Appendix C

Matlab deconvolution scripts

Introduction

The optical absorption kinetics discussed in Chapter 2 occur on the same timescale as the instrument response. Thus, it was necessary to deconvolute the instrument response from the observed kinetics. This appendix presents the Matlab scripts used for this purpose. While they were developed specifically for the kinetics observed in Chapter 2, these scripts can be easily modified to accomodate a wide variety of kinetics equations.

The process used is one of deconvolution by iterative reconvolution. A model of the kinetics is convolved with the response function, and the resulting convolved model is compared to the data. The model is then modified, and the process repeated until the fit between the data and the convolved model cannot be improved. This method has the advantage of working with an arbitrary response function. It is, however, slow compared to methods that assume a Gaussian response function.

Time and space limitations make a full discussion of deconvolution algorithms impractical. For an excellent discussion of deconvolution as related to time-resolved spectroscopy, see *Excited State Lifetime Measurements* by Demas.[1]

This is the main module, optical_decon4.m

```
% fits OD data by first converting it into intensity and then deconvoluting the response
% function by iterative reconvolution.  Data are loaded as .txt single columns.  The file
% names are then converted over to the internal names resp (response) and int (intensity)

global time
global t_r
global interp
global interp_resp
global t_0_data
global t_0_resp
global kr
load bt_420_long.txt;                %CHANGE name of the raw OD data file
load bt_resp.txt;                    %CHANGE name of the response function
                                     %(intensity)
resp = bt_resp;                      % CHANGE
int = 10.^(-(bt_420_long));          %CHANGE converts OD input into intensity
t = 5*(1:(length(int)));             %define a time axis 'time' with 5 ns spacing
time_resp = 5*(1:(length(resp)));    %define a time axis for response function
                                     %'time_resp'
time = 1:(5*length(int));            %define time a axis t with 1 ns spacing
t_r = 1:(5*length(resp));            %define a time axis t_r with 1 ns spacing
interp = spline(t, int, time);       %interpolate data to 1 ns spacing using cubic spline
interp_resp = spline(time_resp, resp, t_r);     %interpolate resp function
m = trapz(interp_resp);
interp_resp = interp_resp/m;         %normalizes the interpolated response function
%_____________________________________________________________________
%input guesses at parameters
% note: this version is for when the decay rate of the excited Ru* is known
c = -0.014;              %CHANGE input guess for preexponential
```

```
kr = 0.037;              %CHANGE input measured Ru* decay exponential
kb = 0.19;               %CHANGE input guess for 2nd exponential
ksep = 0.0154;           %CHANGE input guess for rate of charge separation
t_0_data = 41 ;          % CHANGE input time 0 of data function 41
t_0_resp = 7.4 ;         % CHANGE input time 0 of response function 6
%_____________________________________________________________________
parameters = [c, kb, ksep];
[x,fval]= fminsearch('rutmim2',parameters)  % rutmim2 is a function that contains the
                                            % kinetics equation specific to this system
                                            % fminsearch minimizes rutmim2 with
                                            % respect to the parameters in 'parameters'
out = rutmim_out2(x);                       % out is the non-convolved kinetics model
con_out = convolver2(interp_resp, out, 5*t_0_resp);     %con_out is the convolved
                                                        %kinetics model
resid = interp'-con_out;                    % calculates residual
```

This function, rutmim.m, contains the specific model to be fit.

```
function f = rutmim(p)
% this function returns the kinetics of an A->B->C process,
% given the parameters c, ka, kb, and the time vector t
% note in this version it is assumed that the decay rate of Ru* (kr) is known
global time
global t_0_data
global interp
global t_0_resp
global interp_resp
global kr
w = length(time);
temp = zeros(w,1);
for i = 0 : (w-(5*t_0_data))
  temp(i+(5*t_0_data)) = (p(1)/(p(2)+p(3)-kr))*((1-(p(3)/kr))*exp(-kr*i) - (1 -
(p(3)/(p(2)+p(3))))*exp(-(p(2)+p(3))*i) + (p(3)/kr) - (p(3)/(p(2)+p(3))));
  end
% change the equation in the line above to fit a different kinetics model
% the parameters that are passed to this function will also have to modified accordingly
ideal = 10.^(-temp);    %converts OD into intensity so that it can be convolved with the
                        %resp function
convolved = convolver2(interp_resp, ideal, 5*t_0_resp); % convolves interp_resp and
                                                        % ideal
f = sum(abs(convolved'-interp)); % function returns the sum of the absolute differences
                                 % between the model (convolved) and the data (interp)
                                 % this sum is what is minimized by fmimsearch
```

This function, convolver2.m, convolves one vector with another, in this case, the response function with the model.

```
function f = convolver(a,b,za)          % a is short vector, b is long one
n = length(b);                          % za is the zero point of a
flip = rot90(a);                        % flip = the resp. funct. run in reverse; row vector
m = length(a);
shift = -(m-za):1:(za-1);               % shifts t_0 to t_0 of resp. funct.
f = b;          % initially set conv product=obj (takes care of end effects)
for i = (m-(za-1)):(n-za)               % with t=0 at t_0_resp
      chunk = b(i + shift);             % define chunk of obj to be conv. w/resp
      int = flip.*chunk;                % define the integrand
      f(i) = trapz(int);                % point-wise convolution
end

% This function convolves two vectors.  Beware of the conv. function
% built into Matlab, which is another thing entirely.  Both a and b should
% be column vectors
```

This function, rutmim_out.m, produces a column vector containing the kinetics model.

```
function f = rutmim(p)
global time
global t_0_data
global interp
global t_0_resp
global interp_resp
global kr
w = length(time);
temp = zeros(w,1);
for i = 0 : (w-(5*t_0_data))
  temp(i+(5*t_0_data)) = (p(1)/(p(2)+p(3)-kr))*((1-(p(3)/kr))*exp(-kr*i) - (1 -
(p(3)/(p(2)+p(3))))*exp(-(p(2)+p(3))*i) + (p(3)/kr) - (p(3)/(p(2)+p(3))));
  end
f = 10.^(-temp);

% this function returns the kinetics of an A->B->C process,
% given the parameters c, ka, kb, and the time vector t
% note in this version it is assumed that the decay rate of Ru* (kr) is known
```

REFERENCES

1. Demas, J. N. *Excited State Lifetime Measurements*; Academic Press: New York, 1983.