Study of Cell Orientation Alignment in Response to Cyclic Mechanical Stresses

Arsalan Ahmed
Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109

Abstract

A number of studies have been performed on the subject of 'Cell Orientation Alignment' caused by mechanical stresses applied to sample substrates. Cells in most solid tissues (muscles, tendons, skin) have a characteristic alignment. How does this develop? Current research indicates that when cells on a two-dimensional substrate are subject to stress, either by stretching the substrate, by subjecting them to a well-aligned flow, or by other means, they reorient and align themselves in preferred directions.

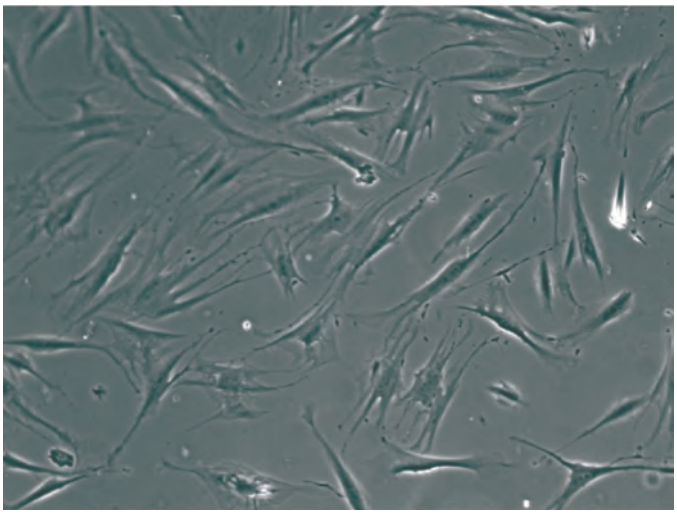

Figure 1: *Cells before stretching*

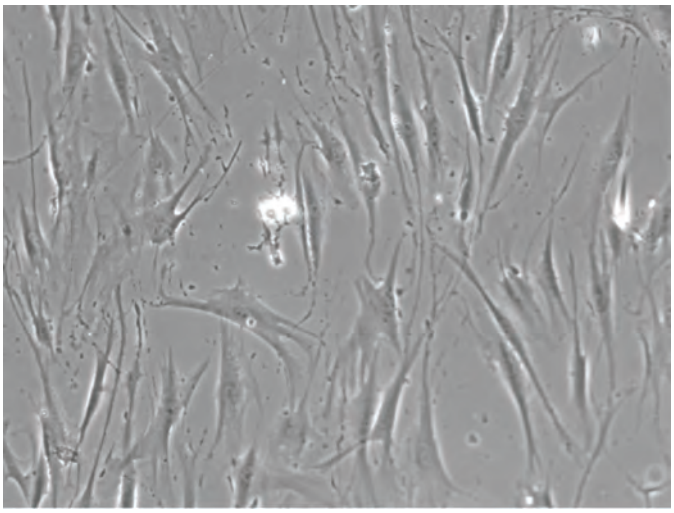

Figure 2: *Cells after stretching*

(Images 1-3 courtesy of Simon Jungbauer and Ralf Kemkemer, Max Planck Institut fuer Metallforschung, Stuttgart)

An understanding of these responses of cells to mechanical stimuli will have implications for mechanical inhibition of cancer cells and promotion of stem cell activity. If the behavior of cells can be formulated mathematically it would make it would aid in detecting cancerous cells, making it easier for medical specialists in eliminating the disease in its initial stages before it spreads around the entire body. It is well established that biological cells (i.e.: Fibroblasts and Rat Embryonic Fibroblasts (REF)) tend to align away from the stretching direction.[1] Most observations suggest that these cells tend to align in the direction of least substrate stress or in other words, they tend to align perpendicular to the direction of applied stress. The exact mechanism by which this process takes place has not been investigated thoroughly. The objective of this research is to perform experimental measurements on sample cells and to use mathematical modeling to help quantitatively describe a cell's morphological response to stretching. Specifically, the goal of this research project is to determine the relationship of cell orientation relative to the horizontal with respect to time. This was done by generating an exponential relation of cell orientation as a function of time and validating it by fitting this mathematical model to actual experimental data.

Overview

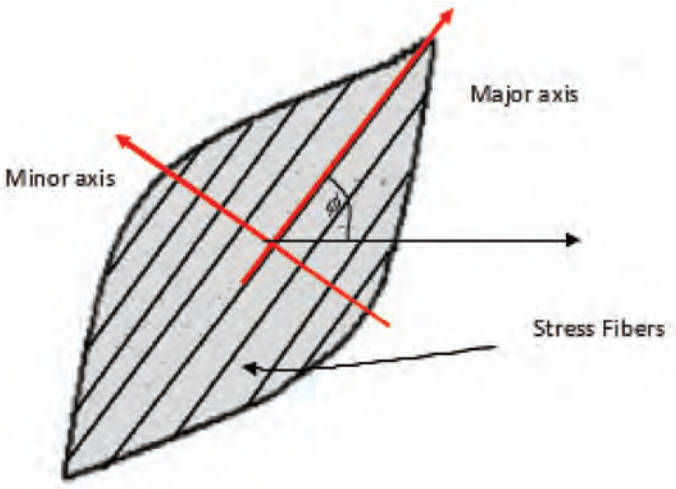

Figure 3: *A schematic model of a biological cell indicating the major & minor axes and the orientation of cell.*[1]

A simple schematic model of a cell is shown in **Figure 3**. A typical 2D cell model is made up of a chain of polymers called stress fibers arranged parallel to each other lying along the major axis of the cell. Orthogonal to the major axis is the minor axis. These axes are useful in defining the orientation of a cell. In **Figure 1**, ϕ corresponds to the cell orientation with respect to the horizontal.

To study the cell orientation of such a model we derived

a mathematical expression that represents the orientation of a cell under cyclic stretching. It is based on the idea of net polymerization of stress fibers.

The strain energy ϕ at an arbitrary orientation cos(ϕ) in a cell sample is given by equation 1:

$$U = U_0(\cos^2\phi)^2 = \frac{1}{2}EAl\varepsilon^2(\cos^2\phi)^2 \quad \text{(Eq\#1)}$$

$$U(m) = U_0(\cos^2\phi)^2 = U_0(\frac{\cos 2\phi + 1}{2})^2 = U_0(\frac{m+1}{2})^2 \quad \text{(Eq\#2)}$$

Where E is Young's modulus, A is the area; l is length, U_0 is the initial strain energy and ϕ is the strain amplitude.

The net polymerization rate = $k^F(U_0 - U(m)) - k^R$ (Eq#3)

In the above expression k^F and k^R are the Forward and Reverse reaction rates.

Equations 1-3 are the primary equations which aided us to construct a concise mathematical formula describing the cell orientation. After performing numerous intermediate operations the final expression reduces down to:

$$cos\emptyset(t) = Z_o[\frac{1 + r\,e^{\left(\frac{-2t\,t}{\tau}\right)}}{1 - r\,e^{\left(\frac{-2t\,t}{\tau}\right)}}] \quad \text{(Eq\#4)}$$

$$r - \frac{Z_o - Z_f}{Z_o + Z_f} \quad \text{(Eq\#5)}$$

Equation #4 is the exponential fitting function which describes the orientation of substrate cells as a function of time. Z_o is the initial fit value and Z_f is the final fit value of the exponential curve, t is time and τ is the time constant. r is simply a ratio of the difference of the fit values to the sum of fit values. Time constant is used to indicate how rapidly an exponential function decays. Equation #4 is the most important equation in our research study. Most of the raw data collected from the experiment was finally utilized to create a fitting function for cell orientation over a period of time.

Experimental Setup

Under a separate study of cellular activities in response to cyclic mechanical stresses, tendon fibroblasts were attached to silicone dishes with grooved surfaces and were bi-axially stretched in parallel and normal directions. The fibroblasts would then align with the microgrooves before any cyclic stresses were applied to the silicone substrate.[2] Such experiments were performed under different temperatures and cyclic frequencies.

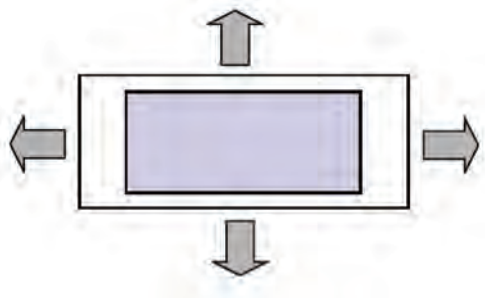

Figure 4: *illustrates a schematic model of a microgrooved silicone dish (cell sample inserted) undergoing cyclic stretching in vertical and horizontal direction.*[2]

Cells were cultured in custom designed microgrooved silicone dishes under controlled temperatures. Three temperatures chosen for such experiments were 17°C, 29°C, and 37°C. Custom built apparatus were used to cyclically stretch the substrate sample under different frequencies ranging from 1-20Hz at the three desired temperatures. Temperatures lower and higher than room temperature (~ 25°C) were chosen to determine if the cell sample behaved uniformly and normally. The silicone dish was placed at the center of the apparatus to maintain uniform stretching of cells both in the horizontal and vertical direction in 2D space (refer to **Figure 4**). The loaded specimen would then be cyclically stretched for about 500 minutes under constant temperature and frequency. Numerical data corresponding to the cells' 2-D coordinates and orientation were collected in intervals of 50 seconds. (*All image processing was done by our collaborators at Max Planck Institut fuer Metallforschung in Stuttgart Germany producing raw data for post processing).*

The recorded data consisted of the following quantities:

Quantity	Description
Pic	Designates different time sets when cell data is recorded
X center	x position of mass center of cell
Y center	Y position of mass center of cell
Minor	axis orthogonal to stress fibers
Major	axis parallel to stress fibers
Perimeter	Perimeter of individual cell sample
Phi	Cell orientation with respect to horizontal in 2D plane

Table 1: *lists all the physical quantities recorded from lab experiments.*

Since the cells are not labeled, the initial task was to sort the data and group in to each single cell. The above listed quantities along with computer codes were used to sort data for specific cells. From there on we investigated the cell orientation in further detail.

Data Analysis

Most of the data processing was done using MATLAB. Most experimental data corresponded to temperatures of 17°C, 29°C & 37°C for different frequencies ranging between 1-20 Hz. The four main type of file used for analysis were: Data input files, sorting file, and two files used for data fitting. Further details are provided below.

Input Data File: input_FL/REF_##C_##Hz.m

After the recorded data was collected in a spreadsheet, it was imported into a Matlab file which included quantities like perimeter, major & minor axes, x & y coordinates, pic number, cos(2ϕ) values, nmax and ntimcut. nmax is the number of data entries and ntimcut is the number of time sets recorded during the experiment. Data was collected into groups of different temperatures and frequency; for instance, 'FL17C_1Hz' indicates Fibroblast Cells at 17 degree Celsius operating under a driving frequency of 1Hz. The same classification was used for Rat Endothelial Fibroblast Cells (REF).

Original Sort File: sort_angle_092508_arsalan.m

The original sorting file was written by Professor Krishna Garikipati and used to produce a complete preliminary analysis of the data files. From the MATLAB data files, the vari-

ables are first separated and stored in individual arrays. The number of cell data for each time step is also noted and stored in the array ncell. Using the time step ntmin with the least number of cells, the data is then sorted according to individual cells. This is done by identifying the x and y coordinates of a particular cell at time step t, and matching them to the closest cell at time step t – 1 and t + 1. This matching process propagates until a cell is fully matched up from the start to the end of the experiment. This is done for all the cells available at time ntmin. The final data array is stored in xs (x-coordinate), ys (y-coordinate), and ϕ's (orientation angle) with individual cell data down the rows, and time steps increasing across the columns.[3]

A check is performed to ensure that the coordinates of individual cells do not exceed a certain limit value. Cells that fail this condition are flagged.

The final data of cells is arranged in three orientation groups: $0^\circ \le \phi \le 30^\circ, 30^\circ \le \phi \le 60^\circ, 60^\circ \le \phi \le 90^\circ$

The cell orientation is then averaged by dividing the orientation by the number of cells (ncell) in that group. If a specific orientation group does not contain any cells, then the analysis associated with that group is ignored. Once all the sorting has been done, the mathematical fitting function described above is implemented in the code. The user is prompted to enter three unknown quantities: 'tau, xib, xi0' for each group set followed by a least square calculation. Further details on how the three quantities were obtained are explained in following subsections. (*All image processing was done by our collaborators at Max Planck Institut fuer Metallforschung in Stuttgart Germany producing raw data for post processing).*

Fitting function #1: recfun1.m

This is a short piece of code that reads in time and experimental data. Using this data it computes the exponential function for the specific orientation group. This function is called in the second fitting function called *recfit1.m* explained below.

Fitting function #2: recfit1.m

This MATLAB file is used for computing the exponential fitting curve and determining the least square value associated between the fit curve and experimental data. Least square value indicates the accuracy of the fit. The closer the value is to 0, the stronger the fit. Boundary conditions are applied to determine tau, xib and xi0. Tau is the time constant of the fit curve, xi0 is the starting value and xib is the final value of the exponential curve. This piece of code uses the MATLAB command "lsqnonlin". 'lsqnonlin' solves nonlinear least-squares problems, including nonlinear data-fitting problems.

Computation Method

To obtain optimal values of xib, xi0 and tau, a number of specific operations are performed. Starting with the input data file, we run this file, which creates an array of all the physical quantities listed in this report followed by compiling *sort_angle_092508_arsalan.m*. We then collect the time and cosphi1, 2, 3 arrays and import them into recfit1.m. cosphi1,2 & 3 correspond to cell orientation groups of $0^\circ \le \phi \le 30^\circ, 30^\circ \le \phi \le 60^\circ, 60^\circ \le \phi \le 90^\circ$ respectively. Performing this iteration helps us in determining optimal values for xib, xi0, tau and least square values for each orientation group.

Computational Results

After performing multiple experiments and data analysis, optimal values were obtained. These results are presented in

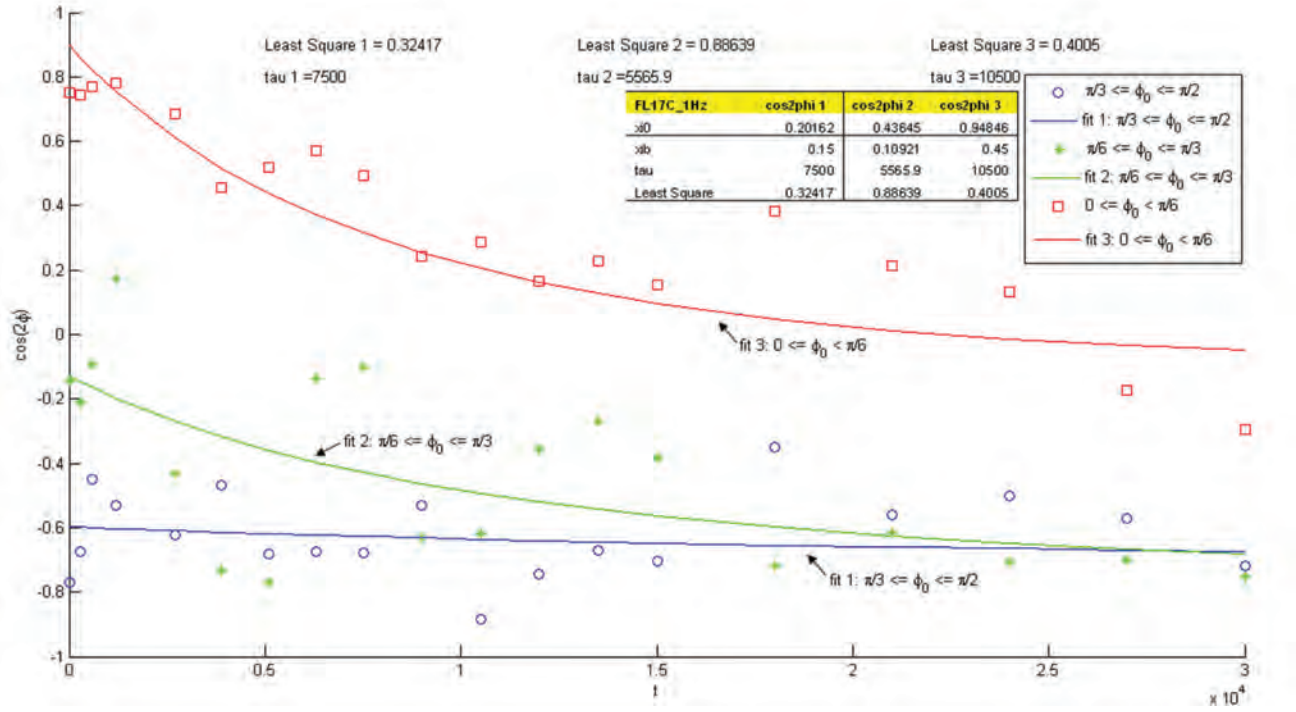

Figure 5: *Data and Curve fit plot for FL 17C 1Hz*

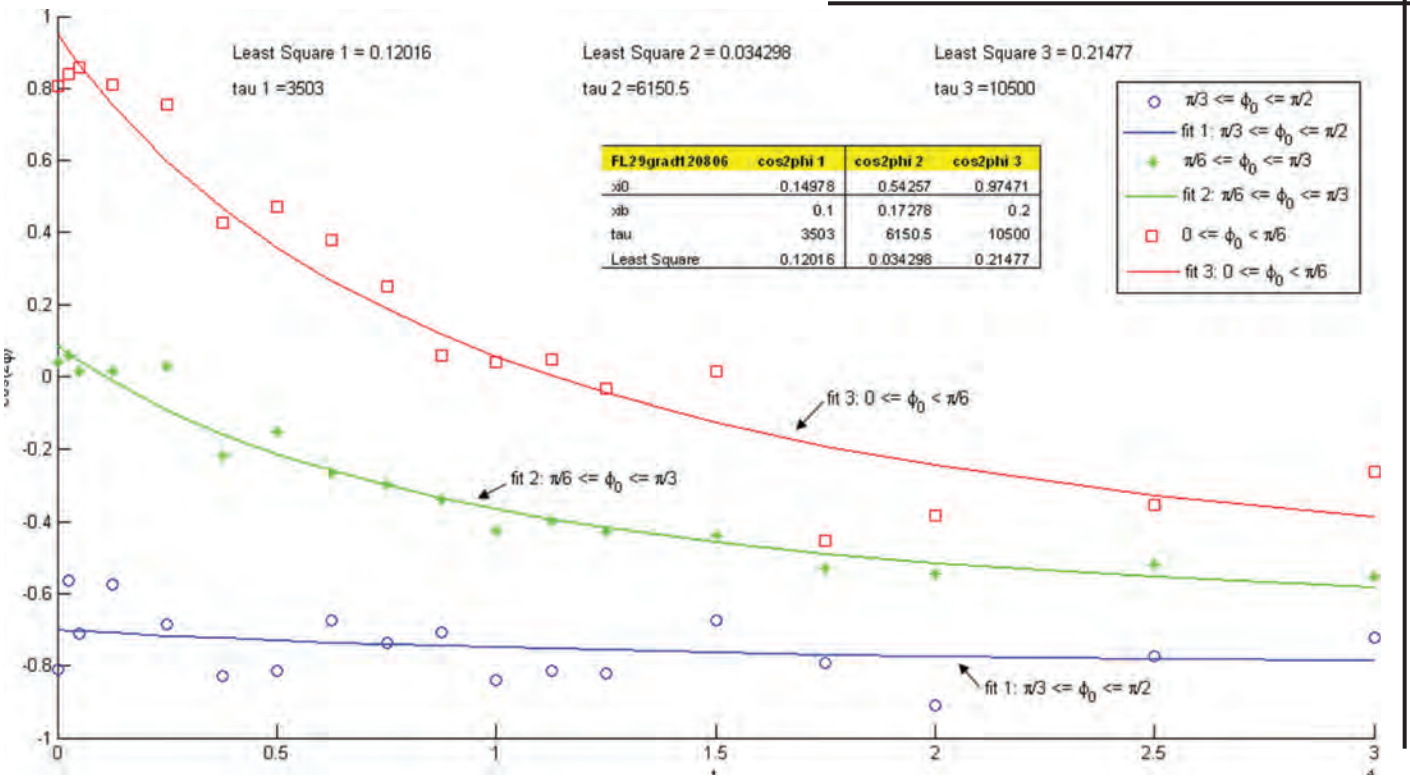

Figure 6: *Data and Curve fit plot for FL 29C grad120806*

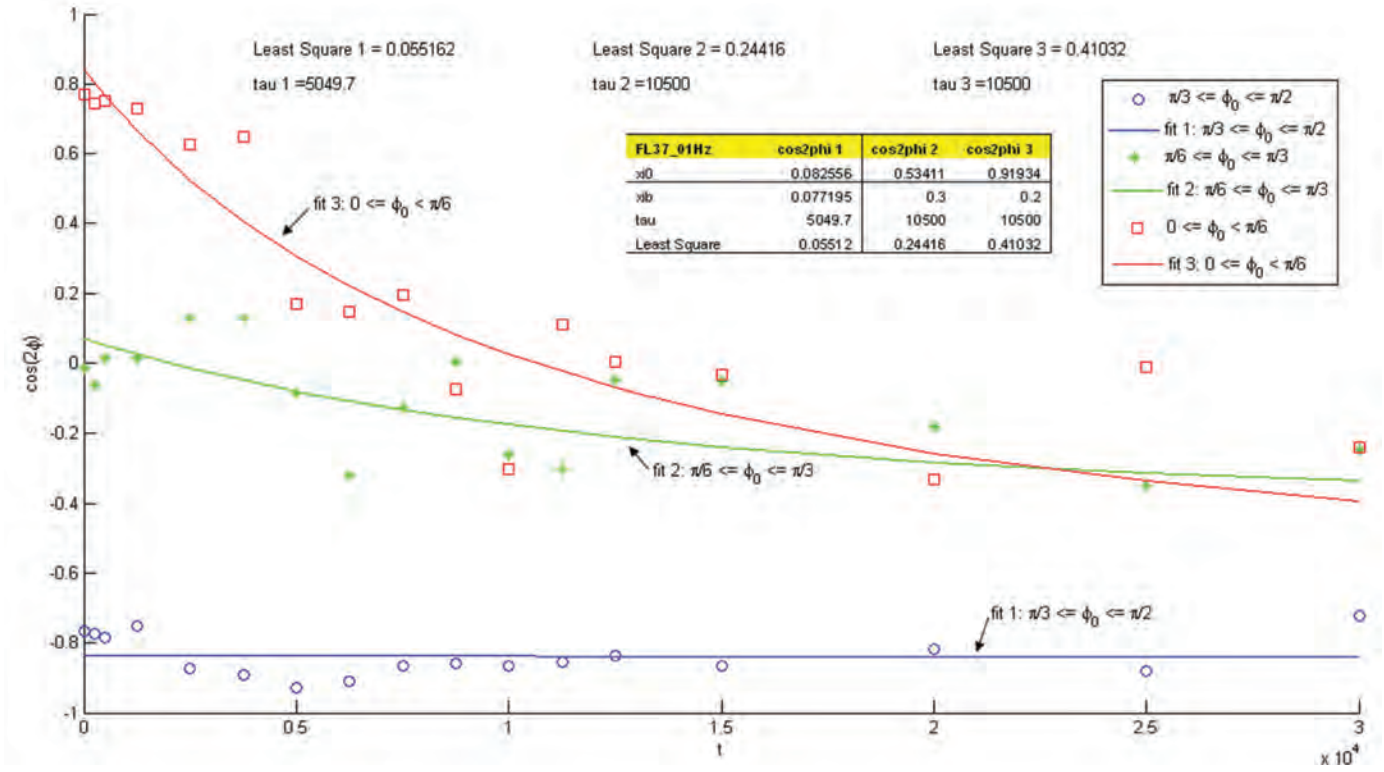

Figure 7: *Data and Curve fit plot for FL 37C 1Hz*

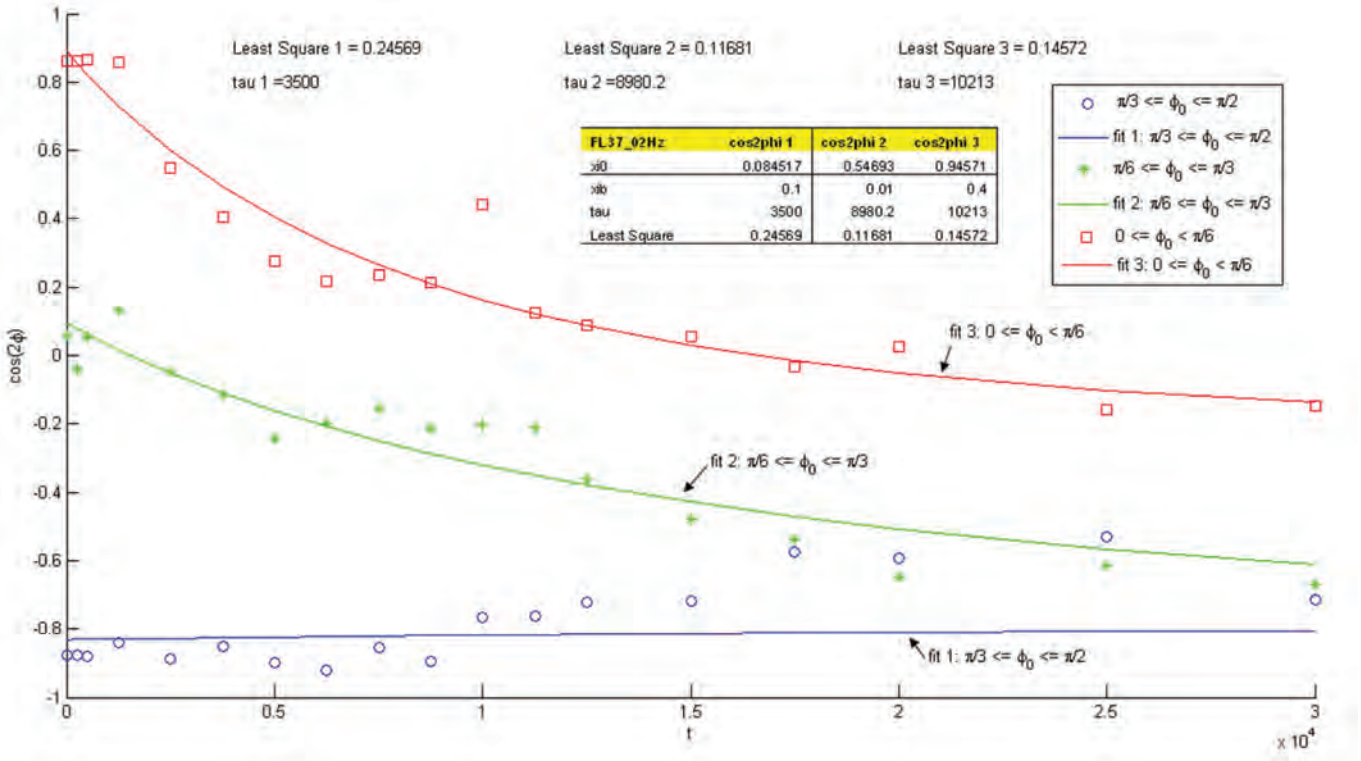

Figure 8: *Data and Curve fit plot for FL 37C 2Hz*

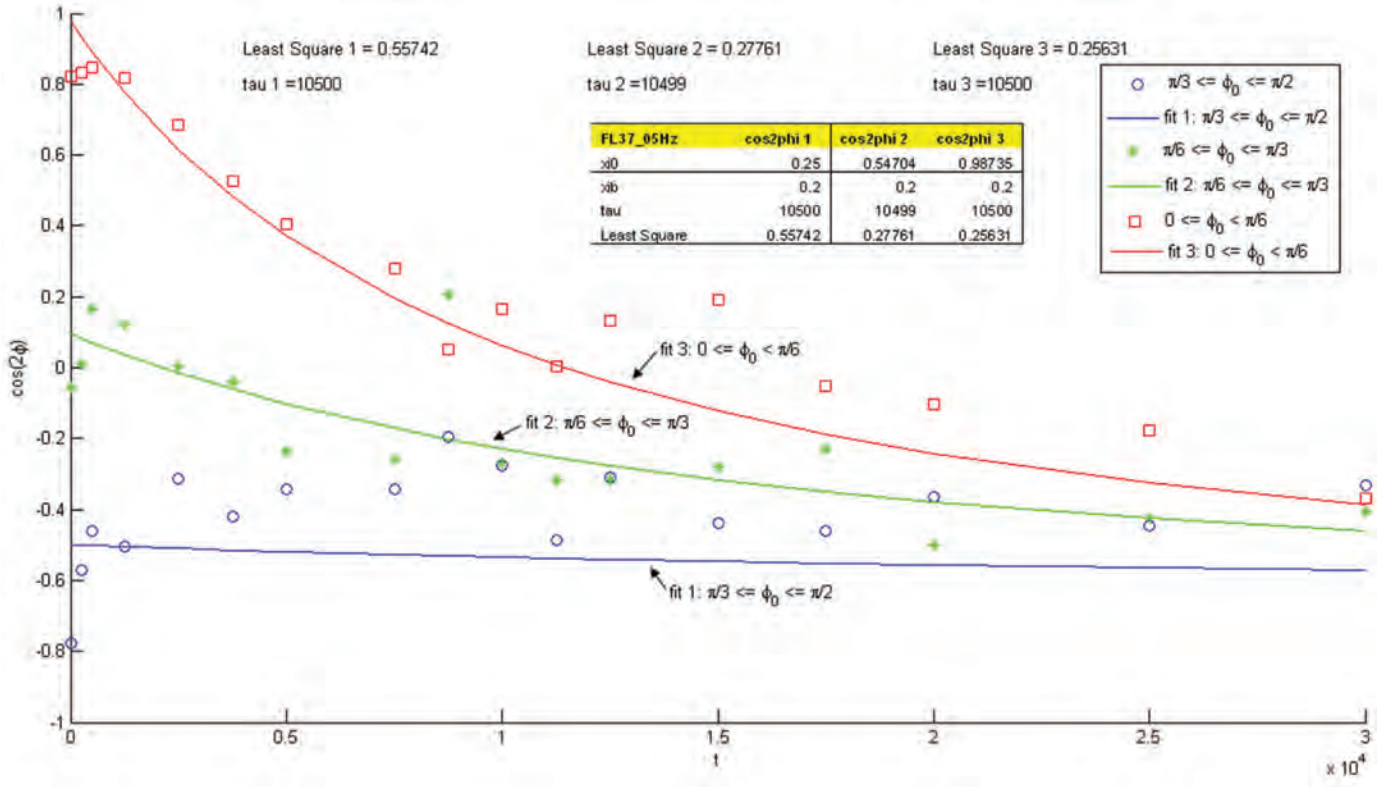

Figure 9: *Data and Curve fit plot for FL 37C 5Hz*

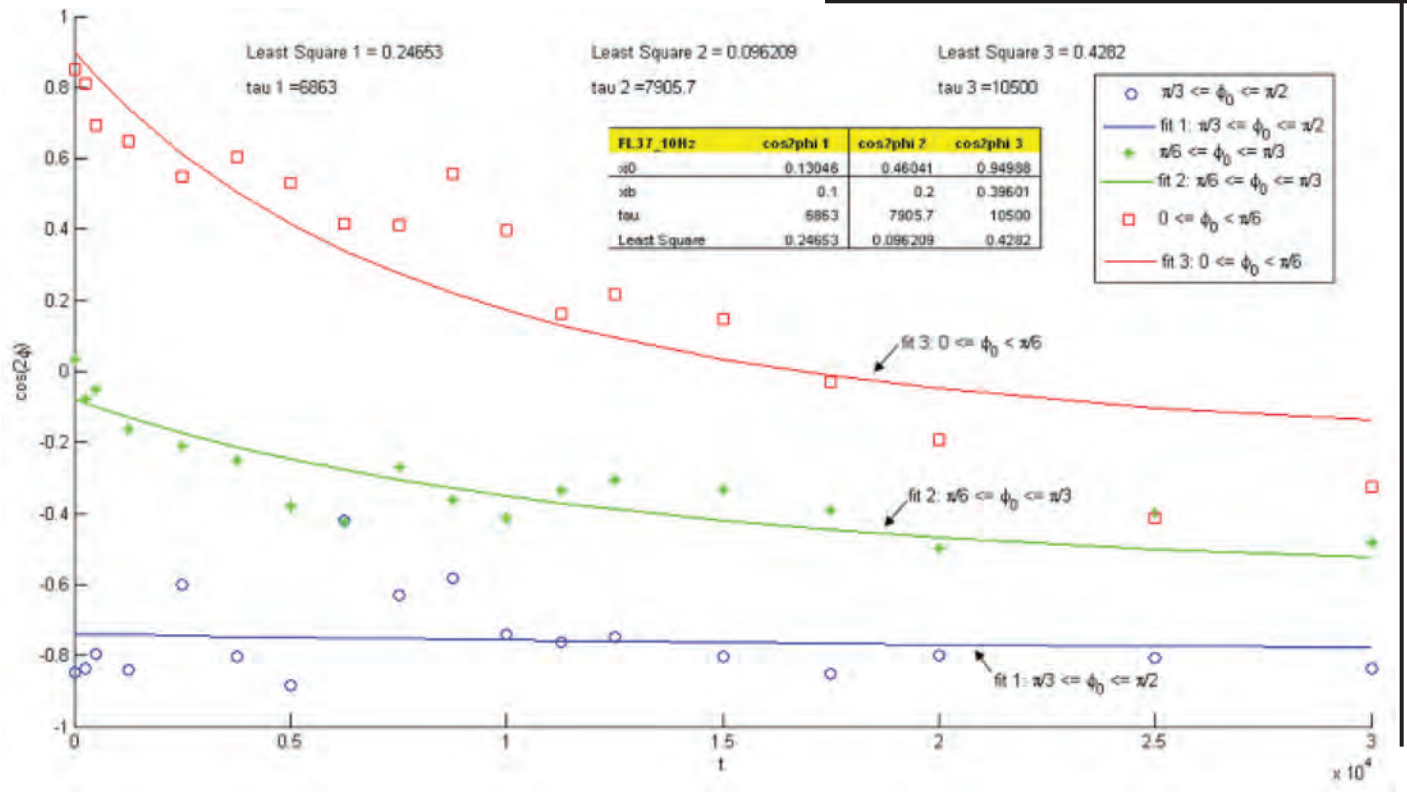

Figure 10: *Data and Curve fit plot for FL 37C 10Hz*

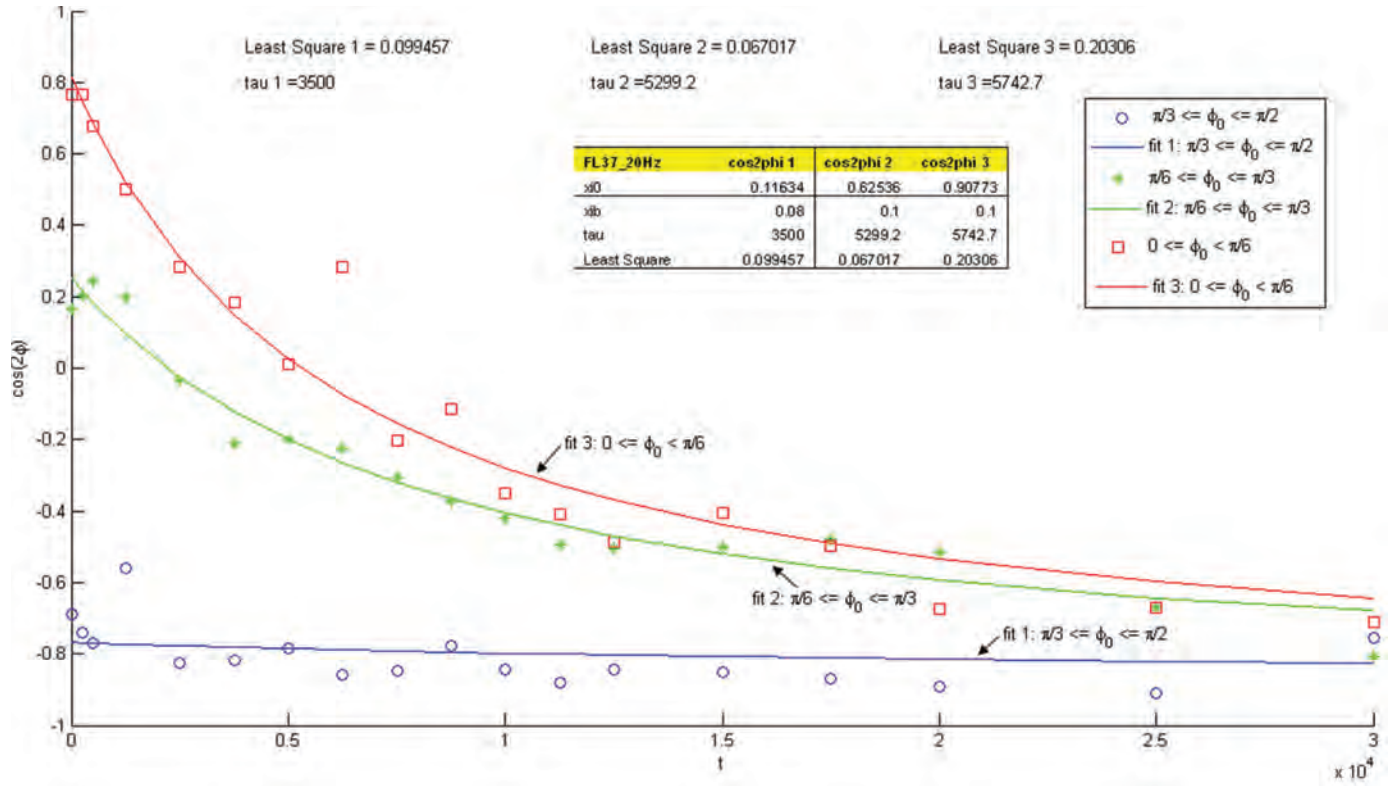

Figure 11: *Data and Curve fit plot for FL 37C 20Hz*

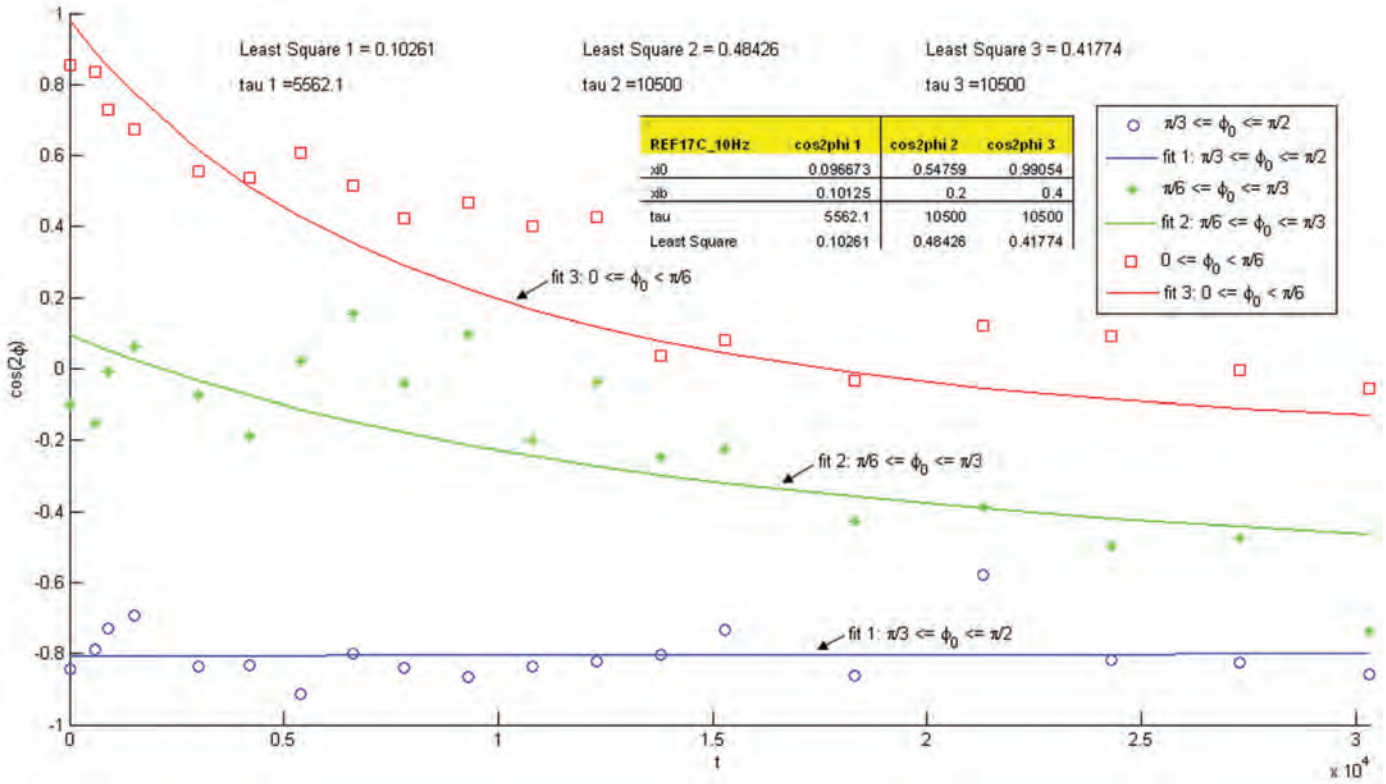

Figure 12: *Data and Curve fit plot for REF 17C 10Hz*

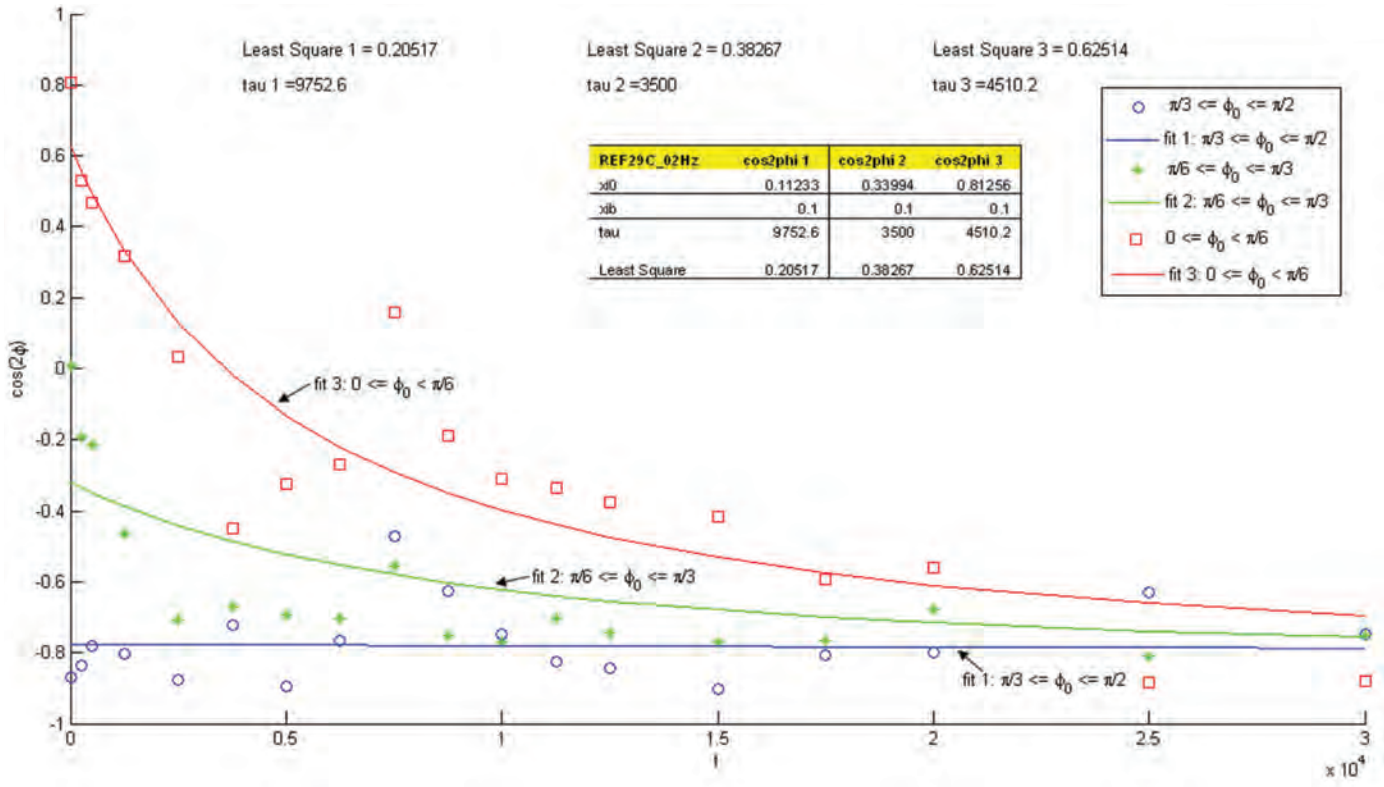

Figure 13: *Data and Curve fit plot for REF 29C 02Hz*

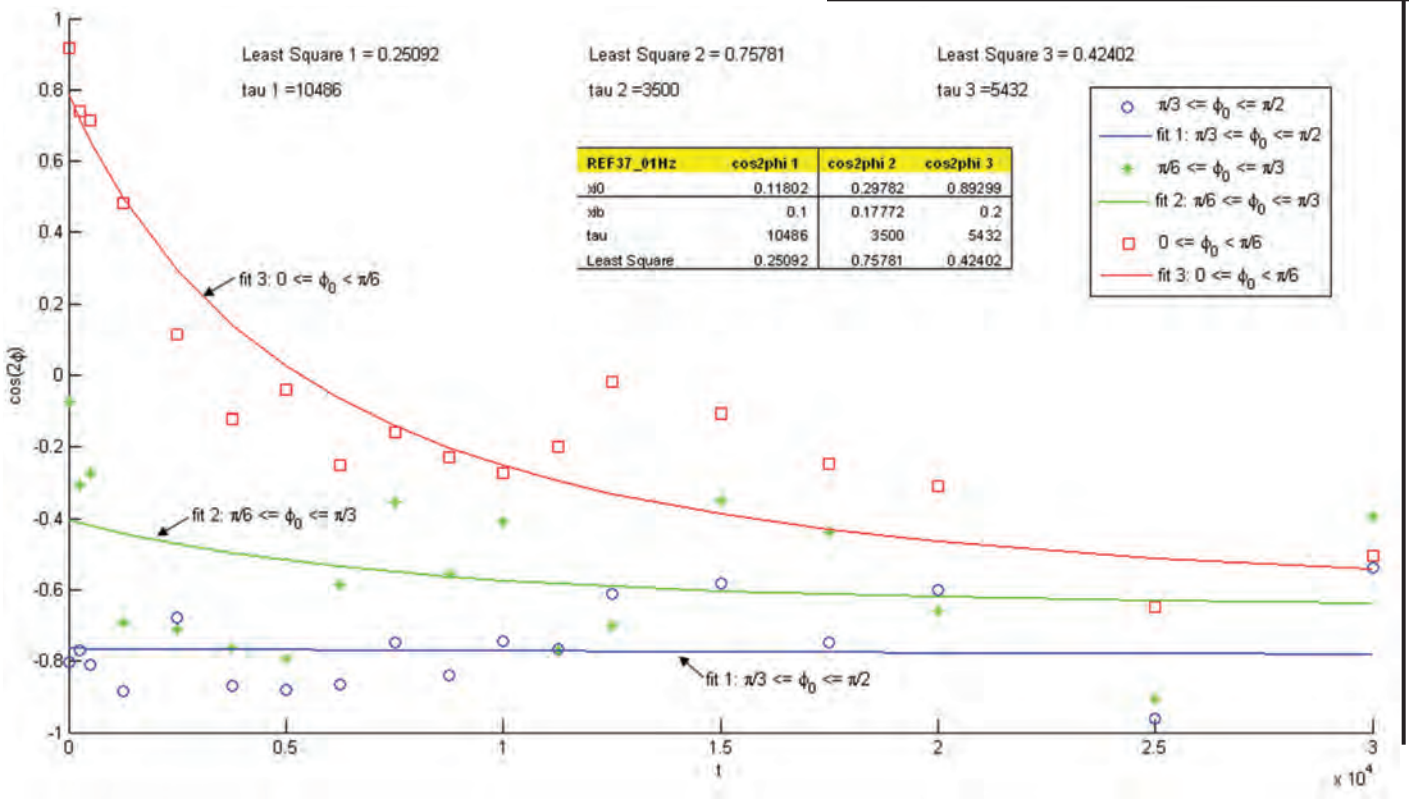

Figure 14: *Data and Curve fit plot for REF 37C 01Hz*

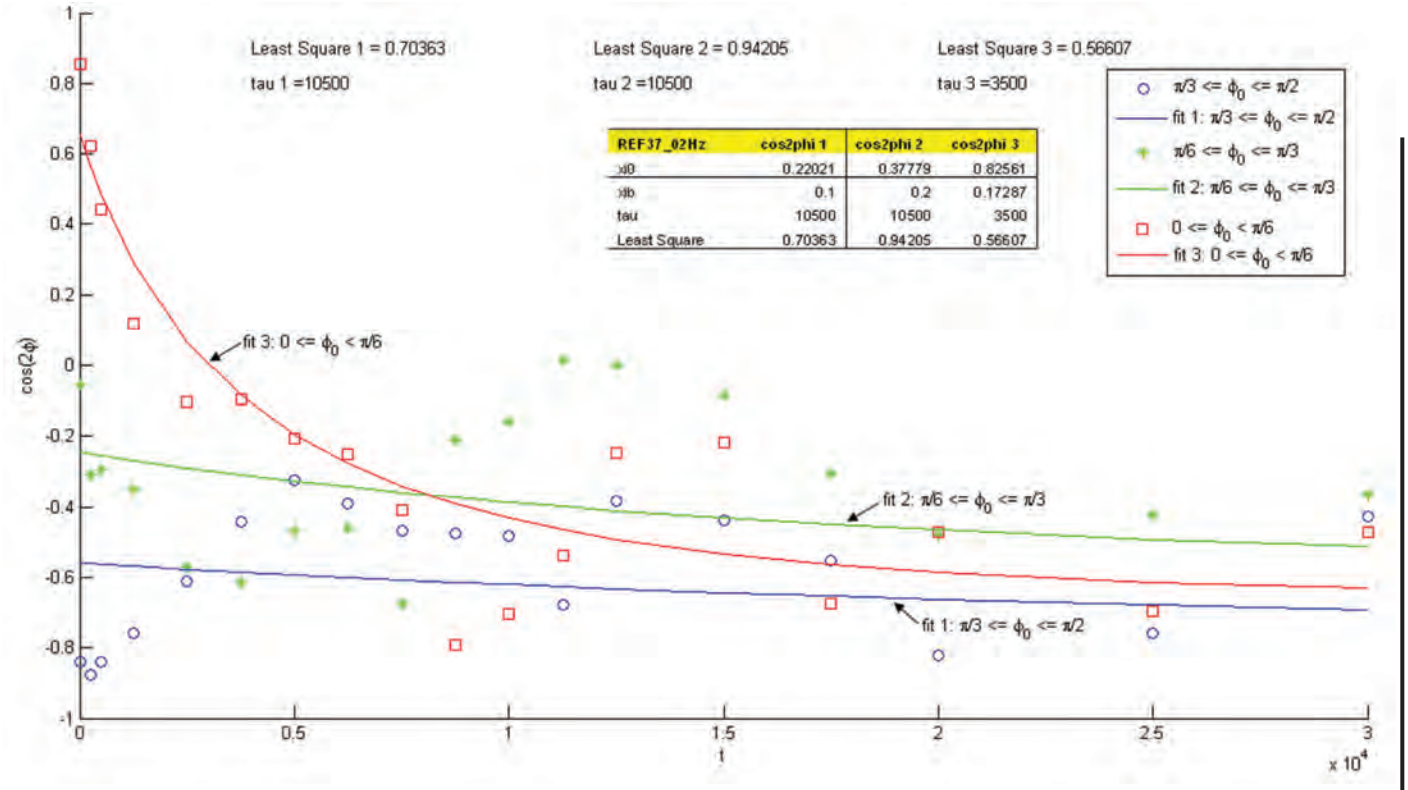

Figure 15: *Data and Curve fit plot for REF 37C 02Hz*

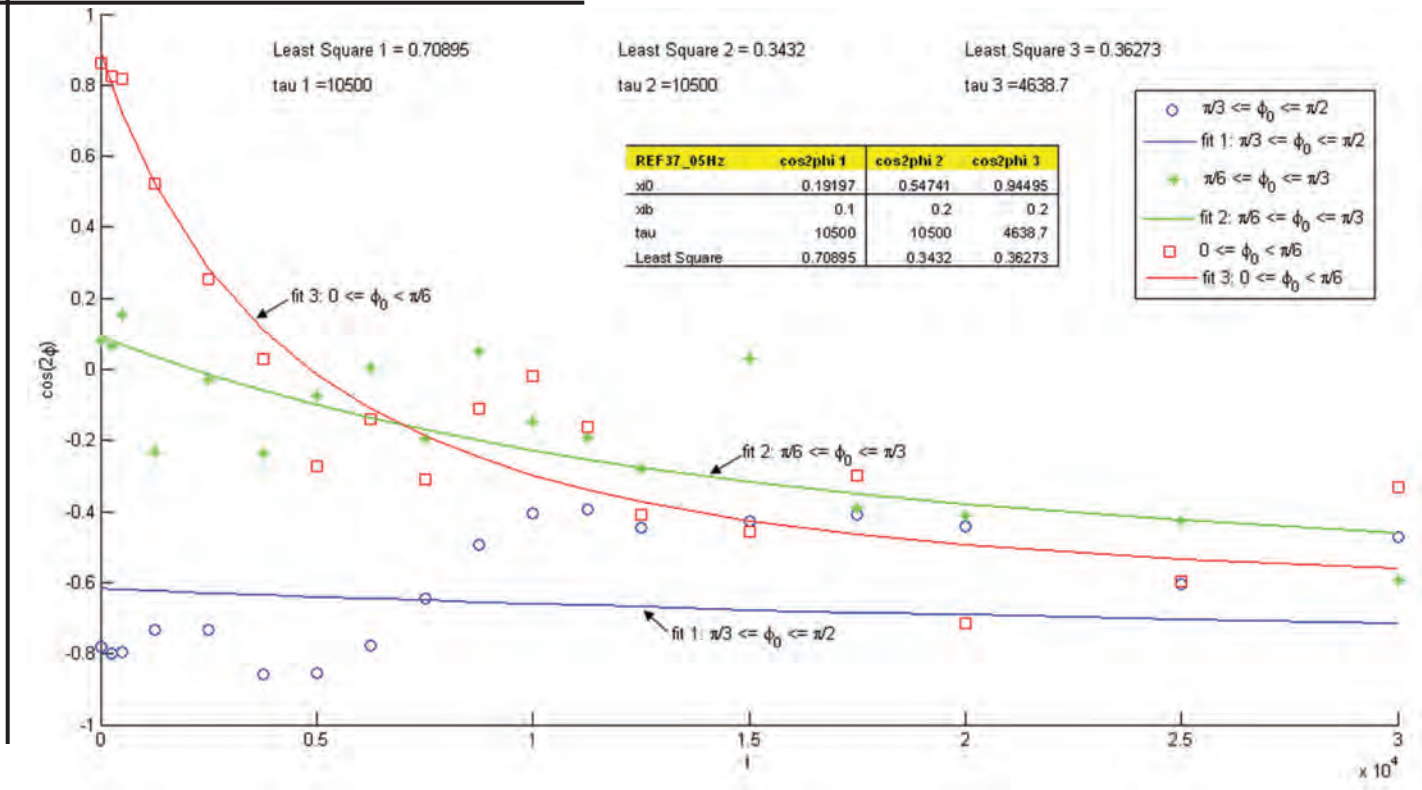

Figure 16: *Data and Curve fit plot for REF 37C 05Hz*

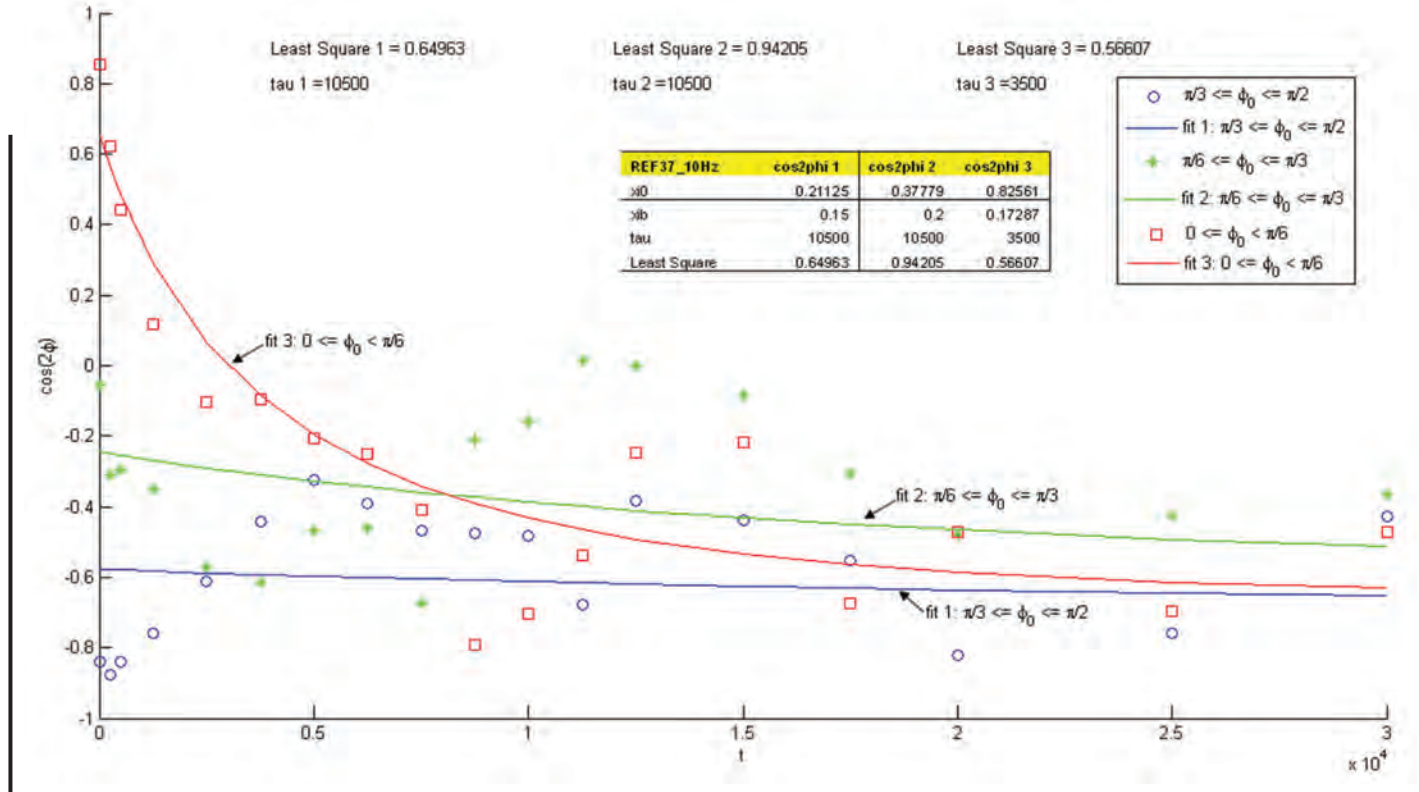

Figure 17: *Data and Curve fit plot for REF 37C 10Hz*

the figures below:

Figures 5-17 illustrate best-fit curves and the least square values for different conditions. These results show that the mathematically generated exponential curves are a good fit to the experimental data. In all cases the least square value is close to zero, indicating a strong correlation between the exponential curve and the actual test data. For different temperature and strain rate the cell orientation obeys the exponential relation with time. This suggests that our mathematical model for cell orientation is reasonably accurate. Thus we have successfully quantified the cellular orientation with respect to time. We observe that in most cases all three orientations in each experiment converge to a common value, indicating that over a long period of time all cells tend to align in the same orientation.

Conclusion

The data analysis of rates of cell-reorientation is able to provide us with useful information about the mechanism that the cell adopts to achieve the observed results. Why is this important? Restate that cells tend to align in same orientation and what this means for any applications. Revisit the idea of inhibition of cancer and promotion of stem cells. The reader need to know why this research is relevant. By characterizing their stress response what has the scientific community gained? These experimental data can be further refined in order to provide results that are close to the actual behavior of the cells and to eliminate many errors associated with our results. The experiments produced very useful results. Different temperatures and frequencies were applied to validate our hypothesis. A strong correlation is present between the experimental data and the mathematical model. In order to further excel in cell research, repeated experiments should be performed under the same conditions in order to make further comparisons. In this way discrepancies between data sets of similar conditions can be easily determined. If such discrepancies exist, this research could be further expanded to re-derive a more accurate model. More physical quantities should be included to make our model more accurate and minimize errors. A better algorithm should be constructed to collect and process raw data, speeding up the computational process.

This research was primarily performed to quantify biological studies in order to prepare accurate mathematical models for future research studies in the field of cell mechanism. A list of MATLAB codes have been provided in Appendix A which have been useful for calculating experimental data.

References

1. H. C. Wang, P. Goldschmidt-Clermont, J. Wille, F. C. Yin, "*Specificity of endothelial cell reorientation in response to cyclic mechanical stretching",* Journal of Biomechanics 34, 2001
2. H. C. Wang, G. Yang, Z. Li, W. Shen, "Fibroblast responses to cyclic mechanical stretching depends on cell orientation to the stretching direction", Journal of Biomechanics 37, 2004
3. Chun Yang Ong, "*Study of Cell Orientation Alignment in Response to Cyclic Mechanical Stresses",* ME 490: Independent Study, 2007

Appendix A: MATLAB Codes

sort_angle_092508_arsalan.m

```
%Sort data on cell positions and orientations over a range of time steps.
%First sort by time.

%time step
ntmstp = 0;
% nmax is the number of data entries in the input file
for i = 1:nmax
   if (pic(i,1) == 100000)
%time step increment
      ntmstp = ntmstp + 1;
%allots different times for different pic values. for instance
%for pic=6:(6-1)*50=250 sec
t(ntmstp) = pic(i+1,1)*1.0;  %(pic(i+1,1))*1.0;
      if (ntmstp > 1)
%measures how many cells are between successive "00's" (a counter:ncell)
         ncell(ntmstp-1) = j ;
      end
      j = 0;
   else
%when pic is not == 0 then input data values are assigned to new set of
%arrays xst,yst,phist,etc.
      j = j+1;
      xst(j,ntmstp)   = x(i,1);
      yst(j,ntmstp)   = y(i,1);
      phist(j,ntmstp) = phi(i,1);
      minorst(j, ntmstp) = minor(i,1);
      majorst(j, ntmstp) = major(i,1);
      perist(j,ntmstp) = peri(i,1);
   end
end
%repetition of ncell above
ncell(ntmstp) = j;
%Find time with minimum number of cells. ntmin is the time number with
minimum number of cells
ntmin = 1;
for i = 2:ntmstp
%simple for loop that checks for the time set that has the minimum number
of cells
%using 'ncell' and 't'
  if (ncell(i) < ncell(ntmin))
%in this for loop 'ntmin' is the array number with the minimum number of
cells.
%'tmin' being the time of the minimum cell group
     ntmin = i;
     tmin = t(ntmin);
  end
end
%Read data from this time into final sorted array
for j = 1:ncell(ntmin)
   xs(j,ntmin) = xst(j,ntmin);
   ys(j,ntmin) = yst(j,ntmin);
   phis(j,ntmin) = phist(j,ntmin);
   minors(j,ntmin) = minorst(j,ntmin);
   majors(j,ntmin) = majorst(j,ntmin);
   peris(j,ntmin) = perist(j,ntmin);
end
%Propagate the sort to times i < ntmin. Always compare data from successive
%times. The algorithm below ensures that ncell in a sorted
%array = ncell(ntmin).
for i = 1:ntmin-1
   k = ntmin-i;
   for j = 1:ncell(ntmin)
      dmin = sqrt((xs(j,k+1)-xst(1,k))^2 + (ys(j,k+1)-yst(1,k))^2);
            xs(j,k) = xst(1,k);
            ys(j,k) = yst(1,k);
            phis(j,k) = phist(1,k);
      for l = 2:ncell(k)
         dchk = sqrt((xs(j,k+1)-xst(l,k))^2 + (ys(j,k+1)-yst(l,k))^2);
               if (dchk < dmin)
            dmin = dchk;
```

```
                xs(j,k) = xst(l,k);
                ys(j,k) = yst(l,k);
            phis(j,k) = phist(l,k);
          end
      end
          dminn(j,i) = dmin;
    end
end
%Propagate the sort to times i > ntmin. Always compare data from successive
%times. The algorithm below ensures that ncell in a sorted
%array = ncell(ntmin).
for i = ntmin+1:ntmstp
    for j = 1:ncell(ntmin)
        dmin = sqrt((xs(j,i-1)-xst(1,i))^2 + (ys(j,i-1)-yst(1,i))^2);
            xs(j,i) = xst(1,i);
            ys(j,i) = yst(1,i);
            phis(j,i) = phist(1,i);
        for l = 2:ncell(i)
          dchk = sqrt((xs(j,i-1)-xst(l,i))^2 + (ys(j,i-1)-yst(l,i))^2);
              if (dchk < dmin)
            dmin = dchk;
                xs(j,i) = xst(l,i);
                ys(j,i) = yst(l,i);
            phis(j,i) = phist(l,i);
          end
      end
          dminn(j,i) = dmin;
    end
end
%Run check on maximum changes in centers of mass
for j = 1:ncell(ntmin)
    if (max(dminn(j,1:ntimcut)) > 100.0)
        nflag(j) = 1;
    else
        nflag(j) = 0;
    end
end
%Obtain cosines
for i = 1:ntmstp
    cosphi1(i) = 0.0;
    cosphi2(i) = 0.0;
    cosphi3(i) = 0.0;
end
%Group cells by initial orientation
ncell1 = 0;
ncell2 = 0;
ncell3 = 0;
for j = 1:ncell(ntmin)
    if (phis(j,1) >= 2*pi/3)  %%%% al angles changed from degrees to radians
        ncell1 = ncell1 + 1;
    elseif ((phis(j,1) < 2*pi/3) & (phis(j,1) >= 2*pi/6))
        ncell2 = ncell2 + 1;
    else
        ncell3 = ncell3 + 1;
    end
end
for j = 1:ncell(ntmin)
    for i = 1:ntmstp
        cosphi(j,i) = cos(phis(j,i)*1.0); %%%%%%%%%%%%%changed from
*pi/180 to *pi/90
        if (phis(j,1) >= 2*pi/3)
            cosphi1(i) = cosphi1(i) + cosphi(j,i);
        elseif ((phis(j,1) < 2*pi/3) & (phis(j,1) >= 2*pi/6))
                cosphi2(i) = cosphi2(i) + cosphi(j,i);
            else
                cosphi3(i) = cosphi3(i) + cosphi(j,i);
        end
    end
end
cosphi1 = (1.0/ncell1)*cosphi1;
cosphi2 = (1.0/ncell2)*cosphi2;
cosphi3 = (1.0/ncell3)*cosphi3;
```

```
%Fitting for cosphi1.......................................
tau = input('Tau for cells above 60 deg: ');
xib= input ('xib for cells above 60 deg: ');
xi0 = input('xi0 for cells above 60 deg: ');

ratio_1=(xi0 - xib)/(xi0 + xib);
for i = 1:18
    fit_1(i) = xib*(1 + ratio_1*exp(-2*xib*t(i)/tau))/(1 - ratio_1*exp(-
2*xib*t(i)/tau));
    %fit_1(i)=2*((fit_1(i)).^2) - 1;  %%%%%%%%EXTRA LINE ADDED
FOR TEST
    fit_1(i) = 2*(fit_1(i)) -1;    %%%%%%%%%%%%%%% NEW LINE
ADDED
end
%least square cosphi1
lsquare_1 = 0;
for i = 1:18
    lsquare_1 = lsquare_1 + (cosphi1(i) - fit_1(i))^2;
end
ls_1 = ['Least Square 1 = ' num2str(lsquare_1)];
tau1 = ['tau 1 =' num2str(tau)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Fitting for cosphi2.......................................
tau = input('Tau for cells between 30 and 60 deg: ');
xib= input ('xib for cells between 30 and 60 deg: ');
xi0 = input('xi0 for cells between 30 and 60 deg: ');
ratio_2=(xi0 - xib)/(xi0 + xib);
for i = 1:18
    fit_2(i) = xib*(1 + ratio_2*exp(-2*xib*t(i)/tau))/(1 - ratio_2*exp(-
2*xib*t(i)/tau));
    %fit_2(i)=2*((fit_2(i)).^2) - 1;  %%%%%%%%EXTRA LINE ADDED
FOR TEST
    fit_2(i) = 2*(fit_2(i)) -1;    %%%%%%%%%%%%%%% NEW LINE
ADDED
end
%least square cosphi2
lsquare_2 = 0;
for i = 1:18
    lsquare_2 = lsquare_2 + (cosphi2(i) - fit_2(i))^2;
end
ls_2 = ['Least Square 2 = ' num2str(lsquare_2)];
tau2 = ['tau 2 =' num2str(tau)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

%Fitting for cosphi3.......................................
tau = input('Tau for cells between 0 and 30 deg: ');
xib= input ('xib for cells between 0 and 30 deg: ');
xi0 = input('xi0 for cells between 0 and 30 deg: ');
ratio_3=(xi0 - xib)/(xi0 + xib);
for i = 1:18
    fit_3(i) = xib*(1 + ratio_3*exp(-2*xib*t(i)/tau))/(1 - ratio_3*exp(-
2*xib*t(i)/tau));
    %fit_3(i)=2*((fit_3(i)).^2) - 1;  %%%%%%%%EXTRA LINE ADDED
FOR TEST
    fit_3(i) = 2*(fit_3(i)) -1;    %%%%%%%%%%%%%%% NEW LINE
ADDED
end
%least square cosphi3
lsquare_3 = 0;
for i = 1:18
    lsquare_3 = lsquare_3 + (cosphi3(i) - fit_3(i))^2;
end
ls_3 = ['Least Square 3 = ' num2str(lsquare_3)];
tau3 = ['tau 3 =' num2str(tau)];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(2);
xlabel('t');
ylabel('cos(2\phi)');
```

```
xlim([0,t(18)]);
ylim([-1.0 1.0]); %CHANGE LIMIT FROM -1<Y<1 TO -2<Y<2
hold on;
plot(t(1:18),cosphi1(1:18),'bo',t(1:18),cosphi2(1:18),'g*',t(1:18),cosphi3(1:1
8),'rs',t(1:18),fit_1(1:18),'-b',t(1:18),fit_2(1:18),'-g',t(1:18),fit_3(1:18),'-r');
legend('\pi/3 <= \phi_0 <= \pi/2','\pi/6 <= \phi_0 <= \pi/3','0 <= \phi_0 < \
pi/6','fit 1: \pi/3 <= \phi_0 <= \pi/2','fit 2: \pi/6 <= \phi_0 <= \pi/3','fit 3: 0
<= \phi_0 < \pi/6');
text(0.4e4, 0.9, ls_1);
text(0.4e4, 0.8, tau1);
text(1.2e4, 0.9, ls_2);
text(1.2e4, 0.8, tau2);
text(2.0e4, 0.9, ls_3);
text(2.0e4, 0.8, tau3);
```

recfun1.m

```
function y=recfun1(b)
global data;

t=data(:,1);
Rexp=data(:,2);
%b(1)=xi0    b(2)=xib     b(3)=tau
Rcal=b(2)*(1 + ((b(1)-b(2))./(b(1)+b(2)))*exp(-2*b(2)*t/b(3))).(1
- ((b(1)-b(2))./(b(1)+b(2)))*exp(-2*b(2)*t/b(3)));    % the calculated
value from the model
Rcal_2theta=2*(Rcal) - 1;
%y=sum((Rcal-Rexp).^2);
y=Rcal_2theta-Rexp;
% the sum of the square of the difference between calculated value
and experimental value
```

recfit1.m

```
global data;
data=[
0            0.8613
250          0.8616
500          0.8661
1250         0.8565
2500         0.5498
3750         0.4057
5000         0.2778
6250         0.2189
7500         0.235
8750         0.2149
10000        0.4424
11250        0.1236
12500        0.0889
15000        0.0549
17500        -0.033
20000        0.0246
25000        -0.1591
30000        -0.1475];                % experimental data FOR "COS
2PHI"
t=data(:,1);
Rexp=data(:,2);
plot(t,Rexp,'ro');                    % plot the experimental data
hold on
b0=[0.875 0.17 5500];   %b0=[0.6 0.001 5000];                    %
start values for the parameters
options=optimset('MaxFunEvals',100000000);   %Max number of
function evaluations
options=optimset('MaxIter',10000000);     %Max number of
function evaluations
lb = [0.75 0.1 3500];                 %Lower bound on b
ub = [1.00 0.4 10500];                %Upper bound on b
b=lsqnonlin('recfun1',b0,lb,ub,options)       % run the lsqnonlin with
start value b0, returned parameter values stored in b
Rcal=(b(2)/1)*(1 + ((b(1)-b(2))./(b(1)+b(2)))*exp(-2*b(2)*t/b(3))).(1 -
((b(1)-b(2))./(b(1)+b(2)))*exp(-2*b(2)*t/b(3)));     % calculate the fitted value
with parameter b
bfinal=b';
%%%%%%% EXTRA LINE TO SHIFT THE CURVE
Rcal_2theta=2*(Rcal) - 1;
plot(t,Rcal_2theta,'b');                % plot the fitted value on the same
graph
xlabel('t')
ylabel('cos(2\phi)')
```