COUPLING THE INDRA AND VAMOS MULTI-DETECTORS FOR SYMMETRY ENERGY STUDIES

PETER COLVIN WIGG

DEPARTMENT OF PHYSICS

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy

June 2015

Abstract

The study of the symmetry energy under well controlled laboratory conditions serves to provide constraints for theoretical models covering a wide range of physical phenomena, from the neutron skin thickness of ^{208}Pb to the structure and dynamics of neutron stars. Of particular interest at the present time is the density dependence of the symmetry energy. The INDRA-VAMOS (E503) experiment intends to extract constraints on the density dependence of the symmetry energy at low densities using isotopic scaling and isospin diffusion measurements. In this report the physics motivation and experiment will be discussed in detail, as well as the calibration of the detectors and recent changes to the analysis code.

Acknowledgements

I wish to express my sincere thanks to Drs. Marielle Chartier, David Joss, Robert Page and Abdou Chbihi for their expert guidance and support, formal and personal, without which this thesis would not exist. I wish to thank the INDRA-VAMOS collaboration and the GANIL staff warmly for their hospitality and assistance over the past few years and for the opportunity to work in such a fascinating field of Nuclear Physics. I would also like to thank the Science and Technology Facilities Council for providing the necessary funding. Finally, I wish to thank my family for their unfailing encouragement and support, in particular my Mother, Father and Grand-parents, whose belief in me appears to be a fundamental constant of the universe.

Contents

List of Figures

List of Tables

Chapter 1

Physics Motivation

1.1 Nuclear Equation of State

The Nuclear Equation of State describes the relationship between the different state variables (pressure, density, temperature, etc.) of a nuclear system. Although when dealing with low mass nuclei, definitions of thermodynamic quantities such as temperature must be made with care. The Equation of State has until recently been studied only at the saturation point of nuclear matter. The saturation point represents the steady state of ordinary nuclear matter and has a nucleon density of $0.17\,\mathrm{fm}^{-3}$.

This restriction of observations to the saturation region introduces significant uncertainties into the Equation of State as one deviates from it. In the past few decades there has been an increasing demand to minimise these uncertainties and to attain a better understanding of the equation of state far from saturation.

1.2 What is the Symmetry Energy?

In highly isospin asymmetric environments the uncertainties in the equation of state are governed by a deficiency in our current understanding of what is known as the symmetry energy. This has an impact on the theoretical models used in such environments and in particular when there are additional complexities introduced by densities deviating from saturation and non-zero temperatures. The symmetry energy is the energy contribution to a nuclear system which results if there are unequal neutron and proton densities present, it stems from the difference in behaviour between neutrons and protons within the nuclear medium.

Perhaps the easiest way to understand this it to look at the Fermi gas model. The Fermi gas model is often used as a very simple description of the nucleus, it treats the nucleons as freely moving (non-scattering) fermions able to occupy states up to a maximum energy (the Fermi energy, at zero temperature) which is determined by the nucleon density. If the neutron and proton densities are treated independently (i.e. not assumed to be equal) then the Fermi energy for protons and neutrons can be different.

Under these conditions the Fermi gas model predicts an additional contribution to the energy of the nucleus [1, p. 154-158]:

$$\Delta E = C\frac{(Z - A/2)^2}{A} \tag{1.1}$$

Where C is a constant and Z and A have their usual meanings as the atomic and mass numbers respectively.

This equation states that if the neutron and proton densities are not equal, then there is an increase in the energy of the nucleus. This contribution to the energy which comes directly from the asymmetry between the neutron and proton densities is known as the symmetry energy. Admittedly, this is something of a misnomer as the energy contribution is driven by asymmetry rather than symmetry but this term is kept for historical reasons and for the sake of consistency. The symmetry energy also makes a more familiar appearance in the liquid drop model [1, p. 213], which accurately predicts the binding energies of moderate to large mass nuclei by treating the nucleus as a homogeneous fluid and fitting the well known Bethe-Weizsäcker formula against the masses of known nuclei. The liquid drop model is a very useful description to have in mind when thinking about the symmetry energy as it is useful to think of proton and neutron motion as the flow of charged and uncharged currents.

The asymmetry of a nucleus is defined as the difference between the neutron and proton densities, normalised to the total nucleon density:

$$\delta = \frac{\rho_n - \rho_p}{\rho} \tag{1.2}$$

Where the total nucleon density: $\rho = \rho_n + \rho_p$. By expanding the expression for the energy of the nucleus around the point of zero asymmetry, the following expression can be extracted:

$$E_A(\rho, \delta) = E_A(\rho, 0) + S_2(\rho)\,\delta^2 + S_4(\rho)\,\delta^4 + S_6(\rho)\,\delta^6 + \ \ldots \tag{1.3}$$

Where E_A denotes the energy per nucleon, $E_A(\rho, 0)$ the density dependent energy per nucleon of symmetric nuclear matter and $S_{2,4,6,\ldots}$ the density dependent functions relating to the asymmetry:

$$S_2(\rho) = \frac{1}{2!}\frac{\partial^2 E_A}{\partial \delta^2}\bigg|_{\delta=0} \qquad S_4(\rho) = \frac{1}{4!}\frac{\partial^4 E_A}{\partial \delta^4}\bigg|_{\delta=0} \qquad S_6(\rho) = \frac{1}{6!}\frac{\partial^6 E_A}{\partial \delta^6}\bigg|_{\delta=0} \tag{1.4}$$

Only even orders of asymmetry appear in Equation 1.3 due to the neutron/proton exchange symmetry which results under the assumption that the Coulomb interaction can be ignored and assuming the charge symmetry of the nuclear force. The higher order terms are generally considered to be negligible and the following parabolic approximation is widely used, although the higher order terms can become important in high density scenarios such as in neutron star physics [2]. The parabolic approximation has been verified using both macroscopic and ab initio microscopic calculations across a broad range of asymmetry [3].

$$E_A(\rho, \delta) = E_A(\rho, 0) + S_2(\rho)\,\delta^2 \tag{1.5}$$

Due to this parabolic approximation, when discussing the symmetry energy it is generally in reference to S_2 but always be aware that the higher order terms still exist and can become relevant. For a qualitative idea of what the symmetry energy (S_2) means, the value of δ can be set to 1 in Equation 1.5 and it can be seen as the difference in energy between pure neutron matter and symmetric nuclear matter for a given density:

$$S_2(\rho) = E_A(\rho, 1) - E_A(\rho, 0) \tag{1.6}$$

The density dependence of the symmetry energy is a particularly important input to many theoretical models and is one of the key areas being studied at the present time. By expanding the symmetry energy around the saturation density of nuclear matter $\rho = \rho_0$ it is possible to explore the region around this value:

$$S_2(\rho) = S_2(\rho_0) + \frac{L_2}{3}\left(\frac{\rho - \rho_0}{\rho}\right) + \frac{K_2}{18}\left(\frac{\rho - \rho_0}{\rho}\right)^2 + \ldots \tag{1.7}$$

Where L_2 and K_2 denote the slope and incompressibility (curvature) parameters and are given by the following expressions:

$$L_2 = 3\rho \left.\frac{\partial S_2(\rho)}{\partial \rho}\right|_{\rho=\rho_0} \tag{1.8}$$

$$K_2 = 9\rho^2 \left.\frac{\partial^2 S_2(\rho)}{\partial \rho^2}\right|_{\rho=\rho_0} \tag{1.9}$$

These two parameters have become useful for comparing the different symmetry energies extracted from theoretical models and experimental data (See Figure 1.6 as an example). Of course the density dependence of S_2 can be expanded around any desirable value of the density and not just that of the saturation point but typically the theoretical models are only well constrained in this region (See Figure 1.1).

There are a number of physical phenomena that are sensitive to both the absolute value of the symmetry energy and to its density dependence. Neutron star physicists in particular have a great interest in this area [4, 5, 6, 7, 8, 9] as the environment is naturally isospin asymmetric and the densities involved range from zero density at the surface to well above saturation density in the core (typically up to several times ρ_0). In this case the density dependence of the symmetry energy governs many features of both the internal structure and of course the dynamics. Typically the surface properties

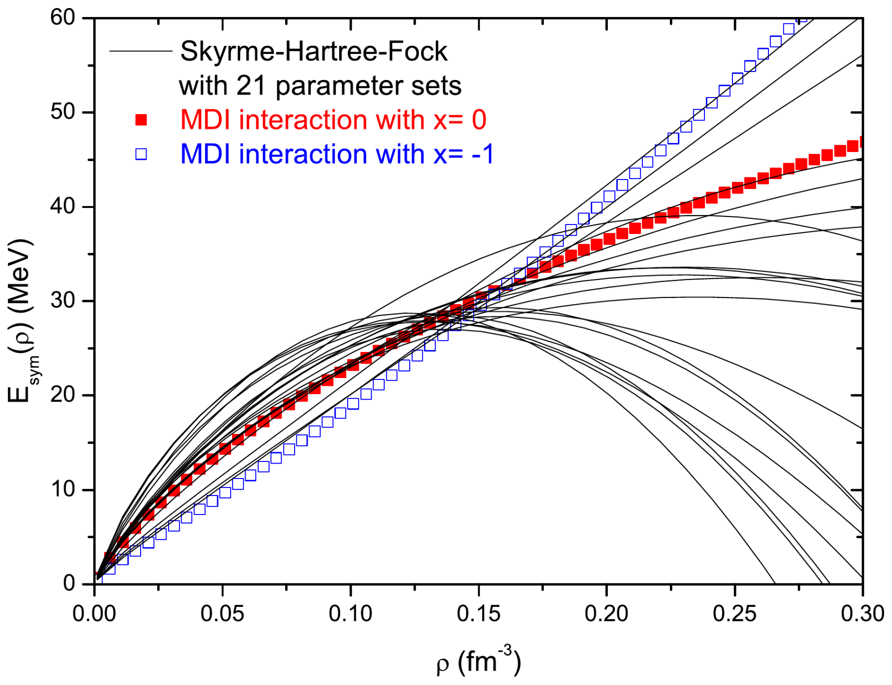

Figure 1.1: This Figure shows the density dependence of the symmetry energy for several nucleon-nucleon interactions and serves to demonstrate the lack of experimental constraint when one moves away from saturation density. Of particular importance is the lack of constraint in the high density region, as this is a critical component for understanding the structure and dynamics of neutron stars. Taken from [3].

of the star can be more easily probed but the core of the star is much more difficult to constrain through direct observation. Neutrino emission from the core as well as gravity waves which are expected to emanate from binary neutron star systems could provide more constraining data but such measurements are difficult to achieve. Aside from neutron stars another astrophysical interest in the symmetry energy can be found in the dynamics of supernova collapse which also requires the symmetry energy and its density dependence as an input for theoretical models.

There are also a great many terrestrial situations in which the symmetry energy plays an important role. For example the neutron skin thickness of heavy nuclei [10] (e.g. ^{208}Pb) is governed by the symmetry energy in much the same way as the radius of a neutron star - increased neutron repulsion (pressure) resulting from the symmetry potential at low densities (See Figure 1.2) increases the radius of the neutron distribution beyond that of the proton distribution.

A few more examples include the dynamics of heavy ion collisions, isovector collective vibrational modes such as the giant and pygmy dipole resonances [12, 13, 14, 15, 16, 17] as well as influencing isobaric analogue states [18]. Generally speaking any situation in which there is isospin transport (neutron/proton current exchange) or collective motion of neutrons and protons in a locally isospin asymmetric environment,

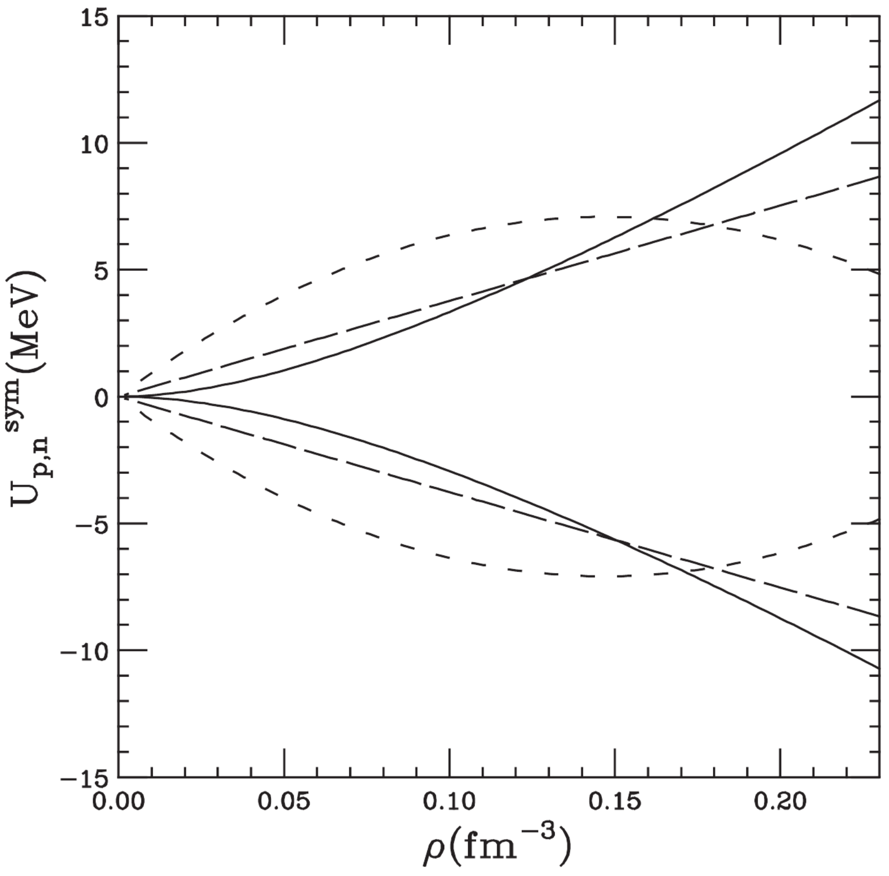

Figure 1.2: Symmetry contribution to the mean field potential for a nucleus with isospin of $(N-Z)/A = 0.2$ (^{124}Sn) for 3 different effective interactions - taken from [11]. The neutrons are represented by the upper curves and the protons by the lower curves. Note that at least in the density range covered by this plot the symmetry potential is repulsive for neutrons and attractive for protons. It is also clear that the choice of interaction can drastically effect the behaviour of neutrons and protons at densities other than the saturation density ($0.17\,\mathrm{fm}^{-3}$).

the symmetry energy will play a significant role. There are many effects besides those mentioned here which are predicted to be sensitive to the symmetry energy, a few others can be found here [19, 20, 21, 22, 23, 24, 25].

1.3 Experimental Observables

The behaviour of the symmetry energy can be studied at different points on the nuclear phase diagram terrestrially by perturbing the system from its saturation point and attempting to “freeze-out” the effects of the symmetry energy before the system returns to saturation. For example in isospin diffusion experiments the interaction time between target and projectile nuclei is kept below the isospin equilibration time (Approximately: $100 - 200\,\mathrm{fm/c}$ - See Figure 1.5) and therefore only partial equilibration is achieved and the isospin content is locked into the nuclei of the exit channel on separation, providing a snapshot of the equilibration process.

In order to study the symmetry energy in a quantitative way, one must construct experimentally verifiable quantities which are sensitive to the symmetry energy and which are ideally insensitive to other competing effects which may make the extraction of the symmetry energy information much more difficult. For the purposes of this report only the two principal experimental observables of the INDRA-VAMOS experiment will be discussed, as there are a great many and not all of them are relevant here.

1.3.1 Isotopic scaling

Isotopic scaling (or isoscaling) refers to a particular phenomenon concerning the yields of detected fragments in multi-fragmenting systems. It has been observed that the ratio of the yield of a particular fragment in two separate reaction systems, which differ only by their mass, follows an isotopic scaling law [26, 27]:

$$R_{21}\left(N,Z\right) = \frac{Y_2\left(N,Z\right)}{Y_1\left(N,Z\right)} = C\exp\left(\alpha N + \beta Z\right) \tag{1.10}$$

Where C is a normalisation constant, α and β are the isoscaling parameters and the values 1 and 2 denote the two reaction systems (By convention $A_2 > A_1$). At first glance it is not immediately apparent how this observable is sensitive to the symmetry energy, however if one delves into the statistical models one finds that the α parameter is directly related to the symmetry energy parameter C_{sym}:

$$\alpha = \frac{4C_{sym}}{T}\left[\left(\frac{Z_1}{A_1}\right)^2 - \left(\frac{Z_2}{A_2}\right)^2\right] \tag{1.11}$$

Where T is the temperature of the source and the $(Z/A)^2$ terms are the source asymmetries of the two systems[28, 29].

For light fragments ($Z < 10$), generally speaking the only fragments which have been well identified, the parameters of α and β are the same (See Figure 1.3), the accuracy of which is verified with the following equations (See right panel of Figure 1.3).

$$S(N) = \frac{Y_2(N,Z)}{Y_1(N,Z)}\exp(-\beta Z), \quad S(Z) = \frac{Y_2(N,Z)}{Y_1(N,Z)}\exp(-\alpha N) \tag{1.12}$$

It is believed that this isoscaling phenomenon is a general property of multi-fragmenting systems [31] and is present in both statistical and dynamical models. It has been extensively studied in recent years and one of the main intentions for the INDRA-VAMOS experiment is to extract the surface and volume symmetry behaviour of the C_{sym} parameter [32].

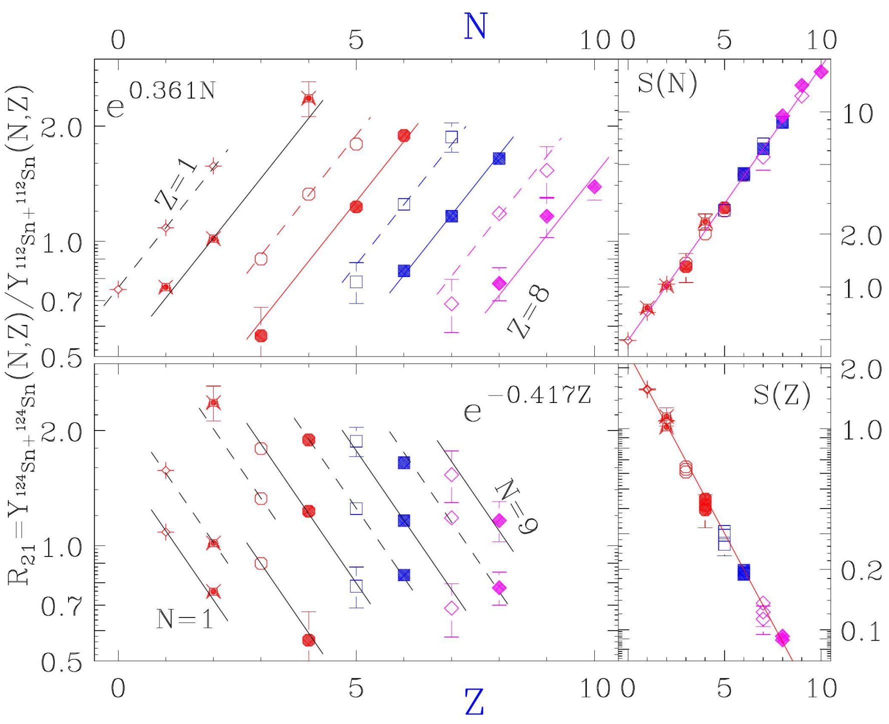

Figure 1.3: Isotopic scaling observed in $^{112}Sn + ^{112}Sn$ and $^{124}Sn + ^{124}Sn$ reactions at 50 AMeV by the MSU group [30]. In the panel on the right it is clear that the isoscaling parameters are equal for the low mass nuclei ($Z < 10$) - see Equation 1.12).

1.3.2 Isospin Diffusion

As stated earlier, the isospin equilibration process between two nuclei can be interrupted before isospin equilibration is complete and therefore provide a snapshot of the equilibration process. There are two processes which drive the migration of isospin between the two nuclei: isospin drift and isospin diffusion. The former, driven by density gradients, the latter by isospin gradients. The resulting neutron and proton currents can be evaluated using the following equation [33, 34]:

$$\mathbf{j}_{p/n} = D^{\rho}_{p/n}\nabla\rho - D^{\delta}_{p/n}\nabla\delta \tag{1.13}$$

Where $D^{\rho}_{p/n}$ and $D^{\delta}_{p/n}$ are the drift and diffusion coefficients respectively. The following pair of expressions outline the symmetry energy dependence of both the drift and diffusion mechanisms [34]:

$$D^{\rho}_{n} - D^{\rho}_{p} \propto 4\delta\,\frac{\partial S_2}{\partial\rho} \tag{1.14}$$

$$D^{\delta}_{n} - D^{\delta}_{p} \propto 4\rho\, S_2 \tag{1.15}$$

It is clear from these expressions that the isospin drift (Equation 1.14) is governed by the density dependence (gradient) of the symmetry energy ($S_2(\rho)$). The isospin diffusion however is governed by the numeric value of the symmetry energy for a given density (Equation 1.15).

In order to measure the effect of isospin transport, theorists formulated what is known as the isospin transport ratio [35]. A quantity which has been specifically devised to enhance the effects of isospin transport and to remove undesirable effects by strategic cancellation. In this way the Coulomb effects as well as those of pre-equilibrium emission and secondary decay are minimised. To construct this ratio one requires data from four different reaction systems with the same bombarding energy. These reaction systems cover all possible permutations for the reaction of two nuclei which differ only by their mass (isospin). Two of the reaction systems must be symmetric and the other two asymmetric (transposing the projectile and target): A+A, A+B, B+A and B+B.

$$R^x_{P,T} = \frac{2\left(x^M - x^{eq}\right)}{\left(x^H - x^L\right)} \tag{1.16}$$

Where x represents an experimental observable with a sensitivity to the asymmetry (such as the isoscaling: R_{21}). H and L refer to the two symmetric reactions (neutron rich and neutron poor respectively) and M refers to the asymmetric reaction. $x_{eq} = (x_L + x_H)/2$. This ratio takes the values ± 1 when no isospin transport has taken place and 0 when complete equilibration has been reached (See Figure 1.4). The sensitivity of the isospin transport ratio to the symmetry energy can be seen in Figures 1.4 and 1.5. Higher values for the symmetry energy correspond to faster equilibration, which naturally effects the degree of isospin transport.

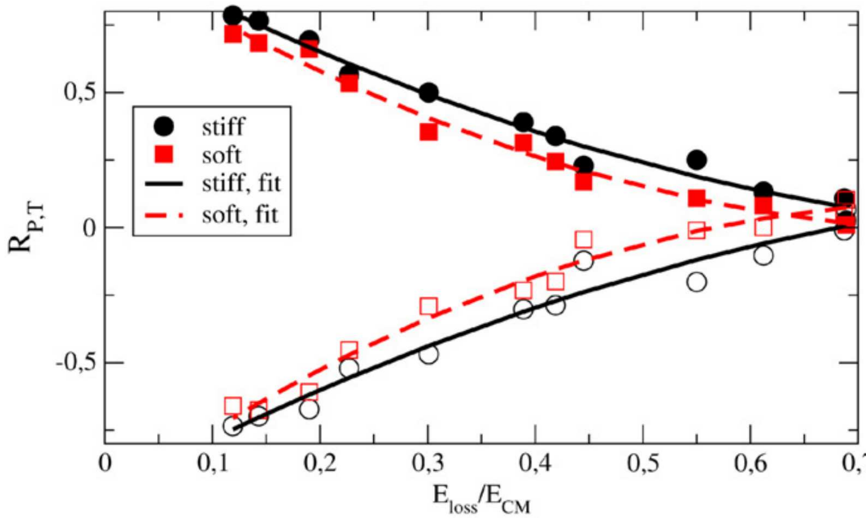

Figure 1.4: The isospin transport ratio as a function of the relative energy loss for different parametrisations of the density dependence of the symmetry energy. Taken from [36]. Soft and Stiff refer to the “flexibility” or the “resistance” of the symmetry energy to changes in density, for more details see the aforementioned paper.

In peripheral and semi-peripheral collisions (as in the INDRA-VAMOS experiment)

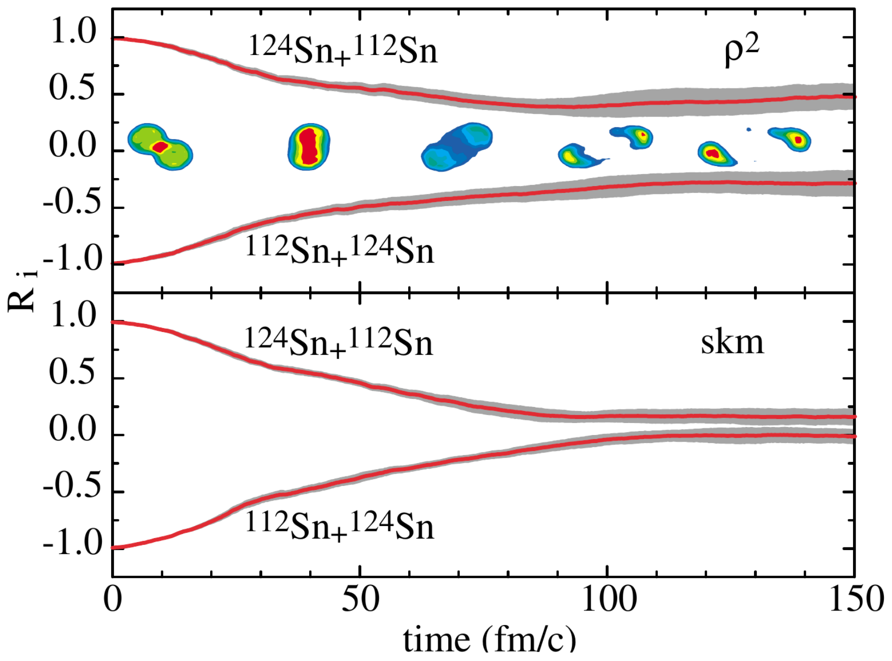

Figure 1.5: The isospin transport ratio as a function of time for two different density parametrisations of the symmetry energy - $Sn + Sn$ collisions at 50 AMeV. Taken from [37]. Notice that the equilibration time can vary significantly depending on the density dependence of the symmetry energy. $300\,\mathrm{fm}/c = 10^{-21}$ s.

isospin transport takes place via a low density "neck" region which acts as a bridge between the two nuclei. The low density of this neck region implies that the isospin transport will be sensitive to the low density behaviour of the symmetry energy.

1.4 Current Constraints

By extracting experimental constraints on the symmetry energy, theoretical models can be validated or invalidated. Of course the constraints which are extracted from the experimental data tend to be model dependent themselves but by combining as many different studies and theories (using different nucleon-nucleon interactions, momentum dependence etc.) as possible, a model independent understanding of the symmetry energy and its effects should be possible. As difficult as the situation may be, there are currently no symmetry energy observables which can be constructed to allow for a model independent extraction of the symmetry energy.

The current constraints from both terrestrial and astrophysical sources are summarised in Figure 1.6. At the moment the L parameter appears to be constrained to the region $44 - 66$ MeV but further constraints are still needed. This figure serves to demonstrate how the collaboration between different experimental and theoretical methods can constrain the density dependence of the symmetry energy and therefore supply theorists with a more stringent set of parameters with which to improve their models.

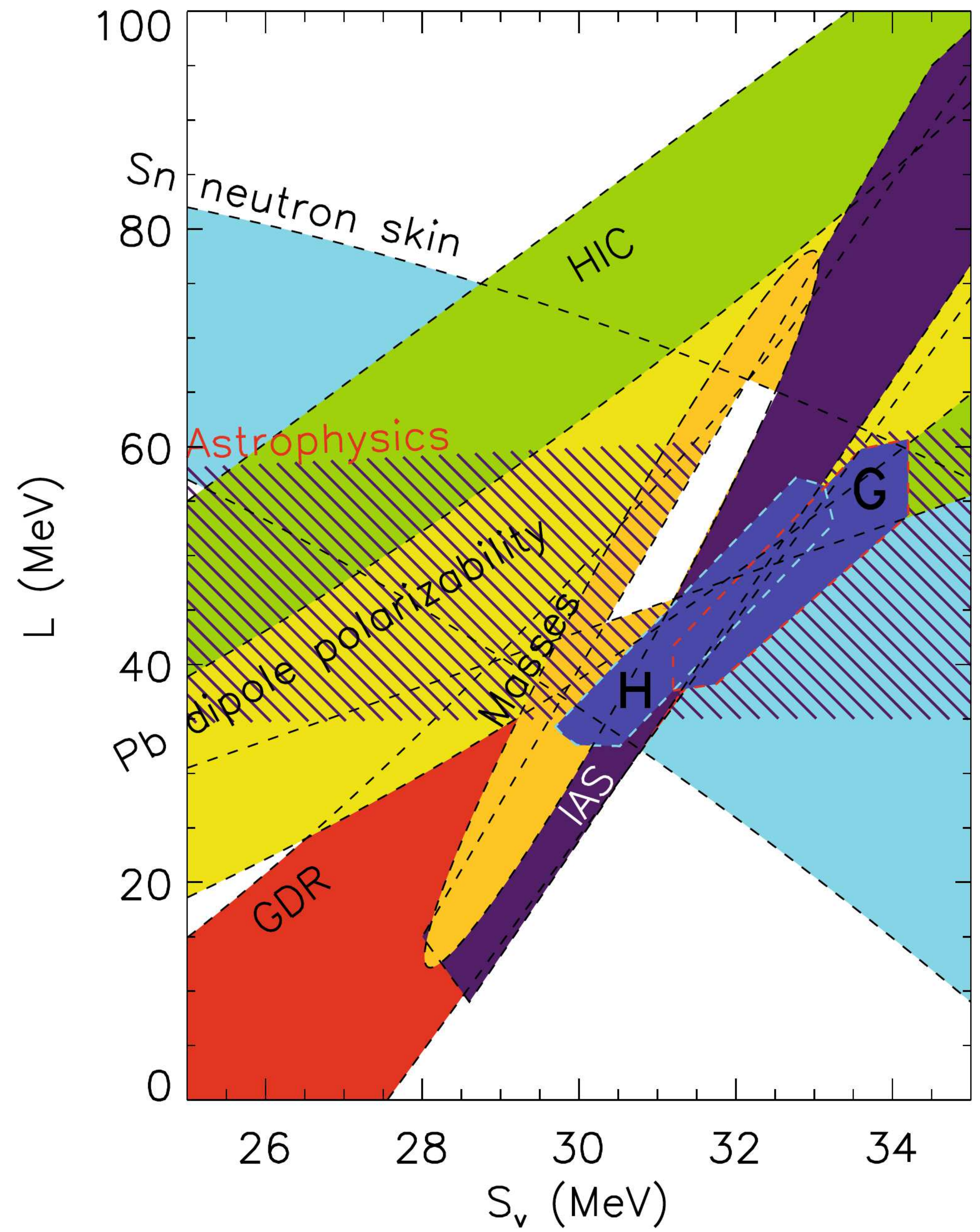

Figure 1.6: Summary of the current constraints on the symmetry energy, taken from [2]. The y-axis is the L parameter of Equation 1.8 and the x-axis is the symmetry energy at saturation density, $S_v \equiv S_2(\rho_0)$. The white region denotes the area of agreement for the experimental constraints. All of the regions are fairly self-explanatory except for those areas named "H" and "G" which represent the constraints extracted by two particular theoretical studies of neutron matter, for more details see the aforementioned paper.

1.5 The INDRA-VAMOS Experiment

There is a general complication when it comes to comparing experimental data and theoretical predictions. The problem is that theoretical models generally only predict the properties of hot reaction fragments prior to secondary decay and these fragments can not be measured experimentally. Unfortunately the primary fragments of the reaction tend to de-excite long before they are detected. It is therefore necessary to pass the theoretical data through secondary decay filtering codes in order to de-excite the fragments. Only by doing so can the experimental data and theoretical predictions be compared. This means that the extracted symmetry energy constraints rely upon the accuracy of the secondary decay code. The terminology for the fragments is modified in light of this situation such that the projectile-like *residue* refers to a de-excited fragment and the projectile-like *fragment* refers to the hot nucleus prior to secondary decay.

Prior to the INDRA-VAMOS experiment the only constraints on the symmetry energy came from experiments which detected either the light fragments of the reaction or the projectile-like residue but not the complete event. The INDRA-VAMOS experiment solves this problem by coupling the INDRA multi-detector to the large acceptance, high resolution spectrometer known as VAMOS. This unique experiment allows the high resolution identification of the projectile-like residue which can then be combined with the light fragments identified in INDRA and the complete event can be reconstructed.

By recording the complete event, fewer assumptions need to be made regarding the isospin content of the projectile-like fragment (hot primary fragment of the reaction) as have been made in previous experiments [37, 38, 39, 40]. Such assumptions are contradictory, as the isospin content is precisely what is to be used when inferring the symmetry energy constraints. The INDRA-VAMOS experiment was designed specifically to minimise this problem. Note that there is still an issue of a similar nature in the INDRA-VAMOS experiment as free neutrons can not be detected, however this is a much less destructive problem as the free neutron content can be inferred as missing mass and of course the neutrons which are bound within the detected fragments are properly accounted for.

The experiment uses a stable calcium projectile and a stable calcium target, the isotopes of which are chosen to be at the extremes of isospin ($A = 40, 48$). The reactions are performed close to the Fermi energy at 35 AMeV (the Fermi energy is typically around $37 - 38$ AMeV) so that a significant proportion of the available centre of mass energy is available to drive the thermalisation of the constituent nucleons and to ensure that the contact time of the two nuclei is similar to the interaction time. This should mean that in peripheral and semi-peripheral collisions there will be a detectable degree of isospin equilibration but not a complete thermalisation of the nucleons. It should therefore be possible to construct the isospin transport ratio and to evaluate the isoscaling in order to extract constraints on the symmetry energy at or below the saturation density of nuclear matter. This will be achieved by comparing the experimental data with transport code models such as Anti-symmetrised Molecular Dynamics (AMD) or Boltzmann-Uehling-Uhlenbeck (BUU) type models.

The E503 experiment was performed in April 2007 at the GANIL facility in Caen, France, it ran over the course of a couple of weeks. The sister experiment to E503 (E494s) ran immediately afterward using the same setup but with a different set of focal plane detectors in VAMOS. For an overview of symmetry energy studies using INDRA the reader is referred to [41].

Chapter 2

Transport Models

There are many theoretical models which can be used to study nuclear matter and the dynamics of heavy ion collisions. The range of validity and techniques used differ from one model to the next and a complete discussion is beyond the scope of this work. However for the conditions of interest in the INDRA-VAMOS experiment (Fermi energy: 35 AMeV, ~ 8.6 AMeV available in the centre of mass) the two most prominent approaches currently favoured are those of either Boltzmann-Uehling-Uhlenbeck based Stochastic Mean Field models or Anti-symmetrised Molecular Dynamics models.

The formulation of transport theories in the nucleonic domain is difficult as the system cannot be treated as purely classical (the de Broglie wavelength of the nucleons is not negligible) and a complete quantum mechanical description is not necessary (the de Broglie wavelength of the nucleons is still small compared to the system size). Under these circumstances quantum effects are present alongside classical behaviour and the transport theories take on what is known as a semi-classical character (Classical behaviour complemented by Fermi-Dirac statistics).

For an overview of theoretical developments in this energy domain the reader is strongly encouraged to consult [42] and its references. The short descriptions given hereafter are an attempt to condense several decades worth of theoretical advancements described therein so that the SMF and AMD models (as required by the INDRA-VAMOS experiment) can be better understood.

2.1 Stochastic Mean Field

Models based on the Boltzmann-Uehling-Uhlenbeck equation [1, p. 468-474] have been the mainstay of heavy ion collision dynamics studies for nearly 30 years. The starting point for these models is a simplistic classical description of the nucleus (a non-relativistic form of the Intra-Nuclear Cascade (INC) model [43]). Such a description yields a classical form of the Boltzmann equation:

$$\frac{\partial f\left(\boldsymbol{r},\boldsymbol{p},t\right)}{\partial t} + \frac{\boldsymbol{p}}{m} \cdot \frac{\partial}{\partial \boldsymbol{r}} f\left(\boldsymbol{r},\boldsymbol{p}\right) = I_{\text{coll}} \tag{2.1}$$

Where I_{coll} is the collision term:

$$I_{\text{coll}}[f] = \int d^3p_2\, d^3p_3\, d^3p_4\, W_{\text{INC}}(12,34) \times \left[f(\boldsymbol{r},\boldsymbol{p}_3,t)\, f(\boldsymbol{r},\boldsymbol{p}_4,t) - f(\boldsymbol{r},\boldsymbol{p}_2,t)\, f(\boldsymbol{r},\boldsymbol{p},t)\right] \tag{2.2}$$

With the collision rate defined as:

$$W_{\text{INC}}(12,34) = \frac{1}{m^2}\frac{d\sigma}{d\Omega}\delta(\boldsymbol{p}+\boldsymbol{p}_2-\boldsymbol{p}_3-\boldsymbol{p}_4)\,\delta\left(\frac{\boldsymbol{p}^2}{2m}+\frac{\boldsymbol{p}_2^2}{2m}+\frac{\boldsymbol{p}_3^2}{2m}+\frac{\boldsymbol{p}_4^2}{2m}\right) \tag{2.3}$$

When $I_{\text{coll}} = 0$ no collisions occur and the equation is known as the Vlasov equation. The exit channels of this model tend to be unbound in the energy range of interest however as there is no attractive mean field component and no Pauli blocking. Also note that the correlation is restricted to low relative momenta and energy (Dirac δ-functions in Equation 2.3).

In order to account for these effects a complete quantum mechanical form of Equation 2.2 must be determined. The Liouville-von Neumann equation provides the basis for deriving such an equation:

$$i\hbar\frac{\partial \boldsymbol{D}}{\partial t} = [\boldsymbol{H}, \boldsymbol{D}] \tag{2.4}$$

Where $\boldsymbol{D}$ is the density matrix containing all the information about the system, typically defined as $\boldsymbol{D} = \sum_\lambda |\Psi_\lambda\rangle\, q_\lambda\, \langle\Psi_\lambda|$ where q_λ denotes the probability that the system is described by the state $|\Psi_\lambda\rangle$. Using this equation directly is difficult as it is an A-body problem, however it is possible to reduce (and truncate) the description of the system using the BBGKY method (see [42]). This method uses reduced density matrices in order to generate a hierarchy of coupled k-body equations which allow the system to be truncated at any particular order, thereby approximating the Liouville-von Neumann equation. The reduced density matrices are determined by partial traces over the A-body density matrix ($\rho_{1\ldots A}$):

$$\rho_{1\ldots k} = \frac{A!}{(A-k)!}\text{Tr}_{k+1,\ldots,A}\,\boldsymbol{\rho}_{1\ldots A} = \frac{1}{A-k}\text{Tr}_{k+1}\,\boldsymbol{\rho}_{1\ldots k+1} \tag{2.5}$$

In doing so the equations of the BBGKY hierarchy take the form:

$$i\frac{\partial \boldsymbol{\rho}_{1\ldots k}}{\partial t} = \sum_{i=1}^{k}\left[\boldsymbol{K}_i + \sum_{j<i}^{k}\boldsymbol{V}_{ij}, \boldsymbol{\rho}_{1\ldots k}\right] + \sum_{i=1}^{k}\text{Tr}_{k+1}\left(\left[\boldsymbol{V}_{ik+1}, \boldsymbol{\rho}_{1\ldots k+1}\right]\right) \tag{2.6}$$

Note that the equation of order k is now coupled to that of $k+1$. By truncating the hierarchy and considering only *two-body* interactions, under the assumption that the two-body density matrix can be decorrelated at a particular instant in time (such that the history of previous correlations is lost between collisions), one ultimately arrives at a quantum Boltzmann equation (For more details refer to [44]):

$$\begin{aligned} i\frac{\partial \boldsymbol{\rho}_1}{\partial t} - \big[\boldsymbol{K}_1 + \boldsymbol{U}_1, \boldsymbol{\rho}_1\big] = I_{\text{coll}} = 2\pi i \text{Tr}_2\big\{\big[\boldsymbol{G}_{12}\mathcal{A}_{12}\big(\boldsymbol{\rho}_1\boldsymbol{\rho}_2\big)\boldsymbol{G}_{12}^{+}\big(1-\boldsymbol{\rho}_1\big)\big(1-\boldsymbol{\rho}_2\big) \\ - \boldsymbol{G}_{12}^{+}\mathcal{A}_{12}\big(\big(1-\boldsymbol{\rho}_1\big)\big(1-\boldsymbol{\rho}_2\big)\big)\boldsymbol{G}_{12}\boldsymbol{\rho}_1\boldsymbol{\rho}_2\big]\delta\big(E_{12} - \boldsymbol{H}_{12}^{0}\big)\big\} \end{aligned} \tag{2.7}$$

Here $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ are the one-body density matrices, the matrix $\boldsymbol{G}_{12}$ is the Brückner matrix for the interaction, $\boldsymbol{U}$ the mean field ($\sim \text{Tr}_2\Re(\boldsymbol{G}_{12}\mathcal{A}_{12}\rho_2)$) and $\mathcal{A}_{12}$ is the anti-symmetrization operator. Note that the mean field effect has been included and that Pauli blocking factors $(1-\boldsymbol{\rho})(1-\boldsymbol{\rho})$ are now present in the equation.

This quantum Boltzmann equation in the *semi-classical* regime can be reduced to the following where the one-body density matrices have been replaced by one-body phase space distributions:

$$\frac{\partial f_1}{\partial t} + \{f_1, h_c\} = I_{\text{coll}}^{\text{UU}} \tag{2.8}$$

Using the notation that $f_i = f(\boldsymbol{r}, \boldsymbol{p}_i, t)$ and the collision term is of Uehling-Uhlenbeck type [45]:

$$I_{\text{coll}}^{\text{UU}} = \frac{\nu}{h^9}\int d\boldsymbol{p}_2\boldsymbol{p}_3\, d\boldsymbol{p}_4\, W(12,34)\big[\big(1-f_1\big)\big(1-f_2\big)f_3f_4 - \big(1-f_3\big)\big(1-f_4\big)f_1f_2\big] \tag{2.9}$$

Here ν specifies the degeneracy and the collision rate is given by:

$$W(12,34) = \frac{1}{m^2}\frac{d\sigma}{d\Omega}\delta\big(\boldsymbol{p}_1 + \boldsymbol{p}_2 - \boldsymbol{p}_3 - \boldsymbol{p}_4\big)\delta\big(\epsilon_1 + \epsilon_2 - \epsilon_3 - \epsilon_4\big) \tag{2.10}$$

Where $\epsilon_i = \boldsymbol{p}_i^2/2m + \boldsymbol{U}_i = h_c(\boldsymbol{r}_i, \boldsymbol{p}_i)$, h_c is the classical Hamiltonian. Note the Pauli blocking factors $(1-f)(1-f)$ present in the collision term and that the correlation is still restricted to low relative energy and momenta via the Dirac-δ functions.

2.1.1 Stochastic Langevin Term

If the timescales of the various interaction processes occurring during the course of the collision can be justifiably separated (decorrelated) then the "relevant", i.e. long timescale interactions can be modelled independently of the those with shorter timescales, these can be approximated by a fluctuating noise term - this is known as the Langevin method.

The typical example given for a Langevin model is that of Brownian motion, a large massive particle moving within a sea of smaller rapidly moving particles. In the first approximation (as in the BUU type model thus far) the motion of the massive particle can be linked to a slow dissipative effect (a frictional force):

$$m\frac{d\boldsymbol{v}}{dt} = -\gamma\boldsymbol{v} \tag{2.11}$$

However such a model does not account for the observed random motion of the massive particle. The Langevin method consists of adding a fluctuation term and replacing the velocity vector with an ensemble of potential velocities (deterministic → stochastic). This term then acts to restore some of the random behaviour of the massive particle which has been neglected in the simple model above:

$$m\frac{d\boldsymbol{v}}{dt} = -\gamma\boldsymbol{v} + \boldsymbol{F}(t) \tag{2.12}$$

The nature of this fluctuation term is that it has a mean value of zero and a correlation function that is localised in time (Markovian), in other words a particular collision does not directly affect the subsequent collisions:

$$\langle\boldsymbol{F}(t)\rangle = 0,\ \langle\boldsymbol{F}(t)\boldsymbol{F}(t')\rangle = \alpha^2\delta(t-t') \tag{2.13}$$

Here α represents the intensity of $\boldsymbol{F}$. By considering the effect of the collision term in Equation 2.8 as a slow dissipative force, the same method can be applied to recover the neglected residual interactions (truncation in BBGKY) of the current model. To achieve this the one-body phase space distributions are replaced with an ensemble of potential one-body distributions and a Langevin fluctuation term is added to counteract the missing physics:

$$f(\boldsymbol{r},\boldsymbol{p},t) \rightarrow \{f_\lambda(\boldsymbol{r},\boldsymbol{p},t), \lambda = 1, \dots\} \tag{2.14}$$

$$\frac{\partial f_\lambda}{\partial t} + \{f_\lambda, h\} = I_{\text{coll}}^{\text{UU}}[f_\lambda] + \delta I_{\text{coll}}^{\text{UU}}(\boldsymbol{r},\boldsymbol{p},t) \tag{2.15}$$

Here the two body collision term $I_{\text{coll}}^{\text{UU}}$ is complemented by the additional fluctuating term $\delta I_{\text{coll}}^{\text{UU}}(\boldsymbol{r},\boldsymbol{p},t)$ to approximate the behaviour of the residual interactions and h denotes the mean field Hamiltonian [34]. One of the specific models that will be used in extracting symmetry energy constraints in the INDRA-VAMOS experiment will be that of the stochastic mean field (SMF) model [36, 46]. For an overview of the current abilities of the SMF model refer to [34] and for a better understanding of the how the fluctuations drive the reaction path [46, 47].

2.2 Anti-symmetrised Molecular Dynamics

The anti-symmetrised molecular dynamics (AMD) approach [48, 49] has its roots in classical molecular dynamics but is enhanced by quantum mechanical considerations to ensure effects such as Pauli blocking are properly taken into account. The model constitutes an N-body problem, unlike the BUU-type models which have their origins in the Time-Dependent-Hartree-Fock model (mean field). As such, the model has only become computationally feasible in recent years due to the increase in available processing power. Within AMD the many body state is now specified by a product of Gaussian single particle states:

$$|\Psi(1,\ldots,A)\rangle = \mathcal{A}_{1\ldots A}\left(\prod_{i=1}^{A}|\mathcal{G}_i\rangle\right) \tag{2.16}$$

For comparison, the Hartree-Fock many body state is expressed without the restriction on the form of the single particle states:

$$|\Psi_{HF}(1,\ldots,A)\rangle = \mathcal{A}_{1\ldots A}\left(\prod_{i=1}^{A}|\phi_i\rangle\right) \tag{2.17}$$

The fact that the AMD single particle states are spatially localised (Gaussian) results in many-body correlations [50]. The precise form of these wave packets is as follows [48]:

$$|\Phi(Z)\rangle = \frac{1}{\sqrt{A!}}\det[\varphi_i(j)], \varphi_i = \phi_{\boldsymbol{Z}_i}\chi_{\alpha_i} \tag{2.18}$$

Where the complex variables $Z \equiv \{\boldsymbol{Z}_i; i = 1,\ldots,A\}$ represent the centroids of the wave packets, “det” is the Slater determinant (imposing the asymmetry) and χ_{α_i} are the spin/isospin states for the nucleons ($|p\uparrow\rangle$, $|p\downarrow\rangle$, $|n\uparrow\rangle$, $|n\downarrow\rangle$). The AMD single particle states take the following spatial form:

$$\langle \boldsymbol{r}|\phi_{\boldsymbol{Z}_i}\rangle = \left(\frac{2\nu}{\pi}\right)^{3/4}\exp\left\{-\nu\left(\boldsymbol{r} - \frac{\boldsymbol{Z}_i}{\sqrt{\nu}}\right)^2 + \frac{1}{2}\boldsymbol{Z}_i^2\right\} \tag{2.19}$$

Where ν is the width parameter. The time evolution of the system (wave packet centroids) is achieved using the time dependent variational method [42, 49] (minimisation of the energy given the functional form of the state):

$$\delta\int dt \frac{\langle\Phi(Z)|\left(i\hbar\frac{\partial}{\partial t} - \boldsymbol{H}\right)|\Phi(Z)\rangle}{\langle\Phi(Z)|\Phi(Z)\rangle} = 0 \tag{2.20}$$

Both the SMF and AMD models share a similar premise: to take a two-body interaction and approximate the residual interactions through an additional Langevin-like fluctuation term. The way that this is achieved is however different in the two models. In SMF, as mentioned above, the residual interaction is simulated by stochastic perturbations of the one-body density distributions, whereas in AMD it is achieved by stochastic perturbations of the wave packets in phase space [50].

AMD models tend to be more computationally intensive and to compound this problem the imposed anti-symmetrization condition means that the $\langle \boldsymbol{r}_i\rangle$ and $\langle \boldsymbol{p}_i\rangle$ coordinates of the model can not be interpreted as the classical position and momentum [42, 48]. These values have to be interpreted when passing into or out of the collision algorithm.

2.3 Regarding Experimental Observables

Comparisons have been made between the two transport models SMF and AMD [50, 51] and an observed difference occurs for the number of pre-equilibrium particles emitted from light fragments ($Z < 3$). The many body correlations in AMD also affect the dynamics of the collision with fragment formation occurring on shorter timescales than in SMF [50].

In any case it seems likely that both SMF and AMD transport codes will be used with the INDRA-VAMOS data so that further comparisons between the fragmentation mechanisms of the two models can be made [51]. For comparison with the experimental data, the results of both models will need to be de-excited by secondary decay codes such as SIMON, an "after-burner" code now embedded in HIPSE [52, 53] or GEMINI [54] and this is likely to reduce any differences between the two models. The de-excitation procedure does tend to weaken the observed isospin effects [39] and this is unavoidable, it can only be offset by using beams with a wider isotopic range in order to enhance the isospin effects (Radioactive Ion Beams).

There are also additional difficulties when comparing experimental and theoretical data, one of the main problems stems from attempting to determine experimentally the global parameters of the reaction on an event-by-event basis, such as the impact parameter and reaction plane. The impact parameter for example cannot be determined event-by-event, it must be estimated using an observable presumed to be correlated with the centrality of the collision (e.g. Transverse kinetic energy, particle multiplicities, ...). Therefore it is difficult to compare experimental data with theoretical predictions of a specific impact parameter.

Chapter 3

Experiment Hardware and Procedures

3.1 The INDRA Multi-Detector Array

The acronym “INDRA” is derived from the French phrase “**I**dentification de **N**oyaux et **D**étection avec **R**ésolutions **A**ccrues” which translates into English as “Identification of Nuclei and Detection with Increased Resolution”. INDRA is a light charged particle multi-detector with low detection thresholds and high resolution, it is highly segmented and has a large solid angle coverage ($\Omega \simeq 0.9 \cdot 4\pi$). This section is an attempt to summarise the information already found in references [55] and [56], for a much more in depth and complete report on how INDRA was constructed, the reader is referred to the aforementioned papers.

3.1.1 Multi-detector Configuration

INDRA is composed of 144 multi-stage identification telescopes made from Silicon, Caesium Iodide, Ionisation Chamber and Phoswich detectors, all housed within 17 ring-like structures (See Figure 3.1). These rings serve to segment the polar angles asymmetrically in the laboratory frame of reference and their composition varies significantly with the ring number. Each ring is additionally segmented, symmetrically, in the azimuthal direction.

A summary of the composition of the INDRA modules is given in Table 3.1 but be aware that this is for demonstrative purposes only, as it represents the configuration of INDRA in the first experimental campaign and not that of the INDRA-VAMOS experiment. The reason for this is simply because it displays INDRA in its entirety, before the coupling modifications which were required to accommodate VAMOS. The configuration for INDRA in the INDRA-VAMOS experiment can be found in Table A.1.

The reason for the asymmetric/symmetric segmentation of the polar/azimuthal angles in INDRA comes purely from the kinematics of the reaction products as observed

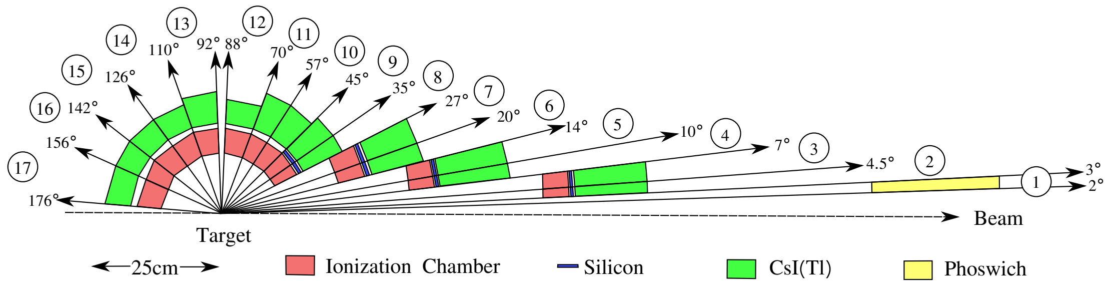

Figure 3.1: A "cut-away" of the INDRA multi-detector (Colourised from [55]). INDRA is symmetric about the axis defined by the beam and so only the top half of it is shown here. The variation in azimuthal segmentation is a necessity of the reaction kinematics, to even out the event rates in the different detector volumes.

in the laboratory frame of reference. In this frame, a moving beam impinges upon a static target and so the reaction products have an additional centre of mass velocity which acts to forward focus them towards the smaller polar angles (towards the beam). The result is that for a uniformly segmented detector the counting rates would be much greater in the forward angle segments than they would be in the backward ones, simply because there is a greater density of reaction fragments present there. This would lead to an unacceptable pile-up and distortion of the front-end detector signals. In order to counter this effect one requires a greater degree of granularity in the forward polar angles, as well as detectors possessing shorter recovery times the closer one gets to the beam (to minimise the pile-up). In the azimuthal direction, however, the reaction products are produced isotropically in the laboratory frame of reference and so the segmentation here can simply be symmetric.

There are more discussions on the reasoning behind the resulting segmentation of the array which can be found in [55]. In essence however it stems from a desire to optimise the competition between increasing the granularity and the resulting reduction of solid angle coverage due to the extra mechanical support required (inactive material).

The mounting of the detectors is certainly not trivial, as the signal paths for the front-end electronics must be minimised and the solid angle of the active detector material must be maximised. Typical constructions are shown for rings 4-5 in Figure 3.2 and rings 8-12 in Figure 3.3. The shapes of the active detector materials are obviously dictated by the geometry and the pre-amplifiers and photo-multiplier bases are mounted on the outer frame and as close to the detecting materials as possible.

	Phoswich NE102-NE115							
Ring No.	θ_{min} [deg]	θ_{max} [deg]	N	$\Delta\phi$ [deg]	e NE102 [mm]	e NE115 [mm]	$\Delta\Omega$ [msr]	d [cm]
1	2	3	12	30	0.5	250	0.37	130

	CsI(Tl)						Si	Ionisation Chamber				
Ring No.	θ_{min} [deg]	θ_{max} [deg]	N	$\Delta\phi$ [deg]	e [mm]	$\Delta\Omega$ [msr]	e [μm]	$\Delta\phi$ [deg]	N	n CsI(Tl)	d [cm]	$\Delta\Omega$ [msr]
2	3	4.5	12	30	138	0.74	300	30	12	3	65.4	2.9
3	4.5	7	24	15	138	1.01	300					
4	7	10	24	15	138	1.70	300	30	12	4	38.4	10.3
5	10	14	24	15	138	3.21	300					
6	14	20	24	15	97	7.01	300	30	12	4	25	37.7
7	20	27	24	15	97	11.2	300					
8	27	35	24	15	90	15.8	300	30	12	4	12	86.0
9	35	45	24	15	90	26.4	300					
10	45	57	24	15	76	39.6	none	30	12	4	12	183
11	57	70	24	15	76	50.3	none					
12	70	88	24	15	48	81.0	none	30	12	2	12	155
13	92	110	24	15	60	82.3	none	45	8	3	12	240
14	110	126	16	22.5	50	93.5	none	45	8	4	12	338
15	126	142	16	22.5	50	73.1	none					
16	142	157	8	45	50	91.2	none	45	8	2	12	144
17	157	176	8	45	50	50.9	none					

Table 3.1: INDRA Configuration from the first experimental campaign (Reproduction from [55]). N: denotes the number of detectors in each ring, e: the thickness of the detector, $\Delta\Omega$: the solid angle, n: the number of CsI(Tl) behind an Ionisation Chamber, d: the distance from the target, θ: polar angle, ϕ: azimuthal angle. For the E503 configuration see Table A.1.

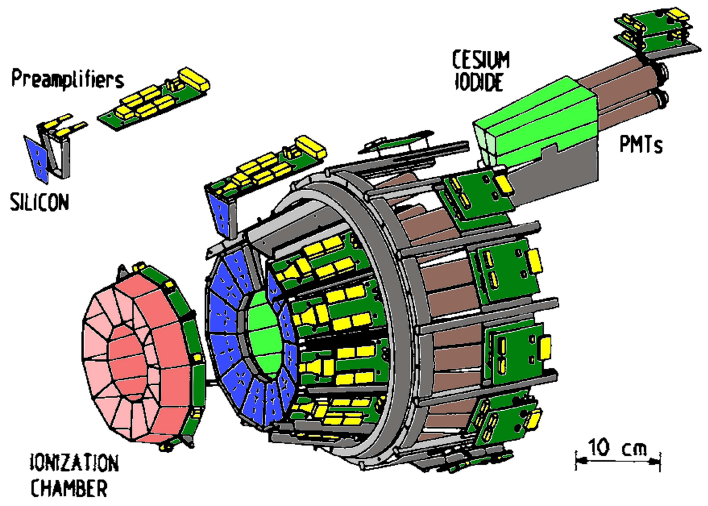

Figure 3.2: An example of the module assembly for the forward rings of INDRA. This particular figure shows the assembly for rings 4-5 (Colourised from [55]). Note the close proximity of the pre-amplifiers and photo-multiplier bases.

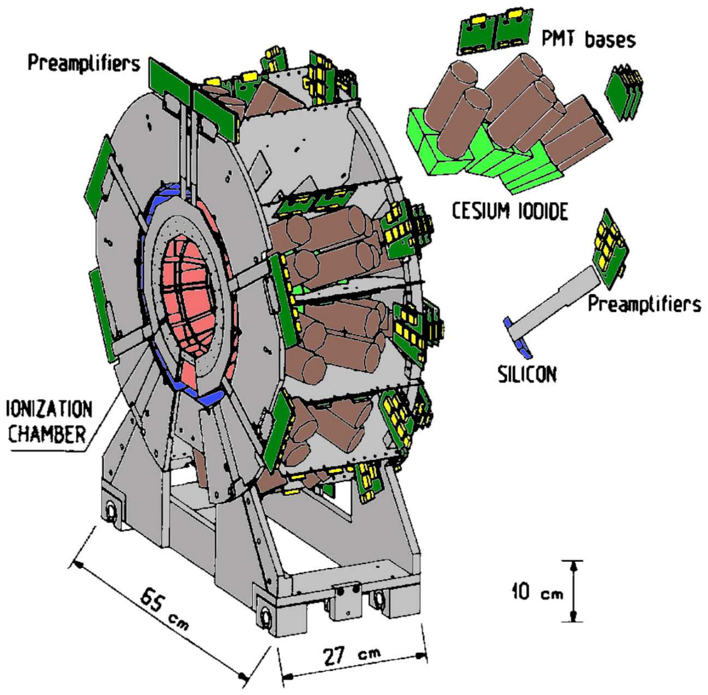

Figure 3.3: An example of the module assembly for rings 8-12 in INDRA (Colourised from [55]). Note the close proximity of the pre-amplifiers and photo-multiplier bases.

3.1.2 Components

Target Holder

The target holder has an additional purpose besides simply holding and positioning the target. It is also used to suppress electrons produced in the interaction between the beam and the target material. Due to the low detection thresholds required in INDRA, these electrons have to be suppressed by applying a positive bias to the target holder. This needs to be done carefully due to the close proximity of the entrance foils of the ionisation chambers. To achieve this, a hollow quartz cylinder was coated internally with a conductive material which allows the continuous operation of the target holder under experimental conditions with a 35 kV bias for several weeks.

The noise induced by the regulation of the high voltage power supply must also be filtered (low-pass) between the supply and the target holder, otherwise the noise induced in the ionisation chamber pre-amplifiers results in a degradation of the resolution.

Ionisation Chamber

There are 96 ionisation chambers present in the INDRA array and they make up the first detection layer. These chambers were designed to have very low detection thresholds and to provide uniform response (signal independent of position). Their exact construction varies with the ring number.

Each of the forward ionisation chamber arrays (Rings 2-7) consists of 12 trapezoidal gas cells arranged into a ring (See Figure 3.4). The housing of these rings is made from a single piece of fibreglass and the entrance window (cathode) is an aluminium coated mylar foil (2.5 μm), common to all of the gas cells. The exit window (anode) is of a similar construction but has separate electrodes for each cell, created by masking the mylar foil with a brass template before application of the aluminium coating.

Within each cell it is critically important to maintain a uniform electric field, as this ensures the invariant signal response of the detector (charge collection behaviour should be independent of position). To that end, field shaping rings were constructed using thick copper traces (1.5 mm traces with 2.5 mm pitch) etched on to printed circuit boards which were then glued on to the walls of each cell. While running field simulations for this construction it was noticed that non-uniformities in the electric field around the edges of the cells and around the anode resulted in a degradation of the resolution. To resolve this issue, a ground reference grid, common to all of the gas cells was connected to the last field shaping ring in each cell. This grid is made of 50 μm Copper-Beryllium wires with a spacing of 5 mm and is located 5 mm from the exit window of each cell.

In addition to keeping a uniform electric field in each of the cells it is also important to cycle the fill gas to ensure its detection characteristics remain the same. If the gas

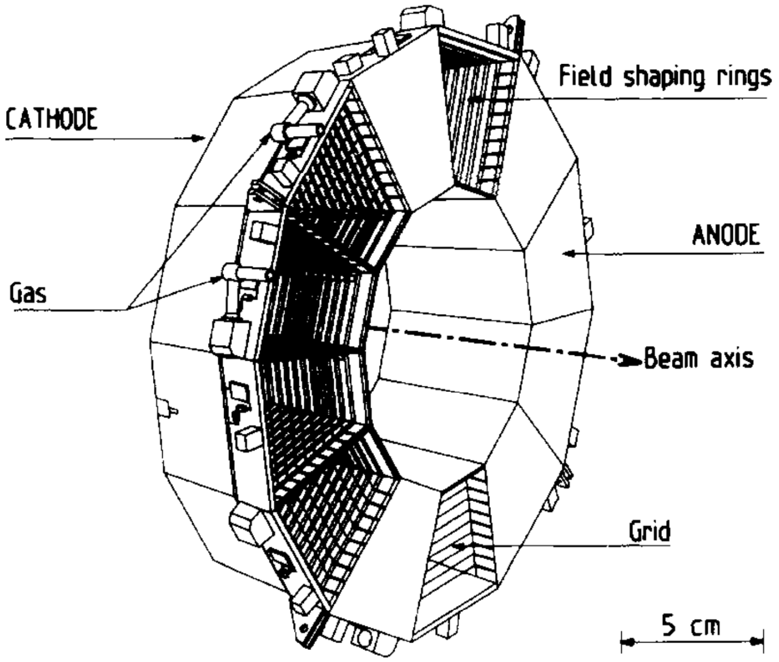

Figure 3.4: Construction of the ionisation chambers for rings 4-5. Taken from [55]. Note the two valves for the ingress and egress of the fill gas, the gas enters at one valve and circulates around the ring until it reaches the other, where it is extracted.

is not cycled its performance will degrade over time as it becomes contaminated by the interactions occurring within it. For this reason, in all but two of the walls separating the gas cells, holes were drilled to allow the circulation of the gas throughout the ring. The gas is then introduced through one cell, it circulates around the all the cells in the ring and then leaves the system at the final cell (adjacent to the entry cell). The pressure of this gas must be carefully tuned to maximise its detection ability while keeping in mind its effect on the downstream detectors. The pressure must also be kept as constant as possible, otherwise one would have to continuously re-calibrate. The fill gas for the ionisation chambers is C_3F_8 and is held at a pressure less than 60 mbar, typically 50 mbar.

The structure of the backward ionisation chamber arrays (Rings 8-17) is different (See Figure 3.5). In these rings there are 24 gas cells rather than 12 and it was decided that construction of the housing from a single piece of fibreglass was impractical. Ultimately it was separated into two parts, an anode part and a cathode part. The walls of the gas cells are composed of multi-layer printed circuit boards fixed together with glue which are also used to route the anode signals out to the pre-amplifiers at the back of the array. The field shaping rings are now common to all of the gas cells and the electrodes are segmented as they could not be glued on in one piece. Due to these complexities the pressure of the fill gas is kept lower than in the forward rings, below 50 mbar and typically 30 mbar. The holes in the cell walls had to be carefully considered to allow a full circulation of the fill gas, as only one entry point and one exit point are used.

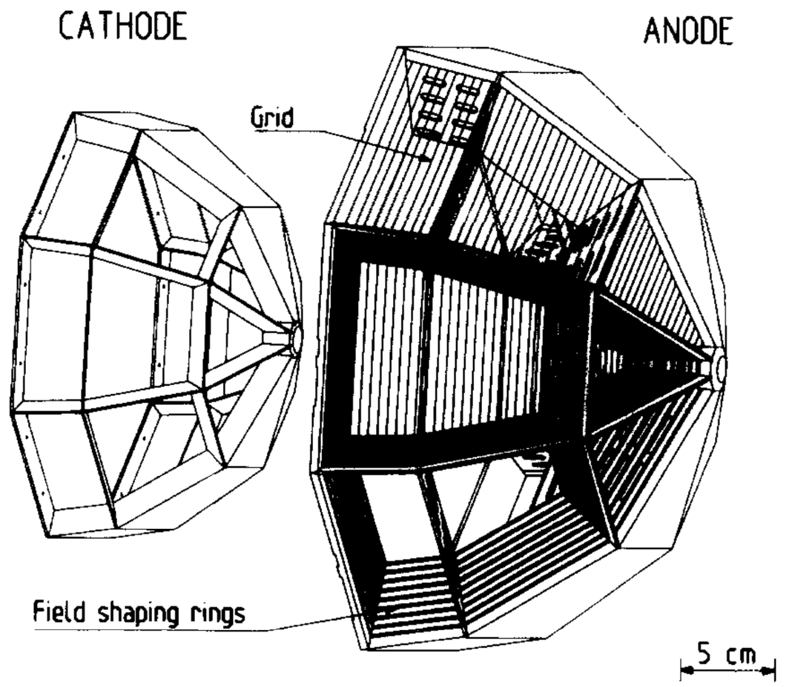

Figure 3.5: Construction of the ionisation chambers for rings 13-17. Taken from [55]. Note that in contrast to the forward rings, the construction had to be split into two halves.

Silicon

There are 180 silicon detectors in INDRA, each with a thickness of either 300 μm or 150 μm depending upon the ring number (For a complete list see Table A.2). The silicon detectors are present only in rings 2-9.

The silicon detectors represent the second stage of detection in INDRA and they are required to cover the exit windows of the ionisation chambers, while minimising the amount of inactive material required to house them. For this reason each silicon wafer is sub-divided into 3 or 4 pads (depending upon the ring number) to match the entrance windows of the downstream caesium iodide detectors, while at the same time maintaining a sufficient active area to cover the exit window of the ionisation chamber (See Figure 3.2).

The proximity of the anode pads of the preceding ionisation chamber to the silicon material presents a minor problem as it results in an unacceptable degree of capacitive coupling. To reduce this effect, the side of the silicon detector nearest the anode must be grounded. The pads had to be manufactured on the p-side of the detector and therefore the n-side had to be grounded. This means that any incident particles enter the silicon on the low electric field side. It is advantageous therefore to overdeplete as many of the silicon detectors in INDRA as possible so as to ensure the maximum possible electric field strength within them - this yields the most efficient charge collection. Wherever this was possible the detectors were over-biased by around 2-3 times their depletion voltage. For the detectors where this was not possible, they had to be carefully examined under experimental conditions with various bias voltages in order to characterise the effect this had on their energy measurements (For more details see [55]).

The silicon wafers are housed within a support frame made of four printed circuit boards. The three boards which make up the sides and base of this support have a thickness of 0.8 mm with the fourth board (the one nearest the pre-amplifier circuit board) having a greater thickness of 6 mm, for the simple reason that it obscures nothing and provides better support for the frame. The silicon wafer is then connected to the pre-amplifier by either a couple of circuit boards fastened to the top of this fourth, thicker, support board or by flat-flex circuits when the distance between the pre-amplifier and the wafer is larger (up to 20 cm).

Caesium Iodide

There are 324 caesium iodide detectors in the INDRA array. Each consists of a scintillation crystal of Thallium activated Caesium Iodide, together with its associated photo-multiplier tube.

The light signal produced in these caesium iodide crystals has two components. A "fast" component coming from the prompt decay of excited states within the crystal and a "slow" component caused by the decay of excited states which have no direct path to the ground state, resulting in a prolonged afterglow. The extended decay time of this signal (on the order of $7\,\mu$s [57]) mandates long pulse shaping times and therefore pile-up can become a significant problem. The crystal characteristics were chosen with this in mind. It also means that the crystals in the forward polar angles of INDRA must have shorter decay times (due to the higher counting rates), this is achieved by having a lower average light output for these crystals.

The crystals had their front and back faces polished and their sides and back face (excluding the window required for the photo-multiplier tube) wrapped in a mylar foil and then again with an aluminium coated mylar foil in order to reduce the leakage of light from within the crystal volume. The optical contact between the photo-multiplier tube and the crystal was achieved without the use of a light-guide to make the assembly easier. Instead the crystals were lengthened by $1.7 - 2$ cm (depending on the ring number) to mimic the effect of the missing light-guides.

A significant amount of experimentation and simulation went into the design and selection of the caesium iodide crystals as well as the choice of photo-multiplier tubes. Owing to the existence of 30 different crystal shapes in the array, a selection of four photo-multiplier tubes (two of which were specially designed and manufactured for the INDRA collaboration) of different diameters and varying number of dynode stages were chosen: 2 × 10-stage models, 2 × 8-stage models (INDRA specific).

In order to reduce cross-talk between the photo-multiplier tubes and the ionisation chamber charge pre-amplifiers the tubes were shielded using a grounded μ-metal (high magnetic permeability) tube, isolated from the photo-cathode.

Nitrogen Laser

In order to ensure the linearity of the scintillator detectors a Nitrogen laser ($\lambda = 337\,\mathrm{nm}$) is pulsed and the response of the various detectors is verified. A simple block diagram showing how the light is distributed to the detectors can be seen in Figure 3.6.

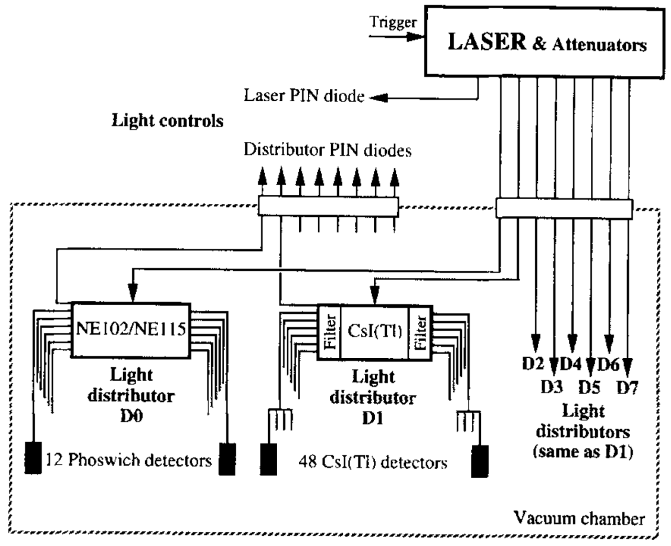

Figure 3.6: A block diagram which shows how the Nitrogen laser pulses are distributed to the scintillation detectors in INDRA. The Nitrogen laser is used to quantify and maintain the linearity of the scintillation detectors. Taken from [56].

3.1.3 Electronics

The electronics associated with INDRA is also highly specialised and it is worth spending some time to get a general understanding of the main design challenges and the hardware. It is important to remember that the design of the INDRA electronics has been influenced by a number of constraining factors.

From the outset it was known that the electronics had to constitute a very low noise system with a large dynamic range, situated as close as possible to the detectors while having a minimum number of connections [56]. It also had to provide remote software control of the module settings, compatibility with the GANIL VME [1] crates (used for

[1]Multi-processor crate running a real-time operating system used for building/filtering the event data and dispatching it to the machines used for control and data storage.

data acquisition) and had to allow the remote visualisation of internal module signals.

In addition to these, many further restrictions were imposed as the design progressed. Among them were such issues as the careful grounding of the system required to reduce interference and minimise crosstalk, the isolation of the high voltage supply lines and naturally the numerous difficulties that inevitably occur when the electronics partially operate inside the vacuum of the reaction chamber.

This section only provides a very brief overview of how the INDRA electronics were constructed and operate, it does not do justice to the level of customisation and the amount of work required to make them operational. For a more in depth report the reader is referred to [56].

Front-end Electronics

The front-end electronics have their own particular subset of requirements as they are needed to process different types of signal depending upon the type of detector with which they are associated.

The photo-multiplier base circuitry for example fulfills a very specific function. Its purpose is to supply charge to the dynodes to replace that which is lost through the photo-multiplication process. In order to maintain the stable operation of the photo-multiplier tube, the inter-dynode voltages must be kept as constant as possible, this means re-charging the dynode capacitors in the shortest possible amount of time. For more traditional scintillation materials this does not pose much of a problem but CsI(Tl) signals have a greater charge output [58, p. 238] and so require a larger current to replenish the upper stage dynode capacitors (the capacitance necessarily increases as the dynode chain is ascended). For this reason the upper stages of the traditional RC chain were replaced with transistors, these transistors act as switches which allow the full bias current to be used when charging the capacitors which supply the upper stages of the dynode chain (See Figure 3.7).

In doing so another benefit is conferred: the power consumption drops around 10-fold, which allows the photo-multiplier bases to be placed inside the reaction chamber (which is under vacuum) without additional cooling.

As a minor point of interest, the bias current needs to increase as one moves toward the forward angles in INDRA due to the higher counting rates experienced there. In other words, the increased rate at which charge is depleted in the dynode capacitors absolutely mandates that the current be increased to replenish them more quickly, in order to maintain the stability of the inter-dynode voltages and so guarantee the continued stable operation of the photo-multiplier tube.

Then there are the ionisation chamber and silicon charge pre-amplifiers. The circuitry for which naturally differs as each material has its own distinct charge collection characteristics which translates into differing demands on the pre-amplifiers for sensi-

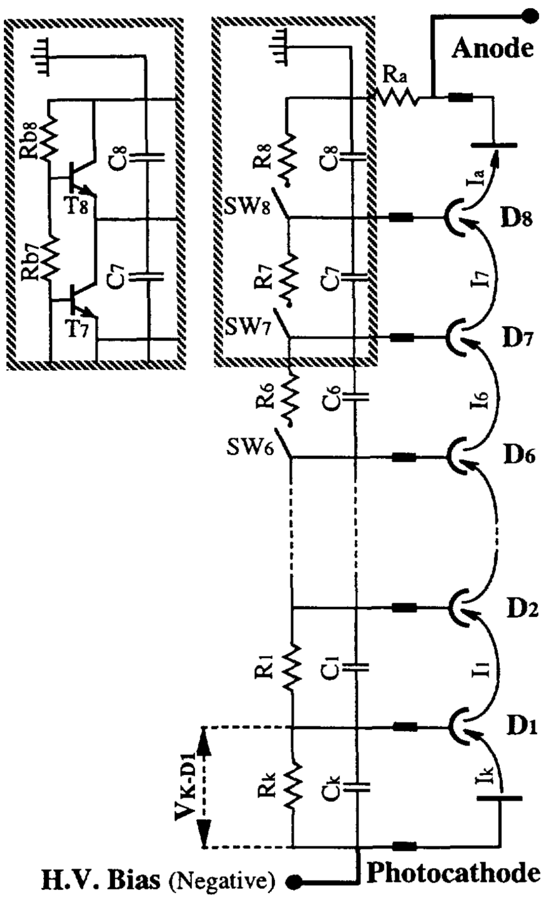

Figure 3.7: The CsI(Tl) photo-multiplier base circuitry for INDRA (Taken from [56]). The upper stages of the traditional RC chain have been replaced with transistors, these transistors act as switches allowing full bias current to be used when charging the upper stage dynode capacitors.

tivity, noise and dynamic range. The charge pre-amplifier for the ionisation chamber for example requires very high sensitivity and low noise, as it does not collect charge very efficiently. To achieve this sensitivity an exceptionally low feedback capacitance was required (0.22 pF), a value so low that no suitable component existed. Ultimately this feedback capacitance was achieved using a pair of parallel copper traces with a separation determined by comparison with a reference charge pre-amplifier of known sensitivity. All of the ionisation chamber charge pre-amplifier sensitivities were then tuned to within 1% of one another and the reference value. The sensitivity required for the silicon charge pre-amplifier, by contrast, was such that the feedback capacitance could be achieved with a standard component.

The operating principle of the charge pre-amplifier is however the same in both cases. Both circuits constitute a "cascode" amplifier, that is a transconductance amplifier (Output current proportional to input voltage) coupled to a current buffer (A unitary gain amplifier in this case, as it draws very little current but yields the same output voltage signal) which prevents the subsequent amplifier circuitry from excessively loading the pre-amplifier stage and interfering with its operation. In both cases the circuit is also terminated by a push-pull transistor amplification stage.

The ionisation chamber charge pre-amplifier (See Figure 3.8) works by passing the signal through a field-effect transistor and a fast-current amplifier (forming an effective transistor with a transconductance equal to that of the field-effect transistor multiplied by the gain of the amplifier) which, in combination with the bipolar junction transistor (forming an effective cascode amplifier) acts to amplify the signal with a gain on the order of 30000.

The silicon charge pre-amplifier (See Figure 3.9) by contrast works by passing the signal through an impedance correcting field-effect transistor and then through a couple of bipolar junction transistors acting as a cascode amplifier whose gain is fixed by varying the current source associated with the first transistor (of amplifying the pair) which alters its impedance. This methods produces a gain on the order of 100000.

The silicon charge pre-amplifiers demand a large dynamic range and this has an engineering consequence as it pushes their power consumption past the point where they can remain uncooled inside the reaction chamber (under vacuum). For this reason a water coolant system had to be accommodated in the INDRA design. The characteristics for each of the charge pre-amplifier circuits can be found in reference [56].

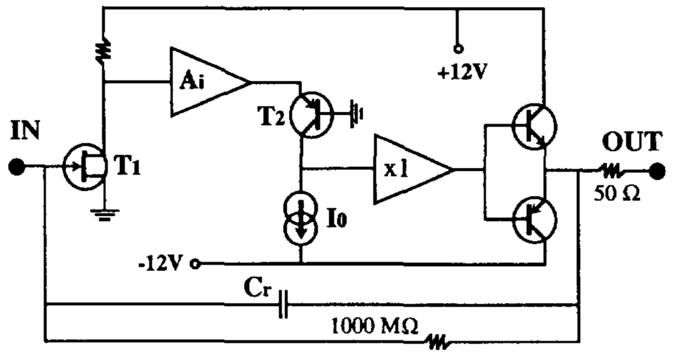

Figure 3.8: Schematic diagram of the Ionisation Chamber charge pre-amplifier circuitry. Taken from [56]. The transistor T_1, together with the fast-current amplifier A_i act in conjunction with the bipolar junction transistor T_2 to form a cascode amplifier with a gain on the order of 30000.

Signal Transmission

The transmission of the pre-amplifier and photo-multiplier (front-end) signals from within the reaction chamber to the amplifier and subsequent circuitry is also critically important. To route these signals from the vacuum of the reaction chamber out to the acquisition electronics in the beam-cave is no trivial matter and for that purpose a special flange was designed. Connections through this flange were made using standard

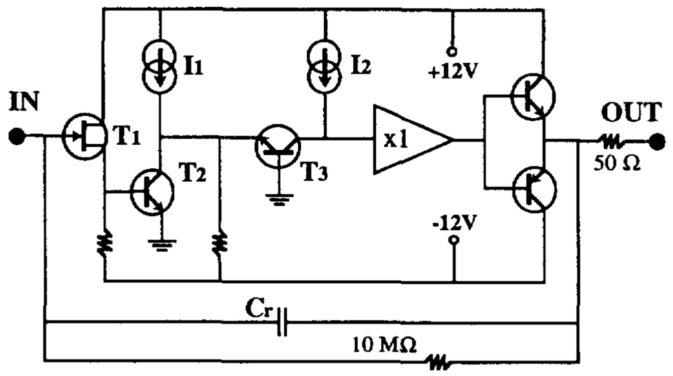

Figure 3.9: Schematic diagram of the Silicon charge pre-amplifier circuitry. Taken from [56]. In this case the two bipolar junction transistors T_2 and T_3 act as a cascode amplifier with a variable gain controlled by the current source I_1.

connectors (53 contact, 2.54 mm pitch) while exercising due regard for the minimisation of crosstalk between the signal lines and the isolation of the high voltage lines (in the case of the scintillation detector connections).

The signal line crosstalk was minimised by having each of the signal lines on the connector surrounded by connections to the ground reference. In the case of the ionisation chamber and silicon detectors this allows 16 signal lines on each connector. For the scintillation detectors the requirements for isolation (spacing) of the high voltage lines from one another results in enough space for 8 signal lines and 8 high voltage lines on each connector. These connectors allowed the coaxial cable of the acquisition electronics to be connected directly to the flanges (10 connectors on each flange) of the reaction chamber which are in turn connected to the front-end electronics. These flanges can be seen in Figure 3.17, at the base of the INDRA reaction chamber.

Any signal degradation induced by the transition between the signal line and the flange is insignificant under the operating conditions typically experienced in this setup. The flange also has another interesting function as it provides the all important electrical contact to the ground reference - the reaction chamber itself.

Acquisition Electronics

The electronics outside the reaction chamber perform many functions but in general it is here that the front-end signals are amplified, discriminated, validated by the trigger, converted into digital signals and then dispatched to the event builder which ultimately sends its output data to the data storage machines.

While designing the acquisition electronics for INDRA it was determined that traditional signal processing standards would not be sufficient and so a custom architecture was used, an extension to the VME standard called VXIbus (VME Extension for Instrumentation). One of the challenges in the design of this system was that the acquisition electronics had to be located within the beam cave, owing to the large number of channels and the desire to minimise the number of connections and reduce signal transmission distances. This means that the acquisition electronics are, of course, inaccessible during experimental operation. The system was therefore built around the need to route internal module signals of significance out to the control room and to allow the remote, software setting of internal module components.

The VXIBus standard allows many of the signal processing functions of traditional modules to be condensed into a relatively small space (minimising the connections), while still allowing the use of a master VME control crate for dealing with the slower CAMAC and high voltage modules. The VXIBus crates successfully condense the output lines for INDRA to just the VME computer links and 20 coaxial cables to allow the visualisation of signals in the control room. The internal signals of the modules are still available on buffered (protected) front panel connectors in the cave, for debugging purposes. A block diagram of a generic VXIBus crate, as used in the Silicon and Ionisation Chamber electronics, is shown in Figure 3.11.

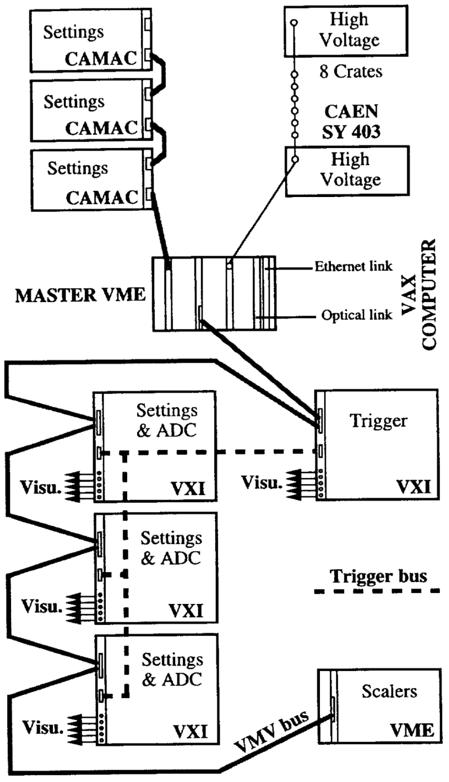

Figure 3.10: A schematic diagram of the INDRA acquisition electronics. Taken from [56]. Each crate provides buffered front panel connections for visualisation of the internal signals on oscilloscopes while in the beam cave ("visu").

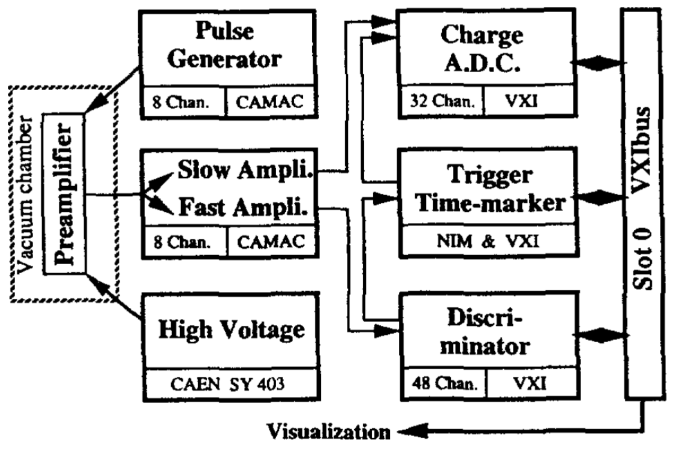

Figure 3.11: An example silicon and ionisation chamber module for INDRA. Notice the fast amplifier provides the timing signals used for the triggering and the slow amplifier is used for the energy measurements. Taken from [56].

3.2 The VAMOS Spectrometer

VAMOS (**VA**riable **MO**de **S**pectrometer) [59, 60, 61] is a magnetic spectrometer with a large solid angle coverage and momentum acceptance, it can operate in several different modes depending upon the nature of the reaction being studied. It belongs to a category of magnetic spectrometers known as "software" ray-tracing spectrometers (See below). Its purpose is to separate and identify the reaction products, precisely how it does so depends upon the operating mode and the detectors being used.

3.2.1 Multi-detector Composition

Broadly speaking, the VAMOS spectrometer has two stages to its operation, an ion-optic stage and a detection stage (See Figure 3.12). The ion-optic stage is made up of two focussing quadrupole magnets (horizontal and vertical), a Wien velocity filter (unused in dispersive mode) and a large acceptance dipole magnet. Its purpose is to separate any reaction fragments within its acceptance so that they can be resolved at the image plane (located downstream). The detection system varies significantly from one experiment to the another but it usually consists of position sensitive devices around and beyond the image plane with increasing stopping power the further downstream from the target. All of the VAMOS hardware is mounted on a rotating platform which allows the spectrometer (ion-optics and detector system) to be positioned at any angle relative to the target position within the range $0 - 100°$. The angle of deflection in the dipole magnet can be varied in the range $0 - 60°$. The momentum and angular acceptance of VAMOS in this experiment are $\pm 5\%$ and $\pm 4°$ respectively.

For the sake of simplicity, VAMOS is henceforth taken to operate only in one mode, the momentum-dispersion mode, where the reaction products are separated at the image plane according to their momentum-to-charge ratio, for a more complete description

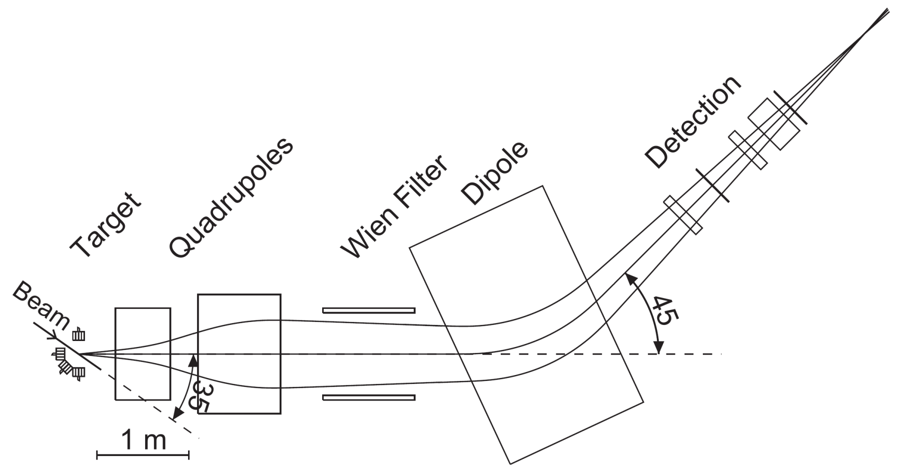

Figure 3.12: A schematic diagram of the VAMOS setup demonstrating a rotated VAMOS platform (35°) and a dipole deflection of 45°. Taken from [59].

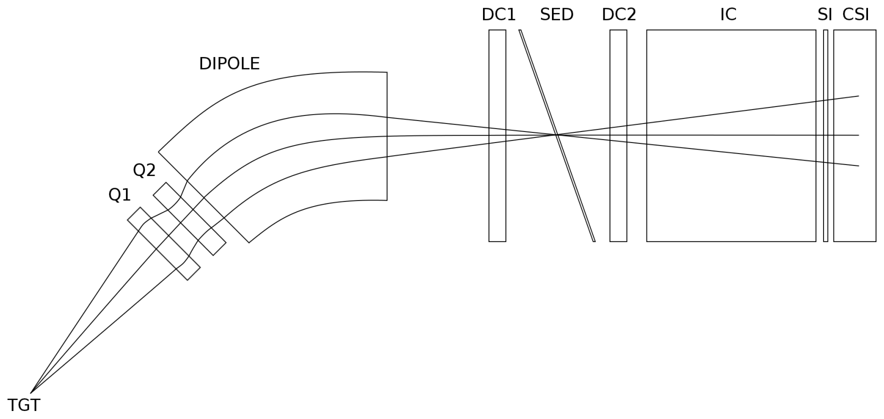

Figure 3.13: The VAMOS focal plane detection setup used in the INDRA-VAMOS (E503) experiment. It consists of two drift chambers, placed either side of an inactive secondary electron detector, an ionisation chamber, a silicon wall and finally a caesium iodide wall.

of the VAMOS spectrometer the reader is referred to reference [59].

The composition of the detection system (See Figure 3.13) in this experiment is as follows: two drift chambers located either side of the image plane, whose purpose is to provide the information necessary to determine the direction in which the fragments are travelling. Then, located as close as possible to the image plane and between the two drift chambers, is a secondary electron detector. This detector generally provides the time reference signal for VAMOS, however in this experiment it is *inactive* and the timing reference instead comes from the downstream silicon detectors. Following the second drift chamber is an ionisation chamber followed by the silicon and caesium iodide detector walls.

In order to determine the emission angles of the reaction products as they leave the target, a ray-tracing procedure is performed off-line, in software which tracks the trajectory of the particles back to the entrance window of VAMOS using ion optical calculations (field aberrations are taken into account to seventh order). The VAMOS hardware however does the heavy-lifting in this respect as it is designed to minimise the aberrations that need to be taken into account in software, nevertheless this procedure remains complex.

3.2.2 Components

Drift Chambers

There are two drift chambers present in the VAMOS detection system and these are located either side of the image plane. These detectors are position sensitive in both the dispersive (horizontal, x) plane and in the non-dispersive (vertical, y) plane (although not with the same resolution). This position sensitivity, coupled with the fact that the image plane resides between them, allows for an extrapolation of the trajectory of a particle $(x_1, y_1) \rightarrow (x_2, y_2)$ so that it is known at the image plane. This is necessary, as in order to transport a particle back through the magnet to the target position (and so to determine its trajectory as it leaves the target) its trajectory must be known *at the image plane*.

Both drift chambers, excluding any manufacturing differences, are identical (See Figure 3.14). They comprise a pair of cathodes separated by a Frisch grid and a set of anode wires, the bottom cathode is segmented in the dispersive plane to provide horizontal position sensitivity. The filling gas is isobutane (C_4H_{10}), typically at around 13 mbar of pressure. The region between the top cathode and the Frisch grid constitutes the active region of the detector and that between the Frisch grid and the segmented cathode, the amplification region. Within the active region a uniform electric field is applied whose effect is to induce any charge carriers produced in the ionisation trail of a passing particle to drift towards their respective electrodes.

Of the various charge carriers produced, the electrons have the greatest mobility and once they pass the Frisch grid they are accelerated towards the anode wires. The electric field in this region is such that the acceleration of these electrons causes further ionisation, which in turn produces more electrons which are then accelerated and so forth - an avalanche. These electrons are then collected at the anode and the fast component of this signal denotes the time of arrival for the electrons.

The amplification of the electrons within this region also induces a signal in the segmented cathode that makes up the base of the drift chambers. The charge distribution measured across the individual pads of the segmented cathode is then fit with a Gaussian distribution in order to determine its horizontal position. The vertical position is determined by multiplying the difference in time between the aforementioned signal, marking the time of arrival for the electrons at the anode wires and the reference signal

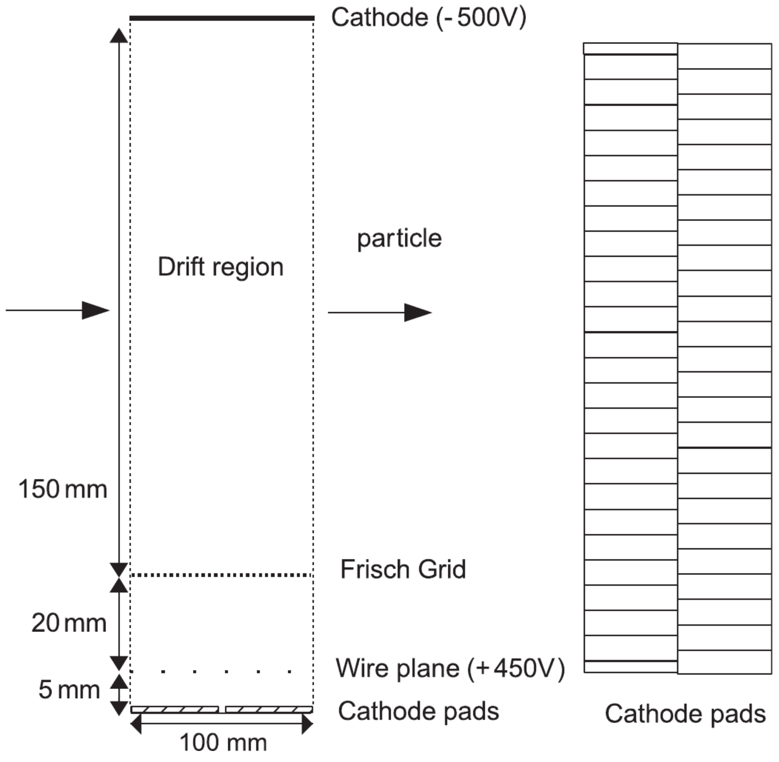

Figure 3.14: A schematic diagram of the VAMOS drift chambers. Taken from [59]. The region above the Frisch grid is the active volume of the chamber and the region below it is the amplification volume. The fast signal from the anode wires is used as the "time of arrival" for the electrons in the drift time calculation.

from the silicon detectors by the drift velocity of the electrons inside the fill gas:

$$y = v(t - t_0) \tag{3.1}$$

The segmented cathodes are each made of two planes of 64 pads, each of which is 6 mm wide and offset by half a pad in order to reduce the non-linearities which occur at the pad edges. The Frisch grid is made from 50 μm $\mathrm{Cu - Be}$ wires spaced at 2 mm in a direction perpendicular to that of the beam and is for all intents and purposes transparent to electrons. The anode wires are made of 20 μm Tungsten with a 10 mm spacing, positioned 20 mm below the Frisch grid. The top cathode is simply a copper plated sheet of glass.

Ionisation Chamber

The ionisation chamber used in VAMOS is a standard design. It consists of a cathode plate, a Frisch grid and a segmented anode. The active area of the detector is confined to the region bounded by the cathode plate and the Frisch grid (as in the drift chambers). The anode is segmented into 7 pads across the dispersion plane to provide horizontal position sensitivity (See Figure 3.15).

Once again a uniform field is applied to the active region which causes any charge carriers produced in the ionisation trail of a passing particle to drift towards their respective electrodes. The purpose of the ionisation chamber is to provide energy loss data which can then be used in conjunction with the energy measurements of other detectors to identify the particles.

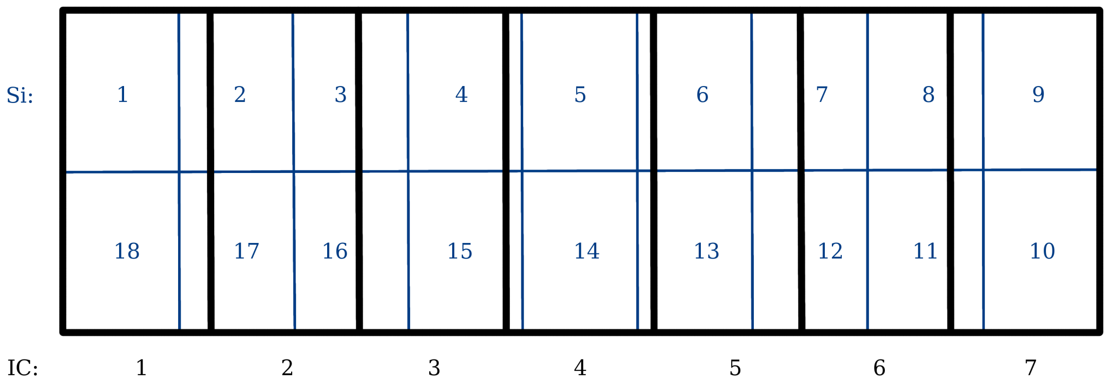

Figure 3.15: The arrangement of the VAMOS ionisation chamber pads with respect to the silicon detectors. The ionisation chamber resides in front of the silicon detectors.

Silicon

There are 18 silicon detectors in VAMOS and they are positioned in such a way that there are 2 rows of 9 wafers, one row above the other, to cover the active area of the image plane (See Figure 3.16). These detectors are mounted directly onto the exit port of the ionisation chamber housing which removes the need for the associated exit window but places them within the fill gas. The wafers have thicknesses in the range $520 - 540\,\mu$m (a list of the individual thicknesses can be found in Table A.3) and have a surface area of $7 \times 5\,\text{cm}^2$. They are directly connected to their own pre-amplifiers which are mounted on the same board as the detector.

Caesium Iodide

The final stage of the VAMOS detector system is made up of 80 Thallium activated Caesium Iodide scintillator detectors which provide the residual energy measurements to complement those of the upstream detectors. Each of the crystals is 1 cm in length and has an entrance window surface area of $2.5 \times 2.5\,\text{cm}^2$. They are located behind the silicon detectors in six rows: four rows of twelve crystals with two rows of sixteen making up the mid-section (See Figure 3.16). The overlap with the silicon detectors and the image plane of VAMOS is not complete but it is sufficient.

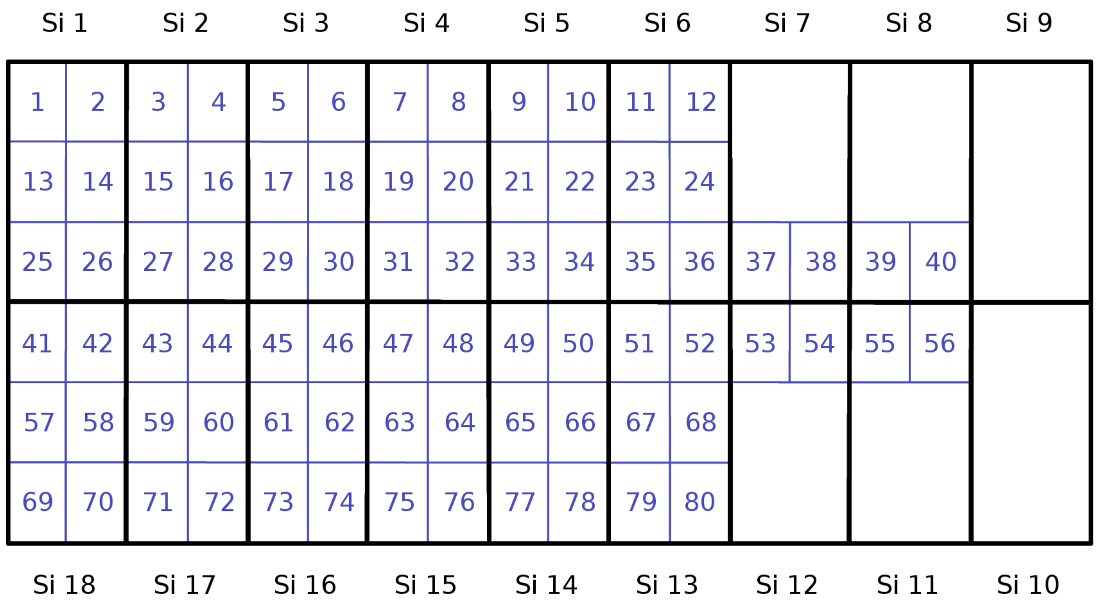

Figure 3.16: The arrangement of the silicon and caesium iodide detectors in VAMOS. The silicon detectors reside in front of the caesium iodide detectors.

As with the silicon detectors, the caesium iodide also resides within the ionisation chamber fill gas. Therefore the separation between the silicon and caesium iodide detectors needs to be taken into account as a dead layer when performing energy loss calculations. The crystals are wrapped using aluminium coated mylar foil which acts as a diffusing material to minimise the leakage of light from within the crystal (as in INDRA).

3.2.3 Electronics

The VAMOS electronics, in contrast to those of INDRA, are made using standard modules and components. Unfortunately, as the detector setup in VAMOS varies from one experiment to another, the electronics are not as well documented as they are for a more static setup like INDRA. Indeed no schematics are currently available for a detailed view of the pre-amplifiers and photo-multiplier bases.

Front-end Electronics

The front-end electronics for the VAMOS detectors were manufactured “in-house” at GANIL as the pre-amplifiers and photo-multiplier bases were of a commonplace design. It is worth noting however that in the case of the silicon pre-amplifiers, several field-effect transistors had to be connected in parallel in order to compensate for the high capacitance load presented by the silicon wafers (surface area of $7 \times 5\,\text{cm}^2$). The drift chamber readout is also fairly unique to the system as each drift chamber uses four application specific integrated circuit (ASIC) chips to multiplex its 128 analog output lines into a single output line.

Signal Transmission

As in INDRA, the transmission of the front-end signals from within the ionisation chamber housing out to the external electronics was achieved using specially designed flanges. The design of the flange and its connections varies with the detector type and in the case of the caesium iodide and ionisation chamber detectors, additionally provides access for the high voltage lines (with due regard for the isolation requirements). The propagation of these signals from the flange out to the external electronics is achieved using standard coaxial cable and, as always, care was taken to minimise the cross-talk between signal lines at the interface by having them surrounded by ground reference connections.

Acquisition Electronics

The VAMOS electronics responsible for the amplification, pulse shaping, digitising, discriminating etc. were all standard NIM/CAMAC/VME modules and follow the usual signal processing path. Although it should be noted that several modules from the TIARA [62] and MAYA [63] electronics were re-tasked for use in the caesium iodide electronics and several were replaced for various reasons. Post-digitisation, the signals are routed out to the VME based acquisition system, more specifically to the event builder which packages the data before writing it out to disk.

3.3 INDRA-VAMOS Coupling

3.3.1 Physical Modifications

As the goal of this experiment was to study peripheral and semi-peripheral interactions between the beam and the target, the VAMOS spectrometer had to be rotated to 4.5°, so that it covered the polar angles $2 - 7°$. This situation presents something of a problem as far as INDRA is concerned, as the acceptance window of VAMOS now encroaches upon the forward acceptance of INDRA. In order to resolve this situation the first three rings in INDRA had to be removed (those covering $2-7°$), namely the two phoswich rings and the first ring containing ionisation chambers, silicon and caesium iodide detectors. In addition, two modules from rings 4 and 5 had to be removed, specifically modules 17 and 18 from each ring. The coupled detectors can be seen in Figure 3.17.

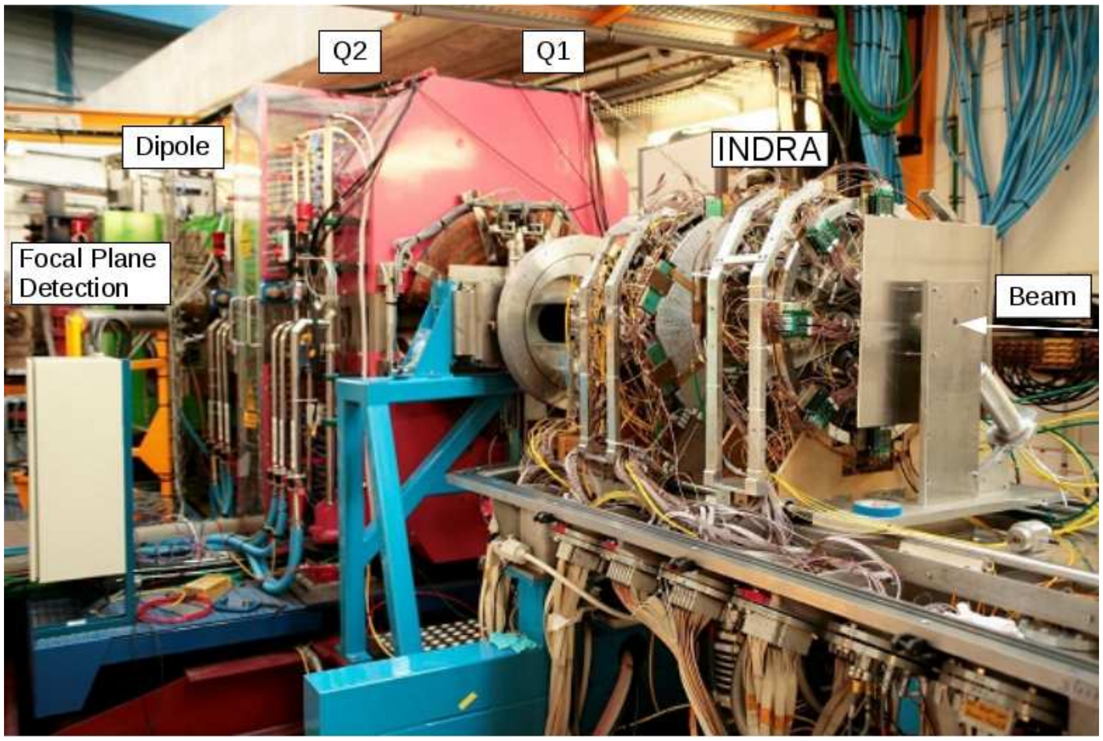

Figure 3.17: Photograph of the INDRA-VAMOS detector setup. The beam enters on the right, strikes the target which is located at the centre of the INDRA array and the reaction fragments are either detected in INDRA or transported through the magnets to the VAMOS focal plane detectors.

3.3.2 Electronic Modifications

In addition to these physical modifications there were electronic alterations required to couple INDRA to VAMOS. In particular the triggering circuitry needed careful consideration and getting the propagation delays and coincidence windows set up correctly proved to be challenging. Ultimately only two triggering conditions were defined, INDRA_SOLO (VAMOS is simply disconnected) and VAMOS_MASTER. Usually both triggers are operated with multiplicity $M \geq 1$. For the study of physics, VAMOS_MASTER is the desired event trigger, as it will trigger off the projectile-like residue which will then call for the read-out of the INDRA data. The combined data is then packaged by the event builder and written to disk.

For the data obtained using this coupling to be of any use, the two multi-detectors have to be correlated, that is, if the counting rates increase in VAMOS they should increase by a corresponding amount in INDRA. This correlation had to be continuously monitored throughout the experiment.

The counting rates turn out to be limited by INDRA, not VAMOS, and the coincidence between the two varies with the $B\rho$ setting of the dipole magnet. The dead-time is usually in the region of $20 - 40\%$ for the physics runs.

3.4 Accelerator

Before the experiment can be performed, the incident ion beam needs to be generated and transported into the cave. This is a task which is a discipline unto itself and so only a very brief overview will be given here.

Firstly, the ions are created at the ion source using a neutral material which contains the desired isotope. This is achieved either by the application of a strong electromagnetic field or by heating it within a high-temperature oven. The general principle of both methods is the same; to excite electrons in order to ionise the material and make it easier to accelerate. The ions leaving the source do however contain a large range of charge states which need to be filtered out at some stage in the optics of the beam line.

Once the ions have been created they are initially accelerated within a small cyclotron named either C01 or C02, depending on the ion source (See Figure 3.18). This cyclotron brings the ion beam within the acceptance window of the first cyclotron, CSS1. This cyclotron accelerates the beam and can optionally feed into another cyclotron, CSS2, which in turn can optionally feed into a secondary ion source (SISSI, see below). In most ion beam experiments however, the purpose of CSS1 and CSS2 is simply to accelerate the ion beam to the desired energy for the experiment. Once this has been done the beam is carefully guided along the beam line to the experimental area.

For the generation of secondary ion beams as mentioned above, there is SISSI (Superconducting Intense Source for Secondary Ions). These secondary beams are generated by focussing a heavy primary beam (created via the aforementioned procedure) onto a thick, rotating target (to dissipate heat) and refocussing the resulting fragments. As the name suggests the focussing of these beams is achieved using superconducting

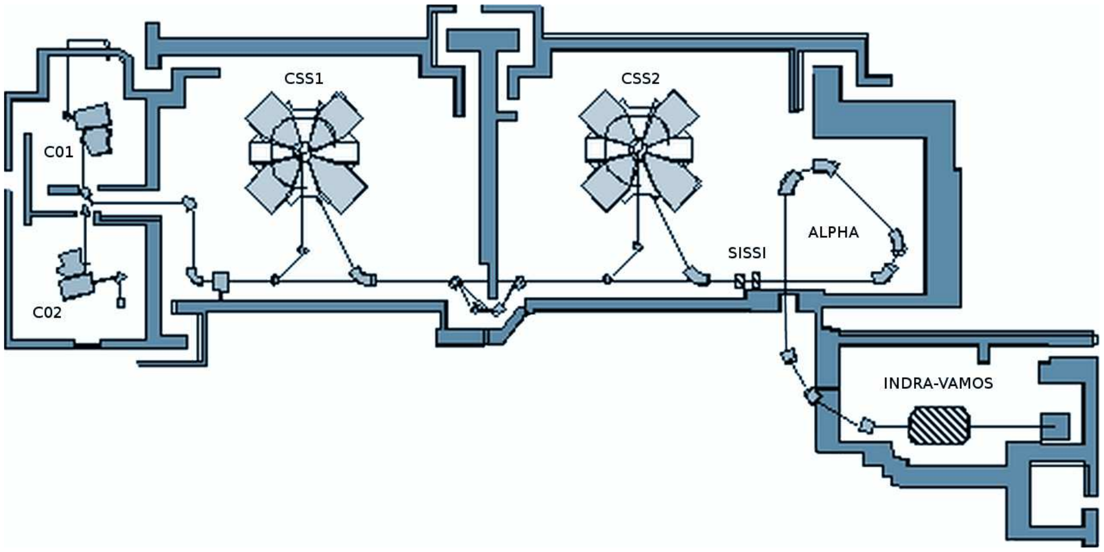

Figure 3.18: Schematic diagram of the beam-line and experimental area. Only the INDRA-VAMOS sections are shown, the other experimental areas have been removed for clarity. Adapted from the image appearing in [64].

solenoids, these are cooled with liquid helium and thermally isolated from the rotating target. In this experiment its principle use is for the generation of cocktail beams to be used in the calibration of INDRA and VAMOS.

3.5 Beam Characteristics and Timing

The high-frequency of the beam is determined by the voltage difference induced across the pair of electrodes in the cyclotron. This frequency has a value of $\nu_{HF} = 8.55\,\text{MHz}$, for this experiment which corresponds to a time period of $t_{HF} = 116.96\,\text{ns}$.

This high-frequency signal is used to denote the beginning of an event in the INDRA-VAMOS acquisition system (the start signal). The termination (or stop) signal is given by VAMOS (as the master trigger). Usually the VAMOS time reference is taken from the fast signal provided by the secondary electron detector, in this experiment however it is inactive and the reference signal is instead taken from the silicon detectors.

3.6 Experiment Procedure

The experimental procedure is fairly straight-forward. In the initial setup phase, which occurs just after the physical and electronic coupling of INDRA and VAMOS, the various detectors are checked to ensure they are operational. Once this preliminary check is complete, it is necessary to take care of the propagation delays and any other intricacies of the electronics to make sure the triggering works as intended. If the trigger works properly the correlation between the two multi-detectors must be verified, indeed this must be continuously monitored throughout the experiment, any mismatch indicates a problem with the triggering. As an example, suppose the INDRA validation signal occurs outside the coincidence window defined by the VAMOS trigger (perhaps due to some unforeseen propagation delay) this would result in the INDRA data failing to read-out.

Once the trigger is working and the multi-detectors are well-correlated, the initial calibration of the detectors needs to be performed. This is achieved using a beam of ^{40}Ar at 35 AMeV with either a ^{197}Au or a ^{48}Ti target, as well as additional data taken from cosmic rays and alpha sources. The linearity of the detectors is also periodically checked throughout the experiment using both pulsers and lasers (for the scintillation detectors).

The procedure for taking the physics data does require a little explanation. In all of the physics runs, VAMOS is acting as the master trigger and is turned to 4.5° so that it covers the polar angle range $2-7°$. In order to cover the widest possible range in detected mass, the VAMOS spectrometer's dipole magnet is swept through 10 $B\rho$ settings (in 8% increments) for each of the calcium-calcium reaction systems. These $B\rho$ settings are governed by the current passing through the magnet. The normalisation of these data sets is achieved using the count rates detected in the INDRA Faraday cup and works for as long as the trigger conditions do not alter. Additional calibrations are also performed towards the end of the experiment with cocktail beams generated using SISSI.

3.6.1 Reaction Systems

System	Energy [AMeV]	Purpose
^{40}Ar + ^{197}Au	35	Calibration/Setup
^{40}Ar + ^{48}Ti	35	Calibration/Setup
Cosmic	-	Calibration
Generated (Pulsers/Laser)	-	Detector linearity
α-source	-	Calibration
^{48}Ca + ^{40}Ca	60	Calibration
^{48}Ca + ^{40}Ca	35	Physics
^{48}Ca + ^{48}Ca	35	Physics
^{40}Ca + ^{197}Au	35	Calibration
^{40}Ca + ^{48}Ca	35	Physics
^{40}Ca + ^{40}Ca	35	Physics
^{16}O + ^{197}Au	28	Cocktail Beam (SISSI)
^{16}O + ^{12}C	28	Cocktail Beam (SISSI)

3.6.2 Target specifications

Target	Thickness [mg/cm^2]
^{40}Ca	1.2
^{48}Ca	1.52
^{12}C	0.02
^{197}Au	1.7
^{48}Ti	0.97

Where ^{48}Ca is in the form of a $CaCO_3$ powder, isotopically enriched to $(96.04\pm0.4)\%$ (3.65% being ^{40}Ca) and stored in Argon gas.

Chapter 4

Calibration and Identification Methods

4.1 INDRA

4.1.1 Principles of Identification

The principal method of identification in INDRA implements what is known as the $\Delta E/E$ identification method. This method combines the detected energies of two different absorbers as a means of distinguishing the different isotopes. The basic principle of this method is to make use of the fact that the two absorbers have different linear stopping powers. The linear stopping power of an absorber is defined as the differential energy loss of an incident particle with respect to the distance it travels through the material:

$$S = -\frac{dE}{\mathrm{d}x} \tag{4.1}$$

The Bethe equation defines the linear stopping power for a particular absorber:

$$S = \frac{1}{(4\pi\epsilon_0)^2}\frac{4\pi e^4 Z^2}{m_0 v^2} \cdot n \cdot B \tag{4.2}$$

Where:

$$B = z\left[\ln\left(\frac{2m_0v^2}{I}\right) - \ln\left(1 - \frac{v^2}{c^2}\right) - \frac{v^2}{c^2}\right] \tag{4.3}$$

In the above equations, Ze is the charge of the incident particle and v is its velocity. z and n denote the atomic number and number density of the absorber atoms (respectively), m_0 is the electron rest mass and I is an empirically determined factor which represents the average excitation/ionisation potential of the material [58, p. 31]. This equation can be reduced for non-relativistic particles ($v/c \ll 1$) such that:

$$S \approx \frac{1}{(4\pi\epsilon_0)^2}\frac{4\pi e^4 Z^2}{m_0 v^2} \cdot n \cdot z \ln\left(\frac{2m_0v^2}{I}\right) \tag{4.4}$$

Or as a simple statement of proportionality:

$$S \propto \frac{Z^2}{v^2} \equiv \frac{AZ^2}{E_{inc}} \tag{4.5}$$

Where A and Z denote the mass and atomic numbers of the incident particle and E_{inc} the incident energy. The linear stopping power of an absorber, in relation to a particular incident particle, therefore depends upon both the atomic number Z and the mass number A of the incident particle in addition to its incident energy. It is precisely this dependence and the fact that it is different for each absorbing material that facilitates the separation of the isotopes in the $\Delta E/E$ plane.

As an example, the correlation between the silicon and caesium iodide detectors of the seventh module of the sixth ring in INDRA, chosen simply as it possesses the largest quantity of data and is one of the 150 μm silicon wafers demonstrating a finer resolution, can be observed in Figure 4.1. This figure demonstrates the isotopic separation of light charged particles in the reaction system ^{40}Ca+^{40}Ca at 35 AMeV. In general the isotopic resolution tends to break down around $Z \approx 6-7$ however Z identification is achieved all the way up to the value of the beam and in principle can go higher.

The means by which the individual particles are assigned their values of Z and A does require some explanation. Initially a grid must be drawn by hand in order to define the curves that several well resolved and visually identifiable particles follow (See Figure 4.1). Generally speaking, curves will be drawn for the low Z isotopes and for a couple of the higher Z values (where the isotopic composition is assumed to be that resulting from β stability). These hand-drawn grids are then used as the input parameters to a fit functional which extrapolates from them to form a complete set of grids covering the entire range of possible Z and A values (See the right panel of Figure 4.2). Details of this fit functional method will not be given explicitly here, the reader is referred to [65]. The resulting particle identification is not as good as if the grids were all drawn by hand, however in INDRA there are 144 modules of silicon and caesium iodide with approximately 30 curves required for each (over 4300 curves) and it is not desirable to spend so much time perfecting these identifications (at least at this stage in the analysis). There are a couple of minor problems with this method, the first is that the fit procedure minimises the fit for the entire range of Z at the expense of the well resolved low-Z values, as it was not originally intended for use over such a large range of isotopes. Another problem is that it tends not to reproduce the observed curvature at very low caesium iodide light output which can lead to mis-identification of particles in this region. The hand drawn identification grids for the silicon/caesium iodide telescopes were made by the author several years ago and are now a part of the E503 repository.

The identification maps containing the ionisation chamber energy against either silicon or caesium iodide residual energy are achieved in much the same way although

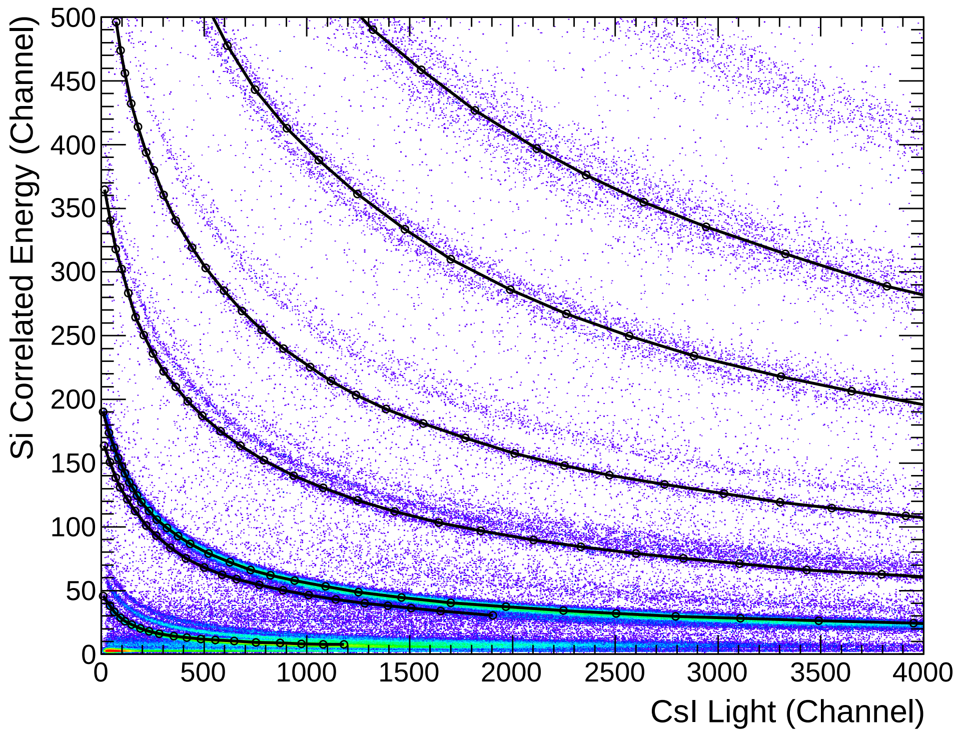

Figure 4.1: An example of the hand drawn $\Delta E/E$ grid for ring six module seven for the system ^{40}Ca + ^{40}Ca at 35 AMeV. The silicon correlated energy, that is, a linear function of low gain and high gain values, is plotted against the caesium iodide total light output. The difference in the linear stopping power of the two materials is what gives rise to the observed isotopic separation. This hand drawn grid (there are also a couple of curves at high-Z) is used as the input to the Tassan-Got fit functional (See text).

it should be noted that the resolution is insufficient as to allow much in the way of isotopic resolution.

In addition to these two forms of identification it is also possible to perform identifications within the caesium iodide crystals *alone*, that is without a second absorber. The reason this is possible is because the light output signals from the crystals are not fully integrated. They are in fact integrated in two specific integration windows: a fast integration of $0 - 400$ ns and a slow integration of $1600 - 3100$ ns. It is this fact that allows the particle identifications to take place. By plotting the fast integration signal against the slow integration signal, in a manner analogous to the $\Delta E/E$ method described above, the isotopes for the low Z nuclei (typically no higher than Beryllium or Lithium) are well separated and can be identified using hand drawn grids (See Figure 4.3).

The identification procedure for an unidentified particle is to start from the stopping

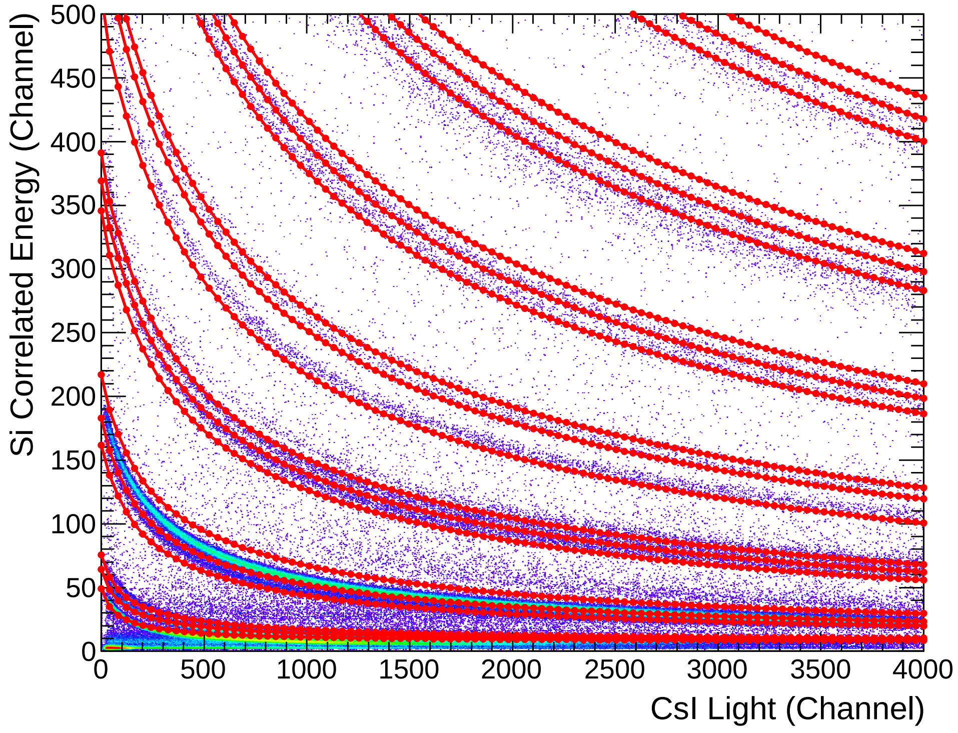

Figure 4.2: An example $\Delta E/E$ grid extrapolated from the initial hand drawn grid using the Tassan-Got fit functional (See text) for ring six module seven for the system ${}^{40}\mathrm{Ca} + {}^{40}\mathrm{Ca}$ at 35 AMeV. The silicon correlated energy, that is, a linear function of low gain and high gain values, is plotted against the caesium iodide total light output. The difference in the linear stopping power of the two materials is what gives rise to the observed isotopic separation. Note that the fit is not perfect due to the large range of Z being fitted.

detector and attempt the aforementioned identifications in sequence moving towards the target until a valid identification is returned. Once this valid identification has been found, various checks must also be made to ensure that undesirable event types (such as pile-up) are properly handled.

4.1.2 Calibration

The energy calibration of the INDRA detectors was achieved using a cocktail beam[1] of low mass ions generated by the reaction ${}^{16}\mathrm{O} + {}^{12}\mathrm{C}$ at 90 AMeV. The target for this reaction was located in the secondary ion source (SISSI) just after the second cyclotron (CSS2) and produced ions in the range $Z = 1 - 8$. These ions were then accelerated and selected by varying the $B\rho$ setting of the alpha spectrometer (See Figure 3.18) and therefore have a well defined energy as they enter the experimental area (selection on

[1]Run 628 in the E503 data set

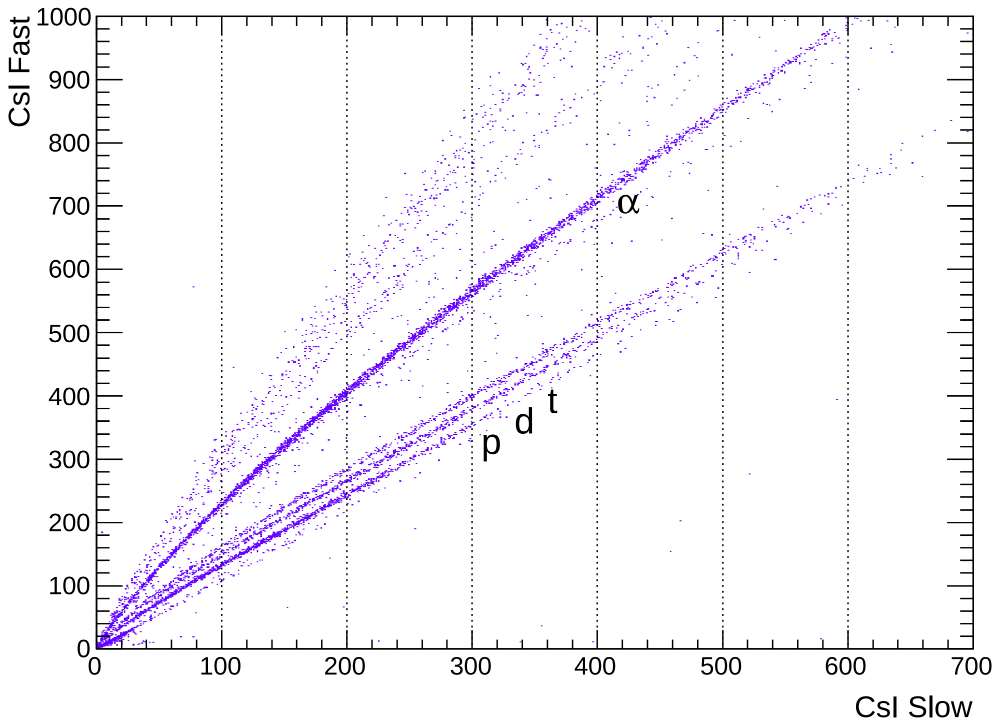

Figure 4.3: Caesium iodide isotopic separation for ring 6 module 1. Only data from a single run is shown for demonstration purposes but even so, proton, deuteron, triton and alpha particles are all easily visible.

particle momentum is functionally equivalent to selection on kinetic energy for a given mass). These ions are then directed onto either a ^{12}C or an ^{197}Au target mounted in the INDRA target holder and are elastically scattered into the INDRA detectors. There are also some additional contributions from inelastic reactions generally involving the ground state and first excited state in ^{12}C. Before the energy calibration of the detectors can take place however they must first be calibrated in terms of voltage.

Ionisation Chamber

The voltage calibration of the ionisation chamber modules requires a set of well known reference voltages. These voltages are provided by a pulse generator[2] which sends voltage signals of known amplitude directly to the ionisation chamber pre-amplifiers. The amplitude of this signal is controlled by a variable attenuation factor present on the output line of the pulse generator, by sweeping through the range of available attenuation one obtains the aforementioned set of reference voltages. These reference signals then induce corresponding responses in the signal processing pathways of the electronics and eventually register as a set of uncalibrated channel numbers in the

[2]Runs 546 and 547 in the E494s data set

energy spectra, or put another way, as a set of raw energy coder values. By fitting the correlation between the known input voltage signals and the resulting energy coder values with a second order polynomial equation one obtains the voltage calibration. This voltage calibration is then easily translated into an energy calibration by using the reference energies of the elastically scattered protons of the aforementioned reaction system. The relationship between the energy calibration and the voltage calibration is taken to be linear.

Silicon

The silicon detector energy calibrations are achieved in precisely the same way. The pulse generator again sends a set of known voltage signals whose amplitude is controlled by variation of the attenuation on its output line, only this time to the silicon pre-amplifiers. These signals induce corresponding responses in the signal processing pathways of the electronics and register as a set of raw coder values in the uncalibrated silicon energy spectra. By fitting this data with a second order polynomial equation the voltage calibration is extracted. This voltage calibration is then converted into an energy calibration using the elastically scattered protons of known energy. Once again the energy calibration function has a linear relationship to the voltage.

Caesium Iodide

The calibration of the caesium iodide detectors is more involved. The reason for this is that the light response function depends not only upon the energy deposited in the crystal but also on the Z and A values of the incident particle. This response function was formulated by the INDRA collaboration specifically for use with their caesium iodide crystals. It is analytically derived from the various physical processes occurring within the crystal when the particle deposits its energy (For a detailed overview see references [66] and [57]):

$$\begin{aligned} L(Z,A,E) =& c_1 \cdot \left\{ E - c_2 \cdot \ln\left(1 + \frac{E}{c_2}\right) + c_2 \cdot c_4 \cdot \frac{\ln\left(\frac{E+c_2}{c_2+c_3}\right)}{1 + \exp\left(\frac{c_3-E}{T}\right)} \right\} \\ & - c_0 \cdot c_1 \cdot c_2 \cdot \ln\left(\frac{c_2}{c_2+c_3}\right) \end{aligned} \tag{4.6}$$

With the terms:

$$c_0 = \frac{c_4}{1 + \exp\left(\frac{c_3}{T}\right)} \tag{4.7}$$

$$c_1 = a_0 \tag{4.8}$$

$$c_2 = A \cdot Z^2 \cdot a_1 \tag{4.9}$$

$$c_3 = A \cdot a_2 \tag{4.10}$$

$$c_4 = a_3 = 0.385 \tag{4.11}$$

$$T = 8 \cdot A \tag{4.12}$$

Where E denotes the energy deposited in the crystal and the fit parameters are given by $a_{0..3}$. For completeness: a_0 is the gain parameter, a_1: the quenching (recombination) parameter, a_2: the energy threshold for δ-ray production parameter and a_3: the parameter representing the fractional energy loss transferred to a δ-ray. The parameters are formulated as they are in the above compound relations ($c_{0..4}$) for simplicity and for consistency with how the formula appears in the analysis code.

It should be noted that this response function is for the total light output of the crystal, which is not what is actually measured, remember the light output signal is integrated in two components (fast and slow, See Figure 4.4) and so the total light must first be calculated from these two components using the approximation that the total light output adheres to the following equation:

$$i(t) = \frac{Q_0}{\tau_0} \exp\left(-\frac{t}{\tau_0}\right) \tag{4.13}$$

Where Q_0 is approximately the total integrated charge and τ_0 is the decay time constant. Coupling this expression with the following, for the same signal in terms of the two partially integrated components (fast and slow):

$$i(t) = \frac{Q_f}{\tau_f} \exp\left(-\frac{t}{\tau_f}\right) + \frac{Q_s}{\tau_s} \exp\left(-\frac{t}{\tau_s}\right) \tag{4.14}$$

Together with that of the measured current at the output of the photo-multiplier base:

$$i_{meas}(t) = \frac{Q_0}{\tau_0 - \tau} \left[\exp\left(-\frac{t}{\tau_0}\right) - \exp\left(-\frac{t}{\tau}\right)\right] \tag{4.15}$$

The values of Q_0 and τ_0 can be determined and thus the total light can be calculated. In the above equations the subscript f denotes the fast integrated component of the signal and s, the slow integrated component, τ is the signal rise time constant ($\tau_0 \gg \tau$).

The calibration of the light response function is achieved using the same elastically scattered protons as in the previous sections. The protons are depositing a known energy (and known Z and A) into the crystal and produce the corresponding coder values for the light output. By correlating this light output with the energy deposited

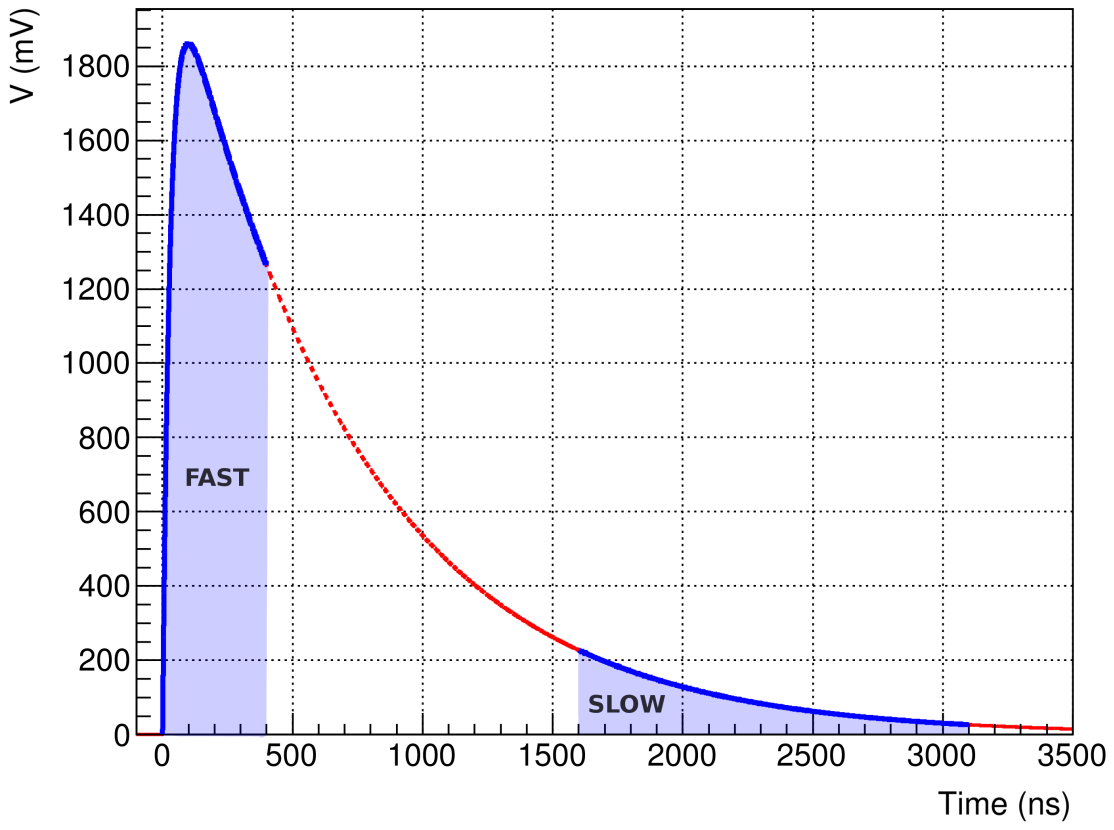

Figure 4.4: Example caesium iodide voltage signal modelled on equation 4.14 demonstrating the fast and slow integration gates (highlighted in blue).

by the protons, the light to energy calibration is achieved. This calibrates the light response function so that when a particle of unknown energy passes through the crystal, the total light computed from its fast and slow components in conjunction with its predetermined Z and A values can be used to calculate its energy loss, by inversion of the light response function.

Timing

The INDRA timing data has not been implemented at this stage in the analysis, as its only purpose at present is to serve as a means to verify the correlation between INDRA and VAMOS, to prove that the two multi-detectors are in fact coupled. The timing information does not need to be calibrated for this as only the *relationship* between INDRA and VAMOS is required. The primary method of verifying the coupling is to plot the low gain silicon signal from ring 6 module 1 in INDRA against the same signal but routed through the VAMOS acquisition electronics (See Figure 4.5).

However another way to see this correlation is by plotting the silicon time in INDRA against the silicon time in VAMOS (See Figure 4.6). Assuming that the electronics are set up correctly with a proper handling of all the propagation delays, event validation signals, coincidence window settings etc. then there should be a correlation between

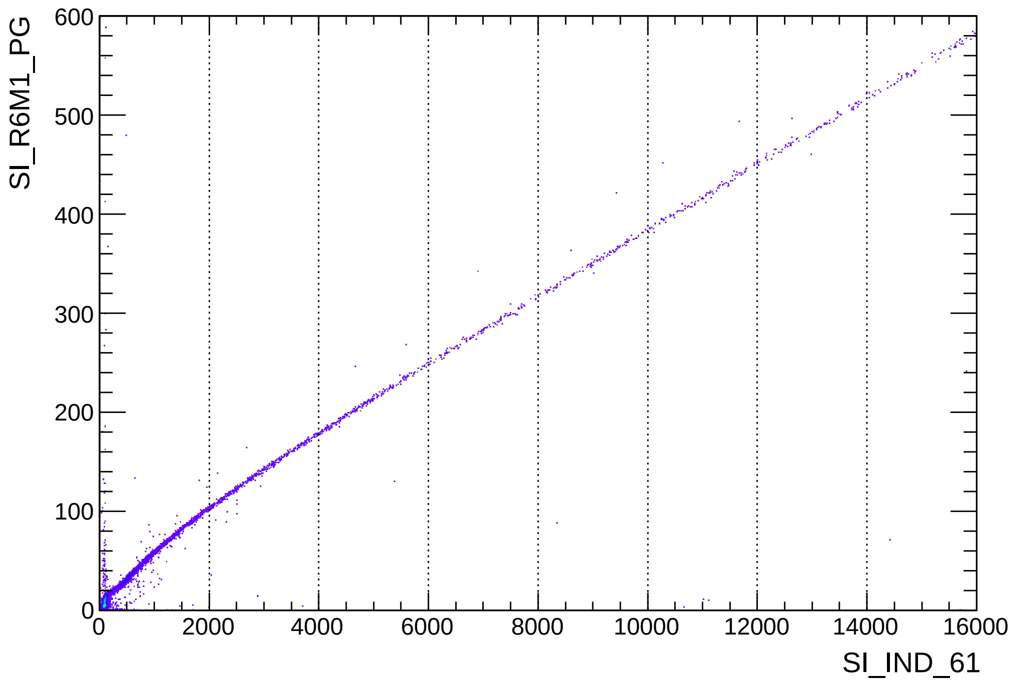

Figure 4.5: The primary method of verifying the correlation between INDRA and VAMOS is to plot the low gain silicon signal for ring 6 module 1 in INDRA against the same signal but routed through the VAMOS acquisition electronics. This behaviour was continuously monitored throughout the experiment.

the two measurements (longer flight times detected in VAMOS results in shorter flight times measured in INDRA due to increased interaction with the target). The timing calibrations are actually possible, as pulse generator runs were performed with this task in mind but there is simply not much use for doing so at present.

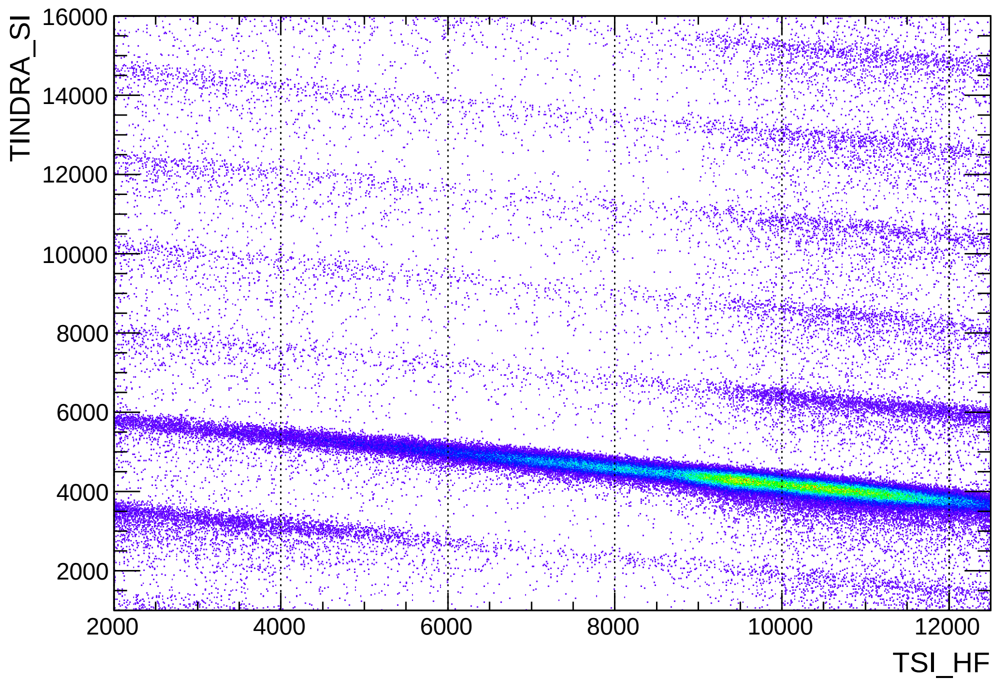

Figure 4.6: An alternative method of verifying the correlation between INDRA and VAMOS is to plot timing correlation between the two. For this, no calibration is needed.

4.2 VAMOS

4.2.1 Principles of Identification

The identification of fragments in VAMOS is necessarily more complicated than in the simple $\Delta E/E$ method used in INDRA. This is mostly due to the fact that the fragments must first traverse a non-trivial set of electro-magnetic fields before they reach the focal plane detection system. The trajectory reconstruction of these fragments through the electro-magnetic system is performed using a set of transfer functions derived using an off-line ray-tracing program and is a necessary step, prior to identification.

The reconstruction procedure treats each fragment as a set of six co-ordinates: $(x_f, y_f, \theta_f, \phi_f, \delta, \hat{l})$, where $x_f, y_f, \theta_f, \phi_f$ define the trajectory of the fragment at the focal plane and δ and $\hat{l}$ are, respectively, the fractional momentum difference and path length difference between the trajectory of the fragment and the reference trajectory. The reference trajectory is defined by setting the field strengths of the magnets so that the beam passes through the focal plane at a right angle. In this experiment the reaction system $^{40}\mathrm{Ar} + {}^{197}\mathrm{Au}$ at 35 AMeV was used to define the reference trajectory[3] ($B\rho_0 = 1.90673\,\mathrm{Tm}$). The drift chamber position measurements are then offset to ensure that this trajectory passes through $(0, 0)$.

[3]Run 129 in the E503 data set

A fragment defined in terms of the above six parameters must then be transported back through the electro-magnetic system of VAMOS to determine its trajectory at the target position. This is achieved through the use of four, seventh order, non-linear polynomial parametrisations determined using numerical ion-optical calculations and accurate field maps which serve as reverse transfer functions:

$$\delta = F_1\left(x_f, \theta_f, y_f, \phi_f\right) \tag{4.16}$$

$$\theta_i = F_2\left(x_f, \theta_f, y_f, \phi_f\right) \tag{4.17}$$

$$\phi_i = F_3\left(x_f, \theta_f, y_f, \phi_f\right) \tag{4.18}$$

$$\hat{l} = F_4\left(x_f, \theta_f, y_f, \phi_f\right) \tag{4.19}$$

As an example, the expression for δ reads:

$$\delta = \sum_{i,j,k,l=0}^{i+j+k+l=7} C_{ijkl} x_f^i \theta_f^j y_f^k \phi_f^l \tag{4.20}$$

Where the constants C_{ijkl} are determined numerically (for more information refer to [59]). The input parameters of this reconstruction are therefore the inferred or detected trajectory of the fragment *at the focal plane* $(x_f, y_f, \theta_f, \phi_f)$. In the particular case of this experiment, these co-ordinates are inferred from the trajectory determined using the vertical (non-dispersive) and horizontal (dispersive) position measurements of the two drift chambers which reside on either side of the focal plane $(x_1, y_1) \rightarrow (x_2, y_2)$. A more direct measurement is not possible in this case as the secondary electron detector, which typically resides at the focal plane, is inactive.

Once the trajectory of the fragment has been reconstructed at the target position, the identification procedure can be used to determine the mass and charge of the fragment (For a schematic representation see Figure 4.7). This identification relies upon the following pair of equations for the determination of mass and mass-to-charge ratio:

$$M = \frac{2E_T}{931.5\beta^2} \tag{4.21}$$

$$\frac{M}{q} = \frac{B\rho}{3.105\beta} \tag{4.22}$$

Where E_T represents the total energy of the fragment (MeV), $B\rho$ the magnetic rigidity (Tm) and β the velocity normalised to the speed of light in vacuum (v/c). The first stage of the identification is to take the reconstructed flight path, $(\hat{l})$ and combine it with the time of flight measurement in order to determine the velocity of the fragment (β). The Z value is already known via the usual $\Delta E/E$ method in the focal plane detectors and the velocity and total energy calculations can be combined to determine the mass (Equation 4.21). In a similar fashion the mass-to-charge ratio is calculated by combining the reconstructed magnetic rigidity ($B\rho$) and the velocity (Equation 4.22). Finally the charge state q can be determined by simply dividing the two equations. This identification procedure is naturally very sensitive to both the energy and time calibrations of the individual detectors.

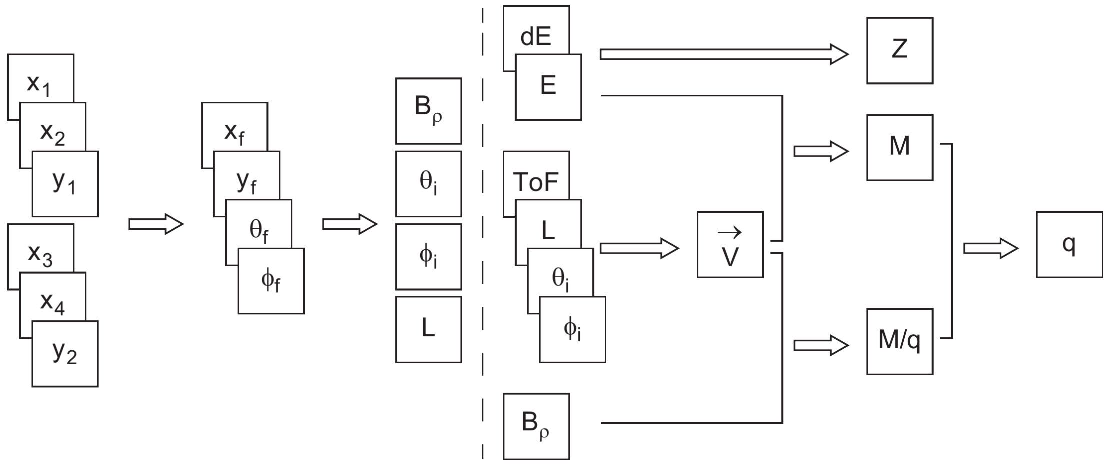

Figure 4.7: A block diagram demonstrating how the identification of fragments is achieved in VAMOS (Taken from [59]). The trajectory of the fragment is reconstructed (left) and combined with energy and time of flight measurements (right) to determine the mass and mass-to-charge ratio according to Equations 4.21 and 4.22. These two values are then simply divided to find the charge state q.

4.2.2 Calibration

In this section the calibration of the detectors of VAMOS will be explored in some detail. Each detector calibration will be examined in turn and finally a discussion on the time of flight calibration will be given. Much of this work was done by Paola Marini and Mark Boisjoli but the author was directly involved in the caesium iodide energy, silicon energy and silicon timing calibrations.

For the purposes of checking energy and time calibrations, a simple "straight-through" simulation was written by the author. It was written around three years ago and despite its simplicity it has proved to be useful. This basic simulation program can be found at the following locations:

- Bazaar:
  `https://code.launchpad.net/~peter-wigg-314159/+junk/vamos_simulation`
- Computing Centre:
  `/afs/in2p3.fr/home/p/pwigg/public/git/vamos_simulation`

Drift Chamber

The drift chambers are currently only calibrated for their voltage and time signals as these are the calibrations which enable the position sensitive part of their operation. The low fractional energy loss of fragments passing through the drift chambers (less than 0.3% of the total energy loss for the elastic ^{40}Ca nucleus) and their comparatively poor energy resolution, combine to make the full energy calibration unprofitable at this stage of the analysis (if indeed it is required at all). This negligence of the energy calibration does not inhibit the identification properties of VAMOS in any perceptible way as the energy loss of the fragments constitutes so small a contribution to the total energy (which determines mass). When it comes to the total energy, range table calculations are more than sufficient at the present time.

For the voltage calibration, a pulse generator is used to send signals of a known voltage directly to the pre-amplifiers[4]. Varying degrees of attenuation are then applied to the output signal of the pulse generator in order to produce a range of known voltages as calibration points. These calibration points (channel, mV) are then fit with a second order polynomial function:

$$V(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 \tag{4.23}$$

This voltage calibration ensures that the pad signals can be used in conjunction with one another to allow the position of the detected fragment in the dispersive (horizontal) direction to be determined. Recall that this procedure uses Gaussian fitting of multiple pad signals in order to determine this co-ordinate.

In order to calibrate the position measurement in the non-dispersive (vertical) direction, the drift time signal must be calibrated. Once again a pulse generator is used to provide the reference signals, only this time it is directed into the stop input of the time amplitude converter. This signal is delayed by varying degrees to provide reference points with a known time interval[5]. These calibration points are then fit with a semi-linear function, i.e. a straight line above some threshold with some curvature allowed for the lower channel numbers such that the total fit is given by:

$$t(x) = b_0 + b_1 \cdot x \qquad (x > b_3) \tag{4.24}$$

$$t(x) = b_0 + b_1 \cdot x + b_2 \cdot (x - b_3)^2 \qquad (x \le b_3) \tag{4.25}$$

The time difference between the fast silicon detector signal (start signal) and this calibrated time, together with the drift velocity of the electrons within the chamber, determines the vertical displacement of the fragment track. In summary, the voltage

[4]Run 48 (Chamber 1) and Run 49 (Chamber 2) in the E503 data set

[5]Run 101 (Chamber 1) and Run 102 (Chamber 2) in the E503 data set

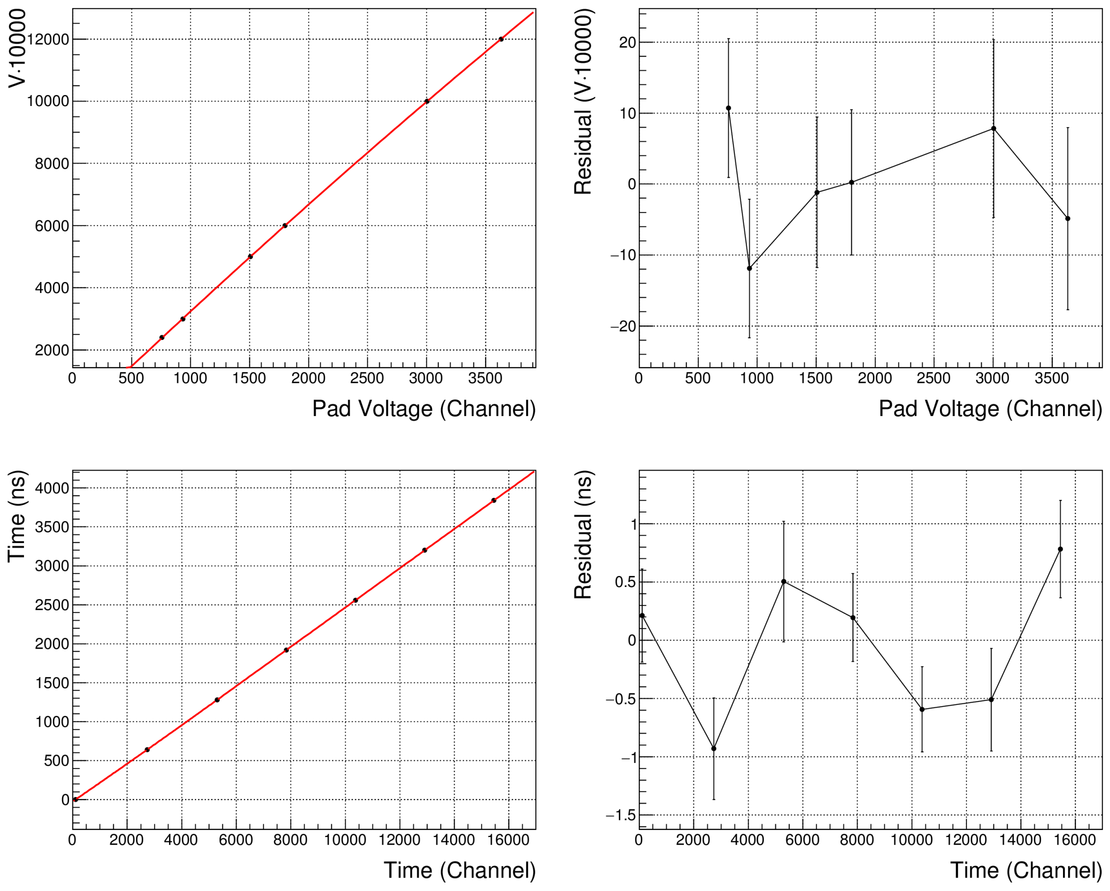

Figure 4.8: An example voltage and time calibration for pad number one in the first row of electrodes of the first drift chamber. The pulse generator calibration points are fit with a simple second order polynomial function (voltage) or a semi-linear function (time). Note that the voltage scale is arbitrary as it does not matter as long as all of the pads are calibrated using the same scale. In principle, an energy reference would be used to map this calibration into a true energy calibration. Error bars are 1σ, estimated using the channel peak width and the calibration function.

calibration enables the position measurement in the dispersive (horizontal) direction and the drift time calibration enables that in the non-dispersive (vertical) direction. The energy calibration is not currently required.

Ionisation Chamber

The energy calibration for the ionisation chamber also uses a pulse generator to provide known voltage signals to the pre-amplifiers[6]. By varying the attenuation of this signal six calibration points are determined (See Figure 4.9). These calibration points are then fit with a simple linear function:

$$V(x) = a_0 + a_1 \cdot x \tag{4.26}$$

[6]Run 615 in the E494s data set

As the number of ion pairs produced per unit energy loss of the incident fragment is an empirical constant and that the electric field strength is such as to ensure near complete charge collection, the energy loss calibration can be achieved with a simple scaling constant. This constant is determined using the calculated energy loss of the elastically scattered ^{40}Ca nucleus and correlating it with the corresponding signal in the ionisation chamber. Therefore the energy calibration is simply given by:

$$E = \alpha \cdot V(x) \tag{4.27}$$

It should be noted that this energy calibration assumes the linear response of the ionisation chamber at low energies (note the difference in channel scales in Figure 4.9) for the simple reason that the energy reference (the elastic peak) loses only 7.46 MeV within the active volume of the chamber. However, as the ionisation chamber contributes less than 0.6% of the total energy for the elastic ^{40}Ca nucleus in VAMOS, this should not prove to be a significant issue for the moment. If the ionisation chamber does not fire for some reason, its calculated energy loss is used in the total energy calculation instead of the calibrated energy.

Silicon

The silicon detector energy calibration is performed in precisely the same manner as for the ionisation chamber. Firstly the calibration of channel number to voltage is determined for each detector using a pulse generator[7]. The voltage of this generator signal is controlled by altering its attenuation and provides five calibration points which are then fit with a second order polynomial function:

$$V(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 \tag{4.28}$$

Once again the empirical observation that the number of electron-hole pairs formed in the ionisation trail of the passing fragment is constant per unit incident energy, allows for this voltage calibration to be transformed into an energy calibration with a simple scaling constant. Again the scaling constant is determined using the calculated energy loss of the elastically scattered ^{40}Ca nucleus[8] in each silicon detector. The complete energy calibration function is therefore as follows:

$$E(x) = \alpha \cdot V(x) \tag{4.29}$$

Note that it is clear from the residuals of the voltage and energy calibrations (Figure 4.10) that the fit is less than perfect, there appear to be other factors at work that prevent the use of a purely polynomial fit function, however having only five reference

[7]Run 448 (Detectors 1-9) and Run 449 (Detectors 10-18) in the E503 data set
[8]Runs 583-591 in the E503 dataset

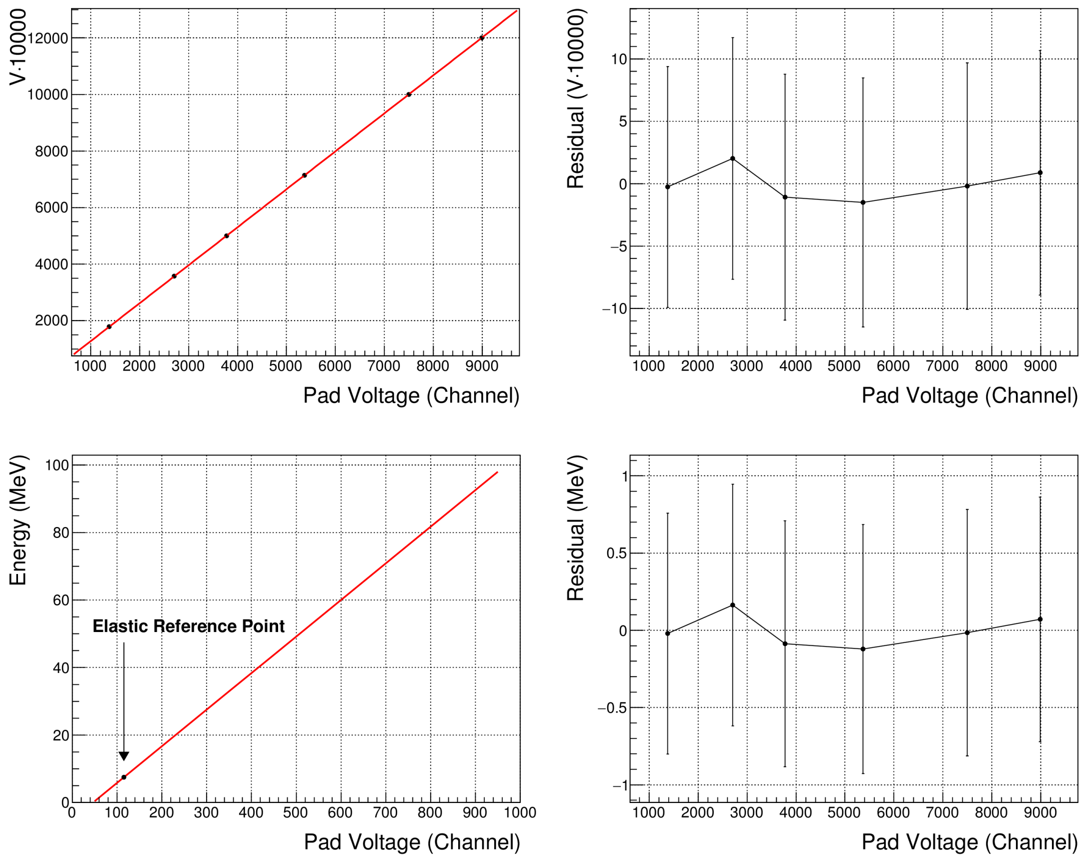

Figure 4.9: Example VAMOS ionisation chamber calibration for chamber number four. The voltage calibration (top panel) is a simple linear fit to the pulse generator data and the elastic ^{40}Ca peak provides the energy reference required to make the transition from voltage to energy (bottom panel). This calibration assumes the linearity of the ionisation chamber response at low energies (note the difference in scale - the elastic nucleus loses only 7.46 MeV in the active volume of the chamber). Error bars are 1σ, estimated using the channel peak width and the calibration function.

points to work with makes extracting the correct form of the ideal calibration function difficult. At present this calibration meets the needs of the experiment - approximately ± 2 MeV across the range, translating into $\sim 0.25\%$ error on the elastic peak, reaching $\sim 10\%$ for a 20 MeV signal.

The time amplitude converter signal is also calibrated with a pulse generator[9] which provides a set of well defined reference points in time by varying the propagation delay on the stop signal. Using these calibration points the channel to nanosecond conversion is achieved with two second order polynomial functions (one of which is almost effectively linear), two are required in order to recover deviations from linearity at high and low channel numbers. The overall fit function is therefore given by the following formulae:

[9]Run 584 in the E494s data set

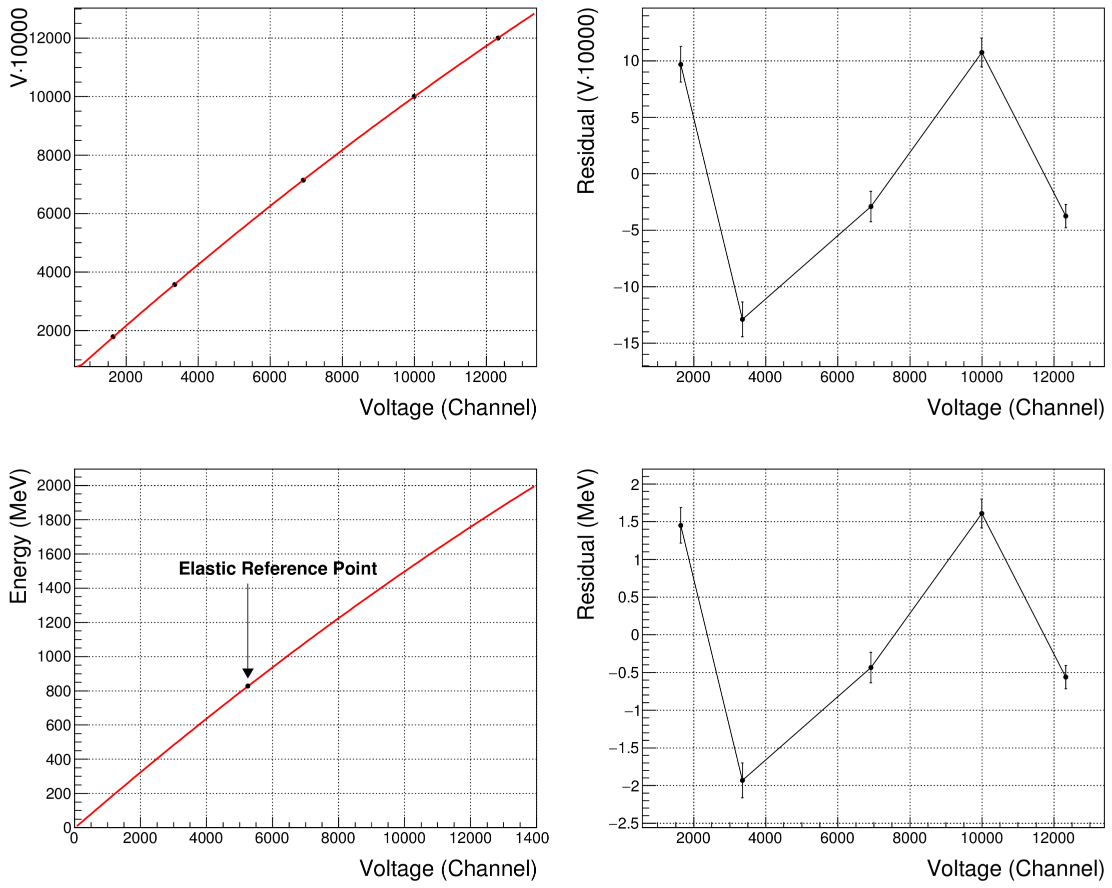

Figure 4.10: Example VAMOS silicon calibration for silicon number fifteen. The channel to mV calibration (top panel) is a second order polynomial fit to the pulser data and the elastic ^{40}Ca peak provides the energy reference used to make the transition from mV to MeV (bottom panel). Error bars are 1σ, estimated using the channel peak width and the calibration function.

$$t(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 \qquad (x > a_3) \qquad (4.30)$$

$$t(x) = a_0 + a_1 \cdot x + a_2{}' \cdot (x - a_3)^2 \qquad (x \leq a_3) \qquad (4.31)$$

To be explicit, the a_0 and a_1 parameters are determined in the region above the cut-off value a_3, these three parameters are then fixed in the region below the cut-off, leaving $a_2{}'$ as the only free parameter for the low channel fit. The result of these fits can be seen in Figure 4.11.

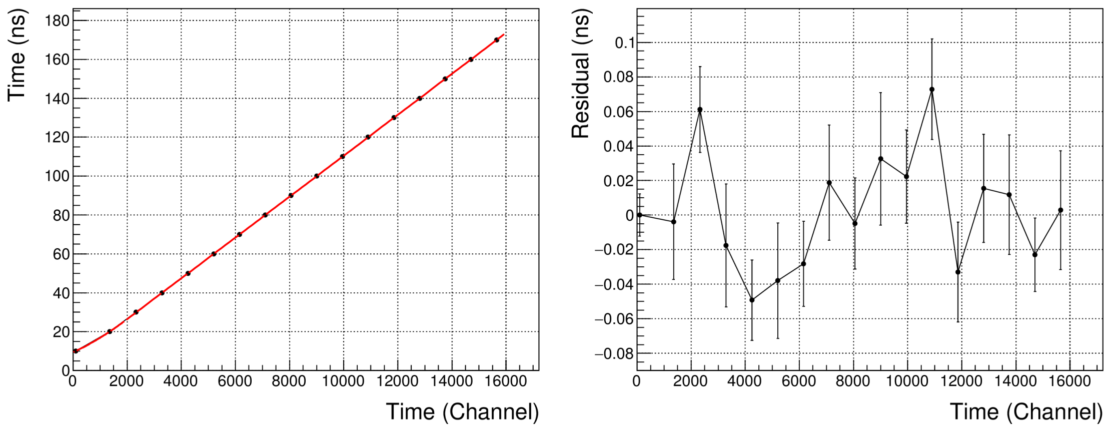

Figure 4.11: Example VAMOS Silicon Time Calibration. The pulse generator data are fit with an almost linear second order polynomial equation (recovering a slight deviation from linearity at high channel numbers) above the "cutoff" channel (1900 in this case) and another second order polynomial equation below this threshold is used to recover stronger deviations at low channel numbers. Error bars are 1σ, estimated using the channel peak width and the calibration function.

Caesium Iodide

As for INDRA, the calibration of the caesium iodide detectors is more involved. The reason being that the light output of the crystal depends upon the Z and A values of the incident fragment in addition to the energy deposited. It should be noted that this calibration also relies heavily upon the silicon energy calibration (see below), if that calibration is incorrect, the following procedure will fail.

For this calibration, a range of well known isotopes is used to provide a set of calibration points for the extra Z and A dependencies. The easiest way to do this is to select nuclei which have a mass-to-charge ratio of two, as these nuclei are easily identified on the raw spectra due to the absence of ^{8}Be, which is unbound. The peaks of these isotopes are then Gaussian fitted to provide their co-ordinates on the silicon raw energy/caesium iodide light output plane. The caesium iodide energy for each of these points is then calculated by simulating the passage of each fragment through the silicon/caesium iodide telescope (taking into account the isobutane dead layer). There now exists a set of points for the various isotopes in the calculated energy/raw light output plane which can be fitted with the following calibration function. Note that this function differs from that appearing in Equation 4.6 as it has been simplified and is known to work well with the particular crystals used in VAMOS.

$$L(Z,A,E) = a_0 \cdot \left[\frac{E}{AZ^2} - a_1 \cdot \ln\left(1 + \frac{E}{a_2 AZ^2}\right)\right] \tag{4.32}$$

Where E denotes the energy deposited in the caesium iodide and the fit parameters have the same meaning as in the INDRA calibration function.

When a nucleus of unknown mass passes through the caesium iodide detector (as is the general case) the energy it deposits in the crystal is a function of its unknown A value, which in VAMOS is determined using the total reconstructed energy loss - a cyclic dependency. In other words the energy loss determination requires the A value, which in turn requires the energy loss determination. The way around this problem is to use an iterative minimisation procedure with an initial estimate for the value of A. This procedure determines the A value by simulating fragment passage through the silicon/caesium iodide detector stack (taking into account inactive detector materials) and minimising the absolute difference between the simulated silicon energy and the true detected energy from the silicon calibration. The A value so determined is then passed as a parameter into the caesium iodide calibration function in order to determine its energy loss. This energy loss is then used to calculate the total reconstructed energy of the nucleus which is then used to determine the value of A in the VAMOS identification routine. If these values of A do not match then this indicates a problem in either (or both) of the caesium iodide or silicon detector calibrations.

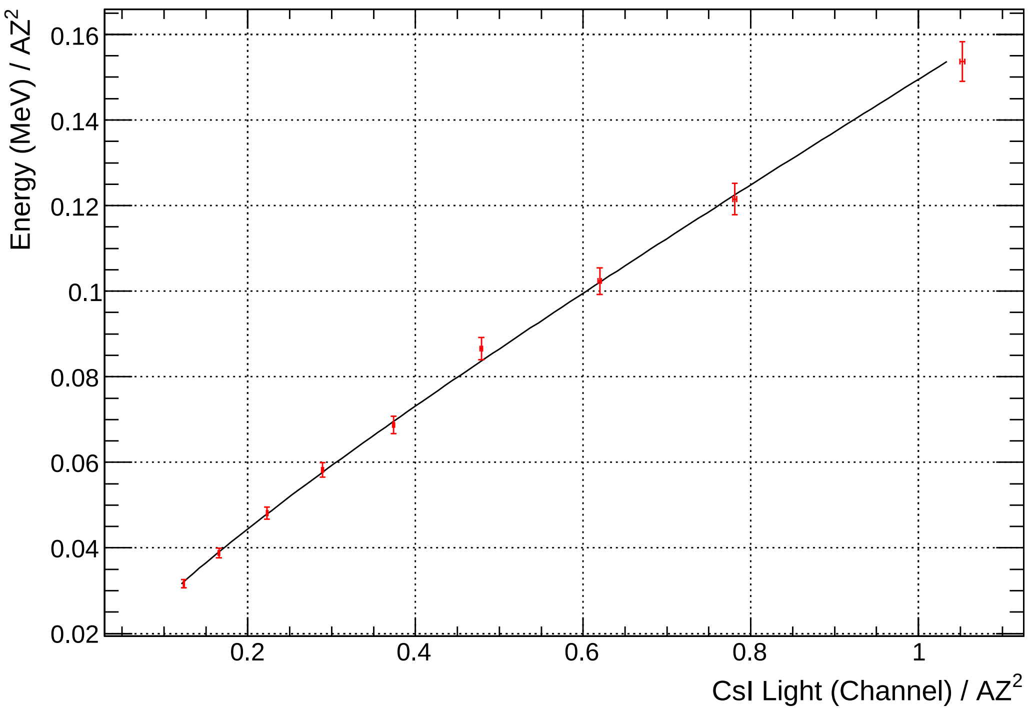

Figure 4.12: Example light output channel to energy calibration for the detector pair: Silicon 01, Caesium Iodide 25. The points on the curve are the result of Gaussian fits to the peaks of well known A/Q = 2 isotopes in the silicon raw energy/caesium iodide raw light output spectrum.

Time of Flight Calibration

The time of flight for a detected fragment is determined using the output of the silicon time amplitude converter. The amplitude of this output signal has a linear relationship to the time difference between the start and stop signals. As stated earlier, the stop signal is taken from the high-frequency signal picked up across the electrodes in the cyclotron and the start signal is provided by the fast amplifier signal of the VAMOS silicon detectors.

The converter signal alone however is not a true time of flight measurement, it simply represents the difference in time between the start and stop signals. Assuming for the moment that the cyclotrons and beam-line operations remain absolutely constant, a detector dependent offset must still be added in order to calibrate the time signal to some well known physical event:

$$t(n) = t_{TAC}(n) + t_0(n) \tag{4.33}$$

Here n denotes the silicon detector number and t_0 the timing offset. The fact that the timing is set individually for each of the detectors is simply a result of their having unique charge collection characteristics and different electronics, which naturally affects their timing. In this experiment the offset is determined by comparison with the calculated time of flight for the elastically scattered ^{40}Ca nucleus. In other words the offset is determined so that $t(n)$ accurately reproduces this calculated value for the elastic peak.

However the above assumption, that the operation of the cyclotrons and beam-line remains constant, is not valid. Throughout the experiment the high frequency signal of the cyclotron does indeed vary from one run to another and even within the same reaction system. There are naturally other sources of variation in the timing signals during the two week long experiment but the aforementioned wandering of the stop signal is probably the principal one. In any case the solution to this problem is to fold in an additional run dependent offset which acts to correct the resulting shift in the mass-charge spectra. The offset is chosen to reposition the A/Q = 2 isotopes at the correct A/Q value. This isotopic selection is chosen as it has linear shape and is easily identified (^{8}Be is unbound and therefore not present in the spectra).

Furthermore, incorrect assignment of an event to its corresponding beam packet can occur if its time of flight is greater than the period of the stop signal ($t_{HF} = 116.96\,\mathrm{ns}$). To fix this problem the time of flight value is shifted by integer values of the stop signal period and is assigned when the difference between the measured energy and that calculated from its velocity is minimised.

Chapter 5

KaliVeda Modifications

In this chapter the modifications to the KaliVeda code that have been made will be discussed in detail. The testing of this code and its current status will be outlined toward the end of the chapter. However before any of these modifications can be discussed, it is necessary to provide some elementary information regarding what KaliVeda is, what it is used for and how the analysis code is structured.

5.1 KaliVeda Overview

The analysis of the INDRA-VAMOS data is achieved using the KaliVeda Analysis Framework. KaliVeda is written almost entirely in C++ but also contains some FORTRAN, it is based on the ROOT analysis framework (`https://root.cern.ch/drupal/`). Its principal use is for the analysis of experimental and simulated nuclear multi-fragmentation data in the INDRA and FAZIA multi-detector arrays. The VAMOS spectrometer is a repository specific addition to this basic functionality for the experiments E494S and E503 (this experiment).

Within KaliVeda a given multi-detector array is defined as a collection of C++ objects representing the various detectors and identification telescopes which constitute the real, physical array. These C++ objects follow a hierarchical structure with absorbing materials as the basis of the detectors and these detectors as the basis of the identification telescopes. By giving these various objects positions in the laboratory frame of reference the multi-detector array is formed. The definition of the absorber volumes is achieved using the ROOT geometry interface and the calculation of energy losses within them is made using KaliVeda's C++ implementation of the VEDALOSS ion range tables [67].

KaliVeda also provides all of the interfaces required for handling and processing the data at an administrative level. For example reading and unpacking the raw data, managing the data repositories and executing various tasks involving the CCIN2P3 Grid Engine (The CPU farm used for batch processing). As a result KaliVeda represents a robust and effective analysis framework for both experimental and simulated data.

For more information on the specific capabilities of the current KaliVeda release please refer to the following links:

- KaliVeda:
  `http://indra.in2p3.fr/KaliVedaDoc/index.html`
- Main Bazaar Repository:
  `https://code.launchpad.net/kaliveda`
- Github Repository:
  `https://github.com/kaliveda-dev/kaliveda`

5.2 Analysis in KaliVeda

KaliVeda currently allows for data analysis tasks of eight different kinds. Three of these are used to perform reconstruction, calibration and identification operations on the data and so serve to promote it to the next stage of the analysis hierarchy which leads ultimately to a data set containing purely physical data - in other words resulting in data that is entirely uncontaminated by any detector specific information.

- Event reconstruction from raw data (raw→recon)
- Identification of reconstructed events (recon→ident)
- Final identification and coherency checks (ident→root)

The remaining five are used to analyse data at these various stages of reconstruction, calibration or identification:

- Analysis of raw data
- Analysis of reconstructed events (recon)
- Analysis of "gene" (Pulser/Laser) data (recon)
- Analysis of partially identified or calibrated reconstructed events (ident)
- Analysis of fully calibrated events (root)

For the purposes of this report however the emphasis will be on the identification of previously reconstructed nuclei (recon→ident).

5.2.1 Analysis Selector Class Example

It is useful at this point to walk through a brief example of a basic analysis program before delving any further into the inner workings of the INDRA-VAMOS code, it should provide some useful insight into how KaliVeda analysis programs generally operate.

In order to conduct any kind of analysis in KaliVeda one has to write an analysis selector class. This class, once compiled and linked against the ROOT and KaliVeda libraries, is loaded and executed by means of the interactive KaliVeda analysis programs (`KaliVeda` or `KaliVedaGUI`).

When working within the framework of KaliVeda this analysis class will generally inherit from `KVSelector` as it provides several virtual member functions which are intended to be overridden in order to define the analysis procedure. The result is that irrespective of the type of analysis being undertaken, the basic structure of the selector class will be the same.

```
class AnalysisClass : public KVSelector
{
    public:

        AnalysisClass();
        virtual ~AnalysisClass();

        virtual void   InitRun();
        virtual void   EndRun();
        virtual void   InitAnalysis();
        virtual Bool_t Analysis();
        virtual void   EndAnalysis();

        ClassDef(AnalysisClass, 1)
};
```

Listing 5.1: Example Analysis Class

These methods are fairly self-explanatory as they execute code at the specified junctures in the analysis program. `InitAnalysis()` at the beginning of the analysis, `EndAnalysis()` at the end of the analysis, `InitRun()` at the beginning of a run, `EndRun()` at the end of a run and finally the code in `Analysis()` is executed for each event in the run.

As an example, in the initialisation of the run, the events can be discriminated according to whether or not they meet certain criteria and the calibration status of the detectors can be displayed.

```
void AnalysisClass::InitRun()
{
    // Define which events are accepted
    GetEvent()->AcceptIDCodes(kIDCode2 | kIDCode3 |
                              kIDCode4 | kIDCode6);
    GetEvent()->AcceptECodes(kECode1 | kECode2);

    // Set parameters for the multi-detectors, print status etc.
    gIndra->PrintCalibStatusOfDetectors();
}
```

Listing 5.2: Example Run Initialisation Method

Most of the time is spent in the `Analysis()` routine however. The general purpose of this routine is to verify that an event is worth processing and if it is, to proceed to iterate over the list of acceptable particles (whether reconstructed, calibrated or identified depending upon the analysis task) and analyse them by running identifications, calibrations, coherency checks or any other desired operations.

```
Bool_t AnalysisClass::Analysis()
{
    // Verify the status of the event
    if (!GetEvent()->IsOK()) return kTRUE;

    // Iterate over the reconstructed particles in accordance with
    // the acceptance conditions defined in InitRun()
    KVINDRAReconNuc *nuc(NULL);
    while ((nuc = GetEvent()->GetNextParticle("ok"))) {
        // Identify, Calibrate, Fill output trees, histograms...
    }

    return kTRUE;
}
```

Listing 5.3: Example Analysis Method

In principle that's all the analysis class itself is really responsible for, it creates the output files (trees, histograms, etc.), it initialises each run, processes each acceptable event in the run (while filling trees, histograms, etc.) and subsequently writes the accumulated data to disk and performs cleanup operations in the various `End()` member functions.

5.2.2 KaliVeda Identification Procedure

The procedure for identifying reconstructed nuclei is as follows. The reconstructed nucleus is defined by the base class `KVReconstructedNucleus` which, together with kinematic and other detector specific data, provides a virtual member function to perform the identification - aptly named `Identify()`. The basic identification operation performed by this function is to call the `Identify()` member function of each identification telescope crossed by the nucleus. The precise nature of these identification functions is obviously highly detector specific and it is here that the real identity of the nucleus is determined.

As an example, take a typical silicon/caesium iodide identification in INDRA. In this case identification grids are used to determine the Z and/or A values of the nucleus. The grid itself is initially defined manually in the silicon energy/caesium iodide light output plane by identifying several well known and easily distinguished isotopes by eye, for example ^{8}Be is not bound and will be missing, therefore the identification of the Be isotopes becomes trivial (See Figure 4.1). Using this initial grid as an input the general identification grid is extracted using the Tassan-Got fit functional method which extrapolates the hand drawn grid across the entire range of expected isotopes (See Figure 4.2 and for more details refer to [65]). The `Identify()` method in this case will therefore be responsible for determining which isotopic band a detected nucleus belongs to based on the calibrated silicon energy and the total caesium iodide light output. The emphasis here is that the identification procedure is unique to this detector pairing.

Due to the detector specific nature of these identification functions the object `KVReconstructedNucleus` is overridden in both INDRA and VAMOS in order to enable their particular identification behaviour.

5.2.3 KaliVeda Calibration Procedure

The calibration of the reconstructed nucleus is achieved in much the same manner as the identification procedure outlined above. When the `Calibrate()` member function is called on the `KVReconstructedNucleus` object, the energy of the nucleus is calculated by summing all of the corrected energy losses in the detectors aligned with the stopping detector (taking into account estimated losses in the target) and its momentum is then set using the dimensions of the identifying telescope. Once again the detector specific energy calibration is obtained via a virtual `GetEnergy()` member function which is overridden in each of the detector classes.

As an example, let us once again take the silicon/caesium iodide example in INDRA. In this case the `GetEnergy()` member function of the silicon detector object will take the raw acquisition data (high-gain and low-gain coder values) and pass them into the channel to voltage calibration function, the resulting voltage is then passed into the voltage to energy calibration function (both of these calibration functions are associated with the detector object) and will then return the calibrated detected energy in the silicon.

The caesium iodide detector on the other hand will take its coder values for the fast and slow components of the light signal and calculate the total corrected light output for the crystal, whereupon, having the prior knowledge of the identified Z and A values (necessary for the light to energy calibration function) it will calculate the total corrected energy detected in the crystal from the light output. These detected energies will then be summed together with estimated losses in the target in order to determine the calibrated energy for the reconstructed nucleus. Once again the emphasis here is on the unique behaviour of the detector pairing, remember also that in this case the caesium iodide light calibration function has a different form for the INDRA and VAMOS crystals.

5.3 INDRA-VAMOS Code

There are currently two versions of the VAMOS code which can be used to perform the reconstruction and identification of the INDRA-VAMOS (E503) data. The original or "old" VAMOS code (as analysed in [68]) is in the process of being replaced as it has become difficult to maintain and understand. It has also not been written within the same object oriented style as the main KaliVeda code. The replacement or "new" code will ultimately be able to perform all of the same reconstruction and identification operations as the old but it will do so with increased rigour and in a more modern, object-oriented fashion which will simplify by orders of magnitude its maintenance and comprehensibility. In fact, one of the main virtues of the new code is that its object-oriented nature renders it almost self documenting (although not entirely so).

In the following sections the changes which have been made by the author to both the old code and the new code will be examined in detail. The development repository for all of the code can be found at the following URL, which also provides the change log. To view the code simply navigate to "Browse the code":

- E503 Development Repository:
  `https://code.launchpad.net/~indra-vamos/kaliveda/kaliveda_e503`

5.3.1 The Original E503 Code

One of the primary tasks necessary before any work could proceed on the new code was to verify the operation of the old code. This work was needed to ensure that the identification corrections which are to be applied in the new code are already functioning correctly in the old code. These identification corrections act to correct distortions in the mass and mass-to-charge ratio spectra caused by various uncontrollable effects such as variations in the behaviour of the accelerator. In testing the old code, one also imparts some degree of reassurance that the old results can be trusted and ensures that any work on the new code can be verified against it. Due to the complexity of the classes which govern the old identification routines however, this procedure took a significant amount of time.

The main operation of the old code can be understood through an examination of just two of its classes: `Identificationv` and `CsICalib`. These two classes account for over $6,000$ lines of code and it is here that all of the identification and identification corrections are made. Unfortunately, even just these two classes cannot be included in this report - even in the appendix - were they to be included they would add a further 90 pages to its length. In any case, the functionality provided by `Identificationv` and `CsICalib` is simple in principle, if not in practice. `CsICalib` serves as a silicon/caesium iodide detector stack used for estimating the A value of the fragment and for handling the isobutane dead-layer. This estimation of the fragment A value is required to evaluate the total energy loss which is a necessary input to the VAMOS identification procedure and is used to determine the true value of its mass (See Figure 4.7). `Identificationv` is where all of the identification procedures and correction functions are co-ordinated and applied.

A cursory glance of these two classes immediately reveals their complexity and the magnitude of potential difficulties awaiting anyone wishing to make alterations to them. The classes as they appear in the development repository[1] have been heavily modified from their original state but remain difficult to comprehend and maintain. Many of the modifications were made simply to ensure the integrity of the memory. The prior use of nearly 90 C-style arrays located on the heap with no bounds checking resulted in significant corruption, which when combined with inconsistent indexing lead to frequent segmentation violations. The logic in the `Identificationv` class has also become impenetrable, with identification codes (which signify the type and quality of the identification) being assigned numeric values without explanation and also used inconsistently. Many modifications were also implemented to increase the processing speed, in particular the lookup operations for the identification grids were previously performed using string comparison against a global-scope grid list which is very slow indeed. In this instance the code was modified to make use of a smaller run-scope grid list (fewer grids translates into shorter lookup times) which could use the fast, in-built hash table lookup operations already present in KaliVeda. All of the aforementioned modifications were implemented with care and over some considerable time to ensure that the results of the identification were not significantly altered. In any case, the unmodified identification classes are still present in the development repository in the directory: `VAMOS/identification/deprecated`.

Just as the operation of the old code can be summarised by these two classes, so too can the operation of these classes be summarised by the actions of just a couple of their member functions. The `Identificationv::Calculate()` function acts as the main identification handler, co-ordinating the use of identification grids, estimations of the fragment A value and implementing the VAMOS identification procedures. It is also responsible for applying any identification corrections as may be required. It is the largest function in the VAMOS code base at around 1300 lines and is difficult to understand and maintain - the processing pathways are generally not very clear and branching occurs unpredictably. While this code may technically function, it does not do so with transparency and much time is wasted when newcomers attempt to understand it. This is one of the principal motivations in the drive to update and clarify the VAMOS code.

The `CsICalib::GetResidualEnergyCsI()` member function is responsible for estimating

[1]`http://bazaar.launchpad.net/~indra-vamos/kaliveda/kaliveda_e503/files` - in the directory `VAMOS/identification`

the mass number of the fragment from the calibrated silicon energy, the caesium iodide light output and the identified Z value. It does this using a bisection method, whereby it repeatedly bisects a range of possible A values with the aim of minimising the difference between the calibrated silicon energy and the same energy as simulated by propagating the candidate particle (Z, potential A value) through the silicon - caesium iodide detector stack. Ultimately two pairs of co-ordinates are determined, one pair in the silicon energy vs. caesium iodide light output plane and the other pair in the silicon energy vs. A value plane. Both pairs straddle the point where the difference between the simulated and measured silicon energies would be zero. Linear interpolation is then used to determine the likely values of caesium iodide energy and A. There are a couple of negative aspects to this procedure, the first is that the range of potential A values is rigid and hard-coded (requiring re-compilation of the KaliVeda source code in order to change it) with no regard as to whether the nucleus to be simulated actually exists, which can waste valuable processing time. The second negative aspect is that it generally requires around 8 simulations per particle in order to converge to its result (something it occasionally fails to do) and once again the flow of the program has become difficult to follow.

The result of processing the data for the elastically scattered ^{40}Ca nucleus with this modified version of the old code can be seen in Figures 5.1 and 5.2. This data provides a useful reference point as the energy and A value of the incident nucleus are well known (34.81 AMeV with $A = 40$). The fact that the energy loss is a little higher than the expected 1392.4 MeV and that the A value returned by the VAMOS identification procedure is also a little high, is not as detrimental as one might think, as these issues are generally accounted for once the identification corrections are made. The reason these points are particularly useful in the context of this report are that they provide values with which the new version of the code must agree in order for the same identification corrections to be applicable. Were this not the case the correction parameters would most likely need to be regenerated for the new code.

It has been mentioned several times already but its importance can not be overstated: the old code has become very difficult to understand and debug, particularly for those without much experience in dealing with it. The myriad hard-coded values which may once have had meaning when the code was originally written have become meaningless through the lack of commentary and are now understood only by those who originally wrote them (and perhaps not even to them at this stage). For this reason it was decided to re-write the VAMOS code. In order to make using the old code a little easier, the author has written a handling class to take care of the basic operations. This code can be found at the following locations:

- Bazaar:
  `https://code.launchpad.net/~peter-wigg-314159/+junk/OldCodeHandler`

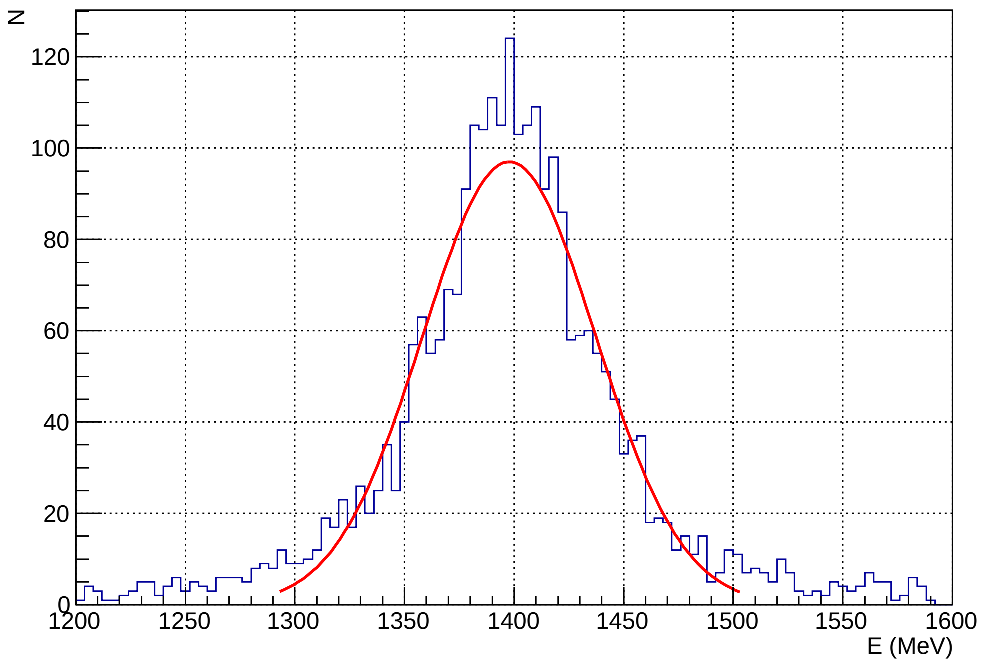

Figure 5.1: Energy measured for the elastically scattered ^{40}Ca nucleus at 34.81 AMeV. The Gaussian fit to this peak results in a measured energy of (1397.8 ± 39.5)MeV. This value is not equal to the expected 1392.4 MeV but as this value is principally used only in the identification procedures it is accounted for in the identification corrections.

- Computing Centre:
  `/afs/in2p3.fr/home/p/pwigg/public/git/OldCodeHandler`

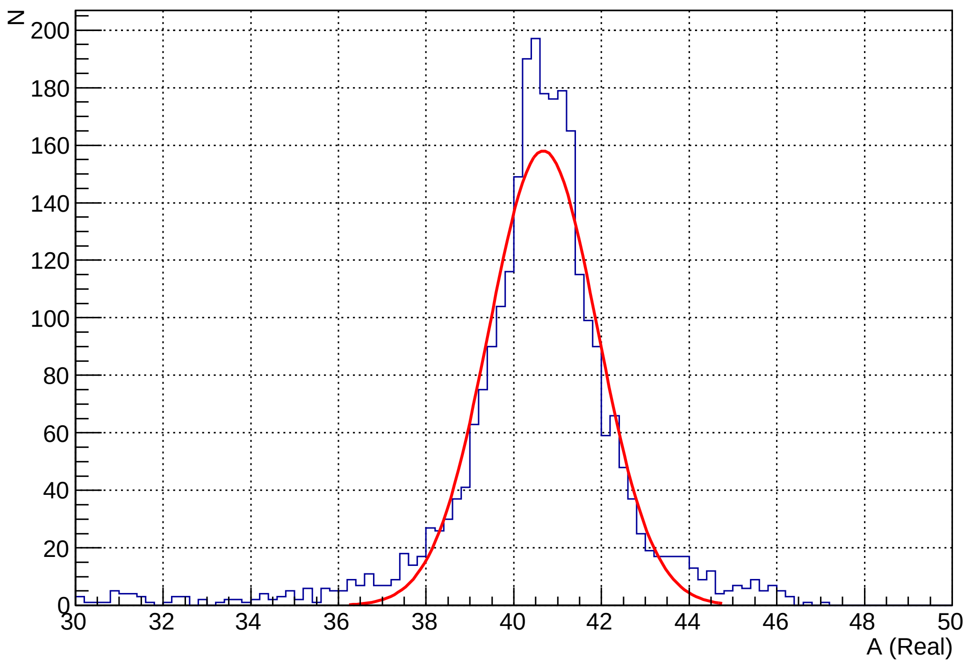

Figure 5.2: The "real" A value (as in floating point) returned by the VAMOS identification procedure for the elastically scattered ^{40}Ca nucleus. The Gaussian fit to this peak results in an A value of (40.7 ± 1.2). This value is not equal to the expected $A = 40$ but this is accounted for in the identification correction procedures of the old code.

5.3.2 The New Code

Much of the re-formulation work for the new version of the code has already been done by Guilain Ademard and Mark Boisjoli. In particular, the trajectory reconstruction code for VAMOS which is shared between E503 and its sister experiment (E494s) together with basic identification classes and the VAMOS specific identification equations are already in place. The implementation of the corrective identifications of the old E503 code however is still missing. It has been the work of the author to migrate the aforementioned corrective identifications into the new framework. For reference, links to both the main INDRA-VAMOS development repository and the E503 specific development repository of the author are provided here:

- The Main INDRA-VAMOS (E503 and E494S) Development Repository:
  `https://code.launchpad.net/~indra-vamos/kaliveda/IV`

- The E503 Development Repository:
  `https://code.launchpad.net/~indra-vamos/kaliveda/kaliveda_e503`

Ultimately the changes made in the E503 repository will be merged back into the main INDRA-VAMOS code, but at the time of writing all of the code changes detailed in this report can only be found in the E503 repository. As a side note: the code alterations for the E503 experiment have been kept within their own namespace to prevent polluting the global namespace during testing. KaliVeda does not typically make use of namepaces and so it is possible that the namespace will be removed once testing is complete, although there is no harm in not doing so (in fact it improves the encapsulation).

Database Classes

Before any of the identification corrections can be applied, the new version of the code must be able to load the old correction parameters in a way that is consistent with the methods already employed in KaliVeda. For this purpose two classes were written to read the correction parameters from their respective parameter files and to assign them to their associated identification telescope objects. The `KVIVDB_e503` class reads the parameters and loads them as database records into KaliVeda, the `KVIVUpdater_e503` class handles the assignment of these records to the various identification telescope objects in VAMOS. The correction parameters for each telescope object are then determined and assigned automatically at the beginning of each run.

Identification Telescopes

In order to assign the identification correction parameters as previously outlined, the basic functionality of the existing identification telescope classes had to be extended. There were initially two classes in the VAMOS code which were used to define the basic elements of the ionisation chamber vs. silicon and silicon vs. caesium iodide telescopes. Due to the fact that these corrections are currently very specialised and designed specifically for use in the E503 experiment, it was decided that rather than to simply add this functionality to the existing classes, it was better to construct two new classes which inherit their basic functionality from the general ones. The resulting telescopes can be found in the `KVIDHarpeeICSi_e503` and `KVIDHarpeeSiCsI_e503` classes. In the case of the ionisation chamber vs. silicon telescopes the identification corrections are currently the only major alteration to their behaviour. The silicon vs. caesium iodide telescopes however warrant a little more attention.

The silicon vs. caesium iodide telescopes have different behaviour. To start with, the residual energy component used by the identification grids is not the caesium iodide energy itself but the total light output of the crystal, calculated using the fast and slow integrated components of the signal. This signal needs to be pedestal subtracted before it is used, unlike the silicon energy signal where the pedestal is automatically taken into account in its calibration function. The most interesting part of the operation of this telescope however is to be found in how it evaluates the A value of the detected fragment. The precise nature of how it does so is explained in detail in the following section but to summarise its operation: an estimation of the value of A is determined by minimising the difference between simulated and calibrated (detected) silicon energy for a set of test fragments.

Threaded Mass Estimator

It has been mentioned earlier in this report that the light to energy calibration function of the caesium iodide crystals depends upon the Z and A value of the incident particle as well as its energy. This results in a problem: how is that the total energy of the fragment can be determined, such that it can be passed into the VAMOS identification equations (in order to determine its A value) if the A value is not already known? It might sound a little confusing so to state it differently: how can the caesium iodide contribution to the total energy be evaluated without already knowing the A value of the fragment? The answer is of course that it can not be and since the caesium iodide contribution is certainly not negligible another method of estimating the A value must be found.

In the new code, the procedure which has been adopted is similar to the bisection method of the old code (outlined earlier in this chapter) although with several significant improvements. It operates on a similar principle, it takes a range of potential A values for any given Z value, simulates the passage of these particles through the detector stack of the silicon vs. caesium iodide telescope (including dead-layers) and minimises the magnitude difference between the simulated and the experimentally observed (from the calibration) silicon energies, termed $\delta = |E_{Si} - E_{Si,sim}|$. As this procedure must occur for every particle that passes through every silicon vs. caesium iodide telescope in the VAMOS setup, it is critically important that it be kept as fast as possible and at the same time using as little memory as possible (memory allocation/deallocation is very expensive).

The old code bisection method could determine the A value of an incident particle within around 8 simulations, typically (See Figure 5.15). It was the goal of the new method of mass estimation to reduce this number as far as possible. One of the first possibilities that springs to mind is to use threading so that multiple simulations can be processed in parallel. That is precisely what the threaded mass estimator (as its name implies) does, currently using two threads. The code for this new mass estimation method can be found in Appendix B.1.

The operation of the mass estimator is summarised in Figure 5.3. Firstly a range of candidates for the A value of the nucleus is determined from its Z value. A potential candidate is defined as any nucleus which, when constructed, has a positive lifetime - that is, a nucleus which is known to exist, for however short a time. One of these values is chosen to provide an initial reference point in (δ, A), the choice of this initial value for A from among the list of possibilities is significant as it will govern the load balancing of the two threads. If the threads are not loaded in a balanced fashion then more processing will be done in one thread than in the other and the increased speed that comes with using two balanced threads will be lost. Naively one might expect the value of $A = 2Z$ to be a good starting point (close to the probable true value for the nucleus) but this turns out not to be the case. It has been determined, empirically, that the optimum starting value is $A = 2.25 \cdot Z$ (which is then cast to an integer) as this value maintains the load balance between the two threads for the range of Z values expected in this experiment (See Figures 5.5 and 5.6).

Once this initial value of A has been chosen its passage is simulated through the detector stack and its δ value is determined. This point now acts as the reference standard against which both threads compare themselves (as they operate independently). Each thread then iterates away from the initial starting point in opposite directions, one decrements A, the other increments it. At each step the threads compare their current result to the reference standard, if the value of δ is larger than the current reference value then the thread will terminate, if its value is lower, then the reference will be updated to this new minimum value and the thread will continue to iterate until δ begins to maximise. In short, both threads search for the minimum delta within their specific range of A values and return as soon as δ starts to maximise.

When both threads have returned their minimum value for δ they are compared to one another in order to determine the true minimum value and therefore the most likely candidate for the A value of the nucleus.

Threaded Mass Estimator Testing

In order to test the operation of the threaded mass estimator (and indeed to compare it against the old bisector method) a set of input nuclei with randomised Z, A and incident energy were generated and their passage simulated through the detector stack. To ensure that the caesium iodide energy to light conversion was not perfect, a random Gaussian deviation with $\sigma = 5.5\%$[2] was added to the simulated light output. This is necessary as the light output for a crystal with a given simulated energy loss is determined using the caesium iodide calibration function and since this is also used to convert the light back into an energy by the mass estimator, its energy determination would be perfect. A further random Gaussian deviation with $\sigma = 2.0\%$[3] was added to the silicon energy input value in order to simulate the observed variance in the true signal. Had these two contributions not been accounted for, the minimised δ would have been exactly zero, which is of course, unphysical. The correlation between

[2]Gaussian fit to the experimental caesium iodide light peak for elastic ^{40}Ca

[3]Gaussian fit to the experimental silicon energy peak for elastic ^{40}Ca

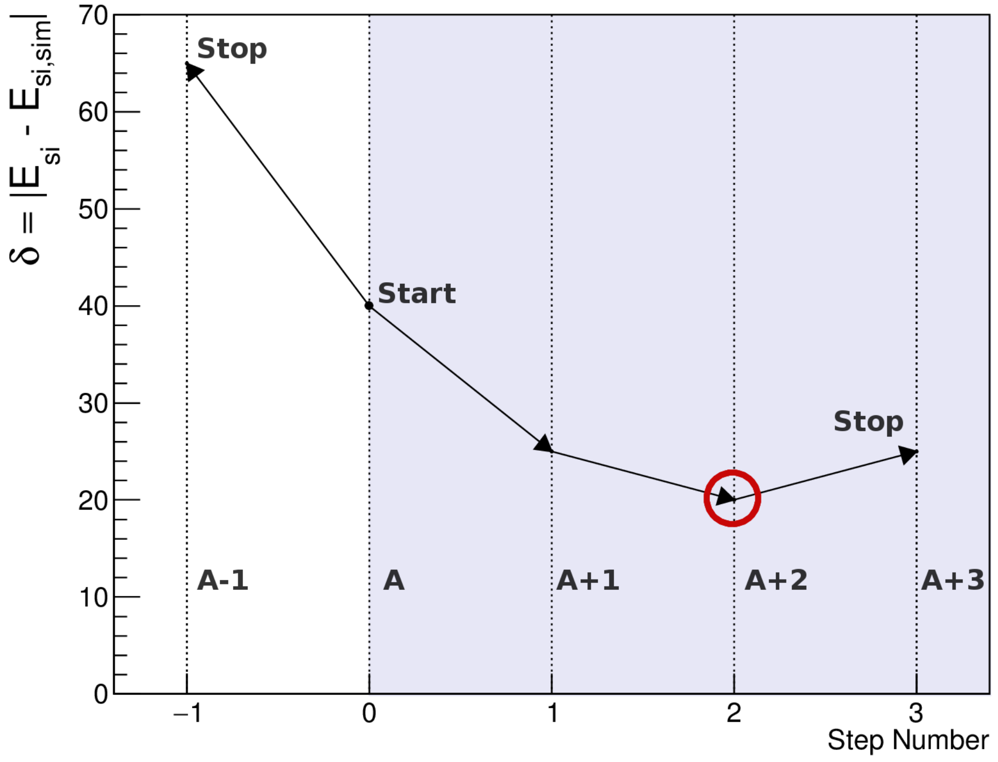

Figure 5.3: This figure demonstrates how the threaded mass estimator works. Both threads have the same initial value for δ, the point marked as Start. Each thread then proceeds to evaluate new values for δ, working away from the starting point. The "forward" thread (shaded region) increases the A value at each step and re-evaluates δ, while the "backward" thread does the same but works in the opposite direction (decreasing A). When either thread encounters a problem (e.g. δ starts maximising) it terminates. When both threads have returned, their minimum values for δ are compared in order to determine the true minimum value and so find the likely value of A (red circle).

the input A value and that determined by the mass estimator can be seen in Figure 5.4. Approximately 37.1% of the 100,000 events generated in this case were identified correctly and 75.6% were found within one mass unit of their true value, across the entire range of Z (The caesium iodide calibration should not be considered valid much above $A = 45$). This indicates that the threaded mass estimator is working well.

The effect of the load balancing can be seen in Figures 5.5 and 5.6. Each thread is processing approximately the same number of events before returning, this means that the optimal number of simulations are occurring in parallel. Technically speaking this is not precisely true, due to the nature of how KaliVeda handles its event simulations the detector stack cannot be accessed by both threads simultaneously (global pointers are used in the core KaliVeda code), a mutex must be locked by whichever thread is

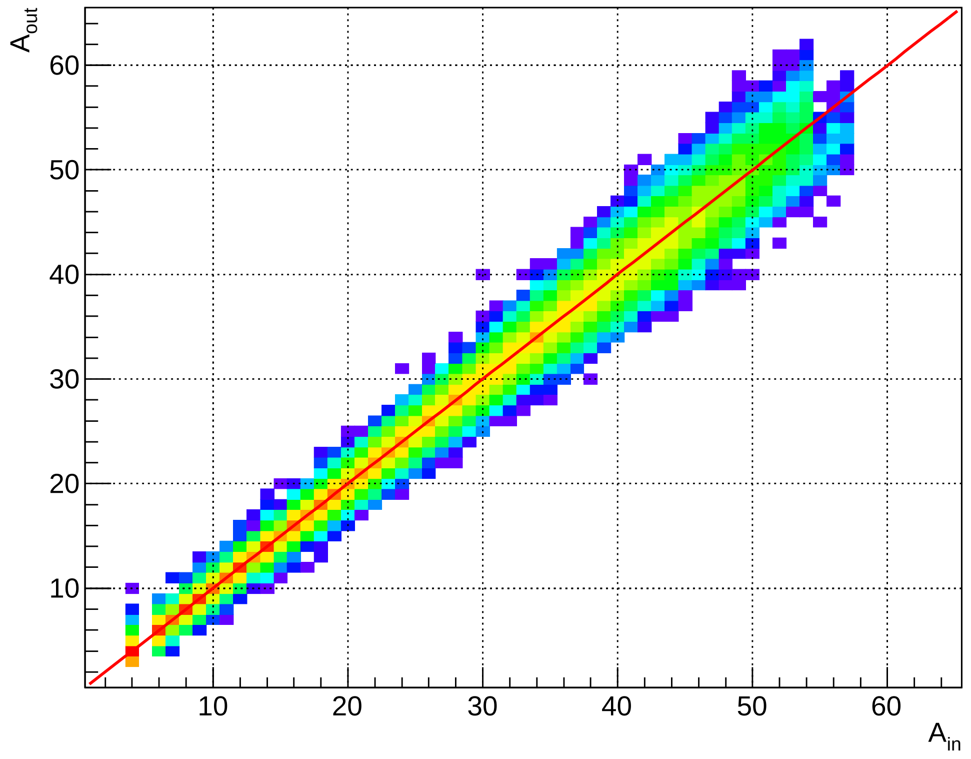

Figure 5.4: Mass number correlation for 100,000 simulated events using the new `ThreadedMassEstimator` class. The straight line overlay denotes $y = x$, that is, a correct determination of the mass number. 37.1% of these events are estimated perfectly, 75.6% are within one mass unit. The scale on the z-axis is logarithmic in order to demonstrate the symmetry about $y = x$ a little better.

running its simulation in order to prevent the other thread from accessing the detector stack at the same time. Now, both threads have a lot of other tasks to perform besides the simulation itself and so this method does speed up the code significantly, perhaps by a factor of $1.5 - 2.0$.

To demonstrate the "energy gap" with which the A value is determined, the final value of $\delta = |E_{Si,sim} - E_{Si}|$ (i.e. the minimum value of δ) which of course determines the A value, is plotted against Z in Figure 5.7 and its fractional value $(\frac{\delta}{E_{si}})$ is likewise plotted in Figure 5.8. Note that the fractional value is generally less than 10% and is less than 1% for the elastic ^{40}Ca nucleus. In combination with the observed A value correlation this would seem to indicate that the mass estimator is working well.

Finally, the residual difference in the A value is shown in Figures 5.9 and 5.10. It can be clearly seen that the majority of events have their A values determined within only a few mass units of the true value, it is also clear that there is a near exponential reduction as one moves away from the true value. The residual A value difference distribution does widen slightly as one moves to larger values of Z but it remains

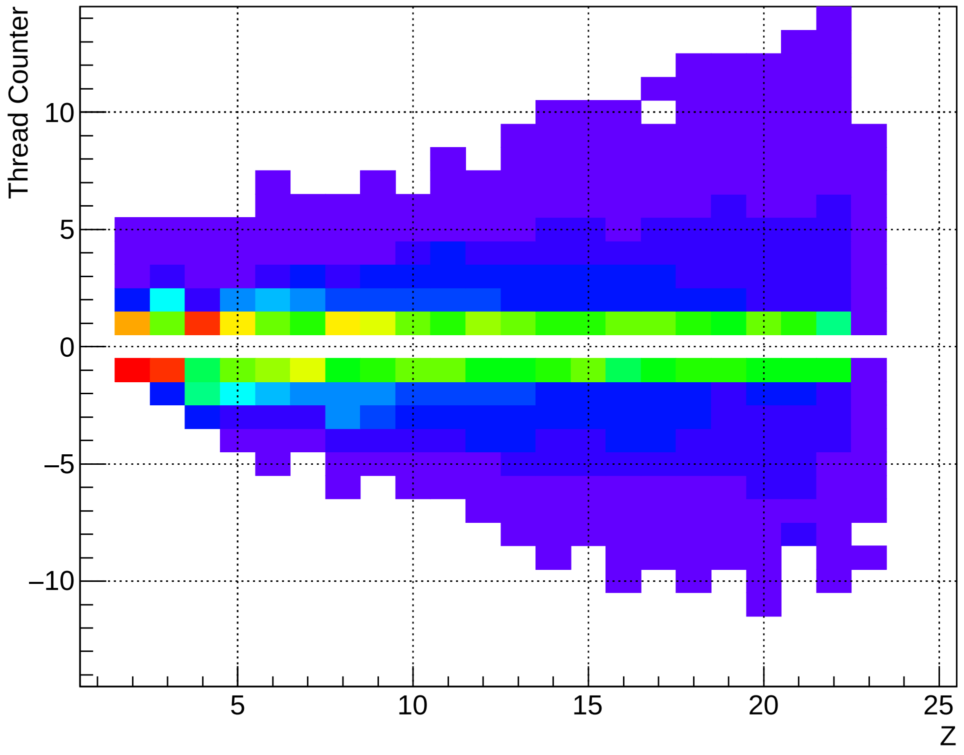

Figure 5.5: This figure demonstrates the number of simulation steps required for the threaded mass estimator to determine its value for A. Simulations in the forward (increasing A) direction are counted as positive, those in the backward (decreasing A) are counted as negative. Remember that these simulations are occurring in parallel and that the zeroth simulation is the initial reference standard (See text). Note that the balance between the threads shifts with increasing Z.

symmetric.

Although in principle there is no real limit to the range of A values which can be processed by the threaded mass estimator, it is effectively limited by the calibration range of the real detectors, in particular the range of Z and A values for which the caesium iodide is calibrated. For this reason a status code is implemented in order to specify whether or not the input data indicates that the estimation routine is operating outside of the calibration range. This cannot be achieved with a simple cut on the A value (as it is unknown) instead a hard cut-off value on the silicon energy is used, above which the mass estimator will still return a result but will mark it as "likely to be outside of the calibration range". It is then up to the user to decide whether the event will undergo further processing in their analysis class.

The effect of this condition on the silicon energy ($E_{Si} < 1000\,\text{MeV}$) can be seen in Figure 5.11, events marked red are deemed outside of the calibration range ($E_{Si} \geq 1000\,\text{MeV}$). It can be seen that in general the mass estimator will still return sensible

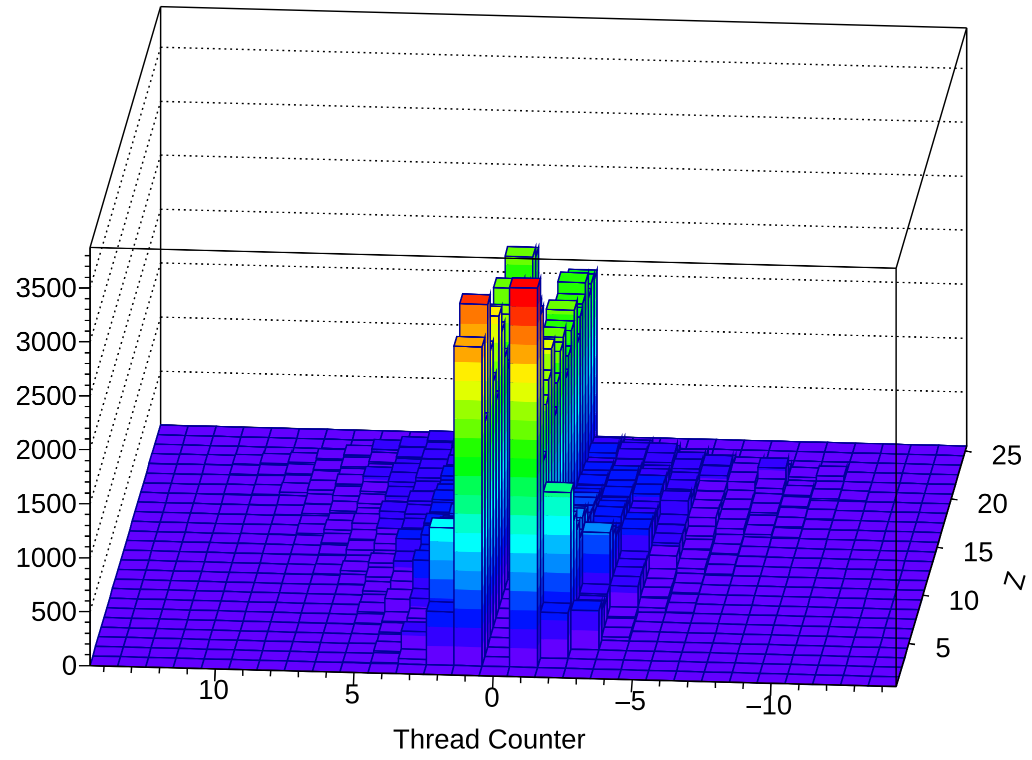

Figure 5.6: This figure contains the same data as in Figure 5.5 but attempts to convey more clearly the fact that most of the events are determined within one step in each thread. As before, this figure demonstrates the number of simulation steps required for the threaded mass estimator to determine its value for A. Simulations in the forward (increasing A) direction are counted as positive, those in the backward (decreasing A) are counted as negative. Remember that these simulations are occurring in parallel and that the zeroth simulation is the initial reference standard (See text).

results outside of the calibration range but above $A = 50$ there are increasing numbers of poorly identified events. The reason for this is simple, the caesium iodide calibration is only valid up to around $A = 45 - 50$, in fact the maximum value of A for which any of the caesium iodide detectors are calibrated is $A = 45$, anything much above this value can result in problems. In this experiment however it is expected that nearly all of the events will be within this calibration range.

The threaded mass estimator is essentially a generic minimisation routine and in principle its behaviour could be extended to other detectors in future, as the detector stack can simply be switched out for another one. Also since δ is defined by the detector stack, the mass estimator can minimise on other variables as well and not just the silicon energy difference as stated here. As an additional side note: the threaded mass estimator was intended to make use of the ROOT class `TThread` due to its platform independence. Ultimately the implementation of `TThread` proved far too difficult to use

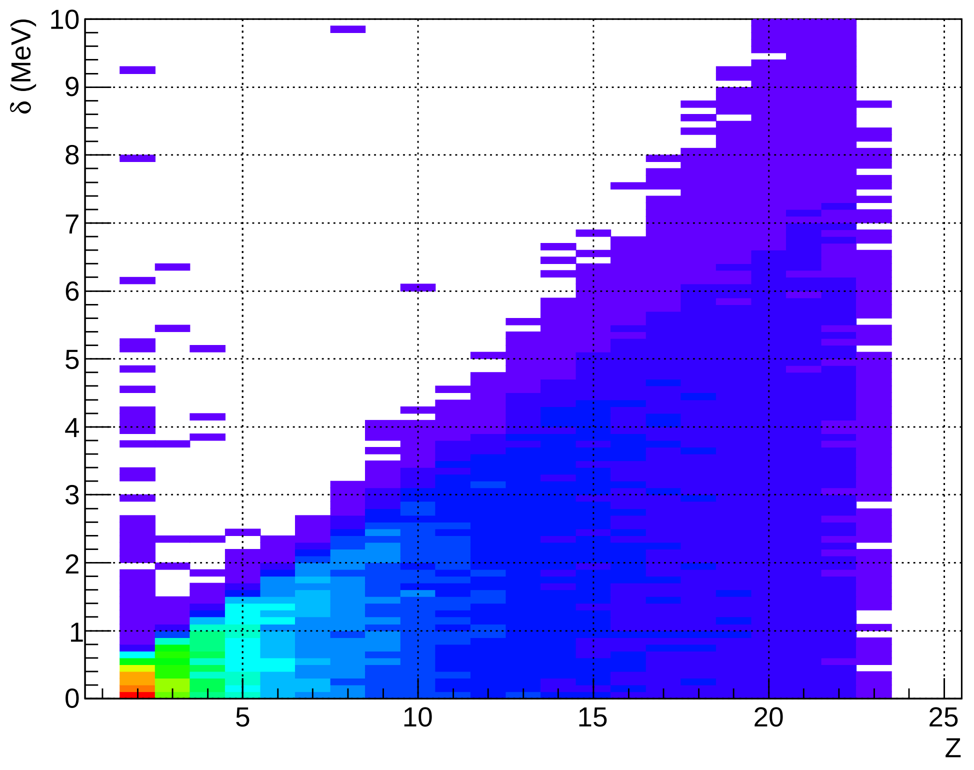

Figure 5.7: This figure shows the minimum value of $\delta = |E_{Si,sim} - E_{Si}|$ in MeV (which determines the A value) as a function of the input value of Z.

in practice. The reason for this is that the kernel recycles the thread id of a successfully joined thread but the cleanup operations in `TThread` can still be trying to use this (now re-assigned) thread id. The result is a segmentation violation after a random number of events which is difficult to debug. This is generally not a problem but in the case of the threaded mass estimator the threads are created and joined very rapidly which seems to exacerbate the issue. Anyway this situation forced the use of posix threads (pthreads) which means that the E503 code can now only run on a machine where these are available. This is not a significant issue however as other parts of the main KaliVeda code are dependent upon linux system headers and the code can still be run without issue at the computing centre.

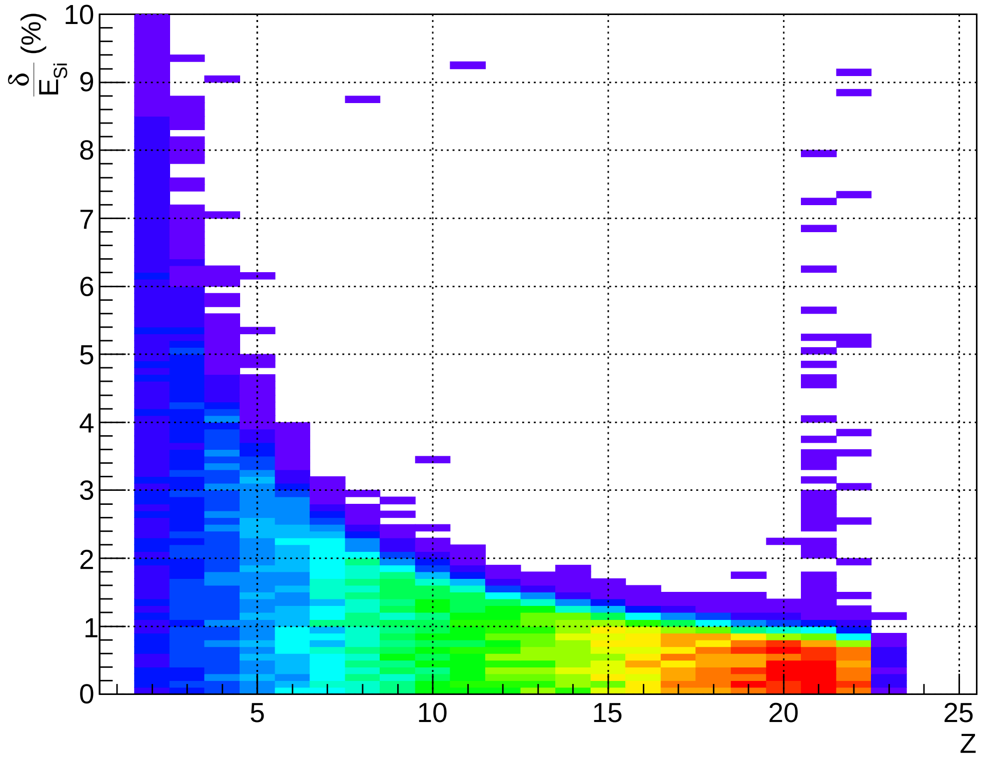

Figure 5.8: This figure shows the fractional minimum value of δ in % (which determines the A value) as a function of the input value of Z. The fractional value is simply defined as: $\frac{\delta}{E_{Si}} \equiv \frac{|E_{Si,sim} - E_{Si}|}{E_{Si}}$.

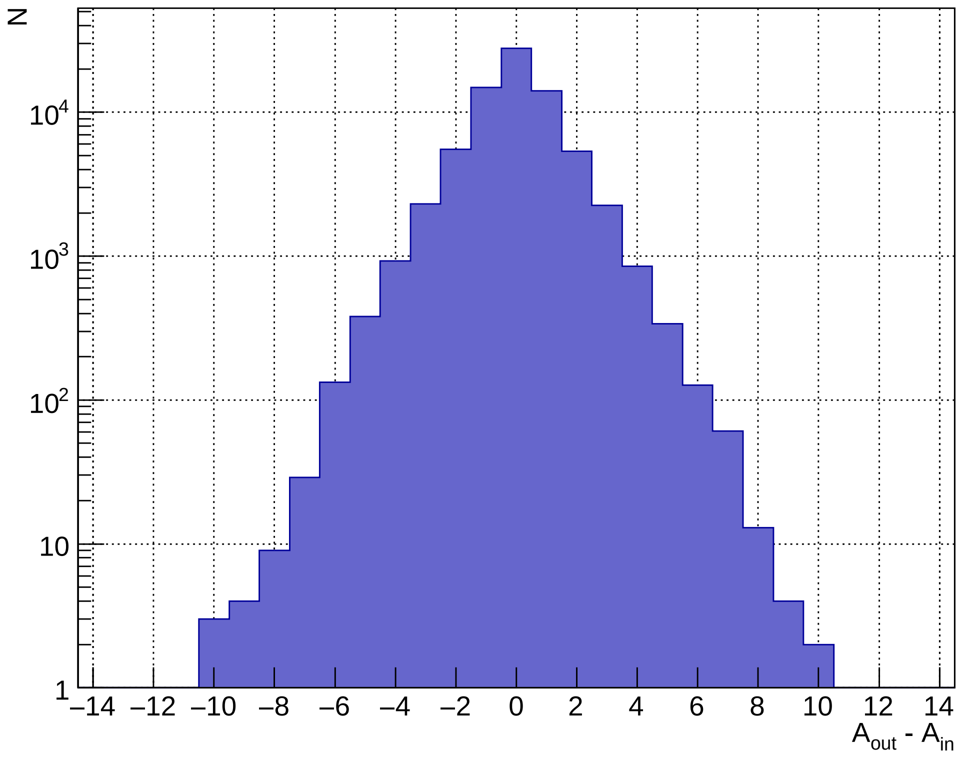

Figure 5.9: The residual mass difference between the input value of A and the value returned by the threaded mass estimator. Note the near exponential reduction as one moves away from the true value.

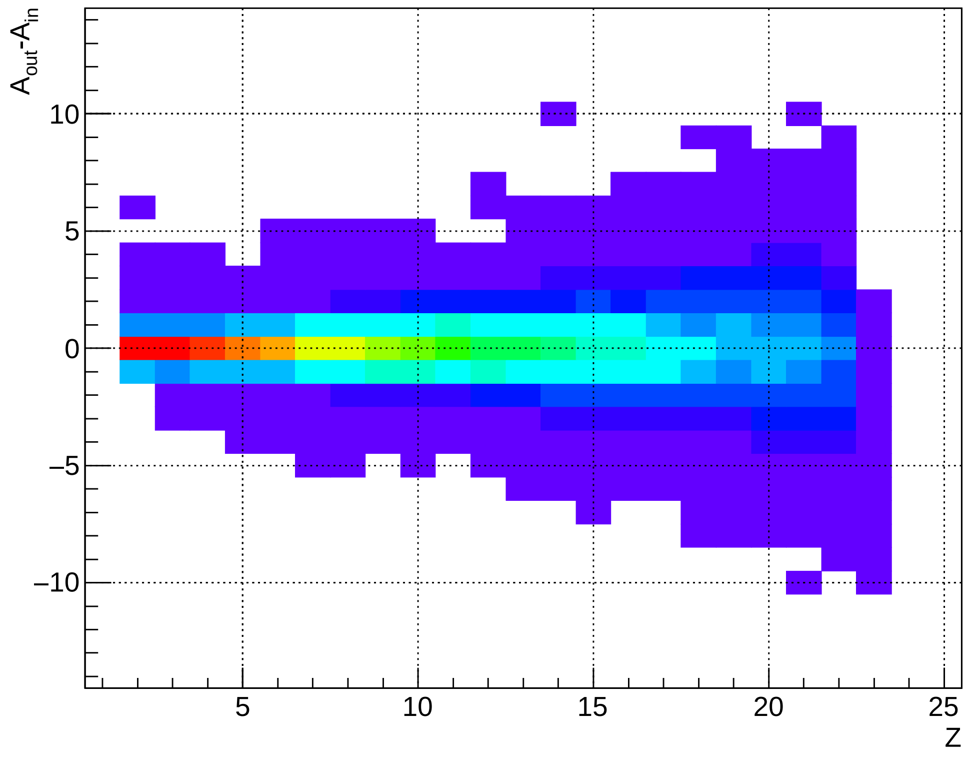

Figure 5.10: The residual mass difference between the input value of A and the value returned by the threaded mass estimator, as a function of Z. As one moves to higher and higher Z values, the residual mass difference distribution does widen slightly.

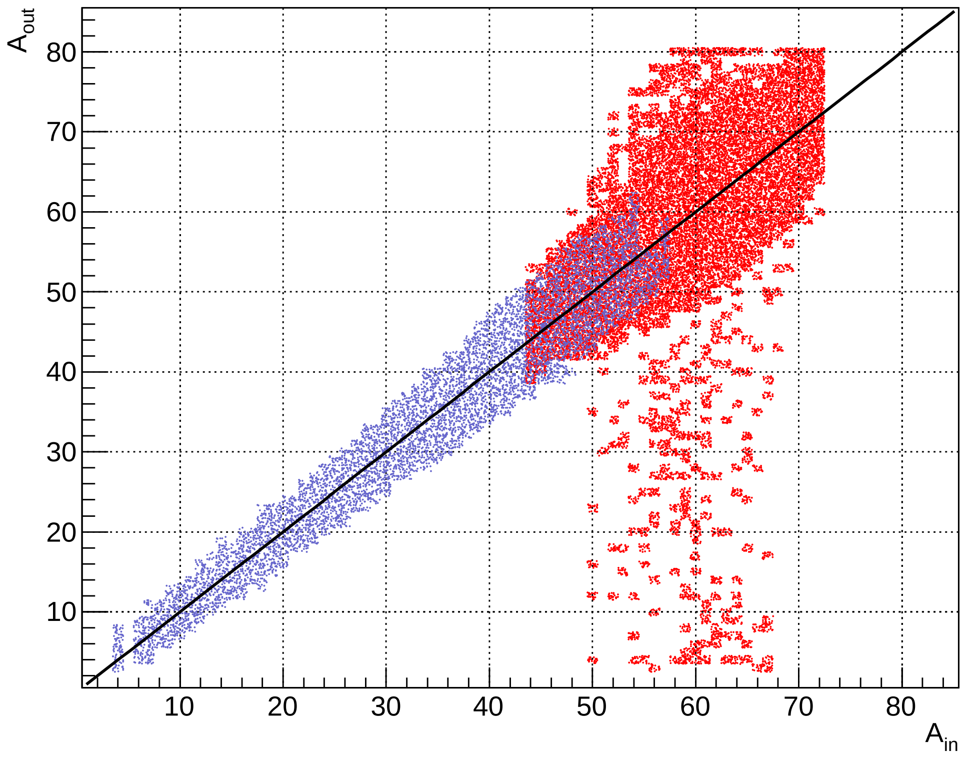

Figure 5.11: This figure demonstrates the effect of the hard cut on silicon energy which is used to restrict the range of A values to approximately that defined by the range of the caesium iodide calibration (See text). This mass number correlation is for the same 100,000 simulated events as in Figure 5.4. The straight line overlay denotes $y = x$, that is, a correct determination of the mass number. 37.1% of these events are estimated perfectly, 75.6% are within one mass unit. The blue points are those returning a valid status code and are therefore within the calibration range, the red points are those which reside outside the calibration range.

Bisector Testing (For Comparison)

If the same procedure is used to test the old bisector algorithm it does not perform quite as well (See Figure 5.12).

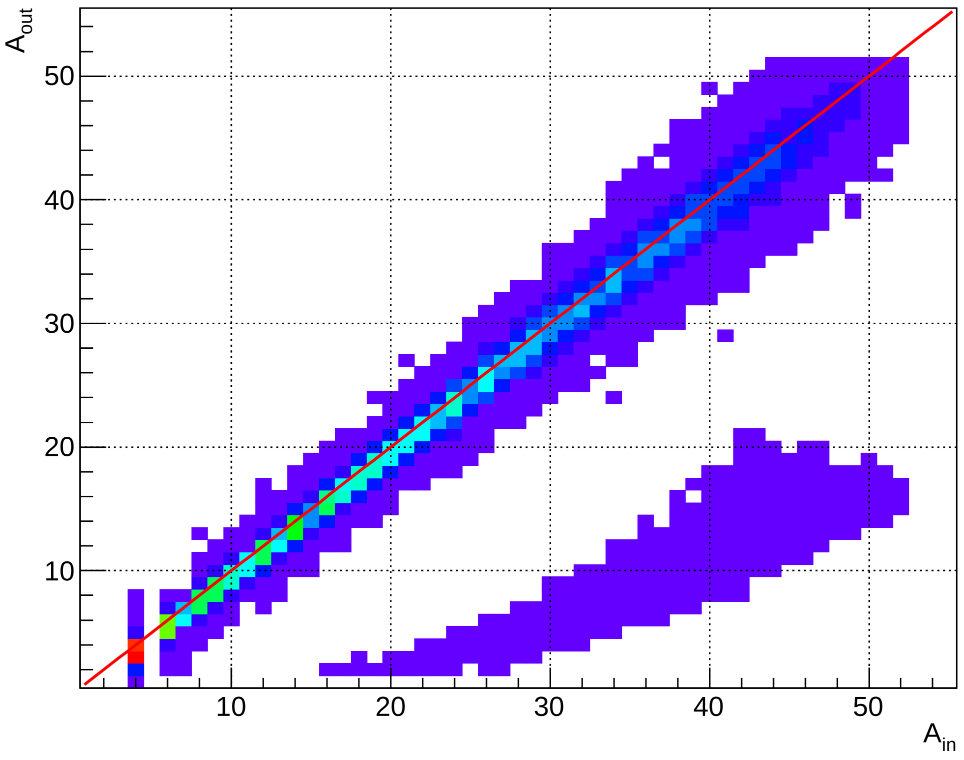

Figure 5.12: A value correlation results of testing the old bisection method. The red line denotes the line of equality $y = x$, i.e. perfect correlation. Note that the distribution is much broader than the new method as it appears to have been pulled below perfect correlation in many cases. Note also the very peculiar band of very poorly identified nuclei. 29.8% of the 100,000 events simulated are identified perfectly, 72.0% are determined within one mass unit.

In this case the A value is determined correctly in 29.8% of the 100,000 events simulated, 72.0% are determined within one mass unit. However, the determination of the mass number seems to be evenly split between $A_{out} = A_{in}$ and $A_{out} = A_{in} - 1$ (See Figure 5.13). In other words, rather than the bin with maximum content occurring at the true value of A it seems to occur evenly split between the aforementioned A values. For some reason not entirely understood at this time there is a shift in the bisector output A value towards a lower A for some events (See Figure 5.14).

The effect of the finite, hard-coded range in A value also results in failed bisection attempts (approximately 3% of events - not shown in the figures) and demonstrates that the method is unsuited for handling a wide range of Z and A input values. There

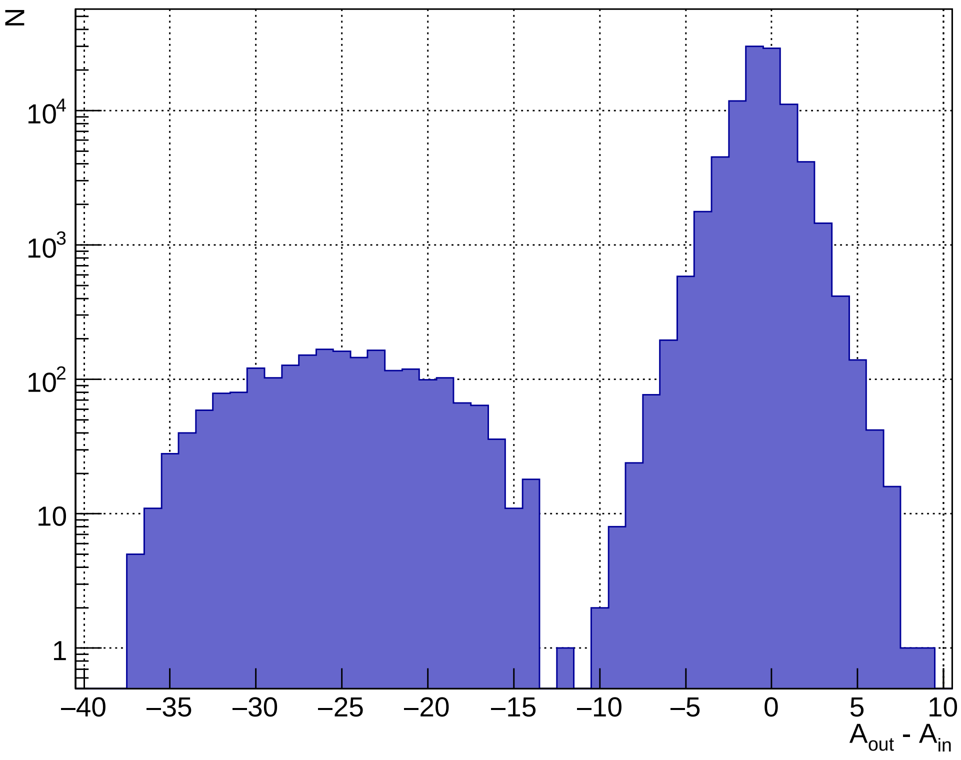

Figure 5.13: The residual A value result from the test of the old bisection method. Note that the peak content does not occur at the true A value as it does in the new method, instead it appears to be evenly split between A_{in} and $A_{in} - 1$. The fall off is still approximately exponential around the true value as it is in the new method. The band of poorly identified nuclei are responsible for the second peak on the left.

are also no safe-guards against processing results which are not within the accepted calibration range for the detectors. This method will process events outside of the calibrated range of Z and A without warning. Such events are perhaps responsible for the band of poorly identified nuclei visible in the figures. On top of these issues, the old code also runs more slowly, returning its result after typically 8 iterations (simulations) per particle. In real terms the event rate is about 40 events/s compared to the 60 events/s of the new threaded method (on the same machine).

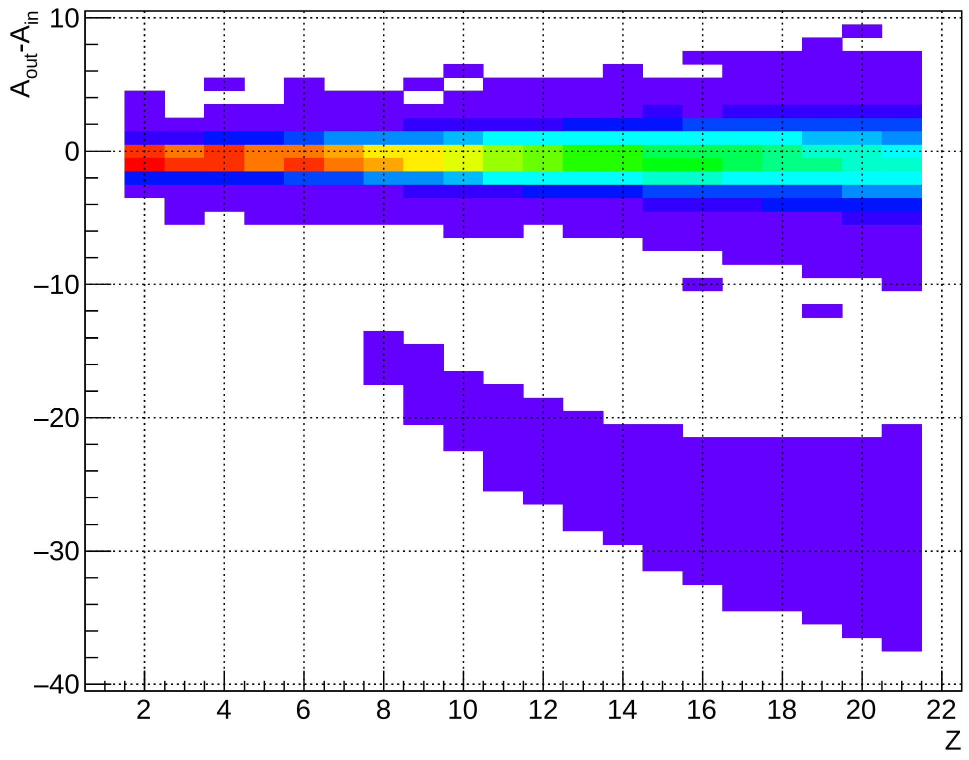

Figure 5.14: The residual A value as a function of Z from the test of the old bisection method. It is clear that the data is not symmetric around the true value as it should be and that the downward shift in the output value of A is an increasing problem at higher values of Z. There is perhaps a hint that the band begins to curve downward at around $Z = 20$, although this may simply be a statistics issue.

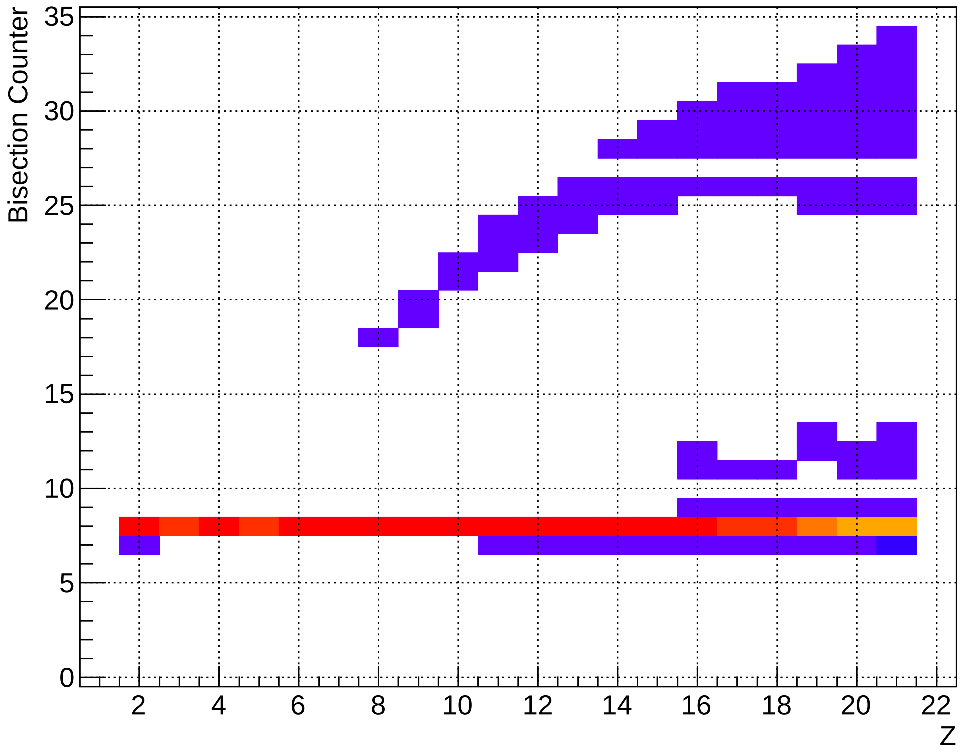

Figure 5.15: The number of simulations required for the old bisection method to return its result for the test input nuclei. The typical value is around 8 simulations for each particle, although this value seems to increase slightly towards the higher Z values. The poorly identified band of nuclei take significantly more iterations to evaluate and this value also increases with the Z value.

Identification Post-Processing

In order to facilitate the identification of fragments using the new mass estimation method and to provide a mechanism for applying the identification correction parameters of the old code, the `IdentifyE503` class was created (See Appendix B.5). This class acts as a post-processing addition to the standard KaliVeda identification procedure. The general idea is that instead of calling the identification methods on the reconstructed nucleus directly, one simply passes this nucleus into this new class, it will then perform all of the identification operations and corrections internally. The major difference to the standard procedure here is that the identification corrections will be used to actively modify the input nucleus properties, setting its A value, total energy etc. The standard identification methods already do this with the identification grids, this class post-processes this data even further to apply the corrections in a way that is more or less transparent to the user. Below is an example of how this class is called in an analysis selector class:

```
void AnalysisClass::InitAnalysis()
{
    ...

    id_handle_ = new e503::IdentifyE503();
    id_handle_->Init();

    ...
}

Bool_t AnalysisClass::Analysis()
{
    ...

    KVVAMOSReconNuc *nucleus(NULL);
    while ((nucleus = GetEvent()->GetNextNucleus("ok"))) {

        UChar_t idrc(id_handle_->Identify(nucleus));
        if (idrc != 0) {
            // Handle any errors
            continue;
        }

        if (nucleus->IsIdentified()) {
            // Fill Trees etc.
        }
    }

    ...
}
```

Listing 5.4: IdentifyE503 Usage Example

Once the nucleus has passed into the `IdentifyE503` class it is initialised and checks are made to ensure that its time-of-flight and flight distance are within the expected range, if this is not the case or if the initialisation fails in any way then the event is not processed any further and an error code is returned. If the particle is correctly initialised then it is identified in the standard KaliVeda manner, by calling the `Identify()` method for each identification telescope that the particle passes through. Remember that it is here (`KVIDHarpeeSiCsI_e503::Identify()`) that the threaded mass estimator is invoked. Once the basic identification is complete, several checks are made to ensure that the particle is well identified, if it is not, the particle is not processed any further. At this stage the particle has a known Z and either a known or a calculated A value (depending upon whether the A identification was possible).

With this data it is then possible to determine the total energy loss of the nucleus by summing the individual energy losses of all of the aligned absorbing materials between the stopping detector and the target (including dead layer corrections). This procedure has to be handled carefully as any calculated energy losses depend sensitively upon the energy calibrations of those detectors which provide the true energy measurements. Final corrections are also made for the strip foil and target energy losses.

At the time of writing only one of the old identification corrections has been implemented as a proof of concept, the mass-to-charge ratio correction. The parameters for the mass-to-charge correction function are loaded using the aforementioned database classes and the correction is applied using the `CorrectAoverQ` member function. It should not be difficult to extend the functionality of the `IdentifyE503` class to include the other corrections. Unfortunately this is not possible within the scope of the current work as there are a couple of problems in the reconstruction which need to be understood first. There are several reconstruction discrepancies between the new code and the old code, in particular the velocity (β) values are not in perfect agreement and this needs to be well understood as it is used extensively in the VAMOS identification equations. The correction parameters will most likely need to be regenerated if this problem turns out to be an issue in the old code. There are also some reconstruction errors for silicon detectors 9 and 10 relating to the setting of the flight path of the nucleus - at present the elastic run 587 (where the elastically scattered ^{40}Ca nucleus is striking these detectors) suffers badly. Once these problems are resolved or if the identification parameters have been regenerated it should be possible to implement them readily in the `IdentifyE503` class.

The primary test of the identification code is whether or not it can reproduce the expected characteristics of the elastically scattered ^{40}Ca nucleus in the experimental data. This is something that the current code is only able to do in part due to the remaining unimplemented identification corrections. The total energy and real A value (as returned by VAMOS) for the elastic peak in the old code can be seen in Figures 5.1 and 5.2 and the same data for the new code can be seen in Figures 5.16 and 5.17.

Both the new code and the old code return approximately the same energy, around $1398 - 1399\,\mathrm{MeV}(\sim 34.95\,\mathrm{AMeV})$, this is larger than the expected theoretical value if one takes the beam energy recorded in the log book ($34.81\,\mathrm{AMeV} \implies 1392.4\,\mathrm{MeV}$),

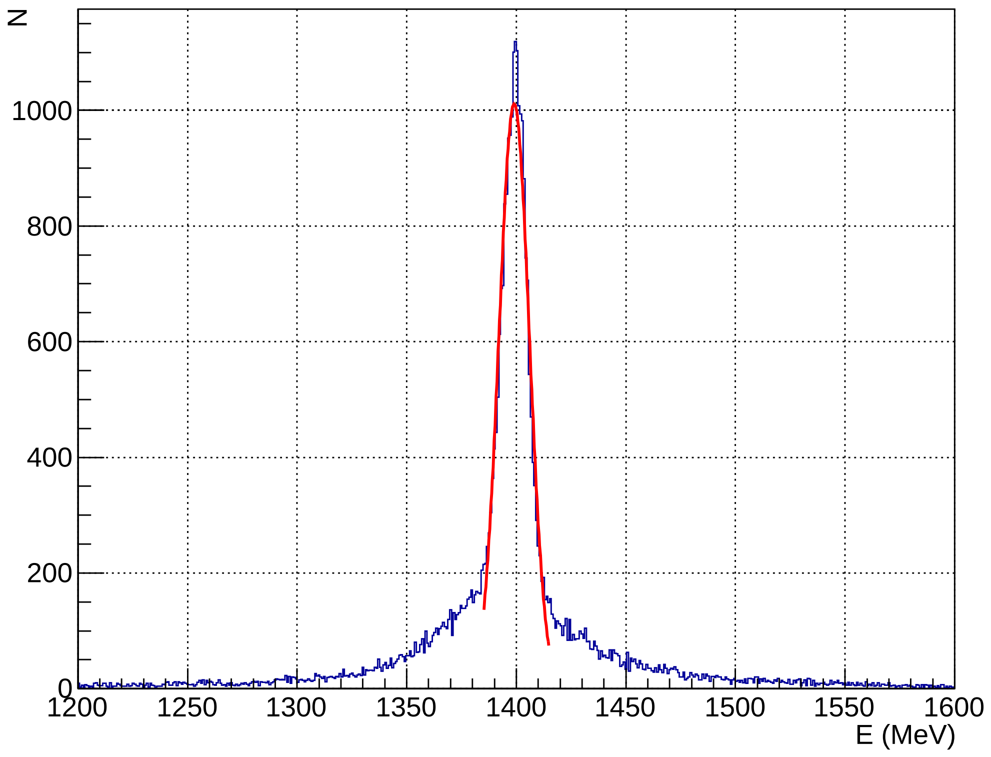

Figure 5.16: Total energy for the elastically scattered ^{40}Ca nucleus as determined using the new version of the code and the threaded mass estimator (For Silicon 15 and Caesium Iodide 48). A Gaussian fit to this peak results in the value (1399.0 ± 6.9)MeV. Note that this value is in close agreement with that of the old code: (1397.8 ± 39.5)MeV but has far better resolution.

however the beam energy is not recorded in the reconstructed experimental data and one must rely upon this single reported value which is given by reaction system rather than by run and may not remain constant throughout the runs of the reaction system.

In any case the fact that the new version of the identification code returns the same energy as the old, indicates that it is operating as expected in this regard. The improvement in the energy resolution of the new code appears to be the result of improvements made to the trajectory reconstruction algorithms by Guilain Ademard and Mark Boisjoli. To test this, the threaded mass estimator was swapped out for the old bisector in the new code (thereby running the new reconstruction routines with the old bisector) and the resolution remains the same for the elastic peak. This can be explained by the fact that for the elastic peak the majority of the calculated energy comes directly from the calibrated energy of the silicon and caesium iodide detectors and so does not rely heavily upon a correct estimation of the A value, remember this is used to calculate the energy loss in the dead layers and the drift chambers. It should be noted however that for lower mass fragments this may not be the case as the dead

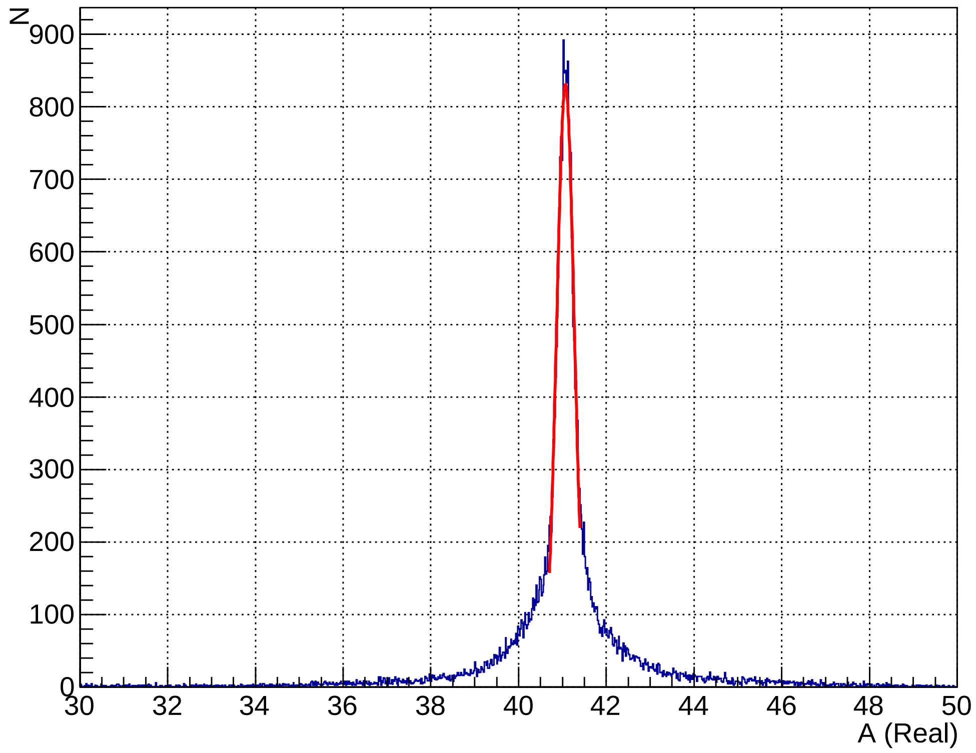

Figure 5.17: The real (as in floating point) A value returned by the VAMOS identification equations (For Silicon 15 and Caesium Iodide 48). A Gaussian fit to the peak results in the value: (41.1 ± 0.2). Note that this value is in close agreement with the old code: (40.7 ± 1.2) but once again the resolution is much improved.

layer and drift volume energy calculations will likely become increasingly significant.

The new and old versions of the code also return approximately the same value for the uncorrected real (as in floating point) A value ($A \sim 41$), note again that the resolution in A is much improved as a result of the improvement in the energy resolution. This high value for A (40 is expected) would seem to indicate that either the calculated energy is too high (which has been observed already) or that there is some issue in the time-of-flight or flight path calculations. This leaves two avenues of approach, either locate and account for the cause of this error or continue with this determination of the mass number and apply identification corrections.

The first method would seem preferable (Only $\sim 6 - 7\,\text{MeV}$ to be accounted for) however, in practice this is difficult as there are many potential sources for this error. For example, at the present time the pressure of the fill gas for the drift chambers and the ionisation chambers is a fixed constant (18 mbar for the drift chambers, 40 mbar for the ionisation chambers) which is potentially not physical. The pressures of these gasses are reported in the log books only sporadically (usually only if there is a problem) which makes the true run by run determination of the pressure difficult. This could perhaps be a potential contribution to the systematic error on the energy calculation observed for the elastic peak, as a drop in pressure would result in less energy being deposited by the incident fragment in the early stages of VAMOS. Or perhaps the error is simply due to variances in the energy calibrations of the detectors, a hint of this possibility is evident in Figure 5.18.

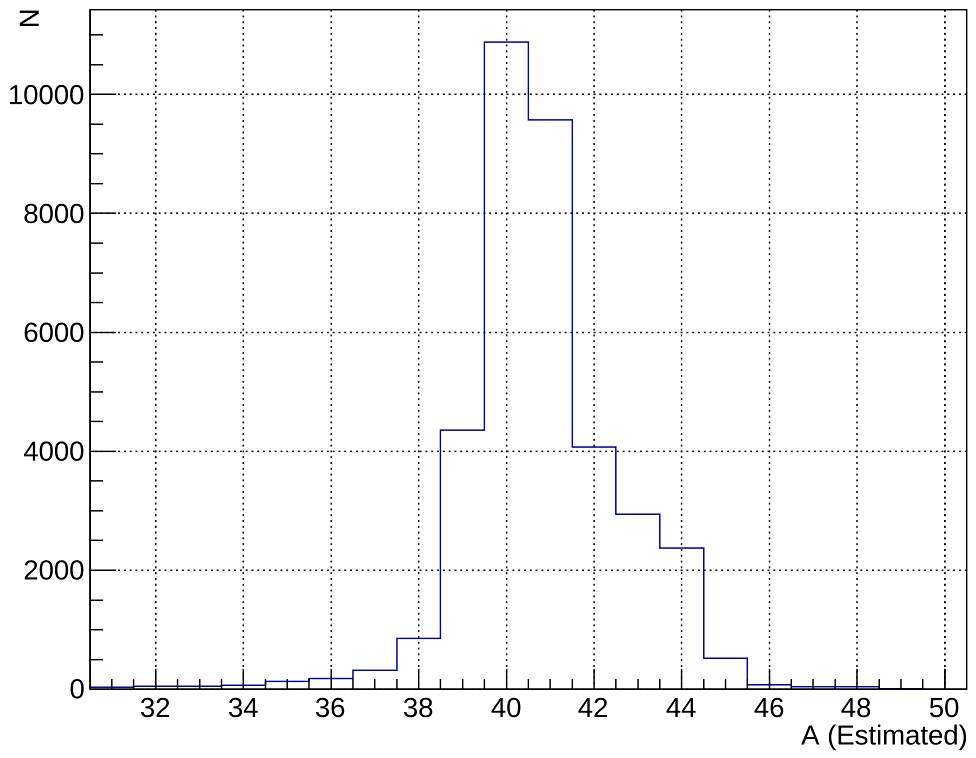

Figure 5.18: The integer A value returned by the threaded mass estimator for the elastically scattered ^{40}Ca nucleus (For Silicon 15 and Caesium Iodide 48). Note that while the peak content is in the expected place ($A = 40$) the weight of the distribution is shifted to higher A values. This might indicate a minor issue in the energy calibrations of the silicon or caesium iodide detectors, however the fact that the total energy and real A values agree with the old code value is what matters as far as the identification corrections are concerned.

In short, finding all of the contributing sources to this error and accounting for them in a way that is applicable to all of the detectors and across the entire range of fragment Z, A, Q and energy expected in the physics runs (i.e. not just for the elastic peak) is likely to be difficult. It is for the reason that the identification correction method is used in the old code, to correct the identification by using the fragments with a mass-to-charge ratio of two as a guide - they are easily identified and should form a straight line in the mass vs mass-to-charge plots.

Therefore, in order to properly identify the fragments in the new code the identification corrections must now be implemented in full. The fact that the new code produces the same uncorrected results (but with better resolution) should mean that the implementation of the identification corrections will be straightforward, although care will be needed to ensure that they are still applicable and work well with the new trajectory reconstruction algorithms. As stated earlier however there are still several trajectory reconstruction issues that need to be investigated before this can be done.

Chapter 6

Conclusion

In summary, due to a collaborative effort (including the author) almost all of the INDRA and VAMOS detectors have now been calibrated for the study of isoscaling and isospin diffusion measurements. The analysis code for VAMOS is still being heavily modified and is the last step before the final physics analysis can take place.

The old version of the identification code for the E503 experiment has been modified to ensure it is working as originally intended. There are still a few problems with the reconstruction algorithm in this old version of the code and it is believed that these are responsible for the poor energy and mass resolution. The improvements made to the old code have been implemented for the sake of verifying the operation of the new code which will eventually replace it.

The new version of the code includes drastic improvements to the old reconstruction methods (the work of Guilain Ademard and Mark Boisjoli) and as a result has much improved energy and mass resolution. The old identification procedure has been implemented in the new version of the code and this allows the implementation of the E503 identification corrections to be made with relative ease, at least in a programming sense, it remains to be seen if these corrections are still appropriate for the new code given the reconstruction improvements. The only correction currently implemented is that of the mass-to-charge ratio "straightening" function, as a proof of concept for the loading and utilisation of the correction parameters. Based purely on the fact that the energy determination between the two versions of the code is in close agreement, it is believed that the remaining corrections can be applied fairly quickly.

Aside from the identification corrections, the old bisection method which is used to estimate the A value of the fragment in VAMOS was tested and found to be failing under certain conditions. It has now been replaced with the threaded mass estimator which performs the same function in less time and in a considerably more robust way.

The new code has been developed under continuous testing and is a lot more stable and accurate than the old. If this remains true then the physics data that can be extracted should have excellent quality and if the event sorting is performed well, it should provide valuable constraints on the symmetry energy behaviour in the region of low density. It should also be noted that while the goal of the experiment is to understand the symmetry energy there is no reason why this data can not be used for other purposes in future.

Note: Should the source code be required in the same state as it exists *at the time of submission of this thesis*, it may be requested by email[1] and a CD can be sent via post. This will hopefully not be necessary as the code should exist within the main KaliVeda branch.

[1]peter.wigg.314159@gmail.com

Appendix A

Appendix

	CsI(Tl)						Si	Ionisation Chamber				
Ring No.	θ_{min} [deg]	θ_{max} [deg]	N	$\Delta\phi$ [deg]	e [mm]	$\Delta\Omega$ [msr]	e [μm]	$\Delta\phi$ [deg]	N	n CsI(Tl)	d [cm]	$\Delta\Omega$ [msr]
4	7	10	22	15	138	1.70	300	30	11	4	38.4	10.3
5	10	14	22	15	138	3.21	300					
6	14	20	24	15	97	7.01	150	30	12	4	25	37.7
7	20	27	24	15	97	11.2	150					
8	27	35	24	15	90	15.8	300	30	12	4	12	86.0
9	35	45	24	15	90	26.4	300					
10	45	57	24	15	76	39.6	none	30	12	4	12	183
11	57	70	24	15	76	50.3	none					
12	70	88	24	15	48	81.0	none	30	12	2	12	155
13	92	110	24	15	60	82.3	none	45	8	3	12	240
14	110	126	16	22.5	50	93.5	none	45	8	4	12	338
15	126	142	16	22.5	50	73.1	none					
16	142	157	8	45	50	91.2	none	45	8	2	12	144
17	157	176	8	45	50	50.9	none					

Table A.1: INDRA Configuration for the INDRA-VAMOS experiment (Modified reproduction from [55]). Note that the silicon detector thicknesses for rings 6 and 7 have been reduced to improve their resolution. N: denotes the number of detectors in each ring, e: the thickness of the detector, $\Delta\Omega$: the solid angle, n: the number of CsI(Tl) behind an Ionisation Chamber, d: the distance from the target, θ: polar angle, ϕ: azimuthal angle.

R \ M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
4	301	301	302	302	301	301	300	300	302	302	303	303	295	295	298	298	-	-	302	302	302	302	301	301
5	301	301	302	302	301	301	300	300	302	302	303	303	295	295	298	298	-	-	302	302	302	302	301	301
6	152	152	153	153	153	153	145	145	152	152	152	152	144	144	144	144	148	148	150	150	150	159	150	150
7	152	152	153	153	153	153	145	145	152	152	152	152	144	144	144	144	148	148	150	150	150	159	150	150
8	305	305	301	301	303	303	150	150	303	303	298	298	305	305	303	303	150	150	303	303	299	299	150	150
9	305	305	301	301	303	303	150	150	303	303	298	298	305	305	303	303	150	150	303	303	299	299	150	150

Table A.2: The INDRA silicon detector thicknesses in the E503 experiment (μm). In order to facilitate the coupling of the VAMOS spectrometer, rings 2-3 and modules 17-18 in rings 4-5 were removed. R denotes the ring number and M the module number.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
522	530	531	532	533	533	534	535	531	535	524	531	529	524	533	537	519	530

Table A.3: The thicknesses of the VAMOS silicon detectors (in μm).

VAMOS Formulae

In these equations c is always given as 29.9792458 cm/ns, all timing information is given in ns, all energies are in MeV, all masses are given as MeV/c^2 and the $B\rho$ value is always in Tm.

$$\frac{M}{Q} = \frac{10cB\rho}{\gamma\beta} \tag{A.1}$$

$$A = \frac{E}{(\gamma - 1)\, u} \tag{A.2}$$

Where $u = 931.494043$ MeV/c^2 (Taken from [69], not the most current value but the one used by KaliVeda).

Calculation of Total CsI Light (INDRA)

$$i(t) = \frac{Q_f}{\tau_f} \exp\left(-\frac{t}{\tau_f}\right) + \frac{Q_s}{\tau_s} \exp\left(-\frac{t}{\tau_s}\right) \tag{A.3}$$

$$i_{meas}(t) = \frac{Q_0}{\tau_0 - \tau}\left[\exp\left(-\frac{t}{\tau_0}\right) - \exp\left(-\frac{t}{\tau}\right)\right] \tag{A.4}$$

$$\begin{aligned}
Q_{meas} &= \int i_{meas}(t)\,\mathrm{d}t && \text{(A.5)}\\
&= \frac{Q_0}{\tau_0 - \tau}\left[\int \exp\left(-\frac{t}{\tau_0}\right)\,\mathrm{d}t - \int \exp\left(-\frac{t}{\tau}\right)\,\mathrm{d}t\right] && \text{(A.6)}\\
&= \frac{Q_0}{\tau_0 - \tau}\left[\tau\exp\left(-\frac{t}{\tau}\right) - \tau_0\exp\left(-\frac{t}{\tau_0}\right) + C\right] && \text{(A.7)}
\end{aligned}$$

$$Q_0 = \int_0^\infty i_{meas}(t)\,\mathrm{d}t \tag{A.8}$$

$$\begin{aligned}
\mathcal{F} &= \int_0^{t_1} i_{meas}(t)\,\mathrm{d}t && \text{(A.9)}\\
\mathcal{S} &= \int_{t_2}^{t_3} i_{meas}(t)\,\mathrm{d}t && \text{(A.10)}
\end{aligned}$$

$$\begin{aligned}
\mathcal{F} &= \frac{Q_0}{\tau_0 - \tau}\left[\tau\exp\left(-\frac{t}{\tau}\right) - \tau_0\exp\left(-\frac{t}{\tau_0}\right) + C\right]_0^{t_1} && \text{(A.11)}\\
&= \frac{Q_0}{\tau_0 - \tau}\left\{\tau_0\left[1 - \exp\left(-\frac{t_1}{\tau_0}\right)\right] - \tau\left[1 - \exp\left(-\frac{t_1}{\tau}\right)\right]\right\} && \text{(A.12)}
\end{aligned}$$

$$\begin{aligned}
\mathcal{S} &= \frac{Q_0}{\tau_0 - \tau}\left[\tau\exp\left(-\frac{t}{\tau}\right) - \tau_0\exp\left(-\frac{t}{\tau_0}\right) + C\right]_{t_2}^{t_3} && \text{(A.13)}\\
&= \frac{Q_0}{\tau_0 - \tau}\left\{\left[\tau\exp\left(-\frac{t_3}{\tau}\right) - \tau_0\exp\left(-\frac{t_3}{\tau_0}\right)\right] - \left[\tau\exp\left(-\frac{t_2}{\tau}\right) - \tau_0\exp\left(-\frac{t_2}{\tau_0}\right)\right]\right\} && \text{(A.14)}\\
&\approx \frac{Q_0}{\tau_0 - \tau}\left[\exp\left(-\frac{t_2}{\tau_0}\right) - \exp\left(-\frac{t_3}{\tau_0}\right)\right] \qquad (\tau_0 \gg \tau) && \text{(A.15)}
\end{aligned}$$

Appendix B

Source Code

B.1 ThreadedMassEstimator Header

```
#ifndef __THREADED_MASS_ESTIMATOR__
#define __THREADED_MASS_ESTIMATOR__

#undef NDEBUG
//#define NDEBUG

// C
#include <cassert>

// CINT can't parse pthread.h
#ifndef __CINT__
#include "pthread.h"
#else
struct pthread_mutex_t;
#endif

// KaliVeda (VAMOS)
#include "MEDetectorStack.h"

namespace e503
{

struct EstimatorInput {
    Int_t z;
    Double_t si_energy;
    Double_t csi_light;
};

struct ThreadStatus {
    static const Int_t kThreadStop;
    static const Int_t kThreadContinue;
};

struct ThreadData {

    // Having a local copy of the possible A values removes the need for high
    // frequency thread locking as we need to access this vector every iteration
    // of each thread. We also get a speed increase as the iterators can simply
    // increment/decrement without jumping around  (as we iterate in different
    // directions in each thread)

    std::vector<Int_t> possible_a_values;

    // Are we iterating forwards?
    Bool_t kForwardIterator;

    // Experimental input data
    Int_t    z_value_expt;
    Double_t si_energy_expt;
    Double_t csi_light_expt;

    // Thread Storage
    Int_t    current_a;
    Double_t current_delta;
    Int_t    minimum_a;
    Double_t minimum_delta;

    // Thread output data
    struct SimulationResult sim_result;

    Int_t counter;
};

struct EstimatorResult {
    Int_t    z_value;
    Int_t    a_value;
    Double_t delta;
    Int_t    forward_counter;
    Int_t    backward_counter;
    Int_t    status;
};

class ThreadedMassEstimator
{
    // Detector stack used to simulate particle passage
    static MEDetectorStack *medetector_stack_;

    static pthread_mutex_t io_mutex_;
    static pthread_mutex_t medetector_mutex_;

    // Thread processing methods
    static void* ThreadExec(void *data);
    static Int_t ThreadProcess(
        struct ThreadData *const data,
        const struct SimulationParameters *const parameters
    );

    Bool_t kInitialised_;
    Float_t load_balance_;

    // Prohibit Copy and Copy Assign
    ThreadedMassEstimator(const ThreadedMassEstimator&);
    ThreadedMassEstimator& operator=(const ThreadedMassEstimator&);

  public:

    ThreadedMassEstimator();
    virtual ~ThreadedMassEstimator();

    Bool_t Init();
    Bool_t IsInitialised() const;

    Bool_t SetIDTelescope(const Char_t *name);
    Bool_t GetPossibleAValues(const Int_t z_value, std::vector<Int_t> *results);

    Bool_t EstimateA(
        const struct EstimatorInput *const input,
        const std::vector<Int_t> *const possible_a_values,
        struct EstimatorResult *const result
    );

    // Accessors
    Float_t get_load_balance() const;

    // Mutators
    void set_load_balance(Float_t load_balance);

    // Negative results indicate error, Positive provide information, 0 should
    // never occur (used as an initialisation value)
    enum ResultStatus {
        kCalibrationRangeError  = -2,
        kNoResultFound          = -1,
        kEqualDeltas            =  1,
        kThreadOneResult        =  2,
        kThreadZeroResult       =  3
    };


    ClassDef(ThreadedMassEstimator, 1); // ThreadedMassEstimator

};

} // e503 namespace

#undef NDEBUG
#endif
```

Listing B.1: ThreadedMassEstimator.h

B.2 ThreadedMassEstimator Source

```
#include "ThreadedMassEstimator.h"

ClassImp(e503::ThreadedMassEstimator)

////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// ThreadedMassEstimator                                                      //
//                                                                            //
// This class estimates the A value of a fragment detected in the            //
// Si:Isobutane:CsI detector stack, given the experimentally measured data    //
// (Z, Silicon Energy, CsI Light).                                            //
//                                                                            //
// It achieves this by minimising the absolute magnitude of the difference    //
// between the experimentally measured silicon energy and the silicon energy  //
// simulated in a set of test nuclei (with fixed Z to determine which nucleus //
// (A value) provides the closest match)                                      //
//                                                                            //
// Estimation will frequently fail at low csi light (approx. < chn #200) as   //
// in this region the bands of the various isotopes for a given Z converge    //
// and simulation errors and approximations will become increasingly          //
// signficant. I've not looked into it in any great detail but the failure    //
// rate seems to decay in an almost exponential way (it's not exponential     //
// though - just a rule of thumb) with the csi light (by approximately chn    //
// #200 there are virtually no failures).                                     //
//                                                                            //
// Author: Peter Wigg <mailto:peter.wigg.314159@gmail.com>                    //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////

namespace e503
{

// Static definitions

const Int_t ThreadStatus::kThreadStop(0);
const Int_t ThreadStatus::kThreadContinue(-1);

// Detector stack
MEDetectorStack* ThreadedMassEstimator::medetector_stack_(
    new MEDetectorStack()
);

pthread_mutex_t ThreadedMassEstimator::io_mutex_;
pthread_mutex_t ThreadedMassEstimator::medetector_mutex_;

// Constructor
```

```
ThreadedMassEstimator::ThreadedMassEstimator() :
    kInitialised_(kFALSE),
    load_balance_(2.25)
{
    // Constructor (Public Interface Method)
    //
    // It is good practice to keep the constructor clear of initialisation code
    // as there is no easy way for constructors to handle errors. Instead you
    // should call Init() immediately after you construct this object.
}

ThreadedMassEstimator::~ThreadedMassEstimator()
{
    // Destructor (Public Interface Method)

    pthread_mutex_destroy(&io_mutex_);
    pthread_mutex_destroy(&medetector_mutex_);

    if (medetector_stack_) {
        delete medetector_stack_;
        medetector_stack_ = NULL;
    }
}

Bool_t ThreadedMassEstimator::Init()
{
    // Init (Public Interface Method)
    //
    // You should call this method immediately after you have constructed the
    // ThreadedMassEstimator object. If the object is already initialised then
    // this method will do nothing
    //

    if (kInitialised_) {
        return kTRUE;
    }

    if (!medetector_stack_->Init()) {
        return kFALSE;
    }

    pthread_mutex_init(&io_mutex_, NULL);
    pthread_mutex_init(&medetector_mutex_, NULL);

    kInitialised_ = kTRUE;

    return kTRUE;
}

Bool_t ThreadedMassEstimator::IsInitialised() const
{
    return kInitialised_;
}

Bool_t ThreadedMassEstimator::SetIDTelescope(const Char_t *name)
{
    // SetIDTelescope (Public Interface Method)
    //
    // Not thread-safe.

    if (!medetector_stack_->SetIDTelescope(name)) {
        return kFALSE;
    }

    return kTRUE;
}

Bool_t ThreadedMassEstimator::GetPossibleAValues(
    const Int_t z_value,
    std::vector<Int_t> *results
```

```
)
{
    // GetPossibleAValues (Public Interface Method)
    //
    // This method populates a vector specified by the user ('results') with a
    // sorted list of possible A values for a given Z. We define a possible
    // nucleus as any nucleus (KVNucleus) having a lifetime greater than zero.
    //
    // Not thread safe.

    assert(results);

    // Destroy all the objects currently in result
    results->clear();

    for (Int_t a_value = 1; a_value < 240; ++a_value) {
        KVNucleus nucleus(z_value, a_value);
        if (nucleus.GetLifeTime() > 0.) {
            results->push_back(a_value);
        }
    }

    return kTRUE;
}

void* ThreadedMassEstimator::ThreadExec(void *thread_data)
{
    // Thread Execution Method (Called when each thread is run)

    //pthread_t self_id = pthread_self();

    struct ThreadData *data(
        static_cast<struct ThreadData*>(thread_data)
    );

    assert(data);

    Bool_t process_enable(kFALSE);

    struct SimulationParameters parameters;
    parameters.z = data->z_value_expt;
    parameters.si_energy = data->si_energy_expt;
    parameters.csi_light = data->csi_light_expt;

    Int_t status(0);
    data->counter = 0;

    if (data->kForwardIterator == kTRUE) {

        // Iterate forwards through the list of possible A values.

        std::vector<Int_t>::iterator itr;
        for (itr = data->possible_a_values.begin();
             itr != data->possible_a_values.end();
             ++itr
            ) {

            if (process_enable) {

                parameters.a = *itr;
                data->current_a = *itr;

                // Process this point
                status = ThreadProcess(data, &parameters);

                // If we are maximising or something went wrong we return
                if (status == ThreadStatus::kThreadStop) {
                    break;
                }

            }

            // Do nothing until we reach the pre-defined starting value
            if (*itr == data->minimum_a) {
                // Remove the inhibitor switch for the next loop
                process_enable = kTRUE;
            }

        }

    } else {

        // Iterate backwards through the list of possible A values.

        std::vector<Int_t>::reverse_iterator ritr;
        for (ritr = data->possible_a_values.rbegin();
             ritr != data->possible_a_values.rend();
             ++ritr
            ) {

            if (process_enable) {

                parameters.a = *ritr;
                data->current_a = *ritr;

                // Process this point
                status = ThreadProcess(data, &parameters);

                if (status == ThreadStatus::kThreadStop) {
                    break;
                }

            }

            // Do nothing until we reach the pre-defined starting value
            if (*ritr == data->minimum_a) {
                // Remove the inhibitor switch for the next loop
                process_enable = kTRUE;
            }


        }

    }

    pthread_exit(NULL);
}

Int_t ThreadedMassEstimator::ThreadProcess(
    struct ThreadData *const data,
    const struct SimulationParameters *const parameters)
{
    // Thread Processing Method (Called from ThreadExec). This method takes care
    // of calculating the delta values and determining the state of the
    // minimisation.  The return value determines whether the current thread
    // will stop or continue processing.

    //pthread_t self_id = pthread_self();

    // We have to lock the detector stack with a mutex when performing the
    // calculation as the energy loss functions are not thread safe - there are
    // several static pointers used.

    ++data->counter;

    pthread_mutex_lock(&medetector_mutex_);
    if (!medetector_stack_->Simulate(parameters, &(data->sim_result))) {
        data->current_delta = -1.;
        pthread_mutex_unlock(&medetector_mutex_);
        return ThreadStatus::kThreadStop;

    }

    data->current_delta = data->sim_result.delta;
    pthread_mutex_unlock(&medetector_mutex_);

    if (data->current_delta < 0.) {

        // This should be impossible but we might as well check

        pthread_mutex_lock(&io_mutex_);
        Error("ThreadedMassEstimator::ThreadProcess",
              "Negative value for current delta!");
        pthread_mutex_unlock(&io_mutex_);

        return ThreadStatus::kThreadStop;
    }

    if (data->minimum_delta < 0.) {

        // This should be impossible but we might as well check

        pthread_mutex_lock(&io_mutex_);
        Error("ThreadedMassEstimator::ThreadProcess",
              "Negative value for minimum delta!");
        pthread_mutex_unlock(&io_mutex_);

        return ThreadStatus::kThreadStop;
    }

    if (data->current_delta < data->minimum_delta) {
        // The current point (A value, Delta) yields a better result than the
        // stored point (a lower value for delta). We therefore overwrite the
        // stored data with the current values;

        data->minimum_delta = data->current_delta;
        data->minimum_a = data->current_a;

        return ThreadStatus::kThreadContinue;

    } else if (data->current_delta > data->minimum_delta) {
        // We are maximising delta which means we are moving away from the
        // minimum point we are searching for so we stop the processing.

        return ThreadStatus::kThreadStop;

    } else {
        // Deltas are equal, this means we cannot be sure we have the correct A
        // value (the current minimum value for A will be used)

        return ThreadStatus::kThreadStop;
    }

}

Bool_t ThreadedMassEstimator::EstimateA(
    const struct EstimatorInput *const input,
    const std::vector<Int_t> *const possible_a_values,
    struct EstimatorResult *const result
)
{
    // EstimateA (Public Interface method)
    //
    // This method calculates an estimate of the A value from a given set of
    // input data (Z, Silicon Energy, CsI Light). The result is written into
    // the address supplied by the user - \'result\' (We store some extra data,
    // for example the CsI Energy for that simulated point)
    //
    // THE RETURN VALUE IS NOT THE VALUE OF A! This function returns a boolean
    // value to indicate success or failure.
    //

    assert(input);
    assert((input->si_energy > 0.) && (input->si_energy < 3000.));

    assert(possible_a_values);
    assert(result);

    if (!input) {
        Error("ThreadedMassEstimator::EstimateA", "Input data is NULL");
        return kFALSE;
    }

    // You should never see this initialisation value in the output, it should
    // always be set to either a positive or a negative number below:
    result->status = 0;

    Bool_t calibration_range_error(kFALSE);

    if (input->si_energy > 1000.) {

        // IMPORTANT:
        //
        // Currently the CsI light to energy calibration function is only valid
        // up to around A=45 and we must warn the user that they are trying to
        // analyse a point which is likely outside of that range. Unfortunately,
        // there are only 3 variables available to us at this time: silicon
        // energy, caesium iodide light output and Z. I have chosen the silicon
        // energy and decided to specifiy the cut-off as 1000 MeV (48-Ca should
        // be around 800-900 MeV). If the CsI is ever re-calibrated to extend
        // this range, then this cut-off value should be increased accordingly.
        //
        // We process this event as normal but the results are extrememly
        // unreliable outside of the CsI calibration range. The
        // calibration_range_error flag is used to set the correct return status
        // after the event has been processed. This way we let the mass
        // estimator do its job but inform the user to be wary of the result.
        //

        calibration_range_error = kTRUE;
    }

    static Long64_t event_number(0);
    ++event_number;

    struct SimulationParameters parameters;
    parameters.z = input->z;

    // Initial simulation for the starting A value (common to both threads)
    //
    // IMPORTANT: The initial A value must be set at approximately the right
    // value (approximately 2*Z) in order to reduce the time taken for its
    // determination. However an additional linear scaling is needed to maintain
    // the balance of processing power between the "forward" thread and the
    // "backward" thread. If this balance is not maintained then all of the
    // processing is done in one thread and you lose the benefit of using two.
    // Empirically, the default scaling factor of 2.25 is sufficient over the
    // range of z values we expect in the E503 experiment.

    parameters.a = static_cast<Int_t>(load_balance_ * input->z);
    parameters.si_energy = input->si_energy;
    parameters.csi_light = input->csi_light;

    // Get the "delta" - the magnitude difference between the real silicon
    // energy and that simulated for our A = load_balance_*Z nucleus.
    // Single thread at this point, no need to lock with mutex
    Double_t delta_init(-1.);

    struct SimulationResult res;
    if (!medetector_stack_->Simulate(&parameters, &res)) {
        return kFALSE;
    }

    delta_init = res.delta;

    // Pack the input data into structs
    struct ThreadData data[2];

    for (Int_t i = 0; i < 2; ++i) {
        data[i].z_value_expt = input->z;
        data[i].si_energy_expt = input->si_energy;
        data[i].csi_light_expt = input->csi_light;

        // Give each thread a local copy of the possible A values
        data[i].possible_a_values = *possible_a_values;

        // Set the (A, delta) as the starting point for both threads
        data[i].minimum_a = parameters.a;
        data[i].minimum_delta = delta_init;

    }

    // Iterating forwards (Increasing A from initial value)
    data[0].kForwardIterator = kTRUE;

    // Iterating backwards (Decreasing A from initial value)
    data[1].kForwardIterator = kFALSE;

    // Create the threads (one for each direction - increasing/decreasing A)

    pthread_t thr[2];
    Int_t status(0);

    /////////////////
    // Create Thread 0
    /////////////////

    status = pthread_create(
                 &thr[0],
                 NULL,
                 ThreadExec,
                 static_cast<void*>(&data[0])
             );

    if (status != 0) {
        Error("ThreadedMassEstimator::EstimateA", "Failed to create thread 0");
        return kFALSE;
    }

    /////////////////
    // Create Thread 1
    /////////////////

    status = pthread_create(
                 &thr[1],
                 NULL,
                 ThreadExec,
                 static_cast<void*>(&data[1])
             );

    if (status != 0) {
        Error("ThreadedMassEstimator::EstimateA", "Failed to create thread 1");
        return kFALSE;
    }

    ////////////////
    // Join Thread 0
    ////////////////

    Int_t thread_return(0);

    status = pthread_join(
                 thr[0],
                 reinterpret_cast<void**>(&thread_return)
             );

    if (status != 0) {
        Error(
            "ThreadedMassEstimator::EstimateA",
            Form("Failed to join Thread 0 (status = %d)", thread_return)
        );
        return kFALSE;
    }

    ////////////////
    // Join Thread 1
    ////////////////

    status = pthread_join(
                 thr[1],
                 reinterpret_cast<void**>(&thread_return)
             );

    if (status != 0) {
        Error(
            "ThreadedMassEstimator::EstimateA",
            Form("Failed to join Thread 1 (status = %d)", thread_return)
        );
        return kFALSE;
    }

    // We now compare the results of the two threads to see which one has the
    // best value for a_min

    Int_t a_min(-1);
    Double_t delta(-1.);
    Int_t result_status(0);

    Bool_t thread_ok[2];
    for (Int_t i = 0; i < 2; ++i) thread_ok[i] = kTRUE;

    if (data[0].minimum_delta < 0.) thread_ok[0] = kFALSE;
    if (data[1].minimum_delta < 0.) thread_ok[1] = kFALSE;

    if (thread_ok[0] && thread_ok[1]) {

        if (data[0].minimum_delta < data[1].minimum_delta) {
            a_min = data[0].minimum_a;
            delta = data[0].minimum_delta;
            result_status = kThreadZeroResult;

        } else if (data[1].minimum_delta < data[0].minimum_delta) {
            a_min = data[1].minimum_a;
            delta = data[1].minimum_delta;
            result_status = kThreadOneResult;

        } else {
            // Equal Deltas: This usually happens when the initial A value
            // provided the minimum delta. This is because both threads store
            // this point as their initial minimum reference)

            a_min = data[0].minimum_a;
            delta = data[0].minimum_delta;
            result_status = kEqualDeltas;
        }

    } else if (thread_ok[0] && !thread_ok[1]) {
        a_min = data[0].minimum_a;
        delta = data[0].minimum_delta;
        result_status = kThreadZeroResult;

    } else if (!thread_ok[0] && thread_ok[1]) {
        a_min = data[1].minimum_a;
        delta = data[1].minimum_delta;
        result_status = kThreadOneResult;

    } else {

        // IMPORTANT: Both threads failed to return a sensible result, you need
        // to handle this in your analysis class!

        result_status = kNoResultFound;

        result->z_value = input->z;
        result->a_value = -1;
        result->delta = -1.;
        result->forward_counter = data[0].counter;
        result->backward_counter = data[1].counter;
        result->status = result_status;

        return kFALSE;
    }

    if (calibration_range_error) {
        result_status = kCalibrationRangeError;
    }

    result->z_value = input->z;
    result->a_value = a_min;
    result->delta = delta;
    result->forward_counter = data[0].counter;
    result->backward_counter = data[1].counter;
    result->status = result_status;

    return kTRUE;
}

Float_t ThreadedMassEstimator::get_load_balance() const
{
    return load_balance_;
}

void ThreadedMassEstimator::set_load_balance(Float_t load_balance)
{
    load_balance_ = load_balance;
}

} // e503 namespace
```

Listing B.2: ThreadedMassEstimator.cpp

B.3 MEDetectorStack Header

```
#ifndef __MEDETECTORSTACK_H__
#define __MEDETECTORSTACK_H__

#undef NDEBUG
//#define NDEBUG

// C
#include <cassert>

// C++
#include <stdexcept>
#include <vector>

// ROOT
#include "Rtypes.h"
#include "TError.h"
```

```
#include "TMath.h"

// KaliVeda (Standard)
#include "KVCalibrator.h"
#include "KVDetector.h"
#include "KVIDTelescope.h"
#include "KVList.h"
#include "KVNucleus.h"
#include "KVTelescope.h"
#include "KVUnits.h"

// KaliVeda (VAMOS)
#include "KVLightEnergyCsIVamos.h"
#include "KVVAMOS.h"

namespace e503
{

struct SimulationParameters {
    Int_t    z;          // Measured Z
    Int_t    a;          // Test value for A
    Double_t si_energy; // Measured Si Energy
    Double_t csi_light; // Measure CsI Light
};

struct SimulationResult {
    Int_t    z;           // Measured Z
    Int_t    a;           // Test A value used
    Double_t si_energy;  // Measured Si Energy
    Double_t gap_energy; // Simulated Gap Energy
    Double_t csi_energy; // Simulated CsI Energy
    Double_t delta;      // Mag. Diff. Si Energy (Sim, Expt)
};

class MEDetectorStack
{
    KVTelescope           *idtelescope_;
    KVLightEnergyCsIVamos *calibrator_;
    KVNucleus             *sim_nucleus_;

    Bool_t kInitialised_;

    // Prohibit Copy and Copy Assign
    MEDetectorStack(const MEDetectorStack&);
    MEDetectorStack& operator=(const MEDetectorStack&);

  public:

    MEDetectorStack();
    virtual ~MEDetectorStack();

    Bool_t Init();
    Bool_t IsInitialised() const;

    Bool_t Simulate(
        const struct SimulationParameters *const parameters,
        struct SimulationResult *const result
    );

    Bool_t SetIDTelescope(const Char_t *name);

    // Accessors
    const KVTelescope*           get_idtelescope() const;
    const KVLightEnergyCsIVamos* get_calibrator()  const;
    const KVNucleus*             get_sim_nucleus() const;

    ClassDef(MEDetectorStack, 1) // MEDetectorStack

};

} // e503 namespace

#undef NDEBUG
#endif
```

Listing B.3: MEDetectorStack.h

B.4 MEDetectorStack Source

```
#include "MEDetectorStack.h"

ClassImp(e503::MEDetectorStack)

////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// MEDetectorStack                                                            //
//                                                                            //
// Si:Isobutane:CsI detector stack used to calculate the absolute difference  //
// between the experimentally measured silicon energy (VAMOS) and that of a   //
// simulated test particle with estimated A value.                            //
//                                                                            //
// Author: Peter Wigg <mailto:peter.wigg.314159@gmail.com>                    //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////

namespace e503
{

MEDetectorStack::MEDetectorStack() :
    idtelescope_(NULL),
    calibrator_(NULL),
    sim_nucleus_(NULL),
    kInitialised_(kFALSE)
{
    // Constructor (Public Interface Method)
    //
    // It is good practice to keep the constructor clear of initialisation code
    // as there is no easy way for constructors to signal errors. Instead you
    // should call Init() immediately after you construct this object.
}

MEDetectorStack::~MEDetectorStack()
{
    // Destructor (Public Interface Method)

    if (idtelescope_) {
        delete idtelescope_;
        idtelescope_ = NULL;
    }

    if (calibrator_) {
        delete calibrator_;
        calibrator_ = NULL;
    }

    if (sim_nucleus_) {
        delete sim_nucleus_;
        sim_nucleus_ = NULL;
    }

}

Bool_t MEDetectorStack::Init()
{
    // Init (Public Interface Method)
    //
    // You should call this method immediately after you have constructed the
    // MEDetectorStack object. If this object is already initialised then this
    // method will do nothing.

    if (kInitialised_) {
        return kTRUE;
    }

    sim_nucleus_ = new KVNucleus();

    // Build default telescope
    idtelescope_ = new KVTelescope();

    KVDetector *si(new KVDetector("Si", 530 * KVUnits::um));
    KVDetector *isobutane(new KVDetector("C4H10", 136.5 * KVUnits::mm));
    isobutane->SetPressure(40.*KVUnits::mbar);
    KVDetector *csi(new KVDetector("CsI", 1.*KVUnits::cm));

    idtelescope_->Add(si);
    idtelescope_->Add(isobutane);
    idtelescope_->Add(csi);

    calibrator_ = new KVLightEnergyCsIVamos();

    kInitialised_ = kTRUE;

    return kTRUE;
}

Bool_t MEDetectorStack::Simulate(
    const struct SimulationParameters *const parameters,
    struct SimulationResult *const result
)
{
    // Simulate (Public Interface Method)
    //
    // This method calculates the magnitude difference between the
    // experimentally measured silicon energy and that simulated for a given
    // test nucleus. The test nucleus takes the measured value of Z but must use
    // an estimate of the A value (which has not been measured).
    //
    // The results are written into the address provided by 'result'.
    //
    // Not thread-safe (You need to wrap each call with a mutex lock). Returns
    // kTRUE or kFALSE as a means of determining whether the simulation was
    // successful or not.

    assert(kInitialised_);

    assert(parameters);
    assert(result);

    assert((parameters->z > 0) && (parameters->z < 120));
    assert((parameters->a > 0) && (parameters->a < 300));
    assert((parameters->si_energy > 0.) && (parameters->si_energy < 3000.));

    // We now need to calculate the incident energy of the nucleus (so that we
    // can simulate its passage through the idtelescope).

    // We have the csi_light, so let's work out the csi_energy first using the
    // light->energy calibration function defined for this caesium iodide
    // detector.

    if (!calibrator_) return kFALSE;
    if (!calibrator_->GetStatus()) return kFALSE;

    Double_t csi_energy(
        calibrator_->Compute(
            parameters->z,
            parameters->a,
            parameters->csi_light
        )
    );
```

```cpp

    //assert((csi_energy > 0.) && (csi_energy < 3000.));

    if ((csi_energy <= 0.) || (csi_energy > 3000.)) {
        // Out of bounds csi energy, discard this event. For some reason there
        // are occasionally energies of 0. MeV returned for non-zero csi light,
        // I'm not sure whether that is a fault in the code or if it is the
        // default return value when TF1::GetX() fails. To see these events
        // enable the assertions and uncomment the above assertion.

        return kFALSE;
    }

    // We then calculate the incident energy for the isobutane layer, which
    // resides behind the silicon layer and in front of the caesium iodide. We
    // calculate this by finding the residual energy just after the silicon
    // layer (using the energy measured in the silicon).

    KVDetector *silicon(idtelescope_->GetDetector(1));
    assert(silicon);

    Double_t isobutane_incident_energy(
        silicon->GetEResFromDeltaE(
            parameters->z,
            parameters->a,
            parameters->si_energy
        )
    );

    if (isobutane_incident_energy <= 0.) {
        // Particle does not punch through the silicon material and so we can
        // not identify it in the SI-CSI telescope.

        return kFALSE;
    }

    if (isobutane_incident_energy > 3000.) {
        // This generally happens when the silicon energy (from the silicon
        // calibration) is too low, this results in an over-estimation of the
        // isobutane incident energy. This event is rejected as this energy loss
        // is far greater than that of the available beam energy (realistically
        // we expect less than 2000 MeV)

        return kFALSE;
    }

    KVDetector *isobutane(idtelescope_->GetDetector(2));
    assert(isobutane);

    Double_t isobutane_energy(
        isobutane->GetDeltaE(
            parameters->z,
            parameters->a,
            isobutane_incident_energy
        )
    );

    assert((isobutane_energy > 0.) && (isobutane_energy < 3000.));

    // We now have all the data we need to calculate the incident energy of the
    // nucleus prior to entering the silicon layer

    Double_t incident_energy(
        parameters->si_energy +
        isobutane_energy +
        csi_energy
    );

    // Now we can simulate the passage of the nucleus through the idtelescope
    // and calculate 'delta' (the absolute difference between the measured
    // silicon energy and that simulated for this nucleus)

    assert((incident_energy > 0.) && (incident_energy < 3000.));

    sim_nucleus_->SetZAandE(parameters->z, parameters->a, incident_energy);
    idtelescope_->DetectParticle(sim_nucleus_);

    Double_t delta(
        TMath::Abs(parameters->si_energy - silicon->GetEnergy())
    );

    // Pack the result struct to contain delta and the gap and csi_energies
    // which have been calculated.

    result->z = parameters->z;
    result->a = parameters->a;
    result->si_energy = silicon->GetEnergy();
    result->gap_energy = isobutane_energy;
    result->csi_energy = csi_energy;
    result->delta = delta;

    idtelescope_->ResetDetectors();

    return kTRUE;
}

Bool_t MEDetectorStack::SetIDTelescope(const Char_t *name)
{
    // SetIDTelescope (Public Interface Method)
    //
    // Build a more realistic Si:CsI id telescope from the basic telescope named
    // above.  This telescope can be used for simulating particle passage as it
    // includes the isobutane dead layer between the silicon and caesium iodide.
    //
    // Not thread-safe.

    assert(kInitialised_);

    KVIDTelescope *idt(gVamos->GetIDTelescope(name));

    if (!idt) return kFALSE;
    if (!idt->InheritsFrom("e503::KVIDHarpeeSiCsI_e503")) return kFALSE;

    // Set the correct material thicknesses etc.
    KVDetector *si(idt->GetDetector(1));
    KVDetector *our_si(idtelescope_->GetDetector(1));

    assert(si);
    assert(our_si);
    our_si->SetThickness(si->GetThickness());

    // XXX: Warning: Isobutane pressure is kept static at the moment (40mbar)

    KVDetector *csi(idt->GetDetector(2));
    assert(csi);

    const Char_t *calibrator_name(
        Form("Light->MeV %s", csi->GetName())
    );

    KVLightEnergyCsIVamos *cal(
        dynamic_cast<KVLightEnergyCsIVamos*>(
            csi->GetCalibrator(calibrator_name)
        )
    );

    if (!cal) return kFALSE;
    if (!cal->GetStatus()) return kFALSE;

    calibrator_ = cal;
```

```cpp

    return kTRUE;
}

Bool_t MEDetectorStack::IsInitialised() const
{
    return kInitialised_;
}

const KVTelescope* MEDetectorStack::get_idtelescope() const
{
    return idtelescope_;
}

const KVLightEnergyCsIVamos* MEDetectorStack::get_calibrator() const
{
    return calibrator_;
}

const KVNucleus* MEDetectorStack::get_sim_nucleus() const
{
    return sim_nucleus_;
}

} // e503 namespace
```

Listing B.4: MEDetectorStack.cpp

B.5 IdentifyE503 Header

```cpp
#ifndef __IDENTIFY_E503_H__
#define __IDENTIFY_E503_H__

#undef NDEBUG
//#define NDEBUG

// C++
#include <cassert>

// ROOT
#include "TVector3.h"

// KaliVeda (Standard)
#include "KVINDRA.h"
#include "KVMaterial.h"
#include "KVTarget.h"

// KaliVeda (VAMOS)
#include "KVVAMOSReconNuc.h"
#include "KVIDHarpeeSiCsI_e503.h"
#include "KVIDHarpeeICSi_e503.h"

#include "StageEnergies.h"
#include "CorrectionData.h"

namespace e503
{

class IdentifyE503
{
    Bool_t kInitialised_;
    StageEnergies energy_data_;

    // Prohibit Copy and Copy Assign
    IdentifyE503(const IdentifyE503&);
    IdentifyE503& operator=(const IdentifyE503&);

    // TODO: Make non-member non-friend?
    Bool_t ApplyCorrections(
        KVVAMOSReconNuc *const n,
        const CorrectionData *const data
    );

    Bool_t ApplyIcSiCorrections(
        KVVAMOSReconNuc *const n,
        const KVIDHarpeeICSi_e503 *const idt,
        const CorrectionData *const data
    );

    Bool_t ApplySiCsiCorrections(
        KVVAMOSReconNuc *const n,
        const KVIDHarpeeSiCsI_e503 *const idt,
        const CorrectionData *const data
    );

    Double_t CorrectAoverQ(
        KVDBParameterSet *const,
        Double_t uncorrected,
        Double_t pid
    );

  public:

    IdentifyE503();
    virtual ~IdentifyE503();

    Bool_t Init();
    UChar_t Identify(KVVAMOSReconNuc *const n);

    // Accessors
    const StageEnergies& get_energy_data() const;

    enum ReturnCodesID {
        kAllOK = 0,
        kBadTimeCal,            //  1
        kBasicIdNotOk,          //  2
        kBothInherited,         //  3
        kICNotInherited,        //  4
        kICSiNullParlist,       //  5
        kInvalidIdCode,         //  6
        kNoIdResult,            //  7
        kNoIdTelescope,         //  8
        kNoTarget,              //  9
        kNotCalibrated,         //  10
        kNotIdentified,         //  11
        kNotInherited,          //  12
        kNotStoppingDetector,   //  13
        kNullParset,            //  14
        kResidualEnergyFail,    //  15
        kSegmentZero,           //  16
        kSiCsINullParlist,      //  17
        kUnCharged,             //  18
        kUnChargedResult,       //  19
        kZIdNotOk,              //  20
        kCorrectionsFailed
    };

    ClassDef(IdentifyE503, 1);

};

// Non-member, Non-Friend Functions

} // namespace e503

#undef NDEBUG
#endif
```

Listing B.5: IdentifyE503.h

B.6 IdentifyE503 Source

```
#include "IdentifyE503.h"

ClassImp(e503::IdentifyE503)

namespace e503
{

////////////////////////////////////////////////////////////////////////////////
//                                                                            //
// e503::IdentifyE503                                                         //
//                                                                            //
// This class is designed as a post-processing addition to the               //
// KVVAMOSReconNuc::Identify() method and its purpose is to calculate the    //
// mass-to-charge ratio, the mass and the charge of the fragments in the     //
// VAMOS multi-detector - it also applies the identification corrections for //
// the E503 experiment.                                                       //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////

IdentifyE503::IdentifyE503() :
    kInitialised_(kFALSE)
{

}

IdentifyE503::~IdentifyE503()
{

}

Bool_t IdentifyE503::Init()
{
    if (kInitialised_) return kTRUE;

    kInitialised_ = kTRUE;
    return kTRUE;
}

Double_t IdentifyE503::CorrectAoverQ(
    KVDBParameterSet *const parameters,
    Double_t uncorrected,
    Double_t pid
)
{
    assert(parameters);
    assert(parameters->GetParamNumber() == 3);
    assert(uncorrected > 0.);
    assert(pid > 0.);

    Double_t corrected(
        uncorrected - (
            parameters->GetParameter(0) +
            (parameters->GetParameter(1) * pid) +
            (parameters->GetParameter(2) * TMath::Power(pid, 2.))
        )
    );

    assert((corrected >= 0.) && (corrected < 10.));

    return corrected;
}

UChar_t IdentifyE503::Identify(KVVAMOSReconNuc *const n)
{
    // Identify() (Public Interface Method)
    //
    // This is the identification method for the E503 experiment. The nucleus is
    // first identified with the usual KVVAMOSReconNuc::Identify() routine -
    // which calls the respective Identify() methods in each of the
    // identification telescopes. This result is then used as the basis for the
    // global VAMOS identification procedure which is used to calculate the
    // mass, charge and mass-to-charge ratio. The E503 identification
    // corrections are subsequently applied.
    //
    // The UChar_t return value is used to pass information about the
    // identification status back to the user without resorting to expensive I/O
    // calls.

    assert(n);
    assert(kInitialised_);

    n->SetFlightDistanceAndTime();

    KVVAMOSCodes codes(n->GetCodes());
    if (codes.GetTCode() == kTCode0) {
        // Badly calibrated time
        return kBadTimeCal;
    }

    assert((n->GetTimeOfFlight() > 0.) && (n->GetTimeOfFlight() < 500.));

    n->Identify();
    if (!n->IsIdentified()) return kNotIdentified;

    KVIDTelescope *idt(n->GetIdentifyingTelescope());
    if (!idt) return kNoIdTelescope;

    KVIdentificationResult *idr(n->GetIdentificationResult(idt));
    if (!idr) return kNoIdResult;
    if (!idr->IDOK) return kBasicIdNotOk;
    if (!idr->Zident) return kZIdNotOk;
    if (idr->Z < 1) return kUnCharged;;

    const Bool_t kICSi(idt->InheritsFrom("e503::KVIDHarpeeICSi_e503"));
    const Bool_t kSiCsI(idt->InheritsFrom("e503::KVIDHarpeeSiCsI_e503"));

    if (!kICSi && !kSiCsI) {
        // The telescope does not inherit from either of the expected classes
        return kNotInherited;
    }

    if (kICSi && kSiCsI) {
        // You can't inherit from both telescopes! What's going on?!
        return kBothInherited;
    }

    if ((idr->IDcode != kIDCode3) && (idr->IDcode != kIDCode4)) {
        // The identification result does not have either of the expected
        // identification codes
        return kInvalidIdCode;
    }

    Double_t total_energy(0.);

    KVVAMOSDetector *stopping_detector(
        static_cast<KVVAMOSDetector*>(n->GetStoppingDetector())
    );

    assert(stopping_detector);

    KVVAMOSDetector *residual_detector(
        static_cast<KVVAMOSDetector*>(idt->GetDetector(2))
    );

    assert(residual_detector);

    // We need the stopping detector to be sure we have the complete energy loss
    // information, otherwise the simulation will give bad results.

    if (stopping_detector != residual_detector) return kNotStoppingDetector;

    KVNucleus sim_nucleus;

    if (idr->Zident && idr->Aident) {
        sim_nucleus.SetZandA(idr->Z, idr->A);
    } else {
        // A Value is automatically calculated.
        sim_nucleus.SetZ(idr->Z);
    }

    assert((sim_nucleus.GetZ() > 0) && (sim_nucleus.GetZ() < 120));
    assert((sim_nucleus.GetA() > 0) && (sim_nucleus.GetA() < 300));

    // A list of all the detectors which are aligned with and between the
    // residual energy detector and the target.
    TList *detector_list(
        residual_detector->GetAlignedDetectors(KVGroup::kBackwards)
    );

    // Iterate over the list of aligned detectors (the list is in reverse order)
    // and add up the energies of all the absorbers, making our way back towards
    // the target. We must do this explicitly as the VAMOS definition of
    // "calibrated detector" may differ from the standard KaliVeda definition.
    // All we care about here is that the detector is calibrated in energy.

    Double_t detector_eloss(0.);
    Double_t absorber_eloss(0.);

    energy_data_.Reset();

    KVVAMOSDetector *detector(NULL);
    TIter next_detector(detector_list);
    while ((detector = static_cast<KVVAMOSDetector*>(next_detector()))) {

        const Bool_t kDetectorIsDC(detector->InheritsFrom("KVDriftChamber"));
        const Bool_t kDetectorIsIC(detector->InheritsFrom("KVHarpeeIC"));
        const Bool_t kDetectorIsSi(detector->InheritsFrom("KVHarpeeSi"));
        const Bool_t kDetectorIsCsI(detector->InheritsFrom("KVHarpeeCsI"));

        KVList *absorbers(detector->GetListOfAbsorbers());

        // For each detector we iterate IN REVERSE over the absorbers (making
        // our way towards the target) and sum either the calculated energy loss
        // in the dead layers or the calibrated energy in the active layer.

        detector_eloss = 0.;

        KVMaterial *abs(NULL);
        TIter next_absorber(absorbers, kIterBackward);
        while ((abs = static_cast<KVMaterial*>(next_absorber()))) {

            absorber_eloss = 0.;

            if (abs == detector->GetActiveLayer()) {

                // The absorber is the active layer of the detector and so we
                // can simply call the detector's GetCalibE function, or in the
                // case of the CsI detector (which requires the Z and A values)
                // GetCorrectedEnergy(). Also the drift chamber is not
                // calibrated in energy and must be calculated.

                if (kDetectorIsCsI) {

                    KVHarpeeCsI *csi(static_cast<KVHarpeeCsI*>(detector));
                    assert(csi);

                    // XXX: Be careful: This only works because the energy of
                    // the nucleus is not used in GetCorrectedEnergy(), should
                    // this change then we will need to rethink.

                    absorber_eloss = csi->GetCorrectedEnergy(&sim_nucleus, -1.,
                        0);

                } else if (kDetectorIsDC) {

                    // Calculate energy loss (not calibrated).
                    KVMaterial *drift_active(static_cast<KVMaterial*>(abs));
                    assert(drift_active);

                    absorber_eloss = drift_active->GetDeltaEFromERes(
                                        sim_nucleus.GetZ(),
                                        sim_nucleus.GetA(),
                                        total_energy
                                    );

                    if (detector->GetNumber() == 1) {
                        energy_data_.dc1_active = absorber_eloss;
                    } else if (detector->GetNumber() == 2) {
                        energy_data_.dc2_active = absorber_eloss;
                    }

                } else {

                    // Other calibrated deterctor
                    absorber_eloss = detector->GetCalibE();

                    if (kDetectorIsIC) {
                        energy_data_.ic_active = absorber_eloss;
                    }

                }

                detector_eloss += absorber_eloss;
                total_energy += absorber_eloss;

                continue;
            }

            if (total_energy <= 0.) {

                // The total_energy should, in principle, already be set by
                // either the silicon or CsI detectors (both have only one layer
                // and it is active) so we should not encounter this condition.

                return kResidualEnergyFail;
            }

            // Total energy is the residual energy since we are working towards
            // the target from the stopping detector
            Double_t incident_energy(
                abs->GetIncidentEnergyFromERes(
                    sim_nucleus.GetZ(),
                    sim_nucleus.GetA(),
                    total_energy
                )
            );

            // Vector normal to absorber surface, pointing from the origin
            // toward the material - in this case simply the z direction (beam
            // direction)
            TVector3 norm(0, 0, 1);

            sim_nucleus.SetEnergy(incident_energy);
            sim_nucleus.SetTheta(n->GetThetaL());
            sim_nucleus.SetPhi(n->GetPhiL());

            absorber_eloss = abs->GetELostByParticle(&sim_nucleus, &norm);

            detector_eloss += absorber_eloss;
            total_energy += absorber_eloss;

        }

        if (kDetectorIsCsI) {
            energy_data_.csi = detector_eloss;

        } else if (kDetectorIsSi) {
            energy_data_.si = detector_eloss;

        } else if (kDetectorIsDC) {

            if (detector->GetNumber() == 1)
                energy_data_.dc1 = detector_eloss;

            else if (detector->GetNumber() == 2)
                energy_data_.dc2 = detector_eloss;

        } else if (kDetectorIsIC) {
            energy_data_.ic = detector_eloss;
        }

        if (kDetectorIsSi && kSiCsI) {

            // Si->CsI dead layer correction.
            //
            // Be careful here! This calculation relies upon the silicon
            // detector having only one absorber, which is calibrated. If the
            // silicon detector has any dead layers then this calculation will
            // not be accurate, as we only make use of the calibrated energy.
            //
            // The alternative would be to use detector_eloss in the incident
            // energy calculation but that is also problematic, as you introduce
            // a cyclic dependency on the calculated total energy! We end up in
            // this situation as the residual energy from the CsI detector is
            // zero and so reliable calculations of the CsI incident energy are
            // not possible.

            // silicon detector incident energy
            assert((detector->GetCalibE() >= 0.) &&
                   (detector->GetCalibE() < 3000.));

            Double_t incident_energy(
                detector->GetEResFromDeltaE(
                    sim_nucleus.GetZ(),
                    sim_nucleus.GetA(),
                    detector->GetCalibE()
                )
            );

            // Vector normal to dead-layer surface, pointing from the origin
            // toward the material - in this case simply the z direction (beam
            // direction)

            TVector3 norm(0, 0, 1);
            KVMaterial isobutane("C4H10", 13.65 * KVUnits::cm);
            isobutane.SetPressure(40.*KVUnits::mbar);

            sim_nucleus.SetEnergy(incident_energy);
            sim_nucleus.SetTheta(n->GetThetaL());
            sim_nucleus.SetPhi(n->GetPhiL());

            absorber_eloss = isobutane.GetELostByParticle(&sim_nucleus, &norm);
            total_energy += absorber_eloss;

            energy_data_.iso_sicsi = absorber_eloss;

            // The silicon detector losses have already been taken into account
            // above.
            continue;

        }

        if (kDetectorIsDC && (detector->GetNumber() == 2)) {

            // Make the SED correction
            //
            // We calculate the incident energy to the second drift chamber
            // from its residual energy and use it as the basis for calculating
            // the energy loss in the sed. Care is needed to set the correct
            // effective thickness of the material for the particle.

            // Vector normal to SED surface, pointing from the origin toward the
            // material - in this case 45 degrees (the beam [0,0,1] sees the
            // thickness as 1.273um)
            TVector3 norm(0, -1, 1);
            KVMaterial sed("Myl", 0.9 * KVUnits::um);

            // total_energy at this point includes the second drift chamber
            // energy losses and so this needs to be subtracted again to get the
            // residual energy.
            Double_t drift_2_incident_energy(
                detector->GetIncidentEnergyFromERes(
                    sim_nucleus.GetZ(),
                    sim_nucleus.GetA(),
                    total_energy - detector_eloss
                )
            );

            // Nucleus is defined just prior to entering the second drift
            // chamber and is used only to evaluate the direction in which it is
            // travelling.

            sim_nucleus.SetEnergy(drift_2_incident_energy);
            sim_nucleus.SetTheta(n->GetThetaL());
            sim_nucleus.SetPhi(n->GetPhiL());

            // Normalised!
            TVector3 momentum(sim_nucleus.GetMomentum());
            Double_t mag(momentum.Mag());

            TVector3 dir;
            dir.SetX(momentum.X() / mag);
            dir.SetY(momentum.Y() / mag);
            dir.SetZ(momentum.Z() / mag);

            Double_t effective_thickness(sed.GetEffectiveThickness(norm, dir));

            // Set the linear thickness of the SED to the correct effective
            // thickness.
            sed.SetThickness(effective_thickness);

            absorber_eloss = sed.GetDeltaEFromERes(
                                 sim_nucleus.GetZ(),
                                 sim_nucleus.GetA(),
                                 drift_2_incident_energy
                             );

            total_energy += absorber_eloss;

            energy_data_.sed = absorber_eloss;

            // The drift chamber energy losses have already been accounted for
            // above.
            continue;
        }

    }

    // Final corrections to the energy for the strip foil and the target

    KVMaterial strip_foil(20.*KVUnits::ug, "C");

    absorber_eloss = strip_foil.GetDeltaEFromERes(
                         sim_nucleus.GetZ(),
                         sim_nucleus.GetA(),
                         total_energy
                     );
    total_energy += absorber_eloss;
    energy_data_.strip_foil = absorber_eloss;

    KVTarget *target(gVamos->GetTarget());

    if (!target) {
        return kNoTarget;
    }

    absorber_eloss = target->GetDeltaEFromERes(
                         sim_nucleus.GetZ(),
                         sim_nucleus.GetA(),
                         total_energy
                     );
    total_energy += absorber_eloss;
    energy_data_.target = absorber_eloss;

    assert((total_energy >= 0.) && (total_energy < 3000.));

    Double_t real_z(idr->PID);

    assert((n->GetTimeOfFlight() > 0.) && (n->GetTimeOfFlight() < 500.));

    Double_t real_a(
        n->CalculateRealA(
            sim_nucleus.GetZ(),
            total_energy,
            n->GetBetaFromToF()
        )
    );

    Double_t pid(real_z + 0.1 * (real_a  - 2. * real_z));

    Int_t z_value(sim_nucleus.GetZ());
    Int_t a_value(sim_nucleus.GetA());

    // TODO: Just occasionally (1/20,000) something weird happens with the
    // energy calculation and we end up with A = 0. Set a flag for now and come
    // back and analyse what is going on another time - I couldn't find anything
    // immediately obvious. I think it's something to do with the nucleus
    // stating that it has stopped in the silicon but there's still some energy
    // detected in the CsI behind it. So essentially we have some missing energy
    // still to account for, I'm not sure why the CsI does not register as being
    // hit...

    if (a_value < 1) {
        return kUnChargedResult;
    }

    CorrectionData data;
    data.idt = idt;
    data.z_value = z_value;
    data.a_value = a_value;
    data.z_real = real_z;
    data.a_real = real_a;
    data.pid = pid;
    data.energy = total_energy;

    if (!ApplyCorrections(n, &data)) {
        return kCorrectionsFailed;
    }

    return kAllOK;
}

// TODO: Implement Calulate energy as a seperate function too?
Bool_t IdentifyE503::ApplyCorrections(
    KVVAMOSReconNuc *const n,
    const CorrectionData *const data
)
{
    assert(n);
    assert(data);

    const KVIDTelescope *const idt(data->idt);
    assert(idt);

    Bool_t status(kFALSE);

    if (idt->InheritsFrom("e503::KVIDHarpeeICSi_e503")) {
        const KVIDHarpeeICSi_e503 *const icsi(
            static_cast<const KVIDHarpeeICSi_e503 *const>(idt)
        );
        assert(icsi);

        status = ApplyIcSiCorrections(n, icsi, data);

    } else if (idt->InheritsFrom("e503::KVIDHarpeeSiCsI_e503")) {
        const KVIDHarpeeSiCsI_e503 *const sicsi(
            static_cast<const KVIDHarpeeSiCsI_e503 *const>(idt)
        );
        assert(sicsi);

        status = ApplySiCsiCorrections(n, sicsi, data);

    }

    return status;
}

Bool_t IdentifyE503::ApplyIcSiCorrections(
    KVVAMOSReconNuc *const n,
    const KVIDHarpeeICSi_e503 *const idt,
    const CorrectionData *const data
)
{
    // Apply corrections for the Ionisation Chamber:Silicon Telescopes
    // TODO: Can any of this code be consoldiated? There is a lot of duplication
    // between ICSi and SiCsI functions

    assert(n);
    assert(idt);
    assert(data);

    KVList *parameters(idt->GetIDCorrectionParameters());
    assert(parameters);

    KVHarpeeIC *ic(static_cast<KVHarpeeIC*>(idt->GetDetector(1)));
    assert(ic);

    // A/Q Straightening parameters
    KVDBParameterSet *a_over_q_straight(
        static_cast<KVDBParameterSet*>(
            parameters->FindObject(
                Form("straight_fct_chiosi_chio%d",
                     ic->GetFiredSegNumber()
                    )
            )
        )
    );

    if (!a_over_q_straight) {
        // No straightening parameters
        return kFALSE;
    }

    Double_t uncorrected_a_over_q(n->GetRealAoverQ());
    Double_t corrected_a_over_q(
        CorrectAoverQ(a_over_q_straight, uncorrected_a_over_q, data->pid)
    );

    // More Parameters
    //Correct();

    // Finally, modify the nucleus:

    // Note: This sets kinetic energy which must be accessed in your selector
    // class by KVVAMOSReconNuc::GetKE();

    n->SetZAandE(data->z_value, data->a_value, data->energy);
    n->SetRealZ(data->z_real);
    n->SetRealA(data->a_real);

    // NOTE: KVVAMOSReconNuc::Calibrate() is extremely slow (lots of TF1::Eval
    // going on) and it reduces the event rate from ~27 events/s down to ~7
    // events/s (3.8x slower) I don't think I need it anyway (I *think*) as I'm
    // setting the energy manually in the code above. As a result the nucleus
    // will not appear calibrated, but the nucleus itself should be OK.

    //n->Calibrate();
    //if (!n->IsCalibrated()) return kNotCalibrated;

    return kTRUE;
}

Bool_t IdentifyE503::ApplySiCsiCorrections(
    KVVAMOSReconNuc *const n,
    const KVIDHarpeeSiCsI_e503 *const idt,
    const CorrectionData *const data
)
{
    // Apply corrections for the Silicon:Caesium Iodide Telescopes

    assert(n);
    assert(idt);
    assert(data);

    KVList *parameters(idt->GetIDCorrectionParameters());
    assert(parameters);

    KVHarpeeSi *si(static_cast<KVHarpeeSi*>(idt->GetDetector(1)));
    assert(si);

    // A/Q Straightening parameters
    KVDBParameterSet *a_over_q_straight(
        static_cast<KVDBParameterSet*>(
            parameters->FindObject(
                Form("straight_fct_si%d",
                     si->GetNumber()
                    )
            )
        )
    );

    if (!a_over_q_straight) {
        // No straightening parameters
        return kFALSE;
    }

    Double_t uncorrected_a_over_q(n->GetRealAoverQ());
    Double_t corrected_a_over_q(
        CorrectAoverQ(a_over_q_straight, uncorrected_a_over_q, data->pid)
    );

    // More Parameters
    //Correct();

    // Finally, modify the nucleus:

    // Note: This sets kinetic energy which must be accessed in your selector
    // class by KVVAMOSReconNuc::GetKE();

    n->SetZAandE(data->z_value, data->a_value, data->energy);
    n->SetRealZ(data->z_real);
    n->SetRealA(data->a_real);

    // NOTE: KVVAMOSReconNuc::Calibrate() is extremely slow (lots of TF1::Eval
    // going on) and it reduces the event rate from ~27 events/s down to ~7
    // events/s (3.8x slower) I don't think I need it anyway (I *think*) as I'm
    // setting the energy manually in the code above. As a result the nucleus
    // will not appear calibrated, but the nucleus itself should be OK.

    //n->Calibrate();
    //if (!n->IsCalibrated()) return kNotCalibrated;

    return kTRUE;
}

const StageEnergies& IdentifyE503::get_energy_data() const
{
    return energy_data_;
}

}
```

Listing B.6: IdentifyE503.cpp

References

[1] C.A. Bertulani and P. Danielewicz. *Introduction to Nuclear Reactions*. Graduate Student Series in Physics. Institute of Physics Publishing, 2004. ISBN 0-7503-0932-6.

[2] J.M. Lattimer and A.W. Steiner. Constraints on the symmetry energy using the mass-radius relation of neutron stars. *European Physical Journal A*, 50(2), 2014.

[3] Bao-An Li, Lie-Wen Chen, and Che Ming Ko. Recent progress and new challenges in isospin physics with heavy-ion reactions. *Physics Reports*, 464(4-6):113 – 281, 2008.

[4] F.J. Fattoyev, W.G. Newton, and B.-A Li. Probing the high-density behavior of symmetry energy with gravitational waves. *European Physical Journal A*, 50(2): 45, 2014.

[5] J.M. Pearson, N. Chamel, A.F. Fantina, et al. Symmetry energy: nuclear masses and neutron stars. *European Physical Journal A*, 50(2):43, 2014.

[6] K. Iida and K. Oyamatsu. Symmetry energy, unstable nuclei and neutron star crusts. *European Physical Journal A*, 50(2):42, 2014.

[7] W.G. Newton, J. Hooker, M. Gearheart, et al. Constraints on the symmetry energy from observational probes of the neutron star crust. *European Physical Journal A*, 50(2):41, 2014.

[8] S. Gandolfi, J. Carlson, S. Reddy, et al. The equation of state of neutron matter, symmetry energy and neutron star structure. *European Physical Journal A*, 50(2): 10, 2014.

[9] T. Fischer, M. Hempel, I. Sagert, et al. Symmetry energy impact in simulations of core-collapse supernovae. *European Physical Journal A*, 50(2):46, 2014.

[10] X. Viñas, M. Centelles, X. Roca-Maza, et al. Density dependence of the symmetry energy from neutron skin thickness in finite nuclei. *European Physical Journal A*, 50(2):27, 2014.

[11] V. Baran, M. Colonna, V. Greco, et al. Reaction dynamics with exotic nuclei. *Physics Reports*, 410:335 – 466, 2005.

[12] L. Trippa, G. Colò, and E. Vigezzi. Giant dipole resonance as a quantitative constraint on the symmetry energy. *Physical Review C*, 77(6), 2008.

[13] A. Klimkiewicz, P. Adrich, M. Fallot, et al. Nuclear symmetry energy and neutron skins derived from pygmy dipole resonances. *Physical Review C*, 76(5), 2007.

[14] A. Carbone, G. Colò, A. Bracco, et al. Constraints on the symmetry energy and neutron skins from pygmy resonances in Ni-68 and Sn-132. *Physical Review C*, 81 (4), 2010.

[15] X. Roca-Maza, M. Brenna, P.F. Bortignon, et al. Giant quadrupole resonances in 208Pb, the nuclear symmetry energy, and the neutron skin thickness. *Physical Review C*, 87(3), 2013.

[16] L.-G. Cao, G. Colò, H. Sagawa, et al. Constraints on the neutron skin and symmetry energy from the anti-analog giant dipole resonance in Pb 208. *Physical Review C*, 92(3), 2015.

[17] G. Colò, U. Garg, and H. Sagawa. Symmetry energy from the nuclear collective motion: constraints from dipole, quadrupole, monopole and spin-dipole resonances. *European Physical Journal A*, 50(2):26, 2014.

[18] P. Danielewicz and J. Lee. Symmetry energy from systematic of isobaric analog states. *AIP Conference Proceedings*, 1423(1):29 – 34, 2012.

[19] M.A. Famiano, T. Liu, W. G. Lynch, et al. Neutron and Proton Transverse Emission Ratio Measurements and the Density Dependence of the Asymmetry Term of the Nuclear Equation of State. *Physical Review Letters*, 97:052701, 2006.

[20] Z. Kohley, L.W. May, S. Wuenschel, et al. Transverse collective flow and midrapidity emission of isotopically identified light charged particles. *Physical Review C*, 83:044601, 2011.

[21] F. Amorini, G. Cardella, G. Giuliani, et al. Isospin Dependence of Incomplete Fusion Reactions at 25 MeV/Nucleon. *Physical Review Letters*, 102:112701, 2009.

[22] E. De Filippo, A. Pagano, P. Russotto, et al. Correlations between emission timescale of fragments and isospin dynamics in ^{124}Sn+^{64}Ni and ^{112}Sn+^{58}Ni reactions at 35A MeV. *Physical Review C*, 86:014610, 2012.

[23] J. B. Natowitz, G. Röpke, S. Typel, et al. Symmetry Energy of Dilute Warm Nuclear Matter. *Physical Review Letters*, 104:202501, 2010.

[24] R. Wada, K. Hagel, L. Qin, et al. Nuclear matter symmetry energy at $0.03 \leq \rho/\rho_0 \leq 0.2$. *Physical Review C*, 85:064618, 2012.

[25] P. Russotto, M.D. Cozma, A. Le Fèvre, et al. Flow probe of symmetry energy in relativistic heavy-ion reactions. *European Physical Journal A*, 50(2):38, 2014.

[26] M.B. Tsang, W.A. Friedman, C.K. Gelbke, et al. Isotopic scaling in nuclear reactions. *Physical Review Letters*, 86(22):5023 – 5026, 2001.

[27] Z. Kohley and S.J. Yennello. Heavy-ion collisions: Direct and indirect probes of the density and temperature dependence of E-sym. *European Physical Journal A*, 50(2), 2014.

[28] M.B. Tsang, W.A. Friedman, C.K. Gelbke, et al. Conditions for isoscaling in nuclear reactions. *Physical Review C*, 64(4), 2001.

[29] A.S. Botvina, O.V. Lozhkin, and W. Trautmann. Isoscaling in light-ion induced reactions and its statistical interpretation. *Physical Review C*, 65(4), 2002.

[30] M.B. Tsang, W.A. Friedman, C.K. Gelbke, et al. Conditions for Isoscaling in Nuclear Reactions. 2001. arXiv:nucl-ex/0106009, image reference only.

[31] C Dorso. Isoscaling: Geometry, correlations and symmetry energy. *Physical Review C*, 73(3), 2006.

[32] A. Ono, P. Danielewicz, W. A. Friedman, et al. Symmetry energy for fragmentation in dynamical nuclear collisions. *Physical Review C*, 70:041604, 2004.

[33] V. Baran, M. Colonna, M. Di Toro, et al. Isospin transport at Fermi energies. *Physical Review C*, 72(6), 2005.

[34] M. Colonna, V. Baran, and M. Di Toro. Theoretical predictions of experimental observables sensitive to the symmetry energy. *European Physical Journal A*, 50 (2):30, 2014.

[35] F. Rami, B. De Schauenburg, P. Wagner, et al. Isospin Tracing: A Probe of Nonequilibrium in Central Heavy-Ion Collisions. *Physical Review Letters*, 84(6): 1120–1123, 2000.

[36] J. Rizzo, M. Colonna, V. Baran, et al. Isospin dynamics in peripheral heavy ion collisions at fermi energies. *Nuclear Physics A*, 806(1-4):79 – 104, 2008.

[37] M.B. Tsang, T.X. Liu, L. Shi, et al. Isospin diffusion and the nuclear symmetry energy in heavy ion reactions. *Physical Review Letters*, 92(6), 2004.

[38] E. Galichet, M. F. Rivet, B. Borderie, et al. Isospin diffusion in 58Ni-induced reactions at intermediate energies. I. Experimental results. *Physical Review C*, 79 (6), 2009.

[39] E. Galichet, M. Colonna, B. Borderie, et al. Isospin diffusion in 58Ni-induced reactions at intermediate energies. II. Dynamical simulations. *Physical Review C*, 79(6), 2009.

[40] M.B. Tsang, Y.X. Zhang, P. Danielewicz, et al. Constraints on the Density Dependence of the Symmetry Energy. *Physical Review Letters*, 102(12), 2009.

[41] G. Ademard, B. Borderie, A. Chbihi, et al. Isospin effects and symmetry energy studies with INDRA. *European Physical Journal A*, 50(2), 2014.

[42] D. Durand, E. Suraud, and B. Tamain. *Nuclear Dynamics in the Nucleonic Regime.* Series in Fundamental and Applied Nuclear Physics. Institute Of Physics Publishing, 2001. ISBN 0-7503-0537-1.

[43] Z. Fraenkel. Review of the intranuclear cascade model for heavy ion reactions. *Nuclear Physics A*, 428:373 – 387, 1984.

[44] W. Botermans and R. Malfliet. Two-body collisions and mean-field theory: The Brüeckner-Boltzmann equation. *Physics Letters B*, 171(1):22–27, 1986.

[45] E.A. Uehling and G.E. Uhlenbeck. Transport phenomena in Einstein-Bose and Fermi-Dirac gases. I. *Physical Review*, 43(7):552–561, 1933.

[46] M. Colonna, M. Di Toro, A. Guarnera, et al. Fluctuations and dynamical instabilities in heavy-ion reactions. *Nuclear Physics A*, 642(3–4):449 – 460, 1998.

[47] M. Colonna, G. Fabbri, M. Di Toro, et al. Fragmentation path of excited nuclear systems. *Nuclear Physics A*, 742:337 – 347, 2004.

[48] A. Ono, H. Horiuchi, T. Maruyama, et al. Fragment formation studied with antisymmetrized version of molecular dynamics with two-nucleon collisions. *Physical Review Letters*, 68:2898–2900, 1992.

[49] A. Ono. Antisymmetrized molecular dynamics with quantum branching processes for collisions of heavy nuclei. *Physical Review C*, 59:853–864, 1999.

[50] J. Rizzo, M. Colonna, and A. Ono. Comparison of multifragmentation dynamical models. *Physical Review C*, 76(2), 2007.

[51] M. Colonna, A. Ono, and J. Rizzo. Fragmentation paths in dynamical models. *Physical Review C*, 82(5), 2010.

[52] D. Lacroix, A. Van Lauwe, and D. Durand. Event generator for nuclear collisions at intermediate energies. *Physical Review C*, 69(5), 2004.

[53] D. Durand. An event generator for the study of nuclear collisions in the Fermi energy domain (I). Formalism and first applications. *Nuclear Physics A*, 541(2): 266–294, 1992.

[54] R. Charity. GEMINI: A Code to Simulate the Decay of a Compound Nucleus by a Series of Binary Decays. In *Joint ICTP-IAEA Advanced Workshop on Model Codes for Spallation Reactions*, page 139, 2008.

[55] J. Pouthas, B. Borderie, R. Dayras, et al. INDRA, a 4π charged product detection array at GANIL. *Nuclear Instruments and Methods in Physics Research, A*, 357: 418–442, 1995.

[56] J. Pouthas, A. Bertaut, B. Borderie, et al. The electronics of the INDRA 4π detection array. *Nuclear Instruments and Methods in Physics Research, A*, 369 (1):222 – 247, 1996.

[57] M. Pârlog, B. Borderie, M.F. Rivet, et al. Response of CsI(Tl) scintillators over a large range in energy and atomic number of ions. Part II: calibration and identification in the INDRA array. *Nuclear Instruments and Methods in Physics Research, A*, 482:693 – 706, 2002.

[58] G.F. Knoll. *Radiation Detection and Measurement*. John Wiley & Sons, 2010. ISBN 9780470131480.

[59] S. Pullanhiotan, M. Rejmund, A. Navin, et al. Performance of VAMOS for reactions near the Coulomb barrier. *Nuclear Instruments and Methods in Physics Research, A*, 593(3):343 – 352, 2008.

[60] Hervé Savajols. VAMOS: A variable mode high acceptance spectrometer for identifying reaction products induced by SPIRAL beams. *Nuclear Instruments and Methods in Physics Research, B*, 204:146–153, 2003.

[61] S. Pullanhiotan, A. Chatterjee, B. Jacquot, et al. Improvement in the reconstruction method for VAMOS spectrometer. *Nuclear Instruments and Methods in Physics Research, B*, 226:4148–4152, 2008.

[62] M. Labiche, W.N. Catford, R.C. Lemmon, et al. TIARA: A large solid angle silicon array for direct reaction studies with radioactive beams. *Nuclear Instruments and Methods in Physics Research, A*, 614:439 – 448, 2010.

[63] C.E. Demonchy, M. Caamaño, H. Wang, et al. MAYA: An active-target detector for binary reactions with exotic beams. *Nuclear Instruments and Methods in Physics Research, A*, 583:341 – 349, 2007.

[64] L. Nalpas. Le Programme "Multifragmentation" d'INDRA, 2015. URL `http://indra.in2p3.fr/multifrag/expe/expe.html`. [Online; accessed 30-April-2015].

[65] L. Tassan-Got. A new functional for charge and mass identification in $\Delta E - E$ telescopes. *Nuclear Instruments and Methods in Physics Research, B*, 194:503 – 512, 2002.

[66] M. Pârlog, B. Borderie, M.F. Rivet, et al. Response of CsI(Tl) scintillators over a large range in energy and atomic number of ions. Part I: recombination and δ-electrons. *Nuclear Instruments and Methods in Physics Research, A*, 482:674 – 692, 2002.

[67] R. Dayras and E. de Filippo. Reference for the INDRA CHIO-Si Identification and the Energy Loss Routines. *CEA Rapport SPhN-95-60*, 1995.

[68] M. Boisjoli. Étude de l'energie de symétrie dans les collisions 40,48Ca+40,48Ca á 35 MeV/A, 2013. Nuclear Experiment. Université de Caen.

[69] P.J. Mohr and B.N. Taylor. CODATA recommended values of the fundamental physical constants: 2002*. *Reviews of Modern Physics*, 77:1–107, 2005.