Appendix: LSTperiod source code (functions and scripts, alphabetically listed).

```
function R=f_LS_Town(t,X,W)
% Calculates the Lomb-Scargle periodogram
% Using the 'refactoring' algorithm in Townsend(2010):
% "FAST CALCULATION OF THE LOMB-SCARGLE PERIODOGRAM USING GRAPHICS
% PROCESSING UNITS"
% t, X are vectors and w is an escalar
% W is an angular frequency: W = 2*pi*f
%

R = ones(length(W),1);
for i = 1:length(W)
    XC=0; XS=0; CC=0;
    SS=0; CS=0;
    Wi = W(i);
    for j = 1:length(t)
        tj = t(j);
        Xj = X(j);
        XC = XC + Xj*cos(Wi*tj);
        XS = XS + Xj*sin(Wi*tj);
        CC = CC + (cos(Wi*tj))^2;
        SS = SS + (sin(Wi*tj))^2;
        CS = CS + sin(Wi*tj)*cos(Wi*tj);
    end
    Tau = (1/(2*Wi))*atan((2*CS)/(CC-SS));   % Calculates tau
    cTau = cos(Wi*Tau);
    sTau = sin(Wi*Tau);

    R1 = (cTau*XC+sTau*XS)^2/(cTau^2*CC+2*cTau*sTau*CS+sTau^2*SS);
    R2 = (cTau*XS-sTau*XC)^2/(cTau^2*SS-2*cTau*sTau*CS+sTau^2*CC);
    R(i)= 0.5*(R1+R2); % Here is the L-S Periodogram
end

% Normalizing by "bandwith": making each estimated 'spectrum area' = 1
R = R/abs(trapz(((W).^(-1)),R));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function residuals = findResids(t, y, X, T)
%FINDSTATS Outputs a vector with some statistics for the residuos of the model:
%               '1' - error
%               '2' - absolute error
%               '3' - error vs model
%               '4' - scale-location
%               '5' - histfit
%               '6' - QQ: y1 vs N
%               '7' - QQ: y1 vs y2


a = X(1); b = X(2);
residuals = zeros(1,7);
yi = sineInterp(t, X, T);
e = y - yi;

%----------------- 1-error ----------------------
Amp = sqrt(a^2+b^2);
residuals(1,1) = Amp;
%----------------- 2-absolute error-----------------------------
phaseR = asin(b/Amp);
residuals(1,2) = phaseR;
%----------------- 3-error vs model -----------------------------
phaseD = asind(b/Amp);
residuals(1,3) = phaseD;
%----------------- 4-scale-location ----------------------
mse = mean(e.^2);
residuals(1,4) = mse;
%----------------- 5-normalised mean squared error (nmse) -------------
nmse = mean(e.^2)/var(y);
residuals(1,5) = nmse;
%----------------- 6-root mean squared error (rmse) -------------
rmse = sqrt(mean(e.^2));
```

```
residuals(1,6) = rmse;
%----------------- 7-normalised root mean squared error (nrmse) -------------
nrmse = sqrt(mean(e.^2)/var(y));
residuals(1,7) = nrmse;

% samplStats = [Amp, phaseR, phaseD, mse, nmse, rmse, nrmse];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function samplStats = findStats(t, y, X, T)
%FINDSTATS Outputs a vector with some statistics for the model:
%               '1' - amplitude
%               '2' - phase (radians)
%               '3' - phase (degrees)
%               '4' - mean squared error (mse)
%               '5' - normalised mean squared error (nmse)
%               '6' - root mean squared error (rmse)
%               '7' - normalised root mean squared error (nrmse)
%               '8' - mean absolute error (mae)
%               '9' - mean  absolute relative error  (mare)
%               '10' - coefficient of correlation (r)
%               '11' - coefficient of determination (d)
%               '12' - coefficient of efficiency (e)
%               '13' - maximum absolute error (maxAbsEr)
%               '14' - maximum absolute relative error (maxAbsRelEr)

a = X(1); b = X(2);
samplStats = zeros(1,14);
yi = sineInterp(t, X, T);
e = y - yi;

%----------------- 1-amplitude ----------------------
Amp = sqrt(a^2+b^2);
samplStats(1,1) = Amp;
%----------------- 2-phaseR-----------------------------
phaseR = atan2(b, a);         % Phase in radians
samplStats(1,2) = phaseR;
%----------------- 3-phaseD -----------------------------
phaseD = atan2d(b, a);        % Phase in degrees
samplStats(1,3) = phaseD;
%----------------- 4-mean squared error (mse) ----------------------
mse = mean(e.^2);
samplStats(1,4) = mse;
%----------------- 5-normalised mean squared error (nmse) -------------
nmse = mean(e.^2)/var(y);
samplStats(1,5) = nmse;
%----------------- 6-root mean squared error (rmse) -------------
rmse = sqrt(mean(e.^2));
samplStats(1,6) = rmse;
%----------------- 7-normalised root mean squared error (nrmse) -------------
nrmse = sqrt(mean(e.^2)/var(y));
samplStats(1,7) = nrmse;
%----------------- 8-mean absolute error (mae) -------------
mae = mean(abs(e));
samplStats(1,8) = mae;
%----------------- 9-mean absolute relative error  (mare) -------------
mare = mean((abs(e./y)));
samplStats(1,9) = mare;
%----------------- 10-coefficient of correlation (r) ----------------
cf = corrcoef(y,yi);
r = cf(1,2);
samplStats(1,10) = r;
%----------------- 11-coefficient of determination (d) -------------
d = r^2;
samplStats(1,11) = d;
%----------------- 12-coefficient of efficiency (e) -------------
e = 1 - sum(e.^2)/sum((y - mean(y)).^2);
samplStats(1,12) = e;
%----------------- 13-maximum absolute error (maxAbsEr) -------------
maxAbsEr = max(abs(e));
samplStats(1,13) = maxAbsEr;
%----------------- 14-maximum absolute relative error (maxAbsRelEr) -------
maxAbsRelEr = max(abs(e./y));
samplStats(1,14) = maxAbsRelEr;
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function X = fitFreq(t, y, Ts)
%FITFREQ Fits y(t)=a*sin(2*pi*t/Ts)+b*cos(2*pi*t/Ts) to the data series
%Calculates linear regression for the model --> Amp*sin(2*pi*t/Ts+theta)
%[a, b] = X = fitFreq(t, y, Ts)
%Uses simple linear inversion: X = A\y,

%*********** t and y should be BOTH COLUMN vectors! ***********
% t = t';
% y = y';
%************************************************************
wt = (2*pi*t)/Ts;

A = [cos(wt), sin(wt)];
X = A\y;
end

function names = getFilesNames()
% Get the all ".dat" files names from atual directory

list=dir(fullfile('*.dat'));
for k=1:length(list)
[a, names{k},c]=fileparts(list(k,1).name);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function fittedData = getFitFreqData(names, T)
%GETFITFREQ Fits a period T to the whole data set


%----------------- Building WaitBar -------------------------
hf=figure('units','pixels','position',[200 200 540 100],...
       'name',' Wait. Work in progress... ',...
       'menubar','none','numbertitle','off','resize','off');
movegui(hf,'center')          % Move the GUI to the center of the screen
ax1=axes('Units','pix','Position',[20 40 500 20]);
set(ax1,'Xtick',[],'Ytick',[],'Xlim',[0 1000]);
box on;axes(ax1)
%------------------------------------------------------------

fittedData = zeros(length(names), 14);
for k = 1:length(names)
    D = load([names{k} '.dat']);
    disp([' Wait. Reading data file ' names{k} '.dat and'])
    disp(['    calculating regression parameters for T = ' num2str(T) '...'])
    t = D(:,1);
    y = D(:,2);
    X = fitFreq(t, y, T);
    fittedData(k,:) = findStats(t, y, X, T);

    %----------------- Updating WaitBar -------------------------
    cla
      rectangle('Position',[0,0,1001-(round(1000*k/length(names))),20],'FaceColor','r');
    text(482,10,[num2str(100-round(100*k/length(names))),'%']);
    pause(0.1)
    %------------------------------------------------------------

end
disp(' ')

%----------------- Closing Waitbar -------------------------
close(hf)
%-----------------------------------------------------------
End

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function guiData = getGuiData3(names, par)
% Build vectors W(i) & P's(i) from: data files & user parameters
% (1) W --> f_min:step:f_Max
```

```
% (2) Read data files one-by-one
% (3) Calculates Pw calling f_LS(t,x,w)


W = 2*pi./linspace(par(2), par(1), par(3))';

Pw = ones(length(W), length(names));

%----------------- Building WaitBar -------------------------
hf=figure('units','pixels','position',[200 200 540 100],...
       'name',' Wait. Work in progress... ',...
       'menubar','none','numbertitle','off','resize','off');
movegui(hf,'center')          %Move the GUI to the center of the screen
ax1=axes('Units','pix','Position',[20 40 500 20]);
set(ax1,'Xtick',[],'Ytick',[],'Xlim',[0 1000]);
box on;axes(ax1)
%------------------------------------------------------------

for k = 1:length(names)
    D = load([names{k} '.dat']);
    disp([' Wait. Reading data file ' names{k} '.dat ' ...
        'and calculating spectrum...'])
    t = D(:,1);
    Y = D(:,2);
    Pw(:,k) = f_LS_Town(t,Y,W);

    %----------------- Updating WaitBar -------------------------
    cla
      rectangle('Position',[0,0,1001-(round(1000*k/length(names))),20],'FaceColor','r');
    text(482,10,[num2str(100-round(100*k/length(names))),'%']);
    pause(0.1)
    %------------------------------------------------------------

end

disp(' ')
disp(' Calculating combined spectra...')
POR = ones(length(W),1); PAND = ones(length(W),1);
for k = 1:length(W)
    POR(k) = sum(Pw(k,:));
    PAND(k) = prod(Pw(k,:));
end

POR = POR/abs(trapz(((W).^(-1)),POR));
PAND = PAND/abs(trapz(((W).^(-1)),PAND));


%----------------- Closing Waitbar -------------------------
close(hf)
%------------------------------------------------------------

% exporting as "periodicities"
guiData = [2*pi./W POR PAND Pw];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Resids = getResiduals(names, T)
%GETFITFREQ Fits a period T to the whole data set


%----------------- Building WaitBar -------------------------
hf=figure('units','pixels','position',[200 200 540 100],...
       'name',' Wait. Work in progress... ',...
       'menubar','none','numbertitle','off','resize','off');
movegui(hf,'center')          % Move the GUI to the center of the screen
ax1=axes('Units','pix','Position',[20 40 500 20]);
set(ax1,'Xtick',[],'Ytick',[],'Xlim',[0 1000]);
box on;axes(ax1)
%------------------------------------------------------------
Resids = cell(length(names), 1);

for k = 1:length(names)
    D = load([names{k} '.dat']);
    disp([' Wait. Reading data file ' names{k} '.dat and'])
    disp(['    calculating residuals statistics for T = ' num2str(T) '...'])
```

```
    t = D(:,1);
    y = D(:,2);
    X = fitFreq(t, y, T);
    yi = sineInterp(t, X, T);
    e = y - yi;
    Resids{k} = [t y yi e];

    %----------------- Updating WaitBar -------------------------
    cla
       rectangle('Position',[0,0,1001-(round(1000*k/length(names))),20],'FaceColor','r');
    text(482,10,[num2str(100-round(100*k/length(names))),'%']);
    pause(0.1)
    %------------------------------------------------------------

end
disp(' ')

%----------------- Closing Waitbar -------------------------
close(hf)
%------------------------------------------------------------
End

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = guiFitFreq_v06(varargin)
% GUIFITFREQ_V06 MATLAB code for guiFitFreq_v06.fig
%      GUIFITFREQ_V06, by itself, creates a new GUIFITFREQ_V06 or raises the existing
%      singleton*.
%
%      H = GUIFITFREQ_V06 returns the handle to a new GUIFITFREQ_V06 or the handle to
%      the existing singleton*.
%
%      GUIFITFREQ_V06('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUIFITFREQ_V06.M with the given input arguments.
%
%      GUIFITFREQ_V06('Property','Value',...) creates a new GUIFITFREQ_V06 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before guiFitFreq_v06_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to guiFitFreq_v06_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help guiFitFreq_v06

% Last Modified by GUIDE v2.5 05-Jun-2017 19:19:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @guiFitFreq_v06_OpeningFcn, ...
                   'gui_OutputFcn',  @guiFitFreq_v06_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

%--------------------------------------------------------------------%
%               *** Variables Storage ***                            %
%--------------------------------------------------------------------%
%     Variable |  'property' - GUI_component / scope                  %
%--------------------------------------------------------------------%
%  filesNames  |           global                                     %
```

```
%     names2   |           global                                  %
%     T_Fit    |           global                                  %
%  T_Fit_Table |           global                                  %
%--------------------------------------------------------------------%



% --- Executes just before guiFitFreq_v06 is made visible.
function guiFitFreq_v06_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to guiFitFreq_v06 (see VARARGIN)


% Choose default command line output for guiFitFreq_v06
handles.output = hObject;

axes1_CreateFcn(hObject, eventdata, handles)

global T_Fit T_Fit_Table filesNames resids
T_Fit_Table = T_Fit;
handles.T_Fit = T_Fit_Table;
resids = getResiduals(filesNames,T_Fit_Table);
handles.statMetds = {'Amplitude', 'Amplitude^2', 'Phase_rad', 'Phase_deg',
'Amplitude_&_Phase',...
    'MSE_mean_squared_error', 'NMSE_normalized_mean_squared_error',...
    'RMSE_root_mean_squared_error', 'NRMSE_norm_root_mean_squared_error',...
    'MAE_mean_absolute_error', 'MARE_mean_abs_relative_error',...
    'R_coefficient_of_correlation', 'D_coefficient_of_determination',...
    'E_coefficient_of_efficiency', 'Max-AE_maximum_absolute_error',...
    'Max-ARE_max_abs_relative_error'};

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guiFitFreq_v06 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = guiFitFreq_v06_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global T_Fit_Table myTable
disp(' ')
disp ([' Creating table with goodness-of-fit for T = ' num2str(T_Fit_Table) ' ...'])

myTable = get(handles.figure1,'UserData');
LST_Table_v01(handles)


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

close(handles.figure1)
```

```
% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu2

% Determine the selected data set.
val = get(hObject,'Value');

global filesNames T_Fit_Table resdMetd
n_files = length(filesNames);

fittedData = get(handles.figure1,'UserData');
data = fittedData;

handles.namesStatMetds = {'Amplitude', 'Amplitude^2', 'Phase - rad', 'Phase - deg',...
    'Phase Histogram', 'Amplitude & Phase',...
    'MSE - mean squared error', 'NMSE - normalized mean squared error',...
    'RMSE - root mean squared error', 'NRMSE - normalized root mean squared error',...
    'MAE - mean absolute error', 'MARE - mean absolute relative error',...
    'R - coefficient of correlation', 'D - coefficient of determination',...
    'E - coefficient of efficiency', 'MaxAE - maximum absolute error',...
    'MaxARE - maximum absolute relative error'};
resdMetd = handles.namesStatMetds{val};
handles.text1 = [resdMetd,'  |  T = ',num2str(T_Fit_Table,'%0.6g')];

switch val
    case 1   % --- Amplitude
        hb = bar(1:n_files,data(:,1)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 2   % --- Amplitude^2
        hb = bar(1:n_files,(data(:,1).^2)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 3   % --- Phase - radians
        hb = bar(1:n_files,data(:,2)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 4   % --- Phase - degrees
        hb = bar(1:n_files,data(:,3)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 5   % --- Phase Histogram (rose)
        ph = data(:,2)';
        hl = rose(ph);
%         x = get(hl,'Xdata');
%         y = get(hl,'Ydata');
%         g=patch(x,y,'b');
        set(hl,'LineWidth',1.5)
        title(handles.text1,'fontsize',12,'FontWeight','bold')
    case 6   % --- Amplitude & Phase
        amp = data(:,1)';
        ph = data(:,2)';
        [x, y] = pol2cart(ph, amp);
```

```
        hl = compass(x, y);
        set(hl,'LineWidth',1.5)
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        for i = 1:n_files
            if x(i)>=0

text(x(i),y(i),filesNames{i},'HorizontalAlignment','left','Color','m','FontSize',9)
            else

text(x(i),y(i),filesNames{i},'HorizontalAlignment','right','Color','m','FontSize',9)
            end
        end
    case 7   % --- Mean squared error (mse)
        hb = bar(1:n_files,data(:,4)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 8   % --- Normalised mean squared error (nmse)
        hb = bar(1:n_files,data(:,5)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 9   % --- Root mean squared error (rmse)
        hb = bar(1:n_files,data(:,6)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 10   % --- Normalised root mean squared error (nrmse)
        hb = bar(1:n_files,data(:,7)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 11   % --- Mean absolute error (mae)
        hb = bar(1:n_files,data(:,8)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 12   % --- Mean  absolute relative error  (mare)
        hb = bar(1:n_files,data(:,9)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 13   % --- Coefficient of correlation (r)
        hb = bar(1:n_files,data(:,10)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 14   % --- Coefficient of determination (d)
        hb = bar(1:n_files,data(:,11)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
```

```
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 15   % --- Coefficient of efficiency (e)
        hb = bar(1:n_files,data(:,12)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 16   % --- Maximum absolute error (maxAbsEr)
        hb = bar(1:n_files,data(:,13)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
    case 17   % --- Maximum absolute relative error (maxAbsRelEr)
        hb = bar(1:n_files,data(:,14)',0.6,'g','EdgeColor',[1,0.5,0.5]);
        title(handles.text1,'fontsize',12,'FontWeight','bold')
        grid on
        xlim([0  n_files+1]);
        hbl = get(hb(1),'BaseLine');
        set(hbl,'Color','red','LineStyle',':')
        set(gca,'XTick',1:n_files,'XTickLabel',filesNames)

end

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function text1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

global T_Fit_Table

set(hObject,'String',[handles.statMetds{1}, '  |  T = ', num2str(T_Fit_Table,'%0.6g')])


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1

handles.statMetds = {'Amplitude', 'Amplitude^2', 'Phase - rad',...
    'Phase - deg', 'Phase Histogram', 'Amplitude & Phase',...
    'MSE - mean squared error', 'NMSE - normalized mean squared error',...
    'RMSE - root mean squared error', 'NRMSE - norm root mean squared error',...
    'MAE - mean absolute error', 'MARE - mean absolute relative error',...
    'R - coefficient of correlation', 'D - coefficient of determination',...
    'E - coefficient of efficiency', 'MaxAE - maximum absolute error',...
    'MaxARE - maximum absolute relative error'};
global filesNames T_Fit_Table
n_files = length(filesNames);

data = getFitFreqData(filesNames, T_Fit_Table);
set(handles.figure1,'UserData',data);

% Plotting 'Amplitude'
```

```
hb = bar(1:n_files,data(:,1)',0.6,'g','EdgeColor',[1,0.5,0.5]);
title([handles.statMetds{1}, '  |  T = ', num2str(T_Fit_Table,'%0.6g')],...
    'fontsize',12,'FontWeight','bold')
grid on
xlim([0  n_files+1]);
hbl = get(hb(1),'BaseLine');
set(hbl,'Color','red','LineStyle',':')
set(gca,'XTick',1:n_files,'XTickLabel',filesNames)
%-------------------


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

myPrintFig(gcf)



% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

guiResiduals_v04(handles)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = guiResiduals_v04(varargin)
% GUIRESIDUALS_V04 MATLAB code for guiResiduals_v04.fig
%      GUIRESIDUALS_V04, by itself, creates a new GUIRESIDUALS_V04 or raises the existing
%      singleton*.
%
%      H = GUIRESIDUALS_V04 returns the handle to a new GUIRESIDUALS_V04 or the handle to
%      the existing singleton*.
%
%      GUIRESIDUALS_V04('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUIRESIDUALS_V04.M with the given input arguments.
%
%      GUIRESIDUALS_V04('Property','Value',...) creates a new GUIRESIDUALS_V04 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before guiResiduals_v04_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to guiResiduals_v04_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help guiResiduals_v04

% Last Modified by GUIDE v2.5 03-Jun-2017 14:29:14

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @guiResiduals_v04_OpeningFcn, ...
                   'gui_OutputFcn',  @guiResiduals_v04_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
%--------------------------------------------------------------------%
%               *** Variables Storage ***                            %
%--------------------------------------------------------------------%
%     Variable |  'property' - GUI_component / scope                  %
%--------------------------------------------------------------------%
%  filesNames  |           global                                     %
%     names2   |           global                                     %
%     T_Fit    |           global                                     %
%  T_Fit_Table |           global                                     %
%--------------------------------------------------------------------%

%-------------- Residuals Visualizations -----------------------
%
%               '1' - error
%               '2' - error+model
%               '3' - error+data
%               '4' - error+model+data
%               '5' - absolute error
%               '6' - error vs model
%               '7' - scale-location
%               '8' - histfit
%               '9' - QQ: y1 vs N
%               '10' - QQ: y1 vs y2
% ------------------------------------------------------------------


% --- Executes just before guiResiduals_v04 is made visible.
function guiResiduals_v04_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to guiResiduals_v04 (see VARARGIN)


% Choose default command line output for guiResiduals_v04
handles.output = hObject;

global T_Fit_Table resids filesNames mainFile error
% resids = getResiduals(filesNames,T_Fit_Table);
mainFile = 1;
error = 1;

axes1_CreateFcn(hObject, eventdata, handles)


% Update handles structure
guidata(hObject, handles);

% UIWAIT makes guiResiduals_v04 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = guiResiduals_v04_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

close(handles.figure1)


% --- Executes on selection change in popupmenu2.
```

```
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu2

% ------------- Popup: select the Method ------------------------


global filesNames T_Fit_Table mainFile error errorNames
error = get(hObject,'Value');

% errorNames = {'Error','Error + Model','Error + Data','Error + Model + Data',...
%     'absolute_error','error_vs_model','scale-location','histfit',...
%     'Q-Q_y1_vs_N','Q-Q_y1_vs_y2'};


% handles.text1 = [filesNames{mainFile},'  |  ', errorNames{error},'  |  T =
',num2str(T_Fit_Table,'%0.6g')];

updateResdsAxes(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global error
error = 1;


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1


global mainFile error
mainFile = 1;
error = 1;

updateResdsAxes(hObject, eventdata, handles)
%-------------------


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

myPrintFig(gcf)


% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu3 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu3
```

```
% ------------ Popup: Select the file ------------------------
global mainFile
mainFile = get(hObject,'value');

updateResdsAxes(hObject, eventdata, handles)
%-------------------


% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

global filesNames mainFile
mainFile = 1;
set(hObject,'String',filesNames);
%-------------------


function updateResdsAxes(hObject, eventdata, handles)
% Updates axes1


global T_Fit_Table error errorNames resids mainFile filesNames
i = mainFile;
j = error;
errorNames = {' Error ', 'Standardized Error', 'Error + Model ',' Error + Data ',...
    ' Error + Model + Data ', ' Absolute Error ',' Error  X  Model ',' Scale-Location ',...
    ' Histfit ', ' Q-Q: y_a  X  N ',' Q-Q:  y_a  X  y_b '};
fName = filesNames{i};
erName = errorNames{j};

text1 = [fName,'  |  ',erName ,'  |  T = ',num2str(T_Fit_Table,'%0.6g')];
% resids{k} = [t y yi e];
data = resids{i};

switch error
    case 1   % --- error
        hb = stem(data(:,1),data(:,4),'fill','--b','LineWidth',1.5);
        set(hb,'BaseValue',0,'MarkerFaceColor','red','MarkerSize',8);
        set(get(hb,'BaseLine'),'Color','b','LineStyle',':','LineWidth',2)
        title(text1,'fontsize',12,'FontWeight','bold')
        ylabel('Error')
        xlabel('Time')
        grid on
    case 2   % --- standardized error
        hb = stem(data(:,1),data(:,4)/var(data(:,4)),'fill','--b','LineWidth',1.5);
        set(hb,'BaseValue',0,'MarkerFaceColor','red','MarkerSize',8);
        set(get(hb,'BaseLine'),'Color','b','LineStyle',':','LineWidth',2)
        title(text1,'fontsize',12,'FontWeight','bold')
        ylabel('Error')
        xlabel('Time')
        grid on
    case 3   % --- error+model
        hb = errorbar(data(:,1),data(:,3),data(:,4),'.r');
        set(hb,'LineStyle','none','LineWidth',1.5)
        hold on
        h1 = plot(data(:,1),data(:,3),':bo');
        set(h1,'MarkerSize',6,'MarkerEdgeColor','b','MarkerFaceColor','b','LineWidth',1)
        title(text1,'fontsize',12,'FontWeight','bold')
        ylabel('Error + Model')
        xlabel('Time')
        legend('Error','Model')
        grid on
        hold off
    case 4   % --- error+data
        hb = errorbar(data(:,1),data(:,2),data(:,4),'.r');
        set(hb,'LineStyle','none','LineWidth',1.5)
```

```
        hold on
        h1 = plot(data(:,1),data(:,2),':bo');
        set(h1,'MarkerSize',6,'MarkerEdgeColor','b','MarkerFaceColor','b','LineWidth',1)
        title(text1,'fontsize',12,'FontWeight','bold')
        ylabel('Error + Data')
        xlabel('Time')
        legend('Error','Data')
        grid on
        hold off
    case 5   % --- error+model+data
        hb = errorbar(data(:,1),data(:,3),data(:,4),'.r');
        set(hb,'LineStyle','none','LineWidth',1.5)
        hold on
        h1 = plot(data(:,1),data(:,3),':gd');
        set(h1,'MarkerSize',6,'MarkerEdgeColor','g','MarkerFaceColor','g','LineWidth',1)
        h2 = plot(data(:,1),data(:,2),':bo');
        set(h2,'MarkerSize',6,'MarkerEdgeColor','b','MarkerFaceColor','b','LineWidth',1)
        title(text1,'fontsize',12,'FontWeight','bold')
        ylabel('Error + Model + Data')
        xlabel('Time')
        legend('Error','Model','Data')
        hold off
    case 6   % --- absolute error
        hb = stem(data(:,1),abs(data(:,4)),'fill','--bo','LineWidth',1.5);
        set(hb,'BaseValue',0,'MarkerFaceColor','red','MarkerSize',8);
        set(get(hb,'BaseLine'),'Color','b','LineStyle',':','LineWidth',2)
        title(text1,'fontsize',12,'FontWeight','bold')
        ylabel('Absolute Error')
        xlabel('Time')
        grid on
    case 7   % --- error vs model
        hb = plot(data(:,3),data(:,4),'bo');
        set(hb,'MarkerSize',8,'MarkerEdgeColor','b','MarkerFaceColor','b')
        title(text1,'fontsize',12,'FontWeight','bold')
        ylabel('Error')
        xlabel('Fitted Values (Model)')
        grid on
    case 8   % --- scale-location
        hb = plot(data(:,3),sqrt(abs(data(:,4))),'bo');
        set(hb,'MarkerSize',8,'MarkerEdgeColor','b','MarkerFaceColor','b')
        title(text1,'fontsize',12,'FontWeight','bold')
        ylabel('Root Squared Error')
        xlabel('Fitted Values (Model)')
        grid on
    case 9   % --- histfit
        hb = histfit(data(:,4));
        title(text1,'fontsize',12,'FontWeight','bold')
        ylabel('Counts')
        xlabel('Error Values')
        grid on
    case 10   % --- QQ: y1 vs N
        hb = qqplot(data(:,4));
        title(text1,'fontsize',12,'FontWeight','bold')
%         ylabel('Root Squared Error')
%         xlabel('Fitted Values (Model)')
        grid on
    case 11   % --- QQ: y1 vs y2
        hb = qqplot(data(:,3),data(:,4));
        title(text1,'fontsize',12,'FontWeight','bold')
%         ylabel('Root Squared Error')
%         xlabel('Fitted Values (Model)')
        grid on

end
%-------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = LST_Table_v01(varargin)
% LST_TABLE_V01 MATLAB code for LST_Table_v01.fig
%      LST_TABLE_V01, by itself, creates a new LST_TABLE_V01 or raises the existing
%      singleton*.
%
%      H = LST_TABLE_V01 returns the handle to a new LST_TABLE_V01 or the handle to
%      the existing singleton*.
```

```
%
%      LST_TABLE_V01('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in LST_TABLE_V01.M with the given input arguments.
%
%      LST_TABLE_V01('Property','Value',...) creates a new LST_TABLE_V01 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before LST_Table_v01_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to LST_Table_v01_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help LST_Table_v01

% Last Modified by GUIDE v2.5 03-Jun-2017 14:31:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @LST_Table_v01_OpeningFcn, ...
                   'gui_OutputFcn',  @LST_Table_v01_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

%---------------- Parameters ---------------------------------
%
%               - Amplitude
%               - Phase
%               - mean squared error (MSE)
%               - normalised mean squared error (NMSE)
%               - root mean squared error (RMSE)
%               - normalised root mean squared error (NRMSE)
%               - mean absolute error (MAE)
%               - mean  absolute relative error  (MARE)
%               - coefficient of correlation (R)
%               - coefficient of determination (D)
%               - coefficient of efficiency (E)
%               - maximum absolute error (Max-AE)
%               - maximum absolute relative error (Max-ARE)
%-------------------------------------------------------------



% --- Executes just before LST_Table_v01 is made visible.
function LST_Table_v01_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to LST_Table_v01 (see VARARGIN)

% Choose default command line output for LST_Table_v01
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes LST_Table_v01 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = LST_Table_v01_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


global names2 myTable T_Fit_Table

handles.T_Fit = T_Fit_Table;
samplEstats = {'Amplitude','Phase-rad','Phase-deg','MSE','NMSE','RMSE',...
    'NRMSE','MAE','MARE','R','D','E','Max-AE','Max-ARE'};
data = myTable;

disp(' ')
disp([' Results for T = ' num2str(T_Fit_Table) ' wrote to "temp_LST_Table.xls"'])

xlswrite('temp_LST_Table.xls', names2', ['T=' num2str(T_Fit_Table)], 'A2')
xlswrite('temp_LST_Table.xls', samplEstats, ['T=' num2str(T_Fit_Table)], 'B1')
xlswrite('temp_LST_Table.xls', data, ['T=' num2str(T_Fit_Table)], 'B2')

% %--- junk code -----
% t=1; y=0;
% if t ~= y
%     h = errordlg({'Ops...Software under Construction!!!'} ,'Wait a moment!...');
%
%     waitfor(h);
% %     close(handles.figure1)
% end
%--------------------



% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% %--- junk code -----
% t=1; y=0;
% if t ~= y
%     h = errordlg([{'   File NOT saved!'};{' '};{' '};{' '};...
%         {' Software under Construction!!!'}],' Wait a moment!...');
%
%     waitfor(h);
% %     close(handles.figure1)
% end
% %--------------------
close(handles.figure1)


% --- Executes during object creation, after setting all properties.
function text1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

global T_Fit_Table
handles.T_Fit = T_Fit_Table;

set(hObject,'String',['  T = ' num2str(handles.T_Fit,'%0.6g')])
```

```
% --- Executes during object creation, after setting all properties.
function uitable1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uitable1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
global names2 myTable

samplEstats = {'Amplitude','Phase-rad','Phase-deg','MSE','NMSE','RMSE',...
    'NRMSE','MAE','MARE','R','D','E','Max-AE','Max-ARE'};

data = myTable;

set(hObject,'RowName', names2)
set(hObject,'ColumnName', samplEstats)
set(hObject,'Data',data)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = LSTperiod(varargin)
% LSTPERIOD MATLAB code for LSTperiod.fig
%      LSTPERIOD, by itself, creates a new LSTPERIOD or raises the existing
%      singleton*.
%
%      H = LSTPERIOD returns the handle to a new LSTPERIOD or the handle to
%      the existing singleton*.
%
%      LSTPERIOD('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in LSTPERIOD.M with the given input arguments.
%
%      LSTPERIOD('Property','Value',...) creates a new LSTPERIOD or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before LSTperiod_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to LSTperiod_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help LSTperiod

% Last Modified by GUIDE v2.5 03-Jun-2017 12:30:44

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @LSTperiod_OpeningFcn, ...
                   'gui_OutputFcn',  @LSTperiod_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

%--------------------------------------------------------------------%
%               *** Variables Storage ***                            %
%--------------------------------------------------------------------%
%     Variable |  'property' - GUI_component / scope                  %
%--------------------------------------------------------------------%
%  filesNames  |           global                                     %
%     tmin     |   'Userdata' - slider1                               %
%     TMax     |   'Userdata' - slider2                               %
%     df       |   'Userdata' - slider3                               %
%     guiData  |   setappdata() & getappdata()   (figure1)            %
%     cBW      |   'Userdata' - edit4                                 %
%     T_Fit    |           global                                     %
```

```
%  T_Fit_Table |           global                                    %
%--------------------------------------------------------------------%
%   cursorMode |   'Userdata' - uitoggletool3                        %
%   (handles)  |                                                     %
%--------------------------------------------------------------------%

% --- Executes just before LSTperiod is made visible.
function LSTperiod_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to LSTperiod (see VARARGIN)

% Initializating Code
% ------------------------------------------------------------------------
global filesNames names2 T_Fit T_Fit_Table
filesNames = getFilesNames();
names2 = filesNames;
T_Fit = 0; T_Fit_Table = 0;
[a, b, c, d] = read_Files(filesNames);
handles.par = [a, b, c, d];
handles.printOption = '.pdf';

initialize(handles)

% ------------------------------------------------------------------------

% Choose default command line output for LSTperiod
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes LSTperiod wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = LSTperiod_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

var_sld3 = get(hObject,'Value');       % getting the value of slider
df = 100+1000*(100-var_sld3);          % Calculating df

set(handles.text8,'String',[num2str(df) ' grid points']); % updating static text
set(handles.slider3,'UserData',df);
handles.parUser(3,1) = df;             % setting df

guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
```

```
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

a = handles.par(:);                         % getting data series limits
cBW = get(handles.edit4,'UserData');
t2 = a(4);
t1 = cBW;
var_sld2 = get(hObject,'Value');              % getting the value of slider
TMax = t1+ var_sld2*(t2-t1)/100;              % Calculating T_Max

set(handles.text7,'String',[num2str(TMax) '  time units']);   % updating static text
set(handles.slider2,'UserData',TMax);
handles.parUser(2,1)=TMax;                    % getting T_Max

updateFreqLimOnAxes (handles)
guidata(hObject, handles);



% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
set(hObject,'SliderStep',[0.001 0.010]);

% Setting slider2 to Max initial value
var = get(hObject, 'Max');
set(hObject, 'Value', var);


% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

a = handles.par;                            % getting data series limits
cBW = get(handles.edit4,'UserData');
t2 = cBW;
t1 = 2*a(1);
var_sld1 = get(hObject,'Value');            % getting the value of slider
tmin = t1+ var_sld1*(t2-t1)/100;            % Calculating t_min

set(handles.text6,'String',[num2str(tmin) '  time units']);  % updating static text
set(handles.slider1,'UserData',tmin);
handles.parUser(1,1)=tmin;                  % sending t_min

updateFreqLimOnAxes (handles)
guidata(hObject, handles);



% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
set(hObject,'SliderStep',[0.001 0.040]);


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global filesNames
tmin = get(handles.slider1,'UserData');
TMax = get(handles.slider2,'UserData');
df = get(handles.slider3,'UserData');

data = getGuiData3(filesNames, [tmin TMax df]);
setappdata(handles.figure1,'guiData',data);

updateAxes (handles)

guidata(hObject, handles);



% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1

value = get(hObject,'Value');
if value == 1
    set(handles.axes1, 'XGrid', 'on','YGrid', 'on');
    set(handles.axes2, 'XGrid', 'on','YGrid', 'on');
    set(handles.axes3, 'XGrid', 'on','YGrid', 'on');
else
    set(handles.axes1, 'XGrid', 'off','YGrid', 'off');
    set(handles.axes2, 'XGrid', 'off','YGrid', 'off');
    set(handles.axes3, 'XGrid', 'off','YGrid', 'off');
end



% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox2

value = get(hObject,'Value');
if value == 1
    set(handles.axes1, 'Yscale', 'log');
    set(handles.axes2, 'Yscale', 'log');
    set(handles.axes3, 'Yscale', 'log');
else
    set(handles.axes1, 'Yscale', 'linear');
    set(handles.axes2, 'Yscale', 'linear');
    set(handles.axes3, 'Yscale', 'linear');
end


% --- Executes during object creation, after setting all properties.
function text6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function text7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function text8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1


function updateSliders (handles)
% Updating all sliders to initial state (with actual parameters)

disp (' Updating sliders...')
a = handles.par;                                % getting actual data series limits
cBW = get(handles.edit4,'UserData');
% ------------- Slider3 -------------
% Setting slider3 to Max initial value
var = get(handles.slider3, 'Max');
set(handles.slider3, 'Value', var);

var_sld3 = get(handles.slider3,'Value');     % getting the value of slider
df = 100+1000*(100-var_sld3);                % Calculating df (number of points)

set(handles.text8,'String',[num2str(df) ' grid points']); % updating static text
set(handles.slider3,'UserData',df);
% ------------- Slider2 -------------
t2 = a(4);
t1 = cBW;
% Setting slider2 to Max initial value
var = get(handles.slider2, 'Max');
set(handles.slider2, 'Value', var);

var_sld2 = get(handles.slider2,'Value');           % getting the value of slider
TMax = t1+ var_sld2*(t2-t1)/100;                   % Calculating T_Max

set(handles.text7,'String',[num2str(TMax) '  time units']);   % updating static text
set(handles.slider2,'UserData',TMax);
% ------------- Slider1 -------------
t2 = cBW;
t1 = 2*a(1);
% Setting slider1 to Min initial value
var = get(handles.slider1, 'Min');
set(handles.slider1, 'Value', var);

var_sld1 = get(handles.slider1,'Value');          % getting the value of slider
tmin = t1+ var_sld1*(t2-t1)/100;                  % Calculating t_min

set(handles.text6,'String',[num2str(tmin) '  time units']);  % updating static text
set(handles.slider1,'UserData',tmin);


function updateFreqLimOnAxes(handles)
% Updating frequency limits in all graphics
global filesNames
tmin = get(handles.slider1,'UserData');  % getting the value of slider1
TMax = get(handles.slider2,'UserData');  % getting the value of slider2
data = getappdata(handles.figure1,'guiData');       % getting data to plot
```

```
plot(handles.axes1, data(:,1), data(:,3))
YlimAxes1 = get(handles.axes1,'Ylim');  % AND Spectrum
plot(handles.axes1, data(:,1), data(:,3),...
    tmin*ones(20,1),linspace(YlimAxes1(1),YlimAxes1(2),20), '>m',...
    TMax*ones(20,1),linspace(YlimAxes1(1),YlimAxes1(2),20),
'<b','MarkerSize',4,'LineWidth',1.5);

plot(handles.axes2, data(:,1), data(:,2))
YlimAxes2 = get(handles.axes2,'Ylim');  % OR Spectrum
plot(handles.axes2, data(:,1), data(:,2),...
    tmin*ones(20,1),linspace(YlimAxes2(1),YlimAxes2(2),20), '>m',...
    TMax*ones(20,1),linspace(YlimAxes2(1),YlimAxes2(2),20),
'<b','MarkerSize',4,'LineWidth',1.5);

plot(handles.axes3, data(:,1), data(:,4:end))
YlimAxes3 = get(handles.axes3,'Ylim');  % Each Spectra separately
plot(handles.axes3, data(:,1), data(:,4:end),...
    tmin*ones(20,1),linspace(YlimAxes3(1),YlimAxes3(2),20), '>m', ...
    TMax*ones(20,1),linspace(YlimAxes3(1),YlimAxes3(2),20), '<b','MarkerSize',5);

ylabel(handles.axes1,'P(\tau)','FontSize',12)
ylabel(handles.axes2,'P(\tau)','FontSize',12)
ylabel(handles.axes3,'P(\tau)','FontSize',12)
xlabel(handles.axes3,'period (time units)','FontSize',12)
axis(handles.axes1,'tight')
axis(handles.axes2,'tight')
axis(handles.axes3,'tight')

names = filesNames;
names = [names 'T_{min}' 'T_{Max}'];
AX=legend(handles.axes3, names, 'Location','Best');
LEG = findobj(AX,'type','text');
set(LEG,'FontSize',7)

datacursormode off;
set(handles.checkbox1, 'Value',0); set(handles.checkbox2, 'Value',0);
global T_Fit
T_Fit = 0;


function updateAxes (handles)
% Updating the actual data set in all graphics
global filesNames
% getting data to plot
data = getappdata(handles.figure1,'guiData');       % getting data to plot

plot(handles.axes1, data(:,1), data(:,3),'LineWidth',1.5)
plot(handles.axes2, data(:,1), data(:,2),'LineWidth',1.5)
plot(handles.axes3, data(:,1), data(:,4:end))

ylabel(handles.axes1,'P(\tau)','FontSize',12)
ylabel(handles.axes2,'P(\tau)','FontSize',12)
ylabel(handles.axes3,'P(\tau)','FontSize',12)
xlabel(handles.axes3,'period (time units)','FontSize',12)
axis(handles.axes1,'tight')
axis(handles.axes2,'tight')
axis(handles.axes3,'tight')

AX=legend(handles.axes3, filesNames, 'Location','Best');
LEG = findobj(AX,'type','text');
set(LEG,'FontSize',7)

datacursormode off;
set(handles.checkbox1, 'Value',0); set(handles.checkbox2, 'Value',0);
global T_Fit
T_Fit = 0;


% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% --------------------------------------------------------------------
function uipushtool2_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to uipushtool2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(gcf,'PaperOrientation','landscape')
set(gcf,'PaperType','a4')
set(gcf,'PaperPositionMode','auto')

printpreview(gcf)
% printdlg(gcf)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
printResults(handles)


% --- Executes when selected object is changed in uipanel5.
function uipanel5_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel5
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%      EventName: string 'SelectionChanged' (read only)
%      OldValue: handle of the previously selected object or empty if none was selected
%      NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)

value = get(hObject,'String');
handles.printOption = value;

guidata(hObject, handles);


function printResults (handles)
% Exporting graphics and saving them in file

value = handles.printOption;

switch (value)
    case '.pdf'
       printAllData(handles, 'pdf')
    case '.jpg'
       printAllData(handles, 'jpeg')
    case '.tif'
       printAllData(handles, 'tiff')
    case '.eps'
       printAllData(handles, 'epsc')
    case '.fig'
       printAllData(handles, 'fig')
end



function printAllData(handles, fileFormat)

global filesNames
% nameOut = 'ResultsLS_gui';

A={'*.eps';'*.epsc';'*.jpeg';'*.jpg';'*.svg';'*.pdf';'*.tif';'*.tiff';...
  '*.png';'*.emf';'*.fig'};
[filename, pathname, filterIndex] = uiputfile(A, 'Save as');

disp(' ')
if isequal(filename,0) || isequal(pathname,0)
   disp(' User selected Cancel')
   return
else
   disp([' User selected >>  ',fullfile(pathname,filename)])
end
fileFormat = A{filterIndex}(3:end);
nameOut = filename;
```

```
data = getappdata(handles.figure1,'guiData');       % Getting the data to plot and print

hf = figure('units','normalized','outerposition',[0 0 1 1]);
movegui(hf,'center')          % Move the GUI to the center of the screen
hold on

subplot(3,1,1)                % Plotting AND spectrum
plot(data(:,1), data(:,3),'LineWidth',1.5);
title('AND', 'FontSize', 12)
ylabel('P(\tau)', 'FontSize', 10)
grid on;
subplot(3,1,2)                % Plotting OR spectrum
plot(data(:,1), data(:,2),'LineWidth',1.5);
title('OR', 'FontSize', 12)
ylabel('P(\tau)', 'FontSize', 10)
grid on;
subplot(3,1,3)                % Plotting Separately spectra
axes3 = plot(data(:,1), data(:,4:end));
ylabel('P(\tau)', 'FontSize', 10)
xlabel('period (time units)', 'FontSize', 11)
grid on;


loc_Auto=(strcmp(fileFormat, 'epsc')|strcmp(fileFormat, 'tiff')|...
    strcmp(fileFormat, 'jpeg'));
if strcmp(fileFormat, 'fig')
    set(gcf,'PaperPositionMode','auto')
    AX=legend(axes3, filesNames, 'Location','Best');
    LEG = findobj(AX,'type','text');
    set(LEG,'FontSize',7)
elseif loc_Auto
    set(gcf,'PaperPositionMode','auto')
%     names = getappdata(handles.figure1,'names');
%     AX=legend(axes3, names, 'Location','Best');
%     LEG = findobj(AX,'type','text');
%     set(LEG,'FontSize',7)
else                                    % Only for .pdf
    set(gcf,'PaperType','a4')
%     set(gcf,'PaperOrientation','landscape')
end
pause(0.5)
saveas (gcf, nameOut, fileFormat)

delete(hf)


% --------------------------------------------------------------------
function uipushtool3_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to uipushtool3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global filesNames
[filename, pathname, filterindex] = uigetfile( ...
{  '*.dat','Data Files (*.dat)'; ...
   '*.*',  'All Files (*.*)'}, ...
   'Select a new data file Set', ...
   'MultiSelect', 'on');

if ~iscell(filename) && ~isequal(filename,0)  % filename ~= 0
    filename = cellstr(filename);
elseif isequal(filename,0) || isequal(pathname,0)
    disp(' ')
    disp(' uigetfile: User pressed cancel.')
    return
end
filesNames = cell(1,length(filename));
for k = 1:length(filename)
    temp_string = filename{k};
    temp_string = temp_string(1:end-4);
    filesNames{k} = temp_string;
end
```

```
[a, b, c, d] = read_Files(filesNames);
handles.par = [a, b, c, d];

initialize(handles)
updateSliders(handles)

guidata(hObject, handles);


% Initializating Code
% --------------------------------------------------------------------
function initialize(handles)

global filesNames

a = handles.par;

halfWidth = (a(4)-2*a(1))/10;
set(handles.edit4,'String',num2str(halfWidth));
set(handles.edit4,'UserData',halfWidth);

global T_Fit
T_Fit = 0;

tmin = 2*a(1);                          % t_min at start
TMax = a(4);                            % T_Max at start
df = 100;                             % df at start
set(handles.slider1,'UserData',tmin);
set(handles.slider2,'UserData',TMax);
set(handles.slider3,'UserData',df);
handles.parUser = [tmin TMax df]';   % t_min, T_Max and df

disp(' Wait. Reading data files and setting start parameters...')

% getting data to plot
data = getGuiData3(filesNames, handles.parUser);
setappdata(handles.figure1,'guiData',data);

plot(handles.axes1, data(:,1), data(:,3),'LineWidth',1.5);     % AND Spectrum
plot(handles.axes2, data(:,1), data(:,2),'LineWidth',1.5);     % OR Spectrum
plot(handles.axes3, data(:,1), data(:,4:end)); % Each Spectrum separately
ylabel(handles.axes1,'P(\tau)','FontSize',12)
ylabel(handles.axes2,'P(\tau)','FontSize',12)
ylabel(handles.axes3,'P(\tau)','FontSize',12)
xlabel(handles.axes3,'period (time units)','FontSize',12)
axis(handles.axes1,'tight')
axis(handles.axes2,'tight')
axis(handles.axes3,'tight')

AX=legend(handles.axes3, filesNames, 'Location','Best');
LEG = findobj(AX,'type','text');
set(LEG,'FontSize',7)

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text6.
function text6_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function uitoggletool3_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to uitoggletool3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cursorMode = datacursormode(gcf);
set(handles.uitoggletool3,'UserData',cursorMode);

datacursormode on;
set(cursorMode,'UpdateFcn',@myDataTip)

function text_to_display = myDataTip(cursorMode, eventData, handles)
pos = get(eventData,'Position');
```

```
text_to_display = {['T = ',num2str(pos(1))]};
global T_Fit
T_Fit = pos(1);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global T_Fit T_Fit_Table filesNames names2
if T_Fit ~= 0
    %--------- Here is the second GUI-2: 'guiFitFreq.fig'------------------
    T_Fit_Table = T_Fit;
    names2 = filesNames;

    % Setting 'datacursormode' to initial value
    cursorMode = get(handles.uitoggletool3,'UserData');
    set(cursorMode,'DisplayStyle','window')
    datacursormode off;
    set(cursorMode,'DisplayStyle','datatip')

    disp(' ')
    disp([' Calculating linear fit for T = ' num2str(T_Fit) ' ...'])
    disp(' ')
    guiFitFreq_v06(handles)
    disp([' LS just calculated for T = ' num2str(T_Fit) '!'])
    disp(' ')
    %-------------------------------------------------------------------------
else
    disp(' ')
    disp(' ** First you need to choose a period, T, to test! **')
    disp(' Hint: use the "Data Cursor" button on the toolbar above')
    disp(' ')
end

% Setting 'T_Fit' to initial value
T_Fit = 0;
guidata(hObject, handles);


function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double

a = handles.par;
str_cBW = get(hObject,'String');
cBW = str2double(str_cBW);
if cBW > a(1) && cBW < a(4)
    set(handles.edit4,'UserData', cBW);
    updateSliders(handles)
else
    set(hObject,'String',num2str(get(handles.edit4,'UserData')));

%   Or bring back the original value of cBW...
%   set(hObject,'String',num2str((a(4)-2*a(1))/10));
%   updateSliders(handles)
%   set(handles.edit4,'UserData', (a(4)-2*a(1))/10);
end
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


h =  msgbox ({'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%                      ',...
              '                                                                   ',...
              '                     Universidade Federal de Santa Catarina -  UFSC',...
              '                                                                   ',...
              '                                         Florianópolis / Brazil ',...
              '                                                                   ',...
              '                                                                   ',...
              '                                                                   ',...
              '                        Copyright (c) 2018,  George Caminha-Maciel ',...
              '                                                                   ',...
              '                                           All rights reserved. ',...
              '                                                                   ',...
              '                                                                   ',...
              '        ==>  PLEASE CITE:                                         ',...
              '                                                                   ',...
              '              * G. Caminha-Maciel & M. Ernesto. " LSTperiod software:   ',...
              '         spectral analysis of multiple irregularly sampled time series.',...
              '         Annals of Geophysics, 2019.',...
',...
              '                                                                   ',...
              '                                                                   ',...
              '%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%                      ',...
              '                                                                   ',...
              '                                                                   '},...
              'Just a moment!... ','help');
btn_h = findobj(h, 'style', 'pushbutton');
set(btn_h, 'String', 'OK...(20s)');

uiwait(h,20);
disp(' '); disp(' Goodbye!')

h=findobj('Type','figure');
delete(h);


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over pushbutton2.
function pushbutton2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on key press with focus on pushbutton2 and none of its controls.
function pushbutton2_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  structure with the following fields (see UICONTROL)
%      Key: name of the key that was pressed, in lower case
%      Character: character interpretation of the key(s) that was pressed
%      Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)


function myPrintFig (handlesFig)
%MYPRINTFIG Print current figure

disp(' Saving figure...')
A={'*.bmp';'*.emf';'*.eps';'*.epsc';'*.jpeg';'*.pdf'; '*.png'; '*.tiff'};
[filename, pathname, filterIndex] = uiputfile(A, 'Save as');


if isequal(filename,0) || isequal(pathname,0)
   disp(' You selected Cancel!')
```

```
   return
else
   disp([' You selected ',fullfile(pathname,filename)])
end

ext = A{filterIndex}(3:end);

switch ext
    case 'bmp'
       dext = '-dbmp';
    case 'emf'
        dext = '-dmeta';
    case 'eps'
       dext = '-deps2';
    case 'epsc'
        dext = '-depsc2';
    case 'jpeg'
       dext = '-djpeg';
    case 'pdf'
        dext = '-dpdf';
    case 'png'
       dext = '-dpng';
    case 'tiff'
        dext = '-dtiff';
end

% set(gca, 'LineWidth', 1);
% set(gca, 'FontName', 'Helvetica');
% set(gca, 'FontSize', 12, 'FontWeight', 'bold');
% set(gcf, 'PaperPositionMode', 'auto');

print(gcf, filename, dext, '-noui')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Tmm, Tm_avg, TM_avg, TMM] = read_Files(names)
% Read data files and send parameters to build the vector W(i)
% Only text data files (ASCII) with extension ".dat"

% Definitions:
% Tmm --> minimal interval for all series
% Tm_avg --> average of all mean-intervals
% TM_avg --> average length of all series
% TMM --> biggest length for all series

T_min = ones(length(names),1);T_Max = ones(length(names),1);
dT_avg = ones(length(names),1);
for k = 1:length(names)
    D{k} = load ([names{k} '.dat']);     % Only for t columm
    T_min(k) = min(abs(diff(D{k}(:,1))));
    T_Max(k) = max(abs(D{k}(end,1)-D{k}(1,1)));
    dT_avg(k) = mean(abs(D{k}(end,1)-D{k}(1,1)));
end

Tmm = min(T_min); Tm_avg = mean(dT_avg);
TM_avg = mean(T_Max); TMM = max(T_Max);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function yi = sineInterp(t, X, T)
%MYINTERP Generates a pure sine on sample (time) points

wf = 2*pi/T;
yi = X(1)*cos(wf*t)+X(2)*sin(wf*t);
end
```