plot 15 shows the 10 times backlash measurement done (symbol '+') at each position with their average value (symbol 'Δ') for AU2.M1X in arcseconds

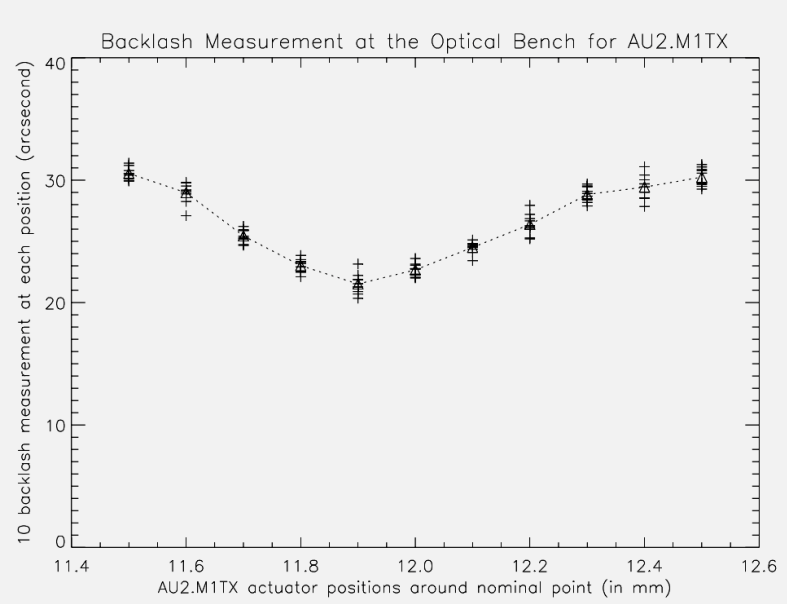


Table 10: Backlash Compensation done for AU2.M1X at the optical bench

Measurement Points	Backlash to be compensated (mm)	Signed compensated Backlash (arcsecond)	Standard Deviation (± arcsecond)	Signed compensated Backlash (Pixel)	Standard Deviation (± pixel)
11.50	0.010051	1.01901	0.58786	0.23713	0.13679
11.60	0.009534	0.42173	0.96097	0.09814	0.22362
11.70	0.008373	-0.51862	1.09894	-0.12068	0.25572
11.80	0.007572	-0.53302	0.90909	-0.12403	0.21155
11.90	0.007078	-0.24288	0.56627	-0.05652	0.13177
12.0	0.007459	0.51563	0.74554	0.11999	0.17349
12.1	0.008062	-1.21390	0.91384	-0.28247	0.21265
12.2	0.008679	0.23256	1.03956	0.05412	0.24191
12.3	0.009487	-0.96355	0.45036	-0.22422	0.10479
12.4	0.009684	-1.25780	0.50609	-0.29269	0.11777
12.5	0.009953	-0.09431	0.96679	-0.02195	0.22497

Plot 16

Plot 16 shows the 10 times compensated backlash measurement (symbol '+') done at each position with their average value (symbol 'Δ') for AU2.M1X in arcseconds

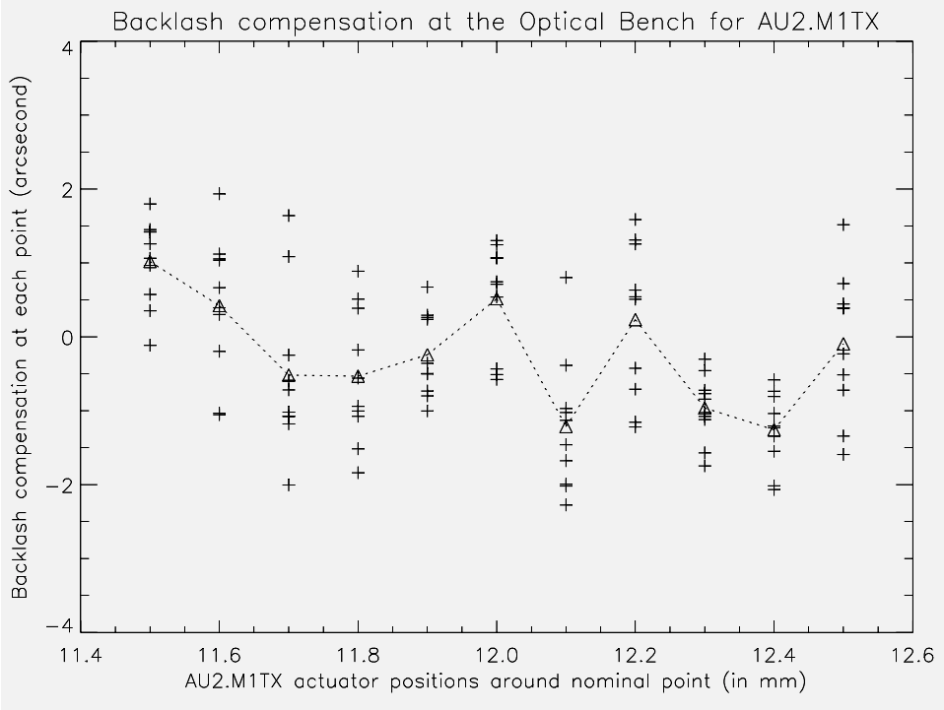


Table 11: Backlash for AU2.M2X at the optical bench

pixel_scale = 10.39179 arcsecond/pixel
nominal position=11.75 mm

Measurement Points	Backlash (arcsecond)	Standard Deviation (± arcsecond)	Backlash (Pixel)	Standard Deviation (± pixel)
11.55	17.1536	1.632426	1.650691	0.157088
11.65	16.2394	0.705733	1.562717	0.0679126
11.75	13.6901	1.384097	1.317399	0.133191
11.85	13.4706	1.581043	1.296276	0.152143
11.95	12.7637	1.220173	1.228248	0.117417
12.05	12.9533	1.327620	1.246494	0.127757
12.15	11.8499	1.660597	1.140311	0.159799
12.25	14.4915	1.730835	1.394519	0.166558
12.35	16.9805	1.724974	1.634029	0.165994
12.45	18.2174	1.125351	1.753062	0.108292
12.55	16.8544	1.969643	1.621899	0.189538

Plot 17

Plot 17 shows the 10 times backlash measurement done (symbol '+') at each position with their average value (symbol 'Δ') for AU2.M2X in arcseconds

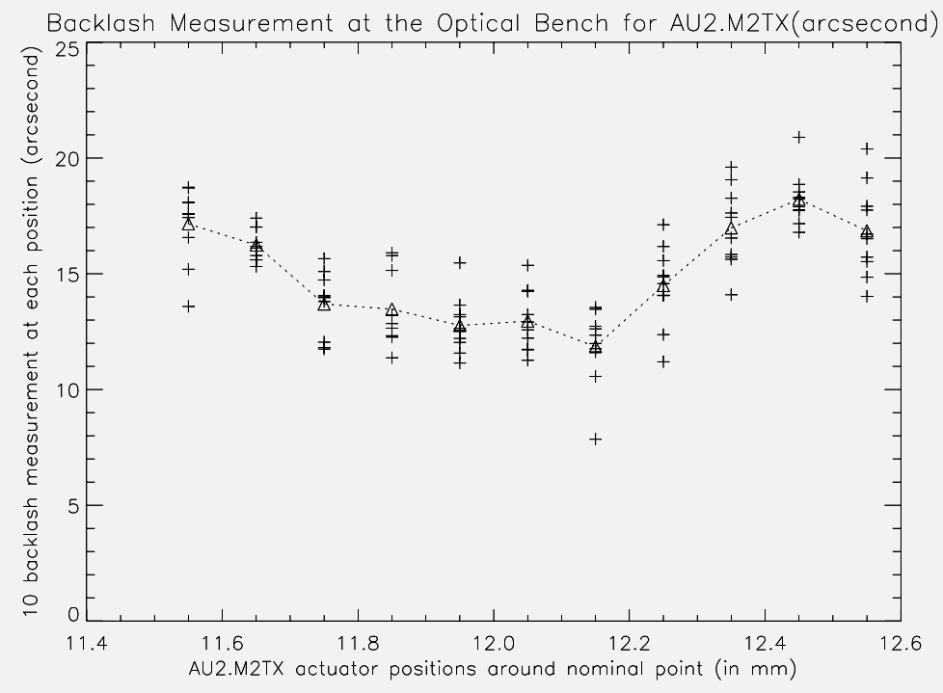


Table 12: Backlash Compensation done for AU2.M2X at the optical bench

Measurement Points	Backlash to be compensated (mm)	Signed compensated Backlash (arcsecond)	Standard Deviation (± arcsecond)	Signed compensated Backlash (Pixel)	Standard Deviation (± pixel)
11.55	0.005644	1.16067	2.51807	0.1116915	0.242314
11.65	0.005343	1.16555	2.28104	0.1121609	0.219504
11.75	0.004504	-0.04842	2.04577	-0.0046597	0.196864
11.85	0.004432	-1.84995	1.90855	-0.1780201	0.183659
11.95	0.004199	1.45209	2.30003	0.1397348	0.221331
12.05	0.004262	0.02898	2.02937	0.0027891	0.195286
12.15	0.003899	0.51647	2.34566	0.0496994	0.225723
12.25	0.004768	2.13387	2.18997	0.2053423	0.210740
12.35	0.005587	-0.41142	1.98515	-0.0395909	0.191030
12.45	0.005993	-0.97656	1.48154	-0.0939740	0.142568
12.55	0.005545	0.77137	2.24459	0.0742286	0.215997

Plot 18

Plot 18 shows the 10 times compensated backlash measurement (symbol '+') done at each position with their average value (symbol 'Δ') for AU2.M2X in arcseconds

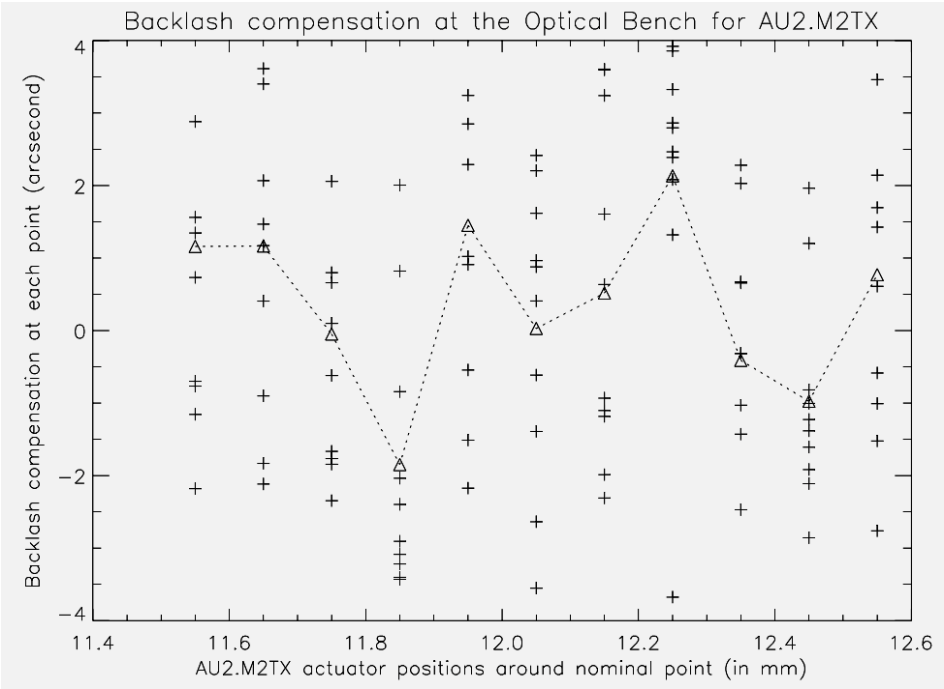


Table 13: Backlash for AU1.M1X at the optical bench

pixel_scale = 4.79556 arcsecond/pixel
nominal position=7.11236 mm

Measurement Points	Backlash (arcsecond)	Standard Deviation (± arcsecond)	Backlash (Pixel)	Standard Deviation (± pixel)
6.5	14.6354	0.852244	3.05186	0.17772
6.6	14.6180	0.513719	3.04824	0.10713
6.7	15.0922	0.820959	3.14712	0.17119
6.8	15.3401	0.565985	3.19881	0.11802
6.9	15.8904	0.747426	3.31357	0.15586
7.0	16.0305	0.449319	3.34278	0.09369
7.1	14.8890	0.769499	3.10475	0.16046
7.2	14.6924	0.654500	3.06376	0.13648
7.3	14.7249	0.581764	3.07053	0.12131
7.4	14.8816	0.924015	3.10319	0.19268
7.5	14.3310	0.493104	2.98838	0.10283

Table 14: Backlash Compensation done for AU1.M1X at the optical bench

Measurement Points	Backlash to be compensated (in mm)	Signed compensated Backlash (arcsecond)	Standard Deviation (± arcsecond)	Signed compensated Backlash (pixel)	Standard Deviation (± pixel)
6.5	0.00435266	-0.863884	1.05458	-0.180142	0.219907
6.6	0.00434750	-0.086742	1.31629	-0.018088	0.274481
6.7	0.00448851	0.073255	0.92193	0.015275	0.192246
6.8	0.00456225	-0.193484	0.96952	-0.040347	0.202170
6.9	0.00472591	-0.478097	1.16217	-0.099696	0.242343
7.0	0.00476757	0.980902	1.11562	0.204543	0.232636
7.1	0.00442809	-0.053634	1.18652	-0.011184	0.247419
7.2	0.00436962	0.239378	0.80945	0.049917	0.168792
7.3	0.00437928	-0.160976	1.18216	-0.033568	0.246511
7.4	0.00442588	-0.202072	1.28653	-0.042137	0.268275
7.5	0.00426212	-0.445290	1.02236	-0.092855	0.213189

Plot 19

Plot 19 shows the 10 times compensated backlash measurement (symbol '+') done at each position with their average value (symbol 'Δ') for AU1.M1X in arcseconds

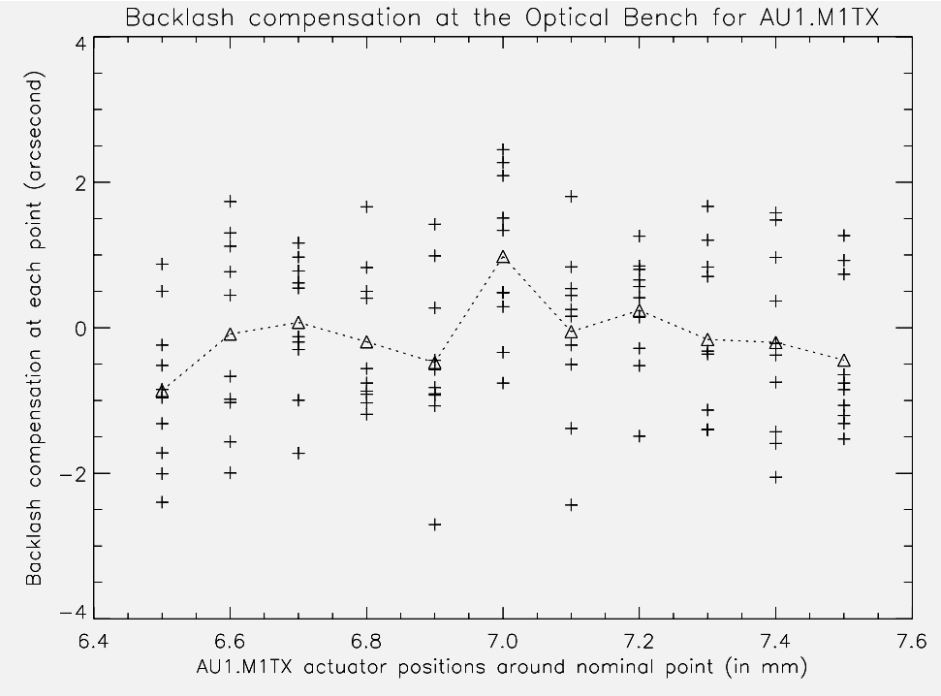


Table 15: Backlash for AU1.M2X at the optical bench

Pixel_scale=13.79302 arcsecond/pixel
Nominal position= 7.29337 mm

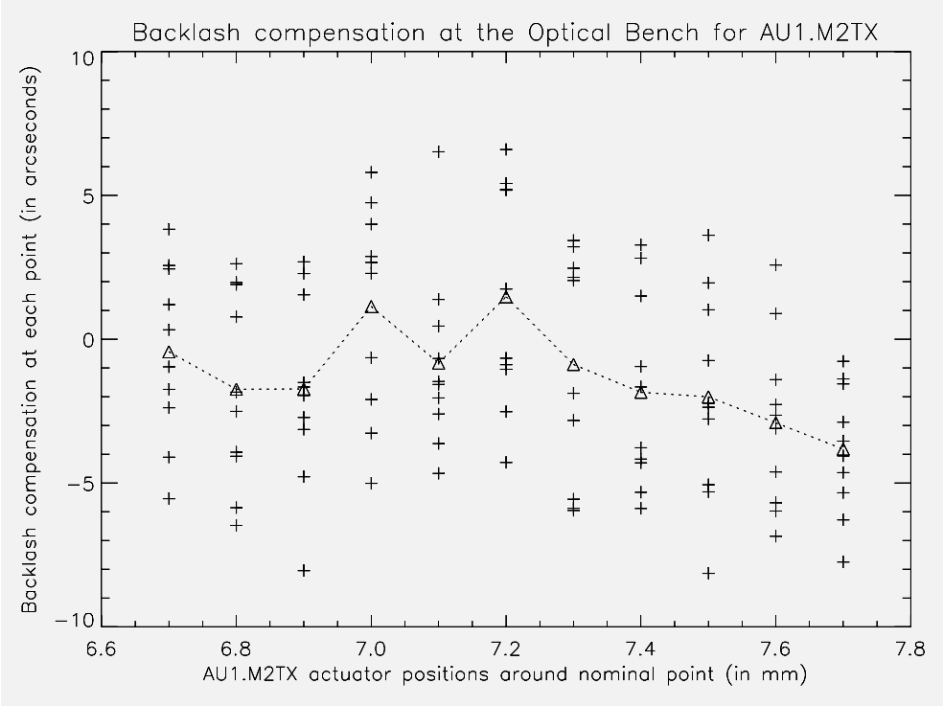
Measurement Points	Backlash (arcsecond)	Standard Deviation (± arcsecond)	Backlash (Pixel)	Standard Deviation (± pixel)
6.7	19.0119	1.92467	1.37837	0.139539
6.8	19.2437	1.62956	1.39518	0.118144
6.9	19.0199	2.42543	1.37896	0.175845
7.0	18.9078	1.57732	1.37083	0.114356
7.1	19.3837	2.15873	1.40533	0.156509
7.2	18.7010	2.42085	1.35583	0.175513
7.3	19.8880	2.32303	1.44189	0.168421
7.4	19.8258	2.55016	1.43738	0.184888
7.5	19.1340	1.60943	1.38723	0.116684
7.6	19.6434	2.14537	1.42416	0.155540
7.7	19.9276	1.91614	1.44476	0.138921

Table16: Backlash Compensation done for AU1.M2X at the optical bench

Measurement Points	Backlash to be compensated (in mm)	Signed compensated Backlash (arcsecond)	Standard Deviation (± arcsec)	Signed compensated Backlash (pixel)	Standard Deviation (± pixel)
6.7	0.0056318	-0.44021	3.0569	-0.031915	0.221623
6.8	0.0057006	-1.74347	3.3781	-0.126402	0.244911
6.9	0.0056342	-1.73220	3.3119	-0.125585	0.240117
7.0	0.0056011	1.13395	3.6594	0.082212	0.265311
7.1	0.0057420	-0.83259	3.1399	-0.060363	0.227649
7.2	0.0055398	1.47268	3.8762	0.106770	0.281029
7.3	0.0058914	-0.88244	3.9694	-0.063977	0.287782
7.4	0.0058729	-1.84951	3.3933	-0.134090	0.246018
7.5	0.0056681	-2.00576	3.6023	-0.145418	0.261172
7.6	0.0058189	-2.91120	3.0359	-0.211063	0.220109
7.7	0.0059031	-3.82341	2.2588	-0.277199	0.163765

Plot 20

Plot 20 shows the 10 times compensated backlash measurement (symbol '+') done at each position with their average value (symbol 'Δ') for AU1.M2X in arcseconds



CONCLUSION FOR TASK 4

In this task, we successfully compensated the backlash for all the actuators for both AU1 and AU2 at the optical bench.

For AU1's M1X and M2X

From the Specification Table (table 7), the repeatability should be 1.1 arcsecond for M1X and 4.2 arcsecond for M2X.

After the backlash compensation of AU1.M1X, we can conclude from the table 14 that the 'signed compensated backlash values' for the actuator positions under observation is within 1.1 arcsecond.

For AU1.M2X, from the table 16 the 'signed compensated backlash values' for the actuator positions under observation is within 4.2 arcsecond.

Hence we can say that the model for finding the backlash value and its compensation is within the expectation of the specification of AU1 and can be used for calculating the backlash value at the other positions of the actuator as well.

For AU2's M1X and M2X

From the Specification Table (table 7), the repeatability should be 1.2 arcsecond for M1X and 3.0 arcsecond for M2X.

After the backlash compensation of AU2.M1X, we can conclude from the table 10 that the 'signed compensated backlash values' for the actuator positions under observation is around 1.2 arcsecond.

For AU2.M2X, from the table 12 the 'signed compensated backlash values' for the actuator positions under observation is within 3.0 arcsecond.

Hence also, we can say that the model for finding the backlash value and its compensation is within the expectation of the specification of AU2 and can be used for calculating the backlash value at the other positions of the actuator desired by the user.

For moving on to the Section of the Focus tracking, we can move the actuators of the AU1 in forward and in backward direction and can apply the compensation of the backlash when required to the 4 actuators of AU1. In this way, we can define the trajectory motion for the tracking of the focus in the real time and we can test the accuracy of the trajectory for tracking the focus or aligning it back to the optical axis of the HOWFS when it get defocused.

12.2) FOCUS TRACKING

TASK 5

To understand the trajectory motion of the prototype (task 1) with the actuator X and Y by using Line-Arc trajectory mode of the XPS controller in XY plane

INTRODUCTION

When the focus of the LGS changes because of the change in the elevation and the zenith angle, the GSAU must move its mirror in the same way or in the same trajectory in order to follow the focus and to align the ray within the FOV of the WFS. So, first we considered the same prototype of AU1, which was used earlier in task 1 and we tried to understand and test the motion/trajectory of the X and Y actuator of M2 of the AU1 prototype by using the inbuilt module of the XPS controller for the trajectory motion in 2D, which is a 'line-arc trajectory' motion.

DEVICES

The devices used are the same as used in Task 1. One more positioner/ linear actuator LTA-HL is installed at the 2nd axes of the XPS controller for the motion in Y direction.

METHOD

What is the trajectory? [25]

Trajectory is defined as a continuous multidimensional motion path that maintains a constant speed throughout the entire path except the acceleration and deceleration periods. It is defined by the user and for its execution in XY plane, a number of straight (line) and curved (arc) segments need to be specified such as-

Trajectory element (segment) - is the geometric shape either a Line (X, Y) or an Arc (Radius, Sweep Angle). The angles are measured in degrees and the positive angles are measured counterclockwise.

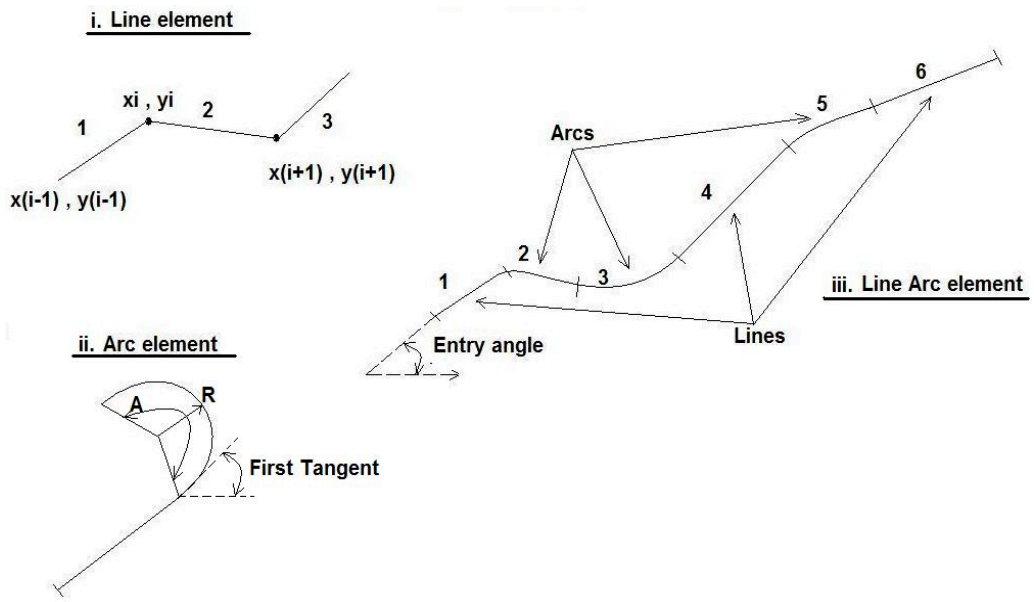
Trajectory velocity - is the tangential linear velocity during its execution

Trajectory acceleration - is the tangential linear acceleration used to start and end the trajectory.

[25] Chapter 9 of the User's manual software tools tutorial of the XPS Controller

A line arc trajectory is a set of consecutive line or arc segments. The line segments are true linear interpolations y = A*x + B, the arc segments are true arcs of circles $(x-x0)^2 + (y-y0)^2 = R^2$

Figure 26



In the **'Line element'**, there are several consecutive line segments that define the trajectory. The ending point of one segment is the starting point for the next consecutive segment. The line arc trajectory is defined by consecutive line and arc segments as shown is fig 26(iii).
As the segment's tangential angles around the connection point of any two consecutive segments may not be continuous, there might be some velocity discontinuities from one segment to the next, so to avoid this, every trajectory have a first element entry angle called as **'FirstTangent'** that is defined in the head of the trajectory data file (it has no effect if the first segment is a line and if its an arc then it is the tangent to the first point of the arc).

Also the user **defines the discontinuity angle**, which is the maximum allowed angle of discontinuity between any two segments having some angle difference.

Note: If the discontinuity angle is very large then during its execution, because of the instantaneous velocity change on both axes, it can produce larger acceleration that can finally result in a shock to the stages.

So to define the line arc trajectory file, certain notation should be followed-

The First line should be the "FirstTangent": "value in degree"
The second line should be the "DiscontinuityAngle": 'value in degrees'
Consecutive line and arc segments can de defined in a following way-
Line=X_i,Y_i
Line=X_{i+1} ,Y_{i+1}

Arc=Radius, SweepAngle etc.

After creating the trajectory file ('.trj'), it is sent to the XPS controller's memory under ...public/trajectories/
There are some predefined functions that can do the precheck and verify the possibility of the execution of the trajectory.

For example,

Line arc trajectory as $Y=X^2$

For creating the $Y=X^2$ trajectory, lets consider some initial theoretical values for X positioner in degree as θ_x. As $Y=X^2$, so $\theta_y = \theta_x^2$.

From equation (2), as we know the modeled function for the conversion of X Positioner's shift in mm to its corresponding value in radians and vice versa, we can get the values in millimeters for θ_x as –

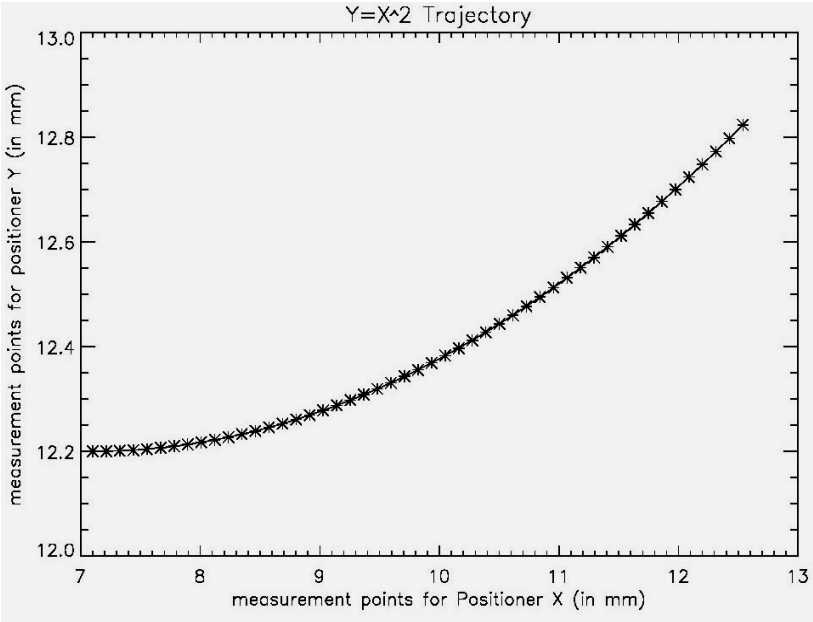
$$xx = a1 \times \sin(\theta_x) - \sqrt{b1^2 - a1^2 \times (1 - \cos(\theta_x))^2} + b1 + sx(0) \quad (37)$$

Similarly for the sake of simplicity, consider again the model function (equation 2) for the conversion of Y Positioner's value from radians to mm-

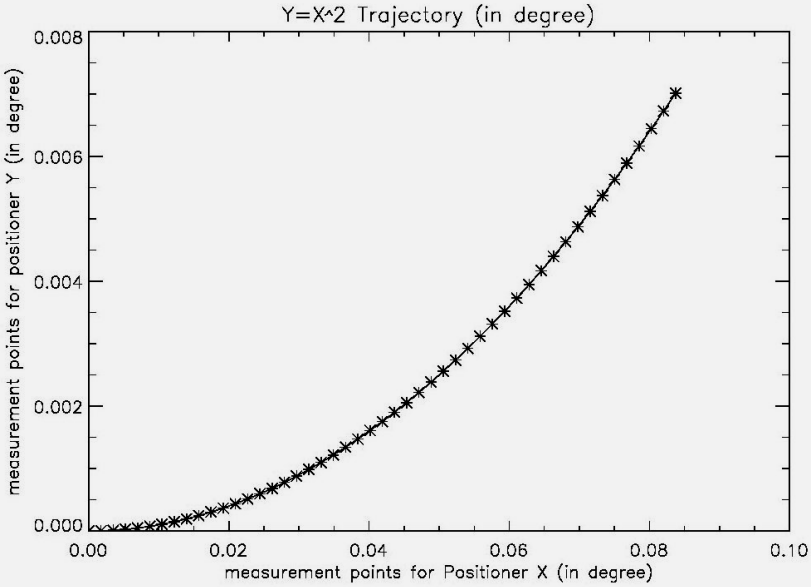
$$yy = a2 \times \sin(\theta_y) - \sqrt{b2^2 - a2^2 \times (1 - \cos(\theta_y))^2} + b2 + sy(0) \quad (38)$$

Following two plots shows the theoretical trajectory for X and Y Positioners in millimeter and in degree respectively in XY plane.

Plot 21



Plot 22



xx(n) and yy(n) are the theoretical measurement points (segments) at which the Positioner X and Y can move in a trajectory. For checking the motion of the trajectory we need to take simultaneous measurements from ACT and for that the smallest step should be find out at which the ACT can take the measurements when the trajectory is in the execution.

Technically the minimum resolution of the actuator prescribed by NEWPORT is 0.05 μm. Following steps has been done for finding the minimum resolution of the actuators in the prototype-

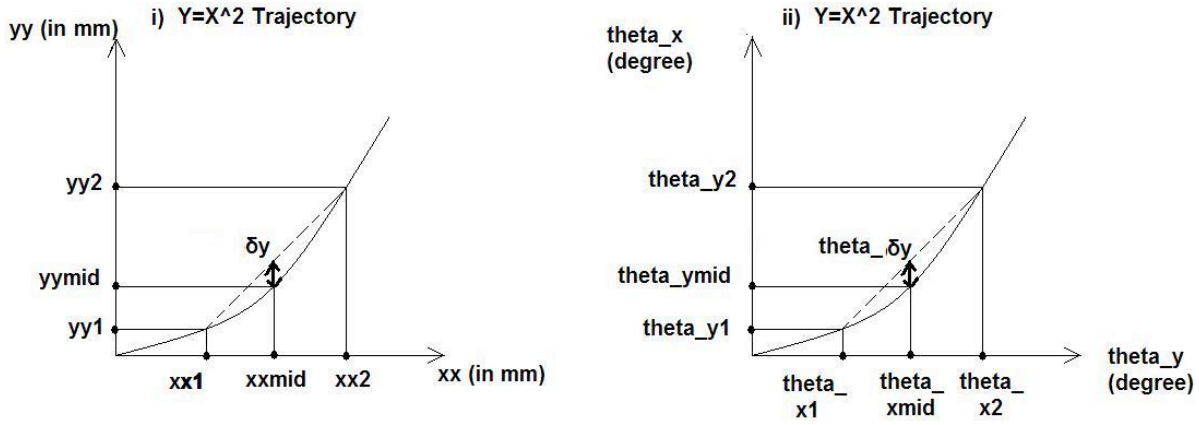
Vectors= xx(n) and yy(n)

As shown in fig 27 (i) –

$$xxmid(n) = \frac{xx(n) + xx(n+1)}{2} \tag{39}$$

$$\delta y(n) = \left(\frac{yy(n+1) - yy(n)}{xx(n+1) - xx(n)}\right) \times (xxmid(n) - xx(n)) + yy(n) - xxmid(n)^2 \tag{40}$$

Figure 27



We have guessed the minimum value for the step as 9 μm approximately. This step value in radians should give the value of θ_{xy} <= 1~2 arc seconds, if not, then the step value should be decreased. For checking this, as shown in fig 27 (ii), the following calculations is done-

Supposing 'n' as the number of measurement points for which the measurement can be taken within the range of ACT-

The corresponding value of the minimum step (9 μm) in radians is, step = $\frac{9e-6}{a1}$

$\theta_x(n)= \theta_x(0)+step*n$

$\theta_y(n)= \theta_x(n)^2$

$$\theta_{xmid}(n) = \frac{\theta_x(n) + \theta_x(n+1)}{2} \tag{41}$$

$$\theta_{\delta y}(n) = (\frac{\theta_y(n+1) - \theta_y(n)}{\theta_x(n+1) - \theta_x(n)}) \times (\theta_{xmid}(n) - \theta_x(n)) + \theta_y(n) - \theta_{xmid}(n)^2 \tag{42}$$

Experimentally, θ_{xy} value found to be less than 1 arc seconds, hence the guess for the minimum resolution of the actuator as 9 μm can be taken into consideration.

So consider the measurement points Δx and Δy (in mm) as the 'n' segments that can be created with a step value of 0.009 mm, that described the parameter LINE (ΔX_n , ΔY_n) of the trajectory file. The x and y values are responsible for the motion of the X and Y Positioners respectively. These values are not the absolute values of the actuator's shift, instead they are defined as relative to the actuator's current position (or trajectory starting point), that is, whatever is the current position, the trajectory consider the current position as the origin and starts executing according to the trajectory file saved in XPS controller's memory (as shown in fig 28).
For moving the X Positioner, the x values are-

Δx(n)= n * 0.009, the corresponding value of Δx in radians as 'θ_{xx}' can be find out by Newton Raphson method as described from equation (3) to (10).

For finding out the value of Δy(n)-
As, $\theta_{xy}(n)= \theta_{xx}(n)^2$, hence putting the value of the θ_{xy} in the equation 2 gives the value of Δy(n).

$$\Delta y(n) = a2 \times \sin(\theta_{\Delta y}(n)) - \sqrt{b2^2 - a2^2 \times (1 - \cos(\theta_{\Delta y}(n)))^2} + b2 \quad (43)$$

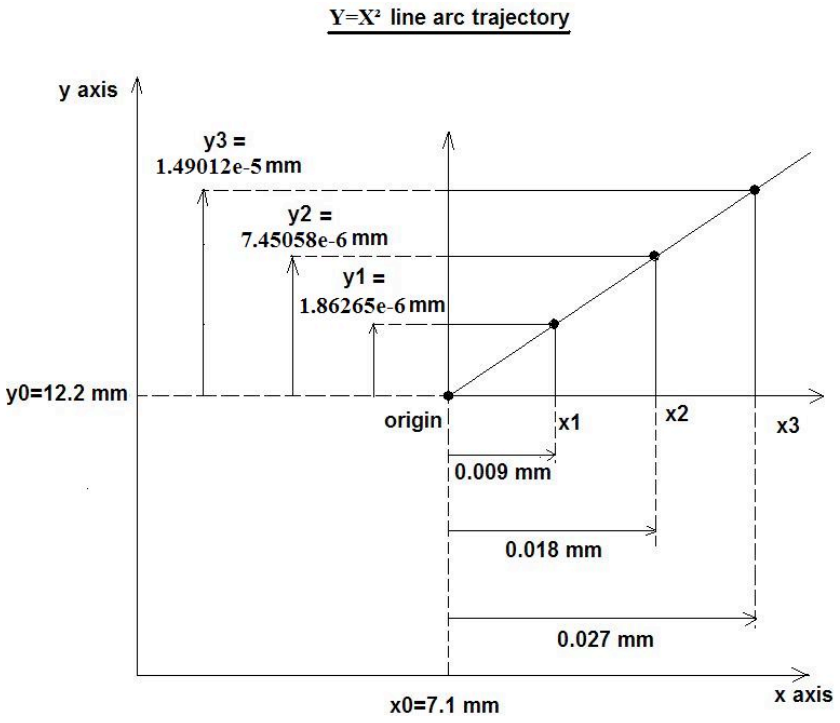
Hence the trajectory file for $Y=X^2$ trajectory (fig 28, showing only 3 line segments) is defined as-

FirstTangent=45;Degrees
DiscontinuityAngle=0.10;Degrees
Line=0.00900000,1.86265e-06
Line=0.0180000,7.45058e-06
Line=0.0270000,1.49012e-05 etc…

After executing the trajectory, the ACT measures the value in XY plane so the error in accuracy can be calculated as-

Accuracy error=$\Delta\theta_y(n)-\Delta\theta_x(n)^2$

Figure 28



Trajectory Verification and Execution

There are some predefined functions in XPS controller, which are used to precheck the maximum and minimum travel requirements, speed and acceleration per positioners before the execution of the trajectory.

XYLineArcVerification (name of the XY group, trajectory file name): This function checks the data of the trajectory file and returns 'OK' if the travel requirement per positioner like maximum possible velocity and acceleration are executable.

XYLineArcVerificationResultGet (groupname.positioner, get_NameOfTheFile, get_MaxPos, get_MinPos, get_Acceleration, get_Velocity): It returns the result of the above function like the name of the last file verified, the value of the maximum velocity if when applied at least one of the positioner will reach its maximum allowed speed (in our example, its 0.00475m/s), maximum possible acceleration ($0.1 m/s^2$).

XYLineArcExecution (name of the XY group, trajectory file name, velocity, acceleration, number of execution): it executes the trajectory with the provided value of velocity and acceleration.

During the simulation, the 'XYLineArcExecution() function via web interface' and the 'idl program for taking the measurements from ACT' were executed simultaneously so that the angle measurements in xy plane can be done simultaneously with the execution of the trajectory.

IDL CODING REFERENCE

Coding 15 is to take the measurements from ACT while actuators are in motion according to the trajectory defined in XPS controller memory.

Coding 16 is to find out the minimum resolution of the actuators by which they can move, that is to be defined in the trajectory file so that when actuators are moving, the measurements can be taken simultaneously at each minimum step.

Coding 17 is to define the line arc trajectory file

Coding 18 is for reading and processing of the measured data (after the trajectory execution).

Coding 19 is for creating numerical code for measuring the approximated values of angle (theta) by Newton-Raphson method for the trajectory

RESULT

After executing the trajectory file, we couldn't check the accuracy of the trajectory motion. Because we executed the trajectory file via web interface and we noted the position of the actuator by running idl command because of which the data taken by the ACT was not taken at the desired position i.e. we couldn't measure the value of the mirror angle at the relative commanded value sent to the trajectory file. While the trajectory was in execution, the value of the actual position returned by the idl command was not sampled at the equal interval.

TASK 6

By taking the AO calibration light source at the nominal position, developing the relation between the change in the focus with the changing position of the linear stage of the AU1 i.e. to find out for how much distance the 5 actuators[26] should move (along with the backlash compensation if required) for aligning the chief ray to the optical axis of the WFS and for tracking the change in the focus

INTRODUCTION

For tracking the focus of the LGS, we use only AU1 of the HOWFS. As the Subaru's LGSAO188 project is currently ongoing, we can't use the Laser. So, for our experiment we used the on-axis calibration light source for developing the relation between the change in the focus of the light spot, on the high-resolution camera, along the optical axis (at the nominal position) and the position of the linear stage (by changing the distance between M1 and M2). For tracking the focus, we created the trajectory file defining the motion of the 5 axes of AU1. We also did the backlash compensation when the 5 actuators reversed their motion while tracking.

DEVICES AT AO OPTICAL BENCH

1. Guide star acquisition unit 1 (AU1) with acquisition camera for HOWFS
2. 5 NEWPORT LTA-HL linear actuators
3. AO calibration light source
4. CCD camera
5. An XPS controller for HOWFS

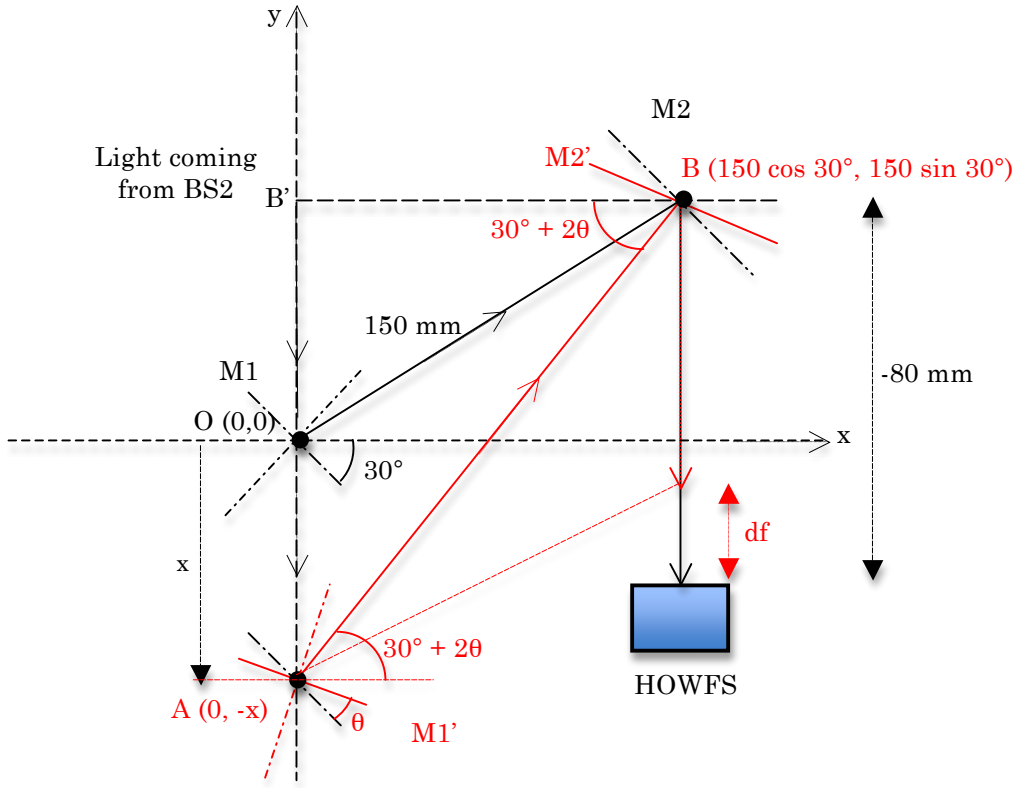
METHOD

The working of the GSAU AU1 is described in the figure 12 on page 20. In figure 12.ii) because of the change in the elevation, the distance between M1 and M2 should change so as the distance traveled by the chief ray to the HOWFS stays the same. But the position of the focus changes by moving the stage MZ, so for compensating the change in the focus, the M1 and M2 mirror should tilt in X plane so as to bring back the focus at the optical axis of the WFS as described in figure 12.iii)

As shown in figure 29, when the beam from the calibration light source is located at the center of AU1.M1X (at the nominal point, the tilt for both the mirror is 30°) and AU1.M1Y and AU2.M2Y tilts are 0°, we can calculate the tilt (θ) of M1X and M2X analytically from the shifting of the stage by distance x mm.

Figure 29

[26] For the tracking of the focus, we will use only 3 actuators AU1.M1X, M2X and MZ as while tracking the AU1.M1Y and M2Y hardly moves.



Note: If we move the stage in the forward direction as shown in the figure 29, the actuators AU1.M1X and M2X will move in the forward direction (as confirmed from the AU1 Drawing of the Subaru's LGSAO188) and the tilt of the mirror M1X and M2X will be same and positive. Similarly by moving the stage backward, the actuators AU1.M1X and M2X moves backward and the tilt of the mirror M1 and M2 will be same but negative (while the actuators moves backward, the compensation of the backlash is mandatory for tracking the focus)

The equation shown below shows the calculation of the M1X and M2X tilt as a function of the motion of the stage.

$$OB' = 150 \times \sin 30^\circ = 75.0 \quad (44)$$

$$B'B = 150 \times \cos 30^\circ = 129.9 \quad (45)$$

$$\tan(30^\circ + 2\theta) = \frac{OB' + x}{B'B} \quad (46)$$

$$\theta = 0.5 \times (\tan^{-1}(\frac{OB' + x}{B'B}) - 30^\circ) \quad (47)$$

Also the change in the focal point (df) on the optical axis of the WFS can be calculated as a linear function of the change in the distance of the stage as-

$$AB = \sqrt{((150\cos 30^\circ)^2 + (150\sin 30^\circ + x)^2)} \quad (48)$$

$$df = x - 230 + 80 + AB \quad (49)$$

,where 80 mm is the distance between M2 and HOWFS and 230 mm is the total distance traveled from M1 to HOWFS at the nominal position (AU1 shown in black in figure 29)

We will move[27] the stage by 0.1 mm step (10 times), which is appropriate for our experiment, as we have considered the positions of the actuator around the nominal point. Moreover with such a small change in the distance between M1 and M2, the Gaussian center of the spot in DS9 can be readable because of the small change in the focal point on the optical axis.

After moving the stage, as the tilt of M1X (θ_{M1X}) and M2X (θ_{M2X}) is calculated analytically, the corresponding amount of shift in the actuator (in mm) of AU1.M1X and M2X can be calculated by the following equations-

We know the relation between θ_{M1X} and AU1.s_{M1X} from the equation 19, as we have considered small number of points we can deduced the linear relation from equation 19 as already done from equation 24-26 (page 56) as-

$$\theta_{M1X} = (9.29908e-01 \times AU1.s_{M1X}) - 1.1683e+01$$

$offset_{M1X} = (9.29908e-01 \times AU1_{M1X0}) - 1.1683e+01$, where $AU1_{M1X0}$ is the nominal position for M1's X positioner of AU1

$$\theta_{M1X} = (9.29908e-01 \times AU1.s_{M1X}) - 1.1683e+01 - offset_{M1X}$$

So, the corresponding actuator position (in mm) can be calculated as-

$$AU1.s_{M1X} = \frac{(\theta_{M1X} + 1.1683e+01 + offset_{M1X})}{9.29908e-01} \quad (50)$$

Similarly, the actuator position fro AU2.M2X from the corresponding tilt θ_{M2X} from the equation 29 to 31 can be calculated as-

$$AU1.s_{M2X} = \frac{\theta_{M2X} + 1.17301e+01 + offset_{M2X}}{9.35911e-01} \quad (51)$$

PROCEDURE

How AU1 is moved at the optical bench for tracking the focus?

The nominal position for AU1.M1 mirror is-

[27] Moving the stage by 0.1 mm implies that starting at the nominal position, the stage is moved forward by 0.1 mm after sending the absolute positions to the XPS Controller

M1X0= 7.11236 mm
M2X0= 7.29337 mm
M1Y0= 12.84165 mm
M2Y0= 12.99195 mm
M1Z0=-43.83951 mm

Total number of actuator positions, n = 10

1. Move the group of 5 actuators of AU1 at the nominal position.

2. Move the whole group together backward by 0.1 mm and again forward at the nominal position. Then move the actuator of the stage MZ by 0.1 mm in forward direction and the actuator of M1X and M2X according to the position calculated from equation 50 and 51 respectively and record the spot position

3. Repeat the procedure for 10 times

4. After reaching at the last specified position, move forward again by 0.1 mm and then backward at the last position and start moving the stage and AU1.M1X and M2X backward by 0.1 mm reaching back to the nominal position and record the spot positions.

5. Calculate the backlash value of AU1.M1X and M2X actuator when the whole group of actuators moved backward

6. Calculate the backlash compensation value for the actuator position of M1X and M2X in the following way-

We calculated the backlash value at every position of AU1.M1X actuator in TASK 3 of Section A, from where we can calculate the interpolated backlash of the positions that we have used in the focus tracking i.e. for AU1.s_{M1X}. Also we know the value of slope 'b_{M1X}' from the linear relation of θ_{M1X} and AU1.s_{M1X} from the equation 19.

Hence, the corresponding backlash value to be compensated in 'mm' for AU1.M1X is,

$$b_{AU1.sM1X} = \frac{\text{int}erpolated\_backlash\_for\_AU1.s_{M1X}(arc\,\text{sec}ond) \times 3600}{b_{M1X}} \quad (52)$$

Similarly for AU1.M2X, the backlash value to be compensated is-

$$b_{AU1.sM2X} = \frac{\text{int}erpolated\_backlash\_for\_AU1.s_{M2X}(arc\,\text{sec}ond) \times 3600}{b_{M2X}} \quad (52)$$

7. After calculating the compensation of backlash for both M1X and M2X actuator, repeat the procedure from 1 to 3 and apply the backlash compensation value calculated in point 6 to the point 4 and then record the spot position.

Defining Trajectory for the tracking of the focus

For defining the trajectory for the focus tracking, the PVT (Position, Velocity and Time) trajectory for multiple dimensions/axes of the XPS controller is used.

Introduction to PVT trajectory[28]

It is a continuous movement of the multi actuators in multi dimensional motion path over several time periods. For each period, each actuator completes a defined displacement from its current position and a defined output velocity at the end of the period.

Trajectory element
An element of the trajectory file (.trj) is defined by-

DT, DP1, VO1, DP2, VO2...DP_n, VO_n

DT: segment duration in seconds.
DP1, DP2,…,DP_n: Actuators (#1, #2,..., #n) displacements during DT
VO1, VO2,…,VO_n: Actuator's output velocities at the end of the DT

Trajectory Conventions
For defining and executing the PVT trajectory, some rules must be followed-

1. The motion must be of multi dimension having multiple axes.
2. The trajectory file defined by the user (.trj) must be stored in the XPS controller's memory under ../public/trajectories.
3. PVT trajectories form a continuous path i.e. each segment's output position is equal to the next segment input position.
4. The input velocity of any element is equal to the output velocity of the previous element. The input velocity for the first element is always zero. The output velocity of the last element must be zero as well.

For the focus tracking, the PVT trajectory file (.trj) for the 5 axes of the AU1 considered the nominal position as the starting point of the trajectory where all the actuators moved relatively, first, from the nominal position to the 10^{th} position (relative position after moving with the step of 0.1 mm by 10 times) and then backward again to the nominal position after applying compensation of the backlash.

For M1X, the relative displacement (RD), in mm = ($AU1.s_{M1X}(n+1)$-$AU1.s_{M1X}(n)$), where 'n' is the number of steps which is 10 here.
And the output velocity (RV), in mm/s = (($AU1.s_{M1X}(n+1)$-$AU1.s_{M1X}(n)$))/DT
Similarly the relative displacements and the relative output velocity for all the other actuators/axes is calculated in the similar manner. For M1Y and M2Y, the displacement and the velocity is '0', as they do not move while tracking the focus.

Table 17: Trajectory file for the focus tracking

[28] Chapter 9 of the User's manual software tools tutorial for XPS controller

DT sec	MZ mm	MZ mm/s	M1X mm	M1X mm/s	M1Y mm	M1Y mm/s	M2X mm	M2X mm/s	M2Y mm	M2Y mm/s
0.1,	0.1,	0.1,	RD,	RV,	0,	0,	RD,	RV,	0,	0,
0.1,	0.1,	0.1,	RD,	RV,	0,	0,	RD,	RV,	0,	0,
…	…	…	…	…	…	…	…	…	…	…
-0.1,	-0.1,	0.1,	-RD,	-RV,	0,	0,	-RD,	-RV	0,	0,
-0.1,	-0.1,	0,	-RD,	0,	0,	0,	-RD,	0	0,	0,

IDL CODING REFERENCE

Coding 20

1) Starting at the nominal position, moving the stage (M1Z0) forward by 0.1 mm by 10 times and then backward by 0.1 mm respectively and finding the value of the change in the focus at every step.

2) Converting the M1X and M2X tilt angle (calculated by the equation 50) to the corresponding actuator position for every change in M1Z0.

3) Finding the interpolated backlash value (in arcsecond) at every actuator position for M1X and M2X calculated from their tilt angle.

4) Converting the interpolated backlash value from arcsecond to the actuator position (in mm), so that the backlash compensation algorithm can be applied.

Coding 21

1) Starting at the nominal position, moving every actuator backward by 0.1 mm and again forward at the nominal position and then moving them again forward by 0.1 mm, 10 times and recording the spot position.

2) When stage is at the 10^{th} position, moving it forward by 0.1 mm and then moving it backward at the same point, taking the 10^{th} position as the starting point, moving every actuator backward by 0.1 mm, 10 times and recording the spot position

3) Compensating the backlash of the actuators when their motion get reversed

Coding 22 is for reading the Gaussian center of the spot measured from coding 21.1), 21.2) and finding the backlash of every actuator. Also reading the spot center of the image after applying compensation in coding 21.3).

Coding 23 is for defining and executing the trajectory file for the tracking of the focus without compensation.

Coding 24 is for defining and executing the trajectory file for the tracking of the focus with compensation.

RESULT

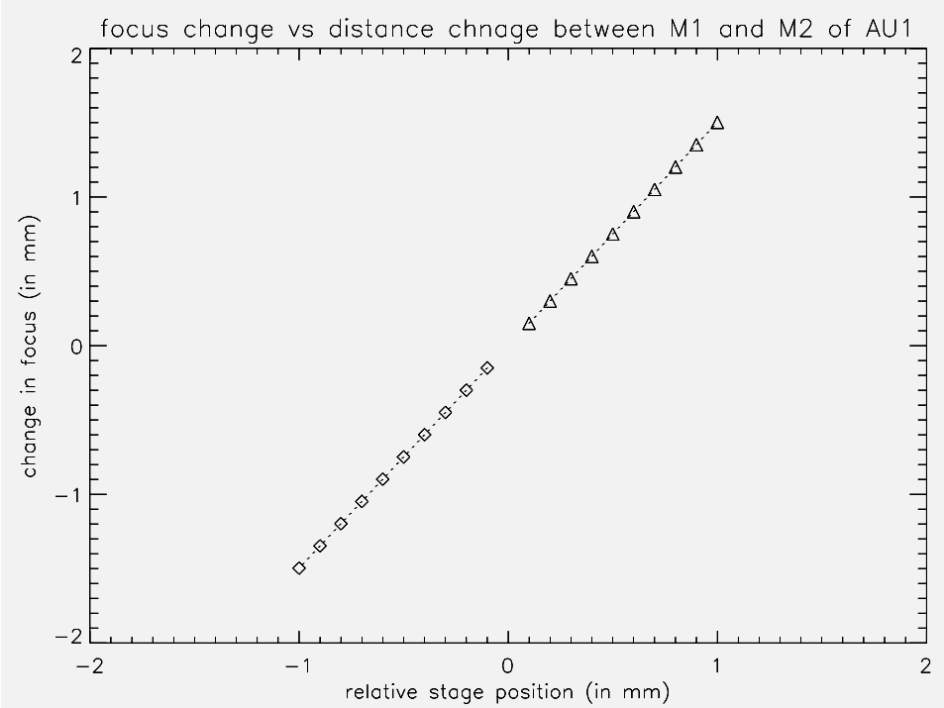
Table 18: Relation between the change in the focus with the change in the distance between M1 and M2

The table below provides the amount of the change in the focus when the stage is moved linearly by 0.1 mm forward and then backward (the absolute values are provided to the XPS controller with the step of 0.1 mm). Also showing the tilt of M1X and M2X actuator as calculated from equation (47).

M1Z (mm)	M1X (degree)	M2X (degree)	FOCUS (mm)
0.1	0.016534	0.016534	0.15002
0.2	0.033056	0.033056	0.30008
0.3	0.049570	0.049570	0.45021
0.4	0.066072	0.066072	0.60036
0.5	0.082562	0.082562	0.75058
0.6	0.099041	0.099041	0.90086
0.7	0.115509	0.115509	1.05116
0.8	0.131966	0.131966	1.20154
0.9	0.148411	0.148411	1.35194
1.0	0.164847	0.164847	1.50241
0.9	0.148411	0.148411	1.35194
0.8	0.131966	0.131966	1.20154
0.7	0.115509	0.115509	1.05116
0.6	0.099041	0.099041	0.90086
0.5	0.082562	0.082562	0.75058
0.4	0.066072	0.066072	0.60036
0.3	0.049570	0.049570	0.45021
0.2	0.033056	0.033056	0.30008
0.1	0.016534	0.016534	0.15002

Plot 24

Plot 24 shows the linear relation between the stage position and the change in the focus



When the AU1's actuator moved in the forward direction in order to follow the focus, we found that the spot position stayed almost at the nominal position. For every step in forward direction, the spot position was moved by < 1 pixel and stayed around the nominal position but as soon as the direction of the actuator get reversed, the position of the focal point changes. Because of the backlash of the actuator of M1X and M2X, the chief ray was no more on the optical axis of the HOWFS.

Table 19

The table below shows the change in the position of the focal point in arcsecond and in pixel for the AU1 when all the actuators moved in the reverse direction. Here we can see that when the actuator reverses their direction there was a sudden change in the focus because of the backlash but for the rest of the positions the change in the focus stays linear which proved the linearity of the actuator LTA-HL.

Change in focus when AU1 moves in reverse direction (arcsecond)	Change in focus when AU1 moves in reverse direction (pixel)
0.0255931	1.27966
0.0332791	1.66395
0.0328249	1.64124
0.0319388	1.59694
0.0333592	1.66796
0.0294972	1.47486
0.0287033	1.43517
0.0355544	1.77772

0.0329039	1.64520
0.0306658	1.53329

Plot 25

The plot 25 shows the change in the focus (in arcsecond) for AU1 when all the actuators reversed their direction. So because of the backlash of the actuators of AU1, the focus shifted from the optical position of the HOWFS.

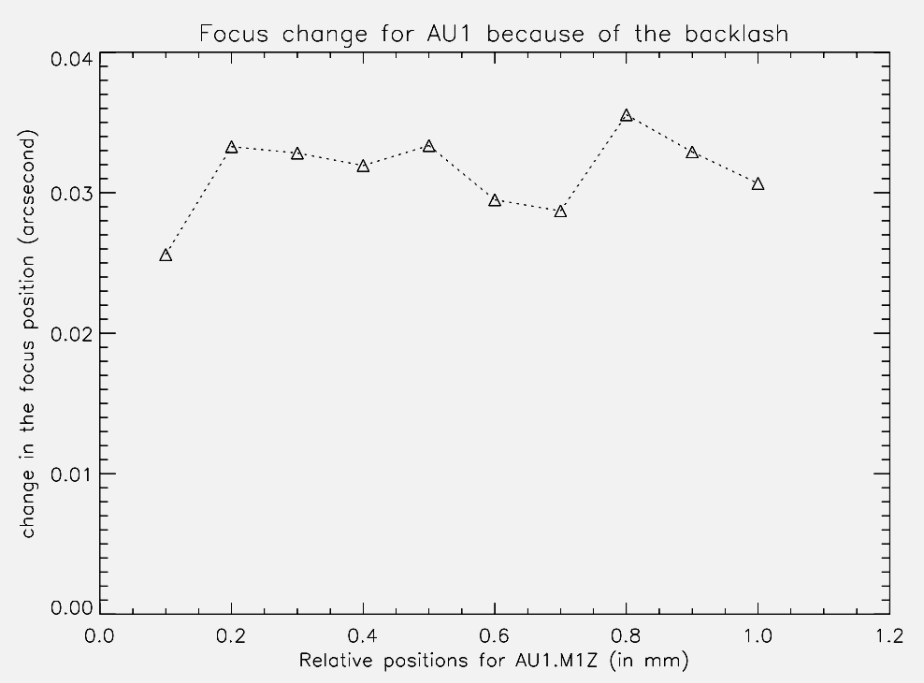


Table 20

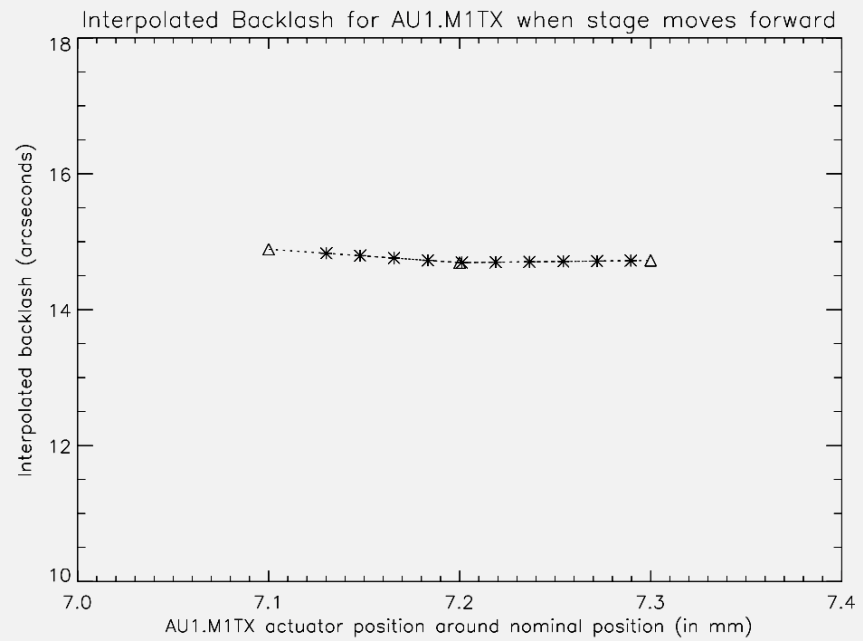
The table below shows the actuator position (in mm) for AU1.M1X and M2X corresponding to the M1X and M2X tilt, their interpolated backlash value (in arcsecond) and also the compensation (in mm) that should be applied to them during the reverse motion of the actuators in order to track the focus

Actuator position for M1X (mm)	Backlash for M1X (arcsec)	Backlash Compensation for M1X (mm)	Actuator position for M2X (mm)	Backlash for M2X (arcsec)	Backlash Compensation for M2X (mm)
7.13014	14.8298	0.0044105	7.31104	19.8024	0.0058661
7.14791	14.7948	0.0044001	7.32869	19.7915	0.0058628
7.16567	14.7599	0.0043897	7.34633	19.7805	0.0058596
7.18341	14.7250	0.0043793	7.36397	19.7696	0.0058563
7.20114	14.6928	0.0043697	7.38158	19.7587	0.0058531
7.21887	14.6986	0.0043715	7.39919	19.7477	0.0058499
7.23657	14.7043	0.0043732	7.41679	19.6316	0.0058154
7.25427	14.7100	0.0043749	7.43437	19.5104	0.0057796

7.27196	14.7158	0.0043766	7.45194	19.3893	0.0057437
7.28963	14.7215	0.0043783	7.46950	19.2684	0.0057079

Plot 26

The plot 26 shows the interpolated backlash value for AU1.M1X



Plot 27

The plot 27 shows the interpolated backlash value for AU1.M2X

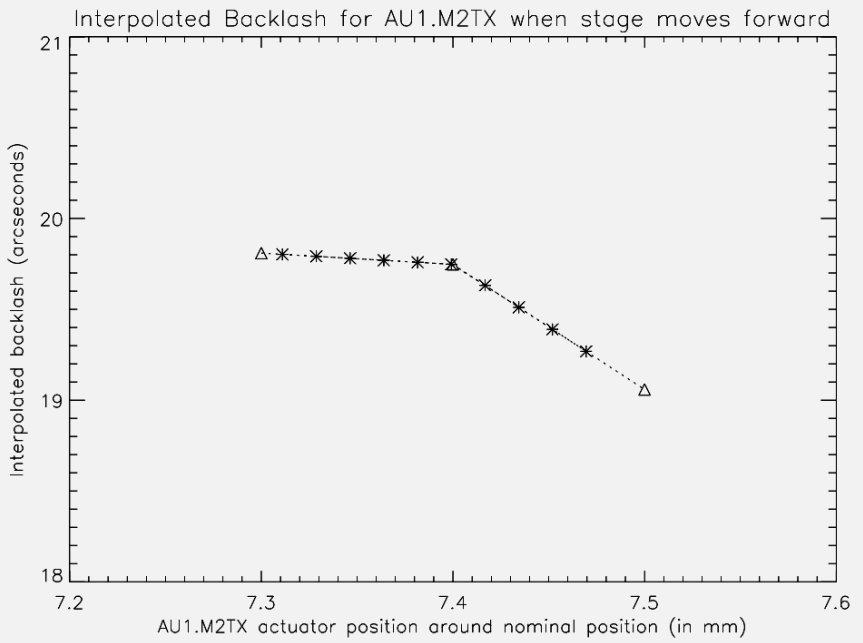


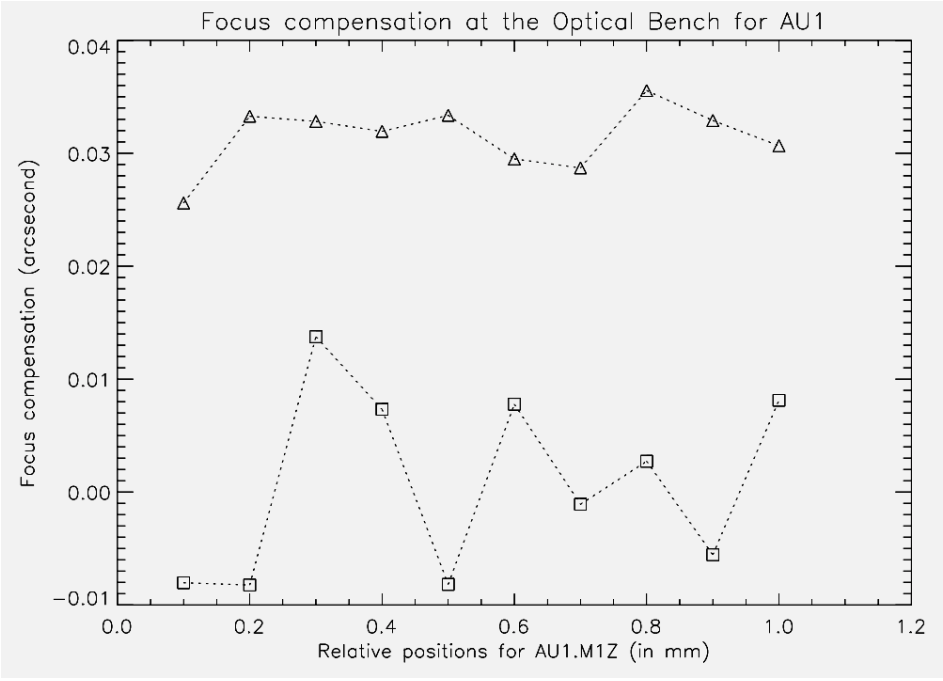
Table 21

Table 21 shows the value of the change in the focus after applying the compensation of the backlash for both AU1.M1X and M2X

Change in focus after the compensation of the backlash (arcsecond)	Change in focus after the compensation of the backlash (pixel)
-0.0080453	-0.402267
-0.0082460	-0.412301
0.013748	0.687409
0.0073099	0.365499
-0.0081905	-0.409523
0.0077709	0.388543
-0.0010926	-0.054629
0.0027179	0.135897
-0.0055642	-0.278209
0.0081204	0.406022

Plot 28

The plot 28 compares the change in the focus before applying compensation of the backlash to the actuators of AU1 (the symbol 'Δ') and the change in the focus after compensating the backlash (the symbol 'square')



CONCLUSION FOR TASK 6

Change in the Focus

We successfully calculated the relation between the linear stage and the change in focus for AU1 which is linear because we have considered only small number of measurement points around the nominal position as shown in the table 18.

We found that when the focus changes then for aligning it back to the optical axis of the WFS, if all the actuators of AU1 moves according to the equations that we have calculated, the focus can be aligned successfully on the optical axis with the error of less than 1 pixel.

Also we have seen that when the actuators moves in the backward direction in order to align the focus on the optical axis, because of the backlash, the position of the focus moves by approximately 2 pixels and if we apply the compensation then we achieved the accuracy of 0.5 pixels in the alignment of the focus on the optical axis.

As the accuracy of the change in the focus is 20mas/pixel on sky, the accuracy of 0.5 pixel or approximately 10 mas on sky after applying the compensation is quite acceptable.

Tracking of the Focus

For tracking the focus, we used the high-resolution camera for viewing the video while trajectory motion. When we executed the trajectory file (table 17) at the summit, we found that the focus can be tracked successfully when the actuators were moving in the forward direction but as soon as their motion reversed, we noticed that even after applying the compensation the focus moves by 0.5 or 1 pixel. As the accuracy of the high resolution camera is 0.339 mas/ pixel, so the movement of 0.5 pixel caused the movement of 0.1695 mas on sky whereas the accuracy for 0.5 pixel movement on sky is 10mas. The reason for this error may be:

1. Error in the trajectory file.
2. Error in the backlash compensation value for the actuator AU1.M1X and M2X.
3. Error in the setup at the optical bench as we noticed the sudden drift of the nominal position before running trajectory file several time.

DISCUSSION

For tracking the focus, the actuators should move the mirrors of the guide star acquisition unit in order to align the focus of the light source on the optical axis of the HOWFS. But because of the mechanical deficiency of the actuator, " The backlash", the deficient movement of the actuators can't follow the focus.

The first section of this report "Backlash Measurement & Compensation" described the concept and the procedure of calculating the backlash and then compensating it. With the help of various sequential subtasks (Task 1 – Task 4), we calculated the backlash for all the actuators of AU1 and AU2 and found that their values were different at the different positions of the actuator measured either at the observational floor or at the optical bench. Also because of the different setup of the acquistion unit at the observational floor and at the optical bench we measured the different backlash value but within the acceptable range which was 0.5 pixel or 10 mas. Also we applied the compensation of the backlash for both AU1 and AU2 at the optical bench and found the compensated backlash value within the specification of the repeatability listed in Table 7 (concluded in Task 4).

As the LGSAO188 project of Subaru Telescope use the AU1 for the HOWFS for the tracking of the focus, so our main concerned was compensating the backlash for X actuator of the mirrors of the AU1.

The second section of this report describe the "Focus Tracking". We tried to understand the trajectory motion in 2D and multiD plane with the help of the XPS Controller. We used the AU1 setup at the optical bench and a high-resolution camera for viewing the motion of the spot at the summit of the Mauna Kea. We created the trajectory file for the focus tracking and executed it. We succeded in tracking the focus in the forward motion of the actuator but not for the backward motion, when actuators reverse their direction. For the forward motion of the actuators, we aligned the spot of the light source on the optical axis of the high-resolution camera but during the reverse motion, we observed the sudden drift in the spot center by 0.5 or 1 pixel which was not acceptable. Moreover we observed the drift in the nominal position due course of time so we could say that the setup error, the measurement error and the backlash compensation error caused the movement of the light spot from the nominal position.

Although at the end of the internship, we couldn't acheived the tracking within the accuracy range but still we described the methods and developed the model for the tracking. We hope to do the corrections in the trajectory file which is defined for the 5 axes of the AU1 and will try to achieve the result before the presentation of the project.

BIBLIOGRAPHY

[1] JOHN W. HARDY, (1998), "Adaptive Optics for Astronomical Telescopes"

[2] N. AGEORGES & C. DAINTY, "Laser Guide Star Adaptive Optics for Astronomy"

[3] NEWPORT, User's Manual Software Tools Tutorial for XPS Controller

[4] Laser guide star AO 188 http://www.naoj.org/Observing/Instruments/AO/

[5] XPS Universal High Performance Motion Controller Driver http://www.newport.com/XPS-Universal-High-Performance-Motion-Controller-/300904/1033/catalog.aspx

[6] Mokoto WATANABE, (2004), " Functional & Performance Requirements for Guide Star Acquisition Units of Subaru LGSAO"

[7] M. Watanabe, et al. (2004), "Design of the Subaru laser guide star adaptive optics module", *Proc., SPIE 5490*

[8] H. Takami, et al. (2004), "Laser guide star AO project at the Subaru Telescope"

[9] H. Takami, Y. Hayano (2004), "Performance of Subaru Adaptive optic system"

[10] Takami, H. et al. (2006), "Status of Subaru laser guide star AO system" in [Advances in Adaptive Optics II], *Proc. SPIE 6272.*

[11] Takami, H. et al. (2006), "The laser guide star facility for Subaru Telescope", *Proc. SPIE 6272*

[12] Y. Hayano, et al. (2009), "Progress of the laser guide star adaptive optics system at Subaru Telescope", *American Institute of Physics Conference Series 1158*

[13] Y. Hayano, et al. (2010), "Commissioning status of Subaru laser guide star adaptive optics system", *Proc. SPIE 7736*

[14] Y. Minowa, et al. (2010), "Performance of Subaru adaptive optics system AO188", *Proc. 7736*

ANNEXE

LIST OF SYMBOLS

$AU1.s_{M1X}$	Actuator positions for AU1's X positioner for M1 for the guide star acquisition unit
$AU1.s_{M1Y}$	Actuator positions for AU1's Y positioner for M1 for the guide star acquisition unit
$AU1.s_{M2X}$	Actuator positions for AU1's X positioner for M2 for the guide star acquisition unit
$AU1.s_{M2Y}$	Actuator positions for AU1's Y positioner for M2 for the guide star acquisition unit
$AU1_{M1X0}$	The nominal position of M1's X positioner at the optical bench which is, 7.11236 mm
$AU1_{M1Y0}$	The nominal position of M2's X positioner at the optical bench which is, 12.84165 mm
$AU1_{M2X0}$	The nominal position of M1's Y positioner at the optical bench which is, 7.29337 mm
$AU1_{M2Y0}$	The nominal position of M2's Y positioner at the optical bench which is, 12.99195 mm
$AU2_{M1X0}$	The nominal position of M1's X positioner at the optical bench which is, 12.0 mm
$AU2_{M1Y0}$	The nominal position of M2's X positioner at the optical bench which is, 12.63 mm
$AU2_{M2X0}$	The nominal position of M1's Y positioner at the optical bench which is, 11.75 mm
$AU2_{M2Y0}$	The nominal position of M2's Y positioner at the optical bench which is, 12.43 mm
n	Number of measurement points
sx(0), mm	Nominal position of the Positioner X for which $\Delta\theta_x=\Delta\theta x(0)$, which is 7.1 mm for the prototype of LGSAU1
sy(0), mm	Nominal position of the Positioner Y for which $\Delta\theta_y=\Delta\theta y(0)$, which is 12.2 mm for the prototype of LGSAU1
sx(n), mm	Actuator's shift from 0 to 25 (measurement points) for the X actuator
sy(n), mm	Actuator's shift from 0 to 25 (measurement points) for the Y actuator
s1(n), mm	Displaced position of the Positioner/Actuator after subtracting from the nominal position
Δs, mm	Backlash value of the actuator
$Text^{number}$	Please see the Bibliography for the references
Δθx(0), radian	Nominal position of the Mirror in X plane, which is '0'
Δθy(0), radian	Nominal position of the Mirror in Y plane, which is '0'
θ(n), radian	Theoretical value of the mirror's angle corresponding to the actuator's shift as calculated from the Newton-Raphson method

$\Delta\theta x(n)$, in radians	Mirror's angle measured from ACT in X plane
$\Delta\theta y(n)$, in radians	Mirror's angle measured from ACT in Y plane
$\Delta\theta x_f / \Delta\theta x_b$	$\Delta\theta x(n)$ in XY plane when the actuator moves forward/backward from the measurement point
$\Delta\theta y_f / \Delta\theta y_b$	$\Delta\theta y(n)$ in XY plane when the actuator moves forward/backward from the measurement point.
$\Delta\theta_x(n)$	overall tilt in the mirror in XY plane because of the X Positioner
$\Delta\theta_y(n)$	overall tilt in the mirror in XY plane because of the Y Positioner

IDL CODING

Backlash Measurement & Compensation

TASK 1

CODING 1

```
PRO testing,start_point,end_point
number_points   =  48.
number_steps    =  10.
step_increment  =  0.5

;; open file

filename  =  'testing'+'_'+strcompress(string(start_point,format='(F5.2)')
,/remove_all)+'_'+ strcompress(string(end_point,format='(F5.2)'),/remove_all)+'.dat'
openw,lun,filename , /get_lun ;,/append

for loop_point  =  0, number_points, 1 do begin
commanded_position =  start_point+(end_point-
start_point)/DOUBLE(number_points)* loop_point
commanded_position1 =  commanded_position
print, 'position sent to XPS ',commanded_position1
wait,15

;; 10 measurements at each point

for loop_step  =  0, number_steps do begin

;; calculate the commanded position
;; move to start position

starting_position   =   start_point + (end_point - start_point)
/DOUBLE(number_points) * loop_point
commanded_position_1   =   sqrt(starting_position*starting_position)
cmd   =   './xps_raw.tcl 133.40.147.71 "GroupMoveAbsolute(GROUP1.POSITIONER,'
+ STRING(commanded_position_1, FORMAT ='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd  =  './xps_raw.tcl 133.40.147.71
"GroupPositionCurrentGet(GROUP1.POSITIONER, double *)"'
SPAWN, cmd, ret

actual_position  =  double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position

;; reverse the direction by 0.5 mm
```

```
commanded_position_1  =   starting_position– step_increment
print, 'position of XPS reversed by -0.5 mm: ', commanded_position_1

cmd  = './xps_raw.tcl 133.40.147.71 "GroupMoveAbsolute (GROUP1.POSITIONER, '
+ STRING(commanded_position_1, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1
cmd  =  './xps_raw.tcl 133.40.147.71 "GroupPositionCurrentGet
(GROUP1.POSITIONER, double *)"'
SPAWN, cmd, ret

actual_position = double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position

;; again moving to the measured point

commanded_position_1 = commanded_position_1 + step_increment
print, 'first data from ACT for the measured point: ',commanded_position_1
cmd = './xps_raw.tcl 133.40.147.71 "GroupMoveAbsolute (GROUP1.POSITIONER, '
+ STRING(commanded_position_1, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1
cmd  =  './xps_raw.tcl 133.40.147.71 "GroupPositionCurrentGet
(GROUP1.POSITIONER, double *)"'
SPAWN, cmd, ret

actual_position = double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
wait,5

;; measure position with auto-collimation telescope

cmd  =  './act_meas.sh meas'
SPAWN, cmd, ret
tmp  =  STRSPLIT(ret, ',', /EXTRACT)
pos_x  = DOUBLE(tmp[4])
pos_y  = DOUBLE(tmp[3])
PRINT, pos_x
PRINT, pos_y
wait, 1

;; move the measurement point by 0.5 mm further

commanded_position_1 = commanded_position_1 + step_increment
print, 'position of measurement point moved by +0.5 mm: ', commanded_position_1
cmd = './xps_raw.tcl 133.40.147.71 "GroupMoveAbsolute (GROUP1.POSITIONER,' +
STRING(commanded_position_1, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
```

```
wait, 1

cmd = './xps_raw.tcl 133.40.147.71 "GroupPositionCurrentGet
(GROUP1.POSITIONER, double *)"'
SPAWN, cmd, ret

actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position

;; coming back to the measurement point

commanded_position_1=commanded_position_1 - step_increment
print, 'second data from ACT for the measured point: ', commanded_position_1

cmd = './xps_raw.tcl 133.40.147.71 "GroupMoveAbsolute (GROUP1.POSITIONER,' +
STRING(commanded_position_1, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 133.40.147.71 "GroupPositionCurrentGet
(GROUP1.POSITIONER, double *)"'
SPAWN, cmd, ret

actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
wait,5

 ;; again measure the new position with auto-collimation telescope

cmd = './act_meas.sh meas'
SPAWN, cmd, ret

tmp = STRSPLIT(ret, ',', /EXTRACT)
pos_x1 = DOUBLE(tmp[4])
pos_y1 = DOUBLE(tmp[3])
PRINT, pos_x1
PRINT, pos_y1
wait,1

printf,lun,commanded_position_1,pos_x,pos_y,pos_x1,pos_y1,FORMAT='(5F12.6)'

endfor
wait,1
endfor
close,lun
END

*****************************************************************************
```

CODING 2

```
;; f is the function and fd is its derivative

PRO numerical1,number_points,start_point,end_point

print, 'parameters:',n_params()
a = 65e-3
b = 22.08e-3

;;start_point=0.0122055
;;end_point=0.0122661
;;number_points=48.

step_increment = (end_point-start_point)/number_points
y = fltarr(number_points+1)
error1 = fltarr(number_points+1)
position = fltarr(number_points+1)
position(0) = start_point
y0 = start_point/a                    ;; initial guess  (in radians)
print,'y0',y0
e = 10e-7
print,'position',position

;; the equation is --------> f(y)= a*sin(y)-sqrt(b^2-a^2*(1-cos(y))^2)+b-position

error = a*sin(y0)-sqrt(b^2-a^2*(1-cos(y0))^2)+b-start_point
print,'error',error

while abs(error) gt e do begin

square = b^2-a^2*(1-cos(y0))^2
fy = a*sin(y0)-sqrt(square)+b-start_point
fd = a*cos(y0) - (2*a^2*(1-cos(y0))*sin(y0))/sqrt(square)
y0 = y0-fy/fd
error = a*sin(y(0))-sqrt(b^2-a^2*(1-cos(y(0)))^2)+b-start_point

endwhile

y(0) = y0
print,'approximate values for y0 is',y0
slope = start_point/y(0)
print,''
print,'slope',slope
print,''

for loop_point=1, number_points,1 do begin
```

```
position(loop_point) = start_point+ (end_point - start_point)/
DOUBLE(number_points)*loop_point
print,'position', position(loop_point)
y(loop_point) = position(loop_point)/slope              ;;  guess value for y
print,'the guess value for
y'+strcompress(string(loop_point),/remove_all)+'is',y(loop_point)

error = a*sin(y(loop_point))-sqrt(b^2-a^2*(1-cos(y(loop_point)))^2)+b-
position(loop_point)
print,'error1', error

while abs(error) gt e do begin

square1 = b^2-a^2*(1-cos(y(loop_point)))^2
print,'square1',square1
fy = a*sin(y(loop_point))-sqrt(square1)+b-position(loop_point)
fd =a*cos(y(loop_point))-
(2*a^2*(1cos(y(loop_point)))*sin(y(loop_point)))/sqrt(square1)
y(loop_point) = y(loop_point)-fy/fd
error = a*sin(y(loop_point))-sqrt(b^2-a^2*(1-cos(y(loop_point)))^2)+b
position(loop_point)
print,'error2', error
print,'y(loop_point)',y(loop_point)

endwhile

print,'function value for y'+strcompress(string(loop_point),/remove_all)+'is',fy
print,'function derivative value',fd
y(loop_point) = y(loop_point)
print,'approximate values of
y'+strcompress(string(loop_point),/remove_all)+'is',y(loop_point)
print,''

endfor

print,''
print,'position values are',position
print,''
print,'values for the angle in radians',y

SET_PLOT, 'PS'
DEVICE,/portrait,filename = 'angle_approximation.ps', /INCHES
plot,position,y,XRANGE=[7.0e-3,20.0e-3],$
TITLE='Angle Approximation by Newton-Raphson Method',$
XTITLE='Mesurement Position(meter)',$
YTITLE='angle (in radians)'
DEVICE, /CLOSE
```

```
;; checking error

for i=0, number_points,1 do begin

error1(i) = a*sin(y(i))-sqrt(b^2-a^2*(1-cos(y(i)))^2)+b-position(i)

endfor

print,'y0',y0
print,''
print,'error1',error1

save,y,error1,position,filename='error.sav'

end
```

****

CODING 3

```
 PRO read_2_data,number_points,start_point,end_point

number_steps = 10.
s=(0.50+findgen (48)*0.5)1e-3      ;; measured points in meters
s1=fltarr(49)
s0=7.1e-3          ;; s0 (in meters) = actuator's position at zero mirror's angle
s1=s-s0            ;; s1 (in meters) = actuator's position after subtracting s0
data=fltarr(5,11,49)
delta_theta1=fltarr(49)
Delta_theta1=fltarr(49)
Delta_theta1_avg=fltarr(49)
total_stddev1=fltarr(49)               ;; standard deviation
std=fltarr(49)
theta=fltarr(49)
total_theta=fltarr(49)
s2=fltarr(49)                          ;; s2=real actuator's position-s0
delta_s=fltarr(49)                     ;; delta_s=s2-s1, final backlash value in mm
a=65e-3
b=22.08e-3

filename='testing'+'_'+strcompress(string(start_point,format='(F5.2)'),/remove_all)+'_
'+strcompress(string(end_point,format='(F5.2)'),/remove_all)+'.dat'

openr, lun, filename, /GET_LUN
readf,lun,data
close, lun

for i=0,number_points,1 do begin
```

```
;; method 1

for loop_step=0, number_steps do begin

Delta_theta=sqrt((data[3,loop_step,i]-data[1,loop_step,i])^2+(data[4,loop_step,i]
            data[2,loop_step,i])^2)
Delta_theta1(loop_step)=0.01745*Delta_theta

endfor

plot,n,data[1,*,i],title='X_backward vs number points', xtitle='number
points',ytitle='x_backward'
Delta_theta1_avg(i)=total(Delta_theta1)/n_elements(Delta_theta1)

;; method 2

;; finding the average of each measurement

print,''
print,'For Measurement point',s1(i)
avg_x_back= total(data[1,*,i])/n_elements(data[1,*,i])
avg_y_back= total(data[2,*,i])/n_elements(data[2,*,i])
avg_x_forw= total(data[3,*,i])/n_elements(data[3,*,i])
avg_y_forw= total(data[4,*,i])/n_elements(data[4,*,i])

print,'avg_x_back',avg_x_back
print,'avg_y_back',avg_y_back
print,'avg_x_forw',avg_x_forw
print,'avg_y_forw',avg_y_forw

;; standard deviation of x and y

back= [variance(data[1,*,i]),variance(data[2,*,i])]
forw=[variance(data[3,*,i]), variance(data[4,*,i])]
var_xy_back=mean(back)
var_xy_forw= mean(forw)

print,'std_xy_forw',sqrt(var_xy_forw)
print,'std_xy_back',sqrt(var_xy_back)

;; total standard deviation

total_var=var_xy_forw+var_xy_back
total_stddev= sqrt(total_var)
print,''
total_stddev1(i)=0.01745*total_stddev

;; delta_theta corresponding to the displacement value in degrees (i)
```

```
delta_theta=sqrt((avg_x_forw-avg_x_back)^2+(avg_y_forw-avg_y_back)^2)
delta_theta1(i)=0.01745*delta_theta

endfor

;; measuring angle theta from the formula

restore,filename='error.sav'                ;; restoring the y (theta) and error value
theta=y
error1=error

;; adding angle's value to the backlash measured before

total_theta=theta+delta_theta1                          ;; in radians

;; actuator's shift for new angle

s2=a*sin(total_theta)-sqrt(b^2-a^2*(1-cos(total_theta))^2)+b

;; result

delta_s=(s2-s1)/2
delta_theta1=delta_theta1/2

;; standard dviation value in micrometers

std=(a*sin(total_stddev1)-sqrt(b^2-a^2*(1-cos(total_stddev1))^2)+b)/2
print,''
print,'the value of angle theta is',theta
print,''
print,'the value of angle delta_theta1 is',delta_theta1
print,''
print,'angle after adding theta+delta_theta=total_theta',total_theta
print,''
print,'s1',s1
print,''
print,'actuator shift s2',s2
print,''
print,'result delta_s (backlash value) is',delta_s
print,''
print,'std deviation for measurement point',std
print,''
print,'average value of the backlash',total(delta_s)/n_elements(delta_s)
print,''
avg_stddev=total(std)/n_elements(std)
print,'average value of standard deviation = ',avg_stddev
print,''
print,' Delta_theta1 (in radians) at the measurement point',Delta_theta1
print,''
```

```
print,' delta_theta1 (in radians) at the measurement point',delta_theta1

delta_s=delta_s*1e+6                ;; expressing backlash value in micro meters
s1=s1*1e+3                          ;; expressing actuator's shift in micro meters
s11=[s1[0:12],s1[14:48]]
delta_s1=[delta_s[0:12],delta_s[14:48]]
delta_theta11=[delta_theta1[0:12],delta_theta1[14:48]]

SET_PLOT, 'PS'
DEVICE,/portrait,filename = 'backlash_mm1.ps', /INCHES
plot,sqrt(s11*s11),delta_s1,PSYM=-2,XRANGE=[-7.0,20.0],YRANGE=[0.0,12.0],$
TITLE='Backlash Measurement',$
XTITLE='Mesurement Position(in mm)',$
YTITLE='Backlash (in micrometer)'
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename = 'backlash_mm.ps', /INCHES
plot,s11,delta_s1,XRANGE=[-7.0,20.0],PSYM=-2,YRANGE=[0.0,12.0],$
TITLE='Backlash Measurement',$
XTITLE='Mesurement Position (in mm)',$
YTITLE='Backlash (in micrometer)'
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename = 'backlash_radians.ps', /INCHES
plot,s11,delta_theta11,PSYM=-2,XRANGE=[-7.0,20.0], $
TITLE='Backlash Measurement',$
XTITLE='Mesurement Position (millimeter)',$
YTITLE='Backlash (in radians)'
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_degrees.ps', /INCHES
plot,s11,57.2985*(delta_theta11),PSYM=-2,XRANGE=[-7.0,20.0],$
TITLE='Backlash Measurement',$
XTITLE='Mesurement Position (millimeter)',$
YTITLE='Backlash (in degrees)'
DEVICE, /CLOSE

;; sky backlash measurement for AU's M2

l1=150e-3
l2=80           ;; in mm
f_tel=110                       ;; in m
delta_x=l2*new_delta_theta1
sky_backlash=delta_x*180*3600/(!PI*110000)      ;; in arcseconds
print,''
print,'sky_backlash',sky_backlash
```

```
print,"
print,'average value of sky backlash
is',total(sky_backlash)/n_elements(sky_backlash)

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'sky_backlash.ps', /INCHES
plot,s11,sky_backlash,PSYM=-2,XRANGE=[-7.0,20.0],$
TITLE='sky backlash',$
XTITLE='Mesurement Position (millimeter)',$
YTITLE='Backlash (in radians)'
DEVICE, /CLOSE

end
```

****

TASK 2

CODING 4

```
PRO summit_test_M1M2_TX, M1TXstart, M2TXstart, number_points, n,
number_steps

;; number_points  =    48.
;; number_steps   =    10.

data  =     fltarr(5,number_points)
filename='x.tab'
openr, lun, filename, /GET_LUN
readf,lun,data
close, lun
step_increment =  0.5

;; nominal positions

M1X0  =    12.5                  ;; in mm
M1Y0  =    12.508
M2X0  =    12.5
M2Y0  =    12.428

M1TX  =    data[1,*]
M1TY  =    data[2,*]             ;; in mm
M2TX  =    data[3,*]
M2TY  =    data[4,*]

print,"
print,'Finding the backlash value for AU2.M1TX'
;; open file
```

```
filename='AU2.M1TX_real'+'_'+strcompress(string(M1TXstart,format='(F5.2)'),/remo
ve_all)+'.dat'
openw,lun,filename , /get_lun ;,/append

count=0

for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU2.M1TY is : ', M1Y0
print,''
print,'nominal position for AU2.M2TY is : ', M2Y0
print,''
print,'moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively'
print,''
print,'measurement position for AU2.M1TY is :',M1Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' + STRING(M1Y0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY is : ', actual_position
print,''
print,'measurement position for AU2.M2TY is :',M2Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' + STRING(M2Y0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY is : ', actual_position
print,''
print,'measurement position for AU2.M2TX is :',M2TX(loop_point)
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(M2TX(loop_point), FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
```

```
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
AU2_M1TX_starting_position  =    M1TX(loop_point)
print,'the measurement position for AU2.M1TX under consideration is
:',AU2_M1TX_starting_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_starting_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 3

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''
print,' moving the actuator backward by n=5 times from the measurement position'
print,''

;;   moving the actuator backward by 5 times from the measurement position

count1=0

for i=0, n-1 ,1 do begin

;; moving the actuator by 0.5 mm in backward direction

count1=count1+1
print,'count1',count1
print,''
AU2_M1TX_commanded_position  =   AU2_M1TX_starting_position -step_increment
print, 'commanded position for AU2.M1TX for moving in backward direction by -0.5
mm is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX : ', actual_position
print,''

 ;;  again moving to the measured point

AU2_M1TX_commanded_position = AU2_M1TX_commanded_position +
step_increment
```

```
print, 'commanded position for AU2.M1TX after coming back to the measurement
position is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU2.M1TX'
print,'*****************************************************'
print,''

count2=0

for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU2.M1TX is -

count2=count2+1
print,'count2',count2
print,''

 ;;  taking 10 times back and forth measurements for AU2.M1TX
 ;;  for moving the measurement position by 0.5 mm in backward direction

print, 'actual position of AU2.M1TX: ', actual_position

AU2_M1TX_commanded_position  =   AU2_M1TX_starting_position -
step_increment
print, 'commanded position for AU2.M1TX for moving in backward direction by -0.5
mm is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX : ', actual_position
;;  again moving to the measured point
```

```
AU2_M1TX_commanded_position  =  AU2_M1TX_commanded_position +
step_increment
print, 'commanded position for AU2.M1TX after coming back to the measurement
position is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX: ', actual_position
print,''

;; measure position with auto-collimation telescope

cmd = './act_meas.sh meas'
print, 'data from ACT when the actuator moved backward and comes back to the
measurement position are :'
SPAWN, cmd, ret
wait,1
tmp = STRSPLIT(ret, ',', /EXTRACT)
pos_back_AU2_M1TX_x = DOUBLE(tmp[4])
pos_back_AU2_M1TX_y = DOUBLE(tmp[3])
PRINT,'pos_back_AU2.M1TX.x', pos_back_AU2_M1TX_x
PRINT,'pos_back_AU2.M1TX.y', pos_back_AU2_M1TX_y
print,''
wait, 1

;;  for moving the measurement position by 0.5 mm in forward direction

AU2_M1TX_commanded_position  =  AU2_M1TX_commanded_position +
step_increment
print, 'commanded position for AU2.M1TX for moving in forward direction by +0.5
mm is ',  AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; coming back to the measurement point
```

```
AU2_M1TX_commanded_position  =  AU2_M1TX_commanded_position -
step_increment
print, 'commanded position for AU2.M1TX after coming back to the measurement
position is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; again measure the new position with auto-collimation telescope

cmd = './act_meas.sh meas'
print, 'data from ACT when the actuator moved forward and comes back to the
measurement position are : '
SPAWN, cmd, ret
wait,1
tmp = STRSPLIT(ret, ',', /EXTRACT)
pos_forw_AU2_M1TX_x = DOUBLE(tmp[4])
pos_forw_AU2_M1TX_y = DOUBLE(tmp[3])
PRINT,' pos_forw_AU2.M1TX.x', pos_forw_AU2_M1TX_x
PRINT,'pos_forw_AU2.M1TX.y', pos_forw_AU2_M1TX_y
print,''
wait,1

printf, lun ,AU2_M1TX_commanded_position ,M2TX(loop_point)
,pos_back_AU2_M1TX_x ,pos_back_AU2_M1TX_y ,pos_forw_AU2_M1TX_x
,pos_forw_AU2_M1TX_y ,FORMAT='(6F12.6)'

endfor
wait,1
endfor

close,lun
print,''
print,''
print,'Finding the backlash value for the AU2.M2TX '

;; open file

filename='AU2.M2TX_real'+'_'+strcompress(string(M2TXstart,format='(F5.2)'),/remo
ve_all)+'.dat'
openw,lun1,filename , /get_lun ;,/append
```

```
printf,lun1,'the values in each columns belongs to AU2_M2TX_commanded_position
,M1TX ,pos_back_AU2_M2TX_x ,pos_back_AU2_M2TX_y ,pos_forw_AU2_M2TX_x
,pos_forw_AU2_M2TX_y respectively'

count=0

for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU2.M1TY is : ', M1Y0
print,''
print,'nominal position for AU2.M2TY is : ', M2Y0
print,''
print,'moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively'
print,''

print,'measurement position for AU2.M1TY is :',M1Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' + STRING(M1Y0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY is : ', actual_position
print,''

print,'measurement position for AU2.M2TY is :',M2Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' + STRING(M2Y0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY is : ', actual_position
print,''
print,'measurement position for AU2.M1TX is :',M1TX(loop_point)
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(M1TX(loop_point), FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1
```

```
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''

AU2_M2TX_starting_position  =  M2TX(loop_point)
print,'the measurement position for AU2.M2TX under consideration is
:',AU2_M2TX_starting_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_starting_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
print,' moving the actuator backward by n=5 times from the measurement position'
print,''

;;   moving the actuator backward by 5 times from the measurement position

count1=0

for i=0, n-1 ,1 do begin

;; moving the actuator by 0.5 mm in backward direction

count1=count1+1
print,'count1',count1
print,''

AU2_M2TX_commanded_position  =   AU2_M2TX_starting_position -
step_increment
print, 'commanded position for AU2.M2TX for moving in backward direction by -0.5
mm is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX : ', actual_position
print,''
```

```
;;  again moving to the measured point
AU2_M2TX_commanded_position  =   AU2_M2TX_commanded_position +
step_increment
print, 'commanded position for AU2.M2TX after coming back to the measurement
position is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU2.M2TX'
count2=0

for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU2.M2TX is -

count2=count2+1
print,'count2',count2
print,''

;;  taking 10 times back and forth measurements for AU2.M2TX
;;  for moving the measurement position by 0.5 mm in backward direction

print, 'actual position of AU2.M2TX: ', actual_position

AU2_M2TX_commanded_position  =   AU2_M2TX_starting_position -
step_increment
print, 'commanded position for AU2.M2TX for moving in backward direction by -0.5
mm is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX : ', actual_position
print,''
```

```
;;  again moving to the measured point

AU2_M2TX_commanded_position  =   AU2_M2TX_commanded_position +
step_increment
print, 'commanded position for AU2.M2TX after coming back to the measurement
position is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX: ', actual_position
print,''

;; measure position with auto-collimation telescope

cmd = './act_meas.sh meas'
print, 'data from ACT when the actuator moved backward and comes back to the
measurement position are :'
SPAWN, cmd, ret
wait,1
tmp = STRSPLIT(ret, ',', /EXTRACT)
pos_back_AU2_M2TX_x = DOUBLE(tmp[4])
pos_back_AU2_M2TX_y = DOUBLE(tmp[3])
PRINT,'pos_back_AU2.M2TX.x', pos_back_AU2_M2TX_x
PRINT,'pos_back_AU2.M2TX.y', pos_back_AU2_M2TX_y
print,''
wait, 1

;;  for moving the measurement position by 0.5 mm in forward direction

AU2_M2TX_commanded_position  =   AU2_M2TX_commanded_position +
step_increment
print, 'commanded position for AU2.M2TX for moving in forward direction by +0.5
mm is ',  AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
 ;; coming back to the measurement point
```

```
AU2_M2TX_commanded_position  =  AU2_M2TX_commanded_position -
step_increment
print, 'commanded position for AU2.M2TX after coming back to the measurement
position is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; again measure the new position with auto-collimation telescope

cmd = './act_meas.sh meas'
print, 'data from ACT when the actuator moved forward and comes back to the
measurement position are : '
SPAWN, cmd, ret
wait,1
tmp = STRSPLIT(ret, ',', /EXTRACT)
pos_forw_AU2_M2TX_x = DOUBLE(tmp[4])
pos_forw_AU2_M2TX_y = DOUBLE(tmp[3])
PRINT,' pos_forw_AU2.M2TX.x', pos_forw_AU2_M2TX_x
PRINT,'pos_forw_AU2.M2TX.y', pos_forw_AU2_M2TX_y
print,''
wait,1

printf, lun1 ,AU2_M2TX_commanded_position ,M1TX(loop_point)
,pos_back_AU2_M2TX_x ,pos_back_AU2_M2TX_y ,pos_forw_AU2_M2TX_x
,pos_forw_AU2_M2TX_y ,FORMAT='(6F12.6)'

endfor
wait,1
endfor
close,lun

END
```

****

CODING 5

```
PRO summit_test_M1M2_TY, M1TYstart, M2TYstart, number_points, n,
number_steps
```

```
;; number_points  =    48.
;; number_steps   =    10.

data     =   fltarr(5,number_points)
filename =   'y.tab'
openr, lun, filename, /GET_LUN
readf,lun,data
close, lun

step_increment =    0.5

;; nominal positions

M1X0     =   12.5                ;; in mm
M1Y0     =   12.508
M2X0     =   12.5
M2Y0     =   12.428

M1TX     =   data[1,*]
M1TY     =   data[2,*]              ;; in mm
M2TX     =   data[3,*]
M2TY     =   data[4,*]

print,''
print,'Finding the backlash value for AU2.M1TY'

;; open file

filename='AU2.M1TY'+'_'+strcompress(string(M1TYstart,format='(F5.2)'),/remove_al
l)+'.dat'
openw,lun,filename , /get_lun ;,/append
printf,lun,'the values in each coloumns belongs to AU2_M1TY_commanded_position
,M2TY ,pos_back_AU2_M1TY_x ,pos_back_AU2_M1TY_y ,pos_forw_AU2_M1TY_x
,pos_forw_AU2_M1TY_y respectively '

count=0

for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU2.M1TX and AU2.M2TX to M1X0 and M2X0 position respectively-

print,'nominal position for AU2.M1TX is : ', M1X0
print,''
print,'nominal position for AU2.M2TX is : ', M2X0
print,''
```

```
print,'moving AU2.M1TX and AU2.M2TX to M1X0 and M2X0 position respectively'
print,''
print,'measurement position for AU2.M1TX is :',M1X0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' + STRING(M1X0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''

print,'measurement position for AU2.M2TX is :',M2X0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' + STRING(M2X0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
print,'measurement position for AU2.M2TY is :',M2TY(loop_point)
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' +
STRING(M2TY(loop_point), FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY is : ', actual_position
print,''
AU2_M1TY_starting_position  =  M1TY(loop_point)
print,'the measurement position for AU2.M1TY under consideration is
:',AU2_M1TY_starting_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' +
STRING(AU2_M1TY_starting_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY is : ', actual_position
print,''
```

```
print,' moving the actuator backward by n=5 times from the measurement position'

;;   moving the actuator backward by 5 times from the measurement position

count1=0

for i=0, n-1 ,1 do begin

;; moving the actuator by 0.5 mm in backward direction

count1=count1+1
print,'count1',count1
print,''
AU2_M1TY_commanded_position  =   AU2_M1TY_starting_position -
step_increment
print, 'commanded position for AU2.M1TY for moving in backward direction by -0.5
mm is : ', AU2_M1TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' +
STRING(AU2_M1TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY : ', actual_position
print,''

;;  again moving to the measured point

AU2_M1TY_commanded_position  =   AU2_M1TY_commanded_position +
step_increment
print, 'commanded position for AU2.M1TY after coming back to the measurement
position is : ', AU2_M1TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' +
STRING(AU2_M1TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU2.M1TY'
count2=0
```

```
for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU2.M1TY is -

count2=count2+1
print,'count2',count2
print,''

;;  taking 10 times back and forth measurements for AU2.M1TY
;;  for moving the measurement position by 0.5 mm in backward direction

print, 'actual position of AU2.M1TY: ', actual_position
AU2_M1TY_commanded_position= AU2_M1TY_starting_position - step_increment
print, 'commanded position for AU2.M1TY for moving in backward direction by -0.5
mm is : ', AU2_M1TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' +
STRING(AU2_M1TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY : ', actual_position
print,''

;;  again moving to the measured point

AU2_M1TY_commanded_position  =   AU2_M1TY_commanded_position +
step_increment
print, 'commanded position for AU2.M1TY after coming back to the measurement
position is : ', AU2_M1TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' +
STRING(AU2_M1TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY: ', actual_position
print,''

;; measure position with auto-collimation telescope

cmd = './act_meas.sh meas'
print, 'data from ACT when the actuator moved backward and comes back to the
measurement position are :'
```

```
SPAWN, cmd, ret
wait,1
tmp = STRSPLIT(ret, ',', /EXTRACT)
pos_back_AU2_M1TY_x = DOUBLE(tmp[4])
pos_back_AU2_M1TY_y = DOUBLE(tmp[3])
PRINT,'pos_back_AU2.M1TY.x', pos_back_AU2_M1TY_x
PRINT,'pos_back_AU2.M1TY.y', pos_back_AU2_M1TY_y
print,''

wait, 1

;;  for moving the measurement position by 0.5 mm in forward direction

AU2_M1TY_commanded_position  =   AU2_M1TY_commanded_position +
step_increment
print, 'commanded position for AU2.M1TY for moving in forward direction by +0.5
mm is ',  AU2_M1TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' +
STRING(AU2_M1TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY : ', actual_position
print,''

;; coming back to the measurement point

AU2_M1TY_commanded_position  =   AU2_M1TY_commanded_position -
step_increment
print, 'commanded position for AU2.M1TY after coming back to the measurement
position is : ', AU2_M1TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' +
STRING(AU2_M1TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY: ', actual_position
print,''

;; again measure the new position with auto-collimation telescope

cmd = './act_meas.sh meas'
```

```
print, 'data from ACT when the actuator moved forward and comes back to the
measurement position are : '
SPAWN, cmd, ret
wait,1
tmp = STRSPLIT(ret, ',', /EXTRACT)
pos_forw_AU2_M1TY_x = DOUBLE(tmp[4])
pos_forw_AU2_M1TY_y = DOUBLE(tmp[3])
PRINT,' pos_forw_AU2.M1TY.x',pos_forw_AU2_M1TY_x
PRINT,'pos_forw_AU2.M1TY.y', pos_forw_AU2_M1TY_y
print,''
wait,1
printf, lun ,AU2_M1TY_commanded_position ,M2TY(loop_point)
,pos_back_AU2_M1TY_x ,pos_back_AU2_M1TY_y ,pos_forw_AU2_M1TY_x
,pos_forw_AU2_M1TY_y ,FORMAT='(6F12.6)'

endfor
wait,1
endfor
close,lun

print,''
print,''
print,'Finding the backlash value for the AU2.M2TY '

;; open file

filename='AU2.M2TY'+'_'+strcompress(string(M2TYstart,format='(F5.2)'),/remove_al
l)+'.dat'
openw,lun1,filename , /get_lun ;,/append

count=0

for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU2.M1TX and AU2.M2TX to M1X0 and M2X0 position respectively-

print,'nominal position for AU2.M1TX is : ', M1X0
print,''
print,'nominal position for AU2.M2TX is : ', M2X0
print,''
print,'moving AU2.M1TX and AU2.M2TX to M1X0 and M2X0 position respectively'
print,''
print,'measurement position for AU2.M1TX is :',M1X0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' + STRING(M1X0,
FORMAT='(F5.2)') + ')"'
```

```
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''
print,'measurement position for AU2.M2TX is :',M2X0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' + STRING(M2X0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
print,'measurement position for AU2.M1TY is :',M1TY(loop_point)
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' +
STRING(M1TY(loop_point), FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY is : ', actual_position
print,''
AU2_M2TY_starting_position  =  M2TY(loop_point)
print,'the measurement position for AU2.M2TY under consideration is
:',AU2_M2TY_starting_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' +
STRING(AU2_M2TY_starting_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY is : ', actual_position
print,''
print,' moving the actuator backward by n=5 times from the measurement position'

;;   moving the actuator backward by 5 times from the measurement position

count1=0
```

```
for i=0, n-1 ,1 do begin

;; moving the actuator by 0.5 mm in backward direction

count1=count1+1
print,'count1',count1
print,''
AU2_M2TY_commanded_position  =   AU2_M2TY_starting_position -
step_increment
print, 'commanded position for AU2.M2TY for moving in backward direction by -0.5
mm is : ', AU2_M2TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' +
STRING(AU2_M2TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY : ', actual_position
print,''

;;  again moving to the measured point

AU2_M2TY_commanded_position  =   AU2_M2TY_commanded_position +
step_increment
print, 'commanded position for AU2.M2TY after coming back to the measurement
position is : ', AU2_M2TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' +
STRING(AU2_M2TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU2.M2TY'

count2=0

for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU2.M2TY is -
```

```
count2=count2+1
print,'count2',count2
print,''

;;  taking 10 times back and forth measurements for AU2.M2TY
;;  for moving the measurement position by 0.5 mm in backward direction

print, 'actual position of AU2.M2TY: ', actual_position
AU2_M2TY_commanded_position  =   AU2_M2TY_starting_position -
step_increment
print, 'commanded position for AU2.M2TY for moving in backward direction by -0.5
mm is : ', AU2_M2TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' +
STRING(AU2_M2TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY : ', actual_position
print,''

;;  again moving to the measured point

AU2_M2TY_commanded_position  =   AU2_M2TY_commanded_position +
step_increment
print, 'commanded position for AU2.M2TY after coming back to the measurement
position is : ', AU2_M2TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' +
STRING(AU2_M2TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY: ', actual_position
print,''

;; measure position with auto-collimation telescope

cmd = './act_meas.sh meas'
print, 'data from ACT when the actuator moved backward and comes back to the
measurement position are :'
SPAWN, cmd, ret
wait,1
tmp = STRSPLIT(ret, ',', /EXTRACT)
pos_back_AU2_M2TY_x = DOUBLE(tmp[4])
```

```
pos_back_AU2_M2TY_y = DOUBLE(tmp[3])
PRINT,'pos_back_AU2.M2TY.x', pos_back_AU2_M2TY_x
PRINT,'pos_back_AU2.M2TY.y', pos_back_AU2_M2TY_y
print,''
wait, 1

;;  for moving the measurement position by 0.5 mm in forward direction

AU2_M2TY_commanded_position  =   AU2_M2TY_commanded_position +
step_increment
print, 'commanded position for AU2.M2TY for moving in forward direction by +0.5
mm is ',  AU2_M2TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' +
STRING(AU2_M2TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY is : ', actual_position
print,''

;; coming back to the measurement point

AU2_M2TY_commanded_position  =   AU2_M2TY_commanded_position -
step_increment
print, 'commanded position for AU2.M2TY after coming back to the measurement
position is : ', AU2_M2TY_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' +
STRING(AU2_M2TY_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY is : ', actual_position
print,''

;; again measure the new position with auto-collimation telescope

cmd = './act_meas.sh meas'
print, 'data from ACT when the actuator moved forward and comes back to the
measurement position are : '
SPAWN, cmd, ret
wait,1
tmp = STRSPLIT(ret, ',', /EXTRACT)
pos_forw_AU2_M2TY_x = DOUBLE(tmp[4])
```

```
pos_forw_AU2_M2TY_y = DOUBLE(tmp[3])
PRINT,' pos_forw_AU2.M2TY.x',pos_forw_AU2_M2TY_x
PRINT,'pos_forw_AU2.M2TY.y', pos_forw_AU2_M2TY_y
print,''
wait,1
printf, lun1 ,AU2_M2TY_commanded_position ,M1TY(loop_point)
,pos_back_AU2_M2TY_x ,pos_back_AU2_M2TY_y ,pos_forw_AU2_M2TY_x
,pos_forw_AU2_M2TY_y ,FORMAT='(6F12.6)'

endfor
wait,1
endfor
close,lun1

END
```

****

CODING 6

```
PRO summit_read_test, M1TXstart, M2TXstart, M1TYstart, M2TYstart,
number_points, number_steps

data_M1TX =    fltarr(6,number_steps,number_points)
data_M2TX =    fltarr(6,number_steps,number_points)
data_M1TY =    fltarr(6,number_steps,number_points)
data_M2TY =    fltarr(6,number_steps,number_points)

data_MX   =    fltarr(5,number_points)
filename  =    'x.tab'
openr, lun1, filename, /GET_LUN
readf,lun1,data_MX
close, lun1

data_MY   =    fltarr(5,number_points)
filename  =    'y.tab'
openr, lun2, filename, /GET_LUN
readf,lun2,data_MY
close, lun2

filename='AU2.M1TX_real_'+strcompress(string(M1TXstart,format='(F5.2)'),/remove
_all)+'.dat'
openr, lun3, filename, /GET_LUN
readf,lun3,data_M1TX
close, lun3

filename='AU2.M2TX_real_'+strcompress(string(M2TXstart,format='(F5.2)'),/remove
_all)+'.dat'
openr, lun4, filename, /GET_LUN
```

```
readf,lun4,data_M2TX
close, lun4

filename='AU2.M1TY_'+strcompress(string(M1TYstart,format='(F5.2)'),/remove_all)
+'.dat'
openr, lun5, filename, /GET_LUN
readf,lun5,data_M1TY
close, lun5

filename='AU2.M2TY_'+strcompress(string(M2TYstart,format='(F5.2)'),/remove_all)
+'.dat'
openr, lun6, filename, /GET_LUN
readf,lun6,data_M2TY
close, lun6

diff_M1TX_x    =  fltarr(number_steps)
diff_M1TX_y    =  fltarr(number_steps)
backlash_M1TX  =  fltarr(number_points)
stddev_M1TX    =  fltarr(number_points)
diff_M1TX      =  fltarr(number_steps)
sum_M1TX       =  0
fitting_M1TX   =  fltarr(number_points)

diff_M2TX_x    =  fltarr(number_steps)
diff_M2TX_y    =  fltarr(number_steps)
backlash_M2TX  =  fltarr(number_points)
stddev_M2TX    =  fltarr(number_points)
diff_M2TX      =  fltarr(number_steps)
sum_M2TX       =  0
fitting_M2TX   =  fltarr(number_points)

diff_M1TY_x    =  fltarr(number_steps)
diff_M1TY_y    =  fltarr(number_steps)
backlash_M1TY  =  fltarr(number_points)
stddev_M1TY    =  fltarr(number_points)
diff_M1TY      =  fltarr(number_steps)
sum_M1TY       =  0
fitting_M1TY   =  fltarr(number_points)

diff_M2TY_x    =  fltarr(number_steps)
diff_M2TY_y    =  fltarr(number_steps)
backlash_M2TY  =  fltarr(number_points)
stddev_M2TY    =  fltarr(number_points)
diff_M2TY      =  fltarr(number_steps)
sum_M2TY       =  0
fitting_M2TY   =  fltarr(number_points)
for i=0,number_points-1,1 do begin

for loop_step=0, number_steps-1,1 do begin
```

```
diff_M1TX_x  =  data_M1TX[2,loop_step,i]-data_M1TX[4,loop_step,i]
diff_M1TX_y  =  data_M1TX[3,loop_step,i]-data_M1TX[5,loop_step,i]
diff_M1TX(loop_step)    =  sqrt(diff_M1TX_x^2+diff_M1TX_y^2)

diff_M2TX_x  =  data_M2TX[2,loop_step,i]-data_M2TX[4,loop_step,i]
diff_M2TX_y  =  data_M2TX[3,loop_step,i]-data_M2TX[5,loop_step,i]
diff_M2TX(loop_step)    =  sqrt(diff_M2TX_x^2+diff_M2TX_y^2)

diff_M1TY_x  =  data_M1TY[2,loop_step,i]-data_M1TY[4,loop_step,i]
diff_M1TY_y  =  data_M1TY[3,loop_step,i]-data_M1TY[5,loop_step,i]
diff_M1TY(loop_step)    =  sqrt(diff_M1TY_y^2+diff_M1TY_x^2)

diff_M2TY_x  =  data_M2TY[2,loop_step,i]-data_M2TY[4,loop_step,i]
diff_M2TY_y  =  data_M2TY[3,loop_step,i]-data_M2TY[5,loop_step,i]
diff_M2TY(loop_step)    =  sqrt(diff_M2TY_x^2+diff_M2TY_y^2)

endfor

stddev_M1TX(i)   =  sqrt(variance(diff_M1TX))
sum_M1TX         =  avg(diff_M1TX)
backlash_M1TX(i) =  sum_M1TX/2

stddev_M2TX(i)   =  sqrt(variance(diff_M2TX))
sum_M2TX         =  avg(diff_M2TX)
backlash_M2TX(i) =  sum_M2TX/2

stddev_M1TY(i)   =  sqrt(variance(diff_M1TY))
sum_M1TY         =  avg(diff_M1TY)
backlash_M1TY(i) =  sum_M1TY/2

stddev_M2TY(i)   =  sqrt(variance(diff_M2TY))
sum_M2TY         =  avg(diff_M2TY)
backlash_M2TY(i) =  sum_M2TY/2

endfor

;; finding the fitting model coefficients for all the actuators for fnding the backlash at
any position

;;backlash_M1TX_interpol=backlash_M1TX(18:26)
;;M1TX_interpol=10+findgen(20)*0.2
backlash_M1TX_interpol=backlash_M1TX(21:23)
M1TX_interpol=11.5+findgen(11)*0.1
;;fitting_M1TX_interpol=INTERPOL(backlash_M1TX_interpol,data_MX[1,18:26],M1
TX_interpol)
fitting_M1TX_interpol=INTERPOL(backlash_M1TX_interpol,data_MX[1,21:23],M1
TX_interpol)
```

```
backlash_M2TX_interpol=backlash_M2TX(21:24)
M2TX_interpol=11.55+findgen(11)*0.1
fitting_M2TX_interpol=INTERPOL(backlash_M2TX_interpol,data_MX[3,21:24],M2
TX_interpol)

backlash_M1TY_interpol=backlash_M1TY(18:26)
M1TY_interpol=10+findgen(20)*0.2
fitting_M1TY_interpol=INTERPOL(backlash_M1TY_interpol,data_MY[2,18:26],M1
TY_interpol)

fitting_M1TX = POLY_FIT (data_MX[1,*], backlash_M1TX,6, MEASURE_ERRORS
=stddev_M1TX,STATUS=status_M1TX,YBAND=YBAND_M1TX,YERROR=yerror_
M1TX,YFIT=yfit_M1TX)

fitting_M2TX = LINFIT(data_MX[3,*], backlash_M2TX, MEASURE_ERRORS
=stddev_M2TX,PROb=prob_M2TX)

fitting_M1TY = POLY_FIT(data_MY[2,*],backlash_M1TY,6, MEASURE_ERRORS
=stddev_M1TY,STATUS=status_M1TY,YBAND=YBAND_M1TY,YERROR=yerror_
M1TY,YFIT=yfit_M1TY)

fitting_M2TY = POLY_FIT(data_MY[4,*],backlash_M2TY,9, MEASURE_ERRORS
=stddev_M2TY,STATUS=status_M2TY,YBAND=YBAND_M2TY,YERROR=yerror_
M2TY,YFIT=yfit_M2TY)

print,'interpolated values for M1TX',fitting_M1TX
print,'interpolated values for M1TY',fitting_M1TY

;; curve fitting model equation for all the actuators

;; for M1TX actuator (4th order equation)

a_M1TX=fitting_M1TX[0]
b_M1TX=fitting_M1TX[1]
c_M1TX=fitting_M1TX[2]
d_M1TX=fitting_M1TX[3]
e_M1TX=fitting_M1TX[4]

fitted_backlash_M1TX=e_M1TX*data_MX[1,*]^4+d_M1TX*data_MX[1,*]^3+c_M1T
X*data_MX[1,*]^2+b_M1TX*data_MX[1,*]^1+a_M1TX
error_M1TX=backlash_M1TX-yfit_M1TX

;; for M2TX (1st order equation)

a_M2TX  = fitting_M2TX[0]
b_M2TX  = fitting_M2TX[1]

fitted_backlash_M2TX=a_M2TX+b_M2TX*data_MX[3,*]
error_M2TX=backlash_M2TX-fitted_backlash_M2TX
```

```
;; for M1TY(4th order equation)

a_M1TY=fitting_M1TY[0]
b_M1TY=fitting_M1TY[1]
c_M1TY=fitting_M1TY[2]
d_M1TY=fitting_M1TY[3]
e_M1TY=fitting_M1TY[4]

fitted_backlash_M1TY=e_M1TY*data_MY[2,*]^4+d_M1TY*data_MY[2,*]^3+c_M1T
Y*data_MY[2,*]^2+b_M1TY*data_MY[2,*]^1+a_M1TY
error_M1TY=backlash_M1TY-yfit_M1TY

;; for M2TY(4th order equation)

a_M2TY=fitting_M2TY[0]
b_M2TY=fitting_M2TY[1]
c_M2TY=fitting_M2TY[2]
d_M2TY=fitting_M2TY[3]
e_M2TY=fitting_M2TY[4]

fitted_backlash_M2TY=e_M2TY*data_MY[4,*]^4+d_M2TY*data_MY[4,*]^3+c_M2T
Y*data_MY[4,*]^2+b_M2TY*data_MY[4,*]^1+a_M2TY
error_M2TY=backlash_M2TY-yfit_M2TY

print,'backlash_M1TX (in arcsecond)',backlash_M1TX
print,''
print,'stddev_M1TX (in arcsecond)',stddev_M1TX
print,''
print,'coefficients of the 4th polynomial equation for M1TX
(ex^4+dx^3+cx^2+bx+a)',fitting_M1TX
print,''
print,'status_M1TX (0: for successful completion)',status_M1TX
print,''
print,'standard deviation error estimate for each point (M1TX):',yband_M1TX
print,''
print,'error_M1TX',error_M1TX
print,''
print,'backlash_M2TX (in arcsecond)',backlash_M2TX
print,''
print,'stddev_M2TX (in arcsecond)',stddev_M2TX
print,''
print,'a_M2TX and b_M2TX (y=A+B*x)',fitting_M2TX
print,''
print,'probability of good fitting for M2TX(if <0.1, the model parameters are
good):',prob_M2TX
print,''
print,'error_M2TX:',error_M2TX
print,''
```

```
print,'backlash_M1TY (in arcsecond)',backlash_M1TY
print,''
print,'stddev_M1TY (in arcsecond)',stddev_M1TY
print,''
print,'coefficients of the 4th polynomial equation for M1TY
(ex^4+dx^3+cx^2+bx+a)',fitting_M1TY
print,''
print,'status_M1TY (0: for successful completion)',status_M1TY
print,''
print,'standard deviation error estimate for each point (M1TY):',yband_M1TY
print,''
print,'error_M1TY',error_M1TY
print,''
print,'backlash_M2TY (in arcsecond)',backlash_M2TY
print,''
print,'stddev_M2TY (in arcsecond)',stddev_M2TY
print,''
print,'coefficients of the 4th polynomial equation for M2TY
(ex^4+dx^3+cx^2+bx+a)',fitting_M2TY
print,''
print,'status_M2TY (0: for successful completion)',status_M2TY
print,''
print,'standard deviation error estimate for each point (M2TY):',yband_M2TY
print,''
print,'error_M2TY',error_M2TY
print,''
print,'bench backlash-fitting backlash (arcseconds)',back-fitting_M1TX_interpol
print,''

restore,filename='/home/ao/testdata/au_backlash/AU2/bench_backlashM1M2X.sav'
bench_backlash_M1TX=bench_backlash_M1TX
bench_backlash_M2TX=bench_backlash_M2TX

restore,filename='/home/ao/testdata/au_backlash/AU2/compensated_backlash.sav'
s_M1X=s_M1X
s_M2X=s_M2X
compensated_backlash_M1TX=compensated_backlash_M1TX
compensated_backlash_M2TX=compensated_backlash_M2TX

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU2.M1TX_actuator.ps', /INCHES
plot,data_MX[1,*],backlash_M1TX,PSYM=-5,linestyle=1,$
TITLE='Backlash Measurement for AU2.M1TX (in arcseconds)',$
XTITLE='AU2.M1TX actuator position (in mm)',$
YTITLE='Measurement in X plane (arcseconds)'
oplot,data_MX[1,*],yfit_M1TX

DEVICE, /CLOSE
```

```
SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'fitting_backlash_for_AU2.M1TX_actuator.ps',
/INCHES
plot,data_MX[1,21:23],backlash_M1TX_interpol,PSYM=-5,linestyle=1,$
TITLE='Backlash Measurement for AU2.M1TX (in arcseconds)',$
XTITLE='AU2.M1TX actuator position (in mm)',$
YTITLE='Measurement in X plane (arcseconds)',YRANGE=[-5,50]
oplot,M1TX_interpol,fitting_M1TX_interpol,PSYM=2
oplot,M1TX_interpol,bench_backlash_M1TX,PSYM=-4
oplot,s_M1X,compensated_backlash_M1TX,PSYM=-6
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU2.M2TX_actuator.ps', /INCHES
loadct,5
plot,data_MX[3,*],backlash_M2TX,PSYM=-2,color=35,linestyle=1,  $
TITLE='Backlash Measurement for AU2.M2TX (in arcseconds)',$
XTITLE='AU2.M2TX actuator position (in mm)',$
YTITLE='Measurement in X plane (arcseconds)'
oplot,data_MX[3,*],fitted_backlash_M2TX
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'fitting_backlash_for_AU2.M2TX_actuator.ps',
/INCHES
plot,data_MX[1,21:24],backlash_M2TX_interpol,PSYM=-5,linestyle=1,$
TITLE='Backlash Measurement for AU2.M2TX (in arcseconds)',$
XTITLE='AU2.M2TX actuator position (in mm)',$
YTITLE='Measurement in X plane (arcseconds)',YRANGE=[-5,50]
oplot,M2TX_interpol,fitting_M2TX_interpol,PSYM=2
oplot,M2TX_interpol,bench_backlash_M2TX,PSYM=-4
oplot,s_M2X,compensated_backlash_M2TX,PSYM=-6
DEVICE, /CLOSE

backlash_M1TY=backlash_M1TY(3:number_points-1)
yfit_M1TY=yfit_M1TY(3:number_points-1)

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU2.M1TY_actuator.ps', /INCHES
plot,data_MY[2,3:number_points-1],backlash_M1TY,PSYM=-2,linestyle=1,$
TITLE='Backlash Measurement for AU2.M1TY (in arcseconds)',$
XTITLE='AU2.M1TY actuator position (in mm)',$
YTITLE='Measurement in Y plane (arcseconds)'
oplot,data_MY[2,3:number_points-1],yfit_M1TY
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'fitting_backlash_for_AU2.M1TY_actuator.ps',
/INCHES
```

```
plot,data_MY[2,18:26],backlash_M1TY_interpol,PSYM=-5,linestyle=1,$
TITLE='Backlash Measurement for AU2.M1TY (in arcseconds)',$
XTITLE='AU2.M1TY actuator position (in mm)',$
YTITLE='Measurement in Y plane (arcseconds)'
oplot,M1TY_interpol,fitting_M1TY_interpol,PSYM=2
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU2.M2TX_actuator.ps', /INCHES
loadct,5
plot,data_MX[3,*],backlash_M2TX,PSYM=-2,color=35,linestyle=1,  $
TITLE='Backlash Measurement for AU2.M2TX (in arcseconds)',$
XTITLE='AU2.M2TX actuator position (in mm)',$
YTITLE='Measurement in X plane (arcseconds)'
oplot,data_MX[3,*],fitted_backlash_M2TX
DEVICE, /CLOSE

backlash_M2TY=backlash_M2TY(3:number_points-2)
yfit_M2TY=yfit_M2TY(3:number_points-2)

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU2.M2TY_actuator.ps', /INCHES
plot,data_MY[4,3:number_points-2],backlash_M2TY,PSYM=-2,linestyle=1,$
TITLE='Backlash Measurement for AU2.M2TY (in arcseconds)',$
XTITLE='AU2.M2TY actuator position (in mm)',$
YTITLE='Measurement in Y plane (arcseconds)'
oplot,data_MY[4,3:number_points-2],yfit_M2TY
DEVICE, /CLOSE

end
```

****

TASK 3

CODING 7

```
PRO summit_bench_M1M2_TX,number_steps,T,n
step_increment =  0.1

;; linear approximation between AU2 mirror angle (both M1 and M2) and the
actuator position

pos_M1X=[0.767,3.108,5.455,7.803,10.151,12.50,14.841,17.174,19.495,21.799,24.081]
mirror_AU2_M1X=[-9.892505,-7.914375,-5.935825,-3.95773,-1.97981,3.5e-
05,1.97981,3.95773,5.935825,7.914375,9.892505]
```

```
pos_M2X=[0.5568,2.9510,5.3485,7.7398,10.1244,12.50,14.8556,17.1904,19.4976,21.7
697,24.0015]
mirror_AU2_M2X=[-9.892505,-7.914375,-5.935825,-3.95773,-1.97981,3.5e-
05,1.97981,3.95773,5.935825,7.914375,9.892505]
number_points=n_elements(mirror_AU2_M1X)
s_M1X=11.5+findgen(number_points)*step_increment
print,'s_M1X',s_M1X
s_M2X=11.55+findgen(number_points)*step_increment
print,'s_M2X',s_M2X
approx_AU2_M1X=fltarr(number_points)
estim_AU2_M1X = LINFIT(pos_M1X,mirror_AU2_M1X,PROb=prob_M1X)
a_M1X  = estim_AU2_M1X[0]
b_M1X  = estim_AU2_M1X[1]
approx_AU2_M1X=a_M1X+b_M1X*s_M1X
print,'a and b(coefficients of the linear equation for M1X)',a_M1X,b_M1X
print,approx_AU2_M1X
approx_AU2_M2X=fltarr(number_points)
estim_AU2_M2X = LINFIT(pos_M2X,mirror_AU2_M2X,PROb=prob_M2X)
a_M2X  = estim_AU2_M2X[0]
b_M2X  = estim_AU2_M2X[1]
approx_AU2_M2X=a_M2X+b_M2X*s_M2X
print,'a and b (coefficients of the linera equation for M2X)',a_M2X,b_M2X
print,approx_AU2_M2X

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'approx_for_AU2.M1TX_actuator.ps', /INCHES
plot,s_M1X,approx_AU2_M1X,PSYM=-2,color=35,linestyle=1,  $
TITLE='Linear approximation for AU2.M1TX (in degrees)',$
XTITLE='AU2.M1TX actuator position (in mm)',$
YTITLE='corresponding tilt angle in X plane (in degrees)'
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'approx_for_AU2.M2TX_actuator.ps', /INCHES
plot,s_M2X,approx_AU2_M2X,PSYM=-2,color=35,linestyle=1,  $
TITLE='Linear approximation for AU2.M2TX (in degrees)',$
XTITLE='AU2.M2TX actuator position (in mm)',$
YTITLE='corresponding tilt angle in X plane (in degrees)'
DEVICE, /CLOSE

;; program for backlash measurement for AU2.M1X (aroud the nominal position)
after the integration of both AU1 and AU2 at the optical bench

;; s_M1X and s_M2X are the actuator positions of M1X and M2X respectively

M1X0= 12.0000  ; mm
M2X0= 11.75000
M1Y0=12.63000
M2Y0=12.43000
```

```
count=0;
for loop_point=4, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU2.M1TY is : ', M1Y0
print,''
print,'nominal position for AU2.M2TY is : ', M2Y0
print,''
print,'moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively'
print,''
print,'measurement position for AU2.M1TY is :',M1Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' + STRING(M1Y0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY is : ', actual_position
print,''
print,'measurement position for AU2.M2TY is :',M2Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' + STRING(M2Y0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY is : ', actual_position
print,''
print,'measurement position for AU2.M2TX is :',s_M2X(loop_point)
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(s_M2X(loop_point), FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
AU2_M1TX_starting_position  =    s_M1X(loop_point)
print,'the measurement position for AU2.M1TX under consideration is :',
AU2_M1TX_starting_position
```

```
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_starting_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''
print,' moving the actuator backward by n=5 times from the measurement position'

;;   moving the actuator backward by 5 times from the measurement position

count1=0

for i=0, n-1 ,1 do begin

;; moving the actuator by 0.1 mm in backward direction

count1=count1+1
print,'count1',count1
print,''
AU2_M1TX_commanded_position  =   AU2_M1TX_starting_position -
step_increment
print, 'commanded position for AU2.M1TX for moving in backward direction by -0.1
mm is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX : ', actual_position
print,''

;;  again moving to the measured point

AU2_M1TX_commanded_position  =   AU2_M1TX_commanded_position +
step_increment
print, 'commanded position for AU2.M1TX after coming back to the measurement
position is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
```

```
print, 'actual position of AU2.M1TX: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU2.M1TX'

count2=0

for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU2.M1TX is -

count2=count2+1
print,'count2',count2
print,''

;;  taking 10 times back and forth measurements for AU2.M1TX
;;  for moving the measurement position by 0.1 mm in backward direction

print, 'actual position of AU2.M1TX: ', actual_position
AU2_M1TX_commanded_position  =   AU2_M1TX_starting_position -
step_increment
print, 'commanded position for AU2.M1TX for moving in backward direction by -0.1
mm is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX : ', actual_position
print,''

 ;;  again moving to the measured point

AU2_M1TX_commanded_position  =   AU2_M1TX_commanded_position +
step_increment
print, 'commanded position for AU2.M1TX after coming back to the measurement
position is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX: ', actual_position
```

```
print,''

;; measuring the pixel positions on the CCD camera for the backward motion

cmd='aqcam 2 dir /home/ao/testdata/au_backlash/backlash_AU2-M1x'
spawn,cmd,ret
cmd='aqcam 2 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 obj '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_back'
spawn,cmd,ret
cmd='aqcam 2 fp '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 2 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+10

;;  for moving the measurement position by 0.1 mm in forward direction

AU2_M1TX_commanded_position  =   AU2_M1TX_commanded_position +
step_increment
print, 'commanded position for AU2.M1TX for moving in forward direction by +0.1
mm is ',  AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; coming back to the measurement point

AU2_M1TX_commanded_position  =   AU2_M1TX_commanded_position -
step_increment
print, 'commanded position for AU2.M1TX after coming back to the measurement
position is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''
```

```
;; measuring the pixel positions on the CCD camera for the forward  motion

cmd='aqcam 2 obj '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw'
spawn,cmd,ret
cmd='aqcam 2 fp '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 2 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+10

endfor

wait,1

endfor

;; program for backlash measurement for AU2.M2X (aroud the nominal position)
after the integration of both AU1 and AU2 at the optical bench

;; s_M1X and s_M2X are the actuator positions of M1X and M2X respectively

M1X0= 12.0000  ; mm
M2X0= 11.75000
M1Y0=12.63000
M2Y0=12.43000

print,''
print,'Finding the backlash value for AU2.M2TX'

count=0

for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU2.M1TY is : ', M1Y0
print,''
print,'nominal position for AU2.M2TY is : ', M2Y0
print,''
print,'moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively'
print,'measurement position for AU2.M1TY is :',M1Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' + STRING(M1Y0,
FORMAT='(F5.2)') + ')"'
```

```
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY is : ', actual_position
print,''
print,'measurement position for AU2.M2TY is :',M2Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' + STRING(M2Y0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY is : ', actual_position
print,''
print,'measurement position for AU2.M1TX is :',s_M1X(loop_point)
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(s_M1X(loop_point), FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''
AU2_M2TX_starting_position  =    s_M2X(loop_point)
print,'the measurement position for AU2.M2TX under consideration is :' ,
AU2_M2TX_starting_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_starting_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
print,' moving the actuator backward by n=5 times from the measurement position'

;;   moving the actuator backward by 5 times from the measurement position

count1=0

for i=0, n-1 ,1 do begin

;; moving the actuator by 0.1 mm in backward direction
```

```
count1=count1+1
print,'count1',count1
print,''
AU2_M2TX_commanded_position  =  AU2_M2TX_starting_position -
step_increment
print, 'commanded position for AU2.M2TX for moving in backward direction by -0.1
mm is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX : ', actual_position
print,''

;;  again moving to the measured point

AU2_M2TX_commanded_position  =  AU2_M2TX_commanded_position +
step_increment
print, 'commanded position for AU2.M2TX after coming back to the measurement
position is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU2.M2TX'
count2=0

for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU2.M2TX is -

count2=count2+1
print,'count2',count2
print,''

;;  taking 10 times back and forth measurements for AU2.M2TX
;;  for moving the measurement position by 0.1 mm in backward direction
```

```
print, 'actual position of AU2.M2TX: ', actual_position
AU2_M2TX_commanded_position  =   AU2_M2TX_starting_position -
step_increment
print, 'commanded position for AU2.M2TX for moving in backward direction by -0.1
mm is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1

cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX : ', actual_position
print,''

;;  again moving to the measured point

AU2_M2TX_commanded_position  =   AU2_M2TX_commanded_position +
step_increment
print, 'commanded position for AU2.M2TX after coming back to the measurement
position is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX: ', actual_position

;; measuring the pixel positions on the CCD camera for the backward motion

cmd='aqcam 2 dir /home/ao/testdata/au_backlash/backlash_AU2-M2x'
spawn,cmd,ret
cmd='aqcam 2 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 obj '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_back'
spawn,cmd,ret
cmd='aqcam 2 fp '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 2 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+10

;;  for moving the measurement position by 0.1 mm in forward direction
```

```
AU2_M2TX_commanded_position  =  AU2_M2TX_commanded_position +
step_increment
print, 'commanded position for AU2.M2TX for moving in forward direction by +0.1
mm is ',  AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; coming back to the measurement point

AU2_M2TX_commanded_position  =  AU2_M2TX_commanded_position -
step_increment
print, 'commanded position for AU2.M2TX after coming back to the measurement
position is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; measuring the pixel positions on the CCD camera for the forward  motion

cmd='aqcam 2 obj '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw'
spawn,cmd,ret
cmd='aqcam 2 fp '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 2 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+10

endfor

wait,1

endfor

end
```

CODING 8

```
PRO summit_aqcam_AU2_M1M2TX,T

;; measuring pixel scale by moving AU2.M1TX only in forward direction

M1X0= 12  ; mm
M2X0= 11.75
step_increment =  0.1
print,'measurement position for AU2.M1TX is :',M1X0-step_increment
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' + STRING(M1X0-
step_increment, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''

;; taking the measurement at 12 mm

print,'measurement position for AU2.M1TX is :',M1X0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' + STRING(M1X0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''
cmd='aqcam 2 dir /home/ao/testdata/au_backlash/AU2-M1x'
spawn,cmd,ret
cmd='aqcam 2 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+3

;; moving to the position 11.80 mm

print,'measurement position for AU2.M1TX is :',M1X0-2*step_increment
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' + STRING(M1X0-
2*step_increment, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
```

```
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''
wait,4

;; moving to the position 11.90 and taking the measurment

print,'measurement position for AU2.M1TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(actual_position+step_increment, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+3

;; taking the measurement at 12.0

print,'measurement position for AU2.M1TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(actual_position+step_increment, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2.5
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''

;; taking the measurement at 12.1

print,'measurement position for AU2.M1TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(actual_position+step_increment, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+3
```

```
;; measuring pixel scale by moving AU2.M2TX only in forward direction

M2X0= 11.75  ;; mm
step_increment =  0.1
print,'measurement position for AU2.M1TX is :',M2X0-step_increment
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' + STRING(M2X0-
step_increment, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''

;; taking the measurement at 11.75 mm

print,'measurement position for AU2.M2TX is :',M2X0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' + STRING(M2X0,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
cmd='aqcam 2 dir /home/ao/testdata/au_backlash/AU2-M2x'
spawn,cmd,ret
cmd='aqcam 2 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+3

;; moving to the position 11.55 mm

print,'measurement position for AU2.M2TX is :',M2X0-2*step_increment
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' + STRING(M2X0-
2*step_increment, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
wait,4
```

```
;; moving to the position 11.65 and taking the measurment

print,'measurement position for AU2.M2TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(actual_position+step_increment, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+3

;; taking the measurement at 11.75

print,'measurement position for AU2.M2TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(actual_position+step_increment, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2.5
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''

;; taking the measurement at 11.85

print,'measurement position for AU2.M2TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(actual_position+step_increment, FORMAT='(F5.2)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+3

end

PRO read_aqcam, M1_f1, M1_f2, M1_f3, M2_f1, M2_f2, M2_f3,number_steps,
number_points
```

```
step_increment=0.1
pos_M1X=[0.767,3.108,5.455,7.803,10.151,12.50,14.841,17.174,19.495,21.799,24.081]
mirror_AU2_M1X=[-9.892505,-7.914375,-5.935825,-3.95773,-1.97981,3.5e-
05,1.97981,3.95773,5.935825,7.914375,9.892505]

pos_M2X=[0.5568,2.9510,5.3485,7.7398,10.1244,12.50,14.8556,17.1904,19.4976,21.7
697,24.0015]
mirror_AU2_M2X=[-9.892505,-7.914375,-5.935825,-3.95773,-1.97981,3.5e-
05,1.97981,3.95773,5.935825,7.914375,9.892505]

number_points=n_elements(mirror_AU2_M1X)

s_M1X=11.5+findgen(number_points)*step_increment
print,'s_M1X',s_M1X

s_M2X=11.55+findgen(number_points)*step_increment
print,'s_M2X',s_M2X

approx_AU2_M1X=fltarr(number_points)

estim_AU2_M1X = LINFIT(pos_M1X,mirror_AU2_M1X,PROb=prob_M1X)
a_M1X  = estim_AU2_M1X[0]
b_M1X  = estim_AU2_M1X[1]

approx_AU2_M1X=a_M1X+b_M1X*s_M1X
print,'a and b(coefficients of the linear equation for M1X)',a_M1X,b_M1X
print,approx_AU2_M1X

approx_AU2_M2X=fltarr(number_points)

estim_AU2_M2X = LINFIT(pos_M2X,mirror_AU2_M2X,PROb=prob_M2X)
a_M2X  = estim_AU2_M2X[0]
b_M2X  = estim_AU2_M2X[1]

approx_AU2_M2X=a_M2X+b_M2X*s_M2X
print,'a and b (coefficients of the linera equation for M2X)',a_M2X,b_M2X
print,approx_AU2_M2X

M1_f1='12.5500_forw_000'+strcompress(string(M1_f1),/remove_all)+'.fits'
M1_f2='12.5500_forw_000'+strcompress(string(M1_f2),/remove_all)+'.fits'
M1_f3='12.5500_forw_000'+strcompress(string(M1_f3),/remove_all)+'.fits'

cmd='imfitpsf -r 3 -x 512 -y 537 /home/ao/testdata/au_backlash/AU2/AU2-M1x/'+
M1_f1
spawn,cmd,ret1
```

```
cmd='imfitpsf -r 3 -x 442 -y 536 /home/ao/testdata/au_backlash/AU2/AU2-M1x/'+
M1_f2
spawn,cmd,ret2

cmd='imfitpsf -r 2 -x 584 -y 537 /home/ao/testdata/au_backlash/AU2/AU2-M1x/'+
M1_f3
spawn,cmd,ret3

tmp1=STRSPLIT(ret1(0),'(,)', /EXTRACT)
x_pixel_M1X_f1=double(tmp1[1])
y_pixel_M1X_f1=double(tmp1[2])

tmp2=STRSPLIT(ret2(0),'(,)', /EXTRACT)
x_pixel_M1X_f2=double(tmp2[1])
y_pixel_M1X_f2=double(tmp2[2])

tmp3=STRSPLIT(ret3(0),'(,)', /EXTRACT)
x_pixel_M1X_f3=double(tmp3[1])
y_pixel_M1X_f3=double(tmp3[2])

print,'ret1(0)for the position 12.0:',ret1(0)
print,'x_pixel_M1X_f1',x_pixel_M1X_f1
print,'y_pixel_M1X_f1',y_pixel_M1X_f1

print,'ret2(0)for the position 11.9:',ret2(0)
print,'x_pixel_M1X_f2',x_pixel_M1X_f2
print,'y_pixel_M1X_f2',y_pixel_M1X_f2

print,'ret3(0)for the position 12.1:',ret3(0)
print,'x_pixel_M1X_f3',x_pixel_M1X_f3
print,'y_pixel_M1X_f3',y_pixel_M1X_f3

pos_M1X=[12.0,11.9,12.1]
theta_M1X=[-0.398503,-0.483227,-0.313780]

print,'theta_M1X (in degree): ', theta_M1X
print,'theta1_M1X-theta0_M1X (in arcminute)',(theta_M1X(1)-theta_M1X(0))*60
print,'theta2_M1X-theta0_M1X (in arcminute)',(theta_M1X(2)-theta_M1X(0))*60
print,''
print,'x_pixel_M1X_f2-x_pixel_M1X_f1 (pixel scale)',x_pixel_M1X_f2-x_pixel_M1X_f1
print,'x_pixel_M1X_f3-x_pixel_M1X_f1 (pixel scale)',x_pixel_M1X_f3-x_pixel_M1X_f1

theta_M1X=theta_M1X*3600
pixel_scale_back_M1X=(theta_M1X(1)-theta_M1X(0))/sqrt((x_pixel_M1X_f2-
x_pixel_M1X_f1)^2+(y_pixel_M1X_f2-y_pixel_M1X_f1)^2)
pixel_scale_forw_M1X=(theta_M1X(2)-theta_M1X(0))/sqrt((x_pixel_M1X_f3-
x_pixel_M1X_f1)^2+(y_pixel_M1X_f3-y_pixel_M1X_f1)^2)

print,''
```

```
print,'pixel_scale_back_M1X',pixel_scale_back_M1X
print,''
print,'pixel_scale_forw_M1X',pixel_scale_forw_M1X

M2_f1='12.5500_forw_000'+strcompress(string(M2_f1),/remove_all)+'.fits'
M2_f2='12.5500_forw_000'+strcompress(string(M2_f2),/remove_all)+'.fits'
M2_f3='12.5500_forw_000'+strcompress(string(M2_f3),/remove_all)+'.fits'

cmd='imfitpsf -r 3 -x 285 -y 534 /home/ao/testdata/au_backlash/AU2/AU2-M2x/'+
M2_f1
spawn,cmd,ret1

cmd='imfitpsf -r 3 -x 316 -y 535 /home/ao/testdata/au_backlash/AU2/AU2-M2x/'+
M2_f2
spawn,cmd,ret2

cmd='imfitpsf -r 3 -x 257 -y 533 /home/ao/testdata/au_backlash/AU2/AU2-M2x/'+
M2_f3
spawn,cmd,ret3

tmp1=STRSPLIT(ret1(0),'(,)', /EXTRACT)
x_pixel_M2X_f1=double(tmp1[1])
y_pixel_M2X_f1=double(tmp1[2])

tmp2=STRSPLIT(ret2(0),'(,)', /EXTRACT)
x_pixel_M2X_f2=double(tmp2[1])
y_pixel_M2X_f2=double(tmp2[2])

tmp3=STRSPLIT(ret3(0),'(,)', /EXTRACT)
x_pixel_M2X_f3=double(tmp3[1])
y_pixel_M2X_f3=double(tmp3[2])

print,'ret1(0)for the position 11.75:',ret1(0)
print,'x_pixel_M2X_f1',x_pixel_M2X_f1
print,'y_pixel_M2X_f1',y_pixel_M2X_f1

print,'ret2(0)for the position 11.65:',ret2(0)
print,'x_pixel_M2X_f2',x_pixel_M2X_f2
print,'y_pixel_M2X_f2',y_pixel_M2X_f2

print,'ret3(0)for the position 11.95:',ret3(0)
print,'x_pixel_M2X_f3',x_pixel_M2X_f3
print,'y_pixel_M2X_f3',y_pixel_M2X_f3

pos_M2X=[11.75,11.65,11.85]
theta_M2X=[-0.557390,-0.641550,-0.473229]

print,'theta_M2X (in degree): ', theta_M2X
print,'theta1_M2X-theta0_M2X (in arcminute)',(theta_M2X(1)-theta_M2X(0))*60
```

```
print,'theta2_M2X-theta0_M2X (in arcminute)',(theta_M2X(2)-theta_M2X(0))*60
print,''
print,'x_pixel_M2X_f2-x_pixel_M2X_f1 (pixel scale)',x_pixel_M2X_f2-x_pixel_M2X_f1
print,'x_pixel_M2X_f3-x_pixel_M2X_f1 (pixel scale)',x_pixel_M2X_f3-x_pixel_M2X_f1

theta_M2X=theta_M2X*3600
pixel_scale_back_M2X=(theta_M2X(1)-theta_M2X(0))/sqrt((x_pixel_M2X_f2-
x_pixel_M2X_f1)^2+(y_pixel_M2X_f2-y_pixel_M2X_f1)^2)
pixel_scale_forw_M2X=(theta_M2X(2)-theta_M2X(0))/sqrt((x_pixel_M2X_f3-
x_pixel_M2X_f1)^2+(y_pixel_M2X_f3-y_pixel_M2X_f1)^2)

print,''
print,'pixel_scale_back_M2X',pixel_scale_back_M2X
print,''
print,'pixel_scale_forw_M2X',pixel_scale_forw_M2X

;; reading the gaussian center for the AU2M1TX having backlash measured at the
optical bench

filename= 'summit_aqcamread_AU2M1TX.dat'
openw,lun,filename , /get_lun ;,/append

x_pixel_back_M1X=fltarr(number_steps)
y_pixel_back_M1X=fltarr(number_steps)
x_pixel_forw_M1X=fltarr(number_steps)
y_pixel_forw_M1X=fltarr(number_steps)

back_M1X=fltarr(number_steps)
forw_M1X=fltarr(number_steps)
back_imgM1=fltarr(number_steps)
forw_imgM1=fltarr(number_steps)

for loop_point=0,number_points-1,1 do begin

for loop_step=0,number_steps-1,1 do begin

back_M1X=strcompress(string(s_M1X(loop_point)),/remove_all)+'_back_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'

forw_M1X=strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'

back_imgM1=READFITS('/home/ao/testdata/au_backlash/AU2/backlash_AU2-
M1x/'+string(back_M1X))
forw_imgM1=READFITS('/home/ao/testdata/au_backlash/AU2/backlash_AU2-
M1x/'+string(forw_M1X))
size_backM1=size(back_imgM1)
size_forwM1=size(forw_imgM1)
```

```
smooth_backM1=smooth(back_imgM1,20,/edge_truncate)
smooth_forwM1=smooth(forw_imgM1,20,/edge_truncate)
temp1_M1=max(smooth_backM1,index1_M1)
temp2_M1=max(smooth_forwM1,index2_M1)
x_backM1=index1_M1 mod size_backM1[1]
y_backM1=index1_M1 / size_backM1[1]
x_forwM1=index2_M1 mod size_forwM1[1]
y_forwM1=index2_M1 / size_forwM1[1]
print,''
print,'index of the brightest pixel for M1X(back)',x_backM1,y_backM1
print,''
print,'index of the brightest pixel for M1X(forw)',x_forwM1,y_forwM1
print,''

;; cropping the image

crop_backM1=back_imgM1[x_backM1-10:x_backM1+10,y_backM1-10:y_backM1+10]
crop_forwM1=forw_imgM1[x_forwM1-10:x_forwM1+10,y_forwM1-10:y_forwM1+10]
fit_backM1=gauss2dfit(crop_backM1,/tilt,param_backM1)
fit_forwM1=gauss2dfit(crop_forwM1,/tilt,param_forwM1)
x_pixel_back_M1X(loop_step)=x_backM1-10+param_backM1[4]+1
y_pixel_back_M1X(loop_step)=y_backM1-10+param_backM1[5]+1
x_pixel_forw_M1X(loop_step)=x_forwM1-10+param_forwM1[4]+1
y_pixel_forw_M1X(loop_step)=y_forwM1-10+param_forwM1[5]+1

print,'gaussian x and y value for
M1X(back)',x_pixel_back_M1X(loop_step),'',y_pixel_back_M1X(loop_step)
print,''
print,'gaussian x and y value for
M1X(forw)',x_pixel_forw_M1X(loop_step),'',y_pixel_forw_M1X(loop_step)
print,''
printf,lun,s_M1X(loop_point),s_M2X(loop_point),x_pixel_back_M1X(loop_step),y_pix
el_back_M1X(loop_step),x_pixel_forw_M1X(loop_step),y_pixel_forw_M1X(loop_step),
FORMAT='(6F12.6)'

endfor

endfor

close,lun

;;reading the gaussian center for AU2M2TX having backlash, measured at the
optical bench

filename= 'summit_aqcamread_AU2M2TX.dat'
openw,lun,filename , /get_lun ;,/append

x_pixel_back_M2X=fltarr(number_steps)
y_pixel_back_M2X=fltarr(number_steps)
```

```
x_pixel_forw_M2X=fltarr(number_steps)
y_pixel_forw_M2X=fltarr(number_steps)

back_M2X=fltarr(number_steps)
forw_M2X=fltarr(number_steps)
back_imgM2=fltarr(number_steps)
forw_imgM2=fltarr(number_steps)

for loop_point=0,number_points-1,1 do begin

for loop_step=0,number_steps-1,1 do begin

back_M2X=strcompress(string(s_M2X(loop_point)),/remove_all)+'_back_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'

forw_M2X=strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'

back_imgM2=READFITS('/home/ao/testdata/au_backlash/AU2/backlash_AU2-
M2x/'+string(back_M2X))
forw_imgM2=READFITS('/home/ao/testdata/au_backlash/AU2/backlash_AU2-
M2x/'+string(forw_M2X))

size_backM2=size(back_imgM2)
size_forwM2=size(forw_imgM2)
smooth_backM2=smooth(back_imgM2,20,/edge_truncate)
smooth_forwM2=smooth(forw_imgM2,20,/edge_truncate)
temp1_M2=max(smooth_backM2,index1_M2)
temp2_M2=max(smooth_forwM2,index2_M2)
x_backM2=index1_M2 mod size_backM2[1]
y_backM2=index1_M2 / size_backM2[1]
x_forwM2=index2_M2 mod size_forwM2[1]
y_forwM2=index2_M2 / size_forwM2[1]
print,''
print,'index of the brightest pixel for M2X(back)',x_backM2,y_backM2
print,''
print,'index of the brightest pixel for M2X(forw)',x_forwM2,y_forwM2
print,''

;; cropping the image

crop_backM2=back_imgM2[x_backM2-10:x_backM2+10,y_backM2-10:y_backM2+10]
crop_forwM2=forw_imgM2[x_forwM2-10:x_forwM2+10,y_forwM2-10:y_forwM2+10]
fit_backM2=gauss2dfit(crop_backM2,/tilt,param_backM2)
fit_forwM2=gauss2dfit(crop_forwM2,/tilt,param_forwM2)
x_pixel_back_M2X(loop_step)=x_backM2-10+param_backM2[4]+1
y_pixel_back_M2X(loop_step)=y_backM2-10+param_backM2[5]+1
x_pixel_forw_M2X(loop_step)=x_forwM2-10+param_forwM2[4]+1
y_pixel_forw_M2X(loop_step)=y_forwM2-10+param_forwM2[5]+1
```

```
print,'gaussian x and y value for
M2X(back)',x_pixel_back_M2X(loop_step),'',y_pixel_back_M2X(loop_step)
print,''
print,'gaussian x and y value for
M2X(forw)',x_pixel_forw_M2X(loop_step),'',y_pixel_forw_M2X(loop_step)
print,''
printf,lun,s_M2X(loop_point),s_M1X(loop_point),x_pixel_back_M2X(loop_step),y_pix
el_back_M2X(loop_step),x_pixel_forw_M2X(loop_step),y_pixel_forw_M2X(loop_step),
FORMAT='(6F12.6)'

endfor

endfor

close,lun

;; backlash measurement for AU2M1X at the optical bench

data1_M1TX =    fltarr(6,number_steps,number_points)

filename= 'summit_aqcamread_AU2M1TX.dat'
openr,lun,filename , /get_lun
readf,lun,data1_M1TX
close,lun

diff_pixelx_M1X= fltarr(number_steps)
diff_pixely_M1X=fltarr(number_steps)
bench_backlash_M1TX=fltarr(number_points)
diff_pixel_M1X=fltarr(number_steps)
stddev_M1TX   =   fltarr(number_points)

for loop_point=0, number_points-1,1 do begin

print,'the measurement point is',data1_M1TX[0,1,loop_point]
print,''

for loop_step=0,number_steps-1,1 do begin

diff_pixelx_M1X  =  data1_M1TX[4,loop_step,loop_point]-
data1_M1TX[2,loop_step,loop_point]
diff_pixely_M1X  =  data1_M1TX[5,loop_step,loop_point]-
data1_M1TX[3,loop_step,loop_point]
diff_pixel_M1X(loop_step)   =  sqrt(diff_pixelx_M1X^2+diff_pixely_M1X^2)

print,'backlash value at every point (in arcseconds) for
AU2M1X',(diff_pixel_M1X(loop_step)*((pixel_scale_forw_M1X-
pixel_scale_back_M1X)/2))
```

```
endfor

stddev_M1TX(loop_point)   =  sqrt(variance(diff_pixel_M1X))
bench_backlash_M1TX(loop_point) = avg(diff_pixel_M1X)*((pixel_scale_forw_M1X-
pixel_scale_back_M1X)/2)

print,''
print,'averaged backlash value for AU2M1TX in arcsecond at bench
is:',bench_backlash_M1TX(loop_point)
print,''

endfor

print,''
print,'backlash value for AU2M1TX in arcsecond at bench is:',bench_backlash_M1TX
print,''
print,'backlash value for AU2M1TX in pixel at bench
is:',bench_backlash_M1TX/((pixel_scale_forw_M1X-pixel_scale_back_M1X)/2)
print,''
print,'standard deviation at each point (in pixel)',stddev_M1TX
print,''
print,'standard deviation at each point (in
arcsecond)',stddev_M1TX*((pixel_scale_forw_M1X-pixel_scale_back_M1X)/2)

data2_M1TX  =    fltarr(6,number_steps*number_points)
difference_M1X  =    fltarr(number_steps*number_points)

filename= 'summit_aqcamread_AU2M1TX.dat'
openr,lun,filename , /get_lun
readf,lun,data2_M1TX
close,lun

for i=0,(number_steps*number_points)-1,1 do begin

difference_M1X(i)=sqrt((data2_M1TX[4,i]-data2_M1TX[2,i])^2+(data2_M1TX[5,i]-
data2_M1TX[3,i])^2)

endfor

print,''

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU2.M1TX_bench_arcsecond.ps',
/INCHES
plot,data2_M1TX[0,*],difference_M1X*((pixel_scale_forw_M1X-
pixel_scale_back_M1X)/2),PSYM=1,$
TITLE='Backlash Measurement at the Optical Bench for AU2.M1TX',$
XTITLE='AU2.M1TX actuator positions around nominal point (in mm)',$
YTITLE='10 backlash measurement at each position (arcsecond)'
```

```
oplot,s_M1X,stddev_M1TX*((pixel_scale_forw_M1X-
pixel_scale_back_M1X)/2),PSYM=-6,linestyle=1
oplot,s_M1X,bench_backlash_M1TX,PSYM=-5,linestyle=1
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU2.M1TX_bench_pixel.ps', /INCHES
plot,data2_M1TX[0,*],difference_M1X,PSYM=1,$
TITLE='Backlash Measurement at the Optical Bench for AU2.M1TX',$
XTITLE='AU2.M1TX actuator positions around nominal point (in mm)',$
YTITLE='10 backlash measurement at each position (pixel)'
oplot,s_M1X,stddev_M1TX,PSYM=-6,linestyle=1
oplot,s_M1X,bench_backlash_M1TX/((pixel_scale_forw_M1X-
pixel_scale_back_M1X)/2),PSYM=-5,linestyle=1
DEVICE, /CLOSE

;; backlash measurement for AU2M2X at the optical bench

data1_M2TX =    fltarr(6,number_steps,number_points)

filename= 'summit_aqcamread_AU2M2TX.dat'
openr,lun,filename , /get_lun
readf,lun,data1_M2TX
close,lun

diff_pixelx_M2X= fltarr(number_steps)
diff_pixely_M2X=fltarr(number_steps)
bench_backlash_M2TX=fltarr(number_points)
diff_pixel_M2X=fltarr(number_steps)
stddev_M2TX   =   fltarr(number_points)

for loop_point=0, number_points-1,1 do begin

print,'the measurement point is',data1_M2TX[0,1,loop_point]
print,''

for loop_step=0,number_steps-1,1 do begin

diff_pixelx_M2X  =  data1_M2TX[4,loop_step,loop_point]-
data1_M2TX[2,loop_step,loop_point]
diff_pixely_M2X  =  data1_M2TX[5,loop_step,loop_point]-
data1_M2TX[3,loop_step,loop_point]
diff_pixel_M2X(loop_step)    =  sqrt(diff_pixelx_M2X^2+diff_pixely_M2X^2)

print,'backlash value at every point (in arcseconds) for
AU2.M2X',(diff_pixel_M2X(loop_step))*((pixel_scale_forw_M2X-
pixel_scale_back_M2X)/2)

endfor
```

```
stddev_M2TX(loop_point)   =  sqrt(variance(diff_pixel_M2X))
bench_backlash_M2TX(loop_point) = avg(diff_pixel_M2X)*((pixel_scale_forw_M2X-
pixel_scale_back_M2X)/2)

print,''
print,'averaged backlash value for AU1M2TX in arcsecond at bench
is:',bench_backlash_M2TX(loop_point)
print,''

endfor

print,''
print,'backlash value for AU2M2TX in arcsecond at bench is:',bench_backlash_M2TX
print,''
print,'backlash value for AU2M2TX in pixel at bench
is:',bench_backlash_M2TX/((pixel_scale_forw_M2X-pixel_scale_back_M2X)/2)
print,''
print,'standard deviation at each point (in pixel)',stddev_M2TX
print,''
print,'standard deviation at each point (in
arcsecond)',stddev_M2TX*((pixel_scale_forw_M2X-pixel_scale_back_M2X)/2)

data2_M2TX  =     fltarr(6,number_steps*number_points)
difference_M2X  =     fltarr(number_steps*number_points)

filename= 'summit_aqcamread_AU2M2TX.dat'
openr,lun,filename , /get_lun
readf,lun,data2_M2TX
close,lun

for i=0,(number_steps*number_points)-1,1 do begin

difference_M2X(i)=sqrt((data2_M2TX[4,i]-data2_M2TX[2,i])^2+(data2_M2TX[5,i]-
data2_M2TX[3,i])^2)

endfor

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU2.M2TX_bench_arcsecond.ps',
/INCHES
plot,data2_M2TX[0,*],difference_M2X*((pixel_scale_forw_M2X-
pixel_scale_back_M2X)/2),PSYM=1, $
TITLE='Backlash Measurement at the Optical Bench for AU2.M2TX(arcsecond)',$
XTITLE='AU2.M2TX actuator positions around nominal point (in mm)',$
YTITLE='10 backlash measurement at each position (arcsecond)'
oplot,s_M2X,stddev_M2TX*((pixel_scale_forw_M2X-
pixel_scale_back_M2X)/2),PSYM=-6,linestyle=1
xyouts,2,12,'Standard deviation for backlash',/normal
oplot,s_M2X,bench_backlash_M2TX,PSYM=-5,linestyle=1
```

```
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU2.M2TX_bench_pixel.ps', /INCHES
plot,data2_M2TX[0,*],difference_M2X,PSYM=1,$
TITLE='Backlash Measurement at the Optical Bench for AU2.M2TX',$
XTITLE='AU2.M2TX actuator positions around nominal point (in mm)',$
YTITLE='10 backlash measurement at each position (pixel)'
oplot,s_M2X,stddev_M2TX,PSYM=-6,linestyle=1
oplot,s_M2X,bench_backlash_M2TX/((pixel_scale_forw_M2X-
pixel_scale_back_M2X)/2),PSYM=-5,linestyle=1
DEVICE, /CLOSE

save,s_M1X,s_M2X,pixel_scale_forw_M1X,pixel_scale_back_M1X,pixel_scale_forw_
M2X,pixel_scale_back_M2X,bench_backlash_M1TX,bench_backlash_M2TX,filename
='/home/ao/testdata/au_backlash/AU2/bench_backlashM1M2X.sav'

end
```

****

CODING 9

```
PRO summit_bench_AU1M1M2TX,number_points,number_steps,T,n
step_increment =  0.1
s_M1X=6.5+findgen(number_points)*step_increment
print,'s_M1X',s_M1X

s_M2X=6.7+findgen(number_points)*step_increment
print,'s_M2X',s_M2X

;; program for backlash measurement for AU1.M1X (aroud the nominal position)
after the integration of both AU1 and AU2 at the optical bench

;; s_M1X and s_M2X are the actuator positions of M1X and M2X respectively

M1X0= 7.11236  ; mm
M2X0= 7.29337
M1Y0= 12.84165
M2Y0= 12.99195

print,''
print,'Finding the backlash value for AU1.M1TX'

count=0;
for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
```

```
;; moving AU1.M1TY and AU1.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU1.M1TY is : ', M1Y0
print,''
print,'nominal position for AU1.M2TY is : ', M2Y0
print,''
print,'moving AU1.M1TY and AU1.M2TY to M1Y0 and M2Y0 position respectively'
print,''
print,'measurement position for AU1.M1TY is :',M1Y0
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TY, ' + STRING(M1Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TY is : ', actual_position
print,'measurement position for AU1.M2TY is :',M2Y0
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TY, ' + STRING(M2Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TY is : ', actual_position
print,''
print,'measurement position for AU1.M2TX is :',s_M2X(loop_point)
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(s_M2X(loop_point), FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
AU1_M1TX_starting_position  =    s_M1X(loop_point)
print,'the measurement position for AU1.M1TX under consideration is :',
AU1_M1TX_starting_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_starting_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
```

```
print,' moving the actuator backward by n=5 times from the measurement position'

;;   moving the actuator backward by 5 times from the measurement position

count1=0

for i=0, n-1 ,1 do begin

;; moving the actuator by 0.1 mm in backward direction

count1=count1+1
print,'count1',count1
print,''
AU1_M1TX_commanded_position  =   AU1_M1TX_starting_position -
step_increment
print, 'commanded position for AU1.M1TX for moving in backward direction by -0.1
mm is : ', AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX : ', actual_position
print,''

;;  again moving to the measured point

AU1_M1TX_commanded_position  =   AU1_M1TX_commanded_position +
step_increment
print, 'commanded position for AU1.M1TX for coming back at the measurement
position is : ', AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU1.M1TX'

count2=0
```

```
 for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU1.M1TX is -

count2=count2+1
print,'count2',count2
print,''

;;  taking 10 times back and forth measurements for AU1.M1TX
;;  for moving the measurement position by 0.1 mm in backward direction

print, 'actual position of AU1.M1TX: ', actual_position

AU1_M1TX_commanded_position  =   AU1_M1TX_starting_position -
step_increment
print, 'commanded position for AU1.M1TX for moving in backward direction by -0.1
mm is : ', AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX : ', actual_position
print,''

 ;;  again moving to the measured point

AU1_M1TX_commanded_position  =   AU1_M1TX_commanded_position +
step_increment
print, 'commanded position for AU1.M1TX for coming back at the measurement
position is : ', AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX: ', actual_position
print,''

;; measuring the pixel positions on the CCD camera for the backward motion

cmd='aqcam 1 dir /home/ao/testdata/au_backlash/AU1/backlash_AU1M1X'
spawn,cmd,ret
cmd='aqcam 1 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
```

```
cmd='aqcam 1 obj '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_back'
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 1 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10

;;  for moving the measurement position by 0.1 mm in forward direction

AU1_M1TX_commanded_position  =   AU1_M1TX_commanded_position +
step_increment
print, 'commanded position for AU1.M1TX for moving in forward direction by +0.1
mm is ',  AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; coming back to the measurement point

AU1_M1TX_commanded_position  =   AU1_M1TX_commanded_position -
step_increment
print, 'commanded position for AU1.M1TX after coming back to the measurement
position is : ', AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; measuring the pixel positions on the CCD camera for the forward  motion

cmd='aqcam 1 obj '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw'
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
```

```
cmd='aqcam 1 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10

endfor
wait,1
endfor

;; program for backlash measurement for AU1.M2X (aroud the nominal position)
after the integration of both AU1 and AU2 at the optical bench

;; s_M1X and s_M2X are the actuator positions of M1X and M2X respectively

M1X0= 7.11236  ; mm
M2X0= 7.29337
M1Y0= 12.84165
M2Y0= 12.99195

;print,''
;print,'Finding the backlash value for AU1.M2TX'

count=0;
for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU1.M1TY and AU1.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU1.M1TY is : ', M1Y0
print,''
print,'nominal position for AU1.M2TY is : ', M2Y0
print,''
print,'moving AU1.M1TY and AU1.M2TY to M1Y0 and M2Y0 position respectively'
print,''
print,'measurement position for AU1.M1TY is :',M1Y0
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TY, ' + STRING(M1Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TY is : ', actual_position
print,''
```

```
print,'measurement position for AU1.M2TY is :',M2Y0
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TY, ' + STRING(M2Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TY is : ', actual_position
print,'measurement position for AU1.M1TX is :',s_M1X(loop_point)
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(s_M1X(loop_point), FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
AU1_M2TX_starting_position  =    s_M2X(loop_point)
print,'the measurement position for AU1.M2TX under consideration is :',
AU1_M2TX_starting_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_starting_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
print,' moving the actuator backward by n=5 times from the measurement position'

;;   moving the actuator backward by 5 times from the measurement position

count1=0

for i=0, n-1 ,1 do begin

;; moving the actuator by 0.1 mm in backward direction

count1=count1+1
print,'count1',count1
print,''
AU1_M2TX_commanded_position  =   AU1_M2TX_starting_position -
step_increment
print, 'commanded position for AU1.M2TX for moving in backward direction by -0.1
mm is : ', AU1_M2TX_commanded_position
```

```
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX : ', actual_position
print,''

;;  again moving to the measured point

AU1_M2TX_commanded_position  =   AU1_M2TX_commanded_position +
step_increment
print, 'commanded position for AU1.M2TX for coming back at the measurement
position is : ', AU1_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU1.M2TX'

count2=0

 for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU1.M2TX is -

count2=count2+1
print,'count2',count2
print,''

;;  taking 10 times back and forth measurements for AU1.M2TX
;;  for moving the measurement position by 0.1 mm in backward direction

print, 'actual position of AU1.M2TX: ', actual_position

AU1_M2TX_commanded_position  =   AU1_M2TX_starting_position -
step_increment
print, 'commanded position for AU1.M2TX for moving in backward direction by -0.1
mm is : ', AU1_M2TX_commanded_position
```

```
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX : ', actual_position
print,''

;;  again moving to the measured point

AU1_M2TX_commanded_position  =   AU1_M2TX_commanded_position +
step_increment
print, 'commanded position for AU1.M2TX for coming back at the measurement
position is : ', AU1_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX: ', actual_position
print,''

 ;; measuring the pixel positions on the CCD camera for the backward motion

cmd='aqcam 1 dir /home/ao/testdata/au_backlash/AU1/backlash_AU1M2X'
spawn,cmd,ret
cmd='aqcam 1 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 obj '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_back'
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 1 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10

;;  for moving the measurement position by 0.1 mm in forward direction

AU1_M2TX_commanded_position  =   AU1_M2TX_commanded_position +
step_increment
print, 'commanded position for AU1.M2TX for moving in forward direction by +0.1
mm is ',  AU1_M2TX_commanded_position
```

```
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; coming back to the measurement point

AU1_M2TX_commanded_position  =  AU1_M2TX_commanded_position -
step_increment
print, 'commanded position for AU1.M2TX after coming back to the measurement
position is : ', AU1_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; measuring the pixel positions on the CCD camera for the forward  motion

cmd='aqcam 1 obj '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw'
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 1 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10

endfor

wait,1

endfor

end
```

CODING 10

```
PRO pixel_scale_AU1M1M2,T

;; measuring pixel scale by moving AU2.M1TX only in forward direction

M1X0= 7.11236  ; mm
M2X0= 7.29337
step_increment =  0.1

 print,'measurement position for AU1.M1TX is :',M1X0-step_increment
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' + STRING(M1X0-
step_increment, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''

;; taking the measurement at 7.11236 mm

 print,'measurement position for AU1.M1TX is :',M1X0
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' + STRING(M1X0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
cmd='aqcam 1 dir /home/ao/testdata/au_backlash/AU1/AU1_M1X'
spawn,cmd,ret
cmd='aqcam 1 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(M1X0),/remove_all)+''
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+3

;; moving to the position 6.91236 mm

 print,'measurement position for AU1.M1TX is :',M1X0-2*step_increment
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' + STRING(M1X0-
2*step_increment, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
```

```
wait,2
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
wait,4

;; moving to the position 7.01236 and taking the measurment

print,'measurement position for AU1.M1TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(actual_position+step_increment, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
cmd='aqcam 1 fp '+strcompress(string(actual_position),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+3

;; taking the measurement at 7.11236

print,'measurement position for AU1.M1TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(actual_position+step_increment, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,2.5
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''

;; taking the measurement at 7.21236

 print,'measurement position for AU1.M1TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(actual_position+step_increment, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
```

```
print, 'actual position of AU1.M1TX is : ', actual_position
cmd='aqcam 1 fp '+strcompress(string(actual_position),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+3

;; measuring pixel scale by moving AU1.M2TX only in forward direction

M2X0= 7.29337  ;; mm

step_increment =  0.1
print,'measurement position for AU1.M2TX is :',M2X0-step_increment
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' + STRING(M2X0-
step_increment, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''

;; taking the measurement at 7.29337 mm

print,'measurement position for AU1.M2TX is :',M2X0
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' + STRING(M2X0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
cmd='aqcam 1 dir /home/ao/testdata/au_backlash/AU1/AU1_M2X'
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(M2X0),/remove_all)+'_'
spawn,cmd,ret
cmd='aqcam 1 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+3

;; moving to the position 7.09337 mm

print,'measurement position for AU1.M2TX is :',M2X0-2*step_increment
```

```
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' + STRING(M2X0-
2*step_increment, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,2

cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
wait,4

;; moving to the position 7.19337 and taking the measurment

print,'measurement position for AU2.M2TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(actual_position+step_increment, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,2

cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
cmd='aqcam 1 fp '+strcompress(string(actual_position),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+3

;; taking the measurement at 7.29337

print,'measurement position for AU1.M2TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(actual_position+step_increment, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,2.5
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''

;; taking the measurement at 7.39337

print,'measurement position for AU1.M2TX is :',actual_position+step_increment
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(actual_position+step_increment, FORMAT='(F9.6)') + ')"'
```

```
SPAWN, cmd, ret
wait,2
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
cmd='aqcam 1 fp '+strcompress(string(actual_position),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+3

end

;*****************************************************************************

PRO read_aqcam_AU1M1M2X,number_steps,number_points

;; pixel scale calculation for AU1M1TX

theta_M1X=fltarr(number_points)
theta_M2X=fltarr(number_points)

step_increment=0.1
s_M1X=6.5+findgen(number_points)*step_increment
print,'s_M1X',s_M1X
s_M2X=6.7+findgen(number_points)*step_increment
print,'s_M2X',s_M2X

;; tilt angle corresponding to the actuator position

theta_M1X = 1.390580e-06*s_M1X^4 + 2.812652e-06*s_M1X^3 + 1.259884e-
04*s_M1X^2 + 9.299078e-01*s_M1X - 1.168302e+01

cmd='imfitpsf -r 2 -x 498 -y 522
/home/ao/testdata/au_backlash/AU1/AU1_M1X/7.1123600030.fits'
spawn,cmd,ret1
cmd='imfitpsf -r 2 -x 427.5 -y 521.5
/home/ao/testdata/au_backlash/AU1/AU1_M1X/7.0123606_back_00031.fits'
spawn,cmd,ret2
cmd='imfitpsf -r 2 -x 568 -y 523
/home/ao/testdata/au_backlash/AU1/AU1_M1X/7.2123545_forw_00032.fits'
spawn,cmd,ret3
tmp1=STRSPLIT(ret1(0),'(,)', /EXTRACT)
x_pixel_M1X_f1=double(tmp1[1])
y_pixel_M1X_f1=double(tmp1[2])

tmp2=STRSPLIT(ret2(0),'(,)', /EXTRACT)
```

```
x_pixel_M1X_f2=double(tmp2[1])
y_pixel_M1X_f2=double(tmp2[2])

tmp3=STRSPLIT(ret3(0),'(,)', /EXTRACT)
x_pixel_M1X_f3=double(tmp3[1])
y_pixel_M1X_f3=double(tmp3[2])

print,'ret1(0)for the position 7.11236:',ret1(0)
print,'x_pixel_M1X_f1',x_pixel_M1X_f1
print,'y_pixel_M1X_f1',y_pixel_M1X_f1

print,'ret2(0)for the position 7.0123606:',ret2(0)
print,'x_pixel_M1X_f2',x_pixel_M1X_f2
print,'y_pixel_M1X_f2',y_pixel_M1X_f2

print,'ret3(0)for the position 7.2123545:',ret3(0)
print,'x_pixel_M1X_f3',x_pixel_M1X_f3
print,'y_pixel_M1X_f3',y_pixel_M1X_f3

pos_M1X=[7.11236,7.0123606,7.2123545]
slope=LINFIT(s_M1X,theta_M1X)
a1_M1X  = slope[0]
b1_M1X  = slope[1]

approx_AU1_M1X=a1_M1X+b1_M1X*pos_M1X

print,'approx_AU1_M1X (in degree): ', approx_AU1_M1X
print,'approx_AU1_M1X1-approx_AU1_M1X0 (in degree)',approx_AU1_M1X(1)-
approx_AU1_M1X(0)
print,'approx_AU1_M1X2-approx_AU1_M1X0 (in degree)',approx_AU1_M1X(2)-
approx_AU1_M1X(0)
print,''
print,'x_pixel_M1X_f2-x_pixel_M1X_f1 (pixel scale)',x_pixel_M1X_f2-x_pixel_M1X_f1
print,'x_pixel_M1X_f3-x_pixel_M1X_f1 (pixel scale)',x_pixel_M1X_f3-x_pixel_M1X_f1

approx_AU1_M1X=approx_AU1_M1X*3600
pixel_scale_back_M1X=(approx_AU1_M1X(1)-
approx_AU1_M1X(0))/sqrt((x_pixel_M1X_f2-x_pixel_M1X_f1)^2+(y_pixel_M1X_f2-
y_pixel_M1X_f1)^2)
pixel_scale_forw_M1X=(approx_AU1_M1X(2)-
approx_AU1_M1X(0))/sqrt((x_pixel_M1X_f3-x_pixel_M1X_f1)^2+(y_pixel_M1X_f3-
y_pixel_M1X_f1)^2)

print,''
print,'pixel_scale_back_M1X',pixel_scale_back_M1X
print,''
print,'pixel_scale_forw_M1X',pixel_scale_forw_M1X

;; pixel scale calculation for AU1M2TX
```

```
cmd='imfitpsf -r 3 -x 592 -y 523
/home/ao/testdata/au_backlash/AU1/AU1_M2X/7.29337_00033.fits'
spawn,cmd,ret1

cmd='imfitpsf -r 3 -x 617 -y 523
/home/ao/testdata/au_backlash/AU1/AU1_M2X/7.1933728_back_00034.fits'
spawn,cmd,ret2

cmd='imfitpsf -r 3 -x 568 -y 523
/home/ao/testdata/au_backlash/AU1/AU1_M2X/7.3933742_forw_00035.fits'
spawn,cmd,ret3

tmp1=STRSPLIT(ret1(0),'(,)', /EXTRACT)
x_pixel_M2X_f1=double(tmp1[1])
y_pixel_M2X_f1=double(tmp1[2])

tmp2=STRSPLIT(ret2(0),'(,)', /EXTRACT)
x_pixel_M2X_f2=double(tmp2[1])
y_pixel_M2X_f2=double(tmp2[2])

tmp3=STRSPLIT(ret3(0),'(,)', /EXTRACT)
x_pixel_M2X_f3=double(tmp3[1])
y_pixel_M2X_f3=double(tmp3[2])

print,'ret1(0)for the position 7.29337:',ret1(0)
print,'x_pixel_M2X_f1',x_pixel_M2X_f1
print,'y_pixel_M2X_f1',y_pixel_M2X_f1

print,'ret2(0)for the position 7.1933728:',ret2(0)
print,'x_pixel_M2X_f2',x_pixel_M2X_f2
print,'y_pixel_M2X_f2',y_pixel_M2X_f2

print,'ret3(0)for the position 7.3933742:',ret3(0)
print,'x_pixel_M2X_f3',x_pixel_M2X_f3
print,'y_pixel_M2X_f3',y_pixel_M2X_f3

pos_M2X=[7.29337,7.1933728,7.3933742]
slope2=LINFIT(s_M2X,theta_M1X)
a1_M2X  = slope2[0]
b1_M2X  = slope2[1]

approx_AU1_M2X=a1_M2X+b1_M2X*pos_M2X

print,'approx_AU1_M2X (in degree): ', approx_AU1_M2X
print,'approx_AU1_M2X1-approx_AU1_M2X0 (in degree)',approx_AU1_M2X(1)-
approx_AU1_M2X(0)
print,'approx_AU1_M2X2-approx_AU1_M2X0 (in degree)',approx_AU1_M2X(2)-
approx_AU1_M2X(0)
```

```
print,''
print,'x_pixel_M2X_f2-x_pixel_M2X_f1 (pixel scale)',x_pixel_M2X_f2-x_pixel_M2X_f1
print,'x_pixel_M2X_f3-x_pixel_M2X_f1 (pixel scale)',x_pixel_M2X_f3-x_pixel_M2X_f1

approx_AU1_M2X=approx_AU1_M2X*3600
pixel_scale_back_M2X=(approx_AU1_M2X(1)-
approx_AU1_M2X(0))/sqrt((x_pixel_M2X_f2-x_pixel_M2X_f1)^2+(y_pixel_M2X_f2-
y_pixel_M2X_f1)^2)
pixel_scale_forw_M2X=(approx_AU1_M2X(2)-
approx_AU1_M2X(0))/sqrt((x_pixel_M2X_f3-x_pixel_M2X_f1)^2+(y_pixel_M2X_f3-
y_pixel_M2X_f1)^2)

print,''
print,'pixel_scale_back_M2X',pixel_scale_back_M2X
print,''
print,'pixel_scale_forw_M2X',pixel_scale_forw_M2X
print,''

;; reading the gaussian center for the AU1M1TX having backlash measured at the
optical bench

filename='read_backlash_AU1M1X.dat'
openw,lun,filename , /get_lun ;,/append

x_pixel_back_M1X=fltarr(number_steps)
y_pixel_back_M1X=fltarr(number_steps)
x_pixel_forw_M1X=fltarr(number_steps)
y_pixel_forw_M1X=fltarr(number_steps)

back_M1X=fltarr(number_steps)
forw_M1X=fltarr(number_steps)
back_imgM1=fltarr(number_steps)
forw_imgM1=fltarr(number_steps)

for loop_point=0,number_points-1,1 do begin

for loop_step=0,number_steps-1,1 do begin


back_M1X=strcompress(string(s_M1X(loop_point)),/remove_all)+'_back_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
forw_M1X=strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
back_imgM1=READFITS('/home/ao/testdata/au_backlash/AU1/backlash_AU1M1X/'+
string(back_M1X))
forw_imgM1=READFITS('/home/ao/testdata/au_backlash/AU1/backlash_AU1M1X/'+
string(forw_M1X))
size_backM1=size(back_imgM1)
size_forwM1=size(forw_imgM1)
```

```
smooth_backM1=smooth(back_imgM1,20,/edge_truncate)
smooth_forwM1=smooth(forw_imgM1,20,/edge_truncate)
temp1_M1=max(smooth_backM1,index1_M1)
temp2_M1=max(smooth_forwM1,index2_M1)
x_backM1=index1_M1 mod size_backM1[1]
y_backM1=index1_M1 / size_backM1[1]
x_forwM1=index2_M1 mod size_forwM1[1]
y_forwM1=index2_M1 / size_forwM1[1]

;; cropping the image

crop_backM1=back_imgM1[x_backM1-10:x_backM1+10,y_backM1-10:y_backM1+10]
crop_forwM1=forw_imgM1[x_forwM1-10:x_forwM1+10,y_forwM1-10:y_forwM1+10]
fit_backM1=gauss2dfit(crop_backM1,/tilt,param_backM1)
fit_forwM1=gauss2dfit(crop_forwM1,/tilt,param_forwM1)
x_pixel_back_M1X(loop_step)=x_backM1-10+param_backM1[4]+1
y_pixel_back_M1X(loop_step)=y_backM1-10+param_backM1[5]+1
x_pixel_forw_M1X(loop_step)=x_forwM1-10+param_forwM1[4]+1
y_pixel_forw_M1X(loop_step)=y_forwM1-10+param_forwM1[5]+1
print,'gaussian x and y value for
M1X(back)',x_pixel_back_M1X(loop_step),'',y_pixel_back_M1X(loop_step)
print,''
print,'gaussian x and y value for
M1X(forw)',x_pixel_forw_M1X(loop_step),'',y_pixel_forw_M1X(loop_step)
print,''

printf,lun,s_M1X(loop_point),s_M2X(loop_point),x_pixel_back_M1X(loop_step),y_pix
el_back_M1X(loop_step),x_pixel_forw_M1X(loop_step),y_pixel_forw_M1X(loop_step),
FORMAT='(6F12.6)'

endfor

endfor

close,lun


;; reading the gaussian center for AU1M2TX having backlash, measured at the
optical bench

filename='read_backlash_AU1M2X.dat'
openw,lun,filename , /get_lun ;,/append

x_pixel_back_M2X=fltarr(number_steps)
y_pixel_back_M2X=fltarr(number_steps)
x_pixel_forw_M2X=fltarr(number_steps)
y_pixel_forw_M2X=fltarr(number_steps)
```

```
back_M2X=fltarr(number_steps)
forw_M2X=fltarr(number_steps)
back_imgM2=fltarr(number_steps)
forw_imgM2=fltarr(number_steps)

for loop_point=0,number_points-1,1 do begin

for loop_step=0,number_steps-1,1 do begin

back_M2X=strcompress(string(s_M2X(loop_point)),/remove_all)+'_back_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
forw_M2X=strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
back_imgM2=READFITS('/home/ao/testdata/au_backlash/AU1/backlash_AU1M2X/'+
string(back_M2X))
forw_imgM2=READFITS('/home/ao/testdata/au_backlash/AU1/backlash_AU1M2X/'+
string(forw_M2X))
size_backM2=size(back_imgM2)
size_forwM2=size(forw_imgM2)
smooth_backM2=smooth(back_imgM2,20,/edge_truncate)
smooth_forwM2=smooth(forw_imgM2,20,/edge_truncate)
temp1_M2=max(smooth_backM2,index1_M2)
temp2_M2=max(smooth_forwM2,index2_M2)
x_backM2=index1_M2 mod size_backM2[1]
y_backM2=index1_M2 / size_backM2[1]
x_forwM2=index2_M2 mod size_forwM2[1]
y_forwM2=index2_M2 / size_forwM2[1]

;; cropping the image

crop_backM2=back_imgM2[x_backM2-10:x_backM2+10,y_backM2-10:y_backM2+10]
crop_forwM2=forw_imgM2[x_forwM2-10:x_forwM2+10,y_forwM2-10:y_forwM2+10]
fit_backM2=gauss2dfit(crop_backM2,/tilt,param_backM2)
fit_forwM2=gauss2dfit(crop_forwM2,/tilt,param_forwM2)
x_pixel_back_M2X(loop_step)=x_backM2-10+param_backM2[4]+1
y_pixel_back_M2X(loop_step)=y_backM2-10+param_backM2[5]+1
x_pixel_forw_M2X(loop_step)=x_forwM2-10+param_forwM2[4]+1
y_pixel_forw_M2X(loop_step)=y_forwM2-10+param_forwM2[5]+1
print,'gaussian x and y value for
M2X(back)',x_pixel_back_M2X(loop_step),",y_pixel_back_M2X(loop_step)
print,"
print,'gaussian x and y value for
M2X(forw)',x_pixel_forw_M2X(loop_step),",y_pixel_forw_M2X(loop_step)
print,"
printf,lun,s_M2X(loop_point),s_M1X(loop_point),x_pixel_back_M2X(loop_step),y_pix
el_back_M2X(loop_step),x_pixel_forw_M2X(loop_step),y_pixel_forw_M2X(loop_step),
FORMAT='(6F12.6)'

endfor
```

```
endfor

close,lun

;; backlash measurement for AU1M1X at the optical bench

data1_M1TX =    fltarr(6,number_steps,number_points)
filename='read_backlash_AU1M1X.dat'
openr,lun,filename , /get_lun
readf,lun,data1_M1TX
close,lun
diff_pixelx_M1X= fltarr(number_steps)
diff_pixely_M1X=fltarr(number_steps)
bench_backlash_M1TX=fltarr(number_points)
diff_pixel_M1X=fltarr(number_steps)
stddev_M1TX   =   fltarr(number_points)

for loop_point=0, number_points-1,1 do begin

print,'the measurement point is',data1_M1TX[0,1,loop_point]
print,''

for loop_step=0,number_steps-1,1 do begin

diff_pixelx_M1X  =  data1_M1TX[4,loop_step,loop_point]-
data1_M1TX[2,loop_step,loop_point]
diff_pixely_M1X  =  data1_M1TX[5,loop_step,loop_point]-
data1_M1TX[3,loop_step,loop_point]
diff_pixel_M1X(loop_step)   =  sqrt(diff_pixelx_M1X^2+diff_pixely_M1X^2)

print,'backlash value at every point (in
arcseconds)',(diff_pixel_M1X(loop_step)*((pixel_scale_forw_M1X-
pixel_scale_back_M1X)/2))

endfor

stddev_M1TX(loop_point)   =  sqrt(variance(diff_pixel_M1X))
bench_backlash_M1TX(loop_point) = avg(diff_pixel_M1X)*((pixel_scale_forw_M1X-
pixel_scale_back_M1X)/2)

print,''
print,'averaged backlash value for AU1M1TX in arcsecond at bench
is:',bench_backlash_M1TX(loop_point)
print,''

endfor
print,'backlash value for AU1M1TX in arcsecond at bench is:',bench_backlash_M1TX
print,''
```

```
print,'backlash value for AU1M1TX in pixel at bench
is:',bench_backlash_M1TX/((pixel_scale_forw_M1X-pixel_scale_back_M1X)/2)
print,''
print,'standard deviation at each point (in pixel)',stddev_M1TX
print,''
print,'standard deviation at each point (in
arcsecond)',stddev_M1TX*((pixel_scale_forw_M1X-pixel_scale_back_M1X)/2)

data2_M1TX =    fltarr(6,number_steps*number_points)
difference_M1X =    fltarr(number_steps*number_points)

filename='read_backlash_AU1M1X.dat'
openr,lun,filename , /get_lun
readf,lun,data2_M1TX
close,lun

for i=0,(number_steps*number_points)-1,1 do begin

difference_M1X(i)=sqrt((data2_M1TX[4,i]-data2_M1TX[2,i])^2+(data2_M1TX[5,i]-
data2_M1TX[3,i])^2)

endfor

print,''

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU1.M1TX_bench_arcsecond.ps',
/INCHES
plot,data2_M1TX[0,*],difference_M1X*((pixel_scale_forw_M1X-
pixel_scale_back_M1X)/2),PSYM=1,$
TITLE='Backlash Measurement at the Optical Bench for AU1.M1TX',$
XTITLE='AU1.M1TX actuator positions around nominal point (in mm)',$
YTITLE='10 backlash measurement at each position (arcsecond)'
oplot,s_M1X,bench_backlash_M1TX,PSYM=-5,linestyle=1
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU1.M1TX_bench_pixel.ps', /INCHES
plot,data2_M1TX[0,*],difference_M1X,PSYM=1,$
TITLE='Backlash Measurement at the Optical Bench for AU1.M1TX',$
XTITLE='AU1.M1TX actuator positions around nominal point (in mm)',$
YTITLE='10 backlash measurement at each position(pixel scale)'
oplot,s_M1X,bench_backlash_M1TX/((pixel_scale_forw_M1X-
pixel_scale_back_M1X)/2),PSYM=-5,linestyle=1
DEVICE, /CLOSE

;; backlash measurement for AU1M2X at the optical bench

data1_M2TX =    fltarr(6,number_steps,number_points)
```

```
filename='read_backlash_AU1M2X.dat'
openr,lun,filename , /get_lun
readf,lun,data1_M2TX
close,lun

diff_pixelx_M2X= fltarr(number_steps)
diff_pixely_M2X=fltarr(number_steps)
bench_backlash_M2TX=fltarr(number_points)
diff_pixel_M2X=fltarr(number_steps)
stddev_M2TX    =    fltarr(number_points)

for loop_point=0, number_points-1,1 do begin

print,'the measurement point is',data1_M2TX[0,1,loop_point]
print,''

for loop_step=0,number_steps-1,1 do begin

diff_pixelx_M2X  =  data1_M2TX[4,loop_step,loop_point]-
data1_M2TX[2,loop_step,loop_point]
diff_pixely_M2X  =  data1_M2TX[5,loop_step,loop_point]-
data1_M2TX[3,loop_step,loop_point]
diff_pixel_M2X(loop_step)    =  sqrt(diff_pixelx_M2X^2+diff_pixely_M2X^2)

print,'backlash value at every point (in
arcseconds)',(diff_pixel_M2X(loop_step))*((pixel_scale_forw_M2X-
pixel_scale_back_M2X)/2)

endfor

stddev_M2TX(loop_point)   =  sqrt(variance(diff_pixel_M2X))
bench_backlash_M2TX(loop_point) = avg(diff_pixel_M2X)*((pixel_scale_forw_M2X-
pixel_scale_back_M2X)/2)

print,''
print,'averaged backlash value for AU1M2TX in arcsecond at bench
is:',bench_backlash_M2TX(loop_point)
print,''

endfor

print,''
print,'backlash value for AU1M2TX in arcsecond at bench is:',bench_backlash_M2TX
print,''
print,'backlash value for AU1M2TX in pixel at bench
is:',bench_backlash_M2TX/((pixel_scale_forw_M2X-pixel_scale_back_M2X)/2)
print,''
print,'standard deviation at each point (in pixel)',stddev_M2TX
```

```
print,''
print,'standard deviation at each point (in
arcsecond)',stddev_M2TX*((pixel_scale_forw_M2X-pixel_scale_back_M2X)/2)

data2_M2TX =    fltarr(6,number_steps*number_points)
difference_M2X =    fltarr(number_steps*number_points)

filename='read_backlash_AU1M2X.dat'
openr,lun,filename , /get_lun
readf,lun,data2_M2TX
close,lun

for i=0,(number_steps*number_points)-1,1 do begin

difference_M2X(i)=sqrt((data2_M2TX[4,i]-data2_M2TX[2,i])^2+(data2_M2TX[5,i]-
data2_M2TX[3,i])^2)

endfor

print,''

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU1.M2TX_bench_arcsecond.ps',
/INCHES
plot,data2_M2TX[0,*],difference_M2X*((pixel_scale_forw_M2X-
pixel_scale_back_M2X)/2),PSYM=1,$
TITLE='Backlash Measurement at the Optical Bench for AU1.M2TX',$
XTITLE='AU1.M2TX actuator positions around nominal point (in mm)',$
YTITLE='10 backlash measurement at each position (arcsecond)'
oplot,s_M2X,bench_backlash_M2TX,PSYM=-5,linestyle=1
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'backlash_for_AU1.M2TX_bench_pixel.ps', /INCHES
plot,data2_M2TX[0,*],difference_M2X,PSYM=1,$
TITLE='Backlash Measurement at the Optical Bench for AU1.M2TX',$
XTITLE='AU1.M2TX actuator positions around nominal point (in mm)',$
YTITLE='10 backlash measurement at each position (pixel scale)'
oplot,s_M2X,bench_backlash_M2TX/((pixel_scale_forw_M2X-
pixel_scale_back_M2X)/2),PSYM=-5,linestyle=1
DEVICE, /CLOSE

save,s_M1X,s_M2X,pixel_scale_forw_M1X,pixel_scale_back_M1X,pixel_scale_forw_
M2X,pixel_scale_back_M2X,bench_backlash_M1TX,bench_backlash_M2TX,stddev_
M1TX,stddev_M2TX,filename='/home/ao/testdata/au_backlash/AU1/bench_backlash
M1M2X.sav'

end
```

TASK 4

CODING 11

```
PRO backlash_compensation_algo_AU1M1M2X,number_points,number_steps,n,T

;; calculating backlash in mm from backlash value in arcseconds for AU1M1X

Restore,filename='/home/ao/testdata/au_backlash/AU1/bench_backlashM1M2X.sav'
bench_backlash_M1TX=bench_backlash_M1TX  ;; backlash in arcseconds
bench_backlash_M2TX=bench_backlash_M2TX  ;; backlash in arcseconds

s_M1X=s_M1X
s_M2X=s_M2X

theta_M1X = 1.390580e-06*s_M1X^4 + 2.812652e-06*s_M1X^3 + 1.259884e-
04*s_M1X^2 + 9.299078e-01*s_M1X - 1.168302e+01
slope=LINFIT(s_M1X,theta_M1X)
a1_M1X  = slope[0]
b1_M1X  = slope[1]
print,'slope',slope
bench_backlash_M1TX=bench_backlash_M1TX/3600
bench_backlash_mm_M1X=bench_backlash_M1TX/b1_M1X
print,'backlash for AU1M1X:',bench_backlash_M1TX
print,'bench_backlash_mm_M1X:',bench_backlash_mm_M1X

;; estimating tilt angle for AU1M2X

slope2=LINFIT(s_M2X,theta_M1X)
a1_M2X  = slope2[0]
b1_M2X  = slope2[1]

print,'slope',slope
bench_backlash_M2TX=bench_backlash_M2TX/3600
bench_backlash_mm_M2X=bench_backlash_M2TX/b1_M2X
print,'backlash for AU1M2X:',bench_backlash_M2TX
print,'bench_backlash_mm_M2X:',bench_backlash_mm_M2X

step_increment=0.1

;; program for backlash measurement for AU1.M1X (aroud the nominal position)
after the backlash compensation integration

;; s_M1X and s_M2X are the actuator positions of M1X and M2X respectively

M1X0= 7.11236  ; mm
M2X0= 7.29337
M1Y0= 12.84165
M2Y0= 12.99195
```

```
count=0

for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU1.M1TY and AU1.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU1.M1TY is : ', M1Y0
print,''
print,'nominal position for AU1.M2TY is : ', M2Y0
print,''
print,'moving AU1.M1TY and AU1.M2TY to M1Y0 and M2Y0 position respectively'
print,''
print,'measurement position for AU1.M1TY is :',M1Y0
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TY, ' + STRING(M1Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TY is : ', actual_position
print,''
print,'measurement position for AU1.M2TY is :',M2Y0
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TY, ' + STRING(M2Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TY is : ', actual_position
print,''
print,'measurement position for AU1.M2TX is :',s_M2X(loop_point)
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(s_M2X(loop_point), FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
AU1_M1TX_starting_position  =    s_M1X(loop_point)
print,'the measurement position for AU1.M1TX under consideration is :',
AU1_M1TX_starting_position
```

```
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_starting_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
print,' moving the actuator backward by n=5 times from the measurement position'

;;   moving the actuator backward by 5 times from the measurement position

count1=0

for i=0, n-1 ,1 do begin

;; moving the actuator by 0.1 mm in backward direction

count1=count1+1
print,'count1',count1
print,''
AU1_M1TX_commanded_position  =   AU1_M1TX_starting_position -
step_increment
print, 'commanded position for AU1.M1TX for moving in backward direction by -0.1
mm is : ', AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX : ', actual_position
print,''

;;  again moving to the measured point

AU1_M1TX_commanded_position  =   AU1_M1TX_commanded_position +
step_increment
print, 'commanded position for AU1.M1TX after coming back to the measurement
position is : ', AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
```

```
print, 'actual position of AU1.M1TX: ', actual_position

endfor

print,' taking 10 times back and forth measurements for AU1.M1TX'

count2=0

for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU1.M1TX is -

count2=count2+1
print,'count2',count2
print,''

;;  taking 10 times back and forth measurements for AU1.M1TX
;;  for moving the measurement position by 0.1 mm in forward direction

AU1_M1TX_commanded_position  =   AU1_M1TX_commanded_position +
step_increment
print, 'commanded position for AU1.M1TX for moving in forward direction by +0.1
mm is ',  AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; coming back to the measurement point

AU1_M1TX_commanded_position  =   AU1_M1TX_commanded_position -
step_increment-bench_backlash_mm_M1X(loop_point)
print, 'commanded position for AU1.M1TX for coming back at the measurement
position is : ', AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''
```

```
;; measuring the pixel positions on the CCD camera for the forward motion

cmd='aqcam 1 dir
/home/ao/testdata/au_backlash/AU1/compensated_backlash_AU1M1X'
spawn,cmd,ret
cmd='aqcam 1 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 obj '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 1 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10

;;  for moving the measurement position by 0.1 mm in backward direction

print, 'actual position of AU1.M1TX: ', actual_position
AU1_M1TX_commanded_position  =   AU1_M1TX_starting_position -
step_increment
print, 'commanded position for AU1.M1TX for moving in backward direction by -0.1
mm is : ', AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX : ', actual_position
print,''

;;  again moving to the measured point

AU1_M1TX_commanded_position  =   AU1_M1TX_commanded_position +
step_increment
print, 'commanded position for AU1.M1TX for coming back at the measurement
position is : ', AU1_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX: ', actual_position
print,''
```

```
;; measuring the pixel positions on the CCD camera for the forward  motion

cmd='aqcam 1 obj '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 1 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10
endfor
wait,1
endfor

;; program for backlash measurement for AU1.M2X (aroud the nominal position)
after the backlash compensation integration

;; s_M1X and s_M2X are the actuator positions of M1X and M2X respectively

M1X0= 7.11236  ; mm
M2X0= 7.29337
M1Y0= 12.84165
M2Y0= 12.99195

count=0

for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU1.M1TY and AU1.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU1.M1TY is : ', M1Y0
print,''
print,'nominal position for AU1.M2TY is : ', M2Y0
print,''
print,'moving AU1.M1TY and AU1.M2TY to M1Y0 and M2Y0 position respectively'
print,''
print,'measurement position for AU1.M1TY is :',M1Y0
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TY, ' + STRING(M1Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
```

```
print, 'actual position of AU1.M1TY is : ', actual_position
print,''
print,'measurement position for AU1.M2TY is :',M2Y0
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TY, ' + STRING(M2Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TY is : ', actual_position
print,''
print,'measurement position for AU1.M1TX is :',s_M1X(loop_point)
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(s_M1X(loop_point), FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
AU1_M2TX_starting_position  =    s_M2X(loop_point)
print,'the measurement position for AU1.M2TX under consideration is :',
AU1_M2TX_starting_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_starting_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
print,' moving the actuator backward by n=5 times from the measurement position'

;;   moving the actuator backward by 5 times from the measurement position

count1=0

for i=0, n-1 ,1 do begin

;; moving the actuator by 0.1 mm in backward direction

count1=count1+1
print,'count1',count1
print,''
```

```
AU1_M2TX_commanded_position  =  AU1_M2TX_starting_position -
step_increment
print, 'commanded position for AU1.M2TX for moving in backward direction by -0.1
mm is : ', AU1_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX : ', actual_position
print,''

;;  again moving to the measured point

AU1_M2TX_commanded_position  =  AU1_M2TX_commanded_position +
step_increment
print, 'commanded position for AU1.M2TX after coming back to the measurement
position is : ', AU1_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU1.M2TX'

count2=0

for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU1.M2TX is -

count2=count2+1
print,'count2',count2
print,''

;;  taking 10 times back and forth measurements for AU1.M2TX
;;  for moving the measurement position by 0.1 mm in forward direction

AU1_M2TX_commanded_position  =  AU1_M2TX_commanded_position +
step_increment
```

```
print, 'commanded position for AU1.M2TX for moving in forward direction by +0.1
mm is ',  AU1_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; coming back to the measurement point

AU1_M2TX_commanded_position  =   AU1_M2TX_commanded_position -
step_increment-bench_backlash_mm_M2X(loop_point)
print, 'commanded position for AU1.M2TX for coming back at the measurement
position is : ', AU1_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; measuring the pixel positions on the CCD camera for the forward motion

cmd='aqcam 1 dir
/home/ao/testdata/au_backlash/AU1/compensated_backlash_AU1M2X'
spawn,cmd,ret
cmd='aqcam 1 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 obj '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 1 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10

;;  for moving the measurement position by 0.1 mm in backward direction

print, 'actual position of AU1.M2TX: ', actual_position
```

```
AU1_M2TX_commanded_position  =  AU1_M2TX_starting_position -
step_increment
print, 'commanded position for AU1.M2TX for moving in backward direction by -0.1
mm is : ', AU1_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX : ', actual_position
print,''

;;  again moving to the measured point

AU1_M2TX_commanded_position  =  AU1_M2TX_commanded_position +
step_increment
print, 'commanded position for AU1.M2TX for coming back at the measurement
position is : ', AU1_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX: ', actual_position
print,''

;; measuring the pixel positions on the CCD camera for the forward  motion

cmd='aqcam 1 obj '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 1 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10

endfor
wait,1
endfor

end
```

CODING 12

```
PRO read_gaussian_center_AU1M1M2X,number_steps,number_points

Restore,filename='/home/ao/testdata/au_backlash/AU1/bench_backlashM1M2X.sav'
s_M1X=s_M1X
s_M2X=s_M2X
bench_backlash_M1TX=bench_backlash_M1TX  ;; backlash in arcseconds
bench_backlash_M2TX=bench_backlash_M2TX  ;; backlash in arcseconds
pixel_scale_forw_M1X=pixel_scale_forw_M1X
pixel_scale_back_M1X=pixel_scale_back_M1X
pixel_scale_forw_M2X=pixel_scale_forw_M2X
pixel_scale_back_M2X=pixel_scale_back_M2X

pixel_scale_M1=(pixel_scale_forw_M1X-pixel_scale_back_M1X)/2
pixel_scale_M2=(pixel_scale_forw_M2X-pixel_scale_back_M2X)/2

;;reading the gaussian center of the images after backlash compensation for
AU1M1TX

filename='read_gaussian_center_AU1M1X.dat'
openw,lun,filename , /get_lun ;,/append

x_pixel_back_M1X=fltarr(number_steps)
y_pixel_back_M1X=fltarr(number_steps)
x_pixel_forw_M1X=fltarr(number_steps)
y_pixel_forw_M1X=fltarr(number_steps)

back_M1X=fltarr(number_steps)
forw_M1X=fltarr(number_steps)
back_imgM1=fltarr(number_steps)
forw_imgM1=fltarr(number_steps)

for loop_point=0,number_points-1,1 do begin

for loop_step=0,number_steps-1,1 do begin

back_M1X=strcompress(string(s_M1X(loop_point)),/remove_all)+'_back_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
forw_M1X=strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
back_imgM1=READFITS('/home/ao/testdata/au_backlash/AU1/compensated_backlas
h_AU1M1X/'+string(back_M1X))
forw_imgM1=READFITS('/home/ao/testdata/au_backlash/AU1/compensated_backlas
h_AU1M1X/'+string(forw_M1X))

size_backM1=size(back_imgM1)
size_forwM1=size(forw_imgM1)
```

```
smooth_backM1=smooth(back_imgM1,20,/edge_truncate)
smooth_forwM1=smooth(forw_imgM1,20,/edge_truncate)
temp1_M1=max(smooth_backM1,index1_M1)
temp2_M1=max(smooth_forwM1,index2_M1)
x_backM1=index1_M1 mod size_backM1[1]
y_backM1=index1_M1 / size_backM1[1]
x_forwM1=index2_M1 mod size_forwM1[1]
y_forwM1=index2_M1 / size_forwM1[1]
print,''
print,'index of the brightest pixel for M1X(back)',x_backM1,y_backM1
print,''
print,'index of the brightest pixel for M1X(forw)',x_forwM1,y_forwM1
 print,''

;; cropping the image

crop_backM1=back_imgM1[x_backM1-10:x_backM1+10,y_backM1-10:y_backM1+10]
crop_forwM1=forw_imgM1[x_forwM1-10:x_forwM1+10,y_forwM1-10:y_forwM1+10]
fit_backM1=gauss2dfit(crop_backM1,/tilt,param_backM1)
fit_forwM1=gauss2dfit(crop_forwM1,/tilt,param_forwM1)
x_pixel_back_M1X(loop_step)=x_backM1-10+param_backM1[4]+1
y_pixel_back_M1X(loop_step)=y_backM1-10+param_backM1[5]+1
x_pixel_forw_M1X(loop_step)=x_forwM1-10+param_forwM1[4]+1
y_pixel_forw_M1X(loop_step)=y_forwM1-10+param_forwM1[5]+1
print,'gaussian x and y value for
M1X(back)',x_pixel_back_M1X(loop_step),'',y_pixel_back_M1X(loop_step)
print,''
print,'gaussian x and y value for
M1X(forw)',x_pixel_forw_M1X(loop_step),'',y_pixel_forw_M1X(loop_step)
print,''
printf,lun,s_M1X(loop_point),s_M2X(loop_point),x_pixel_back_M1X(loop_step),y_pix
el_back_M1X(loop_step),x_pixel_forw_M1X(loop_step),y_pixel_forw_M1X(loop_step),
FORMAT='(6F12.6)'

endfor
endfor
close,lun

data1_M1TX      =    fltarr(6,number_steps,number_points)
data2_M1TX      =    fltarr(6,number_steps*number_points)
difference_M1X  =    fltarr(number_steps*number_points)

diff_pixelx_M1X= fltarr(number_steps)
diff_pixely_M1X=fltarr(number_steps)
compensated_backlash_M1TX=fltarr(number_points)
diff_pixel_M1X=fltarr(number_steps)
stddev_M1TX   =   fltarr(number_points)
```

```
filename='read_gaussian_center_AU1M1X.dat'
openr,lun,filename , /get_lun
readf,lun,data1_M1TX
close,lun

for loop_point=0, number_points-1,1 do begin

print,'the measurement point is',data1_M1TX[0,1,loop_point]
print,''

for loop_step=0,number_steps-1,1 do begin

diff_pixelx_M1X  =  data1_M1TX[4,loop_step,loop_point]-
data1_M1TX[2,loop_step,loop_point]
diff_pixely_M1X  =  data1_M1TX[5,loop_step,loop_point]-
data1_M1TX[3,loop_step,loop_point]
diff_pixel_M1X(loop_step)    =  sqrt(diff_pixelx_M1X^2+diff_pixely_M1X^2)

if (diff_pixelx_M1X lt 0) then begin

diff_pixel_M1X(loop_step)=diff_pixel_M1X(loop_step)*pixel_scale_M1
print,'compensated backlash value at every point (in arcseconds) for
AU1M1X',diff_pixel_M1X(loop_step)

endif else begin

diff_pixel_M1X(loop_step)=(diff_pixel_M1X(loop_step)*pixel_scale_M1)*(-1)

print,'compensated backlash value at every point (in arcseconds) for
AU1M1X',diff_pixel_M1X(loop_step)
endelse

endfor

compensated_backlash_M1TX(loop_point) =  avg(diff_pixel_M1X)

stddev_M1TX(loop_point)   =  sqrt(variance(diff_pixel_M1X))
print,''

endfor

print,'averaged compensated backlash value for AU1M1TX in arcsecond at bench
is:',compensated_backlash_M1TX
print,''
print,'averaged compensated backlash value for AU1M1TX in pixel at bench
is:',compensated_backlash_M1TX/pixel_scale_M1
print,''
print,'standard deviation for AU1M1TX in arcsecond at bench is:',stddev_M1TX
print,''
```

```
print,'standard deviation for AU1M1TX in pixel at bench
is:',stddev_M1TX/pixel_scale_M1
print,''

filename='read_gaussian_center_AU1M1X.dat'
openr,lun,filename , /get_lun
readf,lun,data2_M1TX
close,lun

for i=0,(number_steps*number_points)-1,1 do begin

if ((data2_M1TX[4,i]-data2_M1TX[2,i]) lt 0) then begin
diff=(sqrt((data2_M1TX[4,i]-data2_M1TX[2,i])^2+(data2_M1TX[5,i]-
data2_M1TX[3,i])^2))*1
difference_M1X(i)=diff

endif else begin

diff=(sqrt((data2_M1TX[4,i]-data2_M1TX[2,i])^2+(data2_M1TX[5,i]-
data2_M1TX[3,i])^2))*(-1)
difference_M1X(i)=diff

endelse

endfor

SET_PLOT, 'PS'
DEVICE,/portrait,filename=
'compensated_backlash_for_AU1.M1TX_bench_arcseconds.ps', /INCHES
plot,data2_M1TX[0,*],difference_M1X*pixel_scale_M1,PSYM=1,linestyle=4,$
TITLE='Backlash compensation at the Optical Bench for AU1.M1TX',$
XTITLE='AU1.M1TX actuator positions around nominal point (in mm)',$
YTITLE='Backlash compensation at each point (arcsecond)',YRANGE=[-3,3]
;oplot,s_M1X,stddev_M1TX,PSYM=-6
oplot,s_M1X,compensated_backlash_M1TX,PSYM=-5,linestyle=1
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename=
'compensated_backlash_for_AU1.M1TX_bench_pixel.ps', /INCHES
plot,data2_M1TX[0,*],difference_M1X,PSYM=1,linestyle=4,$
TITLE='Backlash compensation at the Optical Bench for AU1.M1TX',$
XTITLE='AU1.M1TX actuator positions around nominal point (in mm)',$
YTITLE='compensation for 10 measurements at each point (pixel sacle)',YRANGE=[-
1,1]
;oplot,s_M1X,stddev_M1TX,PSYM=-6
oplot,s_M1X,compensated_backlash_M1TX/pixel_scale_M1,PSYM=-5,linestyle=1
DEVICE, /CLOSE
```

```
;; reading the gaussian center of the images after backlash compensation for
AU1M2TX

filename='read_gaussian_center_AU1M2X.dat'
openw,lun,filename , /get_lun ;,/append

x_pixel_back_M2X=fltarr(number_steps)
y_pixel_back_M2X=fltarr(number_steps)
x_pixel_forw_M2X=fltarr(number_steps)
y_pixel_forw_M2X=fltarr(number_steps)

back_M2X=fltarr(number_steps)
forw_M2X=fltarr(number_steps)
back_imgM2=fltarr(number_steps)
forw_imgM2=fltarr(number_steps)

for loop_point=0,number_points-1,1 do begin

for loop_step=0,number_steps-1,1 do begin

back_M2X=strcompress(string(s_M2X(loop_point)),/remove_all)+'_back_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
forw_M2X=strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
back_imgM2=READFITS('/home/ao/testdata/au_backlash/AU1/compensated_backlas
h_AU1M2X/'+string(back_M2X))

forw_imgM2=READFITS('/home/ao/testdata/au_backlash/AU1/compensated_backlas
h_AU1M2X/'+string(forw_M2X))

size_backM2=size(back_imgM2)
size_forwM2=size(forw_imgM2)
smooth_backM2=smooth(back_imgM2,20,/edge_truncate)
smooth_forwM2=smooth(forw_imgM2,20,/edge_truncate)
temp1_M2=max(smooth_backM2,index1_M2)
temp2_M2=max(smooth_forwM2,index2_M2)
x_backM2=index1_M2 mod size_backM2[1]
y_backM2=index1_M2 / size_backM2[1]
x_forwM2=index2_M2 mod size_forwM2[1]
y_forwM2=index2_M2 / size_forwM2[1]
print,''
print,'index of the brightest pixel for M2X(back)',x_backM2,y_backM2
print,''
print,'index of the brightest pixel for M2X(forw)',x_forwM2,y_forwM2
print,''

;; cropping the image

crop_backM1=back_imgM1[x_backM1-10:x_backM1+10,y_backM1-10:y_backM1+10]
```

```
crop_forwM1=forw_imgM1[x_forwM1-10:x_forwM1+10,y_forwM1-10:y_forwM1+10]
crop_backM2=back_imgM2[x_backM2-10:x_backM2+10,y_backM2-10:y_backM2+10]
crop_forwM2=forw_imgM2[x_forwM2-10:x_forwM2+10,y_forwM2-10:y_forwM2+10]
fit_backM2=gauss2dfit(crop_backM2,/tilt,param_backM2)
fit_forwM2=gauss2dfit(crop_forwM2,/tilt,param_forwM2)
x_pixel_back_M2X(loop_step)=x_backM2-10+param_backM2[4]+1
y_pixel_back_M2X(loop_step)=y_backM2-10+param_backM2[5]+1
x_pixel_forw_M2X(loop_step)=x_forwM2-10+param_forwM2[4]+1
y_pixel_forw_M2X(loop_step)=y_forwM2-10+param_forwM2[5]+1
print,'gaussian x and y value for
M2X(back)',x_pixel_back_M2X(loop_step),'',y_pixel_back_M2X(loop_step)
print,''
print,'gaussian x and y value for
M2X(forw)',x_pixel_forw_M2X(loop_step),'',y_pixel_forw_M2X(loop_step)
print,''
printf,lun,s_M2X(loop_point),s_M1X(loop_point),x_pixel_back_M2X(loop_step),y_pix
el_back_M2X(loop_step),x_pixel_forw_M2X(loop_step),y_pixel_forw_M2X(loop_step),
FORMAT='(6F12.6)'

endfor

endfor

close,lun

data1_M2TX      =    fltarr(6,number_steps,number_points)
data2_M2TX      =    fltarr(6,number_steps*number_points)
difference_M2X  =    fltarr(number_steps*number_points)

diff_pixelx_M2X= fltarr(number_steps)
diff_pixely_M2X=fltarr(number_steps)
compensated_backlash_M2TX=fltarr(number_points)
diff_pixel_M2X=fltarr(number_steps)
stddev_M2TX   =   fltarr(number_points)

filename='read_gaussian_center_AU1M2X.dat'
openr,lun,filename , /get_lun
readf,lun,data1_M2TX
close,lun

for loop_point=0, number_points-1,1 do begin

print,'the measurement point is',data1_M2TX[0,1,loop_point]
print,''

for loop_step=0,number_steps-1,1 do begin

diff_pixelx_M2X  =  data1_M2TX[4,loop_step,loop_point]-
data1_M2TX[2,loop_step,loop_point]
```

```
diff_pixely_M2X  =  data1_M2TX[5,loop_step,loop_point]-
data1_M2TX[3,loop_step,loop_point]
diff_pixel_M2X(loop_step)    =  sqrt(diff_pixelx_M2X^2+diff_pixely_M2X^2)
print,'diff_pixel_M2',   diff_pixel_M2X(loop_step)
if (diff_pixelx_M2X lt 0) then begin

diff_pixel_M2X(loop_step)=diff_pixel_M2X(loop_step)*pixel_scale_M2
print,'compensated backlash value at every point (in arcseconds) for
AU1M2X',diff_pixel_M2X(loop_step)

endif else begin

diff_pixel_M2X(loop_step)=(diff_pixel_M2X(loop_step)*pixel_scale_M2)*(-1)

print,'compensated backlash value at every point (in arcseconds) for
AU1M2X',diff_pixel_M2X(loop_step)
endelse

endfor

compensated_backlash_M2TX(loop_point) =  avg(diff_pixel_M2X)

stddev_M2TX(loop_point)   =  sqrt(variance(diff_pixel_M2X))
print,''
print,'averaged backlash value for AU1M2TX in arcsecond at bench
is:',compensated_backlash_M2TX(loop_point)
print,''

endfor

print,'averaged compensated backlash value for AU1M2TX in arcsecond at bench
is:',compensated_backlash_M2TX
print,''
print,'averaged compensated backlash value for AU1M2TX in pixel at bench
is:',compensated_backlash_M2TX/pixel_scale_M2
print,''
print,'standard deviation for AU1M2TX in arcsecond at bench is:',stddev_M2TX
print,''
print,'standard deviation for AU1M2TX in pixel at bench
is:',stddev_M2TX/pixel_scale_M2
print,''

filename='read_gaussian_center_AU1M2X.dat'
openr,lun,filename , /get_lun
readf,lun,data2_M2TX
close,lun

for i=0,(number_steps*number_points)-1,1 do begin
```

```
if ((data2_M2TX[4,i]-data2_M2TX[2,i]) lt 0) then begin
diff=(sqrt((data2_M2TX[4,i]-data2_M2TX[2,i])^2+(data2_M2TX[5,i]-
data2_M2TX[3,i])^2))*1
difference_M2X(i)=diff

endif else begin

diff=(sqrt((data2_M2TX[4,i]-data2_M2TX[2,i])^2+(data2_M2TX[5,i]-
data2_M2TX[3,i])^2))*(-1)
difference_M2X(i)=diff

endelse

endfor

SET_PLOT, 'PS'
DEVICE,/portrait,filename=
'compensated_backlash_for_AU1.M2TX_bench_arcseconds.ps', /INCHES
plot,data2_M2TX[0,*],difference_M2X*pixel_scale_M2,PSYM=1,linestyle=1,$
TITLE='Backlash compensation at the Optical Bench for AU1.M2TX',$
XTITLE='AU1.M2TX actuator positions around nominal point (in mm)',$
YTITLE='Backlash compensation at each point (in arcseconds)'
oplot,s_M2X,compensated_backlash_M2TX,PSYM=-5,linestyle=1
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename=
'compensated_backlash_for_AU1.M2TX_bench_pixel.ps', /INCHES
plot,data2_M2TX[0,*],difference_M2X,PSYM=1,linestyle=1,$
TITLE='Backlash compensation at the Optical Bench for AU1.M2TX',$
XTITLE='AU1.M2TX actuator positions around nominal point (in mm)',$
YTITLE='compensation for 10 measurements at each point (pixel sacle)'
oplot,s_M2X,compensated_backlash_M2TX/pixel_scale_M2,PSYM=-5,linestyle=1
DEVICE, /CLOSE

end
```

****

CODING 13

```
PRO backlash_compensation_algo,number_points,number_steps,n,T

step_increment=0.1
restore,filename='/home/ao/testdata/au_backlash/AU2/fitting_angle.sav'
s_M1X=s_M1X
s_M2X=s_M2X
bench_backlash_M1TX=bench_backlash_M1TX
bench_backlash_mm1=bench_backlash_mm
```

```
bench_backlash_M2TX=bench_backlash_M2TX
bench_backlash_mm_M2X=bench_backlash_mm_M2X

s_M1X=11.5000+findgen(number_points)*step_increment
print,'s_M1X',s_M1X
print,''
s_M2X=11.5500+findgen(number_points)*step_increment
print,'s_M2X',s_M2X

;; program for backlash measurement for AU2.M1X (aroud the nominal position)
after the backlash compensation integration

;; s_M1X and s_M2X are the actuator positions of M1X and M2X respectively

M1X0= 12.0000  ; mm
M2X0= 11.75000
M1Y0=12.63000
M2Y0=12.43000
count=0

for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU2.M1TY is : ', M1Y0
print,''
print,'nominal position for AU2.M2TY is : ', M2Y0
print,''
print,'moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively'
print,''
print,'measurement position for AU2.M1TY is :',M1Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' + STRING(M1Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY is : ', actual_position
print,'measurement position for AU2.M2TY is :',M2Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' + STRING(M2Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
```

```
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY is : ', actual_position
print,''
print,'measurement position for AU2.M2TX is :',s_M2X(loop_point)
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(s_M2X(loop_point), FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
AU2_M1TX_starting_position  =    s_M1X(loop_point)
print,'the measurement position for AU2.M1TX under consideration is :',
AU2_M1TX_starting_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_starting_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''
print,' moving the actuator backward by n=5 times from the measurement position'

;;   moving the actuator backward by 5 times from the measurement position

count1=0

for i=0, n-1 ,1 do begin

;; moving the actuator by 0.1 mm in backward direction
count1=count1+1
print,'count1',count1
print,''
AU2_M1TX_commanded_position  =   AU2_M1TX_starting_position -
step_increment
print, 'commanded position for AU2.M1TX for moving in backward direction by -0.1
mm is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
```

```
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX : ', actual_position
print,''

;;  again moving to the measured point

AU2_M1TX_commanded_position  =   AU2_M1TX_commanded_position +
step_increment
print, 'commanded position for AU2.M1TX after coming back to the measurement
position is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU2.M1TX'

count2=0

for loop_step=0, number_steps-1 do begin

;;  the commanded position for AU2.M1TX is -

count2=count2+1
print,'count2',count2
print,''

;;  taking 10 times back and forth measurements for AU2.M1TX
;;  for moving the measurement position by 0.1 mm in forward direction

AU2_M1TX_commanded_position  =   AU2_M1TX_commanded_position +
step_increment
print, 'commanded position for AU2.M1TX for moving in forward direction by +0.1
mm is ',  AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
```

```
;; coming back to the measurement point

AU2_M1TX_commanded_position  =   AU2_M1TX_commanded_position -
step_increment-bench_backlash_mm(loop_point)
print, 'commanded position for AU2.M1TX for coming back at the measurement
position is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; measuring the pixel positions on the CCD camera for the forward motion

cmd='aqcam 2 dir
/home/ao/testdata/au_backlash/AU2/compensated_backlash_AU2M1X'
spawn,cmd,ret
cmd='aqcam 2 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 obj '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 2 fp '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 2 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+10

 ;;  for moving the measurement position by 0.1 mm in backward direction

print, 'actual position of AU2.M1TX: ', actual_position
AU2_M1TX_commanded_position  =   AU2_M1TX_starting_position -
step_increment
print, 'commanded position for AU2.M1TX for moving in backward direction by -0.1
mm is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX : ', actual_position
print,''
```

```
 ;;  again moving to the measured point

AU2_M1TX_commanded_position  =   AU2_M1TX_commanded_position +
step_increment
print, 'commanded position for AU2.M1TX for coming back at the measurement
position is : ', AU2_M1TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(AU2_M1TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX: ', actual_position
print,''

;; measuring the pixel positions on the CCD camera for the forward  motion

cmd='aqcam 2 obj '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 2 fp '+strcompress(string(s_M1X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 2 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+10

endfor
 wait,1
endfor

;; program for backlash measurement for AU2.M2X (aroud the nominal position)
after the backlash compensation integration

for loop_point=0, number_points-1, 1 do begin

count=count+1
print,'count',count
print,''

;; moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU2.M1TY is : ', M1Y0
print,''
print,'nominal position for AU2.M2TY is : ', M2Y0
print,''
print,'moving AU2.M1TY and AU2.M2TY to M1Y0 and M2Y0 position respectively'
```

```
print,''
print,'measurement position for AU2.M1TY is :',M1Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TY, ' + STRING(M1Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TY is : ', actual_position
print,''
print,'measurement position for AU2.M2TY is :',M2Y0
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TY, ' + STRING(M2Y0,
FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TY is : ', actual_position
print,''
print,'measurement position for AU2.M1TX is :',s_M1X(loop_point)
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M1TX, ' +
STRING(s_M1X(loop_point), FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M1TX is : ', actual_position
print,''
AU2_M2TX_starting_position  =    s_M2X(loop_point)
 print,'the measurement position for AU2.M2TX under consideration is :',
AU2_M2TX_starting_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_starting_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX is : ', actual_position
print,''
print,' moving the actuator backward by n=5 times from the measurement position'

;;   moving the actuator backward by 5 times from the measurement position

count1=0
```

```
 for i=0, n-1 ,1 do begin

;; moving the actuator by 0.1 mm in backward direction
count1=count1+1
print,'count1',count1
print,''

AU2_M2TX_commanded_position  =   AU2_M2TX_starting_position -
step_increment
print, 'commanded position for AU2.M2TX for moving in backward direction by -0.1
mm is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX : ', actual_position
print,''

 ;;  again moving to the measured point

AU2_M2TX_commanded_position  =   AU2_M2TX_commanded_position +
step_increment
print, 'commanded position for AU2.M2TX after coming back to the measurement
position is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX: ', actual_position
print,''

endfor

print,' taking 10 times back and forth measurements for AU2.M2TX'

count2=0

 for loop_step=0, number_steps-1 do begin

  ;;  the commanded position for AU2.M2TX is -

count2=count2+1
print,'count2',count2
```

```
;;  taking 10 times back and forth measurements for AU2.M2TX
;;  for moving the measurement position by 0.1 mm in forward direction

AU2_M2TX_commanded_position  =   AU2_M2TX_commanded_position +
step_increment
print, 'commanded position for AU2.M2TX for moving in forward direction by +0.1
mm is ',  AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position

;; coming back to the measurement point

AU2_M2TX_commanded_position  =   AU2_M2TX_commanded_position -
step_increment-bench_backlash_mm_M2X(loop_point)
print, 'commanded position for AU2.M2TX for coming back at the measurement
position is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of XPS: ', actual_position
print,''

;; measuring the pixel positions on the CCD camera for the forward motion

cmd='aqcam 2 dir
/home/ao/testdata/au_backlash/AU2/compensated_backlash_AU2M2X'
spawn,cmd,ret
cmd='aqcam 2 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 obj '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 2 fp '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw_'
spawn,cmd,ret
cmd='aqcam 2 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+10
```

```
;;  for moving the measurement position by 0.1 mm in backward direction

print, 'actual position of AU2.M2TX: ', actual_position
AU2_M2TX_commanded_position  =   AU2_M2TX_starting_position -
step_increment
print, 'commanded position for AU2.M2TX for moving in backward direction by -0.1
mm is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX : ', actual_position
print,''

;;  again moving to the measured point

AU2_M2TX_commanded_position  =   AU2_M2TX_commanded_position +
step_increment
print, 'commanded position for AU2.M2TX for coming back at the measurement
position is : ', AU2_M2TX_commanded_position
cmd = './xps_raw.tcl 10.0.0.52 "GroupMoveAbsolute (AU2.M2TX, ' +
STRING(AU2_M2TX_commanded_position, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait,1
cmd = './xps_raw.tcl 10.0.0.52 "GroupPositionCurrentGet (AU2.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU2.M2TX: ', actual_position
print,''

;; measuring the pixel positions on the CCD camera for the forward  motion

cmd='aqcam 2 obj '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 2 fp '+strcompress(string(s_M2X(loop_point)),/remove_all)+'_back_'
spawn,cmd,ret
cmd='aqcam 2 fn '+strcompress(string(loop_step),/remove_all)
spawn,cmd,ret
cmd='aqcam 2 snap'
spawn,cmd,ret
wait,T+10
endfor
wait,1
endfor

end
```

CODING 14

```
PRO read_gaussian_center,number_steps,number_points

Restore,filename='/home/ao/testdata/au_backlash/AU2/bench_backlashM1M2X.sav'
s_M1X=s_M1X
s_M2X=s_M2X
bench_backlash_M1TX=bench_backlash_M1TX  ;; backlash in arcseconds for
AU2M1X
bench_backlash_mm=bench_backlash_mm      ;; compensation value for the backlash
for AU2M1X
pixel_scale_forw_M1X=pixel_scale_forw_M1X
pixel_scale_back_M1X=pixel_scale_back_M1X

pixel_scale_M1=(pixel_scale_forw_M1X-pixel_scale_back_M1X)/2

bench_backlash_M2TX=bench_backlash_M2TX  ;; backlash in arcseconds for
AU2M2X
bench_backlash_mm_M2X=bench_backlash_mm_M2X      ;; compensation value for
the backlash for AU2M2X
pixel_scale_forw_M2X=pixel_scale_forw_M2X
pixel_scale_back_M2X=pixel_scale_back_M2X

pixel_scale_M2=(pixel_scale_forw_M2X-pixel_scale_back_M2X)/2

filename='read_gaussian_center_AU2M1X.dat'
openw,lun,filename , /get_lun ;,/append

x_pixel_back_M1X=fltarr(number_steps)
y_pixel_back_M1X=fltarr(number_steps)
x_pixel_forw_M1X=fltarr(number_steps)
y_pixel_forw_M1X=fltarr(number_steps)

back_M1X=fltarr(number_steps)
forw_M1X=fltarr(number_steps)
back_img=fltarr(number_steps)
forw_img=fltarr(number_steps)

for loop_point=0,number_points-1,1 do begin

for loop_step=0,number_steps-1,1 do begin

back_M1X=strcompress(string(s_M1X(loop_point)),/remove_all)+'_back_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
forw_M1X=strcompress(string(s_M1X(loop_point)),/remove_all)+'_forw_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
back_img=READFITS('/home/ao/testdata/au_backlash/AU2/compensated_backlash_
AU2M1X/'+string(back_M1X))
```

```
forw_img=READFITS('/home/ao/testdata/au_backlash/AU2/compensated_backlash_
AU2M1X/'+string(forw_M1X))
size_back=size(back_img)
size_forw=size(forw_img)
smooth_back=smooth(back_img,20,/edge_truncate)
smooth_forw=smooth(forw_img,20,/edge_truncate)
temp1=max(smooth_back,index1)
temp2=max(smooth_forw,index2)
x_back=index1 mod size_back[1]
y_back=index1 / size_back[1]
x_forw=index2 mod size_forw[1]
y_forw=index2 / size_forw[1]
print,''
print,'index of the brightest pixel (back)',x_back,y_back
print,''
print,'index of the brightest pixel (forw)',x_forw,y_forw
print,''

;; cropping the image

crop_back=back_img[x_back-10:x_back+10,y_back-10:y_back+10]
crop_forw=forw_img[x_forw-10:x_forw+10,y_forw-10:y_forw+10]
fit_back=gauss2dfit(crop_back,/tilt,param_back)
fit_forw=gauss2dfit(crop_forw,/tilt,param_forw)
x_pixel_back_M1X(loop_step)=x_back-10+param_back[4]+1
y_pixel_back_M1X(loop_step)=y_back-10+param_back[5]+1
x_pixel_forw_M1X(loop_step)=x_forw-10+param_forw[4]+1
y_pixel_forw_M1X(loop_step)=y_forw-10+param_forw[5]+1
print,'gaussian x and y value
(back)',x_pixel_back_M1X(loop_step),'',y_pixel_back_M1X(loop_step)
print,''
print,'gaussian x and y value
(forw)',x_pixel_forw_M1X(loop_step),'',y_pixel_forw_M1X(loop_step)
print,''
printf,lun,s_M1X(loop_point),s_M2X(loop_point),x_pixel_back_M1X(loop_step),y_pix
el_back_M1X(loop_step),x_pixel_forw_M1X(loop_step),y_pixel_forw_M1X(loop_step),
FORMAT='(6F12.6)'
endfor

endfor

close,lun

data1_M1TX      =    fltarr(6,number_steps,number_points)
data2_M1TX      =    fltarr(6,number_steps*number_points)
difference_M1X  =    fltarr(number_steps*number_points)

diff_pixelx_M1X= fltarr(number_steps)
```

```
diff_pixely_M1X=fltarr(number_steps)
compensated_backlash_M1TX=fltarr(number_points)
diff_pixel_M1X=fltarr(number_steps)
stddev_M1TX    =   fltarr(number_points)

filename='read_gaussian_center_AU2M1X.dat'
openr,lun,filename , /get_lun
readf,lun,data1_M1TX
close,lun

for loop_point=0, number_points-1,1 do begin

print,'the measurement point is',data1_M1TX[0,1,loop_point]
print,''

for loop_step=0,number_steps-1,1 do begin

diff_pixelx_M1X  =  data1_M1TX[4,loop_step,loop_point]-
data1_M1TX[2,loop_step,loop_point]
diff_pixely_M1X  =  data1_M1TX[5,loop_step,loop_point]-
data1_M1TX[3,loop_step,loop_point]
diff_pixel_M1X(loop_step)    =  sqrt(diff_pixelx_M1X^2+diff_pixely_M1X^2)

 if (diff_pixelx_M1X lt 0) then begin

diff_pixel_M1X(loop_step)=diff_pixel_M1X(loop_step)*pixel_scale_M1
print,'compensated backlash value at every point (in arcseconds) for
AU2M1X',diff_pixel_M1X(loop_step)

endif else begin

diff_pixel_M1X(loop_step)=(diff_pixel_M1X(loop_step)*pixel_scale_M1)*(-1)

print,'compensated backlash value at every point (in arcseconds) for
AU2M1X',diff_pixel_M1X(loop_step)
endelse

endfor
compensated_backlash_M1TX(loop_point) =  avg(diff_pixel_M1X)
stddev_M1TX(loop_point)   =  sqrt(variance(diff_pixel_M1X))
endfor
print,''
print,'compensated averaged backlash value for AU2M1TX in arcsecond at bench
is:',compensated_backlash_M1TX
print,''
print,'compensated averaged backlash value for AU2M1TX in pixel at bench
is:',compensated_backlash_M1TX/pixel_scale_M1
print,''
print,'standard deviation for AU2M1TX in arcsecond at bench is:',stddev_M1TX
```

```
print,''
print,'standard deviation for AU2M1TX in pixel at bench
is:',stddev_M1TX/pixel_scale_M1
print,''

filename='read_gaussian_center_AU2M1X.dat'
openr,lun,filename , /get_lun
readf,lun,data2_M1TX
close,lun

for i=0,(number_steps*number_points)-1,1 do begin

if ((data2_M1TX[4,i]-data2_M1TX[2,i]) lt 0) then begin
diff=(sqrt((data2_M1TX[4,i]-data2_M1TX[2,i])^2+(data2_M1TX[5,i]-
data2_M1TX[3,i])^2))*1
difference_M1X(i)=diff

endif else begin

diff=(sqrt((data2_M1TX[4,i]-data2_M1TX[2,i])^2+(data2_M1TX[5,i]-
data2_M1TX[3,i])^2))*(-1)
difference_M1X(i)=diff

endelse

endfor

SET_PLOT, 'PS'
DEVICE,/portrait,filename=
'compensated_backlash_for_AU2.M1TX_bench_arcseconds.ps', /INCHES
plot,data2_M1TX[0,*],difference_M1X*pixel_scale_M1,PSYM=1,$
TITLE='Backlash compensation at the Optical Bench for AU2.M1TX',$
XTITLE='AU2.M1TX actuator positions around nominal point (in mm)',$
YTITLE='Backlash compensation at each point (arcsecond)',YRANGE=[-3,3]
;oplot,s_M1X,stddev_M1TX,PSYM=-6,linestyle=1
;oplot,s_M1X,stddev_M1TX*(-1),PSYM=-6,linestyle=1
oplot,s_M1X,compensated_backlash_M1TX,PSYM=-5,linestyle=1
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename=
'compensated_backlash_for_AU2.M1TX_bench_pixel.ps', /INCHES
plot,data2_M1TX[0,*],difference_M1X,PSYM=1,$
TITLE='Backlash compensation at the Optical Bench for AU2.M1TX',$
XTITLE='AU2.M1TX actuator positions around nominal point (in mm)',$
YTITLE='compensation for 10 measurements at each point (pixel sacle)',YRANGE=[-
1,1]
;oplot,s_M1X,stddev_M1TX/pixel_scale_M1,PSYM=-6,linestyle=1
oplot,s_M1X,compensated_backlash_M1TX/pixel_scale_M1,PSYM=-5,linestyle=1
```

```
DEVICE, /CLOSE

;;reading the gaussian center of the images after backlash compensation for
AU2M2X

filename='read_gaussian_center_AU2M2X.dat'
openw,lun,filename , /get_lun ;,/append

x_pixel_back_M2X=fltarr(number_steps)
y_pixel_back_M2X=fltarr(number_steps)
x_pixel_forw_M2X=fltarr(number_steps)
y_pixel_forw_M2X=fltarr(number_steps)

back_M2X=fltarr(number_steps)
forw_M2X=fltarr(number_steps)
back_img_M2X=fltarr(number_steps)
forw_img_M2X=fltarr(number_steps)

for loop_point=0,number_points-1,1 do begin

for loop_step=0,number_steps-1,1 do begin

back_M2X=strcompress(string(s_M2X(loop_point)),/remove_all)+'_back_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
forw_M2X=strcompress(string(s_M2X(loop_point)),/remove_all)+'_forw_0000'+strco
mpress(string(loop_step),/remove_all)+'.fits'
back_img_M2X=READFITS('/home/ao/testdata/au_backlash/AU2/compensated_back
lash_AU2M2X/'+string(back_M2X))
forw_img_M2X=READFITS('/home/ao/testdata/au_backlash/AU2/compensated_back
lash_AU2M2X/'+string(forw_M2X))
size_back_M2X=size(back_img_M2X)
size_forw_M2X=size(forw_img_M2X)
smooth_back_M2X=smooth(back_img_M2X,20,/edge_truncate)
smooth_forw_M2X=smooth(forw_img_M2X,20,/edge_truncate)
temp1_M2X=max(smooth_back_M2X,index1_M2X)
temp2_M2X=max(smooth_forw_M2X,index2_M2X)
x_back_M2X=index1_M2X mod size_back_M2X[1]
 y_back_M2X=index1_M2X / size_back_M2X[1]
x_forw_M2X=index2_M2X mod size_forw_M2X[1]
y_forw_M2X=index2_M2X / size_forw_M2X[1]
print,''
print,'index of the brightest pixel (back) for AU2M2X',x_back_M2X,y_back_M2X
print,''
print,'index of the brightest pixel (forw) for AU2M2X',x_forw_M2X,y_forw_M2X
print,''

;; cropping the image
```

```
crop_back_M2X=back_img_M2X[x_back_M2X-10:x_back_M2X+10,y_back_M2X-
10:y_back_M2X+10]
crop_forw_M2X=forw_img_M2X[x_forw_M2X-10:x_forw_M2X+10,y_forw_M2X-
10:y_forw_M2X+10]
fit_back_M2X=gauss2dfit(crop_back_M2X,/tilt,param_back_M2X)
fit_forw_M2X=gauss2dfit(crop_forw_M2X,/tilt,param_forw_M2X)
x_pixel_back_M2X(loop_step)=x_back_M2X-10+param_back_M2X[4]+1
y_pixel_back_M2X(loop_step)=y_back_M2X-10+param_back_M2X[5]+1
x_pixel_forw_M2X(loop_step)=x_forw_M2X-10+param_forw_M2X[4]+1
y_pixel_forw_M2X(loop_step)=y_forw_M2X-10+param_forw_M2X[5]+1
print,'gaussian x and y value
(back)',x_pixel_back_M2X(loop_step),'',y_pixel_back_M2X(loop_step)
print,''
print,'gaussian x and y value
(forw)',x_pixel_forw_M2X(loop_step),'',y_pixel_forw_M2X(loop_step)
printf,lun,s_M2X(loop_point),s_M1X(loop_point),x_pixel_back_M2X(loop_step),y_pix
el_back_M2X(loop_step),x_pixel_forw_M2X(loop_step),y_pixel_forw_M2X(loop_step),
FORMAT='(6F12.6)'
endfor

endfor

close,lun

data1_M2TX      =    fltarr(6,number_steps,number_points)
data2_M2TX      =    fltarr(6,number_steps*number_points)
difference_M2X  =    fltarr(number_steps*number_points)

diff_pixelx_M2X= fltarr(number_steps)
diff_pixely_M2X=fltarr(number_steps)
compensated_backlash_M2TX=fltarr(number_points)
diff_pixel_M2X=fltarr(number_steps)
stddev_M2TX   =   fltarr(number_points)

filename='read_gaussian_center_AU2M2X.dat'
openr,lun,filename , /get_lun
readf,lun,data1_M2TX
close,lun

for loop_point=0, number_points-1,1 do begin

print,'the measurement point is',data1_M2TX[0,1,loop_point]
print,''

for loop_step=0,number_steps-1,1 do begin

diff_pixelx_M2X  =  data1_M2TX[4,loop_step,loop_point]-
data1_M2TX[2,loop_step,loop_point]
```

```
diff_pixely_M2X  =  data1_M2TX[5,loop_step,loop_point]-
data1_M2TX[3,loop_step,loop_point]
diff_pixel_M2X(loop_step)    =  sqrt(diff_pixelx_M2X^2+diff_pixely_M2X^2)

 if (diff_pixelx_M2X lt 0) then begin

diff_pixel_M2X(loop_step)=diff_pixel_M2X(loop_step)*pixel_scale_M2
print,'compensated backlash value at every point (in arcseconds) for
AU2M2X',diff_pixel_M2X(loop_step)

endif else begin

diff_pixel_M2X(loop_step)=(diff_pixel_M2X(loop_step)*pixel_scale_M2)*(-1)

print,'compensated backlash value at every point (in arcseconds) for
AU2M2X',diff_pixel_M2X(loop_step)
endelse

endfor

compensated_backlash_M2TX(loop_point) =  avg(diff_pixel_M2X)

stddev_M2TX(loop_point)   =  sqrt(variance(diff_pixel_M2X))

endfor

print,''
print,'compensated averaged backlash value for AU2M2TX in arcsecond at bench
is:',compensated_backlash_M2TX
print,''
print,'compensated averaged backlash value for AU2M2TX in pixel at bench
is:',compensated_backlash_M2TX/pixel_scale_M2
print,''
print,'standard deviation for AU2M2TX in arcsecond at bench is:',stddev_M2TX
print,''
print,'standard deviation for AU2M2TX in pixel at bench
is:',stddev_M2TX/pixel_scale_M2
print,''

filename='read_gaussian_center_AU2M2X.dat'
openr,lun,filename , /get_lun
readf,lun,data2_M2TX
close,lun


for i=0,(number_steps*number_points)-1,1 do begin

if ((data2_M2TX[4,i]-data2_M2TX[2,i]) lt 0) then begin
```

```
diff=(sqrt((data2_M2TX[4,i]-data2_M2TX[2,i])^2+(data2_M2TX[5,i]-
data2_M2TX[3,i])^2))*1
difference_M2X(i)=diff

endif else begin

diff=(sqrt((data2_M2TX[4,i]-data2_M2TX[2,i])^2+(data2_M2TX[5,i]-
data2_M2TX[3,i])^2))*(-1)
difference_M2X(i)=diff

endelse

endfor

SET_PLOT, 'PS'
DEVICE,/portrait,filename=
'compensated_backlash_for_AU2.M2TX_bench_arcseconds.ps', /INCHES
plot,data2_M2TX[0,*],difference_M2X*pixel_scale_M2,PSYM=1,$
TITLE='Backlash compensation at the Optical Bench for AU2.M2TX',$
XTITLE='AU2.M2TX actuator positions around nominal point (in mm)',$
YTITLE='Backlash compensation at each point (arcsecond)',YRANGE=[-3,3]
;oplot,s_M1X,stddev_M1TX,PSYM=-6,linestyle=1
oplot,s_M2X,compensated_backlash_M2TX,PSYM=-5,linestyle=1
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename=
'compensated_backlash_for_AU2.M2TX_bench_pixel.ps', /INCHES
plot,data2_M2TX[0,*],difference_M2X,PSYM=1,$
TITLE='Backlash compensation at the Optical Bench for AU2.M2TX',$
XTITLE='AU2.M2TX actuator positions around nominal point (in mm)',$
YTITLE='compensation for 10 measurements at each point (pixel sacle)',YRANGE=[-
1,1]
oplot,s_M2X,compensated_backlash_M2TX/pixel_scale_M2,PSYM=-5,linestyle=1
DEVICE, /CLOSE

save,s_M1X,s_M2X,compensated_backlash_M2TX,compensated_backlash_M1TX,file
name='compensated_backlash.sav'

end
```

****

TASK 5

CODING 15

PRO testing_1,number_points,x,y

```
;; x and y are the desired starting points for positioners x and y respectively defined
by the user (should be in millimeters)
;; open file

;;filename='y=x^2_trajectory_test.dat'      ;; for y=x^2 trajectory
filename='y=x_trajectory_test.dat'          ;; for y=x trajectory
;;filename='x_axis_test.dat'                ;; testing the trajectory moving only
                                             the X positioner
;;filename='y_axis_test.dat'                ;; testing the trajectory moving only
                                             the Y positioner
;;filename='y=x_1trajectory_test.dat'       ;; for y=x trajectory having different
                                             start and end point

openw,lun,filename , /get_lun ;,/append

cmd1 = './xps_raw.tcl 133.40.147.71 "GroupMoveAbsolute (XY.X,'+STRING(x,
FORMAT='(F5.2)') + ')"'
cmd2 = './xps_raw.tcl 133.40.147.71 "GroupMoveAbsolute (XY.Y, '+STRING(y,
FORMAT='(F5.2)') + ')"'
SPAWN, cmd1, ret1
SPAWN, cmd2, ret2

cmd3 = './xps_raw.tcl 133.40.147.71 "GroupPositionCurrentGet (XY.X, double *)"'
cmd4 = './xps_raw.tcl 133.40.147.71 "GroupPositionCurrentGet (XY.Y, double *)"'
SPAWN, cmd3, ret3
SPAWN, cmd4, ret4

actual_position_for_X=double((STRSPLIT(ret3, /EXTRACT))[1])
print, 'actual position for positioner X: ', actual_position_for_X

actual_position_for_Y=double((STRSPLIT(ret4, /EXTRACT))[1])
print, 'actual position for positioner Y: ', actual_position_for_Y

;; calling the procedure trajectroy

trajectory,number_points,(actual_position_for_X)*1e-3, (actual_position_for_Y)*1e-3
restore,filename='points.sav'
xx=xx                    ;; measurement positions for the actuator in x direction
yy=yy                    ;; measurement positions for the actuator in y direction

count=0

repeat begin

cmd1 = './xps_raw.tcl 133.40.147.71 "GroupPositionCurrentGet (XY.X, double *)"'
cmd2 = './xps_raw.tcl 133.40.147.71 "GroupPositionCurrentGet (XY.Y, double *)"'
SPAWN, cmd1, ret1
SPAWN, cmd2, ret2
```

```
actual_position_for_X=double((STRSPLIT(ret1, /EXTRACT))[1])
print, 'actual position for positioner X: ', actual_position_for_X
actual_position_for_Y=double((STRSPLIT(ret2, /EXTRACT))[1])
print, 'actual position for positioner Y: ', actual_position_for_Y
t= actual_position_for_X

;; taking measurements from ACT while actuators are in trajectory

cmd3 = './act_meas.sh meas'
SPAWN, cmd3,ret3
tmp = STRSPLIT(ret3, ',', /EXTRACT)
pos_x = DOUBLE(tmp[4])
pos_y = DOUBLE(tmp[3])
PRINT, pos_x
PRINT, pos_y
wait, 1
printf,lun,actual_position_for_X,actual_position_for_Y,pos_x,pos_y,FORMAT='(5F12
.6)'
count=count+1
print,'count:',count

endrep until actual_position_for_X ge xx(number_points-1)

elements=count

;;save,elements,filename='n_elements(pos_x)y=x1.sav'
;;save,elements,filename='n_elements(pos_x)y=x^2.sav'
save,elements,filename='n_elements(pos_x)y=x.sav'
;;save,elements,filename='n_elements(pos_x)x_axis.sav'

close,lun

END
```

****

CODING 16

```
PRO trajectory,number_points,x0,y0

;; x0 and y0 are the starting point for the positioner x and y respectively for which
the mirror's angle is zero. This is the user defined floating number.

;;number_points=24.

theta_x=findgen(number_points+1)*0.1*0.01745   ;; in radians
theta_y=theta_x^2
theta_y2=theta_x
```

```
xx=fltarr(number_points+1)
yy=fltarr(number_points+1)
xx_test=fltarr(number_points+1)
yy_test=fltarr(number_points+1)

xx2=fltarr(number_points+1)
yy2=fltarr(number_points+1)
xx_test2=fltarr(number_points+1)
yy_test2=fltarr(number_points+1)

delta_y=fltarr(number_points)
xmid=fltarr(number_points)

a1=65.0e-3                              ;; in m
b1=22.08e-3

a2=88.86e-3
b2=22.01e-3

;;x0=7.1e-3
;;y0=12.2e-3

;; for the trajectory y=x^2

for i=0, number_points,1 do begin

xx(i)=a1*sin(theta_x(i))-sqrt(b1^2-a1^2*(1-cos(theta_x(i)))^2)+b1+x0
yy(i)=a2*sin(theta_y(i))-sqrt(b2^2-a2^2*(1-cos(theta_y(i)))^2)+b2+y0

endfor

;; xx and yy are the corresponding values in meters for the theta_x and theta_y
respectively.

;; for the trajectory y=x

for i=0, number_points,1 do begin

xx2(i)=a1*sin(theta_x(i))-sqrt(b1^2-a1^2*(1-cos(theta_x(i)))^2)+b1+x0
yy2(i)=a2*sin(theta_y2(i))-sqrt(b2^2-a2^2*(1-cos(theta_y2(i)))^2)+b2+y0

endfor

;; to find the minimum resolution value of the actuators

for i=0, number_points-1,1 do begin

xmid(i)=(xx(i)+xx(i+1))*0.5
delta_y(i)=(((yy(i+1)-yy(i))/(xx(i+1)-xx(i)))*(xmid(i)-xx(i)))+yy(i)-xmid(i)^2
```

```
endfor

start_point=delta_y(0)
end_point=delta_y(number_points-1)
print,'xmid',xmid
print,''
print,'avg (xmid)',total(xmid)/n_elements(xmid)
print,''
print,'theta_x',theta_x
print,''
print,'theta_y',theta_y
print,''
print,'xx (in mm)',xx*1e+3
print,''
print,'yy (in mm)',yy*1e+3
print,''
print,'yy2 (in mm)',yy2*1e+3
print,''
print,'delta_y',delta_y
print,''

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'y=x^2.ps', /INCHES
plot,xx,yy,psym=-2,yrange=[12.0,13.0],$
TITLE='Y=X^2 Trajectory',$
XTITLE='measurement points for Positioner X (in mm)',$
YTITLE='measurement points for positioner Y (in mm)'
DEVICE,/CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'y=x.ps', /INCHES
plot,xx,yy2,psym=-2,yrange=[12.0,20.0],$
TITLE='Y=X Trajectory',$
XTITLE='measurement points for Positioner X (in mm)',$
YTITLE='measurement points for positioner Y (in mm)'
DEVICE,/CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'y=x^2 (in degree).ps', /INCHES
plot,theta_x,theta_y,psym=-2,$
TITLE='Y=X^2 Trajectory (in degree)',$
XTITLE='measurement points for Positioner X (in degree)',$
YTITLE='measurement points for positioner Y (in degree)'
DEVICE,/CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'y=x (in degree).ps', /INCHES
plot,theta_x,theta_y2,psym=-2,$
```

```
TITLE='Y=X Trajectory (in degree)',$
XTITLE='measurement points for Positioner X (in degree)',$
YTITLE='measurement points for positioner Y (in degree)'
DEVICE,/CLOSE

;; minimum resolution found= 9 micrometers
;; to check whether the minimum step value in arcsecond is < 1 arcsecond

step=9.0e-6/a
print,''
print,'step',step

theta_mid=fltarr(number_points)

for i=0, number_points,1 do begin

theta_x(i)=theta_x(0)+step*i

endfor

theta_y=theta_x_test^2
theta_y2=theta_x_test

for i=0, number_points-1,1 do begin

theta_mid(i)=(theta_x_test(i)+theta_x_test(i+1))*0.5
delta_y(i)=(((theta_y(i+1)-theta_y(i))/(theta_x(i+1)-theta_x(i)))*(theta_mid(i)
theta_x(i)))+theta_y(i)-theta_mid(i)^2

endfor

print,''
print,'theta_x_test',theta_x_test
print,''
print,'theta_y_test',theta_y_test
print,''
print,'theta_y_test2',theta_y_test2
print,''
print,'delta_y_test',delta_y_test
print,''
print,'delta_y_test value in arcseconds',(delta_y_test*3600*180)/!PI

step_size=9.0e-3                    ;; in mm

xx=xx*1e+3                          ;; in mm
yy=yy*1e+3
yy2=yy2*1e+3

for i=0,number_points,1 do begin
```

```
xx(i)=xx(0)+step_size*i
yy(i)=yy(0)+step_size*i
yy2(i)=yy2(0)+step_size*i

endfor

Line=fltarr(number_points+1)
print,''
print,'xx after adding step (in mm)',xx
print,''
print,'yy after adding step (in mm)',yy
print,''
print,'yy2 after adding step (in mm)',yy2
print,''

save,xx,yy,yy2,filename='points.sav'

end
```

****

CODING 17

```
PRO traj,number_points
step_size=9e-6                    ;; in m

x=fltarr(number_points+1)
y=fltarr(number_points+1)          ;; for y=x trajectory
y1=fltarr(number_points+1)         ;; for y=x^2 trajectory

;; x and y are the relative position values that defines the trajectory defined by the
user

position=fltarr(number_points+1)
theta_x=fltarr(number_points+1)
theta_y=fltarr(number_points+1)
x(0)=step_size

for i=1,number_points,1 do begin

x(i)=step_size*(i+1)   ;; in m

endfor

a1=65.0e-3                        ;; in m
b1=22.08e-3
a2=88.86e-3
b2=22.01e-3
```

```
print,'number of elements in x', n_elements(x)
start_point=x(0)
end_point=x(number_points-1)
position=x

;; calling the procedure numerical1 to find out the angle corresponding to the x
values in meters

numerical1,number_points,start_point,end_point,position,a1,b1
restore,filename='error.sav'              ;; restoring the y (theta) and error value
theta_x=y
;;theta_y=theta_x

;; for y=x trajectory

for i=0,number_points,1 do begin

y(i)=a2*sin(theta_x(i))-sqrt(b2^2-a2^2*(1-cos(theta_x(i)))^2)+b2

endfor

;; for y=x^2 trajectroy

theta_y=theta_x^2

for i=0,number_points,1 do begin

y1(i)=a2*sin(theta_y(i))-sqrt(b2^2-a2^2*(1-cos(theta_y(i)))^2)+b2

endfor

x=x*1e+3                                  ;; in mm
y=y*1e+3
y1=y1*1e+3

print,''
print,'x (in mm) : ',x
print,''
print,'theta_x (in radians): ',theta_x
print,''
print,'y (in mm) : ',y
print,''
print,'y1 (in mm) : ',y1

;; creating y=x^2 trajectory file

filename1='XYfile1.trj'
openw, lun, filename1, /get_lun ;,/append
printf,lun,'FirstTangent=',strtrim(45,1)+';Degrees'
```

```
printf,lun,'DiscontinuityAngle=0.10;Degrees'

for i=1,number_points,1 do begin

Line=string(x(i))+','+strtrim(string(y1(i)),1)
printf,lun,'Line=',strtrim(Line,1)

endfor

close, lun

;; creating Y=X trajectory file

filename1='XYfile2.trj'
openw, lun, filename1, /get_lun ;,/append
printf,lun,'FirstTangent=',strtrim(45,1)+';Degrees'
printf,lun,'DiscontinuityAngle=0.10;Degrees'

for i=0,number_points,1 do begin

Line=string(x(i))+','+strtrim(string(y(i)),1)
printf,lun,'Line=',strtrim(Line,1)

endfor

close, lun

;; creating Y=X trajectory file for different starting points for x and y

filename1='XYfile2.1.trj'
openw, lun, filename1, /get_lun ;,/append
printf,lun,'FirstTangent=',strtrim(45,1)+';Degrees'
printf,lun,'DiscontinuityAngle=0.10;Degrees'

for i=0,number_points,1 do begin

Line=string(x(i))+','+strtrim(string(0),1)
printf,lun,'Line=',strtrim(Line,1)

endfor

close, lun

end
```

****

CODING 18

```
PRO read_trajectory_file_1, number_points

restore, filename='n_elements(pos_x)y=x^2.sav'
row1=elements
print,elements

restore, filename='n_elements(pos_x)y=x.sav'
row2=elements
print,elements

restore, filename='n_elements(pos_x)y=x1.sav'
row3=elements
print,elements

data1=fltarr(4,row1)
data2=fltarr(4,row2)
data3=fltarr(4,row3)

filename='y=x^2_trajectory_test.dat'
openr, lun, filename, /GET_LUN
readf,lun,data1
close, lun

filename='y=x_trajectory_test.dat'
openr, lun, filename, /GET_LUN
readf,lun,data2
close, lun

filename='y=x_1trajectory_test.dat'
openr, lun, filename, /GET_LUN
readf,lun,data3
close, lun

delta_angle1=fltarr(row1)
delta_angle2=fltarr(row2)
delta_angle3=fltarr(row3)

print,'row elements for y=x^2',row1
print,'row elements for y=x',row2
print,'row elements for y=x (for different starting points)',row3

;; for y=x^2 trajectory

for i=0, row1-1,1 do begin

data1[2,i]=0.01745*(-1*data1[2,i]-(-1*data1[2,0]))          ;; expressing in radians
data1[3,i]=0.01745*(-1*data1[3,i]-(-1*data1[3,0]))

endfor
```

```
;; for y=x trajectory

for i=0, row2-1,1 do begin

data2[2,i]=0.01745*(data2[2,i]-data2[2,0])
data2[3,i]=0.01745*(data2[3,i]-data2[3,0])

endfor

;; for y=x trajectory for different starting point

for i=0, row3-1,1 do begin

data3[2,i]=0.01745*(data3[2,i]-(data3[2,0]))
data3[3,i]=0.01745*(data3[3,i]-data3[3,0])

endfor

delta_angle1=data1[3,*]-data1[2,*]^2
delta_angle2=data2[3,*]-(-1*data2[2,*])
delta_angle3=data3[3,*]-(-1*data3[2,*])

print,''
print,'error in accuracy (in radians) for y=x^2
trajectory',(delta_angle1*180*3600)/!PI
print,''
print,'error in accuracy (in radians) for y=x trajectory',(delta_angle2*180*3600)/!PI
print,''
print,'error in accuracy (in radians) for y=x trajectory for different starting
point',(delta_angle3*180*3600)/!PI

var1=variance(delta_angle1)
var2=variance(delta_angle2)
var3=variance(delta_angle3)

print,''
print,'variance for y=x^2 trajectory',var1
print,''
print,'variance for y=x trajectory',var2
print,''
print,'variance for y=x trajectory for different starting point',var3
print,''

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'y=x^2_trajectory.ps', /INCHES
plot,data1[2,*],data1[3,*],$
;;XRANGE=[12.0,12.70],YRANGE=[7.0,7.6],$
;;,PSYM=-2,XRANGE=[-7.0,20.0],$
```

```
TITLE='XY trajectroy',$
XTITLE='measurement in degrees for X plane',$
YTITLE='measurement in degrees for Y plane'
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'y=x_trajectory.ps', /INCHES
plot,data2[2,*],data2[3,*],$
;;XRANGE=[12.0,12.70],YRANGE=[7.0,7.6],$
;;,PSYM=-2,XRANGE=[-7.0,20.0],$
TITLE='Y=X trajectroy',$
XTITLE='measurement in degrees for X plane',$
YTITLE='measurement in degrees for Y plane'
DEVICE, /CLOSE

end
```

****

CODING 19

```
;;f is the function and fd is its derivative

PRO numerical1,number_points,start_point,end_point,position,a,b

print, 'parameters:',n_params()
y=fltarr(number_points+1)
error1=fltarr(number_points+1)

y0=start_point/a                    ;; initial guess  (in radians)
print,'y0',y0
e=10e-10
print,'position',position

;; positions are the values of x that is defined in the 'traj' procedure which are the
relative positions that defines a line arc segment of the trajectory

;; a and b are the distance from the mirror to the lever and from the lever to the
actuator respectively. This is the choice for the user whether to take a and b for
positioner x or for positioner y.
```

Note: Rest of the program is same as coding 2

****

TASK 6

CODING 20

```
PRO focus_mm,number_steps
```

```
;; reading the focus value from the file

dz1=0.1+findgen(number_steps)*0.1
print,'dz1',dz1
tx=fltarr(number_steps)
focus1=fltarr(number_steps)
focus2=fltarr(number_steps)
tilt_M1=fltarr(number_steps)
tilt_M2=fltarr(number_steps)
AB1=fltarr(number_steps)
AB2=fltarr(number_steps)

a=150*sin((30*!PI)/180.)+dz1
b=150*cos((30*!PI)/180.)
tilt_M1=(atan(a/b)-((30*!PI)/180.))*0.5
tilt_M1=(tilt_M1*180.)/!PI
print,'tilt_M1: ',tilt_M1

dz2=-0.1+findgen(number_steps)*(-0.1)
print,'dz2',dz2
tx=fltarr(10)

a=150*sin((30*!PI)/180.)-dz2
b=150*cos((30*!PI)/180.)
tilt_M2=(atan(a/b)-((30*!PI)/180.))*0.5
tilt_M2=(tilt_M2*180.)/!PI
print,'tilt_M2: ',tilt_M2
print,''

AB1=sqrt((150*cos(30*0.01745))^2+(150*sin(30*0.01745)+dz1)^2)
focus1=dz1-230+80+AB1

AB2=sqrt((150*cos(30*0.01745))^2+(150*sin(30*0.01745)+dz2)^2)
focus2=dz2-230+80+AB2

data1=fltarr(4,number_steps)

filename='focus_+dz.dat'
openr,lun1,filename , /get_lun
readf,lun1,data1
close,lun1

data2=fltarr(4,number_steps)

filename='focus_-dz.dat'
openr,lun2,filename , /get_lun
readf,lun2,data2
close,lun2
```

```
print,''
print,'dz1: ',dz1
print,''
print,'focus1: ',focus1
print,''
print,'tilt_M1: ', tilt_M1
print,''
print,'dz2: ',dz2
print,''
print,'focus2: ',focus2
print,''
print,'tilt_M2: ',tilt_M2
print,''

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'focus_vs_stage_position.ps', /INCHES
plot,dz1,focus1,PSYM=-5,linestyle=1,$
TITLE='focus change vs distance chnage between M1 and M2 of AU1',$
XTITLE='relative stage position (in mm)',XRANGE=[-1.5,1.5],$
YTITLE='change in focus (in mm)',YRANGE=[-1.6,1.6]
oplot,dz2,focus2,PSYM=-4,linestyle=1
DEVICE, /CLOSE

;; converting the tilt angle to the actuator position using the linear equation for
AU1M1X

e= 1.390580e-06
d= 2.812652e-06
c= 1.259884e-04
b_M1X= 9.299078e-01
a_M1X= 1.168302e+01
s_M1TX=fltarr(number_steps)   ;; in mm
s_M2TX=fltarr(number_steps)   ;; in mm
s_M1X0=7.11236
s_M2X0=7.29337
tilt_offsetM1 = e*s_M1X0^4 + d*s_M1X0^3 + c*s_M1X0^2 + b_M1X*s_M1X0 -
a_M1X
print,''
print,'tilt_offsetM1: ',tilt_offsetM1
print,''

;; the equation is : tilt_M1=b_M1X*s_M1X0-a_M1X-tilt_offsetM1

s_M1TX=(tilt_M1-5.06918096+a_M1X)/b_M1X
print,'s_M1TX: ',s_M1TX
print,''
```

```
;; converting the tilt angle to the actuator position using the linear equation for
AU1M2X

b_M2X=9.359111e-01
a_M2X=1.173010e+01
tilt_offsetM2 = 1.508692e-06*s_M2X0^4 - 4.099467e-06*s_M2X0^3 + 1.249980e-
05*s_M2X0^2 + b_M2X*s_M2X0 - a_M2X
print,'tilt_offsetM2X: ',tilt_offsetM2
print,''
s_M2TX=(tilt_M1+a_M2X-4.904154061)/b_M2X
print,'s_M2TX: ',s_M2TX
print,''

restore,filename='/home/ao/testdata/au_backlash/AU1/trajectory/bench_backlashAU
1M1M2X.sav'
bench_backlash_M1TX=bench_backlash_M1TX
bench_backlash_M2TX=bench_backlash_M2TX
pos_M1X=s_M1X
pos_M2X=s_M2X

print,'bench backlash for M1Tx: ',bench_backlash_M1TX(6:number_steps-2)
print,''
print,'pos_M1X:',pos_M1X(6:number_steps-2)
print,''
fitting_M1TX_interpol=INTERPOL(bench_backlash_M1TX(6:number_steps-
2),pos_M1X(6:number_steps-2),s_M1TX)
print,'interpolated backlash value for M1TX: ',fitting_M1TX_interpol

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'interpolated_backlash_posstage_M1X.ps', /INCHES
plot,pos_M1X(6:number_steps-2),bench_backlash_M1TX(6:number_steps-2),PSYM=-
5,linestyle=1,$
TITLE='Interpolated Backlash for AU1.M1TX when stage moves forward',$
XTITLE='AU1.M1TX actuator position around nominal position (in
mm)',XRANGE=[7.05,7.35],$
YTITLE='Interpolated backlash (arcseconds)',YRANGE=[10,18]
oplot,s_M1TX,fitting_M1TX_interpol,PSYM=-2,linestyle=1
DEVICE, /CLOSE

print,'bench backlash for M2Tx: ',bench_backlash_M2TX(6:number_steps-2)
print,''
print,'pos_M2X:',pos_M2X(6:number_steps-2)
print,''
fitting_M2TX_interpol=INTERPOL(bench_backlash_M2TX(6:number_steps-
2),pos_M2X(6:number_steps-2),s_M2TX)
print,'interpolated backlash value for M2TX: ',fitting_M2TX_interpol

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'interpolated_backlash_posstage_M2X.ps', /INCHES
```

```
plot,pos_M2X(6:number_steps-2),bench_backlash_M2TX(6:number_steps-2),PSYM=-
5,linestyle=1,$
TITLE='Interpolated Backlash for AU1.M2TX when stage moves forward',$
XTITLE='AU1.M2TX actuator position around nominal position (in
mm)',XRANGE=[7.2,7.6],$
YTITLE='Interpolated backlash (arcseconds)',YRANGE=[18,21]
oplot,s_M2TX,fitting_M2TX_interpol,PSYM=-2,linestyle=1
DEVICE, /CLOSE

theta_M1X = 1.390580e-06*pos_M1X^4 + 2.812652e-06*pos_M1X^3 + 1.259884e-
04*pos_M1X^2 + 9.299078e-01*pos_M1X - 1.168302e+01
slope1=LINFIT(pos_M1X,theta_M1X)
a1_M1TX  = slope1[0]
b1_M1TX  = slope1[1]
print,'slope1 for M1X: ',slope1
fitting_M1TX_interpol=fitting_M1TX_interpol/3600
bench_backlash_mm_M1X=fitting_M1TX_interpol/b1_M1TX
print,''
print,'bench_backlash_mm_M1X:',bench_backlash_mm_M1X

theta_M2X = 1.508692e-06*pos_M2X^4 - 4.099467e-06*pos_M2X^3 + 1.249980e-
05*pos_M2X^2 + b_M2X*pos_M2X - a_M2X
slope2=LINFIT(pos_M2X,theta_M2X)
a1_M2TX  = slope2[0]
b1_M2TX  = slope2[1]
print,'slope2 for M2X: ',slope2
fitting_M2TX_interpol=fitting_M2TX_interpol/3600
bench_backlash_mm_M2X=fitting_M2TX_interpol/b1_M2TX
print,''
print,'bench_backlash_mm_M2X:',bench_backlash_mm_M2X

save,s_M1TX,s_M2TX,bench_backlash_mm_M1X,bench_backlash_mm_M2X,filenam
e='/home/ao/testdata/au_backlash/AU1/trajectory/focus_compensation.sav'

end
```

****

CODING 21

```
PRO focus_compensation,number_steps,T

Restore,filename='/home/ao/testdata/au_backlash/AU1/trajectory/focus_compensatio
n.sav'
s_M1TX=s_M1TX
s_M2TX=s_M2TX
bench_backlash_mm_M1X=bench_backlash_mm_M1X
bench_backlash_mm_M2X=bench_backlash_mm_M2X
```

```
;; First changing the distance between M1 and M2 then tilting the Mirror AU1 M1 and M2 and recording the image of defocused spot

;; nominal position for AU1 M1 and M2

M1X0= 7.11236  ; mm
M2X0= 7.29337
M1Y0= 12.84165
M2Y0= 12.99195
M1Z0=-43.83951

;; moving AU1.M1TY and AU1.M2TY to M1Y0 and M2Y0 position respectively-

print,'nominal position for AU1.M1TX is : ', M1X0
print,''
print,'nominal position for AU1.M2TX is : ', M2X0
print,''
print,'nominal position for AU1.M1TY is : ', M1Y0
print,''
print,'nominal position for AU1.M2TY is : ', M2Y0
print,''

;; moving the actuators at the nominal position

cmd='au1 base'
spawn,cmd,ret
wait,3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TY is : ', actual_position
print,''
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TY is : ', actual_position
print,''
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1Z, double *)"'
SPAWN, cmd, ret
```

```
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1Z is : ', actual_position
print,''

;; moving every actuator backward by 0.1 mm from the nominal position so that
measurement must start at the nominal position

command=-0.1
print,'moving the AU1.M1Z backward by 0.1 mm : '
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M1Z, ' +
STRING(command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3;
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1Z, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1Z is : ', actual_position
print,''
print,'moving the AU1.M1TX backward by 0.1 mm : '
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M1TX, ' +
STRING(command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
print,'moving the AU1.M1TY backward by 0.1 mm : '
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M1TY, ' +
STRING(command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TY is : ', actual_position
print,''
print,'moving the AU1.M2TX backward by 0.1 mm : '
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M2TX, ' +
STRING(command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,'moving the AU1.M2TY backward by 0.1 mm : '
```

```
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M2TY, ' +
STRING(command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TY is : ', actual_position
print,''

;; again moving to the actuator position

cmd='au1 base'
spawn,cmd,ret
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TY is : ', actual_position
print,''
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TY, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TY is : ', actual_position
print,''
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1Z, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1Z is : ', actual_position
print,''
cmd='aqcam 1 dir
/home/ao/testdata/au_backlash/AU1/trajectory/focus_backlash_pos'
spawn,cmd,ret;
cmd='aqcam 1 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(M1X0),/remove_all)+'_nominal_pos_'
spawn,cmd,ret
cmd='aqcam 1 fn '+strcompress(string(0),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
```

```
spawn,cmd,ret
wait,T+10

;; moving the stage in forward direction by 0.1 mm and also moving M1X and M2X
by the corresponding amount to not to lose the focus

displacement1=0.1
count1=0

for i=0,number_steps-1,1 do begin

count1=count1+1
print,'count1: ',count1
print,'the measurement position for AU1.M1Z under consideration is :'
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M1Z, ' +
STRING(displacement1, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1Z, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1Z is : ', actual_position
print,''
AU1_M1TX_command=s_M1TX(i)
print,'the measurement position for AU1.M1TX under consideration is :',
AU1_M1TX_command
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,';
AU1_M2TX_command=s_M2TX(i)
print,'the measurement position for AU1.M2TX under consideration is :',
AU1_M2TX_command
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
```

```
cmd='aqcam 1 dir
/home/ao/testdata/au_backlash/AU1/trajectory/focus_backlash_pos'
spawn,cmd,ret;
cmd='aqcam 1 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret;
cmd='aqcam 1 fp '+strcompress(string(s_M1TX(i)),/remove_all)+'_motion_pos_'
spawn,cmd,ret
cmd='aqcam 1 fn '+strcompress(string(i),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10

endfor

;;moving forward by 0.1 mm

command=0.1
print,'moving the AU1.M1Z forward by 0.1 mm : '
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M1Z, ' +
STRING(command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1Z, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1Z is : ', actual_position
print,''
print,'moving the AU1.M1TX forward by 0.1 mm : '
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M1TX, ' +
STRING(command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''

print,'moving the AU1.M2TX forward by 0.1 mm : '
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M2TX, ' +
STRING(command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3

cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
```

```
;; calculating the backlash when stage moves in negative direction

displacement2=-0.1
count2=0

for i=number_steps-1,0,-1 do begin

count2=count2+1
rint,'count2: ',count2
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M1Z, ' +
STRING(displacement2, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1Z, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1Z is : ', actual_position
print,''
AU1_M1TX_command=s_M1TX(i)
print,'the measurement position for AU1.M1TX under consideration is :',
AU1_M1TX_command
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
AU1_M2TX_command=s_M2TX(i)
print,'the measurement position for AU1.M2TX under consideration is :',
AU1_M2TX_command
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
print,''
cmd='aqcam 1 dir
/home/ao/testdata/au_backlash/AU1/trajectory/focus_backlash_neg'
spawn,cmd,ret
cmd='aqcam 1 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(s_M1TX(i)),/remove_all)+'_motion_neg_'
spawn,cmd,ret
```

```
cmd='aqcam 1 fn '+strcompress(string(count2-1),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10

endfor

;; compensating for the backlash

displacement2=-0.1
count2=0

for i=number_steps-1,0,-1 do begin

count2=count2+1
print,'count2: ',count2
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveRelative (AU1.M1Z, ' +
STRING(displacement2, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1Z, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1Z is : ', actual_position
print,''
AU1_M1TX_command=s_M1TX(i)-bench_backlash_mm_M1X(i)
print,'the measurement position for AU1.M1TX under consideration is :',
AU1_M1TX_command
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M1TX, ' +
STRING(AU1_M1TX_command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M1TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M1TX is : ', actual_position
print,''
AU1_M2TX_command=s_M2TX(i)-bench_backlash_mm_M2X(i)
print,'the measurement position for AU1.M2TX under consideration is :' ,
AU1_M2TX_command
cmd = './xps_raw.tcl 10.0.0.51 "GroupMoveAbsolute (AU1.M2TX, ' +
STRING(AU1_M2TX_command, FORMAT='(F9.6)') + ')"'
SPAWN, cmd, ret
wait, 3
cmd = './xps_raw.tcl 10.0.0.51 "GroupPositionCurrentGet (AU1.M2TX, double *)"'
SPAWN, cmd, ret
actual_position=double((STRSPLIT(ret, /EXTRACT))[1])
print, 'actual position of AU1.M2TX is : ', actual_position
```

```
cmd='aqcam 1 dir /home/ao/testdata/au_backlash/AU1/trajectory/stage_negative'
spawn,cmd,ret
cmd='aqcam 1 et '+ strcompress(string(T),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 fp '+strcompress(string(s_M1TX(i)),/remove_all)+'_stage_neg_'
spawn,cmd,ret
cmd='aqcam 1 fn '+strcompress(string(count2-1),/remove_all)
spawn,cmd,ret
cmd='aqcam 1 snap'
spawn,cmd,ret
wait,T+10

endfor

end
```

****

CODING 22

```
PRO read_gaussian_focus,number_steps

Restore,filename='/home/ao/testdata/au_backlash/AU1/trajectory/focus_compensatio
n.sav'
s_M1X=s_M1TX
s_M2X=s_M2TX
bench_backlash_mm_M1X=bench_backlash_mm_M1X
bench_backlash_mm_M2X=bench_backlash_mm_M2X
dz=0.1+findgen(number_steps)*0.1

M1X0= 7.11236  ; mm
M2X0= 7.29337
M1Y0= 12.84165
M2Y0= 12.99195
M1Z0=-43.83951

;;reading the gaussian center of the images after backlash compensation for
AU1M1TX

x_pixel_pos=fltarr(number_steps)
y_pixel_pos=fltarr(number_steps)
x_pixel_neg=fltarr(number_steps)
y_pixel_neg=fltarr(number_steps)
pos=fltarr(number_steps)
neg=fltarr(number_steps)
pos_img=fltarr(number_steps)
neg_img=fltarr(number_steps)
```

```
;; reading the center of the nominal position

back_img=READFITS('/home/ao/testdata/au_backlash/AU1/trajectory/stage_positive
/7.11236_nominal_00000.fits')
size_back=size(back_img)
smooth_back=smooth(back_img,20,/edge_truncate)
temp1=max(smooth_back,index1)
x_back=index1 mod size_back[1]
y_back=index1 / size_back[1]
print,''
print,'index of the brightest pixel for M1X at the nominal position is: ',x_back,y_back

;; cropping the image

crop=back_img[x_back-10:x_back+10,y_back-10:y_back+10]
fit=gauss2dfit(crop,/tilt,param)
x_pixel=x_back-10+param[4]+1
y_pixel=y_back-10+param[5]+1
print,'gaussian x and y value for M1X(back)',x_pixel,'',y_pixel
print,''

;; reading the spot and finding the backlash value when the stage moves in the
reverse direction

filename='read_focus_backlash.dat'
openw,lun,filename , /get_lun ;,/append
count1=9-indgen(number_steps)

for loop_step=0,number_steps-1,1 do begin

pos=strcompress(string(s_M1X(loop_step)),/remove_all)+'_motion_pos_0000'+strcom
press(string(loop_step),/remove_all)+'.fits'
neg=strcompress(string(s_M1X(loop_step)),/remove_all)+'_motion_neg_0000'+strcom
press(string(count1(loop_step)),/remove_all)+'.fits'
pos_img=READFITS('/home/ao/testdata/au_backlash/AU1/trajectory/focus_backlash
_pos/'+string(pos))
neg_img=READFITS('/home/ao/testdata/au_backlash/AU1/trajectory/focus_backlash
_neg/'+string(neg))

size_pos=size(pos_img)
size_neg=size(neg_img)

smooth_pos=smooth(pos_img,20,/edge_truncate)
smooth_neg=smooth(neg_img,20,/edge_truncate)

temp1=max(smooth_pos,index1)
temp2=max(smooth_neg,index2)
x_pos=index1 mod size_pos[1]
y_pos=index1 / size_pos[1]
```

```
x_neg=index2 mod size_neg[1]
y_neg=index2 / size_neg[1]
print,''
print,'index of the brightest pixel for M1X(pos)',x_pos,y_pos
print,''
print,'index of the brightest pixel for M1X(neg)',x_neg,y_neg
print,''

;; cropping the image

crop_pos=pos_img[x_pos-10:x_pos+10,y_pos-10:y_pos+10]
crop_neg=neg_img[x_neg-10:x_neg+10,y_neg-10:y_neg+10]
fit_pos=gauss2dfit(crop_pos,/tilt,param_pos)
fit_neg=gauss2dfit(crop_neg,/tilt,param_neg)
x_pixel_pos(loop_step)=x_pos-10+param_pos[4]+1
y_pixel_pos(loop_step)=y_pos-10+param_pos[5]+1
x_pixel_neg(loop_step)=x_neg-10+param_neg[4]+1
y_pixel_neg(loop_step)=y_neg-10+param_neg[5]+1
print,'gaussian x and y value for
M1X(pos)',x_pixel_pos(loop_step),'',y_pixel_pos(loop_step)
print,''
print,'gaussian x and y value for
M1X(neg)',x_pixel_neg(loop_step),'',y_pixel_neg(loop_step)
print,''
printf,lun,s_M1X(loop_step),s_M2X(loop_step),x_pixel_pos(loop_step),y_pixel_pos(loo
p_step),x_pixel_neg(loop_step),y_pixel_neg(loop_step),FORMAT='(6F12.6)'

endfor

close,lun

filename='read_focus.dat'
openw,lun1,filename , /get_lun ;,/append

pos1=fltarr(number_steps)
neg1=fltarr(number_steps)
pos_img1=fltarr(number_steps)
neg_img1=fltarr(number_steps)

x_pixel_pos1=fltarr(number_steps)
y_pixel_pos1=fltarr(number_steps)
x_pixel_neg1=fltarr(number_steps)
y_pixel_neg1=fltarr(number_steps)

;; reading the spot center and calculating the compensated backlash value

count=9-indgen(number_steps)

for loop_step=0,number_steps-1,1 do begin
```

```
pos1=strcompress(string(s_M1X(loop_step)),/remove_all)+'_stage_pos_0000'+strcomp
ress(string(loop_step),/remove_all)+'.fits'
neg1=strcompress(string(s_M1X(loop_step)),/remove_all)+'_stage_neg_0000'+strcom
press(string(count(loop_step)),/remove_all)+'.fits'
pos_img1=READFITS('/home/ao/testdata/au_backlash/AU1/trajectory/stage_positive
/'+string(pos1))
neg_img1=READFITS('/home/ao/testdata/au_backlash/AU1/trajectory/stage_negativ
e/'+string(neg1))

size_pos1=size(pos_img1)
size_neg1=size(neg_img1)

smooth_pos1=smooth(pos_img1,20,/edge_truncate)
smooth_neg1=smooth(neg_img1,20,/edge_truncate)

temp1=max(smooth_pos1,index1)
temp2=max(smooth_neg1,index2)

x_pos1=index1 mod size_pos1[1]
y_pos1=index1 / size_pos1[1]

x_neg1=index2 mod size_neg1[1]
y_neg1=index2 / size_neg1[1]
print,''
print,'index of the brightest pixel for M1X(pos)',x_pos1,y_pos1
print,''
print,'index of the brightest pixel for M1X(neg)',x_neg1,y_neg1
print,''

;; cropping the image

crop_pos1=pos_img1[x_pos1-10:x_pos1+10,y_pos1-10:y_pos1+10]
crop_neg1=neg_img1[x_neg1-10:x_neg1+10,y_neg1-10:y_neg1+10]
fit_pos1=gauss2dfit(crop_pos1,/tilt,param_pos1)
fit_neg1=gauss2dfit(crop_neg1,/tilt,param_neg1)
x_pixel_pos1(loop_step)=x_pos1-10+param_pos1[4]+1
y_pixel_pos1(loop_step)=y_pos1-10+param_pos1[5]+1
x_pixel_neg1(loop_step)=x_neg1-10+param_neg1[4]+1
y_pixel_neg1(loop_step)=y_neg1-10+param_neg1[5]+1
print,'gaussian x and y value for
M1X(pos)',x_pixel_pos1(loop_step),'',y_pixel_pos1(loop_step)
print,''
print,'gaussian x and y value for
M1X(neg)',x_pixel_neg1(loop_step),'',y_pixel_neg1(loop_step)
printf,lun1,s_M1X(loop_step),s_M2X(loop_step),x_pixel_pos1(loop_step),y_pixel_pos1
(loop_step),x_pixel_neg1(loop_step),y_pixel_neg1(loop_step),FORMAT='(6F12.6)'

endfor
```

```
close,lun1

;; reading the file for calculating the backlash when stage moves in reverse direction

data=fltarr(6,number_steps)
filename='read_focus_backlash.dat'
openr,lun,filename , /get_lun
readf,lun,data
close,lun

diff=fltarr(number_steps)
difference=fltarr(number_steps)

for i=0,number_steps-1,1 do begin

if ((data[2,i]-data[4,i]) lt 0) then begin

diff=(sqrt((data[2,i]-data[4,i])^2+(data[3,i]-data[5,i])^2))*1
difference(i)=diff

endif else begin

diff=(sqrt((data[2,i]-data[4,i])^2+(data[3,i]-data[5,i])^2))*(-1)
difference(i)=diff

endelse

print,''
print,'backlash when stage moves in reverse direction (in pixels): ',difference(i)
print,'backlash in  arcseconds',difference(i)*0.02
print,''

endfor

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'focus_backlash_pixel.ps', /INCHES
plot,dz,difference,PSYM=-5,linestyle=1,$
TITLE='Focus change for AU1 because of the backlash',$
XTITLE='Relative positions for AU1.M1Z (in mm)',XRANGE=[0,1.1],$
YTITLE=' change in the focus position (pixel)'
;oplot,dz,difference2,PSYM=-6,linestyle=1
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'focus_backlash_arcsecond.ps', /INCHES
plot,dz,difference*0.02,PSYM=-5,linestyle=1,$
TITLE='Focus change for AU1 because of the backlash',$
XTITLE='Relative positions for AU1.M1Z (in mm)',XRANGE=[0,1.1],$
YTITLE='change in the focus position (arcsecond)'
```

```
DEVICE, /CLOSE

;; reading the file for calculating the compensated backlash

data1=fltarr(6,number_steps)
filename='read_focus.dat'
openr,lun1,filename , /get_lun
readf,lun1,data1
close,lun1
diff1=fltarr(number_steps)
difference1=fltarr(number_steps)
diff2=fltarr(number_steps)
difference2=fltarr(number_steps)

for i=0,number_steps-1,1 do begin

if ((510.256989-data1[2,i]) lt 0) then begin
diff1=(sqrt((data1[2,i]-510.256989)^2+(data1[3,i]-522.909851)^2))*1
difference1(i)=diff1

endif else begin

diff1=(sqrt((data1[2,i]-data1[2,0])^2+(data1[3,i]-data1[3,0])^2))*(-1)
difference1(i)=diff1

endelse

print,''
print,'the difference between the nominal position and the spot position when the
stage moves in positive direction (in pixels) : ',difference1(i)
print,'difference in arcseconds',difference1(i)*0.02
print,''

if ((data1[2,i]-data1[4,i]) lt 0) then begin

diff2=(sqrt((data1[2,i]-data1[4,i])^2+(data1[3,i]-data1[5,i])^2))*1
difference2(i)=diff2

endif else begin

diff2=(sqrt((data1[2,i]-data1[4,i])^2+(data1[3,i]-data1[5,i])^2))*(-1)
difference2(i)=diff2

endelse

print,''
print,'after focus compensation (in pixels): ',difference2(i)
print,'focus compensation in arcseconds',difference2(i)*0.02
endfor
```

```
SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'focus_compensation_pixel.ps', /INCHES
plot,dz,difference,PSYM=-5,linestyle=1,$
TITLE='Focus compensation at the Optical Bench for AU1',$
XTITLE='Relative positions for AU1.M1Z (in mm)',XRANGE=[0,1.1],$
YTITLE='Focus compensation (pixel)',YRANGE=[-0.5,2.0]
oplot,dz,difference2,PSYM=-6,linestyle=1
DEVICE, /CLOSE

SET_PLOT, 'PS'
DEVICE,/portrait,filename= 'focus_compensation_arcsecond.ps', /INCHES
plot,dz,difference*0.02,PSYM=-5,linestyle=1,$
TITLE='Focus compensation at the Optical Bench for AU1',$
XTITLE='Relative positions for AU1.M1Z (in mm)',XRANGE=[0,1.1],$
YTITLE='Focus compensation (arcsecond)',YRANGE=[-0.01,0.04]
oplot,dz,difference2*0.02,PSYM=-6,linestyle=1
DEVICE, /CLOSE

end
```

****

CODING 24

```
PRO pvt_traj,number_steps

Restore,filename='/home/ao/testdata/au_backlash/AU1/trajectory/focus_compensatio
n.sav'
s_M1TX=s_M1TX
s_M2TX=s_M2TX
bench_backlash_mm_M1X=bench_backlash_mm_M1X
bench_backlash_mm_M2X=bench_backlash_mm_M2X

M1X0= 7.11236  ; mm
M2X0= 7.29337
M1Y0= 12.84165
M2Y0= 12.99195
M1Z0=-43.83951
t=1   ;;seconds
mz=0.1  ;; mm
filename='pvt_traj.trj'
openw, lun, filename, /get_lun ;,/append
printf,lun,float(t),', ',mz,', ',mz/float(t),', ',s_M1TX(0)-M1X0,', ',(s_M1TX(0)-
M1X0)/float(t),', ', 0, ', ', 0, ', ',s_M2TX(0)-M2X0,', ',(s_M2TX(0)-M2X0)/float(t),', ', 0,',
',0, $
FORMAT='(F8.3,A2,F8.6,A2,
F8.6,A2,F8.6,A2,F8.6,A2,I1,A2,I1,A2,F8.6,A2,F8.6,A2,I1,A2,I1) '
for i=0,number_steps-2,1 do begin
```

```
printf,lun,float(t),', ',mz,', ',mz/float(t),', ',s_M1TX(i+1)-s_M1TX(i),', ',(s_M1TX(i+1)-
s_M1TX(i))/float(t),', ', 0, ', ', 0, ', ',s_M2TX(i+1)-s_M2TX(i),', ',(s_M2TX(i+1)-
s_M2TX(i))/float(t),+', ', 0,', ',0, $
FORMAT='(F8.3,A2,F8.6,A2,
F8.6,A2,F8.6,A2,F8.6,A2,I1,A2,I1,A2,F8.6,A2,F8.6,A2,I1,A2,I1) '

endfor

for j=i,1,-1 do begin

print,''
print,s_M1TX(j)
print,''
printf,lun,float(t),', -',mz,', -',mz/float(t),', -',(s_M1TX(j)-s_M1TX(j-1)),', -',((s_M1TX(j)-
s_M1TX(j-1))/float(t)),', -', 0, ', -', 0, ', -',(s_M2TX(j)-s_M2TX(j-1)),', -',((s_M2TX(j)-
s_M2TX(j-1)))/float(t),+', -', 0,', -',0, $
FORMAT='(F8.3,A3,F8.6,A3,
F8.6,A3,F8.6,A3,F8.6,A3,I1,A3,I1,A2,F8.6,A3,F8.6,A3,I1,A3,I1) '

endfor

printf,lun,float(t),', -',mz,', -',mz/float(t),', -',(s_M1TX(j)-M1X0),', -',0,', -', 0, ', ', 0, ', -
',(s_M2TX(j)-M2X0),', -',0,', -', 0,', -',0, $
FORMAT='(F8.3,A3,F8.6,A3,
F8.6,A3,F8.6,A3,F8.6,A3,I1,A3,I1,A3,F8.6,A3,F8.6,A3,I1,A3,I1) '
close, lun

end
```

****

CODING 25

```
PRO pvt_traj_comp,number_steps

Restore,filename='/home/ao/testdata/au_backlash/AU1/trajectory/focus_compensatio
n.sav'
s_M1TX=s_M1TX
s_M2TX=s_M2TX
bench_backlash_mm_M1X=bench_backlash_mm_M1X
bench_backlash_mm_M2X=bench_backlash_mm_M2X
M1X0= 7.11236  ; mm
M2X0= 7.29337
M1Y0= 12.84165
M2Y0= 12.99195
M1Z0=-43.83951
t=1   ;;seconds
mz=0.1  ;; mm
```

```
filename='pvt_traj_comp.trj'
openw, lun, filename, /get_lun ;,/append
printf,lun,float(t),', ',mz,', ',mz/float(t),', ',s_M1TX(0)-M1X0,', ',(s_M1TX(0)-
M1X0)/float(t),', ', 0, ', ', 0, ', ',s_M2TX(0)-M2X0,', ',(s_M2TX(0)-M2X0)/float(t),', ', 0,',
',0, $
FORMAT='(F8.3,A2,F8.6,A2,
F8.6,A2,F8.6,A2,F8.6,A2,I1,A2,I1,A2,F8.6,A2,F8.6,A2,I1,A2,I1) '

for i=0,number_steps-2,1 do begin

printf,lun,float(t),', ',mz,', ',mz/float(t),', ',s_M1TX(i+1)-s_M1TX(i),', ',(s_M1TX(i+1)-
s_M1TX(i))/float(t),', ', 0, ', ', 0, ', ',s_M2TX(i+1)-s_M2TX(i),', ',(s_M2TX(i+1)-
s_M2TX(i))/float(t),+', ', 0,', ',0, $
FORMAT='(F8.3,A2,F8.6,A2,
F8.6,A2,F8.6,A2,F8.6,A2,I1,A2,I1,A2,F8.6,A2,F8.6,A2,I1,A2,I1) '

endfor

for j=i,1,-1 do begin

print,s_M1TX(j-1)
print,"
print,bench_backlash_mm_M1X(j-1)
print,"
printf,lun,float(t),', -',mz,', ',mz/float(t),', -',(s_M1TX(j)-(s_M1TX(j-1)-
bench_backlash_mm_M1X(j-1))),', ',((s_M1TX(j)-(s_M1TX(j-1)-
bench_backlash_mm_M1X(j-1)))/float(t)),', -', 0, ', -', 0, ', -',(s_M2TX(j)-(s_M2TX(j-1)-
bench_backlash_mm_M2X(j-1))),', ',((s_M2TX(j)-(s_M2TX(j-1)-
bench_backlash_mm_M2X(j-1))))/float(t),+', -', 0,', -',0, $
FORMAT='(F8.3,A3,F12.10,A3,
F12.10,A2,F12.10,A3,F12.10,A3,I1,A3,I1,A3,F12.10,A3,F12.10,A2,I1,A3,I1) '

endfor

close, lun

end

********************************************************************************
********************************************************************************
```