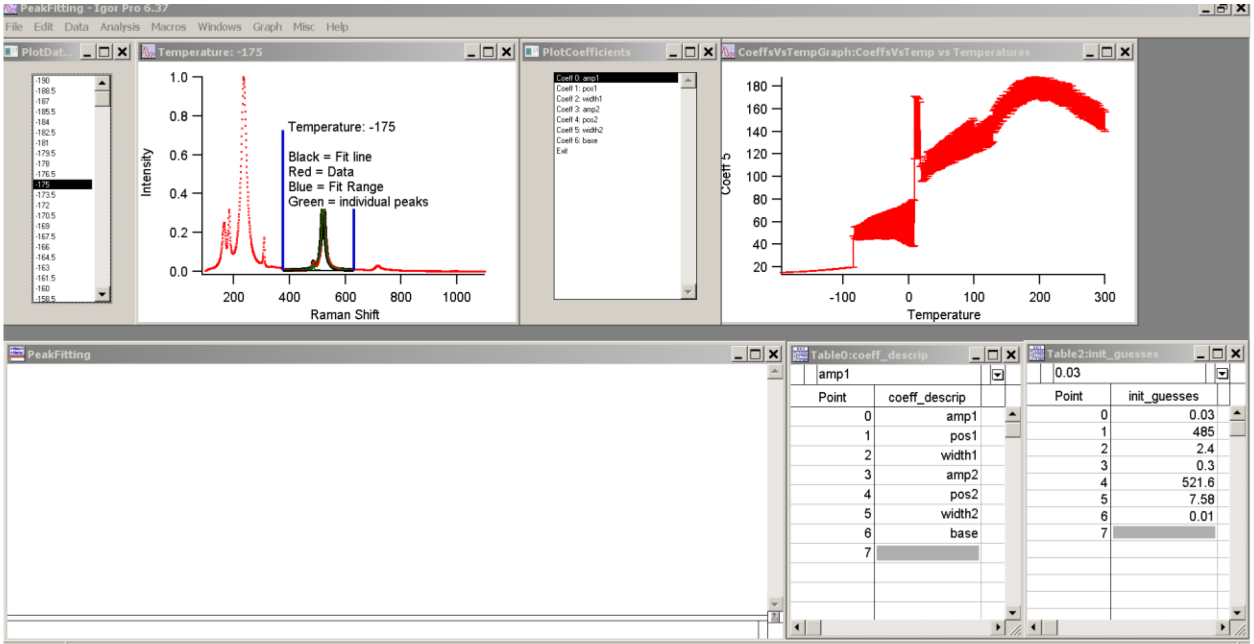
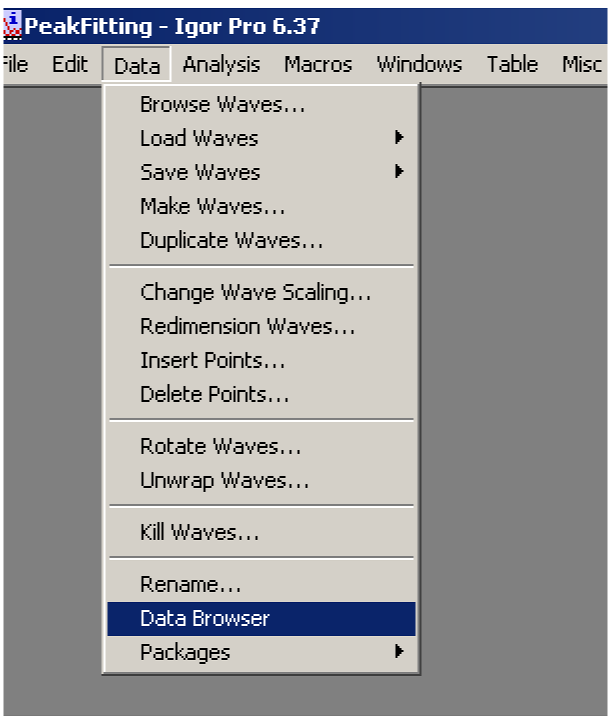
Batch Peak Fitting

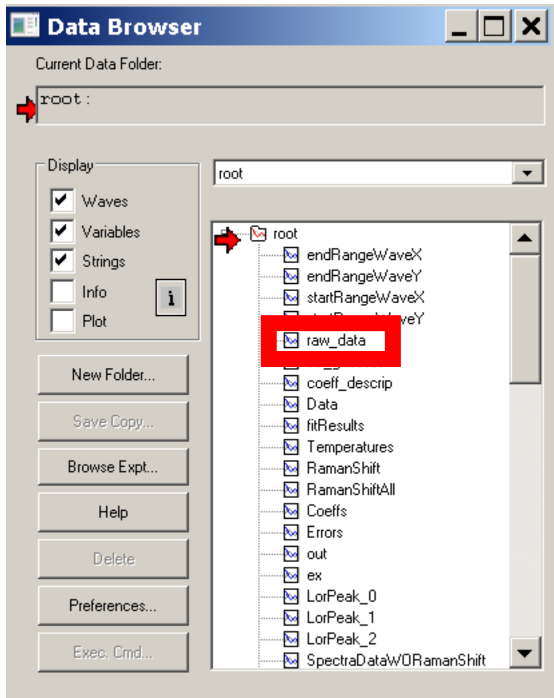
1. Must use Igor Pro 6.37.2
2. Below is what you will see when you open the program



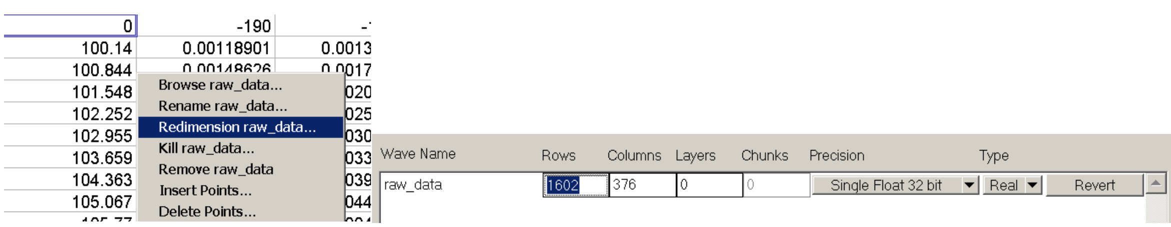
3. Navigate to Data→Data Browser



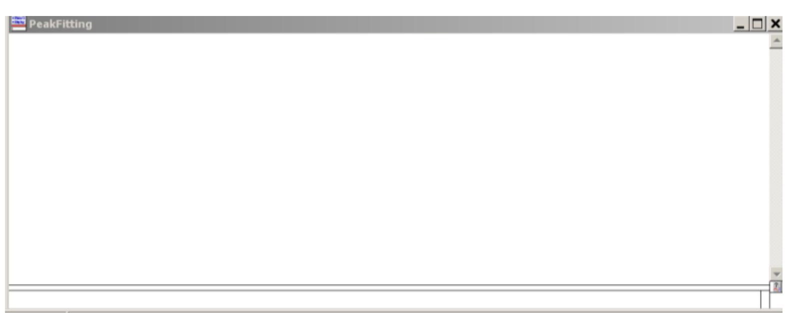
4. This will open up the following panel. The panel shows all the Waves (the Igor variable type) in the experiment (pxp file).



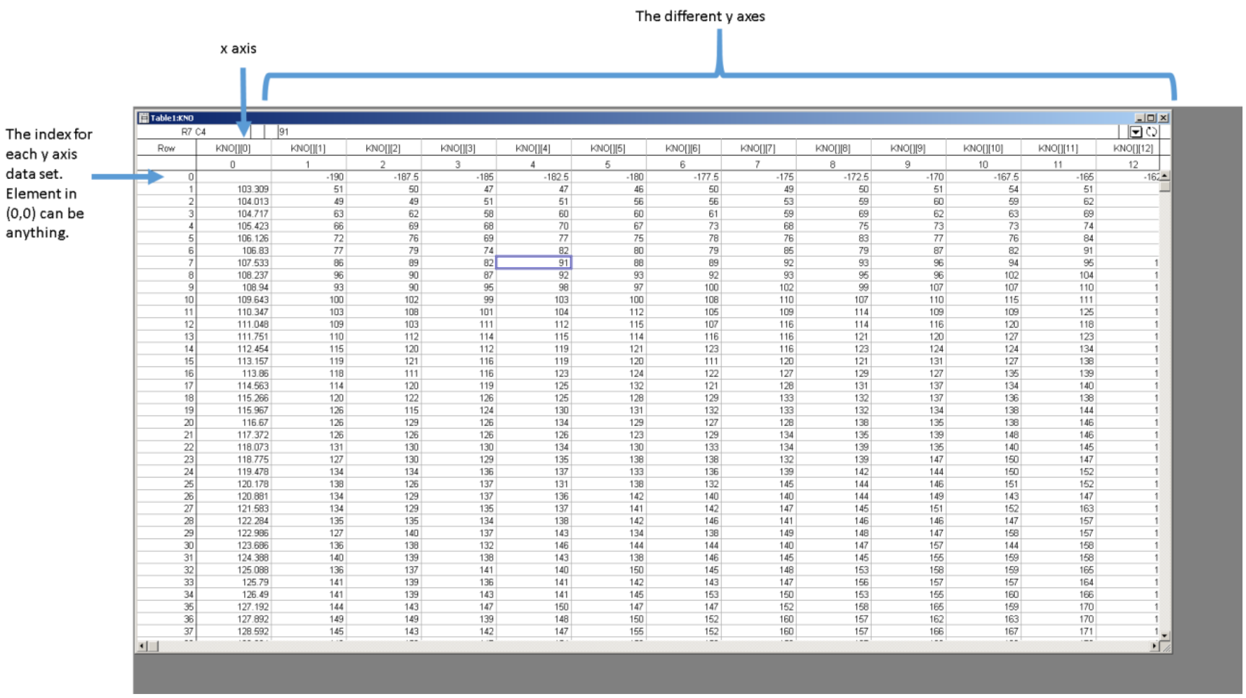
5. We are interested in the wave called raw_data. This is the input file
   a. This is the matrix that will contain the data you want to fit. There are several requirements for the format of this matrix.
   b. The x axis will run be in the first column
   c. The y axes will be in the second, third columns
   d. The first row has no data. The first row contains the index describing each y axes.
   e. Below is an example input wave. Use Make/O/N=(# columns, # rows) NAME_OF_WAVE to make a new wave
   f. To resize the wave right click on an element and select “Redimension…” Act accordingly…



6. Now go to the command window

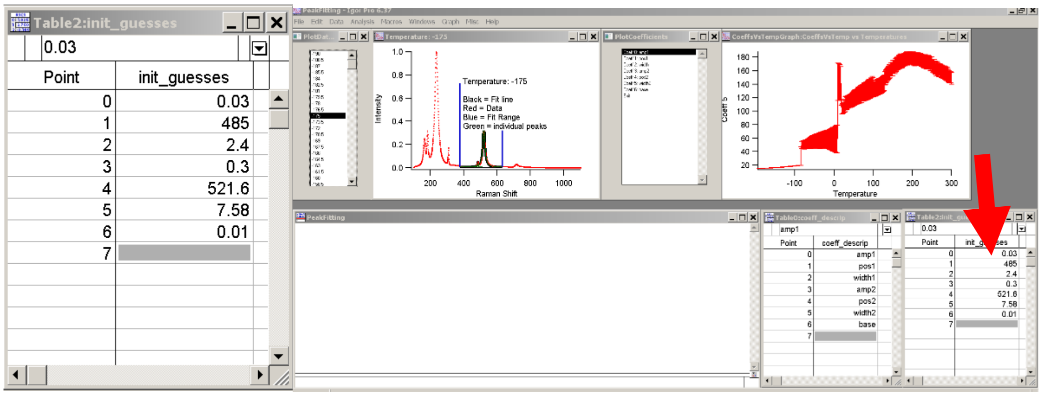


7. Type the following in: PeakFitting(raw_data, start##,end##,"0...1...",init_guesses, coeff_descrip)
   a. raw_data
      i. This is the matrix that will contain the data you want to fit. There are several requirements for the format of this matrix.
      ii. The x axis will run be in the first column
      iii. The y axes will be in the second, third columns
      iv. The first row has no data. The first row contains the index describing each y axes.
      v. Below is an example input wave

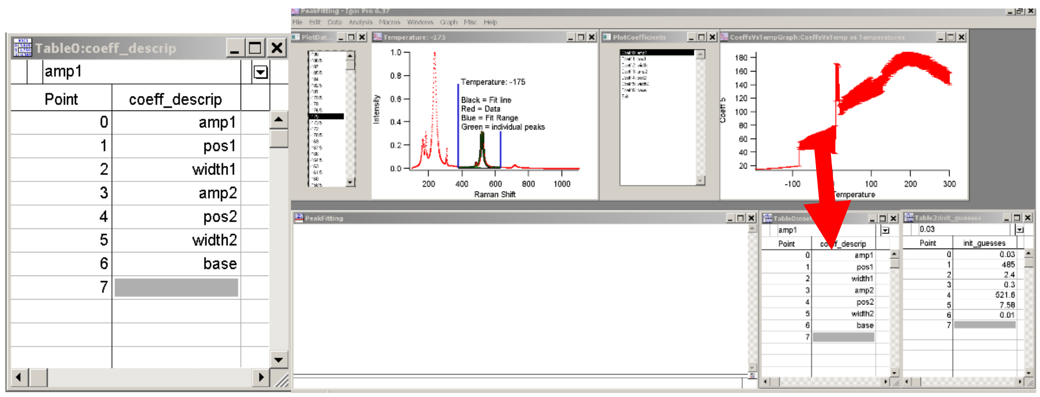


Row	KNO[][0]	KNO[][1]	KNO[][2]	KNO[][3]	KNO[][4]	KNO[][5]	KNO[][6]	KNO[][7]	KNO[][8]	KNO[][9]	KNO[][10]	KNO[][11]	KNO[][12]
	0	1	2	3	4	5	6	7	8	9	10	11	12
0		-190	-187.5	-185	-182.5	-180	-177.5	-175	-172.5	-170	-167.5	-165	-162
1	103.309	51	50	47	47	46	50	49	50	51	54	51	
2	104.013	49	49	51	51	56	56	53	59	60	59	62	
3	104.717	63	62	58	60	60	61	59	69	62	63	69	
4	105.423	66	69	68	70	67	73	68	75	73	73	74	
5	106.126	72	76	69	77	75	78	76	83	77	76	84	
6	106.83	77	79	74	82	80	79	85	79	87	82	91	
7	107.533	86	89	82	91	88	89	92	93	96	94	95	1
8	108.237	96	90	87	92	93	92	93	95	96	102	104	1
9	108.94	93	90	95	98	97	100	102	99	107	107	110	1
10	109.643	100	102	99	103	100	108	110	107	110	115	111	1
11	110.347	103	108	101	104	112	105	109	114	109	109	125	1
12	111.048	109	103	111	112	115	107	116	114	116	120	118	1
13	111.751	110	112	114	115	114	116	116	121	120	127	123	1
14	112.454	115	120	112	119	121	123	116	123	124	124	134	1
15	113.157	119	121	116	119	120	111	120	121	131	127	138	1
16	113.86	118	111	116	123	124	122	127	129	127	135	139	1
17	114.563	114	120	119	125	132	121	128	131	137	134	140	1
18	115.266	120	122	126	125	128	129	133	132	137	136	138	1
19	115.967	126	115	124	130	131	132	133	132	134	138	144	1
20	116.67	126	129	126	134	129	127	128	138	135	138	146	1
21	117.372	126	126	126	126	123	129	134	135	139	148	146	1
22	118.073	131	130	130	134	130	133	134	139	135	140	145	1
23	118.775	127	130	129	135	138	138	132	139	147	150	147	1
24	119.478	134	134	136	137	133	136	139	142	144	150	152	1
25	120.178	138	126	137	131	138	132	145	144	146	151	152	1
26	120.881	134	129	137	136	142	140	140	144	149	143	147	1
27	121.583	134	129	135	137	141	142	147	145	151	152	163	1
28	122.284	135	135	134	138	142	146	141	146	146	147	157	1
29	122.986	127	140	137	143	134	138	149	148	147	158	157	1
30	123.686	136	138	132	146	144	144	140	147	157	144	158	1
31	124.388	140	139	138	143	138	146	145	145	155	159	158	1
32	125.088	136	137	141	140	150	145	148	153	158	159	165	1
33	125.79	141	139	136	141	142	143	147	156	157	157	164	1
34	126.49	141	139	143	141	145	153	150	153	155	160	166	1
35	127.192	144	143	147	150	147	147	152	158	165	159	170	1
36	127.892	149	149	139	148	150	152	160	157	162	163	170	1
37	128.592	145	143	142	147	155	152	160	157	166	167	171	1

   b. start#. Enter a value between 1 and x axis length
      i. x axis length = the number of points in the x axis. Number of rows in the matrix – 1
      ii. This number will be the left bound of the fitting range.
   c. end#
      i. Enter a value between 1 and x axis length
      ii. This number will the right bound of the fitting range
   d. “0…1…”
      i. This is the hold string.
      ii. 1 = hold that variable constant, aligns with the coefficient waves
      iii. Length should equal the coefficient waves
   e. init_guesses
      i. This wave will contain all the initial coefficients. Initial conditions are very important to get good fits. Best to manually fit one data set to get initial guesses

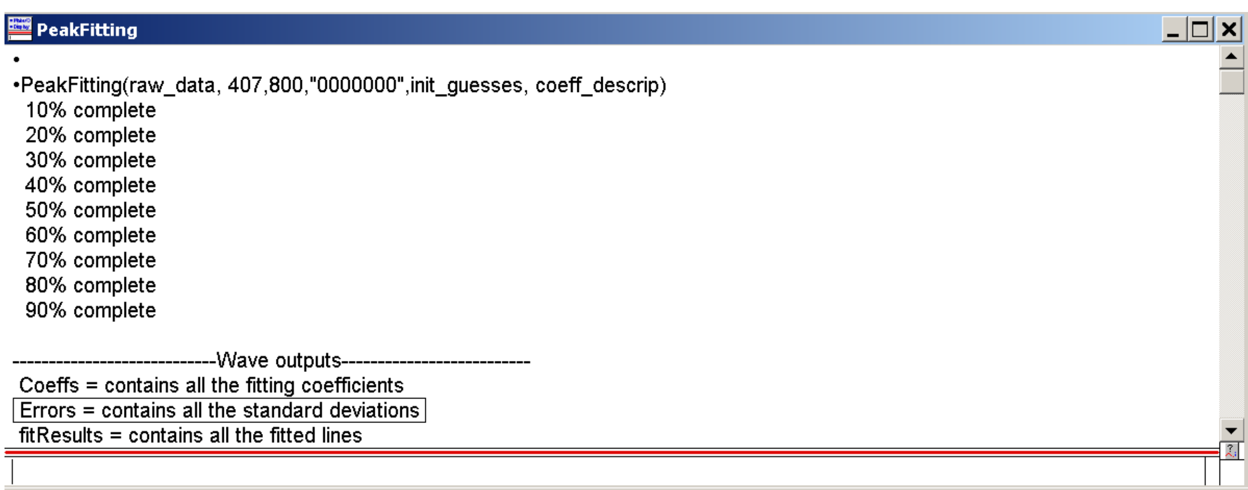


f. coeff_descrip



i. This wave will be a text wave
ii. Each string describes a coefficient
iii. Length must match up with number of coefficients

8. Now run! ☺
9. The result is:

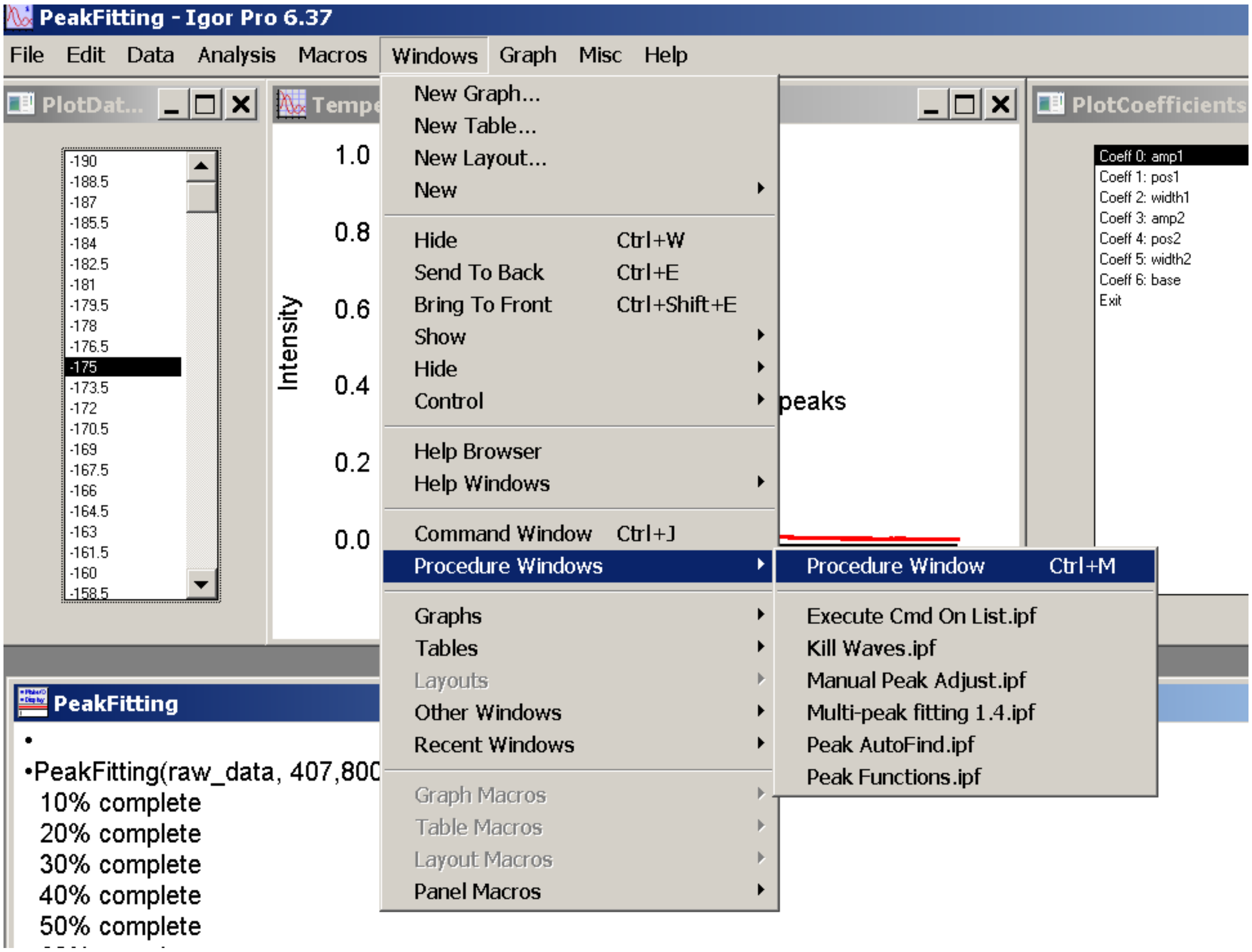


10. In the command window, we see that Coeffs, Errors, and fitResults are created. Use the Data Browser to find these waves
    a. The Coeffs wave contains all the coefficients generated by the fitting

        i. Each column in the Coeffs corresponds to a different y vs x fit. There as many Coeff columns as there are columns in the input data – 1. Minus 1 because the first column is the x axis.
    b. The Errors wave contains all the errors. These errors, are according to Igor, standard deviations for the fitting.
        ii. Same format as Coeffs wave. In fact each element in the Errors wave gives the corresponding error for the element in the Coeffs wave!
    c. The fitResults wave contains the results of the fit: plugging in x values into the fitting equation nwith the associated coefficients
        iii. First column is the x axis binned to the start and end fitting values (values we modify in the initial panel)
        iv. Other columns are just the fits
11. We will also see that
12. The two scrolling lists are the GUI outputs. Click a value and right click to plot.

Changing the Fitting Function

1. We can batch fit any kind of function. It doesn't have to be for spectral analysis. We can also fit a baseline…but this isn't a true baseline in that this baseline is part of the fitting function. A true baseline would baseline subtract the data first. Regardless, such fitting is a useful first approximation.
2. First Navigate to Windows→Procedure Windows→Procedure Window (or Ctrl+M)



3. First we see:

```
Function func(w,x) : FitFunc
	Wave w
	Variable x

	return (w[0] * w[2] / 2) * ( (0.5 * w[2])  /  ( (x - w[1])^2 + (0.5 * w[2])^2 ) ) + (w[3] * w[5] / 2) * ( (0.5 * w[5])  /  ( (x - w[4])^2 + (0.5 * w[5])^2 ) ) + w[6]
End
```

   a. FitFunc is the fitting function
   b. Simply type in the function you wish to fit at the return… (blue arrow)
   c. The values in the coefficient wave will match up sequentially with the descriptions wave and the initial conditions.

d. Currently, in the figure, the function is the sum of two Lorentzians and a baseline
e. The baseline function has to be the last terms. So have the Lorentzians equation and then have the baseline part.

4. We also see:

```
constant NonLorTermsInEqu = 1 // number of parameters in the equation that are not Lorentzian
```

a. This is the number of terms that characterize the baseline.
b. Right now, we only have a constant baseline, so the value is 1

5. We also see:

```
Function FitFuncEval(w, x)
	// Evaluate the fit func for some x values
	// Use the same exact equation as Function func!!!
	Wave w // The coefficients for the function
	Wave x // The x values to evalulate for

	Make /O/N=(DimSize(x,0)) out

	variable i
	for(i = 0; i < DimSize(x, 0); i+=1)
		out[i] = (w[0] * w[2] / 2) * ((0.5 * w[2]) / ( (x - w[1])^2 + (0.5 * w[2])^2) ) + (w[3] * w[5] / 2) * (  (0.5 * w[5])   /   (  (x - w[4])^2 + (0.5 * w[5])^2  )  ) + w[6]
	endfor

	return out

End
```

a. Simply copy and paste what we changed for the FitFunc function (the return… statement) here

6. Last, is the EvalBaseline

```
Function EvalBaseline(w,x)

	Wave w // Coeffs
	Wave x // x values

	Make /O/N = (DimSize(x,0)) out

	variable i
	for(i = 0; i < DimSize(x, 0); i+=1)
		out[i] = w[0]
	endfor

	return out

End
```

a. This is the baseline function. The number of terms after out[i]… should equal the value in Step 4. Just copy the baseline part of your function. Again, this is not a true baseline because it is not baseline subtracting, but rather fitting the baseline as part of the function. Still, this is gives good quick and dirty results.
b. The coefficients for w in EvalBaseline start from 0 and increment by 1

7. That's it! Do not modify anything else. In fact, don't even scroll down further…

8. If you do change the peak fitting function from a Lorentzian, you will also have to modify the below function (it is at the bottom of the code)

```
// Create a Lorentzian function
Function EvalLor(w, x, minBaseline)

    Wave w // The coefficients for the function
    Wave x // The x values to evalulate for
    Variable minBaseline // constant to add to each peak

    Make /O/N=(DimSize(x,0)) out

    variable i
    for(i = 0; i < DimSize(x, 0); i+=1)
        out[i] = (w[0] * w[2] / 2) * ((0.5 * w[2]) / ( (x - w[1])^2 + (0.5 * w[2])^2 ) ) + minBaseline
    endfor

    return out

End
```

9. Change the stuff in the blue brackets to accurately reflect the peak function. For example, if you changed the Lorentzian to a Gaussian, change the blue stuff to a Gaussian peak function.
10. In PeakFittingGauss the parameters are:
    a. Amplitude
    b. Position
    c. FWHM
11. In PeakFittingLor the parameters are:
    a. Amplitude
    b. Position
    c. FWHM

Modified:

August 31, 2016

Added PeakFittingGauss = Uses Gaussians instead of Lorentzians

Geoffrey Xiao (gx26@drexel.edu)

Created:

August 24, 2016

Geoffrey Xiao (gx26@drexel.edu)