Danmarks
Tekniske Universitet

Analytic models in velocity-space tomography for fusion plasmas

by
Andreas Feldt Lomholt Poulsen (s134230)

Supervisor: Mirko Salewski
Assistant supervisor: Birgitte Madsen

Master project
Technical University of Denmark, DTU
June 23, 2018

Abstract

Fusion energy is a potential alternative to fossil fuels that has received a lot of attention over the years because it represents a clean, stable and safe energy source that can last for thousands of years. However, an actual fusion reactor capable of net energy generation during steady-state operation has yet to be realized, and research into the related technologies is still ongoing. One important aspect of operating a fusion reactor is the need to determine the velocity distributions of the fast ions in the plasma. This is done via velocity-space tomography of measured spectra from various diagnostics, but most of the existing models are numerical and slow, though accurate. In order to complement these numerical models, we have written and tested analytic models for velocity-space tomography based on any combination of Collective Thomson Scattering (CTS), Neutron Emission Spectrometry (NES), one- and two-step reaction gamma-ray spectrometry (GRS) and Fast-ion D_α spectroscopy (FIDA). The models were tested by generating synthetic spectra from bi-Maxwellian or slowing-down model distributions and comparing these model distributions to reconstructions obtained from the spectra. The testing revealed some issues with the use of one-step reaction GRS, but we were able to demonstrate that the analytic models can be used to evaluate the suitability of a given diagnostic setup or potential additions thereto for a given situation.

Contents

1 Introduction

Over the past several decades, many governments and scientists have put an increasing amount of effort into finding, improving and implementing alternatives to fossil fuels. This is motivated by the fact that not only are the existing reserves of fossil fuels limited, but the use of fossil fuels also releases greenhouse gases, which contributes to global warming and all of its related problems. [14] [20] [6] [46] Fusion energy is one such alternative that has been the target of ongoing research by multiple countries for many years due to its numerous potential advantages. [53] [9] [12]
First of all, the use of fusion power would not release any greenhouse gases, and fusion power plants would be a much more stable and reliable source of energy than renewable energy sources like windmills or solar panels whose outputs vary with the weather. The power density of a fusion power plant is also much larger than for most renewable energy sources. As a result, it takes significantly less space to achieve a given power output with fusion power than with solar power or wind power. Furthermore, fusion power may not represent a renewable energy source, but the existing stores of easily available fuel could potentially last us thousands if not millions of years. Finally, unlike fission power plants, which also possess most of the previously mentioned qualities, fusion power plants can truly be said to be safe, as it is physically impossible for a fusion power plant to experience a meltdown. This, in particular, should make it much easier to garner public support for fusion power. Fusion power thus has the potential to supply us with stable, clean and safe energy for thousands of years if not more, thereby significantly alleviating the issues with global warming. [7, p. 3-18] [15] [9] [8] [12] Unfortunately, however, nobody has yet managed to construct an actual fusion reactor capable of steady-state operation with a net energy gain, though the ITER-reactor currently being built in France is meant to demonstrate the feasibility of the concept. [53] [15] [7, p. 3-19] Plans are also being made for a follow-up DEMO reactor meant to be the next step between ITER and an actual commercial fusion power plant. [31] [32] [22] [23] As such, the myriad technologies and methods necessary to construct and operate a fusion reactor continue to be the focus of a lot of research. [4] [48] [5] [45] [26] [28] [49] [17] [19]

One important aspect of running a fusion reactor is the need to continuously diagnose and control the fusion plasma. In particular, the velocity distributions of the fast ions are of great interest, as they relate to the heating of the plasma and can be used to investigate certain instabilities. These distributions can be obtained from the spectra measured by various plasma diagnostic instruments via the use of velocity-space tomography. [17] [19] [41] [40] [43] [44] However, most currently used tomographic models are numerical and, while accurate, relatively slow. [38] [39] [36] [11] [50] This makes them unsuited to the task of quickly judging the suitability of a given diagnostic setup for characterizing a given expected velocity distribution function. As such, we have chosen to focus on writing and

testing analytic models for velocity-space tomography for fusion plasmas. These much less time-consuming analytic models can then be used to determine a potentially suitable diagnostic setup for a given situation, after which the numerical models can be used to obtain more accurate results. The testing is carried out by generating synthetic spectra based on known distribution functions and then using the analytic models to reconstruct distribution functions from these spectra. The known and calculated distribution functions can then be compared in order to evaluate the model.

2 Theory

In the field of velocity-space tomography for fusion plasmas, two different coordinate systems are typically used to describe the fast-ion velocity distribution functions. These are the (E, p)-coordinate system and the $(v_\parallel, v_\perp)$-coordinate system. E is the kinetic energy of the fast ion and p is its pitch, which is defined as the ratio between the ion velocity parallel to the magnetic field, $v_\parallel$, and the total ion velocity, $v = \sqrt{v_\parallel^2 + v_\perp^2}$, i.e. $p = \frac{v_\parallel}{v}$. The pitch is also sometimes defined as $p = -\frac{v_\parallel}{v}$, but we will use the positive definition. A complete description of the fast-ion velocities naturally requires two mutually perpendicular velocity components that are perpendicular to the magnetic field, for a total of three velocity components, given that the ions move in 3D. These perpendicular velocity components, $v_{\perp,1}$ and $v_{\perp,2}$, are related to the total perpendicular velocity, $v_\perp$, via the gyro-angle, γ, which is the angle between $v_{\perp,1}$ and $v_\perp$. [17] [19] [38] [42] [41]

$$v_{\perp,1} = v_\perp \cos(\gamma);\ v_{\perp,2} = v_\perp \sin(\gamma) \tag{2.1}$$

However, we know that the ions move in a spiraling motion along the magnetic field. As such, we can assume rotational symmetry around the magnetic field line so long as the gyro-motion of the ions occurs on a time-scale that is short compared to other important time-scales. This assumption of rotational symmetry allows us to switch to a cylindrical coordinate system and integrate out the gyro-angle, thereby reducing the problem to two dimensions and justifying the use of the $(v_\parallel, v_\perp)$-coordinate system. In this cylindrical coordinate system, $v_\parallel$ can be positive or negative while $v_\perp$ can only be positive. The relation between the full 3D distribution function, f^{3D}, and the distribution function, f^{2D}, in the 2D cylindrical coordinate system also becomes $f^{2D} = 2\pi v_\perp f^{3D}$. [17] [19] [38] [42] [35] The (E, p)-coordinate system is similarly two-dimensional since it only contains information about $v_\perp$ and not $v_{\perp,1}$ or $v_{\perp,2}$. All of the distribution functions that will be plotted in this report are thus two-dimensional with no implied third direction. The two coordinate systems are related to one another via the following simple expressions where m is the mass of the fast ion in question. [17] [19] [38]

$$E = \frac{1}{2} m (v_\parallel^2 + v_\perp^2);\ p = \frac{v_\parallel}{\sqrt{v_\parallel^2 + v_\perp^2}} \tag{2.2}$$

$$v_\parallel = p\sqrt{\frac{2E}{m}};\ v_\perp = \sqrt{1-p^2}\sqrt{\frac{2E}{m}} \tag{2.3}$$

The (E,p)-coordinate system is often used because it allows one to read the energies of the ions directly from the plot. [17] [19] A lot of the mathematical expressions are simpler in ($v_\parallel$,$v_\perp$)-coordinates, however, and some features are more easily discernible when using these coordinates. As a result, we will be using both coordinate systems.

The goal of velocity-space tomography, as mentioned in the introduction, is to calculate the fast-ion velocity distribution functions from the spectra measured by various diagnostic instruments. The starting point for the solution of this inverse problem is the corresponding forward problem of calculating the spectra that arise from a given fast-ion velocity distribution function. This forward problem can be solved using eq. 2.4, which relates a part of a measured spectrum, $s(x_1, x_2, \phi)$, to the fast-ion velocity distribution function, $f(E,p)$, via the so-called weight function, $w(x_1, x_2, \phi, E, p)$. [17] [19] [37]

$$s(x_1, x_2, \phi) = \int\int w(x_1, x_2, \phi, E, p) f(E,p)\,\mathrm{d}E\mathrm{d}p \tag{2.4}$$

Note that while this equation is obviously written for the (E,p)-coordinate system, the form would be exactly the same for the $(v_\parallel, v_\perp)$-coordinate system, except for E and p being replaced by $v_\parallel$ and $v_\perp$. The same can be said for the following derivations. In eq. 2.4, x_1 and x_2 denote the part of the measured spectrum that the particular weight function corresponds to. In the case of neutron emission spectrometry, for instance, where one measures a neutron energy spectrum, x_1 and x_2 will denote the interval in neutron energy that the particular weight function corresponds to and $s(x_1, x_2, \phi)$ will be the number of measured neutrons with energies in this interval. Furthermore, while x_1 and x_2 could theoretically have the same value, a real measurement will always be associated with an interval due to the finite spectral resolution of a real measurement instrument.
ϕ is the so-called observation angle between the magnetic field and the line-of-sight of the particular diagnostic instrument. Both the degree of influence and the region of velocity-space that influences a measurement by a particular diagnostic in a given interval are thus given by the corresponding weight function and depend on the interval, the observation angle and the type of diagnostic. For this reason, weight functions are also sometimes called fast-ion velocity-space sensitivity functions, as they indicate the sensitivity of a particular measurement to different regions of velocity-space. Eq. 2.4 can be handled more easily by approximating the integrals with sums, which yields eq. 2.5. [17] [19] [40]

$$S_k = \sum_i \sum_j W_{i,j,k} F_{i,j} \Delta E \Delta p \tag{2.5}$$

The indices i and j denote different grid points in the (E,p)-coordinate system. This means that the sums run over different values of energy and pitch and the

total number of elements, n, is equal to the number of grid points in the considered coordinate system. The index k, meanwhile, denotes the k'th measurement variable bin or interval for the particular diagnostic instrument. One thus has a k and an equation of the form given by eq. 2.5 for every measurement variable bin under consideration. If we then include $\Delta E \Delta p$ in W, collect the different $W_{i,j,k}$-values for a single k in an ordered $(1 \times n)$ row vector and the different $F_{i,j}$-values in a similarly ordered $(n \times 1)$ column vector, eq. 2.5 can be reduced to the following expression.

$$S_k = \mathbf{W}_k \mathbf{F} \tag{2.6}$$

We will thus have an equation of the form given by eq. 2.6 for every measurement variable bin for every diagnostic instrument. However, we know that $\mathbf{F}$ will be the same in all of these equations since the actual velocity distribution function is set. As such, we can combine the m different versions of eq. 2.6 for a single diagnostic instrument into a single expression by collecting the different S_k-values in an ordered $(m \times 1)$ column vector and the different $\mathbf{W}_k$ vectors in an ordered $(m \times n)$ matrix. This yields the expression in eq. 2.7. It is also common to normalize each $\mathbf{W}_k$ vector by the uncertainty of the associated diagnostic instrument in order to account for this uncertainty. [17] [19] [37]

$$\mathbf{S} = \mathbf{W}\mathbf{F} \tag{2.7}$$

Furthermore, if we assume that we have q different diagnostic instruments where m_p is the number of measurement variable bins for instrument p, we can combine the q different versions of eq. 2.7 into a single expression by successively appending the $\mathbf{S}$ and $\mathbf{W}$ for each diagnostic instrument to those for the previous diagnostic instrument. The result is an expression of the same form as eq. 2.7 except that $\mathbf{S}$ is a $(M \times 1)$ column vector and $\mathbf{W}$ is a $(M \times n)$ matrix where $M = \sum_p m_p$. This expression completely defines the forward problem and is used to generate synthetic spectra from known model distribution functions. [17] [13] [37]

In theory, we now only need to invert eq. 2.7 in order to determine the velocity distribution function, $\mathbf{F}$, from the measured spectra, $\mathbf{S}$. However, the $\mathbf{W}$-matrix is almost never quadratic, and the inverse problem is ill-posed. Well-posed problems were defined by Hadamard as those problems whose solutions fulfill the three requirements of existence, uniqueness and stability. This means that the solution must exist, be unique and depend continuously on the data so that it does not change drastically for small changes in the data. If one or more of these requirements is not met, the problem is said to be ill-posed. Inverse problems are often ill-posed and most commonly violate the stability requirement. [17] [13] [37] [44] As such, we cannot simply calculate the inverse matrix of $\mathbf{W}$ and use it to determine the velocity distribution, $\mathbf{F}$, that gave rise to a given set of measured spectra, $\mathbf{S}$. Instead, we need to regularize the inverse problem in order to obtain a solution.

This can be done using a number of different tomographic inversion methods such as truncated singular value decomposition (SVD), maximum entropy regularization, minimum Fisher information regularization and Tikhonov regularization.

[19] We have chosen to focus on Tikhonov regularization and, in particular, first-order Tikhonov regularization because it has been shown to yield some of the best results, as seen in [19]. Tikhonov regularization is based on the principle that the ill-posed inverse problem represented by eq. 2.7 can be formulated as the following ill-posed least-squares problem. [19] [17] [37] [44]

$$\text{minimize} \left\{ ||\mathbf{WF} - \mathbf{S}||^2 \right\} \tag{2.8}$$

The idea is then to replace this ill-posed least-squares problem with a closely related well-posed least squares problem by introducing the $(n \times n)$ regularization matrix $\mathbf{L}$. [19] [17] [37] [44]

$$\text{minimize} \left\{ ||\mathbf{WF} - \mathbf{S}||^2 + \alpha ||\mathbf{LF}||^2 \right\} \tag{2.9}$$

In eq. 2.9, α is a non-negative number that determines the weight of the regularization term. The larger the value of α, the more the solution is regularized. A too large value of α will thus result in the solution being overly regularized and excessively smoothed, while a too low value will result in the regularization being essentially non-existent. A number of methods exist that can be used to determine the "optimal" value of α, but we choose to pick the values of α manually. This is based on the fact that the existing methods are imperfect and require calculation of solutions as well as the fact that the speed of the analytic methods makes adjustments of α simple. The Tikhonov solution, $\mathbf{F}_\alpha$, to the well-posed least-squares problem in eq. 2.9 then has the following form. [19] [17]

$$\mathbf{F}_\alpha = \left(\mathbf{W}^T\mathbf{W} + \alpha \mathbf{L}^T\mathbf{L} \right)^{-1} \mathbf{W}^T\mathbf{S} \tag{2.10}$$

The nature of the Tikhonov regularization depends on the form of the regularization matrix (penalty operator) $\mathbf{L}^T\mathbf{L}$. For zeroth-order Tikhonov regularization, $\mathbf{L}^T\mathbf{L}$ is an identity matrix. This type of Tikhonov regularization penalizes large absolute values of the velocity distribution function. First-order Tikhonov regularization, on the other hand, penalizes large gradients and uses a gradient operator as the penalty operator. [19] [17] [37] In ($v_\parallel$,$v_\perp$)-coordinates, this penalty operator has the form

$$\mathbf{L}^T\mathbf{L} = \nabla_{v_\parallel}^T \nabla_{v_\parallel} + \nabla_{v_\perp}^T \nabla_{v_\perp} \tag{2.11}$$

and in (E, p)-coordinates, it has the form

$$\mathbf{L}^T\mathbf{L} = 2mE\nabla_E^T\nabla_E + \frac{m}{2E}(1 - p^2)\nabla_p^T\nabla_p \tag{2.12}$$

where m is the mass of the fast ion. These expressions were derived in [17]. It should be noted that the gradient operators in eqs. 2.11 and 2.12 are first order finite-difference approximations of the actual partial derivatives. Their forms are given in [1]. Finally, one can perform a mix of zeroth-order and first-order Tikhonov regularization where the penalty operator is equal to the penalty operator for first-order Tikhonov regularization plus a small value (1×10^{-12}, in

our case) times an identity matrix. Though we will include the option to perform zeroth-order Tikhonov regularization or the mix of zeroth- and first-order Tikhonov regularization in our analytic models, we will primarily be focusing on first-order Tikhonov regularization. This is due to the fact that first-order Tikhonov regularization, as previously mentioned, has been shown to yield some of the best results. It is also possible to constrain Tikhonov regularization to only yield non-negative values, and we will include the option to do so, but this significantly increases the computation time. [19] [37]

As mentioned in the introduction, the testing of our analytic models will be done by generating synthetic spectra from known model distribution functions, reconstructing distribution functions from these synthetic spectra and then comparing the reconstructions to the model. In order to more accurately mimic the realistic situation of reconstructing the distribution function from measured spectra, it is common to use a forward grid for the generation of synthetic spectra and then reconstruct the distribution function on a different tomography grid with fewer grid points. Using the same forward and tomography grid is referred to as an inverse crime. [52] [21]
The quality of a reconstructed distribution function obtained via Tikhonov regularization can also be evaluated using the quality factor, Q. The Q-value is calculated as

$$Q = \sum_i \sum_j \frac{|f_{i,j} - f_{tomo_{i,j}}|^2}{f_{i,j}^2} \tag{2.13}$$

where f refers to the model distribution, f_{tomo} refers to the reconstructed distribution, and the sums run over all of the grid points in the considered tomography grid. Determining Q thus requires one to calculate the model distribution on the tomography grid. Q is a measure of how well the reconstruction matches the model distribution and the lower Q is, the better they match. $Q = 0$ corresponds to a perfect match. Note that since Q is calculated by comparing the values at each grid point, the reconstruction with the lowest Q might not be the one that visually seems to resemble the model best. [51]
Before we can make use of Tikhonov regularization, however, we also need to determine the weight functions as the $\mathbf{W}$-matrix enters in eq. 2.10. The expressions for the weight functions depend on the type of diagnostic instrument under consideration. We will thus now go through the five different diagnostic methods that we will be considering and derive the related weight functions. These methods are Collective Thomson Scattering (CTS), Neutron Emission Spectrometry (NES), one- and two-step reaction gamma-ray spectrometry (GRS) and Fast-ion D_α spectroscopy (FIDA).

2.1 CTS weight functions

A CTS-diagnostic is technically a two-part device that consists of a probe beam source and a receiver. It is, however, common to use one of the existing heating devices such as an electron cyclotron resonance heating (ECRH) gyrotron as the probe beam source. When used as a probe beam source, the gyrotron is operated at a different frequency than for heating, such that the electron cyclotron resonances are outside the plasma. The source injects a probe beam with wave vector $\vec{k}^i$ which, for sufficiently large wavelengths of the injected probe beam, scatters off the collective fluctuations of electrons moving under the influence of the fast ions. The resulting scattered wave with wave vector $\vec{k}^s$ is then measured by the receiver. This allows one to determine the wave vector $\vec{k}^\delta = \vec{k}^s - \vec{k}^i$ of the resolved plasma fluctuations and the corresponding frequency, ν^δ, of the plasma wave. Using these values, it is then possible to infer the projected velocity, u, along $\vec{k}^\delta$ of the fast ion setting up the fluctuation via the following relation where k^δ is the length of $\vec{k}^\delta$. [17] [41] [40] [2]

$$\nu^\delta \approx \frac{uk^\delta}{2\pi} \tag{2.14}$$

The distribution, $g(u)$, of these projected velocities is thus the normally desired output spectrum from a CTS-diagnostic. The observation angle, ϕ, between the "line-of-sight" direction of $\vec{k}^\delta$ and the magnetic field line is determined by the way the diagnostic is set up. The weight function for a given interval in the spectrum from a certain CTS-diagnostic is thus given by the probability of measuring projected velocities in that interval for certain values of $(v_\parallel, v_\perp)$ with that diagnostic. Additionally, one should divide this probability by the size of the interval in order to ensure that the weight function is per unit velocity so that it matches the spectra. After all, the spectra will always be densities in the measurement variable, which, in the case of CTS, is velocity. The probability can be determined by integrating the probability density function (pdf) in u for the given values of $v_\parallel, v_\perp$ and ϕ over the considered interval. This leads to the following expression for the weight function. [41] [17] [38]

$$w(u_1, u_2, v_\parallel, v_\perp, \phi) = \frac{1}{u_2 - u_1}\text{prob}(u_1 < u < u_2 | v_\parallel, v_\perp, \phi) = \frac{1}{u_2 - u_1}\int_{u_1}^{u_2} \text{pdf}_u \, \mathrm{d}u \tag{2.15}$$

However, while an expression for pdf_u can be derived, as was done in [41], the finite measurement intervals result in this pdf not accurately reflecting the measurements. Instead, one can make use of the fact that it is possible to relate the projected velocity u to the gyro-angle Γ of the fast ion in order to transform the integral into one over gyro-angle. Due to the previously mentioned assumption of rotational symmetry, every gyro-angle is equally likely and thus the probability density function in Γ must be uniform, i.e. $\text{pdf}_\Gamma = \frac{1}{2\pi}$. The relation between u and Γ can most easily be explained by defining a coordinate system where $\Gamma = 0$ corresponds to the velocity vector of the fast ion lying in the plane defined by

the line-of-sight (LOS) direction and the direction of the magnetic field line. This coordinate system and the projection of an arbitrary fast ion velocity $(v_\parallel, v_\perp)$ onto the LOS are illustrated in Fig. 2.1. [41] [17] [38]

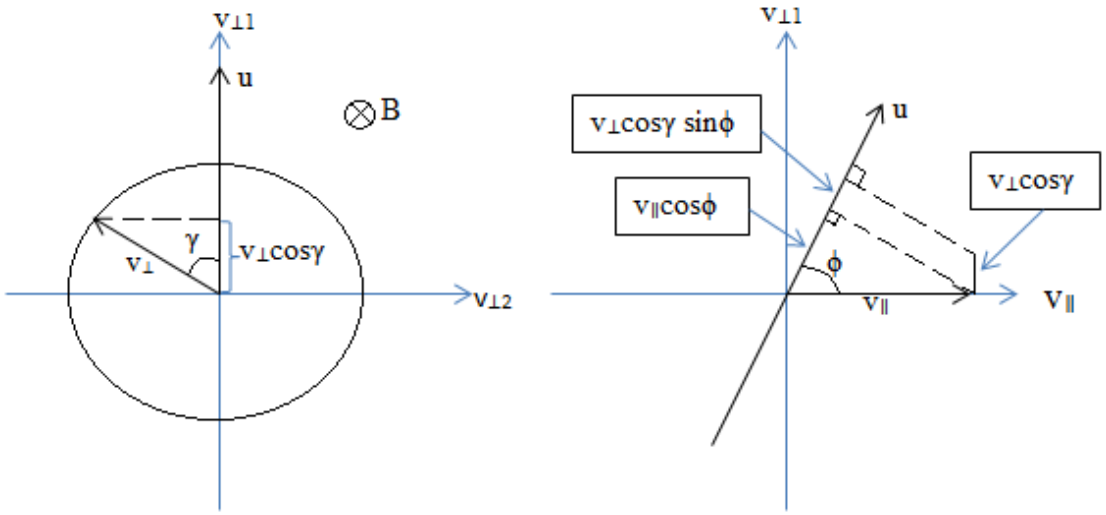


Figure 2.1: An illustration of the projection of a fast ion velocity $(v_\parallel, v_\perp)$ onto the line-of-sight.

Based on Fig. 2.1, we obtain the following expression for the relation between the projected velocity u and the gyro-angle Γ for given values of $v_\parallel, v_\perp$ and ϕ.

$$u = v_\parallel \cos(\phi) + v_\perp \sin(\phi)\cos(\Gamma) \Leftrightarrow \Gamma = \arccos\left(\frac{u - v_\parallel \cos(\phi)}{v_\perp \sin(\phi)}\right) \tag{2.16}$$

Using this relation, we can transform the integration limits in eq. 2.15 from u to Γ. However, the nature of the cosine function means that for every solution Γ_1 to the left side of eq. 2.16, there exists another solution $\Gamma_1' = 2\pi - \Gamma_1$. As a result, we get two integrals instead of one when we transform to Γ-limits. It can also be shown that when we transform from $\mathrm{pdf}_u\, \mathrm{d}u$ to $\mathrm{pdf}_\Gamma\, \mathrm{d}\Gamma$, the integral with the first set of Γ-limits gets a minus sign. [41] [17] [38] As such, we obtain the following new expression for the CTS weight function.

$$w(u_1, u_2, v_\parallel, v_\perp, \phi) = \frac{1}{u_2 - u_1}\left(\int_{\Gamma_1}^{\Gamma_2} -\mathrm{pdf}_\Gamma\, \mathrm{d}\Gamma + \int_{2\pi-\Gamma_1}^{2\pi-\Gamma_2} \mathrm{pdf}_\Gamma\, \mathrm{d}\Gamma\right) \tag{2.17}$$

$$w(u_1, u_2, v_\parallel, v_\perp, \phi) = \frac{1}{u_2 - u_1}\frac{1}{2\pi}\left(\Gamma_1 - \Gamma_2 + 2\pi - \Gamma_2 - (2\pi - \Gamma_1)\right) \tag{2.18}$$

$$w(u_1, u_2, v_\parallel, v_\perp, \phi) = \frac{1}{u_2 - u_1}\frac{\Gamma_1 - \Gamma_2}{\pi} \tag{2.19}$$

The values of Γ_1 and Γ_2 corresponding to u_1 and u_2 are calculated using eq. 2.16. This can also easily be done in (E, p)-coordinates by using the relations in eq. 2.3

to replace $(v_\parallel, v_\perp)$ with (E, p). The result is shown below.

$$\Gamma = \arccos\left(\frac{u - p\sqrt{\frac{2E}{m}}\cos(\phi)}{\sqrt{1-p^2}\sqrt{\frac{2E}{m}}\sin(\phi)}\right) \tag{2.20}$$

It should, however, be noted that the argument of the arccos-function in eqs. 2.16 and 2.20 can potentially lie outside the range $[-1; 1]$, which results in complex values. We thus take the real part of the two gyro-angles in order to obtain physically meaningful results. [17] [38] In this way, it is possible to determine the CTS weight functions. Some example weight functions for certain values of u_1, u_2 and ϕ are shown in $(v_\parallel, v_\perp)$-coordinates in Fig. 2.2.

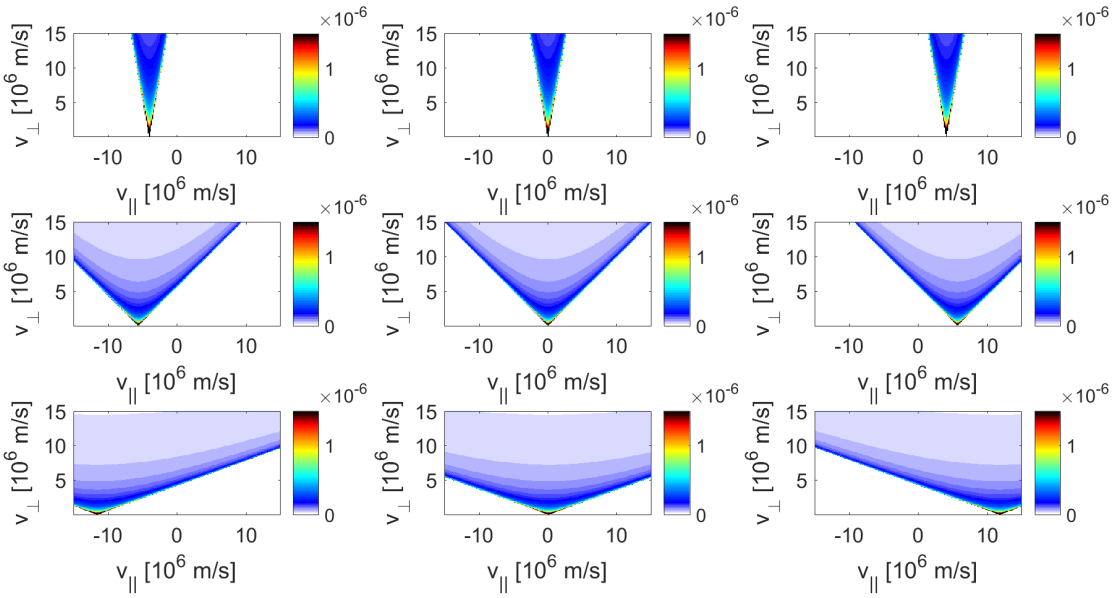


Figure 2.2: Examples of CTS weight functions in $(v_\parallel, v_\perp)$-coordinates. All of the weight functions are plotted for the same interval size $u_2 - u_1 = 0.1 \times 10^6$ m/s. Going from left to right, the weight functions are for $u_1 = -4 \times 10^6$ m/s,$u_1 = 0$ m/s and $u_1 = 4 \times 10^6$ m/s. Going from top to bottom, the observation angles are $\phi = 10°$, $\phi = 45°$ and $\phi = 70°$.

Based on Fig. 2.2, we can easily see that the regions of velocity-space covered by a particular weight function depend on ϕ and the considered u-interval. In the case of existing spectra being used for tomographic inversion, the u-intervals to be used for the weight functions must naturally be equal to those used in the spectra. As previously mentioned, however, we also wish to generate synthetic spectra from known velocity distribution functions so that we can use them to test the various analytic models. We thus need a way to determine the u-intervals that should be used to calculate the weight functions needed for the generation of synthetic spectra. The size of the intervals can be indirectly determined by demanding that all intervals have the same size and setting the number of intervals as an input, but we still need a way to determine the range in u-values that should be partitioned into equal-sized intervals. We also need to ensure that the weight functions arising

from the chosen range in u-values cover the entire region of velocity-space under consideration. To this end, we determine the largest possible positive (u_{max}) and negative (u_{min}) value of the projected velocity for the considered region of velocity-space and let the range run from the negative value to the positive one. After all, the projected velocity can be either positive or negative depending on whether the ion is moving towards or away from the diagnostic. In (E, p)-coordinates, both of these values will have the same magnitude given that one generally always includes the entire pitch-range meaning that all possible directions of the ion velocity are accounted for. The magnitude of the two values is naturally just the velocity of the fast ion with mass m that corresponds to the largest considered E-value.

$$\text{for } (E,p):\ u_{max} = |u_{min}| = \sqrt{\frac{2E}{m}} \tag{2.21}$$

In $(v_\parallel, v_\perp)$-coordinates, the expressions for u_{max} and u_{min} also depend on ϕ. For $\phi \in [0; \pi/2]$, u_{min} is obtained by using eq. 2.16 with the largest negative value of $v_\parallel$, the largest value of $v_\perp$ and setting $\cos(\Gamma) = -1$, while u_{max} is obtained by using eq. 2.16 with the largest positive value of $v_\parallel$, the largest value of $v_\perp$ and setting $\cos(\Gamma) = 1$.

$$\text{for } (v_\parallel, v_\perp), \phi \in [0; \pi/2]:\ u_{min} = v_{\parallel,min}\cos(\theta) - v_{\perp,max}\sin(\theta) \tag{2.22}$$

$$\text{for } (v_\parallel, v_\perp), \phi \in [0; \pi/2]:\ u_{max} = v_{\parallel,max}\cos(\theta) + v_{\perp,max}\sin(\theta) \tag{2.23}$$

For $\phi \in]\pi/2; \pi]$, the formula are the same except that the positive and negative values of $v_\parallel$ are swapped.

$$\text{for } (v_\parallel, v_\perp), \phi \in]\pi/2; \pi]:\ u_{min} = v_{\parallel,max}\cos(\theta) - v_{\perp,max}\sin(\theta) \tag{2.24}$$

$$\text{for } (v_\parallel, v_\perp), \phi \in]\pi/2; \pi]:\ u_{max} = v_{\parallel,min}\cos(\theta) + v_{\perp,max}\sin(\theta) \tag{2.25}$$

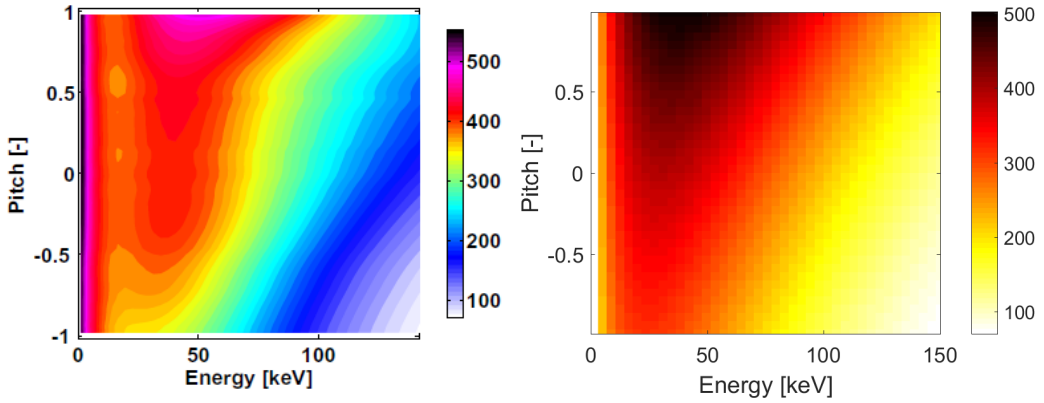
2.2 FIDA weight functions

FIDA spectroscopy somewhat resembles CTS in that it requires the use of a neutral beam injector (NBI) in addition to the actual FIDA-diagnostic. It can thus also be considered a two-part diagnostic. The NBI is used to inject neutral deuterium-atoms into the plasma where they can undergo charge-exchange reactions with fast deuterium-ions. The resulting neutralized fast deuterium-atoms are unaffected by the magnetic field and will continue on straight paths. They will also often be in an excited state after the charge-exchange reaction or become excited through collisions with other particles in the plasma. The excited atoms quickly decay and emit a photon. If the atom decays from the $n = 3$ to the $n = 2$ excited state, the process is called a Balmer-α transition and the resulting photon is a D_α-photon. These photons have wavelengths around 656 nm, but the wavelength measured by the FIDA-diagnostic will be Doppler-shifted by the projected velocity of the emitting fast atoms onto the line-of-sight of the diagnostic. The FIDA-diagnostic can,

however, only detect photons that are emitted towards it. It is technically also possible for thermal deuterium-ions to undergo charge-exchange reactions with the injected neutrals and emit D_α-photons, but the resulting Doppler-shifts will be much smaller due to the smaller velocities of the thermal ions. Furthermore, as the name implies, FIDA spectroscopy involves only considering the photons resulting from fast ions, though the physics are the same in both cases. As such, we may focus exclusively on FIDA spectroscopy, but the methods can, in fact, also be used for regular deuterium-based charge-exchange recombination spectroscopy. [38] [17] [11] [50] [34]
The normally desired output from a FIDA-diagnostic is thus a spectrum of Doppler-shifted D_α-light. Much like with CTS-diagnostics, the weight function for a given interval in the spectrum from a FIDA-diagnostic thus involves the probability of measuring photon-wavelengths in that interval for certain values of $(v_\parallel, v_\perp)$ with that diagnostic. The observation angle ϕ is determined by the way that particular diagnostic is set up. One should also still divide by the size of the interval for reasons similar to those mentioned for the CTS weight functions. However, the FIDA weight functions additionally include a rate part, R, that indicates the total FIDA intensity per fast ion regardless of the wavelength. This is related to the fact that we no longer measure on the fast ions themselves. As such, the FIDA weight functions have the following general form. [38] [17]

$$w(\lambda_1, \lambda_2, v_\parallel, v_\perp, \phi) = \frac{1}{\lambda_2 - \lambda_1} R(v_\parallel, v_\perp) \text{prob}(\lambda_1 < \lambda < \lambda_2 | v_\parallel, v_\perp, \phi) \tag{2.26}$$

It is also worth noting that eq. 2.16 is still perfectly valid even though the diagnostic does not actually measure the projected velocity u. We just have to remember that the LOS direction that we project onto now lies along the actual LOS of the FIDA-diagnostic rather than the direction of $\vec{k}^\delta$. In fact, eq. 2.16 is valid for all of the diagnostics to be considered so long as we use the correct LOS.
Unfortunately, no one has yet managed to derive an analytic model for the FIDA intensity function R. Instead, we have created an ad-hoc model for the R-part based on the numerically calculated R-part that was plotted in [38]. This R-part was calculated for an injection energy of 60 keV using the FIDASIM model. We found that we could recreate the R-part fairly well by modeling it as a bi-Maxwellian distribution (see section 2.6) with $n_i = 1 \times 10^8\,\text{m}^{-3}$, $T_{para} = T_{perp} = 60\,\text{keV}$ and $v_{d\parallel} = 0.4 \times 10^6\,\text{m/s}$. The numerically calculated R-part from [38] is shown in Fig. 2.3a, and our ad-hoc model is shown in Fig. 2.3b. Our ad-hoc model is seen to resemble the numerically calculated FIDA R-part quite well. It is thus suitable for use in the weight functions. This does, however, limit our use of the FIDA weight functions to the region of velocity-space represented by the plots in Fig. 2.3, as we cannot be sure that our ad-hoc model is reasonable outside of this region.



(a) (b)

Figure 2.3: a) FIDA R-part calculated numerically for an injection energy of 60 keV using FIDASIM. Copied from [38] with permission. b) FIDA R-part modeled as a bi-Maxwellian distribution with $n_i = 1 \times 10^8\,\mathrm{m}^{-3}$, $T_{para} = T_{perp} = 60\,\mathrm{keV}$ and $v_{d,para} = 0.4 \times 10^6\,\mathrm{m/s}$.

Furthermore, due to the nature of the FIDA emission process, the expression for the probability part is more complicated than the one used for CTS. For one, when a deuterium atom moving in the magnetic field of a fusion reactor undergoes a Balmer-α transition, it actually emits light at 15 different wavelengths due to so-called Stark splitting caused by the electric field in the reference frame of the moving atom, $\tilde{E}$. This means that the probability part cannot be determined by a single integral, but must involve a sum over the 15 lines. The wavelengths of the 15 Stark lines are related to the wavelength $\lambda_0 = 656.1\,\mathrm{nm}$ of the unshifted D_α-line by the equation [38]

$$\lambda_l = \lambda_0 + s_l \tilde{E} \tag{2.27}$$

where l denotes one of the 15 lines and the constants s_l are [38]

$$\begin{aligned} s_{l=1,\ldots,15} = (&-220.2, -165.2, -137.7, -110.2, -82.64, -55.1, -27.56, 0, \\ &27.57, 55.15, 82.74, 110.3, 138.0, 165.6, 220.9) \times 10^{-18}\,\mathrm{m}^2/\mathrm{V} \end{aligned} \tag{2.28}$$

Line nr. 8 with $s_8 = 0$ is thus the unshifted D_α-line. If there is no electric field in the laboratory reference frame, which is a reasonable assumption in most cases, $\tilde{E}$ is simply given by $\tilde{E} = v_\perp B$ where B is the magnetic field strength. The Stark splitting causes additional complications because a measurement at a given wavelength can potentially have been caused by any of the 15 lines, so long as the Doppler-shift has the appropriate size. Seeing as the Doppler-shift is given by $\lambda - \lambda_0 = u\lambda_0/c$, where u is the projected velocity of the emitting D-atom, the 15 conditions for the 15 lines to each be observable at a given wavelength λ are given by the following expression. [38]

$$\lambda = (\lambda_0 + s_l v_\perp B)\left(1 + \frac{u_l}{c}\right) \tag{2.29}$$

In other words, for every Stark line, there is a corresponding projected velocity u_l of the emitting D-atom that results in that line being observable at the wavelength λ. Since we always consider weight functions for set values of $(v_\parallel, v_\perp)$, and ϕ is determined by the diagnostic setup, the only parameter we can change in order to achieve different projected velocities is the gyro-angle. Combining eq. 2.29 with eq. 2.16 thus yields the following expression.

$$\lambda = (\lambda_0 + s_l v_\perp B)\left(1 + \frac{v_\parallel \cos(\phi) + v_\perp \sin(\phi)\cos(\Gamma_l)}{c}\right) \tag{2.30}$$

We isolate Γ_l.

$$\Gamma_l = \arccos\left(\frac{c\left(\frac{\lambda}{\lambda_0 + s_l v_\perp B} - 1\right) - v_\parallel \cos(\phi)}{v_\perp \sin(\phi)}\right) \tag{2.31}$$

This expression allows one to determine the integration limits in Γ for each of the 15 lines. As with CTS, there exists another solution $\Gamma' = 2\pi - \Gamma$ for every solution Γ, and we take the real part of the gyro-angles in order to obtain physically meaningful results. We thus have a sum with 15 terms of two integrals. Additionally, we need to include the probability that a detected photon for gyro-angle Γ comes from line l in the integrals. This probability is given by [38]

$$\text{prob}(l|\Gamma) = \hat{C}_l\left(1 \pm \sin^2(\phi)\sin^2(\Gamma)\right) \tag{2.32}$$

where the plus is used for σ-lines and the minus is used for π-lines. The σ-lines are lines 2, 3, 7-9, 13 and 14 while the π-lines are lines 1, 4-6, 10-12 and 15. The constants $\hat{C}_l$ are [38]

$$\hat{C}_l = (1, 18, 16, 1681, 2304, 729, 1936, 5490, \\ 1936, 729, 2304, 1681, 16, 18, 1) \times \frac{1}{18860} \tag{2.33}$$

The expression for the probability part now has the following form where pdf_{D_α} is the probability density function for D_α. [38]

$$\text{prob}(\lambda_1 < \lambda < \lambda_2 | v_\parallel, v_\perp, \phi) = \sum_{l=1}^{15}\left(\int_{\Gamma_{2,l}}^{\Gamma_{1,l}} \text{prob}(l|\Gamma)\text{pdf}_{D_\alpha}\, d\Gamma + \int_{\Gamma'_{1,l}}^{\Gamma'_{2,l}} \text{prob}(l|\Gamma)\text{pdf}_{D_\alpha}\, d\Gamma\right) \tag{2.34}$$

This expression is more complicated than the one obtained for CTS. Furthermore, one cannot simply assume that pdf_{D_α} is uniform the way it was done for CTS. It is, in fact, a rather complicated expression that depends on both the used NBI and the ion/electron temperatures as well as the drift velocities. This true pdf is generally treated numerically and is very difficult to work with analytically. [38] Instead, we choose to make use of the simple model for pdf_{D_α} that was presented in [38]. This model has the form

$$\text{pdf}_{D_\alpha} = \frac{1}{2\pi} + a\cos\left(\Gamma + \bar{\Gamma}\right) \tag{2.35}$$

where $\bar{\Gamma}$ is a phase shift, and the amplitude a is modeled by

$$a = \frac{E}{E_0}(1 - p^2) = \frac{v_\perp^2}{v_{\perp 0}^2} \tag{2.36}$$

where $v_{\perp 0} = \sqrt{\frac{2E_0}{m_f}}$ and E_0 is a free parameter. It was shown in [38] that $E_0 = 1\,\text{MeV}$ yields reasonable results for the region of velocity-space that our FIDA weight functions are limited to. Inserting this model for pdf_{D_α} and eq. 2.32 in eq. 2.34 and solving the integrals yields the following expression for the probability part. [38]

$$\begin{aligned}\text{prob}(\lambda_1 < \lambda < \lambda_2 | v_\parallel, v_\perp, \phi) = \sum_{l=1}^{15} \hat{C}_l \Bigg(& \frac{\Gamma_{1,l} - \Gamma_{2,l}}{\pi} \pm \frac{\sin^2(\phi)}{2} \times \\ & \left(\frac{\Gamma_{1,l} - \Gamma_{2,l}}{\pi} - \frac{\sin(2\Gamma_{1,l}) - \sin(2\Gamma_{2,l})}{2\pi} \right) \\ & + 2a\cos(\bar{\Gamma}) \left(\sin(\Gamma_{1,l}) - \sin(\Gamma_{2,l}) \pm \frac{\sin^2(\phi)}{3} \times \left(\sin^3(\Gamma_{1,l}) - \sin^3(\Gamma_{2,l}) \right) \right) \Bigg)\end{aligned} \tag{2.37}$$

The $\pm$-signs are plus for the σ-lines and minus for the π-lines. This expression is most easily applied to an (E, p)-grid by using the relations in eq. 2.3 to calculate the corresponding $(v_\parallel, v_\perp)$-values and inserting these. We now have everything we need to calculate FIDA weight functions. Some example weight functions for $B = 1.74\,\text{T}$, $\bar{\Gamma} = 0$ and certain values of λ_1, λ_2 and ϕ are shown in Fig. 2.4.

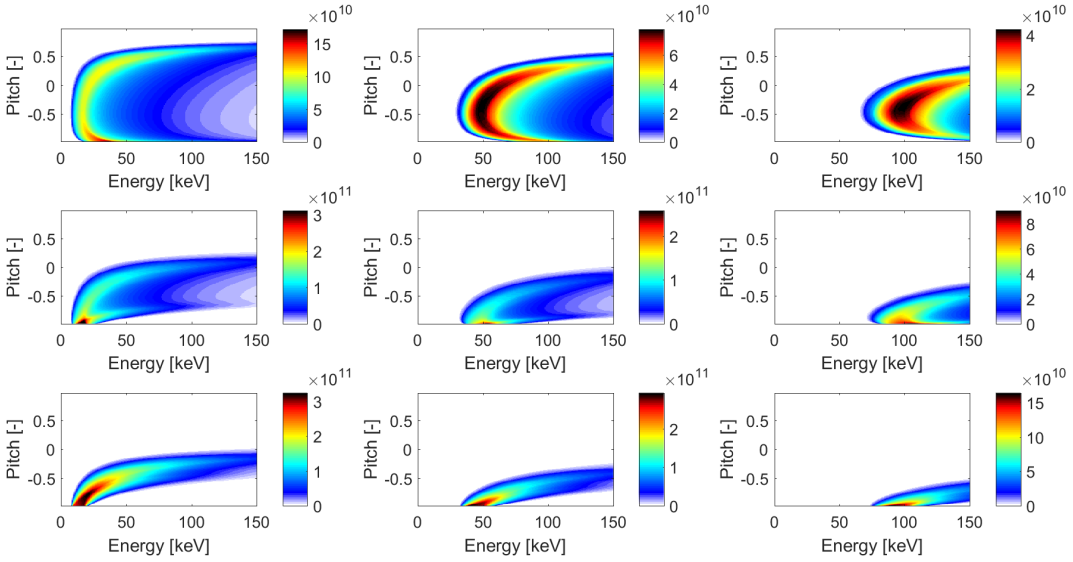


Figure 2.4: Examples of FIDA weight functions for $B = 1.74\,\text{T}$ and $\bar{\Gamma} = 0$ in (E, p)-coordinates. All of the weight functions are plotted for the same interval size $\lambda_2 - \lambda_1 = 1\,\text{nm}$. Going from left to right, the weight functions are for $\lambda_1 = 658\,\text{nm}$, $\lambda_1 = 660\,\text{nm}$ and $\lambda_1 = 662\,\text{nm}$. Going from top to bottom, the observation angles are $\phi = 120°$, $\phi = 155°$ and $\phi = 170°$.

As with CTS weight functions, we still need to determine the range in λ that should be used for the weight functions used to generate the synthetic spectra. This can be done relatively easily by using eq. 2.30 to determine the largest (λ_{max}) and smallest (λ_{min}) wavelength that can be detected for the chosen region of velocity-space. The smallest value will naturally occur for $s_l = s_1 = -220.2 \times 10^{-18}\,\mathrm{m^2/V}$ and the largest for $s_l = s_{15} = 220.9 \times 10^{-18}\,\mathrm{m^2/V}$. Additionally, it is easy to see from eq. 2.30 that the expressions will depend on ϕ in $(v_\parallel, v_\perp)$-coordinates. For $\phi \in [0; \pi/2]$, λ_{max} is obtained by using $v_{\perp,max}$, $v_{\parallel,max}$ and setting $\cos(\Gamma_l) = 1$, while λ_{min} is obtained by using $v_{\perp,max}$, $v_{\parallel,min}$ and setting $\cos(\Gamma_l) = -1$.

$$\text{for } \phi \in [0; \pi/2] : \lambda_{min} = (\lambda_0 + s_1 v_{\perp,max} B)\left(1 + \frac{v_{\parallel,min}\cos(\phi) - v_{\perp,max}\sin(\phi)}{c}\right) \tag{2.38}$$

$$\text{for } \phi \in [0; \pi/2] : \lambda_{max} = (\lambda_0 + s_{15} v_{\perp,max} B)\left(1 + \frac{v_{\parallel,max}\cos(\phi) + v_{\perp,max}\sin(\phi)}{c}\right) \tag{2.39}$$

For $\phi \in [\pi/2; \pi]$, the expressions are the same except that the positive and negative values of $v_\parallel$ are swapped.

$$\text{for } \phi \in [\pi/2; \pi] : \lambda_{min} = (\lambda_0 + s_1 v_{\perp,max} B)\left(1 + \frac{v_{\parallel,max}\cos(\phi) - v_{\perp,max}\sin(\phi)}{c}\right) \tag{2.40}$$

$$\text{for } \phi \in [\pi/2; \pi] : \lambda_{max} = (\lambda_0 + s_{15} v_{\perp,max} B)\left(1 + \frac{v_{\parallel,min}\cos(\phi) + v_{\perp,max}\sin(\phi)}{c}\right) \tag{2.41}$$

In (E, p)-coordinates, one can simply use the relations in eq. 2.3 to obtain the corresponding $(v_\parallel, v_\perp)$-values and then use eqs. 2.38-2.41.

2.3 NES weight functions

Neutron emission spectrometry involves measuring the energy of neutrons that escape the plasma. Unlike CTS and FIDA, it is thus a so-called passive diagnostic method, as it does not influence the plasma itself. The neutrons are produced by various fusion reactions and their energies are related to the velocities of the reactants through conservation of energy and momentum. They can escape the plasma because they are electrically neutral and therefore unaffected by the magnetic field. The two most commonly considered fusion reactions that produce neutrons are [17] [18] [47] [44]

$$\mathrm{D} + \mathrm{D} \longrightarrow {}^3\mathrm{He} + \mathrm{n} \tag{2.42}$$

$$\mathrm{D} + \mathrm{T} \longrightarrow {}^4\mathrm{He} + \mathrm{n} \tag{2.43}$$

where D is deuterium, T is tritium, n is a neutron, ^{3}He is helium-3 and ^{4}He is helium-4 (an α-particle). The energy produced in the D-D fusion reaction is $Q =$

3.27 MeV, and the energy produced in the D-T fusion reaction is $Q = 17.6\,\mathrm{MeV}$. The energy of the neutrons can be measured using a variety of different instruments. The best energy resolution is obtained by using a time-of-flight spectrometer, which consists of two sets of scintillator detectors placed a known distance from one another. By correlating the detection events, it is then possible to identify paired detection events corresponding to a single neutron being detected at both sets of detectors and thereby determine the time it took the neutron to traverse the known distance. This allows one to calculate the energy of the neutron. Another type of measurement instrument called a compact spectrometer relies on analysis of the shape of the light pulse emitted when the neutron passes a single scintillator in order to determine the neutron energy. All of the different instruments have in common that they can only measure the energy of neutrons that move towards the instrument within its field-of-view, though these neutrons can have been created anywhere in the plasma within the field-of-view. This can be compared to the CTS-diagnostic which can measure the projected velocity of fast ions moving towards or away from its line-of-sight. In most cases, the majority of the neutrons are produced by reactions between fast and thermal ions. [17] [18] [47] [44] [16] We will thus focus on weight functions for these types of reactions.

The normally desired output spectrum from a NES-diagnostic is naturally a distribution of neutron energies, i.e. a neutron energy spectrum. Much like with FIDA-diagnostics, the general form of the weight function for a given interval will thus involve a probability part and an R-part that can be said to be related to the fact that we are not measuring directly on the fast ions. It also still involves dividing by the size of the interval for reasons similar to the previously mentioned ones. The probability part now indicates the probability of measuring neutron energies in the given interval for a given situation, and the rate part R denotes the total number of neutrons sent towards the detector per fast ion per second regardless of the neutron energy. [17] [18] [16]

$$w(E_{n,1}, E_{n,2}, v_\parallel, v_\perp, \phi)) = \frac{1}{E_{n,2} - E_{n,1}} R(v_\parallel, v_\perp, \phi))\mathrm{prob}(E_{n,1} < E_n < E_{n,2} | v_\parallel, v_\perp, \phi)) \tag{2.44}$$

The probability part can be derived in much the same way as the one for CTS, except that the expression for the gyro-angle corresponding to a certain neutron energy is obviously different. [17] [18]

$$\mathrm{prob}(E_{n,1} < E_n < E_{n,2} | v_\parallel, v_\perp, \phi)) = \frac{\Gamma_1 - \Gamma_2}{\pi} \tag{2.45}$$

In order to obtain the new expression for $\Gamma(E_n)$, one starts with energy and momentum conservation for the fusion reaction, then makes the reasonable assumption that the velocity of the thermal ion can be neglected compared to that of the fast ion, before finally combining the two equations, as was done in [18]. This

leads to the following alternate expression for u.

$$u = \frac{1}{2}\frac{m_{He}+m_n}{m_f}v_n - \frac{1}{2}\frac{m_{He}-m_f}{m_n}\frac{v_f^2}{v_n} - \frac{m_{He}}{m_f m_n}\frac{Q}{v_n} \tag{2.46}$$

In this expression, m_{He} is the mass of the helium-ion produced by the considered fusion reaction, m_n is the mass of a neutron, m_f is the mass of the fast ion, v_f is the velocity of the fast ion, and $v_n = \sqrt{\frac{2E_n}{m_n}}$ is the measured velocity of the neutron. Combining eq. 2.46 with eq. 2.16 in order to eliminate u, isolating Γ and inserting $v_n = \sqrt{\frac{2E_n}{m_n}}$ then leads to the following expression for $\Gamma(E_n)$. [17] [18]

$$\Gamma = \arccos\left(\frac{1}{v_\perp \sin(\phi)}\left(\frac{1}{2}\frac{m_{He}+m_n}{m_f}\sqrt{\frac{2E_n}{m_n}} - \frac{1}{2}\frac{m_{He}-m_f}{\sqrt{m_n}}\frac{v_\parallel^2+v_\perp^2}{\sqrt{2E_n}} - \frac{m_{He}}{m_f\sqrt{m_n}}\frac{Q}{\sqrt{2E_n}} - v_\parallel\cos(\phi)\right)\right) \tag{2.47}$$

The corresponding expression in (E,p)-coordinates can be obtained by using the relations in eq. 2.3.

$$\Gamma = \arccos\left(\frac{1}{\sqrt{1-p^2}\sin(\phi)}\left(\frac{1}{2}\frac{m_{He}+m_n}{\sqrt{m_f m_n}}\sqrt{\frac{E_n}{E}} - \frac{1}{2}\frac{m_{He}-m_f}{\sqrt{m_f m_n}}\sqrt{\frac{E}{E_n}} - \frac{1}{2}\frac{m_{He}}{\sqrt{m_f m_n}}\frac{Q}{\sqrt{E_n E}} - p\cos(\phi)\right)\right) \tag{2.48}$$

As always, we take the real part of the gyro-angles in order to obtain physically meaningful values.
The complete expression for the rate part is rather difficult to deal with, as it involves a complicated integral and depends on the solid angle of the detector as seen from the plasma. We can, however, obtain a reasonable approximation by neglecting the finite solid angle and the ϕ-dependence of the cross-section. By assuming that the thermal ion velocity is negligible compared to the fast ion velocity, we also reduce the expression to only being dependent on the fast ion velocity rather than the relative velocity, so long as there is no drift. All in all, these approximations yield the expression [17] [18]

$$R(v_\parallel, v_\perp, \phi)) = n_t \sigma(v_f)\sqrt{v_\parallel^2 + v_\perp^2} \tag{2.49}$$

where n_t is the thermal ion density, and σ is the reaction cross section. In (E,p)-coordinates, the expression is mostly the same.

$$R(E,p,\phi) = n_t \sigma(v_f)\sqrt{\frac{2E}{m_f}} \tag{2.50}$$

The simplest way to include drift is to consider purely toroidal drift of the bulk plasma and model the rotating thermal ion distribution as a delta-function located

at a non-zero parallel velocity with zero perpendicular velocity. This changes the expression for the rate part to [17] [18]

$$R(v_\parallel, v_\perp, \phi)) = n_t \sigma(\sqrt{(v_\parallel - v_d)^2 + v_\perp^2})\sqrt{(v_\parallel - v_d)^2 + v_\perp^2} \tag{2.51}$$

where v_d is the toroidal drift velocity. The reaction cross section can be approximated by the following formula that was derived by Bosch and Hale. [3]

$$\sigma = \frac{S(E_{cm})}{E_{cm} \exp\left(\frac{B_G}{\sqrt{E_{cm}}}\right)} \tag{2.52}$$

$$S(E_{cm}) = \frac{A1 + E_{cm}(A2 + E_{cm}(A3 + E_{cm}(A4 + E_{cm}A5)))}{1 + E_{cm}(B1 + E_{cm}(B2 + E_{cm}(B3 + E_{cm}B4)))} \tag{2.53}$$

E_{cm} is the energy in the center-of-mass frame in units of [keV], and the other parameters are constants that depend on the fusion reaction under consideration. The cross section is given in units of millibarn [mb], $1\,\text{mb} = 10 \times 10^{-31}\,\text{m}^2$. The formula is, however, only valid up to certain values of E_{cm} and will give negative cross sections for sufficiently large E_{cm}-values. In order to avoid the problems this might cause, we set all negative cross sections to zero. Given that we neglect the thermal ion velocities, the energy in the center-of-mass frame is simply half the energy of the fast ion in the case of D-D fusion, 2/5 of the energy of fast deuterium in D-T fusion and 3/5 of the energy of fast tritium in D-T fusion. The values of the various constants are shown in table 2.1 below. [3] We also include the values for p-D fusion that are relevant for GRS. Note that in the case of p-D fusion, E_{cm} should be in units of [MeV]. [29]

Table 2.1: Table of constants for the Bosch-Hale approximation to fusion reaction cross sections for neutron-producing D-D and D-T fusion and gamma-ray-emitting p-D fusion.

	D-D	D-T	p-D
B_G	31.397	34.3827	1.07
A1	5.3701×10^4	6.927×10^4	8.09×10^{-4}
A2	3.3027×10^2	7.454×10^8	1.92×10^{-3}
A3	-1.2706×10^{-1}	2.050×10^6	1.21×10^{-2}
A4	2.9327×10^{-5}	5.2002×10^4	-5.26×10^{-3}
A5	-2.5151×10^{-9}	0	6.52×10^{-4}
B1	0	6.38×10^1	0
B2	0	-9.95×10^{-1}	0
B3	0	6.981×10^{-5}	0
B4	0	1.728×10^{-4}	0

We now have everything we need to calculate NES weight functions. The rate part can be calculated using eqs. 2.49-2.53, the values of Γ_1 and Γ_2 corresponding to $E_{n,1}$ and $E_{n,2}$ are calculated using eq. 2.47 and used in eq. 2.45 to determine the probability part, after which the total weight function is calculated using eq. 2.44.

Some example weight functions for D-D fusion with zero drift, $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ and certain values of $E_{n,1}$, $E_{n,2}$ and ϕ are shown in Fig. 2.5. The circular cutoff at large velocities that seems to affect all of the weight functions that approach this region is caused by the previously mentioned limitations of the Bosch-Hale formula for the reaction cross section. The cross sections would be negative for this region and are instead set to zero. As a result, we do not see any weight functions in this region.

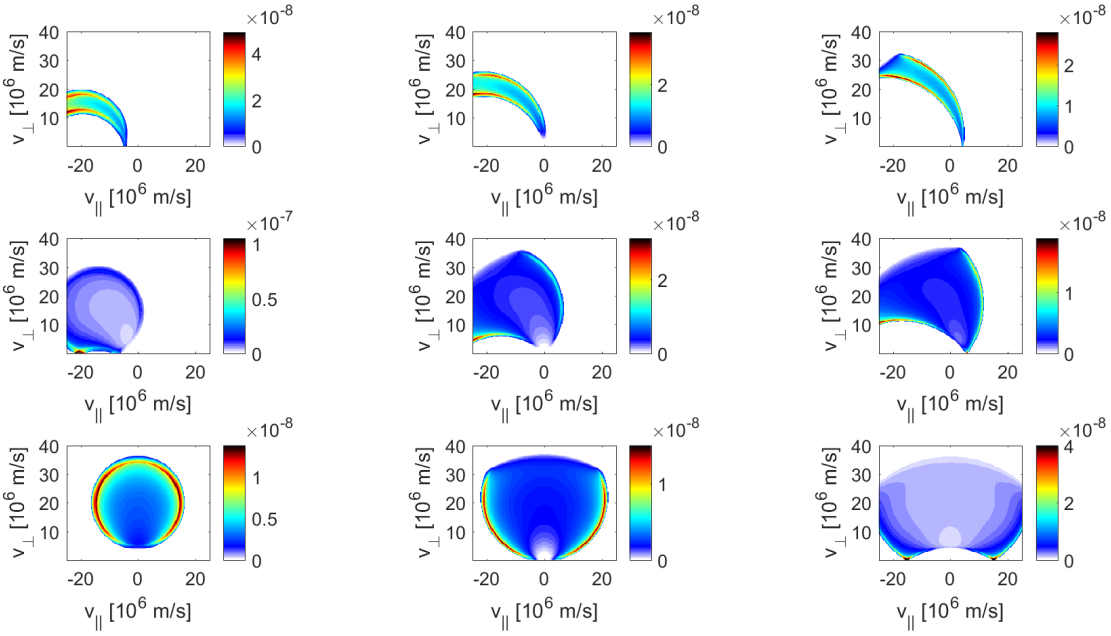


Figure 2.5: Examples of NES weight functions for D-D fusion with zero drift and $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ in $(v_\parallel, v_\perp)$-coordinates. All of the weight functions are plotted for the same interval size $E_{n,2} - E_{n,1} = 0.1\,\mathrm{MeV}$. Going from left to right, the weight functions are for $E_{n,1} = 2\,\mathrm{MeV}$, $E_{n,1} = 2.4\,\mathrm{MeV}$ and $E_{n,1} = 3\,\mathrm{MeV}$. Going from top to bottom, the observation angles are $\phi = 10°$, $\phi = 45°$ and $\phi = 90°$.

Lastly, we again need to determine the neutron energy range to be used for the generation of synthetic spectra. To this end, we determine the boundaries by setting $\cos(\Gamma) = \pm 1$ in eq. 2.16, which corresponds to the line-of-sight velocity being at extremal values, and combining it with eq. 2.46 in order to obtain the following expression. [43]

$$v_\parallel \cos(\phi) \pm v_\perp \sin(\phi) = \frac{1}{2}\frac{m_{He} + m_n}{m_f} v_n - \frac{1}{2}\frac{m_{He} - m_f}{m_n}\frac{v_f^2}{v_n} - \frac{m_{He}}{m_f m_n}\frac{Q}{v_n} \tag{2.54}$$

$$v_\parallel^2 + v_\perp^2 + 2\frac{m_n v_n}{m_{He} - m_f}(v_\parallel \cos(\phi) \pm v_\perp \sin(\phi)) = \frac{m_n}{m_f}\frac{m_{He} + m_n}{m_{He} - m_f} v_n^2 - 2\frac{m_{He} Q}{m_f (m_{He} - m_f)} \tag{2.55}$$

Eq. 2.55 can be rewritten in the form $(v_\parallel - v_{\parallel,0})^2 + (v_\perp - v_{\perp,0})^2 = r_v^2$, which allows us to infer that the weight functions are bounded by circular arcs with $v_\perp > 0$. [43]

$$\left(v_\parallel + \frac{m_n v_n}{m_{He} - m_f}\cos(\phi)\right)^2 + \left(v_\perp \pm \frac{m_n v_n}{m_{He} - m_f}\sin(\phi)\right)^2 = \frac{m_n(m_{He} + m_n)v_n^2 - 2m_{He}Q}{m_f(m_{He} - m_f)} + \frac{m_n^2 v_n^2}{(m_{He} - m_f)^2} \quad (2.56)$$

This gives us

$$v_{\parallel,0} = -\frac{m_n v_n}{m_{He} - m_f}\cos(\phi) \quad (2.57)$$

$$v_{\perp,0} = \pm\frac{m_n v_n}{m_{He} - m_f}\sin(\phi) \quad (2.58)$$

$$r_v = \sqrt{\frac{m_n(m_{He} + m_n)v_n^2 - 2m_{He}Q}{m_f(m_{He} - m_f)} + \frac{m_n^2 v_n^2}{(m_{He} - m_f)^2}} \quad (2.59)$$

for the center and radius of the circular arcs. [43] The right-hand side of eq. 2.56 must naturally be non-negative, as the left-hand side of eq. 2.56 cannot be negative. This implies a minimum energy of the neutrons that can be observed.

$$\frac{m_n(m_{He} + m_n)v_n^2 - 2m_{He}Q}{m_f(m_{He} - m_f)} + \frac{m_n^2 v_n^2}{(m_{He} - m_f)^2} \geq 0 \quad (2.60)$$

We note that $v_n^2 = \frac{2E_n}{m_n}$.

$$(m_{He} + m_n)E_n + \frac{m_n m_f}{m_{He} - m_f}E_n \geq m_{He}Q \quad (2.61)$$

$$E_n \geq E_{n,min} = \frac{(m_{He} - m_f)m_{He}Q}{(m_{He} + m_n)(m_{He} - m_f) + m_n m_f} \quad (2.62)$$

We use this minimum energy as the lower limit of the neutron energy range. Next, we note that the distance from the center of the previously mentioned circular arcs to the origin is $v_0 = \sqrt{v_{\parallel,0}^2 + v_{\perp,0}^2}$.

$$v_0 = \frac{m_n v_n}{m_{He} - m_f} \quad (2.63)$$

The upper circular arc must then have a center with $v_{\perp,0} > 0$, and the lower circular arc must have a center with $v_{\perp,0} < 0$, as indicated by eq. 2.58. As such, the smallest possible fast ion energy that can be covered by the weight function for a given neutron energy must then be given by the following expression. [43]

$$E_{min} = \frac{m_f}{2}(r_v - v_0)^2 \quad (2.64)$$

We should thus be able to ensure coverage of the entire considered region of velocity-space by setting E_{min} equal to the largest fast ion energy that occurs in that region and using the corresponding neutron energy as the upper limit of the neutron energy range. In (E, p)-coordinates, the largest considered fast ion energy is simply the largest E-value to be considered, and in $(v_\parallel, v_\perp)$-coordinates, the largest considered fast ion energy is just $E = \frac{1}{2}m_f(v_{\parallel,max}^2 + v_{\perp,max}^2)$. However, we still need to rewrite eq. 2.64 as an expression for E_n in terms of $E_{min} = E$.

$$E = \frac{m_f}{2}(r_v^2 + v_0^2 - 2r_v v_0) \Leftrightarrow 2r_v v_0 = r_v^2 + v_0^2 - \frac{2E}{m_f} \tag{2.65}$$

Both sides are squared.

$$4r_v^2 v_0^2 = r_v^4 + v_0^4 + 4\frac{E^2}{m_f^2} - 4\frac{E}{m_f}(r_v^2 + v_0^2) + 2r_v^2 v_0^2 \tag{2.66}$$

$$4\frac{E}{m_f}(r_v^2 + v_0^2) = (r_v^2 - v_0^2)^2 + 4\frac{E^2}{m_f^2} \tag{2.67}$$

Based on eqs. 2.59 and 2.63, we see that $r_v^2 - v_0^2 = \frac{m_n(m_{He}+m_n)v_n^2 - 2m_{He}Q}{m_f(m_{He}-m_f)}$.

$$\begin{aligned} 4\frac{E}{m_f}\left(\frac{m_n(m_{He}+m_n)v_n^2 - 2m_{He}Q}{m_f(m_{He}-m_f)} + 2\frac{m_n^2 v_n^2}{(m_{He}-m_f)^2}\right) = \\ \frac{m_n^2(m_{He}+m_n)^2 v_n^4 + 4m_{He}^2Q^2 - 4m_{He}Qm_n(m_{He}+m_n)v_n^2}{m_f^2(m_{He}-m_f)^2} + 4\frac{E^2}{m_f^2} \end{aligned} \tag{2.68}$$

We use $v_n^2 = \frac{2E_n}{m_n}$.

$$\begin{aligned} &4E\left(2(m_{He}+m_n)(m_{He}-m_f)E_n - 2m_{He}(m_{He}-m_f)Q + 4m_n m_f E_n\right) \\ &= 4(m_{He}+m_n)^2E_n^2 + 4m_{He}^2Q^2 - 8m_{He}Q(m_{He}+m_n)E_n + 4(m_{He}-m_f)^2E^2 \end{aligned} \tag{2.69}$$

This is a quadratic equation in E_n. It is solved in the usual way. Furthermore, we only include the result with a plus because the result with a minus was negative in all the cases we tested. The larger result is also the one most likely to actually ensure coverage of the entire considered region of velocity-space. We thus have the following expression for the upper limit of the neutron energy range.

$$E_{n,max} = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \tag{2.70}$$

$$A = 4(m_{He}+m_n)^2 \tag{2.71}$$

$$B = -8\left(m_{He}(m_{He}+m_n)Q + (m_{He}+m_n)(m_{He}-m_f)E + 2m_n m_f E\right) \tag{2.72}$$

$$C = 4m_{He}^2Q^2 + 8m_{He}(m_{He}-m_f)QE + 4(m_{He}-m_f)^2E^2 \tag{2.73}$$

2.4 One-step reaction GRS weight functions

Both types of gamma-ray spectrometry involve measuring the energy of gamma-rays emitted by the plasma. The only difference lies in the processes that lead to the gamma-rays being emitted. Gamma-ray spectrometry is thus also a passive diagnostic method. In the case of one-step reaction GRS, the considered gamma-rays are produced directly by certain fusion reactions as a primary reaction product. Some of these reactions are shown below. [43] [42] [44] [24]

$$\mathrm{D} + \mathrm{p} \longrightarrow {}^{3}\mathrm{He} + \gamma \tag{2.74}$$

$$\mathrm{D} + \mathrm{D} \longrightarrow {}^{4}\mathrm{He} + \gamma \tag{2.75}$$

$$\mathrm{D} + \mathrm{T} \longrightarrow {}^{5}\mathrm{He} + \gamma \tag{2.76}$$

$$\mathrm{T} + \mathrm{p} \longrightarrow {}^{4}\mathrm{He} + \gamma \tag{2.77}$$

However, the reaction cross section is only well-known for the D-p fusion reaction so we will only consider this reaction, though the weight function formula are general. The energy produced by this reaction is $Q = 5.5\,\mathrm{MeV}$. [43] [29] The energy of the emitted gamma-rays can be measured with a number of different detectors. Most of these involve or are based on photo-diodes, but new detectors are still being developed and improved. The aim of these development projects is to simultaneously achieve high-frequency counting capabilities, insensitivity to magnetic field and high energy-resolution in a single detector. [43] [30] [33] As with NES, we will focus on weight functions for gamma-rays produced by reactions between fast and thermal ions.

The normally desired output spectrum from a GRS-diagnostic is a distribution of gamma-ray energies. The process is similar to that for a NES-diagnostic and the weight functions have the same general form. One difference is that the rate part now indicates the total number of gamma-ray photons incident on the detector per fast ion per second regardless of the gamma-ray energy. [43]

$$w(E_{\gamma,1}, E_{\gamma,2}, v_\parallel, v_\perp, \phi)) = \frac{1}{E_{\gamma,2} - E_{\gamma,1}} R(v_\parallel, v_\perp, \phi))\mathrm{prob}(E_{\gamma,1} < E_\gamma < E_{\gamma,2} | v_\parallel, v_\perp, \phi)) \tag{2.78}$$

The probability part can again be derived in much the same way as the one for CTS except for the expression for the gyro-angle corresponding to a certain gamma-ray energy being different. [43]

$$\mathrm{prob}(E_{\gamma,1} < E_\gamma < E_{\gamma,2} | v_\parallel, v_\perp, \phi)) = \frac{\Gamma_1 - \Gamma_2}{\pi} \tag{2.79}$$

The derivation of $\Gamma(E_\gamma)$ is also similar to that for NES. One starts with energy and momentum conservation for the fusion reaction, then makes the reasonable assumption that the velocity of the thermal ion can be neglected compared to that of the fast ion, before finally combining the two equations, as was done in [43].

This leads to the following alternate expression for u. Note that the line-of-sight that we project onto is still the actual line-of-sight of the diagnostic device.

$$u = \frac{(m_f - m_{pr})c}{2E_\gamma}(v_\parallel^2 + v_\perp^2) + \frac{E_\gamma}{2m_f c} + \frac{m_{pr}c(E_\gamma - Q)}{m_f E_\gamma} \tag{2.80}$$

In this expression, m_f is the mass of the fast ion, $c = 3 \times 10^8\,\mathrm{m/s}$ is the speed of light and m_{pr} is the mass of the ion produced by the reaction, e.g. the mass of ^{3}He in the case of the D-p fusion reaction. Next, one combines eq. 2.80 with eq. 2.16 in order to eliminate u. Isolating Γ then yields the following expression for $\Gamma(E_\gamma)$.

$$\Gamma = \arccos\left(\frac{1}{v_\perp \sin(\phi)}\left(\frac{(m_f - m_{pr})c}{2E_\gamma}(v_\parallel^2 + v_\perp^2) + \frac{E_\gamma}{2m_f c} + \frac{m_{pr}c(E_\gamma - Q)}{m_f E_\gamma} - v_\parallel \cos(\phi)\right)\right) \tag{2.81}$$

The corresponding expression in (E, p)-coordinates can be obtained by using the relations in eq. 2.3.

$$\Gamma = \arccos\left(\frac{\sqrt{\frac{m_f}{2E}}}{\sqrt{1-p^2}\sin(\phi)}\left(\frac{(m_f - m_{pr})c}{2E_\gamma}\frac{2E}{m_f} + \frac{E_\gamma}{2m_f c} + \frac{m_{pr}c(E_\gamma - Q)}{m_f E_\gamma} - p\sqrt{\frac{2E}{m_f}}\cos(\phi)\right)\right) \tag{2.82}$$

Alternatively, one can use the relations in eq. 2.2 to calculate the $(v_\parallel, v_\perp$-values corresponding to the (E, p)-coordinates and insert these in eq. 2.81. Regardless, we still take the real part of the gyro-angles in order to ensure that the results are physically meaningful.

The rate part for one-step reaction GRS can also be calculated using eqs. 2.49-2.53. The relevant constants are given in table 2.1. Note, however, that the energy in the center-of-mass frame should be in units of MeV, as was previously mentioned. [43] [29] Given that we neglect the velocity of the thermal ions and only consider D-p fusion, the energy in the center-of-mass frame is simply 1/3 times the energy of a fast proton or 2/3 times the energy of a fast deuterium-ion.

We now have everything we need to calculate one-step reaction GRS weight functions. The rate part can be calculated using eqs. 2.49-2.53, the values of Γ_1 and Γ_2 corresponding to $E_{\gamma,1}$ and $E_{\gamma,2}$ are calculated using eq. 2.81 and used in eq. 2.79 to determine the probability part, after which the total weight function is calculated using eq. 2.78. Some example weight functions for D-p fusion with fast p, zero drift, $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ and certain values of $E_{\gamma,1}$, $E_{\gamma,2}$ and ϕ are shown in Fig. 2.6.

Lastly, we once again need to determine the range in energies to be used for the generation of synthetic spectra. The procedure used to determine this range is the same as the one used for NES and most of it was carried out in [43]. The expressions for v_0 and r_v were shown to be

$$v_0 = \frac{E_\gamma}{(m_{pr} - m_f)c} \tag{2.83}$$

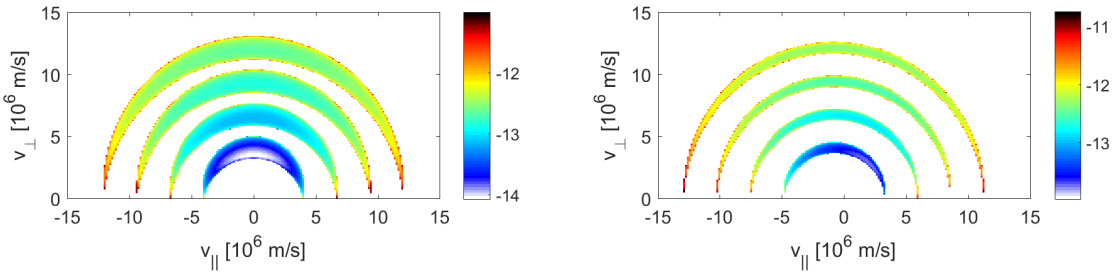


Figure 2.6: Examples of one-step reaction GRS weight functions for D-p fusion with fast p, zero drift and $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ in $v_\parallel, v_\perp$-coordinates in base 10 logarithm. All of the weight functions are plotted for the same interval size $E_{\gamma,2} - E_{\gamma,1} = 1\,\mathrm{keV}$. Going from inside to outside, the weight functions are for $E_{\gamma,1} - Q = 50\,\mathrm{keV}$, $E_{\gamma,1} - Q = 150\,\mathrm{keV}$, $E_{\gamma,1} - Q = 300\,\mathrm{keV}$ and $E_{\gamma,1} - Q = 500\,\mathrm{keV}$. The observation angle is left) $\phi = 90°$, right) $\phi = 30°$.

$$r_v = \sqrt{\frac{m_{pr}}{m_f}\left(\frac{E_\gamma^2}{(m_{pr} - m_f)^2 c^2} + \frac{2(E_\gamma - Q)}{m_{pr} - m_f}\right)} \tag{2.84}$$

and the minimum energy of the observable gamma-rays was shown to be

$$E_\gamma \geq E_{\gamma,min} = \sqrt{2(m_{pr} - m_f)c^2 Q + (m_{pr} - m_f)^2 c^4} - (m_{pr} - m_f)c^2 \tag{2.85}$$

We can use this minimum energy as the lower limit of the gamma-ray energy range. We should also be able to obtain a reasonable upper limit value in the same way as we did for NES, but unfortunately it is not possible to rewrite eq. 2.64 for one-step reaction GRS as an analytic expression for E_γ in terms of E_{min}. Instead, we have to use MATLAB to solve eq. 2.64 for the value of E_γ corresponding to E_{min} being equal to the largest fast ion energy that occurs in the considered region of velocity-space. This value of E_γ is then used as the upper limit of the gamma-ray energy range. In (E, p)-coordinates, one can simply use the relations in eq. 2.3 to obtain the corresponding $(v_\parallel, v_\perp)$-values and then use the same procedure as for $(v_\parallel, v_\perp)$-coordinates.

2.5 Two-step reaction GRS weight functions

As previously mentioned, two-step reaction GRS also involves measuring the energy of gamma-rays emitted by the plasma, and the diagnostics used to do this are the same as the ones used for one-step reaction GRS. The processes that lead to gamma-rays being emitted, however, are different. In two-step reaction GRS, one considers gamma-rays produced as secondary reaction products by two-step reactions, hence the name. One such reaction that has been studied at JET and proposed for use at ITER is the fusion reaction between fast α-particles and thermal ^{9}Be-particles. This is due to the fact that beryllium is the planned first-wall material at ITER. The reaction produces excited $^{12}\mathrm{C}^*$ that rapidly decays to

ground state ^{12}C and emits a gamma-ray photon. [42] [24] [25] The full reaction is [42]

$$\alpha + {}^{9}\mathrm{Be} \longrightarrow {}^{12}\mathrm{C}^{*} + \mathrm{n} \tag{2.86}$$

$$^{12}\mathrm{C}^{*} \longrightarrow {}^{12}\mathrm{C} + \gamma \tag{2.87}$$

The energy produced in this reaction is $Q = 5.6\,\mathrm{MeV}$, but only $Q^{*} = 1.26\,\mathrm{MeV}$ of this energy adds to the kinetic energy of the reaction products. The remaining $E_{\gamma 0} = 4.44\,\mathrm{MeV}$ is needed to populate the excited state of $^{12}\mathrm{C}^{*}$. $E_{\gamma 0}$ is also the rest frame energy of the emitted gamma-ray photon. [42] [25]
The general form of the weight functions for two-step reaction GRS is, perhaps unsurprisingly, the same as the one for one-step reaction GRS. The units of the R-part are also the same.

$$w(E_{\gamma,1}, E_{\gamma,2}, v_{\parallel}, v_{\perp}, \phi)) = \frac{1}{E_{\gamma,2} - E_{\gamma,1}} R(v_{\parallel}, v_{\perp}) \mathrm{prob}(E_{\gamma,1} < E_{\gamma} < E_{\gamma,2} | v_{\parallel}, v_{\perp}, \phi)) \tag{2.88}$$

The rate-part can still be determined using eq. 2.51, but we cannot use the Bosch-Hale formula to obtain the cross sections for the Be-α fusion reaction. Instead, we make use of the fact that others have determined the cross sections experimentally, and they are illustrated in a plot in [42]. The experimentally determined cross sections were found to only depend on the α-particle energy and can thus be represented by a 1D function. A code that uses this 1D function to return the cross sections in [mb] as a function of $(v_{\parallel}, v_{\perp})$ for α-particle energies up to 6 MeV, which corresponds to $v_{\alpha} = 17 \times 10^{6}\,\mathrm{m/s}$, was supplied. The code can also easily be used to obtain the cross sections as a function of (E, p), seeing as the values only depend on the α-particle energy. We use this code to obtain the cross sections for the Be-α fusion reaction. This does limit us to only considering α-particle energies below 6 MeV when working with two-step reaction GRS, but given that the cross sections are known to quickly approach zero for energies above 6 MeV, it does not matter much. The cross sections also rapidly approach zero for energies below 1.9 MeV ($v_{\alpha} = 9 \times 10^{6}\,\mathrm{m/s}$). [42]

The expression for the probability part is made more complex by the two-step nature of the considered reaction. The gamma-ray photon is Doppler-shifted by the projected velocity u_C of the emitting $^{12}\mathrm{C}^{*}$-atom, but we wish to determine the velocity-coordinates of the fast α-particle that underwent the fusion reaction that produced the emitting $^{12}\mathrm{C}^{*}$-atom. As a result, the relation between Γ and E_{γ} must logically involve both the velocity v_{α} of the α-particle and the velocity v_C of the $^{12}\mathrm{C}^{*}$-atom as well as their relative directions. After all, this relation is obtained from momentum and energy conservation. As always, we neglect the velocity of the thermal ^{9}Be. One way to deal with this more complicated system is to split both velocities into a component, u, parallel to the LOS and a component, $\sqrt{v^2 - u^2}$, perpendicular to the LOS and introduce two new angles to keep track of the directions. These angles are $\beta \in [0, \pi]$, the angle between v_C and the LOS, and $\zeta \in [0, 2\pi]$, the angle between the two components perpendicular to

the LOS. Based on this viewpoint, one can then combine momentum and energy conservation, use $v_C = \frac{u_C}{\cos(\beta)}$ for $\beta \neq \pi/2$ and set $m_\alpha = 4m_n$ and $m_C = 12m_n$ in order to obtain the following expression. [42]

$$\frac{Q^*}{6m_n} - v_\alpha^2 - 13\frac{u_C^2}{\cos^2(\beta)} + 8\left(u_\alpha u_C + \cos(\zeta)\sqrt{(v_\alpha^2 - u_\alpha^2)u_C^2\tan^2(\beta)}\right) = 0 \quad (2.89)$$

$v_\alpha = \sqrt{v_\parallel^2 + v_\perp^2}$ is simply the total velocity of the α-particle, and u_α can be found from eq. 2.16. u_C is related to the Doppler-shifted energy E_γ of the detected gamma-ray photon by [42]

$$E_\gamma = E_{\gamma 0}\left(1 + \frac{u_C}{c}\right) \Leftrightarrow u_C = c\left(\frac{E_\gamma}{E_{\gamma 0}} - 1\right) \quad (2.90)$$

Next, we can isolate the square root in eq. 2.89, square and solve the resulting quadratic equation for u_α in order to obtain the expression in eq. 2.91. Squaring the square root in eq. 2.89 does, however, introduce a spurious solution, i.e. a solution of the resulting equation that is not a solution of eq. 2.89, that we need to check for and exclude when using eq. 2.91. In order to allow for the presence of noise, the check is not perfectly strict, and we only exclude solutions where the left side of eq. 2.89 differs from zero by more than $9\,\mathrm{m}^2/\mathrm{s}^2$.

$$u_\alpha = \frac{13u_C^2 + \cos^2(\beta)\left(v_\alpha^2 - \frac{Q^*}{6m_n}\right)}{8u_C\left(\cos^2(\beta) + \sin^2(\beta)\cos^2(\zeta)\right)} \pm \sqrt{\frac{\sin^2(\beta)\cos^2(\zeta)\left(64u_C^2 v_\alpha^2\cos^2(\beta)\left(\cos^2(\beta) + \sin^2(\beta)\cos^2(\zeta)\right) - \left(\cos^2(\beta)\left(\frac{Q^*}{6m_n} - v_\alpha^2\right) - 13u_C^2\right)^2\right)}{64u_C^2\cos^2(\beta)\left(\cos^2(\beta) + \sin^2(\beta)\cos^2(\zeta)\right)^2}} \quad (2.91)$$

Note that we get two values of u_α for every value of u_C. We also need to check for and exclude values where the term in the square root is negative, given that complex values of u_α are not physically meaningful. Having determined u_α, the corresponding value of Γ can then be found using eq. 2.16. As always, there exists a solution $\Gamma' = 2\pi - \Gamma$ for every solution Γ.
The probability part can still be expressed as an integral over a pdf. Due to the introduction of β and ζ and the doubled u_α-values, however, the transformation to Γ-limits yields a more complicated expression than the one for one-step reaction GRS. It has the following form. [42]

$$\mathrm{prob}(E_{\gamma,1} < E_\gamma < E_{\gamma,2}|v_\parallel, v_\perp, \phi)) = \int_{\beta,\zeta}\mathrm{pdf}(\beta,\zeta|v_\parallel, v_\perp, \phi)\sum_{i=1}^{2}\left|\frac{\Gamma_{2,i} - \Gamma_{1,i}}{\pi}\right|\mathrm{d}\beta\mathrm{d}\zeta \quad (2.92)$$

The sum is over the two unrelated intervals in Γ that we obtain for a single interval in E_γ due to the double values of u_α. In other words, the $i = 1$-values

are obtained by using the plus in eq. 2.91 and the $i = 2$-values are obtained by using the minus. It is generally reasonable to assume that the angles β and ζ are uniformly distributed, meaning that $\text{pdf}(\beta, \zeta | v_\parallel, v_\perp, \phi) = \frac{1}{2\pi^2}$. [42]
We now have everything we need to calculate two-step reaction GRS weight functions. Some example weight functions for Be-α fusion with zero drift, $n_t = n_{Be} = 10^{18}$ and certain values of $E_{\gamma,1}$, $E_{\gamma,2}$ and ϕ are shown in Fig. 2.7. The positions of the considered intervals are indicated using $\Delta E_\gamma = E_{\gamma 1} - E_{\gamma 0}$.

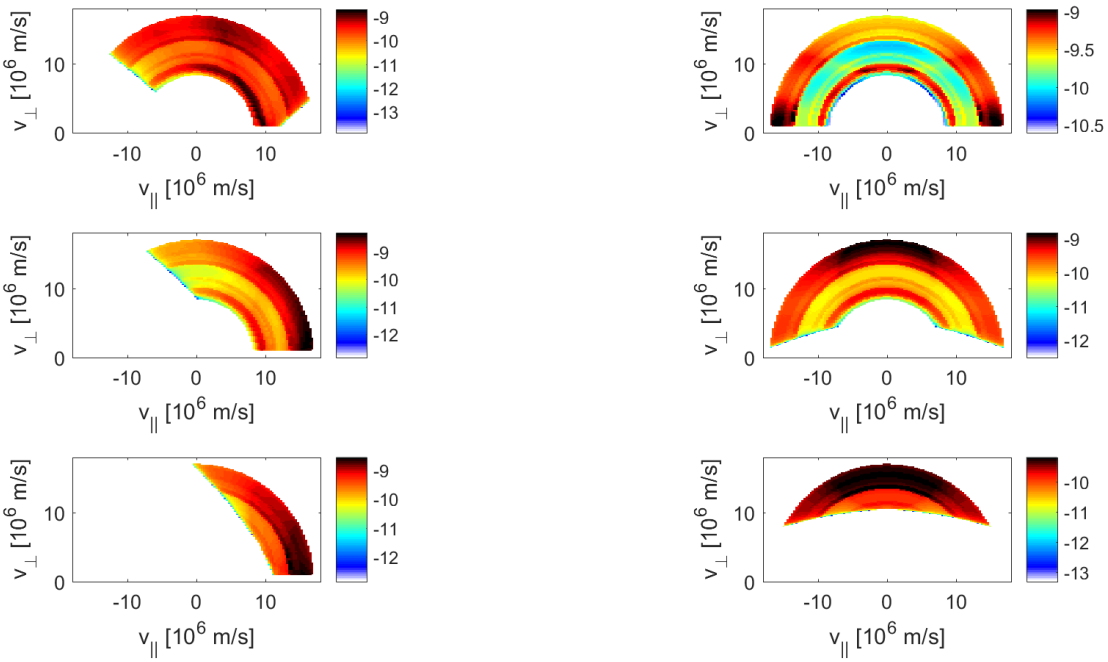


Figure 2.7: Examples of two-step reaction GRS weight functions for Be-α fusion with zero drift and $n_{Be} = 1 \times 10^{18}\,\text{m}^{-3}$ in $v_\parallel, v_\perp$-coordinates in base 10 logarithm. All of the weight functions are plotted for the same interval size $E_{\gamma,2} - E_{\gamma,1} = 1.5\,\text{keV}$. Going from top to bottom, the weight functions are for $\Delta E_\gamma = 15\,\text{keV}$, $\Delta E_\gamma = 45\,\text{keV}$ and $\Delta E_\gamma = 75\,\text{keV}$. Going from left to right, the observation angles are $\phi = 30°$ and $\phi = 90°$.

We note that none of the weight functions cover the low-velocity region below $\sim 9 \times 10^6\,\text{m/s}$ or the high-velocity region above $\sim 17 \times 10^6\,\text{m/s}$. This fits with what we expected based on our knowledge of the fusion cross sections. Two-step reaction GRS is thus only suitable for use in the relatively narrow velocity-space region between $v_\alpha = 9 \times 10^6\,\text{m/s}$ and $v_\alpha = 17 \times 10^6\,\text{m/s}$.
Lastly, we once again need to determine the energy-range to be used for the generation of synthetic spectra. However, we have already concluded that the two-step reaction GRS weight functions only occupy the velocity-space region between $v_\alpha = 9 \times 10^6\,\text{m/s}$ and $v_\alpha = 17 \times 10^6\,\text{m/s}$. Furthermore, we can see based on our plots in Fig. 2.7 and the numerically calculated plots in [42] that weight functions for larger absolute values of ΔE_γ generally cover smaller regions of velocity-space. We also note that increasing ΔE_γ moves the occupied region clockwise and de-

creasing ΔE_γ moves the occupied region counterclockwise, except for the case where $\phi = 90°$. For instance, the weight function for $\Delta E_\gamma = 45\,\text{keV}$ and $\phi = 30°$ in Fig. 2.7 covers a part of the lower-right corner that is not covered by the one for $\Delta E_\gamma = 15\,\text{keV}$ and $\phi = 30°$, despite the former covering an overall smaller region. So long as the right-most and left-most parts of the occupiable region are covered by the weight functions for the outer limits of the energy-range, we can thus be sure that the remainder of the occupiable region will be covered by the weight functions for the values in between. Based on [42] and some checks made using our code, we have found that using $\Delta E_\gamma = \pm 60\,\text{keV}$ ensures coverage of the right- and left-most parts of the occupiable region for any ϕ. The lower limit of the gamma-ray energy range is thus $\Delta E_\gamma = -60\,\text{keV}$, and the upper limit is $\Delta E_\gamma = 60\,\text{keV}$.

Note that all of the considered weight functions involve division by $v_\perp$ in $(v_\parallel, v_\perp)$-coordinates and division by $\sqrt{1-p^2}$ in (E, p)-coordinates. This means that the considered $v_\perp$-values cannot be allowed to go to zero, and the considered p-values cannot be allowed to go to 1 or -1, as this would lead to division by zero. Instead, one uses a minimum $v_\perp$-value that is slightly larger than zero and $p \in]-1; 1[$.
The normalization of the weight functions by the uncertainty of the associated diagnostic also effectively normalizes the synthetic spectra so that spectra from different diagnostics can be compared. This also results in the normalized synthetic spectra becoming unitless, which allows us to make meaningful plots of spectra from multiple different diagnostics.

2.6 Model distribution functions

Finally, we will briefly describe the known model distribution functions that will be used to test our analytic models. The first of these is the so-called bi-Maxwellian distribution function with parallel drift and no perpendicular drift. This function is a good model for the velocity distribution in the center of high-power, high-torque plasmas such as those seen in ASDEX Upgrade. [35] [27] Analytic expressions for the 2D bi-Maxwellian are given in [35]. In $(v_\parallel, v_\perp)$-coordinates, the bi-Maxwellian is given by

$$f^{2D}(v_\parallel, v_\perp) = n_i \frac{m^{3/2}}{\sqrt{2\pi}} \frac{v_\perp}{T_\perp \sqrt{T_\parallel}} \exp\left(-\frac{m(v_\parallel - v_{d\parallel})^2}{2T_\parallel} - \frac{m v_\perp^2}{2T_\perp} \right) \tag{2.93}$$

and in (E, p)-coordinates, it is given by

$$f^{2D}(E, p) = n_i \sqrt{\frac{E}{\pi T_\perp^2 T_\parallel}} \exp\left(-\frac{p^2 E + \frac{1}{2} m v_{d\parallel}^2 - v_{d\parallel} p \sqrt{2mE}}{T_\parallel} - \frac{(1-p^2)E}{T_\perp} \right) \tag{2.94}$$

In both cases, n_i is the ion density, m is the mass of the fast ion, $v_{d\parallel}$ is the parallel drift velocity, $T_\parallel$ is the parallel plasma temperature, and $T_\perp$ is the perpendicular plasma temperature. If $T_\parallel = T_\perp$, the bi-Maxwellian reduces to an isotropic

Maxwellian distribution. [35]
The other model distribution function that we consider is the so-called slowing-down distribution, which is a good model for the velocity distribution that results from using an NBI. [27] It is modeled by assuming that the fast ions are significantly faster than the thermal ions but still slower than the electrons, and is calculated as a sum of so-called Legendre polynomials using a supplied code that is based on the work by Gaffey in [10]. The code can be used for injection energies of 60 keV, which corresponds to the situation that our ad-hoc FIDA R-model was based on, or 1 MeV. An example of the model distribution function obtained for an injection energy of 60 keV at $p = 0.5$ with $E_{crit} = 30$ keV is shown in Fig. 2.8 in (E, p)-coordinates.

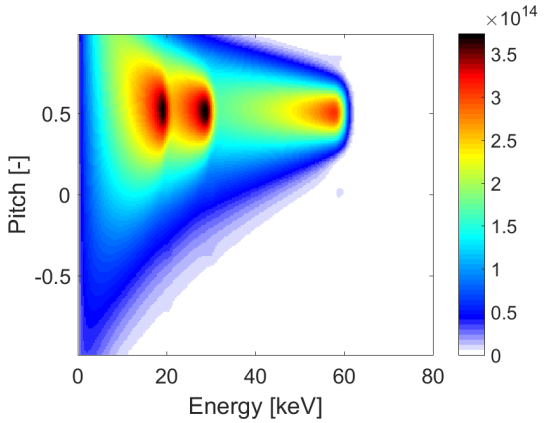


Figure 2.8: Example of the slowing-down distribution function in (E, p)-coordinates for an injection energy of 60 keV at $p = 0.5$ with $E_{crit} = 30$ keV. This example was obtained using the supplied code.

The model distribution can also be a sum of these two distributions.

3 The model code

All of our code was written in MATLAB R2017a so it can be run using this or any newer version of MATLAB. The model consists of a MATLAB function that uses the equations derived in sections 2.1-2.5, eqs. 2.10-2.12 and supplied measurement spectra to return a series of reconstructed velocity distribution functions obtained for the supplied values of the regularization parameter α on a specified grid. The supplied spectra must be normalized by the uncertainty of the associated diagnostics. The function also returns the weight function matrix that was used to reconstruct the velocity distribution functions. The measurement spectra can be from any combination of the five considered diagnostics, and the model can perform 0'th- or 1'st-order Tikhonov regularization or a mix of these. It can also be used to solve the problem with non-negativity constraint such that the reconstructed velocity distribution functions are never negative. The model code

can be seen in appendix B.
Additionally, we have also written another MATlAB function that makes use of the equations derived in sections 2.1-2.6, eq. 2.7 and the model function in order to generate synthetic spectra from a chosen velocity distribution function on a specified forward grid and then plot velocity distribution functions reconstructed from the synthetic spectra on a specified tomography grid. The two grids will always cover the same chosen region of velocity-space, but may have differing numbers of grid points. The synthetic spectra with and without added noise are also plotted in a single plot together with the spectra generated from the reconstructed velocity distribution functions. The chosen velocity distribution is plotted on both the forward grid and the tomography grid in order to allow comparison with the reconstructed distribution functions. Lastly, the function uses eq. 2.13 to calculate the quality factor, Q, for every obtained reconstruction and plot the Q-values as a function of the used α-values. The function can be set to assume any combination of the five considered diagnostics when generating the synthetic spectra. The uncertainty of each assumed diagnostic is calculated as 10 % of the maximum spectrum value obtained by using the non-normalized weight functions for that diagnostic in the forward problem. The added noise is calculated as normally distributed random numbers times the maximum value of the synthetic spectra times a chosen noise level. The code for this other function is shown in appendix C.
Note that although both codes accept any combination of the five considered diagnostics, not all combinations make sense. For instance, the two-step GRS weight functions are based on the fast ions being α-particles whereas the FIDA weight functions are based on the fast ions being deuterium-ions. It would thus not make sense to combine a two-step reaction GRS-diagnostic and a FIDA-diagnostic, as these diagnostics relate to different fast ions. The only diagnostic that can be meaningfully combined with all of the others is CTS given that the CTS weight functions alone are not based on the fast ions being specific types.

4 Results and discussion

To begin with, we will carry out the "inverse crime" of using the same forward and tomography grid with zero added noise as a fundamental check of our code. After all, if one cannot obtain good reconstructions under inverse crime conditions, it certainly will not be possible to obtain good reconstructions under more realistic conditions. We thus perform first-order Tikhonov reconstructions of appropriate model distribution functions for each of the five considered diagnostics where both grids are 40×40 using nine logarithmically spaced values of α between 10^8 and 10^{13}. For the sake of expediency and consistency with later plots, we do not use the non-negativity constraint. We use two views for two-step reaction GRS and tree views for the other diagnostics with 100 points per view. We use the same bi-Maxwellian model distribution (Fig. 4.1a) with $T_{\parallel} = 200\,\mathrm{keV}$, $T_{\perp} = 1.5\,\mathrm{MeV}$,

$n_i = 1 \times 10^{19}\,\mathrm{m}^{-3}$ and zero drift for CTS, NES and one-step reaction GRS and a slowing-down model distribution (Fig. 4.1b) with an injection energy of 60 keV at $p = 0.5$ for FIDA. The bi-Maxwellian is considered in the region where $v_\perp \leq 30 \times 10^6\,\mathrm{m/s}$ and $v_\parallel \in [-10; 10] \cdot 1 \times 10^6\,\mathrm{m/s}$. In the case of two-step reaction GRS, we use a bi-Maxwellian model distribution (Fig. 4.1c) with $T_\parallel = T_\perp = 2\,\mathrm{MeV}$ and $n_i = 1 \times 10^{19}\,\mathrm{m}^{-3}$ and limit ourselves to considering the region between 1.9 MeV and 6 MeV where the weight functions are non-zero. The best reconstruction from the set for each diagnostic, as judged by the Q-value, is shown in Fig. 4.2, and the full sets of reconstructions are shown in appendix A. The best reconstruction was observed to be the one for $\alpha = 10^8$ in all five cases.

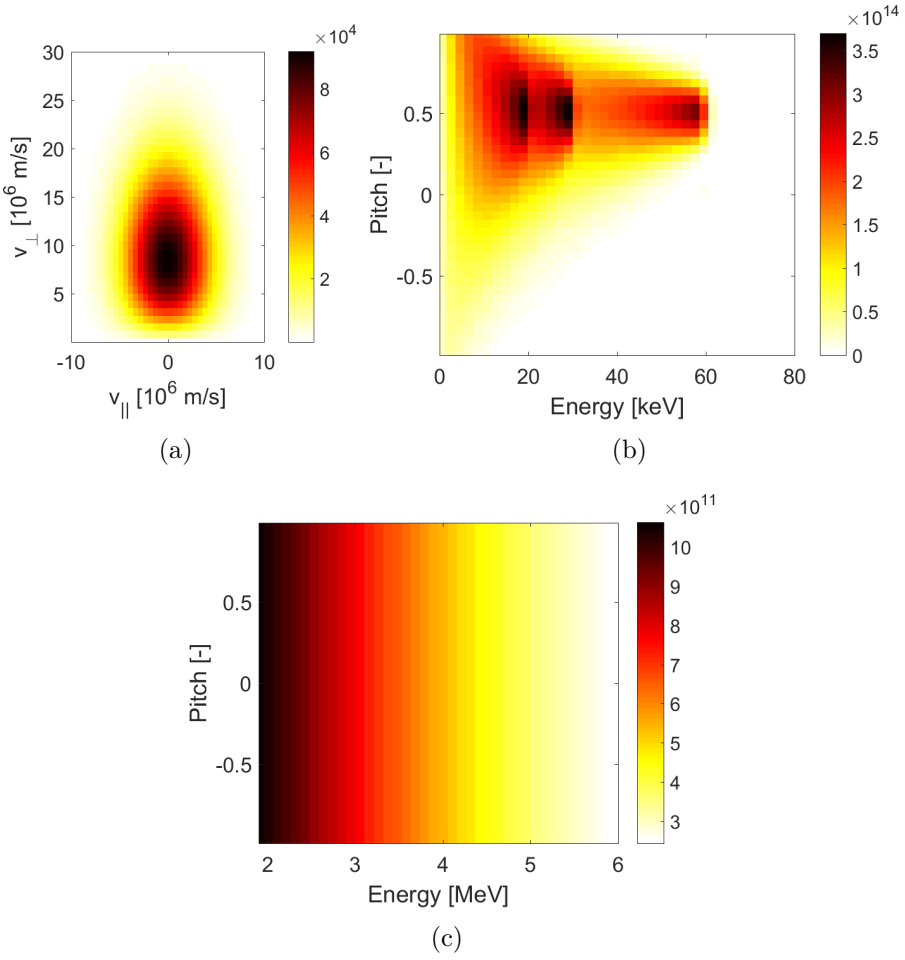


Figure 4.1: Three model distribution functions on 40×40 grids. (a) bi-Maxwellian model with $T_\parallel = 200\,\mathrm{keV}$, $T_\perp = 1.5\,\mathrm{MeV}$, $n_i = 1 \times 10^{19}\,\mathrm{m}^{-3}$ and zero drift. (b) slowing-down model with an injection energy of 60 keV at $p = 0.5$ and $E_{crit} = 30\,\mathrm{keV}$. (c) bi-Maxwellian model with $T_\parallel = T_\perp = 2\,\mathrm{MeV}$, $n_i = 1 \times 10^{19}\,\mathrm{m}^{-3}$ and zero drift.

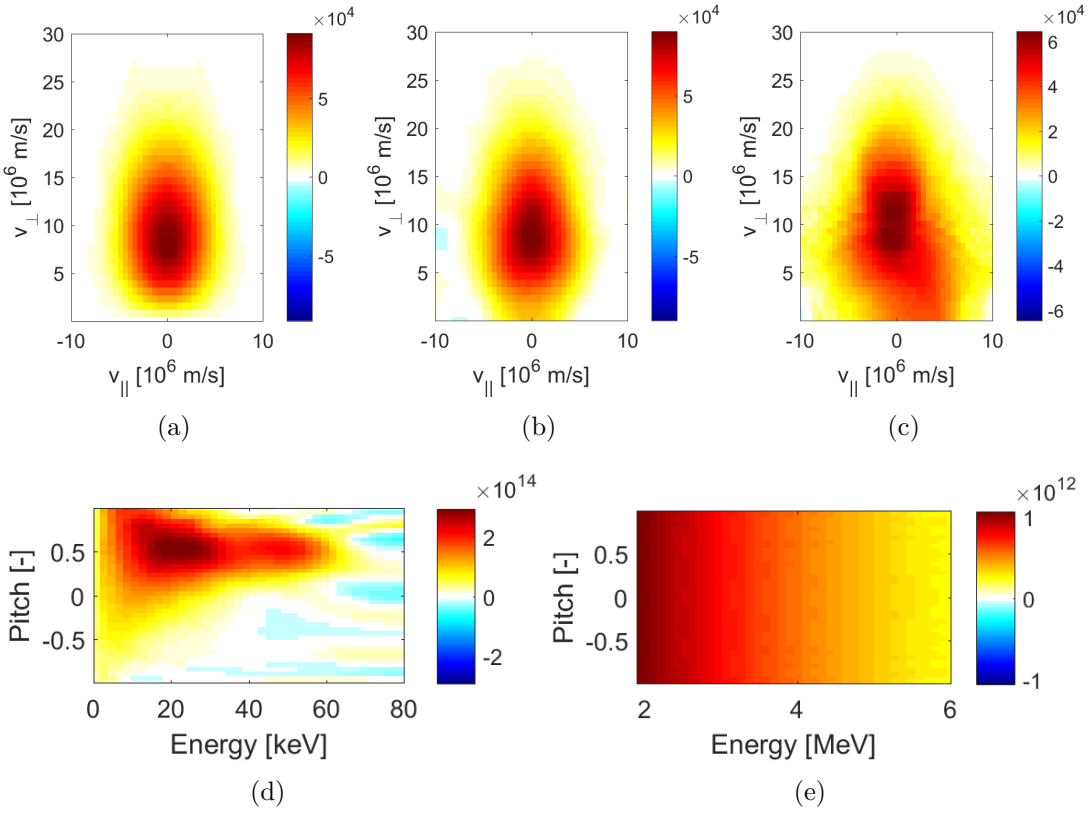


Figure 4.2: First-order Tikhonov reconstructions of the distribution function in (a,b,c) Fig. 4.1a, (d) Fig. 4.1b and (e) Fig. 4.1c made using 100 points per view under inverse crime conditions with zero drift. The used diagnostics were (a) three CTS-views at $\phi = 10°$, $40°$ and $70°$ for fast D-ions, (b) three NES-views at $\phi = 20°$, $45°$ and $85°$ for D-D fusion, (c) three one-step reaction GRS-views at $\phi = 15°$, $45°$ and $80°$ for p-D fusion with fast p, (d) three FIDA-views at $\phi = 10°$, $40°$ and $70°$ for fast D with $B = 1.74\,\mathrm{T}$ and $\bar{\Gamma} = 0$ and (e) two two-step reaction GRS-views at $\phi = 30°$ and $90°$ for Be-α fusion with fast α-particles and $n_{Be} = 1 \times 10^{18}\,\mathrm{m}^{-3}$. $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ for (b) and (c). All five reconstructions were made for $\alpha = 10^8$.

Based on Fig. 4.2, we see that the reconstructions for CTS, NES, FIDA and two-step reaction GRS all resemble their related model distribution functions quite well, though the FIDA-reconstruction contains several spurious features in the region that is empty in Fig. 4.1b. However, most of these spurious features are negative and can thus be avoided by using the non-negativity constraint. The reconstruction for one-step reaction GRS is noticeably worse than the others, but it still resembles the related model distribution function fairly well. We can thus conclude that we were able to obtain at least one fairly good reconstruction for each of the five considered diagnostics under inverse crime conditions. It is also worth noting that all of the reconstructions were underdetermined, given that the grids contain $40 \cdot 40 = 1600$ points and we reconstructed them from spectra containing 300 points (200 in the case of two-step reaction GRS). If we had used more views or more points per view, we would thus almost certainly have obtained even

better reconstructions. In addition to this general conclusion, we made a number of observations regarding the individual diagnostics based on Fig. 4.2 and the full sets of reconstructions in appendix A.

The CTS-reconstructions (Fig. A.1) were seen to resemble the model distribution function (Fig. 4.1a) very well for all of the used values of α. The resemblance is slightly greater for the small values of α than for the largest values of α, but the difference is small.

The NES-reconstructions (Fig. A.2) also resemble the model distribution (Fig. 4.1a) quite well for all of the used values of α, though the resemblance is more clearly seen to diminish with increasing values of α than for CTS. The reconstructions also seem to have been tilted slightly compared to the model distribution, which can most easily be seen in the reconstruction for $\alpha = 10^{13}$ in Fig. A.2. The biggest difference compared to the CTS-reconstructions, however, is the presence of large distribution function values at low velocities that are not present in the model distribution. This spurious feature is seen to grow with increasing α-values and is almost certainly related to differences in the weight functions for CTS and NES. This explanation is supported by the plot (Fig. 4.3) of the sums of the weight functions used for the CTS- and NES-reconstructions, respectively.

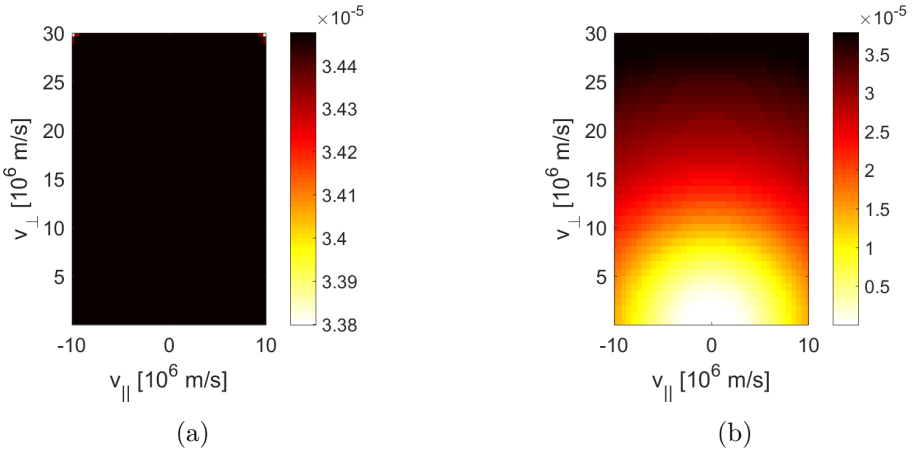


Figure 4.3: (a) Sum of the CTS weight functions used for the reconstructions in Fig. A.1. (b) Sum of the NES weight functions used for the reconstructions in Fig. A.2.

Fig. 4.3 clearly shows that the CTS weight functions cover the entire considered region of velocity-space essentially equally, whereas the NES weight functions have very low values/coverage for the low velocities. Furthermore, the position of the region of low weight function coverage overlaps quite nicely with the position of the spurious feature in the NES-reconstructions. We thus conclude that the spurious feature is most likely caused by the low weight function coverage. The low weight function coverage at low velocities could be related to the fact that there is a minimum energy of the detectable neutrons, as derived in section 2.3. After all,

fast ions with greater velocities will generally result in more energetic neutrons due to conservation of energy. It thus seems reasonable that the fast ions with the lowest velocities will only yield few if any neutrons with energies in the detectable range. This would then be reflected in the weight functions as low values for low velocities. An alternate but related explanation is that the reaction cross sections and thus the rate are very small for the lowest velocities. The cross sections are, after all, known to increase with increasing energy in the center-of-mass frame up to a point [17] [18] [3]. This would also explain the general increase in weight function values with increasing fast ion velocity that we observe in Fig. 4.3b. Both of these explanations are supported by the fact that the CTS weight functions, which involve neither rate functions nor minimum observable projected velocities, cover the entire considered region of velocity-space.

The one-step reaction GRS-reconstructions (Fig. A.3) resemble the model distribution (Fig. 4.1a) fairly well for the small values of α, but the resemblance is seen to rapidly diminish with increasing α. The reconstructions for the largest used values of α only somewhat resemble the model distribution. The bottom part of the reconstructions also seems to have been tilted compared to the model distribution. Finally, a spurious feature similar to the one observed for NES is clearly visible in the one-step reaction GRS-reconstructions, though it also seems to have been tilted. It is observed to grow more rapidly with increasing α-values than the one for NES. This spurious feature is most likely also caused by low weight function coverage at low velocities, which is supported by the plot (Fig. 4.4a) of the sum of the weight functions used for the one-step reaction GRS-reconstructions.

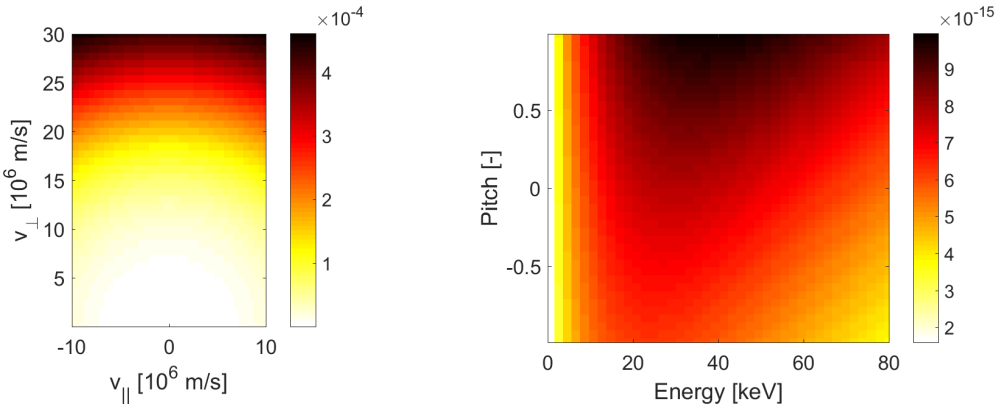


Figure 4.4: (a) Sum of the one-step reaction GRS weight functions used for the reconstructions in Fig. A.3. (b) Sum of the FIDA weight functions used for the reconstructions in Fig. A.4.

In fact, the region with very low weight function coverage is larger than for the NES weight functions, which might explain the more rapid growth of the spurious feature. The most likely causes of this low weight function coverage are the same

as the ones mentioned for NES.

The FIDA-reconstructions (Fig. A.4) resemble the model distribution (Fig. 4.1b) quite well for most of the used values of α. Once again, the resemblance is seen to diminish for increasing values of α, and the two reconstructions for the two largest used values of α only somewhat resemble the model distribution. Most of the reconstructions contain spurious features in the region that is empty in Fig. 4.1b that change as the value of α is increased. These spurious features cannot readily be matched to features in the plot (Fig. 4.4b) of the sum of the weight functions used for the FIDA-reconstructions. As previously mentioned, however, most of these spurious features are negative and can thus be avoided by using the non-negativity constraint. This is demonstrated in Fig. 4.5, which is a reconstruction performed identically to the one in Fig. 4.2d, except that we also used the non-negativity constraint.

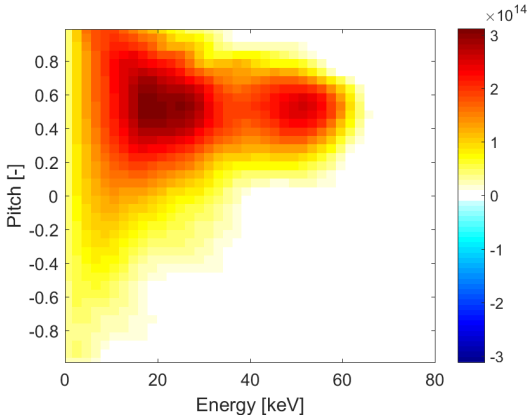


Figure 4.5: First-order Tikhonov reconstruction of the distribution function in Fig. 4.1b made using three FIDA-views at $\phi = 10°$, $40°$ and $70°$ with 100 points per view for fast D-ions with $B = 1.74\,\mathrm{T}$ and $\bar{\Gamma} = 0$ under inverse crime conditions with non-negativity constraint and $\alpha = 10^8$.

By comparing Figs. 4.1b, 4.2d and 4.5, we see that using the non-negativity constraint has not only removed both the negative and positive spurious features, but also improved the resemblance to the model distribution. We thus conclude that FIDA-reconstructions can be significantly improved by using the non-negativity constraint. If the constraint is not used, the reconstructions may contain several spurious features in the region not covered by the actual velocity distribution function.

The two-step reaction GRS-reconstructions (Fig. A.5) resemble the model distribution (Fig. 4.1c) quite well for all of the used values of α. The resemblance is still seen to diminish for increasing values of α. We do not see any noteworthy spurious features, but if we had included the region below 1.9 MeV where the weight functions are essentially zero, we would almost certainly have seen a spu-

rious feature similar to the ones observed for NES and one-step reaction GRS.

We have now evaluated the full sets of reconstructions obtained for each of the five considered diagnostics under inverse crime conditions. We note that the quality of the reconstructions was seen to always decrease with increasing values of α, particularly for one-step reaction GRS. This is most likely caused by the used α-values being larger than necessary for inverse crime conditions. Experience has shown that less regularization is needed when performing reconstructions under inverse crime conditions as compared to more realistic conditions.

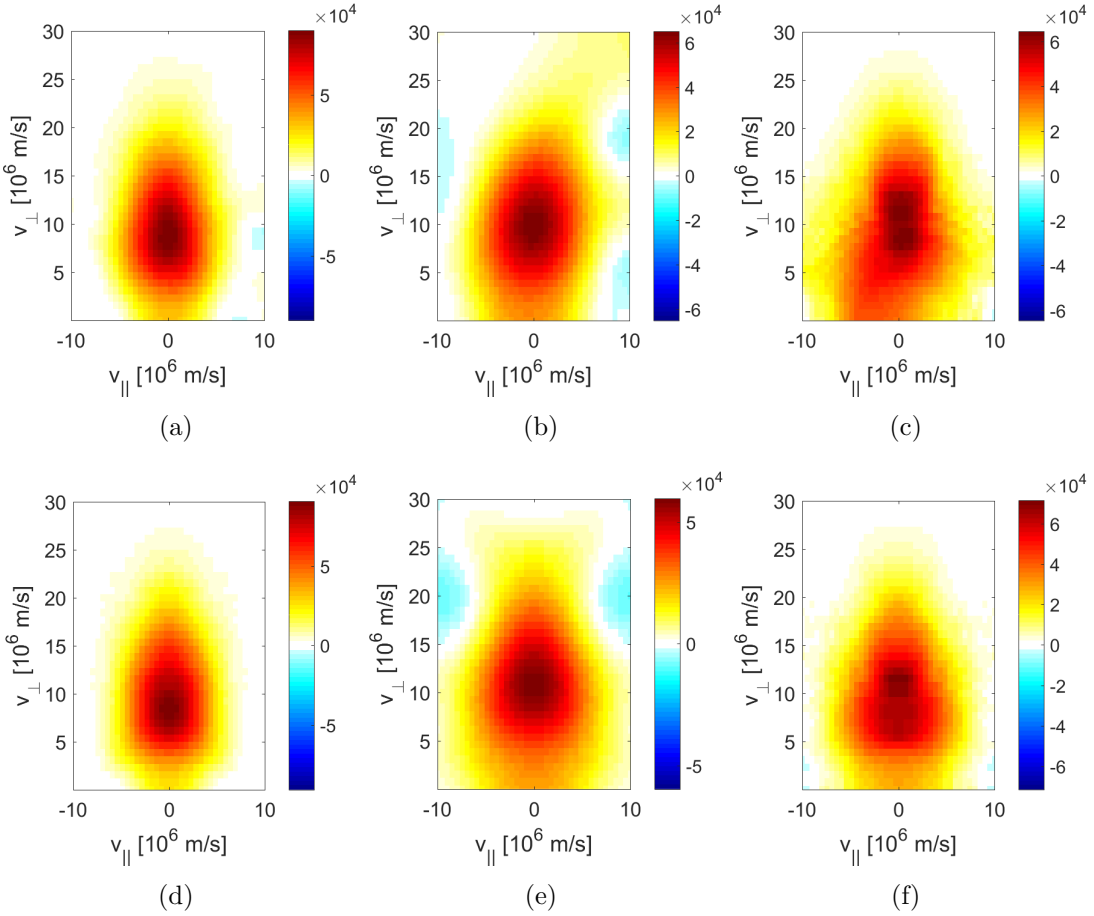


Figure 4.6: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1a made using 100 points per view under inverse crime conditions. The used diagnostics were (a,b) three NES-views at $\phi = 95°$, 135° and 160° for D-D fusion, (c) three one-step reaction GRS-views at $\phi = 100°$, 135° and 165° for p-D fusion with fast p and (d,e) three NES-views for D-D fusion or (f) three one-step reaction GRS-views for p-D fusion with fast p at $\phi = 45°$, 90° and 135°. $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ and zero drift for all six plots. The regularization parameter is (a,c,d,f) $\alpha = 10^8$ and (b,e) $\alpha = 10^{13}$.

We observed that the reconstructions for NES and one-step reaction GRS were tilted slightly compared to the used model distribution. These two diagnostic methods are very similar in nature as they both involve measuring the energy of a fusion reaction product. The cause of the observed tilt is thus likely the same in both cases. Given that the two sets of reconstructions were performed using only views at angles between zero and 90°, the tilt could be related to the fact that the diagnostics are only capable of detecting neutrons/photons moving towards the instrument, unlike CTS. In order to test this hypothesis, we perform some of the inverse crime NES- and one-step reaction GRS-reconstructions again using the corresponding observation angles between 90° and 180° ($\phi' = 180° - \phi$). The resulting reconstructions (Fig. 4.6(a,b,c)) are observed to be tilted the opposite way relative to the model distribution when compared to the previous reconstructions in Fig. A.2 and A.3. Otherwise, they resemble the previous reconstructions. This supports the idea that the tilting is related to the distribution of the views because the diagnostics can only detect neutrons/photons moving towards them. As a test of a possible solution, we perform the same reconstructions again using views distributed symmetrically around $\phi = 90°$. These reconstructions (Fig. 4.6(d,e,f)) are observed to not exhibit any noticeable tilting compared to the model distribution. They also resemble the model distribution at least as well as the previous reconstructions, if not better. We thus conclude that the diagnostic views used for NES and one-step reaction GRS must be evenly distributed around $\phi = 90°$ in order to avoid spurious tilting of the reconstructed distribution functions.
The diagnostics used for FIDA and two-step reaction GRS are also only capable of detecting photons moving towards them, but the related reconstructions (Figs. A.4 and A.5) do not exhibit any noticeable tilting compared to the relevant model distributions. This might be due to the more complicated nature of FIDA and two-step reaction GRS.
In order to deal with the spurious feature seen in the reconstructions for NES and one-step reaction GRS, we alter the code so that the gradient used for first-order Tikhonov regularization is removed from the regions where the weight function coverage is smaller than a chosen percentage of the average weight function coverage. This results in these regions effectively not being reconstructed. One can always set the percentage to zero if one does not wish to disregard any regions in the reconstructions. As an example, we perform another NES-reconstruction identical to the one in Fig. 4.6d, except that the gradient cutoff is set to 1 %. It is plotted next to the reconstruction from Fig. 4.6d for the sake of comparison. We see that the reconstruction with the gradient cutoff (Fig. 4.7b) contains a region at very low velocities that has not been reconstructed. This region covers part of the previously mentioned spurious feature. We could have made this region larger by using a larger cutoff percentage. In this way, it is thus possible to remove part or all of the spurious feature caused by low weight function coverage, though one can discuss whether or not the disregarded region is preferable to the spurious feature. Interestingly, however, we still see the remainder of the spurious feature on the left side of the disregarded region in Fig. 4.7b, but not on the right side.

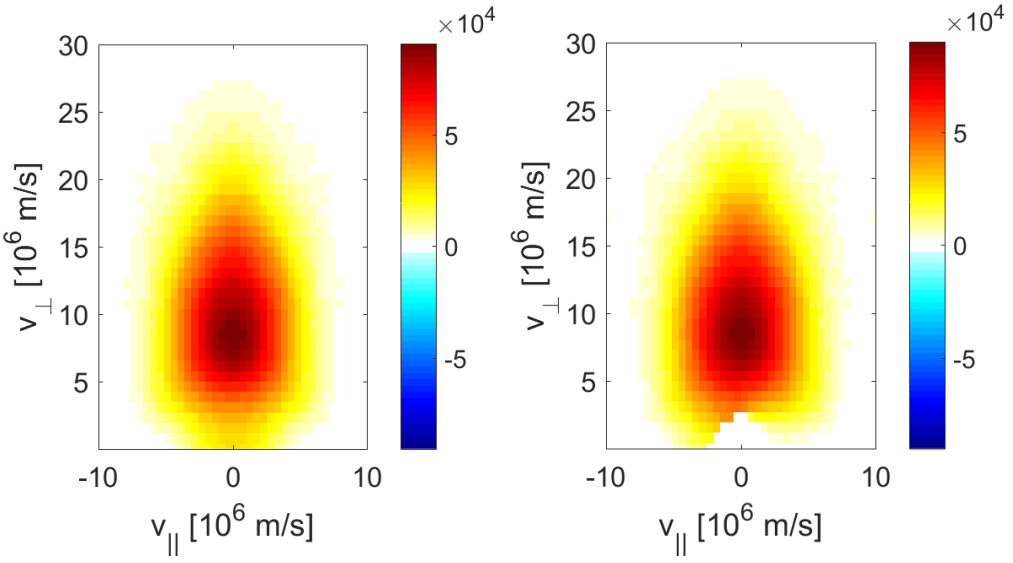


Figure 4.7: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1a made using three NES-views at $\phi = 45°$, $90°$ and $135°$ with 100 points per view for D-D fusion with zero drift, $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ and $\alpha = 10^8$ under inverse crime conditions. (a) no gradient cutoff. (b) 1 % gradient cutoff.

The right side instead resembles the corresponding part of the model distribution (Fig. 4.1a) quite well. This is most likely related to the implicit direction of the gradient and the way it enters in the Tikhonov regularization. We thus conclude that it is possible to use the gradient cutoff to remove the spurious feature from a larger region than just the disregarded one. This makes it an acceptable method for dealing with the spurious features caused by low weight function coverage.

Having concluded that our models work under inverse crime conditions, we move on to considering more realistic conditions where the forward and tomography grids have different sizes and noise is included. We start by performing first-order Tikhonov reconstructions of the previously considered model distributions using the same diagnostics, but with a 30×30 tomography grid and 3 % noise added to the synthetic spectra. We used 3 views with 100 points per view for all of the diagnostics. The views used for NES and one-step reaction GRS were evenly distributed around $\phi = 90°$, in accordance with our earlier findings. The gradient cutoff was 1 % for NES and 0.1 % for one-step reaction GRS. The best reconstruction from the set for each diagnostic, as judged by the Q-value, is shown in Fig. 4.8, and the full sets of reconstructions are shown in appendix A. Note that the inclusion of noise results in the reconstructions being slightly different each time they are performed.

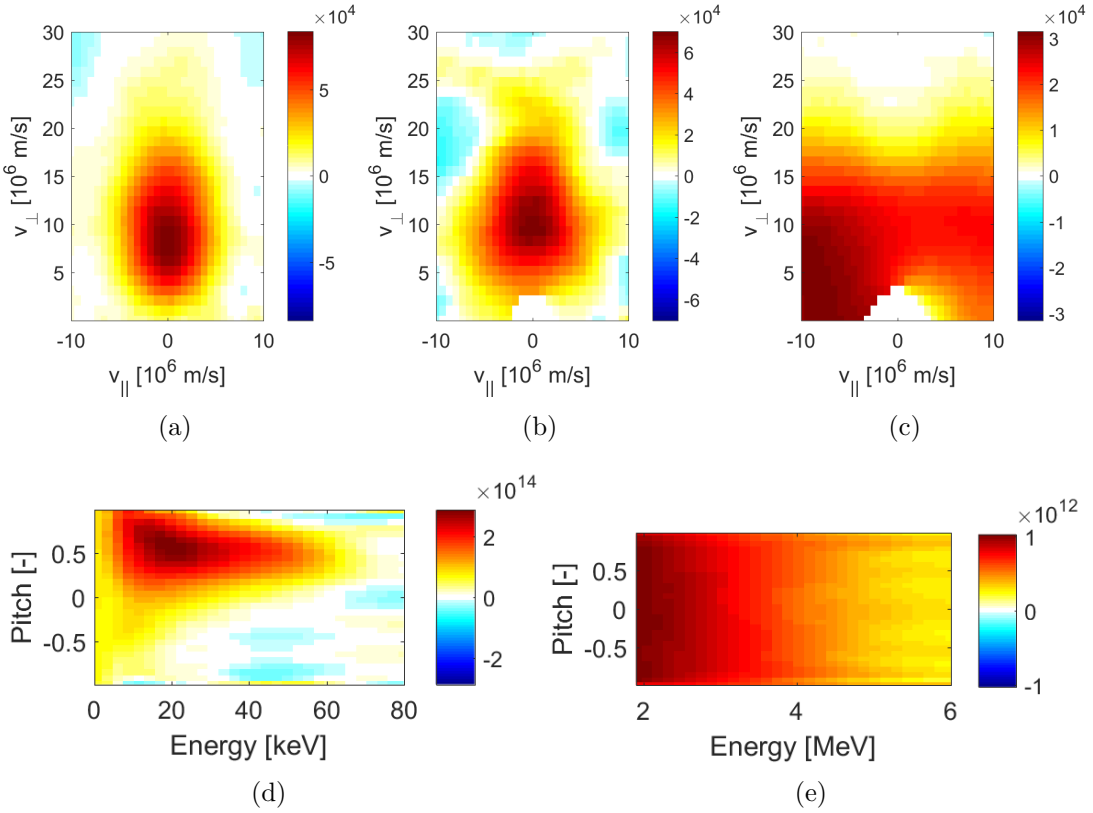


Figure 4.8: First-order Tikhonov reconstructions of the distribution function in (a,b,c) Fig. 4.1a, (d) Fig. 4.1b and (e) Fig. 4.1c made using 100 points per view on a 30×30 grid with 3 % noise added to the spectra and zero drift. The used diagnostics were (a) three CTS-views at $\phi = 10°$, 40° and 70° for fast D-ions, (b) three NES-views at $\phi = 45°$, 90° and 135° for D-D fusion, (c) three one-step reaction GRS-views at $\phi = 45°$, 90° and 135° for p-D fusion with fast p, (d) three FIDA-views at $\phi = 10°$, 40° and 70° for fast D with $B = 1.74\,\mathrm{T}$ and $\bar{\Gamma} = 0$ and (e) three two-step reaction GRS-views at $\phi = 30°$, 60° and 90° for Be-α fusion with fast α-particles and $n_{Be} = 1 \times 10^{18}\,\mathrm{m}^{-3}$. $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ for (b) and (c). The α-value is (a,c,e) 10^{13}, (b) 2.4×10^{12} and (d) 3.2×10^{10}.

Based on Fig. 4.8, we see that the reconstructions for CTS, NES, FIDA and two-step reaction GRS all resemble their related model distribution functions reasonably well, though the two-step reaction GRS-reconstruction does not really capture the pitch-independent nature of the related model distribution. These reconstructions do include some negative spurious features in the regions not covered by the related model distribution functions, but we know that these can be avoided by using the non-negativity constraint. Furthermore, our observations regarding the FIDA-reconstructions under inverse crime conditions indicate that using the non-negativity constraint would also improve the overall quality of the reconstructions. The reconstructions for CTS, NES, FIDA and two-step reaction GRS are thus perfectly acceptable. It is still worth noting that the reconstructed distribution values are seen to be somewhat lower than the model distribution

values, which is a common consequence of the regularization. The reconstruction for one-step reaction GRS, however, does not resemble the related model distribution function particularly well. It also includes features resembling "wings" that are not present in the model distribution. This will be discussed further later on. First, we wish to make some observations based on the full sets of reconstructions. All of the reconstructions (Figs. A.6, A.7, A.8, A.9 and A.10) are observed to transition from mostly noise to more definite structures as the value of α is increased. As indicated by the α-values used in Fig. 4.8, however, the optimal value differs for certain diagnostics. Some of the reconstructions have also clearly been over-regularized. We will thus briefly highlight noteworthy aspects of each set of reconstructions.
The CTS-reconstructions (Fig. A.6) show reasonable resemblance to the model distribution (Fig. 4.1a) for the three largest α-values, and the best reconstruction is the one for the largest α-value, $\alpha = 10^{13}$.
The NES-reconstructions (Fig. A.7) show reasonable resemblance to the model distribution (Fig. 4.1a) for the three largest α-values, and the best reconstruction is the one for the second-largest α-value, $\alpha = 2.4 \times 10^{12}$. The reconstruction for $\alpha = 1 \times 10^{13}$ has been regularized slightly too much, which indicates that this α-value is too large for NES under the considered circumstances.
The FIDA-reconstructions (Fig. A.8) show recognizable structures for all of the used values of α. The three reconstructions for the middle values of α all show reasonable resemblance to the model distribution (Fig. 4.1b), and the best reconstruction is the one for the middle α-value, $\alpha = 3.2 \times 10^{10}$. The reconstructions for the two lowest values of α are more noisy, and the reconstructions for the two largest values of α have been regularized too much, which indicates that these α-values are too large for FIDA under the considered circumstances.
The two-step reaction GRS-reconstructions (Fig. A.9) show reasonable resemblance to the model distribution (Fig. 4.1c) for the two largest used values of α, and the best reconstruction is the one for the largest α-value, $\alpha = 10^{13}$.

None of the one-step reaction GRS-reconstructions for fast p (Fig. A.10) bear more than a poor resemblance to the model distribution (Fig. 4.1a). They are also all seen to include at least a hint of the previously mentioned spurious "wings." Unfortunately, we are not quite certain why these spurious features appear. Our initial theory was that they were related to the different grid sizes and the fact that the weight functions for one-step reaction GRS are narrow circular arcs, as was shown in Fig. 2.6. After all, the actual weight functions exist in specific regions of velocity-space, and our weight functions are essentially obtained by sampling these actual weight functions at the chosen grid points. It is thus not unthinkable for the weight functions obtained by sampling the narrow actual weight functions on two different grids to be slightly displaced in velocity-space with respect to each other. This mismatch between the weight functions used to generate synthetic spectra and the weight functions used to reconstruct the distribution could then lead to spurious features in the reconstructions, such as the observed "wings". This would

also explain why we did not observe the "wings" under inverse crime conditions or for any of the other diagnostics that have, on average, much less narrow weight functions.
As a test of this hypothesis, we performed another set of one-step reaction GRS-reconstructions under identical circumstances, except that we used larger grids. More specifically, we used a 80×80 forward grid and a 60×60 tomography grid. The number of points per view was also changed to 400 in order to maintain the same ratio of spectra points to grid points. This was based on the fact that using more grid points will decrease the size of the individual grid points and the distance between them, which should limit the possible displacement of the weight functions. However, the resulting reconstructions were observed to still depict the spurious "wings." This indicates that the weight function mismatch is not the true cause of these spurious features. We also tried using even larger grids, different views, more views, different model distributions or even removing the noise, but were unable to get rid of the spurious "wings" and obtain reasonable reconstructions.
Fortunately, the problem is less pronounced for one-step reaction GRS with fast D-ions. While we could not obtain significantly better reconstructions of the model distribution in Fig. 4.1a for fast D-ions, we were able to obtain a reasonable reconstruction of a different model distribution by using larger grids. The other parameters were kept the same. The model distribution (Fig. 4.9a) was a bi-Maxwellian with $T_\parallel = 1\,\mathrm{MeV}$, $T_\perp = 1.5\,\mathrm{MeV}$, $n_i = 1 \times 10^{19}\,\mathrm{m}^{-3}$ and zero drift considered in the region where $v_\perp \leq 30 \times 10^6\,\mathrm{m/s}$ and $v_\parallel \in [-15; 15] \cdot 1 \times 10^6\,\mathrm{m/s}$. The best reconstruction, as judged by the Q-value, is shown in Fig. 4.9b, the second-best in Fig. 4.9c, and the full set of reconstructions is shown in appendix A.

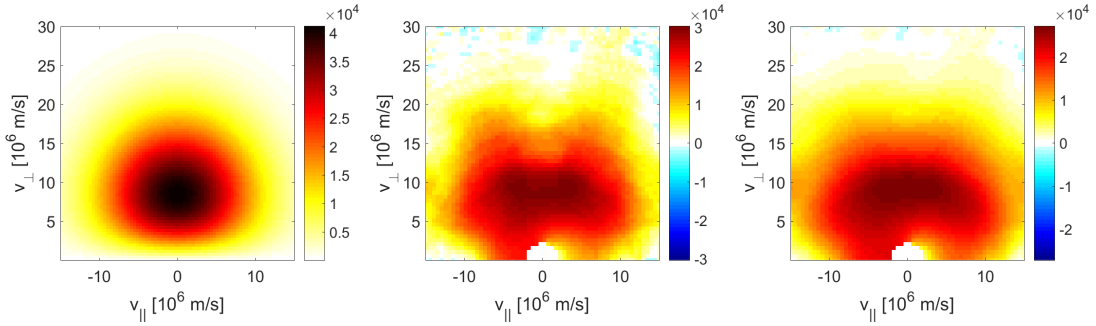


Figure 4.9: (a) bi-Maxwellian model distribution with $T_\parallel = 1\,\mathrm{MeV}$, $T_\perp = 1.5\,\mathrm{MeV}$, $n_i = 1 \times 10^{19}\,\mathrm{m}^{-3}$ and zero drift on a 80×80 grid. (b,c) First-order Tikhonov reconstructions of the model distribution in (a) made using three one-step reaction GRS-views at $\phi = 45°$, $90°$ and $135°$ with 400 points per view for p-D fusion with fast D, zero drift, $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ and a 0.1 % gradient cutoff on a 60×60 grid with 3 % noise added to the spectra. The lowest Q was obtained for (b) with $\alpha = 1.3 \times 10^{11}$, and the second-lowest Q was obtained for (c) with $\alpha = 5.6 \times 10^{11}$.

The one-step reaction GRS-reconstruction in Fig. 4.9b is seen to resemble the model distribution in Fig. 4.9a reasonably well. It does contain a hint of the spurious wings, but the resemblance is still reasonable. The wings are also absent from the second-best reconstruction in Fig. 4.9c that still resembles the model distribution reasonably well. It is thus possible to avoid the spurious wings and obtain reasonable reconstructions by using one-step reaction GRS for fast D-ions, at least in some cases. The full set of reconstructions in Fig. A.11 shows that the wings are visible in the reconstructions for low α-values and disappear in the reconstructions for large α-values.
We were also able to obtain reasonable one-step reaction GRS-reconstructions for fast D-ions of other model distributions, but not always. Considering fast D thus allows us to obtain reasonable reconstructions in situations where considering fast p does not. This is interesting given that considering fast D instead of fast p only changes the fast ion mass and the ratio for the center-of-mass energy. We thus conclude that the one-step reaction GRS model gives better reconstructions for fast D than for fast p, though both cases are affected by spurious features with unknown causes.

We have now demonstrated that it is possible to obtain reasonable reconstructions using any of the five considered diagnostics under realistic conditions, though one-step reaction GRS only works in some cases. As previously mentioned regarding the reconstructions under inverse crime conditions, we could almost certainly have obtained even better reconstructions by using more views or more points per view. This also relates to the practical applications of the models as one can evaluate the suitability of a given diagnostic setup for a given situation via the reconstructions that can be obtained with that setup for that situation.
We now move on to more specific practical applications. To begin with, we consider a situation where we have three FIDA views at $\phi = 10°$, $20°$ and $70°$ and wish to determine the optimal observation angles to use for two new FIDA views. To this end, we first wish to determine the optimal α-value for the three existing views. This is done by performing a series of reconstructions of the model distribution in Fig. 4.1b using the three views with 100 points per view, $B = 1.74\,\mathrm{T}$ and $\bar{\Gamma} = 0$ for various α-values and calculating the associated quality factor, Q, using eq. 2.13. The reconstructions were carried out using a 40×40 forward grid and a 30×30 tomography grid with 1 % noise added to the spectra. The resulting Q-values were plotted in a semi-logarithmic plot as a function of the used α-values, and the plot is shown in Fig. 4.10.
Based on Fig. 4.10 and the data used to make it, we determine that the optimal α-value for the three existing FIDA views is $\alpha = 1.5 \times 10^{10}$. Using this α-value, we then carry out a series of reconstructions of the model distribution in Fig. 4.1b for the three existing FIDA views and two extra views at various angles and calculate the Q-values. The observation angles of the two extra views are changed in increments of $2°$. We only consider angles between $0°$ and $90°$ because we have not observed any need for angles greater than $90°$ for FIDA-reconstructions. The

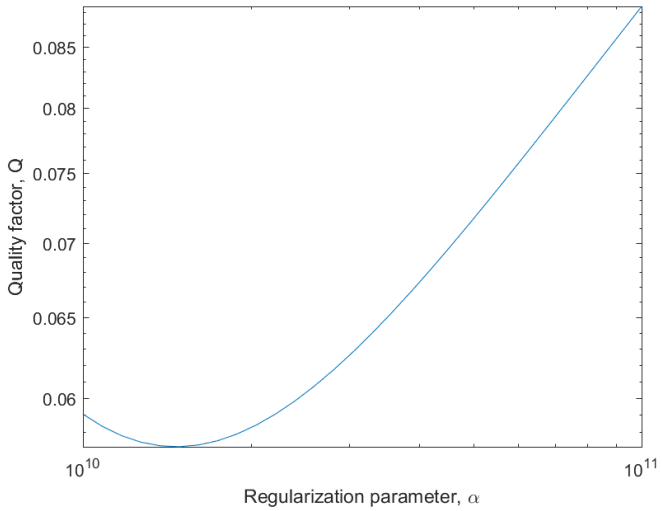


Figure 4.10: Semi-logarithmic plot of Q as a function of α for first-order Tikhonov reconstructions of the distribution function in Fig. 4.1b. The reconstructions were made using three FIDA views at $\phi = 10°$, $20°$ and $70°$ with 100 points per view for fast D-ions with $B = 1.74\,\mathrm{T}$ and $\bar{\Gamma} = 0$ on a 30×30 grid with no gradient cutoff and $1\,\%$ noise added to the spectra.

other parameters are the same as in Fig. 4.10. The resulting Q-values are plotted in a 2D-plot in Fig. 4.11 as a function of the observation angles of the two extra views.
Based on the plot in Fig. 4.11, we note that we generally gain the least by placing the new views at angles around $10°$, $20°$ and $70°$, which makes sense given that these are the angles of the existing views. Interestingly, the worst results are seen to be obtained if one of the new views is placed at an angle $< 10°$, and the other is placed at an angle $< 40°$. This is most likely due to a combination of these placements overlapping with the existing views and being the least suited for reconstructing the used model distribution. Some of the Q-values in this region are actually larger than the minimum Q-value in Fig. 4.10, but this might simply be because the optimal α-value for the three existing views is not the optimal value for five views. In order to avoid this, one could repeat the plot after determining the optimal angles for the two new views with an α-value optimized for the five resulting views. However, given that all of the Q-values are calculated for the same α-value, Fig. 4.11 should be a good indicator of the optimal angles for the new views, regardless of the absolute sizes of the Q-values. The optimal angles are seen to be around $\phi_{1,2} \approx 48°, 58°$ or $\phi_{1,2} \approx 50°, 80°$. This fits quite well with what was shown in a similar plot in [51]. The overall structure in Fig. 4.11 is also generally the same as in the plot in [51]. Any differences could be related to differences in the model distribution or the used tomographic inversion method. We thus conclude that our models can be used to determine the optimal placement of new diagnostics given an existing diagnostic setup.

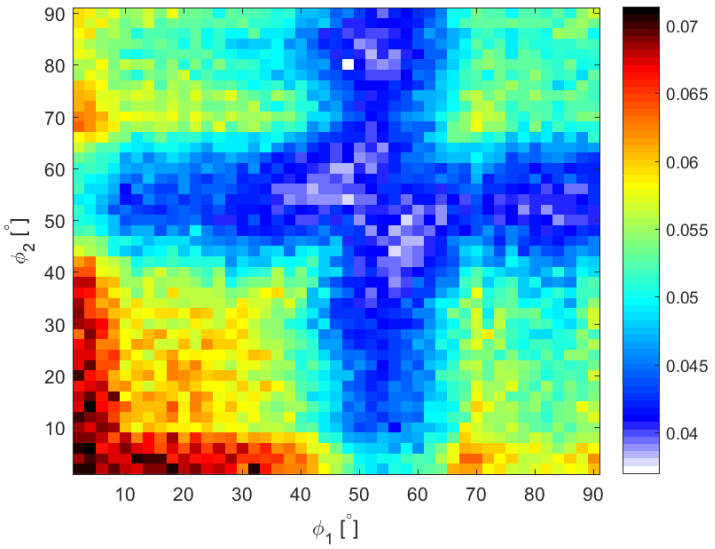


Figure 4.11: 2D-plot of Q as a function of ϕ_1 and ϕ_2 for first-order Tikhonov reconstructions of the distribution function in Fig. 4.1b. The reconstructions were made using five FIDA views at $\phi = 10°$, $20°$, $70°$, ϕ_1 and ϕ_2 with 100 points per view for fast D-ions with $B = 1.74\,\mathrm{T}$, $\bar{\Gamma} = 0$ and $\alpha = 1.5 \times 10^{10}$ on a 30×30 grid with no gradient cutoff and 1 % noise added to the spectra.

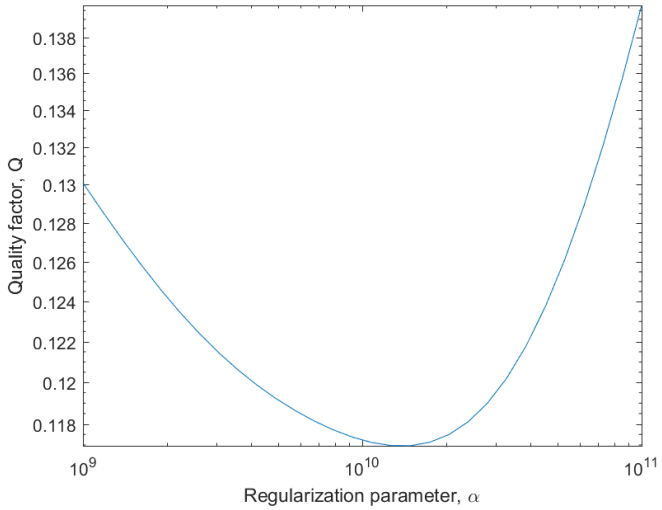


Figure 4.12: Semi-logarithmic plot of Q as a function of α for first-order Tikhonov reconstructions of a 40×40 grid version of the distribution function shown in Fig. 4.9a. The reconstructions were made using three one-step reaction GRS views at $\phi = 45°$, $90°$ and $135°$ with 100 points per view for p-D fusion with fast D-ions, zero drift and $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ on a 30×30 grid with a 0.1 % gradient cutoff and 1 % noise added to the spectra.

As a further test of this practical application, we consider a situation where we have three one-step reaction GRS views for p-D fusion with fast D at $\phi = 45°$, $90°$ and $135°$ and wish to determine the optimal observation angles to use for two complementary NES views for D-D fusion. The procedure is the same as before. We first wish to determine the optimal α-value for the three existing views and thus perform a series of reconstructions of a 40×40 grid version of the model distribution shown in Fig. 4.9a for different α-values. The reconstructions are made using the three views with 100 points per view, zero drift and $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$, and the associated quality factors, Q, are calculated using eq. 2.13. The reconstructions were carried out using a 40×40 forward grid and a 30×30 tomography grid with 1 % noise added to the spectra and a 0.1 % gradient cutoff. The resulting Q-values were plotted in a semi-logarithmic plot as a function of the used α-values, and the plot is shown in Fig. 4.12.
Based on Fig. 4.12 and the data used to make it, we determine that the optimal α-value for the three existing one-step reaction GRS views is $\alpha = 1.5 \times 10^{10}$. Using this α-value, we then carry out a series of reconstructions of the model distribution shown in Fig. 4.9a for the three existing one-step reaction GRS views and two NES views at various angles and calculate the Q-values. The observation angles of the two NES views are changed in increments of $2°$. In this case, we consider angles between $0°$ and $180°$, in accordance with our earlier findings regarding NES and one-step reaction GRS. The other parameters are the same as in Fig. 4.12. The resulting Q-values are plotted in a 2D-plot in Fig. 4.13 as a function of the observation angles of the two NES views.
Based on Fig. 4.13, we note that we generally gain the least when both NES-views are placed at angles between $40°$ and $60°$ or $120°$ and $140°$. This makes sense given that placing the two views at the same observation angle is redundant, but it is interesting that these particular regions give worse results than the diagonal in general. This might be because these are the placements that cause the NES views to most closely overlap with the existing one-step reaction GRS views. After all, two different diagnostics placed at the same observation angle are not necessarily sensitive to the same regions in velocity-space, as indicated by the various weight functions in the theory section. This would also explain why we do not see particularly bad results at all of the placements around $45°$, $90°$ and $135°$, despite these being the angles of the existing one-step reaction GRS views. The optimal angles seem to be around $\phi_{1,2} \approx 95°, 135°$, though there are extensive regions in Fig. 4.13 that give almost as good results. An optimum is also seen around one of the $\phi_{1,2} \approx 25°, 90°$ points, but it is not mirrored in the diagonal so it might not be a true optimum. Interestingly, the observed optimal angles overlap heavily with the angles of the existing one-step reaction GRS views and are not symmetric around $90°$. The overlap can be explained by the previously mentioned different velocity-space sensitivities of different diagnostics. The lack of symmetry might be caused by the symmetrically placed one-step reaction GRS views being sufficient to avoid spurious tilting.

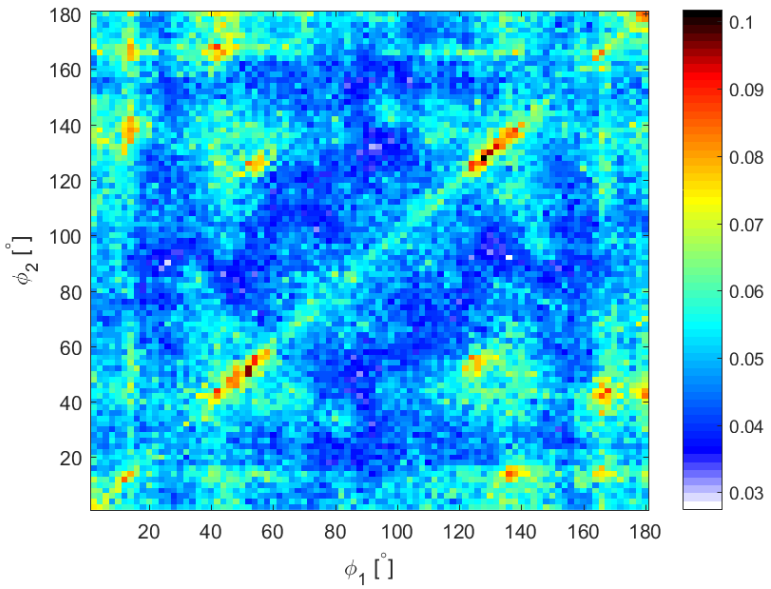


Figure 4.13: 2D-plot of Q as a function of ϕ_1 and ϕ_2 for first-order Tikhonov reconstructions of a 40×40 grid version of the distribution function shown in Fig. 4.9a. The reconstructions were made using three one-step reaction GRS views for p-D fusion with fast D at $\phi = 45°$, $90°$ and $135°$ and two NES views for D-D fusion at ϕ_1 and ϕ_2 with 100 points per view for $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$, zero drift and $\alpha = 1.5 \times 10^{10}$ on a 30×30 grid with a 0.1 % gradient cutoff and 1 % noise added to the spectra.

Alternatively, it is possible that the placements at $\phi_{1,2} \approx 95°, 135°$ improve the reconstructions enough to compensate for the addition of some spurious tilting. It is, however, potentially a good thing that the optimal angles for the NES views overlap with the angles of the existing one-step reaction GRS views, given that it is possible to construct combined NES/GRS diagnostics. [43] [30] [33] All of the Q-values in Fig. 4.13 are lower than the minimum Q-value in Fig. 4.12, which indicates that the chosen α-value is also suitable for the situation with the two added NES views.

As a final demonstration of this practical application of our models, we consider a situation where we have three two-step reaction GRS views for Be-α fusion with fast α-particles at $\phi = 30°$, $60°$ and $90°$ and wish to determine the optimal observation angles to use for two complementary CTS views for fast α-particles. The procedure is once again the same. We first carry out a series of reconstructions of the model distribution in Fig. 4.1c using the existing views for various α-values and calculate the associated Q-values using eq. 2.13. The reconstructions were carried out using a 40×40 forward grid and a 30×30 tomography grid with zero drift, $n_{Be} = 1 \times 10^{18}\,\mathrm{m}^{-3}$, 1 % noise added to the spectra and no gradient cutoff. The Q-values are plotted in a logarithmic plot as a function of the used α-values in Fig. 4.14.

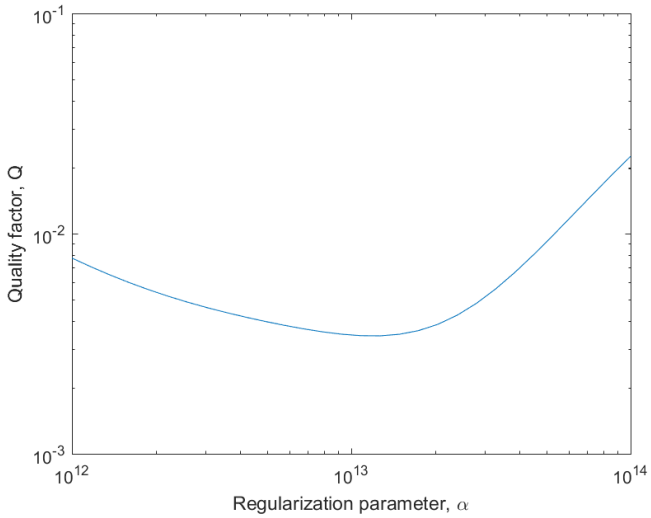


Figure 4.14: Logarithmic plot of Q as a function of α for first-order Tikhonov reconstructions of the distribution function in Fig. 4.1c. The reconstructions were made using three two-step reaction GRS views at $\phi = 30°$, $60°$ and $90°$ with 100 points per view for fast α-particles with zero drift and $n_{Be} = 1 \times 10^{18}\,\mathrm{m}^{-3}$ on a 30×30 grid with no gradient cutoff and 1 % noise added to the spectra.

Based on Fig. 4.14 and the data used to make it, we determine that the optimal α-value for the three existing two-step reaction GRS views is $\alpha = 1.3 \times 10^{13}$. Using this α-value, we then carry out a series of reconstructions of the model distribution shown in Fig. 4.1c for the three existing two-step reaction GRS views and two CTS views at various angles and calculate the Q-values. The observation angles of the two CTS views are changed in increments of 2°. We again only consider angles between 0° and 90° because we have not observed any need for angles greater than 90° for two step reaction GRS- or CTS-reconstructions. The other parameters are the same as in Fig. 4.14. The resulting Q-values are plotted in a 2D-plot in Fig. 4.15 as a function of the observation angles of the two CTS views.
Based on Fig. 4.15, we note that we generally gain the least by either placing both CTS views at angles around 68° or placing them at angles around $\phi_{1,2} \approx 30°, 75°$. We also note that Fig. 4.15 generally shows relatively low Q-values for much larger regions than the two previously considered cases. Furthermore, it contains many more optimal angles than were seen in either of the previous cases. However, almost all of the Q-values in Fig. 4.15 are larger than the minimum Q-value in Fig. 4.14. This indicates that the chosen α-value is ill-suited for the situation with the three two-step reaction GRS views and the two added CTS views. Despite this, Fig. 4.15 should still be a good indicator of the optimal placements of the two CTS views given that all of the Q-values are calculated for the same α-value, as previously mentioned. The optimal angles are seen to be around $\phi_{1,2} \approx 45°, 80°$ or $\phi_{1,2} \approx 30°, 50°$ with a large region of potential optimal angles around $\phi_{1,2} \approx 30°, 50°$.

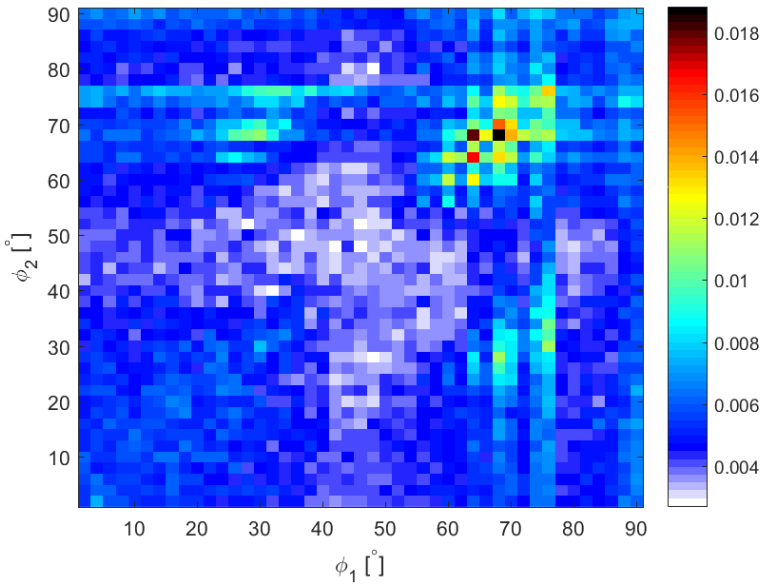


Figure 4.15: 2D-plot of Q as a function of ϕ_1 and ϕ_2 for first-order Tikhonov reconstructions of the distribution function shown in Fig. 4.9c. The reconstructions were made using three two-step reaction GRS views for Be-α fusion with fast α-particles at $\phi = 30°$, $60°$ and $90°$ and two CTS views for fast α-particles at ϕ_1 and ϕ_2 with 100 points per view for zero drift, $n_{Be} = 1 \times 10^{18}\,\mathrm{m}^{-3}$ and the regularization parameter $\alpha = 1.3 \times 10^{13}$ on a 30×30 grid with no gradient cutoff and 1 % noise added to the spectra.

The angles around $\phi_{1,2} \approx 30°, 50°$ partially overlap with the angles of the existing views, but this can be explained by the previously mentioned different velocity-space sensitivities for different diagnostics. It is also worth noting that the Q-values along parts of the diagonal are not noticeably worse than the values in much of the rest of Fig. 4.15. This is interesting given that the diagonal represents redundant placements of the two CTS views. One possible explanation is that the placements represented by the regions that can be compared to parts of the diagonal are not noticeably better than those redundant placements. This might, in turn, be explained by the interaction between the different velocity-space sensitivities of the two different diagnostics. Regardless, we were still able to determine optimal observation angles for the two CTS views.
We have now demonstrated three potential applications of our analytic models to the task of determining suitable additions to existing diagnostic setups. The results can always be used in conjunction with the existing numerical models in order to obtain more accurate data.

5 Conclusion

We have written analytic models for velocity-space tomography based on Tikhonov regularization using any combination of CTS, FIDA, NES, one-step reaction GRS and two-step reaction GRS. These models were tested through the use of bi-Maxwellian and slowing-down model distributions, and we demonstrated that they can be used to evaluate the suitability of a given diagnostic setup or potential additions thereto for a given situation. During our testing, we also found that the views used for NES and one-step reaction GRS need to be evenly distributed around $\phi = 90°$ in order to avoid certain spurious features in the reconstructed distribution functions. Despite some observed issues with the use of one-step reaction GRS, the analytic models are thus quite capable of complementing the existing numerical models.

References

[1] Aster, R. C., Borchers, B., and Thurber, C. H. *Parameter Estimation and Inverse Problems.* Academic Press, 2004.

[2] Bindslev, H., Meo, F., and Korsholm, S. B. ITER Fast Ion Collective Thomson Scattering: Feasibility study. *Risø National Laboratory* (2003).

[3] Bosch, H.-S., and Hale, G. M. Improved formulas for fusion cross-sections and thermal reactivities. *Nucl. Fusion 32, 611* (1992).

[4] Bruzzone, P., Sedlak, K., Stepanov, B., Wesche, R., Muzzi, L., Seri, M., Zani, L., and Coleman, M. Design, Manufacture and Test of a 82 kA React&Wind TF conductor for DEMO. *IEEE Transactions on Applied Superconductivity 26, 4801805* (2016).

[5] Chen, S., Wang, J., Wang, F., Wang, J., and Hu, L. Asymmetrical Common-Cause Failures Analysis Method Applied in Fusion Reactors. *Journal of Fusion Energy 35, p. 221-228* (2016).

[6] Dresselhaus, M. S., and Thomas, I. L. Alternative energy technologies. *Nature 414, 332-337* (2001).

[7] Freidberg, J. *Plasma Physics and Fusion Energy.* Cambridge University Press, 2007.

[8] Fusion for Energy. The Merits of Fusion, 21-06-13. *http://fusionforenergy.europa.eu/understandingfusion/merits.aspx* (2016).

[9] Fusion for Energy. What is Fusion?, 21-06-13. *http://fusionforenergy.europa.eu/understandingfusion/* (2016).

[10] Gaffey, J. D. Energetic ion distribution resulting from neutral beam injection in tokamaks. *J. Plasma Phys. 16, 149* (1976).

[11] Geiger, B., Karpushov, A. N., Duval, B. P., Marini, C., Sauter, O., Andrebe, Y., Testa, D., Marascheck, M., Salewski, M., Schneider, P. A., the TCV Team, and the EUROfusion MST1 Team. Fast-ion transport in low density L-mode plasmas at TCV using FIDA spectroscopy and the TRANSP code. *Plasma Phys. Control. Fusion 59, 115002* (2017).

[12] General Fusion. Energy supply: The #1 challenge of the 21st century, 21-06-18. *http://generalfusion.com/what-are-the-benefits-of-fusion-energy/* (2018).

[13] Hansen, P. C. *Discrete Inverse Problems; Insight and Algorithms.* SIAM, 2010.

[14] Herzog, A. V., Lipman, T. E., and Kammen, D. M. Renewable energy sources. *Available from* `http://www.eolss.com` (2004).

[15] ITER Organization. What is ITER?, 09-06-16. `https://www.iter.org/proj/inafewlines` (2016).

[16] Jacobsen, A. S., Binda, F., Cazzaniga, C., Eriksson, J., Hjalmarsson, A., Nocente, M., Salewski, M., Tardini, G., the JET Contributors, and the ASDEX Upgrade Team. Velocity-space sensitivities of neutron emission spectrometers at the tokamaks JET and ASDEX Upgrade in deuterium plasmas. *Rev. Sci. Instrum. 88, 073506* (2017).

[17] Jacobsen, A. S., Naulin, V., and Salewski, M. Methods to determine fast-ion distribution functions from multi-diagnostic measurements. *PhD Thesis, Department of Physics, Technical University of Denmark* (2015).

[18] Jacobsen, A. S., Salewski, M., Eriksson, J., Ericsson, G., Korsholm, S. B., Leipold, F., Nielsen, S. K., Rasmussen, J., Stejner, M., and the JET EFDA Contributors. Velocity-space sensitivity of neutron spectrometry measurements. *Nucl. Fusion 55, 053013* (2015).

[19] Jacobsen, A. S., Stagner, L., Salewski, M., Geiger, B., Heidbrink, W. W., Korsholm, S. B., Leipold, F., Nielsen, S. K., Rasmussen, J., Stejner, M., Thomsen, H., Weiland, M., and the ASDEX Upgrade Team. Inversion methods for fast-ion velocity-space tomography in fusion plasmas. *Plasma Phys. Control. Fusion 58, 045016* (2016).

[20] Johansson, T. B., Kelly, H., Reddy, A. K. N., Williams, R. H., and Burnham, L., Eds. *Renewable Energy: Sources for Fuels and Electricity.* Island Press, 1993.

[21] Kaipio, J., and Somersalo, E. Statistical inverse problems: Discretization, model reduction and inverse crimes. *Journal of Computational and Applied Mathematics 198, pp. 493-504* (2007).

[22] Kim, K., Im, K., Kim, H. C., Oh, S., Park, J. S., Kwon, S., Lee, Y. S., Yeom, J. H., Lee, C., Lee, G.-S., Neilson, G., Kessel, C., Brown, T., Titus, P., Mikkelsen, D., and Zhai, Y. Design concept of K-DEMO for near-term implementation. *Nuclear Fusion 55, 053027* (2015).

[23] Kim, K., Kim, H. C., Oh, S., Lee, Y. S., Yeom, J. H., Im, K., Lee, G.-S., Neilson, G., Kessel, C., Brown, T., and Titus, P. A preliminary conceptual design study for korean fusion DEMO reactor. *Fusion Engineering and Design 88, p. 488-491* (2013).

[24] Kiptily, V. G., Cecil, F. E., and Medley, S. S. Gamma ray diagnostics of high temperature magnetically confined fusion plasmas. *Plasma Phys. Control. Fusion 48, R59-82* (2006).

[25] Kiptily, V. G., Popovichev, S., and Sharapov, S. E. Gamma-diagnostics of alpha-particles in He-4 and D?T plasmas. *Rev. Sci. Instrum. 74, 1753* (2003).

[26] Linares, J. I., Cantizano, A., Moratilla, B. Y., Martin-Palacios, V., and Batet, L. Supercritical CO2 Brayton power cycles for DEMO (demonstration power plant) fusion reactor based on dual coolant lithium lead blanket. *Energy 98, p. 271-283* (2016).

[27] Moseev, D., and Salewski, M. Bi-Maxwellian -, slowing-down -, and ring velocity distributions of fast ions in magnetized plasmas. *in preparation*.

[28] Nakamura, M., Tobita, K., Someya, Y., Utoh, H., Sakamoto, Y., and Gulden, W. Thermohydraulic responses of a water-cooled tokamak fusion DEMO to loss-of-coolant accidents. *Nuclear Fusion 55, 123008* (2015).

[29] Nocente, M., Garcia-Muñoz, M., Gorini, G., Tardocchi, M., Weller, A., Akaslompolo, S., Bilato, R., Bobkov, V., Cazzaniga, C., Geiger, B., Grosso, G., Herrmann, A., Kiptily, V., Maraschek, M., McDermott, R., Noterdaeme, J. M., Podoba, Y., Tardini, G., and the ASDEX Upgrade Team. Gamma-ray spectroscopy measurements of confined fast ions on ASDEX Upgrade. *Nucl. Fusion 52, 094021* (2012).

[30] Nocente, M., Rigamonti, D., Perseo, V., Tardocchi, M., Boltruczyk, G., Broslawski, A., Cremona, A., Croci, G., Gosk, M., Kiptily, V., Korolczuk, S., Mazzocco, M., Muraro, A., Strano, E., Zychor, I., Gorini, G., and the JET contributors. Gamma-ray spectroscopy at MHz counting rates with a compact LaBr3 detector and silicon photomultipliers for fusion plasma applications. *Rev. Sci. Instrum. 87, 11E714* (2016).

[31] Orsitto, F. P., Villari, R., Moro, F., Todd, T. N., Lilley, S., Jenkins, I., Felton, R., Biel, W., Silva, A., Scholz, M., Rzadkiewicz, J., Duran, I., Tardocchi, M., Gorini, G., Morlock, C., Federici, G., and Litnovsky, A. Diagnostics and control for the steady state and pulsed tokamak DEMO. *Nuclear Fusion 56, 026009* (2016).

[32] Palermo, I., Veredas, G., Gómez-Ros, J. M., Sanz, J., and Ibarra, A. Neutronic design studies of a conceptual DCLL fusion reactor for a DEMO and a commercial power plant. *Nuclear Fusion 56, 016001* (2016).

[33] Rigamonti, D., Muraro, A., Nocente, M., Perseo, V., Boltruczyk, G., Fernandes, A., Figueiredo, J., Giacomelli, L., Gorini, G., Gosk, M., Kiptily, V., Korolczuk, S., Mianowski, S., Murari, A., Pereira, R. C., Cippo, E. P., Zychor, I., Tardocchi, M., and the JET Contributors. Performance of the prototype LaBr3 spectrometer developed for the JET gamma-ray camera upgrade. *Rev. Sci. Instrum. 87, 11E717* (2016).

[34] Salewski, M., Geiger, B., Heidbrink, W. W., Jacobsen, A. S., Korsholm, S. B., Leipold, F., Madsen, J., Moseev, D., Nielsen, S. K., and Rasmussen, J. Doppler tomography in fusion plasmas and astrophysics. *Plasma Phys. Control. Fusion 57, 014021* (2015).

[35] Salewski, M., Geiger, B., Jacobsen, A. S., Abramovic, I., Korsholm, S. B., Leipold, F., Madsen, B., Madsen, J., McDermott, R. M., Moseev, D., Nielsen, S. K., Nocente, M., Rasmussen, J., M., S., Weiland, M., the EUROfusion MST1 team, and the ASDEX Upgrade team. Deuterium temperature, drift velocity, and density measurements in non-Maxwellian plasmas at ASDEX Upgrade. *Nucl. Fusion 58, 036017* (2018).

[36] Salewski, M., Geiger, B., Jacobsen, A. S., Garcia-Muñoz, M., Heidbrink, W. W., Korsholm, S. B., Leipold, F., Madsen, J., Moseev, D., Nielsen, S. K., Rasmussen, J., Stejner, M., Tardini, G., Weiland, M., and the ASDEX Upgrade Team. Measurement of a 2D fast-ion velocity distribution function by tomographic inversion of fast-ion D-alpha spectra. *Nucl. Fusion 54, 023005* (2014).

[37] Salewski, M., Geiger, B., Jacobsen, A. S., Hansen, P. C., Heidbrink, W. W., Korsholm, S. B., Leipold, F., Madsen, J., Moseev, D., Nielsen, S. K., Nocente, M., Odstrčil, T., Rasmussen, J., Stagner, L., Stejner, M., Weiland, M., and the ASDEX Upgrade Team. High-definition velocity-space tomography of fast-ion dynamics. *Nucl. Fusion 56, 106024* (2016).

[38] Salewski, M., Geiger, B., Moseev, D., Heidbrink, W. W., Jacobsen, A. S., Korsholm, S. B., Leipold, F., Madsen, J., Nielsen, S. K., Rasmussen, J., Stejner, M., Weiland, M., and the ASDEX Upgrade Team. On velocity-space sensitivity of fast-ion D-alpha spectroscopy. *Plasma Phys. Control. Fusion 56, 105005* (2014).

[39] Salewski, M., Geiger, B., Nielsen, S. K., Bindslev, H., Garcia-Muñoz, M., Heidbrink, H. H., Korsholm, S. B., Leipold, F., Madsen, J., Meo, F., Michelsen, P. K., Moseev, D., Stejner, M., Tardini, G., and the ASDEX Upgrade Team. Combination of fast-ion diagnostics in velocity-space tomographies. *Nucl. Fusion 53, 063019* (2013).

[40] Salewski, M., Geiger, B., Nielsen, S. K., Bindslev, H., Garcia-Muñoz, M., Heidbrink, W. W., Korsholm, S. B., Leipold, F., Meo, F., Michelsen, P. K., Moseev, D., Stejner, M., Tardini, G., and the ASDEX Upgrade Team. Tomography of fast-ion velocity-space distributions from synthetic CTS and FIDA measurements. *Nucl. Fusion 52, 103008* (2012).

[41] Salewski, M., Nielsen, S. K., Bindslev, H., Furtula, V., Gorelenkov, N. N., Korsholm, S. B., Leipold, F., Meo, F., Michelsen, P. K., Moseev, D., and Stejner, M. On velocity space interrogation regions of fast-ion collective Thomson scattering at ITER. *Nucl. Fusion 51, 083014* (2011).

[42] Salewski, M., Nocente, M., Gorini, G., Jacobsen, A. S., Kiptily, V. G., Korsholm, S. B., Leipold, F., Madsen, J., Moseev, D., Nielsen, S. K., Rasmussen, J., Stejner, M., Tardocchi, M., and the JET Contributors. Velocity-space observation regions of high-resolution two-step reaction gamma-ray spectroscopy. *Nucl. Fusion 55, 093029* (2015).

[43] Salewski, M., Nocente, M., Gorini, G., Jacobsen, A. S., Kiptily, V. G., Korsholm, S. B., Leipold, F., Madsen, J., Moseev, D., Nielsen, S. K., Rasmussen, J., Stejner, M., Tardocchi, M., and the JET Contributors. Fast-ion energy resolution by one-step reaction gamma-ray spectrometry. *Nucl. Fusion 56, 046009* (2016).

[44] Salewski, M., Nocente, M., Jacobsen, A. S., Binda, F., Cazzaniga, C., Ericsson, G., Eriksson, J., Gorini, G., Hellesen, C., Hjalmarsson, A., Kiptily, V. G., Koskela, T., Korsholm, S. B., Kurki-Suonio, T., Leipold, F., Madsen, J., Moseev, D., Nielsen, S. K., Rasmussen, J., Schneider, M., Sharapov, S. E., Stejner, M., Tardocchi, M., and the JET Contributors. MeV-range velocity-space tomography from gamma-ray and neutron emission spectrometry measurements at JET. *Nucl. Fusion 57, 056001* (2017).

[45] Shimwell, J., Kovari, M., Lilley, S., Zheng, S., Morgan, L. W. G., Packer, L. W., and McMillan, J. A parameter study of time-varying tritium production in solid-type breeder blankets. *Fusion Engineering and Design 104, p. 34-39* (2016).

[46] Sims, R. E. H., Rogner, H.-H., and Gregory, K. Carbon emission and mitigation cost comparisons between fossil fuel, nuclear and renewable energy resources for electricity generation. *Energy Policy 31, 1315-1326* (2003).

[47] Sundén, E. A., Ericsson, G., Cecconello, M., Conroy, S., Johnson, M. G., Giacomelli, L., Hellesen, C., Hjalmarsson, A., Källne, J., Ronchi, E., Sangaroon, S., Sjöstrand, H., and

Weiszflog, M. Instrumentation for neutron emission spectrometry in use at JET. *Nuclear Instruments and Methods in Physics Research A 623, 681-685* (2010).

[48] Tamura, H., Yanagi, N., Takahata, K., Sagara, A., Ito, S., and Hashizume, H. Multiscale Stress Analysis and 3D Fitting Structure of Superconducting Coils for the Helical Fusion Reactor. *IEEE Transactions on Applied Superconductivity 26, 4202405* (2016).

[49] Wan, B., Ding, S., Qian, J., Li, G., Xiao, B., and Xu, G. Physics Design of CFETR: Determination of the Device Engineering Parameters. *IEEE Transactions on Plasma Science 42, p. 495-502* (2014).

[50] Weiland, M., Bilato, R., Geiger, B., Schneider, P. A., Tardini, G., Garcia-Muñoz, M., Ryter, F., Salewski, M., Zohm, H., the ASDEX Upgrade Team, and the EUROfusion MST1 Team. Phase-space resolved measurement of 2nd harmonic ion cyclotron heating using FIDA tomography at the ASDEX Upgrade tokamak. *Nucl. Fusion 57, 116058* (2017).

[51] Weiland, M., Geiger, B., Jacobsen, A. S., Reich, M., Salewski, M., Odstrčil, T., and the ASDEX Upgrade Team. Enhancement of the FIDA diagnostic at ASDEX Upgrade for velocity space tomography. *Plasma Phys. Control. Fusion 58, 025012* (2016).

[52] Wirgin, A. The inverse crime. *arXiv:math-ph/0401050* (2004).

[53] World Nuclear Association. Nuclear Fusion Power, 06-04-16. *http://www.world-nuclear.org/information-library/current-and-future-generation/nuclear-fusion-power.aspx* (2016).

Appendix A

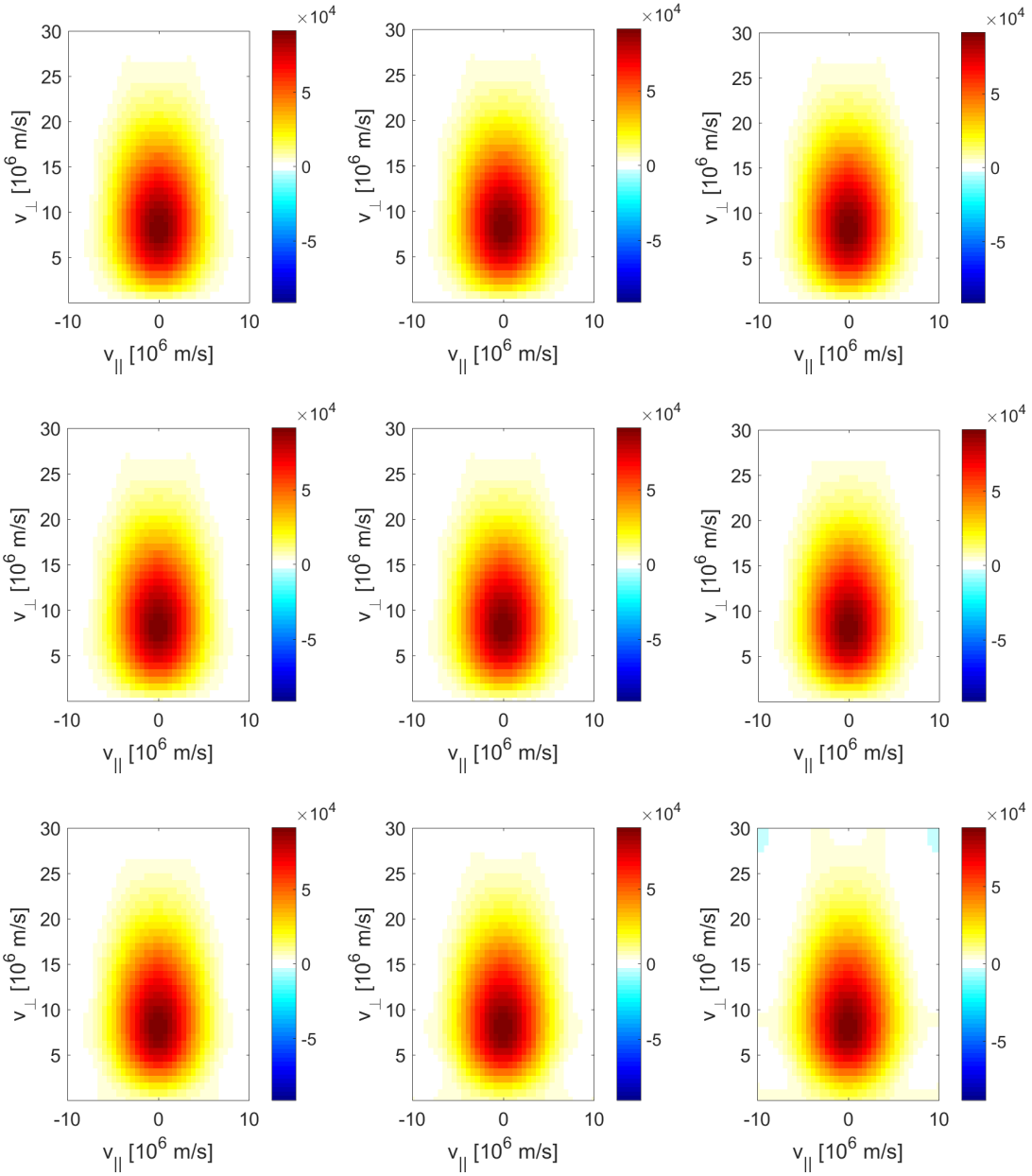


Figure A.1: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1a made using three CTS-views at $\phi = 10°$, $40°$ and $70°$ with 100 points per view for fast D-ions under inverse crime conditions. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

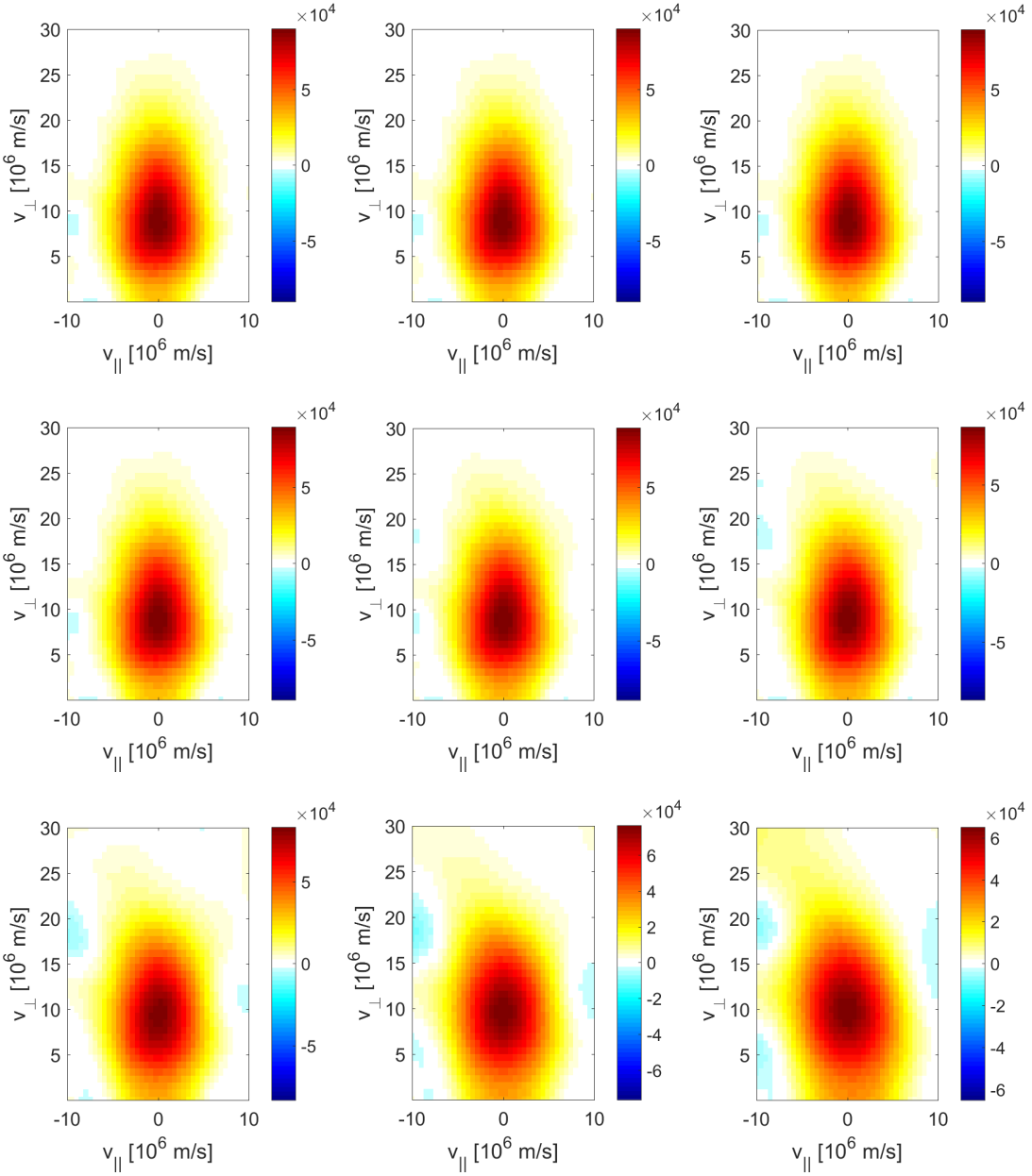


Figure A.2: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1a made using three NES-views at $\phi = 20°$, $45°$ and $85°$ with 100 points per view for D-D fusion with zero drift and $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ under inverse crime conditions. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

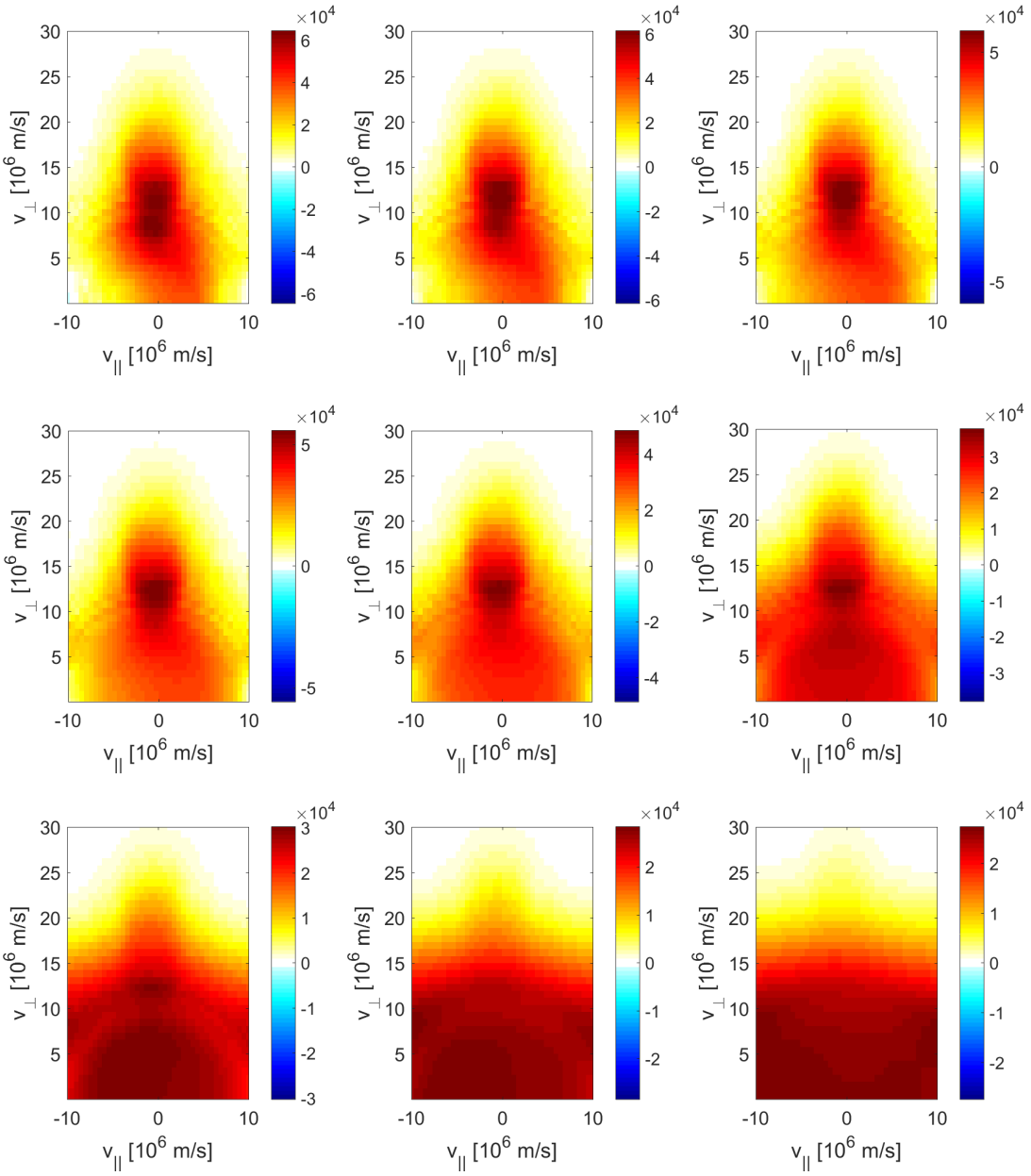


Figure A.3: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1a made using three one-step reaction GRS-views at $\phi = 15°$, $45°$ and $80°$ with 100 points per view for p-D fusion with fast p, zero drift and $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ under inverse crime conditions. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

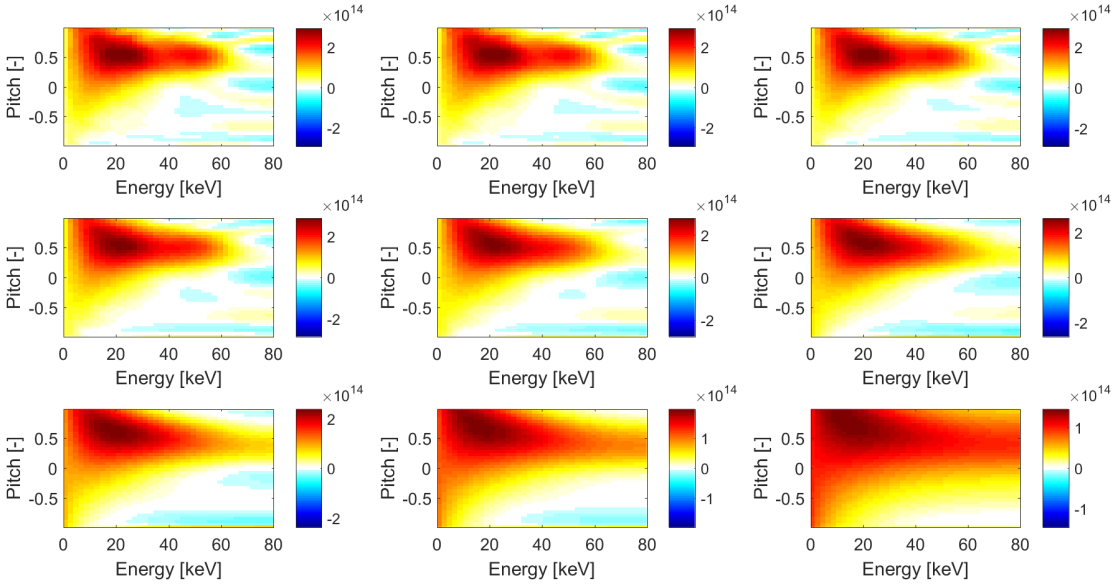


Figure A.4: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1b made using three FIDA-views at $\phi = 10°$, $40°$ and $70°$ with 100 points per view for fast D-ions with $B = 1.74\,\mathrm{T}$ and $\bar{\Gamma} = 0$ under inverse crime conditions. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

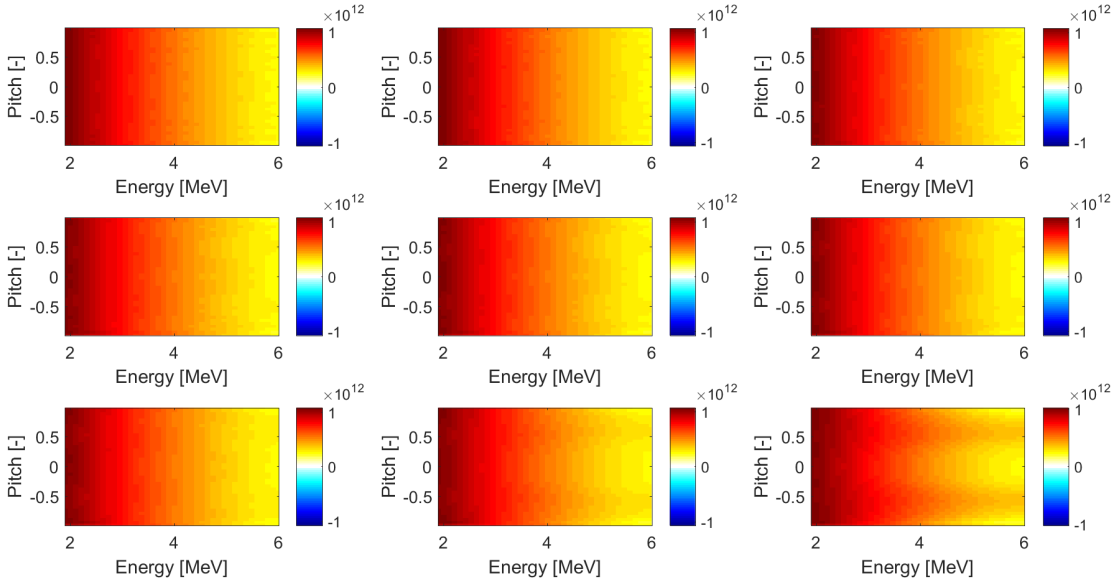


Figure A.5: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1c made using two two-step reaction GRS-views at $\phi = 30°$ and $90°$ with 100 points per view for α-Be fusion with fast α-particles, zero drift and $n_{Be} = 1 \times 10^{18}\,\mathrm{m}^{-3}$ under inverse crime conditions. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

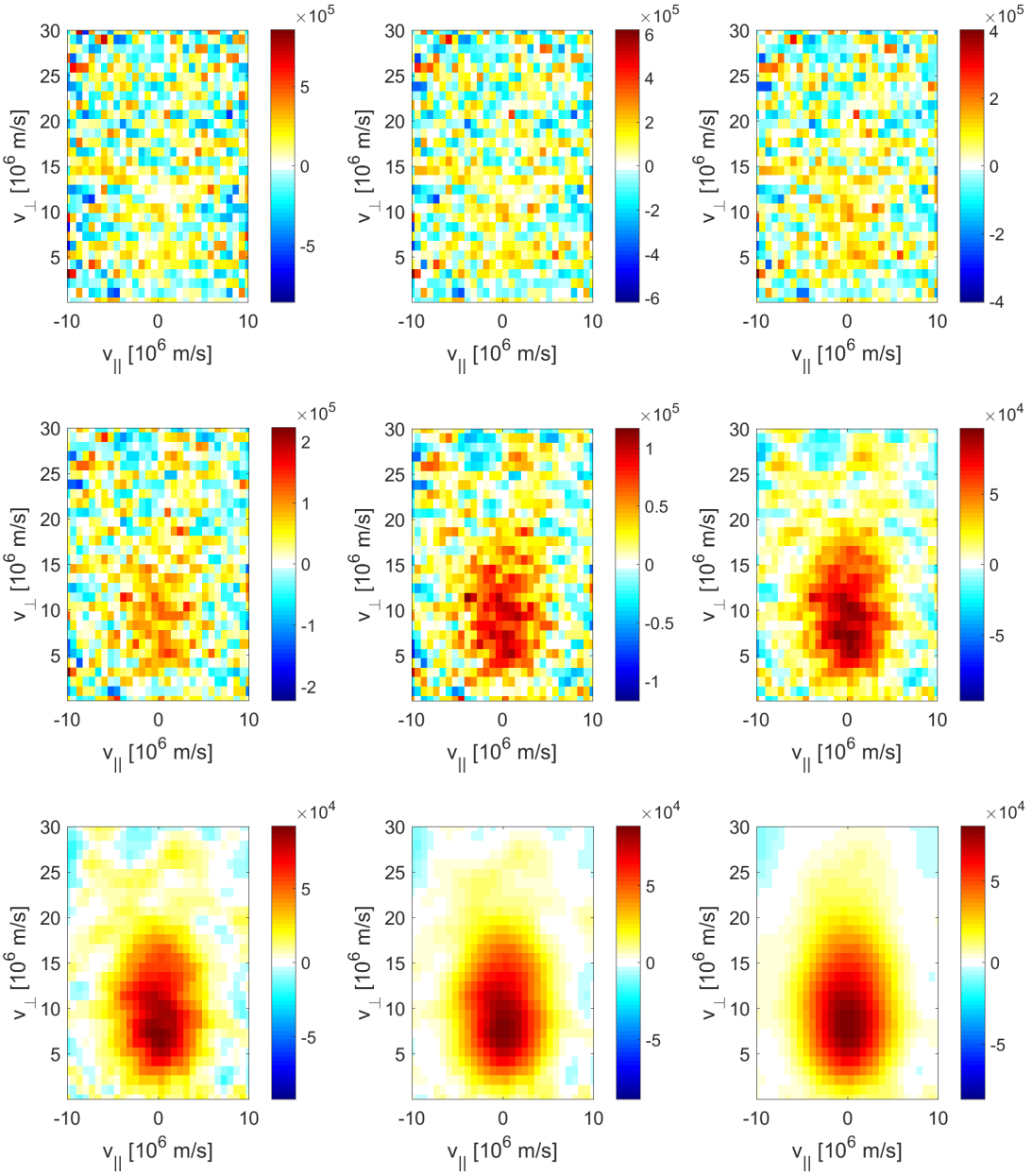


Figure A.6: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1a made using three CTS-views at $\phi = 10°$, $40°$ and $70°$ with 100 points per view for fast D-ions on a 30×30 grid with no gradient cutoff and 3 % noise added to the spectra. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

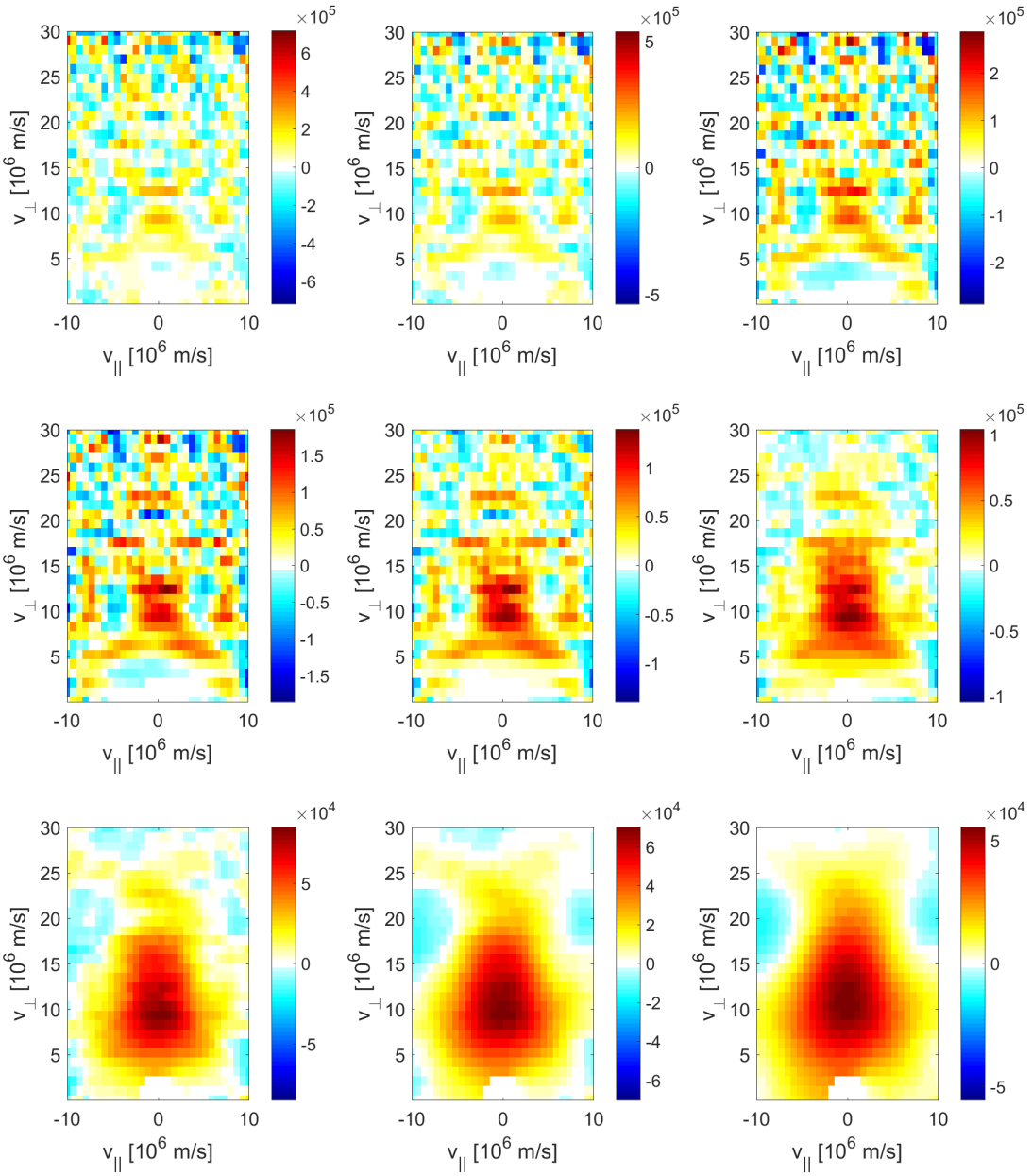


Figure A.7: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1a made using three NES-views at $\phi = 45°$, 90° and 135° with 100 points per view for D-D fusion with zero drift, $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ and a 1 % gradient cutoff on a 30×30 grid with 3 % noise added to the spectra. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

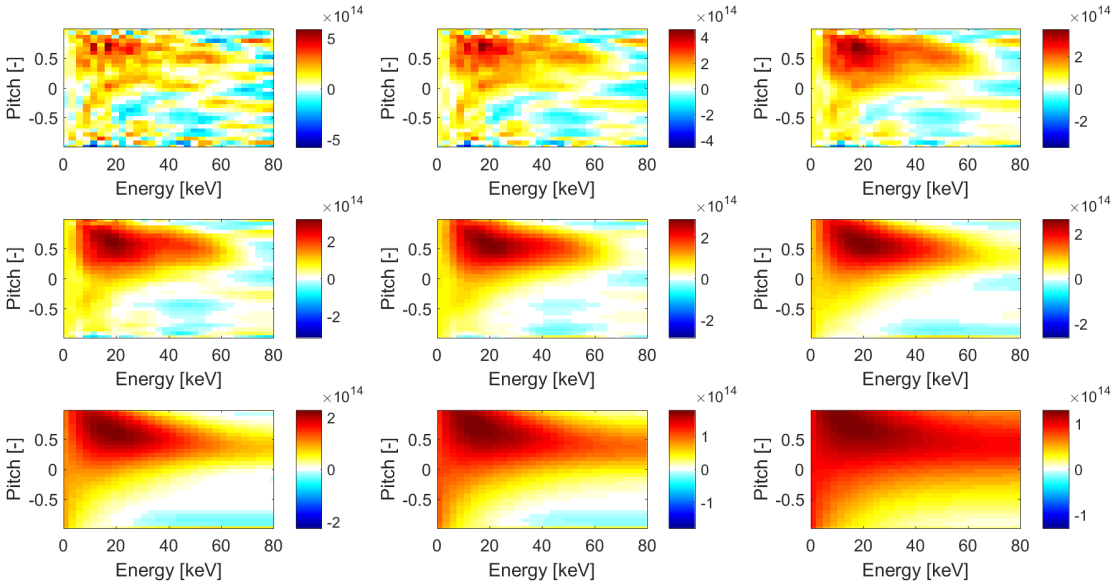


Figure A.8: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1b made using three FIDA-views at $\phi = 10°$, $40°$ and $70°$ with 100 points per view for fast D-ions with $B = 1.74\,\mathrm{T}$ and $\bar{\Gamma} = 0$ on a 30×30 grid with no gradient cutoff and 3 % noise added to the spectra. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

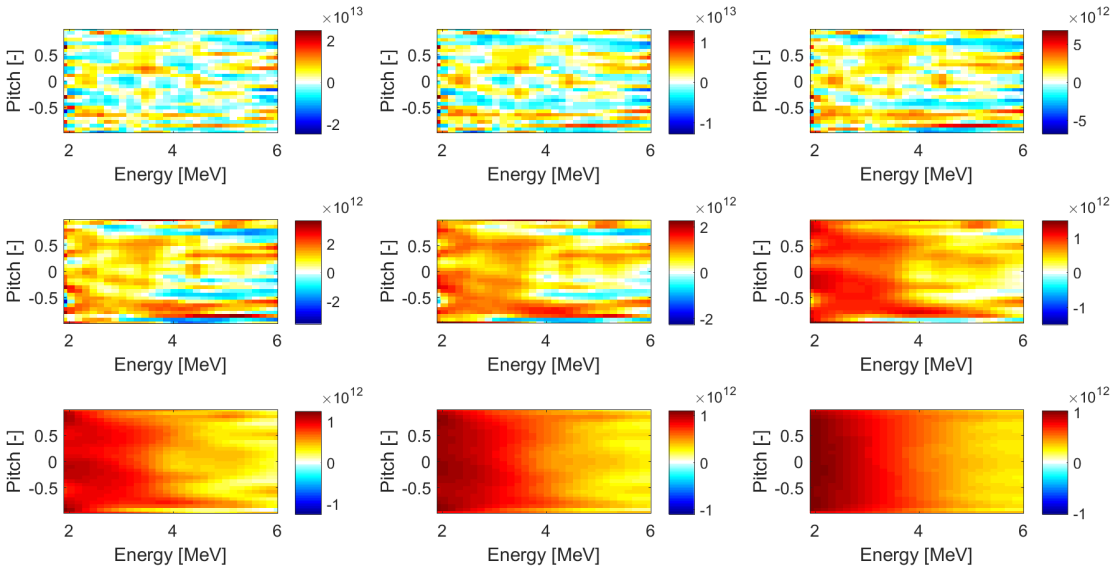


Figure A.9: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1c made using three two-step reaction GRS-views at $\phi = 30°$, $60°$ and $90°$ with 100 points per view for α-Be fusion with fast α-particles, zero drift and $n_{Be} = 1 \times 10^{18}\,\mathrm{m}^{-3}$ on a 30×30 grid with no gradient cutoff and 3 % noise added to the spectra. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

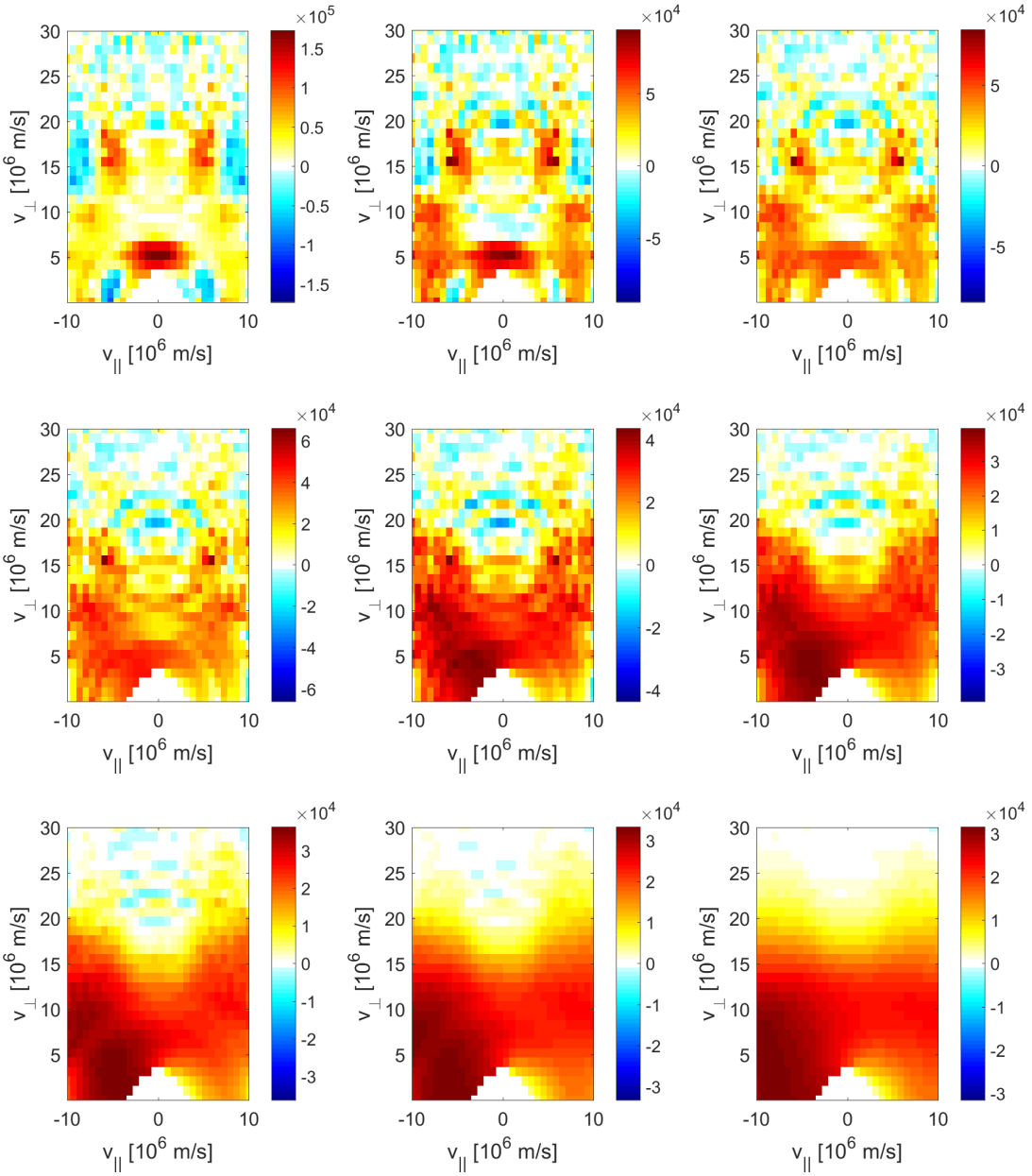


Figure A.10: First-order Tikhonov reconstructions of the distribution function in Fig. 4.1a made using three one-step reaction GRS-views at $\phi = 45°$, 90° and 135° with 100 points per view for p-D fusion with fast p, zero drift, $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ and a 0.1 % gradient cutoff on a 30×30 grid with 3 % noise added to the spectra. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

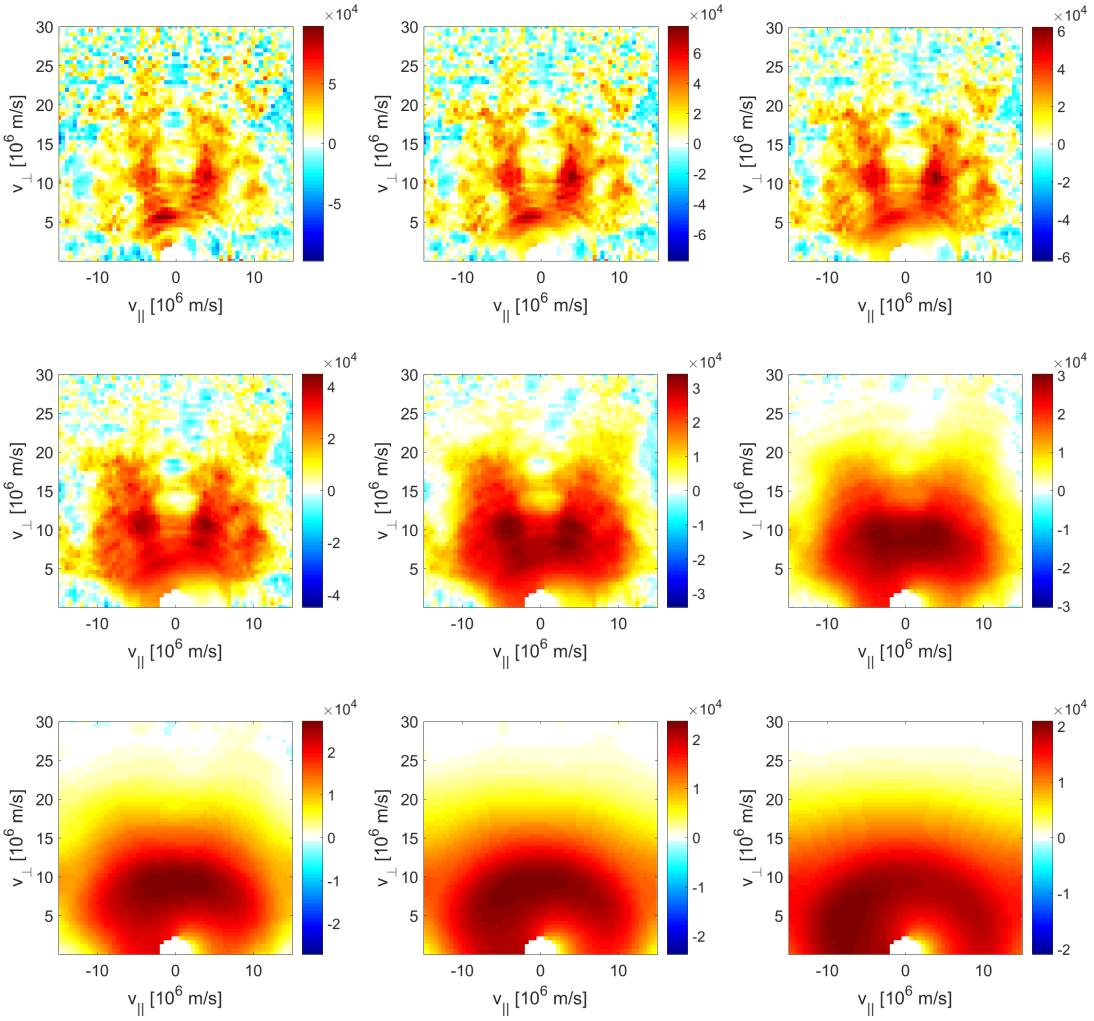


Figure A.11: First-order Tikhonov reconstructions of the distribution function in Fig. 4.9a made using three one-step reaction GRS-views at $\phi = 45°$, $90°$ and $135°$ with 400 points per view for p-D fusion with fast D, zero drift, $n_t = 5 \times 10^{19}\,\mathrm{m}^{-3}$ and a 0.1 % gradient cutoff on a 60×60 grid with 3 % noise added to the spectra. Going from left to right and top to bottom, the values of the regularization parameter are nine logarithmically spaced values between 10^8 and 1×10^{13}.

Appendix B

```
%************************************************************************
% Name:         TomoAnalyticRmix
%
% Purpose:      Contains the function 'TomoAnalyticRmix' which returns
%               reconstructed 2D fast-ion velocity or (energy,pitch)
%               distribution functions and the weight functions used to
%               obtain them from supplied spectra via a tomographic
%               inversion procedure based on Tikhonov regularization. The
%               function is capable of using spectra obtained via
%               Collective Thompson Scattering (CTS), Fast-ion D_alpha
%               spectroscopy (FIDA), Neutron Emission Spectrometry (NES),
%               one-step reaction Gamma-ray spectrometry (1stepGRS),
%               two-step reaction Gamma-ray spectrometry (2stepGRS) or any
%               combination of these.
%               The equations used are explained or derived in the report
%               "Analytic models in velocity-space tomography for fusion
%               plasmas" by Andreas Poulsen.
%               All references refer to this report.
%************************************************************************

function [WTomo,xalpha] = TomoAnalyticRmix(Svec,Spoint,Sdim,Serror,...
    pspacevec1,pspacevec2,vd,phivec,alpha,Tikhonov,uselsqnonneg,...
    grad_cutoff,method,pspace)
%TomoAnalyticRmix Function used for velocity-space tomography
%
% Output parameters
%------------------
% WTomo     - Matrix with each row containing the weight function
%             corresponding to a particular combination of observation
%             angle and spectrum X-value (bin).
%             #rows is equal to length(phivec)*length(Spoint) and #columns
%             is equal to the number of elements in the chosen tomography
%             grid i.e. length(pspacevec1)*length(pspacevec2)
% xalpha    - Matrix with each column containing the reconstructed
%             distribution function obtained for a given value of the
%             regularization parameter alpha.
%             #rows is equal to to the number of elements in the chosen
%             tomography grid i.e. length(pspacevec1)*length(pspacevec2)
%             and #columns is equal to length(alpha)
%
%
% Input parameters
%-----------------
% Svec           - Ordered column vector containing the spectrum Y-values
%                  (signal values) for each observation angle. #values is
%                  equal to sum(Sdim). The values for each observation
%                  angle must be ordered in the same way as the values of
%                  Spoint for that observation angle, and the ordering of
%                  the observation angles must be the same as in phivec.
% Spoint         - Ordered row vector containing the spectrum X-values
%                  (bins) used for every observation angle. The ordering
%                  must be the same as that of the values in Svec.
% Sdim           - Ordered row vector containing the number of measurements
%                  (spectrum values) for each observation angle. The
```

```
%                   ordering must be the same as in phivec. Alternatively,
%                   Sdim may be a single value if the numbr of measurements
%                   was the same for all observation angles
% Serror          - Ordered row vector containing the uncertainties in the
%                   spectrum Y-values. The vector may contain a value for
%                   each observation angle or a value for each value in
%                   Svec. It may also contain a single value.
% pspacevec1      - Ordered row vector containing all of the values of
%                   energy (E) or v_parallel (vpa) to be used for the
%                   tomography grid. [eV] for E and [m/s] for vpa.
% pspacevec2      - Ordered row vector containing all of the values of pitch
%                   (p) or v_perpendicular (vpe) to be used for the
%                   tomography grid. [] for p and [m/s] for vpe.
% vd              - Toroidal drift velocity of the bulk plasma. Only affects
%                   NES, one- and two-step reaction GRS.
% phivec          - Ordered vector containing the used observation
%                   angles in degrees. The values must be ordered in the
%                   same way as in Svec.
% alpha           - Vector containing the desired values of the
%                   regularization parameter to be used for the tomographic
%                   inversion via Tikhonov regularization.
% Tikhonov        - Parameter used to indicate the desired Tikhonov
%                   procedure. The value should be 0 for 0th order Tikhonov,
%                   1 for 1st order Tikhonov or 2 for a mix of 0th and 1st
%                   order Tikhonov.
% uselsgnoneg     - Parameter used to indicate whether or not the output
%                   distribution functions should be constrained to be
%                   non-negative. The value should be 0 for unconstrained
%                   Tikhonov or 1 for non-negative Tikhonov. Note that using
%                   the non-negativity constraint significantly increases
%                   the computation time.
% grad_cutoff     - The gradient used for the Tikhonov reconstruction is
%                   removed from the regions where the weight function
%                   coverage (sum of all weight functions) is less than
%                   grad_cutoff times the mean of the weight function
%                   coverage. This leads to these regions being disregarded
%                   in the reconstructions.
% method          - Optional string array used to indicate the used method
%                   for each diagnostic. Can be CTS, NESDD, NESDT,
%                   GRSDpfastp, GRSDpfastD, FIDA or any combination of
%                   these. #elements must be 1 or equal to length(phivec)
% pspace          - Optional string used to indicate the desired plotting
%                   space for the tomographic inversion. Can be Ep or vv.
%
% Fixed constants
Mp = 1.6726e-27; % [kg] Mass of a proton
Mn = 1.6749e-27; % [kg] Mass of a neutron
Mi = 2*Mp; % [kg] Mass of a deuterium ion
M_He = 3*Mp; % [kg] Mass of a helium-3 particle
n_t = 5e19; % [m^-3] The thermal ion density
Qe = 1.6021917e-19; % [C] The elementary charge
c = 3e8; % [m/s] The speed of light

% List of possible methods
```

```
methodvec=["CTS", "NESDD", "NESDTfastD", "NESDTfastT", "GRSDpfastp",...
    "GRSDpfastD", "FIDA","2stepGRS"];
methods={'CTS','NESDD','NESDTfastD','NESDTfastT','GRSDpfastp',...
    'GRSDpfastD','FIDA','2stepGRS'};

% The function starts by checking whether or not the variables method and
% pspace were specified and give the user the option to select them if they
% were not. In this case, it is assumed that all diagnostics used the
% chosen method. Note that it is possible to specify method and not pspace,
% but it is not possible to specify pspace and not method.

inputval=1;
inputval2=1;
options.Interpreter = 'tex';
options.Default = 'A';
switch nargin % The number of given input parameters is checked
    case {0,1,2,3,4,5,6,7,8,9,10,11}
        disp('Error: Not enough input arguments.')
        inputval=0;
    case 12
        disp('No diagnostic method or plotting space chosen.')
        [indx, tf]=listdlg('PromptString',...
            'Which diagnostic method was used?',...
            'SelectionMode','single', 'ListString',methods);
        if ~tf % If no method is chosen, the function stops
            inputval=0;
        else
            method=methodvec(indx);
            pspaceval=questdlg(...
                'Plot in A) (E,p)- or B) (v_{||},v_{\perp})-space?',...
                'Plotting space Menu', 'A', 'B',options);
            % (E,p)-space is the default choice if one presses enter
            if strcmpi(pspaceval,'')
                % If no plotting space is chosen, the function stops
                inputval=0;
            elseif strcmpi(pspaceval,'A')
                pspace='Ep';
            elseif strcmpi(pspaceval,'B')
                pspace='vv';
            end
        end
    case 13
        disp('No plotting space chosen.')
        pspaceval=questdlg(...
            'Plot in A) (E,p)- or B) (v_{||},v_{\perp})-space?',...
            'Plotting space Menu', 'A', 'B',options);
        % (E,p)-space is the default choice if one presses enter
        if strcmpi(pspaceval,'')
            % If no plotting space is chosen, the function stops
            inputval=0;
        elseif strcmpi(pspaceval,'A')
            pspace='Ep';
        elseif strcmpi(pspaceval,'B')
            pspace='vv';
```

```
        end
    case 14
    otherwise
        disp('Error: Too many input arguments.')
        inputval=0;
end

% The function checks whether or not the specified variables method and
% pspace match any of the possible choices and gives the option to select
% them if they do not.

if inputval
    if sum(strcmpi(method,'CTS'))+ sum(strcmpi(method,'NESDD'))+...
            sum(strcmpi(method,'NESDTfastD'))+...
            sum(strcmpi(method,'NESDTfastT'))+...
            sum(strcmpi(method,'GRSDpfastp'))+...
            sum(strcmpi(method,'GRSDpfastD'))+...
            sum(strcmpi(method,'FIDA'))...
            +sum(strcmpi(method,'2stepGRS'))~=length(method)
        tf=1;
        % The function checks each individual string in the method array
        % and gives the option to select replacements for any invalid ones.
        % If a replacement is not chosen, the function stops.
        for i =1:length(method)
            if ~strcmpi(method(i),methods) && tf==1
                fprintf('Diagnostic nr. %s is not a valid choice.\n',...
                    num2str(i,1))
                [indx, tf]=listdlg('PromptString',...
                    'Which diagnostic method was used?','SelectionMode',...
                    'single', 'ListString',methods);
                method(i)=methodvec(indx);
            end
        end

        if tf==1 && ~strcmpi(pspace,{'Ep','vv'})
            disp('No valid plotting space chosen')
            pspaceval=questdlg(...
                'Plot in A) (E,p)- or B) (v_{||},v_{\perp})-space?',...
                'Plotting space Menu', 'A', 'B',options);
            % (E,p)-space is the default choice
            if strcmpi(pspaceval,'')
                % If no plotting space is chosen, the function stops
                inputval2=0;
            elseif strcmpi(pspaceval,'A')
                pspace='Ep';
            elseif strcmpi(pspaceval,'B')
                pspace='vv';
            end
        elseif tf==0
            % If the invalid methods are not replaced, the function stops
            inputval2=0;
        end
    elseif ~strcmpi(pspace,'Ep') && ~strcmpi(pspace,'vv')
        disp('No valid plotting space chosen.')
```

```
        pspaceval=questdlg(...
            'Plot in A) (E,p)- or B) (v_{||},v_{\perp})-space?',...
            'Plotting space Menu', 'A', 'B',options);
        % (E,p)-space is the default choice
        if strcmpi(pspaceval,'')
            % If no plotting space is chosen, the function stops
            inputval2=0;
        elseif strcmpi(pspaceval,'A')
            pspace='Ep';
        elseif strcmpi(pspaceval,'B')
            pspace='vv';
        end
    end
end

% If both method and pspace are specified properly, the function continues

if inputval && inputval2
    % If the method variable only contains a single string, all diagnostics
    % are assumed to have used that method
    if length(method)==1
        temp=strings(1,length(phivec));
        temp(1:length(phivec))=method;
        method=temp;
    end

    % If Sdim was a single value, it is changed to a vector of length
    % equal to that of phivec where every element has that value
    if length(Sdim)==1
        Sdim=ones(1,length(phivec))*Sdim;
    end

    % If Serror was a single value, it is changed to a vector of length
    % equal to that of Svec where every element has that value
    if length(Serror) == 1
        Serror=Serror.*ones(1,length(Svec));
    % If Serror contained a value for each observation angle, it is changed
    % to a vector of length equal to that of Svec where all
    % elements corresponding to a particular observation angle have the
    % value specified for that observation angle
    elseif length(Serror) == length(phivec)
        temp=ones(1,length(Svec));
        num=0;
        for i =1:length(phivec)
            temp((num+1):(num+Sdim(i)))=temp((num+1):(num+Sdim(i)))*...
                Serror(i);
            num=num+Sdim(i);
        end
        Serror=temp;
    end

    % The grid points in the tomography grid are generated
    [grid1,grid2]=meshgrid(pspacevec1,pspacevec2);
```

```
% The observation angles are converted to radians
phivec=phivec.*pi./180;
Wrow=0;
Sbin=0;
% Pre-allocation of the matrix to hold the weight functions for
% computational efficiency
Wtomo=zeros(length(Svec),numel(grid1));
% The weight functions for each individual diagnostic are calculated
% and stored in the designated matrix
for i=1:length(phivec)
    phi=phivec(i);
    % Parameters relevant for the considered diagnostic method are
    % defined
    if strcmpi(method(i),'CTS')
        methodval = 'CTS';
        M_f =Mi; % [kg] Mass of the fast ion (assumed to be deuterium)
    elseif strcmpi(method(i),'NESDD') % NES for D-D fusion
        methodval = 'NES';
        reaction = 'NESDD';
        M_f =Mi; % [kg] Mass of the fast ion (deuterium)
        % Coefficient used to calculate the fast-ion center-of-mass
        % energy
        CM_coef = 1/2;
        Q = 3.27e6; % [eV] Energy generated in the D-D fusion reaction
    elseif strcmpi(method(i),'NESDTfastD')
        % NES for D-T fusion with fast D
        methodval = 'NES';
        reaction = 'NESDT';
        M_f =Mi; % [kg] Mass of the fast ion (deuterium)
        M_He =4*Mp; % [kg] Mass of an alpha-particle
        % Coefficient used to calculate the fast-ion center-of-mass energy
        CM_coef = 2/5;
        Q = 17.6e6; % [eV] Energy generated in the D-T fusion reaction
    elseif strcmpi(method(i),'NESDTfastT')
        % NES for D-T fusion with fast T
        methodval = 'NES';
        reaction = 'NESDT';
        M_f =3*Mp; % [kg] Mass of the fast ion (tritium)
        M_He =4*Mp; % [kg] Mass of an alpha-particle
        % Coefficient used to calculate the fast-ion center-of-mass
        % energy
        CM_coef = 3/5;
        Q = 17.6e6; % [eV] Energy generated in the D-T fusion reaction
    elseif strcmpi(method(i),'GRSDpfastp')
        % GRS for D-p fusion with fast p
        methodval = 'GRS';
        reaction = 'GRSDp';
        M_f = Mp; % [kg] Mass of the fast ion (proton)
        M_pr = 3*Mp; % [kg] Mass of a helium-3 particle
        % Coefficient used to calculate the fast-ion center-of-mass
        % energy
        CM_coef = 1/3;
        Q = 5.5e6; % [eV] Energy generated in the D-p fusion reaction
    elseif strcmpi(method(i),'GRSDpfastD')
```

```
    % GRS for D-p fusion with fast D
    methodval = 'GRS';
    reaction = 'GRSDp';
    M_f = 2*Mp; % [kg] Mass of the fast ion (deuterium)
    M_pr = 3*Mp; % [kg] Mass of a helium-3 particle
    % Coefficient used to calculate the fast-ion center-of-mass
    % energy
    CM_coef = 2/3;
    Q = 5.5e6; % [eV] Energy generated in the D-p fusion reaction
elseif strcmpi(method(i),'FIDA')
    methodval = 'FIDA';
    M_f = Mi; % [kg] Mass of the fast ion (deuterium)
    gamma_bar=0; % The phase-shift of the cosine model for the pdf
    B=1.74; % [T] The magnetic field
    E0=1e6; % [eV] The free parameter used in the model for the
    % amplitude of the cosine model for the pdf
    lambda0=656.1e-9; % [m] The wavelength of the unshifted D_alpha
    % -line
    C_hat =[1,18,16,1681,2304,729,1936,5490,1936,729,2304,1681,...
        16,18,1]/18860; % Constants related to the probabilities of
    % the various Stark lines
    % Proportionality constants for the Stark splitting wavelength
    % shifts
    sl = [-220.2,-165.2,-137.7,-110.2,-82.64,-55.1,-27.56,0,...
        27.57,55.15,82.74,110.3,138.0,165.6,220.9]*1e-18; % [m^2/V]
    % Signs related to the various Stark lines. + for the
    % sigma-lines and - for the pi-lines
    Stark_sign = [-1,1,1,-1,-1,-1,1,1,1,-1,-1,-1,1,1,-1];
    % Parameters used for the ad-hoc model for the FIDA intensity
    % function R
    ni = 1e8;     % [m^-3] Ion density
    RTpara = 60e3; % [eV] Parallel plasma temperature
    RTperp = 60e3; % [eV] Perpendicular plasma temperature
    Rvparadrift = 0.4e6; % [m/s] Parallel drift velocity
elseif strcmpi(method(i),'2stepGRS')
    methodval= '2stepGRS';
    M_f = 4*Mp; % [kg] Mass of the fast ion (alpha-particle)
    Ega0=4.44e6; % [eV]
    Qdot=1.26e6; % [eV]
    n_Be=1e18; % [m^-3] Density of thermal Be
    betaloop=(90.5:0.5:179.5)/180*pi;
    zetaloop=(0:0.5:179.5)/180*pi;
    Esigma=0:0.1:6.1; % [MeV]
    sigma=[0 0 0 0 0 0 0 0 0 0 ...
        0 0 0 0 0 10 30 100 180 230 ...
        220 200 80 70 60 60 80 100 80 60 ...
        60 60 60 60 60 65 70 75 200 320 ...
        320 320 220 240 260 260 250 280 320 340 ...
        340 320 320 320 320 320 320 320 290 250 ...
        250 0]; % [mb]
end
if strcmpi(methodval,'CTS') && strcmpi(pspace,'Ep')
    ubroadening=3; % Spectral resolution of the measurements
    % divided by the bin width of the spectra
```

```
        % The step sizes in E and p are defined
        dE=pspacevec1(2)-pspacevec1(1);
        dp=pspacevec2(2)-pspacevec2(1);
        % The spectral resolution of the measurements
        dures=ubroadening*(max(Spoint((Sbin+1):(Sbin+Sdim(i))))...
            -min(Spoint((Sbin+1):(Sbin+Sdim(i)))))/(Sdim(i)-1);
        % For each bin in the spectra, the weight function for the
        % chosen tomography grid is calculated using eq. 2.19
        % and stored as a row in a matrix
        for j=(Sbin+1):(Sbin+Sdim(i))
            u=Spoint(j);
            Wrow=Wrow+1;
            % The two gyro-angles (eq. 2.20)
            gamma1 = acos(1./sin(phi).*1./sqrt(1-grid2.^2).*...
                ((u-dures./2)./sqrt(2.*grid1./Mi*Qe)-grid2.*cos(phi)));
            gamma2 = acos(1./sin(phi).*1./sqrt(1-grid2.^2).*...
                ((u+dures./2)./sqrt(2.*grid1./Mi*Qe)-grid2.*cos(phi)));
            % The weight function (eq. 2.19) times the step sizes
            wfcEpCTS=real((gamma1-gamma2))./pi./dures.*dE.*dp;
            % The weight function is normalized by the associated
            % uncertainty
            Wtomo(Wrow,:)=reshape(wfcEpCTS,1,numel(grid1))./...
                Serror(Wrow);
        end
    elseif strcmpi(methodval,'CTS') && strcmpi(pspace,'vv')
        ubroadening=3; % Spectral resolution of the measurements
        % divided by the bin width of the spectra
        % The step sizes in vpa and vpe are defined
        dvpa=pspacevec1(2)-pspacevec1(1);
        dvpe=pspacevec2(2)-pspacevec2(1);
        % The spectral resolution of the measurements
        dures=ubroadening*(max(Spoint((Sbin+1):(Sbin+Sdim(i))))...
            -min(Spoint((Sbin+1):(Sbin+Sdim(i)))))/(Sdim(i)-1);
        % For each bin in the spectra, the weight function for the
        % chosen tomography grid is calculated using eq. 2.19
        % and stored as a row in a matrix
        for j=(Sbin+1):(Sbin+Sdim(i))
            u=Spoint(j);
            Wrow=Wrow+1;
            % The two gyro-angles (eq. 2.20)
            gamma1=acos(1./(sin(phi).*grid2).*(u-dures./2-grid1.*...
                cos(phi)));
            gamma2=acos(1./(sin(phi).*grid2).*(u+dures./2-grid1.*...
                cos(phi)));
            % The weight function (eq. 2.19) times the step sizes
            wfcvvCTS=real((gamma1-gamma2))./pi./dures.*dvpa.*dvpe;
            % The weight function is normalized by the associated
            % uncertainty
            Wtomo(Wrow,:)=reshape(wfcvvCTS,1,numel(grid1))./...
                Serror(Wrow);
        end
    elseif strcmpi(methodval,'NES') && strcmpi(pspace,'Ep')
        Enbroadening=1; % Spectral resolution of the measurements
        % divided by the bin width of the spectra
```

```
        % The step sizes in E and p are defined
        dE=pspacevec1(2)-pspacevec1(1);
        dp=pspacevec2(2)-pspacevec2(1);
        % The spectral resolution of the measurements
        dEnres=Enbroadening*(max(Spoint((Sbin+1):(Sbin+Sdim(i))))...
            -min(Spoint((Sbin+1):(Sbin+Sdim(i))))/(Sdim(i)-1);
        % The values of vpa and vpe corresponding to the chosen
        % (E,p)-grid are calculated (eq. 2.3)
        vpara=grid2.*sqrt(2.*grid1.*Qe./M_f);
        vperp=sqrt(1-grid2.^2).*sqrt(2.*grid1.*Qe./M_f);
        % The center-of-mass energy of the fast ion in [keV]
        E_CM=CM_coef.*M_f.*((vpara-vd).^2+vperp.^2)./Qe/2e3;
        % The rate part of the weight function (eq. 2.51)
        RNES=n_t.*Bosch_Hale_sigma(E_CM,reaction).*10^-31.*...
        sqrt((vpara-vd).^2+vperp.^2);
        RNES(RNES<0)=0; % Negative rates are set to zero
        % For each bin in the spectra, the weight function for the
        % chosen tomography grid is calculated using eq. 2.44
        % and stored as a row in a matrix
        for j=(Sbin+1):(Sbin+Sdim(i))
            En=Spoint(j);
            Wrow=Wrow+1;
            % The two gyro-angles (eq. 2.48)
            gamma1=acos(1./(sqrt(1-grid2.^2).*sin(phi)).*(...
                1./2.*(M_He+Mn)./sqrt(M_f.*Mn).*sqrt((En-...
                dEnres./2)./grid1)-1./2.*(M_He-M_f)./sqrt(M_f.*Mn).*...
                sqrt(grid1./(En-dEnres./2))-1./2.*M_He./sqrt(M_f.*...
                Mn).*Q./(sqrt(grid1.*(En-dEnres./2)))-...
                grid2.*cos(phi)));
            gamma2=acos(1./(sqrt(1-grid2.^2).*sin(phi)).*(...
                1./2.*(M_He+Mn)./sqrt(M_f.*Mn).*sqrt((En+...
                dEnres./2)./grid1)-1./2.*(M_He-M_f)./sqrt(M_f.*Mn).*...
                sqrt(grid1./(En+dEnres./2))-1./2.*M_He./sqrt(M_f.*...
                Mn).*Q./(sqrt(grid1.*(En+dEnres./2)))-...
                grid2.*cos(phi)));
            % The probability part (eq. 2.45)
            probNES=real((gamma1-gamma2)./pi);
            % The total weight function (eq. 2.44) times the step sizes
            wfcEpNES=probNES.*RNES./dEnres.*dE.*dp;
            % The weight function is normalized by the associated
            % uncertainty
            Wtomo(Wrow,:)=reshape(wfcEpNES,1,numel(grid1))./...
                Serror(Wrow);
        end
    elseif strcmpi(methodval,'NES') && strcmpi(pspace,'vv')
        Enbroadening=1; % Spectral resolution of the measurements
        % divided by the bin width of the spectra
        % The step sizes in vpa and vpe are defined
        dvpa=pspacevec1(2)-pspacevec1(1);
        dvpe=pspacevec2(2)-pspacevec2(1);
        % The spectral resolution of the measurements
        dEnres=Enbroadening*(max(Spoint((Sbin+1):(Sbin+Sdim(i))))...
            -min(Spoint((Sbin+1):(Sbin+Sdim(i))))/(Sdim(i)-1);
        % The center-of-mass energy of the fast ion in [keV]
```

```
    E_CM=CM_coef.*M_f.*((grid1-vd).^2+grid2.^2)./Qe/2e3;
    % The rate part of the weight function (eq. 2.51)
    RNES=n_t.*Bosch_Hale_sigma(E_CM,reaction).*10^-31.*...
        sqrt((grid1-vd).^2+grid2.^2);
    RNES(RNES<0)=0; % Negative rate are set to zero
    % For each bin in the spectra, the weight function for the
    % chosen tomography grid is calculated using eq. 2.44
    % and stored as a row in a matrix
    for j=(Sbin+1):(Sbin+Sdim(i))
        En=Spoint(j);
        Wrow=Wrow+1;
        % The two gyro-angles (eq. 2.47)
        gamma1=acos(1./(grid2.*sin(phi)).*(0.5.*(M_He+Mn)./M_f.*...
            sqrt(2.*(En-dEnres./2)./Mn.*Qe)-0.5.*(M_He-...
            M_f)./Mn.*(grid1.^2+grid2.^2)./sqrt(2.*(En-...
            dEnres./2)./Mn.*Qe)-M_He./(M_f.*Mn).*Q.*Qe./sqrt(2.*...
            (En-dEnres./2)./Mn.*Qe)-grid1.*cos(phi)));
        gamma2=acos(1./(grid2.*sin(phi)).*(0.5.*(M_He+Mn)./M_f.*...
            sqrt(2.*(En+dEnres./2)./Mn.*Qe)-0.5.*(M_He-...
            M_f)./Mn.*(grid1.^2+grid2.^2)./sqrt(2.*(En+...
            dEnres./2)./Mn.*Qe)-M_He./(M_f.*Mn).*Q.*Qe./sqrt(2.*...
            (En+dEnres./2)./Mn.*Qe)-grid1.*cos(phi)));
        % The probability part (eq. 2.45)
        probNES=real((gamma1-gamma2))./pi;
        % The total weight function (eq. 2.44) times the step sizes
        wfcvvNES=probNES.*RNES./dEnres.*dvpa.*dvpe;
        % The weight function is normalized by the associated
        % uncertainty
        Wtomo(Wrow,:)=reshape(wfcvvNES,1,numel(grid1))./...
            Serror(Wrow);
    end
elseif strcmpi(methodval,'GRS') && strcmpi(pspace,'Ep')
    Ebroadening=1; % Spectral resolution of the measurements
    % divided by the bin width of the spectra
    % The step sizes in E and p are defined
    dE=pspacevec1(2)-pspacevec1(1);
    dp=pspacevec2(2)-pspacevec2(1);
    % The spectral resolution of the measurements
    dEgares=Ebroadening*(max(Spoint((Sbin+1):(Sbin+Sdim(i)))))...
        -min(Spoint((Sbin+1):(Sbin+Sdim(i)))))/(Sdim(i)-1);
    % The values of vpa and vpe corresponding to the chosen
    % (E,p)-grid are calculated (eq. 2.3)
    vpara=grid2.*sqrt(2.*grid1.*Qe./M_f);
    vperp=sqrt(1-grid2.^2).*sqrt(2.*grid1.*Qe./M_f);
    % The center-of-mass energy of the fast ion in [keV]
    E_CM=CM_coef.*M_f.*((vpara-vd).^2+vperp.^2)./Qe/2e3;
    % The rate part of the weight function (eq. 2.51)
    RGRS=n_t.*Bosch_Hale_sigma(E_CM,reaction).*10^-31.*...
    sqrt((vpara-vd).^2+vperp.^2);
    RGRS(RGRS<0)=0; % Negative rates are set to zero
    % For each bin in the spectra, the weight function for the
    % chosen tomography grid is calculated using eq. 2.78
    % and stored as a row in a matrix
    for j=(Sbin+1):(Sbin+Sdim(i))
```

```
            Ega=Spoint(j);
            Wrow=Wrow+1;
            % The two gyro-angles (eq. 2.82)
            gamma1=acos(1./(vperp.*sin(phi)).*((M_f-M_pr).*c./...
                (2.*(Ega-dEgares./2).*Qe).*(vpara.^2+vperp.^2)+...
                (Ega-dEgares./2).*Qe./(2.*M_f.*c)+M_pr.*c.*...
                (Ega-dEgares./2-Q)./(M_f.*(Ega-dEgares./2))-vpara.*...
                cos(phi)));
            gamma2=acos(1./(vperp.*sin(phi)).*((M_f-M_pr).*c./...
                (2.*(Ega+dEgares./2).*Qe).*(vpara.^2+vperp.^2)+...
                (Ega+dEgares./2).*Qe./(2.*M_f.*c)+M_pr.*c.*...
                (Ega+dEgares./2-Q)./(M_f.*(Ega+dEgares./2))-vpara.*...
                cos(phi)));
            % The probability part (eq. 2.79)
            probGRS=real(gamma1-gamma2)./pi;
            % The total weight function (eq. 2.78) times the step sizes
            wfcEpGRS=probGRS.*RGRS./dEgares.*dE.*dp;
            % The weight function is normalized by the associated
            % uncertainty
            Wtomo(Wrow,:)=reshape(wfcEpGRS,1,numel(grid1))./...
                Serror(Wrow);
        end
    elseif strcmpi(methodval,'GRS') && strcmpi(pspace,'vv')
        Ebroadening=1; % Spectral resolution of the measurements
        % divided by the bin width of the spectra
        % The step sizes in vpa and vpe are defined
        dvpa=pspacevec1(2)-pspacevec1(1);
        dvpe=pspacevec2(2)-pspacevec2(1);
        % The spectral resolution of the measurements
        dEgares=Ebroadening*(max(Spoint((Sbin+1):(Sbin+Sdim(i))))...
            -min(Spoint((Sbin+1):(Sbin+Sdim(i))))/(Sdim(i)-1);
        % The center-of-mass energy of the fast ion in [keV]
        E_CM=CM_coef.*M_f.*((grid1-vd).^2+grid2.^2)./Qe/2e3;
        % The rate part of the weight function (eq. 2.51)
        RGRS=n_t.*Bosch_Hale_sigma(E_CM,reaction).*10^-31.*...
            sqrt((grid1-vd).^2+grid2.^2);
        RGRS(RGRS<0)=0; % Negative rates are set to zero
        % For each bin in the spectra, the weight function for the
        % chosen tomography grid is calculated using eq. 2.78
        % and stored as a row in a matrix
        for j=(Sbin+1):(Sbin+Sdim(i))
            Ega=Spoint(j);
            Wrow=Wrow+1;
            % The two gyro-angles (eq. 2.81)
            gamma1=acos(1./(grid2.*sin(phi)).*((M_f-M_pr).*c./...
                (2.*(Ega-dEgares./2).*Qe).*(grid1.^2+grid2.^2)+...
                (Ega-dEgares./2).*Qe./(2.*M_f.*c)+M_pr.*c.*...
                (Ega-dEgares./2-Q)./(M_f.*(Ega-dEgares./2))-grid1.*...
                cos(phi)));
            gamma2=acos(1./(grid2.*sin(phi)).*((M_f-M_pr).*c./...
                (2.*(Ega+dEgares./2).*Qe).*(grid1.^2+grid2.^2)+...
                (Ega+dEgares./2).*Qe./(2.*M_f.*c)+M_pr.*c.*...
                (Ega+dEgares./2-Q)./(M_f.*(Ega+dEgares./2))-grid1.*...
                cos(phi)));
```

```
        % The probability part of the weight function (eq. 2.79)
        probGRS=real(gamma1-gamma2)./pi;
        % The total weight function (eq. 2.78) times the step sizes
        wfcvvGRS=probGRS.*RGRS./dEgares.*dvpa.*dvpe;
        % The weight function is normalized by the associated
        % uncertainty
        Wtomo(Wrow,:)=reshape(wfcvvGRS,1,numel(grid1))./...
            Serror(Wrow);
    end
elseif strcmpi(methodval,'FIDA') && strcmpi(pspace,'Ep')
    lambdabroadening=1; % Spectral resolution of the measurements
    % divided by the bin width of the spectra
    % The step sizes in E and p are defined
    dE=pspacevec1(2)-pspacevec1(1);
    dp=pspacevec2(2)-pspacevec2(1);
    % The spectral resolution of the measurements
    dlambdares=lambdabroadening*(max(Spoint((Sbin+1):(Sbin+...
        Sdim(i))))-min(Spoint((Sbin+1):(Sbin+Sdim(i)))))/...
        (Sdim(i)-1);
    % The ad-hoc model for the FIDA intensity function R
    RFIDA = ni.*sqrt(grid1./(pi.*RTpara.*RTperp.^2.*Qe.^2))...
        .*exp(-(grid2.^2.*grid1.*Qe+0.5.*Mi.*Rvparadrift.^2-...
        Rvparadrift.*grid2.*sqrt(2.*Mi.*grid1.*Qe))/...
        (RTpara.*Qe)-(1-grid2.^2).*grid1./RTperp).*Qe;
    % The values of vpa and vpe corresponding to the chosen
    % (E,p)-grid are calculated (eq. 2.3)
    vpara=grid2.*sqrt(2.*grid1.*Qe./Mi);
    vperp=sqrt(1-grid2.^2).*sqrt(2.*grid1.*Qe./Mi);
    % For each bin in the spectra, the weight function for the
    % chosen tomography grid is calculated using eq. 2.26
    % and stored as a row in a matrix
    for j=(Sbin+1):(Sbin+Sdim(i))
        lambda=Spoint(j);
        Wrow=Wrow+1;
        probFIDA=zeros(size(grid1));
        for l=1:15 % Sum over the 15 Stark lines
            % The two gyro-angles (eq. 2.31)
            gamma1=acos(1./(vperp.*sin(phi)).*(c.*((lambda-...
                dlambdares./2)./(lambda0+sl(l).*vperp.*B)-1)-...
                vpara.*cos(phi)));
            gamma2=acos(1./(vperp.*sin(phi)).*(c.*((lambda+...
                dlambdares./2)./(lambda0+sl(l).*vperp.*B)-1)-...
                vpara.*cos(phi)));
            %The probability part of the weight function (eq. 2.37)
            probFIDA=C_hat(l).*(real(gamma1-gamma2)/pi+...
                Stark_sign(l).*sin(phi)^2/2.*(real(gamma1-...
                gamma2)/pi-(sin(2.*real(gamma1))-sin(2.*...
                real(gamma2)))/(2*pi))+2.*grid1./E0.*...
                (1-grid2.^2).*cos(gamma_bar).*...
                (sin(real(gamma1))-sin(real(gamma2))+...
                Stark_sign(l).*sin(phi)^2/3.*...
                (sin(real(gamma1)).^3-sin(real(gamma2)).^3)))+...
                probFIDA;
        end
```

```
        % The total weight function (eq. 2.26) times the step sizes
        wfcEpFIDA=probFIDA.*RFIDA.*dE.*dp./dlambdares;
        % The weight function is normalized by the associated
        % uncertainty
        Wtomo(Wrow,:)=reshape(wfcEpFIDA,1,numel(grid1))./...
            Serror(Wrow);
    end
elseif strcmpi(methodval,'FIDA') && strcmpi(pspace,'vv')
    lambdabroadening=1; % Spectral resolution of the measurements
    % divided by the bin width of the spectra
    % The step sizes in vpa and vpe are defined
    dvpa=pspacevec1(2)-pspacevec1(1);
    dvpe=pspacevec2(2)-pspacevec2(1);
    % The spectral resolution of the measurements
    dlambdares=lambdabroadening*(max(Spoint((Sbin+1):(Sbin+...
        Sdim(i))))-min(Spoint((Sbin+1):(Sbin+Sdim(i)))))/...
        (Sdim(i)-1);
    % The ad-hoc model for the FIDA intensity function R
    vthpara=sqrt(2*RTpara*Qe/Mi);
    vthperp=sqrt(2*RTperp*Qe/Mi);
    RFIDA = ni./(pi.^(3/2).*vthpara.*vthperp.^2).*...
        exp(-((grid1-Rvparadrift)./vthpara).^2-...
        (grid2./vthperp).^2).*2.*pi.*grid2;
    % The values of E and p corresponding to the chosen
    % (vpa,vpe)-grid are calculated (eq. 2.2)
    E = 1/2.*Mi.*(grid1.^2+grid2.^2)./Qe;
    p = grid1./sqrt(grid1.^2+grid2.^2);
    % For each bin in the spectra, the weight function for the
    % chosen tomography grid is calculated using eq. 2.26
    % and stored as a row in a matrix
    for j=(Sbin+1):(Sbin+Sdim(i))
        lambda=Spoint(j);
        Wrow=Wrow+1;
        probFIDA=zeros(size(grid1));
        for l=1:15 % Sum over the 15 Stark lines
            % The two gyro-angles (eq. 2.31)
            gamma1=acos(1./(grid2.*sin(phi)).*(c.*((lambda-...
                dlambdares./2)./(lambda0+sl(l).*grid2.*B)-1)-...
                grid1.*cos(phi)));
            gamma2=acos(1./(grid2.*sin(phi)).*(c.*((lambda+...
                dlambdares./2)./(lambda0+sl(l).*grid2.*B)-1)-...
                grid1.*cos(phi)));
            %The probability part of the weight function (eq. 2.37)
            probFIDA=C_hat(l).*(real(gamma1-gamma2)/pi+...
                Stark_sign(l).*sin(phi)^2/2.*(real(gamma1-...
                gamma2)/pi-(sin(2.*real(gamma1))-...
                sin(2.*real(gamma2)))/(2*pi))+2.*E./E0.*...
                (1-p.^2).*cos(gamma_bar).*(sin(real(gamma1))-...
                sin(real(gamma2))+Stark_sign(l).*sin(phi)^2/3.*...
                (sin(real(gamma1)).^3-sin(real(gamma2)).^3)))+...
                probFIDA;
        end
        % The total weight function (eq. 2.26) times the step sizes
        wfcvvFIDA=probFIDA.*RFIDA.*dvpa.*dvpe./dlambdares;
```

```
        % The weight function is normalized by the associated
        % uncertainty
        Wtomo(Wrow,:)=reshape(wfcvvFIDA,1,numel(grid1))./...
            Serror(Wrow);
    end
elseif strcmpi(methodval,'2stepGRS') && strcmpi(pspace,'Ep')
    Ebroadening=1; % Spectral resolution of the measurements
    % divided by the bin width of the spectra
    % The step sizes in E and p are defined
    dE=pspacevec1(2)-pspacevec1(1);
    dp=pspacevec2(2)-pspacevec2(1);
    % The spectral resolution of the measurements
    dEgares=Ebroadening*(max(Spoint((Sbin+1):(Sbin+Sdim(i))))...
        -min(Spoint((Sbin+1):(Sbin+Sdim(i)))))/(Sdim(i)-1);
    % The values of vpa, vpe and vtot corresponding to the chosen
    % (E,p)-grid are calculated (eq. 2.3)
    vpara=grid2.*sqrt(2.*grid1.*Qe./(4*Mn));
    vperp=sqrt(1-grid2.^2).*sqrt(2.*grid1.*Qe./(4*Mn));
    vtotforwardgrid=sqrt(2.*grid1.*Qe./(4*Mn));
    % The rate part of the weight function
    SIGMA=zeros(size(grid1));
    for k =1:length(pspacevec2)
        for j=1:length(pspacevec1)
            [~,index]=min(abs(Esigma-grid1(k,j)/1e6));
            SIGMA(k,j)=sigma(index);
        end
    end
    SIGMAs=medfilt2(SIGMA);
    RGRS=n_Be*SIGMAs.*sqrt((vpara-vd).^2+vperp.^2)*10^(-31);
    % For each bin in the spectra, the weight function for the
    % chosen tomography grid is calculated using eq. 2.88
    % and stored as a row in a matrix
    for j=(Sbin+1):(Sbin+Sdim(i))
        Ega=Spoint(j);
        Wrow=Wrow+1;
        prob2stepGRS=zeros(size(vtotforwardgrid));
        % The integrals are calculated as sums
        for beta=betaloop
            for zeta=zetaloop
                % The two values of u_C (eq. 2.90)
                uC1=((Ega-dEgares/2)/Ega0-1)*c;
                uC2=((Ega+dEgares/2)/Ega0-1)*c;

                % The radicands of eq. 2.91 for the two u_C-values
                radicand1=(sin(beta)^2*cos(zeta)^2*(64*uC1^2*...
                    vtotforwardgrid.^2*cos(beta)^2*(cos(beta)^2+...
                    sin(beta)^2*cos(zeta)^2)-(cos(beta)^2*...
                    (Qdot*Qe/(6*Mn)-vtotforwardgrid.^2)-13*...
                    uC1^2).^2))/(64*uC1^2*cos(beta)^2*...
                    (cos(beta)^2+sin(beta)^2*cos(zeta)^2)^2);
                radicand2=(sin(beta)^2*cos(zeta)^2*(64*uC2^2*...
                    vtotforwardgrid.^2*cos(beta)^2*(cos(beta)^2+...
                    sin(beta)^2*cos(zeta)^2)-(cos(beta)^2*...
                    (Qdot*Qe/(6*Mn)-vtotforwardgrid.^2)-13*...
```

```
    uC2^2).^2))/(64*uC2^2*cos(beta)^2*...
    (cos(beta)^2+sin(beta)^2*cos(zeta)^2)^2);

% Check for and exclude negative values of the
% radicands
r1positive=ones(size(radicand1));
r1positive(radicand1<0)=0;
r2positive=ones(size(radicand2));
r2positive(radicand2<0)=0;

% The four possible values of u_alpha (eq. 2.91)
ua1plus=real((13*uC1^2+cos(beta)^2*...
    (vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*...
    uC1*(cos(beta)^2+sin(beta)^2*cos(zeta)^2))+...
    sqrt(radicand1));
ua2plus=real((13*uC2^2+cos(beta)^2*...
    (vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*...
    uC2*(cos(beta)^2+sin(beta)^2*cos(zeta)^2))+...
    sqrt(radicand2));
ua1minus=real((13*uC1^2+cos(beta)^2*...
    (vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*...
    uC1*(cos(beta)^2+sin(beta)^2*cos(zeta)^2))-...
    sqrt(radicand1));
ua2minus=real((13*uC2^2+cos(beta)^2*...
    (vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*...
    uC2*(cos(beta)^2+sin(beta)^2*cos(zeta)^2))-...
    sqrt(radicand2));

% Check for and exclude solutions that do not keep
% eq. 2.89
check1plus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-...
    13*uC1^2/cos(beta)^2+8*(ua1plus*uC1+...
    cos(zeta)*sqrt((vtotforwardgrid.^2-...
    ua1plus.^2).*uC1^2*tan(beta)^2));
check1minus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-...
    13*uC1^2/cos(beta)^2+8*(ua1minus*uC1+...
    cos(zeta)*sqrt((vtotforwardgrid.^2-...
    ua1minus.^2).*uC1^2*tan(beta)^2));
check2plus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-...
    13*uC2^2/cos(beta)^2+8*(ua2plus*uC2+...
    cos(zeta)*sqrt((vtotforwardgrid.^2-...
    ua2plus.^2).*uC2^2*tan(beta)^2));
check2minus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-...
    13*uC2^2/cos(beta)^2+8*(ua2minus*uC2+...
    cos(zeta)*sqrt((vtotforwardgrid.^2-...
    ua2minus.^2).*uC2^2*tan(beta)^2));
% We only exclude solutions that do not keep eq.
% 2.89 by more than 9 m^2/s^2
checkzero1plus=ones(size(check1plus));
checkzero1plus(log10(abs(check1plus)+1)>1)=0;
checkzero1minus=ones(size(check1minus));
checkzero1minus(log10(abs(check1minus)+1)>1)=0;
checkzero2plus=ones(size(check2plus));
checkzero2plus(log10(abs(check2plus)+1)>1)=0;
```

```
                checkzero2minus=ones(size(check2minus));
                checkzero2minus(log10(abs(check2minus)+1)>1)=0;

                % The checks are combined into acceptance matrices
                % that define the grid points that are acceptable.
                % The other grid points are set to zero.
                acceptplus=r1positive.*r2positive.*...
                    checkzero1plus.*checkzero2plus;
                acceptminus=r1positive.*r2positive.*...
                    checkzero1minus.*checkzero2minus;

                % The four gyro-angles (eq. 2.16)
                gamma1plus=acos((ua1plus-vpara*cos(phi))./...
                    (vperp*sin(phi)));
                gamma2plus=acos((ua2plus-vpara*cos(phi))./...
                    (vperp*sin(phi)));
                gamma1minus=acos((ua1minus-vpara*cos(phi))./...
                    (vperp*sin(phi)));
                gamma2minus=acos((ua2minus-vpara*cos(phi))./...
                    (vperp*sin(phi)));
                % The probability part of the weight function (eq.
                % 2.92) is checked using the acceptance matrices
                prob=abs(real(gamma2plus-gamma1plus))/pi.*...
                    acceptplus+abs(real(gamma2minus-...
                    gamma1minus))/pi.*acceptminus;
                prob2stepGRS=prob2stepGRS+prob;
            end
        end
        prob2stepGRS=prob2stepGRS/(length(betaloop)*...
            length(zetaloop));
        % The total weight function (eq. 2.88) times the step sizes
        wfcEp2stepGRSforward=RGRS.*prob2stepGRS/dEgares*dE*dp;
        Wtomo(Wrow,:)=reshape(wfcEp2stepGRSforward,1,...
            numel(grid1))./Serror(Wrow);
    end
elseif strcmpi(methodval,'2stepGRS') && strcmpi(pspace,'vv')
    Ebroadening=1; % Spectral resolution of the measurements
    % divided by the bin width of the spectra
    % The step sizes in vpa and vpe are defined
    dvpa=pspacevec1(2)-pspacevec1(1);
    dvpe=pspacevec2(2)-pspacevec2(1);
    % The spectral resolution of the measurements
    dEgares=Ebroadening*(max(Spoint((Sbin+1):(Sbin+Sdim(i))))...
        -min(Spoint((Sbin+1):(Sbin+Sdim(i)))))/(Sdim(i)-1);
    % The total velocities and alpha-particle energies are
    % calculated (eq. 2.2)
    vtotforwardgrid=sqrt(grid1.^2+grid2.^2);
    Ealpha=2*Mn*vtotforwardgrid.^2/Qe;
    % The rate part of the weight function
    SIGMA=zeros(size(grid1));
    for k =1:length(pspacevec2)
        for j=1:length(pspacevec1)
            [~,index]=min(abs(Esigma-Ealpha(k,j)/1e6));
            SIGMA(k,j)=sigma(index);
```

```
    end
end
SIGMAs=medfilt2(SIGMA);
RGRS=n_Be*SIGMAs.*sqrt((grid1-vd).^2+grid2.^2)*10^(-31);
% For each bin in the spectra, the weight function for the
% chosen tomography grid is calculated using eq. 2.88
% and stored as a row in a matrix
for j=(Sbin+1):(Sbin+Sdim(i))
    Ega=Spoint(j);
    Wrow=Wrow+1;
    prob2stepGRS=zeros(size(vtotforwardgrid));
    % The integrals are calculated as sums
    for beta=betaloop
        for zeta=zetaloop
            % The two values of u_C (eq. 2.90)
            uC1=((Ega-dEgares/2)/Ega0-1)*c;
            uC2=((Ega+dEgares/2)/Ega0-1)*c;

            % The radicands of eq. 2.91 for the two u_C-values
            radicand1=(sin(beta)^2*cos(zeta)^2*(64*uC1^2*...
                vtotforwardgrid.^2*cos(beta)^2*(cos(beta)^2+...
                sin(beta)^2*cos(zeta)^2)-(cos(beta)^2*...
                (Qdot*Qe/(6*Mn)-vtotforwardgrid.^2)-13*...
                uC1^2).^2))/(64*uC1^2*cos(beta)^2*...
                (cos(beta)^2+sin(beta)^2*cos(zeta)^2)^2);
            radicand2=(sin(beta)^2*cos(zeta)^2*(64*uC2^2*...
                vtotforwardgrid.^2*cos(beta)^2*(cos(beta)^2+...
                sin(beta)^2*cos(zeta)^2)-(cos(beta)^2*...
                (Qdot*Qe/(6*Mn)-vtotforwardgrid.^2)-13*...
                uC2^2).^2))/(64*uC2^2*cos(beta)^2*...
                (cos(beta)^2+sin(beta)^2*cos(zeta)^2)^2);

            % Check for and exclude negative values of the
            % radicands
            r1positive=ones(size(radicand1));
            r1positive(radicand1<0)=0;
            r2positive=ones(size(radicand2));
            r2positive(radicand2<0)=0;

            % The four possible values of u_alpha (eq. 2.91)
            ua1plus=real((13*uC1^2+cos(beta)^2*(...
                vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC1*...
                (cos(beta)^2+sin(beta)^2*cos(zeta)^2))+...
                sqrt(radicand1));
            ua2plus=real((13*uC2^2+cos(beta)^2*(...
                vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC2*...
                (cos(beta)^2+sin(beta)^2*cos(zeta)^2))+...
                sqrt(radicand2));
            ua1minus=real((13*uC1^2+cos(beta)^2*(...
                vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC1*...
                (cos(beta)^2+sin(beta)^2*cos(zeta)^2))-...
                sqrt(radicand1));
            ua2minus=real((13*uC2^2+cos(beta)^2*(...
                vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC2*...
```

```
    (cos(beta)^2+sin(beta)^2*cos(zeta)^2))-...
    sqrt(radicand2));

% Check for and exclude solutions that do not keep
% eq. 2.89
check1plus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-...
    13*uC1^2/cos(beta)^2+8*(ua1plus*uC1+...
    cos(zeta)*sqrt((vtotforwardgrid.^2-...
    ua1plus.^2).*uC1^2*tan(beta)^2));
check1minus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-...
    13*uC1^2/cos(beta)^2+8*(ua1minus*uC1+...
    cos(zeta)*sqrt((vtotforwardgrid.^2-...
    ua1minus.^2).*uC1^2*tan(beta)^2));
check2plus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-...
    13*uC2^2/cos(beta)^2+8*(ua2plus*uC2+...
    cos(zeta)*sqrt((vtotforwardgrid.^2-...
    ua2plus.^2).*uC2^2*tan(beta)^2));
check2minus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-...
    13*uC2^2/cos(beta)^2+8*(ua2minus*uC2+...
    cos(zeta)*sqrt((vtotforwardgrid.^2-...
    ua2minus.^2).*uC2^2*tan(beta)^2));
% We only exclude solutions that do not keep eq.
% 2.89 by more than 9 m^2/s^2
checkzero1plus=ones(size(check1plus));
checkzero1plus(log10(abs(check1plus)+1)>1)=0;
checkzero1minus=ones(size(check1minus));
checkzero1minus(log10(abs(check1minus)+1)>1)=0;
checkzero2plus=ones(size(check2plus));
checkzero2plus(log10(abs(check2plus)+1)>1)=0;
checkzero2minus=ones(size(check2minus));
checkzero2minus(log10(abs(check2minus)+1)>1)=0;

% The checks are combined into acceptance matrices
% that define the grid points that are acceptable.
% The other grid points are set to zero.
acceptplus=r1positive.*r2positive.*...
    checkzero1plus.*checkzero2plus;
acceptminus=r1positive.*r2positive.*...
    checkzero1minus.*checkzero2minus;

% The four gyro-angles (eq. 2.16)
gamma1plus=acos((ua1plus-grid1*cos(phi))./...
    (grid2*sin(phi)));
gamma2plus=acos((ua2plus-grid1*cos(phi))./...
    (grid2*sin(phi)));
gamma1minus=acos((ua1minus-grid1*cos(phi))./...
    (grid2*sin(phi)));
gamma2minus=acos((ua2minus-grid1*cos(phi))./...
    (grid2*sin(phi)));
% The probability part of the weight function (eq.
% 2.92) is checked using the acceptance matrices
prob=abs(real(gamma2plus-gamma1plus))/pi.*...
    acceptplus+abs(real(gamma2minus-...
    gamma1minus))/pi.*acceptminus;
```

```
                        prob2stepGRS=prob2stepGRS+prob;
                    end
                end
                prob2stepGRS=prob2stepGRS/(length(betaloop)*...
                    length(zetaloop));
                % The total weight function
                wfcvv2stepGRSforward=RGRS.*prob2stepGRS/dEgares*dvpa*dvpe;
                Wtomo(Wrow,:)=reshape(wfcvv2stepGRSforward,1,...
                    numel(grid1))./Serror(Wrow);
            end
        end
        % Variable used to keep track of the number of measurement points
        % per diagnostic
        Sbin=Sbin+Sdim(i);
    end
    % The calculated weight function is used to obtain the
    % reconstructed distribution functions
    if strcmpi(pspace,'Ep')
        xalpha=Tikhonov_reg(Wtomo,Svec,pspacevec1,pspacevec2,M_f,...
            Tikhonov,alpha,uselsqnonneg,grad_cutoff,'Ep');
    elseif strcmpi(pspace,'vv')
        xalpha=Tikhonov_reg(Wtomo,Svec,pspacevec1,pspacevec2,M_f,...
            Tikhonov,alpha,uselsqnonneg,grad_cutoff,'vv');
    end
    WTomo=Wtomo;
end

end

function [xalpha] =Tikhonov_reg(W,S,pspacevec1,pspacevec2,M_f,...
    Tikhonov,alpha,uselsqnonneg,gradient_cutoff,pspace)
% Number of points in the grid
n = size(W,2);

%0th order Tikhonov penalty matrix
L0 = eye(n);

% The gradient matrices are determined
if strcmpi(pspace,'Ep')
    [L1,L2] = gradient_matrix(pspacevec1,pspacevec2,M_f,'Ep');
elseif strcmpi(pspace,'vv')
    [L1,L2] = gradient_matrix(pspacevec1,pspacevec2,M_f,'vv');
end

% The regions where the weight function coverage is below the selected
% cutoff are determined
no_weight_coverage = find(sum(W) < gradient_cutoff*mean(sum(W)));

% The gradient is removed from the selected regions with low weight
% function coverage
for i = no_weight_coverage

    L1(i,:) = 0;
```

```
    L1(i,i) = 1;
    L2(i,:) = 0;
    L2(i,i) = 1;
end

% The weight functions are rescaled
scaling_factor = 1/max(max(W));
W = W*scaling_factor;

% The penalty operator L^T*L is calculated (eqs. 2.11 and 2.12)
switch Tikhonov
    case 0 % Zeroth-order Tikhonov
        H = L0'*L0;
    case 1 % First-order Tikhonov
        H = L2'*L2 + L1'*L1;
    case 2 % Mix of zeroth- and first-order Tikhonov
        H = L2'*L2 + L1'*L1+1e-12*(L0')*L0;
    otherwise
        error('L must be 0, 1, 2 (0th or 1st order Tikhonov or a mix)')
end

% Pre-allocation for speed
xalpha = zeros(n,length(alpha));

% The reconstructed distribution function is calculated for every chosen
% value of alpha and saved as a column in a matrix (eq. 2.10)
for i = 1:length(alpha)
    switch Tikhonov
        case 0 % Zeroth-order Tikhonov
            if uselsqnonneg % With the non-negativity constraint
                GalphaL0=zeros(size(L0,2),1);
                WalphaL=double([W; sqrt(alpha(i))*L0]);
                GalphaL=double([S; GalphaL0]);
                xalpha(:,i) = lsqnonneg(WalphaL,GalphaL);
            else % Without the non-negativity constraint
                xalpha(:,i) = (W'*W + alpha(i)*H)\W'*S;
            end
        case 1 % First-order Tikhonov
            if uselsqnonneg % With the non-negativity constraint
                GalphaL1=zeros(2*size(L1,2),1);
                WalphaL=double([W; sqrt(alpha(i))*L1; sqrt(alpha(i))*L2]);
                GalphaL=double([S; GalphaL1]);
                xalpha(:,i) = lsqnonneg(WalphaL,GalphaL);
            else % Without the non-negativity constraint
                xalpha(:,i) = (W'*W + alpha(i)*H)\W'*S;
            end
        case 2 % Mix of zeroth- and first-order Tikhonov
            if uselsqnonneg % With the non-negativity constraint
                GalphaL1=zeros(2*size(L1,2),1);
                GalphaL0=zeros(size(L0,2),1);
                WalphaL=double([W; sqrt(alpha(i))*L1;...
                    sqrt(alpha(i))*L2; 1e-8*sqrt(alpha(i))*L0]);
                GalphaL=double([S; GalphaL1; GalphaL0]);
                xalpha(:,i) = lsqnonneg(WalphaL,GalphaL);
```

```
            else % Without the non-negativity constraint
                xalpha(:,i) = (W'*W + alpha(i)*H)\W'*S;
            end
    end
end
% The previous rescaling is undone
xalpha = xalpha*scaling_factor;
end

function [L1, L2] = gradient_matrix(pspacevec1,pspacevec2,M_f,pspace)
% The number of points along each axis
xdim=length(pspacevec1);
ydim=length(pspacevec2);

if strcmpi(pspace,'Ep')
    pspacevec1=pspacevec1*1.6021917e-19; % Conversion to [J]
end

% The step sizes along both axes
dx=pspacevec1(2)-pspacevec1(1);
dy=pspacevec2(2)-pspacevec2(1);

% The gradient matrices are defined according to eqs. 2.11 and 2.12 and the
% definitions in the book "Parameter Estimation and Inverse Problems" by
% Richard Aster, Brian Borchers and Clifford Thurber
Delta_x=diag(ones(1,xdim*ydim))+ diag(-1*ones(1,(xdim-1)*ydim),-ydim);
Delta_x(1:ydim,:) = 0;

if strcmpi(pspace,'Ep')
    Delta_x=Delta_x/dx*sqrt(2*M_f);
    for i=1:xdim
        Delta_x((i-1)*ydim+1:i*ydim,:) = Delta_x((i-1)*ydim+1:i*ydim,:)*...
            sqrt(pspacevec1(i));
    end
elseif strcmpi(pspace,'vv')
    Delta_x=Delta_x/dx;
end

Dy_c0=zeros(xdim*ydim,1);
Dy_c1=zeros(xdim*ydim,1);

for i = 1:xdim
    Dy_c0((i-1)*ydim+1:i*ydim) = [-1*ones(1,ydim-1) 0];
    Dy_c1((i-1)*ydim+1:i*ydim) = [ones(1,ydim-1) 0];
end

Dy_c1 = Dy_c1(1:end-1);
Delta_y = diag(Dy_c0) + diag(Dy_c1,1);

if strcmpi(pspace,'Ep')
    Delta_y = Delta_y/dy*sqrt(M_f/2);
    for i =1:xdim
        for j =1:ydim
```

```
            Delta_y((i-1)*ydim + j,:) = Delta_y((i-1)*ydim + j,:)*...
                sqrt(1-pspacevec2(j)^2)/sqrt(pspacevec1(i));
        end
    end
elseif strcmpi(pspace,'vv')
    Delta_y = Delta_y/dy;
end
L1 =Delta_x;
L2 =Delta_y;
end

function sigma = Bosch_Hale_sigma(E,reaction)
% This script calculates the NES-related D-D or D-T or the GRS-related
% D-p fusion cross section using the approximation published by Bosch
% and Hale in NF 1992. E is the energy in the center-of-mass frame
% in units of keV. For a stationary target ion, the center-of-mass
% energy is simply the energy of the fast ion in the lab frame times
% the ratio between its mass and the total mass of the fast and
% stationary ions. The cross section is given in units of mb (milli-barns).
% 1 mb = 10^-31 m^2
if strcmpi(reaction,'NESDD') % Constants for D-D fusion (Table 2.1)
    B_G = 31.3970;
    A1 = 5.3701e4;
    A2 = 3.3027e2;
    A3 = -1.2706e-1;
    A4 = 2.9327e-5;
    A5 = -2.5151e-9;
    B1 = 0;
    B2 = 0;
    B3 = 0;
    B4 = 0;
elseif strcmpi(reaction,'NESDT') % Constants for D-T fusion (Table 2.1)
    B_G = 34.3827;
    A1 = 6.927e4;
    A2 = 7.454e8;
    A3 = 2.050e6;
    A4 = 5.2002e4;
    A5 = 0;
    B1 = 6.38e1;
    B2 = -9.95e-1;
    B3 = 6.981e-5;
    B4 = 1.728e-4;
elseif strcmpi(reaction,'GRSDp') % Constants for p-D fusion (Table 2.1)
    E = E/1e3;
    B_G = 1.07;
    A1 = 8.09e-4;
    A2 = 1.92e-3;
    A3 = 1.21e-2;
    A4 = -5.26e-3;
    A5 = 6.52e-4;
    B1 = 0;
    B2 = 0;
    B3 = 0;
    B4 = 0;
```

```
end

S = (A1 + E.*(A2 + E.*(A3 + E.*(A4 + E.*A5)))./...
    (1+E.*(B1+E.*(B2+E.*(B3+E.*B4)))); % (eq. 2.53)
% The cross section is calculated (eq. 2.52)
sigma = S./(E.*exp(B_G./sqrt(E)));

end
```

Appendix C

```
% This function is used to calculate a known distribution function,
% generate synthetic spectra based on this distribution function and then
% use the function TomoAnalyticRmix to reconstruct distribution functions
% from these synthetic spectra. The known distribution function and the
% reconstructed ones are all plotted along with the synthetic spectra with
% and without noise as well as the spectra obtained from the
% reconstructions. The calculated quality factor Q for each reconstruction
% is also plotted as a function of the regularization parameter.
% The equations used are explained or derived in the report
% "Analytic models in velocity-space tomography for fusion plasmas" by
% Andreas Poulsen. All references refer to this report.

clear;

dist =1; %1 for Bi-Maxwellian, 2 for NBI and [1 2] for a sum of these
pspace =2; %1 for (E,p)-space and 2 for (vpa,vpe)-space

%Choose the used method for each diagnostic. 1 for CTS, 2 for NES of D-D
%fusion, 3 for NES of D-T fusion with fast D, 4 for NES of D-T fusion with
%fast T, 5 for one-step reaction GRS of D-p fusion with fast p, 6 for
%one-step reaction GRS of D-p fusion with fast D, 7 for FIDA and 8 for
%two-step reaction GRS of Be-alpha fusion with fast alpha. #elements must
%be 1 or equal to length(phivec)
method =2;

%observation angles, Sdim(i) measurements probing the target velocity space
%for view i,
phivec=[10 40 85];

%number of points in each spectrum: length must be equal to length(phivec)
% or 1, total number of measurements: sum(Sdim) or length(phivec)*Sdim if
% length(Sdim)=1
Sdim=100;

%The gradient used for the Tikhonov reconstruction is removed from the
%regions where the weight function coverage (sum of all weight functions)
%is less than gradient_cutoff times the mean of the weight function
%coverage. This leads to these regions being disregarded in the
%reconstructions
gradient_cutoff=0.00;

%natural constants
Mn   = 1.6749e-27; % [kg] Mass of a neutron
Mp   = 1.6726e-27; % [kg] Mass of a proton
Qe   = 1.6021917e-19; % [C] Elementary charge
c    = 3e8; % [m/s] The speed of light

%deuterium plasma
Mi=2*Mp; % [kg] Mass of a deuterium ion
M_He=3*Mp; % [kg] Mass of a helium-3 particle
n_t = 5e19; % [m^-3] The thermal ion density
vd=0; % [m/s] Toroidal drift of the bulk plasma
```

```
%plasma parameters used for the bi-Maxwellian distribution,
%isotropic means Tpara=Tperp
Tpara=200e3; % [eV] Parallel plasma temperature
Tperp=1.5e6; % [eV] Perpendicular plasma temperature
ni=1e19;     % [m^-3] Ion density
vparadrift=0; % [m/s] Parallel drift velocity

%Parameters used for the NBI distribution
nibeam=1e19; % [m^-3] Beam ion density
Mibeam=2*Mp; % [kg] Mass of the beam ions
Ecritbeam=30e3; % [eV] 30 keV for 60 keV injection,
% 400 keV for 1 MeV injection
Ebirthbeamfull=60e3; %keV
vbirthwidthbeam=3e4; %3e4 for 60 keV injection, 2e5 for 1 MeV injection
pbirthbeam=0.5; % Pitch of the injection
pitchwidthbeam=0.15;

%noiselevel of the signal
noiselevel=0.03;

%0 does 0th order Tikhonov, 1-1st order, 2-mix of 0th and 1st order
Tikhonov=1;
%0 does unconstrained Tikhonov, 1 does non-negative Tikhonov
uselsqnonneg=0;

% Values of the regularization parameter alpha
switch Tikhonov
    case 0
        alpha = logspace(-2,2,9)';
    case 1
        alpha= logspace(8,13,9)';
    case 2
        alpha = logspace(8,11,9)';
    otherwise
        error('L must be 0, 1, 2 (0th or 1st order Tikhonov or a mix)')
end

%%%%parameters for (E,p)-space
%boundaries of the target (E,p)-space
Emin=5e2; % 0.1e6
Emax=80e3;
pmin=-0.99;
pmax=0.99;

%resolution of the tomography grid in (E,p)
Edimtomo=30;
pdimtomo=30;

%resolution of the forward model grid in (E,p)
Edimforward=40;
pdimforward=40;

%%%%parameters for (vpa,vpe)-space
```

```
%boundaries of the target (v_para,v_perp)-space
vparamax=10e6;
vparamin=-10e6;
vperpmin=1e3;
vperpmax=30e6;

%resolution of the tomography grid in (v_para,v_perp)
vparadimtomo=30;
vperpdimtomo=30;

%resolution of the forward model grid in (v_para,v_perp)
vparadimforward=40;
vperpdimforward=40;

%The possible methods
methodvec=["CTS", "NESDD", "NESDTfastD", "NESDTfastT", "GRSDpfastp",...
    "GRSDpfastD", "FIDA", "2stepGRS"];

%If method is a single value, it is changed to a vector of length equal
%to that of phivec where every element has that value
if length(method)==1
    method=ones(1,length(phivec))*method;
end
%The same is done to Sdim
if length(Sdim)==1
    Sdim=ones(1,length(phivec))*Sdim;
end

%A string array denoting the used methods is created
methods=strings(1,length(method));
for i=1:length(method)
    methods(i)=methodvec(method(i));
end

% The forward and tomography grids are defined based on the chosen plotting
% space via vectors that contain the values to be used in the grids. The
% grids are then used to calculate the values of the chosen 2D distribution
% on both the forward and the tomography grid. The axes and their labels
% are also defined

if pspace == 1 % (E,p)-space
    dEforward=(Emax-Emin)/(Edimforward-1);
    dpforward=(pmax-pmin)/(pdimforward-1);

    dEtomo=(Emax-Emin)/(Edimtomo-1);
    dptomo=(pmax-pmin)/(pdimtomo-1);

    Eforward = Emin:dEforward:Emax;
    pforward = pmin:dpforward:pmax;
    [Eforwardgrid,pforwardgrid] = meshgrid(Eforward,pforward);
    forwardgrid1=Eforwardgrid;
    forwardgrid2=pforwardgrid;
```

```
Etomo = Emin:dEtomo:Emax;
ptomo = pmin:dptomo:pmax;
[Etomogrid,ptomogrid] = meshgrid(Etomo,ptomo);
tomogrid1=Etomogrid;
tomogrid2=ptomogrid;

distsimforward=zeros(size(Eforwardgrid));
distsimtomo=zeros(size(Etomogrid));
% The model distribution can be bi-Maxwellian, slowing-down or a sum of
% these
for j =1:length(dist)
    if dist(j) ==1 % bi-Maxwellian distribution in (E,p)-space
        distsimforward=ni.*sqrt(Eforwardgrid./(pi.*Tpara.*Tperp.^2.*...
            Qe.^2)).*exp(-(pforwardgrid.^2.*Eforwardgrid.*Qe+0.5.*...
            Mi.*vparadrift.^2-vparadrift.*pforwardgrid.*sqrt(2.*...
            Mi.*Eforwardgrid.*Qe))/(Tpara.*Qe)-(1-pforwardgrid.^2).*...
            Eforwardgrid./Tperp).*Qe+distsimforward; % eq. 2.94

        distsimtomo=ni.*sqrt(Etomogrid./(pi.*Tpara.*Tperp.^2.*...
            Qe.^2)).*exp(-(ptomogrid.^2.*Etomogrid.*Qe+0.5.*Mi.*...
            vparadrift.^2-vparadrift.*ptomogrid.*sqrt(2.*Mi.*...
            Etomogrid.*Qe))/(Tpara.*Qe)-(1-ptomogrid.^2).*...
            Etomogrid./Tperp).*Qe+distsimtomo; % eq. 2.94
    elseif dist(j) ==2 % Slowing-down distribution in (E,p)-space
        % Supplied code
        vcritbeam=sqrt(2*Ecritbeam*Qe/Mibeam);
        vbirthbeamfull=sqrt(2*Ebirthbeamfull*Qe/Mibeam);
        vbirthbeamhalf=sqrt(2*Ebirthbeamfull/2*Qe/Mibeam);
        vbirthbeamthird=sqrt(2*Ebirthbeamfull/3*Qe/Mibeam);
        pitchkernel=exp(-(pforward-pbirthbeam).^2/pitchwidthbeam^2);
        intkernel=trapz(pforward,pitchkernel);
        pitchkernel=pitchkernel/intkernel;

        vmaxE=sqrt(2*Emax*Qe/Mi);
        vminE=sqrt(2*Emin*Qe/Mi);
        v=vminE:(vmaxE/200):vmaxE;
        [Vforward,Pforward2]=meshgrid(v,pforward);

        VRATIOBCFULL=((Vforward.^3/vbirthbeamfull^3).*(...
            vbirthbeamfull^3+vcritbeam^3)./(Vforward.^3+...
            vcritbeam^3)).^(1/6);
        VRATIOBCHALF=((Vforward.^3/vbirthbeamhalf^3).*(...
            vbirthbeamhalf^3+vcritbeam^3)./(Vforward.^3+...
            vcritbeam^3)).^(1/6);
        VRATIOBCTHIRD=((Vforward.^3/vbirthbeamthird^3).*(...
            vbirthbeamthird^3+vcritbeam^3)./(Vforward.^3+...
            vcritbeam^3)).^(1/6);
        SUMLEGENDREFULL=zeros(size(VRATIOBCFULL));
        SUMLEGENDREHALF=zeros(size(VRATIOBCFULL));
        SUMLEGENDRETHIRD=zeros(size(VRATIOBCFULL));

        for n=0:15
```

```
    legendrep=legendreP(n,Pforward2);
    legendrepbirth=trapz(pforward,pitchkernel.*...
        legendreP(n,pforward));
    SUMLEGENDREFULL=SUMLEGENDREFULL+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCFULL.^(n*(n+1)).*...
        erfc((Vforward-vbirthbeamfull)/vbirthwidthbeam)/2;
    SUMLEGENDREHALF=SUMLEGENDREHALF+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCHALF.^(n*(n+1)).*...
        erfc((Vforward-vbirthbeamhalf)/vbirthwidthbeam)/2;
    SUMLEGENDRETHIRD=SUMLEGENDRETHIRD+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCTHIRD.^(n*(n+1)).*...
        erfc((Vforward-vbirthbeamthird)/vbirthwidthbeam)/2;
end
SUMLEGENDREFULL(SUMLEGENDREFULL<0)=0;
SUMLEGENDREHALF(SUMLEGENDREHALF<0)=0;
SUMLEGENDRETHIRD(SUMLEGENDRETHIRD<0)=0;
fvpbeam=Vforward.^2./(Vforward.^3+vcritbeam^3).*(0.5*...
    SUMLEGENDREFULL+0.3*SUMLEGENDREHALF+0.2*SUMLEGENDRETHIRD);

v_on_E_grid = sqrt(2*Eforwardgrid*Qe/Mibeam);
fEpbeam=interp2(v,pforward,fvpbeam./(Mibeam*Vforward),...
    v_on_E_grid,pforwardgrid,'linear',0);
fEpbeam=nibeam/trapz(pforward,trapz(Eforward,fEpbeam'))*...
    fEpbeam;
distsimforward=fEpbeam+distsimforward;

pitchkernel=exp(-(ptomo-pbirthbeam).^2/pitchwidthbeam^2);
intkernel=trapz(ptomo,pitchkernel);
pitchkernel=pitchkernel/intkernel;
[Vtomo,Ptomo2]=meshgrid(v,ptomo);

VRATIOBCFULL=((Vtomo.^3/vbirthbeamfull^3).*(...
    vbirthbeamfull^3+vcritbeam^3)./(Vtomo.^3+...
    vcritbeam^3)).^(1/6);
VRATIOBCHALF=((Vtomo.^3/vbirthbeamhalf^3).*(...
    vbirthbeamhalf^3+vcritbeam^3)./(Vtomo.^3+...
    vcritbeam^3)).^(1/6);
VRATIOBCTHIRD=((Vtomo.^3/vbirthbeamthird^3).*(...
    vbirthbeamthird^3+vcritbeam^3)./(Vtomo.^3+...
    vcritbeam^3)).^(1/6);
SUMLEGENDREFULL=zeros(size(VRATIOBCFULL));
SUMLEGENDREHALF=zeros(size(VRATIOBCFULL));
SUMLEGENDRETHIRD=zeros(size(VRATIOBCFULL));
for n=0:15
    legendrep=legendreP(n,Ptomo2);
    legendrepbirth=trapz(ptomo,pitchkernel.*...
        legendreP(n,ptomo));
    SUMLEGENDREFULL=SUMLEGENDREFULL+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCFULL.^(n*(n+1)).*...
        erfc((Vtomo-vbirthbeamfull)/vbirthwidthbeam)/2;
    SUMLEGENDREHALF=SUMLEGENDREHALF+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCHALF.^(n*(n+1)).*...
        erfc((Vtomo-vbirthbeamhalf)/vbirthwidthbeam)/2;
    SUMLEGENDRETHIRD=SUMLEGENDRETHIRD+(n+0.5)*legendrep.*...
```

```
                    legendrepbirth.*VRATIOBCTHIRD.^(n*(n+1)).*...
                    erfc((Vtomo-vbirthbeamthird)/vbirthwidthbeam)/2;
            end
            SUMLEGENDREFULL(SUMLEGENDREFULL<0)=0;
            SUMLEGENDREHALF(SUMLEGENDREHALF<0)=0;
            SUMLEGENDRETHIRD(SUMLEGENDRETHIRD<0)=0;
            fvpbeam=Vtomo.^2./(Vtomo.^3+vcritbeam^3).*(0.5*...
                SUMLEGENDREFULL+0.3*SUMLEGENDREHALF+0.2*SUMLEGENDRETHIRD);

            v_on_E_grid = sqrt(2*Etomogrid*Qe/Mibeam);
            fEpbeam=interp2(v,ptomo,fvpbeam./(Mibeam*Vtomo),v_on_E_grid,...
                ptomogrid,'linear',0);
            fEpbeam=nibeam/trapz(ptomo,trapz(Etomo,fEpbeam'))*fEpbeam;
            distsimtomo=fEpbeam+distsimtomo;

        end
    end

    % The labels and axes are defined
    if Emax>=1e6
        axes = [Emin/1e6 Emax/1e6 pmin pmax];
        xlab ='Energy [MeV]';
        xaxisforward = Eforward/1e6;
        xaxistomo = Etomo/1e6;
    elseif Emax<1e6
        axes = [0 Emax/1e3 pmin pmax];
        xlab ='Energy [keV]';
        xaxisforward = Eforward/1e3;
        xaxistomo = Etomo/1e3;
    end
    yaxisforward = pforward;
    yaxistomo = ptomo;
    ylab ='Pitch [-]';
elseif pspace == 2 % (vpa,vpe)-space
    dvparaforward=(vparamax-vparamin)/(vparadimforward-1);
    dvperpforward=(vperpmax-vperpmin)/(vperpdimforward-1);

    dvparatomo=(vparamax-vparamin)/(vparadimtomo-1);
    dvperptomo=(vperpmax-vperpmin)/(vperpdimtomo-1);

    vparaforward = vparamin:dvparaforward:vparamax;
    vperpforward = vperpmin:dvperpforward:vperpmax;
    [vparaforwardgrid,vperpforwardgrid] = meshgrid(vparaforward,...
        vperpforward);
    forwardgrid1=vparaforwardgrid;
    forwardgrid2=vperpforwardgrid;

    vparatomo = vparamin:dvparatomo:vparamax;
    vperptomo = vperpmin:dvperptomo:vperpmax;
    [vparatomogrid,vperptomogrid] = meshgrid(vparatomo,vperptomo);
    tomogrid1=vparatomogrid;
    tomogrid2=vperptomogrid;
```

```
distsimforward=zeros(size(vparaforwardgrid));
distsimtomo=zeros(size(vparatomogrid));
% The model distribution can be bi-Maxwellian, slowing-down or a sum of
% these
for j=1:length(dist)
    if dist(j)==1 % bi-Maxwellian distribution in (vpa,vpe)-space
        %define thermal velocities
        vthpara=sqrt(2*Tpara*Qe/Mi);
        vthperp=sqrt(2*Tperp*Qe/Mi);

        fvpavpe3DbiMaxDriftforward=ni./(pi.^(3/2).*vthpara.*...
            vthperp.^2).*exp(-((vparaforwardgrid-vparadrift)./...
            vthpara).^2-(vperpforwardgrid./vthperp).^2);

        fvpavpe3DbiMaxDrifttomo=ni./(pi.^(3/2).*vthpara.*...
            vthperp.^2).*exp(-((vparatomogrid-vparadrift)./...
            vthpara).^2-(vperptomogrid./vthperp).^2);

        distsimforward=fvpavpe3DbiMaxDriftforward.*2.*pi.*...
            vperpforwardgrid+distsimforward; % eq. 2.93

        distsimtomo=fvpavpe3DbiMaxDrifttomo.*2.*pi.*vperptomogrid+...
            distsimtomo; % eq. 2.93
    elseif dist(j)==2 % Slowing-down distribution in (vpa,vpe)-space
        % Supplied code
        pmin=-0.99;
        pmax=0.99;
        Emax=0.5*Mi*(vparamax^2+vperpmax^2)/Qe;
        Emin=0.5*Mi*vperpmin^2/Qe;
        pdimforward=vperpdimforward;
        Edimforward=vparadimforward;
        pdimtomo=vperpdimtomo;
        Edimtomo=vparadimtomo;

        dEforward=(Emax-Emin)/(Edimforward-1);
        dpforward=(pmax-pmin)/(pdimforward-1);

        dEtomo=(Emax-Emin)/(Edimtomo-1);
        dptomo=(pmax-pmin)/(pdimtomo-1);

        Eforward = Emin:dEforward:Emax;
        pforward = pmin:dpforward:pmax;
        [Eforwardgrid,pforwardgrid] = meshgrid(Eforward,pforward);

        Etomo = Emin:dEtomo:Emax;
        ptomo = pmin:dptomo:pmax;
        [Etomogrid,ptomogrid] = meshgrid(Etomo,ptomo);

        vcritbeam=sqrt(2*Ecritbeam*Qe/Mibeam);
        vbirthbeamfull=sqrt(2*Ebirthbeamfull*Qe/Mibeam);
        vbirthbeamhalf=sqrt(2*Ebirthbeamfull/2*Qe/Mibeam);
        vbirthbeamthird=sqrt(2*Ebirthbeamfull/3*Qe/Mibeam);
        pitchkernel=exp(-(pforward-pbirthbeam).^2/pitchwidthbeam^2);
```

```
intkernel=trapz(pforward,pitchkernel);
pitchkernel=pitchkernel/intkernel;

vmax=sqrt(vparamax^2+vperpmax^2);
vmin=vperpmin;
v=vmin:(vmax/200):vmax;
[Vforward,Pforward2]=meshgrid(v,pforward);

VRATIOBCFULL=((Vforward.^3/vbirthbeamfull^3).*(...
    vbirthbeamfull^3+vcritbeam^3)./(Vforward.^3+...
    vcritbeam^3)).^(1/6);
VRATIOBCHALF=((Vforward.^3/vbirthbeamhalf^3).*(...
    vbirthbeamhalf^3+vcritbeam^3)./(Vforward.^3+...
    vcritbeam^3)).^(1/6);
VRATIOBCTHIRD=((Vforward.^3/vbirthbeamthird^3).*(...
    vbirthbeamthird^3+vcritbeam^3)./(Vforward.^3+...
    vcritbeam^3)).^(1/6);
SUMLEGENDREFULL=zeros(size(VRATIOBCFULL));
SUMLEGENDREHALF=zeros(size(VRATIOBCFULL));
SUMLEGENDRETHIRD=zeros(size(VRATIOBCFULL));

for n=0:15
    legendrep=legendreP(n,Pforward2);
    legendrepbirth=trapz(pforward,pitchkernel.*...
        legendreP(n,pforward));
    SUMLEGENDREFULL=SUMLEGENDREFULL+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCFULL.^(n*(n+1)).*...
        erfc((Vforward-vbirthbeamfull)/vbirthwidthbeam)/2;
    SUMLEGENDREHALF=SUMLEGENDREHALF+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCHALF.^(n*(n+1)).*...
        erfc((Vforward-vbirthbeamhalf)/vbirthwidthbeam)/2;
    SUMLEGENDRETHIRD=SUMLEGENDRETHIRD+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCTHIRD.^(n*(n+1)).*...
        erfc((Vforward-vbirthbeamthird)/vbirthwidthbeam)/2;
end
SUMLEGENDREFULL(SUMLEGENDREFULL<0)=0;
SUMLEGENDREHALF(SUMLEGENDREHALF<0)=0;
SUMLEGENDRETHIRD(SUMLEGENDRETHIRD<0)=0;
fvpbeam=Vforward.^2./(Vforward.^3+vcritbeam^3).*(0.5*...
    SUMLEGENDREFULL+0.3*SUMLEGENDREHALF+0.2*SUMLEGENDRETHIRD);
v_on_E_grid = sqrt(2*Eforwardgrid*Qe/Mibeam);
fEpbeam=interp2(v,pforward,fvpbeam./(Mibeam*Vforward),...
    v_on_E_grid,pforwardgrid,'linear',0);
fEpbeam=nibeam/trapz(pforward,trapz(Eforward,fEpbeam'))*...
    fEpbeam;
JacobianEpvpavpe=Mi*vperpforwardgrid./...
    sqrt(vparaforwardgrid.^2+vperpforwardgrid.^2)/Qe;
E_on_vpavpeMESH=0.5*Mi*(vparaforwardgrid.^2+...
    vperpforwardgrid.^2)/Qe;
p_on_vpavpeMESH=vparaforwardgrid./sqrt(vparaforwardgrid.^2+...
    vperpforwardgrid.^2);
fvpavpe2DtransformedNBI=interp2(Eforward,pforward,fEpbeam,...
    E_on_vpavpeMESH,p_on_vpavpeMESH,'*spline');
fvpavpe2DtransformedNBI=fvpavpe2DtransformedNBI.*...
```

```
    JacobianEpvpavpe;
distsimforward=distsimforward+fvpavpe2DtransformedNBI;

pitchkernel=exp(-(ptomo-pbirthbeam).^2/pitchwidthbeam^2);
intkernel=trapz(ptomo,pitchkernel);
pitchkernel=pitchkernel/intkernel;
[Vtomo,Ptomo2]=meshgrid(v,ptomo);

VRATIOBCFULL=((Vtomo.^3/vbirthbeamfull^3).*(...
    vbirthbeamfull^3+vcritbeam^3)./(Vtomo.^3+...
    vcritbeam^3)).^(1/6);
VRATIOBCHALF=((Vtomo.^3/vbirthbeamhalf^3).*(...
    vbirthbeamhalf^3+vcritbeam^3)./(Vtomo.^3+...
    vcritbeam^3)).^(1/6);
VRATIOBCTHIRD=((Vtomo.^3/vbirthbeamthird^3).*(...
    vbirthbeamthird^3+vcritbeam^3)./(Vtomo.^3+...
    vcritbeam^3)).^(1/6);
SUMLEGENDREFULL=zeros(size(VRATIOBCFULL));
SUMLEGENDREHALF=zeros(size(VRATIOBCFULL));
SUMLEGENDRETHIRD=zeros(size(VRATIOBCFULL));
for n=0:15
    legendrep=legendreP(n,Ptomo2);
    legendrepbirth=trapz(ptomo,pitchkernel.*...
        legendreP(n,ptomo));
    SUMLEGENDREFULL=SUMLEGENDREFULL+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCFULL.^(n*(n+1)).*...
        erfc((Vtomo-vbirthbeamfull)/vbirthwidthbeam)/2;
    SUMLEGENDREHALF=SUMLEGENDREHALF+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCHALF.^(n*(n+1)).*...
        erfc((Vtomo-vbirthbeamhalf)/vbirthwidthbeam)/2;
    SUMLEGENDRETHIRD=SUMLEGENDRETHIRD+(n+0.5)*legendrep.*...
        legendrepbirth.*VRATIOBCTHIRD.^(n*(n+1)).*...
        erfc((Vtomo-vbirthbeamthird)/vbirthwidthbeam)/2;
end
SUMLEGENDREFULL(SUMLEGENDREFULL<0)=0;
SUMLEGENDREHALF(SUMLEGENDREHALF<0)=0;
SUMLEGENDRETHIRD(SUMLEGENDRETHIRD<0)=0;
fvpbeam=Vtomo.^2./(Vtomo.^3+vcritbeam^3).*(0.5*...
    SUMLEGENDREFULL+0.3*SUMLEGENDREHALF+0.2*SUMLEGENDRETHIRD);

v_on_E_grid = sqrt(2*Etomogrid*Qe/Mibeam);
fEpbeam=interp2(v,ptomo,fvpbeam./(Mibeam*Vtomo),v_on_E_grid,...
    ptomogrid,'linear',0);
fEpbeam=nibeam/trapz(ptomo,trapz(Etomo,fEpbeam'))*fEpbeam;

JacobianEpvpavpe=Mi*vperptomogrid./...
    sqrt(vparatomogrid.^2+vperptomogrid.^2)/Qe;
E_on_vpavpeMESH=0.5*Mi*(vparatomogrid.^2+...
    vperptomogrid.^2)/Qe;
p_on_vpavpeMESH=vparaforwardgrid./sqrt(vparatomogrid.^2+...
    vperptomogrid.^2);
fvpavpe2DtransformedNBI=interp2(Etomo,ptomo,fEpbeam,...
    E_on_vpavpeMESH,p_on_vpavpeMESH,'*spline');
fvpavpe2DtransformedNBI=fvpavpe2DtransformedNBI.*...
```

```
                JacobianEpvpavpe;
            distsimtomo=distsimtomo+fvpavpe2DtransformedNBI;
        end
    end

    % The axes and labels are defined
    axes=[vparamin vparamax vperpmin vperpmax]/1e6;
    xlab='v_{||} [10^6 m/s]';
    ylab='v_{\perp} [10^6 m/s]';
    xaxisforward=vparaforward/1e6;
    yaxisforward=vperpforward/1e6;
    xaxistomo=vparatomo/1e6;
    yaxistomo=vperptomo/1e6;
end

% Pre-allocation for speed
%Matrix to hold the weight functions
Wforward=zeros(sum(Sdim),numel(forwardgrid1));
Spoint=zeros(1,sum(Sdim)); %Vector to hold the spectrum bin values
Wrow=0;
Sbin=0;

% The weight functions for each individual diagnostic are calculated
% and stored in the designated matrix

for i=1:length(phivec)
    phi=phivec(i)*pi/180;

    % Parameters that differ for different methods
    if method(i) == 2 %NES for D-D fusion
        M_f=Mi; % [kg] Mass of the fast ion (deuterium)
        Q = 3.27e6; % [eV] Energy released in the D-D fusion reaction
        CM_coef = 1/2; % Coefficient for the fast-ion center-of-mass energy
        reaction = 'NESDD';
    elseif method(i) == 3 %NES for D-T fusion with fast D
        M_f =Mi; % [kg] Mass of the fast ion (deuterium)
        Q = 17.6e6; % [eV] Energy released in the D-T fusion reaction
        CM_coef = 2/5; % Coefficient for the fast-ion center-of-mass energy
        M_He = 4*Mp; % [kg] Mass of an alpha-particle
        reaction = 'NESDT';
    elseif method(i) == 4 %NES for D-T fusion with fast T
        M_f =3*Mp; % [kg] Mass of the fast ion (deuterium)
        Q = 17.6e6; % [eV] Energy released in the D-T fusion reaction
        CM_coef = 3/5; % Coefficient for the fast-ion center-of-mass energy
        M_He = 4*Mp; % [kg] Mass of an alpha-particle
        reaction = 'NESDT';
    elseif method(i) == 5 %One-step reaction GRS for D-p fusion with fast p
        Q = 5.5e6; % [eV] Energy released in the D-p fusion reaction
        CM_coef = 1/3; % Coefficient for the fast-ion center-of-mass energy
        M_f = Mp; % [kg] Mass of the fast ion (proton)
        M_pr = 3*Mp; % [kg] Mass of a helium-3 particle
        reaction = 'GRSDp';
    elseif method(i) == 6 %One-step reaction GRS for D-p fusion with fast D
```

```
    Q = 5.5e6; % [eV] Energy released in the D-p fusion reaction
    CM_coef = 2/3; % Coefficient for the fast-ion center-of-mass energy
    M_f = 2*Mp; % [kg] Mass of the fast ion (deuterium)
    M_pr = 3*Mp; % [kg] Mass of a helium-3 particle
    reaction = 'GRSDp';
elseif method(i) == 7 % FIDA
    gamma_bar=0; % The phase-shift of the cosine model for the pdf
    B=1.74; % [T] The magnetic field
    E0=1e6; % [eV] The free parameter used in the model for the
    % amplitude of the cosine model for the pdf
    lambda0=656.1e-9; % [m] The wavelength of the unshifted
    % D_alpha-line
    C_hat =[1,18,16,1681,2304,729,1936,5490,1936,729,2304,1681,...
        16,18,1]/18860; % Constants related to the probabilities of
    % the various Stark lines
    % Proportionality constants for the Stark splitting wavelength
    % shifts
    sl = [-220.2,-165.2,-137.7,-110.2,-82.64,-55.1,-27.56,0,27.57,...
        55.15,82.74,110.3,138.0,165.6,220.9]*1e-18; % [m^2/V]
    % Signs related to the various Stark lines. + for the
    % sigma-lines and - for the pi-lines
    Stark_sign = [-1,1,1,-1,-1,-1,1,1,1,-1,-1,-1,1,1,-1];
    % Parameters used for the ad-hoc model for the FIDA intensity
    % function R
    Rni=1e8; % [m^-3] Ion density
    RTpara=60e3; % [eV] Parallel plasma temperature
    RTperp=60e3; % [eV] Perpendicular plasma temperature
    Rvparadrift=0.4e6; % [m/s] Parallel drift velocity
elseif method(i) == 8 % Two-step reaction GRS
    Ega0=4.44e6; % [eV]
    Qdot=1.26e6; % [eV]
    n_Be=1e18; % [m^-3] Density of thermal Be
    betaloop=(90.5:0.5:179.5)/180*pi;
    zetaloop=(0:0.5:179.5)/180*pi;
    Esigma=0:0.1:6.1; % [MeV]
    sigma=[0 0 0 0 0 0 0 0 0 0 ...
        0 0 0 0 0 10 30 100 180 230 ...
        220 200 80 70 60 60 80 100 80 60 ...
        60 60 60 60 60 65 70 75 200 320 ...
        320 320 220 240 260 260 250 280 320 340 ...
        340 320 320 320 320 320 320 320 290 250 ...
        250 0]; % [mb]
end

if method(i) ==1 && pspace ==1 % CTS in (E,p)-space
    ubroadening =3; % Spectral resolution of the measurements divided
    % by the bin width of the spectra
    % The value of the upper/lower limit of the spectra bins
    umax=sqrt(2*Emax*Qe/Mi); % [m/s] (eq. 2.21)
    du=(2*umax)/(Sdim(i)-1); % The bin width of the spectra
    dures=ubroadening*du; % Spectral resolution of the measurements
    uvec=(-umax:du:umax); % The bin values of the spectra
    % The calculated bin values are stored in the designated vector
    Spoint((Sbin+1):(Sbin+Sdim(i)))=uvec;
```

```
    % For each bin in the spectra, the weight function for the chosen
    % forward grid is calculated using eq. 2.19 and stored as a row in
    % a matrix
    for u=uvec
        Wrow=Wrow+1;
        % The two gyro-angles (eq. 2.20)
        gamma1forward = acos(1./sin(phi).*1./sqrt(1-...
            pforwardgrid.^2).*((u-dures./2)./sqrt(2.*...
            Eforwardgrid./Mi*Qe)-pforwardgrid.*cos(phi)));
        gamma2forward = acos(1./sin(phi).*1./sqrt(1-...
            pforwardgrid.^2).*((u+dures./2)./sqrt(2.*...
            Eforwardgrid./Mi*Qe)-pforwardgrid.*cos(phi)));
        % The weight function (eq. 2.19) times the step sizes
        wfcEpCTSforward=real((gamma1forward-gamma2forward))./pi./...
            dures.*dEforward.*dpforward;
        Wforward(Wrow,:)=reshape(wfcEpCTSforward,1,...
            numel(Eforwardgrid));
    end
elseif method(i) ==1 && pspace ==2 % CTS in (vpa,vpe)-space
    ubroadening =3; % Spectral resolution of the measurements divided
    % by the bin width of the spectra
    % The value of the upper/lower limit of the spectra bins
    if phi <= pi/2
        umin=vparamin*cos(phi)-vperpmax*sin(phi); % [m/s] (eq. 2.22)
        umax=vparamax*cos(phi)+vperpmax*sin(phi); % [m/s] (eq. 2.23)
    elseif phi > pi/2
        umin=vparamax*cos(phi)-vperpmax*sin(phi); % [m/s] (eq. 2.24)
        umax=vparamin*cos(phi)+vperpmax*sin(phi); % [m/s] (eq. 2.25)
    end
    du=(umax-umin)/(Sdim(i)-1); % The bin width of the spectra
    dures=ubroadening*du; % The spectral resolution of the measurements
    uvec=(umin:du:umax); % The bin values of the spectra
    % The calculated bin values are stored in the designated vector
    Spoint((Sbin+1):(Sbin+Sdim(i)))=uvec;
    % For each bin in the spectra, the weight function for the chosen
    % forward grid is calculated using eq. 2.19 and stored as a row in
    % a matrix
    for u=uvec
        Wrow=Wrow+1;
        % The two gyro-angles (eq. 2.16)
        gamma1forward = acos(1./(sin(phi).*vperpforwardgrid).*...
            (u-dures./2-vparaforwardgrid.*cos(phi)));
        gamma2forward = acos(1./(sin(phi).*vperpforwardgrid).*...
            (u+dures./2-vparaforwardgrid.*cos(phi)));
        % The weight function (eq. 2.19) times the step sizes
        wfcvvCTSforward=real((gamma1forward-gamma2forward))./pi./...
            dures.*dvparaforward.*dvperpforward;
        Wforward(Wrow,:)=reshape(wfcvvCTSforward,1,...
            numel(vparaforwardgrid));
    end
elseif (method(i)==2 || method(i)==3 || method(i)==4) && pspace ==1
    %NES in (E,p)-space
    Enbroadening=1; % Spectral resolution of the measurements divided
    % by the bin width of the spectra
```

```
    % The values of the upper/lower limits of the spectra bins
    % The lower limit of the spectra bins (eq. 2.62)
    Enmin=M_He*(M_He-M_f)*Q/((M_He+Mn)*(M_He-M_f)+Mn*M_f); % [eV]
    % The upper limit of the spectra bins (eqs. 2.70-2.73)
    A=4*(M_He+Mn)^2;
    B=-(8*M_He*(M_He+Mn)*Q+8*(M_He+Mn)*(M_He-M_f)*Emax+16*Mn*M_f*Emax);
    C=4*M_He^2*Q^2+8*M_He*(M_He-M_f)*Q*Emax+4*(M_He-M_f)^2*Emax^2;
    Enmax=(-B+sqrt(B^2-4*A*C))/(2*A); % [eV]
    dEn=(Enmax-Enmin)/(Sdim(i)-1); % The bin width of the spectra
    dEnres=Enbroadening*dEn; % Spectral resolution of the measurements
    Envec=(Enmin:dEn:Enmax); % The bin values of the spectra
    % The calculated bin values are stored in the designated vector
    Spoint((Sbin+1):(Sbin+Sdim(i)))=Envec;
    % The values of vpa and vpe corresponding to the chosen
    % (E,p)-grid are calculated (eq. 2.3)
    vpara=pforwardgrid.*sqrt(2.*Eforwardgrid.*Qe./M_f);
    vperp=sqrt(1-pforwardgrid.^2).*sqrt(2.*Eforwardgrid.*Qe./M_f);
    % The center-of-mass energy of the fast ion in [keV]
    E_CM=CM_coef.*M_f.*((vpara-vd).^2+vperp.^2)./Qe/2e3;
    % The rate part of the weight function (eq. 2.51)
    RNES=n_t.*Bosch_Hale_sigma(E_CM,reaction).*10^-31.*...
        sqrt((vpara-vd).^2+vperp.^2);
    RNES(RNES<0)=0; % Negative rates are set to zero
    % For each bin in the spectra, the weight function for the chosen
    % forward grid is calculated using eq. 2.44 and stored as a row in
    % a matrix
    for En=Envec
        Wrow=Wrow+1;
        % The two gyro-angles (eq. 2.48)
        gamma1forward=acos(1./(sqrt(1-pforwardgrid.^2).*sin(phi)).*(...
            1./2.*(M_He+Mn)./sqrt(M_f.*Mn).*sqrt((En-dEnres./2)./...
            Eforwardgrid)-1./2.*(M_He-M_f)./sqrt(M_f.*Mn).*sqrt(...
            Eforwardgrid./(En-dEnres./2))-1./2.*M_He./sqrt(M_f.*...
            Mn).*Q./(sqrt(Eforwardgrid.*(En-dEnres./2)))-...
            pforwardgrid.*cos(phi)));
        gamma2forward=acos(1./(sqrt(1-pforwardgrid.^2).*sin(phi)).*(...
            1./2.*(M_He+Mn)./sqrt(M_f.*Mn).*sqrt((En+dEnres./2)./...
            Eforwardgrid)-1./2.*(M_He-M_f)./sqrt(M_f.*Mn).*sqrt(...
            Eforwardgrid./(En+dEnres./2))-1./2.*M_He./sqrt(M_f.*...
            Mn).*Q./(sqrt(Eforwardgrid.*(En+dEnres./2)))-...
            pforwardgrid.*cos(phi)));
        % The probability part of the weight function (eq. 2.45)
        probNES=real((gamma1forward-gamma2forward))./pi;
        % The total weight function (eq. 2.44) times the step sizes
        wfcEpNESforward=probNES.*RNES./dEnres.*...
            dEforward.*dpforward;
        Wforward(Wrow,:)=reshape(wfcEpNESforward,1,...
            numel(Eforwardgrid));
    end
elseif (method(i)==2 || method(i)==3 || method(i)==4) && pspace ==2
    % NES in (vpa,vpe)-space
    Enbroadening=1; % Spectral resolution of the measurements divided
    % by the bin width of the spectra
    % The values of the upper/lower limits of the spectra bins
```

```
    % The lower limit of the spectra bins (eq. 2.62)
    Enmin=M_He*(M_He-M_f)*Q/((M_He+Mn)*(M_He-M_f)+Mn*M_f); % [eV]
    % The maximum possible fast ion energy for the considered
    % velocity-space region
    EmaxW=0.5*M_f*(max(abs(vparatomo))^2+vperpmax^2)/Qe;
    % The upper limit of the spectra bins (eq. 2.70-2.73)
    A=4*(M_He+Mn)^2;
    B=-(8*M_He*(M_He+Mn)*Q+8*(M_He+Mn)*(M_He-M_f)*EmaxW+...
        16*Mn*M_f*EmaxW);
    C=4*M_He^2*Q^2+8*M_He*(M_He-M_f)*Q*EmaxW+4*(M_He-M_f)^2*EmaxW^2;
    Enmax=(-B+sqrt(B^2-4*A*C))/(2*A); % [eV]
    dEn=(Enmax-Enmin)/(Sdim(i)-1); % The bin width of the spectra
    dEnres=Enbroadening*dEn; % Spectral resolution of the measurements
    Envec=(Enmin:dEn:Enmax); % The bin values of the spectra
    % The calculated bin values are stored in the designated vector
    Spoint((Sbin+1):(Sbin+Sdim(i)))=Envec;
    % The center-of-mass energy of the fast ion in [keV]
    E_CM = CM_coef.*M_f.*((vparaforwardgrid-vd).^2+...
            vperpforwardgrid.^2)./Qe/2e3;
    % The rate part of the weight function (eq. 2.51)
    RNES=n_t.*Bosch_Hale_sigma(E_CM,reaction).*10^-31.*...
            sqrt((vparaforwardgrid-vd).^2+vperpforwardgrid.^2);
    RNES(RNES<0)=0; % Negative rates are set to zero
    % For each bin in the spectra, the weight function for the chosen
    % forward grid is calculated using eq. 2.44 and stored as a row in
    % a matrix
    for En=Envec
        Wrow=Wrow+1;
        % The two gyro-angles (eq. 2.47)
        gamma1forward=acos(1./(vperpforwardgrid.*sin(phi)).*...
            (0.5.*(M_He+Mn)./M_f.*sqrt(2.*(En-dEnres./2)./Mn.*Qe)-...
            0.5.*(M_He-M_f)./Mn.*(vparaforwardgrid.^2+...
            vperpforwardgrid.^2)./sqrt(2.*(En-dEnres./2)./Mn.*Qe)-...
            M_He./(M_f.*Mn).*Q.*Qe./sqrt(2.*(En-dEnres./2)./Mn.*Qe)-...
            vparaforwardgrid.*cos(phi)));
        gamma2forward=acos(1./(vperpforwardgrid.*sin(phi)).*...
            (0.5.*(M_He+Mn)./M_f.*sqrt(2.*(En+dEnres./2)./Mn.*Qe)-...
            0.5.*(M_He-M_f)./Mn.*(vparaforwardgrid.^2+...
            vperpforwardgrid.^2)./sqrt(2.*(En+dEnres./2)./Mn.*Qe)-...
            M_He./(M_f.*Mn).*Q.*Qe./sqrt(2.*(En+dEnres./2)./Mn.*Qe)-...
            vparaforwardgrid.*cos(phi)));
        % The probability part of the weight function (eq. 2.45)
        probNES=real(gamma1forward-gamma2forward)./pi;
        % The total weight function (eq. 2.44) times the step sizes
        wfcvvNESforward=probNES.*RNES./dEnres.*...
            dvparaforward.*dvperpforward;
        Wforward(Wrow,:)=reshape(wfcvvNESforward,1,...
            numel(vparaforwardgrid));
    end
elseif (method(i)==5 || method(i)==6) && pspace==1
    % One-step reaction GRS in (E,p)-space
    Ebroadening=1; % Spectral resolution of the measurements divided by
    % the bin width of the spectra
    % The values of the upper/lower limits of the spectra bins
```

```
    % The lower limit of the spectra bins (eq. 2.85)
    Egamin=(sqrt(2*(M_pr-M_f)*c^2*Q*Qe+(M_pr-M_f)^2*c^4)-(M_pr-M_f)*...
        c^2)/Qe; % [eV]
    syms x
    % The upper limit of the spectra bins (eq. 2.64 solution)
    Egamax=double(solve(Emax*Qe==0.5*M_f*(sqrt(M_pr/M_f*(x^2/...
        ((M_pr-M_f)^2*c^2)+2*(x-Q*Qe)/(M_pr-M_f)))-...
        x/((M_pr-M_f)*c))^2,x))/Qe; % [eV]
    dEga=(Egamax-Egamin)/(Sdim(i)-1); % The bin width of the spectra
    dEgares=Ebroadening*dEga; % Spectral resolution of the measurements
    Egavec=(Egamin:dEga:Egamax); % The bin values of the spectra
    % The calculated bin values are stored in the designated vector
    Spoint((Sbin+1):(Sbin+Sdim(i)))=Egavec;
    % The values of vpa and vpe corresponding to the chosen
    % (E,p)-grid are calculated (eq. 2.3)
    vpara=pforwardgrid.*sqrt(2.*Eforwardgrid.*Qe./M_f);
    vperp=sqrt(1-pforwardgrid.^2).*sqrt(2.*Eforwardgrid.*Qe./M_f);
    % The center-of-mass energy of the fast ion in [keV]
    E_CM=CM_coef.*M_f.*((vpara-vd).^2+vperp.^2)./Qe/2e3;
    % The rate part of the weight function (eq. 2.51)
    RGRS=n_t.*Bosch_Hale_sigma(E_CM,reaction).*10^-31.*...
        sqrt((vpara-vd).^2+vperp.^2);
    RGRS(RGRS<0)=0; % Negative rates are set to zero
    % For each bin in the spectra, the weight function for the chosen
    % forward grid is calculated using eq. 2.78 and stored as a row in
    % a matrix
    for Ega=Egavec
        Wrow=Wrow+1;
        % The two gyro-angles (eq. 2.82)
        gamma1forward=acos(1./(vperp.*sin(phi)).*((M_f-M_pr).*c./...
            (2.*(Ega-dEgares./2).*Qe).*(vpara.^2+vperp.^2)+...
            (Ega-dEgares./2).*Qe./(2.*M_f.*c)+M_pr.*c.*...
            (Ega-dEgares./2-Q)./(M_f.*(Ega-dEgares./2))-...
            vpara.*cos(phi)));
        gamma2forward=acos(1./(vperp.*sin(phi)).*((M_f-M_pr).*c./...
            (2.*(Ega+dEgares./2).*Qe).*(vpara.^2+vperp.^2)+...
            (Ega+dEgares./2).*Qe./(2.*M_f.*c)+M_pr.*c.*...
            (Ega+dEgares./2-Q)./(M_f.*(Ega+dEgares./2))-...
            vpara.*cos(phi)));
        % The probability part of the weight function (eq. 2.79)
        probGRS=real(gamma1forward-gamma2forward)./pi;
        % The total weight function (eq. 2.78) times the step sizes
        wfcEpGRSforward=probGRS.*RGRS./dEgares.*...
            dEforward.*dpforward;
        Wforward(Wrow,:)=reshape(wfcEpGRSforward,1,...
            numel(Eforwardgrid));
    end
elseif (method(i)==5 || method(i)==6) && pspace==2
    % One-step reaction GRS in (vpa,vpe)-space
    Ebroadening=1; % Spectral resolution of the measurements divided by
    % the bin width of the spectra
    % The values of the upper/lower limits of the spectra bins
    % The lower limit of the spectra bins (eq. 2.85)
    Egamin=(sqrt(2*(M_pr-M_f)*c^2*Q*Qe+(M_pr-M_f)^2*c^4)-...
```

```
        (M_pr-M_f)*c^2)/Qe; % [eV]
    syms x
    % The largest possible fast ion energy for the considered region of
    % velocity-space
    EmaxW=0.5*M_f*(max(abs(vparatomo))^2+vperpmax^2);
    % The upper limit of the spectra bins (eq. 2.64 solution)
    Egamax=double(solve(EmaxW==0.5*M_f*(sqrt(M_pr/M_f*(x^2/...
        ((M_pr-M_f)^2*c^2)+2*(x-Q*Qe)/(M_pr-M_f)))-x/((M_pr-...
        M_f)*c))^2,x))/Qe; % [eV]
    dEga=(Egamax-Egamin)/(Sdim(i)-1); % The bin width of the spectra
    dEgares=Ebroadening*dEga; % Spectral resolution of the measurements
    Egavec=(Egamin:dEga:Egamax); % The bin values of the spectra
    % The calculated bin values are stored in the designated vector
    Spoint((Sbin+1):(Sbin+Sdim(i)))=Egavec;
    % The center-of-mass energy of the fast ion in [keV]
    E_CM = CM_coef.*M_f.*((vparaforwardgrid-vd).^2+...
        vperpforwardgrid.^2)./Qe/2e3;
    % The rate part of the weight function (eq. 2.51)
    RGRS=n_t.*Bosch_Hale_sigma(E_CM,reaction).*10^-31.*...
        sqrt((vparaforwardgrid-vd).^2+vperpforwardgrid.^2);
    RGRS(RGRS<0)=0; % Negative rates are set to zero
    % For each bin in the spectra, the weight function for the chosen
    % forward grid is calculated using eq. 2.78 and stored as a row in
    % a matrix
    for Ega=Egavec
        Wrow=Wrow+1;
        % The two gyro-angles (eq. 2.81)
        gamma1forward=acos(1./(vperpforwardgrid.*sin(phi)).*((M_f-...
            M_pr).*c./(2.*(Ega-dEgares./2).*Qe).*(...
            vparaforwardgrid.^2+vperpforwardgrid.^2)+(Ega-dEgares./...
            2).*Qe./(2.*M_f.*c)+M_pr.*c.*(Ega-dEgares./2-Q)./(M_f.*...
            (Ega-dEgares./2))-vparaforwardgrid.*cos(phi)));
        gamma2forward=acos(1./(vperpforwardgrid.*sin(phi)).*((M_f-...
            M_pr).*c./(2.*(Ega+dEgares./2).*Qe).*(...
            vparaforwardgrid.^2+vperpforwardgrid.^2)+(Ega+dEgares./...
            2).*Qe./(2.*M_f.*c)+M_pr.*c.*(Ega+dEgares./2-Q)./(M_f.*...
            (Ega+dEgares./2))-vparaforwardgrid.*cos(phi)));
        % The probability part of the weight function (eq. 2.79)
        probGRS=real(gamma1forward-gamma2forward)./pi;
        % The total weight function (eq. 2.78) times the step sizes
        wfcvvGRSforward=probGRS.*RGRS./dEgares.*...
            dvparaforward.*dvperpforward;
        Wforward(Wrow,:)=reshape(wfcvvGRSforward,1,...
            numel(vparaforwardgrid));
    end
elseif method(i)==7 && pspace==1 % FIDA in (E,p)-space
    lambdabroadening=1; % Spectral resolution of the measurements
    % divided by the bin width of the spectra
    % The values of vpa and vpe corresponding to the chosen (E,p)-grid
    % are calculated (eq. 2.3)
    vpara=pforwardgrid.*sqrt(2.*Eforwardgrid.*Qe./Mi);
    vperp=sqrt(1-pforwardgrid.^2).*sqrt(2.*Eforwardgrid.*Qe./Mi);
    % The values of the upper/lower limits of the spectra bins
    if phi <= pi/2
```

```
    % The lower limit (eq. 2.38)
    lambdamin = (lambda0+sl(1)*max(max(vperp))*B)*(1+1/c*(min(...
        min(vpara))*cos(phi)-max(max(vperp))*sin(phi))); % [m]
    % The upper limit (eq. 2.39)
    lambdamax = (lambda0+sl(15)*max(max(vperp)*B))*(1+1/c*(max(...
        max(vpara))*cos(phi)+max(max(vperp))*sin(phi))); % [m]
elseif phi > pi/2
    % The lower limit (eq. 2.40)
    lambdamin = (lambda0+sl(1)*max(max(vperp))*B)*(1+1/c*(max(...
        max(vpara))*cos(phi)-max(max(vperp))*sin(phi))); % [m]
    % The upper limit (eq. 2.41)
    lambdamax = (lambda0+sl(15)*max(max(vperp))*B)*(1+1/c*(min(...
        min(vpara))*cos(phi)+max(max(vperp))*sin(phi))); % [m]
end
% The bin width of the spectra
dlambda=(lambdamax-lambdamin)/(Sdim(i)-1);
% Spectral resolution of the measurements
dlambdares=lambdabroadening*dlambda;
% The bin values of the spectra
lambdavec=(lambdamin:dlambda:lambdamax);
% The calculated bin values are stored in the designated vector
Spoint((Sbin+1):(Sbin+Sdim(i)))=lambdavec;
% The ad-hoc model for the FIDA intensity function R
RFIDA = Rni.*sqrt(Eforwardgrid./(pi.*RTpara.*RTperp.^2.*Qe.^...
    2)).*exp(-(pforwardgrid.^2.*Eforwardgrid.*Qe+0.5.*Mi.*...
    Rvparadrift.^2-Rvparadrift.*pforwardgrid.*sqrt(2.*Mi.*...
    Eforwardgrid.*Qe))/(RTpara.*Qe)-(1-pforwardgrid.^2).*...
    Eforwardgrid./RTperp).*Qe;
% For each bin in the spectra, the weight function for the chosen
% forward grid is calculated using eq. 2.26 and stored as a row in
% a matrix
for lambda=lambdavec
    Wrow=Wrow+1;
    probFIDA=zeros(size(Eforwardgrid));
    for l=1:15 % sum over the 15 Stark lines
        % The two gyro-angles (eq. 2.31)
        gamma1=acos(1./(vperp.*sin(phi)).*(c.*((lambda-...
            dlambdares./2)./(lambda0+sl(l).*vperp.*B)-1)-...
            vpara.*cos(phi)));
        gamma2=acos(1./(vperp.*sin(phi)).*(c.*((lambda+...
            dlambdares./2)./(lambda0+sl(l).*vperp.*B)-1)-...
            vpara.*cos(phi)));
        % The probability part of the weight function (eq. 2.37)
        probFIDA=C_hat(l).*(real(gamma1-gamma2)/pi+...
            Stark_sign(l).*sin(phi)^2/2.*(real(gamma1-gamma2)/...
            pi-(sin(2.*real(gamma1))-sin(2.*real(gamma2)))/...
            (2*pi))+2.*Eforwardgrid./E0.*(1-pforwardgrid.^2).*...
            cos(gamma_bar).*(sin(real(gamma1))-sin(real(gamma2))...
            +Stark_sign(l).*sin(phi)^2/3.*(sin(real(gamma1)).^3-...
            sin(real(gamma2)).^3)))+probFIDA;
    end
    % The total weight function (eq. 2.26) times the step sizes
    wfcEpFIDAforward=probFIDA.*RFIDA.*dEforward.*dpforward./...
        dlambdares;
```

```
        Wforward(Wrow,:)=reshape(wfcEpFIDAforward,1,...
            numel(Eforwardgrid));
    end
elseif method(i) ==7 && pspace ==2 % FIDA in (vpa,vpe)-space
    lambdabroadening=1; % Spectral resolution of the measurements
    % divided by the bin width of the spectra
    % The value of vpe0 corresponding to the chosen E0-value
    % is calculated
    vperp0=sqrt(2*E0/Mi*Qe);
    % The values of the upper/lower limits of the spectra bins
    if phi <= pi/2
        % The lower limit (eq. 2.38)
        lambdamin = (lambda0+sl(1)*vperpmax*B)*(1+1/c*(vparamin*...
            cos(phi)-vperpmax*sin(phi))); % [m]
        % The upper limit (eq. 2.39)
        lambdamax = (lambda0+sl(15)*vperpmax*B)*(1+1/c*(vparamax*...
            cos(phi)+vperpmax*sin(phi))); % [m]
    elseif phi > pi/2
        % The lower limit (eq. 2.40)
        lambdamin = (lambda0+sl(1)*vperpmax*B)*(1+1/c*(vparamax*...
            cos(phi)-vperpmax*sin(phi))); % [m]
        % The upper limit (eq. 2.41)
        lambdamax = (lambda0+sl(15)*vperpmax*B)*(1+1/c*(vparamin*...
            cos(phi)+vperpmax*sin(phi))); % [m]
    end
    % The bin width of the spectra
    dlambda=(lambdamax-lambdamin)/(Sdim(i)-1);
    % Spectral resolution of the measurements
    dlambdares=lambdabroadening*dlambda;
    % The bin values of the spectra
    lambdavec=(lambdamin:dlambda:lambdamax);
    % The calculated bin values are stored in the designated vector
    Spoint((Sbin+1):(Sbin+Sdim(i)))=lambdavec;
    % The ad-hoc model for the FIDA intensity function R
    vthpara=sqrt(2*RTpara*Qe/Mi);
    vthperp=sqrt(2*RTperp*Qe/Mi);
    RFIDA = Rni./(pi.^(3/2).*vthpara.*vthperp.^2).*...
        exp(-((vparaforwardgrid-Rvparadrift)./vthpara).^2-...
        (vperpforwardgrid./vthperp).^2).*2.*pi.*vperpforwardgrid;
    % For each bin in the spectra, the weight function for the chosen
    % forward grid is calculated using eq. 2.26 and stored as a row in
    % a matrix
    for lambda=lambdavec
        Wrow=Wrow+1;
        probFIDA=zeros(size(vparaforwardgrid));
        for l=1:15 % Sum over the 15 Stark lines
            % The two gyro-angles (eq. 2.31)
            gamma1=acos(1./(vperpforwardgrid.*sin(phi)).*(c.*...
                ((lambda-dlambdares./2)./(lambda0+sl(l).*...
                vperpforwardgrid.*B)-1)-vparaforwardgrid.*cos(phi)));
            gamma2=acos(1./(vperpforwardgrid.*sin(phi)).*(c.*...
                ((lambda+dlambdares./2)./(lambda0+sl(l).*...
                vperpforwardgrid.*B)-1)-vparaforwardgrid.*cos(phi)));
            % The probability part of the weight function (eq. 2.37)
```

```
            probFIDA=C_hat(l).*(real(gamma1-gamma2)/pi+...
                Stark_sign(l).*sin(phi)^2/2.*(real(gamma1-gamma2)/...
                pi-(sin(2.*real(gamma1))-sin(2.*real(gamma2)))/...
                (2*pi))+2.*vperpforwardgrid.^2./vperp0^2.*...
                cos(gamma_bar).*(sin(real(gamma1))-sin(real(gamma2))...
                +Stark_sign(l).*sin(phi)^2/3.*(sin(real(gamma1)).^3-...
                sin(real(gamma2)).^3)))+probFIDA;
        end
        % The total weight function (eq. 2.26) times the step sizes
        wfcvvFIDAforward=probFIDA.*RFIDA.*dvparaforward.*...
            dvperpforward./dlambdares;
        Wforward(Wrow,:)=reshape(wfcvvFIDAforward,1,...
            numel(vparaforwardgrid));
    end
elseif method(i) ==8 && pspace ==1
    % Two-step reaction GRS in (E,p)-space
    Ebroadening=1; % Spectral resolution of the measurements divided by
    % the bin width of the spectra
    % The values of the upper/lower limits of the spectra bins
    Egamin=Ega0-60e3; % [eV] The lower limit of the spectra bins
    Egamax=Ega0+60e3; % [eV] The upper limit of the spectra bins
    dEga=(Egamax-Egamin)/(Sdim(i)-1); % The bin width of the spectra
    dEgares=Ebroadening*dEga; % Spectral resolution of the measurements
    Egavec=(Egamin:dEga:Egamax); % The bin values of the spectra
    % The calculated bin values are stored in the designated vector
    Spoint((Sbin+1):(Sbin+Sdim(i)))=Egavec;
    % The values of vpa, vpe and vtot corresponding to the chosen
    % (E,p)-grid are calculated (eq. 2.3)
    vpara=pforwardgrid.*sqrt(2.*Eforwardgrid.*Qe./(4*Mn));
    vperp=sqrt(1-pforwardgrid.^2).*sqrt(2.*Eforwardgrid.*Qe./(4*Mn));
    vtotforwardgrid=sqrt(2.*Eforwardgrid.*Qe./(4*Mn));
    % The rate part of the weight function (supplied code)
    SIGMA=zeros(size(Eforwardgrid));
    for k =1:length(pforward)
        for j=1:length(Eforward)
            [~,index]=min(abs(Esigma-Eforwardgrid(k,j)/1e6));
            SIGMA(k,j)=sigma(index);
        end
    end
    SIGMAs=medfilt2(SIGMA);
    RGRS=n_Be*SIGMAs.*sqrt((vpara-vd).^2+vperp.^2)*10^(-31);
    % For each bin in the spectra, the weight function for the chosen
    % forward grid is calculated using eq. 2.88 and stored as a row in
    % a matrix
    for Ega=Egavec
        Wrow=Wrow+1;
        prob2stepGRS=zeros(size(vtotforwardgrid));
        % The integrals are calculated as sums
        for beta=betaloop
            for zeta=zetaloop
                % The two values of u_C (eq. 2.90)
                uC1=((Ega-dEga/2)/Ega0-1)*c;
                uC2=((Ega+dEga/2)/Ega0-1)*c;
                % The radicands of eq. 2.91 for the two values of u_C
```

```
radicand1=(sin(beta)^2*cos(zeta)^2*(64*uC1^2*...
    vtotforwardgrid.^2*cos(beta)^2*(cos(beta)^2+...
    sin(beta)^2*cos(zeta)^2)-(cos(beta)^2*(Qdot*Qe/...
    (6*Mn)-vtotforwardgrid.^2)-13*uC1^2).^2))/(64*...
    uC1^2*cos(beta)^2*(cos(beta)^2+sin(beta)^2*...
    cos(zeta)^2)^2);
radicand2=(sin(beta)^2*cos(zeta)^2*(64*uC2^2*...
    vtotforwardgrid.^2*cos(beta)^2*(cos(beta)^2+...
    sin(beta)^2*cos(zeta)^2)-(cos(beta)^2*(Qdot*Qe/...
    (6*Mn)-vtotforwardgrid.^2)-13*uC2^2).^2))/(64*...
    uC2^2*cos(beta)^2*(cos(beta)^2+sin(beta)^2*...
    cos(zeta)^2)^2);

%Check for and exclude negative values of the radicands
r1positive=ones(size(radicand1));
r1positive(radicand1<0)=0;
r2positive=ones(size(radicand2));
r2positive(radicand2<0)=0;

% The four possible values of u_alpha (eq. 2.91)
ua1plus=real((13*uC1^2+cos(beta)^2*(...
    vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC1*...
    (cos(beta)^2+sin(beta)^2*cos(zeta)^2))+...
    sqrt(radicand1));
ua2plus=real((13*uC2^2+cos(beta)^2*(...
    vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC2*...
    (cos(beta)^2+sin(beta)^2*cos(zeta)^2))+...
    sqrt(radicand2));
ua1minus=real((13*uC1^2+cos(beta)^2*(...
    vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC1*...
    (cos(beta)^2+sin(beta)^2*cos(zeta)^2))-...
    sqrt(radicand1));
ua2minus=real((13*uC2^2+cos(beta)^2*(...
    vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC2*...
    (cos(beta)^2+sin(beta)^2*cos(zeta)^2))-...
    sqrt(radicand2));

% Check for and exclude solutions that do not keep eq.
% 2.89
check1plus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-13*...
    uC1^2/cos(beta)^2+8*(ua1plus*uC1+cos(zeta)*...
    sqrt((vtotforwardgrid.^2-ua1plus.^2).*uC1^2*...
    tan(beta)^2));
check1minus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-13*...
    uC1^2/cos(beta)^2+8*(ua1minus*uC1+cos(zeta)*...
    sqrt((vtotforwardgrid.^2-ua1minus.^2).*uC1^2*...
    tan(beta)^2));
check2plus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-13*...
    uC2^2/cos(beta)^2+8*(ua2plus*uC2+cos(zeta)*...
    sqrt((vtotforwardgrid.^2-ua2plus.^2).*uC2^2*...
    tan(beta)^2));
check2minus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-13*...
    uC2^2/cos(beta)^2+8*(ua2minus*uC2+cos(zeta)*...
    sqrt((vtotforwardgrid.^2-ua2minus.^2).*uC2^2*...
```

```
                    tan(beta)^2));
                % We only exclude solutions that do not keep eq. 2.89
                % by more than 9 m^2/s^2
                checkzero1plus=ones(size(check1plus));
                checkzero1plus(log10(abs(check1plus)+1)>1)=0;
                checkzero1minus=ones(size(check1minus));
                checkzero1minus(log10(abs(check1minus)+1)>1)=0;
                checkzero2plus=ones(size(check2plus));
                checkzero2plus(log10(abs(check2plus)+1)>1)=0;
                checkzero2minus=ones(size(check2minus));
                checkzero2minus(log10(abs(check2minus)+1)>1)=0;

                % The checks are combined into acceptance matrices that
                % define the grid points that are acceptable. The other
                % grid points are set to zero.
                acceptplus=r1positive.*r2positive.*checkzero1plus.*...
                    checkzero2plus;
                acceptminus=r1positive.*r2positive.*...
                    checkzero1minus.*checkzero2minus;

                % The four gyro-angles (eq. 2.16)
                gamma1plus=acos((ua1plus-vpara*cos(phi))./...
                    (vperp*sin(phi)));
                gamma2plus=acos((ua2plus-vpara*cos(phi))./...
                    (vperp*sin(phi)));
                gamma1minus=acos((ua1minus-vpara*cos(phi))./...
                    (vperp*sin(phi)));
                gamma2minus=acos((ua2minus-vpara*cos(phi))./...
                    (vperp*sin(phi)));
                % The probability part of the weight function
                % (eq. 2.92) is checked using the acceptance matrices
                prob=abs(real(gamma2plus-gamma1plus))/pi.*...
                    acceptplus+abs(real(gamma2minus-gamma1minus))/...
                    pi.*acceptminus;
                prob2stepGRS=prob2stepGRS+prob;
            end
        end
        prob2stepGRS=prob2stepGRS/(length(betaloop)*length(zetaloop));
        % The total weight function (eq. 2.88) times the step sizes
        wfcEp2stepGRSforward=RGRS.*prob2stepGRS/dEga*dEforward*...
            dpforward;
        Wforward(Wrow,:)=reshape(wfcEp2stepGRSforward,1,...
            numel(Eforwardgrid));
    end
elseif method(i) ==8 && pspace ==2
    % Two-step reaction GRS in (vpa,vpe)-space
    Ebroadening=1; % Spectral resolution of the measurements divided by
    % the bin width of the spectra
    % The values of the upper/lower limits of the spectra bins
    Egamin=Ega0-60e3; % [eV] The lower limit of the spectra bins
    Egamax=Ega0+60e3; % [eV] The upper limit of the spectra bins
    dEga=(Egamax-Egamin)/(Sdim(i)-1); % The bin width of the spectra
    dEgares=Ebroadening*dEga; % Spectral resolution of the measurements
    Egavec=(Egamin:dEga:Egamax); % The bin values of the spectra
```

```
% The calculated bin values are stored in the designated vector
Spoint((Sbin+1):(Sbin+Sdim(i)))=Egavec;
% The total velocities and alpha-particle energies corresponding to
% the chosen grid are calculated (eq. 2.2)
vtotforwardgrid=sqrt(vparaforwardgrid.^2+vperpforwardgrid.^2);
Ealpha=2*Mn*vtotforwardgrid.^2/Qe;
% The rate part of the weight function (supplied code)
SIGMA=zeros(size(vparaforwardgrid));
for k =1:length(vperpforward)
    for j=1:length(vparaforward)
        [~,index]=min(abs(Esigma-Ealpha(k,j)/1e6));
        SIGMA(k,j)=sigma(index);
    end
end
SIGMAs=medfilt2(SIGMA);
RGRS=n_Be*SIGMAs.*sqrt((vparaforwardgrid-vd).^2+...
    vperpforwardgrid.^2)*10^(-31);
% For each bin in the spectra, the weight function for the chosen
% forward grid is calculated using eq. 2.88 and stored as a row in
% a matrix
for Ega=Egavec
    Wrow=Wrow+1;
    prob2stepGRS=zeros(size(vtotforwardgrid));
    % The integrals are calculated as sums
    for beta=betaloop
        for zeta=zetaloop
            % The two values of u_C (eq. 2.90)
            uC1=((Ega-dEga/2)/Ega0-1)*c;
            uC2=((Ega+dEga/2)/Ega0-1)*c;

            % The radicands of eq. 2.91 for the two values of u_C
            radicand1=(sin(beta)^2*cos(zeta)^2*(64*uC1^2*...
                vtotforwardgrid.^2*cos(beta)^2*(cos(beta)^2+...
                sin(beta)^2*cos(zeta)^2)-(cos(beta)^2*(Qdot*Qe/...
                (6*Mn)-vtotforwardgrid.^2)-13*uC1^2).^2))/(64*...
                uC1^2*cos(beta)^2*(cos(beta)^2+sin(beta)^2*...
                cos(zeta)^2)^2);
            radicand2=(sin(beta)^2*cos(zeta)^2*(64*uC2^2*...
                vtotforwardgrid.^2*cos(beta)^2*(cos(beta)^2+...
                sin(beta)^2*cos(zeta)^2)-(cos(beta)^2*(Qdot*Qe/...
                (6*Mn)-vtotforwardgrid.^2)-13*uC2^2).^2))/(64*...
                uC2^2*cos(beta)^2*(cos(beta)^2+sin(beta)^2*...
                cos(zeta)^2)^2);

            % Check for and exclude negative values of the
            % radicands
            r1positive=ones(size(radicand1));
            r1positive(radicand1<0)=0;
            r2positive=ones(size(radicand2));
            r2positive(radicand2<0)=0;

            % The four possible values of u_alpha (eq. 2.91)
            ua1plus=real((13*uC1^2+cos(beta)^2*(...
                vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC1*...
```

```
    (cos(beta)^2+sin(beta)^2*cos(zeta)^2))+...
    sqrt(radicand1));
ua2plus=real((13*uC2^2+cos(beta)^2*(...
    vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC2*...
    (cos(beta)^2+sin(beta)^2*cos(zeta)^2))+...
    sqrt(radicand2));
ua1minus=real((13*uC1^2+cos(beta)^2*(...
    vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC1*...
    (cos(beta)^2+sin(beta)^2*cos(zeta)^2))-...
    sqrt(radicand1));
ua2minus=real((13*uC2^2+cos(beta)^2*(...
    vtotforwardgrid.^2-Qdot*Qe/(6*Mn)))/(8*uC2*...
    (cos(beta)^2+sin(beta)^2*cos(zeta)^2))-...
    sqrt(radicand2));

% Check for and exclude solutions that do not keep eq.
% 2.89
check1plus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-13*...
    uC1^2/cos(beta)^2+8*(ua1plus*uC1+cos(zeta)*...
    sqrt((vtotforwardgrid.^2-ua1plus.^2).*uC1^2*...
    tan(beta)^2));
check1minus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-13*...
    uC1^2/cos(beta)^2+8*(ua1minus*uC1+cos(zeta)*...
    sqrt((vtotforwardgrid.^2-ua1minus.^2).*uC1^2*...
    tan(beta)^2));
check2plus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-13*...
    uC2^2/cos(beta)^2+8*(ua2plus*uC2+cos(zeta)*...
    sqrt((vtotforwardgrid.^2-ua2plus.^2).*uC2^2*...
    tan(beta)^2));
check2minus=Qdot*Qe/(6*Mn)-vtotforwardgrid.^2-13*...
    uC2^2/cos(beta)^2+8*(ua2minus*uC2+cos(zeta)*...
    sqrt((vtotforwardgrid.^2-ua2minus.^2).*uC2^2*...
    tan(beta)^2));
% We only exclude solutions that do not keep eq. 2.89
% by more than 9 m^2/s^2
checkzero1plus=ones(size(check1plus));
checkzero1plus(log10(abs(check1plus)+1)>1)=0;
checkzero1minus=ones(size(check1minus));
checkzero1minus(log10(abs(check1minus)+1)>1)=0;
checkzero2plus=ones(size(check2plus));
checkzero2plus(log10(abs(check2plus)+1)>1)=0;
checkzero2minus=ones(size(check2minus));
checkzero2minus(log10(abs(check2minus)+1)>1)=0;

% The checks are combined into acceptance matrices that
% define the grid points that are acceptable. The other
% grid points are set to zero.
acceptplus=r1positive.*r2positive.*checkzero1plus.*...
    checkzero2plus;
acceptminus=r1positive.*r2positive.*...
    checkzero1minus.*checkzero2minus;

% The four gyro-angles (eq. 2.16)
gamma1plus=acos((ua1plus-vparaforwardgrid*...
```

```
                        cos(phi))./(vperpforwardgrid*sin(phi)));
                    gamma2plus=acos((ua2plus-vparaforwardgrid*...
                        cos(phi))./(vperpforwardgrid*sin(phi)));
                    gamma1minus=acos((ua1minus-vparaforwardgrid*...
                        cos(phi))./(vperpforwardgrid*sin(phi)));
                    gamma2minus=acos((ua2minus-vparaforwardgrid*...
                        cos(phi))./(vperpforwardgrid*sin(phi)));
                    % The probability part of the weight function
                    % (eq. 2.92) is checked using the acceptance matrices
                    prob=abs(real(gamma2plus-gamma1plus))/pi.*...
                        acceptplus+abs(real(gamma2minus-gamma1minus))/...
                        pi.*acceptminus;
                    prob2stepGRS=prob2stepGRS+prob;
                end
            end
            prob2stepGRS=prob2stepGRS/(length(betaloop)*length(zetaloop));
            % The total weight function (eq. 2.88) times the step sizes
            wfcvv2stepGRSforward=RGRS.*prob2stepGRS/dEga*dvparaforward*...
                dvperpforward;
            Wforward(Wrow,:)=reshape(wfcvv2stepGRSforward,1,...
                numel(vparaforwardgrid));
        end
    end
    % Variable used to keep track of the number of measurement points per
    % diagnostic
    Sbin=Sbin+Sdim(i);
end
% Temporary synthetic spectra are calculated from the forward weight
% functions and the known distribution (eq. 2.7)
Stemp=Wforward*reshape(distsimforward,numel(forwardgrid1),1);

% Pre-allocation for speed
Serror=zeros(1,length(phivec));
% For each observation angle, the uncertainty of the spectrum values is
% estimated as 1/10 of the maximum value in the temporary spectrum for
% that observation angle, and the weight functions for that observation
% angle are normalized by this uncertainty
num=0;
for i=1:length(phivec)
    Serror(i)=0.1*max(Stemp((1+num):(num+Sdim(i))));
    Wforward((1+num):(num+Sdim(i)),:)=Wforward((1+num):(num+Sdim(i)),:)/...
        Serror(i);
    num=num+Sdim(i);
end

% Synthetic spectra are calculated from the normalized forward weight
% functions and the known distribution (eq. 2.7)
Sforward=Wforward*reshape(distsimforward,numel(forwardgrid1),1);
% Noise is added to the synthetic spectra
Sforwardnoise=Sforward+max(Sforward).*randn(sum(Sdim),1)...
    .*noiselevel;
% The normalized tomography weight functions and the reconstructed
% distribution functions are obtained by calling the function
% TomoAnalyticRmix.
```

```
switch pspace
    case 1
        [Wtomo,xalpha]=TomoAnalyticRmix(Sforwardnoise,Spoint,Sdim,...
            Serror,Etomo,ptomo,vd,phivec,alpha,Tikhonov,uselsqnonneg,...
            gradient_cutoff,methods,'Ep');
    case 2
        [Wtomo,xalpha]=TomoAnalyticRmix(Sforwardnoise,Spoint,Sdim,...
            Serror,vparatomo,vperptomo,vd,phivec,alpha,Tikhonov,...
            uselsqnonneg,gradient_cutoff,methods,'vv');
end

% Spectra are calculated from the normalized tomography weight
% functions and the known distribution (eq. 2.7)
Stomo=Wtomo*reshape(distsimtomo,numel(tomogrid1),1);

% Pre-allocation for speed
balpha=zeros(sum(Sdim),length(alpha));
% A set of spectra is calculated from the normalized tomography weight
% functions and the various reconstructed distribution functions
for ialpha=1:length(alpha)
    balpha(:,ialpha)=Wtomo*xalpha(:,ialpha); % (eq. 2.7)
end

% The dimensions of the tomography grid
ydim=size(distsimtomo,1);
xdim=size(distsimtomo,2);
% Pre-allocation for speed
Qvec=zeros(1,length(alpha));
% The Q-factor for each reconstruction is calculated (eq. 2.13)
for i=1:length(alpha)
    Qvec(i)=sum(sum((distsimtomo-reshape(xalpha(:,i),ydim,xdim)).^2))/...
        sum(sum(distsimtomo.^2));
end

load('colormap_rel_diff.mat')

% All of the non-temporary spectra are plotted in a single figure
figure(10);clf;hold on;
plot(Sforward)
plot(Sforwardnoise)
plot(Stomo)
for ialpha=1:length(alpha)
    plot(balpha(:,ialpha))
end

% The chosen known distribution is plotted on the forward grid
fig1=figure(20); clf; hold on; box on;set(gca, 'Layer','top');
set(gcf,'DefaultLineLineWidth',2)
imagesc(xaxisforward,yaxisforward,distsimforward)
% caxis([0 280])
axis equal
axis(axes)
colorbar
```

```
colormap(flipud(hot))
set(gca,'Fontsize',16)
xlabel(xlab)
ylabel(ylab)

% The chosen known distribution is plotted on the tomography grid
fig2=figure(30); clf; hold on; box on;set(gca, 'Layer','top');
set(gcf,'DefaultLineLineWidth',2)
imagesc(xaxistomo,yaxistomo,distsimtomo)
% caxis([0 280])
%axis equal
axis(axes)
%pbaspect([1 3 1])
colorbar
colormap(flipud(hot))
set(gca,'Fontsize',16)
xlabel(xlab)
ylabel(ylab)

% The various reconstructed distribution functions are plotted in a set of
% subplots
figure(40); clf; hold on
for i=1:length(alpha)
    subplot(ceil(sqrt(length(alpha))),ceil(sqrt(length(alpha))),i)
    imagesc(xaxistomo,yaxistomo,reshape(xalpha(:,i),ydim,xdim))
    colorbar
    caxis([-max(abs(xalpha(:,i))) max(abs(xalpha(:,i)))])

    % caxis([0 120])
    %axis equal
    axis(axes)
    set(gca,'ydir','normal')
    xlabel(xlab)
    ylabel(ylab)
    set(gca,'Fontsize',13)
end
set(gcf,'colormap',colormap_rel_diff)

% The Q-factors are plotted as a function of the alpha-values
figure(50); clf;
loglog(alpha,Qvec)
xlabel('Regularization parameter, \alpha')
ylabel('Quality factor, Q')

function sigma = Bosch_Hale_sigma(E,reaction)
% This script calculates the NES-related D-D or D-T or the GRS-related
% D-p fusion cross section using the approximation published by Bosch
% and Hale in NF 1992. E is the energy in the center-of-mass frame
% in units of keV. For a stationary target ion, the center-of-mass
% energy is simply the energy of the fast ion in the lab frame times
% the ratio between its mass and the total mass of the fast and
% stationary ions. The cross section is given in units of mb (milli-barns).
% 1 mb = 10^-31 m^2
```

```
if strcmpi(reaction,'NESDD') % Constants for D-D fusion (Table 2.1)
    B_G = 31.3970;
    A1 = 5.3701e4;
    A2 = 3.3027e2;
    A3 = -1.2706e-1;
    A4 = 2.9327e-5;
    A5 = -2.5151e-9;
    B1 = 0;
    B2 = 0;
    B3 = 0;
    B4 = 0;
elseif strcmpi(reaction,'NESDT') % Constants for D-T fusion (Table 2.1)
    B_G = 34.3827;
    A1 = 6.927e4;
    A2 = 7.454e8;
    A3 = 2.050e6;
    A4 = 5.2002e4;
    A5 = 0;
    B1 = 6.38e1;
    B2 = -9.95e-1;
    B3 = 6.981e-5;
    B4 = 1.728e-4;
elseif strcmpi(reaction,'GRSDp') % Constants for p-D fusion (Table 2.1)
    E = E/1e3;
    B_G = 1.07;
    A1 = 8.09e-4;
    A2 = 1.92e-3;
    A3 = 1.21e-2;
    A4 = -5.26e-3;
    A5 = 6.52e-4;
    B1 = 0;
    B2 = 0;
    B3 = 0;
    B4 = 0;
end

S = (A1 + E.*(A2 + E.*(A3 + E.*(A4 + E.*A5)))./...
    (1+E.*(B1+E.*(B2+E.*(B3+E.*B4)))); % (eq. 2.53)
% The cross section is calculated (eq. 2.52)
sigma = S./(E.*exp(B_G./sqrt(E)));

end
```