Integrated Spectral Analysis and Fitting

Chris Nicholas
Andrew Meigs
Hugh Summers
Allan Whiteford

ADAS 2009

Overview

- ADAS Feature Generation (AFG)
    - Easy access to ADAS special feature models.
    - Provides common access point to the ADAS special feature codes.
    - Consistent interface when utilising each of the models.
    - Graphical exploration tool allows auto generated example code.

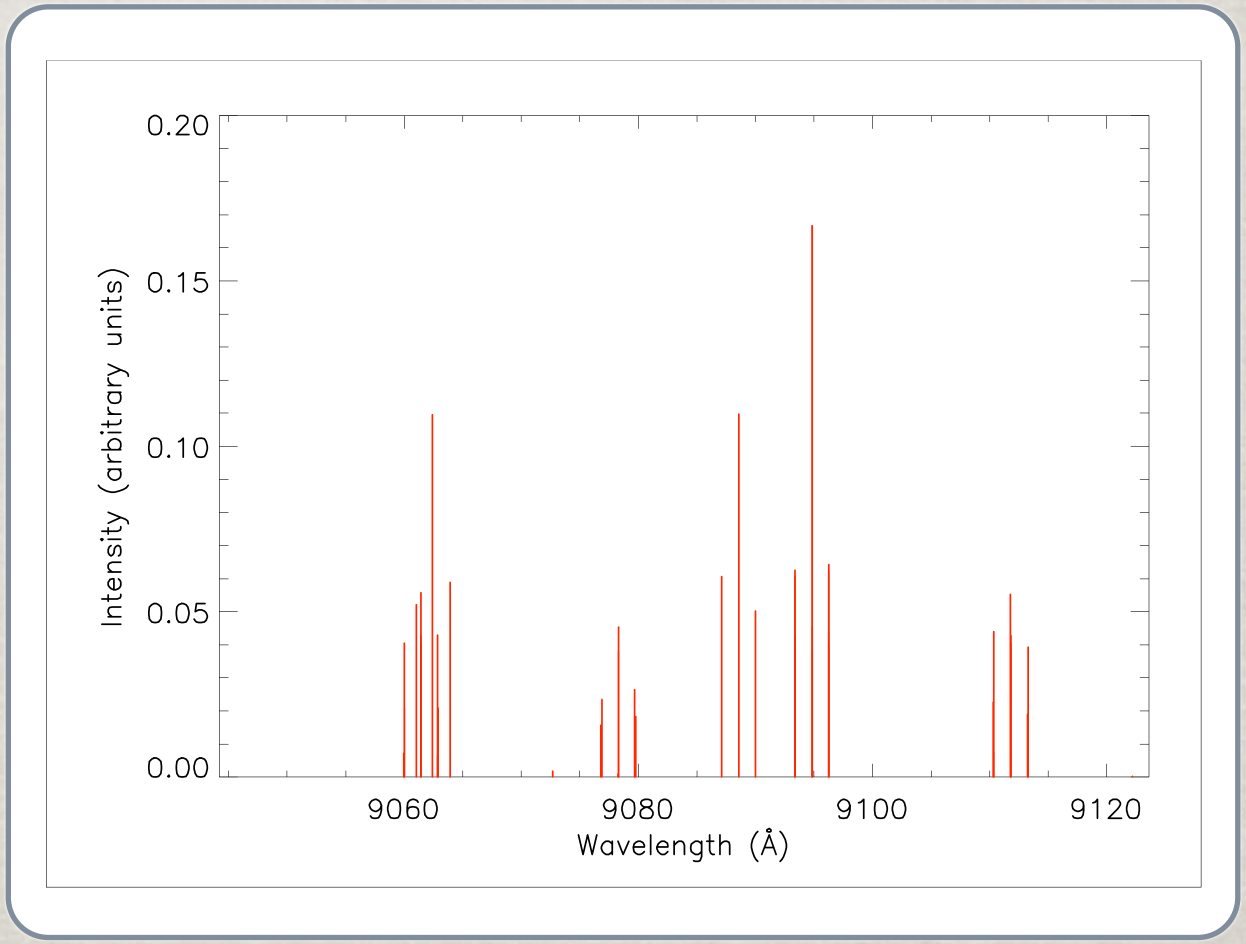
ADAS Feature Generation (AFG) API

- Currently, the supported models include:
  - Motional Stark multiplet
  - Zeeman / Paschen Back
  - H-like Zeeman
- Awaiting completion / inclusion:
  - Heavy species envelope emission
  - He-like soft x-ray resonance and satellite lines

Example AFG Plot

- Quick / easy to generate a special feature using AFG

```
pars=afg('zeeman',/parameters)

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

res=afg('zeeman',calculate=pars)

plot,res.wv,res.intensity,/nodata

for i=0,n_elements(res.wv)-1 do $
    oplot,[res.wv[i],res.wv[i]],[0,res.intensity[i]]
```



Great - so what?

- What else can AFG do?
  - Interactively assists a user regarding parameter input...

Example AFG Query

```
desc = afg(‘zeeman’, /desc)
help, desc, /str
```

```
NAME          STRING    'Zeeman Feature'
TEXT          STRING    'ADAS implementaion of Zeeman features
base'...
PARAMETERS      STRUCT    -> <Anonymous> Array[1]
```

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

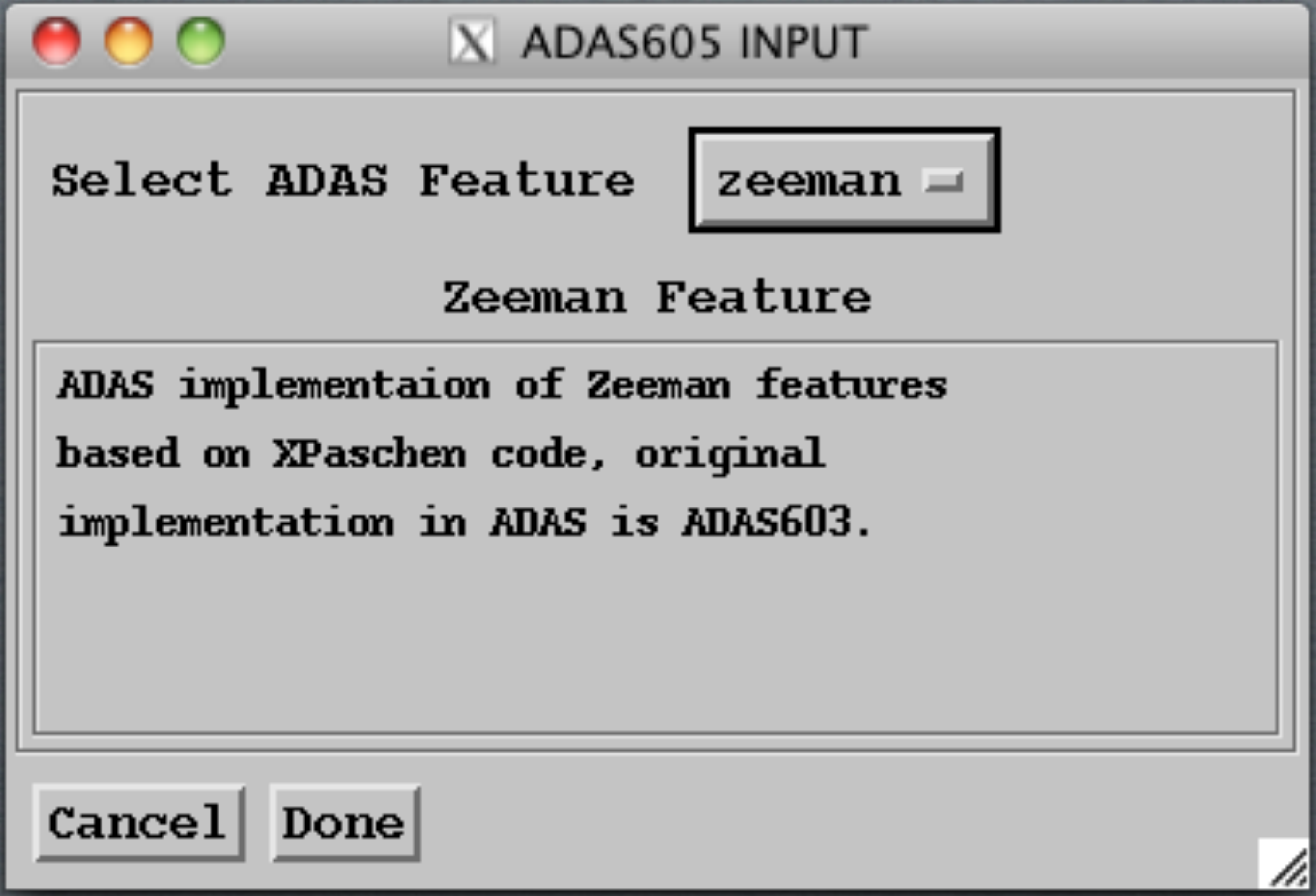
```
POL            STRUCT    -> <Anonymous> Array[1]
OBSANGLE         STRUCT    -> <Anonymous> Array[1]
BVALUE          STRUCT    -> <Anonymous> Array[1]
FINDEX          STRUCT    -> <Anonymous> Array[1]
```

```
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'
LOG            INT            0
ALTERSLIMITS    INT            0
```

Great - so what?

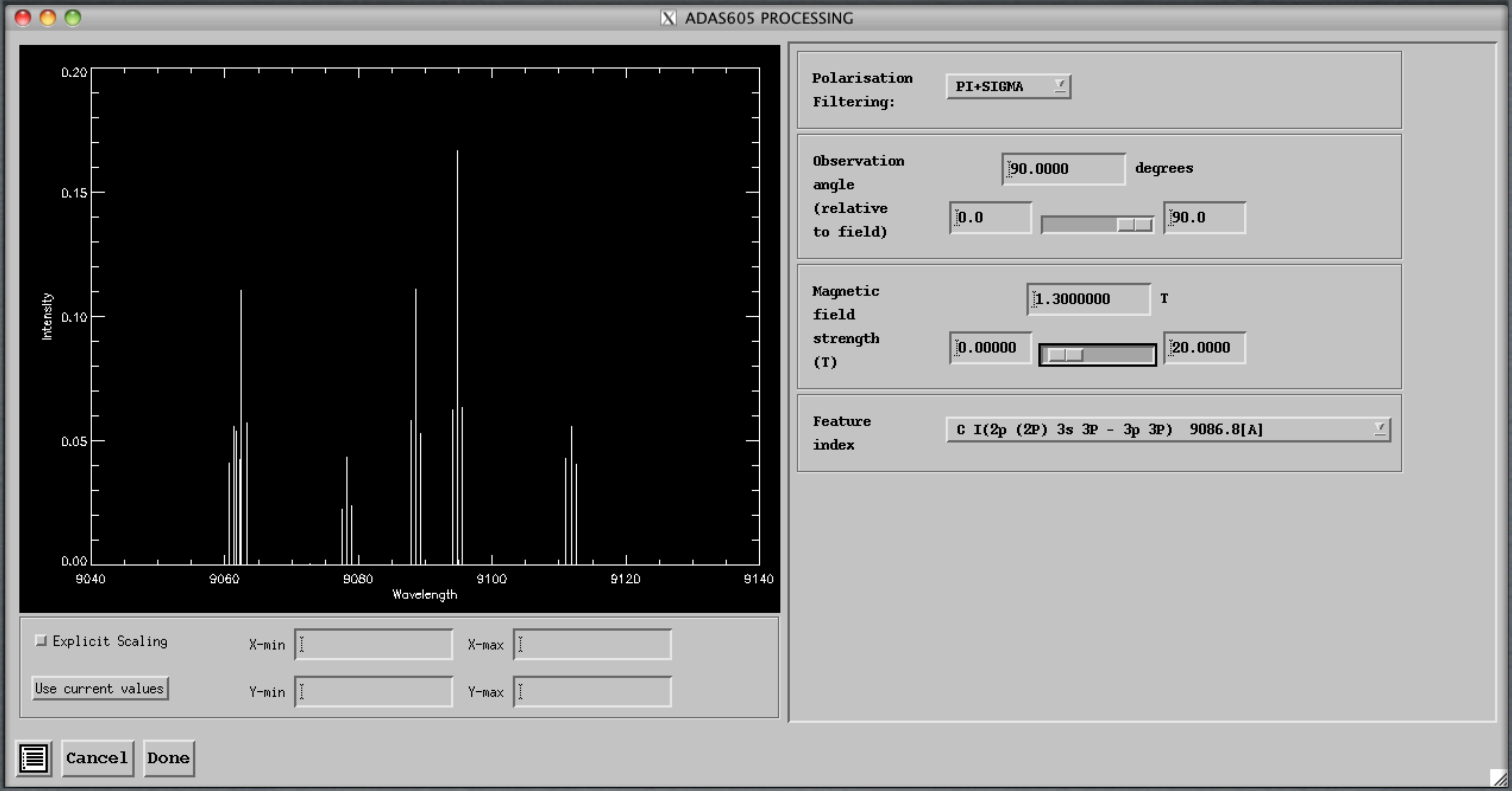
- Why are AFG queries useful?
    - Consistent API between ADAS special feature programs and user created codes - as an example...



Zeeman Feature

ADAS implementaion of Zeeman features based on XPaschen code, original implementation in ADAS is ADAS603.

ADAS 605 input screen



□ Explicit Scaling X-min X-max

Use current values Y-min Y-max

Polarisation Filtering: PI+SIGMA

Observation angle (relative to field) 90.0000 degrees

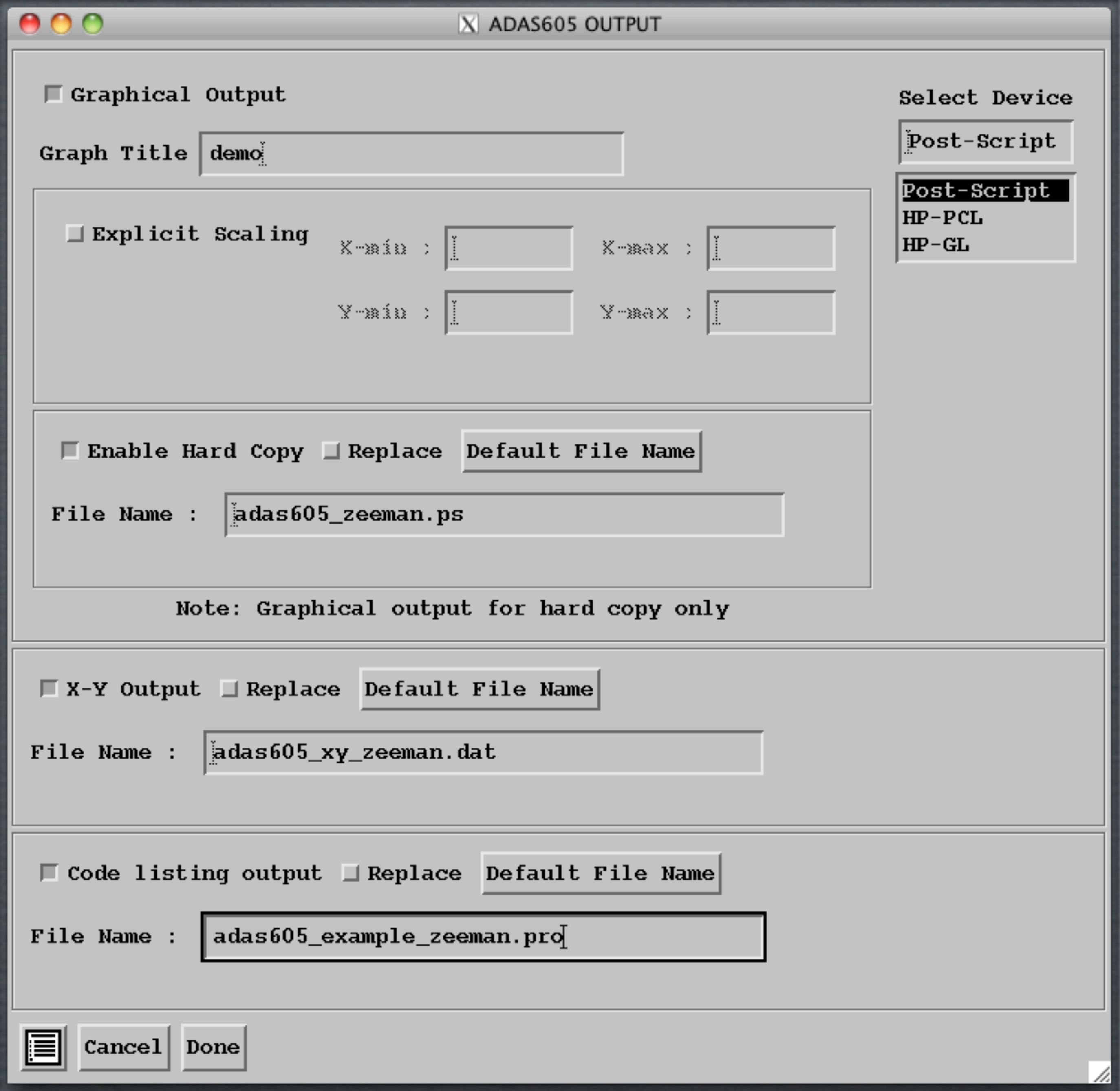
0.0 90.0

Magnetic field strength (T) 1.3000000 T

0.00000 20.0000

Feature index C I(2p (2P) 3s 3P - 3p 3P) 9086.8[A]

ADAS 605 PROCESSING SCREEN


ADAS605 OUTPUT

□ Graphical Output

Graph Title: demo

□ Explicit Scaling

X-min :

X-max :

Y-min :

Y-max :

□ Enable Hard Copy □ Replace

File Name : adas605_zeeman.ps

Note: Graphical output for hard copy only

Select Device: Post-Script

□ X-Y Output □ Replace

File Name : adas605_xy_zeeman.dat

□ Code listing output □ Replace

File Name : adas605_example_zeeman.pro

ADAS 605 OUTPUT SCREEN

Overview

- Framework for Feature Synthesis (FFS)
  - Managed data structure for modelling complex spectra, using a modular approach.
  - Provides a language for defining combination of spectral features that comprise a model.
  - Handles parameter attributes for numerical fitting.
  - Model definition language allows for coupling of parameters.

Model Definition Language

- LISP-like syntax.
- Model files define the main model construct as well as settings for the model:
  - parameter values
  - parameter coupling
  - parameter limits

FFS Model Definition Syntax

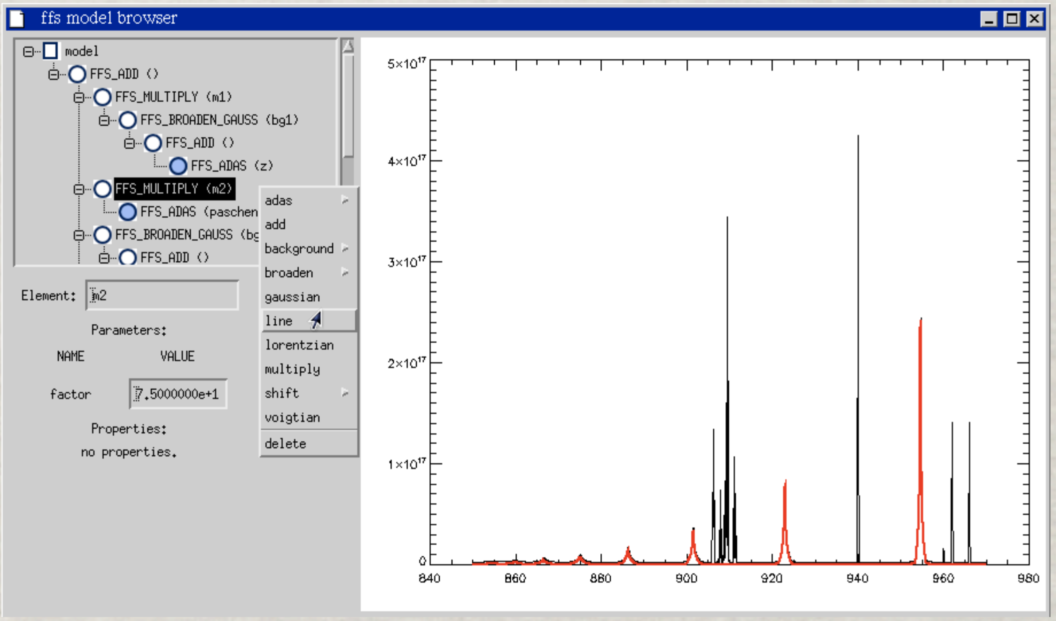
- An example model definition:

```
(model modelname
   (+
      (*
         (broaden_gauss
            (+
               (adas-zeeman z)
            )
         bg1)
      m1)
      (* (adas-paschen_archived paschen) m2)
      (broaden_gauss
         (+
            (line l1)
            (line l2)
            (line l3)
            (line l4)
            (line l5)
            (line l6)
         )
      bg2)
      (background-linear back)
   )
)
```

Format is specified as follows:

(elementclass[-optinput] operands name)

where the operands are further element expressions (optional) and ‘optinput’ allows for an additional string to be passed to that particular feature code.



ffs model browser

```
model
  FFS_ADD ()
    FFS_MULTIPLY (m1)
      FFS_BROADEN_GAUSS (bg1)
        FFS_ADD ()
          FFS_ADAS (z)
    FFS_MULTIPLY (m2)
      FFS_ADAS (paschen
    FFS_BROADEN_GAUSS (bg
      FFS_ADD ()
```

Element: m2

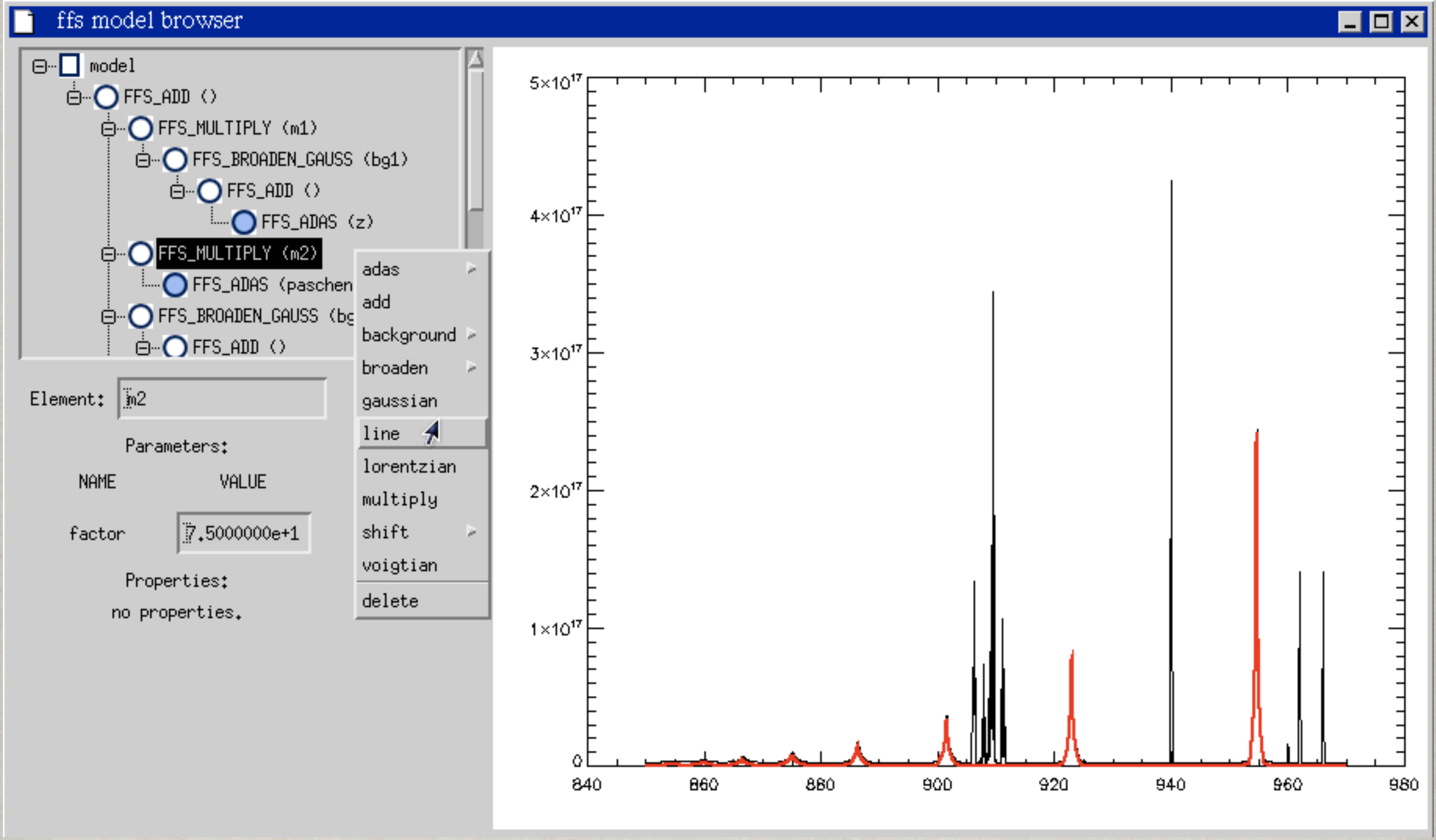
Parameters:

NAME	VALUE
factor	7.5000000e+1

Properties:

no properties.

FFS Model Browser (test version)



```
model
  FFS_ADD ()
    FFS_MULTIPLY (m1)
      FFS_BROADEN_GAUSS (bg1)
        FFS_ADD ()
          FFS_ADAS (z)
    FFS_MULTIPLY (m2)
      FFS_ADAS (paschen
    FFS_BROADEN_GAUSS (bg
      FFS_ADD ()
```

Element: m2

Parameters:

NAME	VALUE
factor	7.5000000e+1

Properties:

no properties.

Model Definition: Parameter Coupling and Limits

- Setting parameter coupling:
  - Line ‘l2’ is to be coupled such that it’s intensity is twice that of line ‘l1’:

    ```
    (couple l2.intensity (* l1.intensity 2.0))
    ```

Format is specified as follows:

(couple *elementname.partocouplename expression*)

The expressions are of the form:

(*operator* operands)

the operators are arithmetic (+, -, *, /, ^) and the operands are numeric values or further model parameters.

- Setting parameter limits:
  - (setmin l1.intensity 0.0)
  - (setlimits l1.intensity 0.0 60.0)

Format is specified as follows:

(setmin *elementname.parname value*)

or

(setlimits *elementname.parname min max*)

- Note that special features coming from AFG have limits imposed automatically using the description structure supplied by AFG.

Model 'Simplification'

- Takes input model definition and provides a more optimal representation of the model.
- Can provide more efficient calculation of model spectra.
- Can enable use of analytical expressions for model partial derivatives with respect to the parameters.
- Opaque to the user - 'simplified' model used for evaluation and partial derivatives, but linked back to original user specified model using the coupling system.
  - This means for an arbitrarily complex model the code implicitly/explicitly does the necessary maths to determine what the analytic partial derivative is.

ELEMENT COMBINATIONS

Consider a Gaussian broadening function:

$$\begin{aligned} &B_g\left\{f\{\dots\},w_g\right\}(x) \\ &=\left[G\{w_g\}*f\{\dots\}\right](x) \\ &=\int_{-\infty}^{+\infty} G\left\{w_g\right\}(x-x')f\{\dots\}(x')dx' \end{aligned}$$

$$G\{w_g\}(x)=\frac{C}{\sqrt{\pi}w_g}\exp\left(-\frac{C^2x^2}{w_g^2}\right)$$

$$I_g\{\lambda_0,\phi,w_g\}(\lambda)=\frac{C\phi}{\sqrt{\pi}w_g}\exp\left(-\frac{C^2(\lambda-\lambda_0)^2}{w_g^2}\right)$$

$$\begin{aligned} \frac{\partial}{\partial\lambda_0}I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda)&=\frac{2C^2\left(\lambda-\lambda_0\right)}{w_g^2}I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda) \\ \frac{\partial}{\partial w_g}I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda)&=\frac{1}{w_g}\left(2C^2\frac{\left(\lambda-\lambda_0\right)^2}{w_g^2}-1\right)I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda) \\ \frac{\partial}{\partial\phi}I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda)&=\frac{1}{\phi}I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda) \end{aligned}$$

$$C=2\sqrt{\ln 2}$$

Apply the gaussian broadener to a gaussian line and we get another gaussian:

$$\begin{aligned} I_{\mathrm{g(new)}}\{\lambda_{0(\mathrm{new})},\phi_{(\mathrm{new})},w_{g(\mathrm{new})}\}(\lambda)&=B_g\left\{I_g\left\{\lambda_0,\phi,w_{g1}\ \right\},w_{g2}\right\}(\lambda) \\ &=\frac{C}{\sqrt{\pi}\sqrt{w_{g_1}^2+w_{g_2}^2}}\exp\left(\frac{-C^2\left(\lambda-\lambda_0\right)^2}{w_{g_1}^2+w_{g_2}^2}\right) \end{aligned}$$

ELEMENT COMBINATIONS

Consider a Gaussian broadening function:

$$
\begin{aligned}
&B_g\left\{f\{\dots\}, w_g\right\}(x)\\
&= \left[G\{w_g\} * f\{\dots\}\right](x)\\
&= \int_{-\infty}^{+\infty} G\left\{w_g\right\}(x-x')f\{\dots\}(x')dx'
\end{aligned}
$$

$$G\{w_g\}(x) = \frac{C}{\sqrt{\pi}w_g}\exp\left(-\frac{C^2x^2}{w_g^2}\right)$$

$$I_g\{\lambda_0, \phi, \boxed{w_g}\}(\lambda) = \frac{C\phi}{\sqrt{\pi}w_g}\exp\left(-\frac{C^2(\lambda-\lambda_0)^2}{w_g^2}\right)$$

$$
\begin{aligned}
\frac{\partial}{\partial\lambda_0}I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda) &= \frac{2C^2\left(\lambda-\lambda_0\right)}{w_g^2}I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda)\\
\frac{\partial}{\partial w_g}I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda) &= \frac{1}{w_g}\left(2C^2\frac{(\lambda-\lambda_0)^2}{w_g^2}-1\right)I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda)\\
\frac{\partial}{\partial\phi}I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda) &= \frac{1}{\phi}I_g\left\{\lambda_0,\phi,w_g\right\}(\lambda)
\end{aligned}
$$

$$C = 2\sqrt{\ln 2}$$

Apply the gaussian broadener to a gaussian line and we get another gaussian:

$$
\begin{aligned}
I_{\mathrm{g(new)}}\{\lambda_{0(\mathrm{new})}, \phi_{(\mathrm{new})}, w_{g(\mathrm{new})}\}(\lambda) &= B_g\left\{I_g\left\{\lambda_0, \phi, w_{g1}\right\}, w_{g2}\right\}(\lambda)\\
&= \frac{C}{\sqrt{\pi}\boxed{\sqrt{w_{g1}^2+w_{g2}^2}}}\exp\left(\frac{-C^2\left(\lambda-\lambda_0\right)^2}{w_{g1}^2+w_{g2}^2}\right)
\end{aligned}
$$

FFS 'Simplification' Example

```
(model original
  (+
    (broaden_gauss
      (+
        (gaussian g1)
        (lorentzian l1)
        (broaden_lorentz
          (+
            (line theline)
            (adas-zeeman az)
          )
        blorentz)
      )
    bgauss)
  )
)
```

```
(model simplified
  (+
    (gaussian new_gauss)
    (voigtian new_voigt)
    (voigtian new_voigt)
    (broaden_voigt
      (+
        (adas-zeeman new_az)
      )
     new_bvoigt)
  )
)
```

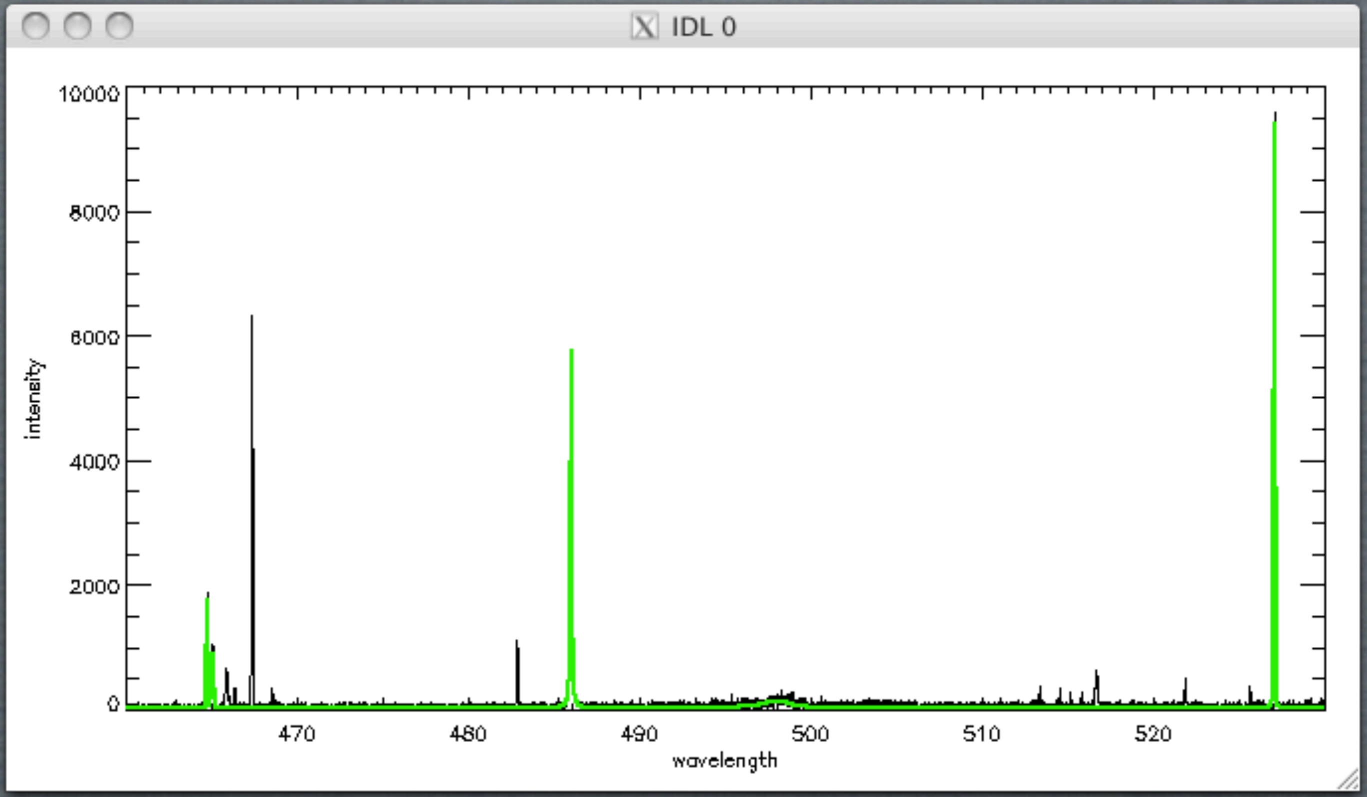
Internally, expressions such as:

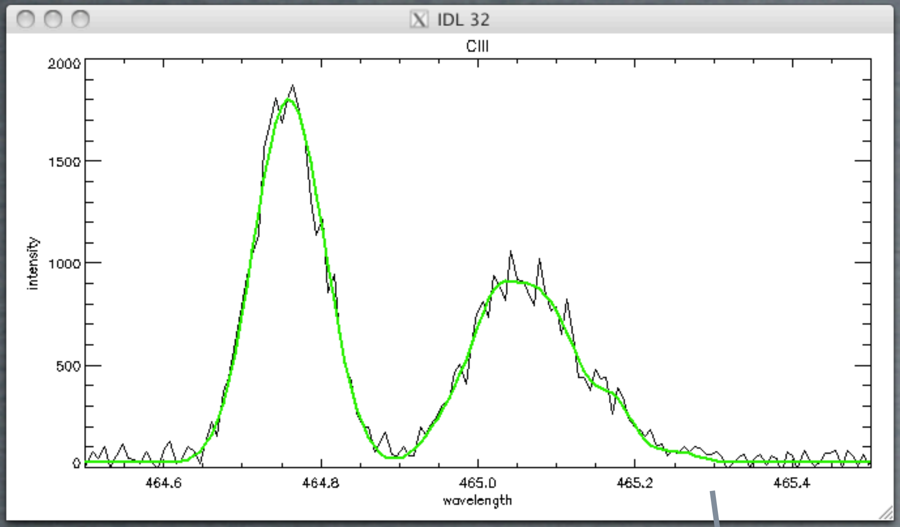
```
(couple new_gauss (^ (+ (^ (* bgauss.fwhm 1.0) 2) (^ g1.fwhm 2)) 0.5) ))
```

are formed to couple the parameters back to the original parameter set.

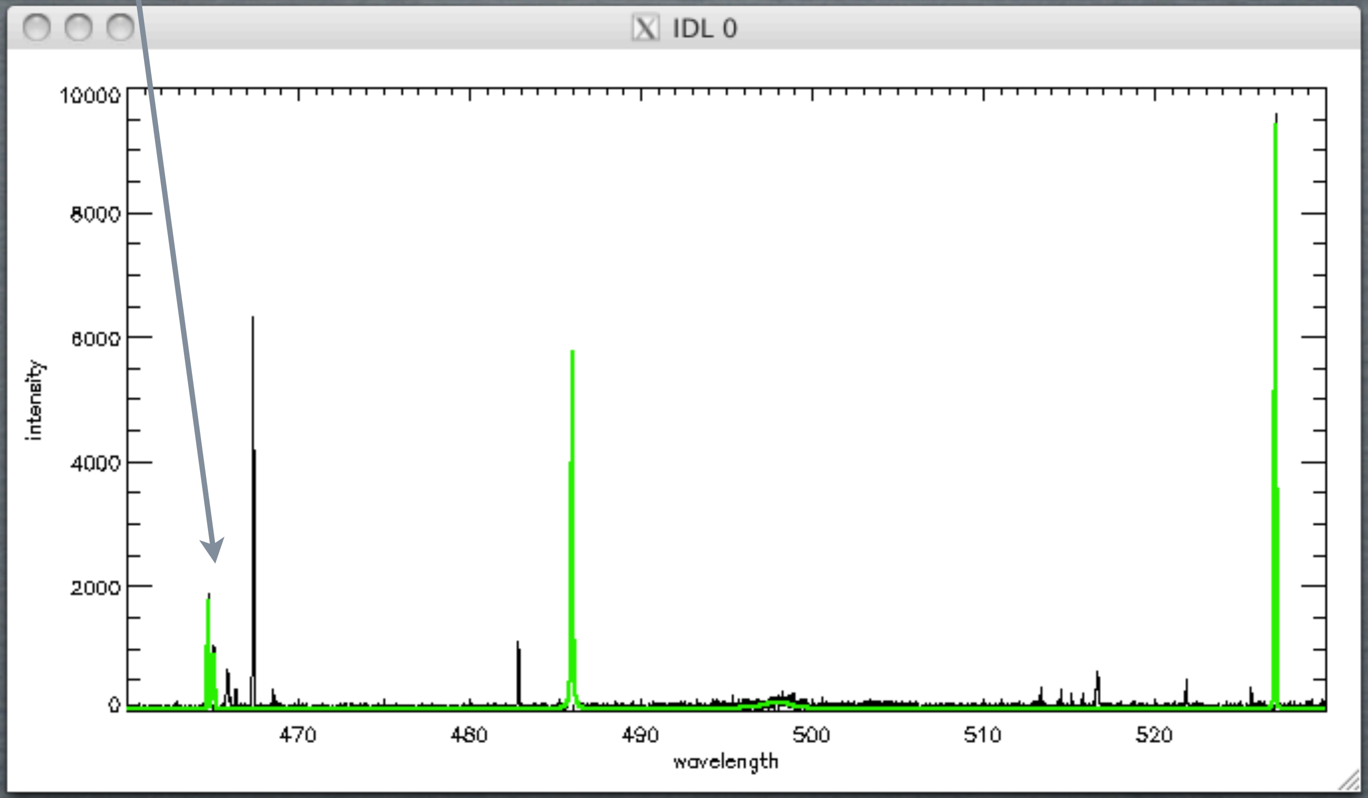
FFS USING AFG

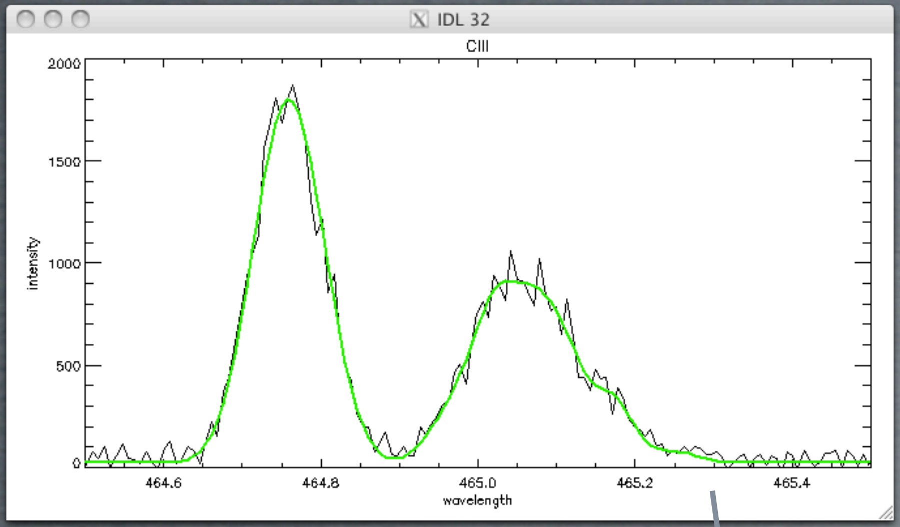
```
(model zeeman

    (+
        (shift-lambda
            (+
                (broaden_gauss
                    (* (adas-zeeman ciiilow) ciiilowmult)
                bg1)
                (broaden_gauss
                    (* (adas-zeeman ciiihigh) ciiihighmult)
                bg2)
            )
        sh1)

        (shift-lambda
            (+
                (broaden_gauss
                    (* (adas-zeeman beiilow) beiilowmult)
                bg3)
                (broaden_gauss
                    (* (adas-zeeman beiihigh) beiihighmult)
                bg4)
            )
        sh2)

        (voigtian dbeta)

        (broaden_gauss
            (* (adas-picket bed) bedmult)
        bg6)

        (background-linear backg)
    )
)
```


IDL 0
10000
8000
6000
4000
2000
0
intensity
470
480
490
500
510
520
wavelength

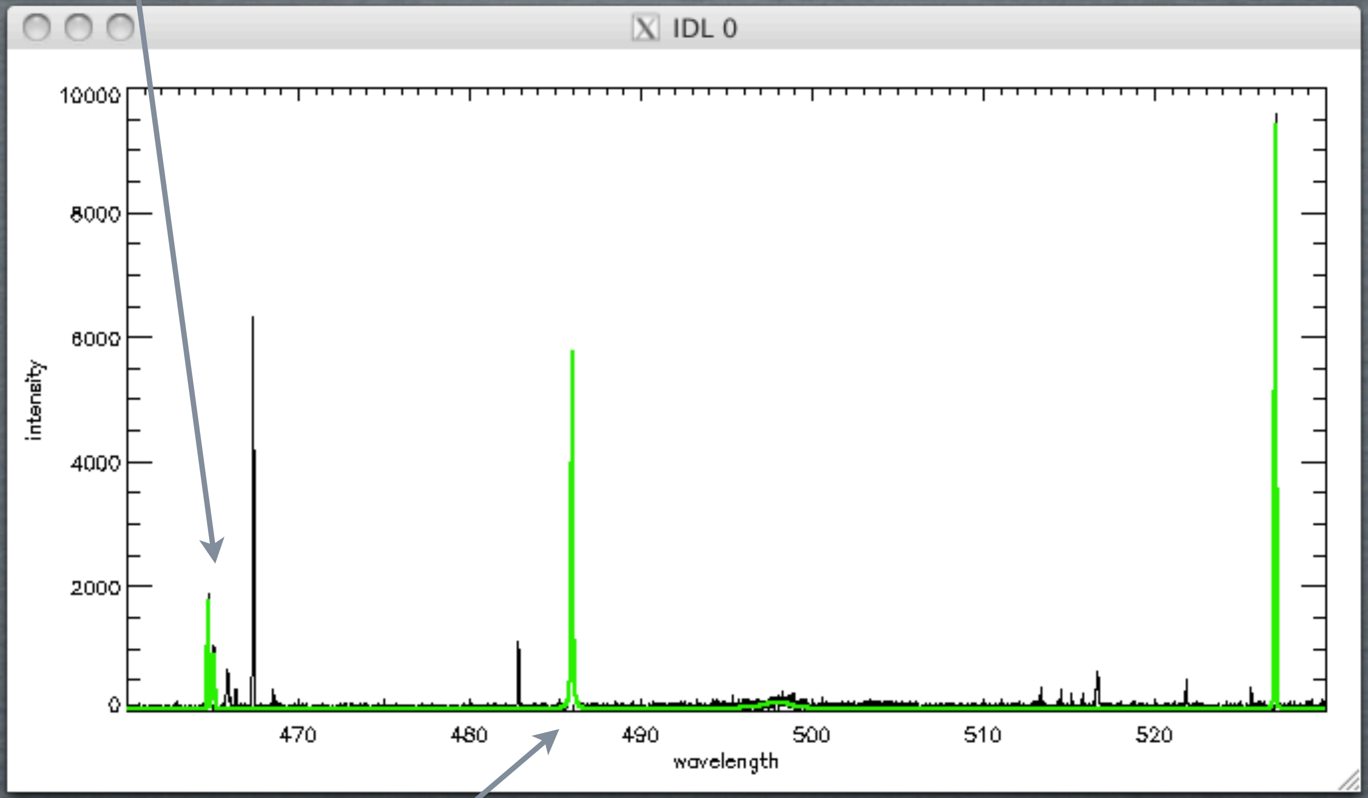


AFG
Zeeman
(ciiihigh &
ciiilow)

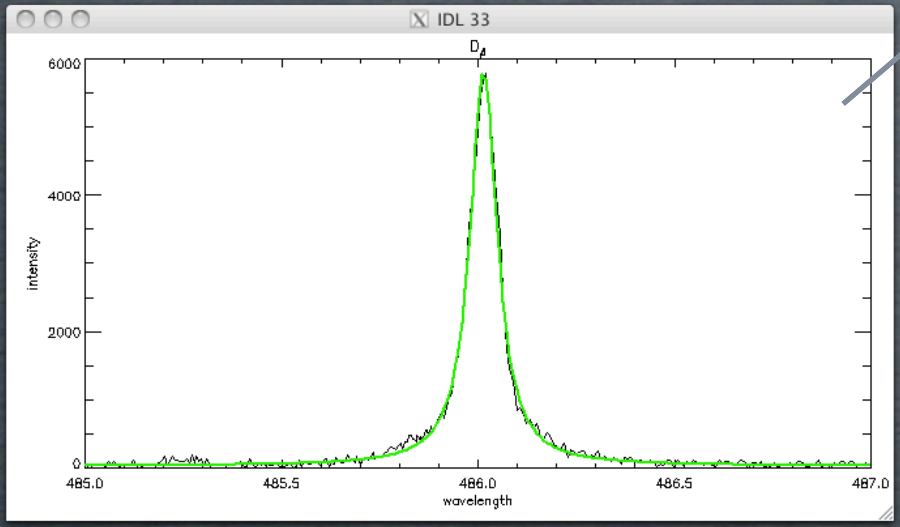


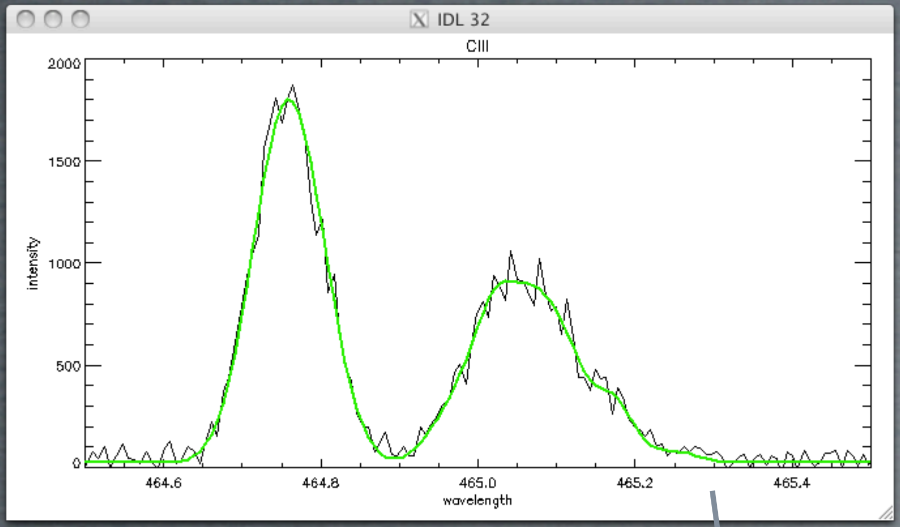


AFG
Zeeman
(ciiihigh &
ciiilow)

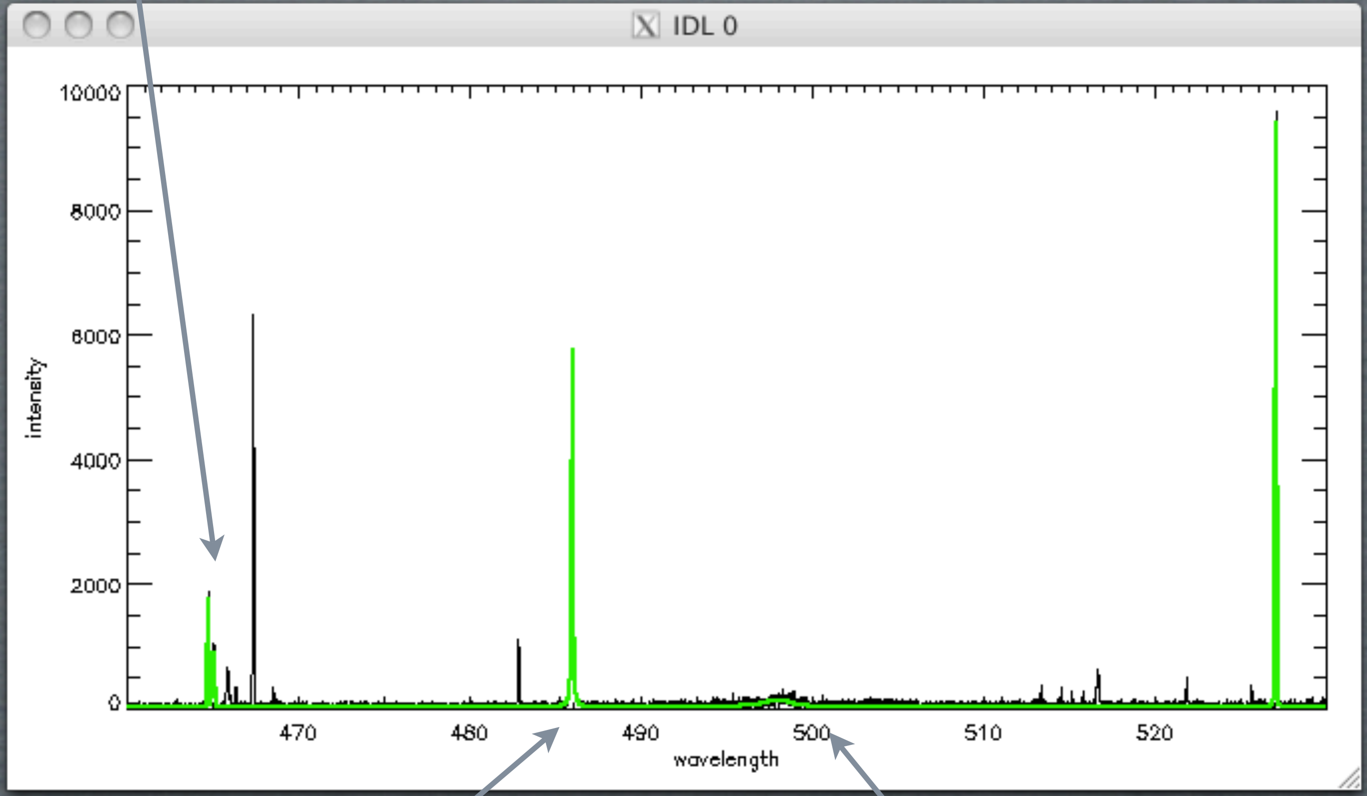


Voigt (dbeta)



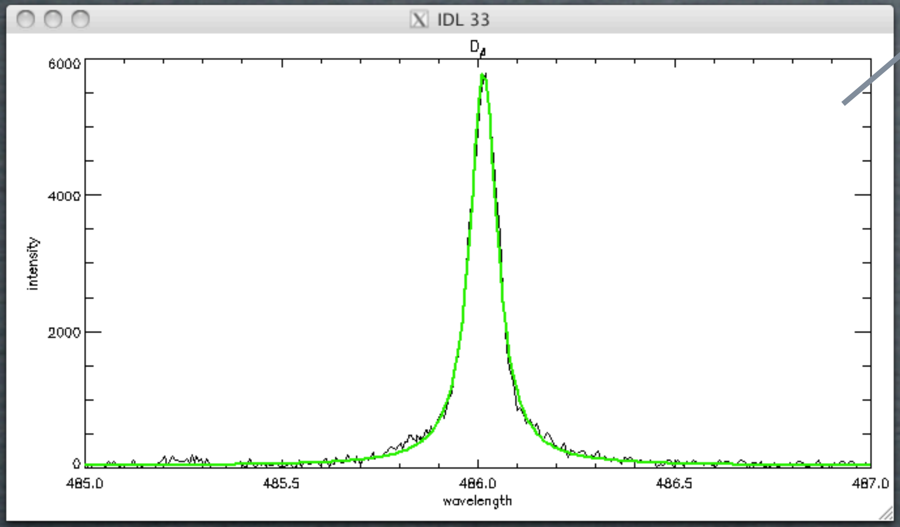


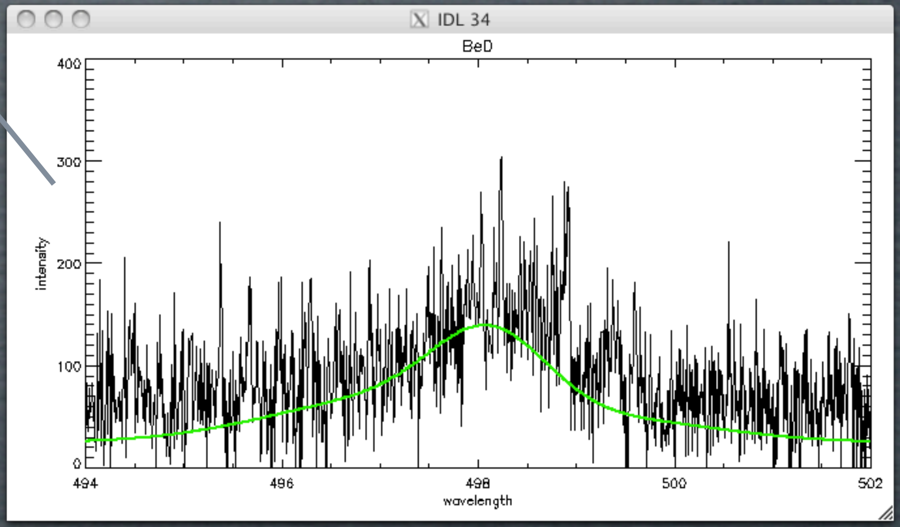
AFG
Zeeman
(ciiihigh &
ciiilow)

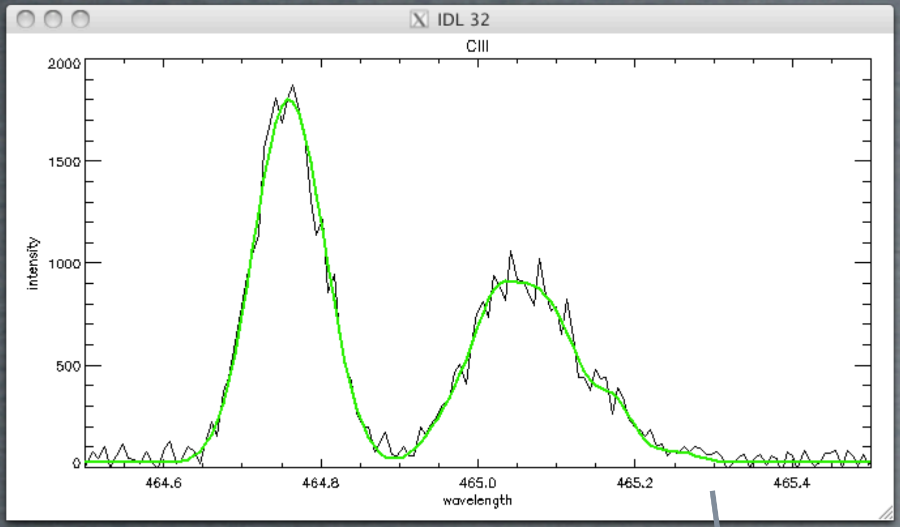


Voigt (dbeta)

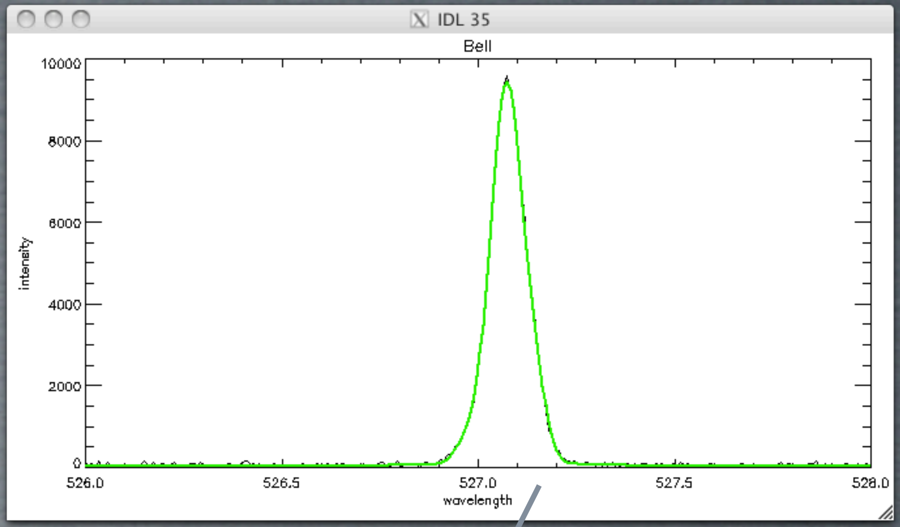
AFG Picket (bed)



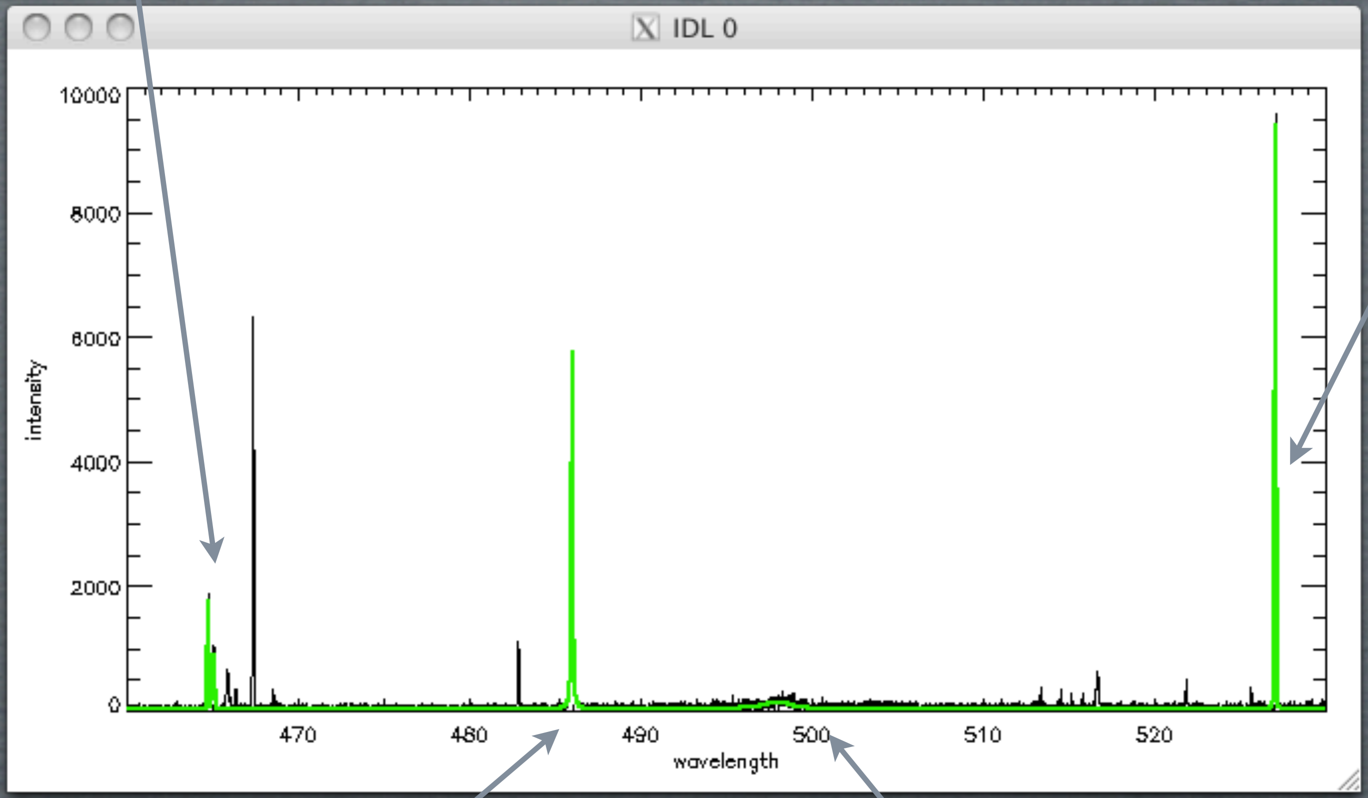




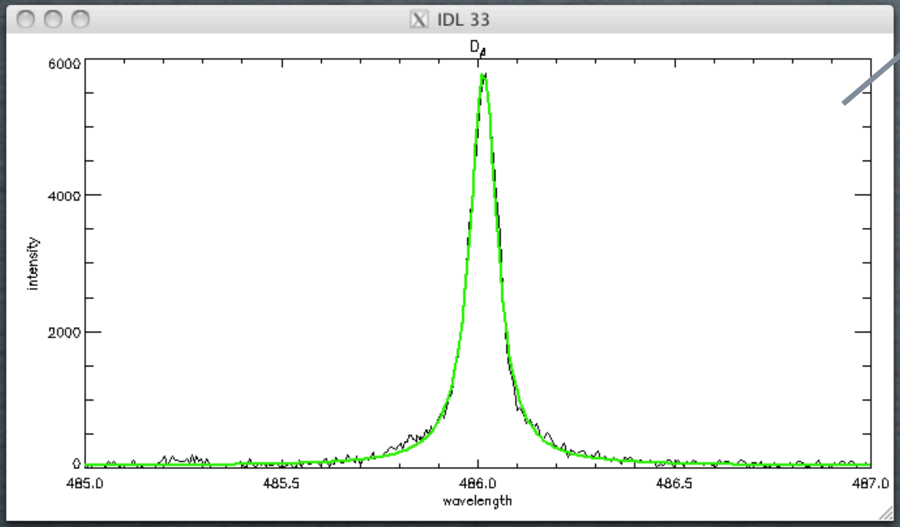
AFG Zeeman (ciiihigh & ciiilow)



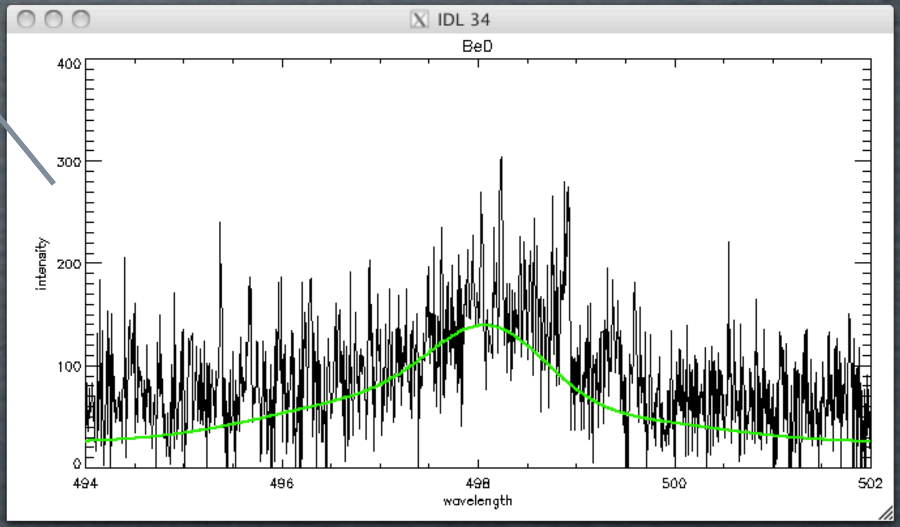
AFG Zeeman (beiihigh & beiilow)



Voigt (dbeta)



AFG Picket (bed)



Summary

- AFG provides easy, common interface to ADAS special feature models.
- FFS will provide unified approach to modelling arbitrarily complex spectra.
- ADAS Feature Generation (AFG) routines are included in the latest release of ADAS.
- GUI to AFG is now available - ADAS 605.
- FFS still has some time to reach maturity and is expected sometime next year.

Adhering to Standards...

- Greenspun's tenth rule of programming:
  - "Any sufficiently complicated C or Fortran program contains and ad-hoc, informally-specified, bug-ridden, slow implementation of half of common LISP."