Data Bank
NEA/DB/DOC(2014)4
www.oecd-nea.org

Analysis of Geel Spectra – AGS

Users' manual

OECD
BETTER POLICIES FOR BETTER LIVES

NEA
NUCLEAR ENERGY AGENCY

Unclassified **NEA/DB/DOC(2014)4**

Organisation de Coopération et de Développement Économiques
Organisation for Economic Co-operation and Development **29-Sep-2014**

English - Or. English

NUCLEAR ENERGY AGENCY
STEERING COMMITTEE FOR NUCLEAR ENERGY

NEA/DB/DOC(2014)4
Unclassified

Data Bank Management Committee

AGS - Analysis of Geel Spectra
Users' Manual

B. Becker, C. Bastian, J. Heyse, S. Kopecky, P. Schillebeeckx

European Commission, Joint Research Centre

Institute for Reference Materials and Measurements
Retieseweg 111, B-2440 Geel, Belgium
September 2014

kiyoshi matsumoto@oecd.org
+33 01 45 24 10 80

JT03362692

English - Or. English

AGS – Analysis of Geel Spectra

Users' Manual

B. Becker, C. Bastian, J. Heyse, S. Kopecky, P. Schillebeeckx

European Commission, Joint Research Centre
Institute for Reference Materials and Measurements
Retieseweg 111, B-2440 Geel, Belgium

September 2014

Foreword

This manual describes the AGS (Analysis of Geel Spectra) code (v.1.0.2), which has been developed to transform count rate spectra of time of flight measurements in an efficient and convenient way into experimental observables.

The output of multi-channel data acquisition systems widely used in experimental physics often consists of counting histograms. These histograms or spectra undergo various mathematical transformations or operations to obtain the final experimental observables. To perform these operations the AGS package has been developed at the European Commission, Joint Research Centre, Institute for Reference Materials and Measurements (EC-JRC-IRMM) [1, 2, 3]. The system consists of a set of stand-alone C programs to transform count rate spectra resulting from time-of-flight experiments in an efficient and convenient way into an observable, transmission or reaction yield. The observables together with their covariance information can be used to determine nuclear reaction model parameters in an adjustment to experimental data [3, 4]. The use of the AGS concept has been recommended in a consultants' meeting organized by the IAEA to report TOF cross section data in the EXFOR data library [5, 6, 7].

Although the AGS package is tailored for neutron time-of-flight (TOF) measurements [4], it can in principle be used for any application involving spectrum transformations. The code is based on a compact formalism performing a full propagation of uncertainties, starting from the uncorrelated uncertainties due to counting statistics. It accounts for both the correlated and uncorrelated uncertainty components, and stores the full covariance information after each operation in a concise, vectorized way. The covariance matrix is split in two parts, separating the uncorrelated and correlated components. The uncorrelated part is represented as a diagonal matrix, while the correlated part is expressed as the product of a rectangular matrix with its own transpose. This structure results in a substantial reduction of data storage volume and provides a structure to verify the various sources of uncertainties through each step of the data reduction process. In addition, it ensures that the resulting covariance matrix is always well defined, i.e. positive definite, and it allows for a convenient way to propagate the uncertainties of parameters resulting from a least squares adjustment.

The development of the AGS code has benefited tremendously over the years from feedback, testing and coding of various users. We acknowledge in particular the contributions of A. Borella from SCK•CEN (Belgium) and F. Gunsing from CEA Saclay (France). We are also very grateful to the Nuclear Energy Agency of the OECD and the Nuclear Data Section of the IAEA for their interest in this work.

Table of contents

1 Data reduction of TOF-experiments **5**
1.1 Time-of-flight technique . . . 5
1.2 Total cross section measurements . . . 6
1.3 Reaction cross section measurements . . . 7
1.4 Dead time correction . . . 9

2 Uncertainty propagation **10**
2.1 Generalized law of uncertainty propagation . . . 10
2.2 AGS formalism for uncertainty propagation of TOF-cross section data . . . 10

3 Data analysis **13**
3.1 Parameter estimation . . . 13
3.2 Estimation of the total covariance of estimated parameters . . . 13
3.2.1 Conventional uncertainty propagation . . . 13
3.2.2 Monte Carlo . . . 14
3.2.3 Marginalization . . . 15

4 The AGS code **16**
4.1 AGS commands . . . 16
4.2 Syntax . . . 19
4.2.1 ags_addval . . . 20
4.2.2 ags_avrg . . . 21
4.2.3 ags_divi . . . 22
4.2.4 ags_divval . . . 23
4.2.5 ags_dtco . . . 24
4.2.6 ags_ener . . . 26
4.2.7 ags_exp . . . 27
4.2.8 ags_fit . . . 28
4.2.9 ags_fpa . . . 31
4.2.10 ags_func . . . 32
4.2.11 ags_fxyp . . . 33
4.2.12 ags_getA . . . 35
4.2.13 ags_getE . . . 36
4.2.14 ags_getET . . . 37
4.2.15 ags_getH . . . 38
4.2.16 ags_getW . . . 39
4.2.17 ags_getXY . . . 40
4.2.18 ags_getY . . . 41
4.2.19 ags_inv . . . 42
4.2.20 ags_lico . . . 43
4.2.21 ags_lift . . . 44
4.2.22 ags_log . . . 45
4.2.23 ags_mpty . . . 46
4.2.24 ags_mult . . . 47
4.2.25 ags_multval . . . 48
4.2.26 ags_putCM . . . 49

4.2.27 ags_putX . . . 50
4.2.28 ags_scan . . . 52
4.2.29 ags_shift . . . 53
4.2.30 ags_subval . . . 54
4.2.31 ags_title . . . 55
4.2.32 ags_tof . . . 56
4.2.33 ags_unav . . . 57

5 Examples **58**
5.1 Transmission . . . 58
5.1.1 Linux . . . 59
5.1.2 Windows . . . 62
5.2 Capture . . . 65
5.2.1 Data import and dead time correction . . . 66
5.2.2 Background . . . 69
5.2.3 Flux . . . 71
5.2.4 Yield . . . 77

Appendix A symbols, nomenclature and abbreviations **79**

References **82**

1 Data reduction of TOF-experiments

In this section a brief overview of the time-of-flight technique and the data reduction of transmission and reaction measurements is given. A more comprehensive review can be found in ref. [4] which discusses more in detail such measurements including a discussion on the uncertainty components.

1.1 Time-of-flight technique

The TOF technique can be applied to determine the velocity v of a neutron from the time t it needs to travel a given distance L. Experimentally this time is derived from the difference between a stop and a start signal, represented by T_s and T_0, respectively. At the GELINA facility of the EC-JRC-IRMM [8] the start signal is produced when the pulsed electron beam passes through a coil just before the beam enters the uranium target. This signal represents the time the neutron is produced. The stop signal or arrival time of the neutron in a transmission experiment is provided by the neutron detector. In a reaction cross section experiment the arrival time is obtained from the detection of the reaction products which are emitted in the neutron induced reaction. The observed TOF t_m becomes

$$t_m = (T_S - T_0) + t_0, \tag{1}$$

where t_0 is a time offset. This time offset, which is mostly due to a difference in cable lengths, can be deduced from a measurement of the TOF of the γ-ray flash produced in the target.

The time-of-flight t_m can be related to the velocity v of the neutron at the moment it leaves the target and enters the detector or sample by

$$v = \frac{L}{t} = \frac{L}{t_m - (t_t + t_d)}, \tag{2}$$

where L is the distance between the outer surface of the neutron-producing target and the front face of the detector or sample, t_t is the time difference between the moment of creation of the neutron and the moment it leaves the target and t_d is the difference between the moment the neutron enters the detector or sample and the time of detection. The kinetic energy of the neutron is given by

$$E = m_n c^2(\gamma - 1), \tag{3}$$

where m_n is the rest mass of the neutron and γ represents the Lorentz factor:

$$\gamma = \frac{1}{\sqrt{1 - (v/c)^2}}, \tag{4}$$

with c the speed of light. At low energies Eq. (3) can be approximated using a numerically more stable cubic expansion in $(v/c)^2$:

$$E = m_n c^2 \left[\frac{1}{2} \left(\frac{v}{c} \right)^2 + \frac{3}{8} \left(\frac{v}{c} \right)^4 + \frac{5}{16} \left(\frac{v}{c} \right)^6 + \ldots \right] . \tag{5}$$

1.2 Total cross section measurements

In a transmission experiment the observed quantity is the fraction of the neutron beam passing through the sample without any interaction. For a parallel neutron beam, which is perpendicular to a homogeneous slab of material, this fraction or transmission T is

$$T = e^{-\sum_k n_k \overline{\sigma}_{tot,k}}, \tag{6}$$

where $\overline{\sigma}_{tot,k}$ is the Doppler broadened total cross section and n_k is the number of atoms per unit area of nuclide k.

Experimentally the transmission T_{exp} is obtained from the ratio of TOF spectra resulting from a sample-in C_{in} and a sample-out measurement C_{out}, after subtraction of the background contributions B_{in} and B_{out}, respectively [9, 10, 11]:

$$T_{exp} = N_T \frac{C_{in} - K\ B_{in}}{C_{out} - K\ B_{out}}. \tag{7}$$

The experimental spectra in Eq. (7) need to be corrected for losses due to the dead time in the detector and electronics chain. All spectra need to be normalized to the same neutron intensity and TOF-bin width. To account for the uncertainty on the normalization due to the beam intensity the factor N_T is introduced. Its uncertainty depends on the stability of both the neutron monitors and the neutron transmission detectors. This uncertainty can be reduced by alternating sequences of sample-in and sample-out measurements. The factor K in Eq. (7) is introduced to include a correlated uncertainty of the used model for B_{in} and B_{out}.

Eq. (7) reveals that the experimental transmission is deduced from a ratio of counting spectra. Therefore, it is independent of the detector efficiency and no absolute measurement of the neutron flux is required. The experimental observable T_{exp} (Eq. (7)) is a direct measure of the theoretical transmission given by Eq. (6) if the measurements are performed in a good transmission geometry, i.e.:

- the sample is perpendicular with respect to a parallel incoming neutron beam;
- all neutrons that are detected have passed through the sample; and
- neutrons scattered by the sample are not detected.

In addition, it requires a constant homogeneous spatial distribution of the sample material. The conditions of an ideal or good transmission geometry can be achieved by a proper collimation of the neutron beam at both the sample and detector position.

The background in a TOF transmission measurement can be considered as a sum of several components [4]:

$$B(t) = B_0 + B_\gamma(t) + B_{no}(t) + B_{ns}(t) + B_{ne}(t)\ , \tag{8}$$

where B_0 is TOF independent and $B_\gamma(t)$, $B_{no}(t)$, $B_{ns}(t)$ and $B_{ne}(t)$ are TOF dependent. For measurements at a moderated neutron beam, the contribution $B_\gamma(t)$ is primarily due to the detection of 2.2 MeV γ-rays resulting from neutron capture in hydrogen present in the moderator [10, 9]. The time distribution of the 2.2 MeV γ-rays is directly related to the slowing down process of the neutrons in the moderator [10, 9]. Additional γ-ray background results from high energy γ-rays and Bremsstrahlung scattered in the target-moderator assembly. Since their energy spectrum is dominated by Compton scattered γ-rays around 250 keV, their contribution

can be significantly reduced by a proper pulse height threshold. In addition, their arrival time corresponds to the one of fast neutrons. The second time dependent component $B_{no}(t)$ results from overlap neutrons, i.e. neutrons which are detected but have been produced in a previous neutron pulse. A third time dependent component $B_{ns}(t)$ originates predominantly from beam neutrons which are scattered inside the detector station. A small background component $B_{ne}(t)$ is due to neutrons scattered in the surroundings (environment) or at other flight paths in case of a multi-user facility.

The background as a function of TOF can be determined by an analytical expression applying the black resonance technique [4]. The free parameters in the analytical expression are determined by a least square fit to resonance dips observed in the TOF-spectra from measurements with black resonance filters. The thickness of the filter is chosen to ensure that the transmission through the filter at an element specific resonance energy is less than 10^{-4}. Elements that can be used as black resonance filters are e.g. Cd, Ag, W, Mo, Co, Al, Na and S. At higher energies relatively thick Si or Ti are applicable. Unfortunately, the presence of both the sample and black resonance filter will alter the background. In particular, the time dependent components $B_{\gamma}(t)$, $B_{no}(t)$ and $B_{ns}(t)$ are very sensitive to the absorption and scattering characteristics of the filter and sample. To avoid bias effects in the background, the change in background level due to the presence of the sample and the filters has to be taken into account [4]. The best accuracy of the background is obtained by measurements with fixed background filters in the beam. Using fixed background filters the impact of the material placed in the beam can be taken into account and the stability of the background level can be controlled at any time. For samples which contain isotopes for which some resonances are black, the corresponding transmission in the sample-in spectra can be used as self-indicating black resonance dips. In most cases the shape of the time dependent components can be considered as fixed, such that only the amplitudes are sensitive to the presence of the filters and sample.

The contribution $B_{no}(t)$ due to overlap neutrons can be estimated from measurements in the same experimental conditions, however, at a lower operating frequency of the accelerator or by an extrapolation of the time response at the end of a cycle.

Functions that are quite commonly used as background corrections are sums of decaying exponential and power functions. A typical function for measurements with a Li-glass scintillator at a moderated neutron beam is [11]

$$B(t) = a_0 + a_1 e^{-\lambda_1 t} + a_2 e^{-\lambda_2 t} + a_3 e^{-\lambda_3 (t+t_0)}. \tag{9}$$

a_0 is the time independent component. The first exponential is due to the 2.2 MeV γ-rays, the second due to neutrons scattered in the surroundings and the third accounts for the contribution of overlap neutrons, which has been estimated by an extrapolation of the spectrum at the end of the cycle. The parameter t_0 is related to the operating frequency of the accelerator.

1.3 Reaction cross section measurements

In a reaction cross section measurement the quantity of interest is the reaction yield, which is the fraction of the neutron beam inducing a reaction in the sample. The theoretical reaction yield $Y_{r,k}$ resulting from a neutron induced reaction (n, r) with nuclide k, can be expressed as a sum of primary $Y_{0,k}$ and multiple interaction events $Y_{m,k}$ [4]

$$Y_{r,k} = Y_{0,k} + Y_{m,k}. \tag{10}$$

The latter are due to a (n,r) reaction after at least one neutron scattering in the sample. For a parallel uniform neutron beam and a homogeneous slab of material, which is placed perpendicular to the neutron beam, the primary yield $Y_{0,k}$ is given by

$$Y_{0,k} = \left(1 - e^{-\sum_j n_j \overline{\sigma}_{tot,j}}\right) \frac{n_k \overline{\sigma}_{r,k}}{\sum_j n_j \overline{\sigma}_{tot,j}}, \tag{11}$$

where $\overline{\sigma}_{r,k}$ is the Doppler broadened reaction cross section and n_j is the number of atoms per unit area of nuclide j. Only for very thin samples and/or small cross sections, such that $\sum_j n_j \overline{\sigma}_{tot,j} \ll 1$, the reaction yield is directly proportional to the reaction cross section, i.e. $Y_{r,k} \approx n_k \overline{\sigma}_{r,k}$.

The experimental yield Y_{exp}, i.e. the observable of a reaction cross section experiment, is defined as the total observed count rate C_r which is corrected for its background contribution B_r and divided by the neutron flux at the sample position:

$$Y_{exp} = N_r \frac{C_r - B_r}{\varphi'} \ , \tag{12}$$

where parameters related to the detection of the reaction product and the absolute value of the neutron flux at a given energy are lumped together into one normalization factor N_r. This factor is determined at an energy where the theoretical yield is well known and only the shape of the neutron flux, i.e. its relative energy dependence denoted by φ', needs to be determined. Such a procedure is only valid when the parameters related to the detection of the reaction product are energy and nuclide independent.

For the analysis of any reaction cross section measurement the absolute or relative neutron flux as a function of neutron energy is essential. Such measurements rely on neutron induced reactions for which the cross sections are known. Since the accuracies of these cross sections directly translate into the accuracy of the experimental yield Y_{exp}, reactions for which the cross sections are considered as a standard [12] are preferred.

A flux measurement based on such standard cross sections is nothing else than a reaction cross section measurement. Therefore, Eq. (12) becomes [4]

$$Y_{exp} = N'_r \frac{C_r - B_r}{C_\varphi - K B_\varphi} \frac{Y_\varphi}{T_\varphi}, \tag{13}$$

where Y_φ is the theoretical yield for the flux measurements, T_φ is the transmission through the flux chamber, and C_φ and B_φ are the total and background count rate for the flux measurement.

To prevent artificial structures in Y_{exp}, due to structures in Y_φ and the finite response of the TOF-spectrometer, standard reactions with a smooth cross section as function of neutron energy, such as the ^{10}B(n,α)^{7}Li reaction in the region below a few hundred keV, are preferred. The ratio $\frac{Y_\varphi}{T_\varphi}$ is given by

$$\frac{Y_\varphi}{T_\varphi} = e^{n_\varphi \bar{\sigma}_{tot,\varphi}} \left(1 - e^{-n_\varphi \bar{\sigma}_{tot,\varphi}}\right) \frac{\bar{\sigma}_{r,\varphi}}{\bar{\sigma}_{tot,\varphi}} \ , \tag{14}$$

where n_φ is the total areal density of the active layer in the flux measurement, $\bar{\sigma}_{r,\varphi}$ the Doppler broadened reaction cross section used to determine the flux and $\bar{\sigma}_{tot,\varphi}$ is the Doppler broadened total cross section for neutron induced reactions in the active layer in the flux measurement.

The background for the flux measurement B_φ can be deduced using the black resonance technique in a similar way as it is done for total cross section measurements (see Section 1.2).

Also for neutron induced fission or charged particle reaction cross section measurements the background B_r is mostly approximated by the black resonance technique. In case of capture measurements the total background contribution B_r of the observed count rate is derived from results of additional measurements without a sample and with a purely scattering sample. B_r can be expressed as [4]

$$B_r(t) = b_0 + K_1 C_0(t) + K_2 R_n(t) \left[C_{scat}(t) - C_0(t) \right] \ , \tag{15}$$

where b_0 is a time independent contribution. C_0 is the TOF-spectrum resulting from measurements with no sample. The third term of the sum takes into account the neutron sensitivity of the detection system due to neutrons that are scattered from the sample. C_{scat} are the counts from measurements with an almost purely scattering sample such as ^{208}Pb. The correction factor R_n is the ratio of the neutron scattering yield of the sample of interest and the scattering sample. The factors K in Eq. (13) and K_1 and K_2 in Eq. (15) are used to include a correlated uncertainty component due to the background model.

1.4 Dead time correction

All count rate spectra obtained from a TOF experiment require a correction due to the dead time of the detection system consisting of the detector, electronics, digitizers and data acquisition system. In case of a fixed (non-extendable) dead time, the correction is well understood.

In a dead time correction procedure the counts $N_0(i)$ of channel i are corrected based on the counts in the previous channels j which fall in the range of the dead time. If an event in these channels occurred then the channel i is blocked due to the dead time of detection system.

In the publication of Moore [13] several models to correct for the dead time are given. Assuming a stable beam the corrected counts $N_c(i)$ in channel i are given by

$$N_c(i) = N_0(i) / \left(1 - \sum_{j=i_0}^{i-1} N_0(j)/N_b \right) \ , \tag{16}$$

where N_b is the total number of bursts. The dead time of the detection system corresponds to the difference in channels $i - i_0$. In case of a varying beam intensity the corrected counts $N_c(i)$ can be determined by

$$N_c(i) = N_b \left\{ -\ln \left(1 - \frac{N_0(i)/N_b}{\left[1 - \sum_{j=i_0}^{i-1} N_0(j)/N_b \right]} \right) \Bigg/ \left[1 - \sigma \ \tanh \left(\sigma \sum_{j=i_0}^{i-1} N_c(j)/N_b \right) \right] \right\} \ , \tag{17}$$

where σ^2 is the relative variance of the beam intensity. Since the dead time correction of channel i depends on the counts in other channels the dead time correction is by definition not a channel-to-channel operation (see Section 2.2).

2 Uncertainty propagation

2.1 Generalized law of uncertainty propagation

Suppose that a quantity $\vec{Z}$ is deduced from the spectra $(\vec{Z}_1, \ldots, \vec{Z}_p)$ and the parameter vector $\vec{b} = (b_1, \ldots, b_q)$ through a function

$$\vec{Z} = f(\vec{Z}_1, \ldots, \vec{Z}_p, \vec{b}). \tag{18}$$

The covariance matrix $\mathbf{V}_{\vec{Z}}$ of $\vec{Z}$ can be derived by applying the generalized law of uncertainties propagation (GLUP) as defined in the Guide to the Expression of Uncertainty in Measurement (GUM) [14]. This law is based on a first-order Taylor series. In case the spectra $(\vec{Z}_1, \ldots, \vec{Z}_p)$ and the parameter $\vec{b}$ are independent, the covariance matrix of $\vec{Z}$ becomes

$$\mathbf{V}_{\vec{Z}} = \sum_{j=1}^{p} \mathbf{D}_h(\vec{Z}_j)\mathbf{V}_{\vec{Z}_j}\mathbf{D}_h^T(\vec{Z}_j) + \mathbf{D}_h(\vec{b})\mathbf{V}_{\vec{b}}\mathbf{D}_h^T(\vec{b}), \tag{19}$$

with $\mathbf{D}_h(\vec{Z}_j)$ and $\mathbf{D}_h(\vec{b})$ being the sensitivity matrices with respect to $\vec{Z}_j$ and $\vec{b}$, respectively. To apply Eq. (19) the full covariance matrices have to be stored and used in the calculations. In addition, the resulting covariance matrix $\mathbf{V}_{\vec{Z}}$ does not yield any information about the original uncertainty components and no differentiation between correlated and uncorrelated components can be made.

2.2 AGS formalism for uncertainty propagation of TOF-cross section data

When the different steps involved in the measurement and data reduction process are fully understood and documented, specific properties of the covariance matrix can be used and a concept can be established such that the impact of each uncertainty component can be identified at all times [1, 2, 3]. In addition, the GLUP can be simplified in terms of data storage and arithmetics.

The result of a TOF experiment are usually counting histograms which can be represented by a vector $\vec{Y}$ of dimension m consisting of count numbers $(y_1, y_2, \ldots, y_m)$. Since the numbers are the result of a counting experiment their uncertainties are not correlated. The covariance matrix $\mathbf{U}_{\vec{Y}}$ of such a histogram contains only diagonal terms and the full covariance information can be stored into a vector $\vec{U}_{\vec{Y}}$. The elements of this vector are the estimates of the variances $\vec{U}_{\vec{Y}} = (u_{y_1}^2, \ldots, u_{y_m}^2)$ of the observed counts $\vec{Y} = (y_1, \ldots, y_m)$. In most cases these estimates of $\vec{U}_{\vec{Y}}$ are based on Poisson statistics.

In the data reduction usually several operations are carried out involving initial counting histograms and a parameter vector $\vec{b}$. The uncertainty on the elements $(b_1, \ldots, b_k)$ introduces a correlated uncertainty component in the uncertainty on the result of the data reduction. The elements of $\vec{b}$ may be correlated with a covariance matrix $\mathbf{V}_{\vec{b}}$ which contains also non-diagonal elements. Such a covariance matrix is by definition symmetric and positive definite. In this case the Cholesky transformation [1] can be applied. The matrix $\mathbf{V}_{\vec{b}}$ can be decomposed as

$$\mathbf{V}_{\vec{b}} = \mathbf{L}_{\vec{b}}\mathbf{L}_{\vec{b}}^T, \tag{20}$$

where $\mathbf{L}_{\vec{b}}$ is a lower triangular matrix of dimension $(k \times k)$ and $\mathbf{L}_{\vec{b}}^T$ denotes the conjugate transpose of $\mathbf{L}_{\vec{b}}$.

Consider now that the data reduction results in a quantity $\vec{Z}$ that is deduced from p counting spectra $(\vec{Y}_1, \ldots, \vec{Y}_p)$ and the parameter vector $\vec{b}$ through the functional relationship $\vec{Z} = f(\vec{Y}_1, \ldots, \vec{Y}_p, \vec{b})$. Each of the spectra has a dimension n. The dimension of the parameter vector is m. Taking into account the specific property of counting spectra and supposing that the relationship f only involves channel-to-channel operations, the covariance matrix of $\vec{Z}$ can be split into an uncorrelated and correlated component:

$$\mathbf{V}_{\vec{Z}} = \underbrace{\sum_{j=1}^{p} \mathbf{D}_f(\vec{Y}_j)\mathbf{U}_{\vec{Y}_j}\mathbf{D}_f^T(\vec{Y}_j)}_{\text{uncorrelated}} + \underbrace{\mathbf{D}_f(\vec{b})\mathbf{V}_{\vec{b}}\mathbf{D}_f^T(\vec{b})}_{\text{correlated}}, \tag{21}$$

The first term is a diagonal matrix containing the uncorrelated component of the covariance matrix of $\vec{Z}$ and can be denoted by $\mathbf{U}_{\vec{Z}}$. Application of the Cholesky transformation on the covariance matrix $\mathbf{V}_{\vec{b}} = \mathbf{L}_{\vec{b}}\mathbf{L}_{\vec{b}}^T$ results in

$$\mathbf{V}_{\vec{Z}} = \mathbf{U}_{\vec{Z}} + \mathbf{D}_f(\vec{b})\mathbf{L}_{\vec{b}}\mathbf{L}_{\vec{b}}^T\mathbf{D}_f^T(\vec{b}). \tag{22}$$

By defining the $\mathbf{S}_{\vec{Z}}(\vec{b})$ matrix of $\vec{b}$ as

$$\mathbf{S}_{\vec{Z}}(\vec{b}) = \mathbf{D}_f(\vec{b})\mathbf{L}_{\vec{b}}, \tag{23}$$

Eq. (22) can be replaced by

$$\mathbf{V}_{\vec{Z}} = \mathbf{U}_{\vec{Z}} + \mathbf{S}_{\vec{Z}}(\vec{b})\mathbf{S}_{\vec{Z}}^T(\vec{b}). \tag{24}$$

Eq. (24) reveals that the full covariance information of $\vec{Z}$ is contained in a vector $\vec{U}_{\vec{Z}}$ with length n and a matrix $\mathbf{S}_{\vec{Z}}$ of dimension (n x m). The former is the sum of all uncorrelated uncertainty components, while the latter represents the contribution of each component resulting in a correlated contribution.

In the previous paragraph the quantity $\vec{Z}$ was derived in one step from a set of spectra based on one functional relationship. The same principles can be applied in case $\vec{Z}$ is obtained from successive operations on spectra. Let us assume that the spectra $\vec{Z}_1$ and $\vec{Z}_2$ have been derived from a set of counting spectra through the relations $\vec{Z}_1 = f_1(\vec{Y}_1, \ldots, \vec{Y}_q, \vec{b}_1)$ and $\vec{Z}_2 = f_2(\vec{Y}_r, \ldots, \vec{Y}_w, \vec{b}_2)$ and that $\vec{Z}$ is given by $\vec{Z} = g(\vec{Z}_1, \vec{Z}_2, \vec{c})$. The covariances of $\vec{Z}_1$ and $\vec{Z}_2$ are given by $\mathbf{V}_{\vec{Z}_1} = \mathbf{U}_{\vec{Z}_1} + \mathbf{S}_{\vec{Z}_1}\mathbf{S}_{\vec{Z}_1}^T$ and $\mathbf{V}_{\vec{Z}_2} = \mathbf{U}_{\vec{Z}_2} + \mathbf{S}_{\vec{Z}_2}\mathbf{S}_{\vec{Z}_2}^T$, respectively. If the counting spectra $\vec{Y}_1, \ldots, \vec{Y}_q, \vec{Y}_r, \ldots, \vec{Y}_w$ and the parameter vectors $\vec{c}$, $\vec{b}_1$, $\vec{b}_2$ are mutually not correlated then the covariance matrix of $\vec{Z}$, with application of GLUP, becomes

$$\mathbf{V}_{\vec{Z}} = \sum_{j=1}^{2} \mathbf{D}_g(\vec{Z}_j)\mathbf{V}_{\vec{Z}_j}\mathbf{D}_g^T(\vec{Z}_j) + \mathbf{D}_g(\vec{c})\mathbf{V}_{\vec{c}}\mathbf{D}_g^T(\vec{c}). \tag{25}$$

The covariance structure of Eq. (25) can only be used if the parameter vectors involved in the functional relationships are mutually independent and each spectrum $\vec{Y}$ is used in only one step. For example, when in the data reduction of a transmission measurement the sample-in and sample-out background are the same, the background correction can not be done separately, but it has to be performed in one single step.

Using the AGS notation of the covariance matrices of $\vec{Z}_1$ and $\vec{Z}_2$ and the Cholesky transformation $\mathbf{V}_{\vec{c}} = \mathbf{L}_{\vec{c}}\mathbf{L}_{\vec{c}}^T$ the covariance matrix $\mathbf{V}_{\vec{Z}}$ is then expressed by

$$\mathbf{V}_{\vec{Z}} = \underbrace{\sum_{j=1}^{2} \mathbf{D}_g(\vec{Z}_j)\mathbf{U}_{\vec{Z}_j}\mathbf{D}_g^T(\vec{Z}_j)}_{\text{uncorrelated}} + \underbrace{\sum_{j=1}^{2} \mathbf{D}_g(\vec{Z}_j)\mathbf{S}_{\vec{Z}_j}(\vec{b_j})\mathbf{S}_{\vec{Z}_j}^T(\vec{b_j})\mathbf{D}_g^T(\vec{Z}_j) + \mathbf{D}_g(\vec{c})\mathbf{L}_{\vec{c}}\mathbf{L}_{\vec{c}}^T\mathbf{D}_g^T(\vec{c})}_{\text{correlated}}. \quad (26)$$

The uncorrelated part of $\mathbf{V}_{\vec{Z}}$ is again a diagonal matrix $\vec{U}_{\vec{Z}}$ while the correlated part can be combined into a new **S**-matrix by using the concatenation

$$\mathbf{S}_{\vec{Z}} = \left[\mathbf{D}_g(\vec{Z}_1)\mathbf{S}_{\vec{Z}_1} \quad \mathbf{D}_g(\vec{Z}_2)\mathbf{S}_{\vec{Z}_2} \quad \mathbf{D}_g(\vec{c})\mathbf{L}_{\vec{c}}\right]. \quad (27)$$

$\mathbf{S}_{\vec{Z}}$ is a matrix with the dimension $(n \times m)$ with $m = k + m_1 + m_2$ where k, m_1 and m_2 are the dimension of the parameter vectors $\vec{c}$, $\vec{b}_1$, $\vec{b}_2$, respectively. Finally, the covariance matrix of $\vec{Z}$ becomes

$$\mathbf{V}_{\vec{Z}} = \mathbf{U}_{\vec{Z}} + \mathbf{S}_{\vec{Z}}\mathbf{S}_{\vec{Z}}^T. \quad (28)$$

The covariance representation in Eq. (28) results in a substantial reduction of storage space when the dimension n of the spectra is much larger than the total number m of correlated uncertainty components. The classical covariance matrix representation requires storage of $n \times (n+1)/2$ numbers. The scheme of Eq. (28) needs only $n \times (m+1)$ numbers. The procedure also ensures that the covariance matrix is always well defined, i.e. symmetric and positive definite. In addition, the correlated and uncorrelated uncertainties are well separated and the impact of each part of the process resulting in a correlated uncertainty can be verified. The procedure, however, requires that all steps in the data reduction process are well understood and described, and that the data reduction starts from spectra which are only subject to uncorrelated uncertainties.

3 Data analysis

3.1 Parameter estimation

The results of a data reduction procedure of TOF data is commonly used for parameter estimation of nuclear reaction models using for example resonance shape analysis codes like REFIT [15]. The estimated parameters can be obtained by a least squares adjustment, that is by minimizing the expression

$$\chi^2(\vec{\theta}) = \left(\vec{Z}_{exp} - \vec{Z}_m(\vec{\theta})\right)^T \mathbf{V}^{-1}_{\vec{Z}_{exp}} \left(\vec{Z}_{exp} - \vec{Z}_m(\vec{\theta})\right), \tag{29}$$

where $\vec{Z}_m(\vec{\theta})$ is the model describing the experimental data $\vec{Z}_{exp}$, with covariance matrix $\mathbf{V}_{\vec{Z}_{exp}}$, and $\vec{\theta}$ are the model parameters. The minimum condition of Eq. (29) is equivalent to a maximum likelihood when the probability distribution of the observable is a normal distribution [16, 17]. The fitted parameters, describing in the best way the experimental data, can be found in an iterative way by searching for the steepest decent. For a linear model the minimum is also found for

$$\vec{\theta} = \left(\mathbf{G}^T_{\vec{\theta}} \mathbf{V}^{-1}_{\vec{Z}} \mathbf{G}_{\vec{\theta}}\right)^{-1} \left(\mathbf{G}^T_{\vec{\theta}} \mathbf{V}^{-1}_{\vec{Z}} \vec{Z}_{exp}\right) , \tag{30}$$

where the sensitivity matrix $\mathbf{G}_{\vec{\theta}}$ has as elements the partial derivatives of $\vec{Z}_{exp}$ with respect to $\vec{\theta}$. The quality of the fit can be verified by comparing the χ^2 per degree of freedom with its expectation value in case the observable follows a normal distribution. For an adequate description of the data this value approaches unity for a large number of degrees of freedom ν [18].

Results of a least squares adjustment might be biased when the model is non-linear and the estimator of the covariance matrix is not well defined, e.g. due to the presence of a correlated normalization component together with a strong scattering of the data around the estimated value (see Becker *et al.* [3] and references therein). This problem, known as Peelle's Pertinent Puzzle (PPP), can be avoided by using only the uncorrelated uncertainty component of $\mathbf{V}_{\vec{Z}_{exp}}$ in the calculation of Eq. (29) and (30) which is easily accessible when the AGS formalism (see Section 2.2) is used in the data reduction.

3.2 Estimation of the total covariance of estimated parameters

If the result of the data reduction is used in parameter estimation by fitting, maintaining the separation of the correlated and uncorrelated uncertainty components in the AGS format (Eq. (28)) is convenient for the determination of the final uncertainty of the fitted parameters. The covariance of the final spectra can be propagated towards the covariance of the fitted parameters by means of conventional uncertainty propagation, Monte Carlo [19] or marginalization [20].

3.2.1 Conventional uncertainty propagation

Suppose that the parameter vector $\vec{\theta}$ with p elements is determined in a least squares fit of a model $F(\vec{\theta})$ to the result of the data reduction $\vec{Z}$. Following conventional uncertainty propagation, the covariance of these estimated parameters is then

$$\mathbf{V}_{\vec{\theta}} = \left(\mathbf{D}^T_F(\vec{\theta}) \mathbf{V}^{-1}_{\vec{Z}} \mathbf{D}_F(\vec{\theta})\right)^{-1} , \tag{31}$$

where the covariance matrix $\mathbf{V}_{\vec{Z}}$ contains both uncorrelated and correlated components. $\mathbf{D}_F(\vec{\theta})$ contains the derivatives of the model with respect to $\vec{\theta}$. If the data vector $\vec{Z}$ consists of a significant number of elements n, then the inversion of its full covariance matrix $\mathbf{V}_{\vec{Z}}$ with dimension $(n \times n)$ in Eq. (31) is computationally very challenging. Using the AGS format of the covariance matrix given in Eq. (28) the inversion of the full covariance matrix can be avoided. Introducing Eq. (28) in Eq. (31) leads to

$$\mathbf{V}_{\vec{\theta}} = \left(\mathbf{D}_F^T(\vec{\theta}) \left(\mathbf{U}_{\vec{Z}} + \mathbf{S}_{\vec{Z}} \mathbf{S}_{\vec{Z}}^T \right)^{-1} \mathbf{D}_F(\vec{\theta}) \right)^{-1} . \tag{32}$$

By using the Woodbury matrix identity [21] the inner term $\left(\mathbf{U}_{\vec{Z}} + \mathbf{S}_{\vec{Z}} \mathbf{S}_{\vec{Z}}^T \right)^{-1}$ can be transformed:

$$\left(\mathbf{U}_{\vec{Z}} + \mathbf{S}_{\vec{Z}} \mathbf{S}_{\vec{Z}}^T \right)^{-1} = \mathbf{U}_{\vec{Z}}^{-1} - \mathbf{U}_{\vec{Z}}^{-1} \mathbf{S}_{\vec{Z}} \left(\mathbf{I} + \mathbf{S}_{\vec{Z}}^T \mathbf{U}_{\vec{Z}}^{-1} \mathbf{S}_{\vec{Z}} \right)^{-1} \mathbf{S}_{\vec{Z}}^T \mathbf{U}_{\vec{Z}}^{-1} , \tag{33}$$

with $\mathbf{I}$ being the identity matrix. On the righthand side of Eq. (33) only the uncorrelated part of the covariance matrix and the term $\left(\mathbf{I} + \mathbf{S}_{\vec{Z}}^T \mathbf{U}_{\vec{Z}}^{-1} \mathbf{S}_{\vec{Z}} \right)$ need to be inverted. The latter is a $(m \times m)$ matrix with m being the number of correlated uncertainty components. The inversion of these two terms can be done significantly more efficient than inverting $\mathbf{V}_{\vec{Z}}$. Inserting Eq. (33) into Eq. (32) gives

$$\mathbf{V}_{\vec{\theta}} = \left(\mathbf{D}_F^T(\vec{\theta}) \mathbf{U}_{\vec{Z}}^{-1} \mathbf{D}_F(\vec{\theta}) - \mathbf{D}_F^T(\vec{\theta}) \mathbf{U}_{\vec{Z}}^{-1} \mathbf{S}_{\vec{Z}} \left(\mathbf{I} + \mathbf{S}_{\vec{Z}}^T \mathbf{U}_{\vec{Z}}^{-1} \mathbf{S}_{\vec{Z}} \right)^{-1} \mathbf{S}_{\vec{Z}}^T \mathbf{U}_{\vec{Z}}^{-1} \mathbf{D}_F(\vec{\theta}) \right)^{-1} . \tag{34}$$

A similar procedure called implicit data covariance method has also been applied in ref. [22].

Eq. (34) can be simplified by introducing $\mathbf{K} = \mathbf{D}_F^T(\vec{\theta}) \mathbf{U}_{\vec{Z}}^{-1} \mathbf{S}_{\vec{Z}}$ which is a $(p \times m)$ matrix:

$$\mathbf{V}_{\vec{\theta}} = \left(\mathbf{D}_F^T(\vec{\theta}) \mathbf{U}_{\vec{Z}}^{-1} \mathbf{D}_F(\vec{\theta}) - \mathbf{K} \left(\mathbf{I} + \mathbf{S}_{\vec{Z}}^T \mathbf{U}_{\vec{Z}}^{-1} \mathbf{S}_{\vec{Z}} \right)^{-1} \mathbf{K}^T \right)^{-1} . \tag{35}$$

By defining the contribution of the uncorrelated uncertainty on the parameter covariance as

$$\mathbf{V}_{\vec{\theta}}^U = \left(\mathbf{D}_F^T(\vec{\theta}) \mathbf{U}_{\vec{Z}}^{-1} \mathbf{D}_F(\vec{\theta}) \right)^{-1} \tag{36}$$

and re-applying the Woodbury matrix identity one obtains

$$\mathbf{V}_{\vec{\theta}} = \mathbf{V}_{\vec{\theta}}^U + \mathbf{V}_{\vec{\theta}}^U \mathbf{K} \left(\mathbf{I} + \mathbf{S}_{\vec{Z}}^T \mathbf{U}_{\vec{Z}}^{-1} \mathbf{S}_{\vec{Z}} - \mathbf{K}^T \mathbf{V}_{\vec{\theta}}^U \mathbf{K} \right)^{-1} \mathbf{K}^T \mathbf{V}_{\vec{\theta}}^U , \tag{37}$$

where the second term is a correction of $\mathbf{V}_{\vec{\theta}}^U$ due to correlated contributions in the data covariance matrix.

3.2.2 Monte Carlo

In case of a Monte Carlo uncertainty propagation, the representation of the correlated uncertainty in terms of the **S**-matrix is very practical. Since each column of the **S**-matrix denotes the sensitivity of the final spectrum to the uncertainty of a specific parameter, one can directly use the **S**-matrix to sample the perturbation of the final spectrum under a first-order Taylor series

expansion assumption. Each column of the matrix is then multiplied with a Gaussian sampled random number. The q^{th} sampled final spectrum $\vec{Z}^{(q)}$ is obtained by

$$\vec{Z}^{(q)} = \vec{Z} + \mathbf{S}_{\vec{Z}}\vec{r}^{(q)}, \tag{38}$$

where $\vec{r}$ is a vector of m Gaussian sampled random numbers. Note that in this approach second order effects are not taken into account. Every sampled spectrum is then used to fit the parameter vector $\vec{\theta}^{(q)}$. The final covariance matrix of the adjusted parameters is obtained using the total covariance theorem of Ref. [19] given by

$$\left(\mathbf{V}_{\vec{\theta}}^{MC}\right)_{(i,j)} = \mathbb{E}\left[\left(\mathbf{V}_{\vec{\theta}}^{U}\right)_{(i,j)}^{(q)}\right] + Cov\left[\theta_i^{(q)}, \theta_j^{(q)}\right] \ . \tag{39}$$

The first term is the average of q covariance matrices due to only the uncorrelated uncertainty component of the data. The second term is a correction of the average taking into account the variation of the fitted parameters due to different samplings.

3.2.3 Marginalization

Using the AGS format of the covariance matrix is also convenient for the application of the marginalization procedure of Ref. [20]. If there is no correlation between the fitted parameters $\vec{\theta}$ and model/experimental parameters $\vec{b}$, such as a normalization, the marginalized covariance matrix is obtained using

$$\begin{aligned}\mathbf{V}_{\vec{\theta}}^{Mar.} &= \underbrace{\left(\mathbf{D}_F^T(\vec{\theta})\mathbf{D}_F(\vec{\theta})\right)^{-1}\mathbf{D}_F^T(\vec{\theta})\left(\mathbf{D}_F(\vec{\theta})\mathbf{V}_{\vec{\theta}}^{U}\mathbf{D}_F^T(\vec{\theta})\right)\mathbf{D}_F(\vec{\theta})\left(\mathbf{D}_F^T(\vec{\theta})\mathbf{D}_F(\vec{\theta})\right)^{-1}}_{\text{uncorrelated}} \\ &+ \underbrace{\left(\mathbf{D}_F^T(\vec{\theta})\mathbf{D}_F(\vec{\theta})\right)^{-1}\mathbf{D}_F^T(\vec{\theta})\left(\mathbf{D}_F(\vec{b})\mathbf{V}_{\vec{b}}\mathbf{D}_F^T(\vec{b})\right)\mathbf{D}_F(\vec{\theta})\left(\mathbf{D}_F^T(\vec{\theta})\mathbf{D}_F(\vec{\theta})\right)^{-1}}_{\text{correlated}},\end{aligned} \tag{40}$$

where $\mathbf{V}_{\vec{b}}$ is the covariance matrix of $\vec{b}$. $\mathbf{D}_F(\vec{\theta})$ and $\mathbf{D}_F(\vec{b})$ contain the derivatives of the model with respect to $\vec{\theta}$ and $\vec{b}$. The parameter uncertainty due to the uncorrelated component of the data covariance matrix if given by $\mathbf{V}_{\vec{\theta}}^{U}$ and defined in Eq. (36). With

$$\mathbf{D}_F(\vec{b})\mathbf{V}_{\vec{b}}\mathbf{D}_F^T(\vec{b}) \approx \mathbf{S}_{\vec{Z}}\mathbf{S}_{\vec{Z}}^T \tag{41}$$

the marginalized covariance matrix is obtained by adding the covariance matrix evaluated after the fit $\mathbf{V}_{\vec{\theta}}^{U}$ and the marginalized correlated uncertainty component:

$$\mathbf{V}_{\vec{\theta}}^{Mar.} \approx \mathbf{V}_{\vec{\theta}}^{U} + \left(\mathbf{D}_F^T(\vec{\theta})\mathbf{D}_F(\vec{\theta})\right)^{-1}\mathbf{D}_F^T(\vec{\theta})\mathbf{S}_{\vec{Z}}\mathbf{S}_{\vec{Z}}^T\mathbf{D}_F(\vec{\theta})\left(\mathbf{D}_F^T(\vec{\theta})\mathbf{D}_F(\vec{\theta})\right)^{-1}. \tag{42}$$

4 The AGS code

The AGS concept of uncertainty propagation (see Section 2.2) has been fully implemented into the AGS code. This code package has been developed at the GELINA time-of-flight facility [8] of the EC-JRC-IRMM as a tool to perform the data reduction of TOF data including a full uncertainty propagation. The data reduction using the AGS code is conceptually broken down into sequential operations. Each operation is performed with a stand-alone program which reads and writes into a common archive file. This archive file does not only contain all spectra and uncertainty components, but also keeps track of the operations, parameters and uncertainties that have been used to produce a specific spectrum. Therefore, all intermediate results of a data reduction procedure are fully documented and can, if needed, be used for further analysis.

The AGS package contains all basic spectrum operations such as arithmetic operations and spectrum manipulations e.g. linear combination of different spectra. In addition, more advanced operations such as dead time correction, least squares background fitting, reading of data histograms as well as of data files in ENDF format [23], conversion from time-of-flight to energy, projection of spectra on different time axes and bin averaging are included. The resulting covariance information can be displayed either in the AGS format ($\mathbf{U}_{\vec{Z}}$ and $\mathbf{S}_{\vec{b}}$ – see Equation (24)) or as a full covariance or correlation matrix. Table 1 and 2 summarize the available commands ordered alphabetically and in categories, respectively.

The AGS code package has been written in the programming language C. Every command is programmed in a separate file using a common set of header files. In these header files common functions such as reading and writing to the archive file are defined. The compilation of the code has been tested under Linux (32 bit) using the GNU Compiler Collection (gcc) and under Windows 7 (64-bit) using Microsoft Visual Studio 10. Note that the AGS code assumes that a *long integer* has the length of 32-bit while a *double* has the length of 64-bit. This means that code can only be compiled on a 64-bit machine if the LLP64 data model is used and not LP64 or ILP64.

In practice, the different steps of a data reduction are best composed into a script such as *bash* under Linux or *batch* scripting under Windows. This allows for an easy way to define input parameters and data paths as well as to rerun a full data reduction procedure. In Section 5 several scripts are given as examples of a full data reduction procedure.

4.1 AGS commands

A typical AGS procedure starts with the creation of an AGS archive file followed by the definition of some experimental parameters (e.g. contents of scalers, dead time coefficient) and the reading in of one or multiple count-rate spectra. Subsequently multiple channel-to-channel operation are carried out on the spectra and the final result including the full covariance information in AGS format is printed out. The archive file stores the results of the operations in a well-defined and structured way, as will be explained in Section 4.2.

Table 1 and Table 2 give an overview of the available AGS commands in alphabetical order and ordered in categories, respectively. In Section 4.2 the common syntax of the AGS commands is explained. The syntax of each command is explained in sections 4.2.1 to 4.2.33.

Table 1: Alphabetic list of AGS commands

Name	Purpose	section
ags_addval	Add a constant to a spectrum	4.2.1
ags_avrg	Division of spectrum by bin widths	4.2.2
ags_divi	Division of two spectra	4.2.3
ags_divval	Division of a spectrum by a constant	4.2.4
ags_dtco	Dead time correction	4.2.5
ags_ener	Conversion of bin boundaries from TOF to energy	4.2.6
ags_exp	Exponential of a spectrum	4.2.7
ags_fit	Fitting a spectrum	4.2.8
ags_fpa	Flight path adjustment	4.2.9
ags_func	Generate spectrum using a function	4.2.10
ags_fxyp	Special spectrum operation	4.2.11
ags_getA	Import a spectrum from an AGS archive file	4.2.12
ags_getE	Import cross section data from an ENDF formated file (file 3) in energy	4.2.13
ags_getET	Import cross section data from an ENDF formated file (file 3) in TOF	4.2.14
ags_getH	Import a histogram (integers)	4.2.15
ags_getW	Import a weighted histogram	4.2.16
ags_getXY	Import a spectrum	4.2.17
ags_getY	Import a histogram	4.2.18
ags_inv	Inverse a spectrum	4.2.19
ags_lico	Linear combination of spectra	4.2.20
ags_lift	Interpolate spectrum on different grid	4.2.21
ags_log	Natural logarithm of a spectrum	4.2.22
ags_mpty	Creation of an empty AGS archive file	4.2.23
ags_mult	Multiplication of two spectra	4.2.24
ags_multval	Multiplication of a spectrum with a constant	4.2.25
ags_putCM	Print covariance matrix	4.2.26
ags_putX	Print spectra and uncertainty components	4.2.27
ags_scan	Print content of AGS archive file	4.2.28
ags_shift	Shift grid boundaries	4.2.29
ags_subval	Subtraction of a constant from a spectrum	4.2.30
ags_title	Print title of AGS archive file	4.2.31
ags_tof	Conversion of bin boundaries from energy to TOF	4.2.32
ags_unav	Multiply spectrum with bin widths	4.2.33

Table 2: List of AGS commands by category

Name	Purpose	section
Creation of AGS library file and data import		
ags_getA	Import a spectrum from an AGS archive file	4.2.12
ags_getE	Import cross section data from an ENDF formated file (file 3) in energy	4.2.13
ags_getET	Import cross section data from an ENDF formated file (file 3) in TOF	4.2.14
ags_getH	Import a histogram (integers)	4.2.15
ags_getW	Import a weighted histogram	4.2.16
ags_getXY	Import a spectrum	4.2.17
ags_getY	Import a histogram	4.2.18
ags_mpty	Creation of an empty AGS archive file	4.2.23
Basic channel-to-channel, arithmetic operations		
ags_addval	Add a constant to a spectrum	4.2.1
ags_divval	Division of a spectrum by a constant	4.2.4
ags_multval	Multiplication of a spectrum with a constant	4.2.25
ags_subval	Subtraction of a constant from a spectrum	4.2.30
Channel-to-channel, spectrum operations		
ags_avrg	Division of spectrum by bin widths	4.2.2
ags_divi	Division of two spectra	4.2.3
ags_dtco	Dead time correction	4.2.5
ags_ener	Conversion of bin boundaries from TOF to energy	4.2.6
ags_exp	Exponential of a spectrum	4.2.7
ags_fit	Fitting a spectrum	4.2.8
ags_fpa	Flight path adjustment	4.2.9
ags_func	Generate spectrum using a function	4.2.10
ags_fxyp	Special spectrum operation	4.2.11
ags_inv	Inverse a spectrum	4.2.19
ags_lico	Linear combination of spectra	4.2.20
ags_lift	Interpolate spectrum on different grid	4.2.21
ags_log	Natural logarithm of a spectrum	4.2.22
ags_mult	Multiplication of two spectra	4.2.24
ags_shift	Shift grid boundaries	4.2.29
ags_tof	Conversion of bin boundaries from energy to TOF	4.2.32
ags_unav	Multiply spectrum with bin widths	4.2.33
Output operations		
ags_putCM	Print covariance matrix	4.2.26
ags_putX	Print spectra and uncertainty components	4.2.27
ags_scan	Print content of AGS archive file	4.2.28
ags_title	Print title of AGS archive file	4.2.31

4.2 Syntax

A typical AGS command starts with the name of the operation (executable), like "ags_lico" in:

```
ags_lico jun1317,A01TOF1,B01FLUX /PAR=1.3,0.78 /ALIAS=COMBO /COV=0.01,-0.005,0.02
```

The first argument always contains the name of the AGS archive file, mostly followed by the names of spectra to be used as operands. In case more than one operand or filenames are present, they should be separated by commas and not by embedded blanks. In the example above the operands are the spectra A01TOF1 and B01FLUX which are stored in the AGS file "jun1317.ags". The other arguments (2^{rd} , 3^{th}, 4^{th}, etc.) are couples made of a literal Qualifier that is preceded by a slash (e.g. /PAR and /ALIAS) and a numeric or literal Value string (e.g. 1.3, 0.78 and COMBO, respectively). An equal sign (or in some cases a blank) is used to separate the Qualifier and the Value string. The Value string consists of one or more literal or numerical values separated by commas. Blanks are not allowed as separators in a Value string. The order of the qualifier/value couples in the command line is not important. Depending on the command, some qualifiers may be mandatory and others may be optional. In case the qualifier is optional, a default value will be assigned to the corresponding parameter that is used in the execution of the command.

In the sections 4.2.1 to 4.2.33 the syntax of each AGS command is defined and explained. Optional qualifiers are put in square brackets [ ] and alternative qualifiers in curly parentheses { .. — .. }. When the command produces one or more spectra as results, the names of the resulting spectra are built by the command. They always start with a prefix which is defined by the number of operations that have been performed before the current operation. The prefix will be an A in case the command is the 1^{st} operation, a B for the 2^{nd} operation, etc.... This prefix will be followed by 2 digits which are always 01 (except in case of *ags_getW*) and further followed by a name inherited from a previous operation, generated by the command or requested using the /ALIAS qualifier. For example, if the command on the top of this page was the 6^{th} operation it will produce a spectrum with a prefix F and full name F01COMBO. The prefix in the archive file is only incremented by operations that produce a new spectrum. For example the command ags_putX only produces an output and no new spectrum. Hence, after the execution of this command the prefix will not be incremented.

When reading vectorized data from ASCII files using *ags_getW*, *ags_getXY* and *ags_getY* it is assumed that data have only uncorrelated uncertainties. Any possible correlation has to be added using AGS commands. In case of *ags_getH* (and under certain conditions also in case of *ags_getW*, *ags_getXY* and *ags_getY* – see command descriptions) the data in the files are assumed to be counting histograms implying that the uncorrelated uncertainty is given by the square root of the counts.

4.2.1 ags_addval

Operation: Adds a constant value a to the spectrum $\vec{Y}$:

$$\vec{Z} = \vec{Y} + a$$

Syntax:

```
ags_addval   <file>,<spect> /VAL=value[,uncertainty] [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /VAL (mandatory)
  value: mandatory
  uncertainty: optional
- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Examples:

- Assuming the second operation in the AGS archive file "archive" is

  ```
  ags_addval archive,A01SIN /VAL=1.,0.01 /ALIAS=NET /DEBUG=0
  ```

  the spectrum "B01NET" is added to the AGS archive file "archive".
- Assuming the third operation in the AGS archive file "archive" is

  ```
  ags_addval archive,A01SIN /VAL=1.,0.01 /DEBUG=0
  ```

  the spectrum "C01SIN" is added to the AGS archive file "archive".

4.2.2 ags_avrg

Operation: Divides each value of $\vec{Y}$ by its bin width, i.e. difference of the X-bounds. The division by a value specified by the qualifier /CMSC is optional.
Syntax:

```
ags_avrg   <file>,<spect> [/CMSC=position] [/ALIAS=..] [/DEBUG=..].
```

Operands: AGS spectrum <spect>

Qualifiers:

- /CMSC (optional) Position of a scaler used for normalization (usually Central Monitor SCaler). The scaler values are defined during the reading of the initial spectrum (using e.g. ags_getH or ags_getW)
- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Examples:

- Assuming the second operation in the AGS archive file "archive" is

  ```
  ags_avrg archive,A01SIN /CMSC=1 /ALIAS=NET /DEBUG=0
  ```

  the spectrum "B01NET" is added to the AGS archive file "archive".
- Assuming the third operation in the AGS archive file "archive" is

  ```
  ags_avrg archive,A01SIN /CMSC=1 /DEBUG=0
  ```

  the spectrum "C01SIN" is added to the AGS archive file "archive".

4.2.3 ags_divi

Operation: Divides spectrum $\vec{Y}_1$ by spectrum $\vec{Y}_2$:

$$\vec{Z} = \frac{\vec{Y}_1}{\vec{Y}_2}$$

Division by zero gives zero.
Syntax:

```
ags_divi   <file>,<spect1>,<spect2> [/ALIAS=..] [/DEBUG=..]
```

Operands: two ags spectra:
<spect1> : numerator
<spect2> : denominator
Qualifiers:

- /ALIAS (optional) Name to be given to the resulting spectrum (Default: DIVI).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the numerator.
Examples:

- Assuming the third operation in the AGS archive file "archive" is

  ```
  ags_divi archive,A01SIN,B01SOUT /ALIAS=NET /DEBUG=0
  ```

  the spectrum "C01NET" is added to the AGS archive file "archive".

- Assuming the third operation in the AGS archive file "archive" is

  ```
  ags_divi archive,A01SIN,B01SOUT /DEBUG=0
  ```

  the spectrum "C01DIVI" is added to the AGS archive file "archive".

4.2.4 ags_divval

Operation: Divides the spectrum $\vec{Y}$ by a constant a:

$$\vec{Z} = \frac{\vec{Y}}{a}$$

Syntax:

```
ags_divval   <file>,<spect> /VAL=value[,uncertainty] [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /VAL (mandatory)
  value: mandatory
  uncertainty: optional
- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Example:

- Assuming the second operation in the AGS archive file "archive" is

  ```
  ags_divval archive,A01SIN /VAL=2.,0.01 /ALIAS=NET /DEBUG=0
  ```

  the spectrum "B01NET" is added to the AGS archive file "archive".
- Assuming the second operation in the AGS archive file "archive" is

  ```
  ags_divval archive,A01SIN /VAL=2.,0.01 /DEBUG=0
  ```

  the spectrum "C01SIN" is added to the AGS archive file "archive".

4.2.5 ags_dtco

Operation: Calculates the dead time correction factor and applies this correction on a spectrum.
Syntax:

```
 ags_dtco   <file>,<spect> /BASE=<b1>,..,<bn> /DTIME=dt1,..,dtn /DTUNC=.. /LPSC=..
/TYPE=.. /COEFF=.. [/L_OR_D=..] [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /BASE (mandatory) Names of n spectra ($1 \leq n \leq 7$) holding all events to be considered in the dead time calculation.
- /DTIME (mandatory) n values of the dead time for the /BASE spectra
- /LPSC (mandatory) Linac Pulse SCaler position in scaler sequence associated to the spectrum during read-in (total number of bursts).
- /DTUNC (mandatory) Uncertainty on dead times
- /TYPE (mandatory) Type of dead time correction =1 or 2 (see additional information).
- /COEFF (mandatory if /TYPE=2) Coefficient for the type 2 correction. It is the relative standard deviation of the beam intensity variation.
- /L_OR_D (optional) L or D indicating the format of the data (long or double). Default is long.
- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: Dead time corrected spectra with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand. The spectrum, named CSUM, contains the cumulative sum of the counts in the spectra specified in the /BASE qualifier. The spectrum, named DTCF, is the dead time correction factor that was applied.
Examples:

- Assuming the fourth operation in the AGS archive file "archive" is

  ```
  ags_dtco archive,C01SIN /BASE=A01BAS /DTIME=2050 /DTUNC=5. /TYPE=1 /LPSC=1 /ALIAS=DEAD
  ```

  the spectra "D01DEAD", "D01CSUM" and "D01DTCF" are added to the AGS archive file "archive". For the dead time correction Eq. (16) is used.

- Assuming the fourth operation in the AGS archive file "archive" is

```
ags_dtco archive,C01SIN /BASE=A01BAS /DTIME=2050 /DTUNC=5. /TYPE=2 /LPSC=1 /COEFF=0.05
```

the spectra "D01SIN", "D01CSUM" and "D01DTCF" are added to the AGS archive file "archive". For the dead time correction Eq. (17) is used. This correction accounts for a 5% variation in the beam intensity.

Additional information: The dead time correction in AGS is based on Eq. (1) and Eq. (10) of the ref. [13] for /TYPE=1 and /TYPE=2, respectively. If /TYPE=1 then:

$$N_c(i) = N_0(i) / \left(1 - \sum_{j=i_0}^{i-1} N_0(j)/N_b \right) \ ,$$

where $N_c(i)$ and $N_0(i)$ are dead time corrected and uncorrected counts in channel i. N_b is the number of bursts. The dead time is given in channels by the difference $I - I_0$. If /TYPE=2 then:

$$N_c(i) = N_b \left\{ -\ln \left(1 - \frac{N_0(i)/N_b}{\left[1 - \sum_{j=i_0}^{i-1} N_0(j)/N_b \right]} \right) \Bigg/ \left[1 - \sigma \ \tanh \left(\sigma \sum_{j=i_0}^{i-1} N_c(j)/N_b \right) \right] \right\} \ ,$$

where σ is the relative standard deviation of the beam intensity specified by /COEFF.

Note that since the dead time correction of channel i depends on the counts in other channels the dead time correction is by definition not a channel-to-channel operation.

4.2.6 ags_ener

Operation: Converts the X-bounds ($\vec{X}_L$ and $\vec{X}_H$) from a time scale to an energy scale using the flight path length specified in the qualifier /FPATH. Above 5 eV the calculation is fully relativistic, below 5 eV a cubic expansion in $(v/c)^2$ up to the third order is applied. The contents of channels with a time-of-flight corresponding to a velocity larger than c are set to 0 and the energy is set to -1.
Syntax:

```
ags_ener   <file>,<spect> /FPATH=.. [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /FPATH (mandatory) Length in meters of the neutron flight path
- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Examples:

- Assuming the second operation in the AGS archive file "archive" is

  ```
  ags_ener archive,A01SIN /FPATH=33.2 /ALIAS=ENY /DEBUG=0
  ```

  the spectrum "B01ENY" is added to the AGS archive file "archive".

- Assuming the third operation in the AGS archive file "archive" is

  ```
  ags_ener archive,A01SIN /FPATH=33.2 /DEBUG=0
  ```

  the spectrum "C01SIN" is added to the AGS archive file "archive".

4.2.7 ags_exp

Operation: Calculates the exponential of a spectrum $\vec{Y}$:

$$\vec{Z} = e^{\vec{Y}}$$

Syntax:

```
ags_exp   <file>,<spect> [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Examples:

- Assuming the second operation in the AGS archive file "archive" is

  ```
  ags_exp archive,A01SIN /ALIAS=NET /DEBUG=0
  ```

  the spectrum "B01NET" is added to the AGS archive file "archive".

- Assuming the third operation in the AGS archive file "archive" is

  ```
  ags_exp archive,A01SIN /DEBUG=0
  ```

  the spectrum "C01SIN" is added to the AGS archive file "archive".

4.2.8 ags_fit

Operation: Fits parts of a spectrum with a non-linear function by χ^2-minimization.
Syntax:

```
ags_fit   <file>,<spect> /FUN=.. {/GROUPS=..|ZONES=..}
           /GUESS=.. [/FIX=..] [/LIKE=..] [/NITER=..]
           [/CHIDIF=..] [/UNC=..] [/VERBOSE=..] [/ALIAS=..] [/POINTS=..]  [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /FUN (mandatory) Character identifier of fitting function. The available functions are given in Table 3.
- /GROUPS (optional, supersedes /ZONES) defines the part of the spectrum that is used for the fit. The regions are defined by lower ($x_{L,i}$) and higher ($x_{H,i}$) boundaries on the X-values, which are defined by:

  $$x_{L,1}, x_{H,1}, K_1, x_{L,2}, x_{H,2}, K_2, \ldots$$

  where $x_{L,i}$ and $x_{H,i}$ are floating point values and K_i is a positive integer.
  If $K_i = 0$ then all channels with $x_{L,i} < x < x_{H,i}$ are averaged to one observation.
  If $K_i = 1$ then all channels in the group are taken individually.
  if $K_i = n$ with $n > 1$ then the channels in the group i are averaged by subgroups of n channels.
- /ZONES (mandatory in absence of /GROUPS) Defines the part of the spectrum that is used for the fit. The regions are defined by lower ($ch_{L,i}$) and higher ($x_{H,i}$) boundaries on the channel number (ch), which are defined by:

  $$ch_{L,1}, ch_{H,1}, K_1, ch_{L,2}, ch_{H,2}, K_2 \ldots$$

  where $ch_{L,i}$ and $ch_{H,i}$ are integer zone limits and K_i is a positive integer defined like above.
- /POINTS (optional) Add additional data points for the fit. The data are read from a data file, in the form /POINTS=filename,c_1,c_2,c_3, where 'filename' is the name of the file in the current directory containing columns with data with (x, y, u_y) in the columns (c_1, c_2, c_3). If c_3 is not given, u_y is taken equal to 1. Lines starting with a '#' are not read.
- /GUESS (mandatory) Starting values of the fitting parameters. The code expects the correct number of parameters depending on the chosen function. Note that in case of /FUN=POLY the degree of the polynomial is chosen according to the number of given parameters.
- /FIX (optional) The indexes of parameters which are kept fixed and are not adjusted. The index numbering starts with 1.
- /LIKE (optional) The name of the target spectrum that defines the x-values for which the analytical function will be computed using the final parameters after the adjustment. By default, the x-values of the operand are taken.

- /NITER (optional) Number of maximum iterations (default: 50)
- /CHIDIF (optional) Maximum difference of the χ^2 at which the iteration will be stopped (default: 0.001)
- /UNC (optional) Additional fitting/output option:
  If /UNC=CHI: the uncertainties of the data points is multiplied by χ^2 and the fit is redone.
- /VERBOSE (optional) If /VERBOSE=1, the data, parameters and covariance matrices are printed on the screen.
- /ALIAS (optional) Name to be given to the spectrum that is calculated with the final parameters. If not present, no spectra will be added to the file.
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand. A spectrum with the name RAWDAT containing the input spectral data (condensed by /GROUPS). The fitted parameters are stored in a spectrum with name FITPAR. A spectrum with the name FIT containing the expansion of the fitted function on target X-bounds.
Example: Assuming the fourth operation in the AGS archive file "archive" is

```
ags_fit archive,C01SIN \
       /ALIAS=BKGIN \
       /FUN=E3P0 \
       /FIX=3,4,6,7,8,9,10 \
       /GUESS=8.5e-8,1.88818e-6,-3.1e-5,0.,2.91358e-7,-1.9e-7,0.,5.6e-8,-4.00e-6,0  \
       /ZONES=19474,19839,4,31141,31388,8,33794,33820,0
```

the spectra "D01BKGIN", "D01RWADAT" and "D01FIT" are added to the AGS archive file "archive".
Additional information: Only the diagonal of the covariance matrix is considered in the χ^2-minimization, which is justified for most applications.

Table 3: List of available functions

/FUN=..		
POLY	$\sum_{k\geq 1} p_k\ x^{k-1}$	
E0P0	p_1	
E1P0	$p_1 + p_2 e^{p_3(x+p_4)}$	
E2P0	$p_1 + \sum_{k=0}^{1} p_{(3k+2)} e^{p_{(3k+3)}\left(x+p_{(3k+4)}\right)}$	
E3P0	$p_1 + \sum_{k=0}^{2} p_{(3k+2)} e^{p_{(3k+3)}\left(x+p_{(3k+4)}\right)}$	
E0P1	p_1	$+ p_2 \left(x+p_4\right)^{p_3}$
E1P1	$p_1 + p_2 e^{p_3(x+p_4)}$	$+ p_5 \left(x+p_7\right)^{p_6}$
E2P1	$p_1 + \sum_{k=0}^{1} p_{(3k+2)} e^{p_{(3k+3)}\left(x+p_{(3k+4)}\right)}$	$+ p_8 \left(x+p_{10}\right)^{p_9}$
E3P1	$p_1 + \sum_{k=0}^{2} p_{(3k+2)} e^{p_{(3k+3)}\left(x+p_{(3k+4)}\right)}$	$+ p_{11} \left(x+p_{13}\right)^{p_{12}}$
E0P2	p_1	$+ \sum_{j=0}^{1} p_{(3j+2)} \left(x+p_{(3j+4)}\right)^{p_{(3j+3)}}$
E1P2	$p_1 + p_1 e^{p_2(x+p_3)}$	$+ \sum_{j=1}^{2} p_{(3j+2)} \left(x+p_{(3j+4)}\right)^{p_{(3j+3)}}$
E2P2	$p_1 + \sum_{k=0}^{1} p_{(3k+2)} e^{p_{(3k+3)}\left(x+p_{(3k+4)}\right)}$	$+ \sum_{j=2}^{3} p_{(3j+2)} \left(x+p_{(3j+4)}\right)^{p_{(3j+3)}}$
E3P2	$p_1 + \sum_{k=0}^{2} p_{(3k+2)} e^{p_{(3k+3)}\left(x+p_{(3k+4)}\right)}$	$+ \sum_{j=3}^{4} p_{(3j+2)} \left(x+p_{(3j+4)}\right)^{p_{(3j+3)}}$

4.2.9 ags_fpa

Operation: Performs a transformation of variables to change the flight path length and includes the re-binning of a spectrum to the time-boundaries of a target spectrum. The transformation is based on:

$$\frac{L_1}{t_{m,1}} = \frac{L_2}{t_{m,2}}$$

This operation should only be applied on raw (weighted) data.
Syntax:

```
ags_fpa   <file>,<spect> /LIKE=.. /FPATH1=.. /FPATH2=.. [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /FPATH1 (mandatory) Fligth path length of the original spectrum.
- /FPATH2 (mandatory) Fligth path length of the target spectrum.
- /LIKE (mandatory) Target spectrum defining the X-values to rebin the original spectrum after the transformation.
- /ALIAS (optional) Name to be given to the imported spectra (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Example: Assuming the second operation in the AGS archive file "archive" is

```
ags_fpa archive,A01SIN /FPATH1=25.0 /FPATH1=26.1 /LIKE=A01SIN /DEBUG=0
```

the spectrum "B01SIN" is added to the AGS archive file "archive".

4.2.10 ags_func

Operation: Generates a spectrum using an analytical function and the input parameter vector $\vec{p}$.
Syntax:

```
ags_func   <file> /LIKE=.. /ALIAS=.. /FUN=.. /PAR=.. [/COV=..] [/DEBUG=..]
```

Operands: none
Qualifiers:

- /LIKE (mandatory) The target spectrum defining the X-values that are used to calculate the analytical function.
- /ALIAS (mandatory) Name to be given to the result.
- /FUN (mandatory) Character identifier of the function. The available functions are given in Table 3.
- /PAR (mandatory) Vector of n parameters $\vec{p} = (p_1, p_2, \ldots, p_n)$. The code expects the correct number of parameters depending on the chosen function. Note that in case of /FUN=POLY the degree of the polynomial is chosen according to the number of given parameters.
- /COV (optional) Covariance matrix $\mathbf{V}_{\vec{p}}$ of the parameters. The number of covariance terms must be either equal to n (diagonal covariance) or to $n(n+1)/2$ (full covariance stored half-column wise). Half-column wise means:

  $$/COV = (\mathbf{V}_{\vec{p}})_{(1,1)}, (\mathbf{V}_{\vec{p}})_{(1,2)}, (\mathbf{V}_{\vec{p}})_{(2,2)}, (\mathbf{V}_{\vec{p}})_{(1,3)}, (\mathbf{V}_{\vec{p}})_{(2,3)}, (\mathbf{V}_{\vec{p}})_{(3,3)}, \ldots, (\mathbf{V}_{\vec{p}})_{(n,n)}$$

  If there is no uncertainty on the parameters /COV may be omitted altogether.
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier.
Example: Assuming the second operation in the AGS archive file "archive" is

```
ags_func archive, /LIKE=A01SIN  /FUN=E1P0 \
             /PAR=1.0E-3,0.0046,6E-6,0.0 \
             /COV=1.0E-8,0.0,0.0,0.0     \
             /DEBUG=0
```

the spectrum "B01SFTH" is added to the AGS archive file "archive".

4.2.11 ags_fxyp

Operation: Performs user defined channel-to-channel operations using a set of input spectra $(\vec{Y}_1, \ldots, \vec{Y}_n)$ and parameters defined in $\vec{p}$. The input spectra should all have the same X-boundaries. The operations are defined in the header file own.h.
Syntax:

```
ags_fxyp   <file>,<sp1>,<sp2>,.. /OWN=.. [/PAR=..] [COV=..] [/ALIAS=..] [/DEBUG=..]
```

Operands: ags spectra <sp1>,<sp2>, ..
Qualifiers:

- /OWN (mandatory) Identifier of the function.
- /PAR (mandatory If required by the function) Vector of n parameters $\vec{p} = (p_1, p_2, \ldots, p_n)$.
- /COV (optional) Covariance matrix $\mathbf{V}_{\vec{p}}$ of the parameters. The number of covariance terms must be either equal to n (diagonal covariance) or to $n(n+1)/2$ (full covariance stored half-column wise). Half-column wise means:

  $$/COV = (\mathbf{V}_{\vec{p}})_{(1,1)}, (\mathbf{V}_{\vec{p}})_{(1,2)}, (\mathbf{V}_{\vec{p}})_{(2,2)}, (\mathbf{V}_{\vec{p}})_{(1,3)}, (\mathbf{V}_{\vec{p}})_{(2,3)}, (\mathbf{V}_{\vec{p}})_{(3,3)}, \ldots, (\mathbf{V}_{\vec{p}})_{(n,n)}$$

  If there is no uncertainty on the parameters /COV may be omitted.
- /ALIAS (mandatory) Name to be given to the resulting spectrum.
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier.
Example: Assuming the fifth operation in the AGS archive file "archive" is

```
ags_fxyp archive,A01SIN,B01SOUT,C01BKGIN,D01BKGOUT \
                     /OWN=c02 /PAR=1.0 /COV=0.0009 \
                     /ALIAS=TX /DEBUG=1
```

the spectrum "E01TX" is added to the AGS archive file "archive".
Additional information: The user can define channel to channel operations in a function that is specified in the file own.h. as own_cX, where X is the number of the function. The functions are then selected using the qualifier /OWN=cX. Up to 30 different functions can be programmed. The functions c01, c02 and c03 are predefined as:

- **c01:** calculation of the transmission $\vec{T}$ using:

  $$\vec{T} = \frac{\vec{Y}_1 - \vec{Y}_3}{\vec{Y}_2 - \vec{Y}_3} ,$$

  where $\vec{Y}_{1,2,3}$ are input spectra.

- **c02:** calculation of the transmission $\vec{T}$ using:

$$\vec{T} = \frac{\vec{Y}_1 - p_1\vec{Y}_3}{\vec{Y}_2 - p_1\vec{Y}_4} \ ,$$

  where p_0 is an input parameter and $\vec{Y}_{0,1,2,3}$ are input spectra.

- **c03:** calculation of the background $\vec{B}$ using:

$$\vec{B}_1 = p_1 + p_2\left(\vec{Y}_1 - p_3\right) + p_4\vec{Y}_2\left[\left(\vec{Y}_3 - p_5\right) - \left(\vec{Y}_1 - p_3\right)\right] \ ,$$

  where $p_{1,2,3,4,5}$ are input parameter and $\vec{Y}_{1,2,3}$ are input spectra.

4.2.12 ags_getA

Operation: Imports a spectrum and its uncertainty components from an external AGS archive file.
Syntax:

```
ags_getA   <file> /FROM=<ext.file>,<spec> /ALIAS=.. [/LIKE=..] [/DEBUG=..]
```

Operands: none
Qualifiers:

- /FROM (mandatory) Name of the external AGS archive file <ext.file> (max 256 char, without suffix .ags), and spectrum name <spec>
- /ALIAS (mandatory) Name to be given to the resulting spectrum.
- /LIKE (optional) Name of the target spectrum from which the X-bounds will be taken
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier.
Example: Assuming the first operation in the AGS archive file "archive" is

```
ags_getA archive /FROM=old_archive,L01NTX /ALIAS=TX
```

the spectrum "A01TX" is added to the AGS archive file "archive".

4.2.13 ags_getE

Operation: Imports cross section data from an ENDF/B file (ASCII) and interpolates them on an existing energy grid.
Syntax:

```
ags_getE   <file> /FROM=<ENDF_file>,MAT,MT /ALIAS=.. [/LIKE=..] [/DEBUG=..]
```

Operands: none <spect>
Qualifiers:

- /FROM (mandatory) Full name of the ENDF file, followed by the MAT number (material) and MT number (reaction) of the evaluation.
- /ALIAS (mandatory) Name to be given to the resulting spectrum.
- /LIKE (optional) Name of the target spectrum from which the energy bounds will be taken.
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier and with the same Xbounds as the /LIKE target spectrum. In absence of a target spectrum, the result has the same single point X data (energy data) as the ENDF file.
Example: Assuming the second operation in the AGS archive file "archive" is

```
ags_getE archive /FROM=endfB10,,525,1 /LIKE=A01CAPT /ALIAS=B10NT /DEBUG=1
```

the spectrum "B01B10NT" is added to the AGS archive file "archive".

4.2.14 ags_getET

Operation: Imports cross section data from an ENDF/B file (ASCII) and interpolates them on an existing time-of-flight grid. Above 5 eV the calculation is fully relativistic, below 5 eV a cubic expansion in $(v/c)^2$ up to the third order is applied. The contents of channels with a time-of-flight corresponding to a velocity larger than c are set to zero.
Syntax:

```
ags_getET   <file> /FROM=<ENDF_file>,MAT,MT /ALIAS=.. /FPATH=.. [/LIKE=..] [/DEBUG=..]
```

Operands: none
Qualifiers:

- /FROM (mandatory) Full name of the ENDF file, followed by the MAT number (material) and MT number (reaction) of the evaluation.
- /FPATH (mandatory) Length of the flight path (in meter).
- /ALIAS (mandatory) Name to be given to the resulting spectrum.
- /LIKE (mandatory) Name of the target spectrum from which the X-bounds will be taken.
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier.
Example: Assuming the second operation in the AGS archive file "archive" is

```
ags_getET archive /FROM=endfB10,525,1 /LIKE=A01CAPT /ALIAS=B10NT /FPATH=15.2 /DEBUG=1
```

the spectrum "B01B10NT" is added to the AGS archive file "archive".

4.2.15 ags_getH

Operation: Imports a counting histogram from an ASCII file (.HST). The file consists of counts in the form of one column. The uncertainty is based on Poisson statistics, i.e. the square root of counts.
Syntax:

```
ags_getH   <file> /FROM= <.....[hst]> /SCALERS=.. [/XCONV= ..]
  {/ACCORD=..| /LIKE=..} [/ALIAS=..] [/DEBUG=..]
```

Operands: none
Qualifiers:

- /FROM (mandatory) Name of the ASCII file (suffix .hst or HST) holding the histogram data.
- /SCALERS (mandatory) Value of up to 32 scalers associated with the histogram
- /ACCORD (mandatory if /LIKE not present) Accordion pattern used as: $z_1, cf_1, z_2, cf_2, \ldots$, where z_i and cf_i are length and compression factor of zone i, respectively.
- /LIKE (optional) Name of the target spectrum from which the X-bounds will be taken. When /LIKE is present, /XCONV and /ACCORD are ineffective.
- /ALIAS (optional) Name to be given to the imported spectra (Default: HIST).
- /XCONV (mandatory if /LIKE not present) Base channel width.
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier.
Example: Assuming the first operation in the AGS archive file "archive" is

```
ags_getH archive /FROM=sample.hst /ALIAS=SIN \
     /ACCORD=1024,8,1024,4,4096,2,5120,4,5120,8,5120,16,5120,32,5120,64,5120,128 \
     /SCALERS=97570690,20164978 /XCONV=0.5
```

the spectrum "A01SIN" is added to the AGS archive file "archive".

4.2.16 ags_getW

Operation: Import vectorized data from the standard input stream with the format:

$$\left[\vec{i}, \vec{Y}_1, \vec{Y}_2, \vec{U}_{\vec{Y}_2}\right] \ ,$$

where $\vec{i}$ is a vector of values (e.g. row numbers) which are skipped in the reading. $\vec{Y}_1$ is a histogram. $\vec{Y}_2$ is a data vector with the uncorrelated variance vector $\vec{U}_{\vec{Y}_2}$. The uncertainty of $\vec{Y}_1$ is calculated based on Poisson statistics, i.e. the square root of counts.
Syntax:

```
cat .. | ags_get_W   <file> /SCALERS=.. [/GLOBAL=<const1>,<const2>,..]
       [/LOCAL=..] {/ACCORD=.. | /LIKE=..} [/XCONV=..] [/ALIAS=..] [/DEBUG=..]
```

Operands: none
Qualifiers:

- /SCALERS (mandatory) Value of up to 32 scalers associated with the histogram
- /ACCORD (mandatory if /LIKE not present) Accordion pattern used as: $z_1, cf_1, z_2, cf_2, \ldots$, where z_i and cf_i are length and compression factor of zone i, respectively.
- /XCONV (mandatory if /LIKE not present) Base channel width.
- /LIKE (mandatory if /ACCORD and /XCONV are not present) Name of the target spectrum from which the X-bounds will be taken
- /ALIAS (optional) Name to be given to the imported spectra.
- /GLOBAL (optional) Length in meters of the neutron flight path
- /LOCAL (optional) Name to be given to the resulting spectrum.
- /ALIAS (optional) Name to be given to the imported spectra (Default: WSUM).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: Two spectrum with a new prefix. <Prefix>01WSUM contains the unweighted spectrum. <Prefix>02WSUM contains the weighted spectrum.
Example: Assuming the first operation in the AGS archive file "archive" is

```
cat AGS_1.ags | ags_getW archive \
     /ACCORD=1024,8,1024,4,4096,2,5120,4,5120,8,5120,16,5120,32,5120,64,5120,128 \
     /SCALERS=97570690,20164978 /XCONV=0.5 /DEBUG=-1
```

the spectrum "A01WSUM" is added to the AGS archive file "archive".

4.2.17 ags_getXY

Operation: Imports vectorized data from an ASCII file containing data with the format:

$$\left[\vec{X}_L, \vec{X}_H, \vec{Y}, \vec{u}_{\vec{Y}}\right] \ ,$$

where $\vec{X}_L$, $\vec{X}_H$, $\vec{Y}$ and $\vec{u}_{\vec{Y}}$ are lower and higher bounds, spectrum and uncertainty, respectively.
Syntax:

```
ags_getXY   <file> /FROM=.. [/LIKE=..] [{/SCALERS=..|/SCALERFILE=..}] [/ALIAS=..]
       [/UNC=SQRT] [/DEBUG=..]
```

Operands: none
Qualifiers:

- /FROM (mandatory) Full name of the datafile
- /LIKE (optional) Name of the target spectrum from which the X-bounds will be taken. If present, the first two columns of the data file are ignored.
- /SCALERS (optional) Value of up to 32 scalers associated with the data
- /SCALERFILE (optional) Name of the file containing scalers
- /ALIAS (optional) Name to be given to the resulting spectrum (Default: ASC).
- /UNC (optional) Uncertainty of the data is assumed to be: $\vec{u}_{\vec{Y}} = \sqrt{\vec{Y}}$. The last column of the data file is ignored.
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or with the default name ASC.
Example: Assuming the first operation in the AGS archive file "archive" is

```
ags_getXY archive /FROM=data.txt /SCALERFILE=scal.txt /ALIAS=SIN
```

the spectrum "A01SIN" is added to the AGS archive file "archive".

4.2.18 ags_getY

Operation: Imports vectorized data from an ASCII file with the format:

$$\left[\vec{Y}, \vec{u}_{\vec{Y}}\right] \ ,$$

where $\vec{Y}$ and $\vec{u}_{\vec{Y}}$ are spectrum and uncertainty, respectively.
Syntax:

```
ags_getY   <file> /FROM= <....> {/SCALERS= ..|/SCALERFILE=.. } [/XCONV= ..]
       {/ACCORD=..| /LIKE=..} [/ALIAS=..] [/UNC=SQRT] [/DEBUG=..]
```

Operands: none
Qualifiers:

- /FROM (mandatory) Full name of the datafile
- /SCALERS (mandatory if /SCALERFILE not present) Value of up to 32 scalers associated with the data
- /SCALERFILE (mandatory if /SCALER not present) Name of the file containing scalers
- /ACCORD (mandatory if /LIKE not present) Accordion pattern used as: $z_1, cf_1, z_2, cf_2, \ldots$, where z_i and cf_i are length and compression factor of zone i, respectively.
- /LIKE (optional) Name of the target spectrum from which the X-bounds will be taken. if /LIKE is present, /XCONV and /ACCORD are ineffective.
- /ALIAS (optional) Name to be given to the imported spectra (Default: ASC).
- /XCONV (mandatory if /LIKE not present) Base channel width.
- /UNC (optional) Uncertainty of the data is assumed to be: $\vec{u}_{\vec{Y}} = \sqrt{\vec{Y}}$. The last column of the data file is ignored.
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier.
Example: Assuming the second operation in the AGS archive file "archive" is

```
ags_getY archive /FROM=data.txt /LIKE=A01SIN /ALIAS=SOUT
```

the spectrum "B01SOUT" is added to the AGS archive file "archive".

4.2.19 ags_inv

Operation: Calculates the inverse of a spectrum $\vec{Y}$:

$$\vec{Z} = \vec{Y}^{-1}$$

Syntax:

```
ags_inv   <file>,<spect> [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Example: Assuming the second operation in the AGS archive file "archive" is

```
ags_inv archive,A01SIN /ALIAS=NET /DEBUG=0
```

the spectrum "B01NET" is added to the AGS archive file "archive".

4.2.20 ags_lico

Operation: Performs a channel-to-channel linear combination of n spectra ($\vec{Y_1}$, ..., $\vec{Y_n}$) using a parameter vector $\vec{p}$ with dimension n, i.e., (p_1, ..., p_n):

$$\vec{Z} = \sum_n p_n \vec{Y_n}$$

Syntax:

```
ags_lico   <file>,<sp1>,<sp2>,.. /PAR=.. [/COV=..] [/ALIAS=..] [/DEBUG=..]
```

Operands: ags spectra <sp1>, <sp2>, ..
Qualifiers:

- /PAR (mandatory) Vector of n parameters $\vec{p} = (p_1, p_2, \ldots, p_n)$
- /COV (optional) Covariance matrix $\mathbf{V}_{\vec{p}}$ of the parameters. The number of covariance terms must be either equal to n (diagonal covariance) or to $n(n+1)/2$ (full covariance stored half-column wise). Half-column wise means:

  $$/COV = (\mathbf{V}_{\vec{p}})_{(1,1)}, (\mathbf{V}_{\vec{p}})_{(1,2)}, (\mathbf{V}_{\vec{p}})_{(2,2)}, (\mathbf{V}_{\vec{p}})_{(1,3)}, (\mathbf{V}_{\vec{p}})_{(2,3)}, (\mathbf{V}_{\vec{p}})_{(3,3)}, \ldots, (\mathbf{V}_{\vec{p}})_{(n,n)}$$

  If there is no uncertainty on the parameters /COV may be omitted altogether.
- /ALIAS (optional) Name to be given to the resulting spectrum (Default : LICO).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or with the default name LICO.
Example:

- Assuming the third operation in the AGS archive file "archive" is

  ```
  ags_lico archive,A01SIN,B01SIN2 /PAR=1,1 /COV=0.0001,0.0,0.0001 /ALIAS=SUM /DEBUG=0
  ```

  the spectrum "C01SUM" is added to the AGS archive file "archive".
- Assuming the third operation in the AGS archive file "archive" is

  ```
  ags_lico archive,A01SIN,B01SIN2 /PAR=1,1 /COV=0.0001,0.0,0.0001 /DEBUG=0
  ```

  the spectrum "C01LICO" is added to the AGS archive file "archive".

4.2.21 ags_lift

Operation: Performs a linear interpolation (linear on the two axes) based on the X-boundaries of a target spectrum.
Syntax:

```
ags_lift   <file>,<spect> /LIKE=.. [/ALIAS=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /LIKE (mandatory) Name of the target spectrum defining the X-boundaries that are used for the interpolation.
- /ALIAS (optional) Name to be given to the lifted spectrum (Default: LIFT)

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Example: Assuming the third operation in the AGS archive file "archive" is

```
ags_lift archive,A01SIN /LIKE=B01SIN2 /ALIAS=INT
```

the spectrum "C01INT" is added to the AGS archive file "archive".

4.2.22 ags_log

Operation: Calculates the natural logarithm of a spectrum $\vec{Y}$.

$$\vec{Z} = \ln \vec{Y}$$

Syntax:

```
ags_log   <file>,<spect> [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Example:

- Assuming the second operation in the AGS archive file "archive" is

  ```
  ags_log archive,A01SIN /DEBUG=0
  ```

  the spectrum "B01SIN' is added to the AGS archive file "archive".

- Assuming the second operation in the AGS archive file "archive" is

  ```
  ags_log archive,A01SIN /ALIAS=LOG /DEBUG=0
  ```

  the spectrum "B01LOG" is added to the AGS archive file "archive".

4.2.23 ags_mpty

Operation: Creates an empty AGS archive file.
Syntax:

```
ags_mpty    <file> [ Title - may be in several words ]
```

Operands: none
Qualifiers: none
Result: A new AGS archive file.
Example: The AGS archive file "archive" is created by

```
ags_mpty archive
```

4.2.24 ags_mult

Operation: Multplies two spectra ($\vec{Y_1}$ and $\vec{Y_2}$)

$$\vec{Z} = \vec{Y_1}\vec{Y_2}$$

Syntax:

```
ags_mult   <file>,<sp1>,<sp2> [/ALIAS=..] [/DEBUG=..]
```

Operands: ags spectra <sp1> and <sp2>
Qualifiers:

- /ALIAS (optional) Name to be given to the resulting spectrum (Default:MULT).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or with the default name MULT.
Example: Assuming the third operation in the AGS archive file "archive" is

```
ags_mult archive,A01SIN,B01IMP /DEBUG=0
```

the spectrum "C01MULT' is added to the AGS archive file "archive".

4.2.25 ags_multval

Operation: Multiplies a spectrum $\vec{Y}$ with a constant value a:

$$\vec{Z} = a\vec{Y}$$

Syntax:

```
ags_multval   <file>,<spect> /VAL=value[,uncertainty] [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /VAL (mandatory)
  value: mandatory
  uncertainty: optional
- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Example: Assuming the second operation in the AGS archive file "archive" is

```
ags_multval archive,A01SIN /VAL=1.0E-5,1.0E-6 /ALIAS=SIN2 /DEBUG=0
```

the spectrum "B01SIN2' is added to the AGS archive file "archive".

4.2.26 ags_putCM

Operation: Prints the covariance matrix V of a spectrum $\vec{Z}$:

$$\mathbf{V}_{\vec{Z}} = \mathbf{U}_{\vec{Z}} + \mathbf{S}_{\vec{Z}}\mathbf{S}_{\vec{Z}}^{T}$$

Syntax:

```
ags_putCM   <file>,<spect> {/GROUPS=..|ZONES=..} [/KIND]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /GROUPS (Optional) defines the part of the spectrum that is printed. The regions are defined by lower ($x_{L,i}$) and higher ($x_{H,i}$) boundaries on the X-values, which are defined by:

  $$x_{L,1}, x_{H,1}, K_1, x_{L,2}, x_{H,2}, K_2, \ldots$$

  where $x_{L,i}$ and $x_{H,i}$ are floating point values and K_i is a positive integer.
  If $K_i = 0$ then all channels with $x_{L,i} < x < x_{H,i}$ are averaged to one observation.
  If $K_i = 1$ then all channels in the group are taken individually.
  if $K_i = n$ with $n > 1$ then the channels in the group i are averaged by subgroups of n channels.

- /ZONES (Mandatory in absence of /GROUPS) defines the part of the spectrum that is printed. The regions are defined by lower ($ch_{L,i}$) and higher ($ch_{H,i}$) boundaries on the channel number (ch), which are defined by:

  $$ch_{L,1}, ch_{H,1}, K_1, ch_{L,2}, ch_{H,2}, K_2 \ldots$$

  where $ch_{L,i}$ and $ch_{H,i}$ are integer zone limits and K_i is a positive integer defined like above.

- /KIND (optional) ONE for single point spectra, if missing assuming spectra with bin boundaries (TWO)

Result: none
Example: Assuming an operation in the AGS archive file "archive" is

```
ags_putCM archive,G01TX /ZONES=200,300,0 > CMout.dat
```

prints the covariance and correlation matrix of channels 200 to 300 of the spectrum G01TX into the ASCII file "CMout.dat".

4.2.27 ags_putX

Operation: Prints the values of a spectrum $\vec{X}$ on the screen together with the covariance information in AGS-format.
Syntax:

```
ags_putX   <file>,<sp> [/GROUPS=..] [/ZONES=..] [/OPT=..] [/DEBUG=..] [/KIND=..] [/FORMAT=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /GROUPS (optional) defines the part of the spectrum that is printed. The regions are defined by lower ($x_{L,i}$) and higher ($x_{H,i}$) boundaries on the X-values, which are defined by:

  $$x_{L,1}, x_{H,1}, K_1, x_{L,2}, x_{H,2}, K_2, \ldots$$

  where $x_{L,i}$ and $x_{H,i}$ are floating point values and K_i is a positive integer.
  If $K_i = 0$ then all channels with $x_{L,i} < x < x_{H,i}$ are averaged to one observation.
  If $K_i = 1$ then all channels in the group are taken individually.
  if $K_i = n$ with $n > 1$ then the channels in the group i are averaged by subgroups of n channels.

- /ZONES (optional, supersedes /ZONES) defines the part of the spectrum that is printed. The regions are defined by lower ($ch_{L,i}$) and higher ($ch_{H,i}$) boundaries on the channel number (ch), which are defined by:

  $$ch_{L,1}, ch_{H,1}, K_1, ch_{L,2}, ch_{H,2}, K_2 \ldots$$

  where $ch_{L,i}$ and $ch_{H,i}$ are integer zone limits and K_i is a positive integer defined like above.

- /KIND (optional) ONE for single point spectra, if missing assuming spectra with bin boundaries (TWO)

- /FORMAT (optional) Specify the format of the output as if it was a c-program (Default: "%16.8E")

- /OPT Line Print options (Default XYA, see below)
  L : Low bound of zone
  H : High bound of zone
  X : Zone center
  Y : Average Y value
  P : Relative uncertainty of Y in per cent
  A : Absolute (total) uncertainties on Y
  C : Covariance contribution: uncorrelated variance + correlated uncertainty on Y
  U : Uncorrelated part
  s : Sammy format sorted by increasing TOF (max 64k channels)
  S : Sammy format sorted by increasing Energy (max 64k channels)
  r : Old REFIT format sorted by increasing TOF (max 64k channels)
  R : Old REFIT format sorted by increasing Energy (max 64k channels)
  O: Print in reverse order
  T: No tab in output (useful for new refit format)

- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: none
Examples:

- Assuming an operation in the AGS archive file "archive" is

```
ags_putX archive,G01TX /OPT=LHYAUC
```

  prints the lower and higher bin boundaries, the spectrum, the absolute uncertainty, the uncorrelated uncertainty, the uncorrelated variance and the S-vector on the screen.

- Assuming an operation in the AGS archive file "archive" is

```
ags_putX archive,G01TX /OPT=LHYAUC > output.dat
```

  prints the lower and higher bin boundaries, the spectrum, the absolute uncertainty, the uncorrelated uncertainty, the uncorrelated variance and the S-vector into a file called "output.dat".

4.2.28 ags_scan

Operation: Printing of the AGS archive file content
Syntax:

```
ags_scan   <file>
```

Operands: none
Qualifiers: none
Result: none
Example: The operation in the AGS archive file "archive"

```
ags_scan archive
```

opens an interactive menu with information on the content of the archive file.

4.2.29 ags_shift

Operation: Performs a transformation of variables based on a linear additive transformation of the X-bounds and a re-binning of the X-bounds of a spectrum. A linear additive transformation is performed based on

$$\vec{t}_{m,2} = \vec{t}_{m,1} + (T_2 - T_1).$$

This operation should only be applied on raw (weighted) data.
Syntax:

```
ags_shift   <file>,<spect> /T1=.. /T2=.. /LIKE=.. [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /T1 (mandatory) Original position (T_1)
- /T2 (mandatory) Target position (T_2)
- /LIKE (mandatory) Name of the target spectrum defining the X-boundaries that will be used to rebin the data.
- /ALIAS (optional) Name to be given to the imported spectra (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Example: Assuming the third operation in the AGS archive file “archive” is

```
ags_shift archive,A01FAST /LIKE=B01FAST /T1=-400.3 T2=-60.5 /DEBUG=1
```

the spectrum “C01FAST’ is added to the AGS archive file “archive”.

4.2.30 ags_subval

Operation: Substracts a constant value a from a spectrum $\vec{Y}$:

$$\vec{Z} = \vec{Y} - a$$

Syntax:

```
ags_subval   <file>,<spect> /VAL=value[,uncertainty] [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /VAL (mandatory)
  value: mandatory
  uncertainty: optional
- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Example: Assuming the second operation in the AGS archive file "archive" is

```
ags_subval archive,A01SIN /VAL=0.0001,0.00001 /DEBUG=1
```

the spectrum "B01SIN' is added to the AGS archive file "archive".

4.2.31 ags_title

Operation: Printing of the AGS archive file title.
Syntax:

```
ags_title   <file>
```

Operands: none
Qualifiers: none
Result: none
Example: The operation in the AGS archive file "archive"

```
ags_title archive
```

prints the title of the AGS archive file "archive".

4.2.32 ags_tof

Operation: Converts the X-boundaries from an energy scale (in eV) to a time scale (in ns) using a flight path lenght L. The calculation is fully relativistic. If the energy is zero or less, than the TOF is set to -1.
Syntax:

```
ags_tof   <file>,<spect> /FPATH=.. [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /FPATH (mandatory) The flight path length in meter.
- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Result: A spectrum with a new prefix and with the name given by the /ALIAS qualifier or inherited from the operand.
Example: Assuming the second operation in the AGS archive file "archive" is

```
ags_tof archive,A01SIN /FPATH=24.5 /ALIAS=TME /DEBUG=1
```

the spectrum "B01TME' is added to the AGS archive file "archive".

4.2.33 ags_unav

Operation: Multiplies a spectrum $\vec{Y}$ with the channel width ($\vec{X}_H$ - $\vec{X}_L$) and the contents of a scaler. (inverse of ags_avrg)
Syntax:

```
ags_unav   <file>,<spect> [/CMSC=position] [/ALIAS=..] [/DEBUG=..]
```

Operands: AGS spectrum <spect>
Qualifiers:

- /CMSC (optional) Position of scaler used for normalization (usually Central Monitor SCaler)
- /ALIAS (optional) Name to be given to the resulting spectrum (Default: name of operand).
- /DEBUG (optional)
  -1: no debug output
  0: very limited debug output (default)
  1: full debug output

Examples:

- Assuming the second operation in the AGS archive file "archive" is

  ```
  ags_unav archive,A01SIN /CMSC=1 /DEBUG=1
  ```

  the spectrum "B01SIN' is added to the AGS archive file "archive".
- Assuming the second operation in the AGS archive file "archive" is

  ```
  ags_unav archive,A01SIN /CMSC=1 /ALIAS=UNA /DEBUG=1
  ```

  the spectrum "B01UNA" is added to the AGS archive file "archive".

5 Examples

5.1 Transmission

In this section an example of the analysis of transmission data is given as a Linux bash script (Section 5.1.1) and as a Windows batch script (Section 5.1.2). The scripts contain the following steps:

1. Executable and data paths are defined
2. An empty AGS archive file is created
3. Scaler, accordion, base channel width and dead-time are specified
4. Count histograms and histograms used for the dead time correction are read for sample-in and sample-out
5. Dead-time correction is performed
6. Bin averaged spectra are generated
7. The background is fitted using black resonance dips
8. The transmission is calculated
9. Print out of spectra

5.1.1 Linux

```
#############################################################################
#                                                                           #
#                     W sample; filter combination Na, Co                   #
#                                                                           #
#############################################################################
#                                                                           #
#                  PRODUCE A TRANSMISSION SPECTRUM                          #
#                                                                           #
#############################################################################
#         1) definition of the paths (of AGS-package and DATA)
#         2) introduction of some identifications
#         3) creation of an empty AGS-file
#############################################################################
AGS_PATH=./../../../AGS_linux/bin

data_path_SI=../data/SAMPLE_IN
data_path_SO=../data/SAMPLE_OUT

IDSI=w_fp2_t01_r01-02_METAL
IDSO=w_fp2_t01_r01-02_OPB

CUTSI=${IDSI}_AGS_1.hst
TCSI=${IDSI}_DT_tc.hst
ADCSI=${IDSI}_DT_adc.hst

CUTSO=${IDSO}_AGS_1.hst
TCSO=${IDSO}_DT_tc.hst
ADCSO=${IDSO}_DT_adc.hst

ags_output=ags_archive_W

echo "step 0 ---> create AGS archive file"
${AGS_PATH}/ags_mpty ${ags_output}

#############################################################################
#         Introduction of parameters necessary for data-reduction:
#         i.e. accordeon, scalers, Time Coder Resolution, time offset,
#              gamma flash position(s), dead time
#############################################################################
acco=16384,1,10240,2,8192,4,8192,8,6144,16,6144,32,4096,64,4096,128,1024,256,1024,512

#         Scalers are given in the following sequence:
#       ...T0...,...CM3...
scal_SI=82490677,453367
scal_SO=81059326,445869

TCR=1.0   # Time Coder Resolution (ns)
DT=3134.  # Dead time (ns)
```

```
#########################################################################
#         Import histogram(s) in ASCII format
#########################################################################
echo "step A ---> import SO-histogram"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${data_path_SO}/${CUTSO} \
/ACCO=${acco} /SCALERS=${scal_SO} /XCONV=${TCR} /ALIAS=SO /DEBUG=1

echo "step B ---> import SI-histogram"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${data_path_SI}/${CUTSI} \
/SCALERS=${scal_SI} /ALIAS=SI /LIKE=A01SO /DEBUG=1

echo "step C ---> import histograms to correct SO for dead-time due \
to events producing a TOF and no amplitude"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${data_path_SO}/${TCSO} \
/SCALERS=${scal_SO} /ALIAS=TCSO /LIKE=A01SO /DEBUG=1

echo "step D ---> import histograms to correct SI for dead-time due \
to events producing a TOF and no amplitude"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${data_path_SI}/${TCSI} \
/SCALERS=${scal_SI} /ALIAS=TCSI /LIKE=A01SO /DEBUG=1

echo "step E ---> import histograms to correct SO for dead-time due \
to events producing a TOF and a amplitude"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${data_path_SO}/${ADCSO} \
/SCALERS=${scal_SO} /ALIAS=ADCSO /LIKE=A01SO /DEBUG=1

echo "step F ---> import histograms to correct SI for dead-time due \
to events producing a TOF and a amplitude"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${data_path_SI}/${ADCSI} \
/SCALERS=${scal_SI} /ALIAS=ADCSI /LIKE=A01SO /DEBUG=1

#########################################################################
#         Perform dead-time corrections
#########################################################################
echo "step G ---> dead time correction for SO"
${AGS_PATH}/ags_dtco ${ags_output},A01SO /BASE=C01TCSO,E01ADCSO \
/DTIME=${DT},${DT} /DTUNC=0.,0. /COEFF=0.000 /TYPE=1 /LPSC=1 /DEBUG=1

echo "step H ---> dead time correction for SI"
${AGS_PATH}/ags_dtco ${ags_output},B01SI /BASE=D01TCSI,F01ADCSI \
/DTIME=${DT},${DT} /DTUNC=0.,0. /COEFF=0.000 /TYPE=1 /LPSC=1 /DEBUG=1

#########################################################################
#         Shift spectra
#########################################################################
echo "step I ---> shift SO spectrum"
${AGS_PATH}/ags_shift ${ags_output},G01SO  /LIKE=A01SO /T1=-10.0 /T2=-20.0

echo "step J ---> shift SI spectrum"
${AGS_PATH}/ags_shift ${ags_output},H01SI /LIKE=A01SO /T1=-10.0 /T2=-20.0
```

```
#########################################################################
#         Generate a Counts/ns-spectrum normalized also to CM3
#########################################################################
echo "step K ---> generate a time of flight spectrum for SO"
${AGS_PATH}/ags_avrg ${ags_output},I01SO /CMSC=2
echo "step L ---> generate a time of flight spectrum for SI"
${AGS_PATH}/ags_avrg ${ags_output},J01SI /CMSC=2

#########################################################################
#         Fitting the background both for SAMPLE-IN and SAMPLE-OUT
#########################################################################
echo "step M ---> fit the background for SO-spectrum"
${AGS_PATH}/ags_fit ${ags_output},K01SO /FUN=E3P0 \
/GUESS=1.12E-05,3.236E-04,-4.955E-05,0.0,1.295E-05,-3.7E-6,0.0,1.075E-03,-3.791E-06,1.25E6 \
/FIX=1,3,4,6,7,8,9,10 \
/ZONES=5241,5380,100,24440,24810,50 \
/ALIAS=BGSO /NITER=100 /CHIDIF=0.00001 /DEBUG=1

echo "step N ---> fit the background for SI-spectrum"
${AGS_PATH}/ags_fit ${ags_output},L01SI /FUN=E3P0 \
/GUESS=1.12E-05,3.15E-04,-4.955E-05,0.0,1.9E-05,-3.7E-6,0.0,1.029E-03,-3.758E-06,1.25E6 \
/FIX=1,3,4,6,7,8,9,10 \
/ZONES=5241,5380,100,24440,24810,50,54438,54606,10 \
/ALIAS=BGSI /CHIDIF=0.00001 /NITER=100 /DEBUG=1

#########################################################################
#         Producing transmission spectrum
#########################################################################
echo "step O ---> generate a net SO-spectrum (subtract background component)"
${AGS_PATH}/ags_lico ${ags_output},K01SO,M01BGSO \
/PAR=1.0,-1.0 /ALIAS=SONET

echo "step P ---> generate a net SI-spectrum (subtract background component)"
${AGS_PATH}/ags_lico ${ags_output},L01SI,N01BGSI \
/PAR=1.0,-1.0 /ALIAS=SINET

echo "step Q ---> generate TRANSMISSION spectrum"
${AGS_PATH}/ags_divi ${ags_output},P01SINET,O01SONET /ALIAS=TRANS

echo "step R ---> generate TRANSMISSION spectrum as function of energy"
${AGS_PATH}/ags_ener ${ags_output},Q01TRANS /FPATH=24.355 /ALIAS=ENE /DEBUG=1

#########################################################################
#         Copy the relevant spectra into an ASCII file
#########################################################################
${AGS_PATH}/ags_putX ${ags_output},K01SO > SO.dat
${AGS_PATH}/ags_putX ${ags_output},L01SI > SI.dat
${AGS_PATH}/ags_putX ${ags_output},M01BGSO > BG_SO.dat
${AGS_PATH}/ags_putX ${ags_output},N01BGSI > BG_SI.dat
${AGS_PATH}/ags_putX ${ags_output},Q01TRANS > T.dat
${AGS_PATH}/ags_putX ${ags_output},R01ENE > T_E.dat
```

5.1.2 Windows

```
@echo off
REM ##############################################################################
REM #                                                                            #
REM #                    W sample; filter combination Na, Co                     #
REM #                                                                            #
REM ##############################################################################
REM #                                                                            #
REM #                 PRODUCE A TRANSMISSION SPECTRUM                            #
REM #                                                                            #
REM ##############################################################################
REM #         1) definition of the paths (of AGS-package and DATA)
REM #         2) introduction of some identifications
REM #         3) creation of an empty AGS-file
REM ##############################################################################
set AGS_PATH=..\..\..\AGS_windows\exe
set data_path_SI=..\data\SAMPLE_IN
set data_path_SO=..\data\SAMPLE_OUT

set IDSI=w_fp2_t01_r01-02_METAL
set IDSO=w_fp2_t01_r01-02_OPB

set CUTSI=%IDSI%_AGS_1.hst
set TCSI=%IDSI%_DT_tc.hst
set ADCSI=%IDSI%_DT_adc.hst

set CUTSO=%IDSO%_AGS_1.hst
set TCSO=%IDSO%_DT_tc.hst
set ADCSO=%IDSO%_DT_adc.hst

set ags_output=ags_archive

echo "step 0 ---> create AGS archive file"
%AGS_PATH%\ags_mpty %ags_output%

REM ##############################################################################
REM #         Introduction of parameters necessary for data-reduction:
REM #         i.e. accordeon, scalers, Time Coder Resolution, time offset,
REM #              gamma flash position(s), dead time
REM ##############################################################################
set acco=16384,1,10240,2,8192,4,8192,8,6144,16,6144,32,4096,64,4096,128,1024,256,1024,512

REM #         Scalers are given in the following sequence:
REM #       ...T0...,...CM3...
set scal_SI=82490677,453367
set scal_SO=81059326,445869

REM # Time Coder Resolution (ns)
set TCR=1.0
REM # Dead time (ns)
set DT=3134.
```

```
REM ########################################################################
REM #         Import histogram(s) in ASCII format
REM ########################################################################
echo "step A ---> import SO-histogram"
%AGS_PATH%\ags_getH %ags_output% /FROM=%data_path_SO%/%CUTSO% ^
/ACCO=%acco% /SCALERS=%scal_SO% /XCONV=%TCR% /ALIAS=SO /DEBUG=1

echo "step B ---> import SI-histogram"
%AGS_PATH%\ags_getH %ags_output% /FROM=%data_path_SI%\%CUTSI% ^
/SCALERS=%scal_SI% /ALIAS=SI /LIKE=A01SO /DEBUG=1

echo "step C ---> import histograms to correct SO for dead-time due ^
to events producing a TOF and no amplitude"
%AGS_PATH%\ags_getH %ags_output% /FROM=%data_path_SO%\%TCSO% ^
/SCALERS=%scal_SO% /ALIAS=TCSO /LIKE=A01SO /DEBUG=1

echo "step D ---> import histograms to correct SI for dead-time due ^
to events producing a TOF and no amplitude"
%AGS_PATH%\ags_getH %ags_output% /FROM=%data_path_SI%\%TCSI% ^
/SCALERS=%scal_SI% /ALIAS=TCSI /LIKE=A01SO /DEBUG=1

echo "step E ---> import histograms to correct SO for dead-time due ^
to events producing a TOF and a amplitude"
%AGS_PATH%\ags_getH %ags_output% /FROM=%data_path_SO%\%ADCSO% ^
/SCALERS=%scal_SO% /ALIAS=ADCSO /LIKE=A01SO /DEBUG=1

echo "step F ---> import histograms to correct SI for dead-time due ^
to events producing a TOF and a amplitude"
%AGS_PATH%\ags_getH %ags_output% /FROM=%data_path_SI%\%ADCSI% ^
/SCALERS=%scal_SI% /ALIAS=ADCSI /LIKE=A01SO /DEBUG=1

REM ########################################################################
REM #         Perform dead-time corrections
REM ########################################################################
echo "step G ---> dead time correction for SO"
%AGS_PATH%\ags_dtco %ags_output%,A01SO /BASE=C01TCSO,E01ADCSO ^
/DTIME=%DT%,%DT% /DTUNC=0.,0. /COEFF=0.000 /TYPE=1 /LPSC=1 /DEBUG=1

echo "step H ---> dead time correction for SI"
%AGS_PATH%\ags_dtco %ags_output%,B01SI /BASE=D01TCSI,F01ADCSI ^
/DTIME=%DT%,%DT% /DTUNC=0.,0. /COEFF=0.000 /TYPE=1 /LPSC=1 /DEBUG=1

REM ########################################################################
REM #         Shift spectra
REM ########################################################################
echo "step I ---> shift SO spectrum"
%AGS_PATH%\ags_shift %ags_output%,G01SO  /LIKE=A01SO /T1=-10.0 /T2=-20.0 /DEBUG=1

echo "step J ---> shift SI spectrum"
%AGS_PATH%\ags_shift %ags_output%,H01SI /LIKE=A01SO /T1=-10.0 /T2=-20.0 /DEBUG=1
```

```
REM ########################################################################
REM #         Generate a Counts/ns-spectrum normalized also to CM3
REM ########################################################################
echo "step k ---> generate a time of flight spectrum for SO"
%AGS_PATH%\ags_avrg %ags_output%,I01SO /CMSC=2

echo "step L ---> generate a time of flight spectrum for SI"
%AGS_PATH%\ags_avrg %ags_output%,J01SI /CMSC=2

REM ########################################################################
REM #         Fitting the background both for SAMPLE-IN and SAMPLE-OUT
REM ########################################################################
echo "step M ---> fit the background for SO-spectrum"
%AGS_PATH%\ags_fit %ags_output%,K01SO /FUN=E3P0 ^
/GUESS=1.12E-05,3.236E-04,-4.955E-05,0.0,1.295E-05,-3.7E-6,0.0,1.075E-03,-3.791E-06,1.25E6 ^
/FIX=1,3,4,6,7,8,9,10 ^
/ZONES=5241,5380,100,24440,24810,50 ^
/ALIAS=BGSO /NITER=100 /CHIDIF=0.00001 /DEBUG=1

echo "step N ---> fit the background for SI-spectrum"
%AGS_PATH%\ags_fit %ags_output%,L01SI /FUN=E3P0 ^
/GUESS=1.12E-05,3.15E-04,-4.955E-05,0.0,1.9E-05,-3.7E-6,0.0,1.029E-03,-3.758E-06,1.25E6 ^
/FIX=1,3,4,6,7,8,9,10 ^
/ZONES=5241,5380,100,24440,24810,50,54438,54606,10 ^
/ALIAS=BGSI /CHIDIF=0.00001 /NITER=100 /DEBUG=1

REM ########################################################################
REM #         Producing transmission spectrum
REM ########################################################################
echo "step O ---> generate a net SO-spectrum (subtract bg component)"
%AGS_PATH%\ags_lico %ags_output%,K01SO,M01BGSO ^
/PAR=1.0,-1.0 /ALIAS=SONET

echo "step P ---> generate a net SI-spectrum (subtract bg component)"
%AGS_PATH%\ags_lico %ags_output%,L01SI,N01BGSI ^
/PAR=1.0,-1.0 /ALIAS=SINET

echo "step Q ---> generate TRANSMISSION spectrum"
%AGS_PATH%\ags_divi %ags_output%,P01SINET,O01SONET /ALIAS=TRANS

echo "step R ---> generate TRANSMISSION spectrum as function of energy"
%AGS_PATH%\ags_ener %ags_output%,Q01TRANS /FPATH=24.355 /ALIAS=ENE /DEBUG=1

REM ########################################################################
REM #         Copy the relevant spectra into an ASCII file
REM ########################################################################
%AGS_PATH%\ags_putX %ags_output%,K01SO > SO.dat
%AGS_PATH%\ags_putX %ags_output%,L01SI > SI.dat
%AGS_PATH%\ags_putX %ags_output%,M01BGSO > BG_SO.dat
%AGS_PATH%\ags_putX %ags_output%,N01BGSI > BG_SI.dat
%AGS_PATH%\ags_putX %ags_output%,Q01TRANS > T.dat
%AGS_PATH%\ags_putX %ags_output%,R01ENE > T_E.dat
```

5.2 Capture

In this section an example of the analysis of capture data is given as Linux bash scripts. The data reduction is broken down into several steps:

1. Data import and dead time correction (Section 5.2.1)
2. Background correction (Section 5.2.2)
3. Data import, dead time correction and alignment of flux measurement (Section 5.2.3)
4. Background correction of flux measurement (Section 5.2.3)
5. Calculation of the capture yield (Section 5.2.4)

5.2.1 Data import and dead time correction

In the following a script is given which is used to import the counting spectra from a sample-in measurement and to perform a dead time correction. In a full analysis, this step has to be performed for sample-in, scattering sample-in and open beam.

```
#########################################################################
#                                                                       #
#            Au sample; filter combination B, S, Na                     #
#                                                                       #
#########################################################################
#                                                                       #
#                 STEP 1 / 3                                            #
#        sum of all ADC's and perform DT correction on capture data     #
#                                                                       #
#########################################################################
AGS_PATH=./../../../AGS_linux/bin
data_path=../data

cap_ses_name=Au197_0.53mm
ID=Au197

#########################################################################
# Introduce values for scalers, accord, DT, T0 and TCR according to     #
#########################################################################
#CM1,CM3,T0,ACOF
scal_cap=58923655,1759362,63690028,55535639
acco_cap=4096,1,4096,2,4096,4,4096,8,4096,16,3072,32,3072,64,4096,128,7168,256

DTadc=2870. #Dead Time with only ADC
DTtc=2870. #Dead Time with only TC
TCR=1.0 #Time coder resolution
g_flash_m=80 #Measured Gamma Flash position
g_flash_t=43 #L/C [ns], --> L=12.938 m <--

#########################################################################
# Define input data names                                               #
#########################################################################
ags_output=s1cw_${ID}.ags
ags_C6D6_WF_1=${cap_ses_name}_WF_1.dat
ags_C6D6_WF_2=${cap_ses_name}_WF_2.dat
ags_C6D6_DT_adc=${cap_ses_name}_DT_adc.hst
ags_C6D6_DT_tc=${cap_ses_name}_DT_tc.hst

#########################################################################
# Create AGS archive file                                               #
#########################################################################
rm ${ags_output}
echo "-->Initialize ags file"
${AGS_PATH}/ags_mpty ${ags_output}
```

```
##########################################################################
# STEP A-B                                                               #
# Import AGL spectra for all (2) ADCs                                    #
##########################################################################
echo "-->step A : Import file C6D6 ADC1"
${AGS_PATH}/ags_getY ${ags_output} /FROM=${data_path}/${ags_C6D6_WF_1} \
 /ACCO=${acco_cap} /SCALERS=${scal_cap} /XCONV=${TCR} /ALIAS=ADC1 /DEBUG=1

echo "-->step B : Import file C6D6 ADC2"
${AGS_PATH}/ags_getY ${ags_output} /FROM=${data_path}/${ags_C6D6_WF_1} \
 /LIKE=A01ADC1  /SCALERS=${scal_cap} /ALIAS=ADC2 /DEBUG=1

##########################################################################
# STEP C-D:                                                              #
# spectra for dead time correction from HST format: counts               #
##########################################################################
echo "-->step C : Import file C6D6 DEAD TIME ADC"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${data_path}/${ags_C6D6_DT_adc} \
 /LIKE=A01ADC1 /SCALERS=${scal_cap} /DEBUG=1

echo "-->step D : Import file C6D6 DEAD TIME TC"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${data_path}/${ags_C6D6_DT_tc} \
 /LIKE=A01ADC1 /SCALERS=${scal_cap} /DEBUG=1

##########################################################################
# STEP E:                                                                #
# sum of adcs spectra and averaging (div by channel time width)          #
##########################################################################
echo "c6d6 spectra sum"
${AGS_PATH}/ags_lico ${ags_output},A01ADC1,B01ADC2 /PAR=1.,1. /ALIAS=SUM

##########################################################################
# STEP F:                                                                #
# dead time correction on the sum of adcs spectra                        #
# DT_adc.hist = sum of counts in both ADC                                #
# DT_tc.hist = sum of counts in both ADC when only time is recoded       #
##########################################################################
echo "dead time correction "
${AGS_PATH}/ags_dtco ${ags_output},E01SUM /BASE=C01HIST,D01HIST \
 /DTIME=${DTadc},${DTtc} /DTUNC=5.,5. /TYPE=1 /LPSC=3 /DEBUG=1

##########################################################################
# STEP G:                                                                #
# shift spectrum - brings Gamma flash at L/C                             #
##########################################################################
echo "-->step F : Shift --> TOF"
${AGS_PATH}/ags_shift ${ags_output},F01SUM /T1=${g_flash_m} \
 /T2=${g_flash_t} /LIKE=E01SUM /ALIAS=SFTD /DEBUG=1
```

```
#############################################################################
# STEP H:                                                                   #
#     NORMALIZAITION  TO A MONITOR                                          #
#############################################################################
echo "capture average  divided by cm3 or ACOF"
${AGS_PATH}/ags_avrg ${ags_output},G01SFTD /CMSC=2
#############################################################################
# saving .dat files these are not AGS steps                                 #
#############################################################################

echo "saving ASCII"
${AGS_PATH}/ags_putX ${ags_output},H01SFTD /OPT=LHXYA > gross_${ID}.dat

#############################################################################
#                                                                           #
#                        END OF THE SCRIPT                                  #
#                                                                           #
#############################################################################
```

5.2.2 Background

In the following a script to perform the background correction of the sample-in spectrum is given. The background correction is based on Eq. (15). It reads the results of the first script (Section 5.2.1) used for the sample-in, scattering sample-in and open beam. In addition a neutron sensitivity correction factor is read. The background corrected capture spectrum is calculated.

```
#########################################################################
#                                                                       #
#            Au sample; filter combination B, S, Na                     #
#                                                                       #
#########################################################################
#                                                                       #
#                 STEP 2 / 3                                            #
#         background subtraction on capture data                        #
#                                                                       #
#########################################################################
AGS_PATH=./../../../AGS_linux/bin

ID=Au197 #sample
IDob=OPB4Au #open beam
YR=YR050.DAT
IDpb=Pb4Au #lead
scal_zero=0,0,0,0
#########################################################################
#  declare spectra for sample, OPB, Pb etc. produced in step 1          #
#            define output name                                        #
#########################################################################
ags_gross=s1cw_${ID}.ags
ags_ob=s1cw_${IDob}.ags
ags_pb=s1cw_${IDpb}.ags
ags_output=s2cw_${ID}.ags

#########################################################################
# Create AGS archive file                                               #
#########################################################################
rm ${ags_output}
echo "-->Initialize ags file"
${AGS_PATH}/ags_mpty ${ags_output}

#########################################################################
# STEP A-B:      Import sample and OPB spectra normalized over CM       #
#                                                                       #
#########################################################################
echo "-->step A : Importing C6D6 averaged - sample spectrum per CM"
${AGS_PATH}/ags_getA ${ags_output} /FROM=${ags_gross},H01SFTD /ALIAS=SAMP

echo "-->step B : Importing C6D6 averaged - OPB per CM"
${AGS_PATH}/ags_getA ${ags_output} /FROM=${ags_ob},H01SFTD /ALIAS=OPB \
/LIKE=A01SAMP /DEBUG=1
```

```
echo "-->step C : Importing C6D6 averaged - OPB per CM"
${AGS_PATH}/ags_getY ${ags_output} /FROM=${YR} /ALIAS=YR /LIKE=A01SAMP \
/SCALERS=${scal_zero} /DEBUG=1

echo "-->step D : Importing C6D6 averaged - Pb per CM"
${AGS_PATH}/ags_getA ${ags_output} /FROM=${ags_pb},H01SFTD /ALIAS=Pb \
/LIKE=A01SAMP  /DEBUG=1

##########################################################################
# STEP E-F:     AMB and OPB subtraction from sample                      #
#                                                                        #
##########################################################################
echo "-->step E : calculate total background"
${AGS_PATH}/ags_fxyp ${ags_output},B01OPB,C01YR,D01Pb /OWN=c03 \
/PAR=5E-4,1,0,1,0 /COV=2.5E-9,0.0009,0,0.0025,0 /ALIAS=BKG /DEBUG=1

echo "-->step F : calculate net spectrum"
${AGS_PATH}/ags_lico ${ags_output},A01SAMP,E01BKG /PAR=1.0,-1.0 /ALIAS=NET

##########################################################################
# saving .dat files these are not AGS steps                              #
##########################################################################
echo "-->saving files..."
${AGS_PATH}/ags_putX ${ags_output},B01OPB  /OPT=XYA  > OPB_${ID}.dat
${AGS_PATH}/ags_putX ${ags_output},D01Pb  /OPT=XYA  > Pb_${ID}.dat
${AGS_PATH}/ags_putX ${ags_output},E01BKG  /OPT=XYA  > BKG_${ID}.dat
${AGS_PATH}/ags_putX ${ags_output},F01NET  /OPT=XYA  > net_${ID}.dat

##########################################################################
#                                                                        #
#                       END OF THE SCRIPT                                #
#                                                                        #
##########################################################################
```

5.2.3 Flux

The first of the following two scripts is used to perform the dead time correction of a flux measurement with a double sided ionization chamber. Both spectra are then aligned by shifting them using the dip in the spectra caused by the 30.38 keV resonance of the sulphur filter. The spectra are then added and projected on the capture flight path. The second script is used to fit the background of the flux measurement and to calculate the net flux spectrum.

Dead time correction and alignment

```
#########################################################################
#                                                                       #
#            Flux IC; filter combination B, S, Na                       #
#                                                                       #
#########################################################################
#                                                                       #
#                 STEP 1 / 2                                            #
#        sum of all ADC's and perform DT correction data                #
#        align adcs and shift to capture                                #
#                                                                       #
#########################################################################
AGS_PATH=./../../../AGS_linux/bin
flx_ses_name=FX_BSNa

agl_data_path=../data/
ID=BSNa

#########################################################################
# Introduce values for scalers, accord, DT, T0 and TCR according to  #
#        'an_rep' output file from last AGL step                        #
#   Divide CM3 value by 1E2 in both flux and capture  !!!!!!!!!      #
#   #CM1 CM3 T0 ACO                                        #
#########################################################################
#CM1 CM3 T0 ACO
scal_flux=1012498868,29961122,1110043004,940560877
acco_flux=6144,1,4096,2,4096,4,3072,8,3072,16,3072,32,3072,64,3072,128

DTadc=3940.    #Dead Time with adc
DTtc=3940.     #Dead Time with only tc
TCR=4.0        #time coder resolution

fpl_f=12.12133 #FPL of flux
fpl_c=12.938   #FPL capture

TSc=5030.3     # position of 30.38 keV S resonance in capture (scaled by ratio of fps)
TS1=3960.3     # position of 30.38 keV S resonance in flux 1
TS2=3938.4     # position of 30.38 keV S resonance in flux 2

#########################################################################
# Define input data names                                               #
#########################################################################
ags_output=s1f_${ID}.ags
ags_FLUX_1=${flx_ses_name}_AGS_1.hst
ags_FLUX_2=${flx_ses_name}_AGS_2.hst
ags_F_DT_ADC=${flx_ses_name}_DT_adc.hst
ags_F_DT_TC=${flx_ses_name}_DT_tc.hst
```

```
##########################################################################
# Get capture reference spectrum                                         #
##########################################################################
ags_capt=s1cw_Au197.ags
echo "--> getting ref. capture"
cp ../Au197/${ags_capt} .

##########################################################################
# Create AGS archive file                                                #
##########################################################################
rm ${ags_output}
echo "-->Initialize ags file"
${AGS_PATH}/ags_mpty ${ags_output}

##########################################################################
# STEP A-B                                                               #
# Import AGL spectra for ADCs  (2 in FP5_10m)                            #
##########################################################################
echo "-->step A : Import file from AGL FLUX1"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${agl_data_path}/${ags_FLUX_1} \
 /ALIAS=FLUX /ACCORD=${acco_flux} /SCALERS=${scal_flux} /XCONV=${TCR}

echo "-->step B : Import file from AGL FLUX2"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${agl_data_path}/${ags_FLUX_2} \
 /ALIAS=FLUX /LIKE=A01FLUX /SCALERS=${scal_flux}

##########################################################################
# STEP C-D                                                               #
# Import dead time ADC and TC                                            #
##########################################################################
echo "-->step C : Import file from AGL FLUX DEAD TIME ADC"
# having 1 adc G and H are the same...beside amp windows on amp
${AGS_PATH}/ags_getH ${ags_output} /FROM=${agl_data_path}/${ags_F_DT_ADC} \
 /LIKE=A01FLUX /SCALERS=${scal_flux}

echo "-->step D : Import file from AGL FLUX DEAD TIME TC"
${AGS_PATH}/ags_getH ${ags_output} /FROM=${agl_data_path}/${ags_F_DT_TC} \
 /LIKE=A01FLUX /SCALERS=${scal_flux}

##########################################################################
# STEP E-F                                                               #
# dead time correction for individual ADCs                               #
##########################################################################
echo "-->step E : dead time correction FLUX1"
${AGS_PATH}/ags_dtco ${ags_output},A01FLUX /BASE=C01HIST,D01HIST \
/DTIME=${DTadc},${DTtc} /DTUNC=0.,0. /TYPE=1 /LPSC=3

echo "-->step F : dead time correction FLUX2"
${AGS_PATH}/ags_dtco ${ags_output},B01FLUX /BASE=C01HIST,D01HIST \
/DTIME=${DTadc},${DTtc} /DTUNC=0.,0. /TYPE=1 /LPSC=3
```

```
#########################################################################
# STEP G                                                                #
# shift ADCs to account for time offset                                 #
#########################################################################
echo "-->step G : Shift flux 1 after DT corrections"
${AGS_PATH}/ags_shift ${ags_output},E01FLUX /T1=${TS1} \
 /T2=${TSc}  /LIKE=F01FLUX /ALIAS=ADC1SFTD

echo "-->step H : Shift flux 2 after DT corrections"
${AGS_PATH}/ags_shift ${ags_output},F01FLUX /T1=${TS2} \
 /T2=${TSc}  /LIKE=F01FLUX /ALIAS=ADC2SFTD

#########################################################################
# STEP I            sum ADCs                                            #
#                       ADC2 5.5% thicker                               #
#########################################################################
${AGS_PATH}/ags_lico ${ags_output},G01ADC1SFTD,H01ADC2SFTD, /PAR=1.,0.945 \
/ALIAS=FLUX

#########################################################################
# STEP J-K    rebin flux to capture                                     #
#       IMPORTANT:to do so, we need  a capture spectrum                 #
#     so first we have to create this spectrum from the s1cw script     #
#########################################################################
echo "-->step J : Import capture spectrum"
${AGS_PATH}/ags_getA ${ags_output} /FROM=${ags_capt},H01SFTD /ALIAS=SUM

echo "-->step K : Adjust flight path "
${AGS_PATH}/ags_fpa ${ags_output},I01FLUX /FPATH1=${fpl_f} /FPATH2=${fpl_c} \
/LIKE=J01SUM /ALIAS=FLXREB

#########################################################################
# STEP L   create spectrum nrml per cm3 or ACOF (still TOF) and         #
#########################################################################
echo "flux average  divided by cm3 or ACOF"
${AGS_PATH}/ags_avrg ${ags_output},K01FLXREB /CMSC=2 /ALIAS=FLX

#########################################################################
# saving .dat files these are not AGS steps                             #
#########################################################################
echo "-->saving files..."
${AGS_PATH}/ags_putX ${ags_output},L01FLX /OPT=XYA > flux_gross_${ID}.dat

#########################################################################
#                                                                       #
#                       END OF THE SCRIPT                               #
#                                                                       #
#########################################################################
```

Background

```
#############################################################################
#                                                                           #
#            Flux IC; filter combination B, S, Na                           #
#                                                                           #
#############################################################################
#                                                                           #
#                 STEP 2 / 2                                                #
#        fit background and calculate net flux spectrum                     #
#                                                                           #
#############################################################################
AGS_PATH=./../../../AGS_linux/bin
ID=BSNa

ags_input=s1f_${ID}.ags
ags_output=s2f_${ID}.ags

#############################################################################
# Create AGS archive file                                                   #
#############################################################################
rm ${ags_output}
echo "-->Initialize ags file"
${AGS_PATH}/ags_mpty ${ags_output}

#############################################################################
# STEP A                                                                    #
# Import AGS gross flux                                                     #
#############################################################################
echo "-->step A : importing gross flux"
${AGS_PATH}/ags_getA ${ags_output} /FROM=${ags_input},L01FLX /ALIAS=FLX

#############################################################################
# STEP B                                                                    #
#         Fitting the background                                            #
#############################################################################
echo "-->step B :  fit the background"
${AGS_PATH}/ags_fit ${ags_output},A01FLX /FUN=E2P0 \
/GUESS=6.56E-07,3.236E-04,-5.30E-05,0.0,5.29E-05,-2.33E-06,1.25E6 \
/FIX=1,4,5,6,7 \
/ZONES=2840,2905,10,9447,9565,10 \
/ALIAS=BKG /NITER=100 /CHIDIF=0.00001 /DEBUG=1

#############################################################################
# STEP C                                                                    #
#  bkg subtraction                                                          #
#############################################################################
echo "-->step B :  calculating net spectrum"
${AGS_PATH}/ags_lico ${ags_output},A01FLX,B01BKG /PAR=1.0,-1.0 /ALIAS=NET
```

```
#############################################################################
# saving .dat files these are not AGS steps                                 #
#############################################################################
echo "-->saving files..."
${AGS_PATH}/ags_putX ${ags_output},B01BKG /OPT=XYA > flux_bkg_fct_${ID}.dat
${AGS_PATH}/ags_putX ${ags_output},C01NET /OPT=XYA > net_flux_${ID}.dat

#############################################################################
#                                                                           #
#                       END OF THE SCRIPT                                   #
#                                                                           #
#############################################################################
```

5.2.4 Yield

In the following a script to calculate the capture yield is given. The calculation of the yield is based on Eq. (13) and (14). The script reads the result of the second script (Section 5.2.2) as well as the net flux spectrum (Section 5.2.3). In addition, the boron total and capture cross sections are read from a ENDF-6 formatted file. The final result is the capture yield as function of TOF, neutron energy and in REFIT format.

```
##########################################################################
#                                                                        #
#            Au sample; filter combination B, S, Na                      #
#                                                                        #
##########################################################################
#                                                                        #
#                 STEP 3 / 3                                             #
#   yield calculation from net flux counts and capture data              #
#                                                                        #
##########################################################################
AGS_PATH=./../../../AGS_linux/bin

fpl_c_real=12.944       #FPL capture
ID_c=Au197 #sample
ID_f=BSNa #flux

scal_zero=0,0,0,0

path_ags_xsec=../X-SEC
b10_jeff22=10b.dat
IC_thick=4.79E-06 #thickness of IC, 10B in at/b at FP5_10m and FP15_30m

##########################################################################
#  declare spectra for flux and net capture produced in step 2           #
#            define output name                                          #
##########################################################################
ags_capt=s2cw_${ID_c}.ags
ags_flux=s2f_${ID_f}.ags
ags_output=s3cw_${ID_c}.ags

##########################################################################
# Create AGS archive file                                                #
##########################################################################
rm ${ags_output}
echo "-->Initialize ags file"
${AGS_PATH}/ags_mpty ${ags_output}

##########################################################################
# STEP A-B: Import sample and flux net spectra                           #
##########################################################################
echo "-->step A : Importing C6D6 averaged - sample spectrum per CM"
${AGS_PATH}/ags_getA ${ags_output} /FROM=${ags_capt},F01NET /ALIAS=CAPT
```

```
cp ../flux/${ags_flux} .
echo "-->step B : Importing flux averaged - per CM"
${AGS_PATH}/ags_getA ${ags_output} /FROM=${ags_flux},C01NET /ALIAS=FLUX \
/LIKE=A01CAPT /DEBUG=1

#########################################################################
# STEP C-D:   Import 10B(n,alfa) and total cross section              #
#########################################################################
echo "-->step C-D : Retrieve 10B(n,alfa) cross section"
${AGS_PATH}/ags_getET ${ags_output} /FROM=${path_ags_xsec}/${b10_jeff22},525,107 \
/LIKE=A01CAPT /ALIAS=B10NAJEF /FPATH=${fpl_c_real}

${AGS_PATH}/ags_getET ${ags_output} /FROM=${path_ags_xsec}/${b10_jeff22},525,1 \
/LIKE=A01CAPT /ALIAS=B10NTJEF /FPATH=${fpl_c_real}

#########################################################################
# STEP E-J: Building flux                                             #
#########################################################################
echo "-->step E-L : Building flux"
${AGS_PATH}/ags_multval ${ags_output},D01B10NTJEF /VAL=${IC_thick} /ALIAS=NSIGT
${AGS_PATH}/ags_exp ${ags_output},E01NSIGT /ALIAS=EXPS
${AGS_PATH}/ags_addval ${ags_output},F01EXPS /VAL=-1 /ALIAS=ABS
${AGS_PATH}/ags_divi ${ags_output},C01B10NAJEF,D01B10NTJEF /ALIAS=SIGR
${AGS_PATH}/ags_mult ${ags_output},G01ABS,H01SIGR /ALIAS=QY
${AGS_PATH}/ags_divi ${ags_output},B01FLUX,I01QY /ALIAS=FLUX

#########################################################################
# STEP K-L: Calculating yield                                          #
#########################################################################
echo "-->step K-L : calculating yield"
${AGS_PATH}/ags_divi ${ags_output},A01CAPT,J01FLUX /ALIAS=Y
${AGS_PATH}/ags_divval ${ags_output},K01Y /VAL=7.927 /ALIAS=YN

#########################################################################
# STEP M: Convert tof to energy                                       #
#########################################################################
echo "-->step M : convert tof to energy"
${AGS_PATH}/ags_ener ${ags_output},L01YN /FPATH=${fpl_c_real} /ALIAS=ENE

#########################################################################
# saving .dat files these are not AGS steps                           #
#########################################################################
echo "-->saving files..."
${AGS_PATH}/ags_putX ${ags_output},J01FLUX /OPT=XYA > Flux_for_Yield_${ID_c}.dat
${AGS_PATH}/ags_putX ${ags_output},L01YN /OPT=XYA > Yield_${ID_c}.dat
${AGS_PATH}/ags_putX ${ags_output},M01ENE /OPT=XYA > Yield_E_${ID_c}.dat

#########################################################################
#                                                                       #
#                       END OF THE SCRIPT                               #
#                                                                       #
#########################################################################
```

A symbols, nomenclature and abbreviations

Table A.1: Symbols

Symbol	
t_m	measured TOF
T_s	start signal
T_0	stop signal
t_0	time offset
t_t	time difference between the moment of neutron creation and the moment it leaves the target
t_d	time difference between the moment the neutron enters the detector or sample and the time of detection
v	neutron speed
c	speed of light
L	distance between the outer surface of the neutron-producing target and the front face of the detector or sample
E	neutron kinetic energy
m_n	neutron mass
γ	Lorentz factor
T	transmission
n_k	areal density of nuclide k [at/b]
$\bar{\sigma}_{tot,k}$	Doppler broadened total cross section of nuclide k
T_{exp}	experimental transmission
C_{in}	sample-in count rate in a transmission experiment
C_{out}	sample-out count rate in a transmission experiment
B_{in}	background contribution of sample-in count rate in a transmission experiment
B_{out}	background contribution of sample-in count rate in a transmission experiment
N_T	normalization of a transmission experiment
K	factor to introduce a correlated uncertainty due to the background model of a transmission experiment
B_0	time independent background component
B_γ	time dependent background component due to 2.2 MeV γ-rays
B_{no}	time dependent background component due to overlap neutrons
B_{ns}	time dependent background component due to scattered neutrons in the detector stations
B_{ne}	time dependent background component due to scattered neutrons in the environment
$Y_{r,k}$	yield of nuclide k and of reaction r
$Y_{0,k}$	primary yield of nuclide k
$Y_{m,k}$	multiple interaction yield of nuclide k
$\bar{\sigma}_{r,k}$	Doppler broadened reaction cross section of nuclide k and reaction r
Y_{exp}	experimental yield
C_r	reaction count rate of a reaction cross section measurement

Continued on next page

Table A.1 – *Continued from previous page*

Symbols	
B_r	background contribution of a reaction cross section measurement
φ	flux
N_r	normalization of a reaction cross section experiment
C_φ	count rate of flux measurement
B_φ	background contribution of the flux measurement
Y_φ	yield of the flux measurement
T_φ	transmission of the flux measurement
$\bar{\sigma}_{r,\varphi}$	Doppler broadened reaction cross section used to determine the flux
$\bar{\sigma}_{tot,\varphi}$	Doppler broadened total cross section used to determine the flux
$N_0(i)$	number of counts in channel i
$N_c(i)$	dead time corrected number of counts in channel i
N_b	number of bursts

Table A.2: Nomenclature

Symbol	
$\vec{Z}$	spectrum vector (capital letter)
$\vec{p}$	parameter vector (small letter)
$\mathbf{M}$	matrix (bold capital letter)
$\mathbf{M}^T$	transposed of matrix $\mathbf{M}$
$\mathbf{M}_{((i,j))}$	element (i,j) of matrix $\mathbf{M}$
$\mathbf{V}_{\vec{Z}}$	covariance matrix of spectrum $\vec{Z}$
$\mathbf{U}_{\vec{Z}}$	covariance matrix of spectrum $\vec{Z}$ containing only the uncorrelated uncertainty component
$\vec{U}_{\vec{Z}}$	vector of the uncorrelated covariance component: $\vec{U}_{\vec{Z}} = diag\left(\mathbf{U}_{\vec{Z}}\right)$
$\vec{u}_{\vec{Z}}$	vector of the uncorrelated uncertainty: $\vec{u}_{\vec{Z}} = \sqrt{\vec{U}_{\vec{Z}}}$
$\mathbf{D}_f(\vec{b})$	sensitivity matrix: partial derivatives of the function $f(x,\vec{b})$ with respect to the parameter vector $\vec{b}$ with dimension m: $$\mathbf{D}_f(\vec{b}) = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{b}}\right) = \begin{pmatrix} \partial f_i/\partial b_1 & \dots & \partial f_1/\partial b_m \\ \vdots & \ddots & \vdots \\ \partial f_n/\partial b_1 & & \partial f_n/\partial b_m \end{pmatrix}$$ where $\partial f_1/\partial b_j$ is the partial derivative of f with respect to b_j evaluated at $x = x_i$
$\left[\vec{b}_1, \vec{b}_2, \vec{b}_3, \dots\right]$	concatenation of several vectors $\vec{b}_1, \vec{b}_2, \vec{b}_3, \dots$ with the same dimension into a matrix $\mathbf{B} = \left[\vec{b}_1, \vec{b}_2, \vec{b}_3, \dots\right]$. The concatenation of several matrices with the same number of rows is symbolized in the same manner.

Table A.3: Abbreviations

Abbreviation	
AGS	Analysis of Geel Spectra
GELINA	GEel LINear Accelerator
EC	European Commission
JRC	Joint Research Centre
IRMM	Institute for Reference Materials and Measurements
TOF	Time-Of-Flight
GLUP	Generalized Law of Uncertainties Propagation
GUM	Guide to the Expression of Uncertainty in Measurement
ENDF	Evaluated Nuclear Data File
FWHM	Full Width Half Maximum

References

[1] C. Bastian, General Procedures and Computational Methods for Generating Covariance Matrices, in Proc. Int. Symposium On Nucl. Data Evaluation Methodology, pp. 642–649, Oct. (1992).

[2] C. Bastian, *AGS a set of UNIX Commands for Neutron Data Reduction*, Proc. SPIE **2867** No. 611 (1997). DOI: 10.1117/12.267862

[3] B. Becker, C. Bastian, F. Emiliani, F. Gunsing, J. Heyse, K. Kauwenberghs, S. Kopecky, C. Lampoudis, C. Massimi, N. Otuka, P. Schillebeeckx, I. Sirakov, J. Instrum. **7**,P11002, DOI: 10.1088/1748-0221/7/11/P11002 (2012).

[4] P. Schillebeeckx, B. Becker, Y. Danon, K. Guber, H. Harada, J. Heyse, A.R. Junghans, S. Kopecky, C. Massimi, M. Moxon, N. Otuka, I. Sirakov, K. Volev, Nuclear Data Sheets **113**, 3054 (2012). DOI: 10.1016/j.nds.2012.11.005

[5] N. Otuka, A. Borella, S. Kopecky, C. Lampoudis, P. Schillebeeckx, J. Korean Phys. Soc. **59**, 1314 - 1317 (2011). DOI: 10.3938/jkps.59.1314

[6] N. Otuka, S. Dunaeva, E. Dupont, O. Schwerer, A. Blokhin, J. Korean Phys. Soc. **59**, 1292 – 1297, (2011). DOI: 10.3938/jkps.59.1292

[7] Consultants' Meeting, 8 to 10 October 2013, IAEA Headquarters, Vienna, Austria. https://www-nds.iaea.org/index-meeting-crp/CM-RF-2013/ (2013).

[8] W. Mondelaers, P. Schillebeeckx, Notiziario Neutroni e Luce di Sincrotrone **11 no.2**, 19 (2006).

[9] D.C. Larson, N.M. Larson, J.A. Harvey, N.W. Hill, and C.H. Johnson, Application of new techniques to ORELA neutron transmission measurements and their uncertainty analysis: the case of natural nickel from 2 keV to 20 MeV, ORNL Report, ORNL/TM-8203 (1983).

[10] S. Kopecky, I. Ivanov, M. Moxon, P. Schillebeeckx, P. Siegler and I. Sirakov, Nucl. Instr. Meth. B **267**, 2345 – 2350 (2009). DOI: 10.1016/j.nimb.2009.04.010

[11] I. Sirakov, B. Becker, R. Capote, E. Dupont, S. Kopecky, C. Massimi, P. Schillebeeckx, Eur. Phys. J. A **49** 144, 1- 10, (2013) DOI: 10.1140/epja/i2013-13144-2

[12] A.D. Carlson, V.G. Pronyaev, D.L. Smith, N.M. Larson, Z. Chen, G.M. Hale, F.-J. Hambsch, E.V. Gai, Soo-Youl Oh, S.A. Badikov, T. Kawano, H.M. Hofman, H. Vonach, S. Tagesen, Nuclear Data Sheets **110**, 3215 (2009).

[13] M.S. Moore, Nuclear Instruments and Methods **169** No. 1, 245–247 (1980). DOI: 10.1016/0029-554X(80)90129-9

[14] Joint Committee for Guides in Metrology (JCGM/WG 1), Evaluation of Measurement Data – Guide to the Expression of Uncertainty in Measurement, JCGM 100:2008 (2008).

[15] M.C. Moxon, T.C. Ware, and C.J. Dean, REFIT-2009 A Least-Square Fitting Program for Resonance Analysis of Neutron Transmission, Capture, Fission and Scattering Data – Users' Guide for REFIT-2009-10, UKNSF(2010)P243 (2010).

[16] F.H. Fröhner, Evaluation and analysis of nuclear resonance data, JEFF Report 18, NEA/OECD (2000).

[17] F.H. Fröhner, Nucl. Sci. Eng. **126** 1 – 18 (1997).

[18] D.L. Smith and N. Otuka, Nuclear Data Sheets **113**, 3006 (2012). DOI: 10.1016/j.nds.2012.11.004

[19] C. De Saint Jean, G. Noguere, B. Habert, B. Iooss, Nucl. Sci. Eng. **161** No. 3, pp.363–370 (2009).

[20] B. Habert, C. De Saint Jean, G. Noguere, L. Leal, Y. Rugama, Nucl. Sci. Eng. **166** No. 3, pp.276–287 (2010).

[21] K.B. Petersen, M.S. Pedersen, The Matrix Cookbook, Version 20121115, Technical University of Denmark (2012) http://www2.imm.dtu.dk/pubdb/p.php?3274 (2014).

[22] N.M. Larson, Report ORNL/TM-9179/R8 and ENDF-364/R2, Oak Ridge National Laboratory, USA, (2008).

[23] M. Herman (Ed.), *ENDF- 6 Formats Manual*, Report BNL-NCS-44945-05-Rev (2005).