The performance of the spectral synthesis is usually dominated by either the matrix calculation step or the convolution step. Because there are a number of different ways to improve the accuracy of a spectrum, the optimal approach depends on which of the two steps is limiting. For example, if the matrix calculation is limiting, which is the case for a large number of spectral lines, the grid size can be reduced without incurring any penalty. Likewise, if the convolution is the limiting step, which is the case for a large number of spectral points, using improved weights such as the optimized weights can be used to reduce the number of convolutions at a given accuracy.

The optimized weights were able to improve the result because now instead of a single parameter per axis (i.e. t) there are three available, which allows for finer optimization. In addition to using the Gaussians part to optimize the Lorentzian error or vice-versa, as was the case with optimized weights, one might consider spreading a spectral line over not 2 bins (on a single axis), but over 3, 4, … .

In the case of 2 bins, which covered all cases in the original paper, only one weight was used, which for the simple weights was a = t. To generalize this approach to more bins, it is convenient to consider the 2-bin case as having two weights, namely and respectively for the bins with index 0 and 1.

As shown before, the error in Fourier can be expressed as a power series in :

With for the line position axis, and for the width axes. Here the comes from the fact that the Lorentzian and Gaussian widths are multiplicative with respect to the spectral coordinate, whereas the line position is additive; it is *not* related to the choice of interpolating the grid linearly vs. exponentially/logarithmically.

The approximated lineshape is identical to the exact lineshape when the 0th order term in eq1 is unity and all higher order terms are zero. This requirement can be used to solve for and , namely:

Solving for and yields the original simple weights, i.e. and \*.

\*Footnote: The power series of the approximated lineshape was developed in Fourier space, as opposed to real space, precisely because eliminating the first order term yields the simple weights. Compared to real space, the error terms are simpler because in Fourier space the lineshape intensity doesn’t scale inversely with the width. In principle the same procedure could have been applied in real space, but the power series would have been more complicated because the lineshapes need to be scaled (normalized) by the widths. The power series of the approximated lineshape is then given by *.* While this optimization would results in slightly smaller errors, the benefit of having a simple expression for the weights was deemed preferable.

The discretization error for this choice of weights is then given by the first non-zero term, which in this case is the second order term:

Extending to more than two bins follows the same recipe, solving for weights setting higher orders to zero. For three bins, the approximated lineshape is developed around:

For an odd number of bins, the range of the grid alignment changes to to reflect the symmetry around the central bin\*, as opposed to for even bins.

\*Footnote: A minor side effect is that the approximated lineshape is no longer continuous when scanning the width-axis. For an even number of bins, the “jump” occurs when t=1(=0), where the approximation is exact. For an odd number of bins, the “jump” happens when t=1/2(=-1/2), which is where the approximation error is largest. When jumping from one set of bins to the next, the sign of the error switches resulting in a sudden ‘step’. In application this is not an issue however, because any given line has only one value for its widths, and will never “traverse” the edge of a bin.

The system of equations to be solved is given by:

Solving yields:

And the discretization error is given by:

The same procedure can be used to calculate weights for an arbitrary number of points; Weights for up to 5 points are given in table 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Terms (n): | -2 | -1 | 0 | 1 | 2 |
|  |  |  | t |  |  |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |

This suggests a general expression for the weight with index using bins:

With , , and the error given by:

The errors so far have been given in Fourier space. For the line position, the transformation to real space is simple because the derivative is with respect to the dimensionless spectral coordinate itself. For the widths, it is slightly more complicated because the derivative is to . This can be calculated by repeatedly taking the derivative and multiplying by . Transformed into real space, the function is calculated by taking the reverse order, i.e., repeatedly multiplying by and *then* taking the derivative:

A list of derivatives is given in table 2:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 |  |  |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

* In practice, it is useful to keep the index always positive; add index, divide later in the convolution step.
* What about weights minimizing the peak in real space?
* What about minimizing the higher order error terms in real space?
* Make connection with finite difference coefficients

There is an inconsistency in the optimized errors. Before coming up with the “optimized error” scheme, I insisted on minimizing the error in the peak. For 2-term this resulted in a=(1-exp(-tDx))/(1-exp(-Dx)). This is the same as setting the 0th and 1st order error terms to zero, leaving only a second order term. Doing the same in Fourier space resulted the “simple weights”. When playing around earlier, I found that the RMS error is minimized when the average of the Real and Fourier “zero term” solution was used. However, this is not consistent with the solution in the paper. This must be resolved.

It appears the difference has to do with proper accounting of negative signs; do the lineshapes need to be multiplied with- or divided by the widths? This needs to be resolved…

Things to check:

* Is minimizing the RMS error in real space *really* the same as minimizing the error in Fourier space?
* In real space, should I multiply with or divide by widths? How have I done it in:
  + Paper
  + N-term applet
  + Old error estimation applet

It appears the function should be divided by the weight, i.e.:

* According to the paper, E\_RMS^2 = cL1^2.S0L1L1 + cL1cL2.S0L1L2 + cL2^2.S0L2L2. This can be computed from the integrals following the paper, but also directly using Lorentzian lineshapes. Compare this and see if the error is really the same. If there is an inconsistency, identify whether there is an error in the paper.
* A: in pure L/G mode, there is only so much room for optimization. By taking the error separately, the proper optimization is indeed retrieved and it is t + t(1-t)/4Dx. However, having both G and L provides more optimization opportunities it seems. In other words, parts of the G are used to produce a better L lineshape.
* Remaining question is 1. Are the terms in the squared error A.29 the right ones? And is the predicted error in FIG3 consistent with the direct numerical calculation (within the approximation error)?

In real space, this transforms to:

N=2:

*0th order:*

*1st order:*

n=3:

when:

for and:

*1st order:*

*2nd order:*

n=4:

0th order:

1st order:

2nd order:

3rd order:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Terms: | -2 | -1 | 0 | 1 | 2 |
|  |  |  | t |  |  |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Terms are something like the following, but with care taken about aligning it in the proper columns:

2-term (Y\_max = 0.740):

3-term (Y\_max = 1.009):

4-term (Y\_max = 1.876):

How much did we win?

Y2 0.740

Y3 1.009

Y4 1.876