MOF-FIT – A Matlab routine for determining breathing angles and other crystal structure deformations of MOFs using a visual fit and a graphical user interface

Brief Instructions

The MATLAB working directory should be populated with the following files:

breathing.m – MATLAB script for dynamic simulation of PXRD patterns of a MOF with varying breathing angle as input.

resgenerator_1.m - MATLAB script for generating a .res file containing atomic coordinates (in P1) for a structure with the input breathing angle θ.

funcslider.m –A freely available MATLAB script for continuously updating sliders to view how a function changes as its input parameters change. (http://www.mathworks.com/matlabcentral/fileexchange/28076-function-parameter-slider/content/funcslider.m)

scattering_factor_parameters.m - A MATLAB script containing atomic scattering factor parameters for powder pattern simulation.

MOF_coords.txt – A text file containing atomic fractional coordinates listed as:

Atom	type	x	y	z	occupancy

pxrd.txt – A text file containing the experimental powder X-ray diffraction data listed in two columns:

2 θ	Intensity

Dynamic simulation of powder X-ray diffraction pattern of a MOF with breathing angle θ:

Input the "*MOF_coords.txt*" (line 19), "*pxrd.txt*" (line 23), and "*MOFbreath.res*" (line 25) filenames, unit cell parameters (lines 34-39), and space group symmetry elements (lines 72-79) into *breathing.m*

See comments in *breathing.m* and detailed instructions below for other parameters that may be modified as needed.

Run the dynamic modeling function via funcslider using the command: funcslider(@breathing)

A dialog window displaying the experimental PXRD pattern will appear. Enter 90° as the "Max" and "Value" breathing angle 90 ° and the corresponding simulated pattern will appear. Use the slider bar to vary the breathing angle from 0-90° until a fit is observed.

Close the Function Slider dialog window when finished.

Generating a .res file with new coordinates based on breathing angle θ:

Open *breathing.m*, scroll to the last line of the script, and remove the comment character (%) preceding the command:
“resgenerator_1(a,b,c,alpha,beta,gamma,atom_type,x,y,z,occupancy,resfilename)”.
Save and close *breathing.m*.

Type “breathing(θ)” where θ is the value of the breathing angle for which the .res file should be generated. (i.e. “breathing(90)”). The generated .res file can be found in the MATLAB working directory with the desired filename input at line 25 in breathing.m. This file contains atomic coordinates (in P1) for the structure generated with the input breathing angle θ.

Full Description and Instructions:

A distortion that is somewhat common in MOFs is known as breathing, in which a pore shaped approximately like a rhombohedral prism can dilate and contract in the presence of different stimuli. The vertices of these rhombus-shaped pores are infinite chains of metal ions and the sides of the rhombus are constructed by organic linkers. As seen in Figure 1, we would like to manipulate the angle θ by moving a slider and compare calculated and experimental powder patterns. In the MOF-FIT, we note that the length, $L/2$, between metal chains and the angle θ uniquely define the axes a and b while leaving c unchanged. The distance L can be determined using the original a and b by noting that

$$L = \sqrt{a^2 + b^2}$$

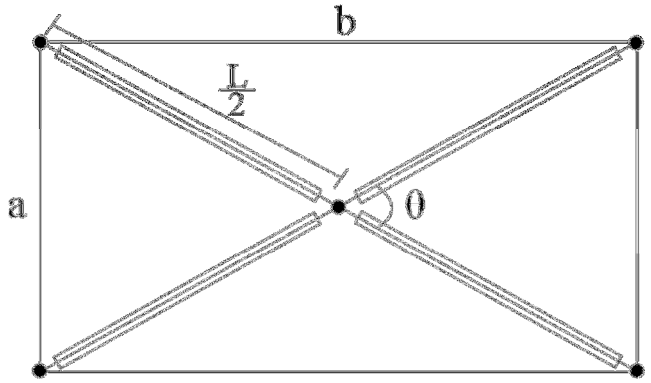


Figure 1 Schematic drawing of a breathing MOF.

Now, specifying an angle θ, we see that we can determine new values of a and b, denoted a' and b' as follows

$$a' = L \sin\frac{\theta}{2}$$
$$b' = L \cos\frac{\theta}{2}$$

With all the alterations complete, the powder X-ray pattern can be calculated.

We begin by calculating the scattering factor

$$sf = \sum_{n=1}^{4} a_n \exp\left(-\frac{b_n}{\lambda^2}\sin\frac{2\theta}{2}\right) + c$$

where the a_n, b_n, and c values are given in crystallographic tables and are different for each atom.[119] The angle 2θ is the angle of the reflection. In order to determine 2θ, we can first find the volume of the unit cell given by

$$V = abc\sqrt{1 - \cos^2\alpha - \cos^2\beta - \cos^2\gamma + 2\cos\alpha\cos\beta\cos\gamma}$$

The spacing between planes, d_{hkl} is then given by

$$\frac{1}{d_{hkl}{}^2} = \frac{1}{V^2}(h^2b^2c^2\sin^2\alpha + k^2a^2c^2\sin^2\beta + l^2a^2b^2\sin^2\gamma + 2hkabc^2(\cos\alpha\cos\beta - \cos\gamma) + 2kla^2bc(\cos\beta\cos\gamma - \cos\alpha) + 2hlab^2c(\cos\alpha\cos\gamma - \cos\beta))$$

from which we can determine the angle 2θ to be

$$2\theta = 2\sin^{-1}\left(\frac{\lambda}{2d_{hkl}}\right)$$

Each atom's subsequent contribution to the overall structure factor, *S*

$$S_{hkl} = sf \cdot e^{-B_{iso}\sin(2\theta/2)/\lambda^2} e^{2\pi i(hx+ky+lz)} \cdot \text{occupancy}$$

where B_{iso} is the isotropic Debye-Waller factor. This can be disregarded with minimal effect on the calculated powder pattern. The individual contributions to the overall structure factor can then be summed and multiplied by its complex conjugate to obtain $F_{hkl}{}^2$.

In the case of preferential orientation along a vector h_{pref}, k_{pref}, l_{pref} for a degree of preferential orientation $p \geq 0$, one model of preferential orientation modeling used here suggests that the squared structure factor can be modified as

$$F_{hkl}{}^{2'} = F_{hkl}{}^2\exp(p\cos 2\phi)$$

for an angle ϕ between the preferred direction and *h*, *k*, *l* given by

$$\phi = \cos^{-1}\left(\frac{hh_{\text{pref}} + kk_{\text{pref}} + ll_{\text{pref}}}{\sqrt{h^2+k^2+l^2}\sqrt{h_{\text{pref}}{}^2 + k_{\text{pref}}{}^2 + l_{\text{pref}}{}^2}}\right)$$

The intensity, *I*, of each reflection is proportional to the combined structure factor $F_{hkl}{}^2$.

Once the structure factors have been determined, the calculated profile can be determined using a pseudo-Voigt fit. This fit consists of a linear combination of a Lorenzian and Gaussian fits. A parameter, η, describes the proportion of the profile described by a Lorenzian. The overall profile, $f(2\theta)$, is then

$$f(2\theta) = \eta L + (1-\eta)G$$

where

$$L(2\theta) = \frac{F_{hkl}{}^2}{(2\theta - 2\theta_0)^2 + (0.5\gamma)^2}$$

and

$$G(2\theta) = F_{hkl}{}^2\exp\left(\frac{-(2\theta-2\theta_0)^2}{\frac{\gamma^2}{4\ln(2)}}\right)$$

with γ representing the full width at half maximum. The last step is to introduce the Lorentz-Polarization factor, which is the product of two terms, the Lorentz factor and the polarization factor. The Lorentz factor arises from three different phenomena: a) the arms of the diffractometer move at a constant angular velocity, but the time afforded to each reflection plane is not constant; b) diffraction for a crystal can occur for angles slightly deviating from Bragg's law; c) the difference in the amount of crystals oriented in such a way that they satisfy Bragg's law. This term, *L*, can be shown to be

$$L = \frac{1}{4\sin^2\theta\cos\theta}$$

The next factor, the polarization factor, arises from the unpolarized nature of the incident X-rays and the fact that the electron off of which the X-ray is scattering emits radiation parallel to the direction of incidence. This factor can be shown to be

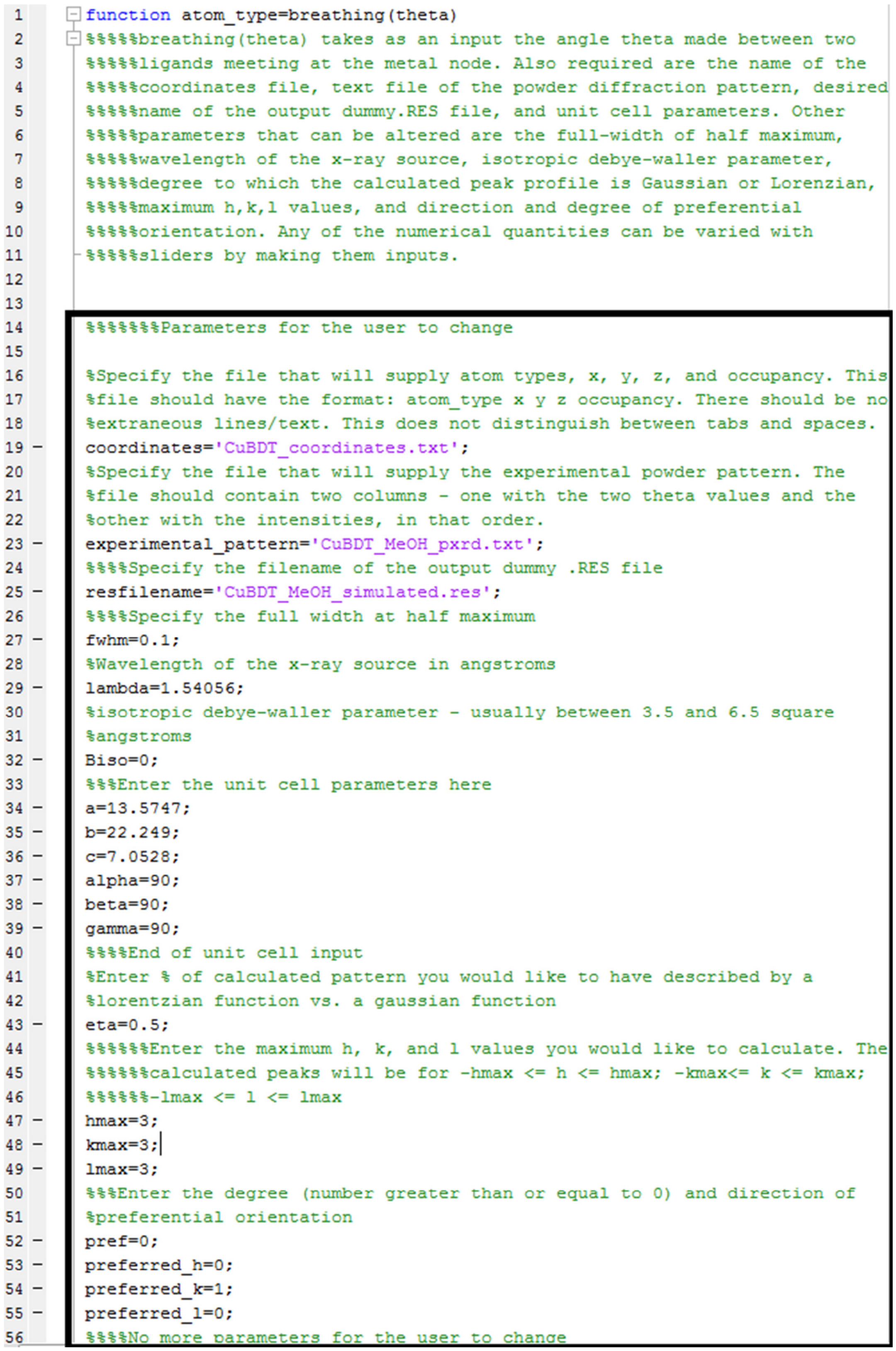
$$P = \frac{1+\cos^2 2\theta}{2}$$

Thus, the product of these two factors, *LP*, alters the intensity of the observed radiation such that the intensity after accounting for these effects, *I'*, is given by

$$I' = I\frac{1+\cos^2 2\theta}{8\sin^2\theta\cos\theta}$$

MOF-FIT is a function implemented in MATLAB that can be used in conjunction with `funcslider.m` (available free for charge at http://www.mathworks.com/matlabcentral/fileexchange/28076-function-parameter-slider/content/funcslider.m) to dynamically model breathing behavior in MOFs. As its input, it takes the angle θ made between ligands connected by a metallic vertex. The output is a plot containing both the experimental and calculated powder patterns as well as a dummy .RES file if desired. As presently constructed the user can move a slidebar to vary the angle θ and view the calculated pattern versus the experimental pattern in real time. A variety of other parameters can be varied in this manner as well.

In order to use MOF-FIT, place the files `breathing.m`, `resgenerator_1.m`, `funcslider.m`, and `scattering_factor_parameters.m` into the working directory in MATLAB. Edit `breathing.m` (Figure 2) to reflect the desired files and parameters needed for the simulation. The parameters that should be edited are indicated in the box.

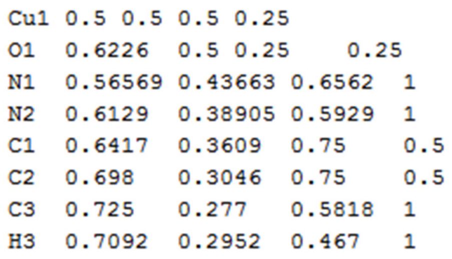


```
function atom_type=breathing(theta)
%%%%%breathing(theta) takes as an input the angle theta made between two
%%%%%ligands meeting at the metal node. Also required are the name of the
%%%%%coordinates file, text file of the powder diffraction pattern, desired
%%%%%name of the output dummy.RES file, and unit cell parameters. Other
%%%%%parameters that can be altered are the full-width of half maximum,
%%%%%wavelength of the x-ray source, isotropic debye-waller parameter,
%%%%%degree to which the calculated peak profile is Gaussian or Lorenzian,
%%%%%maximum h,k,l values, and direction and degree of preferential
%%%%%orientation. Any of the numerical quantities can be varied with
%%%%%sliders by making them inputs.


%%%%%%%Parameters for the user to change

%Specify the file that will supply atom types, x, y, z, and occupancy. This
%file should have the format: atom_type x y z occupancy. There should be no
%extraneous lines/text. This does not distinguish between tabs and spaces.
coordinates='CuBDT_coordinates.txt';
%Specify the file that will supply the experimental powder pattern. The
%file should contain two columns - one with the two theta values and the
%other with the intensities, in that order.
experimental_pattern='CuBDT_MeOH_pxrd.txt';
%%%%Specify the filename of the output dummy .RES file
resfilename='CuBDT_MeOH_simulated.res';
%%%%Specify the full width at half maximum
fwhm=0.1;
%Wavelength of the x-ray source in angstroms
lambda=1.54056;
%isotropic debye-waller parameter - usually between 3.5 and 6.5 square
%angstroms
Biso=0;
%%%Enter the unit cell parameters here
a=13.5747;
b=22.249;
c=7.0528;
alpha=90;
beta=90;
gamma=90;
%%%%End of unit cell input
%Enter % of calculated pattern you would like to have described by a
%lorentzian function vs. a gaussian function
eta=0.5;
%%%%%%Enter the maximum h, k, and l values you would like to calculate. The
%%%%%%calculated peaks will be for -hmax <= h <= hmax; -kmax<= k <= kmax;
%%%%%%-lmax <= l <= lmax
hmax=3;
kmax=3;
lmax=3;
%%%Enter the degree (number greater than or equal to 0) and direction of
%preferential orientation
pref=0;
preferred_h=0;
preferred_k=1;
preferred_l=0;
%%%%No more parameters for the user to change
```

Figure 2 Area for the user to edit in `breathing.m`

All text preceded by the percent sign (%) are comments and not read by MATLAB when executing the script. All comments refer to commands immediately below the comment.
In line 19, the user should replace ‘`CuBDT_coordinates.txt`’ with the name of the file containing the following information:

Atom type x y z occupancy
A sample input file is shown below (Figure 3). The new file name should be surrounded by single quotation marks as indicated in the sample shown in line 19.



```
Cu1 0.5 0.5 0.5 0.25
O1  0.6226  0.5 0.25    0.25
N1  0.56569 0.43663 0.6562  1
N2  0.6129  0.38905 0.5929  1
C1  0.6417  0.3609  0.75    0.5
C2  0.698   0.3046  0.75    0.5
C3  0.725   0.277   0.5818  1
H3  0.7092  0.2952  0.467   1
```

Figure 3 Sample input file

This file can easily be generated from a .RES file by removing the header, deleting the second column in the atom list, and deleting the last column in the atom list.

The next input is on line 23, the experimental powder x-ray pattern file. This should contain two columns – the 2θ values and the intensities.

The next input, on line 25, is the desired filename for the output dummy .RES file used to visualize the calculated structure.

The rest of the inputs are parameters for the simulation. The first parameter to input, `fwhm`, is the full width at half maximum in units of $°2\theta$. The second is the wavelength of the x-rays used. The current value is that of Cu Kα radiation. The next parameter is `Biso`, the Debye-Waller factor, is set to 0 by default. The value is usually very small for molecular systems and still fairly small for other condensed phases. The value is unlikely to make a significant impact on the calculated pattern. This value serves to attenuate the structure factor at high angles with an exponential envelope.

The next set of inputs is the unit cell parameters of the original structure described by the coordinates file.

Since the calculated powder pattern is a pseudo-Voigt profile, it comprises a linear combination of a Gaussian and Lorenzian fit. The parameter eta corresponds to the proportion of the profile will be described by the Lorenzian profile.

The next three parameters are the maximum h, k, and l values for which peak positions and intensities will be calculated.

Parameters may also be input to correct for preferential orientation. The first parameter, `pref`, represents the degree to which the sample is preferentially oriented. This should be a number greater than or equal to 0, where 0 represents no preferential orientation. The next three parameters are the direction of preferential orientation.

Lastly, symmetry elements associated with crystallographic space group of the input structure should be input by editing lines 72-79 (unless the structure contains no symmetry, i.e. P1). The elements are included in matrix format as shown below for the orthorhombic space group Imma:

```
%%%%Input the symmetry elements present here

x=[x2;0.5+x2];y=[y2;0.5+y2];z=[z2;0.5+z2];
```

(0.5+x2 introduces body centering)

```
atom_type_and_number=[atom_type_and_number.',atom_type_and_number.'].';

occupancy=[occupancy;occupancy];
```

The number of elements in the atom_type_and_number and occupancy matrices should reflect the number of elements in the centering matrix.

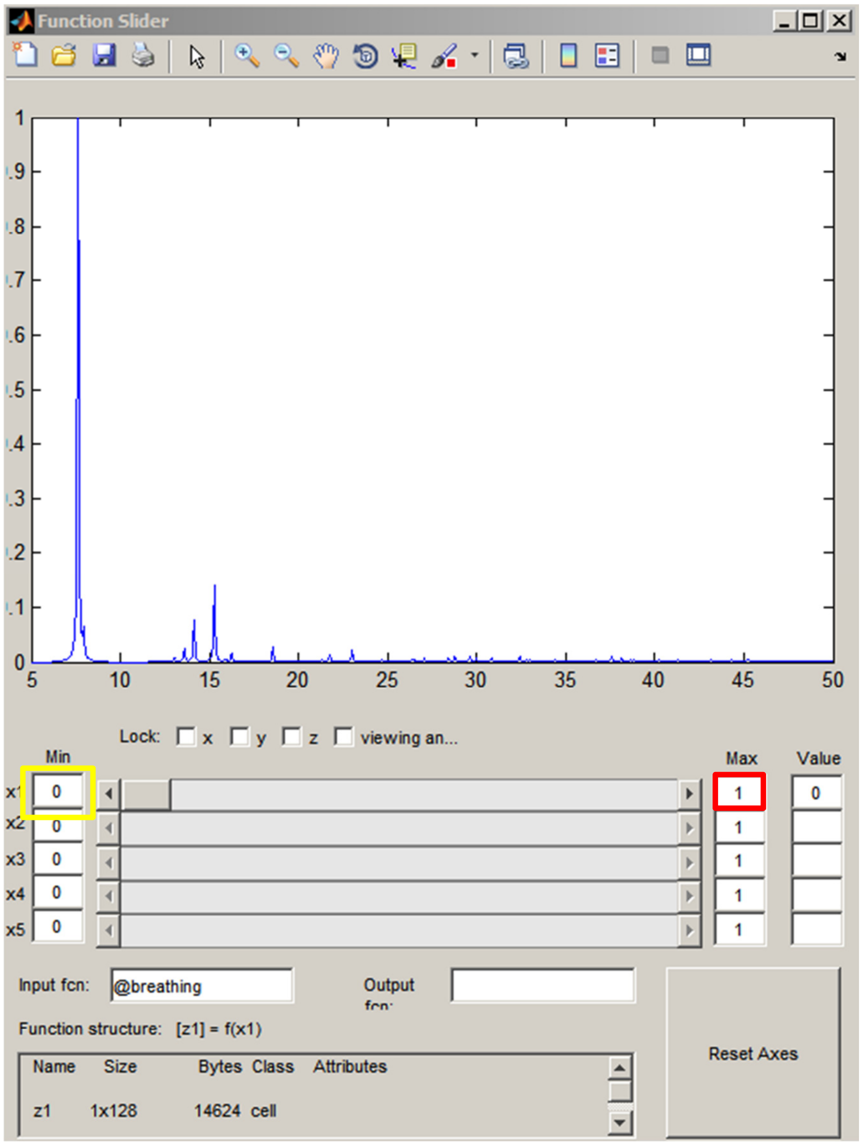
```
x=[x2;0.0-x2;0.0+x2;0.0+x2;0.0-x2;0.0+x2;0.0-x2;0.0-x2];

y=[y2;0.0+y2;0.0-y2;0.0+y2;0.0-y2;0.0-y2;0.0+y2;0.0-y2];

z=[z2;0.5+z2;0.0+z2;0.5-z2;0.0-z2;0.5-z2;0.0-z2;0.5+z2];
```

The 8 remaining symmetry transformations for Imma are entered in matrix format.

```
atom_type_and_number=[atom_type_and_number.',atom_type_and_number.',ato
m_type_and_number.',atom_type_and_number.',atom_type_and_number.',atom_
type_and_number.',atom_type_and_number.',atom_type_and_number.'].';
```

The number of elements in the atom_type_and_number and occupancy matrices reflect the number of elements in the symmetry transformation matrix.

Once all the parameters have been set as desired, one can proceed to run the script in MATLAB. In the command window, the program will run with sliders upon typing `funcslider(@breathing)` and pressing enter (Figure 4). This will bring up a window shown below depicting the experimental powder pattern in blue. An error will be displayed in the command window. This is because the default angle is 0, thus making one of the unit cell parameters 0.



Lock: ☐ x ☐ y ☐ z ☐ viewing an...

	Min	Max	Value
x1	0	1	0
x2	0	1	
x3	0	1	
x4	0	1	
x5	0	1	

Input fcn: @breathing

Output fcn:

Function structure: [z1] = f(x1)

```
Name    Size       Bytes  Class   Attributes

z1      1x128      14624  cell
```

Figure 4 Functional slider interface

The first step is to increase the maximum value to 90, which corresponds to a ligand-metal-ligand angle of 90°. To do this, click inside the textbox indicated in red and type `90` followed by either tab or enter. The slider boxed in yellow can now be moved to the desired angle. The calculated powder pattern will appear in red. During this time, a dummy .RES file will be created every time the calculated pattern changes. If there is a considerable lag between movement of the slider and an update of the calculated pattern, the last line of `breathing.m` can be commented out by placing a percent sign in front of the text. The coding of the .RES file generator is inefficient and can significantly slow down the problem. If this is the case, after obtaining a satisfactory angle, the applet window can be closed. The last line of `breathing.m` can be uncommented and the user can type `breathing(angle)`, where `angle` is the desired value of the angle, in the command window and then press enter to generate the .RES file.

As can be seen in the functional slider applet, the script `funcslider.m` has the ability to accommodate 5 parameters that can be varied using the slider bars. Although the program as currently constructed only uses one of the parameters, it can be easily modified to vary other relevant parameters. For example, if one would like to vary the full width at half maximum of the calculated peaks, one can place a percent sign in front of line 27, thus commenting out the contents of that line. The variable `fwhm` can then be made an argument of the function '`breathing`' by altering the first line to read as follows:

```
function atom_type=breathing(theta, fwhm)
```

The second slide bar will now become active.

breathing.m

```
function atom_type=breathing(theta)
%%%%%breathing(theta) takes as an input the angle theta made between two
%%%%%ligands meeting at the metal node. Also required are the name of the
%%%%%coordinates file, text file of the powder diffraction pattern, desired
%%%%%name of the output dummy.RES file, and unit cell parameters. Other
%%%%%parameters that can be altered are the full-width of half maximum,
%%%%%wavelength of the x-ray source, isotropic debye-waller parameter,
%%%%%degree to which the calculated peak profile is Gaussian or Lorenzian,
%%%%%maximum h,k,l values, and direction and degree of preferential
%%%%%orientation. Any of the numerical quantities can be varied with
%%%%%sliders by making them inputs.


%%%%%%%Parameters for the user to change

%Specify the file that will supply atom types, x, y, z, and occupancy. This
%file should have the format: atom_type x y z occupancy. There should be no
%extraneous lines/text. This does not distinguish between tabs and spaces.
coordinates='CuBDT_coordinates.txt';
%Specify the file that will supply the experimental powder pattern. The
%file should contain two columns - one with the two theta values and the
%other with the intensities, in that order.
experimental_pattern='CuBDT_MeOH_pxrd.txt';
%%%%Specify the filename of the output dummy .RES file
resfilename='CuBDT_MeOH_simulated.res';
%%%%Specify the full width at half maximum
fwhm=0.1;
%Wavelength of the x-ray source in angstroms
lambda=1.54056;
%isotropic debye-waller parameter - usually between 3.5 and 6.5 square
%angstroms
Biso=0;
%%%Enter the unit cell parameters here
a=13.5747;
b=22.249;
c=7.0528;
alpha=90;
beta=90;
gamma=90;
%%%%End of unit cell input
%Enter % of calculated pattern you would like to have described by a
%lorentzian function vs. a gaussian function
eta=0.5;
%%%%%%Enter the maximum h, k, and l values you would like to calculate. The
%%%%%%calculated peaks will be for -hmax <= h <= hmax; -kmax<= k <= kmax;
%%%%%%-lmax <= l <= lmax
hmax=3;
kmax=3;
lmax=3;
%%%Enter the degree (number greater than or equal to 0) and direction of
%preferential orientation
pref=0;
preferred_h=0;
preferred_k=1;
```

```
preferred_l=0;
%%%%No more parameters for the user to change


%%%%%%%%Program starts here



%%%Converting to radians
alpha=alpha*pi/180;
beta=beta*pi/180;
gamma=gamma*pi/180;
theta=theta*pi/180;
%Reads in the atom type with number appended, x, y, and z from the
%specified file
[atom_type_and_number,x,y,z,occupancy]=textread(coordinates,'%s %f %f %f
%f');
%%%%Input the symmetry elements present here
x=[x;0.5+x];y=[y;0.5+y];z=[z;0.5+z];
atom_type_and_number=[atom_type_and_number.',atom_type_and_number.'].';
occupancy=[occupancy;occupancy];
x=[x;0.0-x;0.0+x;0.0+x;0.0-x;0.0+x;0.0-x;0.0-x];
y=[y;0.0+y;0.0-y;0.0+y;0.0-y;0.0-y;0.0+y;0.0-y];
z=[z;0.5+z;0.0+z;0.5-z;0.0-z;0.5-z;0.0-z;0.5+z];
atom_type_and_number=[atom_type_and_number.',atom_type_and_number.',atom_type
_and_number.',atom_type_and_number.',atom_type_and_number.',atom_type_and_num
ber.',atom_type_and_number.',atom_type_and_number.'].';
occupancy=[occupancy;occupancy;occupancy;occupancy;occupancy;occupancy;occupa
ncy;occupancy];
%%%%%%%%%%%Now trying to strip the number from the atom name
atom_type={''};
for n=1:length(atom_type_and_number)
    current=char(atom_type_and_number(n));
    %%%%Change the atom type+number from a cell to text
    if isempty(regexp(current,'\d'));
        atomtype_new=current;
    else
        atomtype_new=current(1:regexp(current,'\d')-1);
    end
    %%%%cut off the part of the name after the first appearance of a number
    cell={atomtype_new};
    atom_type(length(atom_type)+1)=cell;
    %%%Append this new atom name to the list of atom types
end
atom_type=atom_type(2:length(atom_type));
%%%%%%%%%%%%%Done stripping number from atom name
%%%Change of unit cell parameters a and b corresponding to the new theta
%%%value

diagonal=(a^2+b^2)^(0.5);
a=diagonal*sin(theta/2);
b=diagonal*cos(theta/2);



%%%%%%%%%%%%%Setting up the different h,k,l values
l_values_temp=repmat(-1*lmax:lmax,1,(2*kmax+1)*(2*hmax+1));
```

```
k_repeat_unit=repmat(-1*kmax:kmax,(2*lmax+1),1);
k_values_temp=repmat(reshape(k_repeat_unit,1,(2*lmax+1)*(2*kmax+1)),1,(2*hmax
+1));
h_values_temp=reshape(repmat(-
1*hmax:hmax,(2*kmax+1)*(2*lmax+1),1),1,(2*hmax+1)*(2*kmax+1)*(2*lmax+1));
h_values_temp(hmax*(2*kmax+1)*(2*lmax+1)+kmax*(2*lmax+1)+lmax+1)=[];
k_values_temp(hmax*(2*kmax+1)*(2*lmax+1)+kmax*(2*lmax+1)+lmax+1)=[];
l_values_temp(hmax*(2*kmax+1)*(2*lmax+1)+kmax*(2*lmax+1)+lmax+1)=[];
h=h_values_temp;
k=k_values_temp;
l=l_values_temp;
%%%%%%%%%%%%%%%Now we have all the h,k,l values with 0,0,0 taken out

%%%%%%%%%%%%%%Determine the two theta values for each h,k,l
V_cell=a*b*c*(1-cos(alpha)^2-cos(beta)^2-
cos(gamma)^2+2*cos(alpha)*cos(beta)*cos(gamma))^0.5;
one_over_dhkl=1/V_cell.*(h.^2*b^2*c^2*sin(alpha)^2+k.^2*a^2*c^2*sin(beta)^2+l
.^2*a^2*b^2*sin(gamma)^2+2.*h.*k*a*b*c^2*(cos(alpha)*cos(beta)-
cos(gamma))+2*k.*l*a^2*b*c*(cos(beta)*cos(gamma)-
cos(alpha))+2*h.*l*a*b^2*c*(cos(alpha)*cos(gamma)-cos(beta))).^(0.5);
two_theta=2.*asin(one_over_dhkl*lambda/2);
%http://classes.uleth.ca/200401/chem4000a/Lecture10.pdf
%%%%%%%%%%%%twotheta values now determined

%%%%%%%%%%%%%%Calculate the structure factor for each reflection
n=1;
structure_factor=0;
while(n<=length(atom_type))
    %%%%read in the parameters for the scattering factor equation and give
    %%%%them names
    sf_parameters=scattering_factor_parameters(atom_type(n));
    a1=sf_parameters(1);
    b1=sf_parameters(2);
    a2=sf_parameters(3);
    b2=sf_parameters(4);
    a3=sf_parameters(5);
    b3=sf_parameters(6);
    a4=sf_parameters(7);
    b4=sf_parameters(8);
    c_=sf_parameters(9);
    %%%%%Plug the parameters into the scattering factor equation
    scattering_factor=a1.*exp(-b1.*sin(two_theta/2).^2/lambda^2)+a2.*exp(-
b2.*sin(two_theta/2).^2/lambda^2)+a3.*exp(-
b3.*sin(two_theta/2).^2/lambda^2)+a4.*exp(-
b4.*sin(two_theta/2).^2/lambda^2)+c_;%Check
    %%%%%%Account for vibration in the form of the debye-waller factor
    scattering_factor=scattering_factor.*exp(-
Biso.*sin(two_theta/2).^2/lambda^2);
    %%%%%%%Determine the contribution of the current atom to the overall
    %%%%%%%structure factor

structure_factor=structure_factor+scattering_factor.*occupancy(n).*exp(2*pi*1
i.*(h*x(n)+k*y(n)+l*z(n)));
    n=n+1;
end
%%%%%square each term in the structure factor vector
F_squared=structure_factor.*conj(structure_factor);
```

```
%%%%%Correction for preferred orientation
%%Angle between h,k,l and preferrential orientation direction
theta_pref_orient=acos((h*preferred_h + k*preferred_k +
l*preferred_l)./((h.^2+k.^2+l.^2).^0.5*(preferred_h^2+preferred_k^2+preferred
_l^2)^0.5));
if theta_pref_orient>pi/2
    theta_pref_orient=pi-theta_pref_orient;
end
F_squared=F_squared.*exp(pref*cos(2*theta_pref_orient));
%%%%%read in the experimental pattern
[experimental_two_theta,unnormalized_experimental_intensity]=textread(experim
ental_pattern,'%f %f');
%%%%%%convert the calculated two theta values to degrees for easier
%%%%%%plotting
two_theta=180*two_theta/pi;

%%%%%%Construct the calculated pxrd pattern by adding a lorentzian fraction
%%%%%%to a gaussian fraction

%%Gaussian part
c_=fwhm/(2*(2*log(2))^0.5);
n=1;
unnormalized_gauss=0;
while(n<=length(two_theta))
    unnormalized_gauss_contribution=F_squared(n).*exp(-
(experimental_two_theta-two_theta(n)).^2/(2*c_^2));
    unnormalized_gauss=unnormalized_gauss+unnormalized_gauss_contribution;
    n=n+1;
end
gauss_part=unnormalized_gauss/max(unnormalized_gauss);

%%Lorentzian part
n=1;
unnormalized_lorentz=0;
while(n<length(two_theta))
    unnormalized_lorentz_contribution=F_squared(n)./((experimental_two_theta-
two_theta(n)).^2+(0.5*fwhm)^2);

unnormalized_lorentz=unnormalized_lorentz+unnormalized_lorentz_contribution;
    n=n+1;
end
lorentz_part=unnormalized_lorentz/max(unnormalized_lorentz);
%%%%%%%Mix together the Lorentzian and Gaussian parts in the ratio
%%%%%%%specified by eta to generate the pseudo-Voigt function
calculated_intensity=eta*lorentz_part+(1-eta)*gauss_part;
%%%%%%Now put in the Lorentz-Polarization factor:
calculated_intensity=calculated_intensity.*(1+cos(experimental_two_theta*pi/1
80).^2)./(8*sin(experimental_two_theta*pi/180/2).^2.*cos(experimental_two_the
ta*pi/180/2));
calculated_intensity=calculated_intensity/max(calculated_intensity);
%%%%normalize the experimental intensity
experimental_intensity=unnormalized_experimental_intensity/max(unnormalized_e
xperimental_intensity);

%%%%%Plot the calculated pattern in red and the experimental pattern in
%%%%%blue
plot(experimental_two_theta,calculated_intensity,'Color','red');
```

```
hold on
plot(experimental_two_theta,experimental_intensity);
hold off


%%%%Generate a dummy res file to visualize coordinates
resgenerator_1(a,b,c,alpha,beta,gamma,atom_type,x,y,z,occupancy,resfilename);
```

resgenerator_1.m

```
function
y=resgenerator_1(a,b,c,alpha,beta,gamma,atomtypes,x,y,z,occupancy,resfilename
)
%%%This function takes as inputs the unit cell parameters, the atom types
%%%present (SFAC), the numbers of each atom present (UNIT), the atom list
%%%(atomtypes) as a row vector, the number corresponding to each atom type
%%%(atomnumbers) as a row vector, and the fractional coordinates x, y, and
%%%z as row vectors. It outputs a dummy res file that can be used to
%%%visualize the packing.
alpha=alpha*180/pi;beta=beta*180/pi;gamma=gamma*180/pi;
%Merge atoms that end up on the same site
x=mod(x,1);y=mod(y,1);z=mod(z,1);
n=1;
while(n<=length(x))
    m=n+1;
    while(m<=length(x))
        if x(n)==x(m) && y(n)==y(m) && z(n)==z(m)

x(m)=[];y(m)=[];z(m)=[];atomtypes(m)=[];occupancy(n)=occupancy(n)+occupancy(m
);occupancy(m)=[];
        else
            m=m+1;
        end
    end
    n=n+1;
end
occupancy=occupancy+10;
n=1;
SFAC=atomtypes(1);
while(n<=length(atomtypes))
    current=char(atomtypes(n));
    if sum(ismember(SFAC,current))>0
        n=n+1;
    else
        SFAC(length(SFAC)+1)={current};
        n=n+1;
    end
end
n=1;
atomnumbers=[];
UNIT=zeros(1,length(SFAC));
while(n<=length(atomtypes))
    current=char(atomtypes(n));
    atomnumbers(length(atomnumbers)+1)=find(ismember(SFAC,current),1);

UNIT(find(ismember(SFAC,current),1))=UNIT(find(ismember(SFAC,current),1))+1;
    n=n+1;
end

fid=fopen(resfilename,'w');
fprintf(fid,'TITL MATLAB structure\nCELL 0.71073 %3.4f %3.4f %3.4f %3.3f
%3.3f %3.3f\nZERR 1 0 0 0 0 0 0\nLATT -1\n',a,b,c,alpha,beta,gamma);
n=1;
fprintf(fid,'SFAC ');
```

```
while(n<=length(SFAC))
    fprintf(fid,'%s ',char(SFAC(n)));
    n=n+1;
end
fprintf(fid,'\n');
fprintf(fid,'UNIT ');
n=1;
while(n<=length(UNIT))
    fprintf(fid,'%d ',UNIT(n));
    n=n+1;
end
fprintf(fid,'\n');
fprintf(fid,'L.S. 10\nBOND $H\nFMAP 2\nPLAN 10\nWGHT 0.1\nFVAR 0.1 \n');
n=1;
while(n<=length(atomtypes))
    fprintf(fid,'%s %d %1.5f %1.5f %1.5f %1.5f
0.05\n',char(atomtypes(n)),atomnumbers(n),x(n),y(n),z(n),occupancy(n));
    n=n+1;
end
fprintf(fid,'HKLF 4\n\nEND');
y=5;
```

scattering_factor_parameters.m

```
function sf = scattering_factor_parameters(a)
a=char(a);
if strcmp(a,'H')
    sf=[0.493002 10.5109 0.322912 26.1257 0.140191 3.14236 0.040810 57.7997
0.003038];
elseif strcmp(a,'Li')%This is for Li(0)
    sf=[1.1282 3.9546 0.7508 1.0524 0.6175 85.3905 0.4653 168.261 0.0377];
%elseif strcmp(a,'Li')%This is for Li+
    %sf=[0.6968 4.6237 0.7888 1.9557 0.3414 0.6316 0.1563 10.0953 0.0167];
elseif strcmp(a,'C')
    sf=[2.31 20.8439 1.02 10.2075 1.5886 0.5687 0.865 51.6512 0.2156];
elseif strcmp(a,'N')
    sf=[12.2126 0.0057 3.1322 9.8933 2.0125 28.9975 1.1663 0.5826 -11.529];
elseif strcmp(a,'O')
    sf=[3.04850 13.2771 2.2868 5.7011 1.5463 0.3239 0.867 32.9089 0.2508];
elseif strcmp(a,'B')
    sf=[2.0545 23.2185 1.3326 1.021 1.0979 60.3498 0.7068 0.1403 -0.1932];
%elseif strcmp(a(b),'Na')%this is for Na+
%     sf=[3.2565 2.6671 3.93620 6.1153 1.3998 0.2001 1.0032 14.0390 0.404];
elseif strcmp(a,'Na')
    sf=[4.7626 3.285 3.1736 8.8422 1.2674 0.3136 1.1128 129.424 0.676];
elseif strcmp(a,'Si')
    sf=[6.2915 2.4386 3.0353 32.3337 1.9891 0.6785 1.541 81.6937 1.1407];
elseif strcmp(a,'P')
    sf=[6.4345 1.9067 4.1791 27.1570 1.7800 0.526 1.4908 68.1645 11.1149];
elseif strcmp(a,'S')
    sf=[6.9053 1.46790 5.2034 22.2151 1.4379 0.2536 1.5863 56.172 0.8669];
%elseif strcmp(a,'Cl')%this is for Cl-
%     sf=[18.2915 0.0066 7.2084 1.1717 6.5337 19.5424 2.3386 60.4486 -16.378];
elseif strcmp(a,'Cl')
    sf=[11.4604 0.0104 7.1964 1.16620 6.25560 18.5194 1.6455 47.7784 -
9.5574];
elseif strcmp(a,'Cr')
    sf=[10.6406 6.1038 7.3537 0.392 3.324 20.2626 1.4922 98.7399 1.1832];
elseif strcmp(a,'Mn')
    sf=[11.2819 5.3409 7.3573 0.3432 3.01930 17.8674 2.24410 83.7543
1.08960];
elseif strcmp(a,'Fe')
    sf=[11.7695 4.7611 7.3573 0.3072 3.5222 15.3535 2.3045 76.8805 1.0369];
elseif strcmp(a,'Co')
    sf=[12.2841 4.2791 7.3409 0.2784 4.0034 13.5359 2.2488 71.1692 1.0118];
elseif strcmp(a,'Ni')
    sf=[12.8376 3.8785 7.292 0.2565 4.4438 12.1763 2.38 66.3421 1.0341];
elseif strcmp(a,'Cu')
    sf=[13.338 3.5828 7.1676 0.247 5.6158 11.3966 1.6735 64.8126 1.191];
elseif strcmp(a,'Zn')
    sf=[14.0743 3.2655 7.0318 0.2333 5.1652 10.3163 2.41 58.7097 1.3041];
elseif strcmp(a,'Se')
    sf=[17.0006 2.4098 5.8196 0.2726 3.9731 15.2372 4.3543 43.8163 2.8409];
elseif strcmp(a,'Cs')
    sf=[20.3892 3.569 19.1062 0.3107 10.662 24.3879 1.4953 213.904 3.3352];
%elseif strcmp(a,'Cs')%this is for Cs+
    %sf=[20.3524 3.552 19.1278 0.3086 10.2821 23.7128 0.9615 59.4565 3.2791];
elseif strcmp(a,'Mo')
```

```
    sf=[3.7025 0.2772 17.2356 1.0958 12.8876 11.004 3.7429 61.6584 4.3875];
end
```