A P P E N D I X

MATLAB Programs and Functions Utilized in Data Analysis

The programs and functions given on the following pages were utilized in the analysis of time-resolved UV-VIS spectroscopy data from the 10 ps laser.

The function **streak_a.m** (p. 113) was used to convert streak camera image files into text files with data. It was called with the command:

```
[t,y]=streak_a;
```

Once converted, the resulting vectors were saved to a text file.

The file **max_ent.m** (p. 116) was utilized for a cursory fit to the data (when there was no concern over the instrument response). The program actually does a lot more than fitting, but for the purpose of the fits, the function was exited when asked:

```
Do you want to fit with a single MEM...
```

by hitting Ctrl-C.

The program **splice_data1.m** (p. 124) was used to splice data from different timescales together so that analysis could be performed on complete data sets with better data at faster timescales.

The function **crystal.m** (p.126) was utilized to convolve instrument response with a theoretical curve; the convolved curve would be compared to the data, and chi square

would be calculated. It calls another function **multiexp_conv.m** (p.127) that executes the convolution on a spliced data set.

streak_a.m

```
function [X,Y]=streak_a()
%
%
%           function: streak_a
%           action:  bins selected streak camera image data collected
with
%                   streak camera GPIB controller turned off.
%                           into a time vector (t) and
%                           an intensity vextor (y)
%
%           syntax:
%
%             [t,y]=streak;
%
%             remember to end with a semicolon
%             to avoid printing out the results
%             on the screen
%
%
load streak_scl
%
%deltat=[0.31329345703125; 0.9678955078125; 1.904541015625;
4.92431640625; 12.11328125; 21.474609375; 42.72265625; 102.48828125];
%
time=[A B C D E F G H];
%
filename=input('Enter the data file name: ','s');
%
fprintf(' Data in file: %24s\n',filename)
%
fid=fopen(filename,'r');
%
%
if fid < 0
   fprintf(' Error opening : %24s\n',filename)
   return
end
%
%
im=fread(fid,2,'uchar');
IM=char(im');
length=fread(fid,1,'int16');
width=fread(fid,1,'int16');
height=fread(fid,1,'int16');
xoff=fread(fid,1,'int16');
yoff=fread(fid,1,'int16');
filetype=fread(fid,1,'int16');
reserved=fread(fid,51,'int16');
icomment=fread(fid,length-52,'uchar');
comment=char(icomment');
fprintf(1,'\n%34s\n',comment(171:204))
fprintf(1,'%55s\n',comment(207:261))
fprintf(1,'%43s\n',comment(264:284))
for i=1:height-1
   tmp=fread(fid,width,'int16');
```

```
   image(i,1:width)=tmp';
end
%
%
fclose(fid);
%
%
fprintf(1,'\n\nTime Bases:\n')
fprintf(1,'  1 --> 0.2 ns\n')
fprintf(1,'  2 --> 0.5 ns\n')
fprintf(1,'  3 --> 1.0 ns\n')
fprintf(1,'  4 --> 2.0 ns\n')
fprintf(1,'  5 --> 5.0 ns\n')
fprintf(1,'  6 --> 10 ns\n')
fprintf(1,'  7 --> 20 ns\n')
fprintf(1,'  8 --> 50 ns\n')
itst=-1;
while itst<0
   timebase=input('\nEnter the number of the timebase --> ');
   if timebase >= 1 & timebase <= 8
      itst=1;
   end
end
%
%
y=sum(image);
plot(y);
axis([1 640 min(y)-((max(y)-min(y)).*0.025) max(y)+((max(y)-
min(y)).*0.025)])
xlabel('Channel Number')
ylabel('Intensity')
%
%
itst=-1;
while itst<0
   fprintf(1,'\n\nEnter columns numbers to bin in the following
format:\n')
   fprintf(1,'                        [left1 right1; left2
right2; ... ]\n\n')
   comb=input('Enter the columns to bin --->  ');
%
   [a,b]=size(comb);
   if b==2 & a>=1
      itst=1;
   end
end
%
%
j=1;
for i=1:a
   if comb(i,1) >= 1 & comb(i,1) <= 640 & comb(i,2) >= 1 & comb(i,2) <=
640 & comb(i,1) <= comb(i,2)
      COMB(j,1:2)=comb(i,1:2);
      j=j+1;
   end
end
%
```

```
%
[A,B]=size(COMB);
%
%
for i=1:A
   y=sum(image(:,COMB(i,1):COMB(i,2))');
end
%
%
plot(y)
axis([1 512 min(y)-((max(y)-min(y)).*0.025) max(y)+((max(y)-
min(y)).*0.025)])
xlabel('Channel Number')
ylabel('Intensity')
%
%
itst=-1;
while itst<0
   fprintf(1,'\n\nEnter first and last data points in the following
format:\n')
   fprintf(1,'                        [first last]\n\n')
   firlas=input('Enter the data points --->  ');
%
   [a,b]=size(firlas);
   if b==2 & a>=1
      itst=1;
   end
end
%
if firlas(1,1) <= 0 | firlas(1,1) >= firlas(1,2)
   firlas(1,1)=1;
end
%
if firlas(1,2) > 512 | firlas(1,2) <= firlas(1,1)
   firlas(1,2)=512;
end
%
Y=y(firlas(1,1):firlas(1,2))';
%X = 0:deltat(timebase):(firlas(1,2)-firlas(1,1)).*deltat(timebase);
X=time(firlas(1,1):firlas(1,2),timebase);
X=X./1000;
%
%
plot(X,Y)
axis([min(X) max(X) min(Y)-((max(Y)-min(Y)).*0.025) max(Y)+((max(Y)-
min(Y)).*0.025)])
xlabel('Time, ns')
ylabel('Intensity')
%
%
clear time A B C D E F G H
%
%
```

max_ent.m

```
%
%
%
%   script to fit two-column ascii kinetics data
%  to a distribution of rate constants using the maximum entropy method
%
%
clear all
clf
%
%
%
%        define fit function
func=['laplace_mem'];
%
fprintf('\r\r Data must be in two-column or two-row, space delimited,
ascii format:\r column(row)-1 is time; column(row)-2 is intensity\r\r')
filename=input('Enter the data file name --> ', 's');
%
%
Mm=dlmread(filename);
M=Mm(:,1:2);
[rM cM]=size(M);
if cM ~= 2
   M=M';
end
%
%
if cM ~= 2
   [aa, bb]=size(M');
   fprintf('\r\r Data are not in two-column or two-row, space delimited,
ascii format \r')
   fprintf('\r %5i Rows and %5i Columns \r', aa, bb)
   return
end
%
%
t=M(:,1);
t=t-t(1);
y_un=M(:,2);
y=y_un;
%y=y_un./max(y_un); %set max value to 1
%[t,y,wlog]=logtime(te,ye,50);
%[rt ct]=size(t);
%if ct ~= 1
%   t=t';
%end
%[ry cy]=size(y);
%if cy ~= 1
%   y=y';
%end
%[rw cw]=size(wlog);
%if cw ~= 1
%   wlog=wlog';
%end
```

```
%
%
tleng=length(t);    %length of the logarithmically space data record
%
%
fprintf('\r\r Weighting methods: (1) uniform (DEFAULT) (2) relative;
(3) Poisson\r')
weight_method=input('Select a weighting method (1,2,3) --> ');
%
%
if isempty(weight_method)
   weight_method=1;
end
%
%
sml1=sort(y);
test_y=-1;
icnt=1;
while test_y <= 0
   test_y=sml1(icnt);
   icnt=icnt+1;
end
%
%
if weight_method==1
   wt=ones(tleng,1);
elseif weight_method==2
   wt=y_un.^2;
   tty=test_y.^2;
   wt=(max(wt,tty));
   wt=sqrt(wt);
%   wt=sqrt(wt./wlog);
elseif weight_method==3
   tty=sqrt(test_y./max(y_un));
   wt=sqrt(y./max(y_un));
   wt=max(wt,tty);
%   wt=sqrt(wt./wlog);
end
%
%
clf
subplot(3,1,2)
plot(t,y)
pause(0.1)
%
%
ltest=-1;
ltest1=-1;
while ltest < 0
   while ltest1 < 0
      fprintf('\r\r 1/tmin = %12.8e ; 1./tmax = %12.8e \r\r', 1./(t(2)-
t(1)), 1./max(t))
      kmax=input(' Enter the value for the maximum rate constant in the
distribution --> ');
      kmin=input(' Enter the value for the minimum rate constant in the
distribution --> ');
      if kmax > kmin
```

```
         ltest1=1;
      else
         fprintf(' kmax must be greater than kmin !')
      end
   end
   delta=input(' Enter the value for the resolution ratio of adjacent
rate constants [k(i+1)/k(i)] --> ');
   lkspace=log10(kmin):log10(delta):log10(kmax);
   lkleng=length(lkspace);
   if (lkleng > 1) & (tleng > lkleng)
      fprintf('\r\r k-space vector has %4i elements ',lkleng)
      fprintf('\r time vector has %4i elements ',tleng)
      ltest=1;
   elseif (tleng <= lkleng)
      fprintf('\r\r # of fit parameters exceeds number of observables
')
      fprintf('\r k-space vector has %4i elements; time vector has %4i
elements ',lkleng,tleng)
   else
      fprintf('\r\r k-space vector is too short: %4i element ',lkleng)
   end
end
%
%
kspace=ones(1,lkleng);
kspace=kspace.*(10.^lkspace);
mink=min(min(kspace),1./max(t));
kspace=[mink./1e4,kspace];
kleng=length(kspace);       % length of k-space vector
A=t*kspace;
A=exp(-A);
%
%  get the lsqnonneg solution for the initial guesses
%
fprintf('\r\r Finding the lsqnonneg solution for the initial guess \r')
X0=lsqnonneg(A,y);
%
%
ycalc=A*X0;
lsqchisq=((ycalc-y)./wt)'*((ycalc-y)./wt);
%
%
subplot(3,1,1)
plot(t,y-ycalc,'b',[min(t),max(t)],[0,0],'k--')
axis([min(t) max(t) 1.05.*min(y-ycalc) 1.05.*max(y-ycalc)])
ylabel('I-I_{calc}')
subplot(3,1,2)
plot(t,y,'b',t,y,'bo',t,ycalc,'r',[min(t),max(t)],[0,0],'k--')
axis([min(t) max(t) 1.05.*min(min(min(y),min(ycalc)),0)
1.05.*max(max(y),max(ycalc))])
xlabel('time')
ylabel('Intensity')
lsqchisq=(ycalc-y)'*(ycalc-y);
tt=['  \chi^2 = ' num2str(lsqchisq)];
title(tt)
subplot(3,1,3)
bar(lkspace,X0(2:kleng))
```

```
xlabel('log(k)')
ylabel('P(k)')
pause(0.1);
%
%
lamscan=3;
while lamscan > 2
   fprintf('\r\r Do you want to fit with a single MEM regularization
parameter?')
   fprintf('\r Or scan through a range of MEM regularization
parameters? \r')
   lamscan=input(' Enter (1) for a single fit or (2) for a scan: (1 or
2) --> ');
   if (isempty(lamscan))
      lamscan=3;
   elseif (lamscan ~= 1) & (lamscan ~= 2)
      lamscan =3;
   end
end
%
%
if lamscan == 1
%
%
   fprintf('\r\r MEM regularization parameter (0 --> LSQ; inf = MEM)
\r');
   memlsq=input(' Enter a value --->  ');
   memlsq=abs(memlsq);
   old_stol=0.0001;
   old_niter=100;
   if memlsq == 0
       memlsq_1=1;
       memlsq_2=1;
       memlsq_delt=1;
   else
       memlsq_1=log(memlsq);
       memlsq_2=log(memlsq);
       memlsq_delt=1;
   end
   ip2=1;
%
%
else
%
%
   goodnum=-1;
   while goodnum < 0
   fprintf('\r\r')
      memlsq_min=input(' Enter the value for the minimum MEM
regularization parameter --> ');
      memlsq_max=input(' Enter the value for the maximum MEM
regularization parameter --> ');
      if memlsq_max > memlsq_min
         goodnum=1;
      else
         fprintf(' maximum must be greater than minimum !')
      end
```

```
   end
   memlsq_rat=input(' Enter the increment for adjacent MEM
regularization parameter [lamda(i+1)/lambda(i)] --> ');
   memlsq_1=log(memlsq_min);
   memlsq_2=log(memlsq_max);
   memlsq_delt=log(memlsq_rat);
   fprintf('\r\r')
   old_stol=input(' Enter the fitting tolerance value (CR = default =
1e-5) ---> ');
   if isempty(old_stol)
      old_stol=1e-5;
   elseif (old_stol > 1) | (old_stol < 0)
      old_stol=1e-5;
   end
%
   fprintf('\r\r')
   old_niter=input(' Enter the maximum number of iterations (CR =
default = 200) ---> ');
   if isempty(old_niter)
      old_niter=200;
   elseif (old_niter <= 1)
      old_niter=200;
   end
   %
   goodname = -1;
   while goodname < 0
      fprintf('\r\r')
      fname=input(' Enter a filename (including directory, but no
extension) for the fit results ---> ', 's');
      if ~isempty(fname)
         goodname=1;
      end
   end
   ip2=2;
%
%
end
%
%
X00=(X0.*0)+(1./(kleng-1));
X00(1,1)=0;
%%%%%%%%%%%%%%%%%%%
%
init_guess=-1;
while init_guess < 0
   fprintf('\r\rInitial guess:\r')
   init_guess=input(' Enter (1) for NNLS guess, or (2) for Maximum
entropy guess (1 or 2) ---> ');
   if isempty(init_guess)
      init_guess=-1;
   elseif (init_guess < 1) | (init_guess > 2)
      init_guess=-1;
   end
end
%
%
%
```

```
icount=0;
for iijj=memlsq_1:memlsq_delt:memlsq_2
   %
   if init_guess == 1
      pfit=X0;
   else
      pfit=X00;
   end

%
icount=icount+1;
%
%
if lamscan ~= 1
    memlsq=exp(iijj);
end
%
stol=old_stol;
niter=old_niter;
iter=0;
stest=1;
ochisq=-1;
chisq=-1;
lamda=-1;
lista=zeros(kleng,1);
alpha=zeros(kleng);
beta=zeros(kleng,1);
more_fit=1;
while more_fit==1
   fprintf('\r\r\r')
   while (stest>stol)&(iter<niter)
%
%
      [ycalc, memout pfit, chisq, alpha, beta,
lamda]=mrqmin_mem(A,memlsq,y,wt,pfit,kleng,func,lamda,alpha,beta,ochisq
);
      if (iter == 0) | (chisq < ochisq)
         if (iter ~= 0)
            stest=abs((chisq-ochisq)./chisq);
         end
         fprintf('Iteration %3i;  Chi-squared = %12.4e;  Fractional
Change = %12.4e \r',iter,chisq,stest);
         ochisq=chisq;
         subplot(3,ip2,1)
         plot(t,y-ycalc,'b',[min(t),max(t)],[0,0],'k--')
         axis([min(t) max(t) 1.05.*min(y-ycalc) 1.05.*max(y-ycalc)])
         ylabel('I-I_{calc}')
         subplot(3,ip2,2)
         plot(t,y,'b',t,ycalc,'r',[min(t),max(t)],[0,0],'k--')
         axis([min(t) max(t) 1.05.*min(min(min(y),min(ycalc)),0)
1.05.*max(max(y),max(ycalc))])
         xlabel('time')
         ylabel('Intensity')
         lsqchisq=((ycalc-y)./wt)'*((ycalc-y)./wt);
         tt=[' LSQ \chi^2 = ' num2str(lsqchisq)];
         title(tt)
         subplot(3,ip2,3)
```

```
         bar(lkspace,pfit(2:kleng))
         xlabel('log(k)')
         ylabel('P(k)')
         pause(0.1);
      end
      iter=iter+1;
%
%
   end
   if lamscan == 1
      if iter >= niter
         fprintf('\r\r Iteration count of %6i exceeded \r\r',niter)
         niter=2.*niter;
      else
         fprintf('\r\r Change in chi-squared less than %12.4e;
iterations stopped \r\r',stol)
         stol=stol./10;
      end
      fprintf(' Do you want to continue iterating ?\r\r')
      more_fit=input(' stop = 0; continue = 1 or CR: --> ');
      if isempty(more_fit)
         more_fit=1;
      end
   else
      more_fit=0;
   end
%
end
%
%
if lamscan == 1
   fprintf('\r\r')
   isave=input(' Enter a (1) to save the data, 0 or CR to quit ---> ');
   if isempty(isave)
      isave=0;
   end
   if isave == 1
      goodname = -1;
      while goodname < 0
         fprintf('\r\r')
         fname=input(' Enter a filename (including directory, but no
extension) for the fit results ---> ', 's');
         if ~isempty(fname)
            goodname=1;
         end
      end
      evalstr=[fname];
      evalstr=['save ',evalstr,' filename t y ycalc kspace lkspace
kleng A pfit chisq lsqchisq memlsq memout X0'];
      eval(evalstr);
%      save fname filename t y ycalc A pfit chisq lsqchisq memlsq
memout X0
   end
else
   pfit_out(:,icount)=pfit;
   ycalc_out(:,icount)=ycalc;
   chisq_out(icount)=chisq;
```

```
   lsqchisq_out(icount)=lsqchisq;
   memlsq_out(icount)=memlsq;
   memout_out(icount)=memout;
   subplot(3,ip2,4)
   hold on
   plot(lkspace,pfit(2:kleng),'b',lkspace,pfit(2:kleng),'bo')
   xlabel('log(k)')
   ylabel('P(k)')
   hold off
   subplot(3,ip2,5)

plot(log10(memlsq_out),lsqchisq_out,'r',log10(memlsq_out),lsqchisq_out,
'ro')
   xlabel('log(MEM regularization parameter)')
   ylabel('LSQ \chi^2')
   subplot(3,ip2,6)
   plot(log10(lsqchisq_out),log10(-
(memlsq_out.^2)./memout_out),'r',log10(lsqchisq_out),log10(-
(memlsq_out.^2)./memout_out),'ro')
   ylabel('log(1/S)')
   xlabel('log(LSQ \chi^2)')
   pause(1);
   evalstr=[fname];
   evalstr=['save ',evalstr,' filename t y ycalc_out kspace lkspace
kleng A pfit_out chisq_out lsqchisq_out memlsq_out memout_out X0'];
   eval(evalstr);
   %   save fname filename t y ycalc_out A pfit_out chisq_out
lsqchisq_out memlsq_out memout_out X0
end
   %
   %
end
%
%
%%%%%%%%%%%%%%%%%%%
```

splice_data1.m

```
%
%   script to splice together streak camera data from different time
bases
%
%
fprintf(' Data must be in two-column (time,intensity) format\n');
files=input('\n How many data sets will be spliced ? ---> ');
%
clear n
for i=1:files
%
    fprintf('\n Data set #%i:',i)
    filename=input(' enter the file name  --> ', 's');
%
    t_y=importdata(filename);
%
    eval(['time' num2str(i) '=t_y(:,1)-t_y(1,1);']);
    eval(['intensity' num2str(i) '=t_y(:,2);']);
    eval(['n(' num2str(i) ')=length(time' num2str(i) ');']);
end
%
%
[a,b]=max(n);
%
Time=zeros(a,files);
Intensity=zeros(a,files);
%
%
for i=1:files
    if n(i) < n(b)
        del=n(b)-n(i);
        eval(['Time(:,i)=[[min(time' num2str(i) ')-del:1:min(time'
num2str(i) ')-1]''; time' num2str(i) '];'])
        eval(['Intensity(:,i)=[zeros(del,1); intensity' num2str(i)
'];'])
    else
        eval(['Time(:,i)=[time' num2str(i) '];'])
        eval(['Intensity(:,i)=[intensity' num2str(i) '];'])
    end
end
%
% order the matrix in increasing time base
%
[P,Q]=sort(max(Time));
Time=Time(:,Q);
Intensity=Intensity(:,Q);
%
for i=1:files
    [a,b]=max(Intensity(:,i));
    Time(:,i)=Time(:,i)-Time(b,i);
    Intensity(:,i)=Intensity(:,i)./a;
end
%
%
[a,b]=size(Time);
```

```
%
first_pnt=floor(0.9.*a);
last_pnt=a;
%
factor=ones(files,1);
for i=2:files
    ytmp=interp1(Time(:,i),Intensity(:,i),Time(first_pnt:last_pnt,i-
1),'spline');
    ytmp=ytmp(:);

factor(i:files)=factor(i:files).*(sum(Intensity(first_pnt:last_pnt,i-
1)./ytmp)./(last_pnt-first_pnt+1));
end
%
%
Intensity=Intensity*diag(factor);
subplot(3,1,1)
semilogx(Time,Intensity)
%
Tsplice=Time(:,1);
Isplice=Intensity(:,1);
%
for i=2:files
    I=find(Time(:,i)>Time(last_pnt,i-1));
    Tsplice=[Tsplice; Time(I,i)];
    Isplice=[Isplice; Intensity(I,i)];
end
%
W = find(Tsplice>=0);
a=size(Tsplice);
Tsplice = Tsplice(W:a);
Isplice = Isplice(W:a);
subplot(3,1,2)
semilogx(Tsplice,Isplice)
%
subplot(3,1,3)
plot(Tsplice,Isplice)
```

crystal.m

```
function chi = crystal(x,R,E,t)
 %This function finds the chi square of a expression derived from
%convolution and data.

[O,Radj] = multiexp_conv(t,R,x);
%O = conv(R,T);
a = length(E);
%O1 = O(1:a);
chi = (O-E)'*(O-E);
%subplot(2,1,1);
semilogx(t,E,'b',t,O,'g')
%subplot(2,1,2);
%plot(t,O)
%plot(t,O1);
%plot(t(P1:P1+125),R(P1:P1+125),'m');
%plot(R);
```

multiexp_conv.m

```
function [O,Radj]=multiexp_conv(x,R,pram)
%
%   Syntax: [O,Radj]=multiexp_conv(x,R,pram);
%
%                   x = time vector
%                   R = response function (must have same length as x)
%
%   fitting to multi-exponential function: f(x)=alpha_0 +
%                                               alpha_1*exp(-alpha_2*x)
+
%                                               alpha_3*exp(-alpha_4*x)
+
%                                                           .
%                                                           .
%                                                           .
%
%   fit parameters: pram
%                       pram(1) = delta (response function time-shift
value)
%                       pram(2) = alpha_0
%                       pram(3) = alpha_1
%                       pram(4) = k_1
%                       pram(5) = alpha_2
%                       pram(6) = k_2
%                               .
%                               .
%                               .
%
x=x(:);
R=R(:);
pram=pram(:);
%
len=length(x);
exps=(length(pram)-2)./2;
%
if length(R) ~= len
    fprintf('Unequal lengths of time vector and response function\r');
    return
end
%
XI=x-pram(1);
Radj=interp1(x',R',XI','spline',0);
Radj=Radj./trapz(x,Radj);
Radj=Radj(:);
%
Rsums=zeros(len,exps+1);
%
for i=2:len
    %
    Rsums(i,1)=Rsums(i-1,1)+(pram(2).*0.5.*(x(i)-x(i-
1)).*(Radj(i)+Radj(i-1)));
    %
    for j=1:exps
        %
        exdx=exp(-pram(2.*(j+1)).*(x(i)-x(i-1)));
```

```
        Rsums(i,j+1)=(Rsums(i-
1,j+1).*exdx)+(pram((2.*j)+1).*0.5.*(x(i)-x(i-1)).*((Radj(i-
1).*exdx)+Radj(i)));
        %
    end
    %
end
%
%
O=sum(Rsums,2);
%
%
%subplot(2,1,1)
%semilogx(x,Radj./max(Radj),'b',x,O,'r')
%subplot(2,1,2)
%plot(x,Radj./max(Radj),'b',x,O,'r')
%
%
```