ACTIVE OPTICAL SYSTEMS

AN008

Fitting a Surface with Influence Functions

Author: **Justin D. Mansell, Ph.D.**
Active Optical Systems, LLC
Revision: **12/26/08**

In this application note, we explain how a set of influence functions can be used to create a known surface shape. We will do this here by way of an example piece of Matlab code. The entire script is in an appendix at the end of this note, but we will walk through it a section at a time here to explain what it is doing.

Setup and Showing the DM Actuator Pattern

This section of the script initializes Matlab and shows the influence function shape. The routines LoadDM() and ShowDM() are available from AOS. They load a .DM file into a structure array and display the array respectively. Figure 1 shows the result of the ShowDM() function.

```
%this script fits an arbitrary
shape with DM influence functions
close all; clear all; clc;
%rand('seed',123);
limiting=0;
showIFs=0;

%load the DM configuration
dm = LoadDM('MDM1-32S-001.dm');
figure;
ShowDM(dm); axis image;
```

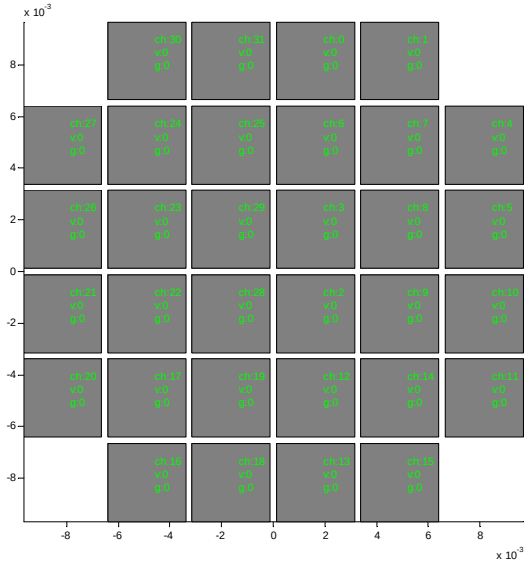

Figure 1 - Results of the ShowDM() function in Matlab

Load the Influence Functions

This section of code loads the influence functions generated by the AOS DMController code. The user must specify some parameters of the DM here including the maximum peak-to-valley motion of the DM (also known as the focal throw), the diameter of the DM in meters, the wavelength of the test light in meters, and the amount of the DM that is being used for fitting (dmScaleFactor), which in this case is the recommended central 80%. The influence functions are stored in a cell array called IF. The user has the option of displaying the influence functions by setting

the showIFs variable to a non-zero value above.

```
%load influence functions
files=dir('outputIF*.txt');
PVdm = 10e-6;
Ddm = 25.4e-3;
wavelength = 0.633e-6;
dmScaleFactor=0.8;
figure;
for ii=1:length(files);
    IF{ii}=csvread(files(ii).name);
    nxy = min([size(IF{ii},1)
size(IF{ii},2)]);
    IF{ii} = IF{ii}(1:nxy,1:nxy);
    if (ii==1)
        nxy = size(IF{1},1);
        dxy = Ddm/(nxy-4);
        x = (-nxy/2:1:nxy/2-1)*dxy;
    end;
    if (showIFs)
        clf;
        imagesc(x,x,IF{ii}); axis
image; colorbar;
        title(ii); pause(0.001);
    end;
end;
```

Create Apertures and Scale IFs

This next section of code uses the information about the size of the DM and the mesh spacing to create a 2D grid of radii and then that is used to create two apertures: one of the DM itself (apDM) and the other the fit aperture (apFit). Then the influence functions are scaled so that the peak to valley sum of all the influence functions is 10 microns, the normal throw of a 25.4 mm diameter membrane DM.

```
%create a DM aperture and a fit
aperture
[xx,yy]=meshgrid(x,x);
r = sqrt(xx.^2+yy.^2);
apDM = (r<=Ddm/2);
apFit = (r<=Ddm/2*dmScaleFactor);

%scale the DM IFs so that the sum
of all of them is equal in
amplitude to
%10 waves
dm = 0.0 .* IF{1};
for ii=1:length(IF); dm=dm+IF{ii};
end;
PV = max(max(dm)) - min(min(dm));
for ii=1:length(IF);
IF{ii}=IF{ii}./PV.*PVdm; end;
```

Create a Sample Grid

The next section of code creates a set of samples over the used DM aperture for fitting. For this code, it is assumed that it is a regular square grid of points, but almost any grid with enough points will work. The user specifies the number of samples per axis over the DM aperture (nxySample). The code then creates a 2D array of sample points with one at the center and removes all the points outside the used area of the DM. The code checks to see if there are sufficient points to continue and then shows the points overlaid on a DM influence function (Figure 2). Finally, the code finds the 2D grid points closest to the real-number sample coordinates.

```
%create a sample grid
nxySample = 14;
dxySample = Ddm/(nxySample);
xs=(-nxySample/2:1:nxySample/2-
1)*dxySample;
[xxs,yys]=meshgrid(xs,xs);
rs=sqrt(xxs.^2+yys.^2);
rSample = rs(:);
xSample = xxs(:); ySample = yys(:);
ind =
find(rSample<Ddm/2*dmScaleFactor);
xSample = xSample(ind);
ySample = ySample(ind);

%check if enough samples
if (length(xSample)<length(IF))
    error('Not enough samples for
fitting');
end;

%show the sample grid on top of the
first IF
figure;
imagesc(x,x,IF{1});
hold on;
```

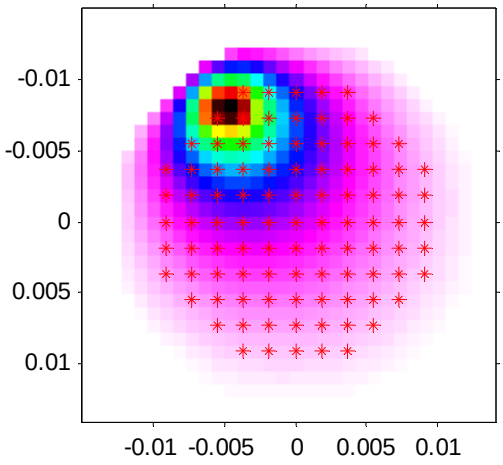

Figure 2 - Sample points as red asterisks overlaid on the first DM influence function

```
plot(xSample,ySample,'r*');

%find the coordinates of samples
for ii=1:length(xSample)
    [mv,xi(ii)] = min(abs(x-
xSample(ii)));
    [mv,yi(ii)] = min(abs(x-
ySample(ii)));
end;
```

Creating Poke and Control Matrices

This next section of code creates the poke and control matrices. In this example, we are using phase control, but in most AO applications slope control is actually used since the wavefront sensor is usually a Hartmann sensor and the absolute phase is not known. This script can be easily modified to do slope control instead by taking the gradient of the surfaces and operating on it instead of operating on the surface itself.

The code starts by creating a poke matrix (poke) that is a sample of each of the influence functions at each of the sample points specified above and displaying it. Then a pseudo-least-squares inverse of the poke matrix is generated using a single-value-decomposition (SVD) technique. This was done instead of the Matlab function pinv() because this technique allows us to easily remove low-gain modes if necessary. The script plots the svd gains. In this example, there was not a precipitous drop in the curve, so we left all the modes in,

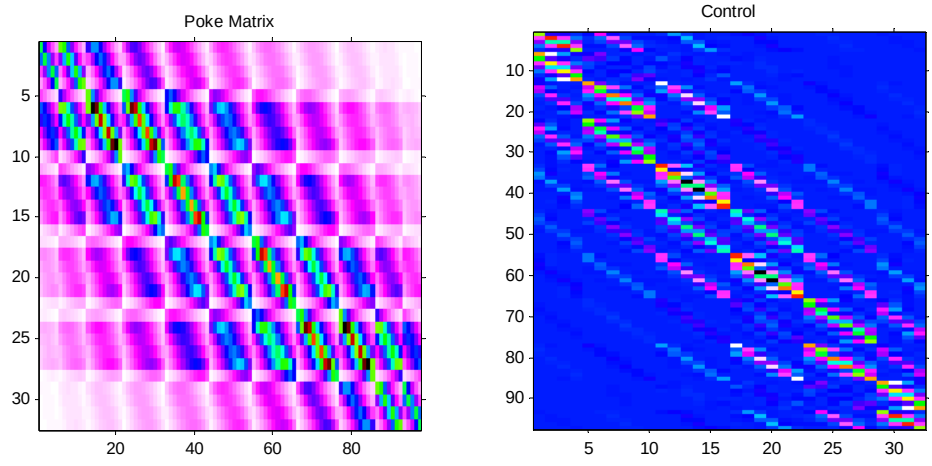

Figure 3 - Poke and Control Matrices

(modesRemoved=0), but left the code in to allow the user to change this easily. The inverted poke matrix is then stored (control) and displayed (Figure 3).

```
%sample the influence functions to
create a poke matrix
for ii=1:length(IF)
    for jj=1:length(xi)
        poke(ii,jj) =
IF{ii}(xi(jj),yi(jj));
    end;
end;
figure;
imagesc(poke); title('Poke
Matrix');

%invert the poke matrix to create
the control matrx
[u,ss,v]=svd(poke);
sv=diag(ss); gains=sv;
svi = 1.0./sv;
si = zeros(size(ss,2),size(ss,1));
%show the svd gains
figure; semilogy(sv,'*b-');
title('svd gains-AOA recon');

%remove some modes
modesRemoved = 0;
for ii=1:size(svi,1)-modesRemoved;
    si(ii,ii) = svi(ii);
end;
control=v*si*u';

%show the control matrix
figure; imagesc(control);
title('Control');
```

Creating an Example Phase Screen

In this example, we created our own phase screen based on Kolmogorov turbulence theory. The screen was generated based on a PSD function that was found on p. 35 of Robert Tyson's book Principles of Adaptive Optics (equation 2.28). The user can specify the aperture diameter and the D/r0 (Dap_over_r0) and the phase screen is generated. In the code in the appendix we left in some commented-out code that is useful for visualizing the results. For an end-user of this code, this is the point at which you might load the desired phase screen. Figure 4 shows an example phase screen generated from this script. To generate a consistent phase screen, remove the comment from the rand() function that seeds the random number generator in the first section of code.

```
%% create a phase screen
Dap = 30.0e-2;  Dap_over_r0=2;
r0 = Dap/Dap_over_r0;
nxy = 128; dxy = Dap/nxy;
xp = (-nxy/2:1:nxy/2-1)*dxy;
[xxp,yyp]=meshgrid(xp,xp);
rp = sqrt(xxp.^2+yyp.^2);
K = fftshift(rp);
PHIp = (0.023/r0.^(5/3)) .* K.^(-
11/3);
PHIp(1,1)=0;
PHI = sqrt(PHIp) .*
exp(j*2*pi*rand(nxy,nxy));
phi = ifft2((PHI));
phase = real(phi) ./ (2*pi) .*
wavelength;
figure; imagesc(xp,xp,phase); axis
image; title('Phase (waves)');
colorbar;
hold on;
theta = 0:0.1:2.0*pi;
xcircle = Dap/2 * cos(theta);
ycircle = Dap/2 * sin(theta);
plot(xcircle,ycircle,'r--
','linewidth',2');
```

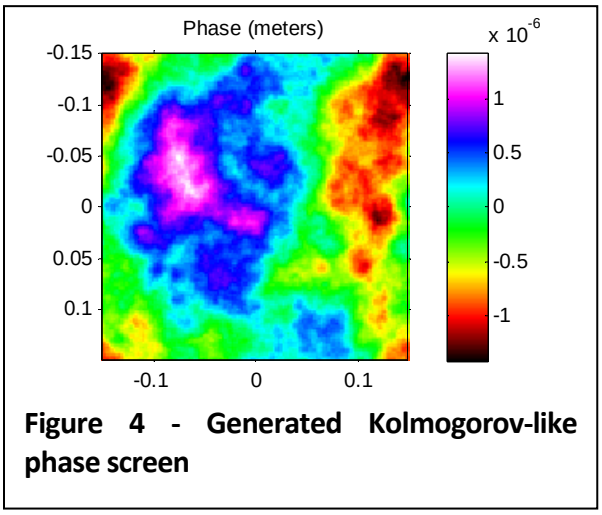

Figure 4 - Generated Kolmogorov-like phase screen

Sampling the Desired Shape

This next section of code puts the desired shape on the same grid as the influence functions and then samples it using the same sample grid generated above. We removed the mean over the fit area to avoid any significant piston term causing a large actuator throw. We also add in here potential magnification between the DM and the output aperture (Magnification).

```
%% sample the desired shape with
magnification
Magnification = Dap / Ddm;
shape =
interp2(xp./Magnification,(xp./Magn
ification)',phase,x,(x)');
%interpolate onto the same grid
shape(find(isnan(shape)==1))=0;

%remove the mean over the fit
aperture area
shapeFit = shape .* apFit;
shapeVec = shapeFit(:);
ind = find(shapeVec~=0);
shape = shape -
mean(shapeVec(ind));
shape = shape .* apFit;

%sample it on the desired grid
for jj=1:length(xi)
    vec(jj) = shape(xi(jj),yi(jj));
end;
```

Generating DM Commands

This next section of code takes the sampled shape as a vector and multiplies it by the control matrix generated earlier to create an vector of actuator forces or commands. The commands are then displayed as a bar chart (Figure 5).

To simulate throw limitations of the DM, we assumed a 50% throw bias and added code to limit the DM force range to ±50%. Then we display the desired forces, the limited forces, and the difference between them. In this case, we set limiting = 0 in the first section of code, so no actuator limiting was used.

```
%fit to the desired shape
commands = control' * vec';
figure; bar(commands);
title('Commands');

if (limiting)
    %implement limiting at -0.5 to
0.5
    c0 = commands;
    commands = c0 .* (c0>-0.5) +
(c0<-0.5).*-0.5;
    commands = commands .* (c0<0.5)
+ (c0>0.5).*0.5;
    figure; bar([c0 commands c0-
commands]); title('Limited
Commands'); colormap(jet);

legend('requested','limited','diffe
rence');
end;
```

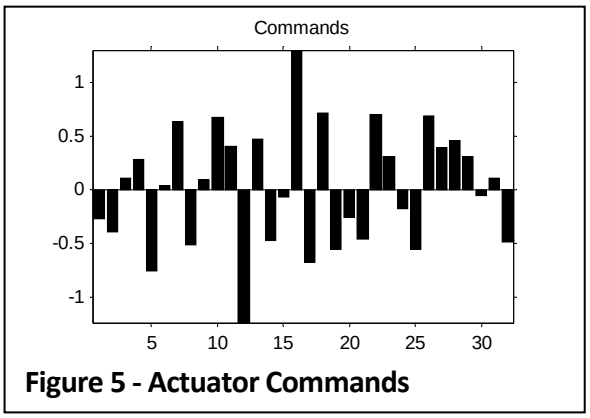

Figure 5 - Actuator Commands

Constructing the DM Surface and Analysis

This final section of code constructs the DM surface by doing a sum of the influence functions weighted by the commands and then displays and analyzes the result. The next section of the code saves the output to a new DM file that can be loaded into the AOS software. The line commented out at the end

of the file saving allows the user to read in the output file and display it for verification.

In this case, we found that the phase over the DM area was 540 nm rms. After subtracting the DM surface, we the rms difference was 154 nm. Using the Marechal approximation (Strehl Ratio ≈ $\exp(-\phi_{rms\_radians}^2)$), we calculated the Strehl ratio to be 3e-13 uncompensated and 9.5% compensated at a 633 nm wavelength (the Marechal approximation assumptions are clearly invalid for the initial case).

Figure 6 shows the results generated in this section. The white circles are the sample point locations. The difference plot is aperture with the fit aperture (apFit) generated above.

```
%create the resulting DM shape
dm=0.0 .* IF{1};
for ii=1:length(IF);
dm=dm+IF{ii}.*commands(ii); end;

%save the output DM file
maxCounts = 255;
dmf2 = dmf;
cmdAverage = mean(commands);
for ii=1:length(IF);
    dmf2(ii).v =
round((commands(ii) - cmdAverage +
0.5) .* maxCounts);
    if (dmf2(ii).v>maxCounts)
dmf2(ii).v=maxCounts; end;
    if (dmf2(ii).v<0) dmf2(ii).v=0;
end;
end;
SaveDM('MDM1-32S-001-fit.dm',dmf2);
% this line tests the file output
by displaying the re-read file
% dm2=LoadDM('MDM1-32S-001-
fit.dm'); nf; ShowDM(dm2);


%show the resulting DM shape
figure;
subplot(1,3,1); imagesc(x,x,shape
.* apDM); axis image; colorbar;
title('Desired Shape');
hold on; plot(x(xi),x(yi),'wo');
cax = caxis;
subplot(1,3,2); imagesc(x,x,dm);
axis image; colorbar; title('DM
Shape');
hold on; plot(x(xi),x(yi),'wo');
caxis(cax); colorbar;
subplot(1,3,3); imagesc(x,x,apFit
.* (shape - dm)); axis image;
colorbar; title('Difference');
hold on; plot(x(xi),x(yi),'wo');
caxis(cax); colorbar;

%calculate RMS wavefront error and
estimated Strehl ratio
delta = apFit .* (shape - dm);
dv=delta(:); ind = find(dv~=0); dvs
= dv(ind); %look at all the non-
zero elements in vector form
rmswfe_m=sqrt(sum((dvs-
mean(dvs)).^2)/length(ind))
rmswfe_rads = rmswfe_m / wavelength
* 2 * pi
SR = exp(-1.0 * (rmswfe_rads).^2)

%look at what the Strehl ratio
would be without compensation
delta = apFit .* (shape);
dv=delta(:); ind = find(dv~=0); dvs
```

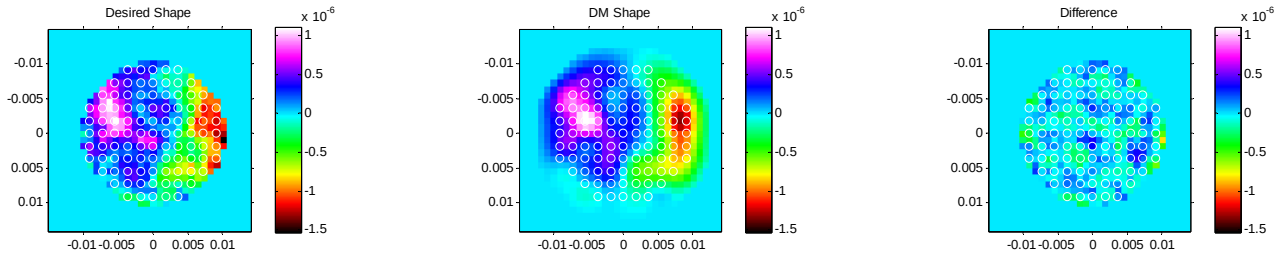

Figure 6 - Result of fitting the phase screen with DM influence functions.

```
= dv(ind); %look at all the non-
zero elements in vector form
rmswfe_m_uncomp=sqrt(sum((dvs-
mean(dvs)).^2)/length(ind));
rmswfe_rads_uncomp =
rmswfe_m_uncomp / wavelength * 2 *
pi;
SR_uncomp = exp(-1.0 *
(rmswfe_rads_uncomp).^2)
```

Conclusions

In this application note, we showed how to fit an arbitrary surface using DM influence functions. We did the fitting in phase space in this script, but it can easily be modified to work in gradient (slope) space or even in Laplacian (curvature) space. This same idea can be also easily extended to performing adaptive optics.

Appendix A: Entire Matlab Code

```
%this script fits an arbitrary shape with DM influence functions
close all; clear all; clc;
%rand('seed',123);
limiting=0;
showIFs=0;

%load the DM configuration
dm = LoadDM('MDM1-32S-001.dm');
figure;
ShowDM(dm); axis image;

%load influence functions
files=dir('outputIF*.txt');
PVdm = 10e-6;
Ddm = 25.4e-3;
wavelength = 0.633e-6;
dmScaleFactor=0.8;
figure;
for ii=1:length(files);
    IF{ii}=csvread(files(ii).name);
    nxy = min([size(IF{ii},1) size(IF{ii},2)]);
    IF{ii} = IF{ii}(1:nxy,1:nxy);
    if (ii==1)
        nxy = size(IF{1},1);
        dxy = Ddm/(nxy-4);
        x = (-nxy/2:1:nxy/2-1)*dxy;
    end;
    if (showIFs)
        clf;
        imagesc(x,x,IF{ii}); axis image; colorbar;
        title(ii); pause(0.001);
    end;
end;

%create a DM aperture and a fit aperture
[xx,yy]=meshgrid(x,x);
r = sqrt(xx.^2+yy.^2);
apDM = (r<=Ddm/2);
apFit = (r<=Ddm/2*dmScaleFactor);

%scale the DM IFs so that the sum of all of them is equal in amplitude to
%10 waves
dm = 0.0 .* IF{1};
for ii=1:length(IF); dm=dm+IF{ii}; end;
PV = max(max(dm)) - min(min(dm));
for ii=1:length(IF); IF{ii}=IF{ii}./PV.*PVdm; end;

%create a sample grid
nxySample = 14;
dxySample = Ddm/(nxySample);
xs=(-nxySample/2:1:nxySample/2-1)*dxySample;
[xxs,yys]=meshgrid(xs,xs);
rs=sqrt(xxs.^2+yys.^2);
```

```
rSample = rs(:);
xSample = xxs(:); ySample = yys(:);
ind = find(rSample<Ddm/2*dmScaleFactor);
xSample = xSample(ind);
ySample = ySample(ind);

%check if enough samples
if (length(xSample)<length(IF))
    error('Not enough samples for fitting');
end;

%show the sample grid on top of the first IF
figure;
imagesc(x,x,IF{1});
hold on;
plot(xSample,ySample,'r*');

%find the coordinates of samples
for ii=1:length(xSample)
    [mv,xi(ii)] = min(abs(x-xSample(ii)));
    [mv,yi(ii)] = min(abs(x-ySample(ii)));
end;

%sample the influence functions to create a poke matrix
for ii=1:length(IF)
    for jj=1:length(xi)
        poke(ii,jj) = IF{ii}(xi(jj),yi(jj));
    end;
end;
figure;
imagesc(poke); title('Poke Matrix');

%invert the poke matrix to create the control matrx
[u,ss,v]=svd(poke);
sv=diag(ss); gains=sv;
svi = 1.0./sv;
si = zeros(size(ss,2),size(ss,1));
%show the svd gains
figure; semilogy(sv,'*b-'); title('svd gains-AOA recon');

%remove some modes
modesRemoved = 0;
for ii=1:size(svi,1)-modesRemoved;
    si(ii,ii) = svi(ii);
end;
control=v*si*u';

%show the control matrix
figure; imagesc(control); title('Control');

%% create a phase screen
Dap = 30.0e-2;  Dap_over_r0=2;
r0 = Dap/Dap_over_r0;
nxy = 128; dxy = Dap/nxy;
xp = (-nxy/2:1:nxy/2-1)*dxy;
```

```
[xxp,yyp]=meshgrid(xp,xp);
rp = sqrt(xxp.^2+yyp.^2);
K = fftshift(rp);
PHIp = (0.023/r0.^(5/3)) .* K.^(-11/3);
PHIp(1,1)=0;
PHI = sqrt(PHIp) .* exp(j*2*pi*rand(nxy,nxy));
%figure; subplot(1,2,1); imagesc(abs(PHI)); colorbar; title('PHI');
%subplot(1,2,2); imagesc(angle(PHI)); colorbar; title('PHI');
phi = ifft2((PHI));
% figure('position',[360         278        1032         420]);
% subplot(1,2,1); imagesc(imag(phi)); colorbar; title('imag');
% subplot(1,2,2); imagesc(real(phi)); colorbar; title('real');
phase = real(phi) ./ (2*pi) .* wavelength;
figure; imagesc(xp,xp,phase); axis image; title('Phase (waves)'); colorbar;
hold on;
theta = 0:0.1:2.0*pi;
%xcircle = Ddm/2 * cos(theta); ycircle = Ddm/2 * sin(theta);
xcircle = Dap/2 * cos(theta); ycircle = Dap/2 * sin(theta);
plot(xcircle,ycircle,'r--','linewidth',2');

%% sample the desired shape with magnification
Magnification = Dap / Ddm;
shape = interp2(xp./Magnification,(xp./Magnification)',phase,x,(x)');
%interpolate onto the same grid
shape(find(isnan(shape)==1))=0;

%remove the mean over the fit aperture area
shapeFit = shape .* apFit;
shapeVec = shapeFit(:);
ind = find(shapeVec~=0);
shape = shape - mean(shapeVec(ind));
shape = shape .* apFit;

%sample it on the desired grid
for jj=1:length(xi)
    vec(jj) = shape(xi(jj),yi(jj));
end;

%fit to the desired shape
commands = control' * vec';
figure; bar(commands); title('Commands');

if (limiting)
    %implement limiting at -0.5 to 0.5
    c0 = commands;
    commands = c0 .* (c0>-0.5) + (c0<-0.5).*-0.5;
    commands = commands .* (c0<0.5) + (c0>0.5).*0.5;
    figure; bar([c0 commands c0-commands]); title('Limited Commands');
colormap(jet);
    legend('requested','limited','difference');
end;

%create the resulting DM shape
dm=0.0 .* IF{1};
for ii=1:length(IF); dm=dm+IF{ii}.*commands(ii); end;
```

```
%save the output DM file
maxCounts = 255;
dmf2 = dmf;
cmdAverage = mean(commands);
for ii=1:length(IF);
    dmf2(ii).v = round((commands(ii) - cmdAverage + 0.5) .* maxCounts);
    if (dmf2(ii).v>maxCounts) dmf2(ii).v=maxCounts; end;
    if (dmf2(ii).v<0) dmf2(ii).v=0; end;
end;
SaveDM('MDM1-32S-001-fit.dm',dmf2);
% this line tests the file output by displaying the re-read file
% dm2=LoadDM('MDM1-32S-001-fit.dm'); nf; ShowDM(dm2);

%show the resulting DM shape
figure;
subplot(1,3,1); imagesc(x,x,shape .* apDM); axis image; colorbar;
title('Desired Shape');
hold on; plot(x(xi),x(yi),'wo');
cax = caxis;
subplot(1,3,2); imagesc(x,x,dm); axis image; colorbar; title('DM Shape');
hold on; plot(x(xi),x(yi),'wo');
caxis(cax); colorbar;
subplot(1,3,3); imagesc(x,x,apFit .* (shape - dm)); axis image; colorbar;
title('Difference');
hold on; plot(x(xi),x(yi),'wo');
caxis(cax); colorbar;

%calculate RMS wavefront error and estimated Strehl ratio
delta = apFit .* (shape - dm);
dv=delta(:); ind = find(dv~=0); dvs = dv(ind); %look at all the non-zero
elements in vector form
rmswfe_m=sqrt(sum((dvs-mean(dvs)).^2)/length(ind))
rmswfe_rads = rmswfe_m / wavelength * 2 * pi
SR = exp(-1.0 * (rmswfe_rads).^2)

%look at what the Strehl ratio would be without compensation
delta = apFit .* (shape);
dv=delta(:); ind = find(dv~=0); dvs = dv(ind); %look at all the non-zero
elements in vector form
rmswfe_m_uncomp=sqrt(sum((dvs-mean(dvs)).^2)/length(ind));
rmswfe_rads_uncomp = rmswfe_m_uncomp / wavelength * 2 * pi;
SR_uncomp = exp(-1.0 * (rmswfe_rads_uncomp).^2)
```