```
%Function L2 Adjustment using method of indirect observations
%by James Sapcoe, Nov 20, 96
%Input Bmatrix and Fmatrix from v+B*Delta=F
%Output paramters(x) and residuals(v)
%Syntax [x,v]=l2(b,f)

function [x,v]=L2(B,f)

x=inv(B'*B)*(B'*f);
v=f-B*x;
```

```
%Function L1 Adjustment using Linear Programing
%requires lp.m and qb.m (optimization toolbox)
%by Nakarin Satthamnuwong , Nov 19,96
%updated Oct 2001--linprog now replaces lp
%Input B matrix and F matrix from v+B*Delta=F
%i.e. method of indirect observations
%Output paramters(x) and residuals(v)
%Syntax [x,v]=l1(b,f)
% updated 24 nov 2008 for new linprog arguments

function [x,v]=L1(B,f)

[n,u]=size(B);

% rename some things

b=f;                                %fill b matrix
%f=[zeros(1,2*u) ones(1,2*n)];       %fill f' (Objective function)
f=[zeros(2*u,1) ; ones(2*n,1)];      % fill f
for i=1:n;                          %fill A matrix
  for j=1:u
    A(i,2*j-1)=B(i,j);
    A(i,2*j)=-B(i,j);
    end
  A(i,2*u+2*i-1)=1;
  A(i,2*u+2*i)=-1;
  end

%T=lp(f,A,b,zeros(2*u+2*n,1),[],[],n);   %use lp (obsolete) function
%keyboard
%T=linprog(f,A,b,A,b,zeros(2*u+2*n,1),[],[]);   %use linprog (new) function
options=optimset('LargeScale','off','Simplex','on');
T=linprog(f,[],[],A,b,zeros(2*u+2*n,1),[],[],options); % new args R2007a, R2008a

% calculate parameters

for j=1:u
  x(j)=T(2*j-1)-T(2*j);
  end;

% calculate residuals

for j=1:n
  v(j)=T(2*u+2*j-1)-T(2*u+2*j);
  end

x=x';
v=v';
```

```
function varargout = inner6(varargin)
% INNER6 Application M-file for inner6.fig
%    FIG = INNER6 launch inner6 GUI.
%    INNER6('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 19-Jan-2003 12:07:56

if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % Use system color scheme for figure:
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);

        if nargout > 0
                varargout{1} = fig;
        end

        % i put something here to initialize the fig, see if it works
        % yes, it works
        set(handles.four_par_radio,'value',1);
        set(handles.l2_radio,'value',1);
        set(handles.output_edit_text,'string','');
        set(handles.fiducial_edit_text,'string','');
        set(handles.measurement_edit_text,'string','');
        set(handles.message_text,'string','');

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.


% --------------------------------------------------------------------
function varargout = get_fiducial_pushbitton_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.get_fiducial_pushbitton.
[filename,pathname]=uigetfile({'*.*','All Files (*.*)'},'Select File');
if (filename ~= 0)
  set(handles.fiducial_edit_text,'string',strcat(pathname,filename));
else
  set(handles.fiducial_edit_text,'string','');
  end


% --------------------------------------------------------------------
function varargout = get_measurement_pushbutton_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.get_measurement_pushbutton.
[filename,pathname]=uigetfile({'*.*','All Files (*.*)'},'Select File');
if (filename ~= 0)
  set(handles.measurement_edit_text,'string',strcat(pathname,filename));
else
  set(handles.measurement_edit_text,'string','');
  end


% --------------------------------------------------------------------
function varargout = get_output_pushbutton_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.get_output_pushbutton.
[filename,pathname]=uiputfile({'*.*','All Files (*.*)'},'Select File');
if (filename ~= 0)
  set(handles.output_edit_text,'string',strcat(pathname,filename));
else
  set(handles.output_edit_text,'string','');
  end


% --------------------------------------------------------------------
function varargout = fiducial_edit_text_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.fiducial_edit_text.


% --------------------------------------------------------------------
function varargout = measurement_edit_text_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.measurement_edit_text.


% --------------------------------------------------------------------
function varargout = output_edit_text_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.output_edit_text.
% --------------------------------------------------------------------
function varargout = four_par_radio_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.four_par_radio.
set(handles.four_par_radio,'value',1);
set(handles.six_par_radio,'value',0);

% --------------------------------------------------------------------
function varargout = six_par_radio_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.six_par_radio.
set(handles.six_par_radio,'value',1);
set(handles.four_par_radio,'value',0);


% --------------------------------------------------------------------
function varargout = l1_radio_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.l1_radio.
set(handles.l1_radio,'value',1);
set(handles.l2_radio,'value',0);


% --------------------------------------------------------------------
function varargout = l2_radio_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.l2_radio.
set(handles.l2_radio,'value',1);
set(handles.l1_radio,'value',0);


% --------------------------------------------------------------------
function varargout = result_listbox_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.result_listbox.


% --------------------------------------------------------------------
function varargout = run_button_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.run_button.

global RESID_X_PLOT;
global RESID_Y_PLOT;
global FID_X_PLOT;
global FID_Y_PLOT;
global FID_NUM_PLOT;
global NUM_FID;
global POINT_ID
global POINT_X
global POINT_Y
global NUM_POINT

%A=cell(3,1);
%num=1;
%resx=0.55;
%resy=0.66;
%s=sprintf('%5d %8.2f %8.2f',num,resx,resy);
%A(1)=cellstr(s);
%num=2;
%resx=0.77;
%resy=0.88;
%s=sprintf('%5d %8.2f %8.2f',num,resx,resy);
%A(2)=cellstr(s);
%set(handles.result_listbox,'string',A)

% four parameter model
%
```

```
% x =  aX + bY + c
% y = -bx + aY + d
%
% | vx | + | -X  -Y  -1    0 | | a | = | -x |
% | vy |   | -Y    X   0  -1 | | b |   | -y |
%                             | c |
%                             | d |
% scale=sqrt(a^2 + b^2)
%
% six parameter model
%
% x =  a0 + a1*X + a2*Y
% y =  b0 + b1*X + b2*Y
%
% | vx | + | -1 -X -Y  0  0  0| | a0 | = | -x |
% | vy |   |  0  0  0  1 -X -Y| | a1 |   | -y |
%                               | a2 |
%                               | b0 |
%                               | b1 |
%                               | b2 |
% scale_x=sqrt(a1^2 + b1^2)
% scale_y=sqrt(a2^2 + b2^2)

% open and read fiducial data

fidfile=get(handles.fiducial_edit_text,'string');
[fnum,fx,fy]=textread(fidfile,'%f %f %f');
[numfid,n]=size(fnum);

% open and read observation data (fids + pass & control points)

phofile=get(handles.measurement_edit_text,'string');
[pid,pr,pc]=textread(phofile,'%s %f %f');
[numfp,n]=size(pid);
nump=numfp - numfid;

% get option and do four parameter transformation if requested

do4par=get(handles.four_par_radio,'value');
if(do4par == 1)
  if(numfid < 2)
    disp('not enough points');
    return
    end
  B=zeros(numfid*2,4);
  f=zeros(numfid*2,1);
  ndx=1;
  for i=1:numfid
    B(ndx  ,:)=[-fx(i) -fy(i) -1  0];
    f(ndx  )= -pr(i);
    B(ndx+1,:)=[-fy(i)  fx(i)  0 -1];
    f(ndx+1)= -pc(i);
    ndx=ndx+2;
    end
  doL2=get(handles.l2_radio,'value');
  if(doL2 == 1)
    [par,v]=L2(B,f);
  else
    [par,v]=L1(B,f);
    end

  scale=sqrt(par(1)^2 + par(2)^2);
  pix_per_mm=scale;
  mm_per_pix=1/scale;
  vr=zeros(numfid,1);
  vc=zeros(numfid,1);
  % invert and apply transform to data points
  % store in global variables
  mx=[par(1) par(2); -par(2) par(1)];
  mx_inv=inv(mx);
  NUM_POINT=nump;
  POINT_ID=cell(nump,1);
  POINT_X=zeros(nump,1);
  POINT_Y=zeros(nump,1);
  for i=1:nump
    POINT_ID(i)=pid(numfid + i);
    temp=[pr(numfid + i); pc(numfid + i)] - [par(3); par(4)];
    cal=mx_inv*temp;
    POINT_X(i)=cal(1);
    POINT_Y(i)=cal(2);
    end
  end


% ===================== put 6-par code here ========================

% get option and do six parameter transofrmation if requested

do6par=get(handles.six_par_radio,'value');
if(do6par == 1)
  if(numfid < 3)
    disp('not enough points for six-par');
    return
    end
  B=zeros(numfid*2,6);
  f=zeros(numfid*2,1);
  ndx=1;
  for i=1:numfid
    B(ndx  ,:)=[-1 -fx(i) -fy(i) 0  0  0];
    f(ndx  )= -pr(i);
    B(ndx+1,:)=[0  0  0 -1 -fx(i) -fy(i)];
    f(ndx+1)= -pc(i);
    ndx=ndx+2;
    end
  doL2=get(handles.l2_radio,'value');
  if(doL2 == 1)
    [par,v]=L2(B,f);
  else
    [par,v]=L1(B,f);
    end


  scale_x=sqrt(par(2)^2 + par(5)^2);
  scale_y=sqrt(par(3)^2 + par(6)^2);
  scale=(scale_x + scale_y)/2;
  pix_per_mm=scale;
  mm_per_pix=1/scale;
  vr=zeros(numfid,1);
  vc=zeros(numfid,1);
  % invert and apply transform to data points
  % store in global variables
  mx=[par(2) par(3); par(5) par(6)];
  mx_inv=inv(mx);
  NUM_POINT=nump;
  POINT_ID=cell(nump,1);
  POINT_X=zeros(nump,1);
  POINT_Y=zeros(nump,1);
  for i=1:nump
    POINT_ID(i)=pid(numfid + i);
    temp=[pr(numfid + i); pc(numfid + i)] - [par(1); par(4)];
    cal=mx_inv*temp;
    POINT_X(i)=cal(1);
    POINT_Y(i)=cal(2);
    end
  end

% ===================================================================

ndx=1;
for i=1:numfid
  vr(i)=v(ndx);
  vc(i)=v(ndx+1);
  ndx=ndx+2;
  end

% put r,c residuals into x,y system

vy= -vr*mm_per_pix;
vx=  vc*mm_per_pix;

% scale up so that we can see something

exag=500.0;
RESID_X_PLOT=vx*exag;
RESID_Y_PLOT=vy*exag;
FID_X_PLOT=fx;
FID_Y_PLOT=fy;
FID_NUM_PLOT=fnum;
NUM_FID=numfid;

% list to the list box

A=cell(numfid+2,1);
s='      Residuals (mm)';
A(1)=cellstr(s);
s='Fid. ID      vX       vY';
A(2)=cellstr(s);
for i=1:numfid
  s=sprintf('%8d %8.3f %8.3f',fnum(i),vx(i),vy(i));
  A(i+2)=cellstr(s);
  end
set(handles.result_listbox,'string',A);


% --------------------------------------------------------------------
function varargout = graph_button_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.graph_button.

global RESID_X_PLOT;
global RESID_Y_PLOT;
global FID_X_PLOT;
global FID_Y_PLOT;
global FID_NUM_PLOT;
global NUM_FID;

%x=[1;2;3;4;5;6;7;8;9;10];
%y=[1;0;1;0;1;0;1;0;1;0];
%plot(x,y,'-');
```

```
x=[-125;-125;125;125;-125];
y=[-125;125;125;-125;-125];
plot(x,y,'b-','Linewidth',3);
hold on;
axis equal;
axis([-140 140 -140 140]);

clear x;
clear y;
x=zeros(2,1);
y=zeros(2,1);
for i=1:NUM_FID
  x(1)=FID_X_PLOT(i);
  y(1)=FID_Y_PLOT(i);
  x(2)=FID_X_PLOT(i) + 3;
  y(2)=y(1);
  plot(x,y,'b-','Linewidth',3);
  s=sprintf('%1d',FID_NUM_PLOT(i));
  text(FID_X_PLOT(i)+10,FID_Y_PLOT(i),s);
  end

for i=1:NUM_FID
  x(1)=FID_X_PLOT(i);
  y(1)=FID_Y_PLOT(i);
  x(2)=FID_X_PLOT(i) + RESID_X_PLOT(i);
  y(2)=FID_Y_PLOT(i) + RESID_Y_PLOT(i);
  plot(x,y,'r-','Linewidth',1);
  end
title('Residual Vectors');

% --------------------------------------------------------------------
function varargout = quit_button_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.quit_button.
% write data to outfile and quit

global POINT_ID;
global POINT_X;
global POINT_Y;
global NUM_POINT;

% you could correct here for lens distortion and atmospheric refraction
% or do in another program.

%test if outfile ok

outfile=get(handles.output_edit_text,'string');
[m,n]=size(outfile);
if(n < 1)
  disp('cannot write the output file');
else
  ofid=fopen(outfile,'wt');
  for i=1:NUM_POINT
    s=char(POINT_ID(i));
    fprintf(ofid,'%10s ',s);
    fprintf(ofid,'%10.3f %10.3f\n',POINT_X(i),POINT_Y(i));
    end
  fclose(ofid);
  end

closereq
```