T.C.
DOKUZ EYLUL UNIVERSITY
HEALTH SCIENCE INSTITUTE

SOFTWARE MODULE DEVELOPMENT FOR HIGH RESOLUTION PET SYSTEM

PINAR CELIK

BASIC NUCLEAR MEDICINE MASTER PROGRAM

MASTER THESIS

IZMIR–2006

T.C.
DOKUZ EYLUL UNIVERSITY
HEALTH SCIENCE INSTITUTE

SOFTWARE MODULE DEVELOPMENT FOR HIGH RESOLUTION PET SYSTEM

BASIC NUCLEAR MEDICINE MASTER PROGRAM

MASTER THESIS

PINAR ÇELİK

I. Advisor: Assoc. Prof. Dr. Gamze ÇAPA KAYA

II. Advisor: Prof. Dr. Uwe PIETRZYK

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Examining Committee Members

Assoc. Prof. Dr. Gamze ÇAPA KAYA
DEU, Nuc. Med. _____

Prof. Dr. Uwe PIETRZYK
Juelich Research Center, IME _____

Prof. Dr. Hatice DURAK
DEU, Nuc. Med. _____

Prof. Dr. Berna DEĞIRMENCİ
DEU, Nuc. Med. _____

Asst. Prof. Dr. Özhan ÖZDOĞAN
DEU, Nuc. Med. _____

LIST OF CONTENTS

LIST OF EQUATIONS

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

LIST OF OUTPUTS

LIST OF ABBREVATIONS

PET	Positron emission tomography
CCC	Crystal Clear Collebration
IDL	Interactive Data Language
^{68}Ge	Germanium-68
^{18}F	Flour-18
$\Delta N/\Delta t$	Average decay rate
Λ	Decay constant
N	Number of radioactive atom
A	Activity
N(t)	Number of radioactive atoms as a function of time
N(0)	Number of radioactive atoms at the initial time
$e^{-\Delta t}$	Decay function
$T_{1/2}$	Half life
β	Beta
α	Alfa
β^+	Positron
β^-	Negatron
e-	electron
C	light speed
m	Mass
E	Enerji
υ	Neutrino
υ^-	Antineutrino
P	Proton
n	Neutron
γ	Gamma rays
I	Intensity
I_o	Incident intensity
x	Depth
μ	Linear attenuation coefficient
^{18}F	Flour 18
^{18}O	Oxygen 18

FDG	Flouro deoxy glucose
$H_2[^{18}O]$	Water molecule of Oxygen 18
$H_2[^{15}O]$	Water molecule of Oxygen 18
$C[^{15}O]$	Carbonmonoxide
$[^{13}N]H_3$	Amonia
^{13}N	Nitrogen 13
^{11}C	Carbon 11
^{137}Cs	Cesium 137

ACKNOWLEDGMENT

I WOULD LIKE TO THANK …

… to Prof. Dr. Gul Guner, Prof. Dr. Ulrich Scherer and Prof. Dr. Hatice Durak for their help providing me to be Socrates- Erasmus student in Germany.

… to my advisor Assoc. Dr. Gamze Capa Kaya for her sister like behavior and great support in my thesis.

… to my advisor Prof. Dr. Uwe Pietrzyk in Germany for his goodwill, patience and providing me all opportunities for my study. He behaved me as if I was his college not only his student. I will never forget his and his wife's wonderful Turkish food party that they have prepared for me.

… to Dr. Maryam Khodaverdi whom I feel as my second advisor in Germany for her kindly, friendly and genial approach.

… to Dirk Jahnsen who tries to teach me programming from the beginning level.

… to Dr. Cristoph Palm who was always kind and helpful to me.

… to Onur Tugcu, who was a student of the Mechanical Engineering Department of Fachhochschule Aachen. But now he is one of the precious students of the Maltepe University in Computer Engineering Program. Thanks are not enough for him. If he wasn't there this study would not finish and therefore I'm grateful for meeting him.

… to Prof. Dr. Aysegul Temiz Artmann for her moral support when I felt hopeless. I never forget her kindness. I also thank her student Peter for his helps.

… to Sabine Brinker who does her best in International Student Affairs of Fachhochschule.

… to my lovely friend Tugba Vilken with whom I lived my best days in Germany.

… to Omer Sekeroglu who was always with me and helped me improving my self confidence to succeed in this thesis.

… to all my friends and teachers in Nuclear Medicine Department of Dokuz Eylul University, to my room mates and everyone whom I took help and support at Juelich Research Center Institute of Medicine and also to my sweet friends who are living in guest house in Solarcampus.

… lastly to my parents, my sister and her husband and all my relatives for their endless love, support, trust and encouragement during my thesis study.

(DOKUZ EYLÜL ÜNIVERSITESI NÜKLEER TIP AD VE SAĞLIK BILIMLERI ENSTITÜSÜ SAYESINDE TANIDIĞIM ,TÜM ÖĞRENCILIK HAYATIM BOYUNCA BANA DESTEĞINI, SEVGISINI VE ARKADAŞLIĞINI VERMIŞ OLAN …

Her kafam karıştığında sizin o sihirli odanızdan düzelmiş olarak çıktım.Bende çok emeğiniz var. Size nasıl teşekkür edilir ki?...Siz bize Allah'ın bir lütfusunuz Hatice Hocam. İyi ki varsınız.

Hep mantığın sesiydin, hep doğruları söyledin ve yol gösterdin. Bazen anladım seni, eh bazen de geç anladım☺ Kardeşin adaşımdı, kardeşin yerine koyup sevdin. Her şey için teşekkürler Gamze Ablcığım. O tatlı iki fıstıkla mutluluklar diliyorum.

Bana hep güvendiniz, beni hep dinleyip değer verdiniz. Her zaman kapınızı açtınız.İyi niyetiniz ve yardımlarınız için ,bana gözden kaçırdığım en önemli şeyleri hatırlattığınız için teşekkür ederim Berna Hocam.

Beni bu bölüme almak için çok uğraştınız, ama ne yapalım olmadı.. Belki bu bölümde yardım etmek için en çok çaba harcadığınız kişilerdenim. Erkan Hocam.bana verdiğiniz değer ve yardımlarınız için çok teşekkür ederim

Recep Ağabey örnek alınacak çok yönün var. Sakinliğin ve olaylara hakimiyetini her zaman takdir ediyorum. Yardımların için teşekkürler.Umarım ailenle hep mutlu olursun.

Özhan Ağabey kafan hep bir şeylerle meşguldür☺ Hep çalışkan, titiz biri olarak hatırlayacağım seni. Umarım çalışkanlığının mükafatını kat kat görürsün.Yardımseverliğin ve arkadaşlığın için teşekkürler.

Ağabeyimsin, nükleer tıpı seçmemin sebebisin. Umarım kardeşliğine layık olmuşumdur İsmail Ağabeyciğim. Bazen kızdın bana, bazen şımarmayayım diye övmedin☺ Ama ben anladım gözlerinden neyi yapıp, neyi yapmamam gerektiğini. Öğrettiğin her şey için, dostluğun için sana çok teşekkür ederim.

Canım Bağnu Ablam benim. Hep yanımdaydın, hep bana destek oldun. Sırlarımı paylaştım seninle. Hem iyi bir hoca, hem meslektaş, hem arkadaş oldun. Ömür boyu dost kalırız inşallah. Her şey için teşekkürler.

Biraz geç kaynaştık ne yazık ki .O ciddi duruşun altında yumuşacık, harika bir kalp var. Az da olsa bunu keşfedebildiğim için çok mutluyum. Dünyana tam olarak girebilenler çok daha şanslıdırlar sanırım.. Teşekkürler her şey için Türkan Abla.

Her zaman İstanbul beyefendisiydin. Bir kere şu bölümde sesinin yükseldiğini görmedim.. Bana verdiğin akıllar için teşekkürler. İyi ki tanıdım seni Özden Ağabey.

Kağancığım benim, canım dostum...Sana uzun laflar etmeye gerek yok ki. Benim için değerin anlatılmaz çünkü, sen bunu zaten çok iyi biliyorsun....

Yaşar bana takılmalarını çok özleyeceğim. İlerde iyi bir nükleer tıp uzmanı olarak çok güzel yerlere geleceğine eminim. Yolun açık olsun.

Hasan gittiğim yerde ben kime takılıp, dalga geçeceğim ya da kimin laflarına takacağım bakalım. İlerde sana bol şans ve bol para diliyorum kardeşim(Dileklerin en güzelini yaptım sanırım.

Cafer her sabah günün yorumunu yapmak keyifliydi. Sizin memleketten bir kart atarsın artık ilerde☺ Ailenle mutluluklar.

Sadetciğim düğünündeki güzelliğin hala gözümün önünde. Umarım hayatın hep böyle güzel olur, işte de evde de hep mutlu olursun.

Yeni arkadaşlarım Erdem ve Tarık. Dilerim Nükleer Tıp size uğur getirir, iyi ki gelmişiz dersiniz. İkinize de bol şanslar.

O gülüşünü hep hatırlayacağım Mukaddes Hemşire Hanım. Ha bir de elinde liste "Allaah yine para istemek için geliyor" dediğimiz o sevimli yürüyüşünü☺ Her şey gönlünce olsun.

Tibetciğim hem arkadaşım, hem terapistim oldun. O uzun konuşmalarımızı hiç unutmayacağım.. Sana ömrünce mutluluklar ve o güzel evinde huzur dolu günler diliyorum canım.

Burcu Gülsüm, içtiğimiz günleri ve dostluğunu çok özleyeceğim. Aramızdaki yaş farkına rağmen frekanslarımızın çok uyduğuna inanıyorum. Ha bu arada bu Fener'den de adam olmaz söyliyim☺

Hüseyin Baba. Vallahi seni çok özleyeceğim. Her Orhan duyduğumda, her Fener maçında sen geleceksin aklıma. Samimiyetin hiç kaybolmasın.

Süslü Gülşahım benim. İnşallah bir kız çocuğun olur da biraz da o nasiplenir bu zevklerinden. Her şey gönlünce olsun.

Gülsümcüğüm hayatlarımız bir bakıma benzeşiyor. Umarım ben de senin gibi mutlu olurum. Dostluğun için teşekkürler.

Yeni anne Ebru Hanım☺ O dünya güzeli kızını kucağına aldığında içindeki huzur gözlerinden okunuyor. Bize de nasip olur inşallah. Her şey istediğin gibi olsun.

İkinci taze annemiz Serapcık. Sen ve oğlun bir mucize gerçekleştirdiniz. Hayatta hep böyle doğru kararlar vermen dileğiyle. Gülümsemen hiç eksilmesin.

Sana da her şey için teşekkür ediyorum Raziyeciğim, özellikle otlandığım sigaralar için☺ Mutluluklar.

Özlerciğim benim. Her şeyin en güzelini hakediyorsun. İsteklerine bir an önce kavuşmanı ve seni çok mutlu görmeyi diliyorum. İyi niyetin ve arkadaşlığın için teşekkürler.

İnşallah bir an önce iyi olacaksın, bize hikayelerini anlatıp, rüyalarımızı yorumlayacaksın Güler Ablacığım. O gülen yüzün hiç solmasın.

Her istediğine kavuşman dileğiyle Senarcığım. Yüksek lisansında başarılar.

Sizlere de her şey için teşekkürler Sema Hanım ve Ramazan Bey. Hakkınızı helal edin.

Bölümümüzden ayrılan Cengiz, Gülhan, Sarı Burcu, Küçük Burcu, Yasemin Hanım,, Fatma Hemşire Hanım, Meryem, Aytül, Nesli ve Figen, sevgili Manisa grubu; Feray, Dilek, Yasemin, Gökçen ve adını unutmuş olabileceğim (kendilerinden özür diliyorum) tüm sevgili arkadaşlarım, bana verdiğiniz değer, paylaştığımız şeyler, arkadaşlığınız ve yardımlarınız için çok teşekkür ediyorum.

Bu bölümü çok sevdim burda çok şey öğrendim. İsimlerini saydığım ve sayamadığım tüm bu saygıdeğer insanları bu bölüm sayesinde tanıdım. Bana kucağını açan, sevgisini, ve bilgisini esirgemeyen bu yeri çok özleyeceğim. Ömür boyu bağımızın kopmaması dileğiyle. Hepinize sevgilerimi yolluyorum.

Ayrıca öğrenim hayatım boyunca her zaman bana yardımcı olmak için ellerinden geleni yapan Dokuz Eylül Üniversitesi Sağlık Bilimleri Enstitüsü Müdürü Sayın Gül Hocam ve enstitü çalışanları Bahriye Hanım, Şencan Hanım, Asiye Hanım, Füsun Hanım, Alpaslan Bey, Ferhat Bey ve her iki Nevzat Beyler sizlere ve ismini sayamadığım bana emeği geçmiş olan herkese ayrı ayrı teşekkürü bir borç biliyorum.

…HERKESE TEŞEKKÜR EDERİM)

ABSTRACT

SOFTWAREMODULE DEVELOPMENT FOR HIGH RESOLUTION PET SYSTEMS

Aim: ClearPET is a small animal PET scanner device which is made by the CCC (Crystal Clear Collaboration) and used at the Juelich Research Center. A program based on IDL 6.1 (Interactive Data Language) named Module_Check written is aimed to be used for quality control of the ClearPET.

Material, Method: The Module_Check program was written to evaluate the information which come from each module group and are stored in .ang files during the detection. In the program, the user shall input the address of the data. After that, the necessary graphs and outputs can be selected. Module_Check was tested with the ^{68}Ge and blank scan data acquired while the gantry was in fixed state, and ^{18}F, ^{68}Ge and blank scan data taken during gantry rotation. Maximum, minimum and average values and their standard deviations were found by program. Module_Check was tested whether or not it could find errors which were already known by checking ASCII formatted data manually.

Results: Graphics taken while the gantry was in fixed state was linear. Graphics of the ^{68}Ge and ^{18}F counts taken during gantry rotation was like sinus curve. Minimum and maximum angle values were observed at about 90° and 270° in these sinusoidal graphics. Averages of minimum and maximum angle values of ^{68}Ge counts were determined as 53°±15° and 227°±12°. Besides, averages of minimum and maximum angle values of module group counts taken with ^{18}F source were calculated as 246°±16° and 75°±16°. Counter to ^{68}Ge, minimum angle value was between 180°-270° and while maximum value was between 0°-90° in ^{18}F measurements. Blank scan measurement gives a noisy line though taken during gantry rotation. % deviation between minimum and maximum counts for rotating gantry was 16.9 for ^{68}Ge and 5.6 for ^{18}F. This program can detect the errors. The program showed that Module Group 1 measurements were inaccurate for the blank scan data of in which error was known before. Additionally, the program found out that the measurements of Module Group 3 from 289th to 335th in ^{18}F experiment were inaccurate. This error wasn't known before the creation of the module group graphics.

Conclusion: Module_Check program runs correctly. This program may be suggested in order to use finding the errors of cassettes (module groups) of the ClearPET.

Keywords: ClearPET, Module_Check, module group, gantry rotation, sinusoidal graphic, error detection.

ÖZET

YÜKSEK REZOLÜSYONLU PET SISTEMLERI İÇIN MODÜL GELIŞTIRILMESI

Amaç: ClearPET CCC (Crystal Clear Collaboration) tarafından yapılan ve Jülich Araştırma Merkezi'nde kullanılan küçük hayvanların taranması için yapılmış bir PET cihazıdır. CearPET'in kalite kontrolünde kullanılması amacıyla Module_Check isimli IDL 6.1 (Interactive Data Language) tabanlı bir program yazılmıştır.

Materyal, Metod: Module_Check modül gruplarından gelen ve dedeksiyon boyunca .ang dosyalarında depolanan bilgileri değerlendirmek için yazılmıştır. Kullanıcı verinin adresini programa girmelidir. Sonrasında gereksinim duyulan grafikler ve çıktılar seçilebilir. Module_Check gantry sabit konumdayken çekilmiş ^{68}Ge ve boş tarama verileri ve gantry dönerken alınan ^{18}F, ^{68}Ge ve boş tarama verileri kullanılarak test edilmiştir. Maksimum, minimum, ortalama değerler ve bunların standart sapmaları program tarafından bulunmuştur. Ayrıca daha önceden ASCII formatlı verilerin manuel olarak kontrol edilmesiyle bulunmuş bir hatanın Module_Check tarafından saptanıp saptanamayacağı test edilmiştir.

Bulgular: Gantry sabitken alınan verilerin grafikleri lineer görünümdedir. Gantry dönerken alınan ^{68}Ge ve ^{18}F sayımlarının grafikleri sinüs eğrisine benzemektedir. Bu sinüs şeklindeki grafiklerde minimum ve maksimum açı değerleri yaklaşık 90° ve 270°'lerde gözlenmiştir. ^{68}Ge için minimum ve maksimum açıların ortalama değerleri 53°±15°ve 227°±12°olarak bulunmuştur. Ayrıca ^{18}F kaynağı ile alınan modül grup sayımları için minimum ve maksimum açı değerlerinin ortalamaları 246°±16°ve 75°±16°'dir. ^{18}F ölçümlerinin tam tersi, ^{68}Ge'un maksimum açı değerleri 0°-90° aralığında iken, minimum açı değerleri 180°-270° aralığında yer almaktadır. Boş tarama ölçümleri gantry dönerken alınmasına rağmen gürültülü bir çizgi şeklinde görünmektedir. Gantry dönerken alınan sayımlarda minimum ve maksimum arasındaki % sapma ^{68}Ge için 16.9, ^{18}F için 5.6'dır. Bu program hataları dedekte edebilmektedir. Program yanlışlığı daha önceden bilinen boş tarama verilerinde 1. Modül Grup sayımlarının hatalı olduğunu saptamıştır. Buna ek olarak program ^{18}F deneyindeki 3. Modül Grup'ta 289.ölçümden 335.'ye kadar olan ölçümlerin hatalı olduğunu saptamıştır. Bu hata modül grup grafikleri oluşturulmadan önce bilinmemekteydi.

Sonuç: Module_Check programı doğru olarak çalışmaktadır. Bu programın ClearPET kasetlerinin hatalarını bulması için kullanılması tavsiye edilebilir.

Anahtar Kelimeler: ClearPET, Module_Check, modül grubu, gantry dönüşü, sinüs şeklindeki grafik, hata dedeksiyonu.

1 INTRODUCTION

Positron Emission Tomography (PET) is a radiotracer imaging technique, in which tracer compounds labeled with positron-emitting radionuclides are injected into the subject of the study. PET has become an important tool for the early detection of disease, the understanding of basic molecular aspects of living organisms and the evaluation of medical treatment. In time scintillator materials were developed which improve stopping power and shorten the detection time with increasing the count rates and sensitivity. In recent years new techniques were introduced to increase the sensitivity using PET acquisition. PET is being used to examine the biological function in animals besides humans.

The ClearPET^{TM} LYSO/LuYAP phoswich scanner is a high performance small animal PET system that has been developed within the Crystal Clear Collaboration (CCC) and exists in Juelich Research Center. The gantry in which the 20 cassettes (modules) are fixed allows rotation of the detector modules around the field of view. ClearPET has approximately 1mm high resolution.

Getting data from a device is not sufficient enough itself. To reach good results we should be sure that images are true and accurate. That is why quality controls are very important for devices. To obtain significant results from the detector counts of the ClearPET the data were arranged via plotting graphs. A program based on IDL 6.1 (Interactive Data Language) was written to evaluate the information which come from each module group and are stored in .ang files during the detection. The data stored in ASCII format were read for ^{68}Ge and ^{18}F sources and blank scan. Some data files were measured during gantry rotation. As a result cassette position (angle) versus module group counts are plotted. On the other hand for the results of fixed gantry state measurements, the number of interval versus module group counts were plotted differently. The purpose of all these evaluations is to determine the detector errors as this program also works as a controller.

This thesis is made in Juelich Research Center- Germany, where I have been as Socrates –Erasmus exchange student and stayed between1 March- 31 August 2005. All the used data are taken from this center.

2 BASIC INFORMATIONS

2. 2 BASIC CHARACTERISTICS OF RADIATION

2.2.1 ACTIVITY, DECAY CONSTANT AND HALF LIFE

If a radionuclide has N number of radioactive atoms, the average decay rate for the sample is given as;

$$\Delta N / \Delta t = -\lambda N . \qquad \textbf{2. 1}$$

Where λ is decay constant for the radionuclide and its unit is $(\text{time})^{-1}$. The average decay rate ΔN/Δt is the activity of the sample.

A sample has an activity A of 1 Bq if it is decaying at a rate of 1 s^{-1} (1 dps*)

$$A(Bq) = \left|\Delta N / \Delta t\right| = \lambda N \qquad \textbf{2. 2}$$

The traditional unit for activity is curie (Ci), which is defined as 3.7×10^{10} dps*.

$$A(Ci) = \lambda N /(3.7 \times 10^{10}) \qquad \textbf{2. 3}$$

From the Equation 2.2 we can find the number of radioactive atoms as a function of time.

$$N(t) = N(0)e^{-\lambda t} \qquad \textbf{2. 4}$$

---

* disintegration per second

The factor $e^{-\lambda t}$ is the decay function, which is the fraction of radioactive atoms remaining after time t.

Since the activity A is proportional to the number of atoms N (Equation 2.2), the decay factor also applies to activity versus time. It means that the activity decays with time. Exponential decay is characterized by disappearance of a constant fraction of activity of number of atoms present per unit time interval.

The half life $T_{1/2}$ and decay constant λ of a radionuclide are related as [1]

$$T_{1/2} = \ln 2 / \lambda \qquad \textbf{2. 2}$$

$$\lambda = \ln 2 / T_{1/2} \qquad \textbf{2. 3}$$

2.2.2 INTERACTION OF PARTICLE RADIATION WITH MATTER

High energy charged particles, such as beta (β) or alpha (α) particles lose energy and slow down as they pass through matter. Except for differences in sign, the forces experienced by positive electron (positron) and negative electron (negatron) (β^+ and β^- particles) are identical. The collisions, which occur between charged particles and atoms (or molecules), involve electrical forces of attraction or repulsion. Sometime the strength is sufficient to separate an electron from its orbit in the atom, which is called as ionization.

Part of this transmitted energy is used to overcome the binding energy of the electron and the remainder is given to the ejected secondary electron as kinetic energy. Inner shell electron leads to the emission of characteristic X rays (Figure 2.1) or Auger electrons.

Another type of interaction occurs when the charged particle penetrates the orbital electron cloud of an atom and interacts with its nucleus. The particle will be deflected by the strong electrical forces exerted on it by the nucleus and loses energy. This energy appears as a photon called Bremsstrahlung [1] (Figure 2.1).

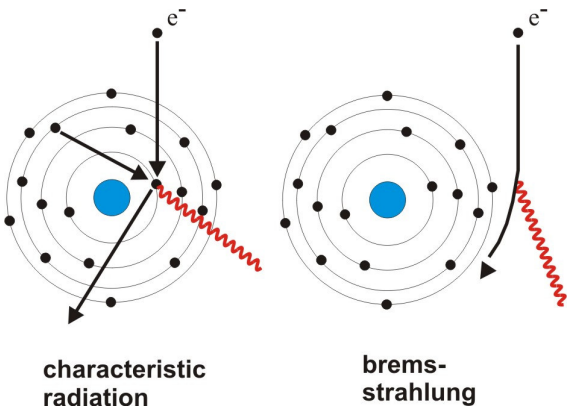


Figure 2.1: Characteristic radiation and Brehmsstrahlung [2].

2.2.2.1 Positron and Annihilation Process

Positron is the antiparticle of the electron. It has the same mass as the electron (9.11×10^{-31}kg or 0.511keV/c^2). The magnitude of its charge is also the same (1.6×10^{-19} Coulombs). But the charge of the positron is positive, opposite to the electron.

Antimatter has very special physical characteristics. When a particle contacts with its antiparticle, both particles convert into another form. Their mass converts to radiation energy, in accordance to of Einstein's Energy Equation $E=mc^2$.

In beta decay (β^-), a neutron rich nucleus converts one neutron to a proton and emits an electron and an antineutrino ($\bar{\upsilon}$).

$$n \rightarrow e^- + p + \bar{\upsilon} \qquad \textbf{2.7}$$

In positron decay (β^+) a proton in the nucleus converts to a neutron. In this reaction a positron and a neutrino (υ) are emitted.

$$p \rightarrow e^+ + n + \upsilon \qquad \textbf{2.8}$$

When positrons meet with matter, they come to a halt in about 10^{-12} s. Once their trajectory ends, there is a possibility that the positron annihilates with an electron (Figure 2.2) [3].

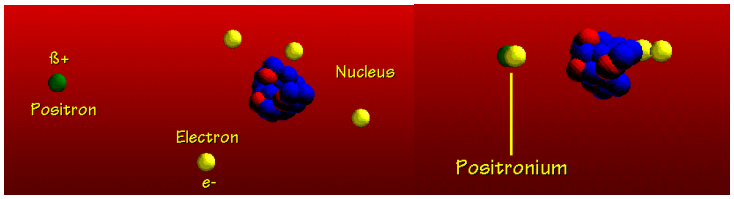


Figure 2. 2: The positron combines with an ordinary electron of a nearby atom in an annihilation reaction forming positronium as an intermediate [4].

The annihilation of a positron and electron is a matter-antimatter reaction process. During this process, all the mass that is annihilated is converted to different forms of energy.

Energy is conserved. Each photon will have energy of 511 keV. Momentums are also conserved. In a system of two photons, they will be emitted collinearly in opposite directions (Figure 2.3). 0.01% of anti-matter reactions result in more than two photons being emitted. This event can be neglected [3].

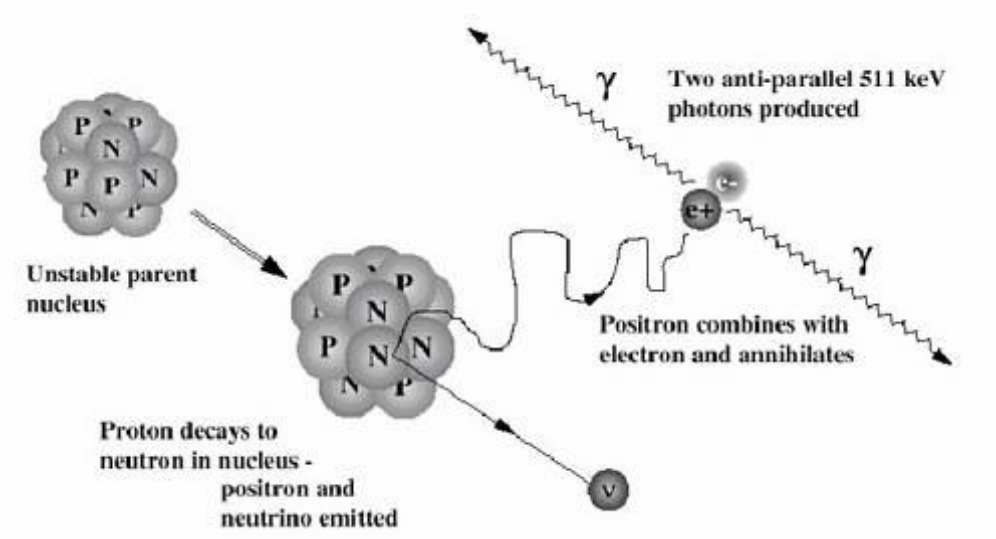


Figure 2. 3: When a positron comes in contact with an electron, the two particles annihilate turning the mass of the two particles into two 511 keV gamma rays that are emitted at 180° to each other. [5].

2.2.3 PHOTON INTERACTION MECHANISM

High energy photons (Gamma rays (γ), X-Rays, annihilation radiation and Bremsstrahlung) transfer their energy to matter in interaction with atoms, nuclei and electrons [1].

2.2.3.1 Compton Scattering

When photons collide with orbital electrons of an atom, they create an inelastic collision. For example when a gamma ray with a medium energy of between 2 KeV and 2 MeV collides with an atom, inelastic collision causes it to transfer some energy to some orbital electrons, which are knocked off the orbit. The photon changes direction because of this energy loss. This is called Compton scattering (Figure 2.4). Compton scattering increases with the electron number of the collided material [6].

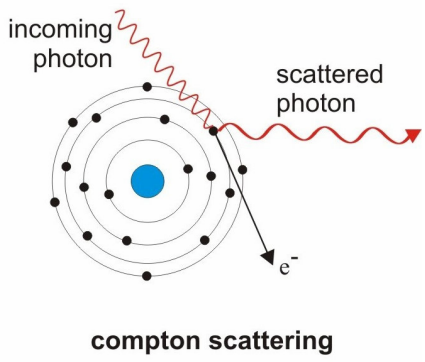


Figure 2. 4 Compton Scattering: [2].

2.2.3.2 Photoelectric Effect

When a photon hits an atom, it may also lose all its energy and knock off an orbital electron. The energy of the photon is used for removing and accelerating the electron to a certain velocity. This ionizing particle is what appears in the phosphors of the scintillation detectors [6] (Figure2.5).

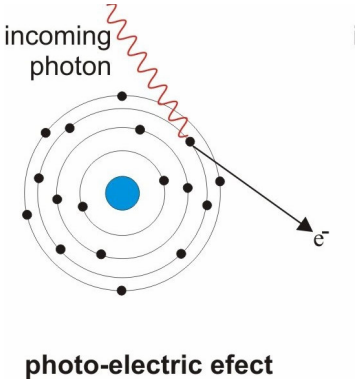


Figure 2. 5: Photoelectric Effect [2].

2.2.3.3 Attenuation

If a photon beam passes through the matter both interactions processes described above cause attenuation. Photons which undergo photoelectric absorption and Compton scattering are manipulated before they reach a detector. If photons come from a source both scattering and absorption reduce the number of photons reaching the detector. In some situations more photons are detected because some are scattered into the detector.

The intensity of photons can be written as

$$I = I_0 e^{-\mu x} \qquad \textbf{2. 9}$$

I_0 represents the incident intensity, x is the depth (units of length), μ is the linear attenuation coefficient, which has dimension of $(\text{length})^{-1}$ [7].

2.3 POSITRON EMISSION TOMOGRAPHY

2.3.1 BASIC PRINCIPLES OF A PET STUDY

A PET Centre requires a cyclotron and a radiotracer production system of PET radiopharmaceuticals, PET camera and computers for acquisition, reconstruction and analysis.

PET is based on the tracer principle. Tracers in nuclear medicine are compounds labeled with a radionuclide that emits radiation. The tracers used in PET are labeled with short lived positron emitting radio nuclides (Table 3.2). 2-[^{18}F] fluoro-2-deoxy-D-glucose (FDG) is one of the most important tracer in all fields of PET. It is used in oncology, neurology and cardiology.^{18}F is used for labeling. The chemical reaction is

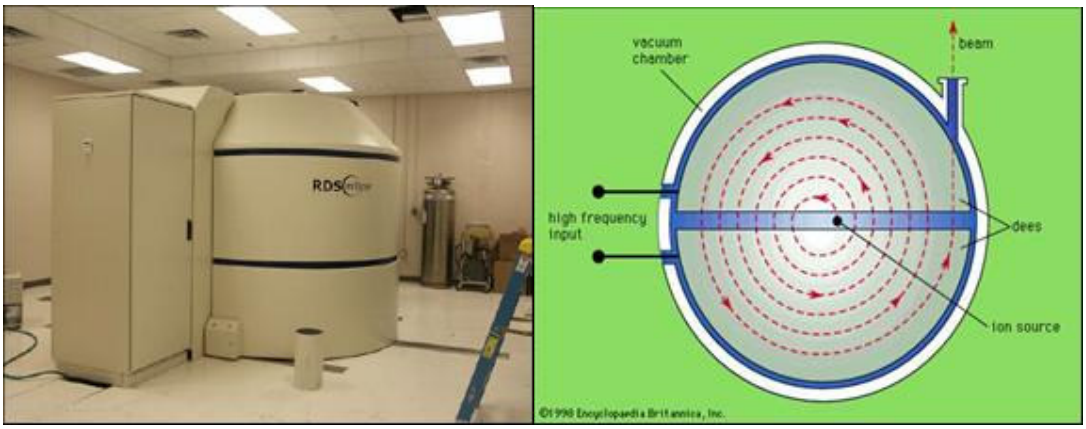
$$^{18}_{8}O+^{1}_{1}p\rightarrow^{18}_{9}F+^{1}_{0}n \qquad \textbf{2.10}$$

^{18}O target water is very expensive because the natural abundance of its stable isotope is only % 0.2. A liquid target of H_2 [^{18}O] produces ^{18}F in the form of fluoride ions in an aqueous solution ($^{18}F)_{aq}$. This reaction has a high cross-section, and also very high yield. In PET studies other tracers are used, too, such as ^{13}N for myocardial perfusion studies or ^{15}O to measure blood flow in cerebral perfusion, brain activation and cardiac perfusion (Table 2.1) [3].

Table 2.1: Most common PET radiopharmaceuticals and their physiologic imaging applications [3].

Radiopharmaceutical	Physiologic imaging application
$[^{15}O]_2$	Cerebral oxygen metabolism and extraction
$H_2[^{15}O]$	Cerebral and myocardial blood flow
$C[^{15}O]$	Cerebral and myocardial blood volume
[^{11}C]-N-methylspiperone	Cerebral dopamine receptor binding
[^{18}F]-fluorodeoxyglucose (FDG)	Cerebral and myocardial glucose metabolism and tumour localization
$[^{13}N]H_3$	Myocardial blood flow
[^{11}C]-acetate	Myocardial metabolism

The short half-lives of PET radio nuclides present several advantages such reduced patient dose, radiation exposure and possibility of repeated measurements. The evident disadvantage is that the radio nuclides must be produced locally, within the PET Centre or very close distance to working area. It is also needed an accelerator (10-20 MeV) capable of producing all desired radio nuclides. Cyclotrons are used in radionuclide production, including PET radio nuclides. They use magnetic and electric fields to accelerate the charged particles to high energies. The accelerating particles move in a circular orbit in a large magnetic field. They can produce high beam currents and it supplies high yield when producing radio nuclides. Most cyclotron- produced radio nuclides are neutron poor and therefore decay by positron emission or electron capture (Figure 2.6) [3].



a **b**

Figure 2.6: a) RDS Eclipse PET Cyclotron. b) The components of a simplified cyclotron.

Most common radionuclides and their half lives are shown in Table 2.2. Radioactively labeled tracers are injected to the patient. Then the tracer circulates through the body into an organ or lesion. The radiation in the body is detected by an external PET detector. So diseases can be investigated with an in-vivo technique [3].

Table 2.2: Most common used radio nuclides and their half lives [3].

Radionuclide	Half-life (min.)
^{18}F	109.8
^{11}C	20.4
^{15}O	2.05
^{13}N	9.96

A PET camera has scintillator based radiation detectors set up in a circle (Figure 2.7). Each detector consists of a scintillators crystal coupled to a photomultiplier tube (PMT). Images are formatted with coincidence detection systems, where annihilation radiation is used. In this type of radiation two photons are produced at the same time and with the same energy (511 keV). They are always emitted opposite to each other and collinear. These properties are used in coincidence detection techniques to form events from two gamma photons of the same annihilation [3]. A positron annihilation must have occurred somewhere on the line of two detector placed across each other (Figure 2.7).

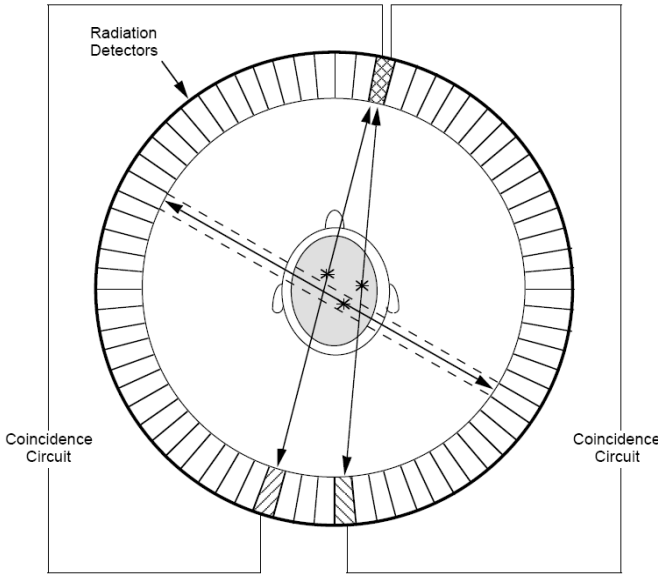


Figure 2.7: The PET camera surrounds the patient's body with multiple rings of gamma detectors, each of which can operate in coincidence with multiple opposing detectors [8].

After the acquisition, events are histogrammed in to projection, which are reconstructed to create an image. The images are then analyzed to extract qualitative and quantitative information.

PET cameras may come with radioactive sources of their own. A source based on ^{68}Ge has a half life about 270 days and needs to be replaced annually. This positron emission source is used for transmission scan to finally get attenuation map for correcting the images. Newer systems sometimes have transmission scanning, which is based on single photons emitted from ^{137}Cs or ^{133}Ba sources. These sources have a very long half life [3].

2.3.2 ANNIHILATION DETECTION

2.3.2.1 Coincidence Detection

PET uses the coincidence detection of positron annihilation radiation during the imaging. With these techniques, the location of positron annihilation can be detected. Two detectors which are positioned at opposite sides determine the distribution of positron emitting radioactivity (Figure 2.8). The detector outputs are connected to an electronic device. This device is called as “coincidence circuit and determines the relative time for each annihilation event [3].

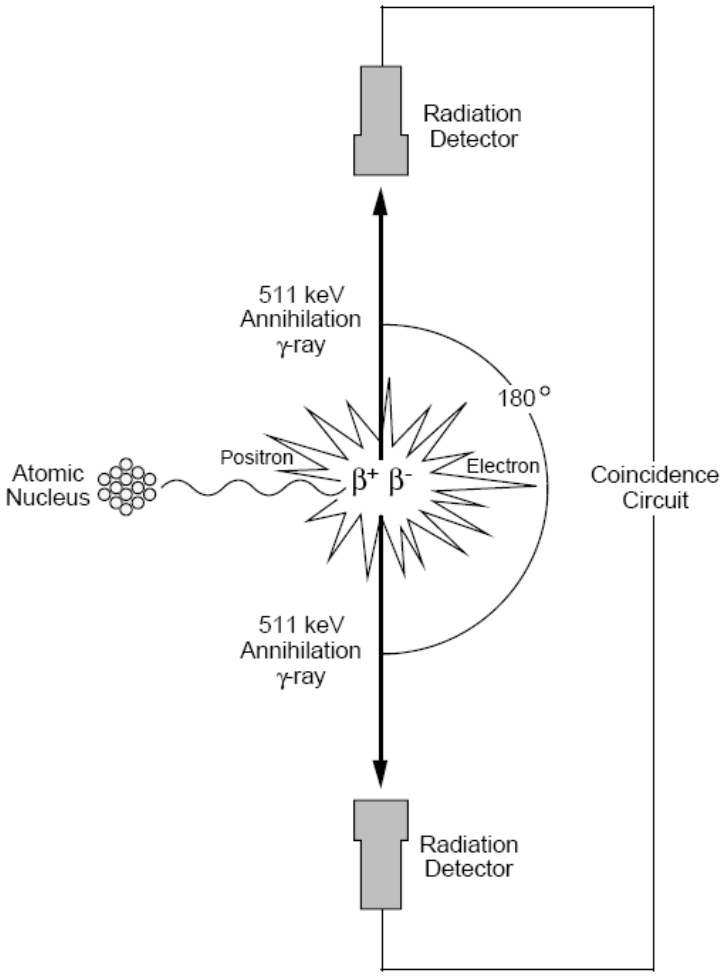


Figure 2.8: The PET camera employs "paired" gamma detectors linked in a coincidence circuit that only records decay events when photons trigger both detectors "simultaneously" [8].

2.3.2.2 Positron Range Effect

When a nucleotide decays, the energy emission is shared in between positrons and neutrinos. This causes the positron emissions to form a continuous spectrum. PET positron emitters mostly decay via process called positron decay. (^{18}F; 97% positron decay, 3% electron capture decay, ^{68}Ga; 88% positron decay and 12% electron capture decay) (Table 2.3). Relative probabilities of these decays are computed using the atomic number (Z) of the element.

A PET positron source or the patient emits positrons in an isotropic (towards all angles) way. The positrons interact with matter much like beta radiation. They make inelastic collisions with the matter. A small amount of Bremsstahlung X-Rays are also created. A large

amount of positrons cause tissue ionization. Positron ray paths are very erratic. Each positron stops at different distance from the point of emission. They rarely cover their maximum range. Their effect/distance ratio decreases roughly exponentially. Most of them stop half way in between source and maximum distance.

Emitted positrons cover different distances within the volume of effect. For example, positrons emitted from ^{15}O travel different distances within 2.5 mm, causing a blurry image of the emission. Most positrons travel less than 3 mm before undergoing an anti-matter reaction with an electron (Table 2.3). The photons emitted as a result, can be detected by PET scans [3].

Table 2.3: Positron emitting radio nuclides and their decay characteristics (Crump Institute for Molecular Imaging) [3].

Isotope	β^+ Fraction	Max. Energy	Range (mm)
^{11}C	0.99	0.96	0.4
^{13}N	1.00	1.20	0.7
^{15}O	1.00	1.74	1.1
^{18}F	0.97	0.63	0.3
^{22}Na	0.90	0.55	0.3
^{62}Cu	0.98	2.93	2.7
^{68}Ga	0.88	1.90	1.2
^{82}Rb	0.96	3.15	2.8
^{124}I	0.22	3.16	2.8

2.3.2.3 The Effect of Energy and Time Window in Detection

Two kinds of window are used for coincidence detection. One of them is energy window. With this window, photons with energy within a certain range are selected. This accepts only annihilation photons and rejects most scattered radiation and other events with different energy. The other window is coincidence timing window. It is set in the coincidence circuit and it records the maximum time allowed between the detection of two photons. So it determines a valid coincidence has occurred. If the coincidence time window is too long, the detected photons might originate from different annihilations and the counts of random

coincidences may be too large. If it is too short, the user may lose the true coincidence data. It is very important to set the coincidence window up correctly.

If annihilation occurs exactly at the middle between two detectors, the detectors shall detect the events at the same time. But if annihilation occurs closer to a detector than the other, a time lag occurs in between two detections, which depends on the difference in distance the photon has to travel.

Line of Response (LOR) is the line along which the annihilation occurred between two coincidence detectors. It is not an exact position, it only means the annihilation, occurred somewhere along the line which connected the two detectors (Figure 2.9) [3].

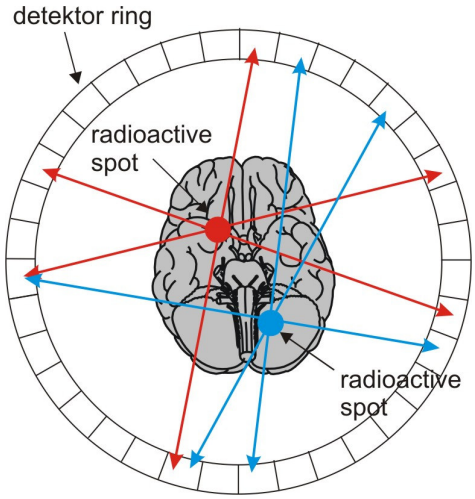


Figure 2.9: Red and blue arrows represent different LORs between detectors [2].

2.3.2.4 Types of Coincidences

PET may detect four types of logical events. A “true coincidence” occurs when both photons from the annihilation reaction are detected within the coincidence time window. These photons must not have interacted with anything else (Figure 2.10).

A “scattered coincidence” happens when the two photons experienced at least one Compton scattering event before the detection. This event causes LORs to detect true coincidences which are not correct. Scattered coincidences contribute the statistical noise to the signal and appears as a background to the true coincidence distribution. The amount of these events depends on the object geometry (Figure 2.10).

A random coincidence event is caused by photons from different reactions being detected within the coincidence time window. Object geometry also changes the number of these events. These uniform events can be estimated and subtracted from the final data in order to get a more accurate result (Figure 2.10).

A multiple coincidence event occurs when multiple photons hit the same LOR at the same interval. This event will be discarded, since it is impossible to determine which LOR these photons should be assigned to. Multiple coincidences can prevent reading the position and direction of a reaction [9].

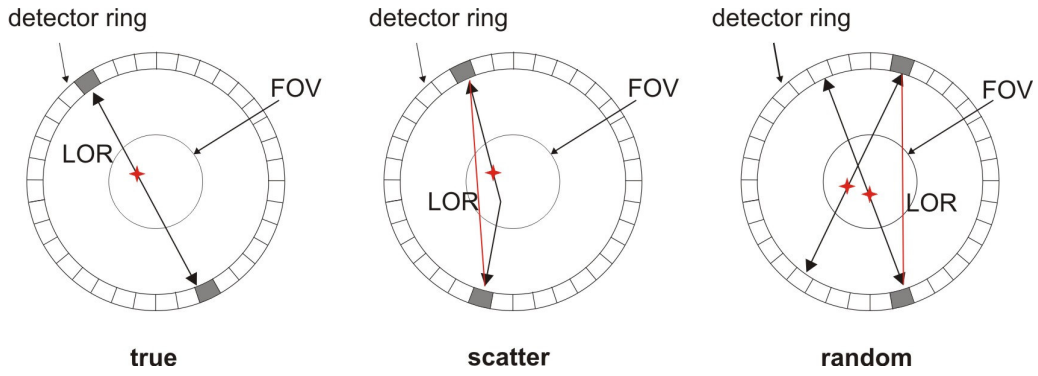


Figure 2.10: Types of Coincidences in PET [2].

2.3.3 PYHSICSICAL PROPERTIES OF PET CAMERA

2.3.3.1 PET Camera Design

PET cameras are made of a set of detectors in a circular orbit. They surround the object to be detected. The detectors individually detect the annihilation photons independent of each other. In each detector, there is Photo Multiplier Tubes (PMT) on which are positioned scintillator blocks [3]. A detector block with 4 PMTs is seen in Figure 2.11.

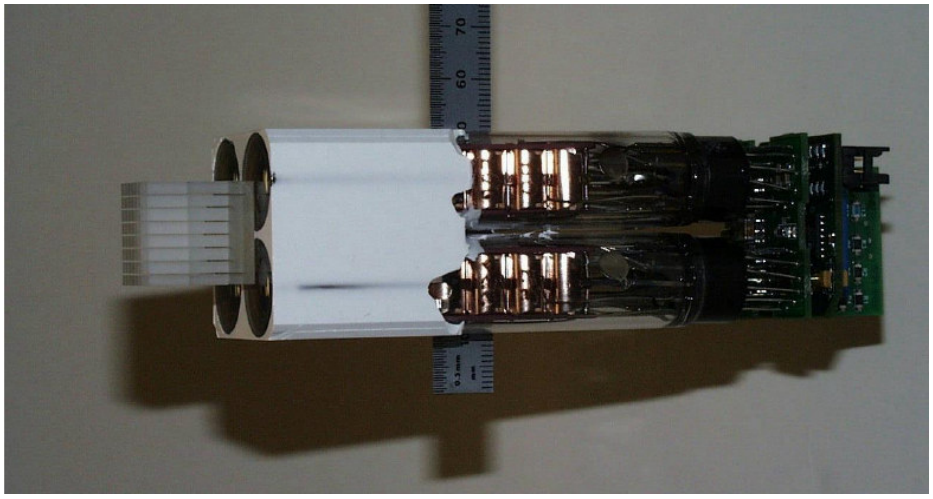


Figure 2.11: A LSO phoswich detector block on a set of 4 PMTs. The scintillators block consists of two LSO layers with different light decay times, which are cut into an 8x8 crystal matrix and glued on a light guide [10].

The primary photons excite with the crystal. These go through the optical coupling and strike the photocathode of the PMT which emits photoelectrons into the vacuum. As seen in Figure 2.12 these photoelectrons are then directed by the focusing electrode voltages towards the electron multiplier where electrons are multiplied by the process of secondary emission. The electron multiplier section consists of electrodes called dynodes. Each dynode is charged with about 100 volts more positive charge than the previous dynode in the chain. As electrons are emitted from a previous dynode they are focused to the next dynode by means of this increasing positive voltage. The multiplied electrons are collected by the anode as an output signal. Because of secondary-emission multiplication, photomultiplier tubes provide extremely high sensitivity. A photon striking the photocathode would usually yield the emission of a single electron but the multiplier can create a final output of one million electrons for each electron emitted [23].

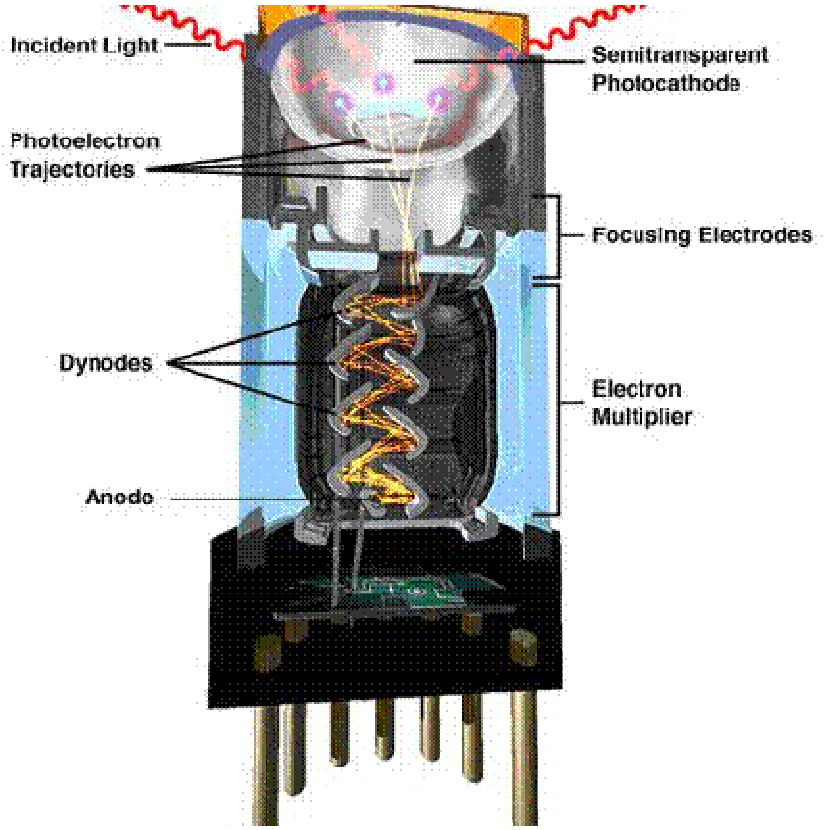


Figure 2.12: Electron amplification in PMT [24].

Cut block detectors were developed in order to reduce the size of detectors. In such detectors, the scintillator block is cut and polished in order to transport the scintillator light to the PMTs. The addressing is done by comparing relative output signals from the PMTs for each event. This design provides uniform sampling and increase intrinsic spatial resolution [3].

2.3.3.2 Scintillator Materials

The properties of the scintillator material have a major effect on the PET camera performance. The scintillator has two basic functions. It interacts with the annihilating photons either by photoelectric absorption or Compton scattering. These interactions cause energy depositions in the scintillator. The scintillator converts this energy into light, which can be detected by the photomultiplier tubes. In a good scintillator the absorption coefficients must be high. It shall have a high atomic number, so it can cause a photoelectric effect. Its decay time must be short which means its spectral response must be rapid. Additionally its

structural stability and hygroscopic sensitivity must be good. Of course it is an advantage if the cost of scintillator is low and if it can be manufactured easily. The crystals must be homogeneous, which means they are made of same material and they have same geometry [3].

Table 2.4: Properties of important PET scintillators [3].

Material	Density (g/cm^3)	Atomic Numbers	Effective Atomic Number	$\mu(cm^{-1})$ @511 keV	Decay Time(ns)	Conversion Efficiency relative to NaI(Tl)
NaI(Tl)	3.67	11,53	51	0.34	230	100%
BGO[1]	7.13	83,32,8	75	0.91	300	15%
GSO(Ce)[2]	6.71	58,64,14,8	59	0.72	60	40%
CsF	4.61	55,9	52	0.42	2.5	6%
BaF_2	4.89	56,9	54	0.44	0.6/620[3]	4/20%
YSO(Ce)[4]	4.54	58,39,14,8	39	-	70	85-118%
LSO(Ce)[5]	7.4	58,71,14,8	66	0.79	40	75%

Now BGO is the most common scintillator in PET cameras. NaI (Tl) is used on, hybrid PET/ SPECT cameras and some PET cameras [3]. LSO (Ce) shows great performance in the new PET scanner generation. It is accepted as a good scintillator by the PET users. Its attenuation is nearly as good as BGO, and has very good scintillator efficiency. It has a very short decay time which improves the count rate efficiency. As seen in Table 2.4, LSO has a density of 7.4 g/cm^3 and its decay time is 40 ns [3]. LuYAP has a density of 7.7g/cm^3 and scintillation decay time of 20 and 160 ns [11].

The crystal thickness is also very important parameter too. It determines the attenuated photon fraction in the detector. This fraction is called the “intrinsic efficiency”. This specialty affects the sensitivity and coincidence count rate performance of the PET directly [3]. Table 2.5 lists the actual photo peak detection efficiencies for different types of detectors.

[1] Bismuth Germanate Oxide ($Bi_4Ge_3O_{12}$)
[2] Gadolinium Oxyorthosilicate (Gd_2SiO_5:Ce)
[3] means two components of light input
[4] Yitrium Oxyorthosilicate (Y_2SiO_5:Ce)
[5] Lutetium Oxyorthosilicate (Lu2SiO_5:Ce)

Table 2.5: Photo peak efficiencies for different scintillators and geometries [3].

Photon Energy	3/8'' NaI(Tl)	5/8'' NaI(Tl)	1'' NaI(Tl)	3.0cm BGO
140 keV	85%	94%	99%	-
511 keV	9-13%	17-21%	36%	93%

2.3.3.3 Attenuation Properties

The interaction behavior of photons with the scintillator is dependent on the density (ρ) and the effective atomic number of the material. They are quantified by the linear attenuation coefficient (μ) which is dependent on the photon energy. The thickness of the detector and its attenuation coefficient effects the attenuation of the incident photon, so the sensitivity of the camera is also determined. The density and atomic number are figured out using the probability of the photoelectric absorption, Compton scattering and pair production. The percentages of the Photoelectric Absorption and Compton Scattering for different crystals are listed in Table 2.6. Photoelectric absorption is preferable in the detector. Compton scattering may lead to misplaced events and blurring effect at the image. If the photon is scattered, the event can be rejected by the energy window and the coincidence effect is completely ignored [3].

Table 2.6: The percentages of the Photoelectric Absorption and Compton Scattering in different crystal materials for 511 keV photons [3].

Crystal Material	Photoelectric Absorption (%)	Compton Scattering (%)
BGO	43	57
NaI(Tl)	18	82
LSO	29	71

2.3.3.4 Decay Time

The time while light is emitted by the scintillator is called decay time. This emission shows an exponential decay curve. The decay time is the time for 67% of the total light to be emitted. If this time is short, then the count rate performance of the camera will be high.

The light output characteristics of PET scintillators show the effects of conversion efficiency and decay time. NaI (Tl) has the greatest total light output compared to other scintillators. LSO has a faster output but lower total light output. The BGO has a low total output and long decay time (Table 2.4).

Linearity and the spectral distribution of the light output are very important. Absorbed radiation energy and the amount of light output from the scintillators must be linear. The spectral distribution refers to the color of the scintillation light given off by the scintillator. A poor match between the scintillator light output spectrum and the PMT detection response causes a smaller signal output from the PMT [3].

2.3.4 DATA PROCESSING IN PET

2.3.4.1 Detector Electronics and Sinograms

An electronic circuitry is responsible for reading out and analyzing detector signal. Preamplifier receives the PMT output signal. The preamplifier outputs go to a threshold (lower level discriminated) which eliminates small signals from electronic noise. Then these signals go to the coincidence testing circuit [3] (Figure 2.13).

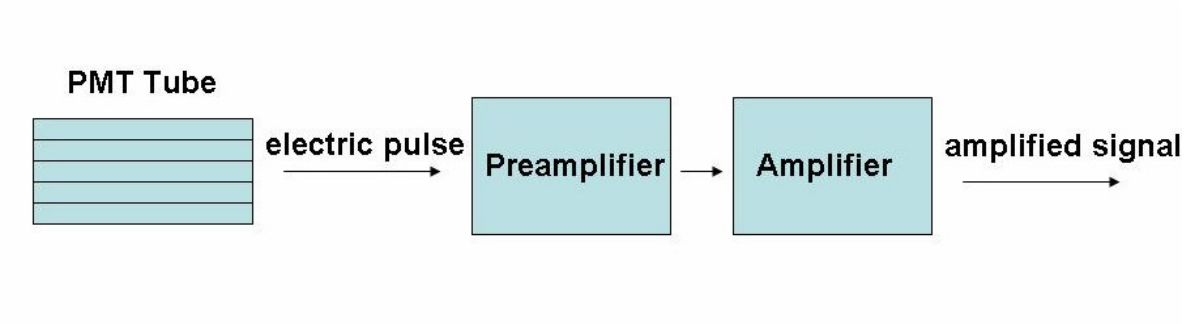


Figure 2.13: The path of annihilation photons becoming signals .

When the signal from one detector enters the coincidence circuit, a timing window is opened. If the second signal comes within a defined time (about 12 to 15 ns) these events are called as coincidence events. After the coincidence is determined, the energy of each photon is tested by a single channel analyzer (SCA). Then the processing continues with LOR determination and sorting.

The LOR is calculated from the signal addresses and their corresponding geometric locations. In the camera, the LOR is the line that connects the position of the two detected photons. The coincidence detection of two photons form a coincidence hit pattern in the detector, from which the LOR of the event is calculated. The detected LORs are then acquired in a matrix by their angles (ɵ) and radial position (r). The term sinogram is used because each point on the image traces out the form of a sine wave of the sinogram [3]. An example forming an image is given in Figure 2.14.

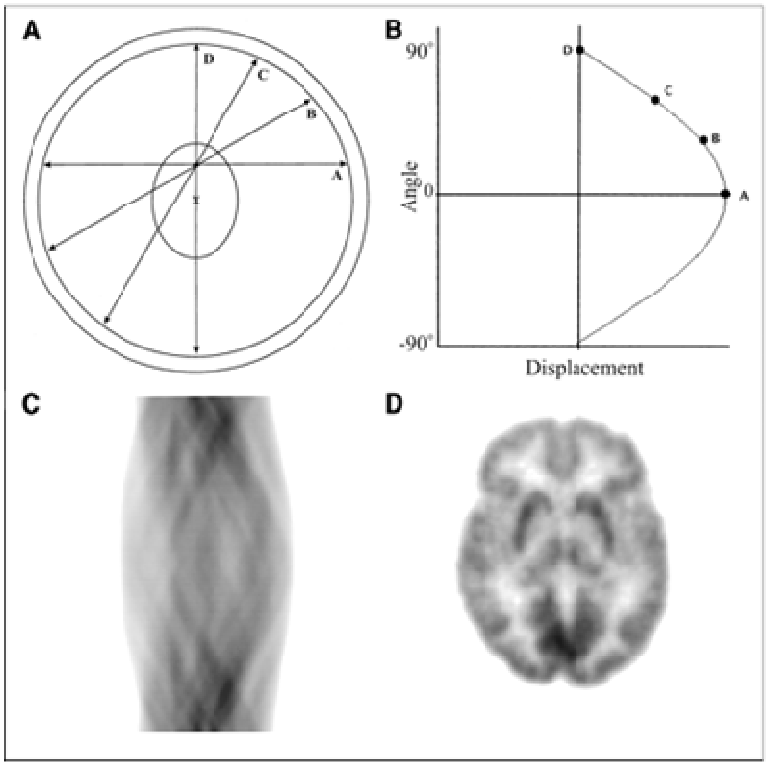


Figure 2.14: Coincidence events are categorized by plotting each LOR as function of its angular orientation versus its displacement from centre of gantry. Four LORs passing through locus of interest are labeled A, B, C, and D. These 4 LORs are plotted on this sinogram. Reconstructed brain image corresponding to sinogram in (C) is shown [12].

.

2.3.4.2 Reconstruction and Back projection

PET images are taken in the form of sinograms or projection matrixes. Tomographic reconstruction is done via the "inverse radon transposition" mathematical method. In order to construct a set of slices from these images, filtered back projection (FBP) or iterative reconstruction (IR) method is used.

The data from each projection is streaked back over the image matrix at the acquisition angle. The final back projection is the sum of the back projections from each angle. During the back projection extra counts are left on the image, which result in artifacts, looking like a star. If numbers of back projected frames are increased, this star artifact turns into a blurry point.

Simple back projection causes a very blurry image which must be corrected. Filtering is required to remove this blurring which leads to "Filtered Back projection". "Pre Filtering" before back projection and "Post filtering" after back projection may be done. Smoothing filter may be applied before or after reconstruction to reduce the image noise.

Filtering is a mathematical method for manipulation and modification of the image. In PET, the image data is converted into a function of spectral frequency by the "Fourier transformation". The low frequencies in the image change the background. The high frequency components are the sharp image details. The image noise is at all frequencies, but mostly at higher frequencies.

There are different kinds of filters. The "Hann" or "Gaussian" filter can be chosen to determine the shape. The shape and sharpness of the filter can be controlled with "order" and "roll-off" properties of the "Butterworth" filter. Most of the filters don't change the image significantly, but very sharp windows may cause "rippling" artifacts.

Another method of reconstructing an image from the acquired sinograms is "Iterative Reconstruction". It is more accurate than filtered back projection, because the reconstructed images are less noisy in this technique. However it needs large memory and long processing time. Most popular modern methods are

- ML-EM (maximum likelihood- expectation maximization)
- OSEM (ordered subset-expectation maximization) [3].

2.3.5 IMAGE CORRECTIONS

2.3.5.1 Random corrections

Random coincidences reduce image contrast and cause errors in quantification. Modern PET cameras have sophisticated electronics for correction of the random coincidences. They are corrected by subtracting the measured or estimated random rate from the emission and transmission data for each detector pair.

They are corrected in two ways

- The delayed coincidence window method,
- The singles approximation method.

In first method; the delayed coincidence window measures the count rate which is delayed in time. So it is possible to differentiate the true annihilation and delayed annihilation. All random coincidences are measured in a second time window.

In the second method, the random rate for each detector pair is estimated with the measured singles rate (Equation 2.11). The random coincidence rate is related to the singles rate of each detector.

$$R_r = 2\tau R_1 R_2 \qquad \textbf{2.11}$$

Where R_r is the accidental (random) coincidence count rate, R_1 is the single rate for detector 1, R_2 is single rate for detector 2 and 2τ is width of the coincidence window [3].

2.3.5.2 Attenuation Correction

The attenuation of photons causes losses in counts, image artifacts and quantitative inaccuracies. Attenuation correction improves contrast and reduces image artifacts. Attenuation is a result of the interaction of one or both of the annihilation photons in the radioactive source (or patient). Photon absorption affects the attenuation.

The emission scan is the main part of the acquisition. The data from the emission scan is used to reconstruct the radiotracer distribution in the source (phantom or patient). The

transmission scan is performed in order to measure and correct the non-uniform attenuation of photons which come from the radioactive source.

Transmission scanning is made with a long lived positron radionuclide, usually ^{68}Ge, which has half life of 271 days. The annihilation photons from the decay in the source pass through the patient and are detected in coincidence. Since the energy of annihilation photons from transmission source is the same as that from the emission source, transmission scan has to be performed before the injection of the radionuclide. This technique allows distinguishing the transmitted photons from the emission photons. However the position of the patient must be same as before the injection. Sometimes significant problems occur because of repositioning.

Transmission scanning needs long scanning time, 10 to 20 minutes. The time depends on activity of the transmission source, the size of patient and the count rate capabilities of the detectors. To reduce this scanning time the transmission source activity might be increased, but the high count rates cause high dead time losses.

An alternative method of transmission scanning has recently been developed which uses single photons, such as ^{137}Cs with 662 keV of energy and 26.6 years half life. Using separate acquisition window allows for correcting post injection transmission scans. Singles transmission scans require less time because of higher count rates. In this technique only the detectors of the opposite side of the patient detects high activity [3] (Figure 2.15)

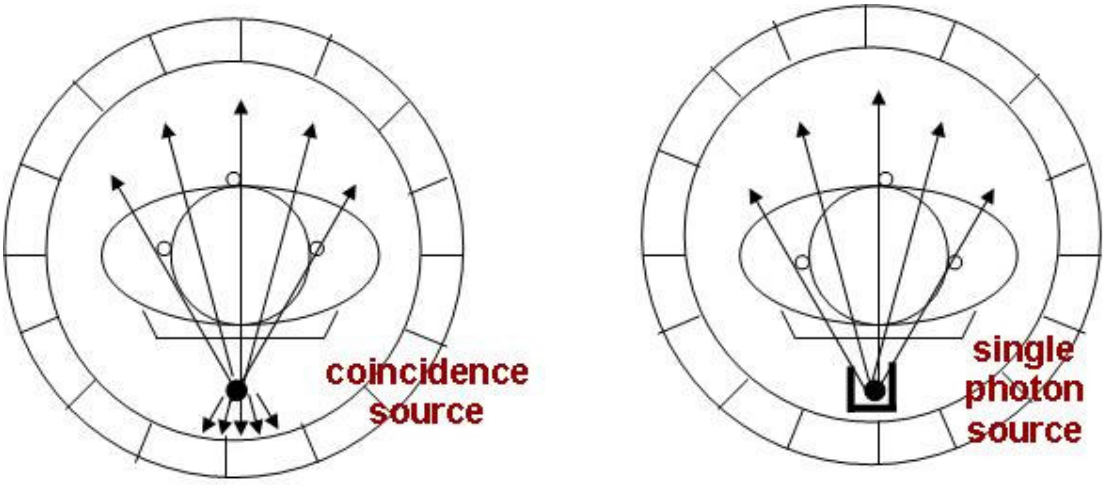


Figure 2.15: Transmission scanning with coincidence sources and single photon sources are compared. In the coincidence counting system, the count rates experienced by detectors close to source are very high, requiring a lower activity source. In singles transmission scanning only the detectors on the opposite side of the patient count hence, allow higher activity sources to be used [3].

2.3.5.3 Scatter Correction

A photon which scatters due to impact looses some energy. These photons appear as background noise in the image. This noise reduces the contrast and has to be subtracted from the image. Scatter errors form a smooth parabolic shaped density curve to estimate and subtract from the total data. This “contrast fixing” results in some accuracy loss, since each imaging has different scatter amount. This subtraction can also be done with subtracting scatter image from original using pixel by pixel subtraction algorithm. Another method for scatter correction is known as “de-convolution” of the image to counter the blurring [3].

2.3.6 CAMERA PERFORMANCE CHARACTERISTICS

2.3.6.1 Spatial Resolution

Spatial resolution is the ability to form separable images of closely positioned object of small size. It depends on the detector size, addressing logic and physical effects of non-colinearity and positron range. Spatial resolution depends firstly on the detector size. The full width half maximum (FWHM) of the point spread function is about 0.4 to 0.5 of the width of detector element. For example, a camera with detectors of 10mm width has a basic resolution of 5mm. Physical effects of non-colinearity and positron ranges are the natural physical limitations of spatial resolution [3].

2.3.6.2 Sensitivity

The sensitivity of a PET camera is the count rate the camera measures for a given activity in the field of view. The units of sensitivity are cps/μCi/ml. Sensitivity is related to injected dose, the imaging time and the acquired counts. It affects the statistical accuracy of the data and the image noise. The sensitivity of a PET camera depends on the individual detector efficiencies, total geometric efficiency, energy windows, acquisition mode, and dead time losses.

The efficiency of the individual detectors for counting single photons depends on the scintillator material, the scintillator thickness and the energy of the gamma ray. However, these efficiencies are especially reduced using coincidence counting because the two detectors have to count a photon at the same time. The geometric efficiency of the whole camera is determined by the physical design of the camera such as the distance between detectors. The energy windows and acquisition mode affects the sensitivity, too. Wider energy windows increase the sensitivity, but with an increase of random coincidence, too.

Dead time reduces the sensitivity at higher count rates. The sensitivity of PET cameras is 30 to 100 times better than the gamma cameras. It is mainly because the gamma camera's collimator greatly hinders the photons because of the photons, which come from different angles. PET camera detectors have higher intrinsic efficiency [3].

2.3.6.3 Count Rate Characteristics

The count rate characteristic of a camera is the capability to process detected events in a given time. It includes the terms of resolving time, which processes the data in a given time and dead time, which is the fraction of time in which other events are not allowed to be detected while PET calculates the prior event. The count rate depends on the sensitivity (Figure 2.16). In an ideal camera, the count rate is proportional to the activity. The count rate reaches a maximum, and then decreases as the activity increases. The dead time causes the count losses, too [3].

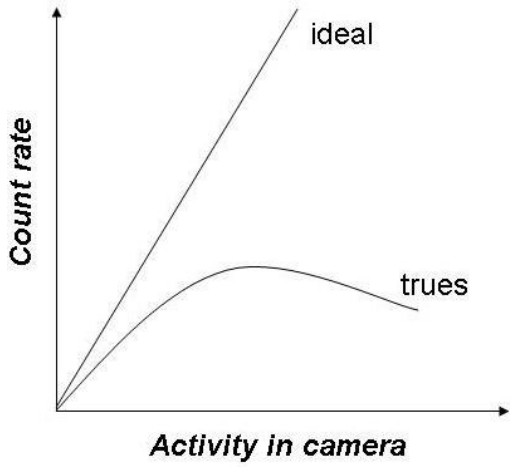


Figure 2.16 : Count rate characteristics of a PET camera. The true count rate deviates from the ideal because of dead time losses.

2.3.6.4 Energy Resolution

Energy resolution is the ability of separating different energies. It is specified as the FWHM of the pulse height. The energy resolution depends on the amount of scintillator light detected by the PMTs, the scintillator material and the amount of signal integration done by the electronics [3].

2.3.7 PET QUALITY CONTROL

Routine quality control is required in PET cameras to check and ensure that the system runs without problem. It must include the following procedures;

- System calibration
- Performance
- Normalization scans
- Blank scan and daily check [3].

2.3.7.1 System Calibration

The output signals are very sensitive to changes in the PMT and pre-amplifier gains. These gains may vary with external temperature, count rate, gamma ray energy, changes in high voltage supply stray local magnetic fields, PMT age and camera motion. Gain changes affect the detector sensitivity and may cause addressing errors in certain detector design.

The "tuning" state of the camera describes the relative sensitivity of the detector. It is affected by the variation of the individual PMT's gain and these sensitivity fluctuations result in a non-uniform image. The tuning must be done periodically with resetting gains of PMT and the preamplifier. This re-tuning is called "system calibration". While doing this calibration, a specific source is placed in the FOV. Sometimes ^{68}Ge filled uniform activity is used as a source. During the calibration, the PMT high voltages, preamplifier gains and photo peak windows are adjusted. After this process, the block detector addressing map is recalibrated. This adjustment may be done weekly or monthly according to the requirement of the system. If the daily QC test is not good enough, then system calibration is required [3].

2.3.7.2 Performance

The performance of a PET camera must be evaluated. Spatial resolution, sensitivity, count rate characteristics and energy resolution which are explained in section 2.3.6 affect the performance.

2.3.7.3 Normalization Scan

Normalization scan is done to measure the sensitivity variations between PMT pairs in the scanner. The normalization table is used to correct non-uniformities in the scan. High count rate scan is done for the normalization. Most of the cameras use the same source, which is used for the transmission scan. The data from normalization is used to correct all scan data (blank, transmission and emission) for the variations in detector efficiencies. These data are also analyzed to detect PMT and electronics failure [3].

2.3.7.4 Blank Scan and Daily Check

The daily quality control testing of camera consists of acquisition of 'blank' (without object in the port) scan. First, the blank scan is used in combination with transmission scan to determine the attenuation correction factors. Then data from blank scan are used to compare sensitivities of the detectors. Diagonal streaks in the sinogram show that there is a problem with detectors. They may be fairly tuned, weak or failing. Other larger streak patterns or areas of missing counts are sign of electronics problems. The daily check should be performed at the beginning of the day before doing any scan [3].

2.4 CLEARPET

2.4.1 SMALL ANIMAL PETS

The evolution of PET imaging for small animals has led to the development of dedicated PET scanner designs with high resolution and sensitivity for imaging small animals such as mice and rats [13]. Small animal models are developed to study the mechanism of human diseases [14].

Technological developments in the area of scintillators, semiconductor detectors and electronics lead to new projects [15]. There are several small animal PET scanner models. In Table 2.7 we can see their properties; such as detector type, dimension, number of modules, volumetric spatial resolution and absolute sensitivity.

Table 2.7: Crystal based animal PET-Scanners [14].

Animal PET Scanner	Detector Type	Detector Dimension	Number of Modules	Volumetric Spatial Resolution(mm^3)	Absolute sensitivity(cps/kBq)
MicroPET4 Model P4	LSO	8X8 (2.1x2.1x10mm^3)	42	6.4*	14.3
MicroPET4 Model R4	LSO	8X8 (2.1x2.1x10mm^3)	24	5.1*	24.5
MADPET	LSO	8x2 (3.7x3.7x12mm^3)	6	17	0.35
Hammersmith rat PET	BGO	8x7 (30x6x3 mm^3)	16	23	43
YAP-PET	YAP:Ce	(2x2x30 mm3)	4	5.8*	17.3
SHR-7700	BGO	8x4 (2.8x6.9x30mm^3)	4x60	22	23
ANIPET	BGO	(36x36x20 mm^3)		35.8	8
Clear PET First Prototype	LSO and Lu-YAP	48 (2x2x10 mm^3)	4x16	3.4	17

* in the centre of FOV

2.4.2 PROPERTIES OF ClearPET™

2.4.2.1 Foundations of ClearPET™

One of the important studies in the development of a Small Animal PET scanner of the second generation is made by the Crystal Clear Collaboration (CCC) and used at the Research Centre Juelich (Forschungszentrum Jülich) [16]. Crystal Clear, an international collaboration of research institutes, to develop new generations of scanners for PET, was setup in 1990. The members are CERN, Forschungszentrum Jülich, the Institute of Nuclear Problems in Minsk, and the Institute for Physical Research in Ashtarak, the Laboratório de Instrumentação e Física Experimental de Partículas (LIP) in Lisbon, Sungkyunkwan University School of Medicine in Seoul, and the Université Claude Bernard in Lyon, the Université de Lausanne and the Vrije Universiteit Brussel (VUB) [17]**.** After doing some research over the gamma ray detection it has developed a high resolution small animal PET scanner named ClearPET™ [15].

Every member in Crystal Clear Collaboration has another task. For example CERN is working on crystals and Forschungszentrum Jülich on electronics. They are several scanners whose names begin with ClearPET but they have another extension, which are same in concept with some variations. For example ClearPET Neuro in Forschungszentrum Jülich can tilt up to 90° toward the front (Figure 2.20). I will describe of ClearPET Neuro in the coming sections (Figure 2.17).

Figure 2.17: Front view of the ClearPET Neuro

2.4.2.2 Physical Specifications of ClearPET Neuro

The ClearPET System is a high performance PET scanner with high resolution and high sensitivity by using new technologies in crystal and electronics [18]. ClearPET systems were developed to achieve a high spatial resolution less than 2mm. New generation small PET scanners have improved spatial resolution about 1mm [11].

The detector technology in ClearPET is based on two different types of lutetium scintillators. Each element consists of two layers of crystals with different decay times. One layer is formed from cerium-doped lutetium oxyorthosilicate (LSO) scintillator material; the other contains cerium-doped lutetium yttrium aluminate perovskite (LuYAP) scintillators [17]. Each crystal has a volume of 2x2x10 mm^3 LSO and 2x2x10 mm^3 Lu-YAP crystals which are coupled to multi-channel photomultiplier tubes (Hamamatsu R7600) as seen in Figure 2.18. Total crystal thickness is 20mm and its matrix is 8x8 [18].

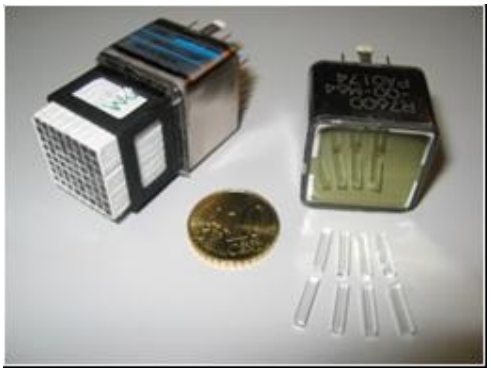


Figure 2.18: Multi-channel photo multiplier tube (7600M64, Hamamatsu)

ClearPET has in total 80 detector modules distributed among 20 detector cassettes which are arranged in a ring configuration. There is a 10mm gap in between each detector cassette. To compensate for this axial gap there is shift of 7.1mm (¼ of PMT) between every other cassette [19] (Figure 2.19).

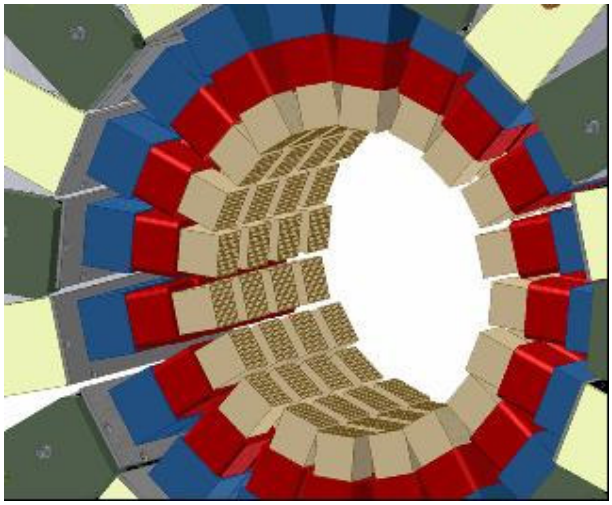


Figure 2.19: 20 PMTs per ring and every other cassette is shifted by ¼ PMT lengths center to center [20].

The gantry allows rotation over 360 degrees. It offers the possibility to measure larger animals due to its adjustable detector diameter. FOV is adjustable from 130 to 300cm. Axial depth is 110mm [18]. Gantry can tilt up to 90° toward the front to measure non-human primates in upright sitting position (Figure 2.20).

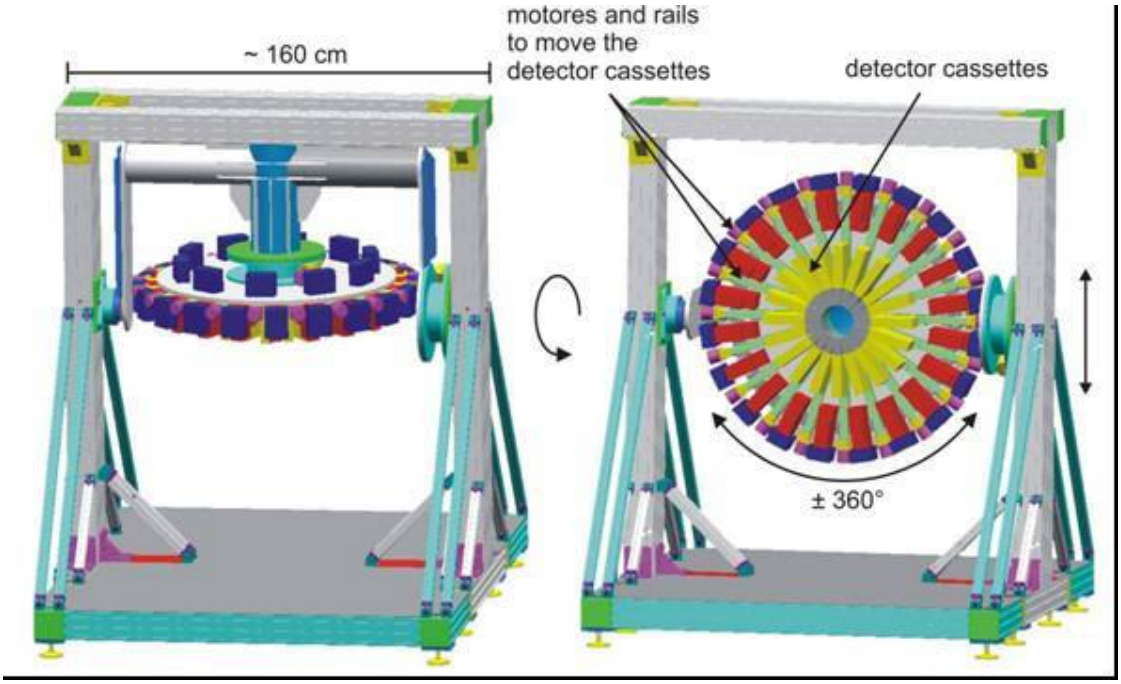


Figure 2.20: Construction chart of primate version of PMT-ClearPET prototype. At the right side the idle position of the scanner is seen. At the left sight we see how the gantry can tilt up upwards to forward.

The PMT contains 8x8 crystals. During acquisition several counts are stored. As it is seen in Figure 2.21 counts of singles belong to main crystal (m). Counts of neighbors (n) specify activated neighbor crystal of the main crystal. Counts of far crystals (f) come from the crystals which are not neighbors, but are affected from the main crystal. In addition multiple rays can be given as an example for the counts in which their energies are out of range.

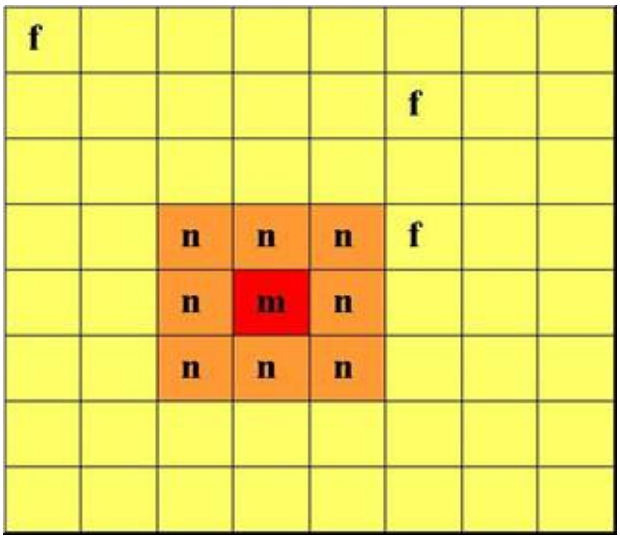


Figure 2.21: An example for main crystal (m) and affected crystals which are in neighborhood (n) and far (f) positions.

Each cassette consists of four modules. 20th cassette matches with module group zero and first cassette matches with 19th module group. Cassettes' positions are seen in Figure 4.6. There are 18° degrees between each cassette. Cassette 1 exists at 0°, cassette 2 exists at 18°… and cassette 20 exists at 342° (Figure 2.22: a). If gantry rotates 180° cassette 11 comes to 0° (Figure 2.22: b)

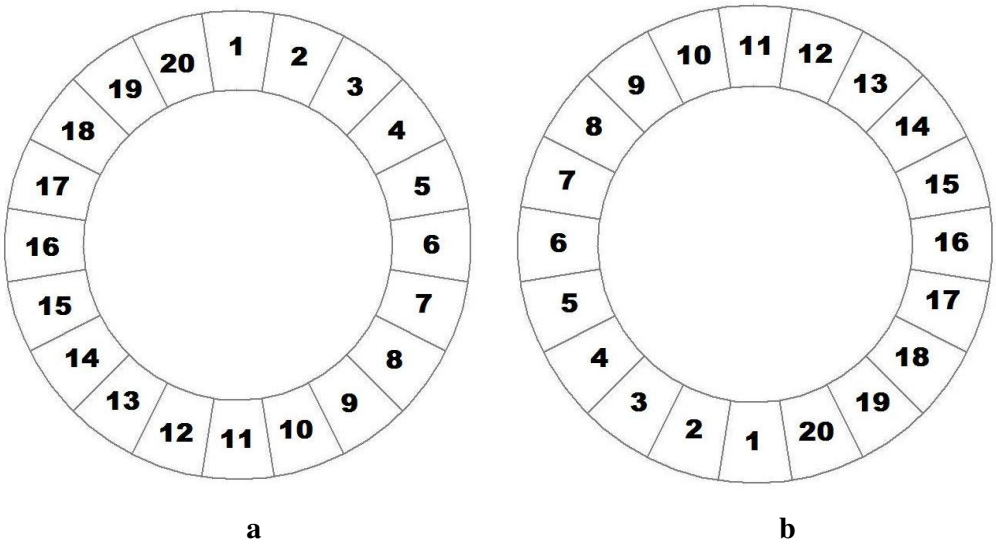


Figure 2.22: a) Cassette positions when the gantry doesn't rotate. b) Cassette positions when the gantry rotates 180°.

As a reflector material ~0.3 mm Tyvek paper is used. The PMTs have a sensitive area of 18.1x18.1mm^2. According to the previous experiences, simulating a point source centered in the FOV (threshold: 350keV) resulted in sensitivities of 4.2% for a 4x20 PMT [19]. Average intrinsic spatial resolution (FWHM) is 1.48mm. Coincidence timing resolution is 2.0 ns [18].

2.4.2.3 Data Acquisition

Scintillators in a cassette are connected to four PMTs. Signals which come from these PMTs are sent to four corresponding decoder blocks, which filter this data and give an output. The output includes trigger, digital and analog signal, which goes later to ADC. Decoder blocks go to a programmable circuit called FPGA, which register the signals. The output from FPGA contains time, energy and position data, and is sent to preprocessing PCs from an optical connection.

There are five preprocessing PCs, one for every four cassettes (Figure 2.23). Preprocessing PCs do mathematical data processing. Calculations for time mark refinement, pulse shape discrimination and position encoding are done here. Pre-processing PCs are networked. They send all their data to a single Master PC which is used to view device output and shows pre-processed data in graphical or simple text form. The software of master PC also lets the user control the PMT, the pre-processor PCs and sends the data to storage [21].

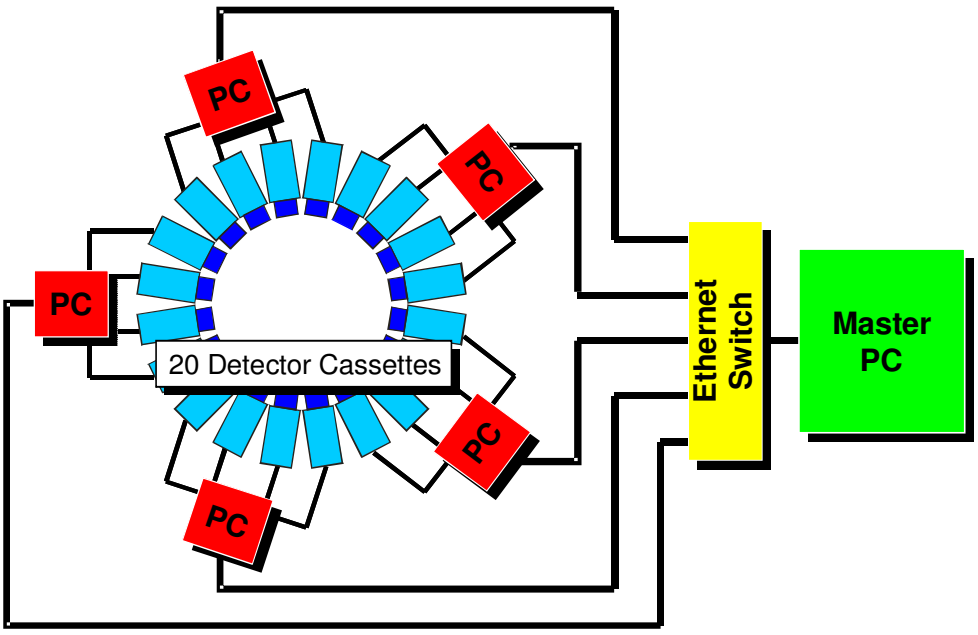


Figure 2.23: Data acquisition architecture [20].

2.4.2.4 Data Format

The data format which stores all available LOR coordinates and detection times is called “list mode”. Each imaging session can be stored in this form and will take hundreds of megabytes of storage space. List Mode Format (LMF) gives information about energy, time and gantry position. In LMF, there is one header file which is plain text and stores information about the measurement and geometry. After the LMF header, device output for events is stored in binary files. A LMF binary file starts with some information called LMF encoding header, followed by the LMF event record in the same file. LMF encoding header describes scanner topology and LMF event record tells us which event type was detected (single hit or coincidence hit), it gives the event time (absolute for single, relative for coincidence), gantry position, crystal position and energy [21].

During acquisition data are stored as .buf file and .ang file. Buf files are received from the measurement directly. There is one .buf file per cassette. Buf Files are binary event data which include all singles with time, event position and energy information. They contain 64 bit words and consists of 38 bit time data, 8 bit energy data in units of 5 keV, 6 bit pixel number data, 4 bit neighbor crystal data, 1 bit crystal layer data, which shows the type of crystal, 2 bit detector ring data and 5 bit cassette data.

Ang files are in ASCII format and inform about the gantry position of cassettes and count rate of cassettes that they gather within a certain time interval. Data in .ang files are stored in 21 line groups which are called intervals.

In the first line, **first value** is an integer which shows the interval number (Figure 2.24 and Figure 2.25). **Second integer value** shows the current gantry angle and there is no rotation if all intervals have the same value. This recorded value is given in units of 1/10 degree (Figure 2.24). For example in Figure 2.24 if the first and the second intervals gantry angle was zero the gantry wouldn't rotate. However, here we have one degree rotation. Gantry can rotate at maximum (360 ±5) °. After the gantry completes one rotation (0° to 360°),it rotates backwards and takes measurements from that side (360° to 0°). **The third integer value** in this line shows the bed position. The bed can not be moved during the measurements, but its data may become significant in future. **The fourth value**, which is a floating point, displays the starting time of the interval and is measured in seconds. **Fifth value**, which is floating point type, shows the duration time of measurement in this interval (Figure 2.25). The acquired data consist of several interval groups. These intervals are determined according to total acquision time and given time duration. For example in Figure 2.24, time duration is assigned as one second. But the second data acquision begins with 0.125 s delay and therefore second interval counts begin with 1.125^{th} s.

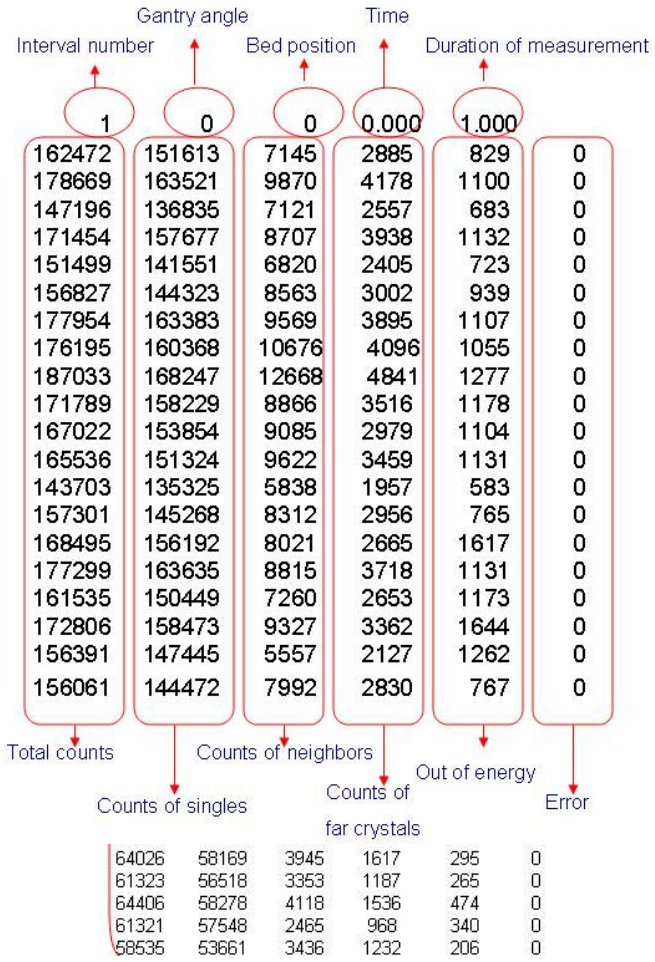
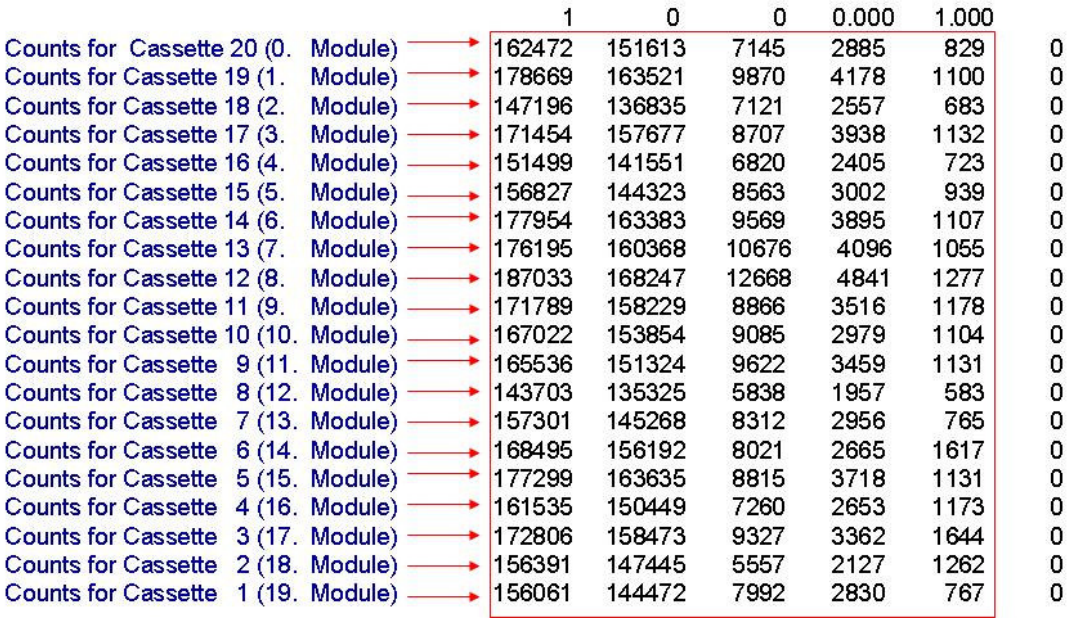


Figure 2.24: The acquired data are stored as several intervals. The first value in the first line of each interval shows measurement number. Second values show rotation degree. If there is no rotation this value takes zero in each interval.

The following lines display the count rate values of the 20 detector cassettes. There are integers in each column which show the counts of certain events. **In the second column**, the counts of singles are shown. **In the third column** the count of the responding neighbors are shown. **In the fourth column**, far crystals' counts are stored. In **the fifth column**, counts with an energy level which is out of a certain range are stored. The sixth column is an error column. **The first column** shows the sum of all these counts (Figure 2.25).

Figure 2.25: Explanation of the intervals' components.

20 cassettes present in the detector ring. The detector counts in each interval begin from 20^{th} detector and continue up to first detector respectively (Figure 2.26).



	1	0	0	0.000	1.000	
Counts for Cassette 20 (0. Module)	162472	151613	7145	2885	829	0
Counts for Cassette 19 (1. Module)	178669	163521	9870	4178	1100	0
Counts for Cassette 18 (2. Module)	147196	136835	7121	2557	683	0
Counts for Cassette 17 (3. Module)	171454	157677	8707	3938	1132	0
Counts for Cassette 16 (4. Module)	151499	141551	6820	2405	723	0
Counts for Cassette 15 (5. Module)	156827	144323	8563	3002	939	0
Counts for Cassette 14 (6. Module)	177954	163383	9569	3895	1107	0
Counts for Cassette 13 (7. Module)	176195	160368	10676	4096	1055	0
Counts for Cassette 12 (8. Module)	187033	168247	12668	4841	1277	0
Counts for Cassette 11 (9. Module)	171789	158229	8866	3516	1178	0
Counts for Cassette 10 (10. Module)	167022	153854	9085	2979	1104	0
Counts for Cassette 9 (11. Module)	165536	151324	9622	3459	1131	0
Counts for Cassette 8 (12. Module)	143703	135325	5838	1957	583	0
Counts for Cassette 7 (13. Module)	157301	145268	8312	2956	765	0
Counts for Cassette 6 (14. Module)	168495	156192	8021	2665	1617	0
Counts for Cassette 5 (15. Module)	177299	163635	8815	3718	1131	0
Counts for Cassette 4 (16. Module)	161535	150449	7260	2653	1173	0
Counts for Cassette 3 (17. Module)	172806	158473	9327	3362	1644	0
Counts for Cassette 2 (18. Module)	156391	147445	5557	2127	1262	0
Counts for Cassette 1 (19. Module)	156061	144472	7992	2830	767	0

Figure 2.26: Counts for cassettes (module groups). Cassette numbers begin from 20 and module group numbers begin from 0.

3 MATERIALS AND METHODS

A program based on IDL 6.1 (Interactive Data Language) has written to evaluate the information which comes from each module group and are stored in .ang files during the detection. IDL is software for data analysis and data visualization which provides 2D and 3D graphing.

As it described before, the ClearPET data are stored in ASCII file. An example is given in Appendix A. These data belong to blank measurements. Data consist of 10 intervals. Also 10 different counts data are taken from cassettes. Time duration of each measurement was 1 s. There is no rotation, because all angle values are zero.

Data can be stored in more than one file with ang extension. For example 360 intervals can be kept as 10 files in which 36 intervals exist in each file that should be read in order.

In the program named Module_check (in Appendix B), the user shall input the address of the data. After that, the wanted graphs can be selected. The details of the program can be found in the following section.

3.1 PROGRAM EXPLANATION

In this section all program is explained one by one. The fractions of the program are pointed out with green color and the outputs with red color.

```
pro Module_check

;#####################################################
;Author: PINAR CELIK
;   RESEARCH CENTER JUELICH
;   GUEST STUDENT FROM DOKUZ EYLUL UNIVERSITY
;
;E-mail: pcelik@gmail.com
;
;#####################################################
```

First we input the working directory from the user. FILE_SEARCH function reads the folder for specified file extension and returns the file names. ‘file_names’ are used as an array of strings which store the file names which end with .ang inside the specified folder.

```
folder= ''
print, 'Enter data folder: '
```

```
read, folder

file_names= FILE_SEARCH(folder, '*.ang', COUNT= NUMFILES)
```

Number of intervals is asked from the user which is used later inside a loop to read all the interval data from the ang file.

```
print,'Enter the number of intervals of data to read from
file:'
NUMINTERVALS= 36
read, NUMINTERVALS

NUMCASSETTES=20 ; WE ALWAYS HAVE 20 CASSETTES
```

Now we create the data structures which will store the headers for each interval in the ang files and the cassette data under each header. One structure called cassette, the other called header are created with to store all the data of intervals.

Header structure: This structure has components to store all header information. Its field names are num, startangle, bedpos, time, intervaldur which help us reading data which are stored in this format:

1 0 0 0.000 1.000

Cassette structure: This structure has fields named num, angle and value which are arrays of long data type for storing big numbers. It helps us reading data which are stored in this format:

162472 151613 7145 2885 829 0

Twenty of these data are repeated for each interval. Also the number of the cassette and its current position on the gantry are stored inside this structure.

```
measurement= {cassette, num:0, angle:0, value:fltarr(6)}
definitions={header,num:0,startangle:0,bedpos:0,time:0.0,inter
valdur:0.0}
```

"data" is a 3d array of cassette structure. After filling it, we can use it as in the following example: If we want to get the angle of cassette 3 in interval 5 of file 6, we can write:

myangle= data[3][5][6].angle

"def" is a 2d array of header structure. After filling it, we can use it like that: If we want to get the time of the header which belongs to interval 3 in file 5, then we write:

mytime= def[3][5].time

```
data= replicate({cassette}, NUMINTERVALS, NUMCASSETTES,
NUMFILES)
def=replicate({header},NUMINTERVALS, NUMFILES)
```

Then we initialize some counting variables for our loops which will be used when reading the data from files into instances of our struct using loops.

We needed these temps because we could not read into array elements directly. So we read into temps and copied them to the array elements later.

```
row= 0          ; row iterator
interval= 0     ; interval iterator
measurement= 0
temp1= 0L       ; header temps
temp2= 0L
temp3= 0L
temp4= 0L
temp5= 0L
temp6= 0L
tempf1=0.0
tempf2=0.0
```

Here is a 2 D array which stores the sum of each module group in each interval. When modulsum is filled, it can be used to retrieve the sums of all module groups inside required intervals. For example, to get the sum of all module groups inside interval 6 of file 2, we do:
mysum= modulsum(6, 2)

```
modulsum=lonarr(NUMINTERVALS, NUMFILES)

;################################
; READING FILES, FILLING ARRAYS
; fill header arrays, fill row and column arrays for each
interval
;################################
```

When the user enters the number of groups, and the entered number is larger than the groups which exist inside the file, we need a mechanism which catches the error which occurs when trying to read from a finished file.

```
on_ioerror, cant_read_anymore
```

We need 3 loops, cassette reading loop nested inside interval reading loop, which is nested inside file reading loop. This way, we can read each cassette data inside each interval inside each file. For example:

file number: 1

interval number: 1

cassette number: 1

cassette number: 2

.

.

cassette number: 20

interval number: 2

cassette number: 1

cassette number: 2

.

.

cassette number: 20

.

.

and so on, then file 2, and so on, until the max number of files inside the given directory.

```
for eachfile= 0, NUMFILES-1 do begin
```

Straightforward, open the file and assign a resource number to the file.

```
openr,file, file_names[eachfile], /get_lun
```

```
for interval= 0,NUMINTERVALS-1 do begin
```

Now “file” can be used for the reading function called readf.

We read each header data each time we loop over an interval. Like it was said before, we first assign the data into temporary values, then our header struct array “def”’s values. We also divide the angle data into 10 because the data given to us is 10 times bigger than the real data.

```
readf, file,temp1,temp2, temp3, tempf1, tempf2
def[interval,eachfile].num=temp1
def[interval,eachfile].startangle=temp2/10
```

```
    def[interval,eachfile].bedpos=temp3
    def[interval,eachfile].time=tempf1
    def[interval,eachfile].intervaldur=tempf2

    for row= 0,NUMCASSETTES-1 do begin
```

Here, like with the headers, we read each cassette value for the current interval being looped over. We fill them into our 3 D structure array called data.

```
        readf, file,temp1,temp2, temp3, temp4, temp5, temp6
        data[interval,row,eachfile].value[0]= temp1
        data[interval,row,eachfile].value[1]= temp2
        data[interval,row,eachfile].value[2]= temp3
        data[interval,row,eachfile].value[3]= temp4
        data[interval,row,eachfile].value[4]= temp5
        data[interval,row,eachfile].value[5]= temp6
```

Here we calculate the current cassette number for our cassette. Cassette numbers are read in reverse.

```
        data[interval,row,eachfile].num=NUMCASSETTES-row

data[interval,row,eachfile].angle=def[interval,eachfile].start
angle
```

Like mentioned above, here, we control the possibility if the calculated cassette angle (using the above formula) becomes negative.

```
        if data[interval,row,eachfile].angle LT 0 then begin

data[interval,row,eachfile].angle=def[0,eachfile].startangle+d
ata[interval,row,eachfile].angle
        endif
```

Now we have all the cassette data for this interval. Here we fill our modulsum array declared above. Since we know each cassette data, we can find the sum of all “total counts per module group”. Total count data is stored in the first element of our value array in the cassette structure.

```
       ;sum of all cassette counts for each interval, store
for later
        modulsum(interval, eachfile)=modulsum(interval,
eachfile)+data[interval,row,eachfile].value[0]
```

```
    endfor ; row
endfor ; interval
```

After reading each file, we free the resource and close the file so that our resources (IDL calls them “lun”s) do not finish when reading.

```
free_lun,file,/FORCE
close,file ; close and free file slot each time we open a new
file
```

Now at this point, if there was no io error, the program will continue the file loop with the next file. However if there was an error reading from the file, (such an error happens when there is nothing more to read) then the program will jump from the failed read attempt into the label called: cant_read_anymore.

```
continue
```

Here is where the program jumps from the failed read attempt. This place is for telling the user which file and which interval and which cassette our reading attempt failed. We also close the file which had this error, since we can’t read from it anymore.

```
cant_read_anymore:
print, 'Read error in file ', file_names[eachfile], 'data
interval ', interval, ' after cassette ', row, '!'
print, 'Error: Unexpected number of intervals in file.'

endfor ; file
```

Sometimes ang data has data about a rotating gantry. To find out if our gantry is rotating, we simply compare the angles of the first two header data in def. We create a variable called “rotating” which shows if the measurement is done with rotating gantry. First we set rotating to 1, which means it is rotating. If first two headers have the same starting angle, then we make rotating 0, which means there is no rotation.

```
;If header angle changes, rotating
rotating=make_array(10, value= 1)
for eachfile= 0, NUMFILES-1 do begin
if def[0,eachfile].startangle eq def[1,eachfile].startangle
then begin
    rotating[eachfile]= 0
endif
endfor
```

Finally we finished reading and calculating all needed data.

```
;#############################################################
; PLOT 1: Show each module pro interval number, with no
rotation
;
#############################################################
```

Here we are about to plot “total counts” data for each cassette in each interval. As mentioned above, “total counts” data is in the value[0](first element of value array in cassette structure), and we need to graph:
value[0] of the first cassette in each interval,
value[0] of the second cassette in each interval and so on. If we wish show “counts for singles” we shall take second element of value array in cassette structure, we graph
value[1]. This will give us 20 graphs.

Like in reading, we can not simply plot the data into a graph from our structures. So we copy the data into axis data arrays and then plot those. Here we create two arrays. X axis array has the “interval number” and Y axis array has the “total counts” data for each interval. These arrays will be refilled each time we loop for a new graph.

```
tempx= lonarr(NUMINTERVALS*NUMFILES)
tempy= lonarr(NUMINTERVALS*NUMFILES)
```

Here we set the name of our Y axis. We left a little space to place a number at the end of this text to show which cassette we are graphing

```
titley= 'Values for Module:   '

for eachrow= 0, NUMCASSETTES- 1 do begin
```

Loop through each row in each file

```
    for eachfile= 0, NUMFILES-1 do begin
```

at each interval

```
        for eachinterval= 0, NUMINTERVALS- 1 do begin
```

Filling Y axis data for current interval (total counts)

```
            tempy(eachfile* NUMINTERVALS+ eachinterval)=
data[eachinterval, eachrow, eachfile].value[0]
```

Filling X axis data current interval number (all interval numbers spanning all files are stored in the X axis data) Maximum number of X axis data can only be NUMINTERVALS* NUMFILES

```
            tempx(eachfile* NUMINTERVALS+ eachinterval)=
eachfile* NUMINTERVALS+ eachinterval


        endfor
    endfor
```

Here is a little routine to put the current processed module group number at the end of the y title string. As we mentioned above, this is why we left extra space at the end of that title.

```
    titlenum= string(eachrow)
    titlenum= strtrim(eachrow, 2)
```

We added the module group number after character position of the y title string.

```
    strput, titley, titlenum, 18
```

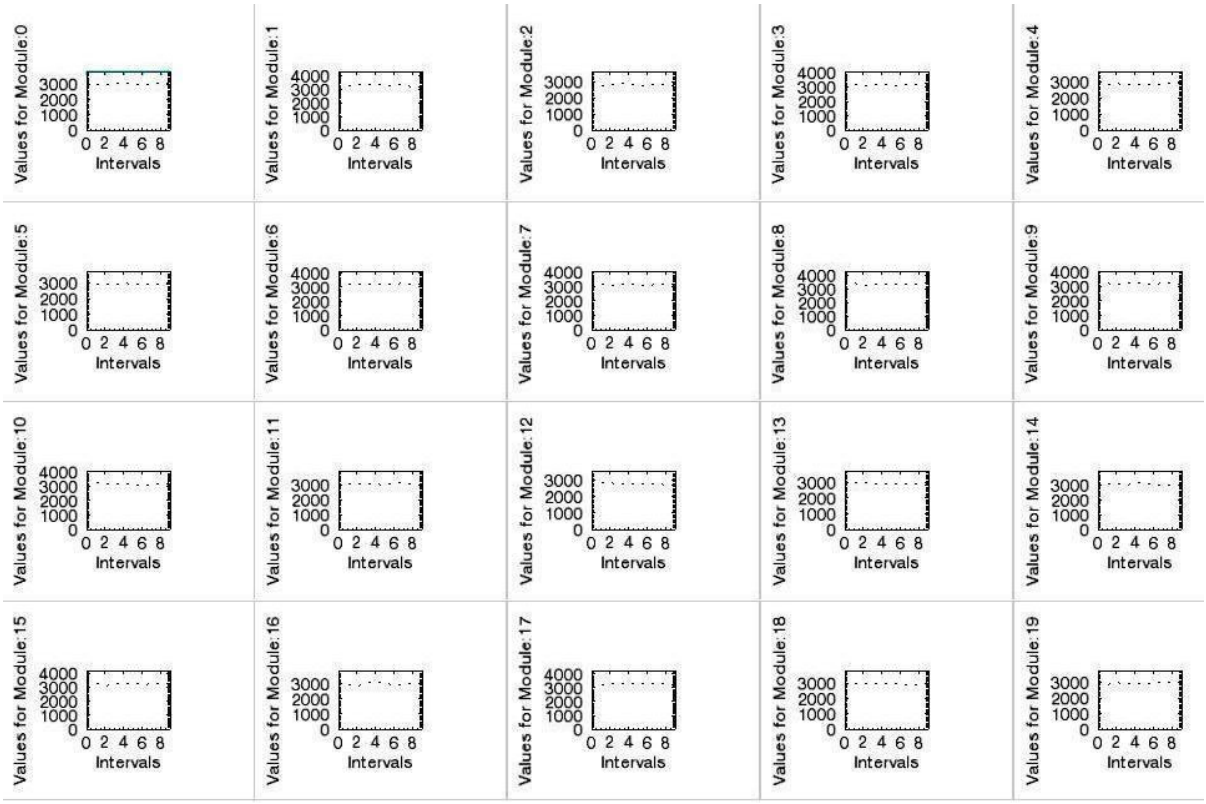
Then we plot the graphics.

```
    iplot, tempx, tempy,$
        linestyle=1,$
```

Here we calculate a y range maximum so that our graph looks a bit better. We used the formula (MAX Y VALUE FOR THE GRAPH)* 5/4

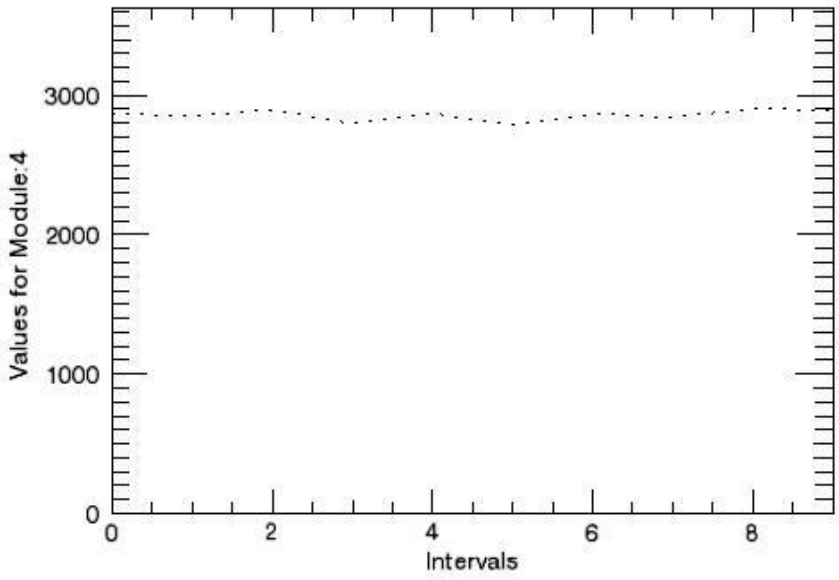
```
        yrange=[0,5* max(tempy)/ 4],$
        view_grid=[5,4],/view_next ,$
        title='Counts',$
        xtitle= 'Angles', $
        ytitle= titley
endfor
```

As an example we can draw intervals versus module group's counts for blank measurement (Appendix A). Graph 3.1 shows all module groups together.



Graph 3.1: Interval versus module group's values for all cassettes.

Graph 3.2 shows one of 20 module groups (module group 1= cassette 19).



Graph 3.2: Intervals versus 4th module group counts

```
;################################
; PLOT 2: Show each module pro angle
;################################

;a) for cassette positions (For long half-life radioisotopes
like 68Ge)
```

In the case of the gantry rotation, we plot “total counts” data for each cassette pro angle.

```
tempx= lonarr(NUMINTERVALS*NUMFILES)
tempy= fltarr(NUMINTERVALS*NUMFILES)
tempxcassette=lonarr(NUMINTERVALS*NUMFILES)
```

In these graphics we may indicate minimum and maximum of Y

```
minxarray= lonarr(NUMCASSETTES)
maxxarray= lonarr(NUMCASSETTES)


titley= 'Values for Module:   '
```

Here we set the name of our X axis. We left a little space to place cassette position angles where Y value is at minimum and maximum.

```
titlex= 'Cassette Position (xmin=    , xmax=    )'


for eachrow= 0, NUMCASSETTES- 1 do begin

    for eachfile= 0, NUMFILES-1 do begin

        for eachinterval= 0, NUMINTERVALS- 1 do begin


            tempy(eachfile* NUMINTERVALS+ eachinterval)=
data[eachinterval, eachrow, eachfile].value[0]??

            tempx(eachfile* NUMINTERVALS+ eachinterval)=
eachfile* NUMINTERVALS+ eachinterval
```

We also calculate each angle according to the beginning angle we read from the header. Cassettes are separated with 18 degrees angle difference. So we start from the header angle and the first cassette’s angle is that header angle. Then we calculate the other angles by

using a circular formula which starts over from 360 if it goes to negative values. For example:

Header angle is 0, this means that cassette 20 sits at;

0-((0+1)*18)= -18 (=342)

Cassette 19 sits at;

0-((1+1)*18)= -36(= 324)

and so on. All angles for the cassettes are calculated like this in a loop.

```
      tempxcassette=tempx-((eachrow+1)*18)

      endfor
    ; print, tempy
  endfor

  titlex= 'Cassette Position (xmin=    , xmax=    )'

  titlenum= string(eachrow)
  titlenum= strtrim(titlenum, 2)

  strput, titley, titlenum, 18
```

We found here minimum and maximum value of Y axis.

```
  minx= where(tempy eq min(tempy))
  maxx= where(tempy eq max(tempy))

  tmix= tempxcassette[minx[0]]
  tmax= tempxcassette[maxx[0]]
```

Then we calculate the angle if it goes to negative values.

```
  if tmix lt 0 then tmix=tmix+ 360
  minxarray[eachrow]= tmix
  if tmax lt 0 then tmax=tmax+ 360
  maxxarray[eachrow]= tmax

  minxstring= string(tempxcassette[minx[0]])
  minxstring= strtrim(minxstring, 2)
  maxxstring= string(tempxcassette[maxx[0]])
  maxxstring= strtrim(maxxstring, 2)

  strput, titlex, minxstring, 24
  strput, titlex, maxxstring, 35
```

We can print angles of cassette positions, where “total count” take minimum and maximum values.

```
print, 'tempxcassette[minx]: ', tempxcassette[minx]
print, 'tempxcassette[maxx]: ', tempxcassette[maxx]
```

If we print min and maximum cassette angles' values we get the values like in Output 3.1.

Output 3.1: Minimum and maximum angle values of the cassette positions. It starts from 0 and ends with 19th value.

```
tempxcassette[minx][19]:       64
tempxcassette[maxx][19]:       241

tempxcassette[minx]:       47
tempxcassette[maxx]:       245

tempxcassette[minx]:       60
tempxcassette[maxx]:       232

tempxcassette[minx]:       57
tempxcassette[maxx]:       214

tempxcassette[minx]:       59
tempxcassette[maxx]:       216

tempxcassette[minx]:       67
tempxcassette[maxx]:       198

tempxcassette[minx]:       50
tempxcassette[maxx]:       226

tempxcassette[minx]:       80
tempxcassette[maxx]:       205

tempxcassette[minx]:       23
tempxcassette[maxx]:      -120

tempxcassette[minx]:       71
tempxcassette[maxx]:      -107

tempxcassette[minx]:       61
tempxcassette[maxx]:      -125

tempxcassette[minx]:       43
tempxcassette[maxx]:      -143

tempxcassette[minx]:       38
tempxcassette[maxx]:      -107

tempxcassette[minx]:       49
tempxcassette[maxx]:      -125

tempxcassette[minx]:       77
tempxcassette[maxx]:      -131

tempxcassette[minx]:       59
tempxcassette[maxx]:      -149

tempxcassette[minx]:      -297
tempxcassette[maxx]:      -142

tempxcassette[minx]:       17
tempxcassette[maxx]:      -127

tempxcassette[minx]:       14
tempxcassette[maxx]:      -134

tempxcassette[minx]:      -295
tempxcassette[maxx]:      -139
```

But as seen from output some values are negative. Like mentioned above if we print tmix and tmax, we can find positive values for each module group.

```
print, 'tempxcassette[minx]: ', tmix
print, 'tempxcassette[maxx]: ', tmax
```

So the output looks like Output 3.2;

Output 3.2: Real minimum and maximum angle values for the cassettes.

```
tempxcassette[minx]:      64
tempxcassette[maxx]:      241

tempxcassette[minx]:      47
tempxcassette[maxx]:      245

tempxcassette[minx]:      60
tempxcassette[maxx]:      232

tempxcassette[minx]:      57
tempxcassette[maxx]:      214

tempxcassette[minx]:      59
tempxcassette[maxx]:      216

tempxcassette[minx]:      67
tempxcassette[maxx]:      198

tempxcassette[minx]:      50
tempxcassette[maxx]:      226

tempxcassette[minx]:      80
tempxcassette[maxx]:      205

tempxcassette[minx]:      23
tempxcassette[maxx]:      240

tempxcassette[minx]:      71
tempxcassette[maxx]:      253

tempxcassette[minx]:      61
tempxcassette[maxx]:      235

tempxcassette[minx]:      43
tempxcassette[maxx]:      217

tempxcassette[minx]:      38
tempxcassette[maxx]:      253

tempxcassette[minx]:      49
tempxcassette[maxx]:      235

tempxcassette[minx]:      77
tempxcassette[maxx]:      229

tempxcassette[minx]:      59
tempxcassette[maxx]:      211

tempxcassette[minx]:      63
tempxcassette[maxx]:      218

tempxcassette[minx]:      17
tempxcassette[maxx]:      233
```

```
tempxcassette[minx]:        14
tempxcassette[maxx]:       226

tempxcassette[minx]:        65
tempxcassette[maxx]:       221
```

We can print averages of minimum and maximum angle values (Output 3.3).

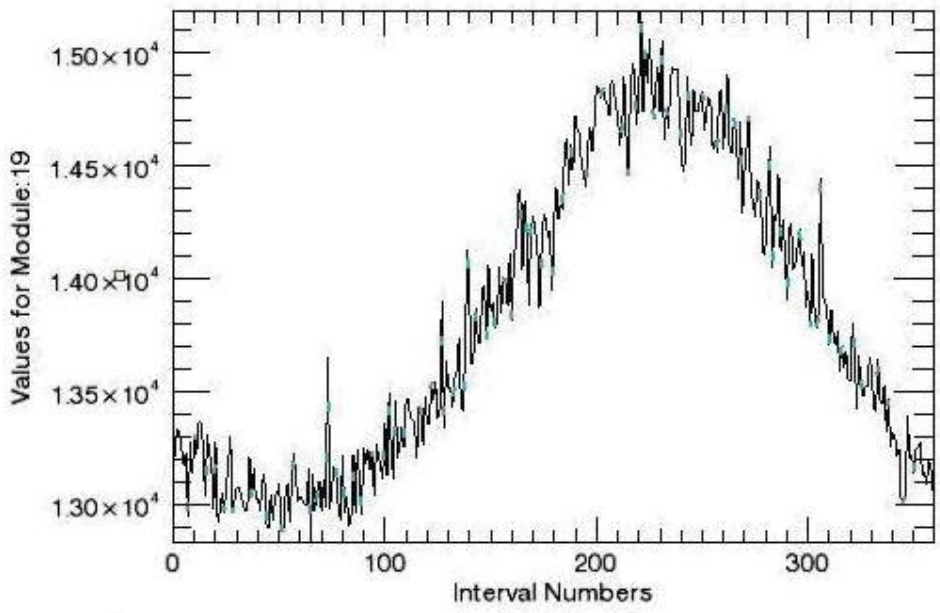
```
print, 'average of minxes: ', median(minxarray)
print, 'average of maxxes: ', median(maxxarray)
```

Output 3.3: Averages of minimum and maximum angle values.

```
average of minxes:      59.0000
average of maxxes:      229.000
```

If we don't indicate y range in Graph 3.3, the program assigns appropriate y range itself.

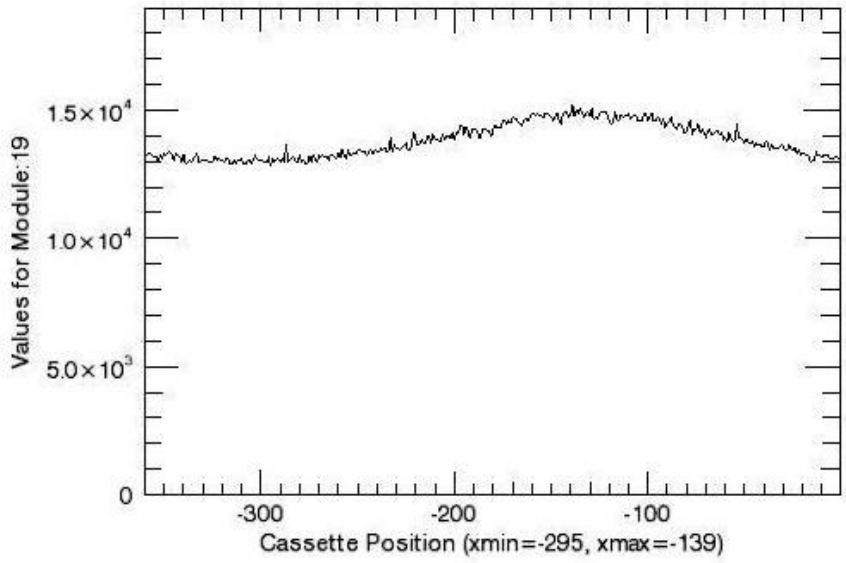
```
iplot, tempx, tempy,$
xtitle='Interval Numbers',$
```



Graph 3.3: An example graph for no indicated y range.

Here we can formulate a y range that starts from 0 (Graph 3.4)

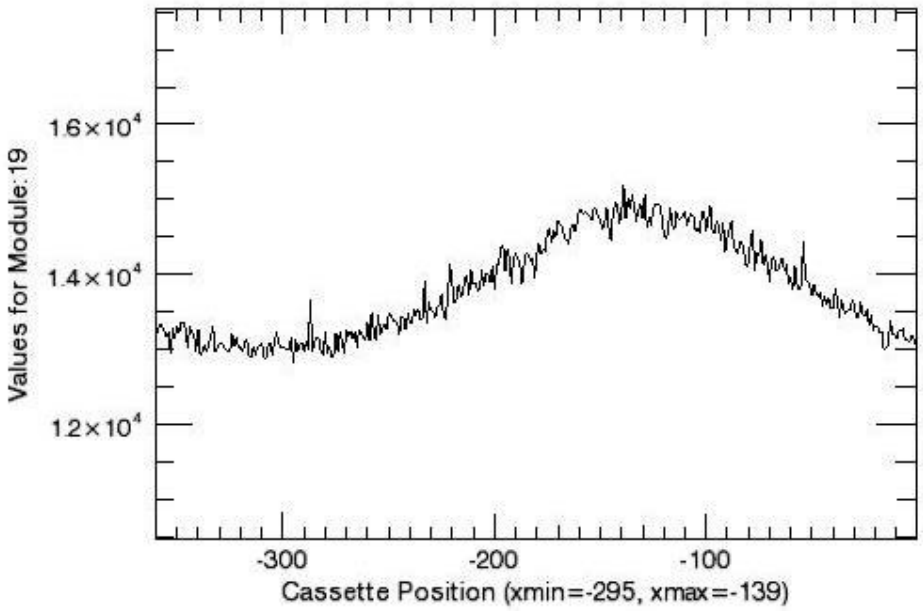
```
iplot,tempxcassette, tempy,$ ;for cassette angle
   ;linestyle=1,$;for dot curve

   yrange=[0,max(tempy)+ max(tempy)/ 4],$;beginning at 0
```



Graph 3.4: An example graph for y axis starts from 0.

Y values can be located between minimum and maximum values (Graph 3.5).

```
        yrange=[min(tempy)-(max(tempy)-
min(tempy)),max(tempy)+(max(tempy)-min(tempy))],$
```



Graph 3.5: Y values located between minimum and maximum values.

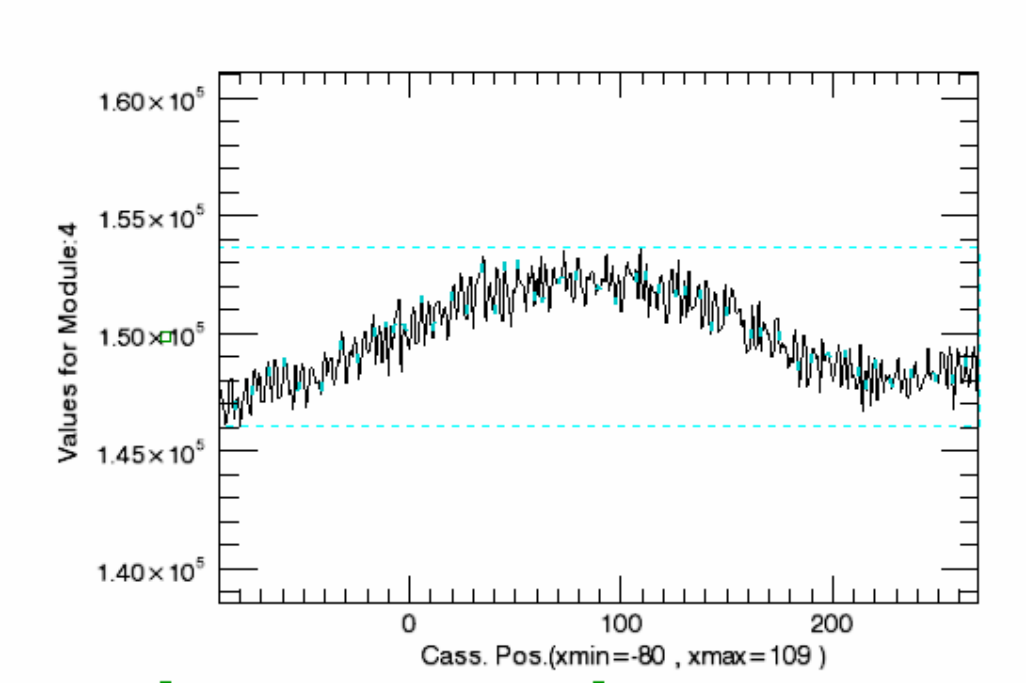
```
        ;view_grid=[5,4],/view_next ,$
        title='Counts',$
        xtitle= titlex, $
        ;xtitle='Cassette Positions',$
        ytitle= titley

endfor

;print,'tempy 0: ', tempy[0]


; ********************************************
; b) Decay Correction for F18 data only
```

If we plot any graph related with ^{18}F source we shall consider its short half life (109, 8 min= 6588 s) and make decay correction. As known from the Equation 2.4 we shall multiply each count value with exp(-(0.693/6588)) to reach the initial count values.

```
titley= 'Values for Module:   '
titlex= 'Cassette Position (xmin=    , xmax=    )'


for eachrow= 0, NUMCASSETTES- 1 do begin
    for eachfile= 0, NUMFILES-1 do begin

        for eachinterval= 0, NUMINTERVALS- 1 do begin
            tempy(eachfile* NUMINTERVALS+ eachinterval)=
data[eachinterval, eachrow, eachfile].value[0] $
                /exp(-(0.693/6588)*
def[eachinterval,eachfile].time)
            tempx(eachfile* NUMINTERVALS+ eachinterval)=
eachfile* NUMINTERVALS+ eachinterval

         tempxcassette=tempx-((eachrow+1)*18); For cassette
position(For F18)
        endfor
       ;print, tempy
    endfor

    titlex= '       Cass. Pos.(xmin=    , xmax=    )'

    titlenum= string(eachrow)
    titlenum= strtrim(titlenum, 2)
    strput, titley, titlenum, 18

       minx= where(tempy eq min(tempy))
    maxx= where(tempy eq max(tempy))
```

```
    tmix= tempxcassette[minx[0]]
    tmax= tempxcassette[maxx[0]]

    if tmix lt 0 then tmix=tmix+360
    minxarray[eachrow]= tmix
    if tmax lt 0 then tmax=tmax+360
    maxxarray[eachrow]= tmax


    minxstring= string(tempxcassette[minx[0]])
    minxstring= strtrim(minxstring, 2)
    maxxstring= string(tempxcassette[maxx[0]])
    maxxstring= strtrim(maxxstring, 2)

    strput, titlex, minxstring, 23
    strput, titlex, maxxstring, 34



;    print, titlex

;    print, titlenum, ': minx: ', minx
;    print, 'translated x: ', tempxcassette[minx]

 ;   iplot, tempx, tempy,$

     print, 'tempxcassette[minx]: ', tempxcassette[minx]
     print, 'tempxcassette[maxx]: ', tempxcassette[maxx]

;     iplot,tempxcassette, tempy,$ ;for cassette angle
;        ;linestyle=1,$
;        ;yrange=[0,max(tempy)+ max(tempy)/ 4],$
;         yrange= [min(tempy)-
median(tempy)/20,max(tempy)+median(tempy)/20],$
;
;        ;view_grid=[5,4],/view_next ,$
;        title='Counts',$
;        xtitle= titlex, $
;        ytitle= titley


Endfor
```

In Graph 3.6, an example plot is given for ^{18}F data for cassette position versus one of the module group.



Graph 3.6 Module group 4 values plotted for ^{18}F data.

```
print, 'average of minxes: ', median(minxarray)
print, 'average of maxxes: ', median(maxxarray)



print, 'totalcounts 0: ', data[1, 0, 0].value[0]
print, 't: ', def[1,0].time
print, 'expon: ', exp(-(0.693/6588)* def[1,0].time)
print,'result 0: ', data[1, 0, 0].value[0]/exp(-(0.693/6588)*
def[1,0].time)

;#############################################################
; PLOT 2:Sum of each module in each interval and file
;#############################################################
```

As above, we loop over each interval and file and sum all the "total counts" (value[0]) for each module. We get one value for each module group.

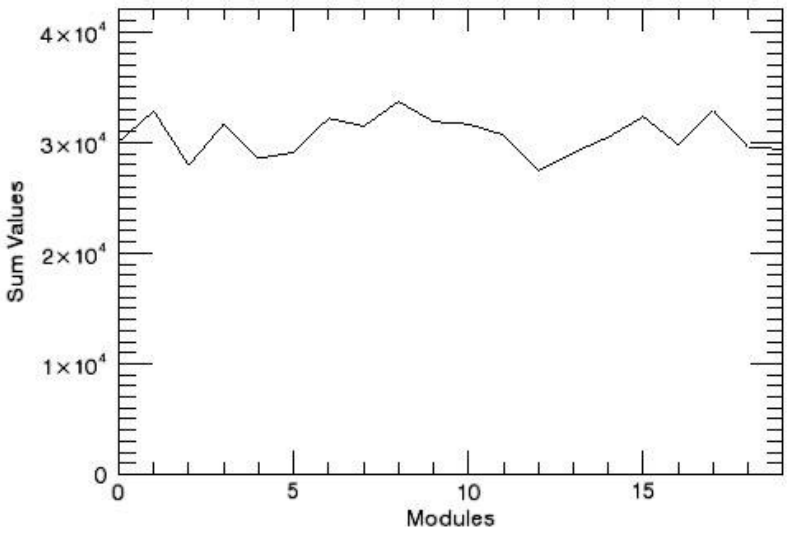
We will fill our X plotting array with just numbers 0-19 and our Y plotting array will have the sums for each module group. Example: for module group 0, we get value[0] from interval 1 of file 1, add to that the value[0] from interval 2 of file 1 and so on.

```
tempx= indgen(NUMCASSETTES)
tempy= lonarr(NUMCASSETTES)

for eachrow= 0, NUMCASSETTES- 1 do begin
    for eachfile= 0, NUMFILES-1 do begin
```

```
        for eachinterval= 0, NUMINTERVALS- 1 do begin
            tempy(eachrow)= tempy(eachrow)+ data[eachinterval,
eachrow, eachfile].value[1]
       endfor
    endfor
endfor
```

We can plot sum of each module in each interval (Graph 3.7)

```
iplot,tempx,tempy,$
       yrange=[0,100000000],$
       title='Sum',$
       xtitle='Modules', $
       ytitle='Sum Values'
```



Graph 3.7: Each module groups versus sum of value[0] in each interval.

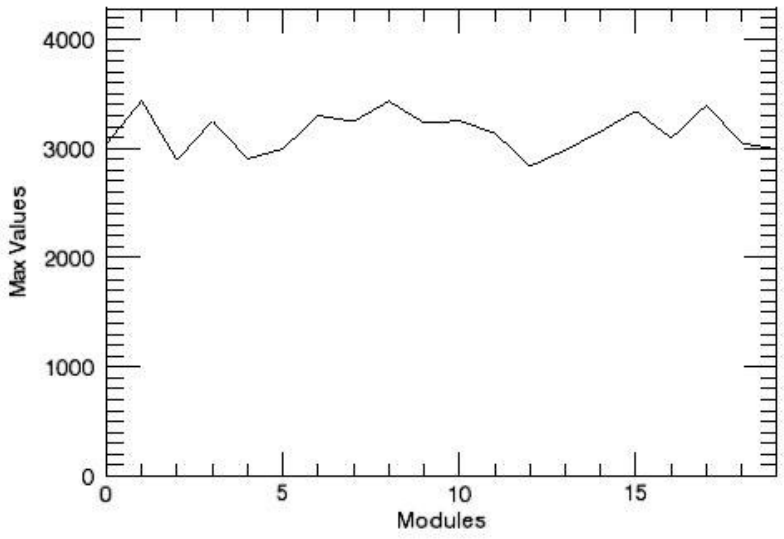
```
;#############################################################
; PLOT 3:Max of each module group in each interval and file
;#############################################################
```

Here we loop through each "total counts" in each interval and file per module group (like we do in plot 2), but we loop only to find the maximum value from these "total counts". After we finish searching through all files and intervals for the module group, we loop for the next module group and find the next maximum "total counts" to plot. We put those maximums to our Y axis array (tempmax). And our X axis array is like the above, module group numbers (from 0 to 19)

```
tempx= indgen(NUMCASSETTES)
tempy= lonarr(NUMINTERVALS*NUMFILES)
tempmax=lonarr(NUMCASSETTES)

for eachrow= 0, NUMCASSETTES- 1 do begin
    for eachfile= 0, NUMFILES-1 do begin
        for eachinterval= 0, NUMINTERVALS- 1 do begin
            tempy(eachfile* NUMINTERVALS+ eachinterval)=
data[eachinterval, eachrow, eachfile].value[0]
            tempmax(eachrow)=max(tempy)
       endfor
    endfor
endfor
```

We can plot sum of each module in each interval (Graph 3.8)

```
;iplot,tempx,tempmax,$
;       yrange=[0,200000],$
;       title='Max',$
;        xtitle='Modules', $
;        ytitle='Max Values'
```



Graph 3.8: Each module groups versus maximums of value[0] inGraph 3.1: Interval versus module group's values for all cassettes.

```
;#############################################################
; PLOT 4:Min of each module group in each interval and file
;#############################################################
```

Exactly like plot 3, we loop through each interval in each file to find the minimum of "total counts" this time. Then we plot it like the above.

```
tempx= indgen(NUMCASSETTES)
tempy= lonarr(NUMINTERVALS*NUMFILES)
tempmax=lonarr(NUMCASSETTES)

for eachrow= 0, NUMCASSETTES- 1 do begin
    for eachfile= 0, NUMFILES-1 do begin
        for eachinterval= 0, NUMINTERVALS- 1 do begin
            tempy(eachfile* NUMINTERVALS+ eachinterval)=
data[eachinterval, eachrow, eachfile].value[0]
            tempmax(eachrow)=min(tempy)
       endfor
    endfor
endfor

;iplot,tempx,tempmax,$
;       yrange=[0,200000],$
;       title='Min',$
;        xtitle='Modules', $
;        ytitle='Min Values'


;#############################################################
; PLOT 5:
;#############################################################
```

Now we make a menu which asks if the user wants to plot all interval graphs (with respect to module groups) or just the intervals which fall in between an inputted range of time.

```
print, 'Now we can plot the desired intervals.'
print, '1.Do you want to plot All? ', NUMINTERVALS, '
intervals times ', NUMFILES, ' files'
print, '2.Do you want to plot a time range (seconds)'
```

As usual we create temporary X and Y axis arrays. This time these arrays will be refilled for each interval's module's "total counts" data (Yaxis); and module group number (X axis).

```
tempx= lonarr(NUMCASSETTES)
tempy= lonarr(NUMCASSETTES)
```

We initialize the answer variable (var), this will be used later:

```
var= 2
```

We will repeat asking while the choice is not 1 or 2 (the only two possible choices), so we create a label to go to (ask:)

```
ask:
read, var

if (var ne 1) and (var ne 2) then begin
    print,'choose 1 or 2'
    goto, ask
endif
```

In case the user wants to draw all the graphs (choice 1), we do the following loop:

```
if (var eq 1)  then begin
```

Choice is 1

```
for eachfile= 0, NUMFILES-1 do begin
```

Loop through each file

```
    for eachinterval= 0, NUMINTERVALS- 1 do begin
```

and each interval

```
        for eachrow= 0, NUMCASSETTES- 1 do begin
```

and each module group

```
            tempy(eachrow)= data[eachinterval, eachrow,
eachfile].value[1]]
            tempx(eachrow)= eachrow
```

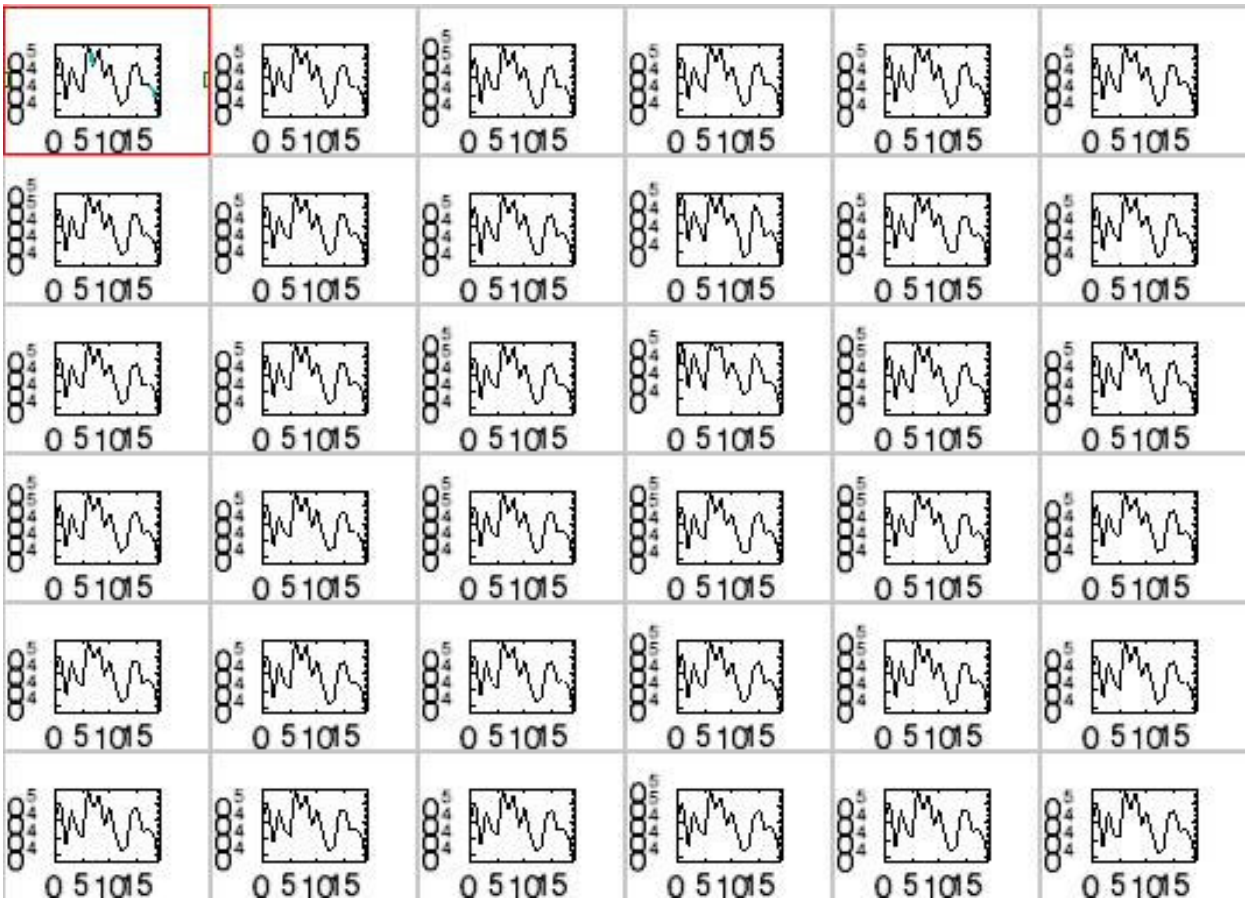
Store the module group data in tempy, store the module group number in tempx, keep on looping

```
          endfor
```

Here we have the “total counts” data for each module group in tempy, ready to plot (Graph 3.9)

```
         iplot,tempx,tempy,$
         view_grid=[6,6],/view_next ,$
        ;yrange=[0,max(tempy)+ max(tempy)/ 4],$;beginning at 0
```

```
        title='Row Values',$
        xtitle= 'Modules', $
        ytitle= 'Module Counts'
    endfor

endfor

endif
```



Graph 3.9: Count Values in each interval versus Module Groups are shown for 36 intervals

Our other menu item:

```
if (var eq 2) then begin
```

Choice is 2; get the time range from the user

```
first_input=0
second_input=0

print,'enter beginning of time range and enter end of time
range'
```

```
read, first_input, second_input

viewvalue=0
```

In each file

```
for eachfile= 0, NUMFILES-1 do begin

    for eachinterval= 0, NUMINTERVALS- 1 do begin
```

Check if the current interval falls in between the time range inputted by the user.

```
      if (def[eachinterval, eachfile].time ge first_input) and
$
             (def[eachinterval, eachfile].time le second_input)
then begin
```

Counting the graphs which fall into our time interval using this "if"

```
       viewvalue=viewvalue+1

       endif
    endfor
endfor
```

Since the number of graphs is dynamic, when we draw the graphs, we want them to be aligned nicely. For example if there are 4 graphs, we want them to appear in a 2x2 graph matrix:

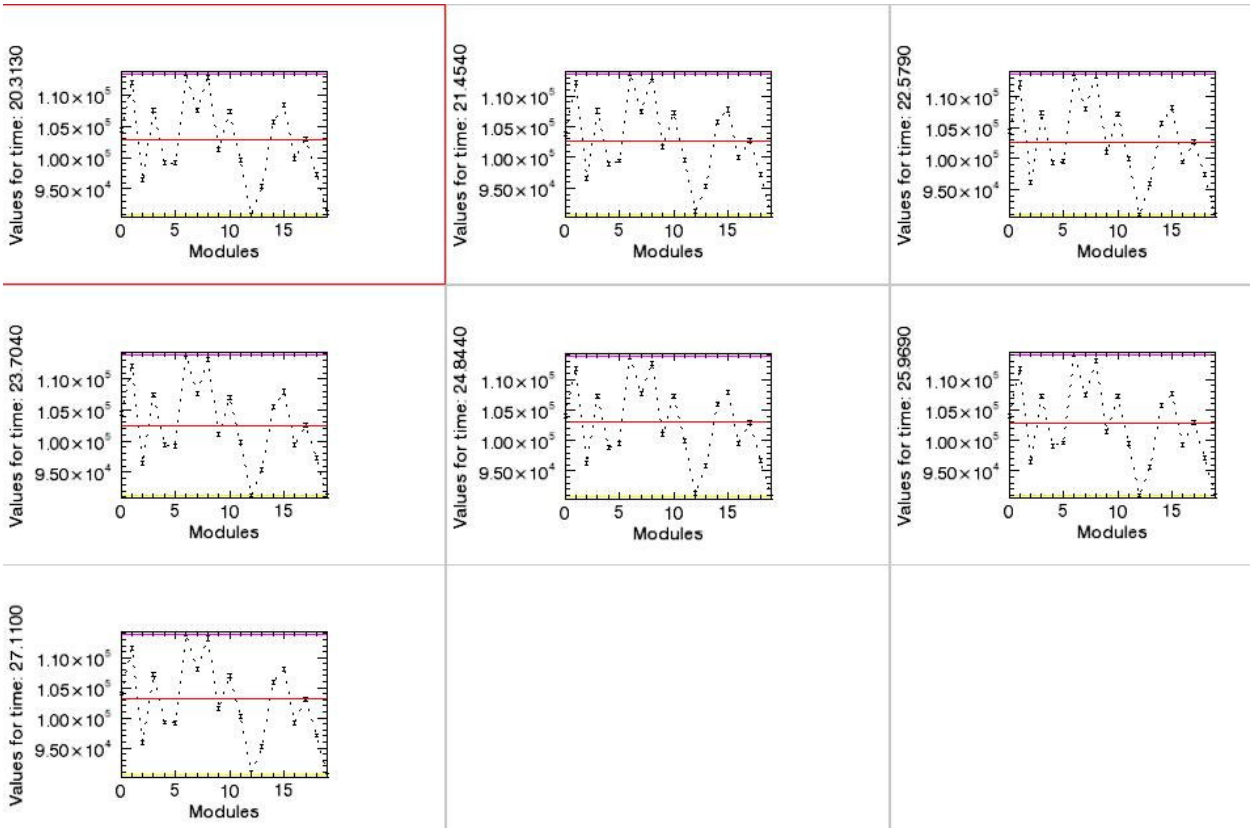
[graph1] [graph2]

[graph3] [graph4]

So we make a formula. We take the square root of the number of graphs to find out how big a square we can fit this number of graphs into. But the sides of our square needs to be whole number, and it can not be smaller than the square root of the graphs, so we round this square root upwards using the ceil function like this:

```
viewvalue_x=ceil(sqrt(viewvalue))
viewvalue_y=ceil(sqrt(viewvalue))
```

For example if viewvalue is seven, ceil(sqrt(7))= ~ceil(2.65)= 3 then viewvalue_x =3 and viewvalue_y =3, so we get a 3x3 matrix (Graph 3.10)



Graph 3.10: Seven graphs are shown 3x3 matrixes

Now it is drawing time!

```
for eachfile= 0, NUMFILES-1 do begin
```

In each file

```
    for eachinterval= 0, NUMINTERVALS- 1 do begin
```

Temporarily store all the module “total counts” data for each interval

```
        for eachrow= 0, NUMCASSETTES- 1 do begin
            tempy(eachrow)= data[eachinterval, eachrow,
eachfile].value[0]
            tempx(eachrow)= eachrow
        endfor
```

and draw it only if it falls within the desired range

```
        if (def[eachinterval, eachfile].time ge first_input)
and $
            (def[eachinterval, eachfile].time le second_input)
then begin
```

Like before, we manipulate the Y title string of each graph and put a descriptive number at the end of it. Here we place the current time of the interval data after position 17 of the title string.

```
            titley= 'Values for time:          '
            titlenum= string(def[eachinterval, eachfile].time)
            titlenum= strtrim(def[eachinterval,
eachfile].time, 2)
            strput, titley, titlenum, 17
```

We calculate the height of our error bars

```
            err=sqrt(tempy)
```

We make an array for the Y values of our median line

```
            med=make_array(NUMCASSETTES,value=median(tempy))
```

We make an array for the Y values of our min value line

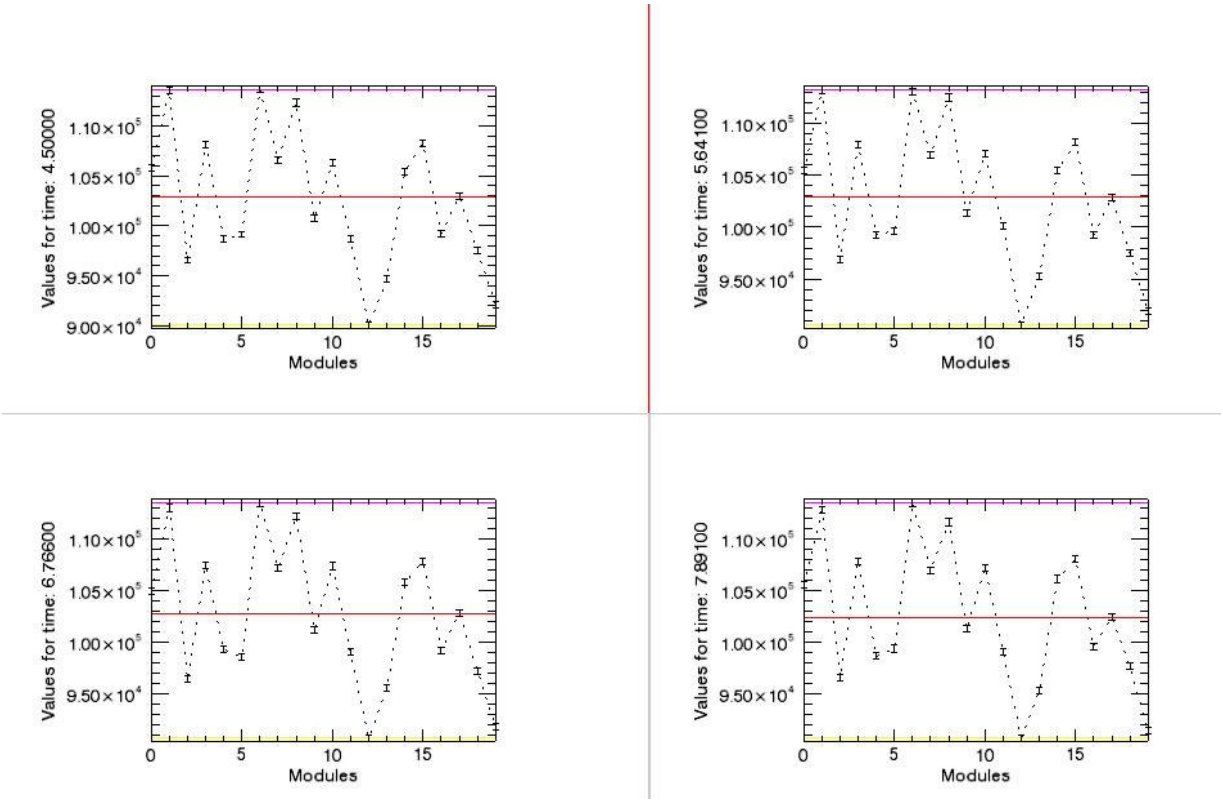
```
            minimum=make_array(NUMCASSETTES,value=min(tempy))
```

We make an array for the Y values of our max value line

```
            maximum=make_array(NUMCASSETTES,value=max(tempy))
```

Here, we draw (Graph 3.11);

```
          iplot,tempx,tempy, linestyle=1,$
          yerror=err,/y_errorbar, $

          view_grid=[viewvalue_x,viewvalue_y],/view_next
          iplot,tempx,med,thick=2,color=[255,0,0],/overplot

iplot,tempx,minimum,thick=2,color=[255,255,0],/overplot

iplot,tempx,maximum,thick=2,color=[255,0,255],/overplot,$
          ;/histogram

          stylename='Meshif', $
          title='Row Values',$
          xtitle= 'Modules', $ ; write maximum after module
          ytitle= titley
```



Graph 3.11: Input time is given between 4 s and 8 s. The program found out graphs for the relevant time interval

We need to place the median value and standard deviation after one of the axis strings as well.

```
print,med
print,meanabsdev(tempy)


       endif
    endfor
endfor

endif

end; PROGRAM
```

3.2 SPECIFICATIONS OF THE USED DATA

^{68}Ge and ^{18}F sources and blank scan measurements data taken from ClearPET is used in this study. Module_Check program is tested with total counts of the single photons (first column of the stored data). The specification of the sources and data are explained below.

3.2.1. ^{68}Ge Data with Gantry Rotation

^{68}Ge rod source is placed in the center of FOV, then gantry is rotated 1° in each step to complete overall 360° and 360 measurements are taken in step and shoot mode. Duration of each measurement is 2 s and total effective measurement time is 12 min. While the data are stored 360 intervals are saved in one .ang file.

3.2.2. ^{18}F Data with Gantry Rotation

590 μCi ^{18}F is put homogeneously in water filled cylinder phantom which has 80mm diameter and 110 mm length and the phantom is placed in the center of FOV. The gantry is rotated 1° in each step to complete overall 360° and 360 measurements are taken in step and shoot mode. Duration of each measurement is 1 s and total effective measurement time is 6 min. While the data are stored, 360 intervals are saved in 10 .ang files separately in which each file has 36 intervals.

3.2.3. Blank Scan Data with Gantry Rotation

The gantry is rotated 1° in each step to complete overall 360° and 360 measurements are taken in step and shoot mode while there was no activity in FOV. Duration of each measurement is 2 s and total effective measurement time is 12 min. While the data are stored, 360 intervals are saved in 10 .ang files separately in which each file has 36 intervals.

3.2.4. Inaccurate Blank Scan Data with Gantry Rotation

The gantry is rotated 3° in each step to complete overall 180° and 60 measurements are taken in step and shoot mode while there was no activity in FOV. The signals of blank scan measurements come from the ^{176}Lu crystals where they exist in the PMTs. Duration of

each measurement is10 s and total effective measurement time is 10 min. While the data are stored, 360 intervals are saved in 10 files separately in which each file has 36 intervals. Measurements taken from 19th cassette are inaccurate in this file.

3.2.5. ^{68}Ge Data without Gantry Rotation

^{68}Ge rod source is placed in the center of FOV, 100 measurements are taken without rotating gantry. Duration of each measurement is1 s and total effective measurement time is 100 s. While the data are stored 100 intervals are saved in one file.

3.2.6. Blank Scan Data without Gantry Rotation

While there was no activity in FOV 10 measurements are taken without rotating gantry. The signals of blank scan measurements come from the ^{176}Lu crystals where they exist in the PMTs. Duration of each measurement is1 s and total effective measurement time is 10 s. While the data are stored 10 intervals are saved in one file.

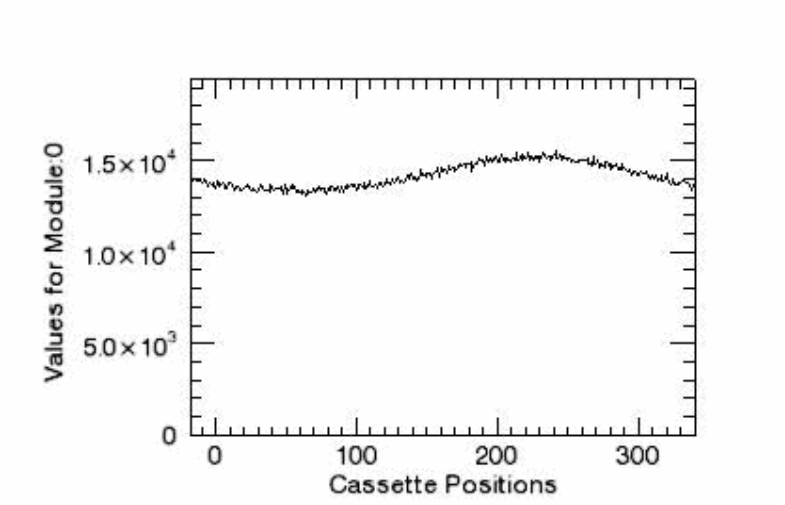
4 RESULTS

4.1 GRAPHS and OUTPUTS

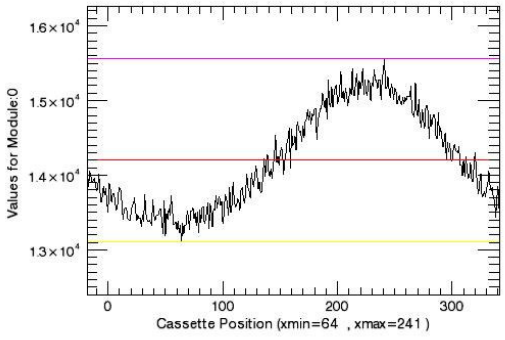
The packed program named Module_Check is written. This program is read the counts of the module groups for ^{68}Ge and ^{18}F sources and blank scan which are stored as ASCII format. Then it converts these data to a comprehensible condition via plotting graphs and finding average, minimum and maximum values and their standard deviations. The results of fixed gantry state measurements and measurements during gantry rotation are given below.

4.1.1 ^{68}Ge Data with Gantry Rotation

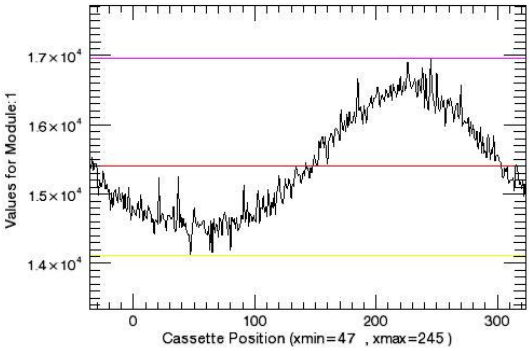
Cassette positions versus each module group counts graphs are shown in Graph 4.2. A graph of Module Group 0 is given in Graph 4.1 as an example whose y axis begins from zero. Graphs in Graph 4.2 take part between minimum and maximum y values which look like sinus waves. Minimum, maximum and average Y values are shown as lines in these graphs.



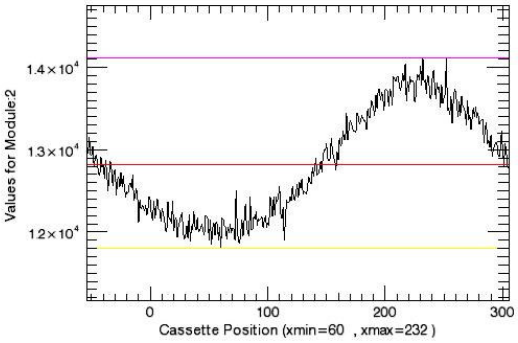
Graph 4.1: An example graph for cassette positions versus values for module groups for ^{68}Ge data with gantry rotation. Y values begin from zero.



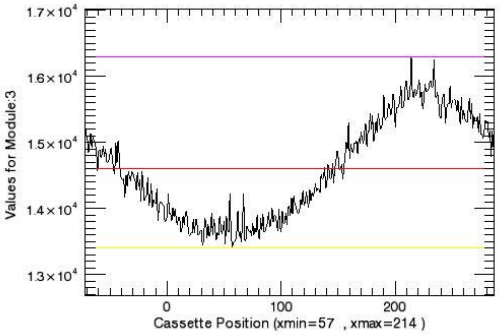
Graph 4.2:a



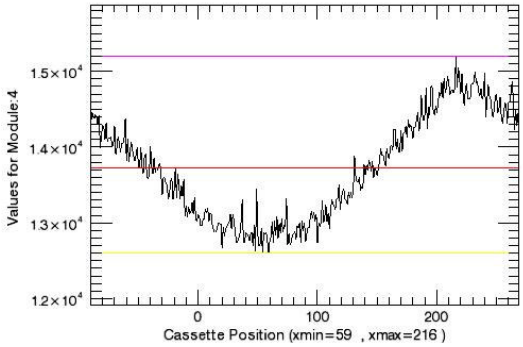
Graph 4.2:b



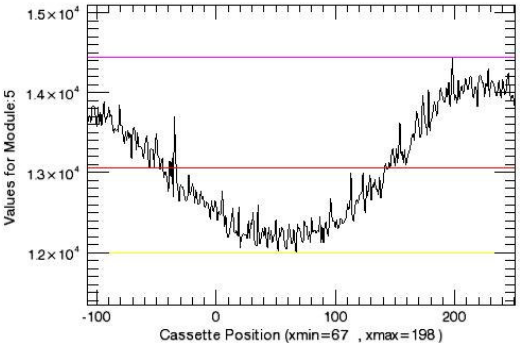
Graph 4.2:c



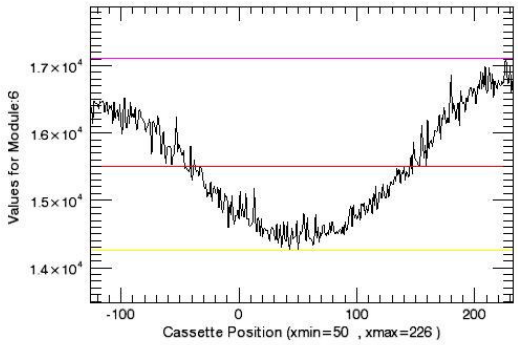
Graph 4.2:d



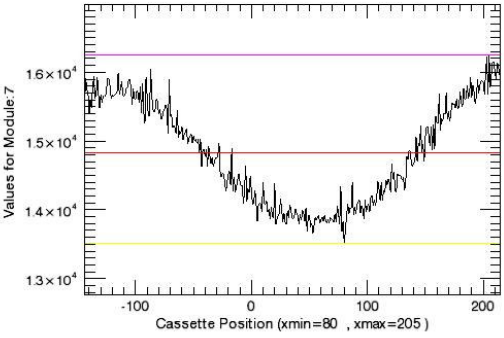
Graph 4.2:e



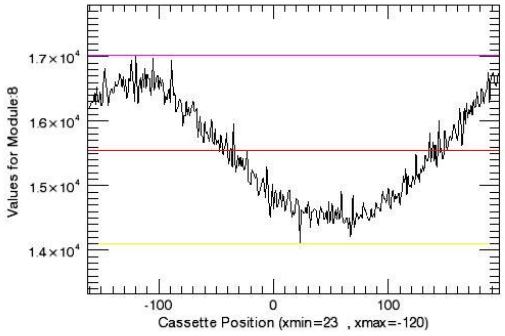
Graph 4.2:f



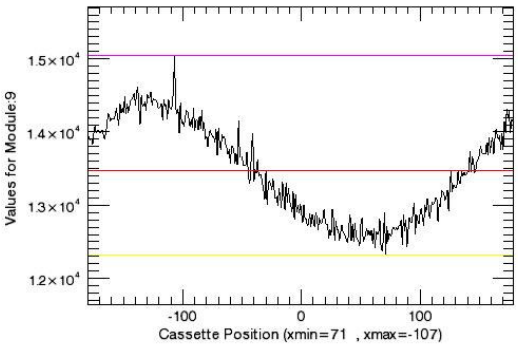
Graph 4.2:g



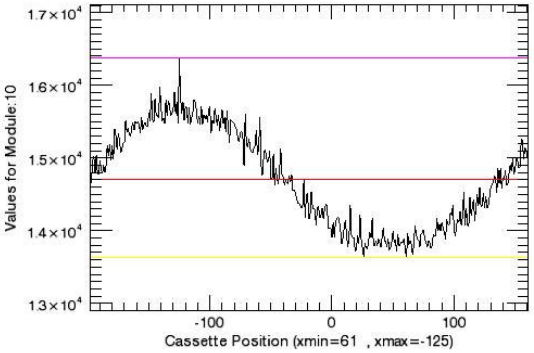
Graph 4.2:h



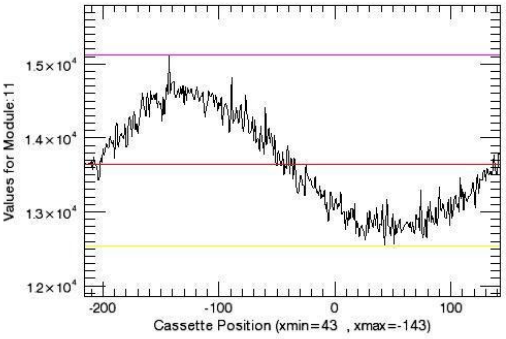
Graph 4.2:i



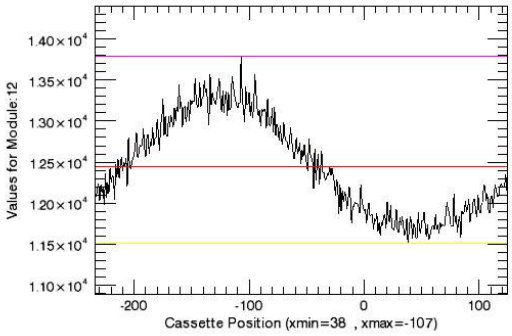
Graph 4.2:j



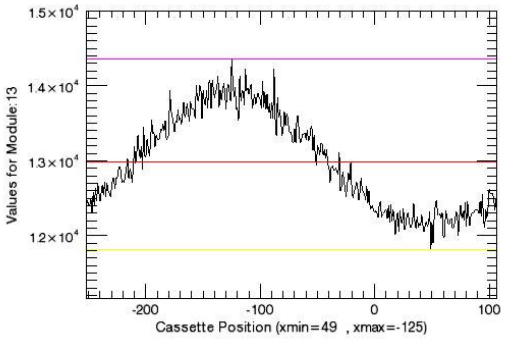
Graph 4.2:k



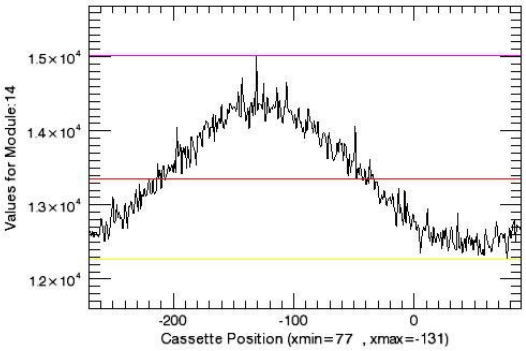
Graph 4.2:l



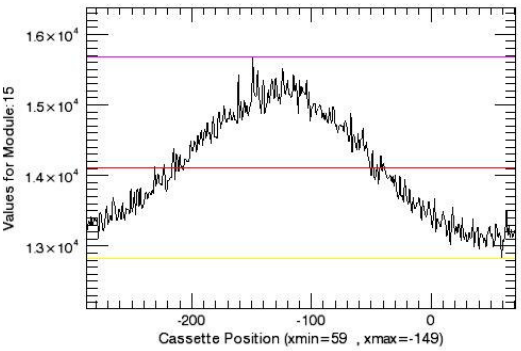
Graph 4.2:m



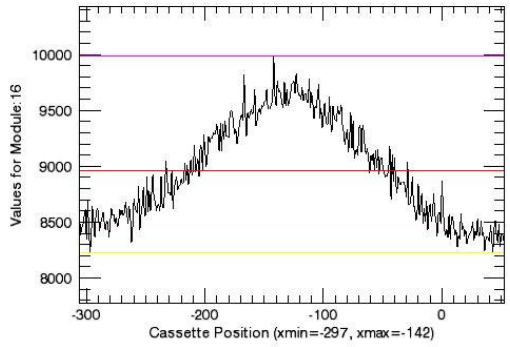
Graph 4.2:n



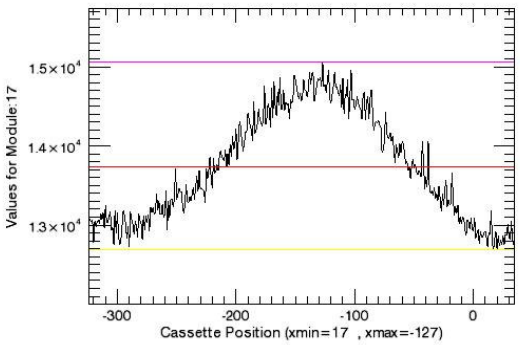
Graph 4.2:o



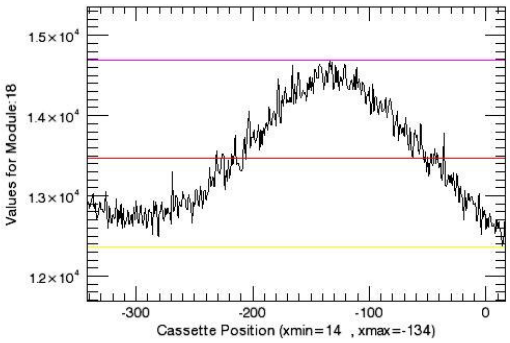
Graph 4.2:p



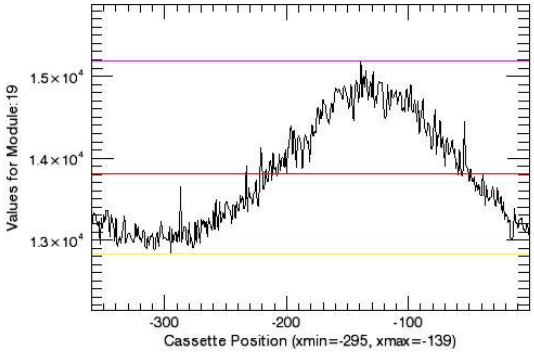
Graph 4.2:r



Graph 4.2:s



Graph 4.2:t



Graph 4.2:u

Graph 4.2: Cassette positions versus values for module groups for all ^{68}Ge data with gantry rotation

As the cassettes are rotated beginning values of each cassette are different. Angle values which correspond with minimum count values are listed in Output 4.1.

Output 4.1: Minimum angle values of the cassette positions for ^{68}Ge data with gantry rotation. It starts from 0 and ends with 19th value.

```
tempxcassette[minx](0):      64
tempxcassette[minx](1):      47
tempxcassette[minx](2):      60
tempxcassette[minx](3):      57
tempxcassette[minx](4):      59
tempxcassette[minx](5):      67
tempxcassette[minx](6):      50
tempxcassette[minx](7):      80
tempxcassette[minx](8):      23
tempxcassette[minx](9):      71
tempxcassette[minx](10):     61
tempxcassette[minx](11):     43
tempxcassette[minx](12):     38
tempxcassette[minx](13):     49
tempxcassette[minx](14):     77
tempxcassette[minx](15):     59
tempxcassette[minx](16):    -297
tempxcassette[minx](17):     17
tempxcassette[minx](18):     14
tempxcassette[minx](19):    -295
```

Some values are negative in Output 4.1. Negative values are summed with 360° to find real angle values (Output 4.2).

Output 4.2: Real minimum angle values of the cassettes for ^{68}Ge data with gantry rotation.

```
tempxcassette[minx](0):      64
tempxcassette[minx](1):      47
tempxcassette[minx](2):      60
tempxcassette[minx](3):      57
tempxcassette[minx](4):      59
tempxcassette[minx](5):      67
tempxcassette[minx](6):      50
tempxcassette[minx](7):      80
tempxcassette[minx](8):      23
tempxcassette[minx](9):      71
tempxcassette[minx](10):      61
tempxcassette[minx](11):      43
tempxcassette[minx](12):      38
tempxcassette[minx](13):      49
tempxcassette[minx](14):      77
tempxcassette[minx](15):      59
tempxcassette[minx](16):      63
tempxcassette[minx](17):      17
tempxcassette[minx](18):      14
tempxcassette[minx](19):      65
```

The averages and standard deviations of the values seen in Output 4.2 are given in Output 4.3.

Output 4.3: Average and standard deviation of minimum angle values for ^{68}Ge data with gantry rotation.

```
average of minxes:      53.2000
standard deviation of minex:      14.4600
```

Module groups take at 53°±14° minimum values.

Likewise the angle values for maximum counts of each module are given in Output 4.4.

Output 4.4: Maximum angle values of the cassette positions. It starts from 0 and ends with 19th value.

```
tempxcassette[maxx](0):      241
tempxcassette[maxx](1):      245
tempxcassette[maxx](2):      232
tempxcassette[maxx](3):      214
tempxcassette[maxx](4):      216
tempxcassette[maxx](5):      198
tempxcassette[maxx](6):      226
tempxcassette[maxx](7):      205
tempxcassette[maxx](8):     -120
tempxcassette[maxx](9):     -107
tempxcassette[maxx](10):      -125
tempxcassette[maxx](11):      -143
tempxcassette[maxx](12):      -107
tempxcassette[maxx](13):      -125
tempxcassette[maxx](14):      -131
tempxcassette[maxx](15):      -149
tempxcassette[maxx](16):      -142
tempxcassette[maxx](17):      -127
tempxcassette[maxx](18):      -134
tempxcassette[maxx](19):      -139
```

Real angle values are seen in Output 4.5.

Output 4.5: Real maximum angle values of the cassettes for ^{68}Ge data with gantry rotation.

```
tempxcassette[maxx](0):      241
tempxcassette[maxx](1):      245
tempxcassette[maxx](2):      232
tempxcassette[maxx](3):      214
tempxcassette[maxx](4):      216
tempxcassette[maxx](5):      198
tempxcassette[maxx](6):      226
tempxcassette[maxx](7):      205
tempxcassette[maxx](8):      240
tempxcassette[maxx](9):      253
tempxcassette[maxx](10):     235
tempxcassette[maxx](11):     217
tempxcassette[maxx](12):     253
tempxcassette[maxx](13):     235
tempxcassette[maxx](14):     229
tempxcassette[maxx](15):     211
tempxcassette[maxx](16):     218
tempxcassette[maxx](17):     233
tempxcassette[maxx](18):     226
tempxcassette[maxx](19):     221
```

Average and standard deviations are given in Output 4.6.

Output 4.6: Average and standard deviation of minimum angle values for ^{68}Ge data with gantry rotation.

```
average of maxxes:      227.400
standard deviation of maxxes:     12.2000
```

Module groups take at 227°±12° maximum values.

Average count values of each module group are given in Output 4.7.

Output 4.7: Average values of the module group counts for ^{68}Ge data with gantry rotation. It starts from 0 and ends with 19th value.

```
average of module (0) counts:     14198.9
average of module (1) counts:     15411.7
average of module (2) counts:     12821.7
average of module (3) counts:     14595.5
average of module (4) counts:     13720.3
average of module (5) counts:     13063.3
average of module (6) counts:     15508.5
average of module (7) counts:     14833.1
average of module (8) counts:     15544.1
average of module (9) counts:     13471.3
average of module (10) counts:    14711.7
average of module (11) counts:    13647.2
average of module (12) counts:    12442.8
average of module (13) counts:    12978.5
average of module (14) counts:    13350.4
average of module (15) counts:    14104.4
average of module (16) counts:    8957.64
average of module (17) counts:    13732.0
average of module (18) counts:    13467.9
average of module (19) counts:    13803.4
```

Averages of all modules and standard deviation are given in Output 4.9.

Output 4.8: Average and standard deviation of the module group mean values for ^{68}Ge data with gantry rotation.

```
average of all module means:      13718.2
standart deviation of all module means:      926.328
```

Average count value of all modules is 13718±926.

Minimum count values of each module groups are seen in Output 4.9.

Output 4.9: Minimum values of the module group counts for ^{68}Ge data with gantry rotation. It starts from 0 and ends with 19th value.

```
minimum of module (0) counts:     13104.0
minimum of module (1) counts:     14108.0
minimum of module (2) counts:     11806.0
minimum of module (3) counts:     13410.0
minimum of module (4) counts:     12602.0
minimum of module (5) counts:     11998.0
minimum of module (6) counts:     14254.0
minimum of module (7) counts:     13506.0
minimum of module (8) counts:     14102.0
minimum of module (9) counts:     12315.0
minimum of module (10) counts:    13638.0
minimum of module (11) counts:    12537.0
minimum of module (12) counts:    11506.0
minimum of module (13) counts:    11811.0
minimum of module (14) counts:    12269.0
minimum of module (15) counts:    12820.0
minimum of module (16) counts:    8222.00
minimum of module (17) counts:    12688.0
minimum of module (18) counts:    12362.0
minimum of module (19) counts:    12830.0
```

Averages of minimums and standard deviations of these averages are given in Output 4.10.

Output 4.10: Average and standard deviation of the module group minimum values for ^{68}Ge data with gantry rotation.

```
average of all module minimums:      12594.4
standart deviation of all module minimums:      852.360
```

Minimum count value of all modules is 12594±852.

Likewise maximum count values of each module groups are seen in Output 4.11.

Output 4.11: Maximum values of the module group counts for ^{68}Ge data with gantry rotation. It starts from 0 and ends with 19th value

```
maximum of module (0) counts:     15557.0
maximum of module (1) counts:     16957.0
maximum of module (2) counts:     14117.0
maximum of module (3) counts:     16295.0
maximum of module (4) counts:     15193.0
maximum of module (5) counts:     14446.0
maximum of module (6) counts:     17104.0
maximum of module (7) counts:     16251.0
maximum of module (8) counts:     17029.0
```

maximum of module (9) counts: 15042.0
maximum of module (10) counts: 16379.0
maximum of module (11) counts: 15117.0
maximum of module (12) counts: 13788.0
maximum of module (13) counts: 14356.0
maximum of module (14) counts: 15024.0
maximum of module (15) counts: 15677.0
maximum of module (16) counts: 9986.00
maximum of module (17) counts: 15056.0
maximum of module (18) counts: 14686.0
maximum of module (19) counts: 15191.0

Averages of maximums and standard deviations of these averages are given in Output 4.12.

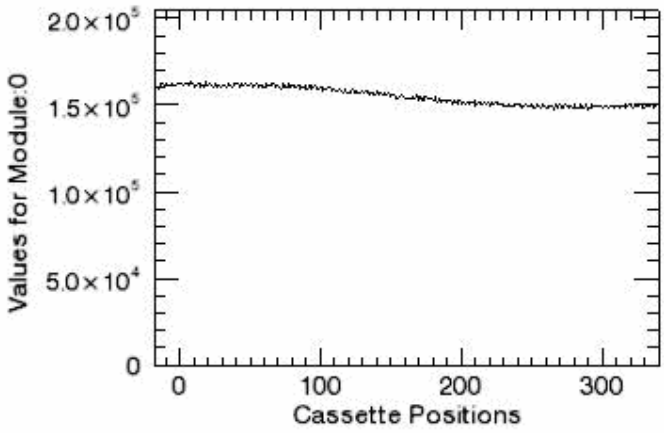
Output 4.12: Average and standard deviation of the module group maximum values for ^{68}Ge data with gantry rotation.

average of all module maximums: 15162.5
standart deviation of all module maximums: 1000.75

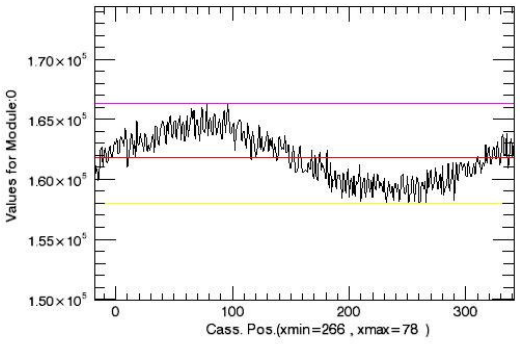
Maximum count value of all modules is 15162±1000

4.1.2 ^{18}F Data with Gantry Rotation

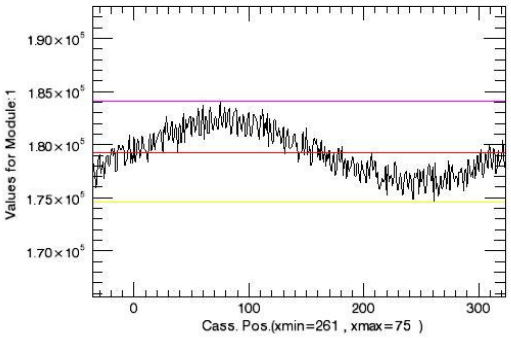
Cassette positions versus module groups are plotted (Graph 4.3 and Graph 4.4)



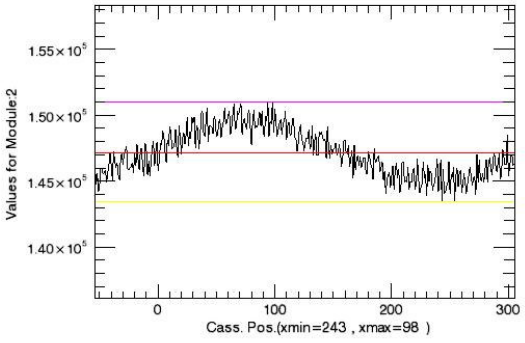
Graph 4.3: An example graph for cassette positions versus values for module groups for ^{18}F data with gantry rotation. Y values begin from zero.



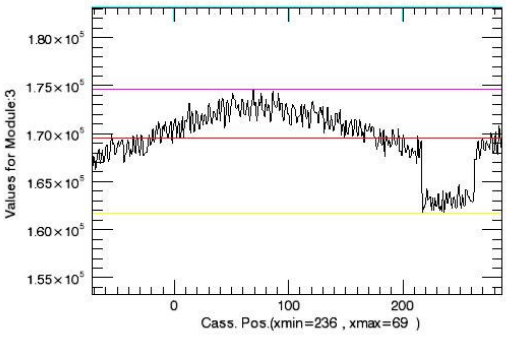
Graph 4.4:a



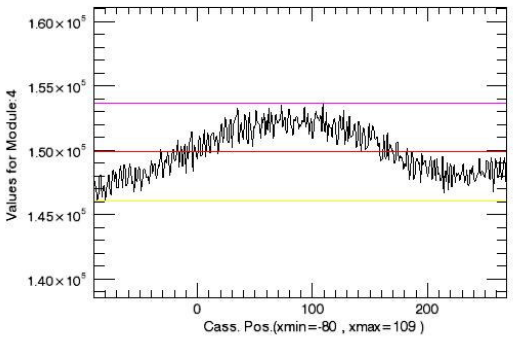
Graph 4.4:b



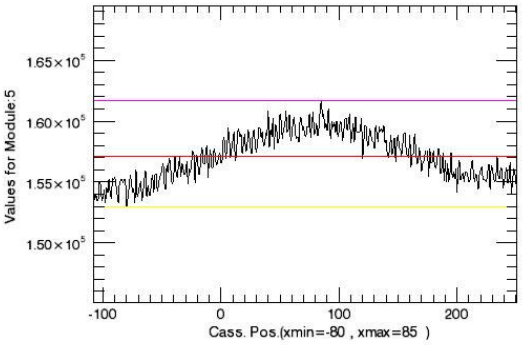
Graph 4.4:c



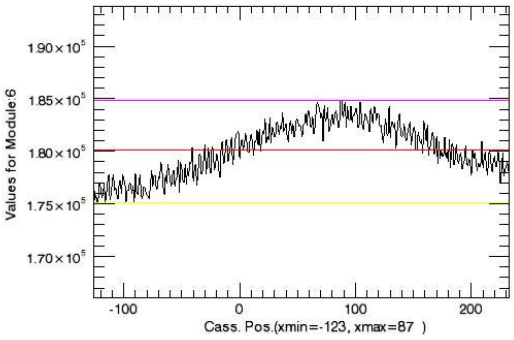
Graph 4.4:d



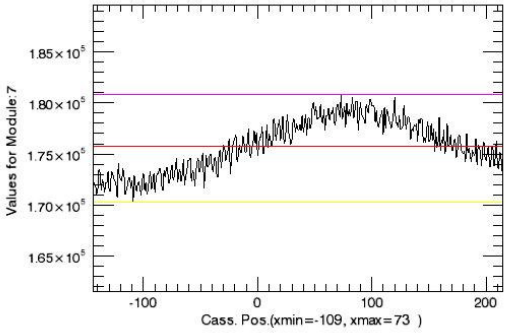
Graph 4.4:e



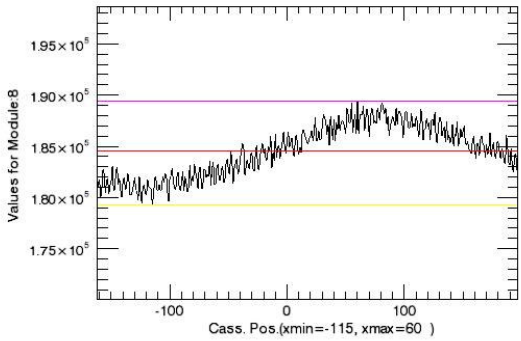
Graph 4.4:f



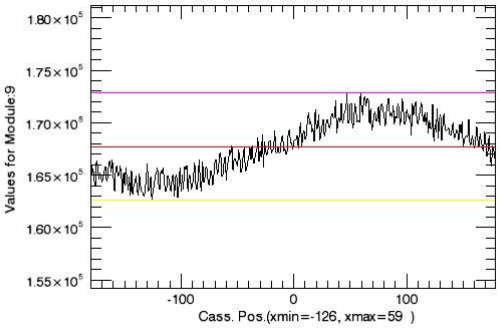
Graph 4.4:g



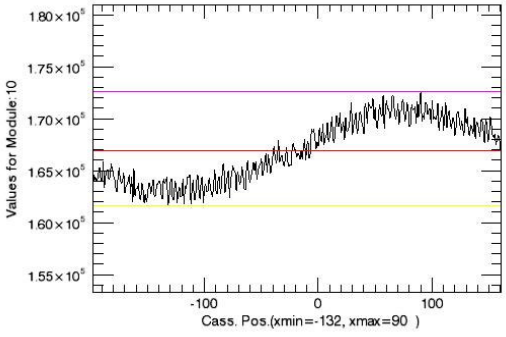
Graph 4.4:h



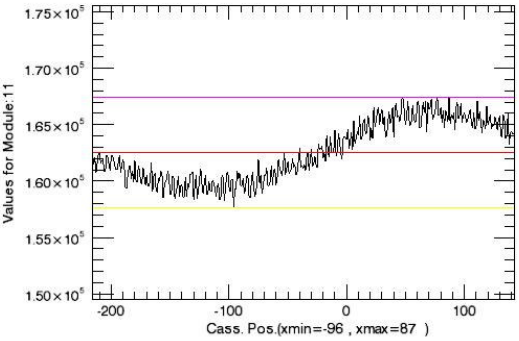
Graph 4.4:i



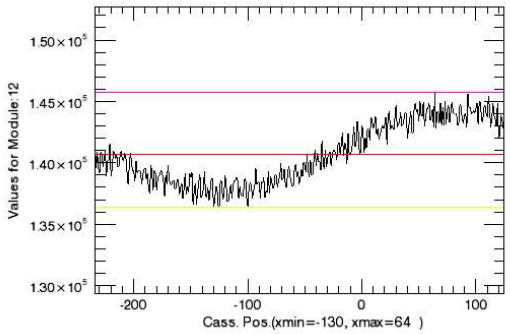
Graph 4.4:j



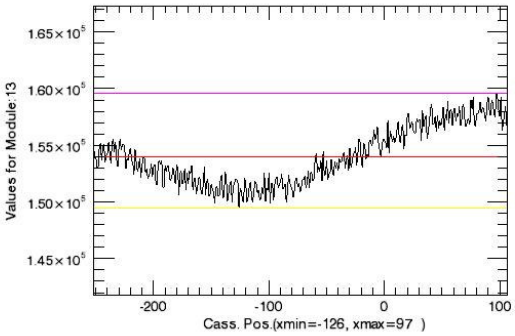
Graph 4.4:k



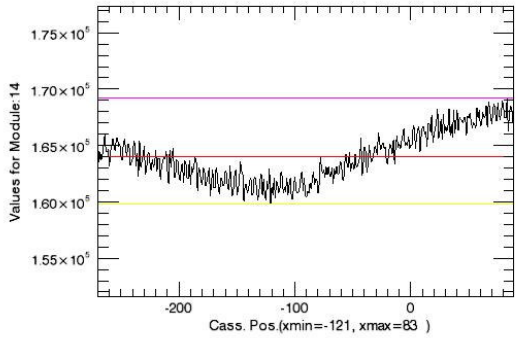
Graph 4.4:l



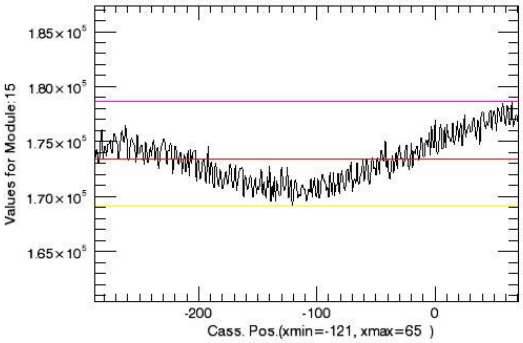
Graph 4.4:m



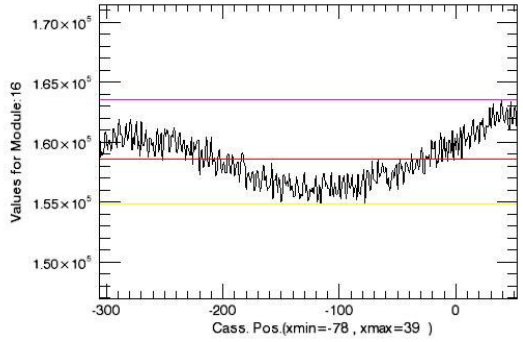
Graph 4.4:n



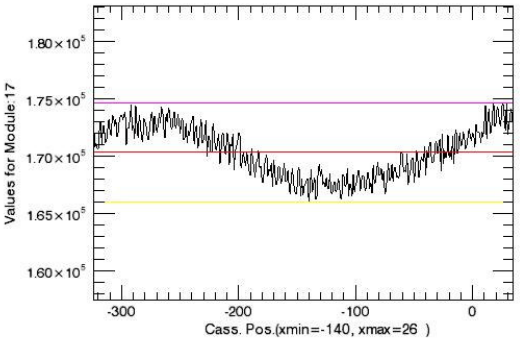
Graph 4.4:o



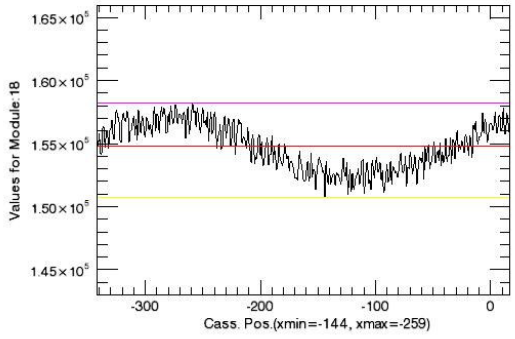
Graph 4.4:p



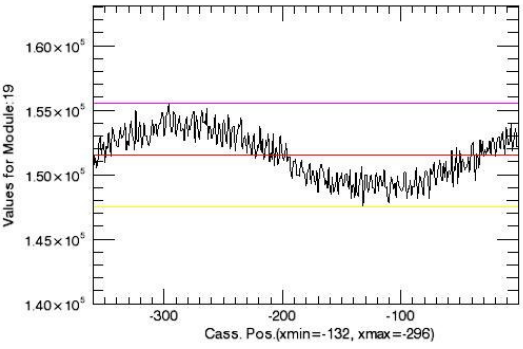
Graph 4.4:r



Graph 4.4:s



Graph 4.4:t



Graph 4.4:u

Graph 4.4: Cassette positions versus values for module groups for all ^{18}F data with gantry rotation

As seen in Graph 4.4:d there are measurement mistakes in Module Group 3 counts. Sinus wave is not continuous.

Output 4.13: Minimum angle values of the cassette positions for ^{18}F data with gantry rotation. It starts from 0 and ends with 19th value.

```
tempxcassette[minx](0):      266
tempxcassette[minx](1):      261
tempxcassette[minx](2):      243
tempxcassette[minx](3):      236
tempxcassette[minx](4):      -80
tempxcassette[minx](5):      -80
tempxcassette[minx](6):     -123
tempxcassette[minx](7):     -109
tempxcassette[minx](8):     -115
tempxcassette[minx](9):     -126
tempxcassette[minx](10):    -132
tempxcassette[minx](11):     -96
tempxcassette[minx](12):    -130
tempxcassette[minx](13):    -126
tempxcassette[minx](14):    -121
tempxcassette[minx](15):    -121
tempxcassette[minx](16):     -78
tempxcassette[minx](17):    -140
tempxcassette[minx](18):    -144
tempxcassette[minx](19):    -132
```

Output 4.14: Real minimum angle values of the cassettes for ^{18}F data with gantry rotation

```
tempxcassette[minx](0):      266
tempxcassette[minx](1):      261
tempxcassette[minx](2):      243
tempxcassette[minx](3):      236
tempxcassette[minx](4):      280
tempxcassette[minx](5):      280
tempxcassette[minx](6):      237
tempxcassette[minx](7):      251
tempxcassette[minx](8):      245
tempxcassette[minx](9):      234
tempxcassette[minx](10):     228
tempxcassette[minx](11):     264
tempxcassette[minx](12):     230
tempxcassette[minx](13):     234
tempxcassette[minx](14):     239
tempxcassette[minx](15):     239
tempxcassette[minx](16):     282
tempxcassette[minx](17):     220
tempxcassette[minx](18):     216
tempxcassette[minx](19):     228
```

Output 4.15: Average and standard deviation of minimum angle values for ^{18}F data with gantry rotation.

```
average of minxes:      245.650
standart deviation of minex:    16.4450
```

All module groups take at 246°±16° minimum values.

Output 4.16: Maximum angle values of the cassette positions for ^{18}F data with gantry rotation. It starts from 0 and ends with 19th value.

```
tempxcassette[maxx](0):      78
tempxcassette[maxx](1):      75
tempxcassette[maxx](2):      98
tempxcassette[maxx](3):      69
tempxcassette[maxx](4):      109
```

tempxcassette[maxx](5): 85
tempxcassette[maxx](6): 87
tempxcassette[maxx](7): 73
tempxcassette[maxx](8): 60
tempxcassette[maxx](9): 59
tempxcassette[maxx](10): 90
tempxcassette[maxx](11): 87
tempxcassette[maxx](12): 64
tempxcassette[maxx](13): 97
tempxcassette[maxx](14): 83
tempxcassette[maxx](15): 65
tempxcassette[maxx](16): 39
tempxcassette[maxx](17): 26
tempxcassette[maxx](18): -259
tempxcassette[maxx](19): -296

Output 4.17: Real maximum angle values of the cassettes for ^{18}F data with gantry rotation.

tempxcassette[maxx](0): 78
tempxcassette[maxx](1): 75
tempxcassette[maxx](2): 98
tempxcassette[maxx](3): 69
tempxcassette[maxx](4): 109
tempxcassette[maxx](5): 85
tempxcassette[maxx](6): 87
tempxcassette[maxx](7): 73
tempxcassette[maxx](8): 60
tempxcassette[maxx](9): 59
tempxcassette[maxx](10): 90
tempxcassette[maxx](11): 87
tempxcassette[maxx](12): 64
tempxcassette[maxx](13): 97
tempxcassette[maxx](14): 83
tempxcassette[maxx](15): 65
tempxcassette[maxx](16): 39
tempxcassette[maxx](17): 26
tempxcassette[maxx](18): 101
tempxcassette[maxx](19): 64

Output 4.18: Average and standard deviation of maximum angle values for ^{18}F data with gantry rotation.

average of maxxes: 75.4500
standart deviation of maxxes: 16.0500

All module groups take at 75°±16° maximum values.

Output 4.19: Average values of the module group counts for ^{18}F data with gantry rotation. It starts from 0 and ends with 19th value.

average of module (0) counts: 161813.
average of module (1) counts: 179253.
average of module (2) counts: 147127.
average of module (3) counts: 169547.
average of module (4) counts: 149934.
average of module (5) counts: 157102.
average of module (6) counts: 180132.
average of module (7) counts: 175788.
average of module (8) counts: 184531.
average of module (9) counts: 167712.
average of module (10) counts: 166889.
average of module (11) counts: 162573.
average of module (12) counts: 140683.
average of module (13) counts: 154007.
average of module (14) counts: 163987.
average of module (15) counts: 173413.
average of module (16) counts: 158614.
average of module (17) counts: 170402.
average of module (18) counts: 154775.
average of module (19) counts: 151499.

Output 4.20: Average and standard deviation of the module group mean values for ^{18}F data with gantry rotation.

average of all module means: 163489.
standart deviation of all module means: 9676.30

Average count value of all modules is 163489±9676.

Output 4.21: Minimum values of the module group counts for ^{18}F data with gantry rotation. It starts from 0 and ends with 19th value.

minimum of module (0) counts: 158011.
minimum of module (1) counts: 174637.
minimum of module (2) counts: 143463.
minimum of module (3) counts: 161727.
minimum of module (4) counts: 146046.
minimum of module (5) counts: 152947.
minimum of module (6) counts: 175077.
minimum of module (7) counts: 170315.
minimum of module (8) counts: 179277.
minimum of module (9) counts: 162595.
minimum of module (10) counts: 161633.
minimum of module (11) counts: 157645.
minimum of module (12) counts: 136380.
minimum of module (13) counts: 149477.
minimum of module (14) counts: 159788.
minimum of module (15) counts: 169139.
minimum of module (16) counts: 154870.
minimum of module (17) counts: 165970.
minimum of module (18) counts: 150737.
minimum of module (19) counts: 147533.

Output 4.22: Average and standard deviation of the module group minimum values for ^{18}F data with gantry rotation.

average of all module minimums: 158863.
standart deviation of all module minimums: 9152.44

Minimum count value of all modules is 158863±9152.

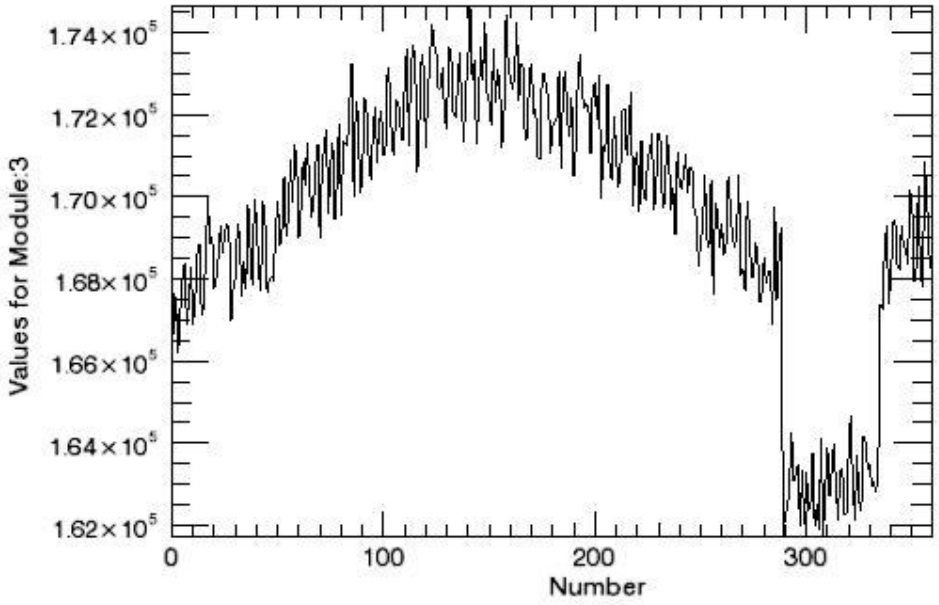
Output 4.23: Maximum values of the module group counts for ^{18}F data with gantry rotation. It starts from 0 and ends with 19th value

maximum of module (0) counts: 166344.
maximum of module (1) counts: 184084.
maximum of module (2) counts: 151014.
maximum of module (3) counts: 174661.
maximum of module (4) counts: 153649.
maximum of module (5) counts: 161661.
maximum of module (6) counts: 184827.
maximum of module (7) counts: 180848.
maximum of module (8) counts: 189461.
maximum of module (9) counts: 172835.
maximum of module (10) counts: 172644.
maximum of module (11) counts: 167467.
maximum of module (12) counts: 145702.
maximum of module (13) counts: 159616.
maximum of module (14) counts: 169193.
maximum of module (15) counts: 178710.
maximum of module (16) counts: 163577.
maximum of module (17) counts: 174645.
maximum of module (18) counts: 158246.
maximum of module (19) counts: 155534.

Output 4.24: Average and standard deviation of the module group maximum values for ^{18}F data with gantry rotation.

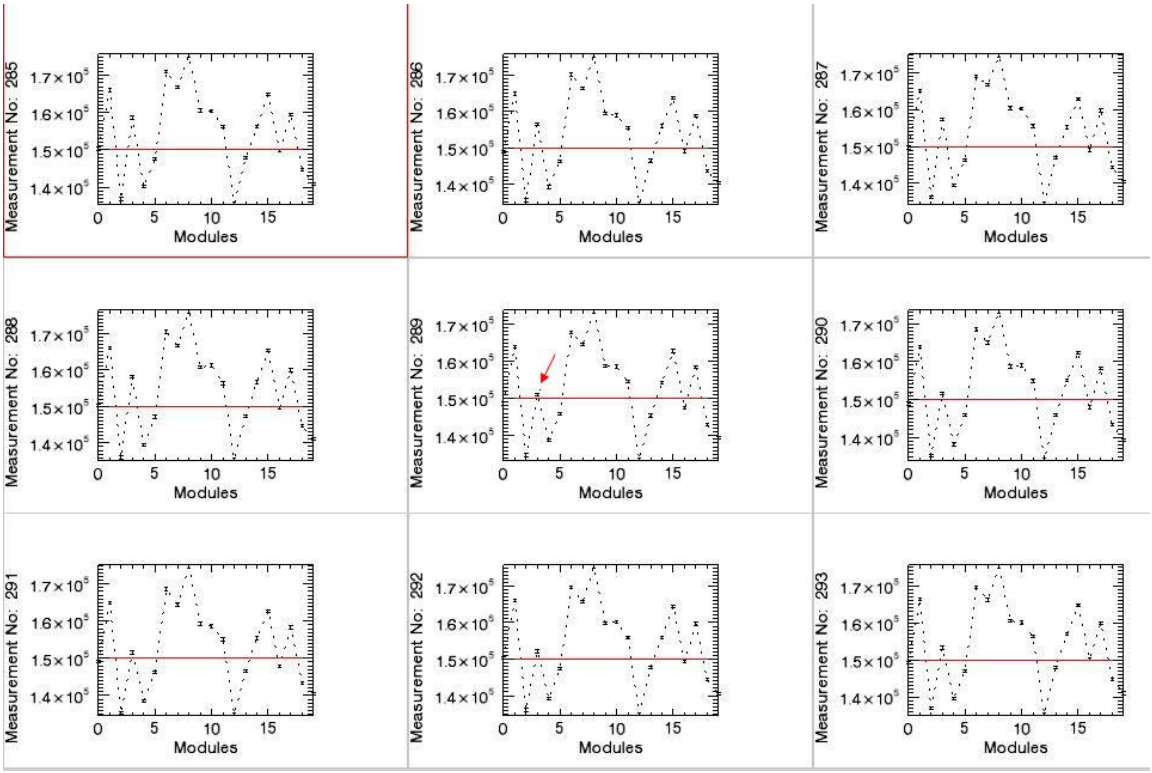
average of all module maximums: 168236.
standart deviation of all module maximums: 9954.133

Maximum count value of all modules is 168236±9954.

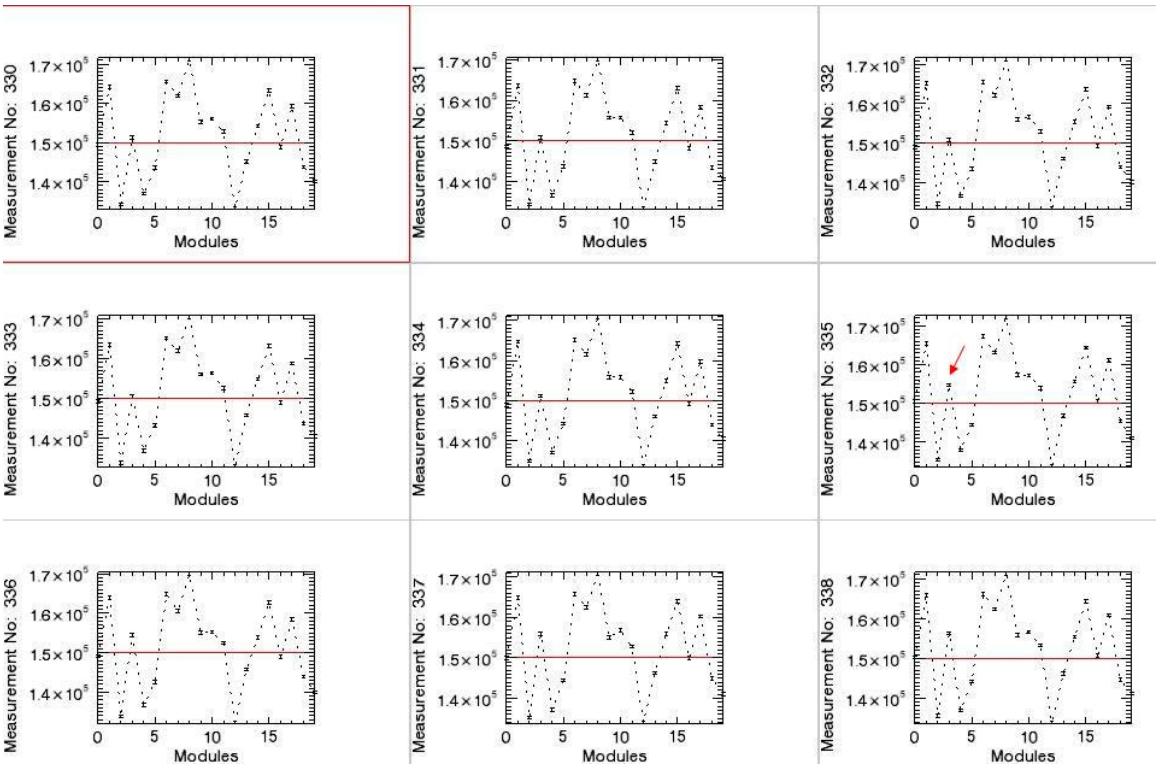


Graph 4.5: Enlarged graph for values of Module Group 3 versus measurement number.

As mentioned before there are mistakes in Module Group 3 counts seen in Graph 4.5 too. The error begins approximately at 290th measurement and continues up to 335th measurement. It is possible to find exact values where the error starts and finishes. We can give an input measurement number to the program and the program can find out graphs for the relevant interval likewise in Graph 3.11. For fixing the beginning point we can give measurement number between 285 and 293. As seen in Graph 4.6 the third module counts begin to drop at 289th measurement. If we apply same method for finding finishing point we can choose interval between 330 and 338 (Graph 4.7). Module Group 3 counts begin to increase at 335th measurement again.



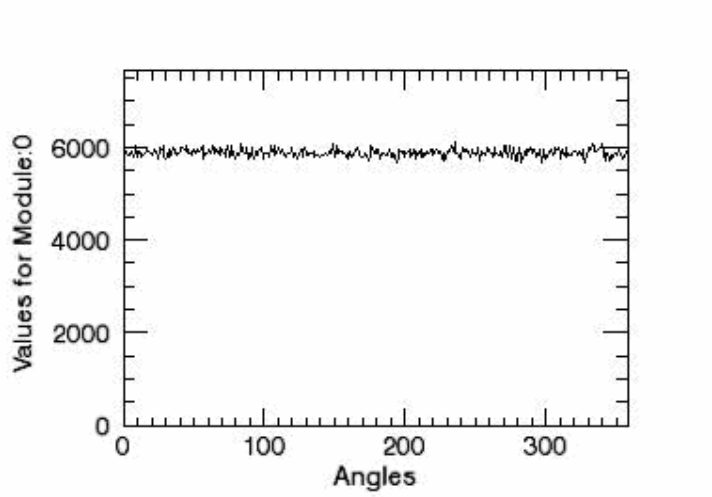
Graph 4.6: Input interval is given between 285 and 293. Third module values begin to decrease at 289th measurement



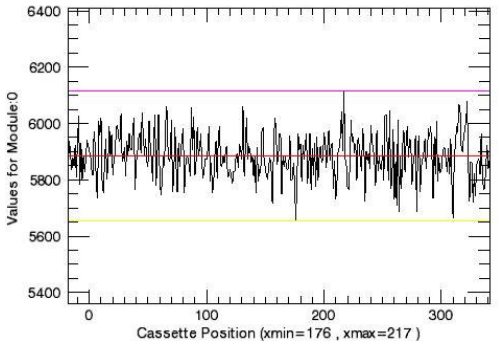
Graph 4.7: Input interval is given between 330 and 338. Third module values begin to increase at 335th measurement

4.1.3 Blank Scan Data with Gantry Rotation

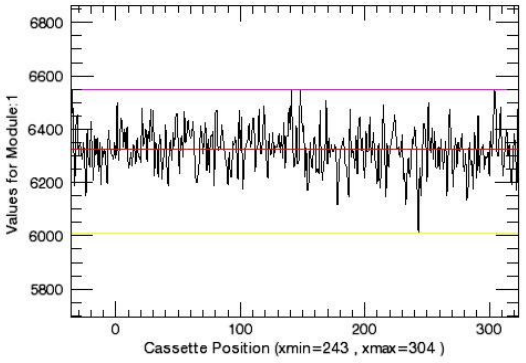
Cassette positions versus module groups are plotted (Graph 4.8 and Graph 4.9)



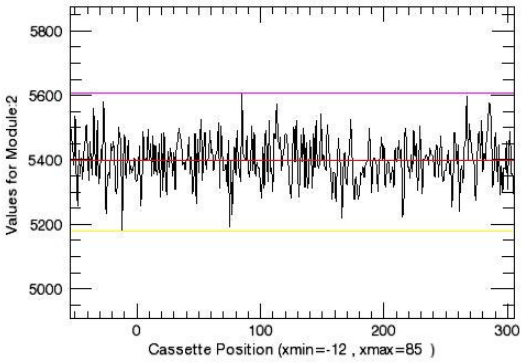
Graph 4.8: An example graph for cassette positions versus values for module groups for Blank Scan Data with Gantry Rotation. Y values begin from zero.



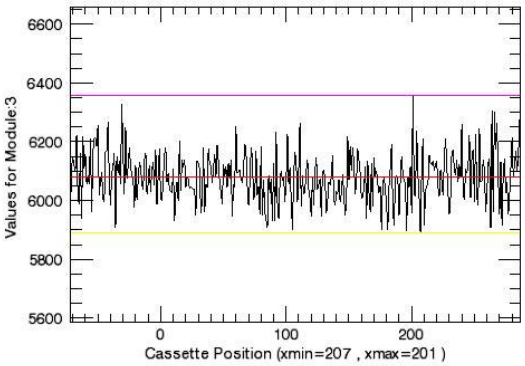
Graph 4.9:a



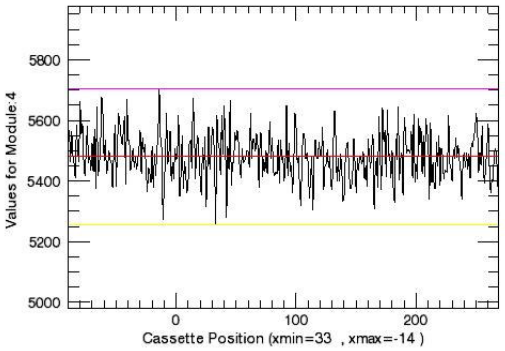
Graph 4.9:b



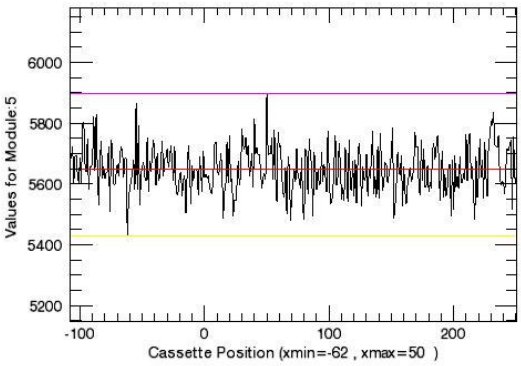
Graph 4.9:c



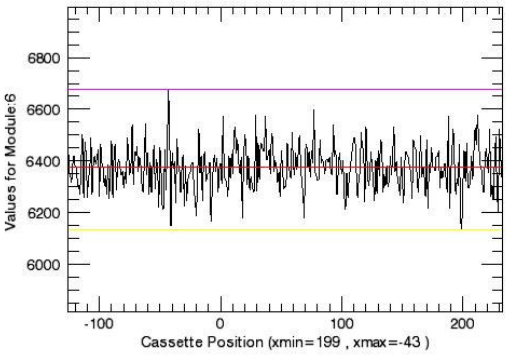
Graph 4.9:d



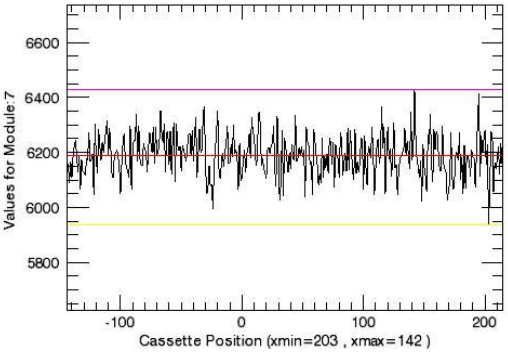
Graph 4.9:e



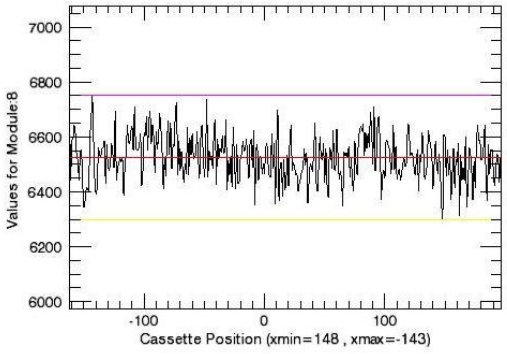
Graph 4.9:f



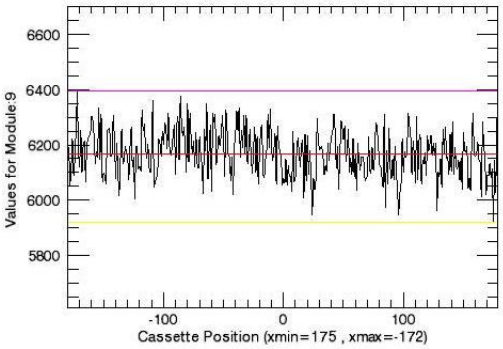
Graph 4.9:g



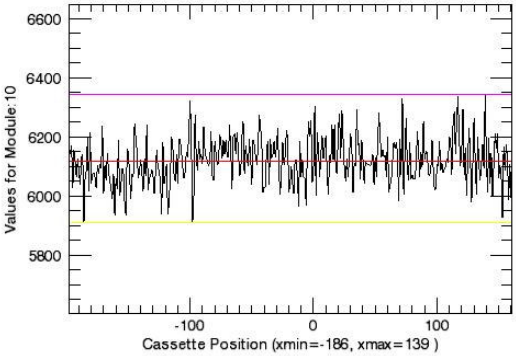
Graph 4.9:h



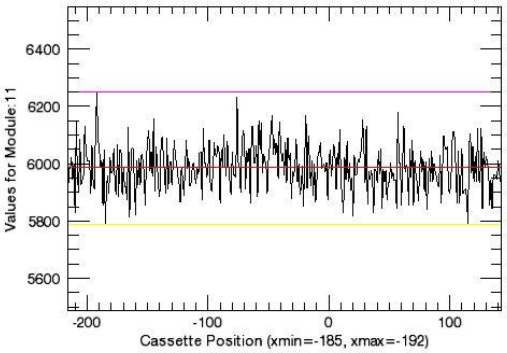
Graph 4.9:i



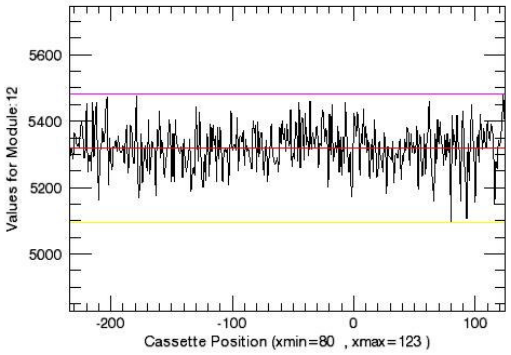
Graph 4.9:j



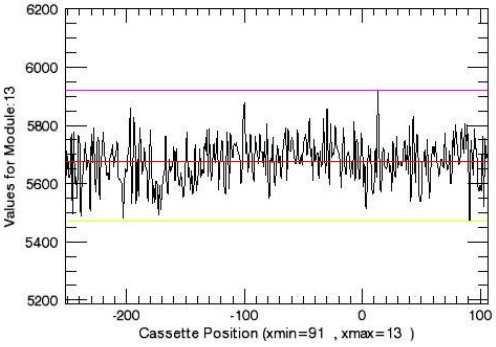
Graph 4.9:k



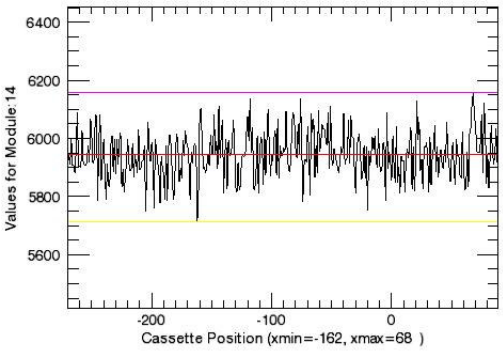
Graph 4.9:l



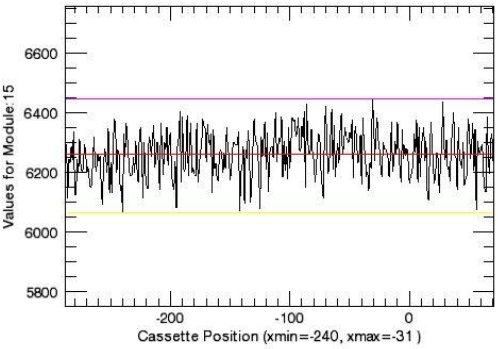
Graph 4.9:m



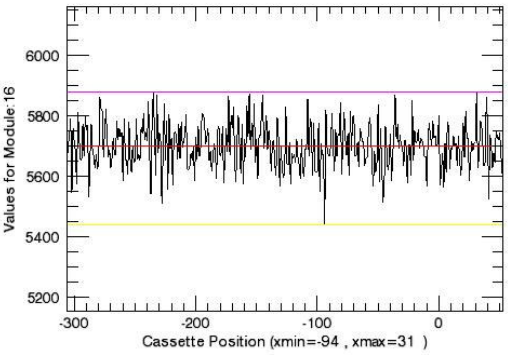
Graph 4.9:n



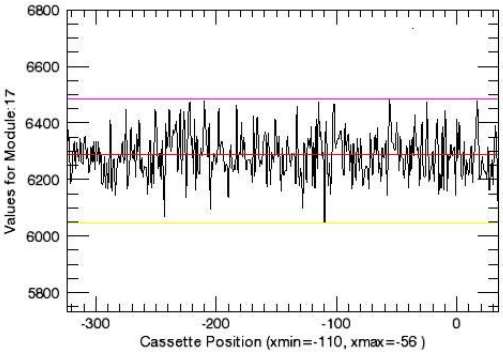
Graph 4.9:o



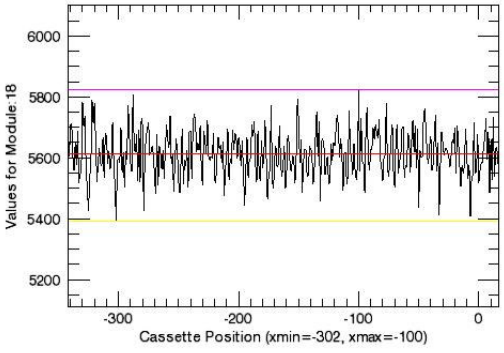
Graph 4.9:p



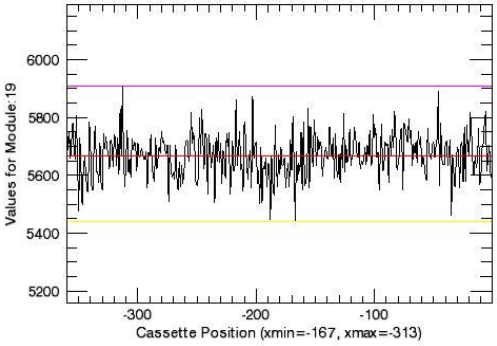
Graph 4.9:r



Graph 4.9:s



Graph 4.9:t



Graph 4.9:u

Graph 4.9: Cassette positions versus values for module groups for all Blank Scan Data with Gantry Rotation

As seen in Graph 4.8 and Graph 4.9 black scan graphs are noisy and their shapes don't look like sinus waves like ^{18}F and ^{68}Ge plots.

Output 4.25:Minimum angle values of the cassette positions for Blank Scan data with gantry rotation. It starts from 0 and ends with 19th value.

tempxcassette[minx](0): 176
tempxcassette[minx](1): 243
tempxcassette[minx](2): -12
tempxcassette[minx](3): 207
tempxcassette[minx](4): 33
tempxcassette[minx](5): -62
tempxcassette[minx](6): 199
tempxcassette[minx](7): 203
tempxcassette[minx](8): 148
tempxcassette[minx](9): 175
tempxcassette[minx](10): -186
tempxcassette[minx](11): -185
tempxcassette[minx](12): 80
tempxcassette[minx](13): 91
tempxcassette[minx](14): -162
tempxcassette[minx](15): -240
tempxcassette[minx](16): -94
tempxcassette[minx](17): -110
tempxcassette[minx](18): -302
tempxcassette[minx](19): -167

Output 4.26: Real minimum angle values of the cassettes for Blank Scan data with gantry rotation

tempxcassette[minx](0): 176
tempxcassette[minx](1): 243
tempxcassette[minx](2): 348
tempxcassette[minx](3): 207
tempxcassette[minx](4): 33
tempxcassette[minx](5): 298
tempxcassette[minx](6): 199
tempxcassette[minx](7): 203
tempxcassette[minx](8): 148
tempxcassette[minx](9): 175
tempxcassette[minx](10): 174
tempxcassette[minx](11): 175
tempxcassette[minx](12): 80
tempxcassette[minx](13): 91
tempxcassette[minx](14): 198
tempxcassette[minx](15): 120
tempxcassette[minx](16): 266
tempxcassette[minx](17): 250
tempxcassette[minx](18): 58
tempxcassette[minx](19): 193

Output 4.27: Average and standard deviation of minimum angle values for Blank Scan data with gantry rotation.

average of minxes: 181.750
standart deviation of minex: 58.7500

All module groups take at 182°±59° minimum values.

Output 4.28: Maximum angle values of the cassette positions for Blank Scan data with gantry rotation. It starts from 0 and ends with 19th value.

tempxcassette[maxx](0): 217
tempxcassette[maxx](1): 304
tempxcassette[maxx](2): 85
tempxcassette[maxx](3): 201

tempxcassette[maxx](4): -14
tempxcassette[maxx](5): 50
tempxcassette[maxx](6): -43
tempxcassette[maxx](7): 142
tempxcassette[maxx](8): -143
tempxcassette[maxx](9): -172
tempxcassette[maxx](10): 139
tempxcassette[maxx](11): -192
tempxcassette[maxx](12): 123
tempxcassette[maxx](13): 13
tempxcassette[maxx](14): 68
tempxcassette[maxx](15): -31
tempxcassette[maxx](16): 31
tempxcassette[maxx](17): -56
tempxcassette[maxx](18): -100
tempxcassette[maxx](19): -313

Output 4.29: Real maximum angle values of the cassettes for Blank Scan data with gantry rotation.

tempxcassette[maxx](0): 217
tempxcassette[maxx](1): 304
tempxcassette[maxx](2): 85
tempxcassette[maxx](3): 201
tempxcassette[maxx](4): 346
tempxcassette[maxx](5): 50
tempxcassette[maxx](6): 317
tempxcassette[maxx](7): 142
tempxcassette[maxx](8): 217
tempxcassette[maxx](9): 188
tempxcassette[maxx](10): 139
tempxcassette[maxx](11): 168
tempxcassette[maxx](12): 123
tempxcassette[maxx](13): 13
tempxcassette[maxx](14): 68
tempxcassette[maxx](15): 329
tempxcassette[maxx](16): 31
tempxcassette[maxx](17): 304
tempxcassette[maxx](18): 260
tempxcassette[maxx](19): 47

Output 4.30: Average and standard deviation of maximum angle values for ^{18}F data with gantry rotation.

average of maxxes: 177.450
standart deviation of maxxes: 90.8500

All module groups take at 177°±91° maximum values.

Output 4.31: Average values of the module group counts for Blank Scan data with gantry rotation. It starts from 0 and ends with 19th value.

average of module (0) counts: 5883.74
average of module (1) counts: 6324.31
average of module (2) counts: 5398.76
average of module (3) counts: 6078.69
average of module (4) counts: 5481.45
average of module (5) counts: 5647.99
average of module (6) counts: 6374.13
average of module (7) counts: 6190.20
average of module (8) counts: 6524.139
average of module (9) counts: 6168.80
average of module (10) counts: 6119.10
average of module (11) counts: 5987.17
average of module (12) counts: 5320.57
average of module (13) counts: 5677.01
average of module (14) counts: 5944.47
average of module (15) counts: 6261.34
average of module (16) counts: 5700.79
average of module (17) counts: 6286.47
average of module (18) counts: 5613.58

average of module (19) counts: 5668.60

Output 4.32: Average and standard deviation of the module group mean values for Blank Scan data with gantry rotation.

average of all module means: 5932.61
standart deviation of all module means: 300.098

Average count value of all modules is 5933±300.

Output 4.33: Minimum values of the module group counts for Blank Scan data with gantry rotation. It starts from 0 and ends with 19th value.

minimum of module (0) counts: 5654.00
minimum of module (1) counts: 6011.00
minimum of module (2) counts: 5178.00
minimum of module (3) counts: 5890.00
minimum of module (4) counts: 5254.00
minimum of module (5) counts: 5430.00
minimum of module (6) counts: 6135.00
minimum of module (7) counts: 5935.00
minimum of module (8) counts: 6300.00
minimum of module (9) counts: 5918.00
minimum of module (10) counts: 5909.00
minimum of module (11) counts: 5787.00
minimum of module (12) counts: 5094.00
minimum of module (13) counts: 5471.00
minimum of module (14) counts: 5713.00
minimum of module (15) counts: 6065.00
minimum of module (16) counts: 5439.00
minimum of module (17) counts: 6047.00
minimum of module (18) counts: 5392.00
minimum of module (19) counts: 5442.00

Output 4.34: Average and standard deviation of the module group minimum values for Blank Scan data with gantry rotation.

average of all module minimums: 5703.20
standart deviation of all module minimums: 297.480

Minimum count value of all modules is 5703±297.

Output 4.35: Maximum values of the module group counts for Blank Scan data with gantry rotation. It starts from 0 and ends with 19th value

maximum of module (0) counts: 6117.00
maximum of module (1) counts: 6549.00
maximum of module (2) counts: 5606.00
maximum of module (3) counts: 6356.00
maximum of module (4) counts: 5704.00
maximum of module (5) counts: 5899.00
maximum of module (6) counts: 6678.00
maximum of module (7) counts: 6429.00
maximum of module (8) counts: 6752.00
maximum of module (9) counts: 6395.00
maximum of module (10) counts: 6344.00
maximum of module (11) counts: 6250.00
maximum of module (12) counts: 5481.00
maximum of module (13) counts: 5919.00
maximum of module (14) counts: 6157.00
maximum of module (15) counts: 6446.00
maximum of module (16) counts: 5878.00
maximum of module (17) counts: 6486.00
maximum of module (18) counts: 5822.00
maximum of module (19) counts: 5908.00

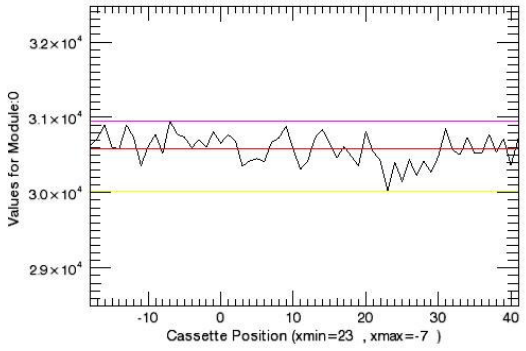
Output 4.36: Average and standard deviation of the module group maximum values for Blank Scan data with gantry rotation

average of all module maximums: 6158.80
standart deviation of all module maximums: 309.700

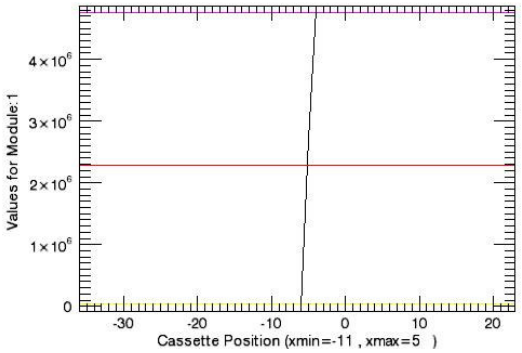
Maximum count value of all modules is 6158±310.

4.1.4 Inaccurate Blank Scan Data with Gantry Rotation

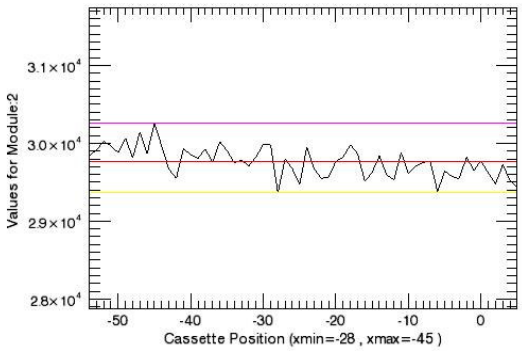
Cassette positions versus module groups are plotted (Graph 4.10)



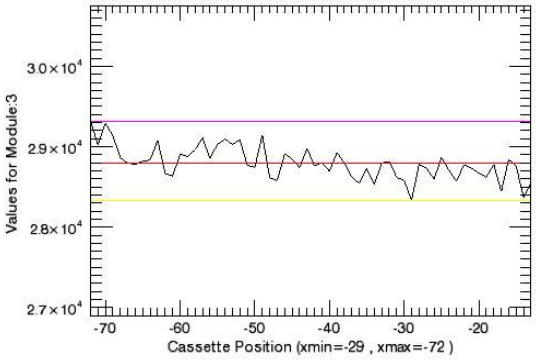
Graph 4.10:a



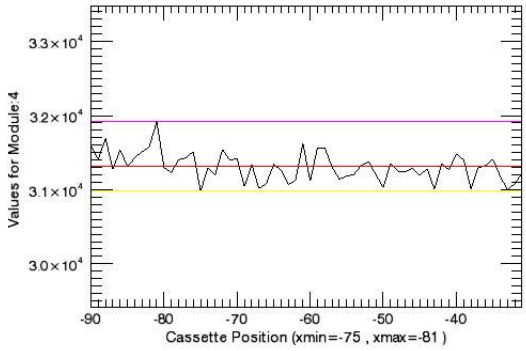
Graph 4.10:b



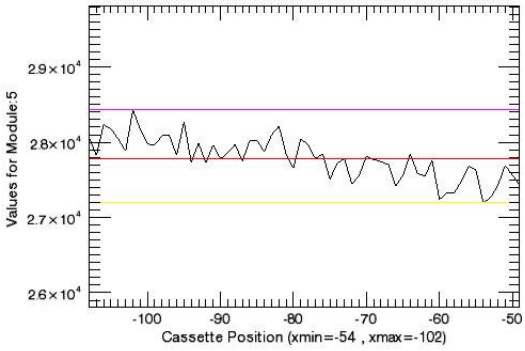
Graph 4.10:c



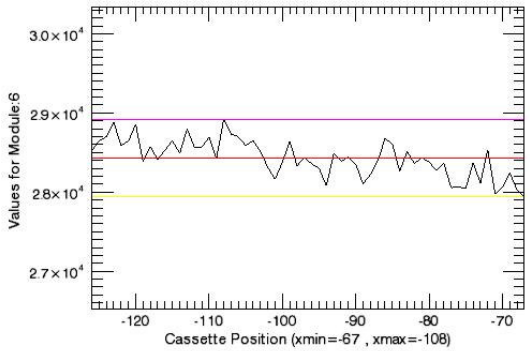
Graph 4.10:d



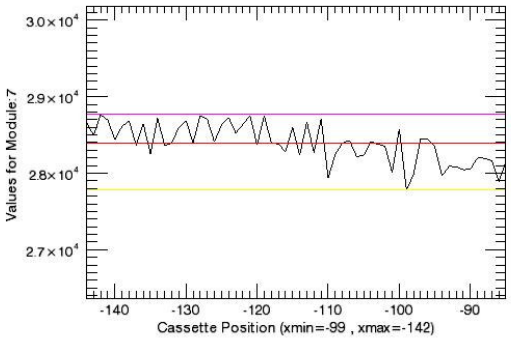
Graph 4.10:e



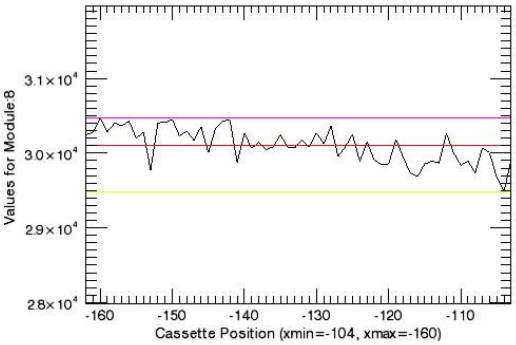
Graph 4.10:f



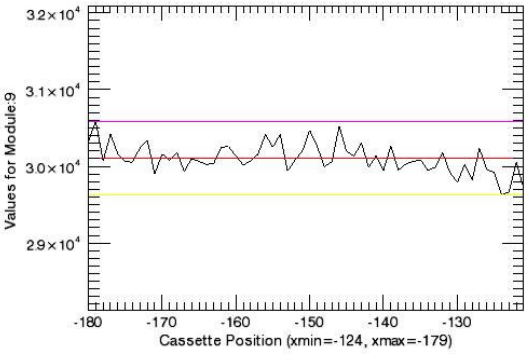
Graph 4.10:g



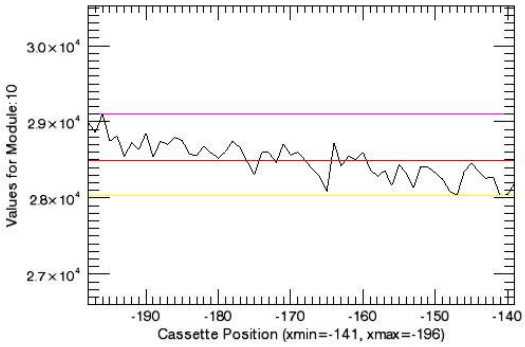
Graph 4.10:h



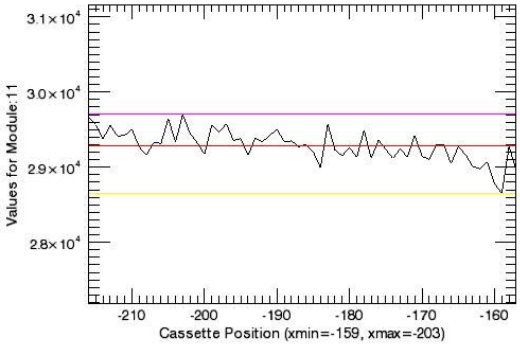
Graph 4.10:i



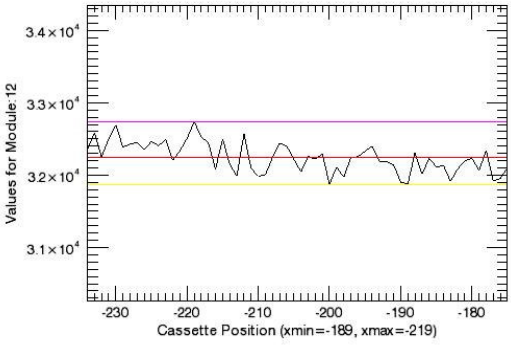
Graph 4.10:j



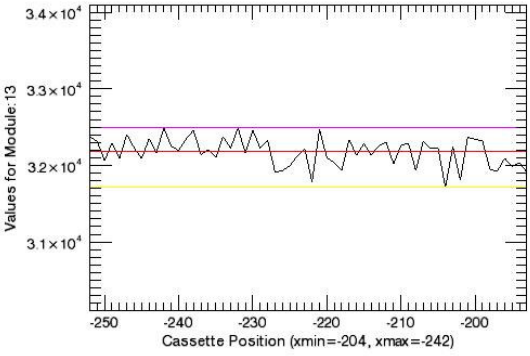
Graph 4.10:k



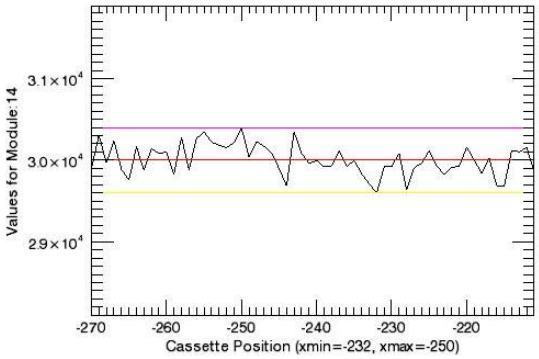
Graph 4.10:l



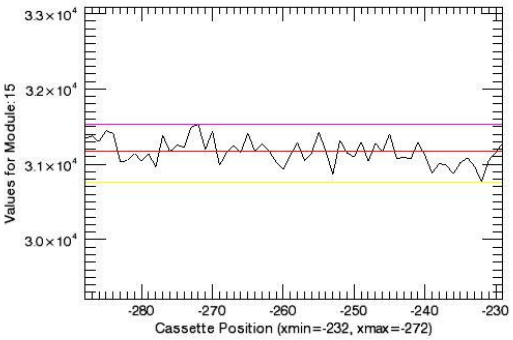
Graph 4.10:m



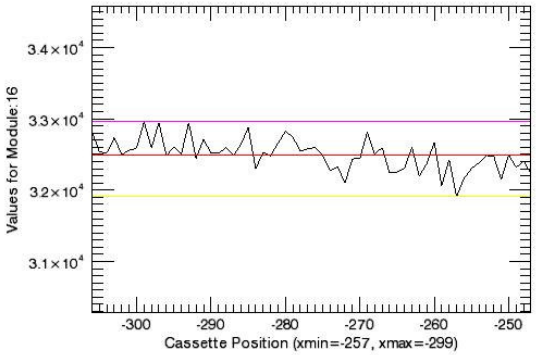
Graph 4.10:n



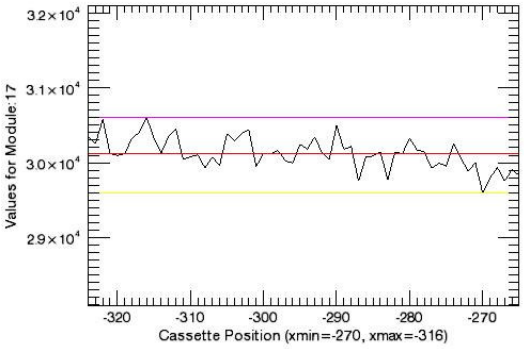
Graph 4.10:o



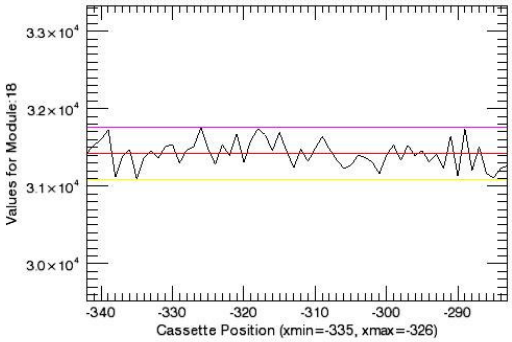
Graph 4.10:p



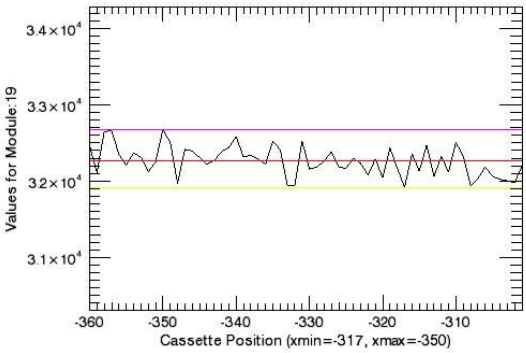
Graph 4.10:r



Graph 4.10:s



Graph 4.10:t



Graph 4.10:u

Graph 4.10: Cassette positions versus values for module groups for all Inaccurate Blank Scan Data with Gantry Rotation

As it mentioned in Section 3.4.2 measurements taken from 19th Cassette (Module Group 1) are inaccurate in this blank scan file. It is clear from Graph 4.10:b too.

Output 4.37: Average values of the module group counts for Inaccurate Blank Scan data with gantry rotation. It starts from 0 and ends with 19th value.

average of module (0) counts: 15291.3
average of module (1) counts: 1.14016e+006
average of module (2) counts: 14880.6
average of module (3) counts: 14399.2
average of module (4) counts: 15654.13

```
average of module (5) counts:     13893.5
average of module (6) counts:     14214.5
average of module (7) counts:     14196.9
average of module (8) counts:     15051.9
average of module (9) counts:     15051.7
average of module (10) counts:    14247.5
average of module (11) counts:    14645.0
average of module (12) counts:    16122.6
average of module (13) counts:    16089.4
average of module (14) counts:    15004.0
average of module (15) counts:    15584.0
average of module (16) counts:    16248.7
average of module (17) counts:    15063.8
average of module (18) counts:    15710.6
average of module (19) counts:    16130.4
```

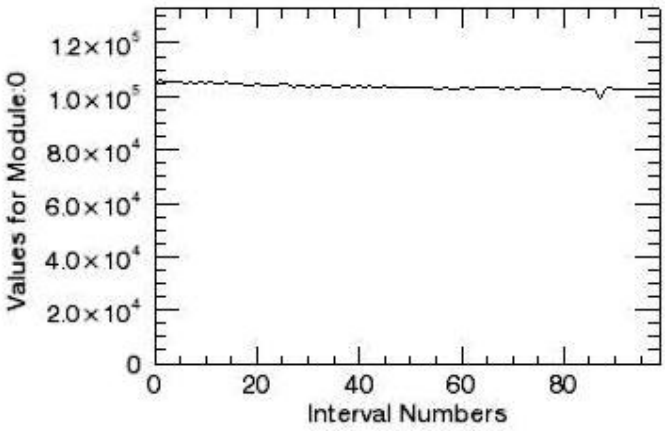
When average counts of the modules are printed like Output 4.37, it is appear that Module 1 counts are very high from other module counts. So the Module 1 counts increase average counts of all module means (Output 4.38)

Output 4.38: Average of the module group mean values for Inaccurate Blank Scan data with gantry rotation.

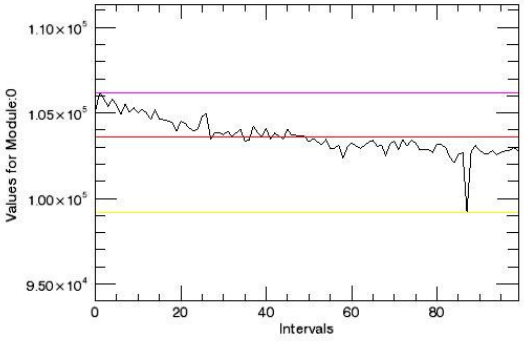
```
average of all module means:     71382.0
```

4.1.5 ^{68}Ge Data without Gantry Rotation

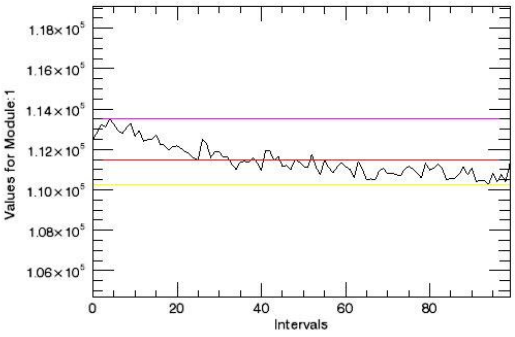
Interval numbers versus module groups are plotted (Graph 4.11 and Graph 4.12)



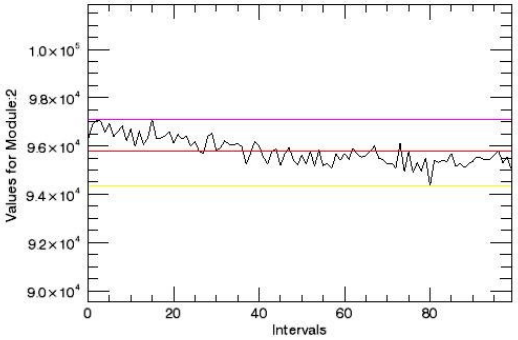
Graph 4.11: An example graph for interval numbers versus values for module groups for ^{68}Ge Data without Gantry Rotation. Y values begin from zero.



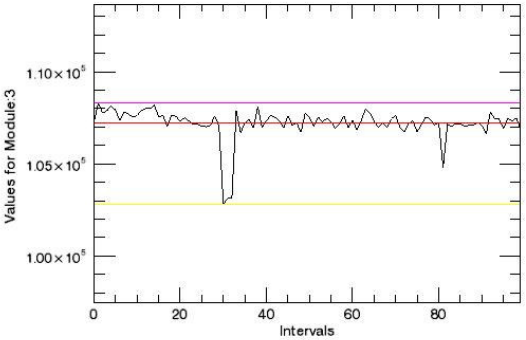
Graph 4.12:a



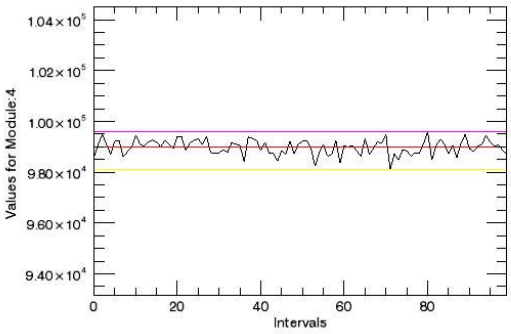
Graph 4.12:b



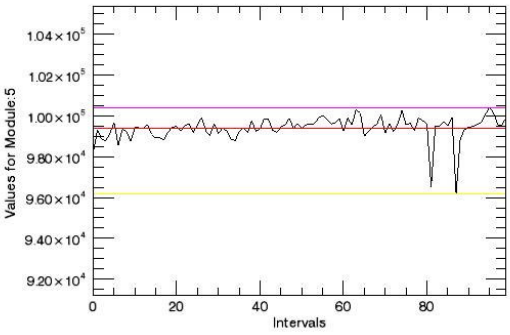
Graph 4.12:c



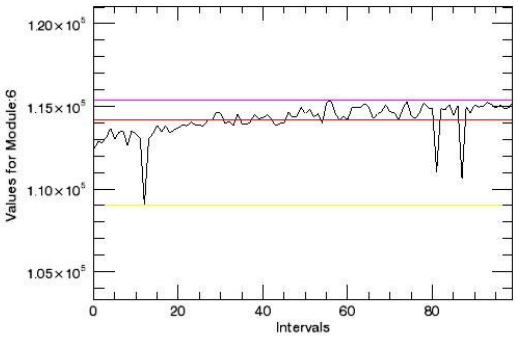
Graph 4.12:d



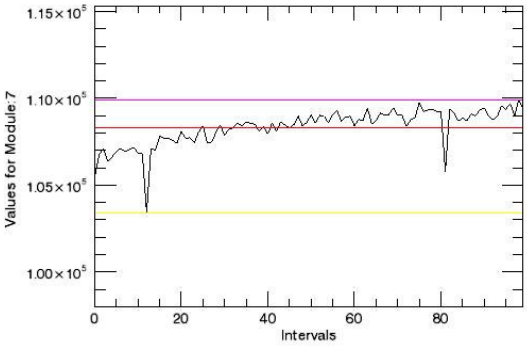
Graph 4.12:e



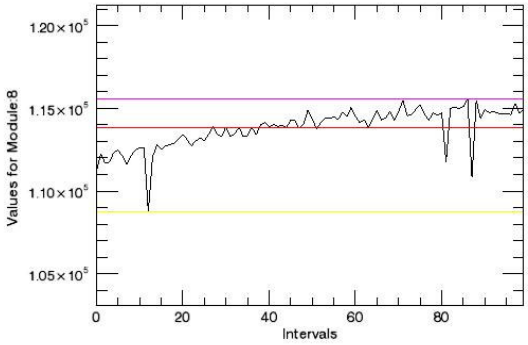
Graph 4.12:f



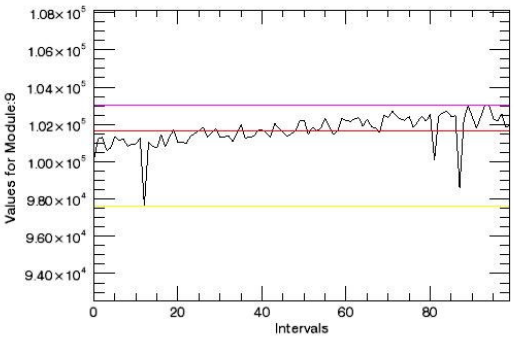
Graph 4.12:g



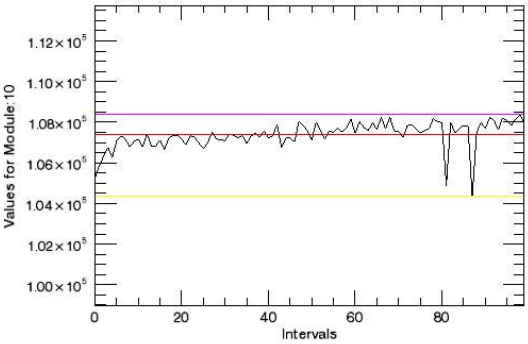
Graph 4.12:h



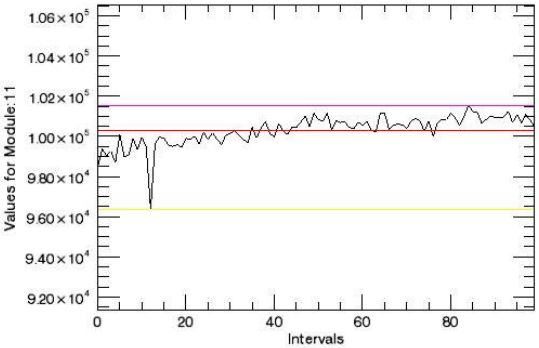
Graph 4.12:i



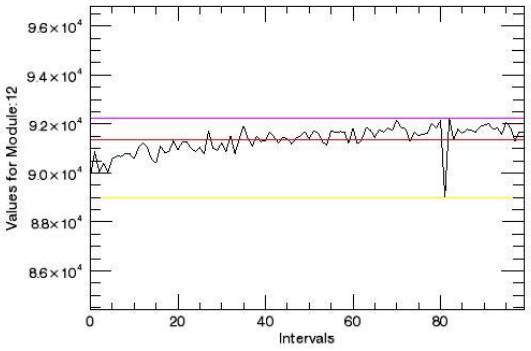
Graph 4.12:j



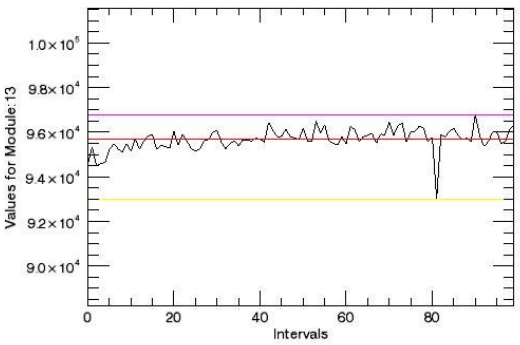
Graph 4.12:k



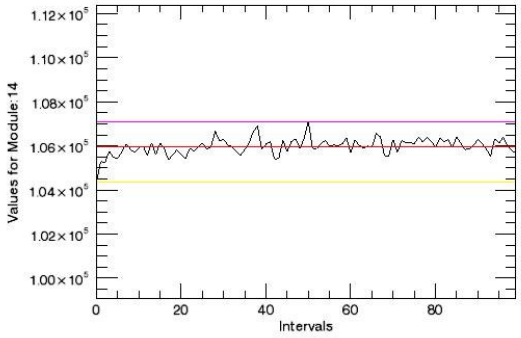
Graph 4.12:l



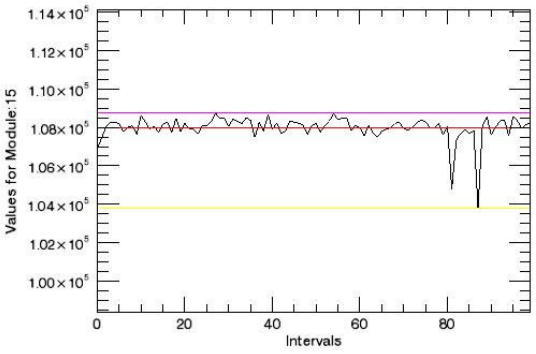
Graph 4.12:m



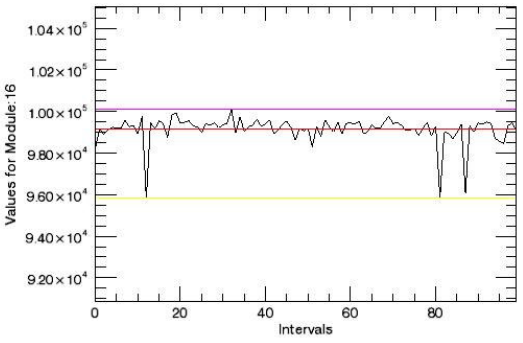
Graph 4.12:n



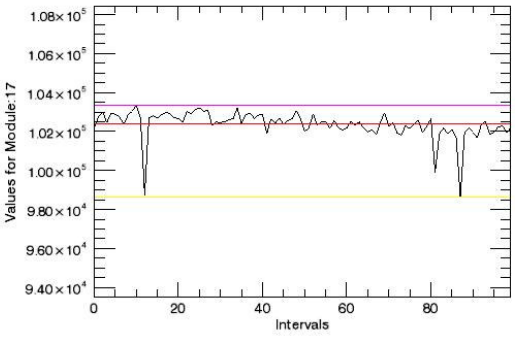
Graph 4.12:o



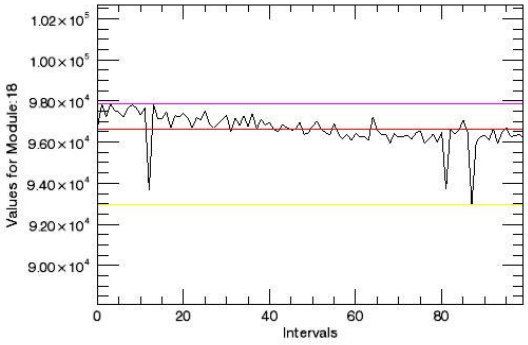
Graph 4.12:p



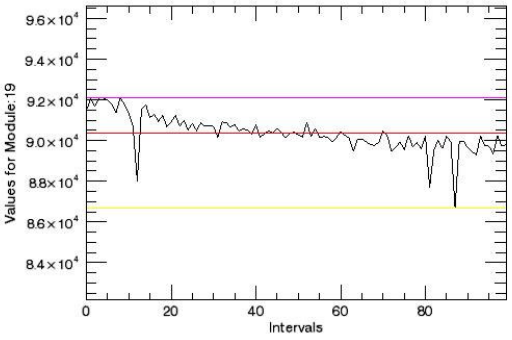
Graph 4.12:r



Graph 4.12:s



Graph 4.12:t



Graph 4.12:u

Graph 4.12: Intervals versus values for module groups for all ^{68}Ge Data without Gantry Rotation

As seen in Graph 4.12 deviations are seen in some measurement values. However, when Y axes begin from zero likewise in Graph 4.11, these deviations seem not much big**.** It is tested with calculation of standard deviations in the fallowing outputs.

Output 4.39:Average values of the module group counts for ^{68}Ge Data without Gantry Rotation. It starts from 0 and ends with 19th value.

average of module (0) counts: 103629.
average of module (1) counts: 111477.
average of module (2) counts: 95788.2
average of module (3) counts: 107204.
average of module (4) counts: 98981.0
average of module (5) counts: 99407.4
average of module (6) counts: 114172.
average of module (7) counts: 108337.
average of module (8) counts: 113804.
average of module (9) counts: 101676.
average of module (10) counts: 107380.
average of module (11) counts: 100300.
average of module (12) counts: 91335.8
average of module (13) counts: 95677.0
average of module (14) counts: 105977.
average of module (15) counts: 107991.
average of module (16) counts: 99140.5
average of module (17) counts: 102380.
average of module (18) counts: 96651.5
average of module (19) counts: 90372.5

Output 4.40: Average and standard deviation of the module group mean values for ^{68}Ge data without gantry Rotation.

average of all module means: 102584.
standart deviation of all module means: 5671.46

Average count value of all modules is 102584±5671.

Output 4.41: Minimum values of the module group counts ^{68}Ge data without gantry rotation. It starts from 0 and ends with 19th value.

minimum of module (0) counts: 99200
minimum of module (1) counts: 110271
minimum of module (2) counts: 94330
minimum of module (3) counts: 102827
minimum of module (4) counts: 98101
minimum of module (5) counts: 96181
minimum of module (6) counts: 109014
minimum of module (7) counts: 103405
minimum of module (8) counts: 108772
minimum of module (9) counts: 97621
minimum of module (10) counts: 104342
minimum of module (11) counts: 96362
minimum of module (12) counts: 88988
minimum of module (13) counts: 93003
minimum of module (14) counts: 104378
minimum of module (15) counts: 103766
minimum of module (16) counts: 95819
minimum of module (17) counts: 98625
minimum of module (18) counts: 92934
minimum of module (19) counts: 86696

Output 4.42: Average and standard deviation of the module group minimum values for ^{68}Ge Data without Gantry Rotation.

average of all module minimums: 99231.8
standart deviation of all module minimums: 5292.10

Minimum count value of all modules is 99232±5292.

Output 4.43: Maximum values of the module group counts for ^{68}Ge Data without Gantry Rotation. It starts from 0 and ends with 19th value

maximum of module (0) counts: 106178
maximum of module (1) counts: 113534
maximum of module (2) counts: 97088
maximum of module (3) counts: 108318
maximum of module (4) counts: 99579
maximum of module (5) counts: 100420
maximum of module (6) counts: 115357
maximum of module (7) counts: 109922
maximum of module (8) counts: 115581
maximum of module (9) counts: 103059
maximum of module (10) counts: 108380
maximum of module (11) counts: 101556
maximum of module (12) counts: 92250
maximum of module (13) counts: 96790
maximum of module (14) counts: 107094
maximum of module (15) counts: 108756
maximum of module (16) counts: 100107
maximum of module (17) counts: 103337
maximum of module (18) counts: 97865
maximum of module (19) counts: 92113

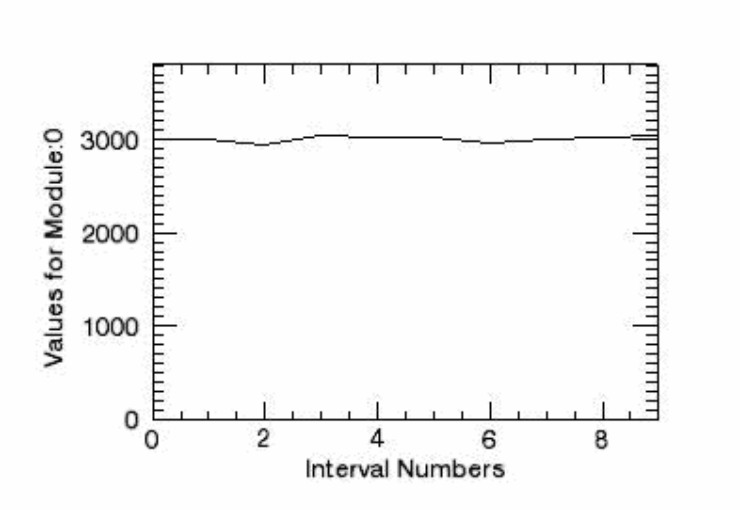
Output 4.44: Average and standard deviation of the module group maximum values for ^{68}Ge Data without Gantry Rotation.

average of all module maximums: 103864.
standart deviation of all module maximums: 5834.22

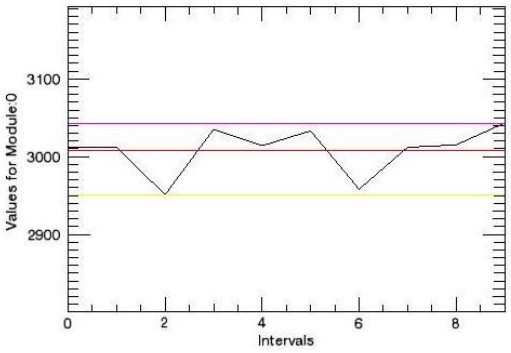
Maximum count value of all modules is 103864±5834.

4.1.6 Blank Scan Data without Gantry Rotation

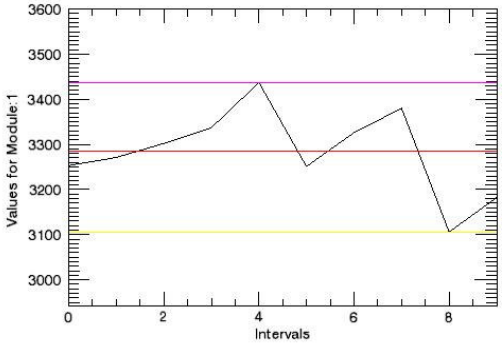
Cassette positions versus module groups are plotted (Graph 4.13 and Graph 4.14)



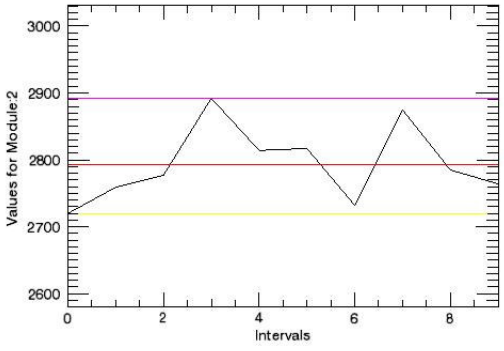
Graph 4.13: An example graph for interval numbers versus values for module groups for Blank Scan Data without Gantry Rotation. Y values begin from zero.



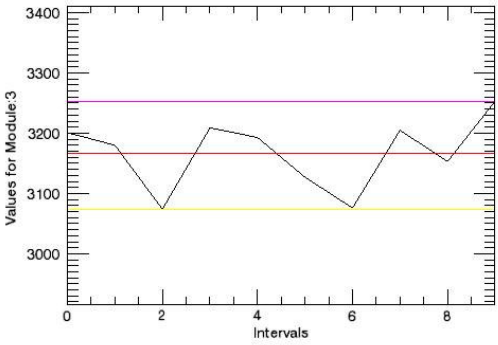
Graph 4.14:a



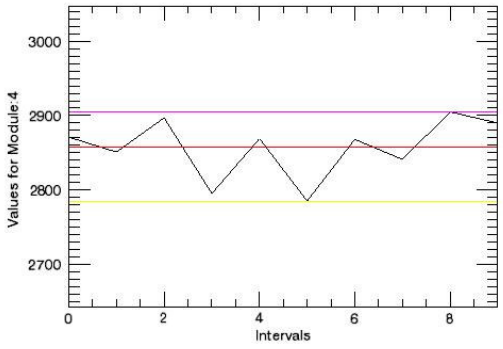
Graph 4.14:b



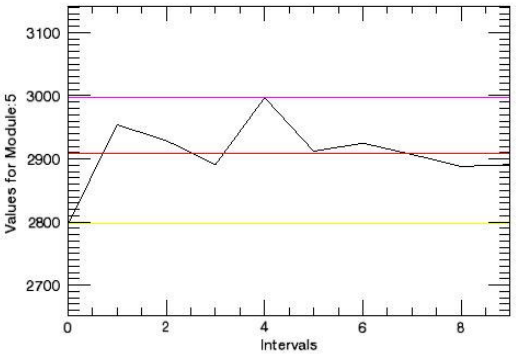
Graph 4.14:b



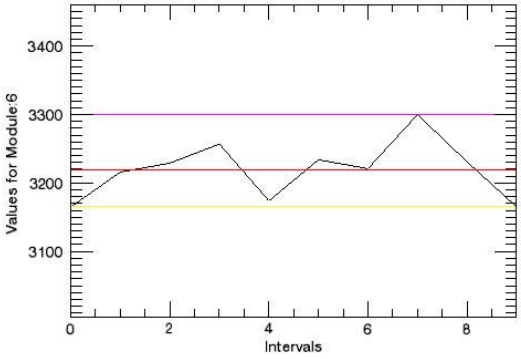
Graph 4.14:d



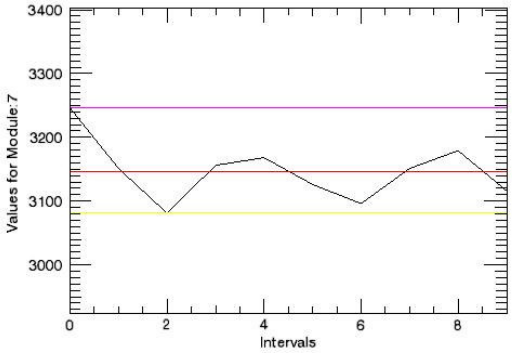
Graph 4.14:e



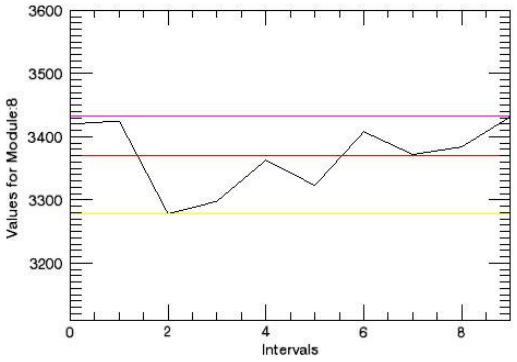
Graph 4.14:f



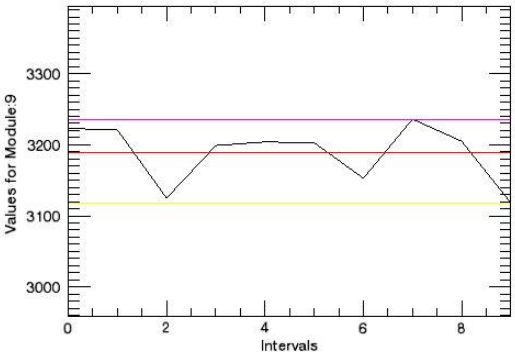
Graph 4.14:g



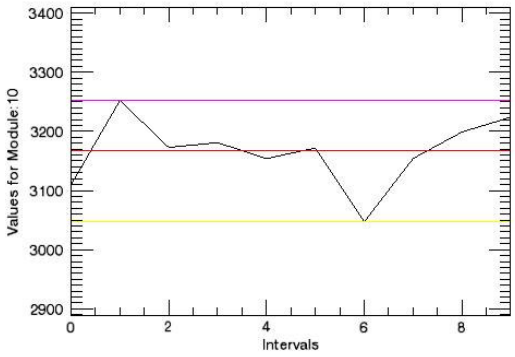
Graph 4.14:h



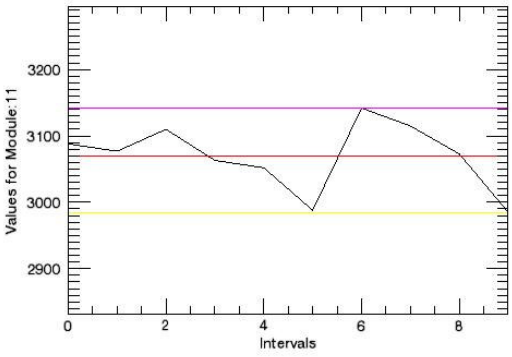
Graph 4.14:i



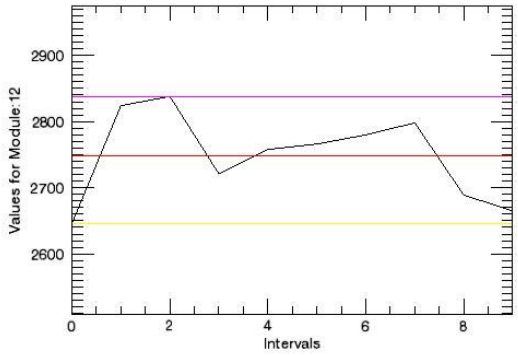
Graph 4.14:j



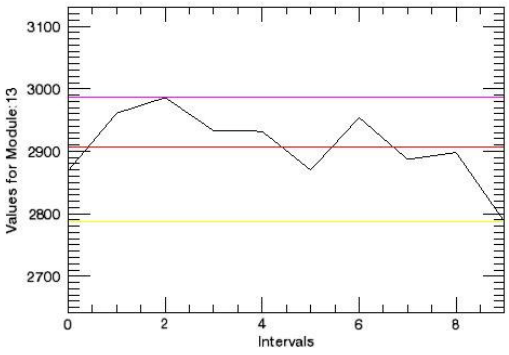
Graph 4.14:k



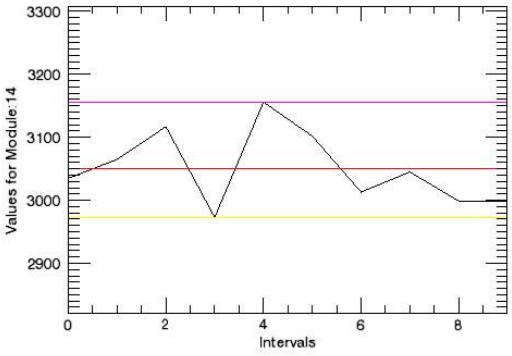
Graph 4.14:l



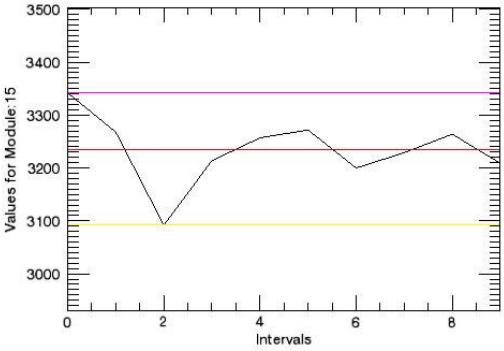
Graph 4.14:m



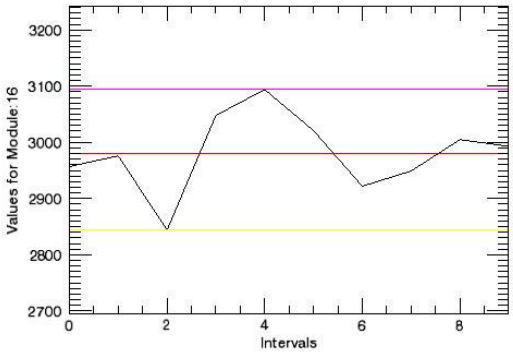
Graph 4.14:n



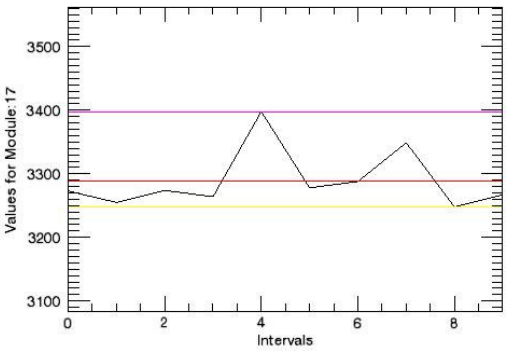
Graph 4.14:o



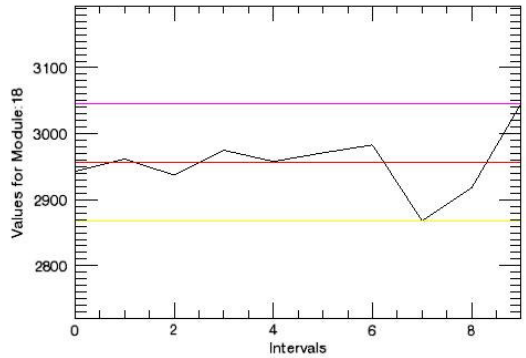
Graph 4.14:p



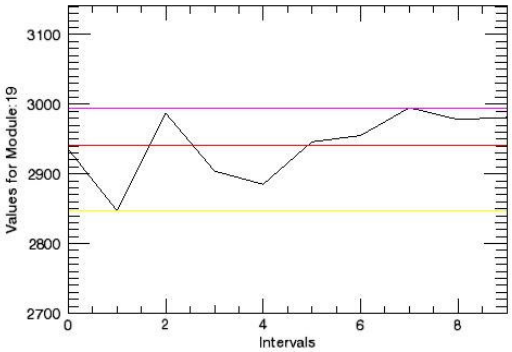
Graph 4.14:r



Graph 4.14:s



Graph 4.14:t



Graph 4.14:u

Graph 4.14: Intervals versus values for module groups for all Blank Scan Data without Gantry Rotation

Output 4.45: Average values of the module group counts for Blank Scan Data without Gantry Rotation. It starts from 0 and ends with 19th value.

average of module (0) counts: 3008.50
average of module (1) counts: 3284.50
average of module (2) counts: 2793.50
average of module (3) counts: 3167.10
average of module (4) counts: 2857.20
average of module (5) counts: 2909.10
average of module (6) counts: 3219.20
average of module (7) counts: 3147.00
average of module (8) counts: 3370.30
average of module (9) counts: 3188.70
average of module (10) counts: 3166.60
average of module (11) counts: 3069.30
average of module (12) counts: 2748.50
average of module (13) counts: 2907.70
average of module (14) counts: 3050.40
average of module (15) counts: 3234.130
average of module (16) counts: 2980.90
average of module (17) counts: 3289.40
average of module (18) counts: 2956.10
average of module (19) counts: 2941.30

Output 4.46: Average and standard deviation of the module group mean values for Blank Scan Data without Gantry Rotation.

average of all module means: 3064.50
standart deviation of all module means: 149.180

Average count value of all modules is 3065±149.

Output 4.47: Minimum values of the module group counts Blank Scan Data without Gantry Rotation. It starts from 0 and ends with 19th value.

minimum of module (0) counts: 2951
minimum of module (1) counts: 3106
minimum of module (2) counts: 2720
minimum of module (3) counts: 3074
minimum of module (4) counts: 2785
minimum of module (5) counts: 2797
minimum of module (6) counts: 3165
minimum of module (7) counts: 3081
minimum of module (8) counts: 3278
minimum of module (9) counts: 3118
minimum of module (10) counts: 3047
minimum of module (11) counts: 2985
minimum of module (12) counts: 2646
minimum of module (13) counts: 2787
minimum of module (14) counts: 2973
minimum of module (15) counts: 3092
minimum of module (16) counts: 2844
minimum of module (17) counts: 3248
minimum of module (18) counts: 2868
minimum of module (19) counts: 2847

Output 4.48: Average and standard deviation of the module group minimum values for Blank Scan Data without Gantry Rotation.

average of all module minimums: 2970.60
standart deviation of all module minimums: 149.040

Minimum count value of all modules is 2971±149.

Output 4.49: Maximum values of the module group counts for Blank Scan Data without Gantry Rotation. It starts from 0 and ends with 19th value

maximum of module (0) counts: 3043
maximum of module (1) counts: 3437
maximum of module (2) counts: 2892
maximum of module (3) counts: 3253
maximum of module (4) counts: 2905
maximum of module (5) counts: 2997
maximum of module (6) counts: 3300
maximum of module (7) counts: 3246
maximum of module (8) counts: 3432
maximum of module (9) counts: 3236
maximum of module (10) counts: 3252
maximum of module (11) counts: 3142
maximum of module (12) counts: 2838
maximum of module (13) counts: 2986
maximum of module (14) counts: 3156
maximum of module (15) counts: 3342
maximum of module (16) counts: 3094
maximum of module (17) counts: 3398
maximum of module (18) counts: 3046
maximum of module (19) counts: 2995

Output 4.50: Average and standard deviation of the module group maximum values for Blank Scan Data without Gantry Rotation.

average of all module maximums: 3149.50
standart deviation of all module maximums: 155.700

Maximum count value of all modules is 3149±156.

4.2 SUMMARY OF RESULTS

The results are summarized in Table 4.1.Minimum and maximum angles and their standard deviations are listed. Average, minimum and maximum counts and their standard deviations are indicated with deviation between minimum and maximum counts.

Table 4.1: Summary of results

Source	Rotation	Min. Angle	Max. Angle	Ave. Count	Min. Count	Max. Count	%Dev. (Min-Max)
^{68}Ge	+	53±14	227±12	13718±926	12594±852	15162±1000	16.9
^{18}F	+	246±16	75±16	16348±9676	158863±9152	168236±9954	5.6
Blank	+	182±59	171±91	5933±300	5703±297	6158±310	7.4
^{68}Ge	-	-	-	102584±5671	99232±5292	103864±5834	4.5
Blank	-	-	-	3065±149	2971±149	3149±156	5.6

5. DISCUSSION

The program named Module_Check was written and then it was tested with acquired data whether to check it worked correctly. The Module_Check is providing an important facility for examination of the cassette of the ClearPET. Consecutive counted data which are composed of several lines and columns in ASCII format can be arranged with this program. Mostly sizes of acquired data are huge. Therefore, they can not be checked easily manually. The modules can be evaluated with Module_Chech via plotting graphs, finding maximum, minimum, average values and their standard deviations as well. The data of ^{68}Ge, ^{18}F and blank scan measurement which was done during gantry rotation and without rotation was used to test this program.

The program was checked whether it really finds the errors. The blank scan measurements which were taken without gantry rotation are used for testing and it was already known that it contained inaccurate data. The results were compared with the ASCII formatted data. Module_Check determined the wrong module group. Additionally, the program found errors in ^{18}F data which is normally very difficult to found out with scanning ASCII data. Furthermore it determined the localization and the starting and finishing point of the error. Anyway checking the ASCII data is not a practical approach in daily use. Hence, Module_Check may be useful especially in determining module errors and localizations of sharp count changes.

Module_Check was tested with the ^{68}Ge and blank scan data which were obtained while the gantry was in fixed state. Graphics of both data seem linear. Afterwards the program was tested with the ^{68}Ge and ^{18}F and blank scan. Measurements was taken while the gantry rotation. Graphics of the ^{68}Ge and ^{18}F sources look like sinus curve.

The reason for the sinus curve of graphics obtained with ^{68}Ge and ^{18}F sources while the gantry rotation may be arising from varying factors as FOV problems, gravitational force while the gantry rotates and the effect of earth's magnetic field.

A source in a tomographic acquisition must exactly be in the center. The source outside of the center produces also a sinusoidal course of count distribution in single projections. Normally the sources are putted in the FOV very carefully. However, unforeseen mistakes may be made. Tomographic viewing is obviously very sensitive on external effects. Also, the warmth development in the cassette is important too.

The PMTs of Cassette 1 is looking down at 0° and they are looking up after the gantry rotates 180° (Figure 2.22).The electronic connections of PMTs can become slack as affected from gravitational force while the gantry rotates. Therefore count variation can occur.

Another reason of the sinusoidal variation can be the effect of earth's magnetic field. Tripathi et al., who searched the effect of earth's magnetic field on PMTs in their study [22], found out that the anode signal and photoelectron collection efficiency can vary depending on the PMT orientation. They made clear that the angle 0° corresponds to the case when the short pin on the PMT was facing local magnetic north, with the photocathode facing vertically downward. The PMT was rotated clockwise (while looking down) from this starting position. They plotted graph which shows the fractional deviation of the anode signal from the average of all data points versus angle from starting point. This graph seems like sinus wave. Minimum value is nearly at 90° and maximum value is at 270°.

Minimum and maximum angle values of the sinus curves were obtained to determine whether there were any systematic behavior in minimums and maximums. Averages of minimum and maximum angle values of ^{68}Ge counts were determined. Minimum angle value is between 0°-90° while maximum one is between 180°-270°. Opposite to ^{68}Ge, the minimum angle value of ^{18}F was between 180°-270° while the maximum value is between 0°-90°. The results of ^{68}Ge are concordant with the study mentioned above. On the contrary, minimum values of ^{18}F take place in third 90° and maximum values are at first 90°. Normally, all measurements were taken while the gantry rotates CW. We bring the gantry back to the starting position when we take sequential measurements with different sources. Otherwise, the gantry takes measurements from 360° to 0° as it was mentioned in Section 2.4.2.4. In that case, Cassette 1 exists instead of Cassette 19. For example, 4th Cassette (16th Module Group) measured maximum value in the first test while 14th Cassette (6th Module Group) measures in the second. Probably ^{18}F experiment was made while the gantry is rotating CCW. Therefore, minimum and maximum values may differ from ^{68}Ge results.

However, blank scan measurement seems a noisy line, also not like sinusoidal in spite of taken during gantry rotation. The graphic is very noisy. Therefore the sinusoidal variation may not be recognized even if the counts are affected from earth's magnetic field.

LSO and LuYAP are used as crystal materials. Lutetium is radioactive therefore we can collect counts despite no source in the FOV. ^{68}Ge and ^{18}F rotating measurements are taken when the sources are centered in the FOV. Putting the source in the center of the FOV can not create any problem, because in the blank scan the sources are in the gantry and they

also rotate while the gantry rotates. Therefore, this measurement may look like the other measurements which are taken when the gantry didn't rotate.

% deviation between minimum and maximum counts of ^{68}Ge taken during gantry rotation is bigger than ^{18}F as noise of ^{68}Ge measurements is bigger than ^{18}F's. Duration of each measurement was 2 s in ^{68}Ge and 1 s in ^{18}F however they both take 360 measurements. It can be a reason of the big deviation of ^{68}Ge counts.

Average, minimum and maximum values of the module counts can be found with Module_Check. We may use this property to make statistical calculations in the future. Here we only showed that this program may be useful for arranging data and finding errors.

This study was made one year ago. For the present condition, the ClearPET is revising nowadays. It is working on crystal blocks again and is studying to increase the light output of crystals. Also the warmth development in the cassette is important. Additionally to this, small ventilators are incorporated into the cassettes, so that the electronics does not superheat and we have stable ratios. Module_Check may be help for further revisions of ClearPET in the future.

6. CONCLUSION AND SUGGESTION

The program named Module_Check based on IDL 1.6 (Interactive Data Language) has written to evaluate the information which comes from each module group and are stored in .ang files during the detection. A useful program used by checking counts of cassettes was prepared in this study.

It is seen that the program is functional and runs correctly and can give lots of information about measurements. Count variations and module which affect the quality of images can be shown. This program may be suggested in order to use finding the errors of cassettes. Module_Check may be revised according to requirements of daily use. It may help providing further developments of ClearPET.

APPENDIX A

DATA OF THE BLANK SCAN WITH NO ROTATION

1	0	0	0.000	1.000	
3012	2315	480	200	17	0
3253	2444	524	268	17	0
2720	2015	516	178	11	0
3201	2452	471	250	28	0
2871	2259	409	192	11	0
2797	2106	481	190	20	0
3165	2367	519	261	18	0
3246	2405	566	251	24	0
3420	2417	667	306	30	0
3223	2417	519	257	30	0
3110	2268	566	244	32	0
3088	2268	538	254	28	0
2646	2012	440	179	15	0
2869	2129	511	213	16	0
3035	2378	452	178	27	0
3342	2532	541	238	31	0
2957	2329	439	173	16	0
3273	2454	516	246	57	0
2943	2381	390	153	19	0
2935	2238	488	197	12	0
2	0	0	1.125	1.000	
3012	2342	450	209	11	0
3271	2415	569	265	22	0
2759	2100	437	206	16	0
3180	2440	457	261	22	0
2851	2155	490	197	9	0
2954	2208	506	221	19	0
3216	2382	531	293	10	0
3151	2290	571	251	39	0
3425	2426	651	313	35	0
3221	2450	501	245	25	0
3252	2437	548	245	22	0
3077	2235	567	250	25	0
2824	2162	471	177	14	0
2961	2238	503	205	15	0
3065	2431	434	173	27	0
3268	2414	566	270	18	0
2976	2326	440	192	18	0
3255	2476	499	232	48	0
2962	2367	403	172	20	0
2847	2163	470	202	12	0
3	0	0	2.250	1.000	
2951	2288	449	200	14	0
3302	2409	593	283	17	0
2777	2052	494	210	21	0
3074	2330	456	259	29	0
2897	2250	457	183	7	0
2929	2228	494	183	24	0
3229	2409	514	272	34	0
3081	2265	560	230	26	0
3278	2340	615	291	32	0
3125	2385	506	209	25	0

3173	2402	540	209	22	0
3110	2301	565	228	16	0
2838	2215	429	185	9	0
2986	2238	521	207	20	0
3117	2421	480	192	24	0
3092	2336	500	244	12	0
2844	2200	442	179	23	0
3274	2447	530	245	52	0
2937	2394	380	147	16	0
2987	2267	509	193	18	0
4	0	0	3.375	1.000	
3035	2332	456	227	20	0
3337	2505	548	259	25	0
2892	2183	489	216	4	0
3209	2428	526	233	22	0
2795	2147	454	190	4	0
2891	2125	552	200	14	0
3257	2430	552	250	25	0
3156	2328	556	250	22	0
3298	2365	612	296	25	0
3199	2403	538	233	25	0
3181	2386	539	235	21	0
3063	2258	541	231	33	0
2721	2147	414	154	6	0
2933	2238	476	205	14	0
2973	2370	403	184	16	0
3214	2408	505	273	28	0
3048	2402	434	182	30	0
3264	2488	484	248	44	0
2975	2397	374	180	24	0
2904	2210	442	233	19	0
5	0	0	4.500	1.000	
3014	2321	451	225	17	0
3437	2540	595	278	24	0
2814	2174	453	180	7	0
3193	2442	465	260	26	0
2869	2239	436	180	14	0
2997	2257	519	204	17	0
3174	2302	560	290	22	0
3168	2296	578	266	28	0
3363	2378	661	300	24	0
3204	2478	475	224	27	0
3154	2351	560	219	24	0
3052	2232	546	250	24	0
2758	2173	418	157	10	0
2932	2178	532	206	16	0
3156	2514	442	162	38	0
3257	2454	528	252	23	0
3094	2416	446	210	22	0
3398	2579	528	244	47	0
2958	2395	395	150	18	0
2885	2187	497	177	24	0
6	0	0	5.625	1.000	
3033	2325	480	218	10	0
3251	2424	539	274	14	0
2817	2119	482	205	11	0
3127	2391	458	252	26	0
2785	2128	458	192	7	0

2912	2126	510	246	30	0
3234	2410	532	265	27	0
3126	2324	561	214	27	0
3323	2397	586	322	18	0
3203	2399	537	233	34	0
3172	2382	541	226	23	0
2988	2208	517	238	25	0
2766	2175	419	162	10	0
2870	2155	511	190	14	0
3102	2436	467	175	24	0
3272	2464	540	240	28	0
3021	2366	450	183	22	0
3278	2499	515	203	61	0
2971	2414	382	163	12	0
2946	2251	443	245	7	0
7	0	0	6.766	1.000	
2958	2336	422	186	14	0
3326	2453	579	278	16	0
2732	2046	477	198	11	0
3076	2320	479	253	24	0
2868	2256	430	172	10	0
2925	2237	470	203	15	0
3221	2385	548	265	23	0
3096	2252	538	278	28	0
3408	2415	674	289	30	0
3153	2382	502	231	38	0
3047	2305	511	212	19	0
3142	2262	601	246	33	0
2780	2181	443	147	9	0
2954	2237	508	193	16	0
3013	2323	486	179	25	0
3200	2416	523	236	25	0
2922	2285	437	176	24	0
3288	2483	531	223	51	0
2983	2377	416	179	11	0
2955	2275	483	183	14	0
8	0	0	7.891	1.000	
3012	2344	457	197	14	0
3380	2472	620	273	15	0
2875	2171	500	196	8	0
3205	2434	504	237	30	0
2841	2205	457	172	7	0
2907	2158	507	211	31	0
3300	2409	597	270	24	0
3151	2303	573	253	22	0
3372	2419	613	315	25	0
3236	2384	554	266	32	0
3154	2397	520	218	19	0
3115	2253	572	266	24	0
2798	2193	425	173	7	0
2887	2165	503	208	11	0
3045	2365	472	182	26	0
3229	2460	482	261	26	0
2949	2283	467	183	16	0
3349	2531	534	242	42	0
2868	2322	365	168	13	0
2995	2253	501	219	22	0
9	0	0	9.016	1.000	

3015	2340	451	208	16	0
3106	2332	502	249	23	0
2785	2064	507	206	8	0
3153	2375	470	275	33	0
2905	2262	455	184	4	0
2888	2168	487	207	26	0
3231	2369	563	279	20	0
3179	2319	572	269	19	0
3384	2341	692	331	20	0
3205	2441	498	232	34	0
3199	2410	520	244	25	0
3073	2258	541	256	18	0
2689	2129	401	150	9	0
2898	2188	482	209	19	0
2999	2334	446	196	23	0
3264	2491	490	255	28	0
3005	2349	437	195	24	0
3248	2438	533	226	51	0
2918	2345	390	168	15	0
2978	2250	492	216	20	0

10	0	0	10.141	1.000	
3043	2353	473	207	10	0
3182	2402	536	235	9	0
2764	2043	497	209	15	0
3253	2415	524	291	23	0
2890	2185	475	218	12	0
2891	2160	489	218	24	0
3165	2370	530	235	30	0
3116	2248	602	240	26	0
3432	2433	674	299	26	0
3118	2335	543	214	26	0
3224	2440	546	216	22	0
2985	2215	534	221	15	0
2665	2047	461	146	11	0
2787	2108	469	190	20	0
2999	2321	484	171	23	0
3209	2440	527	215	27	0
2993	2337	432	203	21	0
3267	2451	509	250	57	0
3046	2504	355	164	23	0
2981	2288	451	218	24	0

APPENDIX B

THE PROGRAM MODULE_CHECK

```
pro Module_Check


;#####################################################
;Author: PINAR CELIK
;   RESEARCH CENTER JUELICH
;   GUEST STUDENT FROM DOKUZ EYLUL UNIVERSITY
;
;E-mail: pcelik@gmail.com
;
;#####################################################


folder= ''
print, 'Enter data folder: '
read, folder

file_names= FILE_SEARCH(folder, '*.ang', COUNT= NUMFILES)



print,'Enter the number of intervals of data to read from
file:'
NUMINTERVALS= 36
read, NUMINTERVALS

NUMCASSETTES=20 ; WE ALWAYS HAVE 20 CASSETTES


measurement= {cassette, num:0, angle:0, value:fltarr(6)}
definitions={header,num:0,startangle:0,bedpos:0,time:0.0,inter
valdur:0.0}


data= replicate({cassette}, NUMINTERVALS, NUMCASSETTES,
NUMFILES)
def=replicate({header},NUMINTERVALS, NUMFILES)



row= 0          ; row iterator
interval= 0        ; interval iterator
measurement= 0
temp1= 0L       ; header temps
temp2= 0L
```

```
temp3= 0L
temp4= 0L
temp5= 0L
temp6= 0L
tempf1=0.0
tempf2=0.0

modulsum=lonarr(NUMINTERVALS, NUMFILES)

;################################
; READING FILES, FILLING ARRAYS
; fill header arrays, fill row and column arrays for each
interval
;################################

on_ioerror, cant_read_anymore

for eachfile= 0, NUMFILES-1 do begin

openr,file, file_names[eachfile], /get_lun


for interval= 0,NUMINTERVALS-1 do begin

    readf, file,temp1,temp2, temp3, tempf1, tempf2
    def[interval,eachfile].num=temp1
    def[interval,eachfile].startangle=temp2/10
    def[interval,eachfile].bedpos=temp3
    def[interval,eachfile].time=tempf1
    def[interval,eachfile].intervaldur=tempf2

    for row= 0,NUMCASSETTES-1 do begin
        readf, file,temp1,temp2, temp3, temp4, temp5, temp6
        data[interval,row,eachfile].value[0]= temp1
        data[interval,row,eachfile].value[1]= temp2
        data[interval,row,eachfile].value[2]= temp3
        data[interval,row,eachfile].value[3]= temp4
        data[interval,row,eachfile].value[4]= temp5
        data[interval,row,eachfile].value[5]= temp6


        data[interval,row,eachfile].num=NUMCASSETTES-row


data[interval,row,eachfile].angle=def[interval,eachfile].start
angle

        if data[interval,row,eachfile].angle LT 0 then begin

data[interval,row,eachfile].angle=def[0,eachfile].startangle+d
ata[interval,row,eachfile].angle
```

```
        endif

       ;sum of all casette counts for each interval, store for
later
        modulsum(interval, eachfile)=modulsum(interval,
eachfile)+data[interval,row,eachfile].value[0]
    endfor ; row
endfor ; interval

free_lun,file,/FORCE
close,file ; close and free file slot each time we open a new
file

continue

cant_read_anymore:
print, 'Read error in file ', file_names[eachfile], 'data
interval ', interval, ' after casette ', row, '!'
print, 'Error: Unexpected number of intervals in file.'

endfor ; file


;If header angle changes, rotating
rotating=make_array(10, value= 1)
for eachfile= 0, NUMFILES-1 do begin
if def[0,eachfile].startangle eq def[1,eachfile].startangle
then begin
    rotating[eachfile]= 0
endif
endfor

;################################
; PLOT 1: Show each module pro interval
;################################


tempx= lonarr(NUMINTERVALS*NUMFILES)
tempy= lonarr(NUMINTERVALS*NUMFILES)



titley= 'Values for Module:   '

for eachrow= 0, NUMCASSETTES- 1 do begin


    for eachfile= 0, NUMFILES-1 do begin


       for eachinterval= 0, NUMINTERVALS- 1 do begin
```

```
            tempy(eachfile* NUMINTERVALS+ eachinterval)=
data[eachinterval, eachrow, eachfile].value[0]


            tempx(eachfile* NUMINTERVALS+ eachinterval)=
eachfile* NUMINTERVALS+ eachinterval


        endfor

    endfor



    titlenum= string(eachrow)
    titlenum= strtrim(eachrow, 2)
    strput, titley, titlenum, 18

    iplot, tempx, tempy,$
        linestyle=1,$

        yrange=[0,5* max(tempy)/ 4],$
       ; view_grid=[5,4],/view_next ,$
        title='Counts',$
        xtitle= 'Intervals', $
        ytitle= titley
endfor

;################################
; PLOT 2: Show each module pro angle
;################################

;a) for cassette positions (For long halflifed radiosotopes
like Ge-68)

tempx= lonarr(NUMINTERVALS*NUMFILES)
tempy= fltarr(NUMINTERVALS*NUMFILES)
tempxcassette=lonarr(NUMINTERVALS*NUMFILES)


minxarray= lonarr(NUMCASSETTES)
maxxarray= lonarr(NUMCASSETTES)


titley= 'Values for Module:   '


titlex= 'Cassette Position (xmin=    , xmax=    )'
```

```
for eachrow= 0, NUMCASSETTES- 1 do begin

    for eachfile= 0, NUMFILES-1 do begin

        for eachinterval= 0, NUMINTERVALS- 1 do begin


            tempy(eachfile* NUMINTERVALS+ eachinterval)=
data[eachinterval, eachrow, eachfile].value[0]

            tempx(eachfile* NUMINTERVALS+ eachinterval)=
eachfile* NUMINTERVALS+ eachinterval


        tempxcassette=tempx-((eachrow+1)*18)

        endfor
        ;print, tempy
    endfor

    titlex= 'Cassette Position (xmin=    , xmax=    )'

    titlenum= string(eachrow)
    titlenum= strtrim(titlenum, 2)

    strput, titley, titlenum, 18

    minx= where(tempy eq min(tempy))
    maxx= where(tempy eq max(tempy))

    tmix= tempxcassette[minx[0]]
    tmax= tempxcassette[maxx[0]]


    if tmix lt 0 then tmix=tmix+ 360
    minxarray[eachrow]= tmix
    if tmax lt 0 then tmax=tmax+ 360
    maxxarray[eachrow]= tmax



    minxstring= string(tempxcassette[minx[0]])
    minxstring= strtrim(minxstring, 2)
    maxxstring= string(tempxcassette[maxx[0]])
    maxxstring= strtrim(maxxstring, 2)

    strput, titlex, minxstring, 24
    strput, titlex, maxxstring, 35
```

```
    print, 'tempxcassette[minx]: ', tempxcassette[minx]
    print, 'tempxcassette[maxx]: ', tempxcassette[maxx]

    print, 'tempxcassette[minx]: ', tmix
    print, 'tempxcassette[maxx]: ', tmax

    print, 'average of minxes: ', median(minxarray)
    print, 'average of maxxes: ', median(maxxarray)


 iplot, tempx, tempy,$
 xtitle='Interval Numbers',$
     iplot, tempxcassette, tempy,$ ;for cassette angle
        ;linestyle=1,$;for dot curve


       yrange=[0,max(tempy)+ max(tempy)/ 4],$;begining at 0




        ;yrange=[min(tempy)-(max(tempy)-
min(tempy)),max(tempy)+(max(tempy)-min(tempy))],$


        ;view_grid=[5,4],/view_next ,$
        title='Counts',$
        xtitle= titlex, $

        xtitle='Cassette Positions',$
        ytitle= titley

endfor

print,'tempy 0: ', tempy[0]


; ********************************************
; b) Decay Correction for F18 data only


titley= 'Values for Module:   '
titlex= 'Cassette Position (xmin=    , xmax=    )'


for eachrow= 0, NUMCASSETTES- 1 do begin
    for eachfile= 0, NUMFILES-1 do begin

        for eachinterval= 0, NUMINTERVALS- 1 do begin
            tempy(eachfile* NUMINTERVALS+ eachinterval)=
data[eachinterval, eachrow, eachfile].value[0]  $
```

```
            /exp(-
(0.693/6588)*def[eachinterval,eachfile].time)
            tempx(eachfile* NUMINTERVALS+ eachinterval)=
eachfile* NUMINTERVALS+ eachinterval

         tempxcassette=tempx-((eachrow+1)*18); For caseete
position(For F18)
        endfor
       ;print, tempy
    endfor

    titlex= '       Cass. Pos.(xmin=    , xmax=    )'

    titlenum= string(eachrow)
    titlenum= strtrim(titlenum, 2)
    strput, titley, titlenum, 18

   ; stempy=smooth(tempy,sfactor); for smooth. sfactor is at
beginning

    minx= where(tempy eq min(tempy))
    maxx= where(tempy eq max(tempy))

    tmix= tempxcassette[minx[0]]
    tmax= tempxcassette[maxx[0]]

    if tmix lt 0 then tmix=tmix+360
    minxarray[eachrow]= tmix
    if tmax lt 0 then tmax=tmax+360
    maxxarray[eachrow]= tmax


    minxstring= string(tempxcassette[minx[0]])
    minxstring= strtrim(minxstring, 2)
    maxxstring= string(tempxcassette[maxx[0]])
    maxxstring= strtrim(maxxstring, 2)

    strput, titlex, minxstring, 23
    strput, titlex, maxxstring, 34



 print, titlex

  print, titlenum, ': minx: ', minx
  print, 'translated x: ', tempxcassette[minx]

   iplot, tempx, tempy,$

    print, 'tempxcassette[minx]: ', tempxcassette[minx]
    print, 'tempxcassette[maxx]: ', tempxcassette[maxx]
```

```
     iplot,tempxcassette, tempy,$ ;for cassette angle
        ;linestyle=1,$
        ;yrange=[0,max(tempy)+ max(tempy)/ 4],$
         yrange= [min(tempy)-
median(tempy)/20,max(tempy)+median(tempy)/20],$

        ;view_grid=[5,4],/view_next ,$
        title='Counts',$
        xtitle= titlex, $
        ytitle= titley


endfor

print, 'average of minxes: ', median(minxarray)
print, 'average of maxxes: ', median(maxxarray)



print, 'totalcounts 0: ', data[1, 0, 0].value[0]
print, 't: ', def[1,0].time
print, 'expon: ', exp(-(0.693/6588)* def[1,0].time)
print,'result 0: ', data[1, 0, 0].value[0]/exp(-(0.693/6588)*
def[1,0].time)




end


;Plot 2:
tempx= indgen(NUMCASSETTES)
tempy= lonarr(NUMCASSETTES)

for eachrow= 0, NUMCASSETTES- 1 do begin
    for eachfile= 0, NUMFILES-1 do begin
        for eachinterval= 0, NUMINTERVALS- 1 do begin
            tempy(eachrow)= tempy(eachrow)+ data[eachinterval,
eachrow, eachfile].value[0]
       endfor
    endfor
endfor

iplot,tempx,tempy,$
       yrange=[0, 5* max(tempy)/ 4],$
       title='Sum',$
       xtitle='Modules', $
       ytitle='Sum Values'
```

```
;#################################################################
; Plot 3:
;#################################################################

tempx= indgen(NUMCASSETTES)
tempy= lonarr(NUMINTERVALS*NUMFILES)
tempmax=lonarr(NUMCASSETTES)

for eachrow= 0, NUMCASSETTES- 1 do begin
    for eachfile= 0, NUMFILES-1 do begin
        for eachinterval= 0, NUMINTERVALS- 1 do begin
            tempy(eachfile* NUMINTERVALS+ eachinterval)=
data[eachinterval, eachrow, eachfile].value[0]
            tempmax(eachrow)=max(tempy)
       endfor
    endfor
endfor

iplot,tempx,tempmax,$
       yrange=[0, 5* max(tempmax)/ 4],$
       title='Max',$
       xtitle='Modules', $
       ytitle='Max Values'

;#################################################################
; Plot 4:
;#################################################################

tempx= indgen(NUMCASSETTES)
tempy= lonarr(NUMINTERVALS*NUMFILES)
tempmin=lonarr(NUMCASSETTES)

for eachrow= 0, NUMCASSETTES- 1 do begin
    for eachfile= 0, NUMFILES-1 do begin
        for eachinterval= 0, NUMINTERVALS- 1 do begin
            tempy(eachfile* NUMINTERVALS+ eachinterval)=
data[eachinterval, eachrow, eachfile].value[0]
            tempmin(eachrow)=min(tempy)
       endfor
    endfor
endfor

iplot,tempx,tempmin,$
       yrange=[0,5* max(tempmin)/ 4],$
       title='Min',$
       xtitle='Modules', $
       ytitle='Min Values'
```

```
;#################################################################
; PLOT 5:
;#################################################################

print, 'Now we can plot the desired intervals(s).'
print, '1.Do you want to plot All? ', NUMINTERVALS, '
intervals times ', NUMFILES, ' files'
print, '2.Do you want to plot a time range'

tempx= lonarr(NUMCASSETTES)
tempy= lonarr(NUMCASSETTES)

var= 2
ask:
read, var

if (var ne 1) and (var ne 2) then begin
    print,'choose 1 or 2'
    goto, ask
endif



if (var eq 1) then begin

for eachfile= 0, NUMFILES-1 do begin

    for eachinterval= 0, NUMINTERVALS- 1 do begin

        for eachrow= 0, NUMCASSETTES- 1 do begin
            tempy(eachrow)= data[eachinterval, eachrow,
eachfile].value[1]
            tempx(eachrow)= eachrow
        endfor

        iplot,tempx,tempy,$
        view_grid=[6,6],/view_next ,$
        ;yrange=[0,max(tempy)+ max(tempy)/ 4],$;begining at 0
        title='Row Values',$
        xtitle= 'Modules', $
        ytitle= 'Module Counts'
    endfor

endfor

endif
```

```
if (var eq 2) then begin

first_input=0
second_input=0
print,'enter beginning of time range and enter end of time
range'
read, first_input, second_input

viewvalue=0
for eachfile= 0, NUMFILES-1 do begin
    for eachinterval= 0, NUMINTERVALS- 1 do begin
      if (def[eachinterval, eachfile].time ge first_input) and
$
            (def[eachinterval, eachfile].time le second_input)
then begin

       viewvalue=viewvalue+1

       endif
    endfor
endfor

viewvalue_x=ceil(sqrt(viewvalue))
viewvalue_y=ceil(sqrt(viewvalue))


for eachfile= 0, NUMFILES-1 do begin

    for eachinterval= 0, NUMINTERVALS- 1 do begin

        for eachrow= 0, NUMCASSETTES- 1 do begin
            tempy(eachrow)= data[eachinterval, eachrow,
eachfile].value[0]
            tempx(eachrow)= eachrow
        endfor

        if (def[eachinterval, eachfile].time ge first_input)
and $
            (def[eachinterval, eachfile].time le second_input)
then begin

            titley= 'Values for time:         '
            titlenum= string(def[eachinterval, eachfile].time)
            titlenum= strtrim(def[eachinterval,
eachfile].time, 2)
            strput, titley, titlenum, 17
            err=sqrt(tempy)
            med=make_array(NUMCASSETTES,value=median(tempy))
            minimum=make_array(NUMCASSETTES,value=min(tempy))
```

```
            maximum=make_array(NUMCASSETTES,value=max(tempy))

            iplot,tempx,tempy, linestyle=1,$
          yerror=err,/y_errorbar, $

            view_grid=[viewvalue_x,viewvalue_y],/view_next
          iplot,tempx,med,thick=2,color=[255,0,0],/overplot

iplot,tempx,minimum,thick=2,color=[255,255,0],/overplot

iplot,tempx,maximum,thick=2,color=[255,0,255],/overplot,$



            ;/histogram
            stylename='Meshif', $

            title='Row Values',$
            xtitle= 'Modules', $; write maximum after module
            ytitle= titley

Print,med
print,meanabsdev(tempy)


       endif
    endfor
endfor

endif



end; PROGRAM
```

REFERENCES

[1] Cherry S R, Sorenson J A, Phelps M E, Physics in Nuclear Medicine, Sounders, third edition, 2003

[2] Khodaverdi M, Designstudie eines μCT-Zusatzes für einen hochauflösenden Positronen-Emissions-Tomographen, PhD Thesis, Forschungszentrum, Jülich, 2004

[3] Pointon B, NMED 5510 Positron emission tomography with dedicated and dual coincidence cameras, British Columbia Institute of Technology, 7/2000

[4] Let's play PET,
http://www.crump.ucla.edu/software/lpp/nuclearphysics/imagerecon.html
retrieved at 01.07.2005

[5] PET in more detail, Nuclear Physics& Tomography, Positron emission tomography, Austin& Repatriation Medical Centre, Melbourne, Australia
http://www.petnm.unimelb.edu.au/pet/detail/nucphysics.html
retrieved at 28.07.2005

[6] SPWLA Glossary, Society of Petrophysicist and Well Log Analysts
http://www.spwla.org/
retrieved at 26.07.2005

[7] Lilley J, Nuclear Physics, Jonn Willey &Sons Ldt, 08/2002

[8] Purdue University, Industrial and Physical Pharmacy.
http://www.ipph.purdue.edu/faculty/Green-PET-Camera.pdf
retrieved at 28.07.2005

[9] Physics. University of Southern California. PET Imaging Centre.
https://petcenter.hsc.usc.edu/USC_Site/html/physics.html
retrieved at 01.08.2005

[10] Wienhard K, Positron Emission Tomography
http://www.nupecc.org/iai2001/report/B23.pdf
retrieved at 05.08.2005

[11] Weber S, Christ D, Kurzeja M, Engels R, Kemmerling G, Halling H, Comparison of LuYAP, LSO, and BGO as scintillators for high resolution PET detectors, IEEE Transactions on Nuclear Science, 10.1109/TNS.2003.817952, 2003; 50(3):1370-1372

[12] Fahey F H, Data Acquisition in PET Imaging, Journal of Nuclear Medicine Technology, 2002; 30(2): 39-49

[13] Surti S, Karp J S, Perkins A E, Cardi C A et al, Imaging performance of a PET: a small animal PET camera, IEEE Trans Med Imaging, 2005;24(7):844-52.

[14] Scaefers K P, Imaging small animals with positron emission tomography, Nuklearmedizin 2003;3:86-89

[15] Auffray E, Bruyndonckx P, Devroede O,Fedorov A et. al, The Clear PET project, Nuclear Instruments and Methods in Physics Research 2004; 257: 171-174

[16] Heinrich U, Auslegung des Detektorsystems für einen hochauflösenden Positronen-Emissions- Tomographen mit hoher Sensitivitaet, PhD Thesis, Forschungszentrum Juelich, 2003

[17] Tavernier S, Clear PET offers improved insight into animal brains, Cern Courier
http://www.cerncourier.com/main/article/45/6/19
retrieved at 23.08.2005

[18] Ray-test
http://www.raytest.de/index2.html
retrieved at 03.03.2006

[19] Heinrichs U, Pietrzyk U, Ziemons K, Design optimization of the PMT-ClearPET prototypes based on simulation studies with GEANT3, IEEE Transactions on Nuclear Science 2003;50:1428-1432

[20] Heinrichs U, Auffray E, Barbier R, Brandenburg G et al, The ClearPET™: A high resolution high sensitivity dual-layer phoswich small animal PET scanner CERN: European Organisation of Nuclear of Nuclear Research, Physics Department. Electronic Systems Support Group.
http://ess.web.cern.ch/ESS/IEEE2004/IndustrialProgramme/SeminarPresentations/CERN-Raytest.pps
retrieved at 02.07.2005

[21] Niang B, Diadem analysis tool for a high-resolution small animal PET system, Master of Science thesis, 2004

[22] A. Tripathi, K. Arisaka, T. Ohnuki, P. Ranin, Effect of earth's magnetic field

on production Photonis PMTs, UCLA-Cosmic,2001;04

[23] Photo Multiplier Tubes
http://technology.niagarac.on.ca/courses/batp9301/PMT.html
retrieved at 04.09.2006

[24] The Photomultiplier Tube
http://laxmi.nuc.ucla.edu:8248/M248_99/autorad/Scint/pmt_diagram.GIF
retrieved at 04.09.2006