

right click to select protective marking
@U

[Click to enter Group Name]	Reference:	[Click to enter Reference]			
	Issue:	[Click to enter Issue]			
	Date:	20 August 2007			
<p><i>STARK SPECTRA SIMULATION CODE</i></p> <p><i>[Click to enter Reference]</i></p>					
<p>SUMMARY</p> <p>Description of the full 3D code – written in IDL – that simulates the Stark spectra emitted by neutral beam particles in the MAST tokamak.</p>					
	Name and Organisation	Signature	Date		
Prepared By:	Maarten De Bock Experiments Department				
Checked By:	[Checked by Name]				
Approved By:	[Approved by Name] [Dept]				

right click to select protective marking

TABLE OF CONTENTS

Table of Contents.....	2
1 Introduction.....	3
1.1 MSE at MAST.....	3
1.1.1 MSE Phase I.....	3
1.1.2 MSE Phase II and the Stark Spectrum Simulation Code.....	4
1.2 Why a new code if there is an existing one?.....	5
1.2.1 Bugs in the code.....	5
1.2.2 Looks like a bug but is actually OK.....	6
1.2.3 Possible improvements.....	7
1.2.4 IDL vs. FORTRAN.....	9
2 Description of the stark spectrum simulation code.....	10
2.1 General overview.....	10
2.2 Calculating the Stark spectrum.....	13
2.2.1 The setting file.....	13
2.2.2 The spatial integration points and coordinate systems.....	15
2.2.3 The Equilibrium Model.....	18
2.2.4 The Neutral Beam Model.....	20
2.2.5 Integrating polarisation angles and including depolarisation.....	21
2.3 Applying a filter.....	21
2.3.1 Different filter shapes.....	21
2.3.2 Effect of spread of incident angles.....	21
2.4 Plotting the results.....	21
2.5 Statistical error calculation.....	21
3 Some results of the Stark code.....	23

right click to select protective marking

1 INTRODUCTION

1.1 MSE at MAST

Motional Stark Effect (MSE) is commonly used in tokamaks to measure the pitch angle of the magnetic field. MSE measurements can be used as a constraint in equilibrium reconstruction codes or directly interpreted as a measure for the current (re)distribution. An MSE diagnostic analyses the light emitted by neutral beam particles – usually H_α or D_α . Due to the neutral particle velocity (\mathbf{v}), with respect to the magnetic field (\mathbf{B}), the particle experiences an electric field ($\mathbf{E} = \mathbf{v} \times \mathbf{B}$). This field causes degenerate energy levels of the neutral beam particles to split up, with multiple emitted transition lines as a result. The transition lines can be divided in 2 groups: π and σ , that emit light polarised parallel and perpendicular to the electric field respectively. Examining the polarisation of the emitted π or σ light thus reveals the angle of the electric field and consequently the pitch angle of the magnetic field (assuming the particle velocity is known).¹

In large aspect ratio tokamaks the curvature of the magnetic field is low. This means that over the observation volume – i.e. the intersection between viewing cone and neutral beam – the change in pitch angle is small and can usually be neglected. Moreover, the magnetic field is high, causing the Doppler broadened π and σ components of the H_α -emission to be well separated. One can therefore easily filter out the π or σ component and directly relate the measured polarisation angle to the magnetic pitch angle.

In a spherical tokamak like MAST, however, the magnetic field is low. This causes the π and σ lines to overlap. Also, the curvature of the magnetic field is large. Therefore the pitch angle changes significantly over the observation volume. The overall polarisation angle of the observed light is the result of π and σ light integrated over all polarisation angles in the observation volume. Relating the overall polarisation angle to the local pitch angle is hence more challenging.

1.1.1 MSE Phase I

As a proof-of-principle a 2 channel pilot MSE system was installed at MAST¹. The setup of the system was similar to the MSE diagnostic at the JET tokamak².

A narrow band pass filter (1.0 Angstrom) is used to select a wavelength band with a large fraction of either π or σ dominated polarised light. Two photo-elastic modulators (PEM's), with an azimuthal angle of 45° between them, are driven at a slightly different frequency. They cause a modulation of the phase of the polarised light, which is

¹ Another possibility is to measure the intensity ratio of the π or σ emission lines, which also contains information on the angle of the electric field. This technique is, however, not described in this report.

right click to select protective marking

translated in an intensity modulation by an analysing polariser (with the polarisation angle in between the 2 PEM angles). This intensity signal is detected by an avalanche photo diode (APD).

The different frequency components of the detected signal contain the information on the polarised light. Especially the amplitude ratio of the frequency components with double the PEM frequencies is of interest, as it reveals the polarisation angle.

The results of this 2 channel pilot system were encouraging enough to consider the implementation of a multi channel MSE system for routine operation.

1.1.2 MSE Phase II and the Stark Spectrum Simulation Code

The multi-channel MSE system has the same basic setup of the pilot system:

The PEM's, polariser and collection lens remain the same. The number of channels is

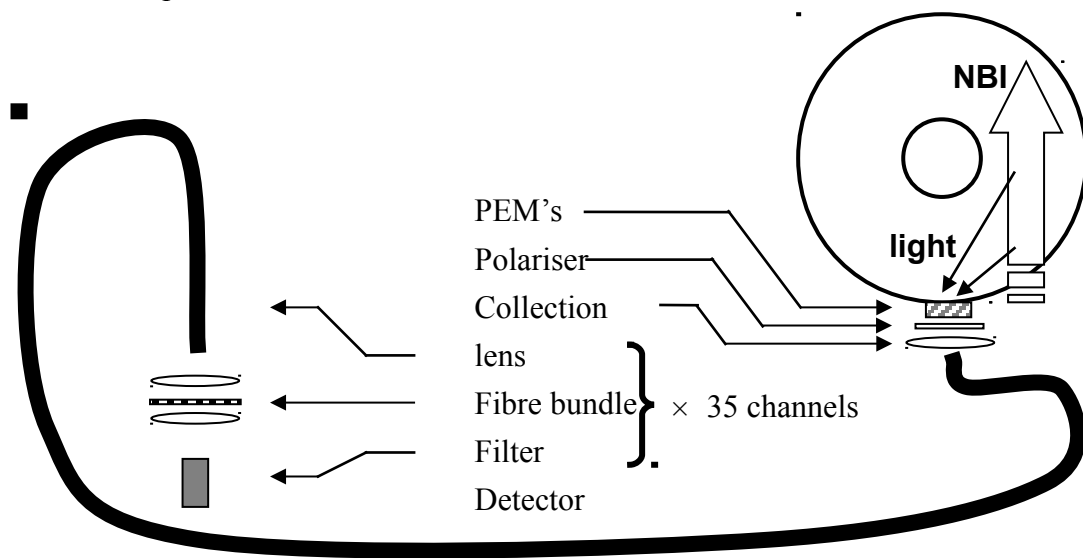


Figure 1 Sketch of the MSE system at the MAST tokamak.

increased to 35 (+5 spare). This means one needs 35 fibre bundles (one for each channel). In the pilot system each fibre bundle consisted of 36 fibres of 400(core)/430(clad) μm . For the multi-channel system the number of fibres per bundle is 19 (again with a diameter of 400(core)/430(clad) μm seems the most likely option).

Apart from 35 fibre bundles one also needs 35 narrow band pass filters. In the pilot system the Stark spectrum was 'tuned' to the filters, by changing the neutral beam voltage and hence the Doppler shift. Because the multi-channel system should be applicable as a routine diagnostic, the filters should be 'tuned' to the Stark spectra and not the other way around. This is done in two ways:

- The set of filters is larger than 35, namely 42, so that for each beam voltage the most appropriate filter can be chosen for each channel.
- Thermal tuning is used for fine tuning the filters. I.e. shifting the peak transition wavelength of the filter to the wavelength that has the largest intensity of polarised light. It also allows to select either the π or σ dominated polarised

right click to select protective marking

light.

Finally 42 detectors are needed, one for each filter. These will, just as for the pilot system, be avalanche photo diodes (APD's). A detector, filter and imaging optics are combined into a filterscope. There is a total of 42 filterscopes, 35 of which are active for a given beam voltage.

To get an idea about what measurements were expected – which polarisation angle, which intensity and which polarised fraction – the pilot system used a FORTRAN code (written by P. Carolan and modified by M. Kuldkepp) to simulate the Stark spectra at MAST. This code was, however, not fully 3D and used a rather crude model for the neutral beam profile and magnetic field.

It was therefore decided to rewrite the FORTRAN code into a full 3D code that could simulate the total MSE spectrum at MAST for a given (EFIT) equilibrium and an advanced neutral beam model. It is this code that is described in this document. Its results are the intensity, polarisation angle and polarised fraction as a function of wavelength. The effect of several types of band pass filters on the 'unfiltered' spectra is included as well.

This code can be used vary the geometry of the fibre bundle, the shape and width of the band pass filter, the position of the collection lens, etc. . These parameters all have an influence on the MSE spectra and hence on the performance of the diagnostic. The code can be used to quantify the optimum settings for these parameters.

Secondly, the simulation code also allows gaining insight on how the measurements of the MSE diagnostic should be interpreted. The latter is not only important for the direct interpretation of the measurements, but is also necessary in order to implement the MSE constraint in equilibrium codes – like EFIT – correctly.

1.2 Why a new code if there is an existing one?

The reason for rewriting the FORTRAN code written by P. Carolan and M. Kuldkepp is that it had some shortcomings. These are reported below:

1.2.1 Bugs in the code

- in SINPUT.f:
 - o function reads the q-axis value from the STARK.INP file, but puts it in the QWALL variable. Same thing for the q-wall value that is put in the QAXIS variable.
 - o Line 75
- in STARK2.f:

right click to select protective marking

- The overall polarisation angle α is calculated as the regular average of the ‘local’ polarisation angles β over all divergence and emission angles and all spatial integration points: $\alpha = (\beta_1 + \beta_2 + \dots + \beta_n)/n$. This regular average is, however, not correct for polarisation angles. Instead α is given as the angle where following polarised intensity function is at a maximum: $I_1 \cos(\alpha - \beta_1)^2 + I_2 \cos(\alpha - \beta_2)^2 + \dots + I_n \cos(\alpha - \beta_n)^2$, where I_i is the polarised intensity of the light with polarisation angle β_i . The analytical solution of the weighted average for 2 polarisation angles β_1 and β_2 with polarised intensities I_1 and I_2 is: $\alpha = \beta_1 + \frac{1}{2} \text{atan}\left(\frac{\sin(2[\beta_2 - \beta_1])}{I_1/I_2 + \cos(2[\beta_2 - \beta_1])}\right)$. In section 2.2.5 this weighted average is explained in more detail.
- Lines 812,813,906
- Further on it gets even worse as the alpha is multiplied by RADDEG multiple times (the first time is OK: radians converted into degrees, but the consequent times are nonsense)
- Lines 941 (is OK),966,991,1015

1.2.2 Looks like a bug but is actually OK

- in STARK2.f:
 - the transformation from the beam coordinate system to the tokamak coordinate system (and also the transformation from the collection lens coordinate system to the tokamak coordinate system), is done by only 2 rotations. In 3D, however, one expects 3 rotations to get from one coordinate system to the other. In the code however the beam and collection lens coordinate systems are not chosen arbitrary. In both systems the Z-axis is chosen such that the first rotation (around the Z-axis) immediately transforms the Y-axis of the beam/collection lens system into the Y-axis of the tokamak system. The second rotation (around the Y-axis) puts the X-axis into the right position, and a rotation around the X-axis is no-longer necessary!
 - So, despite the fact that it looks as if there is one rotation too little, thanks to a clever chosen Z-axis the coordinate transformation by 2 rotations is correct!
 - Lines 247-260 and 473-486

right click to select protective marking

1.2.3 Possible improvements

The main reason for rewriting the code are, however, not the ‘minor’ bugs (they can be debugged), but the fact that – thanks to modern, fast computers – quite a number of improvements can be made to the code, in order to obtain a more accurate result. Implementing these improvements in the existing FORTRAN code required a lot of modifications. It was therefore decided that rebuilding the code from scratch was a better option. A list of points that could be improved upon is given below:

- in TOKFLD.f:
 - In the derivation of the poloidal field B_θ from the q-profile the toroidal field B_ϕ at major radius R is used. This results in a B_θ that depends on R and a pitch angle that is a flux function (which it shouldn't be). It would be better to use B_ϕ at the centre of the flux surface (R_0), hence making B_θ constant over the flux surface (for circular flux surfaces it should be).
 - Even better would be, not to use a ‘build-in’ model for the B-field, but to use ‘real’ equilibria from an equilibrium code like EFIT. The treatment of the equilibrium in the new code is discussed in section 2.2.3.
 - Lines 16-18
- in STARK2.f:
 - The integration points along the line of sight are not really along the line of sight. They lie on the projection of the line of sight in the equatorial plane. So effectively the FORTRAN code is 2D (instead of 3D).
 - An improvement would be to create a set of 3D spatial integration points that resembles the actual image of the fibre bundle at the position of the neutral beam. This is discussed in section 2.2.2.
 - Lines 351,352
- in STARK2.f:
 - For the integration over both beam divergence and collection lens a fixed number of azimuth angles and ‘weight’ separated annuli is used (see Figure 2a).
 - Because the angles ‘captured’ by the collection lens and the angles of the neutral beam particles have a big influence on the (measured/simulated) polarisation angle, it is very important to perform the integration over the collection lens and beam divergence as accurately as possible. The beam divergence is discussed under the next bullet. The integration over the collection lens is discussed here. By using an increasing number of azimuth angles/segments for each annulus, a more homogeneous distribution of the sample points can be achieved (see Figure 2b). This technique is e.g. also

right click to select protective marking

used in ray tracing packages like Z-MAX. (see more in section 2.2.5)

- Lines 165,608,609 and 687,688,689

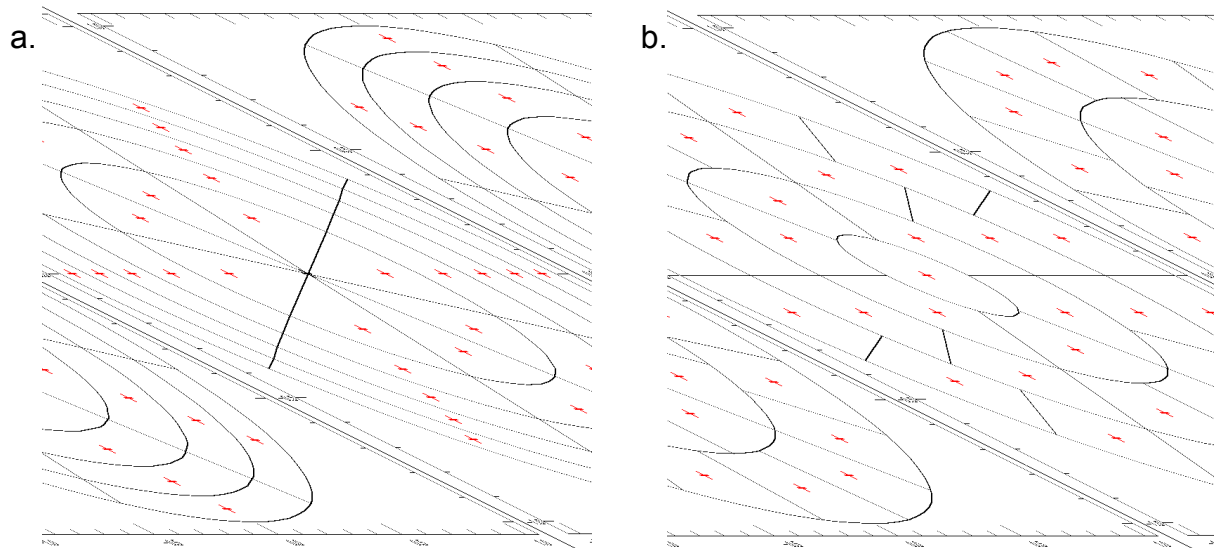


Figure 2 Figure a. represents the weighted sampling as used for both the collection lens as the beam divergence in the FORTRAN code. It has a fixed number of azimuth angles and annuli – 8 and 5 in this case, resulting in 40 sampling points. The area sampled by each point is equal. In figure b. an improved weighted sampling is shown: the number of sampling points is similar (37), the area sampled by each point is still equal, but due to an increasing number of azimuth angles for each annulus the sample points are spread out more homogeneously.

- in STARK2.f:

- In the FORTRAN code the beam is assumed to have an infinite width and the intensity of the light emitted along the beam is equal. Also the average beam velocity direction is assumed to be in the direction of the beam axis. The divergence is assumed to have a square distribution (from 0 to the half-divergence angle) around the average beam velocity direction.
- These are quite large assumptions, so there is quite some room to improve the beam model. First of all a beam code can be used to determine a realistic 3D beam emission profile (which supposedly will be Gaussian transverse to the beam axis and exponentially decaying parallel to the beam axis.). (see section 2.2.4)
- Secondly, due to an extended beam source, consisting of several beamlets with each a geometrical focussing and a divergence, the velocity distribution of the beam particles is definitely not a square – not even a Gaussian – distribution around an average velocity. However, taking into account the beam source geometry and divergence, the beam velocity distribution at a given point in 3D space can be determined. This is shown in Figure 3. (see section 2.2.4 for more information)

right click to select protective marking

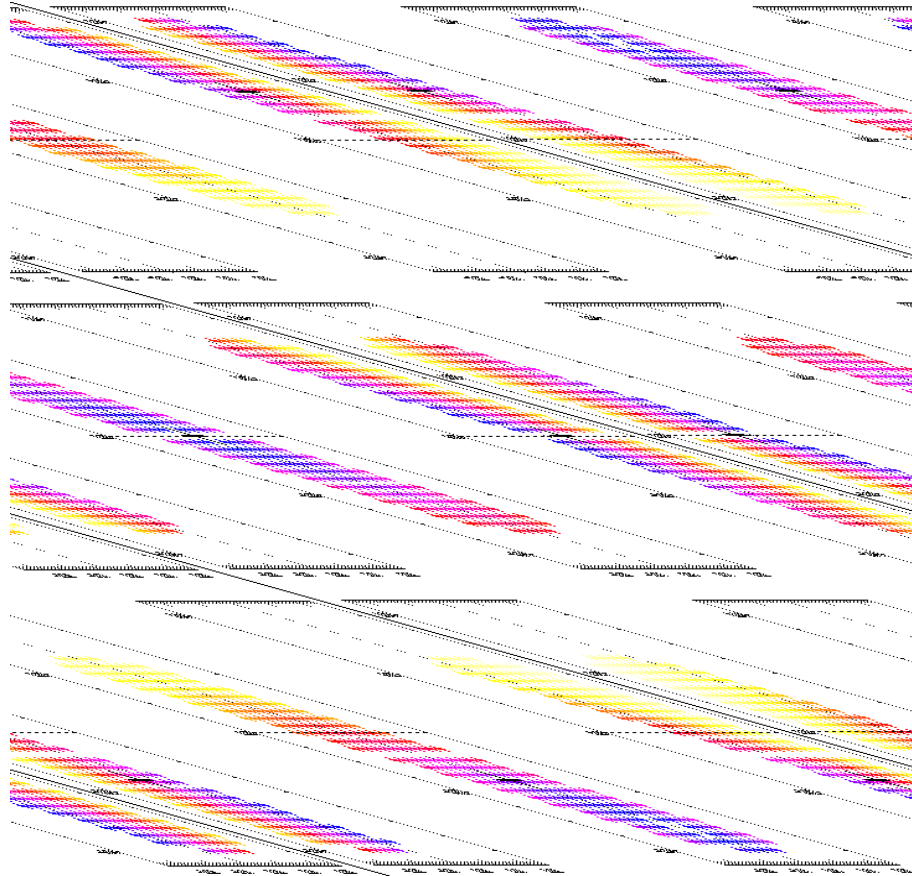


Figure 3 The velocity distribution of the beam particles at 9 spatial points around the beam axis, 6.7m downstream from the beam source, is drawn. The horizontal (geometrical) focus lies 14m from the beam source, the vertical focus 5.17m. The divergence of each beamlet is assumed to be Gaussian, with a half $1/e$ angle of 0.7° . The centre plot shows the velocity distribution on the beam axis, the top row contains the velocity distribution 5cm above the beam axis, the bottom row that 5cm below the beam axis, the left column shows the velocity distribution 5cm left of the beam axis and the right column shows it 5cm right of the beam axis. The distance to the point (0,0) in the plots (where the dashed lines cross) is the sine of the angle between the velocity vector and the beam axis. The colour indicates the probability of the velocity direction (blue being the highest, yellow the lowest probability). The black + indicates the (weighted) average velocity direction.

1.2.4 IDL vs. FORTRAN

The new code is written in IDL and no longer in FORTRAN. Apart from the author's lack of FORTRAN expertise, reasons for doing so are:

- IDL's plotting capabilities
- Easy handling of vectors and matrices

One of the disadvantages of IDL is the fact that it is an interpreted language, not a compiled one. This makes the execution quite slow. It is therefore important to avoid the use of for-loops as much as possible. In a later stage it might be worth 'translating' the IDL code to a compiled language.

right click to select protective marking

2 DESCRIPTION OF THE STARK SPECTRUM SIMULATION CODE

2.1 General overview

The directory structure of the IDL Stark Code is shown in Figure 4. The main directory – ‘stark’ in the figure – contains the [.pro](#) IDL routine files. The subdirectories ‘beam’, ‘equi’, ‘filter’, and ‘physics’ contain data and information about the beam model, the equilibrium, the filters and the physical constants, respectively. The ‘runs’ subdirectory contains the settings and output for several runs of the code. Each subdirectory in ‘runs’ represents a different run of the code. In Figure 4 there’s only one run present: ‘template’.

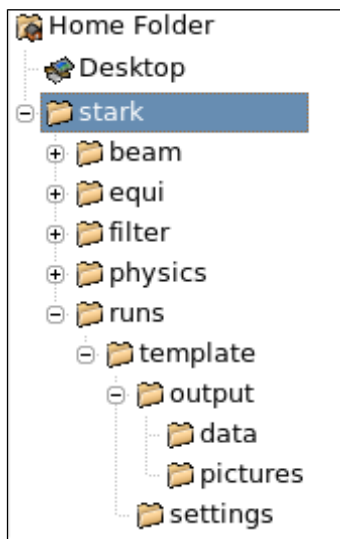


Figure 4 Overview of the IDL Stark code directory structure

Such a ‘run’-directory, like ‘template’, contains a file ‘batch.xml’ – the batch-file with the instructions for running the code – and the subdirectories ‘settings’ - that of contains the setting files for running the code – and ‘output’. The ‘output’ directory is in its turn divided in a ‘data’-directory – contain binary xdr-files with all the output data produced by the code and a ‘pictures’-directory – containing ‘human-readable’ output: eps-figure files and html-files.

The IDL Stark Code is a modular code. The ‘batch.xml’-file determines which modules should be run with which setting, input and output files and in what sequence. There are 6 basic modules. At the moment of this writing 4 of these modules are implemented, one still has to be tested and one still needs to be written. These modules are:

1. [spectrum_calc.pro](#): calculates the 'raw' (unfiltered) spectrum.
2. [spectrum_filter.pro](#): applies a filter to the spectrum.
3. [spectrum_plot.pro](#): plots the geometry, emission intensities and spectra.
4. [spectrum_compare2.pro](#): plots the comparison of 2 sets of spectra.

right click to select protective marking

5. [spectrum_merge.pro](#): merges several spectra into one spectrum (e.g. needed for the triplex setup or overlap with emission spectra of the SW beam). *to be tested*

6. [spectrum_compareN.pro](#): plots the comparison of N sets of spectra. *to be done*

Each of these modules has specific setting, input and output files.

There is a 7th module – [spectrum_noise.pro](#) – that calculates the noise on the detected signal and, from that noise level, the statistical error in the measured polarisation angle. This module is not (yet) integrated in the Stark Spectrum Simulation Code, but works as a stand-alone routine (see section 2.5).

To run the code:

- Go to the upper ‘stark’ directory
- Start IDL
- Type “stark, ‘name of the run’”, e.g.:>> stark, ‘template’

The main program – [stark.pro](#) – will then go through the file ‘batch.xml’ in the run-directory and will execute the instructions it finds there.

A typical ‘batch.xml’-file looks like this:

```
<stark_run description="contains the instructions for running the STARK code">

  <calc  description="calculates the unfiltered spectrum seen from the midplane port"
    settingfile="spectrum_midplane.xml"
    outputfile ="spectrum_midplane.xdr"
  />

  <filter description="Filter the midplane port spectra"
    settingfile="1.0A_Lorentzfilter.xml"
    inputfile  ="spectrum_midplane.xdr"
    outputfile ="spectrum_midplane_filter.xdr"
  />

  <calc  description="calculates the unfiltered spectrum seen from the top port"
    settingfile="spectrum_top.xml"
    outputfile ="spectrum_top.xdr"
  />

  <filter description="Filter the top port spectra"
    settingfile="1.0A_Lorentzfilter.xml"
    inputfile  ="spectrum_top.xdr"
    outputfile ="spectrum_top_filter.xdr"
  />

  <plot  description="Plot the midplane port spectra and intensities"
    settingfile="plot_settings.xml"
    inputfile  ="spectrum_midplane_filter.xdr"
    prefix     ="Midplane port"
  />

  <plot  description="Plot the top port spectra and intensities"
    settingfile="plot_settings.xml"
    inputfile  ="spectrum_top_filter.xdr"
    prefix     ="Top port"
  />

  <compare2  description="Compare the top port with the midplane port spectra"
    settingfile="plot_settings.xml"
    inputfiles =" spectrum_midplane_filter.xdr, spectrum_top_filter.xdr"
    labels     ="Midplane port, Top port"
    prefix     ="Midplane vs. top port"
  />

</stark_run>
```

right click to select protective marking

This batch-file will calculate the Stark spectra as seen from a top port and a midplane port, apply a filter with a Lorentzian band shape and a 1.0 Angstrom FWHM to both, plot both sets of spectra separately and finally compare both sets of spectra by overlaying the plots.

One can see in above sample batch-files that several file-types are used by the code:

- The batch-file is an xml-file. The top tag is named 'stark_run'. The second (which are also the last) level tags have either 'calc', 'filter', 'plot', 'compare2', 'merge' or 'compareN' as name. This of course refers to the different modules. The attributes of the 2nd level tags contain the setting, input and output file names.
- The setting files are xml-files. They have 3 levels of tags: the top tag is named 'stark_settings', the second level tags are there to group several settings, the third level tags contain the actual settings. For the 3rd level tags 2 attributes are required: the type – this can be 'string', 'integer', 'float', ... – and the value – which is a comma separated list of the settings.
- The output files, who are also the input files for the plot routines, are in the binary xdr-format that IDL uses to store (and restore) data.

File-types that do not appear in the shown batch-file example, but are also used by the code:

- The figure files created by the code are in the Encapsulated Postscript format (eps).
- Formatted text output (e.g. tables with data) is saved in HTML files.
- Some numerical data (e.g. atomic data tables for the Stark transition, the filter data ...) is given as numbers in text files. Empty and IDL-style comment-lines (starting with ';') are ignored, the numbers are read in sequentially and interpreted as floating point.

right click to select protective marking

2.2 Calculating the Stark spectrum

This paragraph briefly describes how the Stark spectrum is calculated. The module calculating the spectrum is [spectrum_calc.pro](#) and it is called by adding a `<calc ... />`-tag to the ‘batch.xml’-file. This tag looks like this:

```
<calc    description="calculates the unfiltered spectrum seen from the midplane port"
        settingfile="spectrum_midplane.xml"
        outputfile ="spectrum_midplane.xdr"
/>
```

It has 3 attributes, of which 2 are required (‘settingfile’ and ‘outputfile’). The ‘description’ attribute can be used for describing what kind of spectra will be calculated (which port, how many fibres, how many channels, ...). The ‘settingfile’ attribute contains the filename of the xml-setting file that has all the parameters for calculating the spectra. Its layout is described in the next subsection. The ‘outputfile’ attribute contains the filename of the xdr-file that will contain the result of the calculation. The setting and output filenames should include the extension (‘.xml’ and ‘.xdr’), but not the directory. The code will look for the setting file in the ‘settings’-subdirectory and will write the output file to the ‘output/data’-subdirectory.

2.2.1 The setting file

The setting file for the Stark spectrum calculation has – like all setting files used in the Stark Spectra Simulation Code – 3 levels. The top-level – ‘stark_settings’, a second level grouping related settings and a third level with the actual settings. In Figure 5 the top and second level of a calculation setting file is shown (the third level is collapsed).

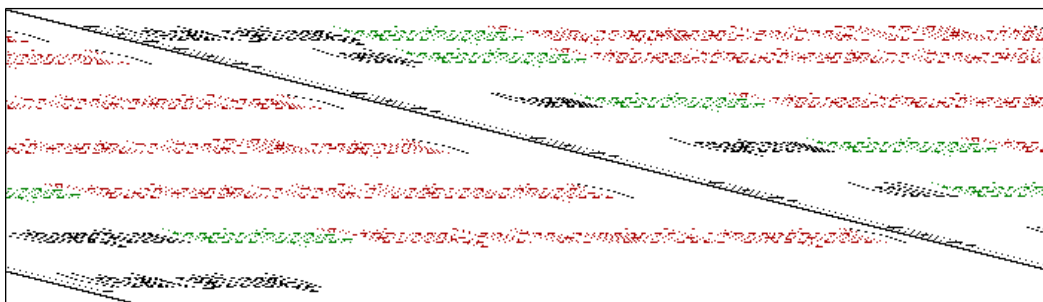
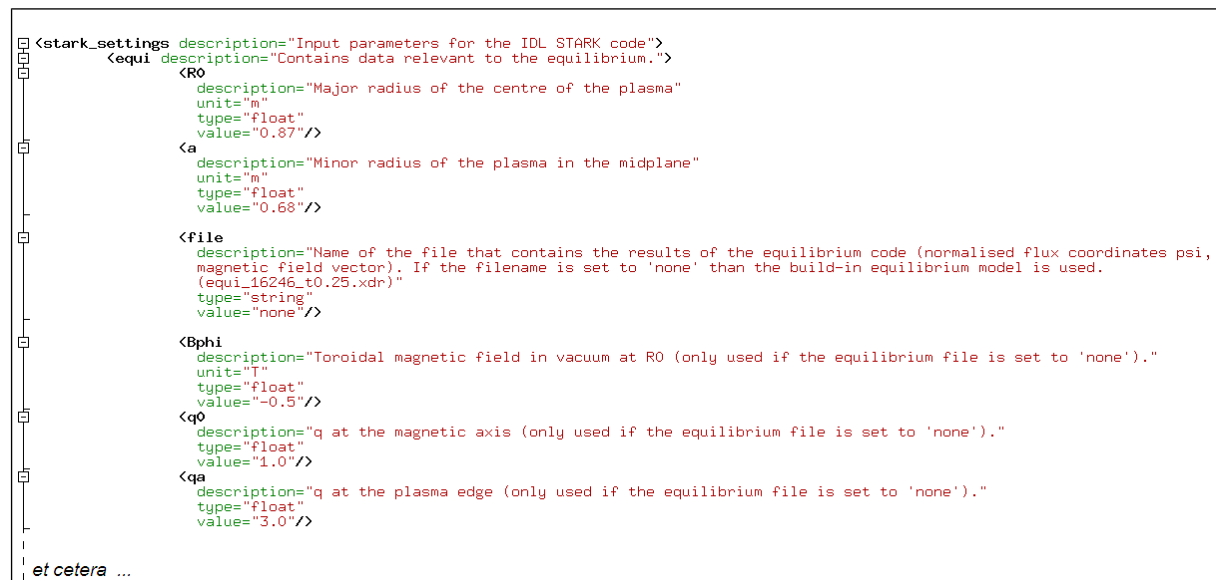


Figure 5 The top and second level of a Stark spectrum calculation setting file (the third level is collapsed). The settings are divided in 5 groups: ‘equi’, ‘beam’, ‘spectrum’, ‘coll’ and ‘integration’ that contain the settings for the used equilibrium, the used beam model, the spectra, the collection optics and the numerical integration, respectively.

In Figure 6 a part of the setting file is shown with the third level expanded. One sees that each setting can have 4 attributes: ‘description’, ‘unit’, ‘type’ and ‘value’. The first two are not required; they are just there for information purposes. The last two – ‘type’ and ‘value’ – are required. The ‘type’ attribute can contain ‘string’, ‘float’, ‘double’, ‘integer’, ‘byte’ or ‘long’. It specifies the data type of the ‘value’ attribute. The ‘value’ attribute contains the actual setting.

right click to select protective marking



```
<stark_settings description="Input parameters for the IDL STARK code">
  <equi description="Contains data relevant to the equilibrium.">
    <R0
      description="Major radius of the centre of the plasma"
      unit="m"
      type="float"
      value="0.87"/>
    <a
      description="Minor radius of the plasma in the midplane"
      unit="m"
      type="float"
      value="0.68"/>
    <file
      description="Name of the file that contains the results of the equilibrium code (normalised flux coordinates psi,
      magnetic field vector). If the filename is set to 'none' then the build-in equilibrium model is used.
      (equi_16246_t0.25.xdr)"
      type="string"
      value="none"/>
    <Bphi
      description="Toroidal magnetic field in vacuum at R0 (only used if the equilibrium file is set to 'none')."
      unit="T"
      type="float"
      value="-0.5"/>
    <q0
      description="q at the magnetic axis (only used if the equilibrium file is set to 'none')."
      type="float"
      value="1.0"/>
    <qa
      description="q at the plasma edge (only used if the equilibrium file is set to 'none')."
      type="float"
      value="3.0"/>
  </equi>
  et cetera ...
</stark_settings>
```

Figure 6 A part of the fully expanded setting file for the Stark spectrum calculation. Each setting is represented by a separate 3rd level xml-tag. The name of the tag is the name of the setting. The ‘description’ and ‘unit’ attributes are there as extra information, the ‘type’ attribute specifies the data type of the setting and the ‘value’ attribute specifies its value.

The settings for the Stark spectrum calculation are divided in 5 sections: ‘equi’, ‘beam’, ‘spectrum’, ‘coll’ and ‘integration’.

- **equi:** contains the settings for the used equilibrium. This is e.g. the name of the file containing the equilibrium and – in case this filename is unspecified - settings for the build-in equilibrium model.
- **beam:** contains the settings for the beam model. I.e. the beam geometry, the beam voltage, a filename containing beam density profiles (and parameters for the build-in beam density model if this file is not specified), et cetera ...
- **spectrum:** contains the total number of channels (even the ones for which no spectrum will be calculated), the major radius of the most outboard channel, the filename containing the atomic Stark data, the size of the wavelength bins, the wavelength ‘tolerance’ (□ the length of the wavelength vector), the size of the polarisation angle bins, the polarisation angle ‘tolerance’, the fraction of emitted light used to determine the spatial resolution, et cetera ...
- **coll:** contains the settings for the collection optics. I.e. the position of the lens, the aperture, the opening angle corresponding with a distance of 1mm on the fibre plate, the core and total diameter of the fibres and the layout of the fibre bundle for one channel.
- **integration:** contains the settings for numerical integration. This includes a list of the channels that need to be calculated, the number of slices along the image of the fibre over the beam (sampling) width, the number of annuli and the increase in segments for the integration over the collection lens, the horizontal and vertical ‘resampling’ of the PINI-beamlets needed for the integration over the beam velocity distribution and the length of the normalised flux coordinates

right click to select protective marking

vector. It also contains a switch that disables the calculation of the spectrum, so that the code only returns the intensity as function of spatial position (makes those runs where only that information is wanted remarkably faster!).

For more information about all the different settings we would like to refer to the setting files themselves. The ‘description’ attributes should give enough information about each setting.

2.2.2 The spatial integration points and coordinate systems

Integration points

The 3D spatial integration grid should cover the area from where the emission is collected as good as possible, with a minimal number of spatial integration points (for reasons of ‘speed’). We choose for a set of integration points that resembles the image of the fibre bundle at the position of the neutral beam. Because the distance between the collection lens and the neutral beam is much larger than the distance between the fibres and the lens, we can approximate the image of a fibre at the position of the neutral beam as a cylinder. As a height of this cylinder we take twice the ‘half sampling width’ W of the neutral beam, as it is defined in ‘beam’ section of the setting file. Typically, two times the beam’s half 1/e width is a good value for W . We divide each cylinder in slices, where the number of slices is defined in the ‘integration’ section of the setting file. As integration points we then take the central points of cylinder slices. The volume each integration point is sampling is given by the volume of the cylinder slice. This is shown in Figure 7.

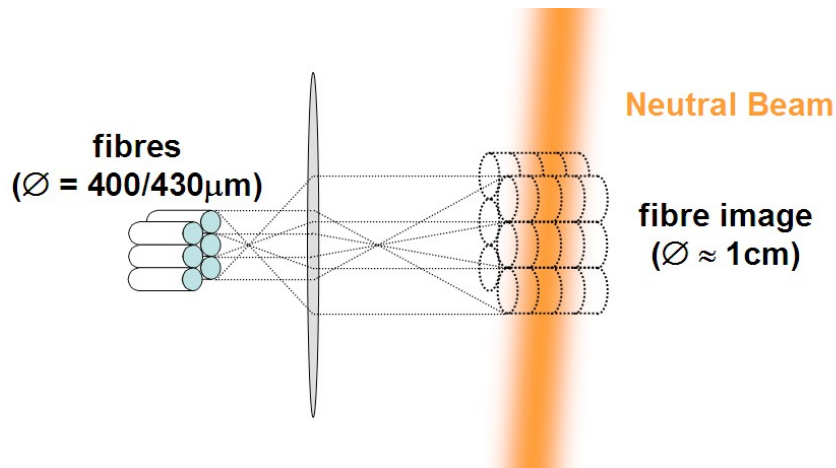


Figure 7 A sketch of the image of a fibre bundle at the position of the neutral beam. Because the distance between the neutral beam and the lens is much larger than the distance between the fibres and the lens, the image of a fibre at the position of the neutral beam can be approximated by a cylinder. This cylinder can be divided into a number of slices. We take the centre of such a cylinder-slice as our integration point – hence we have $\text{\#integration points} = \text{\#fibres} \times \text{\#slices}$ – and its volume as the emitting volume element. Within a fibre bundle the fibres are closest packed.

right click to select protective marking

The total number of integration points per channel is $\#slices \times \#fibres/channel$, where $\#slices$ is given in the ‘integration’ section of the setting file and $\#fibres/channel$ is derived from the fibre bundle layout in the ‘coll’ section of the setting file.

In a Cartesian coordinate system, with the line-of-sight of the channel as x -axis and the crossing of the line-of-sight with the beam axis as origin, the x -coordinates of the integration points are regularly distributed and centred on $x=0$. The distance between the integration points in x -direction is $W/\#slices$, W being the beam sampling with given in the ‘beam’ section of the setting file.

The y - and z -coordinates are centred on $(y=0, z=0)$ and depend on the layout of the fibre bundle and the diameter of the fibre image.

The fibre bundle layout is defined in the ‘coll’ section of the setting file. It is a list where the length of the list is the number of columns and each element gives the number of fibres in that column. A closest packing of the fibres is assumed. E.g. "10, 9" means that one channel consists of a fibre bundle with 19 fibres: 10 in the first column, 9 in the second (see Figure 8(a)).

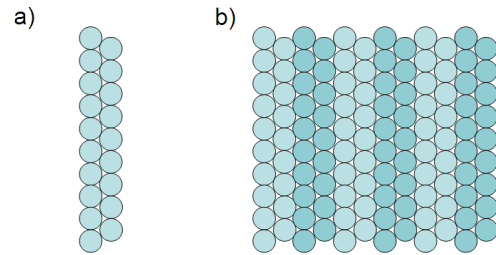


Figure 8 In a) a “10,9” fibre bundle layout is given for one channel. In b) the resulting total layout for 6 channels is shown.

The diameter of the fibre image is calculated using the opening angle corresponding with a distance of 1mm on the fibre plate, the fibre diameter – both given in the ‘coll’ section of the setting file – and the distance to the collection lens. It is different for every fibre and every slice. However, the differences will be small (due to the large overall distance to the collection lens). Therefore it was assumed that the fibre image diameter is constant within one channel and equal to the fibre image diameter on the beam axis.

To convert the xyz -coordinates from this line-of-sight coordinate system to the machine-coordinate system (see below), the machine-coordinates of the point where the line-of-sight crosses the neutral beam axis needs to be calculated. For the outermost channel this depends on the major radius it is looking at (a parameter set in the ‘spectrum’ section of the setting file), the position of the collection lens (given in the ‘coll’ section of the setting file) and the beam axis (defined by the beam duct coordinates and beam angles given in the ‘beam’ section of the setting file). From the outermost channel, we just work our way in, assuming that all channels are closest packed (see Figure 8(b)). Keep in mind that this means that the channels have an equal separation in distance on the fibre plate and in opening angles, but do NOT have an equal separation in major radius.

The calculation of the coordinates of the integration points in the machine-coordinate system is done by the routine [calc_gp.pro](#).

right click to select protective marking

Coordinate systems

In the code there are 3 Cartesian coordinate systems are being used. The main system is the machine coordinate system. This has the vertical axis – bottom to top – as z -axis. The x -axis goes from west to east, the y -axis from south to north. The origin of this coordinate system is the centre of the tokamak.

A second coordinate system is the beam system. In this system the neutral beam axis is the x -axis, with the positive direction defined by the velocity direction of the neutral beam. The y -axis is defined lie within the xy -plane of the machine coordinate system and the direction of the z -axis is defined such that the projection of the beam- z -axis onto the machine- z -axis is positive. The origin of this coordinate system is the beam duct.

The third coordinate system is the emission system. In fact, there is more than one emission system; there is one for every integration point. The x -axis is defined as the emission direction from the emission/integration point towards the collection lens (i.e. the line-of-sight). The y -axis lies in the xy -plane of the machine coordinate system and the z -axis is defined such that the projection on the machine- z -axis is positive. The origin is the emission/integration point.

The 3 types of coordinate systems are indicated in Figure 9. Transformations between the different coordinate systems are performed by the [coordtrans.pro](#) function.

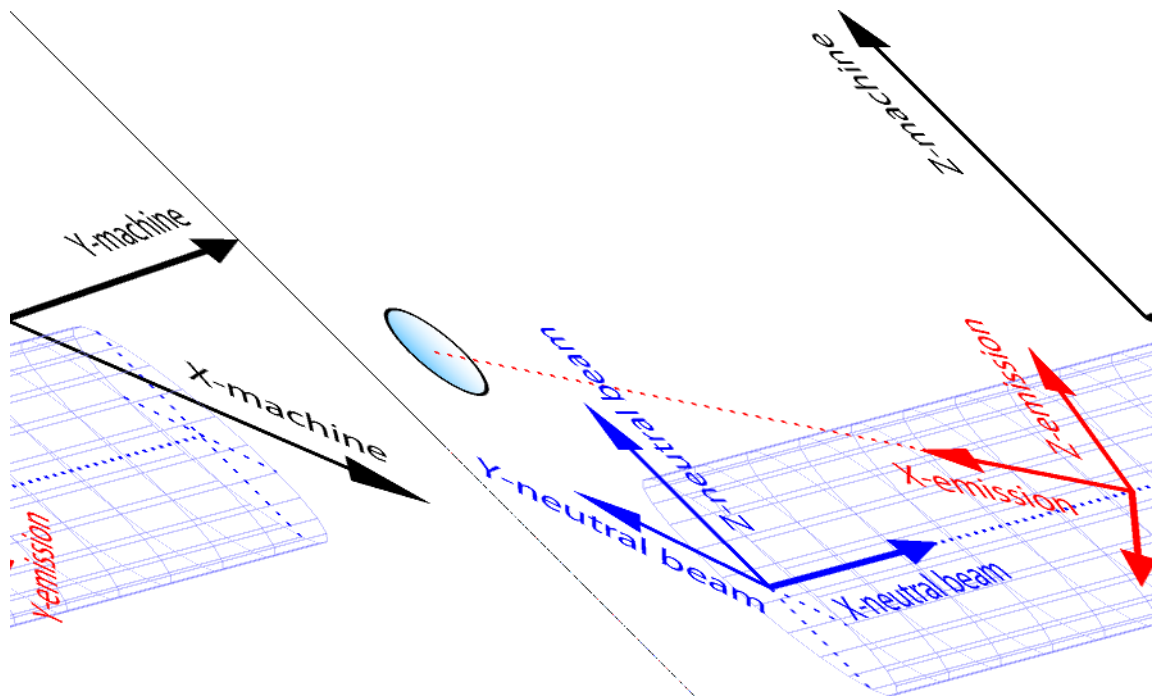


Figure 9 The different types of coordinate systems used in the code are shown. Note that the y -axis of all coordinate systems lie in the same plane: the xy -plane of the machine coordinate system.

right click to select protective marking

2.2.3 The Equilibrium Model

The Stark spectrum is a result of the electric field experienced by the emitting neutral atoms of the neutral beam. The ‘motional’ term of this electric field is given by $\mathbf{v} \times \mathbf{B}$, where \mathbf{v} and \mathbf{B} are the velocity vector of the beam particles and the magnetic field vector, respectively. \mathbf{v} will be discussed in the next section; this section focuses on the magnetic field \mathbf{B} . A second important equilibrium parameter is the normalised flux coordinate ψ . This can give us an idea about how the spatial resolution of the MSE system is affected by flux surface compression.

To find out both the magnetic field vector \mathbf{B} and the normalised flux coordinate ψ at each integration point, we need to use a model for the plasma equilibrium. There are two possibilities for this in the Stark Spectra Simulation Code:

1. Calculate the equilibrium, using the build-in model that assumes elliptical flux surfaces.
2. Read the equilibrium from a xdr-file. These equilibrium-files should be placed in the ‘equi’ directory (see Figure 4). As long as the xdr-file has the right internal structure, any equilibrium code can be used to supply the equilibrium. For MAST, the easiest way to create an equilibrium file is to use the routine [create_equifile.pro](#) that is located in the ‘equi’ directory. It reads the EFIT data for a given shot and time and writes it – in the correct format – to a xdr-equilibrium file.

Build-in equilibrium model

The build-in equilibrium model is used when the ‘file’-setting in the section ‘equi’ of the setting file is set to “none”. If this is the case, \mathbf{B} and ψ will be calculated based on following settings (in the ‘equi’ section):

- R_0 : the centre of the plasma
- a : the plasma radius in the equatorial plane ($z_{\text{machine}}=0$)
- $B_{\phi,0}$: the (vacuum) toroidal, magnetic field at R_0
- q : the q-profile – $q(\psi) = q(0) + [q(a) - q(0)] \cdot \psi^{q_{idx}}$
- B_{par} : the paramagnetic field –
 $B_{par}(\psi) = B_{par}(0) + [B_{par}(a) - B_{par}(0)] \cdot \psi^{B_{par,idx}}$
- S_{shaf} : the Shafranov shift – $\psi = 0 \Leftrightarrow R = R_0 + S_{shaf}$
- κ : plasma elongation – plasma height = $2 a \kappa$

The equilibrium is calculated by the [calc_equi.pro](#) routine. This is done in 2 steps. The first step is to calculate the normalised flux coordinate ψ for each point (x,y,z) – in the machine coordinate system. Because of the toroidal symmetry we can reduce (x,y,z) to (R,z) , with $R = \sqrt{x^2 + y^2}$ the major radius. ψ , and the poloidal angle θ , are then found by solving following set of equations:

right click to select protective marking

$$\begin{cases} R = R_0 + S_{shaf}(1 - \psi^2) + a \cdot \psi \cdot \cos(\theta) \\ z = a \cdot \kappa \cdot \psi \cdot \sin(\theta) \end{cases} \quad (2.2.1)$$

The second step is calculating the \mathbf{B} -vector. The vacuum, toroidal field B_ϕ is simply proportional to $1/R$. To get the total toroidal field the paramagnetic field is added to that: $B_{par}(\psi) = B_{par}(0) + [B_{par}(a) - B_{par}(0)] \cdot \psi^{B_{par,idx}}$. The poloidal field B_θ is approximated by use of the q -profile: $q(\psi) = q(0) + [q(a) - q(0)] \cdot \psi^{q_{idx}}$.

$$\begin{cases} B_\phi(\psi, \theta) = B_{\phi,0} \frac{R_0}{R(\psi, \theta)} + B_{par}(\psi) \\ B_\theta(\psi) = \frac{a \cdot \psi \cdot B_{\phi,0} \cdot R_0}{[R_0 + S_{shaf}(1 - \psi^2)]^2 \cdot q(\psi)} \end{cases} \quad (2.2.1)$$

(Here $R_0 + S_{shaf}(1 - \psi^2)$ is the major radius of the centre of the flux surface, hence $B_{\phi,0} \cdot R_0 / [R_0 + S_{shaf}(1 - \psi^2)]$ is B_ϕ at the centre of the flux surface. And $a \cdot \psi$ is the 'minor radius'.)

In Figure 10(a) the flux surfaces (ψ going from 0.1 up to 1.0) of the build-in equilibrium model are shown.

Read-in equilibrium model

The above build-in equilibrium model is of course a very rough approximation of the real equilibrium. One can however use the equilibrium calculated by a proper equilibrium code within the Stark Spectra Simulation Code. For this a xdr-file, containing the equilibrium information, should be put in the 'equi' directory, and the setting 'file' in the 'equi' section of the setting file should contain the filename of this xdr-file.

The xdr-equilibrium file should contain following variables:

- R: $[#R \times 1]$ - array with the R -coordinates
- Z: $[#z \times 1]$ - array with the z -coordinates
- Bfld: $[3 \times #R \times #z]$ -array with the B_R -, the B_z - and the B_ϕ -components of \mathbf{B} -field at the (R,z) -coordinates.
- Rm: scalar containing the major radius of the magnetic axis
- fluxcoord: $[#R \times #z]$ -array with the normalised flux coordinates ψ at the (R,z) -coordinates.

The routine [read_equi.pro](#) then reads the xdr-equilibrium file and interpolates the flux coordinates and \mathbf{B} -field to from the R - and z -arrays to the R and z -coordinates of the integration points.

For MAST, the easiest way to create a xdr-equilibrium file is to use the [create_equifile.pro](#) routine that is found in the 'equi' directory. The routine has 3 keyword parameters: shotno (of the discharge you want to use the equilibrium from), time (at which you want the equilibrium, in seconds) and filename (of the xdr-file

right click to select protective marking

you're going to save). The routine then uses the MAST IDL-library routine [read_flux.pro](#) to get the equilibrium of the given discharge at the given time and saves it in the correct format to the given xdr-file.

In Figure 10(b) the flux surfaces (ψ going from 0.1 up to 1.0) of the read-in equilibrium (MAST discharge #16246 at $t=0.250$ s) are shown.

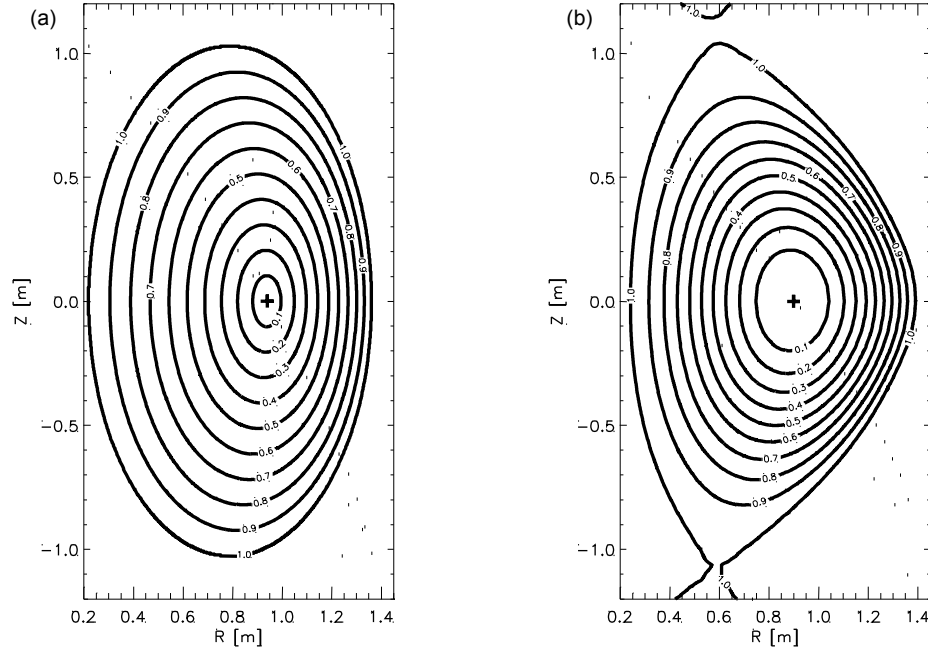


Figure 10 The flux surfaces of the build-in model (a) and a read-in equilibrium (b) (MAST discharge #16246 at $t=0.250$ s) are shown. The magnetic field at R_0 , the Shafranov shift S_{shaf} and the plasma radius a are similar for both sets of equilibria. Nonetheless one notices quite a difference between both equilibria. It is therefore recommended to use ‘read-in’ equilibria created by a dedicated equilibrium code, rather than using the build-in model.

2.2.4 The Neutral Beam Model

There are two aspects of the neutral beam that influence the Stark Spectrum:

- The density-profile of beam particles determines the emission intensity profile in 3D
- The velocity distribution of beam particles determines the Doppler shift, the Stark splitting and the polarisation angle

The neutral beam model used in the Stark code is therefore split in two parts: a model for the beam emission in 3D and a model for the velocity distribution. For the beam emission model there is again the choice between a ‘build-in’ and a ‘read-in’ model. For the velocity distribution only a ‘build-in’ model exists.

Build-in beam emission model

Blabla

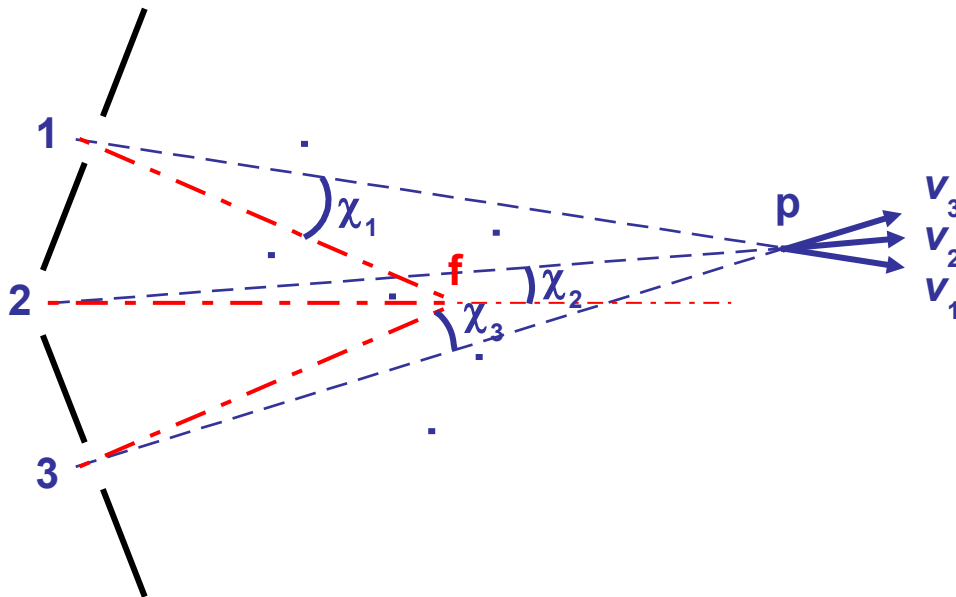
right click to select protective marking

Read-in beam emission model

Blabla

Beam velocity distribution

Blabla



2.2.5 Integrating polarisation angles and including depolarisation

2.3 Applying a filter

2.3.1 Different filter shapes

2.3.2 Effect of spread of incident angles

2.4 Plotting the results

2.5 Statistical error calculation

The calculation of the statistical error is done by the routine [spectrum_noise.pro](#). For the moment this isn't a full module integrated in the code yet, but it is to be in the future. It is run as follows:

```
>> spectrum_noise, 'noise_settings.xml', 'input_file.xdr',  
    'plot_settings.xml', 'prefix', 'pict_dir'
```

The routine will plot the noise level where “polarised fraction $\times \sqrt{\text{total intensity}}$ ” is at a maximum (i.e. at blue shifted π , σ or red shifted π) and the corresponding statistical error in the polarisation angle α . The result will also be saved in a html-file. The ‘plot_settings.xml’ file determines whether the result is plotted to the screen or to an

right click to select protective marking

eps-file. ‘prefix’ and ‘pict_dir’ define the prefix of the html and eps-filenames and the directory where to save these files, respectively.

Below the – analytical – derivation of the statistical error in the polarisation angle α is given:

When the polarisation angle is determined by a PEM system (as it is for MAST), then it is given by

$$\alpha = \frac{1}{2} \text{atan} \left(\frac{I_{2,\omega_2} J_2(A_1)}{I_{2,\omega_1} J_2(A_2)} \right), \quad (2.5.1)$$

Here $J_2(x)$ is the 2nd Bessel function, A_1 and A_2 are the amplitudes of the oscillating retardances of the two PEM’s: $A_1 \cos(\omega_1 t)$ and $A_2 \cos(\omega_2 t)$, respectively. I_{2,ω_1} and I_{2,ω_2} are the (measured) intensities of the frequency components of the detected signal at double the modulation frequency of PEM₁ (ω_1) and PEM₂ (ω_2). These intensities depend on the number of collected photons (N), the polarised fraction pf , the 2nd Bessel function of A_1 or A_2 and the polarisation angle α .

$$\begin{aligned} I_{2,\omega_1} &= \left(\frac{e}{C_{ADP}} \text{amp} \right) \times \left(\frac{J_2(A_1)}{\sqrt{2}} \cdot pf \cdot \cos(2\alpha) \right) \times (QE \cdot N) \\ I_{2,\omega_2} &= \left(\frac{e}{C_{ADP}} \text{amp} \right) \times \left(\frac{J_2(A_2)}{\sqrt{2}} \cdot pf \cdot \sin(2\alpha) \right) \times (QE \cdot N) \end{aligned} \quad (2.5.2)$$

The first factor in (2.5.2) is the proportionality factor that converts the (detected) photons into voltage, with ‘amp’ the amplification, ‘e’ the electron charge and ‘ C_{ADP} ’ the capacity of the detector. The second factor describes the effect of looking into the frequency components $2 \cdot \omega_1$ and $2 \cdot \omega_2$. The last factor is the number of detected photons. I.e. the photons that fall onto the detector in a sampling time t_s times the quantum efficiency ‘QE’.

Considering the detection of photons as a Poisson process, the photon noise level is given by $\sigma_N = \sqrt{QE \cdot N}$. The amplification process in the APD (Avalanche Photo Diode) detector introduces extra noise, known as ‘excess noise’. The noise on the signal then is:

$$\sigma_{2,\omega_1} = \sigma_{2,\omega_2} = \left(\frac{e}{C_{ADP}} \text{amp} \right) \sqrt{QE \cdot N \cdot \text{ExcessNoise}} \quad (2.5.3)$$

This only takes into account the noise caused by the photons, not the noise due to the detectors dark current. One can however look upon the dark current as being caused by ‘dark’ photons: $N_{dark} = I_{dark} \frac{t_s}{e} \frac{1}{QE \cdot \text{ExcessNoise}}$. The total signal noise then becomes:

$$\sigma_{2,\omega_1} = \sigma_{2,\omega_2} = \left(\frac{e}{C_{ADP}} \text{amp} \right) \sqrt{(N + N_{dark}) \cdot QE \cdot \text{ExcessNoise}} \quad (2.5.4)$$

With I_{2,ω_1} and I_{2,ω_2} , and their errors σ_{2,ω_1} and σ_{2,ω_2} , given, we can use the general error

right click to select protective marking

propagation formulas to derive the error in α : σ_α . The two error propagation formulas that we need are:

$$\sigma_{A/B} = \sqrt{\left(\frac{\sigma_A}{A}\right)^2 + \left(\frac{\sigma_B}{B}\right)^2} \frac{A}{B} \quad (2.5.5)$$

$$\sigma_{f(x)} = \frac{df(x)}{dx} \sigma_x \quad (2.5.6)$$

Using (2.5.2) and (2.5.5) one finds that:

$$\begin{aligned} \sigma_{I_{2,\omega_2} J_2(A_1)/I_{2,\omega_1} J_2(A_2)} &= \tan(2\alpha) \sqrt{\frac{1}{(J_2(A_1) \cdot \cos(2\alpha))^2} + \frac{1}{(J_2(A_2) \cdot \sin(2\alpha))^2}} \\ &\times \frac{\sqrt{2 \cdot \text{ExcessNoise} \cdot (N + N_{\text{dark}})}}{pf \cdot N \sqrt{\text{QE}}} \end{aligned} \quad (2.5.7)$$

Using (2.5.1), (2.5.6) and (2.5.7) the error in α can be derived as:

$$\begin{aligned} \sigma_\alpha &= \frac{\tan(2\alpha)}{(1 + \tan^2(2\alpha))} \sqrt{\frac{1}{(J_2(A_1) \cdot \cos(2\alpha))^2} + \frac{1}{(J_2(A_2) \cdot \sin(2\alpha))^2}} \\ &\times \frac{\sqrt{\text{ExcessNoise} \cdot (N + N_{\text{dark}})}}{pf \cdot N \sqrt{2 \cdot \text{QE}}}, \end{aligned} \quad (2.5.8)$$

which does depend on the angle α . However, if A_1 and A_2 are chosen to be equal ($A_1=A_2=A$), then (2.5.8) can be reduced to:

$$\sigma_\alpha = \frac{\sqrt{\text{ExcessNoise}}}{J_2(A) \sqrt{2 \cdot \text{QE}}} \frac{\sqrt{N + N_{\text{dark}}}}{pf \cdot N}, \quad (2.5.9)$$

which does not depend on α .

3 SOME RESULTS OF THE STARK CODE

right click to select protective marking

¹ M. Kuldkepp, M.J. Walsh, et al., **Motional Stark Effect diagnostic pilot experiment for MAST**, Review of Scientific Instruments 77 (2006) 10E905

² N.C. Hawkes, **Design Study of a Motional Stark Effect Diagnostic for JET**, JET-R(96)10