ACTIVE OPTICAL SYSTEMS

AN012

Control Matrices Generation for Hartmann Wavefront Sensor Adaptive Optics

Author: **Justin D. Mansell, Ph.D.**

Active Optical Systems, LLC

Revision: **11/29/09**

In this application note, we describe how an adaptive optics control matrix can be generated for an adaptive optics system comprising a deformable mirror and a Hartmann wavefront sensor. There are many different variations on how to create a control matrix, but we will focus on what we believe to be the simplest version.

In a typical AOS adaptive optics system, an aberrated beam of light reflects from a deformable mirror and illuminates a Hartmann wavefront sensor. Figure 1 illustrates this typical setup. The wavefront sensor measures the average gradient (aka slope) of the wavefront over each sub-aperture in an array of sub-apertures to produce a vector of wavefront slope measurements. There are two common conventions for the ordering of the x and y slopes: xxyy and xyxy. We use the xyxy convention at AOS.

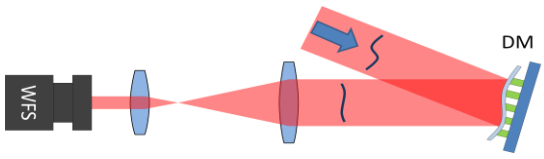

Figure 1 - Typical AOS Adaptive Optics System Optical Setup

To complete an adaptive optics system with this optical setup, we need to convert the wavefront slope measurements into commands to the deformable mirror. In the AOS software, this is accomplished with either metric AO searching or through a control matrix. We will only discuss the control matrix approach in this application note. When the vector of measured slopes is multiplied by the control matrix it produces a vector of commands that if applied to the DM (in addition to any commands that currently exist on the DM) will produce the minimum RMS wavefront slopes.

Making a Poke Matrix

In the AOS software, the adaptive optics control matrix is generated by creating a pseudo-inverse of a measured poke matrix. The poke matrix is a matrix that measures the response of each actuator on the HWFS. In the AOS software, the poke matrix is generated by actuating each actuator or group of actuators individually, measuring the

wavefront slopes, and assembling the vector of slope measurements into a matrix. This matrix describes the relationship between commands to the deformable mirror and response of the wavefront sensor. Mathematically, this can be written as

$$\nabla\phi = P \cdot c$$

where $\nabla\phi$ is the vector of wavefront slopes, P is the poke matrix, and c is the vector of DM commands. It is important to note that the DM commands are not voltages, but instead should be thought of as forces since many different DMs (e.g. membrane deformable mirrors) have a parabolic response to voltage. This poke matrix can saved as a comma-separated value (CSV) file from the AOS software.

The best poke matrix only contains the effect of the actuator motion. It is usually good to create a new reference to eliminate the system static aberrations and isolate the DM influence functions (aka pokes). To isolate the actuator motion from the other system aberrations, the AOS software has an option to poke up (positive command) and down (negative command) and record the difference between these measurements into the poke matrix. To avoid any static image noise, the AOS software also has the ability to subtract a background image from each frame.

There is another option that enables the user to remove the slope vector averaged over all the pokes to remove the background aberration. The average slope removal is controversial because it results in a very low-gain SVD mode and, as such, a very high condition number, which is the ratio of the highest SVD gain to the lowest SVD gain (see Matlab's cond() function). This low gain mode can be removed from the control matrix using the SVD techniques described later.

Actuator Efficacy & Slaving

One simple technique to quantify the efficacy of each actuator on the AO system is to examine the peak-to-valley (max minus min) slope vector for each actuator poke. This value should be large relative to the noise floor of the wavefront sensor for each of the actuators. If the value is close or comparable to the noise floor, the actuator should be removed from control if it is poorly illuminated or slaved to (set to the same value as) an adjacent well-observed actuator. This can be done in the AOS software by deactivating an actuator or grouping it to its neighbor in the DM Controller.

Noise Rejection through Simulation

To achieve better results, the poke matrix can be processed to remove noise from the matrix. The poke matrix can be analyzed to help determine the mapping of the deformable mirror actuators to the sub-apertures. Then modeling of the DM can be used to create a poke matrix without any noise from the measurement. The AOS software supports this, by enabling saving of the poke matrix and loading of a new control matrix. These techniques are advanced and will not be discussed further here.

Making a Control Matrix from a Poke Matrix

Once a good poke matrix has been established, this poke matrix needs to be inverted to create a control matrix that relates the measured wavefront slopes to DM commands. Mathematically this can be represented as

$$\begin{aligned} c &= pseudoinverse(P) \cdot \nabla\phi \\ &= \Gamma \cdot \nabla\phi \end{aligned}$$

where Γ is the control matrix. Since the poke matrix is not square (it is 2 times the number of sub-apertures by the number of actuators),

it cannot be inverted classically, but instead requires a different mathematical technique.

The AOS software uses singular value decomposition (SVD) to create a pseudo-inverse. In SVD[1], the poke matrix is decomposed into the product of three matrices U, S, and V. U is a set of orthonormal output basis vectors that are found by finding the eigenvectors of $P \cdot P^*$ where the * operator is the transpose. V is a set of orthonormal input basis vectors that are found by finding the eigenvectors of $P^* \cdot P$. S is a diagonal matrix of singular values that are found as the square root of the eigenvalues of the eigenvectors in U and V.

The SVD decomposition can be used to calculate a pseudo-inverse of the poke matrix, P, as

$$\Gamma = P^+ = V \cdot S^+ \cdot U^*$$

where the $^+$ operator is the pseudo-inverse. The pseudo-inverse of S is found by taking the reciprocal of each diagonal element of the S matrix.

The condition number of a matrix (found from the Matlab command cond()) is the ratio of the highest gain to the lowest gain. If a matrix is formed of orthogonal vectors, then the condition number will be 1. Since in most deformable mirrors the influence functions are not orthogonal, it is expected that the condition number will not be unity, but lower values are generally better. Unfortunately, the number cannot typically help diagnose the source of a problem, so a more detailed examination is required.

It is common to examine the magnitudes of the gains from the SVD calculation to determine the quality of the poke matrix and help identify any problems. If there is one mode that is much higher gain than the others, it usually indicates the presence of a static aberration in the measurements. If the gains span more than about 2 orders of magnitude, then the matrix may be poorly conditioned for adaptive optics. Examination of a system with a large range of SVD gains usually shows influence functions with limited orthogonality, multiple actuation of a single actuator, or too many poorly observed actuators.

SVD Mode Removal

There are many different variations on creating a control matrix, but one that is commonly used is the removal of SVD modes from the control matrix. This is accomplished by setting the gain corresponding to the SVD mode being removed to zero in the S matrix before calculating the pseudo-inverse. Removing modes from the control matrix is not the same as removing actuators from the control matrix. We only recommend removing one or two modes from the control matrix. If it is necessary to remove more than that to make the system operate well, it is likely that there are poorly observed actuators that should be removed from control or slaved (aka grouped) to their neighbors.

Evaluating the Control Matrix with Poke-Control Product

After the generation of the control matrix, one way of evaluating the quality of the control matrix is to multiply it by the poke matrix and compare it with an identity matrix. A single number metric is the RMS of the difference between the poke-control product and the identity matrix, which would ideally be zero. If it is significantly non-zero then the control matrix may be poorly conditioned for adaptive optics.

Using a Control Matrix in Integrator Control

After we have established a control matrix, the AOS software is setup for integrator control of the wavefront. The commands for

the DM are calculated in each step of the AO loop by taking the commands from the last loop and adding the measured slopes times the control matrix times the AO system gain, or

$$c(t) = c(t-1) + gain \cdot \Gamma \cdot \nabla\phi .$$

There are other methods of implementing AO systems using control matrices, but this is one of the most commonly used and has proven to be very effective.

Example: Analysis of a Measured Poke Matrix

To better illustrate the topics discussed here, we have put together a Matlab script to analyze a measured poke matrix. The code for this is shown in the Appendix. Some of the dependent functions are not included, but they can be commented out to make the script run.

The first thing that the script does is to load the data saved from the AOS software. The functions to do this come with the current version (1.8) of the AOS software. This is a poke matrix that was measured for a 32 actuator square grid membrane DM (layout shown in Figure 2).

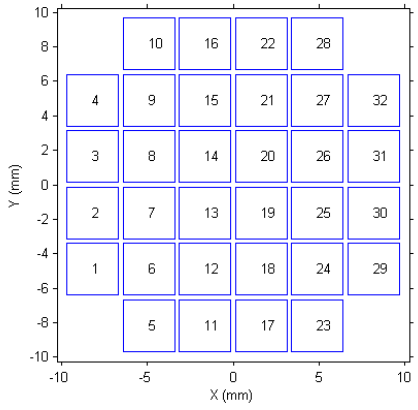

Figure 2 - Layout and actuator numbering of the 32-channel square-grid MDM

Figure 3 shows the poke matrix that was loaded from the file.

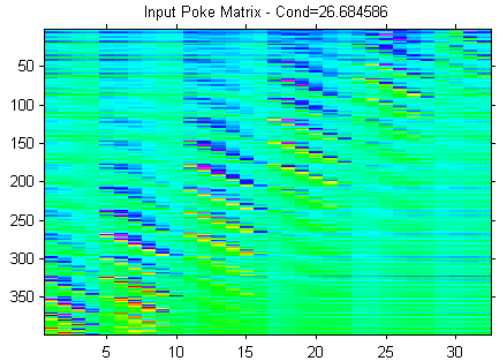

Figure 3 - Measured Poke Matrix

The condition number was 27 indicating a fairly good poke matrix, but analysis of the SVD gains, shown in Figure 4, showed an area of potential improvement. The highest gain was significantly higher than the next highest gain, indicating a problem with a background static aberration in the system.

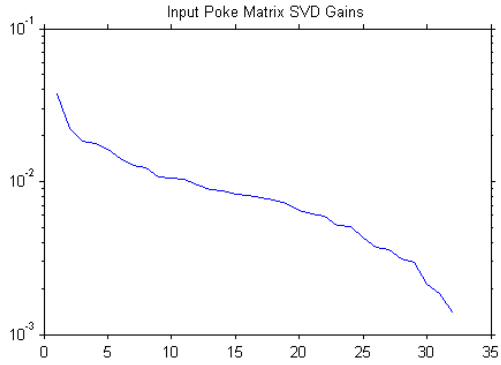

Figure 4 - SVD Gains from Measured Poke Matrix

We averaged all the measured slopes in the poke matrix together to show the background pattern. Figure 5 shows the average slopes in the x and y directions.

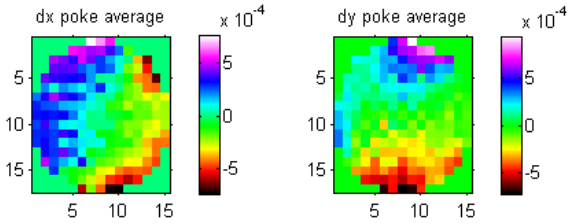

Figure 5 - Average Slopes from Poke Matrix

A significant portion of the 6.9 mradian poke range was this background pattern alone, so we subtracted this average poke from the poke matrix and created a much cleaner poke matrix, shown in Figure 6. The horizontal streaks that were in the measured poke matrix have been removed and the pattern is much clearer.

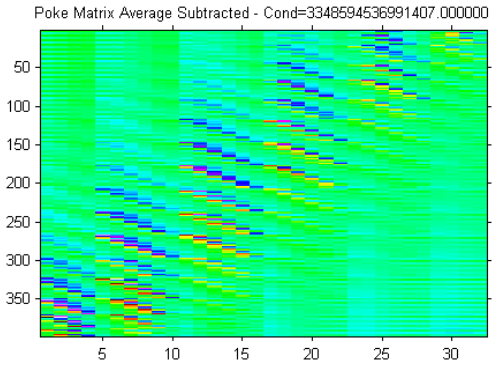

Figure 6 - Poke Matrix with the Average Slope Removed

Unfortunately, the condition number is much higher (3.3e15). Figure 7 shows a section of the SVD gains of the new average-subtracted matrix. Examination of the SVD gains showed that all the range was coming from the last mode. If it were not considered, the ratio of low to high gains was only 15.5.

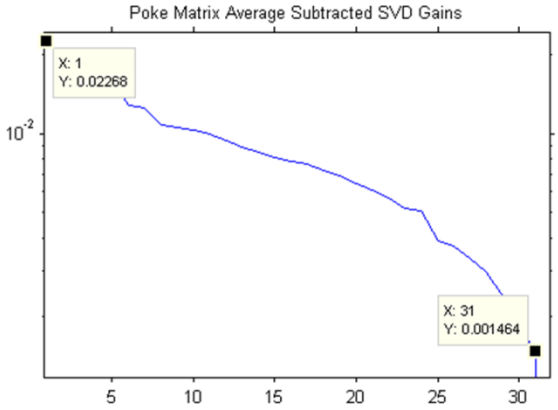

Figure 7 - SVD Gains After Average Subtraction

Poke Amplitude Analysis

After removing the static aberration from the background, we examined the amplitude of the measurement of the influence function (poke) by subtracting the maximum slope from the minimum slope for each actuator to evaluate the efficacy of each actuator. Figure 8 shows the peak-to-valley poke amplitude for each influence function. This plot shows that some of the actuators do not have as strong an influence on the wavefront sensor as others, but there are not any that are so low as to warrant removal from the matrix or slaving since they are all significantly above the wavefront sensor noise floor.

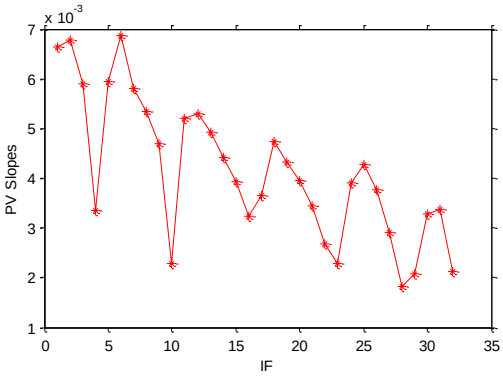

Figure 8 - Peak to Valley Slopes for Each Influence Function

SVD Mode & Control Matrix Analysis

We continued the analysis of the SVD terms by plotting the orthonormal modes in DM (input) and wavefront-sensor (output) space, shown in Figure 9 and Figure 10 respectively. Modes in these plots are ordered from highest to lowest gain. To provide a more direct comparison, the modes in the wavefront sensor space were reconstructed into phase. As is expected in this type of system, the lowest spatial frequency modes have the highest gain.

Outside of the script we multiplied the poke and control matrices after removing one SVD mode and found that the RMS of the difference between the identity matrix and the poke-control matrix product was 3e-16. When we created a control matrix with 10 modes removed, the RMS increased to 9.4%.

Conclusions

In this application note we have shown how we create a control matrix from a poke matrix using singular value decomposition (SVD). We also interpreted the results of the SVD and showed how it can be used to create better control matrices.

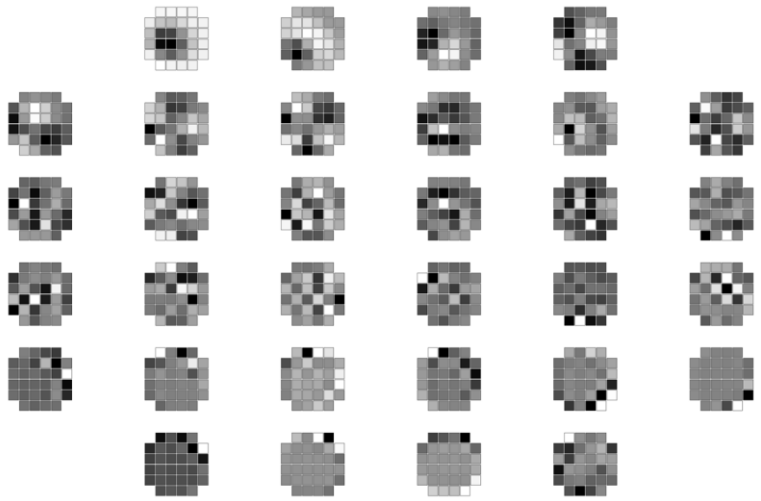

Figure 9 - SVD Modes in DM Space

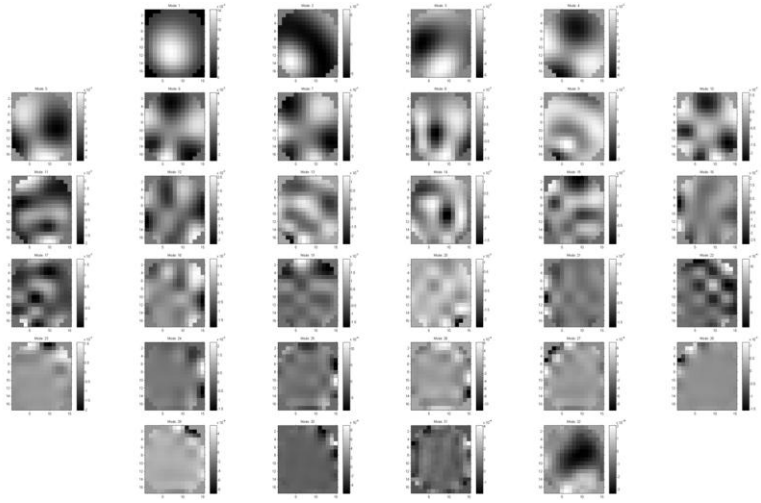

Figure 10 - SVD Modes in the HWFS space

Appendix: Matlab Poke Matrix Analysis Code

```
% this script analyzes the poke and control matrices
setup;
showIFs=1;
ppt=1;

%load the data
poke=csvread('CM_poke.csv');
Nact = min(size(poke));
Nsub = max(size(poke))/2;
[wfs,aois]=LoadWFS('refFlat.wfs');
vec = [2:5 7:30 32:35];
%[wfs,aois]=LoadWFS('refDMcenter.wfs');

%show the poke matrix
nf; imagesc(poke);
title(sprintf('Input Poke Matrix - Cond=%f',cond(poke)));
if (ppt) ToPPT(); end;

[pinv,gains]=svdinverse(poke);
nf; semilogy(gains);
title('Input Poke Matrix SVD Gains');
if (ppt) ToPPT(); end;
%condition number (cond) is the ratio of highest to lowest SVD gains
%values near 1 are a well conditioned matrix

%% subtract the average slopes from the poke matrix
pokeOrig = poke;
pokeavg = mean(poke')';
for ii=1:Nact;
    poke(:,ii) = poke(:,ii)-pokeavg;
end;

%show the average pokes
nf;
for ii=1:length(aois);
    xi = aois(ii).xindex+1;
    yi = aois(ii).yindex+1;
    dx(xi,yi) = pokeavg(2*ii-1);
    dy(xi,yi) = pokeavg(2*ii);
end;
subplot(1,2,1); show(dx); title('dx poke average');
subplot(1,2,2); show(dy); title('dy poke average');
if (ppt) ToPPT('Average Poke Wavefront Slopes'); end;

%show the new poke matrix
nf; imagesc(poke);
title(sprintf('Poke Matrix Average Subtracted - Cond=%f',cond(poke)));
if (ppt) ToPPT(); end;

[pinv,gains]=svdinverse(poke);
nf; semilogy(gains);
title('Poke Matrix Average Subtracted SVD Gains');
if (ppt) ToPPT(); end;

%% show the IFs in slope space
if (showIFs)
    cmin=min(poke(:)); cmax=max(poke(:));
    for ii=1:Nact;
        for jj=1:Nsub;
            xi = aois(jj).xindex + 1;
            yi = aois(jj).yindex + 1;
            dx(xi,yi)=poke(2*jj-1,ii);
            dy(xi,yi)=poke(2*jj,ii);
        end;
%         clf;
```

```
%         subplot(1,2,1); show(dx); caxis([cmin cmax]); colorbar;
%         title('x slopes');
%         subplot(1,2,2); show(dy); caxis([cmin cmax]); colorbar;
%         title(sprintf('y slopes: %i',ii));
%         drawnow; pause(0.1);
        clf;
        show([dx dy]); caxis([cmin cmax]); colorbar;
        title('Wavefront Slopes');
        text(1,             1.5,  'X Slopes');
        text(size(dx,1)+1,  1.5,  'Y Slopes');
        drawnow; pause(0.1);
        if (ppt) ToPPT(gcf,'IFs',[6 6 vec(ii)],ii==1); end;
    end;
end;

%plot peak to valley slopes
nf; plot(max(poke)-min(poke),'r*-');
ylabel('PV Slopes');
xlabel('IF');

%% invert the poke matrix and show the control matrix & svd gains
modesRemoved=1;
[ctrl,gains,modes]=svdinverse(poke,modesRemoved);
nf; imagesc(ctrl); title('Control');
nf; semilogy(gains,'r*-'); title('SVD Gains');

%% show the SVD modes in DM space
rmax = ceil(sqrt(Nact));
cmax = rmax;
dm = LoadDM('MDM1-32S-001.dm');
for ii=1:Nact;
    %v=modes(ii,:); %find the DM modes - WRONG ORIENTATION
    v=modes(:,ii); %find the DM modes
    clf;
    ShowDM(dm,v,[true],true);
    axis image;
    axis off;
    zoomOut(0.05);
    drawnow; pause(1.0);
    if (ppt)
        ToPPT(gcf,'SVD Modes',[rmax cmax vec(ii)],ii==1);
    end;
end;

%% show the SVD modes in WFS space
rmax = ceil(sqrt(Nact));
cmax = rmax;
for ii=1:Nact;
    %v=modes(ii,:); %find the DM modes - WRONG ORIENTATION
    v=modes(:,ii); %find the DM modes
    v2 = poke*v(:); %convert the modes to WFS space
    for jj=1:Nsub;
        xi = aois(jj).xindex + 1;
        yi = aois(jj).yindex + 1;
        dx(xi,yi)=v2(2*jj-1);
        dy(xi,yi)=v2(2*jj);
    end;
    intensity = (dx~=0);
    z = Southwell2(intensity,dx,dy,1,1);
    clf;
    subplot(1,2,1); show(dx);
    title('x slopes');
    subplot(1,2,2); show(dy);
    title(sprintf('y slopes: %i',ii));
    drawnow; pause(0.1);
    if (ppt)
        ToPPT(gcf,'SVD Modes',[rmax cmax vec(ii)],ii==1);
    end;
end;
```

```
%% show the SVD modes in WFS space
rmax = ceil(sqrt(Nact));
cmax = rmax;
for ii=1:Nact;
    %v=modes(ii,:); %find the DM modes - WRONG ORIENTATION
    v=modes(:,ii); %find the DM modes
    v2 = poke*v(:); %convert the modes to WFS space
    for jj=1:Nsub;
        xi = aois(jj).xindex + 1;
        yi = aois(jj).yindex + 1;
        dx(xi,yi)=v2(2*jj-1);
        dy(xi,yi)=v2(2*jj);
    end;
    intensity = (dx~=0);
    z = Southwell2(intensity,dx,dy,1,1);
    clf;
    show(z); colormap(gray);
    title(sprintf('Mode: %i',ii));
    drawnow; pause(0.1);
    if (ppt)
        ToPPT(gcf,'SVD Modes',[rmax cmax vec(ii)],ii==1);
    end;
end;


% function [Minv,gains,modes]=svdinverse(M,varargin)
% %function [Minv,gains,modes]=svdinverse(M,[modesRemoved])
% % does the SVD inverse of a matrix with mode removal
% % Example:
% % [D,gain,modes]=svdinverse(M,1);
% % for ii=1:size(modes,2);
% %     modePhs{ii} = reshape(M * modes(:,ii),ny,nx);
% %     nf; show(modePhs{ii}); title(ii); pause(0.01);
% % end;
% if (nargin==2)
%     modesRemoved=varargin{1};
% else
%     modesRemoved=0;
% end;
%
% [u,ss,v]=svd(M);
% sv=diag(ss); gains=sv;
% svi = 1.0./sv;
% si = zeros(size(ss,2),size(ss,1));
% %nf; semilogy(sv,'*b-'); title('svd gains-AOA recon');
% for ii=1:size(svi,1)-modesRemoved;
%     si(ii,ii) = svi(ii);
% end;
% Minv=v*si*u';
% if (nargout>=3)
%     %calculate the mode shapes
%     modes = v;
% end;
% return;
```

References

[1] http://en.wikipedia.org/wiki/Singular_value_decomposition