ASSIGNMENT 8: SOLUTION

E. ARNOLD, M-S. LIU, R. PRABHU & C.S. RICHARDS
THE UNIVERSITY OF CAMBRIDGE

Written in XeLaTeX

THE ISF

PREREQUISITES

The chapters on *Fourier transforms* and *scattering functions* from the *Theory Handbook*.

INTRODUCTION

It is a recurring theme to require a model of the *intermediate scattering function* (ISF) to compare against data from helium spin-echo spectroscopy; with this being one of the most important techniques employed by the Cambridge group, it is imperative that the reader (if they study at the university) becomes acquainted with its calculation and interpretation. In a similar spirit to the tutorial on *Monte Carlo* techniques, we will calculate the ISF for the system in the *Molecular Dynamics* tutorial; this will naturally lead to fitting it to a Gaussian distribution, using analytic techniques created in 2004. This will reveal Brownian-like behaviour on increasingly small time scales.

THE SCATTERING AMPLITUDES

When a beam is incident on a surface, it will scatter. Let the amplitude of the scattered beam be A. We can express this scattered amplitude in terms of the parallel momentum transfer $\Delta\mathbf{K}$ of the beam, and the trajectory of a particle $\mathbf{r}(t)$, with the expression

$$A(\Delta\mathbf{K}, t) = \exp(-i\Delta\mathbf{K} \cdot \mathbf{r}(t)).$$

The so-called intermediate scattering function $I(\Delta\mathbf{K}, t)$ is the autocorrelation of the scattering amplitude $A(t)$. This can be written as:

$$I(\Delta\mathbf{K}, t) = \mathcal{F}^{-1}[|\mathcal{F}[A(\Delta\mathbf{K}, t)]|^2],$$

where $\mathcal{F}$ is the temporal Fourier transform, and $\mathcal{F}^{-1}$ its inverse.

ANALYTIC FORM OF ISF

Equations

The most simple system we can model is a particle diffusing on a flat surface, undergoing Brownian-like motion. Unusually, there is an analytic solution for the ISF for this system [1]! This analytical solution can be written as:

$$I(\Delta\mathbf{K}, t) = e^{-\chi^2(\eta t - \phi(t))},$$

where $\phi(t)$ is a function of time, and χ is a function of $\Delta\mathbf{K}$. These two quantities are given by:

$$\phi(t) = 1 - \exp(\eta t);$$
$$\chi(\Delta\mathbf{K}) = \frac{\Delta\mathbf{K}}{\eta}\sqrt{\frac{k_B T_s}{m}},$$

where k_B is the Boltzmann constant, and T_s is the temperature of the substrate surface.

Small Time Scales

What can we do with the analytical expressions above? Given the entire purpose of the ^{3}HeSE experiment is to probe extremely small time scales, it is natural to attempt to calculate the Taylor expansion of the system for small times. This procedure, including simplification, can be written elegantly as:

$$\begin{aligned} I(\Delta\mathbf{K}, t) &= \exp(-\chi^2(\eta t - \phi(t)));\\ &= \exp(-\chi^2[\eta t - (1 - e^{-\eta t})]);\\ &= \exp(-\chi^2[\eta t - 1 + (1 - \eta t + \frac{1}{2}\eta^2 t^2 + O(t^3))]);\\ &\approx \exp(-\frac{1}{2}\eta^2\chi^2 t^2), \end{aligned}$$

which takes the form we would expect - the ISF is a decaying Gaussian envelope. How does this relate to the position evolution of the particles? Given particles moving under Brownian motion are expected to be normally distributed, it is expected that their van Hove pair correlation function should also be normally distributed. Recalling that the ISF is the Fourier transform of the van Hove pair correlation function, and that the Fourier transform of a Gaussian is also Gaussian, it is reassuring that our analytical expression is normally distributed.

TASKS

Throughout this section, use your data from the *Molecular Dynamics* tutorial.

1. Plot the scattering amplitude $A(\Delta\mathbf{K} \cdot \mathbf{r}(t))$ as a function of time.
2. Calculate, and plot, the intermediate scattering function (ISF) as a function of time.
3. Fit the function I to the ISF, where:

$$I = A_0 \exp(-\alpha|t|),$$

   and A_0 and α are constants, determined by the fitting function. You may notice that this does not fit as well compared to the trajectory calculated in the Monte Carlo simulation. What could be the reason? Try to follow the same train of thought as the discussion on Brownian motion in the theory section of this tutorial.
4. Plot the ballistic part of the ISF. Attempt to fit it to a Gaussian function of the form:

$$I = A_0 \exp(-\frac{x^2}{2\sigma^2}),$$

   where A_0 and σ are constants, determined by the fitting function. Plot the constant σ against ΔK.
5. Attempt the same procedure for the intermediate scattering function after a long time period.

EXTENSION

1. We have provided both an *a priori* and *a posteriori* description of how the scattering particle undergoes ballistic motion. Try to simulate the trajectory again for different surface temperature T_s. How does the change in temperature (corresponding to the system energy - see the equipartition theorem) change the behaviour of the diffusing particles?
2. To eliminate random error, and to improve the quality of the data, repeat the experiment. How can we quantify the behaviour for a large number of particles?
3. Simulations usually involve an absorbate/substrate potential. How would you includea substrate potential ΔV in your simulation? In reality, the assumption used here of the surfaces being flat will not be valid. Thus understanding how to include the potential energy surface is vital.

SUMMARY

In this tutorial, we have seen how it is possible to describe the behaviour of diffusing particles with fitting functions.

REFERENCES

1. Vega et al. J. Phys.: Condens. Matter 16 (2004)

SOLUTION

THE ISF: MAIN TASK

Task 1: Question

Plot the scattering amplitude $A(\Delta\mathbf{K} \cdot \mathbf{r}(t))$ as a function of time.

Task 1: Solution

Using the formula provided in the tutorial, we can calculate the scattering amplitude function. First, we must set up a domain of $\Delta\mathbf{K}$.

```
% find the range of the parallel momentum transfer
% linspace creates evenly spaced points between 0 and 5
KVec = linspace(0, 5, 30);
```

There is an analytic formula for the scattering amplitude, which is given in this document. The scattering amplitude function can be calculated from this with:

```
% calculate the scattering amplitude
AKt = exp(-1i*(KVec'*r(1,:)+KVec'*r(2,:)));
```

The code above calculates the scattering amplitude in the momentum domain. This function, in the frequency domain, can be obtained using the fast Fourier transform `fft`.

```
% The first entry of fft is the function to be Fourier
    transformed
% The second entry specifies the number of points used to take
    discrete Fourier transform (DFT). We let MatLab run
    automatically and did not specify any value.
% The third entry specifies the direction along which to take
    the Fourier transform. In this case, the argument 2 tells
    MatLab to take each row as input.
AKw=fft(AKt, [], 2);
```

The scattering amplitude function is complex-valued; we will plot the real part, imaginary part, and modulus of the function against time, for a given value of the parallel momentum transfer ($\Delta\mathbf{K}$). We first specify how many runs (along with the increments in their domain) we wish the program to calculate with:

```
% We wish to investigate A of the first numK values of Delta K
    ...
```

```
numK=10;

% ...of the numTime time steps
numTime=100;
```

Our first output is the plot of real part of scattering amplitude function versus time.

```
% initialise figure
fig=figure;

% Suppress figure refreshing
hold on

% Set labels
xlabel('Time/ps')
ylabel('$\Re(A(\Delta K, t))$', 'Interpreter', 'latex')

% iterate through a certain number (numK's) of parallel momentum
    transfers Delta K
for i = 1:numK
    % calculate the real part of the scattering amplitude and
        plot it
    % Use linspace for evenly spaced points across the numTime
        time range
    plot(linspace(0, dt*numTime, numTime), real(AKt(i, 1:numTime
        )))
end
```

We draw our attention to our next output. The plot of the imaginary part follows almost the exact same procedure.

```
% initialise a figure
figure;

% Suppress figure refreshing
hold on

% Set labels
xlabel('Time/ps')
ylabel('$\Im(A(\Delta K, t)$', 'Interpreter', 'latex')


% iterate through a certain number (numK) of parallel momentum
    transfers Delta K
for i = 1:numK
    % calculate the imaginary part of the scattering amplitude
        and plot it
    plot(linspace(0, dt*numTime, numTime), imag(AKt(i, 1:numTime
        )))
end
```

Let us move on to the trajectory traced out by the real and complex parts of the scattering amplitude $A(\Delta\mathbf{K}, t)$ in the complex plane. We can plot this with:

```
% initialise a figure
figure;

% Suppress figure refreshing
hold on
```

```
52  hold on
53
54  %Set labels
55  xlabel('$\Re(A(\Delta K,t))$', 'Interpreter','latex')
56  ylabel('$\Im(A(\Delta K,t))$', 'Interpreter', 'latex')
57
58  % iterate through a certain number (numK) of parallel momentum
        transfers Delta K
59  for i = 1:numK
60      % plot the real part against the imaginary part of the
            scattering amplitude
61      plot(real(AKt(i,1:numTime)),imag(AKt(i,1:numTime)))
62  end
63
64  % set the axis sizes to be in the same ratio
65  axis equal
```

The result of the above three plots can be found in figure 1. The trajectory traces out a an almost perfect circle.

Task 2: Question

Calculate, and plot, the intermediate scattering function (ISF) as a function of time.

Task 2: Solution

For a given value of $\Delta\mathbf{K}$, we can calculate the ISF using the scattering amplitude function as derived in the tutorial.

```
1  IKt=ifft(AKw.*conj(AKw), [], 2);
```

In this task, we will plot the ISF as a function of time. As we have many different ISFs, dependent on the parallel momentum transfer, we will construct a loop that iterates through the values of the parallel momentum transfer $\Delta\mathbf{K}$ (which we denote `KVec` in the code), plotting the graph for each transfer on the same figure. Plotting the data from `-dt*nstep/2` to `dt*nstep/2` is achieved with:

```
2      % iterate through values of Delta K
3      % ind is an arbitrary index
4      for ind = 2:length(KVec)
5          % New figure environment
6          figure;
7
8          % The 'prepareCurveData' function comprises of many
               steps
9          %   takes the real part of each entry
10         [xData, yData] = prepareCurveData(linspace(-dt*nstep/2,
               dt*nstep/2, nstep), fftshift(real(IKt(ind, :))./
               nstep));
11
12         %Plot the data and specify the shape of the marker and
               the colour accordingly
13         plot(xData, yData, 'x', 'Color', '#48399e');
14
15         hold on;
```

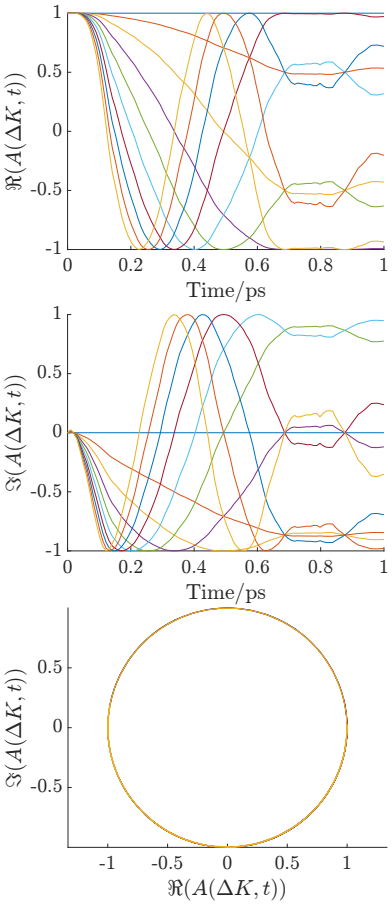

Figure 1 | Plot of the real, imaginary parts of the scattering amplitude versus time, and the trajectory it traces out in the complex plane.

The plot should look similar to figure 2

We would like to see how well the data fit to the analytic solution. Setting up the analytical function can easily be achieved:

```
% Equations based on the analytic solution provided in the tutorial
% The '@' operator is a short hand that creates local functions
phi      = @(t) 1-exp(-eta*abs(t));
chi      = @(DK) DK*sqrt(Kb*Ts/mass)/eta;
isf      = @(t,DK) exp(-chi(DK)^2*(eta*abs(t)-phi(t)));


% isf_fast is the ballistic part of the motion that uses approximations derived in the tutorial
isf_fast = @(t,DK) exp(-chi(DK)^2*eta^2*t.^2/2);
```

We then plot the analytic ISF on the given domain.

```
        % The plot has been shifted so it centres at the origin
        % Hence we use linspace from -100 to 100
        % The function 'isf' defined above takes two inputs
        %  time range  -100 to 100 is centralised
        %  KVec  is the momentum transfer
        plot(linspace(-100,100,500), isf(  linspace(-100,100,500), KVec(ind)  ), '-');

        % Set axes labels
        xlabel('Time (centralised)')
        ylabel('ISF')

        % Limit the x-axis range to the domain
        xlim([-100 100]);
    end
```

The resulting graph should appear like figure 3.

The same plot with different values of $\Delta\mathbf{K}$ can easily plotted.

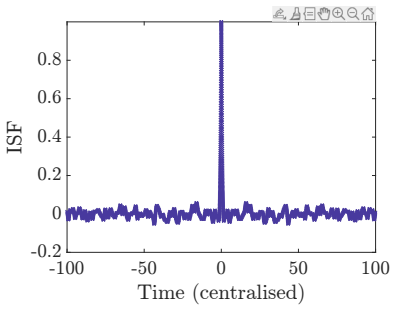

Figure 2 | Plot of the ISF from the Molecular Dynamics tutorial for a particular momentum transfer.

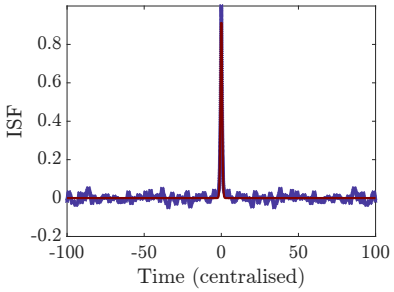

Figure 3 | Analytic solution (red) for the ISF compared to the simulated result (purple).

Task 3: Question

Fit the function I to the ISF, where:

$$I = A\exp(-\alpha|t|),$$

and α is a constant, determined by the fitting function. You may notice that this does not fit as well compared to the trajectory calculated in the Monte Carlo simulation. What could be the reason?

Try to follow the same train of thought as the discussion on Brownian motion in the theory section of this tutorial.

Task 3: Solution

In this section we are going to fit the ISF to the function above over a short time range. We achieve this using the `excludeData` function. We again construct a loop iterating through values of the parallel momentum transfer `KVec`:

```
for ind = 2:length(KVec)
    % excludePoints is now all points outside the range of
        [-0.01, 0.01]
    excludedPoints =~excludedata(xData, yData, 'Domain', [-.01
        .01]);
```

The exponential is in the form of $ae^{-b|x|}$. Fitting this data to the exponential function can be achieved with:

```
    % Set up fittype and options.
    ft = fittype( 'a*exp(-b*abs(x))', 'independent', 'x', '
        dependent', 'y' );
```

The regression method we employ is the least square method. We evaluate the quality of the fitting by the sum of squares of the distance between a data point its corresponding fitted value. The following line specifies the method.

```
    % Specify the fitting method, i.e. least square method
    opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
```

The next few lines tell the program to not display any intermediate step, to set the lower bounds of the coefficients being used in the fitting, and to initialise the values of the coefficients.

```
    % Do not show the intermediate steps
    opts.Display = 'Off';

    % The coefficients has to be non negative
    opts.Lower = [0 0];

    % Initialise the first pair of coefficients
    opts.StartPoint = [0.5 20];

    % Tell MatLab to ignore the points outside range of
        consideration
    opts.Exclude = excludedPoints;
```

Finally, fit the function with data. Extract the coefficients, in order to plot them as a function of the parallel momentum transfer $\Delta\mathbf{K}$.

```
    % Fit model to data.
    [fitresult, gof] = fit( xData, yData, ft, opts );

    %Store the coefficients, b, of varying Delta K, into an
        array called DKs
    DKs(ind)=fitresult.b;
end
```

We can see how the coefficient α (assigned the variable `b` in the program) varies with $\Delta\mathbf{K}$ with the figure in the margin. This figure is plotted with the code:

```
figure
%Plot the coefficients against Delta K
%Set the marker shape and colour accordingly
plot(KVec(2:end), DKs(2:end), 'x')
```

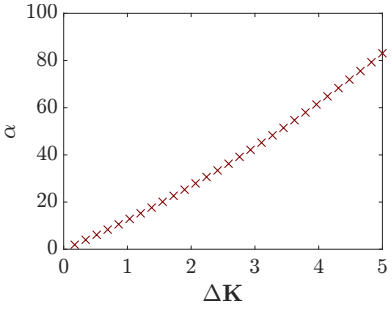

Figure 4 | Plot of the value of the exponential coefficient (α) dervied from the exponential curve fitted to the simulated data in short time range against the momentum transfer ($\Delta\mathbf{K}$).

Task 4: Question

Plot the ballistic part of the ISF. Attempt to fit it to a Gaussian function of the form:

$$I = A\exp(-\frac{x^2}{2\sigma^2})$$

where A and σ are constants, determined by the fitting function. Plot the constant σ against $\Delta\mathbf{K}$.

Task 4: Solution

This task requires us to perform another non-linear fitting. This can be easily achieved.

```
% The first section has exactly the same setup as task 3
for ind = 2:length(KVec)

    % Use the 'prepareCurveData' function to prepare the data
        for fitting
    [xData, yData] = prepareCurveData(linspace(-dt*nstep/2, dt*
        nstep/2, nstep), fftshift(real(IKt(ind, :))./nstep));

    % Select short time range data by excluding the rest
    excludedPoints=~excludedata(xData, yData, 'Domain', [-.01
        .01]);

    % Specify the fitting type as a Gaussian function
    % Different modes of Gaussian curve is available to MatLab
    % The '1' in 'gauss1' tells MatLab that this is an average
        Gaussian distribution with only one term
    % If instead it was 'gauss2', MatLab will fit the data to
        the sum of two different Gaussian distributions
    f = fit(xData, yData, 'gauss1');

    % Store the standard deviations sigma in an array, derived
        from c1
    sigmas(ind) = f.c1/sqrt(2);
end
```

After the standard deviations σ have been recorded, we plot their values against `KVec` as follows.

```
% initialise a figure
figure;

% Plotting standard deviation, sigma, against momentum transfer
plot(KVec(2:end), sigmas(2:end), 'x')
```

The result can be seen in figure 5.

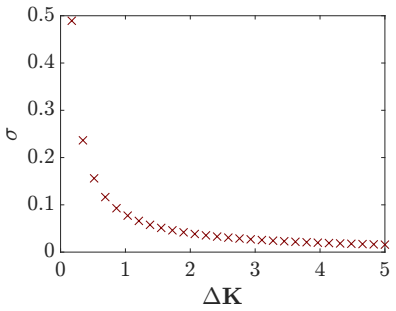

Figure 5 | Plot of the standard deviation σ derived from the Gaussian curve fitted to the short-time-range simulated data against the momentum transfer $\Delta\mathbf{K}$.

Task 5: Question

Attempt the same procedure for the intermediate scattering function after a long time period.

Task 5: Solution

The rest of the process is very much similar to what has been described above, and will be left as an exercise to the reader.