ACTIVE OPTICAL SYSTEMS

AN010

Example of Working with AOS's .WFS Files in MATLAB

Author: **Justin D. Mansell, Ph.D.**
Active Optical Systems, LLC

Revision: **8/30/09**

In this application note, we show an example of how to work with our .WFS files in Matlab. This example was analysis of a set of .WFS files from the measurement of a 1 meter focal length lens on a 1550nm Shack-Hartmann Wavefront Sensor. Through this analysis we confirmed that the slopes being measured were not correct due to an improperly set threshold in the centroid operation. We also verified the wavefront curvature fitting section of the AOS Adaptive Optics code. This Application Note was written to eliminate any dependencies except the scripts that come with the latest versions of the AOS Adaptive Optics software. You may need to add the directory containing the Matlab scripts to your Matlab path in order to run this code. The entire script is in the appendix of this application note.

Reading and Displaying a .WFS File

The first section of this script loads and displays the data recorded by the AOS 1550nm Shack-Hartmann wavefront sensor. The code for this section is:

```
% This script is an example of analysis of some 1550nm WFS data taken while
% measuring a 1m focal length lens.  It does some basic processing and
% analysis on an AOS .WFS file
close all; clear all; clc; format compact;
f=1.0; %we took this data measuring a 1m focal length lens
fb='1000mm';

% read in the WFS data
[wfs,aois]=LoadWFS(sprintf('%s.wfs',fb));
pixelSize=wfs.pixSizeX; %this camera has square pixels
efl = wfs.separation;
figure; showWFS(aois); axis image; colorbar;


% convert the slope data from a vector into a 2D grid using the index
% parameters
for ii=1:length(aois);
    dx1(aois(ii).xindex+1,aois(ii).yindex+1) = aois(ii).dx;
    dy1(aois(ii).xindex+1,aois(ii).yindex+1) = aois(ii).dy;
end;
figure;
subplot(1,2,1); imagesc(dx1); axis image; colorbar;
```

```
title('X Slopes');
subplot(1,2,2); imagesc(dy1); axis image; colorbar;
title('Y Slopes');
```

The LoadWFS function reads from the .WFS file information about the wavefront sensor and the areas of interest or AOIs that define the section of the camera behind each of the Hartmann sub-apertures. The wfs structure in this file as read into Matlab contains the following fields:

```
wfs =
    separation: 0.0051
     threshold: 0.0000
      pixSizeX: 2.5000e-005
      pixSizeY: 2.5000e-005
         dxsub: 1.4992e-004
         dysub: 1.4959e-004
```

Unless otherwise specified, all units in these files are in meters-kilograms-seconds (mks). Separation is the distance from the Hartmann array to the camera. Threshold is the global relative threshold on a 0 to 1 scale. pixSizeX and pixSizeY are the pixel dimensions of the camera. dxsub and dysub are the average sub-aperture spacings in the two axes calculated from the AOI data.

The aoi data is a vector of structures containing information about the AOIs and measurements made in the AOIs. The first AOI structure contains the following information:

```
>> aois(1)
ans =
        x1: 4
        x2: 10
        y1: 2
        y2: 8
        xr: 6.5877
        yr: 4.5304
    xindex: 0
    yindex: 0
        xc: 7.1723
        yc: 4.9363
        dx: 0.0029
        dy: 0.0020
         I: 3.5929
         p: 0
```

x1, x2, y1, and y2 are the coordinates of the AOI boundaries in pixels. xr and yr are the reference spot positions. xindex and yindex are the zero-based indices for mapping to two-dimensions. xc and yc are the measured centroid locations. dx and dy are the measured wavefront slopes. I is the sum of the normalized pixel intensities above threshold. p is the wavefront phase.

This section of code loads the file “1000mm.wfs” and displays the slopes as a quiver plot, the phase as a false-color image, and the slopes as false-color images. The resulting graphs are as follows:

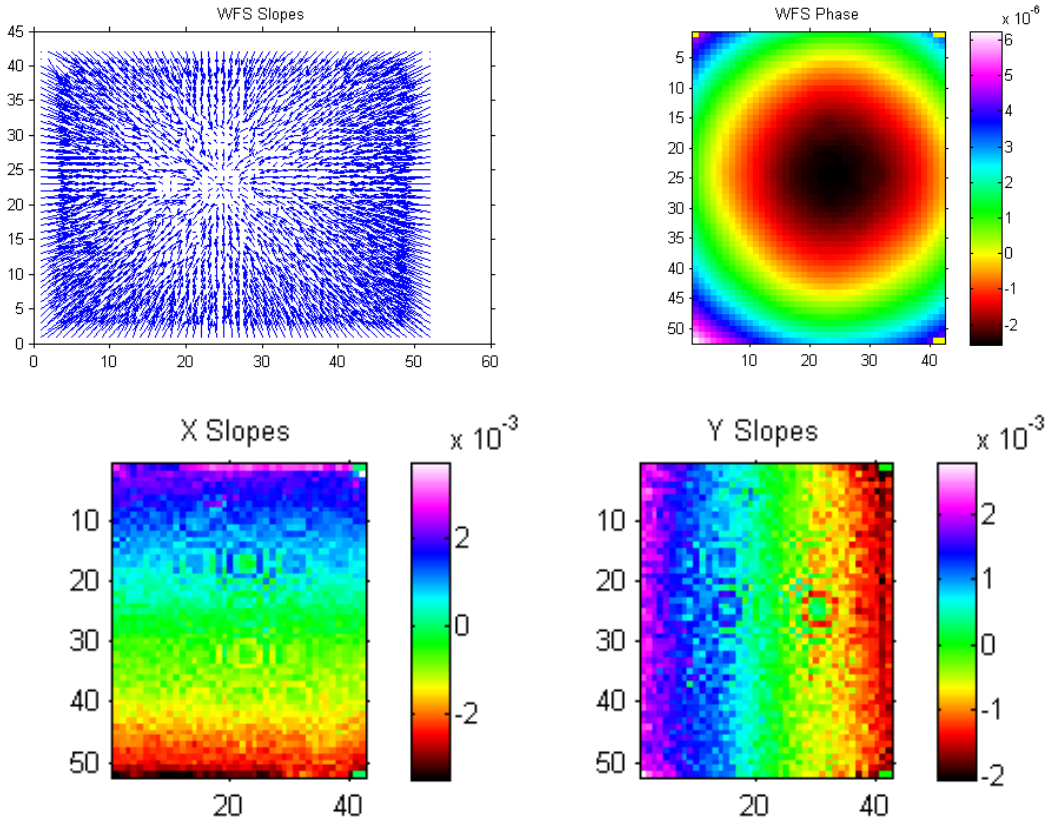


Figure 1 - Results read in from the .WFS file

Calculating Slopes based on Theory

We chose a lens for this analysis because it was easy to obtain and easy to calculate the correct solution base on theory. A collimated beam passing through an ideal lens will have a wavefront phase, ϕ, given by,

$$\phi = \frac{r^2}{2f},$$

where r is the lateral radius and f is the lens focal length. The wavefront slopes in the x and y directions are then given by,

$$\frac{\partial \phi}{\partial x} = \frac{x}{f} \text{ and } \frac{\partial \phi}{\partial y} = \frac{y}{f}.$$

We apply these formulas in the next section of code to calculate the ideal slopes using the lateral dimensions given by the xr and yr fields in the aois structure. The following code calculates the slopes for a 1-m lens based on theory:

```
%% calculate the slopes based on theory
aois2=aois;
xc = 147; % determined imperically
yc = 135; % determined imperically
for ii=1:length(aois);
    aois2(ii).dx = -(aois(ii).xr-xc).*pixelSize./f;
    aois2(ii).dy = -(aois(ii).yr-yc).*pixelSize./f;
```

```
    r = sqrt((aois(ii).xr-xc).^2 + (aois(ii).yr-yc).^2).*pixelSize;
    aois2(ii).p = r.^2 ./ (2*f);
end;

% show the theory-calcualted WFS data
figure; showWFS(aois2);

% convert the theory calculated WFS data into a 2D array
for ii=1:length(aois);
    dx2(aois2(ii).xindex+1,aois2(ii).yindex+1) = aois2(ii).dx;
    dy2(aois2(ii).xindex+1,aois2(ii).yindex+1) = aois2(ii).dy;
end;
sf = PV(dx2) ./ PV(dx1)

figure;
subplot(1,2,1); imagesc(dx2); axis image; colorbar;
title('Calculated X Slopes');
subplot(1,2,2); imagesc(dy2); axis image; colorbar;
title('Calculated Y Slopes');
```

This section of code produces the following plots of the theoretical answers. When comparing the magnitude of the slopes, it is clear that there is a difference in magnitude from the measurements.

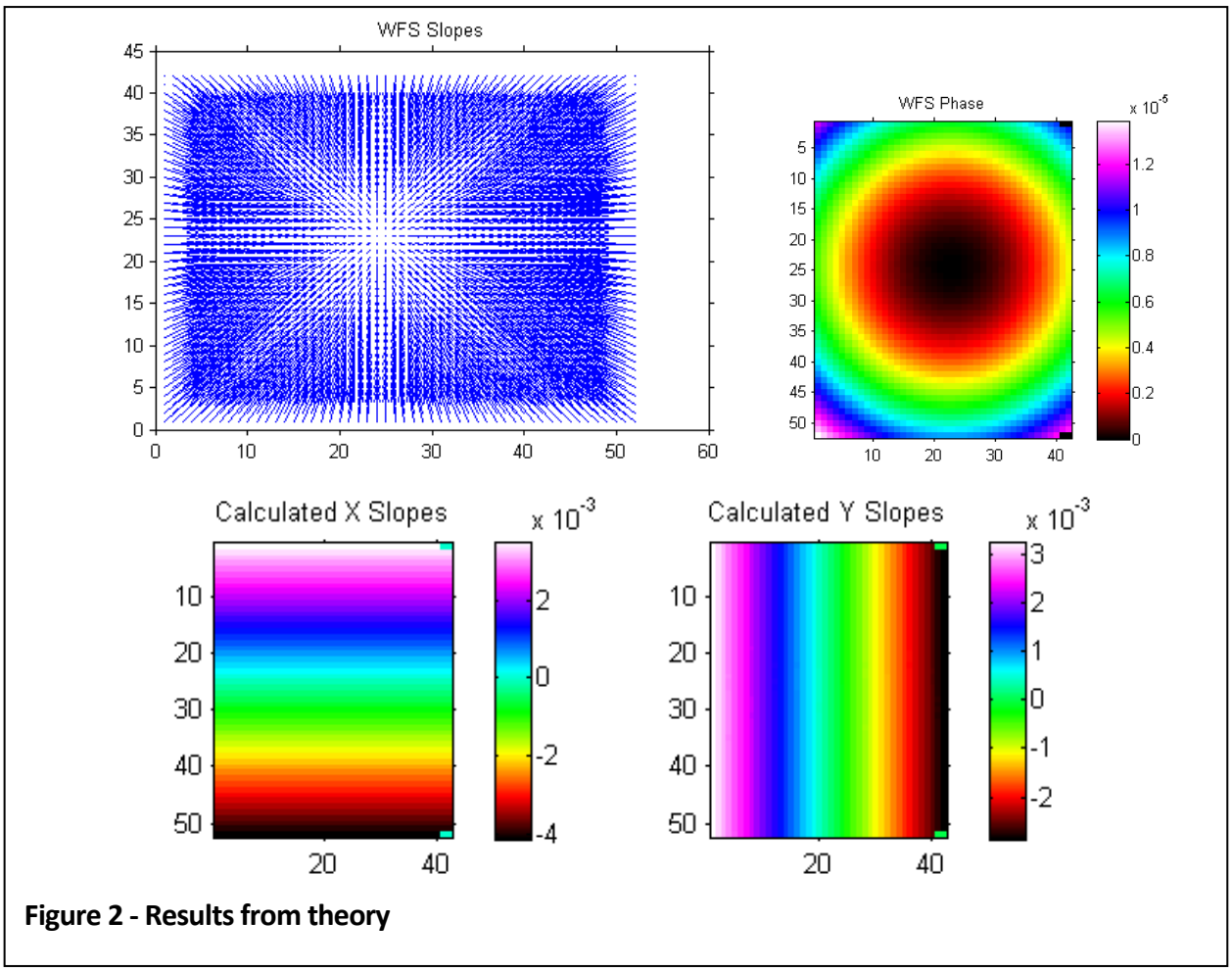


Figure 2 - Results from theory

Comparing Measured and Theoretical Phase

The next section of code compares the measured and ideal reconstructed phase.

```
%% now look at the measured phase
%convert the phase into a 2D array
for ii=1:length(aois);
    z1(aois(ii).xindex+1,aois(ii).yindex+1) = aois(ii).p;
    z2(aois2(ii).xindex+1,aois2(ii).yindex+1) = aois2(ii).p;
    I(aois2(ii).xindex+1,aois2(ii).yindex+1) = aois2(ii).I;
end;
figure;
subplot(1,3,1); imagesc(z1); axis image; colorbar; title('Lab Measured');
subplot(1,3,2); imagesc(z2); axis image; colorbar; title('Theory
Calcualted');
dz = z1-z2; dz=dz-mean(dz(:)); dz=dz.*(I>0); %remove the mean and zero the
low intensities
subplot(1,3,3); imagesc(dz); axis image; colorbar; title('Lab - Theory');
```

The results from this section are shown below. The theoretical phase is larger than the lab measurement by a factor of about 8.5 µm to 14 µm or about 1.6x.

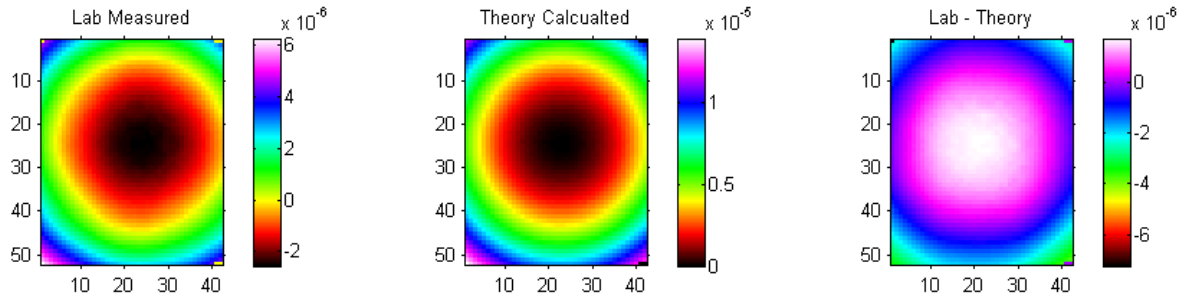


Figure 3 - Comparison of the measured and theoretical wavefront phase.

Analysis of the Ratio of Spot Motion in Pixels to Measured Slopes

The next code was written to analyze the ratio between the centroid motion in pixels and the calculated slopes.

```
%% analyze scale factors
%look at the ratio of the centroid shift to the reported slopes to make
%sure the paramters are correct
ratioExpected = efl/pixelSize
c=1;
for ii=1:length(aois);
    if (aois(ii).I>0)
        deltaxpix(ii) = aois(ii).xc-aois(ii).xr;
        deltaypix(ii) = aois(ii).yc-aois(ii).yr;
        slopex(ii) = aois(ii).dx;
        slopey(ii) = aois(ii).dy;
```

```
        ratiox(c) = deltaxpix(ii)./slopex(ii);
        ratioy(c) = deltaypix(ii)./slopey(ii);
        c=c+1;
    end;
end;
AverageRatios=[mean(ratiox) mean(ratioy)]
```

The results from the Matlab command window are shown here:

```
ratioExpected =
   204
AverageRatios =
  204.0000  204.0000
```

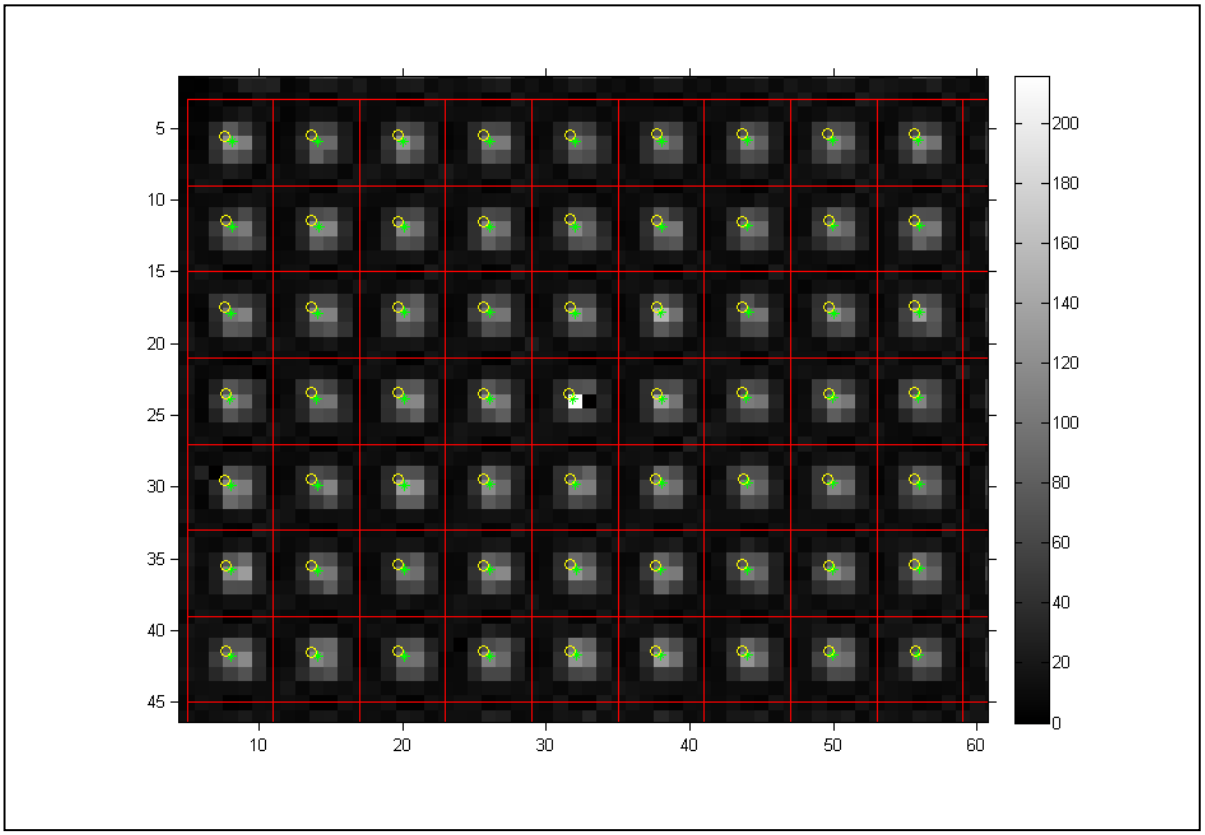
This result shows that the problem was not associated with any problem with improperly set Hartmann array to camera separation or pixel size.

Examination of the Image Overlaid with AOI Information

In this next section of code we load the Hartmann sensor image and overlay the AOIs in red, centroids as green asterisks, and original reference spot positions as yellow circles.

```
%% look at the image & overlay AOIs
%make the measured centroids green stars and
% the reference locations yellow circles
img=imread(sprintf('%s.tif',fb));
img=double(img(:,:,1));
figure; imagesc(img); colormap(gray); axis image; colorbar;
for ii=1:length(aois)
    hold on;
    x1=aois(ii).x1;
    x2=aois(ii).x2;
    y1=aois(ii).y1;
    y2=aois(ii).y2;
    plot([x1 x2 x2 x1 x1]+1,[y1 y1 y2 y2 y1]+1,'r-');
    plot(aois(ii).xc+1,aois(ii).yc+1,'g*');
    plot(aois(ii).xr+1,aois(ii).yr+1,'yo');
end;
slopeMax = (sqrt(size(img,1).^2+size(img,2).^2)*pixelSize/2) / f
PV_WF = ((sqrt(size(img,1).^2+size(img,2).^2)*pixelSize).^2/4) / (2*f)
```

The upper left section of the image is shown below. Examination of the image showed some unusual pixel gains (see x=32, y=24), but a generally good trend of spot motion away from the center of the image. The pixel values that are clearly not part of the focal spots are reporting fairly high magnitudes. There are some that are showing ~40 counts out of 255 in this image. When there is this high of a background in an image, we usually see a trend of spot centroids being closer to the center of the AOIs, thus resulting in a lower slope. We confirmed this by going back to the AOS software and reanalyzing the data with a larger global threshold (10%). This analysis produced results that were much closer to the known lens focal length of 1.0 meters.



The last two lines of this code calculate the ideal max slope magnitude and max wavefront phase amplitude (aka sagittus) if the beam were perfectly centered on the wavefront sensor. (Note that in the previous theoretical calculations, we found the slopes in the two axes, but not the maximum radial slope magnitude.) The theoretical peak-to-valley center wavefront amplutide of 13.1 µm is much closer to the ~14 µm generated before. The difference is due to the center of the wavefront not being centered on the camera. The Matlab command window results are shown here:

```
slopeMax =
    0.0051
PV_WF =
  1.3120e-005
```

Fitting the Slope to a Plane and Extracting Wavefront Curvature

Even though this data was now known to be faulty due to the low threshold on the centroiding, we can still use it to verify the AOS software's wavefront radius of curvature determination algorithm. To do this we fit the slopes in the two axes with a plane and then calculated the curvature in each axis as the inverse of the linear slope coefficient or

$$\frac{\partial \phi}{\partial x} = \frac{1}{ROC} x = mx + b \rightarrow ROC = \frac{1}{m}.$$

The following code was used for the fitting.

```
%% fit the slopes to a plane
%since the input wavefront is primarily curvature, we can fit the
%wavefront slopes with a plane to determine the curvature
c=1;
```

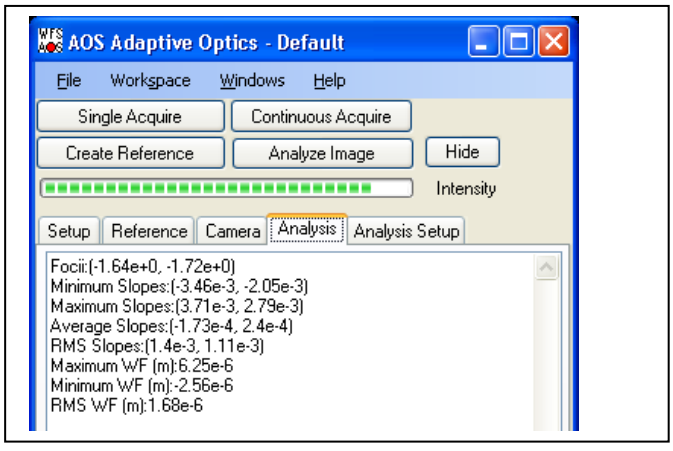
```
for ii=1:length(aois)
    if (aois(ii).I>0)
        Mx(c,:)=[aois(ii).xc*pixelSize 1];
        My(c,:)=[aois(ii).yc*pixelSize 1];
        dxv(c)=aois(ii).dx;
        dyv(c)=aois(ii).dy;
        c=c+1;
    end;
end;
Mxinv=pinv(Mx);
Myinv=pinv(My);
xcoef = Mxinv * dxv(:)
ycoef = Myinv * dyv(:)
EstimatedROCs = 1./[xcoef(1) ycoef(1)]

% the slope fits
aois3=aois;
for ii=1:length(aois)
    aois3(ii).dx = xcoef(1).*aois(ii).xc.*pixelSize + xcoef(2);
    aois3(ii).dy = ycoef(1).*aois(ii).yc.*pixelSize + ycoef(2);
    % convert to 2D
    dx3(aois3(ii).xindex+1,aois3(ii).yindex+1) = aois3(ii).dx;
    dy3(aois3(ii).xindex+1,aois3(ii).yindex+1) = aois3(ii).dy;
end;
figure;
subplot(1,2,1); imagesc(dx3); axis image; colorbar; title('Fit Slopes X');
subplot(1,2,2); imagesc(dy3); axis image; colorbar; title('Fit Slopes Y');

% show the difference between the measured and fit slopes
figure;
subplot(1,2,1); imagesc(dx3-dx1); axis image; colorbar; title('Fit Slopes -
Measured Slopes X');
subplot(1,2,2); imagesc(dy3-dy1); axis image; colorbar; title('Fit Slopes -
Measured Slopes Y');
```

The results from this section of the script are shown below with a screen-shot from the AOS software showing that the curvature estimated by the AOS software and determined from the fits to the wavefront slope data in Matlab are the same.

```
xcoef =
   -0.6087
    0.0023
ycoef =
   -0.5825
    0.0021
EstimatedROCs =
   -1.6427   -1.7167
```



Conclusions

This analysis in Matlab eventually showed us that we had a problem identifying the spot positions in the lab due to an improperly set threshold. It verified that the AOS software was calculating the wavefront curvature based on a fit to the measured wavefront slope data. Most importantly, this application note illustrates how AOS .WFS files can be worked with in Matlab.

Appendix: Complete Matlab Code

```
% This script is an example of analysis of some 1550nm WFS data taken while
% measuring a 1m focal length lens.  It does some basic processing and
% analysis on an AOS .WFS file
close all; clear all; clc; format compact;
R=1.0; %we took this data measureing a 1m focal length lens
fb='1000mm';

% read in the WFS data
[wfs,aois]=LoadWFS(sprintf('%s.wfs',fb));
pixelSize=wfs.pixSizeX; %this camera has square pixels
efl = wfs.separation;
figure; showWFS(aois); axis image; colorbar;

% convert the slope data from a vector into a 2D grid using the index
% parameters
for ii=1:length(aois);
    dx1(aois(ii).xindex+1,aois(ii).yindex+1) = aois(ii).dx;
    dy1(aois(ii).xindex+1,aois(ii).yindex+1) = aois(ii).dy;
end;
figure;
subplot(1,2,1); imagesc(dx1); axis image; colorbar;
title('X Slopes');
subplot(1,2,2); imagesc(dy1); axis image; colorbar;
title('Y Slopes');

%% calculate the slopes based on theory
aois2=aois;
xc = 147; % determined imperically
yc = 135; % determined imperically
for ii=1:length(aois);
    aois2(ii).dx = -(aois(ii).xr-xc).*pixelSize./R;
    aois2(ii).dy = -(aois(ii).yr-yc).*pixelSize./R;
    r = sqrt((aois(ii).xr-xc).^2 + (aois(ii).yr-yc).^2).*pixelSize;
    aois2(ii).p = r.^2 ./ (2*R);
end;

% show the theory-calcualted WFS data
figure; showWFS(aois2);

% convert the theory calculated WFS data into a 2D array
for ii=1:length(aois);
    dx2(aois2(ii).xindex+1,aois2(ii).yindex+1) = aois2(ii).dx;
    dy2(aois2(ii).xindex+1,aois2(ii).yindex+1) = aois2(ii).dy;
end;
sf = PV(dx2) ./ PV(dx1)

figure;
subplot(1,2,1); imagesc(dx2); axis image; colorbar;
title('Calculated X Slopes');
subplot(1,2,2); imagesc(dy2); axis image; colorbar;
title('Calculated Y Slopes');

%% now look at the measured phase
%convert the phase into a 2D array
for ii=1:length(aois);
    z1(aois(ii).xindex+1,aois(ii).yindex+1) = aois(ii).p;
    z2(aois2(ii).xindex+1,aois2(ii).yindex+1) = aois2(ii).p;
    I(aois2(ii).xindex+1,aois2(ii).yindex+1) = aois2(ii).I;
end;
figure;
subplot(1,3,1); imagesc(z1); axis image; colorbar; title('Lab Measured');
subplot(1,3,2); imagesc(z2); axis image; colorbar; title('Theory Calcualted');
dz = z1-z2; dz=dz-mean(dz(:)); dz=dz.*(I>0); %remove the mean and zero the low intensities
subplot(1,3,3); imagesc(dz); axis image; colorbar; title('Lab - Theory');

%% analyze scale factors
%look at the ratio of the centroid shift to the reported slopes to make
%sure the paramters are correct
ratioExpected = efl/pixelSize
c=1;
for ii=1:length(aois);
    if (aois(ii).I>0)
        deltaxpix(ii) = aois(ii).xc-aois(ii).xr;
        deltaypix(ii) = aois(ii).yc-aois(ii).yr;
        slopex(ii) = aois(ii).dx;
```

```
        slopey(ii) = aois(ii).dy;

        ratiox(c) = deltaxpix(ii)./slopex(ii);
        ratioy(c) = deltaypix(ii)./slopey(ii);
        c=c+1;
    end;
end;
AverageRatios=[mean(ratiox) mean(ratioy)]

%% look at the image & overlay AOIs
%make the measured centroids green stars and
% the reference locations yellow circles
img=imread(sprintf('%s.tif',fb));
img=double(img(:,:,1));
figure; imagesc(img); colormap(gray); axis image; colorbar;
for ii=1:length(aois)
    hold on;
    x1=aois(ii).x1;
    x2=aois(ii).x2;
    y1=aois(ii).y1;
    y2=aois(ii).y2;
    plot([x1 x2 x2 x1 x1]+1,[y1 y1 y2 y2 y1]+1,'r-');
    plot(aois(ii).xc+1,aois(ii).yc+1,'g*');
    plot(aois(ii).xr+1,aois(ii).yr+1,'yo');
end;
slopeMax = (sqrt(size(img,1).^2+size(img,2).^2)*pixelSize/2) / R
PV_WF = ((sqrt(size(img,1).^2+size(img,2).^2)*pixelSize).^2/4) / (2*R)

%% fit the slopes to a plane
%since the input wavefront is primarily curvature, we can fit the
%wavefront slopes with a plane to determine the curvature
c=1;
for ii=1:length(aois)
    if (aois(ii).I>0)
        Mx(c,:)=[aois(ii).xc*pixelSize 1];
        My(c,:)=[aois(ii).yc*pixelSize 1];
        dxv(c)=aois(ii).dx;
        dyv(c)=aois(ii).dy;
        c=c+1;
    end;
end;
Mxinv=pinv(Mx);
Myinv=pinv(My);
xcoef = Mxinv * dxv(:)
ycoef = Myinv * dyv(:)
EstimatedROCs = 1./[xcoef(1) ycoef(1)]

% the slope fits
aois3=aois;
for ii=1:length(aois)
    aois3(ii).dx = xcoef(1).*aois(ii).xc.*pixelSize + xcoef(2);
    aois3(ii).dy = ycoef(1).*aois(ii).yc.*pixelSize + ycoef(2);
    % convert to 2D
    dx3(aois3(ii).xindex+1,aois3(ii).yindex+1) = aois3(ii).dx;
    dy3(aois3(ii).xindex+1,aois3(ii).yindex+1) = aois3(ii).dy;
end;
figure;
subplot(1,2,1); imagesc(dx3); axis image; colorbar; title('Fit Slopes X');
subplot(1,2,2); imagesc(dy3); axis image; colorbar; title('Fit Slopes Y');

% show the difference between the measured and fit slopes
figure;
subplot(1,2,1); imagesc(dx3-dx1); axis image; colorbar; title('Fit Slopes - Measured Slopes X');
subplot(1,2,2); imagesc(dy3-dy1); axis image; colorbar; title('Fit Slopes - Measured Slopes Y');
```