MODULE 4
Modelling and analysing special spectral features - A unified approach.

Demonstration script

Hugh Summers, Christopher Nicholas,
Martin O'Mullane and Alessandra Giunta

September 27, 2013

Contents

1 Demo (a) Show spectral features

```
------------------------------
DEMO A: Show spectral features
------------------------------
PURPOSE: To explore the behaviour of the Stark split manifold of the
(beam)  hydrogen Balmer-alpha emission as a function of beam energy, plasma
parameters  and external electric field.

COMMENTS: ADAS605 is a pedagogical tool to interactively see the effect of
varying the conditions which affect a spectrral feature. The afg framework
(ADAS feature generator) supplies the feature and the appropriate widgets,
and their ranges, are automatically generated thus making it simple to add
new features. Note that the available feature are limited to beam Stark
features and Zeeman-split line emission for now.

DEMO a: Using ADAS605 with the interactive ADAS windows
1. Run interactively ADAS605 for Stark feature.
  (output file: demo_a_adas605.ps).
```

1.1 Demo (a) Figures

1.1.1 Demo (a-1) demo_a/demo_a_adas605.pdf

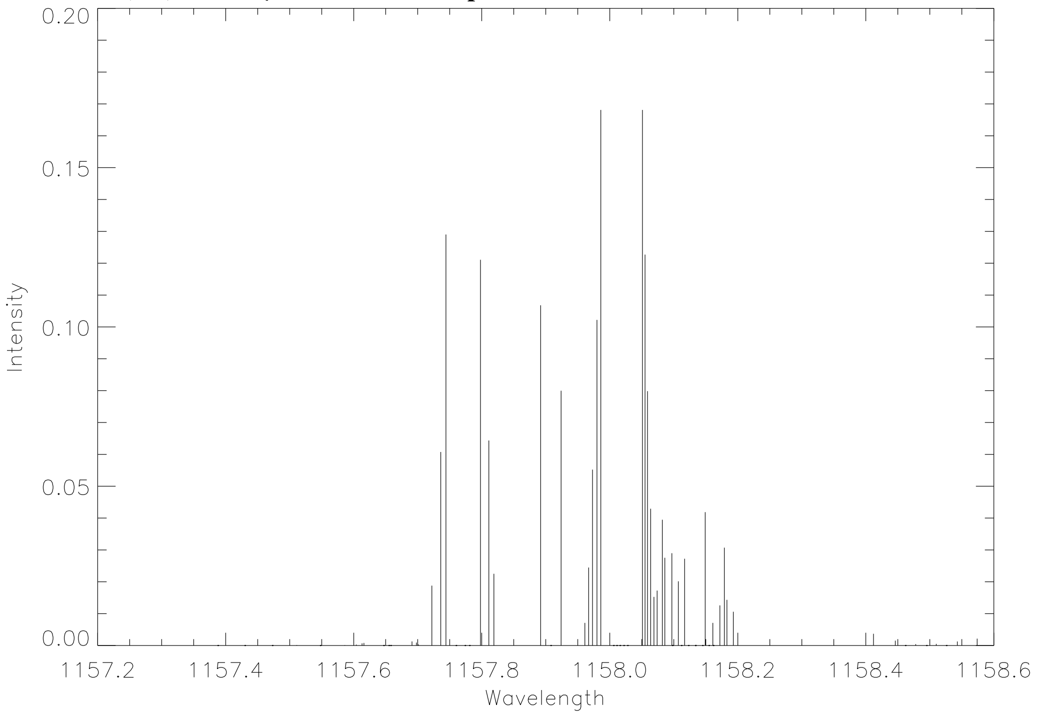

Figure 1: Stark feaures from ADAS605.

2 Demo (b) Fitting spectral features

```
----------------------------------
DEMO B: Fitting spectral features.
----------------------------------
PURPOSE: Use the FFS (Framework for Feature Synthesis) routines to fit a
Zeeman feature.

EXAMPLE: A sample JET spectrum, distributed in adas/arch603, is the
BeIII multiplet, 1s 2s 3S - 1s 2p 3P  at 372.17nm, on a relatively clean
background with a nearby interfering line. The purpose of the
demonstration is to setup a model, fit the spectrum with ffs and  deduce
the local magnetic field.

COMMENTS: The FFS framework is an optimized, least squares fitting
system based on physics-based parametrized spectral features. Individual
lines, background models and broadening operators are also part of the
system to allow a sophisticated description of the target spectrum,
specified with a bespoke model definition language (MDL). Multiple
features and coupling of parameters are fundamental to the design of the
system.


DEMO b: Fit Zeeman features using FFS routines
1. Take an experimental spectra (example from JET).
2. Setup a ffs model defined by the MDL file 'zeeman_be2.mdl'.
3. Plot the data with the model fit.
Program: demo_b.pro
Samples of input files: simplify_output.mdl, zeeman_be2.mdl
Sample of output file:.demo_b.ps
```

2.1 Demo (b) Figures

2.1.1 Demo (b-1) demo_b/demo_b.pdf

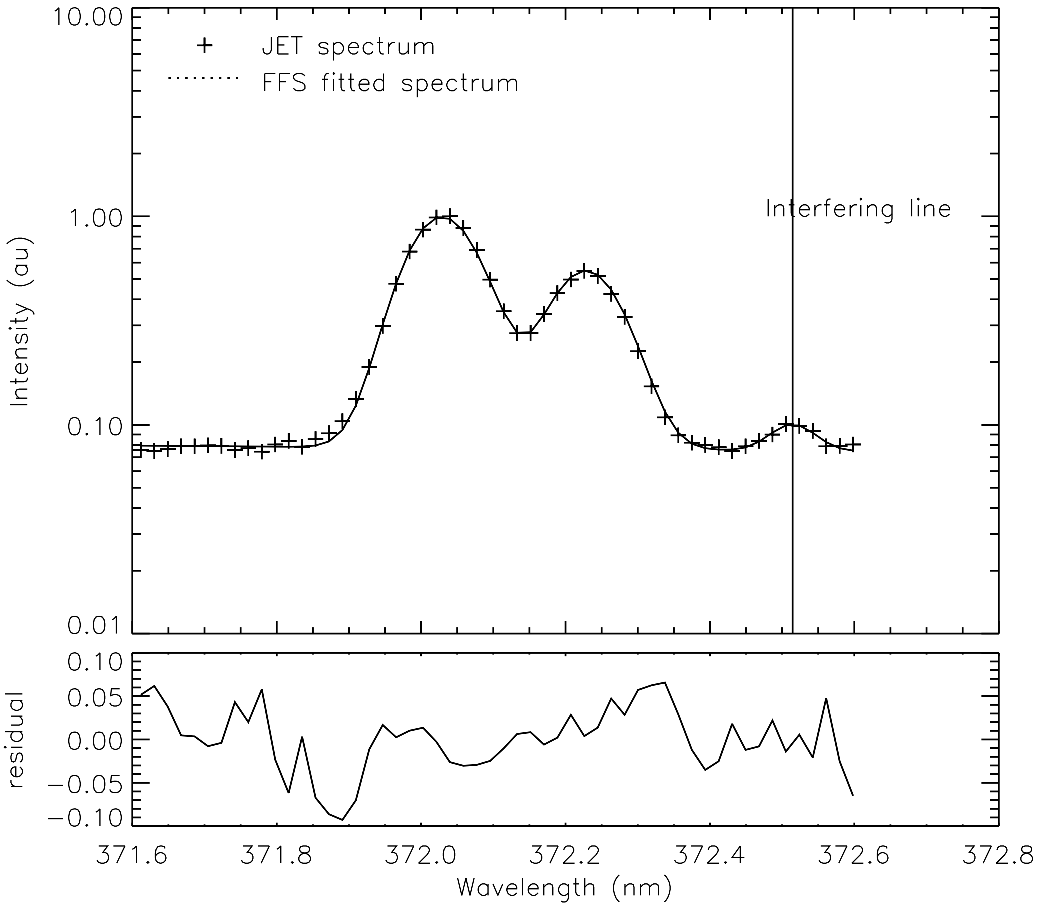

Figure 2: JET spectrum fitted using FFS fit.

2.2 Demo (b) Procedures

2.2.1 Demo (b-1) demo_b/demo_b.pro

```
;-----------------------------------------------------------------------
; Using FFS (Framework for Feature Synthesis) routines
; fit a Zeeman feature (with a background and interfering
; ordinary line) to a sample spectrum, of the multiplet
; BeIII 1s 2s 3S - 1s 2p 3P (372.17nm).
;-----------------------------------------------------------------------


pro demo_b


; Example JET spectrum from central ADAS

adascent = getenv('ADASCENT')
restore, adascent + '/arch603/jet_test.dat'

; Get the wavelength in nm and restrict to +/- 0.5nm about the feature
; Set the error to be square root of the counts

x = adas_vector(low=header.wavemin, high=header.wavemax, num=header.nsize,/linear)
x = x / 10.0
y = spectrum

ind = where(x GT 371.6 and x LT 372.6, count)
x   = x[ind]
y   = y[ind]
err = sqrt(y)


; Setup a ffs model defined by the MDL file 'zeeman_be2.mdl'
;
; 1. make a model object (container)
; 2. parse the MDL file
; 3. add the parsed MDL to the model
; 4. Set the independent x values in the model
; 5. make a fit object and a data structure with wavelength, intensity and error
; 6. fit the model to the data

model  = obj_new('ffs_model', modelname='zeeman_be2', /debug)
parser = obj_new('ffs_parser', file='zeeman_be2.mdl', /debug)

tmp = parser->apply(model)
tmp = model->setxdata(x)

tmp = model->pdsetup()
tmp = model->pdsetup()
```

```
fit  = obj_new('ffs_fit')
data = {x:x, y:y, error:err}
tmp  = fit->apply(model, data)

; Parameters of fit

parnames = model->getparnames(/free, /full)
parvals  = model->getparvals(/free)
errors   = fit->geterrors()



; Plot the data with the model fit and print the fitted parameters

;set_plot,'ps'
;device, /isolatin1, font_index=8
;device, bits=8, filename='demo_b.ps',  $
;        font_size = 14, xsize=18.0, ysize=16.0, $
;        yoffset=7.0, /color
;device, /helvetica

window,0,ret=2

fitspec  = *(model->getresult())
residual = (fitspec.intensity - y) / y

plot_io, [371.6, 372.6], [0.01,2], /nodata, xtickname=replicate(' ', 10), $
         ytitle = 'Intensity (au)', position=[0.15, 0.3, 0.95, 0.95]

oplot, x, y, psym=1

oplot, fitspec.wavelength, fitspec.intensity

; Label graph

plots, [371.7], [6.6], psym=1
oplot, [371.65, 371.75], [4.6, 4.6], line=1

xyouts, 371.78, 6, 'JET spectrum'
xyouts, 371.78, 4, 'FFS fitted spectrum'

; Plot fitted position of the ordinary line

oplot, [parvals[6],parvals[6]], [0.01, 10]
xyouts, 0.9999 * parvals[6],  1.0, 'Interfering line'

; Plot residuals
```

```
plot, [371.6, 372.6], [-0.1, 0.1], /nodata, $
         xtitle = 'Wavelength (nm)', $
         ytitle = 'residual', position=[0.15, 0.10, 0.95, 0.28], /noerase

oplot, x, residual


;device,/close
;set_plot,'x'



; Print out the fitted parameters


for j = 0, n_elements(parnames)-1 do $
    print, j, parnames[j], parvals[j], errors[j], $
           format = '(i2, " : ", a20, " = ", 2f12.4)'
; Write the parsed model to a file in order to see all the limits

isok = parser->writedefinition(model, 'simplify_output.mdl')


stop

end
```

2.3 Demo (b) Drivers

2.3.1 Demo (b-1) demo_b/simplify_output.mdl

```
Model written automatically by FFS
----------------------------------

Editing this file by hand is ok but please
note that FFS will not retain any comments
or formatting changes which you may add if
asked to read and then re-save this model.

(model
    (add
        (background-linear backg)
        (multiply (broaden_gauss (shift (adas-zeeman be2zeeman) sh) bg) be2mult)
        (gaussian g1)
    )
)

(setval backg.c 0.079600883)
(setval backg.m -0.0048565742)
(setval be2zeeman.obsangle 0.0000000)
(setval be2zeeman.bvalue 1.8503528)
(setval sh.lambda -0.069918945)
(setval bg.fwhm 0.10742071)
(setval be2mult.factor 0.058118828)
(setval g1.pos 372.51450)
(setval g1.fwhm 0.070975998)
(setval g1.area 0.0019231882)

(setval be2zeeman.pol 0)
(setval be2zeeman.findex 8)
(setval bg.trap 10.000000)
(setval g1.trap 5.0000000)

(setmin backg.c -Infinity)
(setmax backg.c Infinity)
(setmin backg.m -Infinity)
(setmax backg.m Infinity)
(setmin be2zeeman.obsangle 0.0000000)
(setmax be2zeeman.obsangle 90.000000)
(setmin be2zeeman.bvalue 0.97000000)
(setmax be2zeeman.bvalue 4.0000000)
(setmin sh.lambda -0.10000000)
(setmax sh.lambda 0.20000000)
(setmin bg.fwhm 0.050000000)
(setmax bg.fwhm 1.0000000)
(setmin be2mult.factor 0.010000000)
(setmax be2mult.factor 100.00000)
```

```
(setmin g1.pos 372.40000)
(setmax g1.pos 372.60000)
(setmin g1.fwhm 0.0000000)
(setmax g1.fwhm Infinity)
(setmin g1.area 1.0000000e-05)
(setmax g1.area Infinity)


(free backg.c backg.m be2zeeman.bvalue sh.lambda
            bg.fwhm be2mult.factor g1.pos g1.fwhm g1.area)

(fixed be2zeeman.obsangle)
```

2.3.2 Demo (b-1) demo_b/zeeman_be2.mdl

```
Setup model of BeIII Zeeman feature on a liner background with
one contaminating ordinary line around 372.5nm

The Zeeman feature
  - can be shifted in wavelength (sh - constrained to -0.1 to 0.2 nm)
  - is Doppler broadened with a Gaussian with FVWM constrained to 0.05 to 1.0
  - is multiplied by a factor constrained to 0.01 to 100.0
  - the field is constrained between 1-4T
  - all pi and sigma components are used
  - the observation angle is 0 degrees

The ordinary line
  - position is constrained between 372.4 - 372.6 nm
  - fwhm and area are not constrained

The background
  - is linear
  - unconstrained constant value and slope

(model
    (add
       (background-linear backg)
       (* (broaden_gauss (shift-lambda (adas-zeeman be2zeeman) sh) bg) be2mult)
       (gaussian g1)
    )
zeeman)

(setval backg.m 0.01)
(setval backg.c 0.1)


(setval g1.pos 372.5)
(setlimits g1.pos 372.4 372.6)

(setval g1.area 1.0)
```

```
(setval g1.fwhm 0.07)


(setval sh.lambda -0.06)
(setlimits sh.lambda -0.1 0.2)

(setval be2zeeman.findex 8)
(setval be2zeeman.pol 0)
(setval be2zeeman.obsangle 0.0)
(fixed be2zeeman.obsangle)
(setval be2zeeman.bvalue 1.0)
(setlimits be2zeeman.bvalue 1.0 4.0)

(setval be2mult.factor 1.0e-1)
(setlimits be2mult.factor 1.0e-2 1.0e2)

(setval bg.fwhm 0.07)
(setlimits bg.fwhm 0.05 1.0)
```