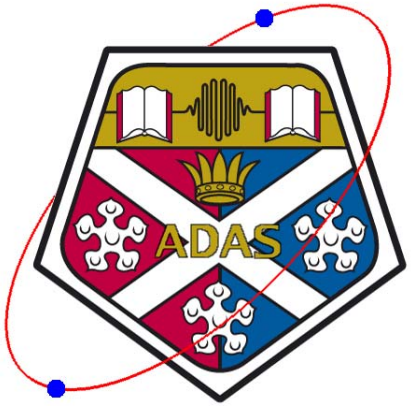
Generalized Spectral Feature Synthesis and Fitting

The Team

A. Meigs, C. Nicholas, A. Whiteford

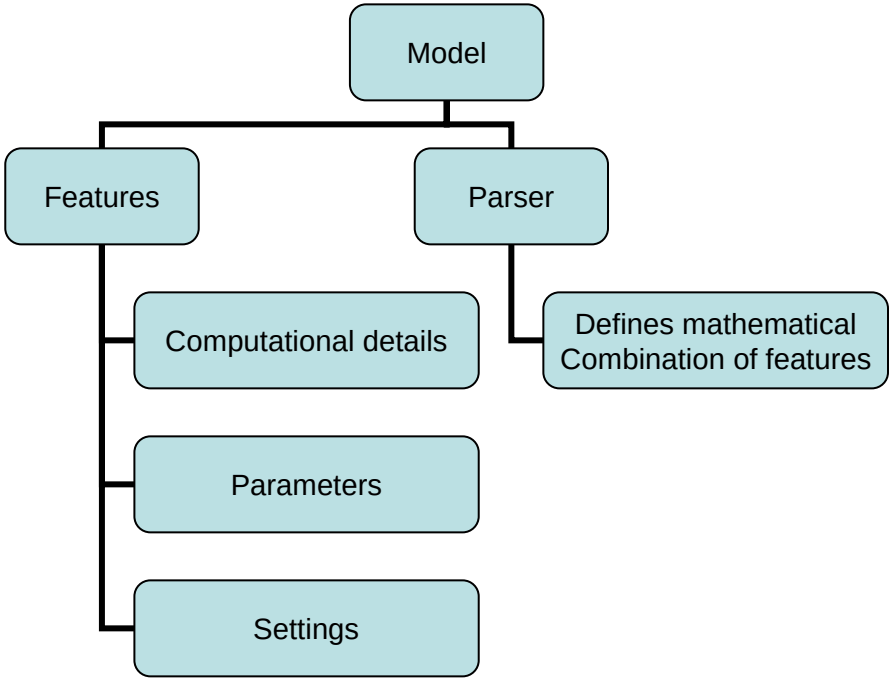


Outline

- Spectral Feature Simulation and Fitting: Concepts
- Simple examples: Gaussians, Lorentzians and Voigts
- **F**ramework for **F**eature **S**ynthesis (ffs)
- Examples of model generation
- ADAS **S**pectral **F**eature **F**ramework (sff)
- Examples of SFF introspection calls (Zeeman, High-Z envelopes)
- Where we are and where we are going…

Spectral Feature Simulation and Fitting: Concepts

- Modularity: spectral features share concepts/operations thru basic building blocks
- Simple language for combination of features (lisp-like)
- Support linear superposition of features plus basic operations: convolution (for Doppler and Instrumental response), product, etc
- Computation is analytical where possible, numerical where not
- Interface to ADAS features
- User programmable extension thru definition of interface, classes and operations.



Model = instr-function @ (doppler-feat1 @ line1 + doppler-feat2 @ line2 + gaussian + zeeman…)

Simple example of FFS calls from command line

- Creating a model (name gslo) using a parser string
- Linear superposition of a Gaussian (name gs1) and a Lorentzian (name lo1):

```
IDL> parstr = '(model gslo (+ (gaussian gs1) (lorentzian lo1)))‘
```

- Object creation step using parser string

```
IDL> model = obj_new('ffs_model',calcstring=parstr)
```

- Note (currently) parameters are zero’ed:

```
IDL> print,model->)->getparvals(elementname=‘gs1’)
      0.0000000      0.0000000      0.0000000

IDL> print, model->getParvals(elementname=‘lo1’)
      0.0000000      0.0000000      0.0000000
```

- Set the parameters for the Lorentzian:

```
IDL> model->setparvals,elementname=‘lo1’, [4620.0,15.0,6e14], /fitting
IDL> print, model->getparvals(elementname=‘lo1’)
      4620.0000      15.000000   6.0000002e+14
```

- Now set the Gaussian:

```
IDL>model->setparvals, elementname=‘gs1’, [4650.0,5.0,1e14],/fitting
```

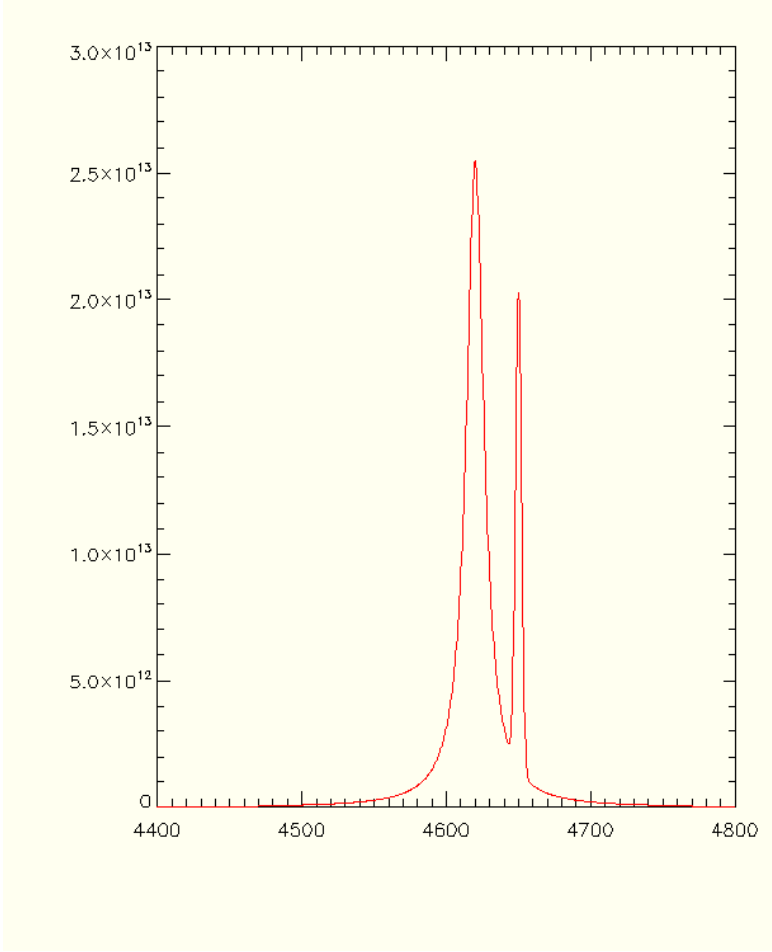
Simple example of FFS calls from command line

- Now set the wavelength grid

  IDL> model->setxdata, 4000.0+0.5*indgen(2000)
- And evaluate the model:

  IDL> y = model->evaluate()
- And plot it:

  IDL> mpi_plot_mod, y.wavelength, y.intensity



Simple example of FFS calls from command line

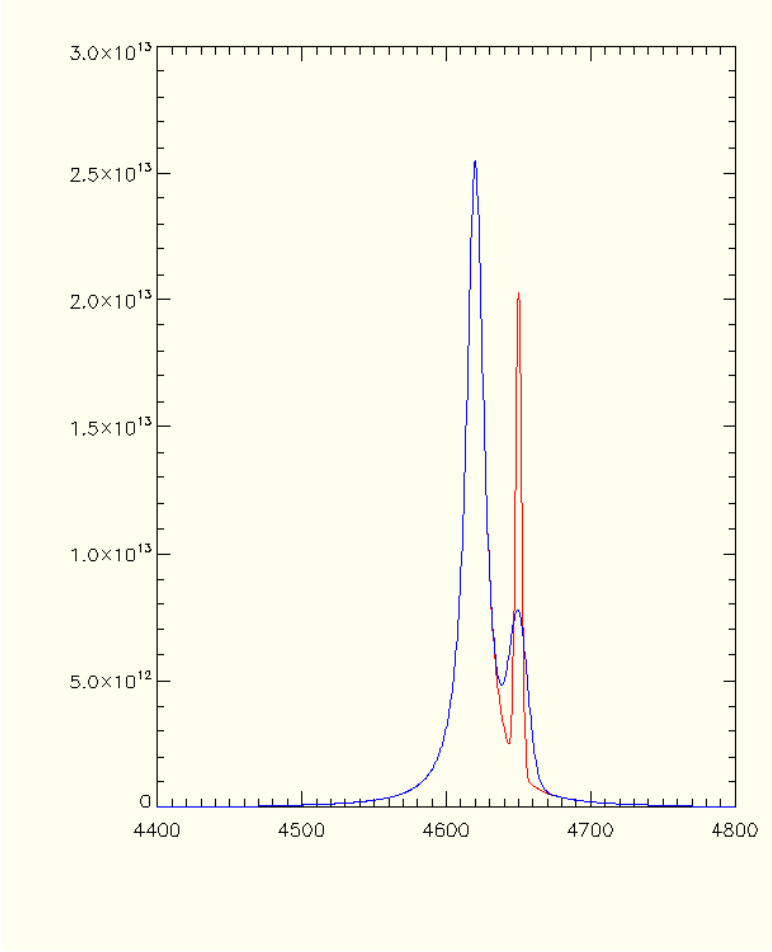
- Changing the width of gs1:

  IDL> model->setparvals, elementname='gs1', $
  parname='fwhmg', 15.0

- Re-evaluate the model and overplot:

  IDL> y = model->evaluate()

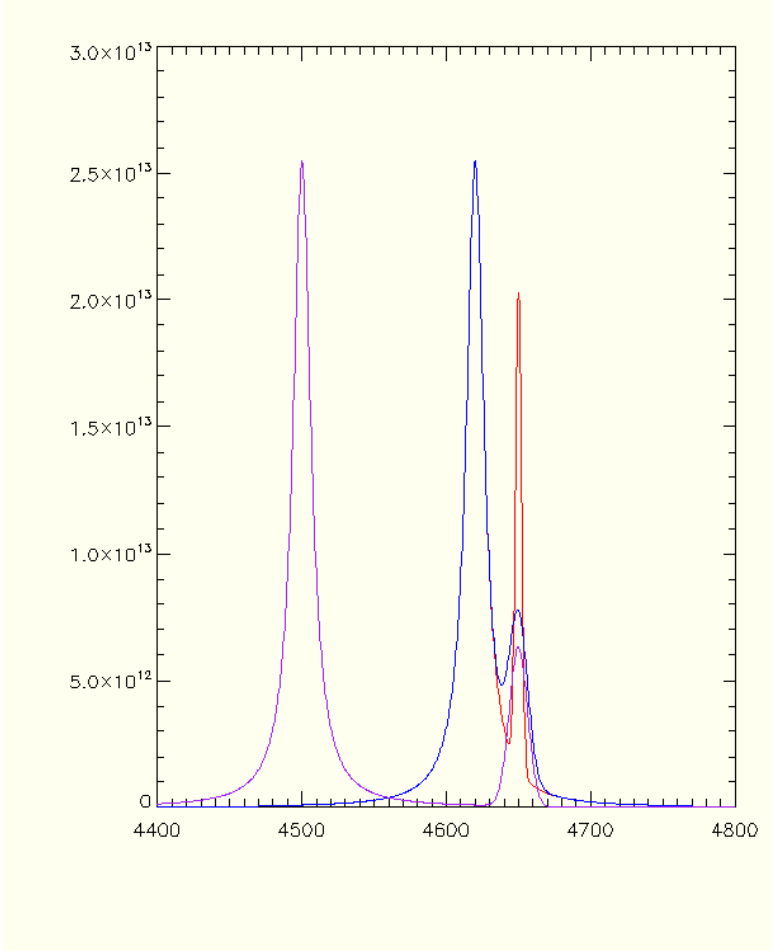
  IDL> mpi_plot_mod, y.wavelength, y.intensity, $
  /overplot, color='blue'



Simple example of FFS calls from command line

- Change the position of the Lorentzian:

```
IDL> model->setparvals, parname='ctr',
        elementname='lo1', 4500.0
IDL> y = model->evaluate()
IDL> mpi_plot_mod, y.wavelength, y.intensity,
        /overplot, color='purple'
```

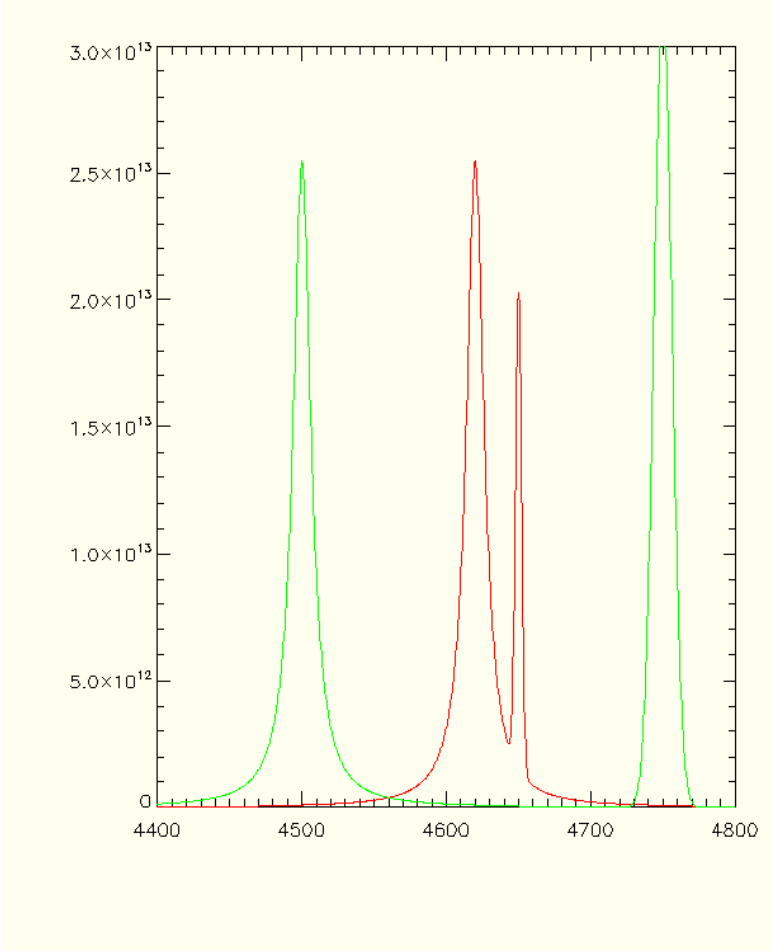


Simple example of FFS calls from command line

- Finally change the postion and intensity of the Gaussian:

```
IDL> model->setparvals, parname='ctr',
        elementname='gs1', 4750.0
IDL> model->setparvals, parname='area',
        elementname='gs1', 5e14
IDL> y = model->evaluate()
IDL> mpi_plot_mod, y.wavelength, y.intensity,
        /overplot, color='green'
```

- Only the original settings for the model (red) and the final (green) are shown



More "Atomic" Example

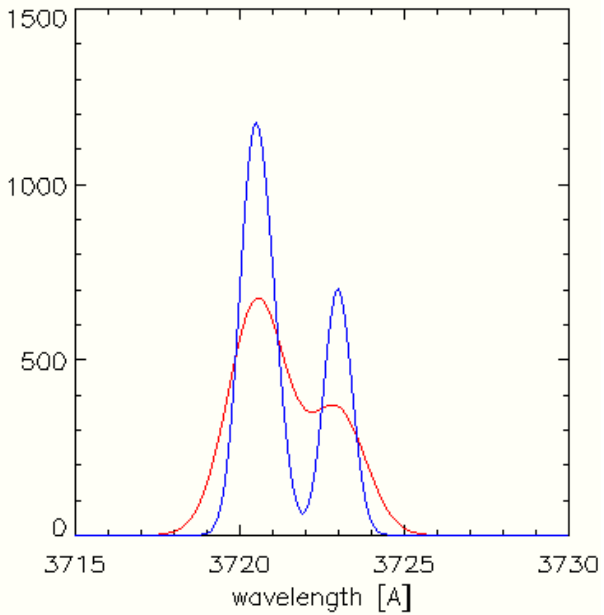
```
IDL> parstr = '(model beiii (broaden-gaussian (+ (line be1) (line be2) (line be3)) gsbroad))'

IDL>  model = obj_new('ffs_model',calcstring=parstr)
```

```
IDL>  print, model->getElementNames()
be1 be2 be3  gsbroad
IDL>  print, model->getparnames()
pos intensity pos intensity pos intensity fwhmg trap
IDL>  print, model->getparnames(elementname='gsbroad',/fitting)
Fwhmg
```

```
IDL> model->setparvals, elementname='be1', parname='pos', 3720.36
IDL> model->setparvals, elementname='be2', parname='pos', 3720.92
IDL> model->setparvals, elementname='be3', parname='pos', 3722.98
IDL> model->setparvals, elementname='be1', parname='intensity', 1000.0
IDL> model->setparvals, elementname='be2', parname='intensity', 500.0
IDL> model->setparvals, elementname='be3', parname='intensity', 750.0
IDL> model->setparvals, elementname='gsbroad', parname='fwhmg', 2.0
```

```
IDL> model->setxdata, 3700+indgen(500)*.1
IDL> y = model->evaluate()
IDL> mpi_plot_mod, y.wavelength, y.intensity,color='red'
```

```
IDL> model->setparvals, elementname='gsbroad', parname='fwhmg', 1.0
IDL> y = model->evaluate()
IDL> mpi_plot_mod, y.wavelength, y.intensity,color='blue',/overplot
```



ADAS Spectral Feature Framework (sff)

- SFF provides access to ADAS special feature codes.
- Currently, the supported models include:
  - Heavy species envelope emission
  - Motional Stark multiplet
  - Zeeman / Paschen Back
- Awaiting completion / inclusion:
  - He-like soft x-ray resonance and satellite lines
  - Balmer series / series limit
- Modular system allows plug-in of future ADAS special features.
- Wrapper object, **ffs_adas**, allow use in ffs as ffs_elements

C. Nicholas

The Special Feature Framework (SFF) API – example query

```
o = obj_new('zeeman')
desc = o->getDesc()
```

```
help, desc, /str
```

```
   NAME            STRING    'Zeeman Feature'
   PARAMETERS      STRUCT    -> <Anonymous> Array[1]
```

```
help, desc.parameters, /str
```

```
   POL             STRUCT    -> <Anonymous> Array[1]
   OBSANGLE        STRUCT    -> <Anonymous> Array[1]
   BVALUE          STRUCT    -> <Anonymous> Array[1]
   FINDEX          STRUCT    -> <Anonymous> Array[1]
```

```
help, desc.parameters.obsangle, /str
```

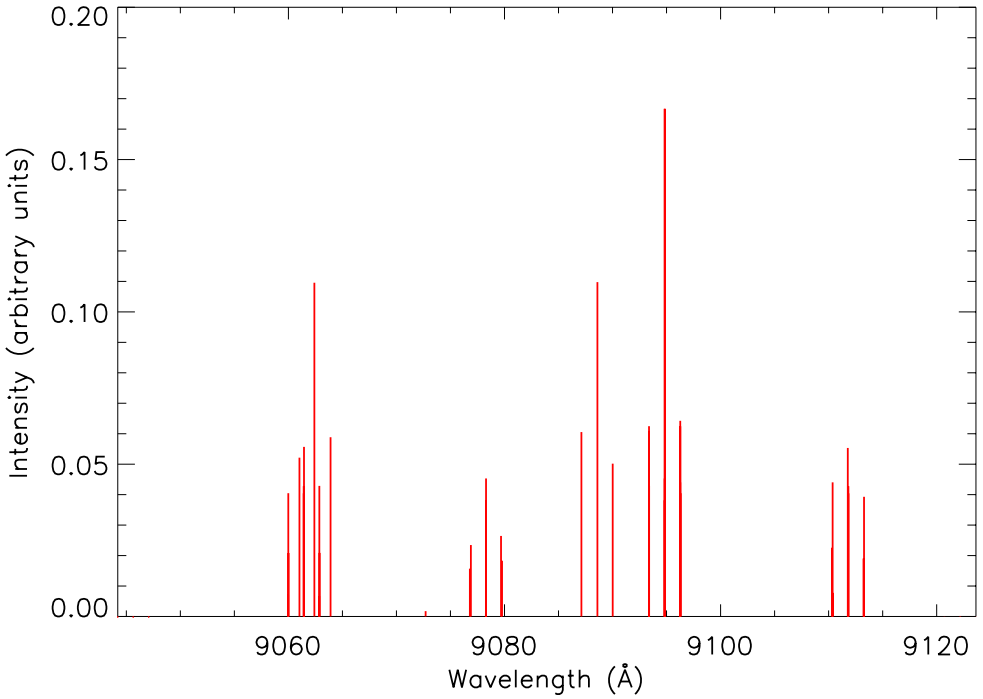
```
   DESC            STRING    'Observation angle (relative to field)'
   TYPE            STRING    'float'
   UNITS           STRING    ‘degrees'
   MIN             STRING    '0.0'
   MAX             STRING    '90.0'
   DISPTYPE        STRING    'continuous'
```

C. Nicholas

The Special Feature Framework (SFF) API – example plot

- Can easily produce plot:

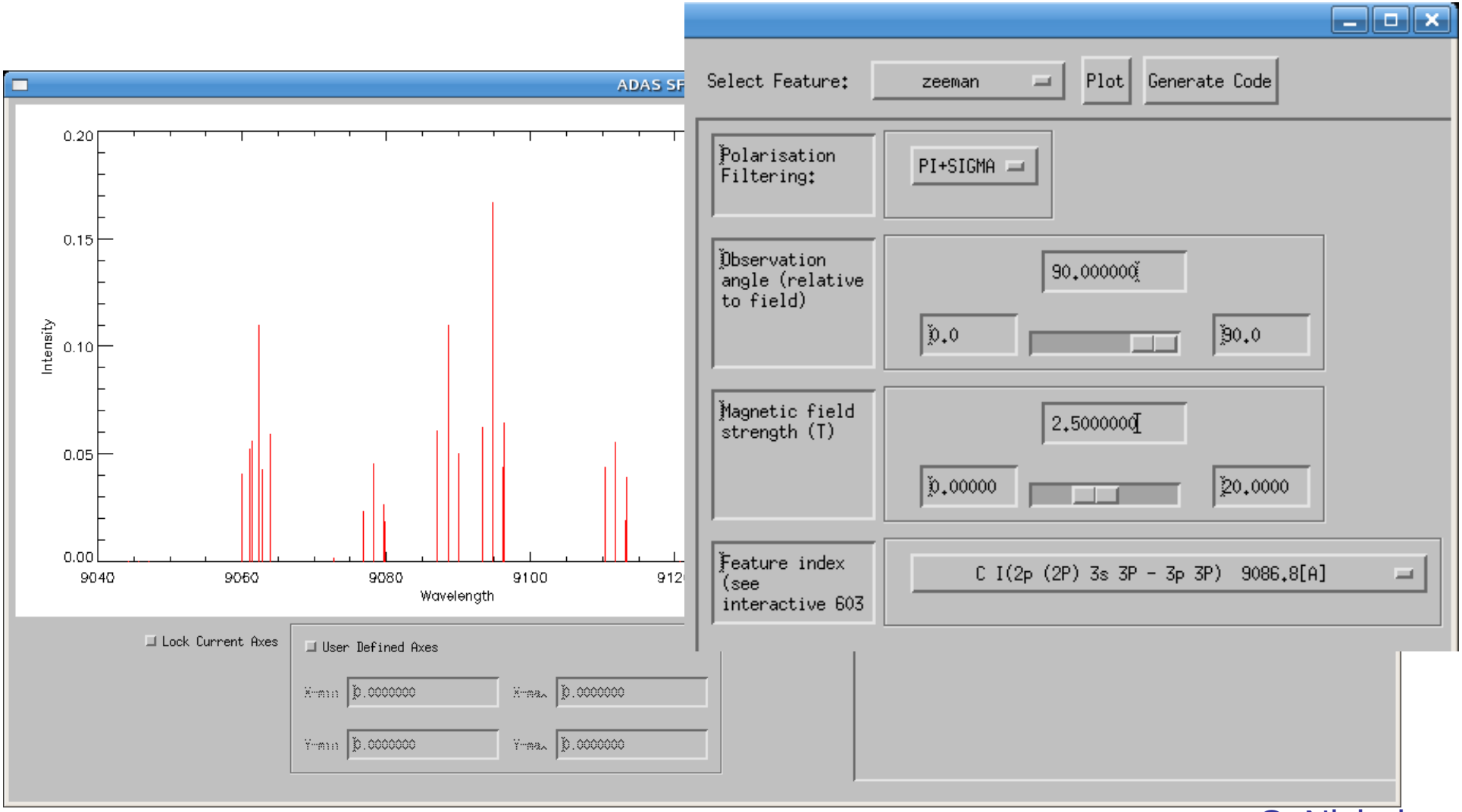
```
o = obj_new('zeeman')
pars = o->getPars()

pars.pol=1
pars.obsangle=90.0
pars.bvalue=2.5
pars.findex=15

o->setPars, PARS=pars

o->calc

PLOT, o->getWv(), o->getIntensity(), $
XTITLE='Wavelength (A), $
YTITLE='Intensity (arbitrary units)'
```



C. Nicholas

Examples of SFF introspection calls

The Special Feature Framework (SFF) GUI



C. Nicholas

What's next

- Convolution operator: how to handle analytical convolution
- Smarten up the parser (existing code too numerical, needs some inclusion of symbolic maths)
- Parser file to include initial parameter values, limits and fitting flags
- Ad-hoc addition of new elements to a model with up date of parser string (and hopefully deletion of elements)
- Related is creation of a parser editor which "knows" the allowed operators and features/elements.
- Coupling to be handled by ffs_model and ffs_parser

- Finish ADAS sff routines
- Data class to define common data format and operations (slicing, units, calibration info…)
- Re-implementation of the gui's to modify elements (modify old project's gui codes)
- Fitting routine (or wrapper to existing) which interfaces with ffs
- Fit some JET data
  - Density limit experiment from March– Balmer and Paschen series)
  - Create cxrs model and compare to CXSfit…

More complicated FFS parser file to create model

```
(model example model
        (+    (broaden-gaussian (line l1) b1)
                    (broaden-gaussian (line l2) b2)
        )
)
;Set initial values for the model:
(setval l1.pos                  5)
(setval l1.intensity    3)
(setval b1.fwhmg        4)
(setval l2.pos                  9.5)
(setval l2.intensity    7)
(setval b2.fwhmg        3)
;Set that all parameters are free:

(free l1.pos l1.intensity b1.fwhmg l1.pos l1.intensity b1.fwhmg)

;Set limits on parameters:
(setmin l1.pos 2)
(setmax l1.pos 10)
;or...
(setlim l2.pos 7 15)
```