NBASS User Manual

Ken Liao

October 11, 2017

Contents

1 New changes 2
2 Introduction 3
3 Installation 4
4 A sample run 5
5 Input Parameters 8
5.1 `param` 8
5.2 `detector_param` 9
5.3 `beam_param` 10
5.4 `view_param` 11
5.5 `mesh_param` 12
5.6 `equil_param` 13
5.7 `prof_param` 15
6 Program Output 16

1 New changes

Version 1.0: First release.

2 Introduction

This is a user manual for version 1.0 of NBASS and is included with the NBASS program distribution. The author can be contacted via the email `kenliao@physics.utexas.edu`.

NBASS is a synthetic diagnostic for polarization and spectral Motional Stark Effect diagnostics used in tokamak experiments. The code simulates the intensity and polarization of the Balmer H_α emission spectrum of the neutral beam. It takes into account the combined Stark and Zeeman effect, non-statistical beam atom excited state population, and radial electric field. To give accurate broadened line shapes, spectra are numerically integrated over a spatial grid which accounts for the finite emitting volume, finite acceptance area, and beam source area. Small differences in angles and plasma parameters over the grid result in a blur of spectra. Bremsstrahlung and charge exchange recombination emission contributions to the spectrum are also calculated.

An overview of the code as well as an explanation of calculation method can be found in the manuscript to be published in Computer Physics Communications[**?**]. A README file included with the distribution contains instructions for installing the program and running a sample test case.

Each run of the NBASS code is organized into a subdirectory of the `runs` directory. Whenever NBASS is called, the first argument is the name of the run subdirectory. NBASS will look in the subdirectory for a file named `parameters.pro` for the input parameters for the run. E.g. `IDL> nbass,'test'`

To create a new run, make a new subdirectory in the `runs` directory and copy the `parameters.pro` from an existing subdirectory to the new directory and edit the file as necessary. A full list of parameters is given in the Input Parameters section of the manual.

The best way to understand the parameters file is to view the sample parameters file `runs/test/parameters.pro` and look at how the parameters are organized.

3 Installation

The latest version of NBASS can be obtained from githab by logging into github.com and going to
`https://github.com/ut-ifs/nbass`
or searching in github for `ut-ifs/nbass`. Download and extract the archive to any suitable directory. Choose a runs directory to store input and output of runs of the code. By default, the runs directory is in the `'runs'` subdirectory in the program directory.

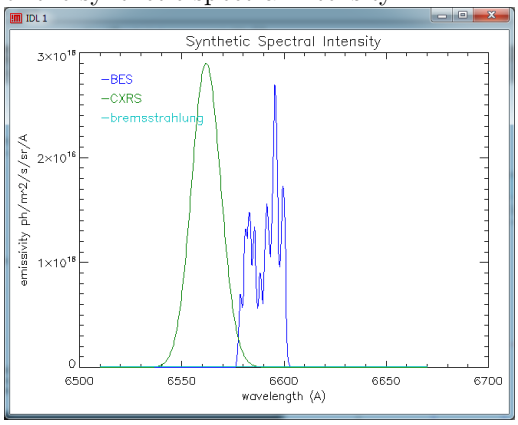
Edit the `'run_nbass'` file. Change the `NBASS_PATH` and `NBASS_RUNS_PATH` variables to contain the install and run paths. Now it should be possible to run the sample case below.

Typically, you will also want to install ALCBEAM since the ALCBEAM output is used by NBASS, but ALCBEAM is not needed to run NBASS if the output file is copied from elsewhere. ALCBEAM can be obtained from github by logging into github.com and going to
`https://github.com/ut-ifs/alcbeam`
or searching in github for `ut-ifs/alcbeam`.

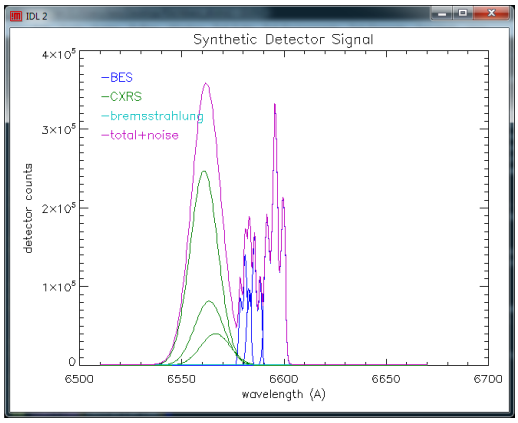
4 A sample run

The follow steps show how to run the NBASS example run. In the following transcript, anything after a line starting with > indicates something to be typed in.

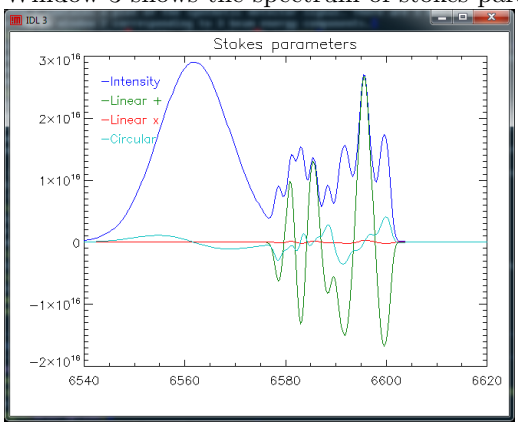
1. Change to the directory where NBASS was installed.
2. `> ./run_nbass`
3. `> nbass,'test'`
   This will instruct NBASS to look in the runs directory for a directory called `'test'`. The parameters will be loaded by from the `runs/test/parameters.pro` file. Wait a few minutes for the calculation to complete. Output from the code will be saved to the file `"<nbass installation directory>"/runs/test/test_4.sav"`
4. `> plot_results,'test','test_4.sav'`
   This will plot results from a previous run which was saved to `'test_4.sav'`
   The plots should look like the following figures: Window 1 contains a plot of the synthetic spectral intensity.



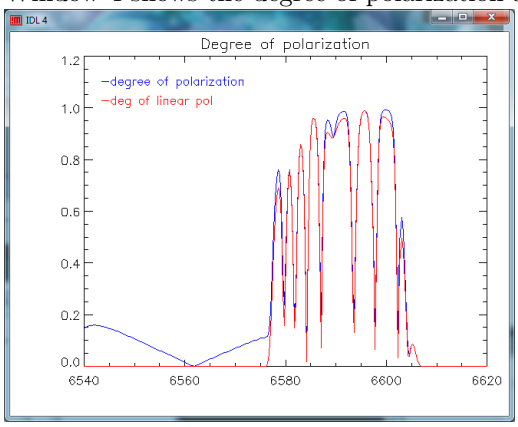
   Window 2 contains a plot of the synthetic detector signal. There are 3 BES curves and 3 CXRS curves plotted in window 2 corresponding to 3 beam energy components.



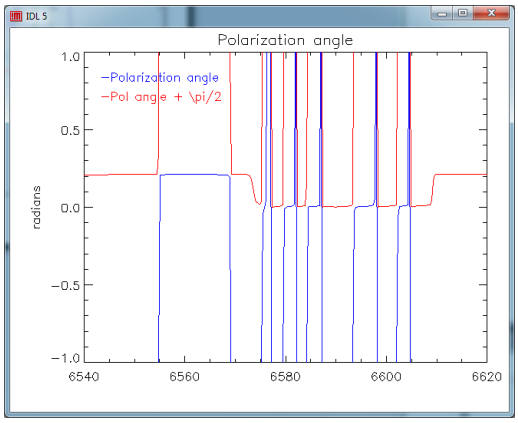
Window 3 shows the spectrum of stokes parameters.



Window 4 shows the degree of polarization as a function of wavelength.



Window 5 shows the polarization angle as a function of wavelength.



5. `> filterstokes,'test','test_4.sav',bandwidth=2.0,center=6600.0`
   This will apply a bandpass filter to the results of the previous run, approximately centered on the full energy pi line. The textual output shows the Stokes parameters for the light that passes the bandpass filter. Window 1 shows the spectrum, degree of polarization, and filter function. Windows 3-5 shows what happens if the filter center wavelength is scanned over a short range.

   It is possible to create a parameters file that accepts additional arguments during runtime. This is useful for scanning a parameter value without creating many directories. In fact, the test parameters accepts a parameter called "channel". Try this:

6. `> nbass,'test',channel=10`
   This will rerun NBASS with a slightly different set of parameters. The output will be saved to 'test_10.sav' A very simple tool can be used to compare two runs.

7. `> compare_results,['test/test_4.sav','test/test_10.sav']`
   This will plot the spectra from both runs on the same plot, allowing quick comparison.

8. `> export_ascii,'test','test_4.sav','runs/test/test_4.txt'`
   This will export the output of the NBASS run to a plain ASCII file.

5 Input Parameters

The parameters are organized into the folowing groups:

`param:` general parameters

`detector_param:` parameters relating to the spectrometer

`beam_param:` parameters relating to the beam

`view_param:` parameters relating to the view geometry

`mesh_param:` parameters relating to the mesh

`equil_param:` parameters relating to the magnetic equilibrium

`prof_param:` parameters relating to plasma profiles

The parameters within each group are listed in the following subsections.

5.1 param

`filename:` *string.* filename for results save file

`enable_bes:` *boolean.* output Stark BES spectrum

`enable_cxrs:` *string.* output CXRS spectrum

`enable_edge:` *boolean.* incomplete, should be set to 0

`enable_brem:` *boolean.* output bremsstrahlung

`linemodel:` *string.*	'gaussian'	Gaussian line shape
	'erf'	difference in error functions line shape
	'delta'	Dirac delta function line shape

`cs_effects:` *boolean.* make adjustments in CX cross section and width for finite plasma temperature

`calczeeman:` *string.*	'1'	use full Zeeman calculation for CXRS lines
	'Blom'	use Blom Zeeman calculation for CXRS lines
	'0'	ignore Zeeman effect for CXRS lines

`nonstatistical:` *boolean.* use non-statistically populated beam excited states in Stark model

`autocenter:` *boolean.* automatically center the wavelength range of the detector on the BES spectrum, overriding the wve parameter in detector_param

`n_gen:` *int.* number of noisy spectra to generate from a Poisson distribution

5.2 detector_param

`l_instr`: *double.* instrument width in angstroms of spectrometer/detector, assumed to be Gaussian

`disp`: *double or double[npix].* dispersion in angstroms/pixel

`npix`: *int.* number of pixels

`wve`: *double[npix].* wavelength of each pixel (arranged monotonically)

`sens`: *double or double[npix].* transmission×QE×etendue in cm^2/steradian

`int_time`: *double.* integration time in seconds

`detstokes`: *4x4 matrix.* multiply results by an arbitrary stokes matrix [optional]

`detgain`: *double.* Number of detector counts per photon detected

`darknoise`: *double.* Detector dark noise counts

5.3 beam_param

`num_beams:` *int.* number of neutral beams

`alcbeam_file:` *string or string[num_beams].* filename or array of filenames of ALCBEAM output files for each neutral beam

`beam_ripple:` *double.* beam_ripple = stddev(E_beam)/E_beam

5.4 view_param

`ap_shape`: *string.* shape of aperture used to generate mesh points

'point'	single point
'hex7'	7 point hexagonal grid
'hex19'	19 point hexagonal grid
'grid'	arbitrarily specified grid

`ap_rad`: *double.* for 'point','hex7','hex19' this is aperture radius (meters). for 'grid' this is size of grid point

`ap_gridx`: *double array.* positions of each point in aperture grid in H,V coordinates (meter)

`ap_gridy`: *double array.* positions of each point in aperture grid in H,V coordinates (meter)

`sp_shape`: *string.* shape of spot used to generate mesh points

'point'	single point
'hex7'	7 point hexagonal grid
'hex19'	19 point hexagonal grid
'grid'	arbitrarily specified grid

`sp_rad`: *double.* for 'point','hex7','hex19' this is spot radius (meters). for 'grid' this is size of grid point

`sp_gridx`: *double array.* positions of each point in spot grid in H,V coordinates (meter)

`sp_gridy`: *double array.* positions of each point in spot grid in H,V coordinates (meter)

`spot_pos`: *double[3].* XYZ position of spot (meters) in the plasma where view is focused

`optic_pos`: *double[3].* effective XYZ position of the optics (meters). Approximated by lens center position

5.5 mesh_param

`ds:` *double.* step length (meters) along viewing chord for calculation mesh

`cut_r1:` *double.* calculation boundary inner wall (meters)

`cut_r2:` *double.* calculation boundary outer wall (meters)

`cut_z1:` *double.* calculation boundary bottom (meters)

`cut_z2:` *double.* calculation boundary top (meters)

`chordmodel:` *string.* options for handling finite chord volume

'ray'	use only one ray from lens to plasma (faster)
'grid'	use multiple rays, controlled by ap_shape and sp_shape

`num_pini:` *int.* number of points to use in the beam source to model beam divergence

5.6 equil_param

`Bmodel:` *string.*	'miller'	use a Miller equilibrium with poloidal field generated from a known q profile
	'efit_file'	read efit data from a g file
	'efit_mds'	read efit data from mdsplus. Requires some tokamak specific implementation

`Rmajor:` *double.* Miller major radius (meters)

`aminor:` *double.* Miller minor radius (meters)

`B_tor0:` *double.* magnetic field at geometric center Rmajor (Tesla)

`upperelong:` *double.* Miller elongation for upper half of plasma, assumed to be constant across flux surfaces

`lowerelong:` *double.* Miller elongation for lower half of plasma, assumed to be constant across flux surfaces

`uppertri:` *double.* Miller triangulation for upper half of plasma at r/a=0.95. model: $\delta(r) = \text{uppertri}/0.95^2 \times (r/a)^2$

`lowertri:` *double.* Miller triangulation for lower half of plasma at r/a=0.95. model: $\delta(r) = \text{lowertri}/0.95^2 \times (r/a)^2$

`z0:` *double.* z coordinate position of midplane (meters)

`shafranov0:` *double.* Shafranov shift in center. model: $\Delta(r) = \text{shafranov0} \times (1 - (r/a)^2)$

`qprof:` *double[].* Table of q-values to use to generate poloidal field

`qprof_rho:` *double[].* normalized poloidal flux associated with q values

`efit_file:` *string.* name of a g-file to open

`efit_shot:` *double.* milliseconds

`efit_time:` *double.* seconds

`Ermodel:` *string.*	'none'	Use $E_r = 0$
	'tabulated'	Define E_r using parameters Er_r,Er_val
	'forcebalance'	Define E_r using parameters Er_diamagnetic, Er_vtheta, Er_vphi. $E_r = \frac{1}{n*Z*e}\frac{\mathrm{d}p_i}{\mathrm{d}r} - v_{\theta,i}B_\phi + v_{\phi,i}B_\theta$ (c.f. Wesson 2004 4.19.4)

`Er_r:` *double[n].* array of Rmid locations where Er is given

`Er_val:` *double[n].* array of Er values (V/m)

`Er_diamagnetic:` *double[n].* $\text{V/m} = \text{kg} \times \text{m} \times \text{s}^{-3} \times \text{A}^{-1} = \text{m}^3 \times \text{C}^{-1} \times \text{Pa/m}$

`Er_vtheta:` *double[n].* poloidal ion velocity used for calculating Er. m/s

`Er_vphi:` *double[n].* toroidal ion velocity used for calculating Er. m/s

Note: `Rmajor, aminor, B_tor0, upperelong, lowerelong, uppertri, lowertri, z0 shafranov0, qprof,` and `qprof_rho` only need to be defined for `Bmodel=miller`. `efit_file` is only needed for `Bmodel=efit_file`. `efit_shot` and `efit_time` is only needed for `Bmode=efit_mds`.

5.7 prof_param

`main_ion`: *string.* should be 'H', 'D', or 'He'

`impurities`: *string[nimp].* array of atomic symbols for impurities to include

`imp_z`: *double[nimp].* charge number of each impurity. allowed to be non-integer for partially ionized species

`imp_fr`: *double[nimp].* fraction of total impurity density for each impurity

`ne_coord`: *string.* 'rhopsi' or 'rmid'. Coordinate to use for ne

`ne_x`: *double[].* array of ne measurement positions

`ne_y`: *double[].* array of ne measurements in cm^{-3}

`te_coord`: *string.* 'rhopsi' or 'rmid'. Coordinate to use for Te

`te_x`: *double[].* array of Te measurement positions

`te_y`: *double[].* array of Te measurements in keV

`zeff_coord`: *string.* 'rhopsi' or 'rmid'. Coordinate to use for Zeff

`zeff_x`: *double[].* array of Zeff measurement positions

`zeff_y`: *double[].* array of Zeff measurements

6 Program Output

The output is saved to an IDL save file located in the run subdirectory. The filename is given by the `filename` parameter. The IDL save file is a proprietary binary file, but for convenience, a tool called `export_ascii` is included to export the full contents of the save file to an ASCII text file.

The output file can be opened in IDL by typing `restore,'filename'` at the IDL prompt, replacing 'filename' with the actual filename.

The output is organized into a three tree structures: `parameters`, `data`, and `result`. The `parameters` tree contains the input parameters. The `data` tree contains some useful data generated during internal steps of the program calculation. The `result` contains the final spectra generated by the program. The tree structures are shown below.

```
PARAMETERS
├── VERSION
├── PARAM
│   └── ...
├── DETECTOR_PARAM
│   └── ...
├── BEAM_PARAM
│   └── ...
├── VIEW_PARAM
│   └── ...
├── MESH_PARAM
│   └── ...
├── EQUIL_PARAM
│   └── ...
└── PROF_PARAM
    └── ...
```

The `parameters` tree is abbreviated here because the contents are all explained above.

- DATA
  - CHORD — chord geometry
    - NPOINT — number of grid points used in the viewing chord
    - X — x coordinate of each grid point (m)
    - Y — y coordinate of each grid point (m)
    - Z — z coordinate of each grid point (m)
    - R — R coordinate of each grid point (m)
    - PHI — ϕ coordinate of each grid point (radian)
    - RMID — Midplane radius of the flux surface that contains each grid point
    - RHO — Square root of normalized poloidal flux of each grid point
    - VECT — unit vector for ray associated with grid point
    - DS — step length between grid points along ray (m)
    - NRAY — number of rays between lens and plasma
  - FIELD — electric and magnetic fields at each grid point
    - BX — B_x Tokamak frame (T)
    - BY — B_y Tokamak frame (T)
    - BZ — B_z Tokamak frame (T)
    - BR — B_R Tokamak frame (T)
    - B_TOR — B_T Tokamak frame (T)
    - B_TOT — $|B|$ at each grid point
    - B_HAT — Unit vector $\mathbf{b}$ at each grid point
    - EX — E_x Tokamak frame (V/m)
    - EY — E_y Tokamak frame (V/m)
    - EZ — E_z Tokamak frame (V/m)
    - ER — E_R Tokamak frame (V/m)
    - E_TOT — $|E|$ Tokamak frame (V/m)
    - E_HAT — Unit vector $\mathbf{E}/|E|$ at each grid point
  - ANGLE — geometric angles (radians)
    - ALPHA — angle between beam axis and viewing ray
    - PSI — angle between $\mathbf{b}$ and viewing ray
    - PHI — angle between $s_\perp$ and $v_\perp$[1]
    - PHIL — $\pi/2 -$ PHILCOMP
    - PHILCOMP — angle between E field (beam frame) and $s_\perp$[1]
    - THETA — angle between $\mathbf{b}$ and beam axis
    - THETA1 — angle between the toroidal direction and beam axis
    - GAMMA — angle between E field (beam frame) and viewing ray[2]
    - PITCH — pitch angle defined using $\tan(\texttt{PITCH}) = \frac{B_{\rm pol}}{B_{\rm tor}}$
    - VECTH — see below
    - VECTV — see below
    - XVECT — unit vector $s_\perp/|s_\perp|$[1]
    - YVECT — unit vector perpendicular to $\mathbf{b}$ and XVECT
  - PROFILES — input profiles interpolated to grid points
    - N_E — n_e at each grid point (cm^{-3})
    - T_E — T_e at each grid point (eV)
    - Z_EFF — $Z_{\rm eff}$ at each grid point

- MAIN_ION ..... Atomic symbol for main ion
- IONS ..... Atomic symbols for plasma ion species
- N_ION ..... Density of each ion at each grid point
- Z_ION ..... Charge of each ion
- T_I .... Ion temperature at each point (assumed equal for all species)

LORENTZE_NORM . Amplitude of Lorentz E field for each beam component

DANGLE_FG ..... contains broadening terms due to finite beam grid area
- ALPHA ..... variation in ALPHA due to finite grid
- PSI ..... variation in PSI due to finite grid
- PHI ..... variation in PHI due to finite grid
- THETA ..... variation in THETA due to finite grid
- GAMMA ..... variation in GAMMA due to finite grid

DANGLE_AP ..... contains broadening terms due to finite aperature size
- ALPHA ..... variation in ALPHA due to finite aperture
- PSI ..... variation in PSI due to finite aperture
- PHI ..... variation in PHI due to finite aperture
- THETA ..... variation in THETA due to finite aperture
- GAMMA ..... variation in GAMMA due to finite aperture

DANGLE_SP ..... contains broadening terms due to finite spot size
- ALPHA ..... variation in ALPHA due to finite spot size
- PSI ..... variation in PSI due to finite spot size
- PHI ..... variation in PHI due to finite spot size
- THETA ..... variation in THETA due to finite spot size
- GAMMA ..... variation in GAMMA due to finite spot size

[1] $s_\perp$ and $v_\perp$ are the projection of the viewing ray and beam axis to the plane perpendicular to **b**

[2] When E(tokamak frame)=0, GAMMA = $\sin(\texttt{PSI})\sin(\texttt{PHI})$

RESULT

- WVE ..... [npix] Wavelength of each pixel in the spectrum (A)
- NOISY ..... [npix,n_gen] Simulated spectra, including noise (counts)
- NOISY_BK .... npix,n_gen] Simulated spectra, including noise, with beam turned off (counts)
- PURE ..... [npix] Simulated total (BES+CXRS+edge+bremsstrahlung) spectrum without noise (counts)
- PURE_BK [npix] Simulated spectrum without noise, with beam turned off (counts)
- SENS ..... Same as the input parameter with the same name
- SCALEFACTOR ..... Conversion factor from ph/m^2/s/sR/A to counts
- BES_SPEC ... [npix] Simulated beam emission spectrum (ph/m^2/s/sR/A)
- CXRS_SPEC ..... [npix] Simulated CXRS spectrum (ph/m^2/s/sR/A)
- EDGE_SPEC ..... N/A
- BREM_SPEC . [npix] Simulated bremsstrahlung spectrum (ph/m^2/s/sR/A)
- BES_COUNTS ..... [npix] Simulated beam emission spectrum (counts)
- CXRS_COUNTS ..... [npix] Simulated CXRS spectrum (counts)
- EDGE_COUNTS ..... N/A
- BREM_COUNTS ..... [npix] Simulated bremsstrahlung spectrum (counts)
- BES_DATA ..... beam emission polarization data
  - STOKES_S0 ..... [npix,nbeam] $E_H E_H^* + E_V E_V^*$ (ph/m^2/s/sR/A)
  - STOKES_S1 ..... [npix,nbeam] $E_H E_H^* - E_V E_V^*$ (ph/m^2/s/sR/A)
  - STOKES_S2 ..... [npix,nbeam] $2\Re(E_H E_V^*)$ (ph/m^2/s/sR/A)
  - STOKES_S3 ..... [npix,nbeam] $2\Im(E_H E_V^*)$ (ph/m^2/s/sR/A)
  - POLDEGREE ... [npix,nbeam] degree of polarization=$\sqrt{s_1^2 + s_2^2 + s_3^2}/s_0$
  - SPSI ..... [npix,nbeam] polarization angle
  - SCHI ..... [npix,nbeam] polarization ellipticity
- CXRS_DATA ..... CXRS polarization data
  - STOKES_S0 ..... [npix,nbeam] $E_H E_H^* + E_V E_V^*$ (ph/m^2/s/sR/A)
  - STOKES_S1 ..... [npix,nbeam] $E_H E_H^* - E_V E_V^*$ (ph/m^2/s/sR/A)
  - STOKES_S2 ..... [npix,nbeam] $2\Re(E_H E_V^*)$ (ph/m^2/s/sR/A)
  - STOKES_S3 ..... [npix,nbeam] $2\Im(E_H E_V^*)$ (ph/m^2/s/sR/A)

npix is the number of pixels. n_gen is the number of randomized spectra to generate (based on the pure spectrum with added noise). nbeam is the number of beam energy components.