# Data Analysis Tools for Molecular Beam Spectroscopy

**Michael Bongard**
**Department of Physics**
**St. Olaf College, Northfield, MN**
**bongard@stolaf.edu**

## Abstract

Molecular beam spectroscopy is a branch of experimental quantum mechanics that analyzes radio-frequency hyperfine spectra of molecules to determine nuclear properties of atoms. At St. Olaf College, numerous software tools have been developed to analyze increasingly complex spectra. A new tool, Linefit, has been developed for more efficient analysis of spectrometer data sets. A brief discussion of the fitting engine, the Nelder-Mead simplex method,  is provided, as well as the current status of development.

## Introduction

Over the past three years I have conducted interdisciplinary research in physics and computer science at St. Olaf College, working with the molecular beam spectroscopy (Molbeam) research group. I have focused on developing new software tools that will make the scientific research more productive, including safety and data acquisition systems. In this paper, I will be detailing my current project for the team, a data analysis system. A brief discussion of the physics that motivates new tool development is in order to place the project in perspective.

Research in molecular beam spectroscopy has been an ongoing effort at St. Olaf College for over twenty years. Our group's work focuses on the acquisition and analysis of high precision radio-frequency spectra of alkali halide molecules. A sample portion of the RbF spectrum is illustrated in Figure 1. Each peak seen is the result of a transition between hyperfine energy levels of the nuclei of diatomic molecules. These spectra can be accurately predicted by quantum mechanics with knowledge of experimentally-obtained factors. These factors are known as atomic hyperfine coupling constants, and are a measure of the geometry of an atomic nucleus.

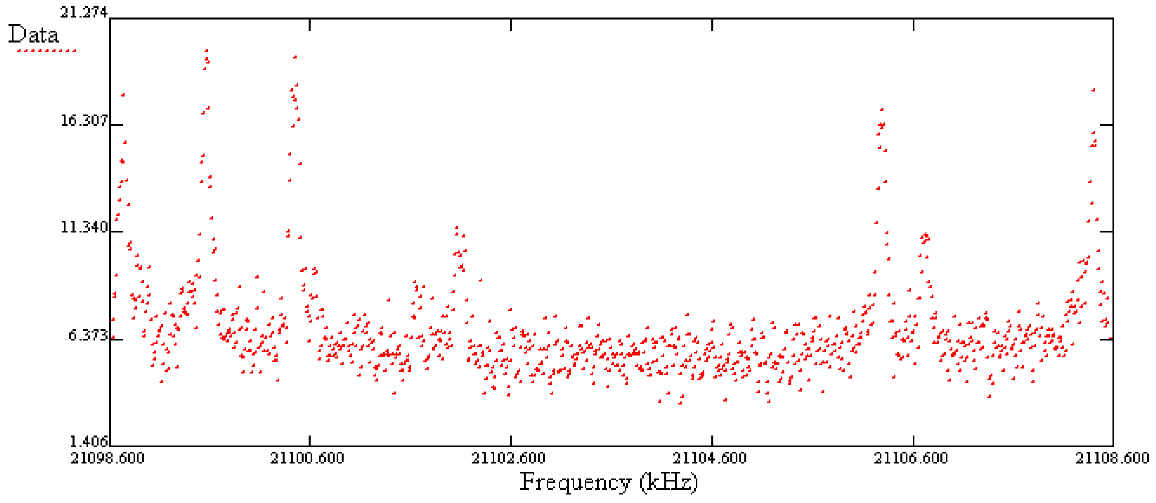

Figure 1: RbF Spectral Data

Through analysis of acquired spectra these hyperfine constants can be calculated. Our research group has written a suite of computer programs over the years to accomplish this task in a two-step process. The first step is handled by a program called Linefit. It takes an individual observed transition and a line assignment as input. A line assignment is a set of quantum numbers that describe the observed transition peaks in the spectrum. Using this information, a transition frequency and its associated experimental uncertainty is calculated. The second step is accomplished with a program called Specfit, and involves combining many individual transition frequencies and their uncertainties to calculate a set of hyperfine coupling constants that satisfy all of the transitions.

Specfit and Linefit have been implemented in many computer languages and for many hardware platforms over the years. Periodically they have been extended (or rewritten) to handle increasingly complex spectra that our group has investigated (Olson et. al., 1994). The current incarnation of these programs are programmed for use in the MathCad computer algebra system for use in a Windows environment.

The most time consuming and computationally intensive aspect of our data analysis is in the Linefit routines. My research has focused on development of a more efficient implementation of Linefit as a standalone Windows application for use by our research group.

## Linefit Requirements

Linefit is not only a tool to fit data. It is also capable of assisting researchers in making the line assignments through quantum mechanical simulations. These theoretical predictions relate to the Stark spectrum and the associated Rabi line shape that correspond to the assigned transitions. While the theory regarding the Stark spectrum and the Rabi line shape (Rabi, 1937) is beyond the scope of this paper, it is sufficient to state that every transition peak's intensity and frequency location is directly proportional to the frequency and amplitude of a Stark component ("stick") in our predictions. Due to the configuration of our experimental apparatus, all transition peaks take the form of the Rabi line shape, a resonance peak that has an explicit mathematical form. In Figure 2, a theoretical prediction of the Rabi line shape is drawn in blue and the Stark spectrum is shown in green on top of the RbF data from Figure 1.

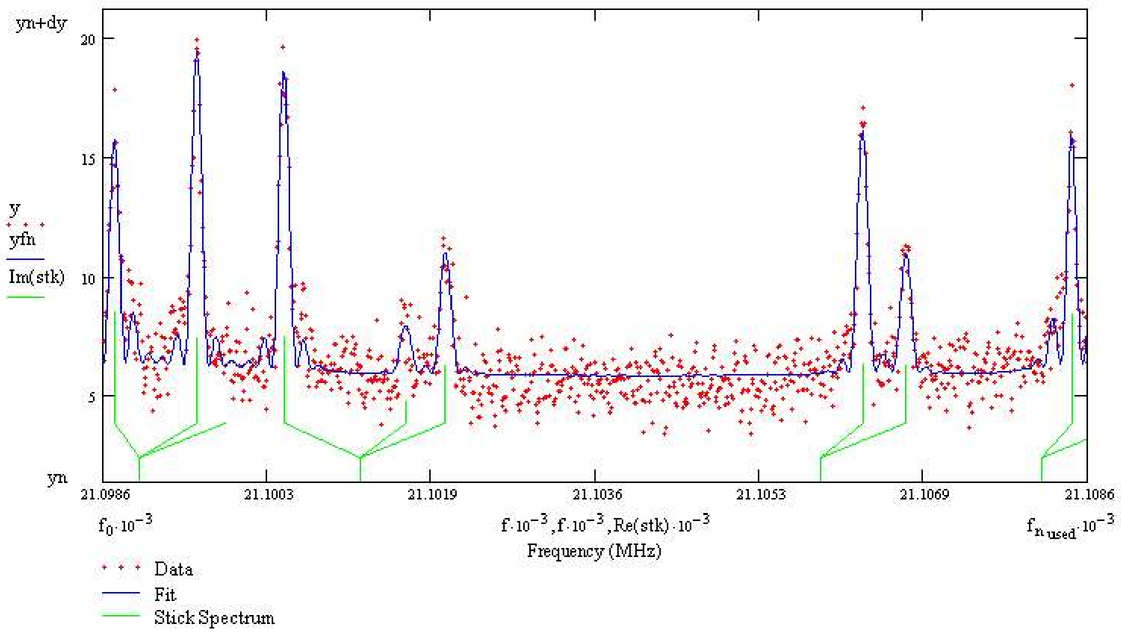

Figure 2: Theoretical Prediction Overlays

The most important aspect of a Linefit implementation is that it is capable of performing a nonlinear multidimensional minimization technique efficiently. The goal of this technique is to determine optimal Rabi function parameter values such that the total difference between the theoretical prediction values and the experimental data is minimized. These parameters consist of the frequencies and amplitude of the Stark sticks, as well as four additional terms relating to the background noise height and overall efficiency of the experimental acquisition itself.

## MathCad Limitations

The choice of minimization technique for our group's MathCad implementation is the Levenberg-Marquardt method (Weisstein, 2004). This method takes as input an objective function to minimize, as well as the first derivatives of this function with respect to each parameter in the objective function.

This derivative requirement is a major drawback for two reasons. First, since every data set is different, the objective function and all its various derivatives need to be re-coded on a case-by-case basis. For simple simulations, this is not a difficult task; however, with newer spectra from $^{6}$LiI, RbF, RbCl, and RbBr it can take hours – if not days – to set up a calculation. The fact that the code is largely uncommented only aggravates the situation. Second, the computation of the fits suffers a performance penalty due to MathCad overhead, both in program interpretation and symbolic derivative evaluation.

There have been serious stability issues with MathCad as well. The Linefit routines and the symbolic derivative information required for the Marquardt minimization can cause the algebra system to crash – even when saving recently entered changes! As this process of manually computing and entering the objective function derivatives is tedious and error-prone, stability is of utmost importance.

Finally, the routines have proven to be difficult to extend to handle complex spectra. Different isotopes of a molecule each produce different transitions at different frequencies. Oftentimes peaks from multiple isotopes all lie within the same set of data. Each isotope needs its own objective function, effectively doubling the user set-up and computational time for the fitting. Support for overlapping spectra is still weak. As such, the typical approach is to re-acquire data to focus only on one isotope's transition. This new data acquisition can be in excess of three days.

## New Design Goals

In light of the drawbacks of the MathCad implementation, I began work on a replacement version of Linefit. It is designed to be run in a Windows environment. Data files utilized in the calculations are compatible between MathCad and Win32 implementations.

In addition to the basic requirement of compatibility with the prior implementation, I also centered the design around speed, stability, and generality. That is, given the same set of data and line assignment, the fitting process should be completed in a shorter amount of time with uncertainty less than or equal to the MathCad implementation. The program must not crash during normal use.

I  intend for the code to be easily extensible. As a result, it is heavily commented, and extensive documentation has been written regarding internal functions and various support libraries.

The issue of generality has been my new addition to the Linefit process. The idea is that a researcher should only need to look at a data set and make a line assignment, foregoing all the manual drudge work of reprogramming the fitting routines on a case-by-case basis. It must also be capable of handling overlapping lines to some extent. This flexibility is a great asset to achieving efficient analysis of thousands of existing data sets!

The generality requirement delegates the generation of the fitting function to the Linefit program. I also made the design choice to avoid taking derivatives at all costs, for performance and numerical stability reasons. This restriction means that the Levenberg-Marquardt minimization technique is not a viable option. Instead, I have chosen to use the downhill simplex method of Nelder and Mead (1965).

## Downhill Simplex Method

The downhill simplex method is a nonlinear multidimensional minimization technique that stands apart from other such minimization techniques. Its distinction lies in that the method only requires knowledge of an objective function of N parameters; other techniques use various derivatives to some extent. This aspect of the downhill simplex method enables analysis of a dynamically generated objective function that is based on the number of Stark sticks present in the line assignment.

The method involves the use of a geometric figure called a *simplex*. In an N-dimensional space, a simplex is a figure with N+1 vertices. Thus, in two dimensions a simplex is a triangle, in three it is a tetrahedron, and so on for higher dimensions. Each of the vertices of the simplex represents a unique set of parameters for the objective function. A simplex also has the property that if the objective function is evaluated at all vertices of the figure, one will have a maximum value. Through manipulation of this maximal vertex, the position of the simplex in its parameter space can be altered to move towards a minimum.

Linefit's implementation of the simplex algorithm is based on the algorithm presented in Numerical Recipes in C++ (Press, Vetterling, Teukolsky, Flannery, 2002) and consists of the following steps. First, evaluate the objective function at all vertices of the simplex to determine the “worst” and “best” points. Next, a reflection test is performed. This means that the worst point is reflected through the surface defined by the remaining points in the simplex and the objective function is re-evaluated at that point. If there is a decrease in objective function value, the test succeeds and the point is extended by a factor of two along the reflection path and the objective is re-evaluated. Whichever point has the lowest objective value is retained as the final position for the vertex and the iteration of the algorithm terminates. In the event of a reflection test failure, a one-dimensional contraction is performed. This contraction moves the worst point towards the surface of the other vertices by a factor of one-half. The objective is re-evaluated at the new point. If there is an improvement, the iteration terminates with the contracted position as the new vertex location. Otherwise, a full contraction is performed. This causes all vertices other than the best point to have their distances to the best point reduced by a factor of one-half. Figure 3 depicts the various steps of the algorithm. Fitting currently terminates after either 5,000 iterations have passed or the fractional change in optimal parameter values is smaller than a user-specified tolerance factor.

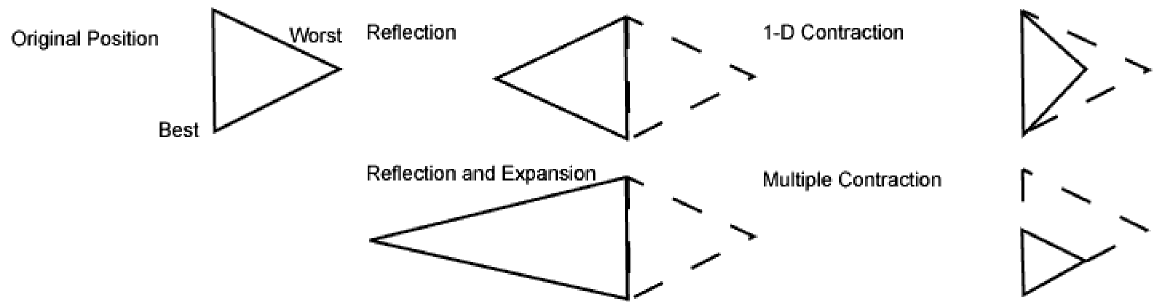

Figure 3: Simplex Manipulations

## Development Challenges and Progress

Throughout the course of development of Linefit there have been various technological hurdles to be jumped. Some have been mundane issues, such as compiler template compliance issues, while others have been interesting from a design perspective.

Progress has been slower than initially anticipated, due to the need to develop various utility libraries. These libraries include a linear algebra package and graphing utilities. While numerous such libraries already exist (and may even be freely available for use), one of the goals of my research program is to produce as much original work as possible. This approach also provides more intimate knowledge of the algebra and graphing subsystems, providing more locations for optimization in future revisions of the code.

Many of the original MathCad Linefit subroutines were applicable to their Win32 counterparts. Oftentimes these algorithms needed to be analyzed and re-interpreted for use in C++, as the MathCad programming language is untyped and utilizes other features of the algebra system, including arrays that are memory-safe and grow automatically. Translation between the strongly typed C++ and untyped MathCad was a difficult challenge!

Generalizing the fitting process was an interesting challenge. The problem dealt with generating a set of parameters to fit for based on the line assignment and letting the simplex fit routine know of the form of these parameters. Existing MathCad routines assumed a particular form for the parameter set, which I chose to replicate. I also devised a scheme that would encode parameter array parsing information in an array that was passed to functions requiring this information.

I also handled overlapping lines with the Win32 Linefit, through the introduction of what I call “line sets.” Each line set can be thought of as an independent source of information in the observed spectrum. Common examples include overlapping lines from different isotopomers of the molecule or lines showing up from an unexpected vibrational sequence. All known troublesome data sets can be modeled with two overlapping line sets. To provide tolerance for the unexpected, the Linefit fitting engine can handle an arbitrary number of line sets; however, the user interface currently allows up to four as input. This can be easily extended in the future if circumstances require it. All line set effects are taken into account when the objective function and its parameters are being generated for the simplex routines.

The first release candidate of Linefit was completed in mid-January 2004. Current work focuses on minor bug fixes and documentation (Bongard, 2004), with a 1.0 release expected by mid-March. Future research includes optimization of the fitting process and incorporation of new features that have been under simultaneous development for the MathCad Linefit program.

Progress with Linefit is likely to continue throughout the duration of molecular beam spectroscopy research at St. Olaf. It is clear that the speed, stability, and generality offered by the new incarnation of Linefit will be serving the team for years to come.

## References

Bongard, M. W. (2004, March 4). Linefit Design Notes: From Vision to Executable. Retrieved March 4, 2004, from http://www.stolaf.edu/people/bongard/CIS/Project/DesignNotes.html.

Nelder, J. A., Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7,* 308-313.

Olson, D., Cederberg, J., Soulen, P., Ton, H., Urberg, K., Mock, B. (1994). Lineshape Analysis of Second-Order Molecular Beam Transitions. *Journal of Molecular Spectroscopy, 166,* 158-168.

Press, W. H., Vetterling, W. T., Teukolsky, S. A., Flannery, B. P. (2002). *Numerical Recipes in C++: The Art of Scientific Computing, Second Edition.* Cambridge, UK: Cambridge University Press.

Rabi, I. I. (1937). Space Quantization in a Gyrating Magnetic Field. *Physical Review, 51,* 652-654.

Weisstein, E. W. (2004, March 4). Levenberg-Marquardt Method. *MathWorld*--A Wolfram Web Resource. Retrieved March 4, 2004, from http://mathworld.wolfram.com/Levenberg-MarquardtMethod.html.

## Acknowledgements

The author would like to thank James Cederberg, David Nitz, Duane Olson, and fellow Molbeam researchers for their support of Linefit development, St. Olaf College and the Hardy Endowed Chair for funding this research, as well as the St. Olaf Center for Integrative Studies for providing the opportunity to continue this work throughout the academic year as a capstone senior project.