1. Theoretical

1.1 Spin 1/2's in a linear chain

The ALCHIM experiment in a fatty acid molecule can be modeled as a linear chain of \( N \) spin \( \frac{1}{2} \) nuclei. The Hamiltonian is

\[
\mathcal{H} = 2\pi J \sum_{i=1}^{N-1} \vec{I}_i \cdot \vec{I}_{i+1}.
\]  

(1.1.1)

In this Hamiltonian only nearest neighbor couplings are included which are assumed to be equal.

1.1.1: The 2 Spin Case

In this case the Hamiltonian reads

\[
\mathcal{H} = 2\pi J \vec{I}_1 \cdot \vec{I}_2 = \pi J \left( \vec{L}^2 - \vec{I}_1^2 - \vec{I}_2^2 \right)
\]  

(1.1.1)
where $\vec{L}^2$ is the square of the total angular momentum operator for the two spin systems. In this total spin basis, where
\[
\vec{L}^2 |L,M\rangle = l(l+1) |L,M\rangle \quad \text{and} \quad \vec{l}^2 |L,M\rangle = \frac{1}{2}(l + 1) |L,M\rangle = \frac{1}{4} |L,M\rangle ,
\]
the Hamiltonian is diagonal. These states written in terms of the single spin $\frac{1}{2}$ states are
\[
|L = 1, M = 1\rangle = \left| \frac{1}{2}, \frac{1}{2} \right> \\
|L = 1, M = -1\rangle = \left| -\frac{1}{2}, -\frac{1}{2} \right> \\
|L = 1, M = 0\rangle = \frac{1}{\sqrt{2}} \left( \left| \frac{1}{2}, -\frac{1}{2} \right> + \left| -\frac{1}{2}, \frac{1}{2} \right> \right) \\
|L = 0, M = 0\rangle = \frac{1}{\sqrt{2}} \left( \left| \frac{1}{2}, -\frac{1}{2} \right> - \left| -\frac{1}{2}, \frac{1}{2} \right> \right)
\]
(1.1.1.3)

The Hamiltonian operating on these states gives
\[
\mathcal{H} |L = 1, M = 1\rangle = \pi J \left( 2 - \frac{3}{4} - \frac{3}{4} \right) |L = 1, M = 1\rangle = \frac{1}{2} \pi J |L = 1, M = 1\rangle \\
\mathcal{H} |L = 1, M = -1\rangle = \frac{1}{2} \pi J |L = 1, M = -1\rangle \\
\mathcal{H} |L = 1, M = 0\rangle = \frac{1}{2} \pi J |L = 1, M = 0\rangle \\
\mathcal{H} |L = 0, M = 0\rangle = \pi J \left( 0 - \frac{1}{4} - \frac{3}{4} \right) |L = 0, M = 0\rangle = \frac{3}{2} \pi J |L = 0, M = 0\rangle ,
\]
(1.1.1.4)
or in matrix form
\[
\mathcal{H} = \frac{1}{2} \pi J \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & -3
\end{pmatrix}.
\]
(1.1.1.5)

The propagator then becomes
\[
\text{Pr} = \text{Exp}\{-i\mathcal{H}\} = \begin{pmatrix}
  e^{-i\frac{\pi}{2}Jt} & 0 & 0 & 0 \\
  0 & e^{-i\frac{\pi}{2}Jt} & 0 & 0 \\
  0 & 0 & e^{-i\frac{\pi}{2}Jt} & 0 \\
  0 & 0 & 0 & e^{i\frac{\pi}{2}Jt}
\end{pmatrix}.
\]

(1.1.6)

The total Z-component of the spin magnetization is

\[
I_{Z}^{\text{Tot}} = I_{Z,1} + I_{Z,2} = \begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & -1 & 0 & 0 \\
  0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0
\end{pmatrix},
\]

(1.1.7)

while the difference in the Z-component of the magnetization is

\[
I_{Z}^{\Delta} = I_{Z,1} - I_{Z,2} = \begin{pmatrix}
  0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1 \\
  0 & 0 & 1 & 0
\end{pmatrix}.
\]

(1.1.8)

The initial density matrix is

\[
\rho(0) = I_{Z,3} = \frac{1}{2}(I_{Z}^{\text{Tot}} + I_{Z}^{\Delta})
\]

(1.1.9)

while the observable is

\[
\mathcal{O} = I_{Z,2} = \frac{1}{2}(I_{Z}^{\text{Tot}} - I_{Z}^{\Delta}).
\]

(1.1.10)

The transfer function is given by

\[
S(t) = \text{Tr}\{\mathcal{O} \text{Pr} \rho(0) \text{Pr}^{\dagger}\} = \frac{1}{2} \text{Tr}\{(I_{Z}^{\text{Tot}} - I_{Z}^{\Delta}) \text{Pr}(I_{Z}^{\text{Tot}} + I_{Z}^{\Delta}) \text{Pr}^{\dagger}\}.
\]

Since \([I_{Z}^{\text{Tot}}, \mathcal{H}] = 0\) and \(\text{Tr}\{I_{Z}^{\text{Tot}} I_{Z}^{\Delta}\} = 0\),

\[
S(t) = \frac{1}{2} \left[ \text{Tr}\{I_{Z}^{\text{Tot}} I_{Z}^{\Delta}\} - \text{Tr}\{I_{Z}^{\Delta} \text{Pr} I_{Z}^{\Delta} \text{Pr}^{\dagger}\} \right].
\]

(1.1.11)

Using that \(\text{Tr}\{I_{Z}^{\text{Tot}} I_{Z}^{\Delta}\} = 2\) and
\[
Pr I_z^A Pr^\dagger = \begin{pmatrix}
    e^{-i\pi Jt} & 0 & 0 & 0 \\
    0 & e^{-i\pi Jt} & 0 & 0 \\
    0 & 0 & e^{-i\pi Jt} & 0 \\
    0 & 0 & 0 & e^{i\pi Jt}
\end{pmatrix}
\begin{pmatrix}
    0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 1 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
    e^{i\pi Jt} & 0 & 0 & 0 \\
    0 & e^{i\pi Jt} & 0 & 0 \\
    0 & 0 & e^{i\pi Jt} & 0 \\
    0 & 0 & 0 & e^{-i\pi Jt}
\end{pmatrix}
\]

\[
Tr \left\{ I_z^A Pr I_z^A Pr^\dagger \right\} = e^{i2\pi Jt} + e^{-i2\pi Jt} = 2 \cos(2\pi Jt).
\]

This should be normalized by \( S_o = Tr \left\{ I_z^A \right\} = 1 \); therefore, the transfer function for 2 spins is

\[
S(t) / S_o = \frac{1}{2} \left[ 1 - \cos(2\pi Jt) \right]
\]

(1.1.1.12)

The first inflection point of the transfer curve occurs when \( Jt = \frac{1}{4} \) and the quasiequilibrium reached occurs when the time dependent terms die away and has a value of \( S_{QE} (t \to \infty) / S_o = \frac{1}{2} \).

1.1.2: The 3 and more Spin Cases

The 3 and more spin cases are calculated similarly; however, the Hamiltonian is no longer diagonal in the total spin basis as the coupling between all spin pairs is not equal. For this reason, we used the product basis and Mathematica to diagonalize the Hamiltonian. The transfer function is now calculated as

\[
S(t) / S_o = Tr \left\{ \rho(0) e^{i\Delta H} \right\} / Tr \left\{ I_z^A \right\}
\]

\[
S(t) / S_o = Tr \left\{ I_{z,N} U^\dagger e^{-i\mathcal{H}_t} U I_{z,1} e^{i\mathcal{H}_t} U^\dagger \right\} / Tr \left\{ I_z^A \right\},
\]

(1.1.2.1)

where \( U \) is the unitary transformation that diagonalizes \( H \). Analytical expressions can be determined for the 3 and 4 spin cases.
3 Spins

\[ S(t)/S_0 = \frac{7}{18} - \frac{1}{3}\cos(\pi Jt) - \frac{1}{6}\cos(2\pi Jt) + \frac{1}{9}\cos(3\pi Jt) , \] (1.1.2.2)

4 spins

\[ S(t)/S_0 = \frac{9}{32} + \frac{5}{48}\cos(2\pi Jt) - \frac{1}{8}\cos(\sqrt{2}\pi Jt) + \frac{1}{32}\cos(2\sqrt{2}\pi Jt) \]

\[ -\frac{5}{96}(2 + \sqrt{2})\cos[(2 - \sqrt{2})\pi Jt] - \frac{5}{96}(2 - \sqrt{2})\cos[(2 + \sqrt{2})\pi Jt] \]

\[ + \frac{1}{24}\cos[(1 - \sqrt{3})\pi Jt] + \frac{1}{24}\cos[(1 + \sqrt{3})\pi Jt] \]

\[ - \frac{1}{96}(4 + \sqrt{2} - \sqrt{6})\cos[(1 - \sqrt{2} - \sqrt{3})\pi Jt] - \frac{1}{96}(4 - \sqrt{2} + \sqrt{6})\cos[(1 + \sqrt{2} - \sqrt{3})\pi Jt] \]

\[ - \frac{1}{96}(4 + \sqrt{2} + \sqrt{6})\cos[(1 - \sqrt{2} + \sqrt{3})\pi Jt] - \frac{1}{96}(4 - \sqrt{2} - \sqrt{6})\cos[(1 + \sqrt{2} + \sqrt{3})\pi Jt] \]

(1.1.2.3)

Numerical solutions were found up to 7 spins. The solutions all had the form of a cosine series

\[ S(t)/S_0 = a_0 + \sum_{i=1}^{N_{\text{max}}} a_i \cos(\pi \nu_i t) , \] (1.1.2.4)

where the \( \nu_i \) are typically irrational multiples of the J-coupling. Simulations were prohibitively long with Mathematica for calculations with more than 7 spins. Plots of the simulations are shown in the online resources Figure 2. Analytical derivatives of the transfer function were taken and the inflection points were determined. A plot of the simulated inflection point as a function of the number of spins in the chain is also shown in online resources Figure 2. The correlation is linear showing that these transfer functions can be used as a molecular ruler. The Quasiequilibrium values are shown in the table 1, which demonstrate that the size of the detected signal is expected to decrease as the chain of spins becomes
longer. This is as expected as the isotropic Hamiltonian distributes the initial magnetization over all the spins.

Table 1: Simulated Quasiequilibrium Intensity

<table>
<thead>
<tr>
<th>N spins</th>
<th>Quasi Equilibrium Intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.500</td>
</tr>
<tr>
<td>3</td>
<td>0.389</td>
</tr>
<tr>
<td>4</td>
<td>0.288</td>
</tr>
<tr>
<td>5</td>
<td>0.242</td>
</tr>
<tr>
<td>6</td>
<td>0.197</td>
</tr>
</tbody>
</table>

1.1.3: Example Mathematica Notebook to calculate the 5 spin case

Comments are presented behind # symbol and should not be included in Mathematica notebook.

```
sx = {{0, 1}, {1, 0}}/2;       #define the spin ½ matrices
sy = {{0, -I}, {I, 0}}/2;
sz = {{1, 0}, {0, -1}}/2;
Id = {{1, 0}, {0, 1}};
Iz1 = KroneckerProduct[Id, Id, Id, sz]; #define the product basis matrices
Iz5 = KroneckerProduct[sz, Id, Id, Id];
Iz1Iz2 = KroneckerProduct[Id, Id, Id, sz];
Iz2Iz3 = KroneckerProduct[Id, Id, sz, sz];
Iz3Iz4 = KroneckerProduct[Id, sz, sz, Id];
Iz4Iz5 = KroneckerProduct[sz, sz, Id, Id];
```
Ix1Ix2 = KroneckerProduct[Id, Id, Id, sx, sx];
Ix2Ix3 = KroneckerProduct[Id, Id, sx, sx, Id];
Ix3Ix4 = KroneckerProduct[Id, sx, sx, Id, Id];
Ix4Ix5 = KroneckerProduct[sx, sx, Id, Id, Id];
Iy1Iy2 = KroneckerProduct[Id, Id, Id, sy, sy];
Iy2Iy3 = KroneckerProduct[Id, Id, sy, sy, Id];
Iy3Iy4 = KroneckerProduct[Id, sy, sy, Id, Id];
Iy4Iy5 = KroneckerProduct[sy, sy, Id, Id, Id];

#define the Hamiltonian

H = (Ix1Ix2 + Iy1Iy2 + Iz1Iz2) + (Ix2Ix3 + Iy2Iy3 + Iz2Iz3) + (Ix3Ix4 + Iy3Iy4 + Iz3Iz4) + (Ix4Ix5 + Iy4Iy5 + Iz4Iz5);
sol = Eigensystem[H];

#Find the Eigenvalues and Eigenvectors
#Find the Normalization constants for the eigenvectors
NormSq = Table[N[sol[[2]][[i]].sol[[2]][[i]]], {i, 1, 2^5}];
Uinv = sol[[2]]/Sqrt[NormSq];

#Find the similarity transform for diagonalizing the Hamiltonian
U = Transpose[Uinv];

#Find the Propagator
DiagPr = DiagonalMatrix[Exp[-I*2*Pi*sol[[1]]*t]];
Pr = U.DiagPr.Uinv;
Prinv = U.DiagonalMatrix[Exp[I*2*Pi*sol[[1]]*t]].Uinv;

#Find the density matrix at time t
rhot = Prinv.IZ1.Pr;

#Calculate the normalized transfer function
Signal5 = ExpandAll[Tr[IZ5.rhot]/Tr[IZ1.IZ1]];
Signal5 = Chop[FullSimplify[ComplexExpand[Signal5]]]  #The 5 spin transfer function
FindRoot[D[Signal5, {t, 2}], {t, .75}]  #Finds the first inflection point
Plot[Signal5, {t, 0, 3}]  #Plot the 5 spin transfer function
2. Data Analysis

2.1 Analytical Fourier Model of Transfer Functions

In order to analyze the experimental transfer functions, it is useful to convert them into analytical functions. Equation 1.1.2.4 demonstrates that the transfer functions are a cosine series. This suggests that the experimental functions can be represented by a cosine Fourier series,

\[ S(n_{\text{cycle}}) = a_0 + \sum_{k=1}^{N_{\text{max}}} a_k \cos \left( \frac{2\pi k n_{\text{cycle}}}{n_{\text{cycle}}^\text{max}} \right) ; \]  

(2.1.1)

however, taking a simple complex fast Fourier transform of the experimental transfer function is not ideal. Figure 7A in the online resources shows the experiment transfer function for octanoic acid while 7B shows the real and imaginary parts of the Fourier coefficients, \( c_k \). The transfer function is then represented as

\[ S(n_{\text{cycle}}) = c_0 + \sum_{k=1}^{N_{\text{max}}} \text{Re} \left[ c_k \right] \cos \left( \frac{2\pi k n_{\text{cycle}}}{n_{\text{cycle}}^\text{max}} \right) + \sum_{k=1}^{N_{\text{max}}} \text{Im} \left[ c_k \right] \sin \left( \frac{2\pi k n_{\text{cycle}}}{n_{\text{cycle}}^\text{max}} \right) ; \]  

(2.1.2)

where the imaginary part is due to the fact that \( S(n_{\text{cycle}}) \) is not even about its center. \( n_{\text{cycle}}^\text{max} \) is the maximum number of DIPSI cycles applied during the isotropic mixing period, while \( N_{\text{max}} \) is the maximum number of Fourier components used in the expansion. From the Fourier transform of the data, there is the same number of Fourier components as observed points in the transfer curve. Using less than this allows Fourier filtering of the high frequency noise from the transfer curve. Figure 7C shows the model transfer function calculated by 2.1.2 using 50 complex Fourier components. The function is periodic with a period of \( n_{\text{cycle}}^\text{max} \), but discontinuous at the endpoints leading to high frequency errors at the discontinuities, 7D. By properly symmetrizing the data these errors can be removed. 7E shows a properly symmetrized transfer function. The original data were reversed and then the point at zero removed before being prepended to the unreversed data. This makes the transfer function even about the midpoint and smooth at the limits of the period. The Fourier coefficients for the symmetrized transfer function are shown in Figure 7F, where it is clear that the coefficients are now purely real. Figure 7G shows three full cycles of the model function again calculated with only 50 complex Fourier components. The model transfer function calculated from the symmetrized data lacks the high frequencies errors. Figure 7H show that this model function (dashed red) accurately represents the measured transfer function (solid blue).
2.2 Analytical Derivatives of Transfer Functions

To find the inflection point of the transfer functions, the maximum of the first derivative must be found. This requires determining both the first and second derivatives of the transfer function, which is straightforward as these functions are just a cosine series,

\[ S(t) = a_0 + \sum_{k=1}^{N_{\text{max}}} a_k \cos(2\pi \nu_k t) . \]  

In the theoretical expressions, \( \nu_k \) is proportional to the J-coupling, while in the experimentally determined transfer functions,

\[ \nu_k = \frac{k}{n_{\text{cycle}}} \]  

and the time, \( t \) is \( n_{\text{cycle}} \). The first derivative of 2.2.1 is

\[ \frac{dS(t)}{dt} = -\sum_{k=1}^{N_{\text{max}}} (2\pi \nu_k) a_k \sin(2\pi \nu_k t) , \]  

while the second derivative is

\[ \frac{d^2S(t)}{dt^2} = -\sum_{k=1}^{N_{\text{max}}} (2\pi \nu_k)^2 a_k \cos(2\pi \nu_k t) . \]  

The first inflection point of the transfer function is the first root of 2.2.4 found by solving

\[ \sum_{k=1}^{N_{\text{max}}} \nu_k^2 a_k \cos(2\pi \nu_k t) = 0 . \]

2.3: Bayesian model for fitting ALCHIM transfer functions of mixtures

In order to extract fractions of the lipid components of a mixture, a proper statistical analysis of the data must be performed. We will follow a Bayesian approach. To produce the measured ALCHIM transfer function, we integrate the peak of choice as a function of the number of mixing cycles, normalized to the initially excited peak. This normalized integral after \( n_{\text{cycle}} \) mixing cycles is \( S_{\text{exp}}(n_{\text{cycle}}) \).

We assume that the noise in this integral is distributed as a Gaussian, so that the probability of measuring the value \( S_{\text{exp}}(n_{\text{cycle}}) \) is

\[ P(n_{\text{cycle}}) = \frac{1}{\sqrt{2\pi \sigma}} e^{-\frac{(S_{\text{model}}(n_{\text{cycle}}) - S_{\text{exp}}(n_{\text{cycle}}))^2}{2\sigma^2}} , \]  

(2.3.1)
where $\sigma$ is the rms noise on the integral and $S_{\text{model}}$ is the model integral given by

$$
S_{\text{model}}(n_{\text{cycle}}) = \sum_{i=1}^{n_{\text{components}}} f_i S_i(n_{\text{cycle}}).
$$

(2.3.2)

$f_i$ is the fraction of the $i^{\text{th}}$ component of the sample and sum of the $f_i$ must be less than or equal to 1. The sum is equal to one if all components of the mixture are being specified. $S_i$ is the integral of the $i^{\text{th}}$ measured pure component of the mixture. The probability for the entire transfer function is the likelihood function, $P(S_{\text{exp}}|\hat{f}, \sigma)$, i.e. the probability of the data given the parameters of the model, $\hat{f}$ the set of $f_i$'s in 2.3.2, and is the product of the probability of each point

$$
P(S_{\text{exp}}|\hat{f}, \sigma) = \prod_{n_{\text{cycle}}=1}^{n_{\text{max}}} P(n_{\text{cycle}}) = \frac{1}{(2\pi)^{n_{\text{max}}/2} \sigma_n^{n_{\text{max}}}} \exp \left[ -\frac{1}{2\sigma^2} \chi^2 \right],
$$

(2.3.3)

where the $\chi^2$ function is given by

$$
\chi^2 = \sum_{n_{\text{cycle}}} \left( S_{\text{model}}(n_{\text{cycle}}) - S_{\text{exp}}(n_{\text{cycle}}) \right)^2.
$$

(2.3.4)

The posterior probability, $P(\hat{f}, \sigma|S_{\text{exp}})$, the probability of the parameters given the data, is proportional to the product of the likelihood function and the prior probability for the parameters. The $f_i$'s are constrained to be between 0 and 1 and to sum to less than or equal to 1. If you assume uniform prior distributions for these parameters, they only contribute to the normalization factor of the posterior probability. The uniform prior in ln($\sigma$) is proportional to $1/\sigma^2$. Thus, we can write our posterior probability as

$$
P(\hat{f}, \sigma|S_{\text{exp}}) = \frac{N}{\sigma_n^{n_{\text{max}}}} e^{-\frac{1}{2\sigma^2} \chi^2},
$$

(2.3.5)

where the normalization factor, $N$, ensures that the integral of the probability distribution over all values of the parameters is 1. This probability distribution allows the determination of the parameters of the model as well as the noise in the data. The noise parameter can be integrated out so as to concentrate on
the relevant parameters, the mixture fractions. 95% confidence intervals are calculated from the 1D projections of the posterior probability, \( P(f_i, \sigma|S_{\exp}) \), as

\[
\int_{f_{i,\text{lower}}}^{f_{i,\text{upper}}} P(f_i, \sigma|S_{\exp}) df_i = 0.95
\]  

(2.3.6)

subject to the constraint that \( f_{i,\text{upper}} - f_{i,\text{lower}} \) is minimized. This formalism was used throughout this paper to fit the lipid fractions from the ALCHIM transfer function. An example Mathematica workbook will be presented in the next section, which shows how the calculation can be performed for a binary mixture.

2.4: Example Mathematica Notebook to fit the ALCHIM transfer function of a binary mixture

Comments are presented behind # symbol and should not be included in Mathematica notebook.

cmp = {Enter comma separated text of values of ncycle to be used in analysis};
S1 = {Enter the integral values for component one acquired at the ncycle above};
S2 = {Enter the integral values from component two acquired at the ncycle above};
Sexp = {Enter the integral values for the mixture to be analyzed acquired at the ncycle above};
Ndata = Length[Sexp];

ListPlot[{Transpose[{ncycle, S1}], Transpose[{ncycle, S2}]}]  #Plot basis transfer functions
DataPlot = ListPlot[Transpose[{ncycle, Sexp}]]  #Plot transfer function of mixture

Smodel = f1*S1 + f2*S2;  #define the two component model
ChiSq = Total[(Sexp - Smodel)^2];  #define the chi squared function
Nsteps = 100;  #define the number of steps in the fractions
Nnoise = 50;  #define the number of step in the noise

f1Range = {f1_lower, f1_upper};  #define the search regions
f2Range = {f2_lower, f2_upper};  #replace the placeholders in the brackets with
NoiseRange = {noise_lower, noise_upper};  #reasonable numbers

#calculate needed tables
f1Vals = Table[i, {i, f1Range[[1]], f1Range[[2]], (f1Range[[2]] - f1Range[[1]])/(Nsteps - 1)}];
f2Vals = Table[i, {i, f2Range[[1]], f2Range[[2]], (f2Range[[2]] - f2Range[[1]])/(Nsteps - 1)}];
NoiseVals = Table[i, {i, NoiseRange[[1]], NoiseRange[[2]], (NoiseRange[[2]] - NoiseRange[[1]])/(Nnoise - 1)}];
ChiSqTable = Array[tmp, {Nsteps, Nsteps}];
PostProbTable = Array[tmp, {Nsteps*Nsteps}];
PostProbf1 = Table[0, {i, 1, Nsteps, 1}];
PostProb2 = Table[0, {i, 1, Nsteps, 1}];
PostProbNoise = Table[0, {i, 1, Nnoise, 1}];

For[i = 1, i <= Nsteps, i++,
    f1val = f1Vals[[i]];
    For[j = 1, j <= Nsteps, j++,
        f2val = f2Vals[[j]];
        ChiSqTable[[i]][[j]] = ChiSq /. f1 -> f1val /. f2 -> f2val;
    ];
];

# Calculate Chi squared over grid of fraction values

# Calculate 1D posterior probabilities of fractions and noise
For[i = 1, i <= Nsteps, i++,
    For[j = 1, j <= Nsteps, j++,
        pptmp = 0;
        For[k = 1, k <= Nnoise, k++,
            Noiseval = NoiseVals[[k]];
            ppval = Exp[-0.5*(ChiSqTable[[i]][[j]]/Noiseval^2)]/(Noiseval^(Ndata + 2));
            pptmp = pptmp + ppval;
            PostProbNoise[[k]] = PostProbNoise[[k]] + ppval;
            PostProbf1[[i]] = PostProbf1[[i]] + ppval;
            PostProbf2[[j]] = PostProbf2[[j]] + ppval;
        ];
        PostProbTable[[i - 1]*Nsteps + j]] = {f1Vals[[i]], f2Vals[[j]], pptmp};
    ];
];

#Normalize the posterior probabilities
Cnorm = 0.0;
Cnorm = Total[ PostProbTable[[All, 3]] ];
PostProbTable[[All, 3]] = PostProbTable[[All, 3]]/Cnorm;

Cnorm = 0.0;
Cnorm = Total[PostProbNoise];
PostProbNoise = PostProbNoise/Cnorm;

Cnorm = 0.0;
Cnorm = Total[PostProbf1];
PostProb1 = PostProb1/Cnorm;

Cnorm = 0.0;
Cnorm = Total[PostProb2];
PostProb2 = PostProb2/Cnorm;

#find the maxima in the posterior probabilities
maxpp = Max[PostProbNoise];
i = 1;
While[PostProbNoise[[i]] != maxpp, i++];
noisemaxi = i;

maxpp = Max[PostProbf1];
i = 1;
While[PostProbf1[[i]] != maxpp, i++];
f1maxi = i;

maxpp = Max[PostProbf2];
i = 1;
While[PostProbf2[[i]] != maxpp, i++];
f2maxi = i;

#output most probable values
f1Vals[[f1maxi]]
f2Vals[[f2maxi]]
NoiseVals[[noisemaxi]]

f1Prob = Transpose[{f1Vals, PostProbf1}];
f2Prob = Transpose[{f2Vals, PostProbf2}];
NoiseProb = Transpose[{NoiseVals, PostProbNoise}];

#Plot 1D posterior probabilities
Plotf1Prob = ListPlot[f1Prob, Joined -> True, PlotRange -> {0, 0.1}]
Plotf2Prob = ListPlot[f2Prob, Joined -> True, PlotRange -> {0, 0.1}]
PlotNoiseProb = ListPlot[NoiseProb, Joined -> True, PlotRange -> {0, 0.1}]

#find errors in fractions
#error in f1
f1maxi

Total[ PostProbf1[[lower ;; upper]] ]

#vary index lower and upper until output is 0.95
f1mp = f1Vals[[f1maxi]]  #index will be between 1 and Nsteps
f1error = (f1Vals[[upper]] - f1Vals[[lower]])/2

f2maxi  #error in f2
Total[ PostProb2[[lower ;; upper]] ]  #vary index lower and upper until output is 0.95
f2mp = f2Vals[[f2maxi]]
f2error = (f2Vals[[upper]] - f2Vals[[lower]])/2  #Plot data with best fit model

SmodelList = Transpose[{ncycle, Smodel /. f1 -> f1Vals[[f1maxi]] /. f2 -> f2Vals[[f2maxi]]}];
ModelPlot = ListPlot[SmodelList, Joined -> True, PlotStyle -> Hue[0.0]];
Show[DataPlot, ModelPlot]

References: