Appendix A

MATLAB Code for X-ray Data and 2D Correlation Analysis

Due to the acquisition of x-ray data in many file formats, MATLAB code was written in order to facilitate data processing. After determination of beam center and sample-to-detector distance using FIT2D software,[1] the following algorithms were employed.

General code is also provided for conventional and hetero-spectral two-dimensional (2D) correlation analysis, as well as 2D moving window analysis.

A.1 Save Experimental Parameters

This function asks for necessary experimental parameters: beam center ('x0','y0'), sample to detector distance ('L'), detector pixel size ('pixsize'), and x-ray wavelength ('lamda') and saves them in Matlab matrix 'BNLsaxsApr2009.mat'.

```
function getparameters
x0=input('Enter x coordinate of center:');
y0=input('Enter y coordinate of center:');
L=input('Enter sample-to-detector distance in mm:');
pixsize=input('Enter pixel size in microns:');
lamda=input('Enter wavelength in angstroms:');
lamda=lamda*10^-10;
pixsize=pixsize*10^-6;
L=L*10^-3;
x0=x0*pixsize;
y0=y0*pixsize;
save('BNLsaxsApr2009.mat'); % save parameter matrix
end
```

A.2 Open Image Files

This function looks for x-ray images (.tif) in directory 'dname' and creates a movie of I(q) versus q and saves the corresponding 3D matrix. 'TempA' and 'TempB' are the slope and y-intercept of the ramp temperature profile, respectively. Variables 'timeA' and 'timeB' are the slope and y-intercept of the time dependence on the frame number. 'IC1A' and 'IC1B' are the slope and y-intercept of the IC1 value (prior to sample) dependence on the frame number. It load background matrices 'B' which should be double-format and corrected for IC1 and any difference for acquisition time. This function calls radialintegrationBNL.

```
function waxsdataBNL(dname,TempA,TempB,timeA,timeB,IC1A,IC1B,f)
%% prepare for analysis
```

```
%load background matrices (only IC1 normalized)
%load('Bsaxs1s.mat');
%load('Bwaxs1s.mat');
load('Bsaxs7s.mat');
%load('PEL161sri871.mat');

disp(['File selected for analysis: ', dname]);

% create new directories
if isequal(exist(strcat(dname,'\movies')),0)
   mkdir(dname,'movies');
end
if isequal(exist(strcat(dname,'\IvsQ')),0)
   mkdir(dname,'IvsQ');
end

waxsfolder=dname;
waxsfiles=dir(waxsfolder);
wsize=size(waxsfiles);
numfiles=wsize(1);

%get movie name
sfile=waxsfiles(10).name;
index=strfind(sfile,'i'); %'r' before for ramps
newdname=sfile(1:index);
moviefile=strcat(waxsfolder,'\movies\',newdname,'_IvsQ');
mov=avifile(moviefile,'compression','none','fps',2);

%find start of files
track=1;
while waxsfiles(track).isdir~=0
   track=track+1;
end
intensity=zeros(250,3,numfiles-4);
ii=1;
for count=track:numfiles-2;
   if waxsfiles(count).isdir==0
      sfilename=waxsfiles(count).name;
      wholepathwaxs=strcat(waxsfolder,'\',sfilename);
      imagemat=imread(wholepathwaxs,'tiff');
      imagemat=double(imagemat)+1;
      n=strfind(sfilename,'.');
      frame=str2double(sfilename(n+1:n+4));
      time=timeA*frame+timeB;
      Temp=TempA*time+TempB;
```

```
        IC1=IC1A*frame+IC1B;
        imagemat=imagemat/IC1-f*B;
        IvsQ=radialintegrationBNL(imagemat);
        intensity(:,1,ii)=Temp;
        intensity(:,2,ii)=IvsQ(:,1);
        intensity(:,3,ii)=IvsQ(:,2);
        ii=ii+1;
        plot(IvsQ(:,1),IvsQ(:,2));
        %axis([0 1.94 0 1]); %for WAXS
        axis([0 0.12 0 3]); %for SAXS
        set(gca,'YGrid','on');
        time2=strcat('Temp=',num2str(Temp),' \circC');
        text(0.07,.75,time2);
        F=getframe(gcf);
        mov=addframe(mov,F);
    end
end
mov=close(mov);
intname=strcat(waxsfolder,'\IvsQ\',newdname,'_intmat');
save(intname,'intensity');
end
```

A.3 Extract I(q) vs. q

This function requires the input of a 2D matrix containing intensity values (double) corresponding to the x-ray scattering image (‘imagemat’). It returns I(q) vs. q in matrix ‘IvsQ’. It loads the experimental parameters saved in A.1. Start and end pixel counts can be adjusted as needed. It is important to resize matrices ‘allpixels’ and ‘sortedpix’ accordingly.

```
function [IvsQ]=radialintegrationBNL(imagemat)

%load('BNLwaxsApr2009.mat');
load('BNLsaxsApr2009.mat');
%%
allpixels=zeros(160801,2); %WAXS: 463761; SAXS: 160801
pixelnum=1;

for xcount=312:712 %WAXS: 168:848; SAXS: 312:712
    for ycount=311:711 %WAXS: 173:853; SAXS: 311:711
        r=(((0.5*pixsize*(2*xcount-1)-x0)^2+(0.5*pixsize*(2*ycount-1)-y0)^2)^0.5);
        q=4*10^-10*pi*sin(0.5*atan(r/L))/lamda;
        allpixels(pixelnum,1)=q;
        allpixels(pixelnum,2)=imagemat(ycount,xcount);
        pixelnum=pixelnum+1;
    end
```

```
end
%%
sortedpix=zeros(160801,2); %WAXS: 463761; SAXS: 160801
[temp,IX]=sort(allpixels);
for count=1:160801 %WAXS: 463761; SAXS: 160801
   sortedpix(count,1)=temp(count,1);
   sortedpix(count,2)=allpixels(IX(count,1),2);
end

[n,xout]=hist(sortedpix(:,1),250);  %can change bin as desired
sizen=size(n);
bin=sizen(2);
Ivsq=zeros(bin,2);
mark=1;
for count=1:bin
   Ivsq(count,1)=xout(count); %set binned q
   ave=0;
   for ii=mark:mark+n(count)-1
      ave=ave+sortedpix(ii,2)/n(count);
   end
   Ivsq(count,2)=ave;
   mark=mark+n(count);
end

%no dezinger
IvsQ=Ivsq;

end
```

A.4 Lorentz Correction

This function creates and saves the Lorentz-corrected intensity matrix calculated from the intensity matrix created in A.2. It also saves each frame as a separate ascii file for crystallinity evaluation in Origin or other software.

```
function lorentz

[filename,pathname]=uigetfile('*.mat','Select matrix for ascii extraction');
disp(['File selected: ', filename])
A=load(strcat(pathname,filename));
mat=A.intensity;

[m,n,k]=size(mat);
lormat=mat;
temp=zeros(189,2); %don't include Temperature values
for count=1:k
```

```
    for q=1:m
        lormat(q,3,count)=lormat(q,2,count)^2*lormat(q,3,count);
    end
    temp(:,1)=lormat(1:189,2,count); %1st column is Temp values
    temp(:,2)=lormat(1:189,3,count)+0.5;
    if count<10
        name=strcat('L52cw','00',num2str(count));
    elseif (count>9) && (count<100)
        name=strcat('L52cw','0',num2str(count));
    else
        name=strcat('L52cw',num2str(count));
    end
    save(strcat(pathname,name),'temp','-ascii');
end
len=length(filename);
name=filename(1:len-11);
save(strcat(pathname,name,'_lormat'),'lormat');
end
```

A.5 Long Period

This function approximates the long period from the peak of the scattering curve on the frame range from ‘start’ to ‘finish’. Note: not valid for noisy data.

```
function [Lp]=getLp(start,finish)

%% load lorentzian corrected matrix
[filename,pathname]=uigetfile('Select Lorentzian corrected matrix');
A=load(strcat(pathname,filename));
%intensity=A.intensity;
intensity=A.lormat;
%intensity=A.intmat;
%% adjust matrix size
[m1,n,k]=size(intensity);
intensity(130:m1,:,:)=[]; %was 205 for BNL materials
[m2,n,k]=size(intensity);
intensity(1:10,:,:)=[]; %22 for PEL52,PEL161, and MS3 at BNL; 37 for EH4 for
BNL; 10 for LBL materials
[m,n,k]=size(intensity);
%%
Lp=zeros(finish-start+1,2);
track=1;
for count=start:finish
    Lp(track,1)=intensity(1,1,count);
    [c,i]=max(smooth(intensity(:,3,count)));
    Lp(track,2)=0.2*pi/intensity(i,2,count);
```

```
    track=track+1;
end

figure(1);
plot(Lp(:,1),Lp(:,2));
xlabel('Temperature (\circC)');
ylabel('Long period (nm)');
plottitle=input('Enter plot title: ','s');
title(plottitle);

len=length(filename);
name=filename(1:len-4);
save(strcat(pathname,name,'_Lpmat'),'Lp');
save(strcat(pathname,name,'_Lpmat'),'Lp','-ascii');
nameplot=strcat(pathname,name,'_Lpgraph');
saveas(gcf,nameplot,'fig');
saveas(gcf,nameplot,'jpg');
end
```

A.6 SAXS Integrated Intensity

This function calculates the SAXS integrated intensity on a frame interval from ‘start’ to ‘finish’.

```
function [Area]=getArea(start,finish)

%% load lorentzian corrected matrix
[filename,pathname]=uigetfile('Select Lorentzian corrected matrix');
A=load(strcat(pathname,filename));
intensity=A.lormat;
%intensity=A.intmat;
%% adjust matrix size
[m1,n,k]=size(intensity);
intensity(205:m1,:,:)=[];
%[m2,n,k]=size(intensity);
%intensity(1:37,:,:)=[]; %22 for all paper materials; 37 for EH4
[m,n,k]=size(intensity);
%%
Lp=zeros(finish-start+1,2);
track=1;
for count=start:finish
    Area(track,1)=intensity(1,1,count);
    A=0;
    for q=2:m
        A=A+(intensity(q,3,count)+intensity(q-1,3,count))*(intensity(q,2,count)-
intensity(q-1,2,count))/2;
```

```
    end
    Area(track,2)=A;
    track=track+1;
end

figure(1);
plot(Area(:,1),Area(:,2));
xlabel('Temperature (\circC)');
ylabel('Invariant');
plottitle=input('Enter plot title: ','s');
title(plottitle);

len=length(filename);
name=filename(1:len-4);
save(strcat(pathname,name,'_Areamat'),'Area');
save(strcat(pathname,name,'_Areamat'),'Area','-ascii');
nameplot=strcat(pathname,name,'_Areagraph');
saveas(gcf,nameplot,'fig');
saveas(gcf,nameplot,'jpg');
end
```

A.7 Conventional 2D Correlation Analysis

```
function corrmat
%% Conventional 2D correlation analysis on a ax2xm matrix containing
%% intensity values at equally spaced points along a perturbation

%% load matrix
[filename,pathname]=uigetfile('Select mutli-spectra matrix for analysis');
disp(['File selected: ', filename])
A=load(strcat(pathname,filename));
sepmat=A.sepmat; %adjust if matrix name is not sepmat (ex:
    sepmat=A.matrix_name;)

%%
[a,b,m]=size(sepmat);
range=a;

figure(1);
hold on;
for speccount=1:m
    plot(sepmat(:,1,speccount),sepmat(:,2,speccount)); %plot all spectra
    y(speccount,:)=sepmat(:,2,speccount);
end
xlabel('Raman Shift (cm^{-1})'); %set axis
ylabel('Intensity'); %set axis
```

```
plottitle=input('Enter plot title:','s');
title(plottitle);
hold off;

len=length(filename);
name=filename(1:len-8);
nameall=strcat(pathname,name);
saveas(gcf,nameall,'fig');
saveas(gcf,nameall,'jpg');

%% calculate differential matrix and plot (not necessary)
diffmat=zeros(range,2,m-1);
figure(5);
hold on;
for count=1:m-1
   diffmat(:,1,count)=sepmat(:,1,count);
   diffmat(:,2,count)=sepmat(:,2,count+1)-sepmat(:,2,count);
   plot(diffmat(:,1,count),diffmat(:,2,count));
end
xlabel('Raman Shift (cm^{-1})'); %set axis
ylabel('Difference in Intensity'); %set axis
title({plottitle;'Change in Spectra'});
hold off;
namediff=strcat(pathname,name,'_diff');
saveas(gcf,namediff,'fig');
saveas(gcf,namediff,'jpg');

depvar=sepmat(:,1,1); % spectral variables
dim=length(depvar);

%% calculate perturbation-averaged spectrum
yave=zeros(1,dim);
for count=1:m
   yave=yave+y(count,:);
end
yave=yave/m;

%% plot perturbation-averaged spectrum
figure(2);
plot(depvar,yave);
xlabel('Raman Shift (cm^{-1})'); %set axis
ylabel('Intensity'); %set axis
title({plottitle;'Averaged Spectra'});
nameave=strcat(pathname,name,'_ave');
saveas(gcf,nameave,'fig');
```

```
saveas(gcf,nameave,'jpg');

%% save average spectrum
avemat(:,1)=depvar;
avemat(:,2)=yave;
nameavemat=strcat(pathname,name,'_avemat');
save(nameavemat,'avemat');

%% calculate dynamic spectrum
for count=1:m
   ys(count,:)=y(count,:)-yave;
end

%% calculate synchronous spectrum
Phi=(1/(m-1))*ys'*ys;

%% if want to set noise threshold to 5%
%[a,a]=size(Phi);
%maxPhi=max(max(Phi));
%for count1=1:a
%    for count2=1:a
%        if abs(Phi(count1,count2))<0.05*maxPhi
%            Phi(count1,count2)=0;
%       end
%    end
%end

%% plot and save synchronous spectrum
figure(3);
contour(depvar',depvar',Phi);
%imagesc(depvar',depvar',Phi); %if contour is too noisy
hold on;
plot(depvar,depvar);
xlabel('Raman Shift (cm^{-1})');
ylabel('Raman Shift (cm^{-1})');
title({'Synchronous 2D Correlation Spectra:';plottitle});
axis xy;
hold off;
namePhi=strcat(pathname,name,'_Phi');
saveas(gcf,namePhi,'fig');
saveas(gcf,namePhi,'jpg');
namePhimat=strcat(pathname,name,'_Phimat');
save(namePhimat,'Phi');

%% calculate asynchronous spectrum
```

```
N=makeN(m); % calculate Hilbert-Noda matrix (see A.8)
Psi=(1/(m-1))*ys'*N*ys;

%% if want to set noise threshold to 5%
%[b,b]=size(Psi);
%maxPsi=max(max(Psi));
%for count1=1:b
%    for count2=1:b
%        if abs(Psi(count1,count2))<0.05*maxPsi
%            Psi(count1,count2)=0;
%        end
%    end
%end

%% plot and save asynchronous spectrum
figure(4);
contour(depvar',depvar',Psi');
%imagesc(depvar',depvar',Psi'); %if countour is too noisy
hold on;
plot(depvar,depvar);
xlabel('Raman Shift (cm^{-1})');
ylabel('Raman Shift (cm^{-1})');
title({'Asynchronous 2D Correlation Spectra';plottitle});
axis xy;
hold off;
namePsi=strcat(pathname,name,'_Psi');
saveas(gcf,namePsi,'fig');
saveas(gcf,namePsi,'jpg');
namePsimat=strcat(pathname,name,'_Psimat');
save(namePsimat,'Psi');

%% calculate, plot, and save autocorrelation intensity
v=diag(Phi);
figure(6);
plot(depvar',v);
xlabel('Raman Shift (cm^{-1})');
ylabel('Autocorrelation Intensity');
title({'Autocorrelation Spectrum';plottitle});
nameapow=strcat(pathname,name,'_pow');
saveas(gcf,nameapow,'fig');
saveas(gcf,nameapow,'jpg');
apowmat(:,1)=depvar;
apowmat(:,2)=v;
nameapowmat=strcat(pathname,name,'_apowmat');
save(nameapowmat,'apowmat');
```

```
%%
end
```

A.8 Hilbert-Noda Matrix

```
function [N]=makeN(m)
%% calculate Hilbert-Noda matrix
N=zeros(m);
for j=1:m
   for k=1:m
      if j==k
         N(j,k)=0;
      else
         N(j,k)=1/(pi*(k-j));
      end
   end
end
end
```

A.9 2D Hetero-Spectral Analysis

```
function heterocorr
%% Conduct 2D hetero-spectral correlation analysis on two matrices (here
%% SAXS and WAXS data) that contain the same number of points along the
%% perturbation. This is written for ax3xm matrices where there are a
%% spectral variables (which are located in column 2) and m points along
%% the perturbation. Column 3 contains the intensity values. Column 1
%% contains temperatures (not used here).

%% load matrices
[filename1,pathname1]=uigetfile('Select SAXS matrix');
disp(['File selected: ', filename1])
A=load(strcat(pathname1,filename1));
sepmat1=A.intensity; %for matrix named intensity or change to
     sepmat1=A.matrix_name;
[filename2,pathname2]=uigetfile('Select WAXS matrix');
disp(['File selected: ', filename2])
A=load(strcat(pathname2,filename2));
sepmat2=A.intensity; %for matrix named intensity or change to
     sepmat2=A.matrix_name;

%% establish matrix size -- note: it is required that m1=m2
[a1,b1,m1]=size(sepmat1);
[a2,b2,m2]=size(sepmat2);
%%
for speccount=1:m1
```

```
   y1(speccount,:)=sepmat1(:,3,speccount); %change if only 2 column
matrix
end
for speccount=1:m2
   y2(speccount,:)=sepmat2(:,3,speccount); %change if only 2 column matrix
end
len=length(filename1);
name=filename1(1:len-4);
nameall=strcat(pathname1,name);

depvar1=sepmat1(:,2,1); % spectral variables set 1
depvar2=sepmat2(:,2,1); % spectral variables set 2
dim1=length(depvar1);
dim2=length(depvar2);

%% calculate average spectra for each dataset
yave1=zeros(1,dim1);
yave2=zeros(1,dim2);
for count=1:m1
   yave1=yave1+y1(count,:);
end
yave1=yave1/m1;
for count=1:m2
   yave2=yave2+y2(count,:);
end
yave2=yave2/m2;

%% plot average spectra for each dataset
figure(1);
plot(depvar1,yave1);
xlabel('q (A^{-1})');
ylabel('Intensity');
title('Averaged Spectra');
nameave=strcat(pathname1,name,'_ave1');
saveas(gcf,nameave,'fig');
saveas(gcf,nameave,'jpg');
namedep1=strcat(pathname1,name,'_depvar1');
save(namedep1,'depvar1');
figure(2);
plot(depvar2,yave2);
xlabel('q (A^{-1})');
ylabel('Intensity');
title('Averaged Spectra');
nameave=strcat(pathname1,name,'_ave2');
saveas(gcf,nameave,'fig');
```

```
saveas(gcf,nameave,'jpg');
namedep2=strcat(pathname1,name,'_depvar2');
save(namedep2,'depvar2');

%% calculate dynamic spectra for each dataset
for count=1:m1
   ys1(count,:)=y1(count,:)-yave1;
end
for count=1:m2
   ys2(count,:)=y2(count,:)-yave2;
end

%% calculate synchronous spectra
Phi=(1/(m1-1))*ys1'*ys2;
[a,a]=size(Phi);
maxPhi=max(max(Phi));
for count1=1:a
   for count2=1:a
      if abs(Phi(count1,count2))<0.05*maxPhi
         Phi(count1,count2)=0;
      end
   end
end

%% plot synchronous spectrum
figure(3);
contour(depvar2',depvar1',Phi);
%imagesc(depvar',depvar',Phi); %if contours are too noisy
xlabel('q (A^{-1})'); %set axis
ylabel('q (A^{-1})'); %set axis
title('Synchronous 2D Correlation Spectra:'); %set title
axis xy;

%% save synchronous spectrum
namePhi=strcat(pathname1,name,'_Phi');
saveas(gcf,namePhi,'fig');
saveas(gcf,namePhi,'jpg');
namePhimat=strcat(pathname1,name,'_Phimat');
save(namePhimat,'Phi');
end
```

A.10 2D Moving Window Analysis

```
function [movwin]=MW2D(intensity,win)
```

```
%% conduct moving window 2D analysis on matrix 'intensity' of window
of size 'win'
[m,n,k]=size(intensity);
%% load desired spectra into y (general spectra)
for count=1:k
   y(count,:)=intensity(:,2,count); %for 2 column matrix
end
%% now conduct analysis on smaller windows`
depvar=intensity(:,1,1); %spectral variable
movwin=zeros(k-win+1,m); %2D moving window plot
ycord=zeros(k-win+1,1); %perturbation coordinate
for start=1:(k-win+1)
   for count=1:win
      ywin(count,:)=y(count+start-1,:);
   end
   yave=zeros(1,m);
   %calculate average spectrum
   for count=1:win
      yave=yave+ywin(count,:);
   end
   yave=yave/win;
   %calculate dynamic spectra
   for count=1:win
      ys(count,:)=ywin(count,:)-yave;
   end
   Phi=(1/(win-1))*ys'*ys; %calculate synchronous spectrum
   movwin(start,:)=diag(Phi); %take autocorrelation intensity
    frame=start+floor(win/2); % take representative value of frame (or perturbation)
     for 'window'
   time=10*frame; %set time dependence on frame (or each perturbation point)
   ycord(start)=time;
end

%% plot 2D moving window plot
contour(depvar,ycord,movwin,10); %set to 10 contours
xlabel('q (Angstroms^{-1})'); %set axis
ylabel('Temperature'); %set axis
end
```

A.11 ACKNOWLEDGEMENTS

Motivation for this work came from Dr. Eleni Pavlopoulou (University of Athens, Greece) and Dr. Luigi Balzano (Technical University of Eindhoven, Netherlands). Understanding of x-ray data analysis was facilitated by Dr. Lucia Fernandez Ballester

(DUBBLE, ESRF, Grenoble, France), Dr. Timothy Gough (University of Bradford, UK), Dr. Giuseppe Portale (DUBBLE, ESRF, Grenoble, France), and Dr. Wim Bras (DUBBLE, ESRF, Grenoble, France). Code development was assisted by Eliot Gann (ALS, Lawrence Berkeley National Lab), Andrew Metcalf (Caltech) and Meisam Hajimorad (Caltech). This work was greatly assisted by the National Science Foundation (NSF) International Research and Education in Engineering (IREE) initiative.

A.12 REFERENCES

1. Hammersley, A. FIT2D. http://www.esrf.eu/computing/scientific/FIT2D/