UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

Dipartimento di Fisica Giuseppe Occhialini

Characterization of a simulated Positron Emission Tomography (PET) apparatus

Matteo Bagnalasta

Simone Capelli

Arianna Chiapparini

Giulia Peveri

Prof. Chiara Brofferio

Prof. Francesco Terranova

Laboratorio di Misure Nucleari e Subnucleari

Anno accademico 2017/2018

Contents

Part I

Introduction

Chapter 1

Positron Emission Tomography

The Positron Emission Tomography (PET) [1] is a non-invasive technique used in nuclear medicine that allows to observe metabolic processes in the body as an aid to diagnose and determine the gravity of a disease. In particular, it is most often used with cancer, heart disease and brain disorders.
The pictures from a PET scan provide a functional information different from the anatomical one obtained by other types of diagnostic techniques, such as Computerized Tomography (CT) or Magnetic Resonance Imaging (MRI). This is the reason why they are usually used in parallel.
PET procedure permits the determination of biological functions, metabolism and pathology following the administration of short-lived positron-emitting *radiopharmaceuticals*, that are drugs given to the patient, containing a radioactive atom that decays in a more stable one mostly by β^+ decay. For example, it is often used the fludeoxyglucose ^{18}F, a polysaccharide enriched with a radioactive isotope. Due to the fact that cancerous cells have a faster metabolism than healthy ones, the isotope gathers in those regions. For this reason, PET permits a quantitative imaging.
After the radioactive atom emits a positron, it quickly loses its energy in the surrounding medium and, coming to rest, it combines with a free electron. The two particles annihilate and their energy is converted into back-to-back 511 keV photons that travel until they reach detectors, set in a multiple coincidence configuration. For clinical purposes the detectors are positioned in order to form rings around the patient, this way they can gather
both linear and angular information at the same time.
In figure 1.1 it is depicted the clinical process.

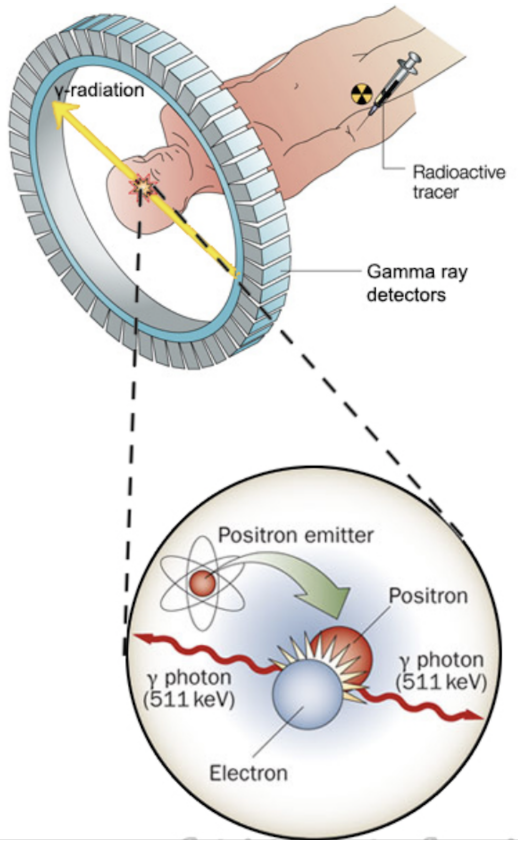


Figure 1.1: *Positron Emission Tomography scheme*

In order to obtain the final image, the radiation detectors convert the γ energy into electrical signals that are then processed by a computer, which reconstructs the distribution of radioactivity in the desired organ.
This process starting point is the *sinogram*, a digital image that holds the linear and angular information of the studied radioisotope activity. An example of an acquisition process and its relative sinogram is depicted in figure 1.2.
There are two types of image reconstruction process: analytical and iterative. The most commonly used techniques are respectively *Filtered Back Projection (FBP)* and *Simultaneous Algebraic Reconstruction Technique (SART)*.

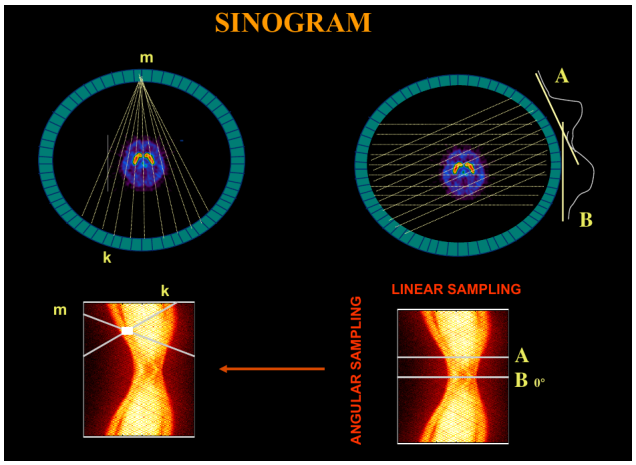
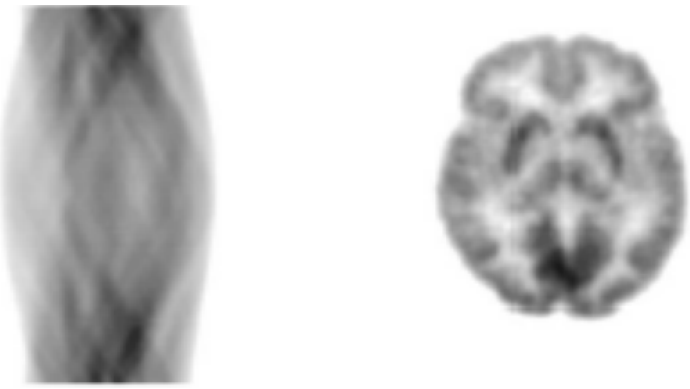


Figure 1.2: *Acquisition process and relative sinogram. The yellow lines indicate the detectors in a coincidence configuration*



(a) (b)

Figure 1.3: *A sinogram (a) and the image (b) reconstructed through FBP*

Chapter 2

Experimental setup

The purpose of the experiment is to simulate a two-dimensional PET apparatus to obtain the position of a radioactive source using two scintillators.
To support the detectors, an electric chain, which has to be characterized and optimized for the goal, and a rotating plate, connected to an electric engine, are needed.

2.1 The source

The source of radiation (annihilation photons) used in this project is a ^{22}Na source. The ^{22}Na decades with an half-life of 2.6 years with a β^+ decay on ^{22}Ne.

$$^{22}Na \rightarrow ^{22} Ne^* + e^+ + \nu_e$$

Then, the positron emitted in this decay annihilates, with the consequent emission of two photons with energy of 0.511 MeV. Furthermore, the ^{22}Ne is usually produced in the excited state $J^P = 2^+$. This involves the decay of the Ne atom on the ground state through the emission of a photon with energy of 1.274 MeV.

$$e^- + e^+ \rightarrow \gamma\gamma \qquad\qquad ^{22}Ne^* \rightarrow ^{22} Ne + \gamma$$

Even if the focus is on the 511 keV peak, two peaks appear in the spectrum, as shown in figure 2.1.

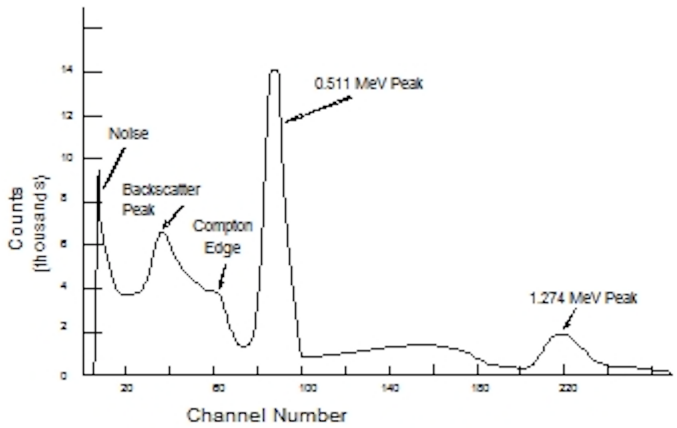
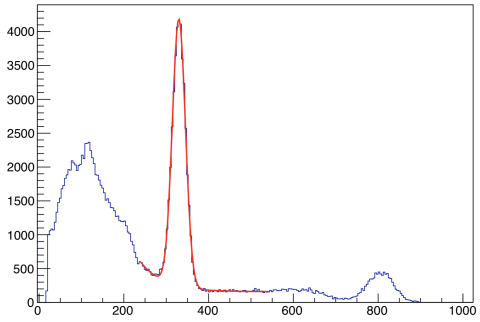
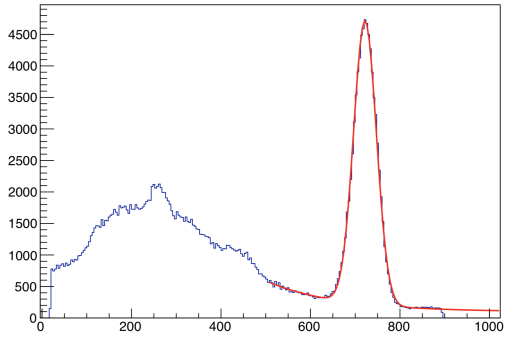


Figure 2.1: ^{22}Na *expected spectrum*

In particular, after 300 s of data acquisition, the obtained spectrum is the one depicted in fig. 2.2a. The Compton edge is barely visible, while the two peaks are well distinguished. The other figure (fig.2.2b) presents a focus on the 511 keV peak, obtained by increasing the gain of the amplifier.



(a) ^{22}Na *spectrum obtained after 300 s of acquisition*



(b) *Spectrum focused on the 511 keV peak*

Figure 2.2

Both figures are fitted preliminary with a Gaussian in the peak area and with a combination of an exponential and a constant function in the background area to get a good parameters starting point. Then the whole spectrum is refitted with a composition of Gaussian, exponential and constant function (see A for details).

2.2 The detectors and the rotating plate

The detectors used are two NaI inorganic scintillators, which purpose is to detect the two annihilation photons emitted back-to-back. The signal, obtained from the photomultiplier tube and from the preamplifier output of the scintillators, is then driven into the readout chain.
The scintillators are fixed on a support and face a rotating plate where the ^{22}Na source is positioned. The support permits to select different angles between the detectors, within 7.5° steps.
The rotating plate is controlled by an engine through its dedicated software (TMCL) and it has six slits in which the source can be positioned at different distances from the center. This device is useful to simulate a real PET, where the patient is completely surrounded by detectors, through an appropriate change of coordinates (see chapter 7.1).

2.3 Electronic readout chain

The electronic readout chain is composed by:

- **ORTEC 572 Spectroscopy Amplifier**: this module is used to set the signal amplification and the shaping, i.e. the *gain* and the *shaping time*;
- **ORTEC 490A AMP & Signal Channel Analyzer (SCA)**: this module can work as an amplifier or as a SCA, but it is used, during the experiment, as a SCA to select the energetic window (see section 3.2);

- **LeCroy 222 Dual Gate and Delay Generator**: it can produce a standard fast NIM output (approximately 2 nsec rise time) when the incoming signal has an amplitude, which falls in a boundary defined by the user. This module is useful in the coincidence part to open the logic GATE;
- **MCA (Multi Channel Analyzer) TRUMP PCI**: it is inserted inside the computer and it is controlled by the program MAESTRO, which permits to acquire the spectrum of shaped signals coming from the amplifier. The spectrum can be registered on 1024 or 2048 channels.

It is always used also an oscilloscope to check signals, in particular to avoid saturation.

Part II

Characterization

Chapter 3

Scintillators characterization

The setup should be optimized to detect in time the two 511 keV photons produced back-to-back, assuming that the annihilation occurs when the positron and the electron are at rest.
The characterization is focused on:

- the optimization of the readout chain, in particular on the bias voltage, the detector gain and the shaping time;
- the synchronization between the linear and the GATE signals, through the autocoincidence, and the setting of the SCA window;
- the calibration of the geometrical configuration for back-to-back photons.

This work on scintillator calibration is crucial to decide which one should be used as detector and which as a GATE generator during the data acquisition.
Through this entire process and also when the data are taken, the source used is the same (number 34).

3.1 Electronic readout chain

The experimental setup shown in figure 3.1 is used in the first part of the calibration. Each detector is analyzed through this chain to get spectra at different voltages.
In particular, the signal is first shaped and amplified by the preamplifier and its long tailed signal is then conducted through a coaxial cable to the output of the linear amplifier. The signal is again amplified and Gaussian shaped, which width depends on the *shaping time*, and then driven through another coaxial cable to the MCA TRUMP.

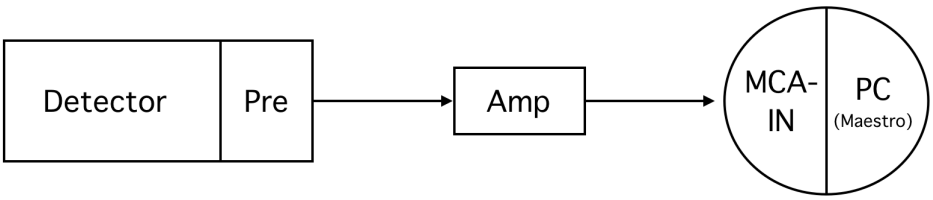


Figure 3.1: *Experimental setup for electronic readout chain calibration*

3.1.1 Bias voltage

Since the purpose of this part is to determine the best working voltage for each detector, the gain and the shaping time should be fixed. In particular the shaping time is set to 1 μs, while the gain is adjusted for each measure in order to maintain the 511 keV full-energy-peak of the spectrum approximately in the same position.
Spectra are taken with a bias voltage from 600 V to over 1000 V and the 511 keV peak is fitted to obtain its full width at half maximum (FWHM) and its mean, the peak centroid position. With these information it is possible to obtain the resolution at every voltage with the formula:

$$Resolution = FWHM/mean$$

The following image (Fig. 3.2) shows the resolution trend depending on the voltage for each scintillator. It can be observed that the resolution improves with the increase of the voltage due to the fact that low bias voltage leads to low amplitude PMT signals. Since the interest is on a condition of stability, it is necessary to move to higher voltage, where, indeed, it can be observed almost a plateau, mostly for number 0 detector.

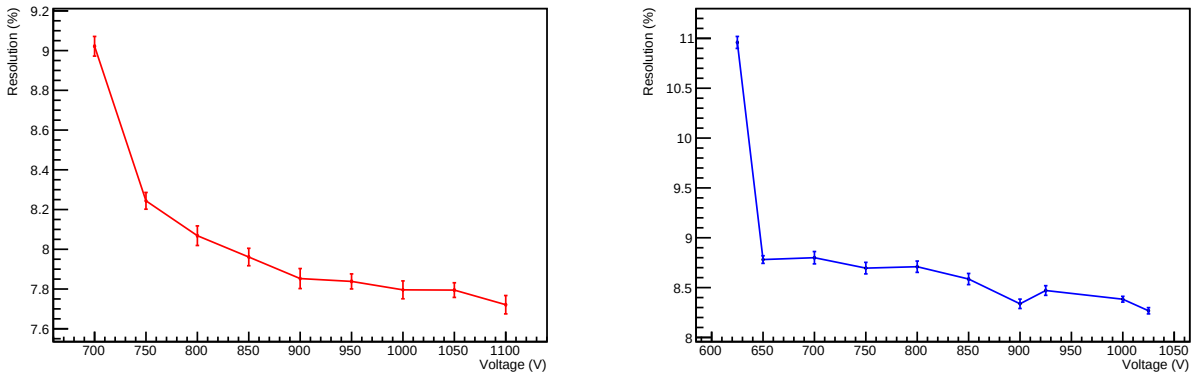


Figure 3.2: *The resolution as a function of the bias voltage for the detector number 1 (red) and number 0 (blue)*

It is also considered the signal rate of the detectors, defined as:

$$Signal\ Rate = peak\,area\,/\,livetime$$

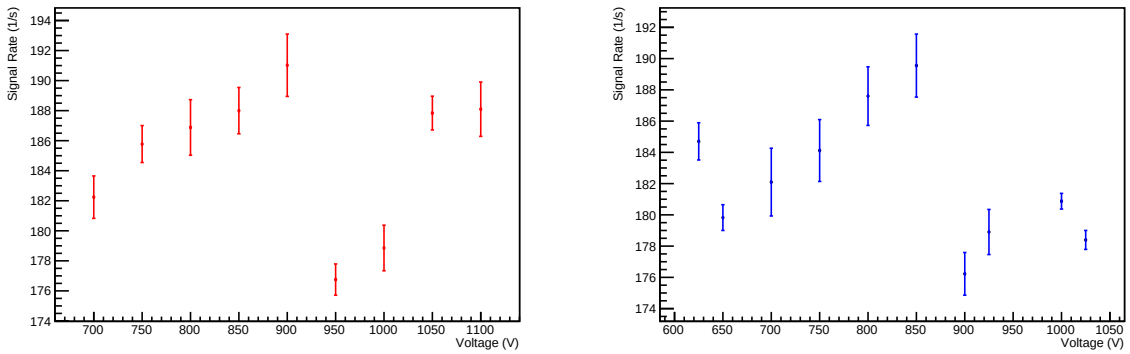


Figure 3.3: *Signal rate as a function of the voltage for detector 1 (red) and for detector 0 (blue)*

In the pictures 3.3 the signal rate is studied as a function of the voltage: it can be observed that the signal rate maximum variations are between 7-9%. The value of the bias voltage is chosen as a compromise between a good resolution and an high signal rate. Thus, the voltage is set to 1100 V for the detector 1 and to 1000 V for the detector 0. Thanks to this analysis, the detector 1 is chosen to be the spectrometer, since it has a better resolution. As a consequence, the detector 0 works as a GATE generator in coincidence measures.

3.1.2 Stability of the electronic

It is important to determine the stability of the readout chain because the focus of the experiment is on long time measures. It is known, indeed, that the gain of the system drifts for hours after the lighting of the power supply, until it stabilizes. This causes an oscillation of the position of the peak, increasing its width. Thus, spectra are taken for weeks checking the centroid of the peak and also the temperature of the environment in order to find a link between them. This analysis is important to set the correct energetic window for the experiment.

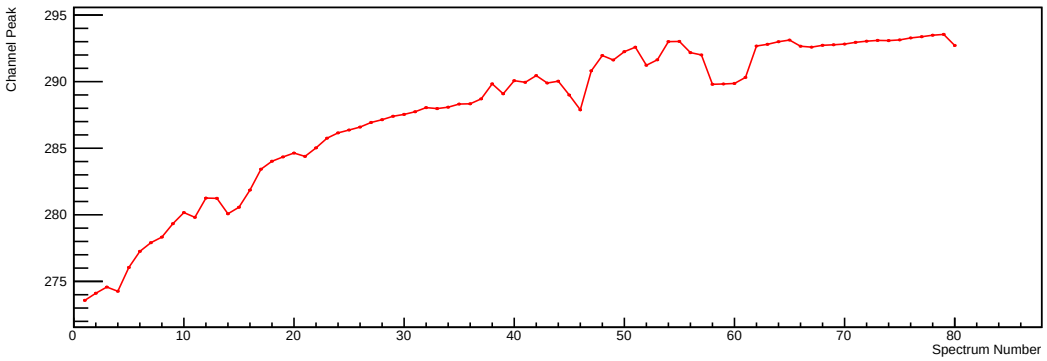


Figure 3.4: *Peak drift over about 40 hours*

The figure 3.4 shows the peak drift during about 40 hours (spectra are taken every half an hour). As it can be observed, the system requires more than a day to stabilize. Even after this period, there can still be a periodic-like gain drift because of the fluctuation of the external temperature during day-night cycle. To study this correlation, spectra are taken for a longer period of time (about 18 days), recording also, through an external thermometer, the temperature.
The picture 3.5, indeed, presents the trend of the gain drift compared to the temperature trend. It can be noticed that there is no periodicity in temperature fluctuations, as at first it could be imagined, and there is also no visible correlation between the drifts of the gain and the variations of the temperature.
Owing to these observations, it is decided to take into account the maximum variation of the peak channel observed in fig.3.5 as a relevant factor to decide the correct energetic window to be used for the next experiment steps (see chapter 3.2). The maximal fluctuation amplitude is about 15 channels (∼5%). This is the benchmark value.

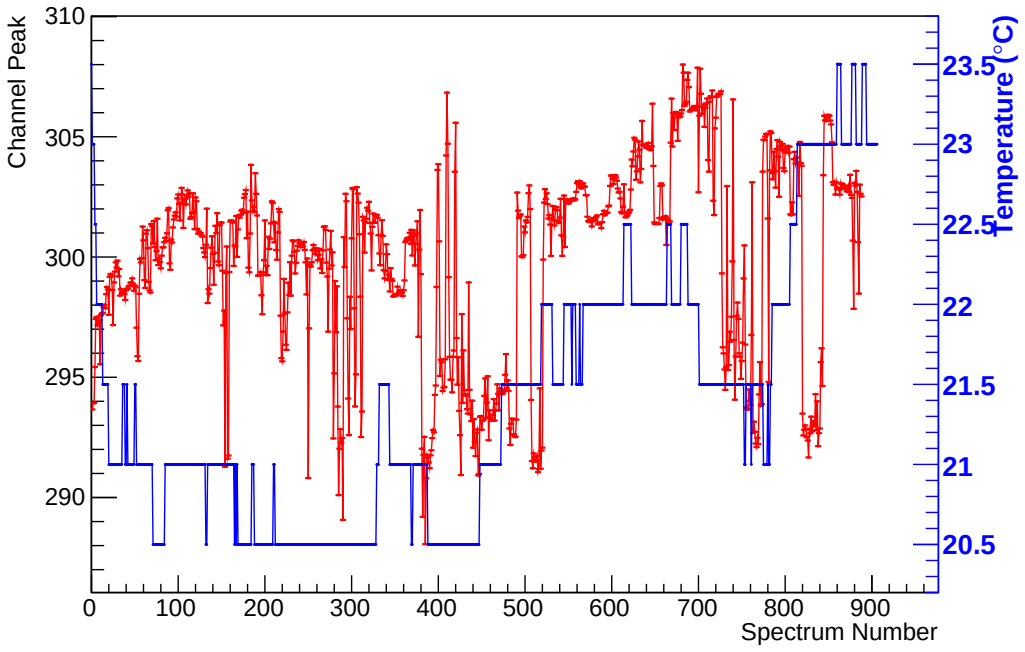
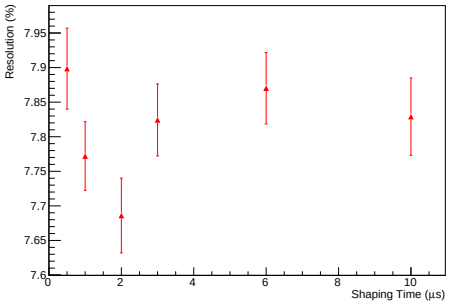


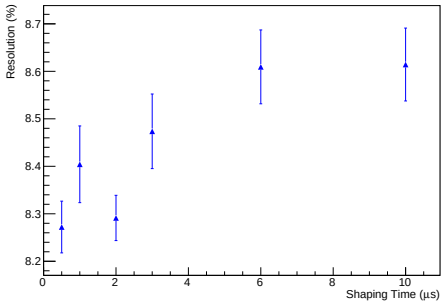
Figure 3.5: *Peak drift in red and temperature variations in blue*

3.1.3 Shaping time

The last feature to consider in this setup is the shaping time (ST) of the linear amplifier. It is analyzed taking spectra varying the ST (between 0.5 μs and 10 μs), keeping fixed the bias voltage, already optimized.
It is expected that the shaping time affects the resolution because it performs a cut on the frequency of the input signal, that has to be optimized when the frequency window is centered on the signal frequencies. Moreover the ballistic deficit and the pile-up effect should be considered. The first degrades the resolution by reducing the shaping time, while the second degrades the resolution by increasing the ST.

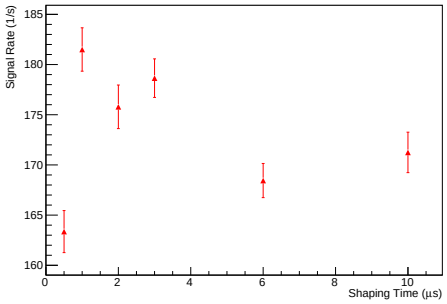


(a) *Resolution as function of the shaping time for the detector 1*

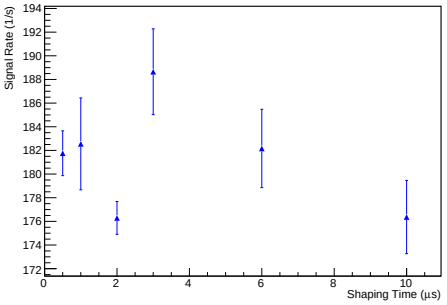


(b) *Resolution as function of the shaping time for the detector 0*

Figure 3.6



(a) *Signal rate as function of the shaping time for the detector 1*



(b) *Signal rate as function of the shaping time for the detector 0*

Figure 3.7

As it can be observed from the fig 3.6a and 3.6b, the increasing of the shaping time affects both detectors resolution of about 4,5% (comparing to the one reported in fig. 3.2). It is still visible, though, that in general the detector number 1 has a better resolution.
For detector 1 a shaping time of 2 μs is chosen, which is the value where it occurs the best compromise between resolution and signal rate. While, for detector number 0, the shaping time choice, necessary to optimize the autocoincidence configuration, is discussed in section 3.2.

3.2 Autocoincidence

To properly set the energetic window on the SCA and the time synchronization between the linear and the GATE signals, it is necessary to change the electronic chain. In particular, only one detector is involved in this part, the one which has been chosen to be the GATE. In this setup the output signal of the detector is used both as input and as GATE for the MCA. It is indeed split into two signals: one sent in the AMP and the other in the SCA. The figure 3.8 shows the electronic chain.

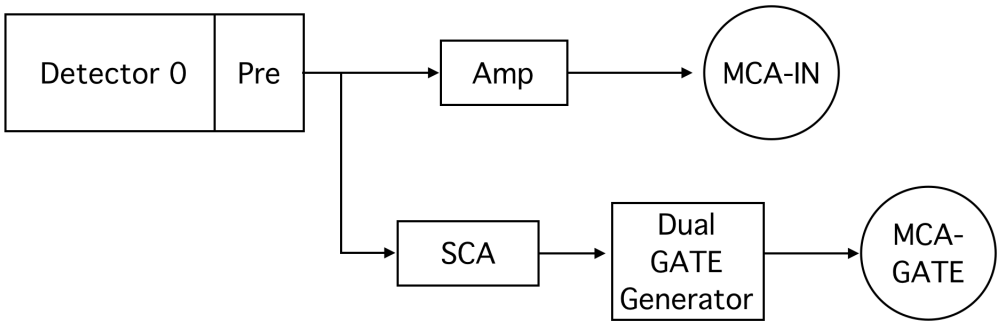


Figure 3.8: *Electronic chain for autocoincidence*

Once obtained the spectrum in this mode, it is necessary to set the energetic window, in particular to define the boundary energetic levels, the lower and the upper one, for the events analyzed by the MCA. In this way, only the events in the 511 keV peak can be selected, leaving out Compton events and the second peak of the ^{22}Na spectrum. A large energetic window increases the dead time of the MCA: it is useful, then, to set a narrow window. Nevertheless it must not be too narrow, because of the drift of the peak. As said in paragraph 3.1.2, the position of the peak drifts with time dependence and this fact brings to enlarge the window.
In detail, it has to be consider:

- the program requirements: MAESTRO asks that the GATE signals are at least 0.5 μs advanced and 0.5 μs delayed from the impulse coming from the amplifier;
- the GATE signal time window: to contain all the signals it is necessary to wide the time window of the GATE signal form 1 μs to 10 μs;
- the peak drift: from 3.1.2 analysis, it has been understood that the peak drifts for a maximum of 15 channels ($\sim$5%);
- the energetic window on the SCA: it is properly set to select mostly the event of the 511 keV peak;
- the shaping time for the Gaussian peak: firstly it is convenient to choose a ST value which guarantees the best resolution (0.5 μs according to 3.6b figure). Nevertheless, after observing, through the oscilloscope, that the Gaussian peak is not centered in the logic GATE window, 2 μs is chosen, the second minimum value of resolution because it permits to better center the peak.

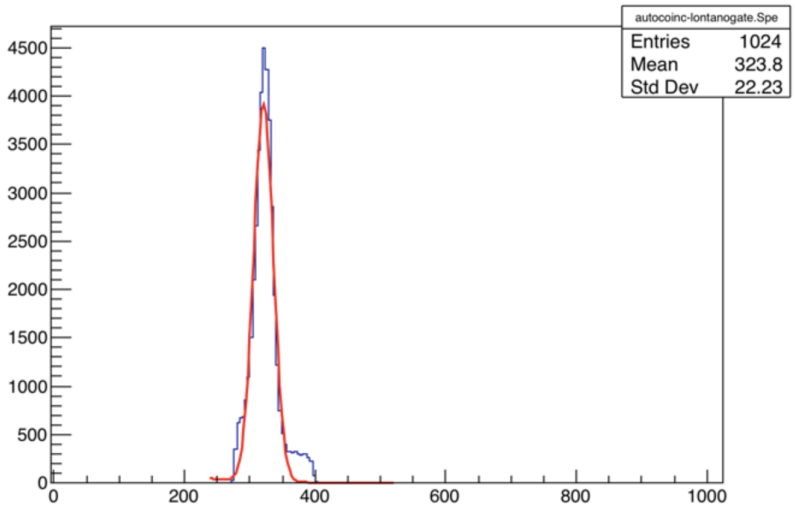


Figure 3.9: *Autocoincidence spectrum. The x-axis corresponds to the rebinned MCA channels*

3.3 Coincidence

After setting the parameters in the autocoincidence system, it is necessary also to study the coincidence setup, which is the one used during the PET campaign. The electronic chain is almost the same as before with the only difference that now both detectors are involved (fig. 3.10).
The focus of this calibration part is on the geometrical configuration, that can be properly set in order to optimize the detector efficiency. The detectors, as already depicted in 2.2, are fixed on a support and face a rotating plate where the source is positioned. In this particular configuration the support is moved so that they could face each other. As it can be seen from the following picture and as already mentioned, the detector 0 is used as the GATE signal generator, while the detector 1 is used as the spectrometer.

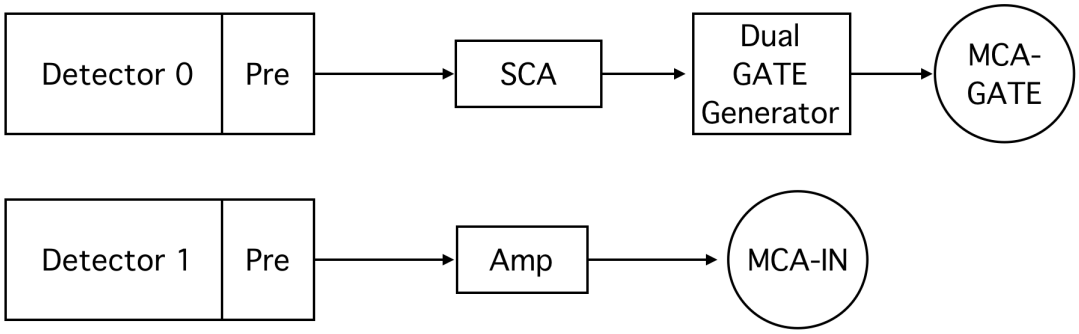


Figure 3.10: *Electronic chain for coincidence*

3.3.1 Alignment of the detectors

Firstly the height of the detectors is studied. The source is placed in the center of the plate, while the two detectors are moved vertically and spectra are acquired.
Initially the height of the detectors is varied keeping them in a collinear configuration, to set the height where the efficiency of detection (signal rate) is at its maximum.
Then the detector 0, the GATE, is kept fixed at the height obtained from the previous configuration and the other moved vertically to check that the best detection actually occurs when they are collinear and aligned to the source.
In figure 3.11 a maximum, which corresponds to the height where the scintillators are aligned with the source, can be seen and in figure 3.12 it is proved that the maximum occurs when the detectors and the source are on the same axis.

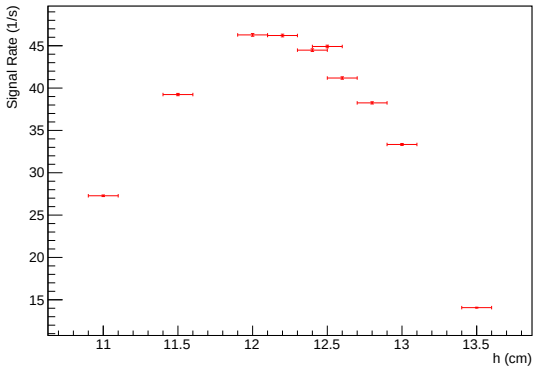


Figure 3.11: *Signal Rate as a function of the height when both detectors move*

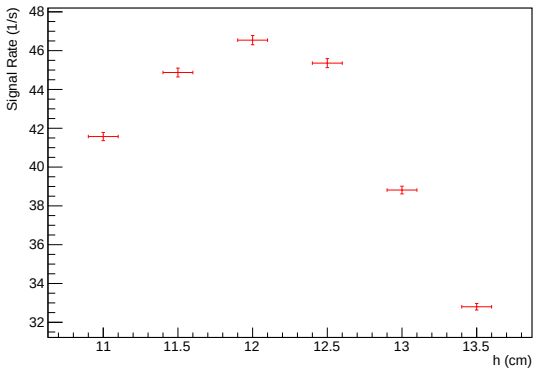


Figure 3.12: *Signal Rate as a function of the height when only detector 1 moves*

The horizontal alignment of the detector is also roughly checked for completeness and it is observed that the best detection occurs when the scintillators are facing each other, as expected. Anyway this analysis is not reported because it is not a useful tool, since PET measures are performed varying the angle between the detectors.

Chapter 4

Rotating plate characterization

As mentioned in paragraph 2.2, the source is placed on a rotating plate, while in a real PET system the patient is fully surrounded by detectors. The plate movements are controlled by a PC interface called TMCL, which permits also to create personalized routines and to download them on the module. Once the position of the scintillators is set, the plate rotates at defined positions and data are acquired in coincidence. Since this tool has a central role in data acquisition, it is really important to perform a complete study of this system.
While testing the device, some issues emerged and it is important to report them.
Firstly, when it changes direction, it takes few seconds to move again and, during this time interval, the internal counter runs the steps. As a result, if the prescription is, for example, to come back to zero after a rotation, it does not reach the point, stopping some steps before (approximately 20000). This happens only during changes of directions but to be sure to avoid it, the routine for data acquisition starts with a preliminary part focused on preventing this situation. More details can be found in section 4.3.
The second comment is about the time counter. It seems indeed that it loses linearity for long time acquisitions. This problem can be solved dividing longer counting times in shorter ones, using iterative cycles.

4.1 Rotating plate parameters

To find the optimal rotating plate parameters, the velocity and the angular acceleration, TMCL provides a tool called *StallGuard Profiler*, which scans a given range of velocities and shows which is the best in relation to the load.
Looking at the profile, the value 100 corresponds to a green area, which indicates that there is effectively no vibration of the motor at that velocity. Owing to this analysis, 100 is chosen as velocity value and the angular acceleration is set to 50. These settings are fixed now and used during all the data acquisition.

```
SAP 4, 0, 100  //Set Velocity
SAP 5, 0, 50  //Set Acceleration
```

4.2 Rotating plate linearity

The rotating plate bases its movements on *steps*. In particular a 360° angle corresponds to 160000 steps. Since the focus of the experiment is on angular measurements, it is necessary to study if there is a linear relation between angles and steps.
To this purpose it is not used a routine; the plate, instead, is controlled directly by TMCL interface. Using the function *move to position*, it is rotated of 10000 steps from the absolute position 0, that has been saved before, to 160000. At every position the angle between the source line and the detector 0 is obtained through a trigonometric relation between the radius of the plate and the position of the source.

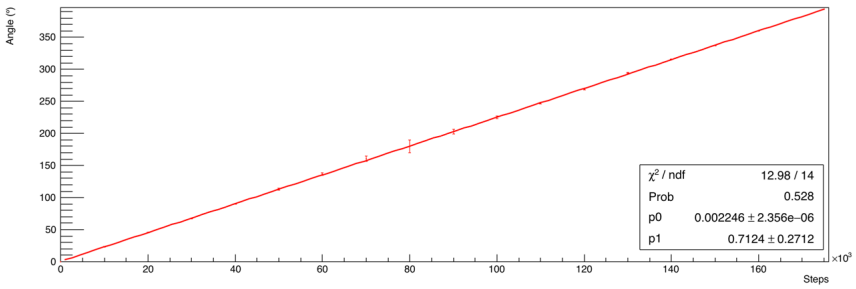


Figure 4.1: *Linear relation between angles and steps*

The figure 4.1 shows clearly that the relation between steps and angles is linear.

4.3 Sub-routine for data acquisition

Once studied in detail the rotating plate characteristics, it is possible to create the routine necessary to data acquirement. This routine is implemented to avoid complications due to the issues reported in the introduction to this chapter: the changing of direction and the time counting.
Firstly, the parameters are set with values previously commented in section 4.1; moreover some thresholds are set up, for example for stall detection:

```
SAP 205, 0, 6       //Setup StallGuard
```

A sub-routine is then implemented to prevent the problem in changing direction. The plate is moved of 50000 steps clockwise and then 30000 in the opposite direction, the same of data acquisition.

```
MVP ABS, 0, -50000
WAIT POS, 0, 0
MVP ABS, 0, -20000
WAIT POS, 0, 0
CSUB WaitOneMeasure
```

Lastly, it is necessary to solve the problem of the time counter. It is, therefore, declared a variable `TotalTime`, which is the time duration of each acquisition, of about 30 minutes, and a main routine that iteratively moves the plate of 1600 steps ($\sim 3.6°$)[1] is created; it is then entered a sub-routine `WaitOneMeasure`, made of another "loop", which counts one minute (1 tick = 10 milliseconds) and compares the time counted, also saving it in EEPROM memory, with the variable `TotalTime`. If it is less than `TotalTime`, the time counter refreshes. Doing this procedure the time passed from the start of acquisition is saved in a variable external to `OneMeasureLoop`, therefore the time counter can be reset, avoiding the problem reported above.
After 30 minutes of counting, the plate moves of other 1600 steps and so on, till it does a full circumference (160000 steps).

```
TotalTime = 30 // Duration of the acquisition (30 minutes)

MainLoop:
MVP REL, 0, 1600
CSUB WaitOneMeasure
GAP 1, 0
COMP 160000
JC LT, MainLoop

WaitOneMeasure:   WAIT POS, 0, 0
                  SGP 0, 2, 0 // Initialize timer to zero
                  OneMeasureLoop: WAIT TICKS, 0, 6000
                                  GGP 0, 2
                                  CALC ADD, 1
                                  AGP 0, 2
                                  GGP 0, 2
                                  COMP TotalTime
                                  JC LT, OneMeasureLoop
                  RSUB
```

The full routine is written in appendix B.
To acquire spectra is, therefore, necessary a synchronization between the routine and the acquisition program (*Maestro*). To this purpose a program to automatize also Maestro acquisitions is written. It can be found after the routine in appendix B.

[1] At the beginning of data acquisition this value is set to 800 steps. This change is due to the fact that is better, for data analysis, to have a lot of shorter acquisitions than a few of longer ones

Part III

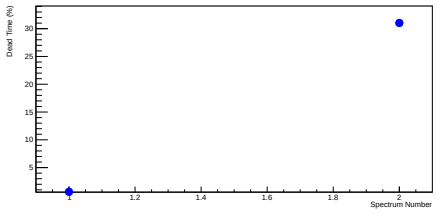
Data Analysis

Chapter 5

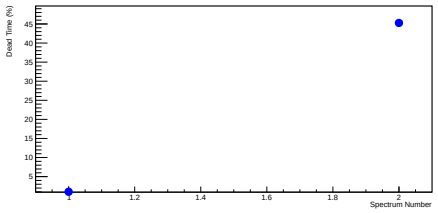
Preliminary Considerations

5.1 Dead time

The dead time is a parameter that has to be controlled in order to keep it as low as possible, because a high dead time causes a loss of events. When the source is too close to the spectrometer, the system paralyzes because there is an overlapping of events. The following graph shows this effect in the worst possible configuration of this system (pos5, see figure 6.2).



(a) *Dead time in pos5 in autocoincidence set up*



(b) *Dead time in pos5 only using spectrometer*

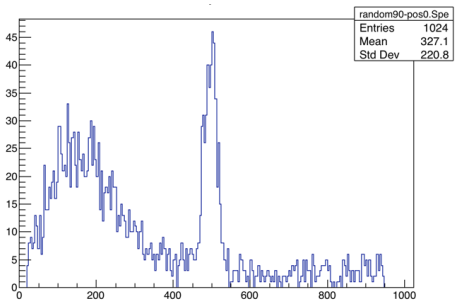
Figure 5.1

The picture 5.1a depicts two points, respectively where the source is close to the detector and far from it, in an autocoincidence set up. Similarly, picture 5.1b shows the two positions of the source, acquiring a spectrum only through the spectrometer (not an autocoincidence setup). As expected, in both cases, the dead time increases when the source gets close to the spectrometer.
The maximum dead time measured in this position is 45%. Changing the source slit, it is observed that the dead time decreases considerably.

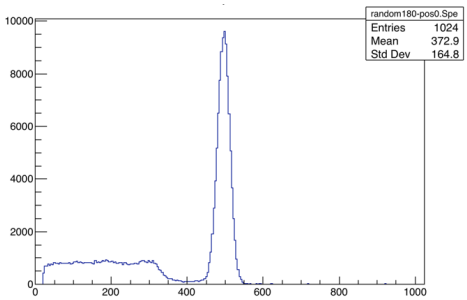
5.2 Random coincidence

In every coincidence system problems related to random coincidences occur, namely when the system records an event as a coincidence when it is not. This happens, for example, when the gate is open and a background signal is detected.
Therefore it is useful to evaluate the random coincidences rate. To this purpose, spectra have been acquired in different configurations. Firstly, the configuration with an angle of 90° between the detectors and the source in pos0 (see figure 6.2) is tested. The expectation

is, then, not to obtain counts from real coincidences, but instead only contributes from random coincidences. On the other hand, a spectrum with an angle of 180° between the detectors, with the source in the same position, is also acquired. Since in this situation the rate of real coincidences should reach a maximum, the two situations are compared.



(a) *Spectrum obtained with an angular distance between the detectors of 90° and with the source in pos0*



(b) *Spectrum obtained with an angular distance between the detectors of 180° and with the source in pos0*

Figure 5.2:

It is observed that the signal rate in the maximum for figure 5.2a is 1.1 s^{-1}, while for figure 5.2b is 149.79^{-1}. It is clear that random coincidences are negligible, they represent indeed $\sim 0.73\%$ of the total counts.

Chapter 6

Geometrical Reconstruction

Geometrical reconstruction is the first type of reconstruction used and it is a more intuitive one, based essentially on trigonometrical relations between some defined angles. These designed observables are depicted in figure 6.1. In particular, α is defined as the supplementary angle of the one described by the source before crossing the LOR (Line Of Response), when detector 0 is at its original position, opposite to detector 1 ($\gamma = 0°$). On the other hand β is the angle described by the source before crossing the LOR, when detector 0 is moved from its original position by a $\gamma \neq 0°$ angle. Lastly, ρ is the distance of the source from the center of the rotating plate, which is the quantity to be found, and R is the distance between each detector and the center of the plate.

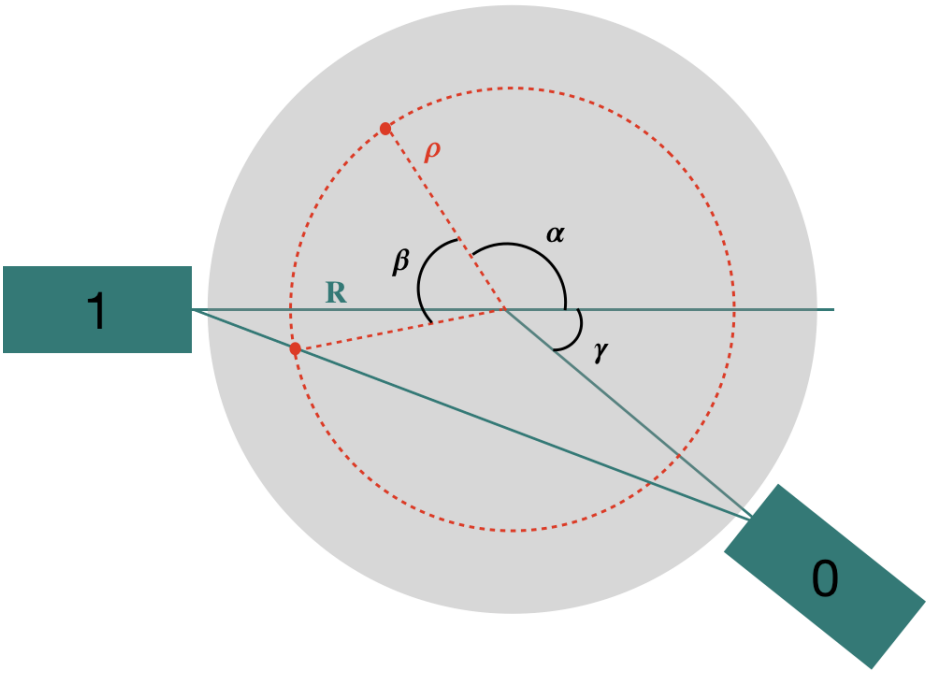


Figure 6.1: *Geometrical configuration with emphasis on designed observables*

To acquire the needed data, the source can be positioned in 6 different slits, as pictured in figure 6.2. After the placement of the source, the routine (described in appendix B) begins, synchronized with Maestro. When all the spectra are acquired, they are analyzed through the macro in appendix A to obtain a signal rate distribution as a function of the angular position (see figure 6.3). Fitting this distribution is essential to calculate the angular position of the source. In particular, every distribution presents two peaks due to the fact that the source always crosses the LOR two times.

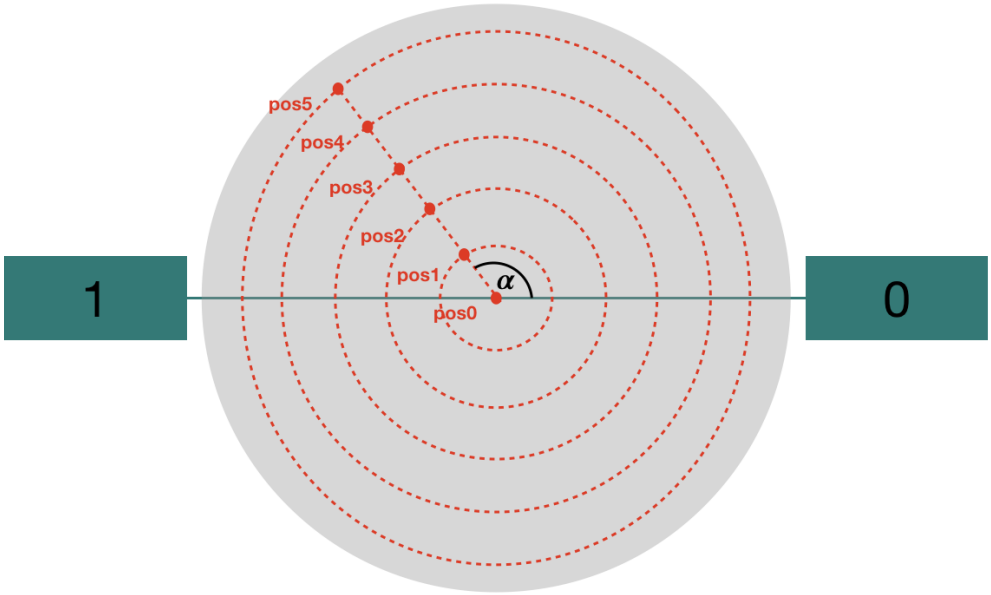


Figure 6.2: *Geometrical configuration with emphasis on different positioning slits*

The peaks, showed in figure 6.3, are both fitted with a Gaussian. To estimate the error, the σ of the Gaussian is considered.
This distribution seems to work well for the acquired data. If the detectors are point-like, the signal rate graph will present a very narrow peak in the positions where the source crosses the LOR. Instead, having a width of centimeters ($\sim 6\,\text{cm}$), the signal rate increases also before and after the maximum, spreading the peak.

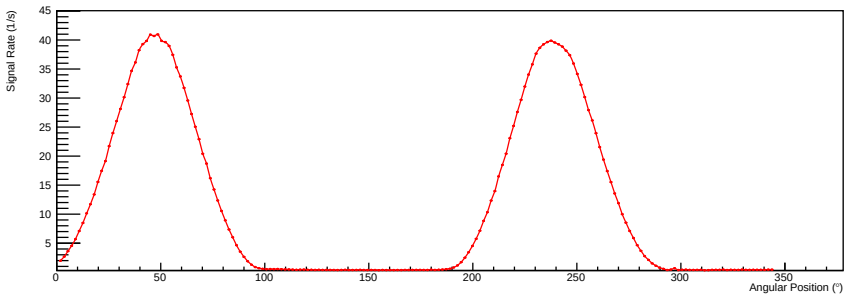


Figure 6.3: *An example of distribution to obtain α with the source in position 1*

6.1 α measures

To measure α, the angle between the detectors is set to 180°, which means γ equal to zero (as in figure 6.2), and spectra are taken varying the position of the source in the slits.
In particular, α is measured setting the source in positions 1, 2, 3 and 4. To follow, the obtained results:

Position	α_1 (°)	$\sigma_{\alpha 1}$ (°)	α_2 (°)	$\sigma_{\alpha 2}$ (°)
pos1	132,66	20	122,51	21,5747
pos2	131,28	10,5744	119,67	10,4232
pos3	132,24	10,4488	124,82	10,7956
pos4	131,45	8,02595	125,26	8,11991
pos4	128,97	6,6316	122,11	6,49673
pos4	129,38	6,61016	122,62	6,46132

Figure 6.4: *Measures of α obtained from the analysis*

In particular, the derived mean value is:

$$\overline{\alpha} = (126.55 \pm 2.47)^{\circ}$$

This value is obtained mediating over all measures, weighting according to their errors.
It is observed that there is no agreement with the expected value of about 135° (this value depends on the implemented routine described in section 4.3).
It is also relevant to report that the angular distance between the two peaks has to be 180° but all data samples present bigger values. In particular, the first α peak seems shifted of about 6° from the expected value, while the second α peak presents a shift of 12°. The shifting relation seems, indeed, linear. It is, then, necessary a more thorough approach to this problem, which is therefore analyzed in the next section.

6.2 The systematic error

To study this systematic error, firstly α angles are analyzed and then the hypothesis of the linear shift is also tested on β ones.
The graph 6.5 depicts on x-axis the angle $(\pi - \alpha)$ described by the source from its starting position and on y-axis the difference between the expected value of this angle and the one obtained from the data acquisitions. In particular, the difference between expected and measured is only known with precision when the source crosses the LOR. Indeed, two points for each measures of α are reported. Each pair of points is then connected with a line, to underline the linear relation between them.
Moreover, it can be observed that there are two different types of data: in blue, the measures of α obtained when the source is in pos4 (the ones actually used in the further analysis steps), while in red, the measures of α with the source in position 1, 2 and 3. Since, this second set presents unrelated measures, only the first, the set of all position 4, is fitted to obtain the linear shifting relation.

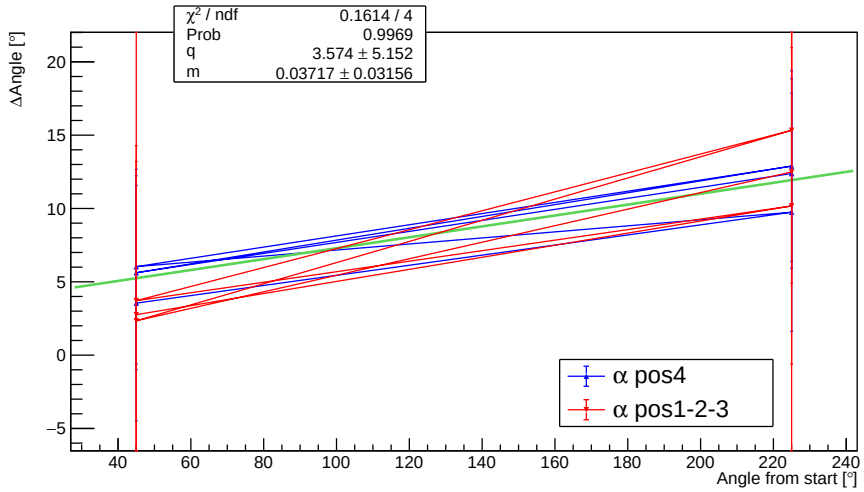


Figure 6.5: *Extrapolation (in green) of the shifting relation from α measures*

The extrapolated relation, depicted in green line in figure 6.5, is then applied to β measures, represented by the blue squares in figure 6.6. This graph is built exactly like the one above, reporting the variation from the expectation value of the angle on y-axis and the evaluated β angle on x-axis.
As it can be observed, applying the obtained shifting relation (black dots) to β measures leads to cancel the shift. This fact proves that this shift is, indeed, a systematic error because the correcting relation works for two set of unrelated measures (α and β). Nevertheless this is not surprising, actually it is perfectly reasonable if the assessment of the systematic error is correct, because almost the same configuration apparatus is used to measure both observables.

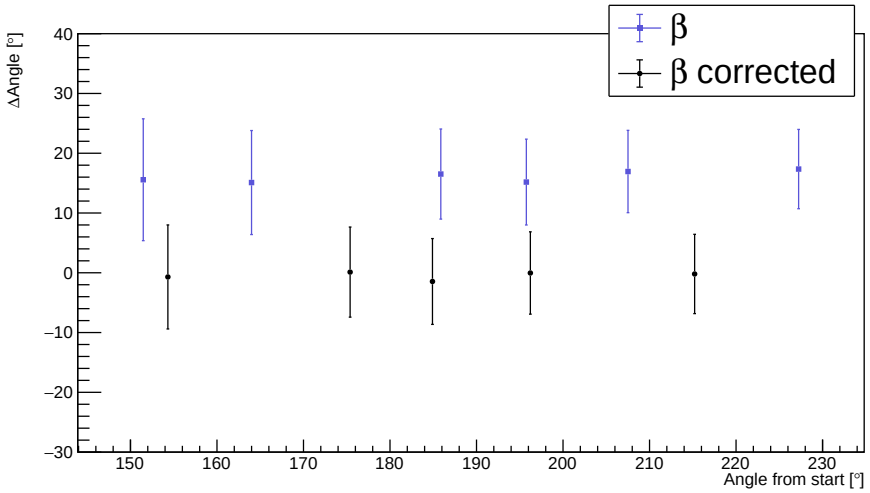


Figure 6.6: *Shifted β angles in blue, corrected β angles in black*

Furthermore, the extrapolated function is used to correct α values and to follow, the results are reported:

Position	α_1 (°)	$\sigma_{\alpha 1}$ (°)	α_2 (°)	$\sigma_{\alpha 2}$ (°)
pos1	137,99	20	134,91	21,5747
pos2	136,67	10,5744	132,18	10,4232
pos3	137,59	10,4488	137,14	10,7956
pos4	136,83	8,02595	137,55	8,11991
pos4	134,44	6,6316	134,53	6,49673
pos4	134,84	6,61016	135,015	6,46132

Figure 6.7: *Measures of α after the correction*

It can be observed that, as expected, the mean value of α, obtained after the correction, agrees with the expectation:

$$\overline{\alpha} = (135.48 \pm 2.47)^{\circ}$$

All the following reported measures for the others observables are obtained correcting this systematic error. However, the sinogram and the corresponding reconstruction, derived from the unadjusted data, are recorded to observe the interesting effect of this systematic error in estimating the source position on the plate (see section 7.2).
It is important to understand why this problem occurs. A lot of hypothesis are proposed, mostly regarding the geometrical configuration (as for example parallax error and γs not perfectly collinear). Nevertheless, another option is taken into account: Maestro job and measurements routine could be not synchronized.
To test the possibilities, α measures are taken for two complete rotations of the plate. Indeed, it is observed that the shift of the peaks is not periodical but increases linearly

with the rotation. This proves that the systematic error does not depend on the geometrical configuration. Since it has already been demonstrated that the rotation of the plate preserves linearity between steps and angles, it is understood that the problem is due to `OneMeasureLoop` subroutine (see section 4.3), which requires an additional ~ 1.7 seconds to restart every minute; this leads to an increasing linear offset. For 100 spectra of 30 minutes each, the total delay is about 85 minutes; this effect leads to the loss of almost 3 spectra, corresponding to about 10 degrees shift. This delay is consistent with the systematical error that has already been estimated.

6.3 β measures

To measure β, it is necessary to move the detector 0 from its position by a γ angle. Various γ angles are tried but also various positions of the source. Nevertheless, it is observed that, in some cases, the two peaks of the obtained spectra merge together. This is due to the fact that the source crosses the LOR in two points that are different but really close.
Before studying this situation, it has to be decided the position of the source. It is put in pos4. From the consideration above, indeed, it is easy to observe that the positions, closer to the center of the plate, correspond to smaller circles traveled by the source and so to a major probability to end up in that "tangent" LOR case, or even worse, the source can not cross the LOR at all. However, in position 5 the dead time, when the source is right in front of the detector, is too high (see section 5.1). Position 4 seems the best compromise between the two conditions.
Focusing now on the fact reported above, it is useful to understand the borderline case conditions, to avoid pointless measurements. First of all, it is observed that in general the angular width of a peak is $\sim 36°$. To fit correctly the two peaks is necessary to see both of them well defined, as in figure 6.8. To find the corresponding condition of γ angle, some trigonometric relations are used:

$$h = R\,sin(\gamma/2) \qquad \rho = h/cos(\epsilon)$$

$$\gamma = 2\,arcsin\left(\frac{d\,cos(\epsilon)}{R}\right)$$

Imposing $\epsilon = 36°$, γ limit value is about $65°$, so if γ has a greater value the two peaks will start to merge and it would be difficult to fit them. The R value here considered is the radius of the disc, which is smaller than the real interaction radius. For this reason the γ value is a little overestimated, but for the purpose of this analysis is enough precise. Anyway, an attempt to measure the real point, in which interactions occur, is done in the next section.

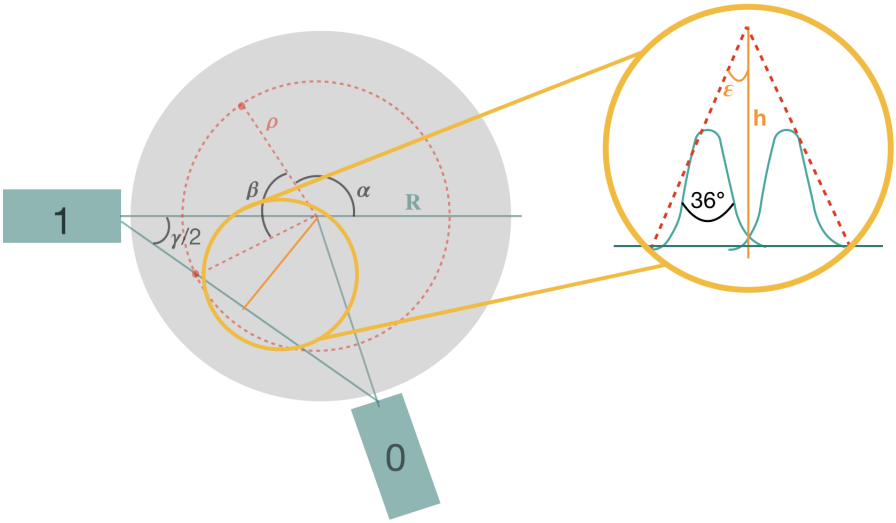


Figure 6.8: *Configuration with a focus on the LOR crossing geometry*

To follow, the results obtained for β:

γ (°)	Position	β_1 (°)	$\sigma_{\beta 1}$ (°)
7,5	pos4	47,1	6,63302
15	pos4	48,87	6,70629
22,5	pos4	51,24	6,88635
30	pos4	53,63	7,17959
37,5	pos4	57,22	7,53636
52,5	pos4	62,48	8,70097
60	pos4	67,76	10,1937
15	pos1	82,68	20,69
15	pos1	79,54	20,5028
15	pos5	45,84	6,27411

Figure 6.9: *Measures of β obtained from the analysis*

6.4 R measures

The last observable that enters in ρ formula is R, the distance between the center of the plate and the detectors. Nevertheless it is not obvious that the interaction occurs right on the surface of the detectors. Thus, it is not sufficient to measure the distance between the detectors, it is instead necessary to try another approach.
R can be obtained inverting the relation that is then used to calculate ρ, because it is easier to measure ρ radius, since it is the distance of the source from the plate center:

$$R = \frac{\rho\,(-sin(\alpha+\beta) - tan\left(\frac{\gamma}{2}\right)\,cos(\alpha+\beta))}{tan\left(\frac{\gamma}{2}\right)}$$

The source is so positioned in two different slits at a known distance from the center: $p1 = 3.3$ cm and $p4 = 13.2$ cm. These two particular positions are chosen because they represent two selectable borderline conditions. Position 5, indeed, can not be chosen because of the high dead time value.
When the source is closer, the photons seems to travel deeper in the crystal and R is slightly bigger, while, in the opposite case, the γs interacts near the surface of the crystal. In this way, it is obtained an R value mediated on these two situations.
To follow the results for R measures:

γ (°)	Position	R (cm)	σ_R (cm)
15	pos1	18,08	6,44942
15	pos1	17,09	6,7358
52,5	pos4	20,81	3,38087
60	pos4	21,15	2,89678

Figure 6.10: *Measures of R obtained from the analysis*

Again, all measures are mediated, weighting on their errors, to obtain the value of the distance between the center of the plate and the interaction point:

$$\overline{R} = (20.37 \pm 1.97)\,cm$$

Considering that the distance between each detector and the center of the plate is 18 cm and that the length of the crystal is about 5.3 cm, the obtained value is reasonable and in agreement with measurements.

6.5 ρ obtained value

The last step of geometrical reconstruction is obtaining ρ, the distance of the source from the plate center. Having already computed α, β and R values, it can be derived using this formula:

$$\rho = \frac{tan\left(\frac{\gamma}{2}\right) R}{-sin(\alpha+\beta) - tan\left(\frac{\gamma}{2}\right) cos(\alpha+\beta)}$$

This formula is discussed in appendix D. To follow the results for ρ measures:

γ (°)	Position	ρ (cm)	σ_ρ (cm)
7,5	pos4	12,1	13,5545
15	pos4	12,95	7,80335
22,5	pos4	12,88	5,22108
30	pos4	12,9	4,01896
37,5	pos4	12,55	3,0879

Figure 6.11: *Measures of ρ obtained from the analysis*

The observations, about the derivation of the others mean values, are valid also in this case and so the obtained ρ value is:

$$\overline{\rho} = (12.72 \pm 2.11)\, cm$$

The distance, between the center of the plate and the source, measured in laboratory is 13.2 cm, therefore the two values are in agreement.
It is interesting to notice that to evaluate R and ρ, different measures are used. The reason for this choice is to avoid correlations between the two results.

Chapter 7

Image Reconstruction

Since geometrical reconstruction is more a proof of concept than a precise reconstruction method, this section is focused on analyzing acquired data using more sophisticated techniques. In first place, the standard retroprojective approach for image reconstruction is adopted.
This approach is based on a mathematical concept called *Radon Inverse Transform* which is an analytical method used to reconstruct three-dimensional and two-dimensional objects through their projections.
The *Radon Transform* is defined as:

$$R_\phi(S)[f(x,y)] = \int_{-\infty}^{\infty} f(x,y)\delta(S - xcos\phi - ysin\phi)dxdy$$

where ϕ and S are the observables depicted in figure 7.1 and $f(x,y)$ the function of interest (in CT the attenuation coefficient $\mu(x,y)$, in PET the activity profile of the radioactive source).
Using Radon Inverse and having hypothetically an infinite number of projection at an infinite number of angles, it is theoretically possible to fully reconstruct the object analyzed ($f(x,y)$). Since this is an ideal situation, a lot of algorithms have been developed to optimize the reconstruction results.
One of the easiest way to approximate the Radon Inverse mechanism is through retroprojection. This method, indeed, consists in discretizing the area to be reconstructed using a grid and retroprojecting the obtained data along the correct direction.
Python is the programming language chosen to implement the reconstruction algorithm because it provides a useful package called *scikit-image*, a collection of algorithms for image processing.

7.1 Changing coordinates

In order to apply the retroprojection algorithm, the position of each measure in the (S,ϕ) coordinates has to be determined. In particular, S is the distance of the LOR from the center of the plate and ϕ is the angle between the line perpendicular to the LOR and the x-axis. In the end, $\theta(t)$ is the coordinate that represents the movement of the detectors during the time of the acquisition.

Changing coordinates can be done using some simple trigonometric relations

$$\sigma = \frac{\pi}{2} - \frac{\gamma}{2} \qquad \pi = \phi + \gamma + \theta(t) + \sigma$$

to obtain ϕ and S:

$$\phi = arctan\left(\frac{cos\left(\frac{\gamma}{2} + \theta(t)\right)}{sin\left(\frac{\gamma}{2} + \theta(t)\right)}\right)$$

$$S = R\,cos\left(\phi + \theta(t)\right)$$

These coordinates are computed for every data sample.

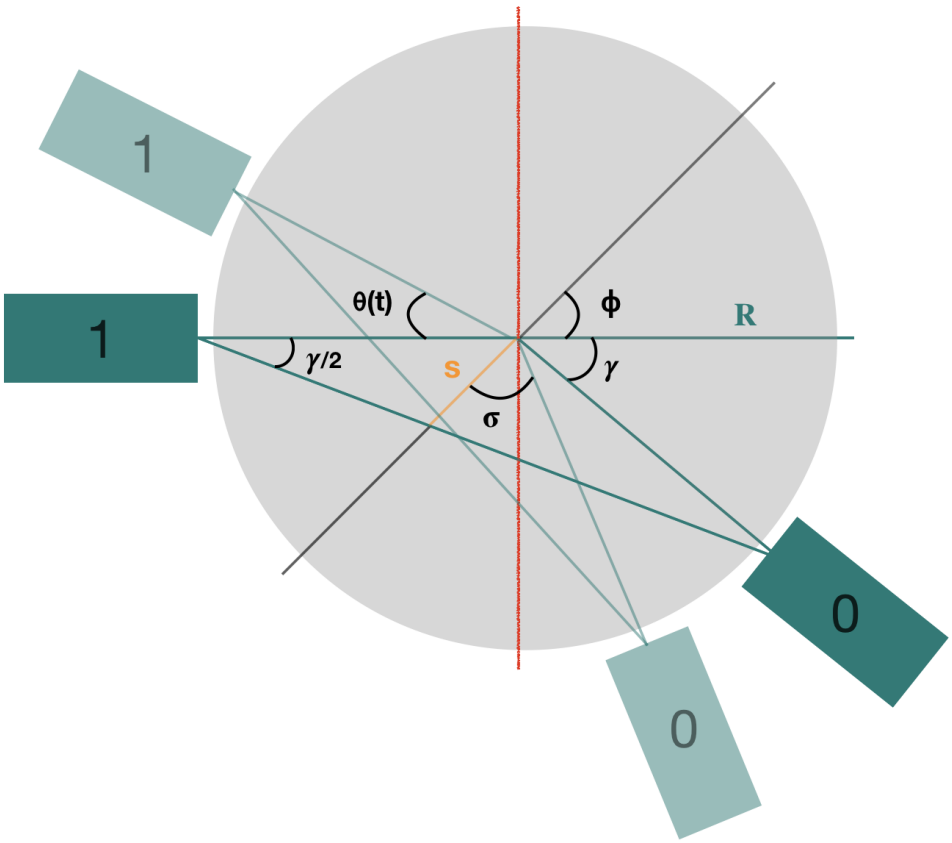


Figure 7.1: *Observables involved in changing coordinates*

7.2 Sinogram

The sinogram in figure 7.3 is a two-dimensional histogram of the signal rate in the (S, ϕ) space.
Each sample, that corresponds to a fixed γ data acquisition, is composed by two values of S: $-|S|$ and $|S|$. The source, indeed, crosses the LOR two times. From the definition of S reported in the paragraph above, it can be observed that it changes sign when it crosses the red line, as in figure 7.2a. As a consequence, the two peaks correspond to S with different sign. The figure 7.2b instead depicts the change of sign of ϕ variable. For each of the two values of S, ϕ varies from 90° to -90°, with steps of 3.6° (the motor step).

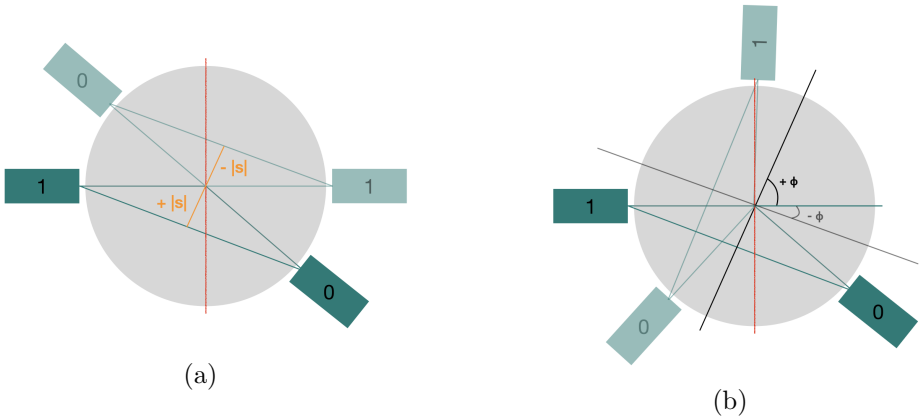


Figure 7.2: *Focus on the sign of S and ϕ variables*

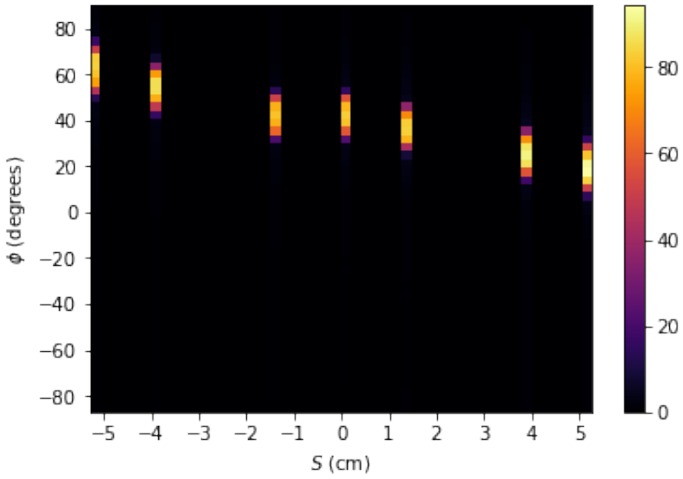


Figure 7.3: *Sinogram* (S, ϕ)

Owing to the external position of the source and the periodic definition of ϕ and S, not every data sample acquired is reported in the sinogram 7.3. In fact, as shown in figure 7.4, values of γ over 45° lead to an inevitable translation of some ϕ values from the region $-|S|$ to the $|S|$ one. This considerably affects the reconstructed image with artifacts. For this reason acquisitions with ϕ over 45° are not reconstructed.

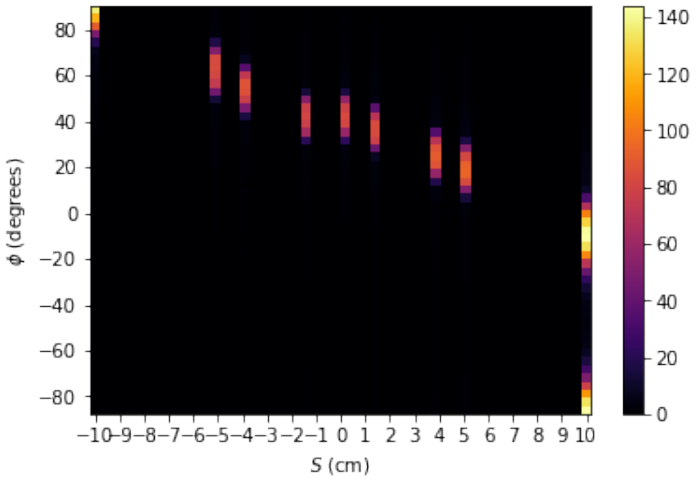


Figure 7.4: *Sinogram including the acquisition with* $\gamma = 60°$ *(*$|S| = 10.7$*)*

Since the geometrical systematic error, that has been discussed in section 6.2, affects the measure of the angle position of the source on the plate, a correction of the angle θ is necessary. Changing θ leads to a correction of ϕ ad S coordinates, because of the relations reported in section 7.1.
The sinogram with the correction of the systematic error is reported in figure 7.5.

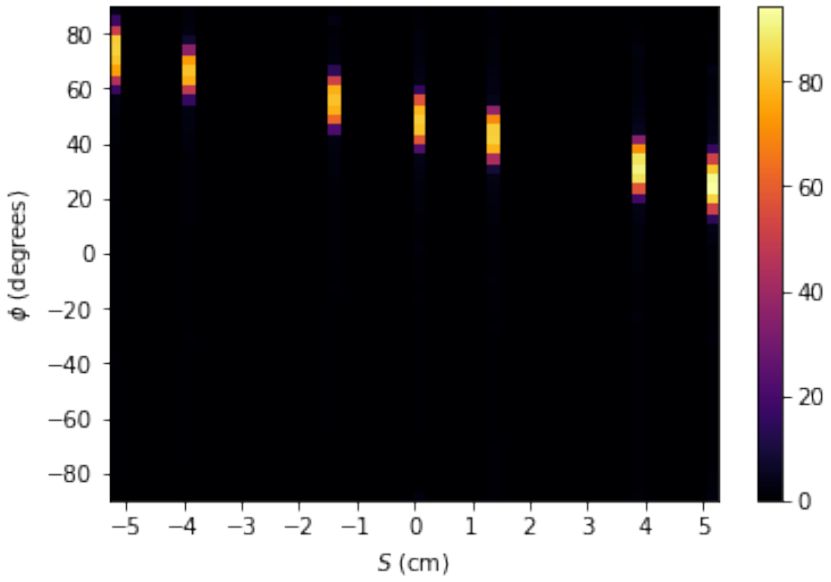


Figure 7.5: *Sinogram after the systematic error correction*

7.3 Filtered Back Projection reconstruction

Applying the scikit-image function *iradon* to the sinogram in figure 7.3 (more details in appendix C), the reconstruction in figure 7.6 is obtained. The filter chosen for the reconstruction is the commonly used *Shepp Logan Filter*.
The color scale used in the following reconstructed images is the same adopted in the sinograms.

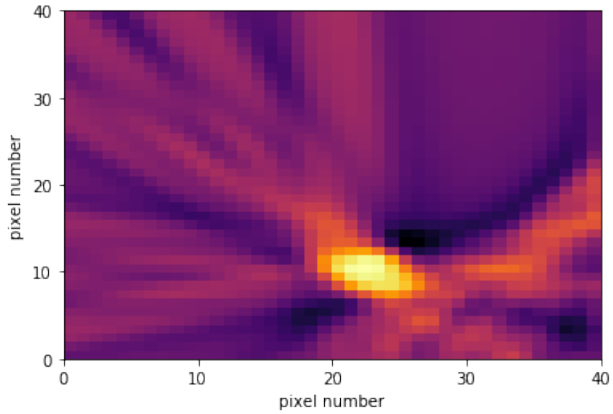


Figure 7.6: *Filtered Back Projection reconstruction of the data affected by the systematic error (40x40 pixels)*

In order to reconstruct the data corrected from the estimated systematic error, the function *iradon* is also applied to the sinogram in figure 7.5. The result is shown in figure 7.7.

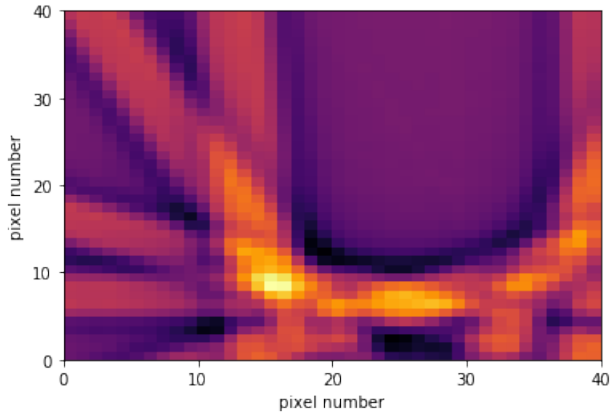


Figure 7.7: *Filtered Back Projection reconstruction of the corrected data (40x40 pixels)*

From figure 7.7 a translation of the position of the reconstructed source is clearly visible. This translation, as discussed in section 7.5, leads to a correct evaluation of the position source thanks to the fact that the systematic error has been corrected. Although, the image 7.7 shows a widening of the high activity region, on account of the fact that the extrapolated correction represents only an average behavior.

7.4 SART reconstruction

The second method for image reconstruction used is the Simultaneous Algebraic Reconstruction Technique (SART), provided by the scikit-image function *iradon_sart* (more details in appendix C). The algorithm is iterative and can be considered as an iterative solver of a system of linear equations. Applying the *iradon_sart* to the sinogram 7.3, the reconstructed image 7.8 is obtained.

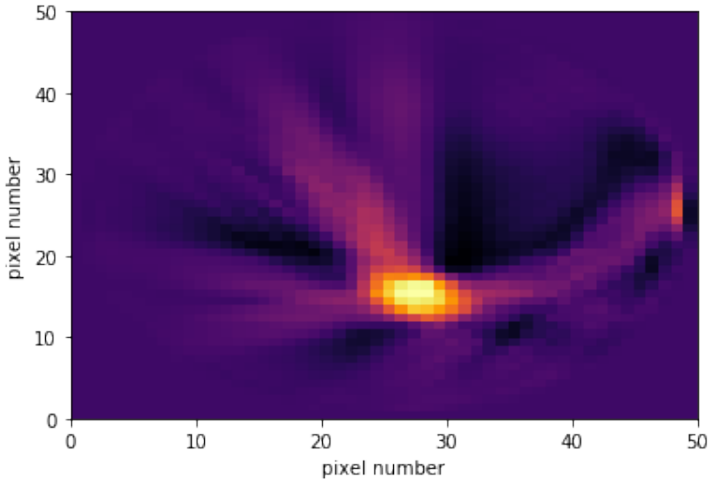


Figure 7.8: *SART reconstruction of the data affected by the systematic error (50x50 pixels)*

As it has been done in section 7.3, in order to obtain a reconstructed image corrected from the systematic error, the function *iradon_sart* is also applied to the sinogram 7.5. The result is shown in figure 7.11.

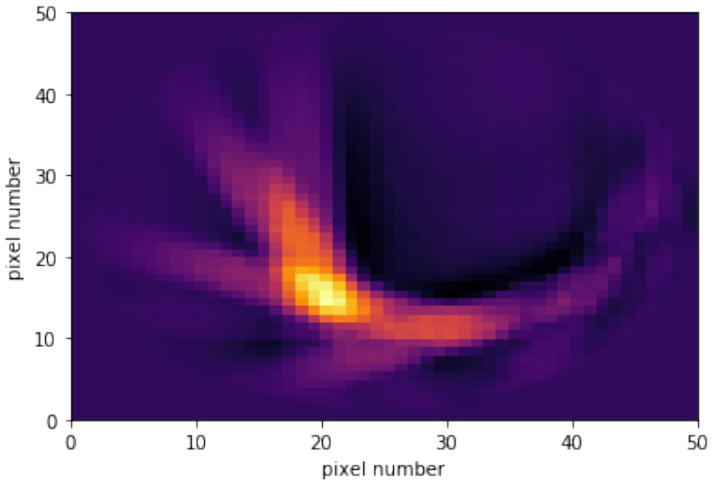


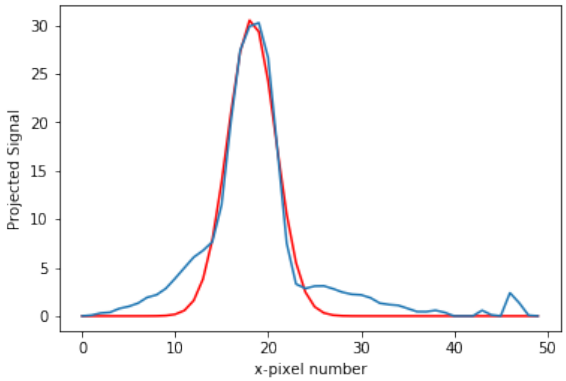
Figure 7.9: *SART reconstruction of the corrected data (50x50 pixels)*

Also with the SART technique the shift of the position of the reconstructed source is clearly visible.

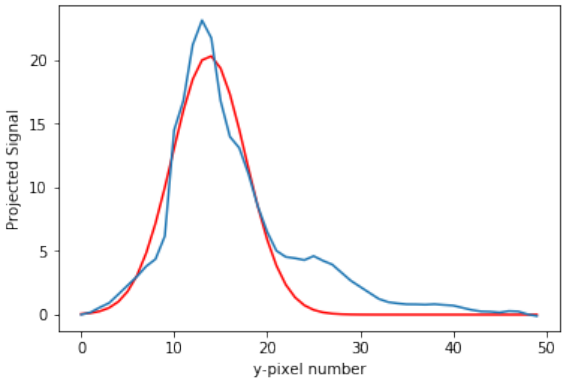
SART technique is the best method to reconstruct the activity profile of the source used in the experiment because it requires less angles of acquisition (ϕ) than the FBP technique. A FBP reconstruction with few angles ϕ (as in this case) can lead to artifacts in the image, as can be seen in figure 7.7. For this reason, the position is derived only from SART reconstructed image.

7.5 Reconstructed position of the source

In order to obtain a quantitative information about the source position, the signal rate of the reconstructed image 7.11 is projected on the x and y axis. The fitted profile of the signal rate is reported in figure 7.10.



(a) *Gaussian fit for the x position*



(b) *Gaussian fit for the y position*

Figure 7.10: *Signal rate projected on the x and y axis from the SART reconstructed image*

The horizontal axis of the plots in figures 7.10 corresponds to the pixel position of the projected signal rate. In order to convert the pixel number value into the real position of the source on the plate, the horizontal axis values are rescaled using the R value of 20.37 cm, derived in section 6.4. To follow the results:

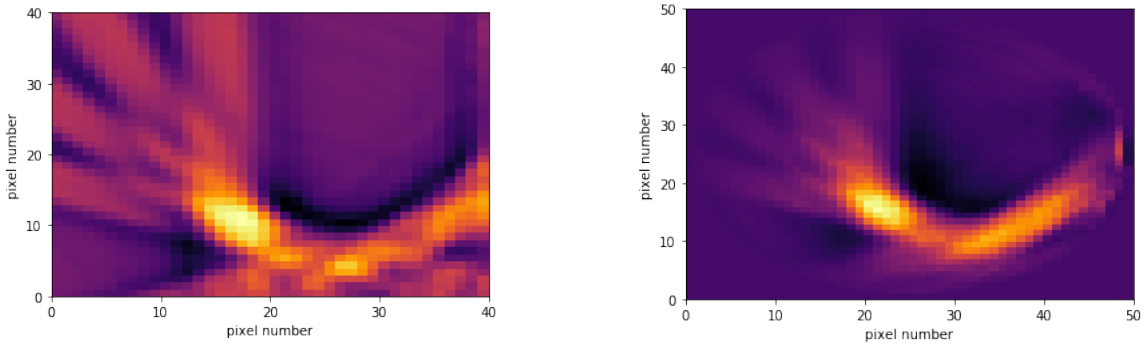
AIR	SART Reconstruction	σ_{SART}	Expected value
x (cm)	-9,17	3,79	-9,3
y (cm)	10,36	2,44	9,3
ρ (cm)	13,83	3,12	13,2
α (°)	131,53	13,53	135

Figure 7.11: *Results for air reconstruction*

Chapter 8

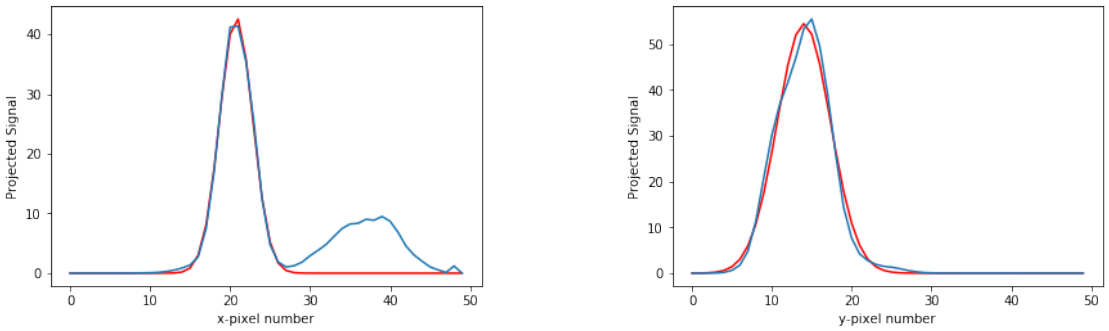
Measurements in water

Since water and the overall human body have a similar attenuation coefficient μ (0.096 cm^{-1} at 511 keV for water [2]), the source is put in a Marinelli beaker filled with water distributed in the toroidal volume (height: 5 cm, radius of the torus: 1.3 cm). The source is then positioned in pos3 and not in pos4 anymore, otherwise in this position the dimensions of the box would exceed the plate border. The reconstructed images are reported in figure 8.1 and 8.2.



(a) *FBP reconstruction* (b) *SART reconstruction*

Figure 8.1: *Image reconstruction for the water measurements*



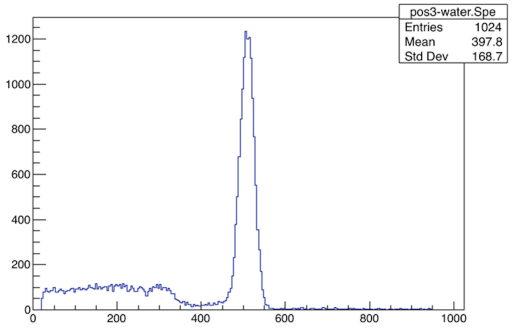
(a) *Gaussian fit for the x position* (b) *Gaussian fit for the y position*

Figure 8.2: *Signal rate projected on the x and y axis from the SART reconstructed image*

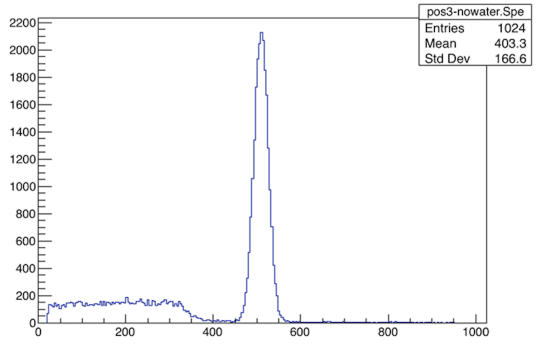
WATER	SART Reconstruction	σ_{SART}	Expected value
x (cm)	-7,02	2,61	-7
y (cm)	8,13	1,97	7
ρ (cm)	10,74	2,27	10
α (°)	130,78	12,58	135

Figure 8.3: *Results for water reconstruction*

The results show that the reconstruction of the image correctly provides the position of the source with uncertainties comparable to the ones measured in the air configuration. Nevertheless, it occurs a problem in evaluating water attenuation because measurements in air are done in pos4. To verify the attenuation of water, two spectra, with same time duration, are taken, both in pos3: the first using beaker Marinelli, the second without it.



(a) *Spectrum taken with beaker Marinelli*



(b) *Spectrum taken without beaker Marinelli*

Figure 8.4: *Study on the attenuation of signal rate in water*

It is observed that, as awaited, the count rate is lower when the source is in Marinelli beaker (1200 s^{-1} in respect to 2200 s^{-1}). This fact is mostly due to Compton scattering of the γs. These photons lose part of their energy and do not contribute to the 511 keV energetic window.

Chapter 9

Conclusion

The purpose of the experiment was to simulate a PET system to reconstruct the position of the radioactive source geometrically and using image reconstruction techniques. Both these approaches seem to derive the correct position within reasonable uncertainties, mostly due to the big acceptance of the detector and to the fact that not all the possible angles are sampled. The real problem encountered through the experiment has been the systematic error, which correction has a fundamental role in obtaining the expected results.
Surely it would have been better to put the source in pos3 to compare the results in air with the ones in water. It would have also been more useful a data acquisition starting angle $(\pi - \alpha)$ closer to the LOR, in $\gamma = 0°$ configuration. Indeed, it would have permitted to include more data sample in the analysis because the problem reported in sinogram 7.4 would probably not occur.
Anyway, considering the overall experiment, the obtained results are satisfactory.

Appendix A

ROOT macro for data analysis

```
1  //composit function for fit the spectrum
2  double funzionefit (double *x, double *par)
3  {
4    double ris = par[0]*exp(-par[1]*x[0]) + par[3]*exp(-((x[0]-par[4])*(x[0]-par[4])))/(2*par[5]*
         par[5])) + par[2];
5    return ris;
6  }
7
8  //propagation formula for the resolution
9  double propagazione (double mu, double mu_err, double sigma, double sigma_err)
10 {
11   double err = TMath::Sqrt( pow((sigma_err/mu),2) + pow(((sigma*mu_err)/(mu*mu)),2) );
12   return err;
13 }
14
15 //Get the m-th max value in the graph
16 double CercaMax(int m = 1, TGraph* gr = NULL)
17 {
18   if(m > 2)
19   { exit(EXIT_FAILURE); }
20
21   double x,y,max1,max2, x1,x2;
22   x1 = 0;
23   x2 = 0;
24   max1 = 0;
25   max2 = 0;
26   for(int i = 0; i < gr->GetN(); i++)
27   {
28     gr->GetPoint(i,x,y);
29     if( y > max1 && max2 == 0)
30     {
31       x1 = x;
32       max1 = y;
33     }
34     if( y > max2 && y < max1 && (x-x1)>6)
35     {
36       x2 = x;
37       max2 = y;
38     }
39   }
40   if ( m == 2 )
41   { return x2;  }
42   else
43   { return x1;  }
44 }
45
46 //Analize the signal rate graph to get alfa or beta angle
47 void GetAngle(char c = 'a', string percorso = "", double tick = 1600, double *v)
48 {
49   //parameters from alpha fit
50   double q = 3.574;
51   double m = 0.03717;
52
53   string fileAngolo = percorso + "angolo.txt";
54   double angleDetectors = v[4];
55   stringstream ss;
56   ss << angleDetectors;
57   string num = ss.str();
58   ss.str("");
59   ss.clear();
60
61   TGraph* angolo = new TGraph(fileAngolo.c_str());
62
63     gStyle->SetOptFit(0000);
64   TCanvas *c1 = new TCanvas("c1","1", 1500, 500);
65   c1 -> cd();
66   string titolo = "Signal Rate al rotare del piatto motorizzato con angolo " + num + " gradi";
```

```
67    angolo->SetTitle(titolo.c_str());
68    angolo->SetMarkerStyle(20);
69    angolo->SetMarkerColor(4);
70    angolo->SetLineColor(2);
71    angolo->GetYaxis()->SetTitle("Conteggi/s");
72    angolo->GetXaxis()->SetTitle("Numero file spettro");
73
74    double picco1, picco2;
75    picco1 = CercaMax(1, angolo);
76    picco2 = CercaMax(2, angolo);
77
78    double picco_ampiezza = 5;
79    TF1* gauss1 = new TF1 ("gauss1","gaus",picco1-picco_ampiezza,picco1+picco_ampiezza);
80    TF1* gauss2 = new TF1 ("gauss2","gaus",picco2-picco_ampiezza,picco2+picco_ampiezza);
81    angolo->Fit("gauss1","RQ");
82    angolo->Fit("gauss2","R+Q");
83    angolo->Draw("AP");
84
85    picco1 = gauss1->GetParameter(1);
86    picco2 = gauss2->GetParameter(1);
87
88    if(picco1 > picco2)
89    {
90      double scambio = picco1;
91      picco1 = picco2;
92      picco2 = scambio;
93    }
94    double separazione = (picco2-picco1)*(360*tick/160000.0);
95
96    cout << "\nPosizione dei picchi: " << picco1 << " | " << picco2 << endl;
97    cout << "Corrispondenti ad angoli di: " << picco1*(360*tick/160000.0) << "    | " << picco2
          *(360*tick/160000.0) << "  " << endl;
98    cout << "(Separati da una distanza di circa " << separazione << "  )." << endl;
99    if(c == 'a')  //alpha analysis
100   {
101     double alfa = (picco1*tick)*(90.0/40000.0)*(1-m)-q;
102     alfa = 180.0 - alfa;
103     double alfa2 = (picco2*tick)*(90.0/40000.0)*(1-m)-q;
104     alfa2 = 360.0 - alfa2;
105
106     v[0] = alfa;
107     v[1] = (gauss1->GetParameter(2)*tick)*(90.0/40000.0);
108     cout << "Alfa1_corr: " << v[0] << "    +- " << v[1] << "  " << endl;
109     cout << "Alfa2_corr: " << alfa2 << "    +- " << (gauss2->GetParameter(2)*tick)
            *(90.0/40000.0) << "  " << endl;
110
111 /*    The following code must be executed only for acquire data for correct sistematic error
112     double val1 = (picco1*tick)*(90.0/40000.0) - 45.0;
113     double val1_err = v[1];
114     double val1_rel = val1*100.0/45.0;
115     double val1_rel_err = val1_err;
116
117     double val2 = (picco2*tick)*(90.0/40000.0) - 225.0;
118     double val2_err = (gauss2->GetParameter(2)*tick)*(90.0/40000.0);
119     double val2_rel = val2*100.0/225.0;
120     double val2_rel_err = val2_err;
121
122     string fileLin = percorso + "linearita.txt";
123     ofstream lin (fileLin.c_str(), ios::out);
124     lin << "picco1: " << 45.0 <<  " " << val1 << " " << gauss1->GetParError(1)*(90.0/40000.0)
            << " " << val1_err << endl;
125     lin << "relat1: " << 45.0 <<  " " << val1_rel << " " << gauss1->GetParError(1)
            *(90.0/40000.0) << " " << val1_rel_err << endl;
126     lin << "picco2: " << 225.0 <<  " " << val2 << " " << gauss2->GetParError(1)*(90.0/40000.0)
             << " " << val2_err << endl;
127     lin << "relat2: " << 225.0 <<  " " << val2_rel << " " << gauss2->GetParError(1)
            *(90.0/40000.0) << " " << val2_rel_err << endl;
128     lin.close();
129 */
130   }
131   else if ( c == 'b' ) //beta analysis
132   {
133    double beta = (picco1*tick)*(90.0/40000.0)*(1-m)-q;
134    v[2] = beta;
135    v[3] = (gauss1->GetParameter(2)*tick)*(90.0/40000.0);
136    cout << "beta: " << (picco1*tick)*(90.0/40000.0) << endl;
137    cout << "Beta_corr: " << v[2] << "    +- " << v[3] << "  " << endl;
138
139 /*    The following code must be executed only for acquire data for correct sistematic error
140    double b2 = (picco2*tick)*(90.0/40000.0);
141    double deltaB = b2 - (270-v[4]-beta);
142    string fileLinB = percorso + "linearita.txt";
143    ofstream linb (fileLinB.c_str(), ios::out);
144    linb << "picco2: " << b2 << " " << deltaB << " " << gauss2->GetParError(1)*(90.0/40000.0)
           << " " << v[3];
145
146    double b2_corr = b2 - (b2*m+q);
147    double deltaB_corr = b2_corr - (270-v[4] - (beta-(beta*m+q)));
148    linb << "\ncorret: " << b2_corr << " " << deltaB_corr << " " << gauss2->GetParError(1)
           *(90.0/40000.0) << " " << v[3];
149    linb.close();
150 */
151   }
```

```
152 }
153
154 //get the angle between detector from the folder name
155 double Gamma(string str = "")
156 {
157   std::size_t found1 = str.find("angolo");
158   std::size_t found2 = str.find("-pos");
159   string num = str.substr(found1+6, found2-found1-6);
160
161   stringstream ss;
162   ss << num;
163   double numero;
164   ss >> numero;
165   return numero;
166 }
167
168 //Calculate the effective distance of detector from center
169 void RaggioDet(double *v)
170 {
171   double gamma = v[4]*(TMath::Pi()/180.0);
172   double alfa = v[0]*(TMath::Pi()/180.0);
173   double beta = v[2]*(TMath::Pi()/180.0);
174
175   double a = v[1]*(TMath::Pi()/180.0);
176   double b = v[3]*(TMath::Pi()/180.0);
177
178   double R_err_p = (cos(alfa + beta) + sin(alfa + beta)/tan(gamma/2.0) );
179   double R_err_ab = v[5] * (cos(alfa + beta)/tan(gamma/2.0) - sin(alfa + beta));
180
181   v[6] = -( v[5] * (cos(alfa + beta) + sin(alfa + beta)/tan(gamma/2.0) ));
182   v[8] = sqrt( R_err_p*R_err_p*v[7]*v[7] + R_err_ab*R_err_ab*(a*a + b*b) );
183 }
184
185 //Calculate the distance of source between center, as a function of R, alfa, beta, gamma
186 void RaggioSorg(double *v)
187 {
188   double gamma = v[4]*(TMath::Pi()/180.0);
189   double alfa = v[0]*(TMath::Pi()/180.0);
190   double beta = v[2]*(TMath::Pi()/180.0);
191
192   double a = v[1]*(TMath::Pi()/180.0);
193   double b = v[3]*(TMath::Pi()/180.0);
194
195   double denom = sin(alfa + beta) + cos(alfa + beta)*tan(gamma/2.0);
196
197   double rho_err_R = tan(gamma/2.0)/denom;
198   double rho_err_ab = (-v[6] * tan(gamma/2.0) * (cos(alfa + beta) - sin(alfa + beta)*tan(gamma
        /2.0)))/ (denom*denom);
199
200   v[5] = -( (v[6] * tan(gamma/2.0))/denom) ;
201   v[7] = sqrt( rho_err_R*rho_err_R*v[8]*v[8] + rho_err_ab*rho_err_ab*(a*a + b*b) );
202 }
203
204 //Calculate sinogram variables
205 void sngVar(double R, double angDet, int numSpe, int tick, double SgnRt, string path)
206 {
207    //parameters from alpha fit
208   double q = 3.574;
209   double m = 0.03717;
210
211   //numSpe start from 1, but first theta is zero
212   double theta = ((numSpe-1)*tick*180.0)/80000.0;
213   theta       = ((theta - m*theta -q)*TMath::Pi())/180.0;
214   angDet      = (angDet*TMath::Pi())/180.0;
215   double phy    = atan( cos(angDet/2 + theta)/sin(angDet/2 + theta) );
216   double H      = R * cos(theta + phy );
217
218   string nomeOutput = path+"sinogram.txt";
219   ofstream sinogram;
220   if(numSpe == 1)
221   {
222     cout << "\nCalcolo i dati per il sinogramma\n";
223     ofstream sinogram (nomeOutput.c_str(),ios::out);
224   }
225   else
226   {
227     ofstream sinogram (nomeOutput.c_str(),ios::app);
228   }
229   //H = distance of the LOR between the center
230   //phy = angle from the normal of the LOR and the X axis
231   sinogram << phy*(180.0/TMath::Pi()) << "," << H << "," << SgnRt << "\n";
232   sinogram.close();
233 }
234
235 void analisi(std::string fileSpe = "", double gamma = 0,int count = 0, int step = 0)
236 {
237   gStyle->SetOptFit(1111);
238   gStyle->SetOptStat(11);
239
240   if (fileSpe == "")
241   {
242     std::cout << "devi inserire come argomento il file .Spe da analizzare.\n";
243     exit(EXIT_FAILURE);
```

```
244   }
245
246   std::size_t found = fileSpe.find_last_of("/");
247   string path = fileSpe.substr(0,found+1);
248   string nomeFile = fileSpe.substr(found+1);
249   nomeFile = nomeFile.substr(0,nomeFile.length()-4);
250
251   ifstream origine (fileSpe.c_str(),ios::in);
252   std::stringstream leggi;
253   std::string riga;
254   std::string nomeHisto;
255   int liveTime, deadTime, totTime;
256   int Nch, entry;
257   for (int i = 1; i < 13; i++)
258   {
259     getline(origine, riga);
260     if (i == 2)
261     { nomeHisto = riga;  }
262
263     else if (i == 10)
264     {
265
266       leggi << riga;
267       leggi >> liveTime >> totTime;
268       leggi.str("");
269       leggi.clear();
270     }
271     else if (i == 12)
272     {
273
274       leggi << riga;
275       leggi >> Nch >> Nch;
276       Nch++;
277       leggi.str("");
278       leggi.clear();
279     }
280   }
281   TH1F* histo = new TH1F(fileSpe.c_str(),nomeHisto.c_str(), Nch, 0, Nch);
282   int TotEntries = 0;
283   for(int i = 0; i < Nch; i++)
284   {
285     getline(origine, riga);
286     leggi << riga;
287     leggi >> entry;
288     histo->SetBinContent(i,entry);
289     TotEntries += entry;
290     leggi.str("");
291     leggi.clear();
292   }
293   int rebinning = 4;
294
295   histo->Rebin(rebinning);
296   int binmax = (int) histo->GetNbinsX();
297   (histo->GetXaxis())->SetRange(binmax/3,binmax);
298
299   int binPeak = rebinning * histo->GetMaximumBin();
300   (histo->GetXaxis())->SetRange(0,binmax);
301
302   int peak_low = binPeak * 0.95;
303   int peak_high = binPeak * 1.05;
304   int fondo1_low = binPeak * 0.7;
305   int fondo1_high = binPeak * 0.85;
306   int fondo2_low = binPeak * 1.2;
307   int fondo2_high = binPeak * 1.6;
308
309   TH1F* histo0 = new TH1F(*histo);
310
311   //fit function
312   TF1* gausss = new TF1 ("gausss","gaus",peak_low,peak_high);
313   TF1* exponential = new TF1 ("exponential","[0]*exp(-[1]*x)+[2]",fondo1_low,fondo1_high);
314   TF1* constant = new TF1 ("constant", "pol0", fondo2_low,fondo2_high);
315
316   histo->Fit(gausss,"R0NQ");
317   histo->Fit(constant,"R+0NQ");
318   exponential->SetParameter(2,constant->GetParameter(0));
319   exponential->SetParLimits(1,0.,1.);
320
321   histo->Fit(exponential,"R+0Q");
322   TF1* fitfunc = new TF1 ("fitfunc",funzionefit, fondo1_low, fondo2_high, 6);
323   fitfunc->SetParameter(0,exponential->GetParameter(0));
324   fitfunc->SetParameter(1,exponential->GetParameter(1));
325   fitfunc->SetParameter(2,exponential->GetParameter(2));
326   fitfunc->SetParameter(3,gausss->GetParameter(0));
327   fitfunc->SetParameter(4,gausss->GetParameter(1));
328   fitfunc->SetParameter(5,gausss->GetParameter(2));
329   fitfunc->SetParLimits(1,0.,1.);
330   fitfunc->SetParNames("Exp-Const", "Exp", "Const", "Gaus-Const", "Mean", "Sigma");
331   histo0->Fit("fitfunc", "RQ");
332   TF1* fondo = new TF1 ("fondo","[0]*exp(-[1]*x)+[2]",fondo1_low,fondo2_high);
333   fondo->SetParameter(0,fitfunc->GetParameter(0));
334   fondo->SetParameter(1,fitfunc->GetParameter(1));
335   fondo->SetParameter(2,fitfunc->GetParameter(2));
336
```

```
337     //get fit information
338     double mean = fitfunc->GetParameter(4);
339     double mean_err = fitfunc->GetParError(4);
340     double sigma = fitfunc->GetParameter(5);
341     double sigma_err = fitfunc->GetParError(5);
342     double FWHM = sigma * 2.35;
343     double FWHM_err = sigma_err * 2.35;
344     double ris = (FWHM/mean)*100;
345     double ris_err = propagazione(mean, mean_err, FWHM,FWHM_err)*100;
346     double areaFondo = fondo->Integral(mean-sigma,mean+sigma);
347     double areaFit   = fitfunc->Integral(mean-sigma,mean+sigma);
348     double peakArea  = areaFit - areaFondo;
349     double SignalRate =(double) peakArea/liveTime;
350     double chi2 = (histo0->GetFunction("fitfunc"))->GetChisquare() / (histo0->GetFunction("
        fitfunc"))->GetNDF();
351     cout << "Tot Entries: " << TotEntries << " N. spe: " << count <<  " Signal rate: " <<
        SignalRate << " chi2_rid: " << chi2 << endl;
352     cout << "Area = " << peakArea << " LiveTime = " << liveTime << " TotTime = " << totTime << "
         DeadTime = " << (totTime-liveTime)/totTime << endl;
353
354     //output
355     string angolo = path+"angolo.txt";
356     ofstream destinazioneAngolo (angolo.c_str(),ios::app);
357     destinazioneAngolo << count << " " << SignalRate << " " << endl;
358     destinazioneAngolo.close();
359 }
360
361 void spectrum2(string dati = "dati.txt")
362 {
363    string files[] = {}; //vettore che contiene tutti i files da analizzare in una volta
364
365    ifstream lettore;
366    lettore.open(dati.c_str(), ios::in);
367    string DaAnalizzare;
368    int dimFiles = 0;
369
370    getline(lettore,DaAnalizzare);
371    double gamma = Gamma(DaAnalizzare);
372    cout << "Raccolta ad angolo tra i detector di " << gamma << "  " << endl;
373    std::size_t found = DaAnalizzare.find_last_of("/");
374    string path = DaAnalizzare.substr(0,found+1);
375
376   //get from file the number of step between movements
377    string moto = path + "esempiocodice.tmc";
378   ifstream motorino (moto.c_str(),ios::in);
379   std::stringstream ss;
380   std::string riga;
381   int step;
382   for (int i = 1; i < 23; i++)
383   {
384     getline(motorino, riga);
385   }
386   ss << riga;
387   ss >> riga >> riga >> riga;
388   ss >> step;
389   ss.str("");
390   ss.clear();
391   motorino.close();
392   cout << "Numero di step per questa raccolta: " << step << endl;
393
394    ifstream test;
395    string angle = path + "angolo.txt";
396      test.open(angle.c_str());
397
398      if(test.good() == 0)
399    {
400     cout << "Inizio l'analisi dei files.." << endl;
401       do
402     {
403       cout << "\n" << DaAnalizzare << endl;
404       dimFiles++;
405       analisi(DaAnalizzare,gamma,dimFiles,step);
406     }
407     while(getline(lettore,DaAnalizzare));
408     cout << "Ho analizzato " << dimFiles << " files." << endl;
409    }
410    else
411    {
412     cout << "Il file angolo.txt esiste gi  nel percorso '" << path << "'" << endl;
413    }
414    lettore.close();
415
416    double vet[9];    //vector of data passed trought function
417
418    vet[0] = 135.48;    // alpha angle, deg
419    vet[1] = 2.47;      // alfa error, deg
420    vet[2] = 0.0;       // beta angle
421    vet[3] = 0.1;       // beta error
422    vet[4] = gamma;     // 180 - angle between detector
423    vet[5] = 13.2;      // rho: distance of source from center of plate
424    vet[6] = 20.37;     // R: detecctor radius
425    vet[7] = 0.1;       // rho error
426    vet[8] = 1.55;      // R error
```

```
427
428       if (gamma == 0)
429        {
430         cout << "Analizzo alfa." << endl;
431         GetAngle('a', path, step, vet);
432        }
433        else
434        {
435         cout << "Analizzo beta." << endl;
436         GetAngle('b', path, step, vet);
437        }
438        cout << "\nLa posizione corrente    pos4.\nSe vuoi cambiare posizione indicane il numero,
               altrimenti digita 'n'." << endl;
439        cout << "( rho = 3.3 * pos[N] )" << endl;
440        int pos;
441        char ans = '4';
442        do
443        {
444         cin >> ans;
445         pos = ans - '0';
446         if ( ans == 'n')
447         {
448           cout << "Ok rimane pos4: " << vet[5] << " cm" << endl;
449         }
450         else if (pos < 0 || pos > 5)
451         {
452           cout << "La posizione non    quella corretta. riprova :)" << endl;
453         }
454         else
455         {
456           vet[5] = (double) (pos * 3.3);
457           cout << "Nuovo rho = " << vet[5] << " cm" << endl;
458         }
459        } while ((pos < 0 || pos > 5) && ans != 'n');
460
461      if (gamma != 0)
462      {
463       RaggioDet(vet);
464       vet[6] = 20.37;
465       vet[8] = 1.97;
466       cout << "Ho calcolato R. Vuoi ricalcolare rho? (y/n)" << endl;
467       do
468       {
469         cin >> ans;
470         if (ans == 'y')
471          {
472           cout << "Ok analizzo rho." << endl;
473           RaggioSorg(vet);
474          }
475         else
476         {
477           cout << "Devi dire si o no! riprova :)" << endl;
478
479         }
480         } while (ans != 'y' && ans != 'n');
481      }
482      cout << "Completata raccolta informazioni geometriche." << endl;
483
484      if(gamma != 0)
485      {
486       ifstream lettoreSGNrate;
487       lettoreSGNrate.open(angle.c_str(), ios::in);
488       stringstream ss1;
489       string temp;
490       double count, SignalRate;
491
492       while(getline(lettoreSGNrate,temp))
493       {
494         ss1 << temp;
495         ss1 >> count;
496         ss1 >> SignalRate;
497         ss1.clear();
498         ss1.str("");
499         //Calculate the sinogram data
500         sngVar(vet[6], gamma, count, step, SignalRate, path);
501       }
502      }
503
504       ofstream risultati;
505       string ris = path + "risultati.txt";
506       risultati.open(ris.c_str(), ios::out);
507       risultati << "Percorso: " << path << endl;
508       risultati << "alfa: " << vet[0] << " +- " <<  vet[1] << endl;
509       risultati << "beta: " << vet[2] << " +- " <<  vet[3] << endl;
510       risultati << "gamma: " <<  vet[4] << endl;
511       risultati << "rho: " << vet[5] << " +- " << vet[7] << endl;
512       risultati << "R: " << vet[6] << " +- " << vet[8] << endl;
513       risultati.close();
514
515       cout << "\nRisultati estratti da questa raccolta: (salvati nel file risultati.txt)" <<
              endl;
516       cout << "alfa: " << vet[0] << "    alfa_err: " <<  vet[1] << "  " << endl;
517       cout << "beta: " << vet[2] << "    beta_err: " <<  vet[3] << "  " << endl;
```

```
518         cout << "gamma: " <<  vet[4] << "  " << endl;
519         cout << "rho: " << vet[5] << " cm rho_err " << vet[7] << " cm" << endl;
520         cout << "R: " << vet[6] << " cm R_err: " << vet[8] << " cm" << endl;
521         cout << "\nThat's All Folks! grazie per aver analizzato con noi :)" << endl;
522 }
```

Appendix B

Programs for data acquisition campaigns

Routine for the rotating plate:

```
TotalTime = 30 // Per acquisition. In minutes

SAP 203, 0, 2048    //Disable Mixed Decay
SAP 5, 0, 50    //Set Acceleration
SAP 4, 0, 100    //Set maximum Velocity
SAP 205, 0, 6    //Setup StallGuard

MVP ABS, 0, -50000
WAIT POS, 0, 0
MVP ABS, 0, -20000
WAIT POS, 0, 0
CSUB WaitOneMeasure

MainLoop:
MVP REL, 0, 1600
CSUB WaitOneMeasure
GAP 1, 0
COMP 160000
JC LT, MainLoop

WaitOneMeasure:   WAIT POS, 0, 0
                  SGP 0, 2, 0 // Initialize timer to zero
                  OneMeasureLoop: WAIT TICKS, 0, 6000
                                  GGP 0, 2
                                  CALC ADD, 1
                                  AGP 0, 2
                                  GGP 0, 2
                                  COMP TotalTime
                                  JC LT, OneMeasureLoop
                  RSUB
```

Program for Maestro acquisition:

```
SET_DETECTOR 1
SET_PRESET_CLEAR
SET_PRESET_REAL 1795  //the acquisition lasts 1795 seconds
LOOP 100              //100 loops needed to acquire 360 degrees
CLEAR
START
WAIT
DESCRIBE_SAMPLE "Misura ??? "
SAVE "C:\Documents and Settings\lab Brof\Desktop\PET 2017\..."
WAIT 5                //wait 5 seconds between the ending
  //acquisition and the starting one
SET_DETECTOR 1
END_LOOP
```

Appendix C

Python3 program for image reconstruction

```
import numpy as np
import pandas as pd
import os
from scipy import odr
from skimage.transform import iradon, rescale
from skimage.transform import iradon_sart
from skimage.transform import radon
import math
import matplotlib.pyplot as plt
from pylab import *
from random import *

file='acqua_corretto.txt'

phi_s_ncr = np.recfromcsv(file, names=['phi','s','ncr'])
datisinogramma = np.asarray(phi_s_ncr)
phi = phi_s_ncr['phi']
s = phi_s_ncr['s']
n_bins = 50

#SINOGRAM
plt.clf()
my_xticks = np.arange(-100, 90, 20)
my_yticks = np.arange(-8, 11, 1)
plt.xticks( my_xticks)
plt.yticks( my_yticks)
sinogram, xax, yax, imagesinogram = plt.hist2d(phi,s, bins= n_bins , weights=phi_s_ncr['ncr'],
      cmap=plt.cm.inferno, cmin=None, cmax=None, hold=None, data=None)
plt.savefig("sinogramma.png",dpi=300)
#plt.colorbar()
plt.xlabel("$\phi$ (degrees)")
plt.ylabel("$S$ (cm)")
plt.title('Sinogram')
plt.show()

#SINOGRAM TRASP
plt.clf()
my_yticks = np.arange(-100, 90, 20)
my_xticks = np.arange(-11, 11, 1)
plt.xticks( my_xticks)
plt.yticks( my_yticks)
sinogram_tras, xax_tras, yax_tras, imagesinogram = plt.hist2d(s,phi, bins= n_bins , weights=
     phi_s_ncr['ncr'], cmap=plt.cm.inferno, cmin=None, cmax=None, hold=None, data=None)
plt.savefig("sinogramma_tras.png",dpi=300)
plt.colorbar()
plt.ylabel("$\phi$ (degrees)")
plt.xlabel("$S$ (cm)")
#plt.title('Sinogramma')
plt.show()

#HEX SINOGRAM
plt.clf()
my_xticks = np.arange(-100, 90, 20)
my_yticks = np.arange(-8, 11, 1.5)
plt.xticks( my_xticks)
plt.yticks( my_yticks)
plt.hexbin(phi_s_ncr['phi'], phi_s_ncr['s'], C=phi_s_ncr['ncr'], cmap=plt.cm.inferno)
plt.savefig("sinogramma_hex.png",dpi=300)
#plt.show()

#SART RECONSTRUCTION
reconstruction_sart = iradon_sart(sinogram, theta=None)
plt.clf()
```

```
plt.grid(True)
reconstruction_sart_arr = np.asarray(reconstruction_sart)
reconstruction_tras_sart=reconstruction_sart_arr
#xt=np.arange(-25,25,1)
#yt=np.arange(-25,25,1)
my_xticks = np.arange(0, 60, 10)
my_yticks = np.arange(0,60, 10)
plt.xticks( my_xticks)
plt.yticks( my_yticks)
plt.ylabel("pixel number")
plt.xlabel("pixel number")
plt.pcolor(reconstruction_tras_sart, cmap=plt.cm.inferno)
#plt.colorbar()
plt.savefig("ricostruzione_sart.png",dpi=300)
#plt.title('SART reconstruction (50x50 pixels)')
plt.show()

m,n = reconstruction_sart.shape

#FBP RECONSTRUCTION
reconstruction_fbp = iradon(sinogram, theta=None, output_size=40, filter="shepp-logan",
    interpolation="cubic",circle=False)
plt.clf()
#xt=np.arange(-20,20,1)
#yt=np.arange(-20,20,1)
my_xticks = np.arange(0, 60, 10)
my_yticks = np.arange(0,60, 10)
plt.xticks( my_xticks)
plt.yticks( my_yticks)
plt.grid(True)
plt.ylabel("pixel number")
plt.xlabel("pixel number")
reconstruction_fbp_arr = np.asarray(reconstruction_fbp)
reconstruction_tras_fbp = reconstruction_fbp_arr
plt.pcolor(reconstruction_tras_fbp, cmap=plt.cm.inferno)
#plt.colorbar()
plt.savefig("ricostruzione_fbp",dpi=300)
#plt.title('FBP reconstruction (40x40 pixels zoomed)')
plt.show()


#X AND Y PROJECTIONS

proiezionex = []
proiezioney = []

for i in range(0 , m):
    cumsumx = 0
    #variance_num=0
    for j in range(0 , n):
        cumsumx += reconstruction_tras_sart[i][j]
        #media = cumsumx/n
        #variance_num += (reconstruction_tras_sart[i][j] - media)*(reconstruction_tras_sart[i
            ][j] - media)
        #variance = variance_num/(n-1)
    #cumsumx=cumsumx*variance
    proiezionex.append(cumsumx)
proiezionex_arr = np.asarray(proiezionex) #era una lista ora    un array

for i in range(0, n):
    cumsumy = 0
    variance_num=0
    for j in range (0,m):
        cumsumy += reconstruction_tras_sart[j][i]
        #media = cumsumy/m
        #variance_num += (reconstruction_tras_sart[j][i] - media)*(reconstruction_tras_sart[j
            ][i] - media)
        #variance = variance_num/(m-1)
    #cumsumy=abs(cumsumy)
    #cumsumy=cumsumy*variance
    proiezioney.append(cumsumy)
proiezioney_arr = np.asarray(proiezioney)

xmaxpos=0
for i in proiezionex_arr:
    if i != max(proiezionex):
        xmaxpos+=1
    if i == max(proiezionex):
        break
ymaxpos=0
for i in proiezioney_arr:
    if i != max(proiezioney):
        ymaxpos+=1
    if i == max(proiezioney):
        break

#NOT FITTED GRAPHS

plt.clf()
plt.plot(proiezioney_arr)
plt.title('X projection')
plt.ylabel("Projected singnal")
plt.xlabel("x-pixel number")
```

```
plt.show()
fig, ax = plt.subplots()

plt.clf()
plt.plot(proiezionex_arr)
plt.title('Y projection')
plt.ylabel("Projected singnal")
plt.xlabel("y-pixel number")
plt.show()

#FITTED X GRAPH

from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

def gauss(x, *p):
    a, b, c = p
    y = a*np.exp(-np.power((x - b), 2.)/(2. * c**2.))
    return y

py_initial = [30.0, 20.0, 20.]
x = np.arange(0,n_bins,1)
y = proiezioney_arr
popt, pcov = curve_fit(gauss, x, y, p0 = py_initial)
y_fit = gauss(x, *popt)

fig, ax = plt.subplots()
ax.errorbar(x,y) #,e)
ax.plot(x, y_fit, color = 'red')
ax.set_xlabel(r'x-pixel number')
ax.set_ylabel(r'Projected Signal')
#ax.set_title('Gaussian fit for the x position')
plt.show()

R=20.37
xrealmedia= (popt[1]/n_bins)*(2*R)
xrealsigma= (abs(popt[2])/n_bins)*(2*R)
#valpixel:x=50:2R

print('> PIXEL:' , 'X-center:' , popt[1],'Sigma:', abs(popt[2]))
print('> CONVERTED:' ,  'X-center:', xrealmedia ,'Sigma:', xrealsigma)
xpos = (xrealmedia - R)
print('> X position on the circle:', xpos)

#FITTED Y GRAPH

px_initial = [10.0, 30.0, 10.]
x = np.arange(0,n_bins,1)
y = proiezionex_arr
popt, pcov = curve_fit(gauss, x, y, p0 = px_initial)
y_fit = gauss(x, *popt)

fig, ax = plt.subplots()
ax.errorbar(x,y)
ax.plot(x, y_fit, color = 'red')
ax.set_xlabel(r'y-pixel number')
ax.set_ylabel(r'Projected Signal')
#ax.set_title('Gaussian fit for the y position')
plt.show()

yrealmedia= (popt[1]/n_bins)*(2*R)
yrealsigma= (abs(popt[2])/n_bins)*(2*R)

print('> PIXEL:' , 'Y-center:' , popt[1],'Sigma:', popt[2])
print('> CONVERTED:' , 'Y-center:', yrealmedia ,'Sigma:', yrealsigma )
ypos = yrealmedia - R
print('> Y position on the circle:', ypos)

#FINAL RESULT
rho = math.sqrt(xpos**2 + ypos**2)
alpha = 180 - math.degrees(math.asin(ypos/rho))
print('rho:', rho)
print('alpha:', alpha)
print('E questo e\' quanto')
```

Appendix D

Derivation of the formula used in geometrical method to calculate ρ

To calculate the position of the source ρ in respect of the plate center is sufficient to know:

- $\frac{\gamma}{2}$ angle. This angle is the angle at the circumference, while γ is the angle at the center. γ angle is known because it is the angle of which it is moved detector 0 from its original position;
- β angle which is the angle described by the source before crossing the LOR, when detector 0 is moved from its original position by a $\gamma \neq 0°$ angle;
- α angle which is the supplementary angle of the one described by the source before crossing the LOR (Line Of Response), when detector 0 is at its original position, opposite to detector 1 ($\gamma = 0°$);
- R is the distance between the plate center and the point in which the interaction occurs.

All this observables are depicted in figure D.1. The formula used is:

$$\rho = \frac{tan\left(\frac{\gamma}{2}\right)\ R}{-sin(\alpha+\beta) - tan\left(\frac{\gamma}{2}\right)\ cos(\alpha+\beta)}$$

In particular this formula is obtained firstly intersecting two lines, function of the known angles:

$$t : y = tan\left(\alpha+\beta\right) x$$

$$r : y = tan\left(\frac{\gamma}{2}\right)(x+R)$$

From their intersection, the position (x,y) of the source on the plate is derived:

$$x\ =\ \frac{tan\left(\frac{\phi}{2}\right) R}{tan(\alpha+\beta) - tan\left(\frac{\phi}{2}\right)}$$

$$y\ =\ tan(\alpha+\beta)\, x$$

Substituting x inside y, finally it is possible to evaluate the position of the source through:

$$\rho\ = \sqrt{x^2+y^2}$$

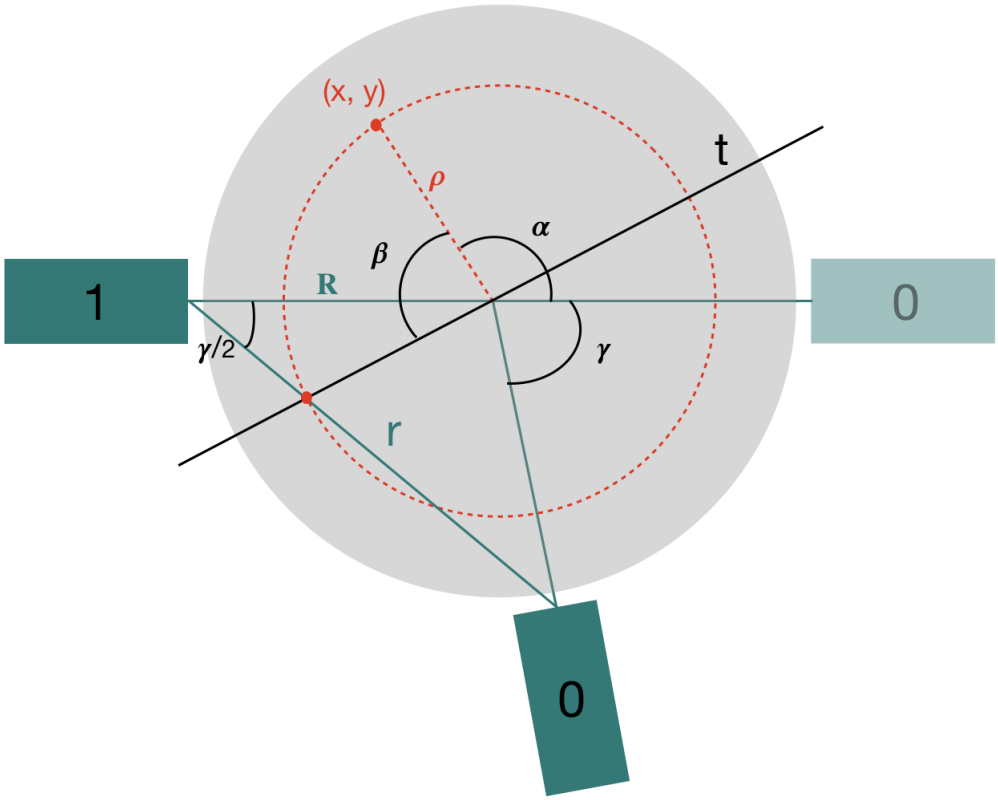


Figure D.1: *Observables necessary to obtain* ρ

Bibliography

[1] *Positron emission tomography (PET) scanning.* Rao, Dandamudi V., PhD, Magills Medical Guide (Online Edition), 2013;

[2] *PET attenuation coefficient from CT images: experimental evaluation of the transformation of CT into PET 511-keV attenuation coefficient.* Burger C., et al. Eur J Nucl Med Mol Imaging, 2002.