**ME 610 – HW3**

**Model Correlation and Update**

**Victor Cavalcanti**

**OBJECTIVE**

The objective of this assignment is to perform a TAM/TEST target mode correlation between the ­modes of a reduced TAM and the data acquired from a vibration test on ‘beam601.mat’, a 41-DOF beam. The mode shapes, frequencies and damping of the test system will be acquired by computing the drive-point FRF and applying parameter extraction techniques.

**PROCEDURE**

The first task is to generate a TAM that will accurately predict the target modes of interest (FEM modes 4 through 8) using the accelerometers located at nodes 1, 13, 17, 29 and 41. **Modal reduction** was used to generate a TAM, **since a modal TAM will always perfectly predict the target modes and frequencies as long as you have as many DOF as number of modes of interest.** Cross-Orthogonality **(Appendix A)** as well as frequency errors will be used to compare the FEM/TAM. Results from a static reduction **(Appendix A)** will be shown for contrast.

To generate a modalTAM, begin by partitioning your DOF vector and modes into the DOF you wish to keep, and those you wish to reduce out of your system: (

(1)

We then generate a Ritz transformation of the form:

(2)

Which, when applied to our sorted Mass and Stiffness matrices, will yield the matrices for our reduced system:

(3) (4)

The second task was to generate a drive point FRF of the tap test data from the file ‘taptest.mat’. The tap test was performed at node location 1 and the data was acquired at a 1024 Hz sample rate. The FRF matrix for the k-th frequency and its assembly for all frequencies is given below:

(5)

(6)

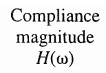
Where , is the complex FFT coefficient of the output vector (acceleration in this case), is the complex FFT coefficients for the input vector (tap test). The first term in brackets is the Auto-Spectral Density and the second is the Cross-Spectral Density.

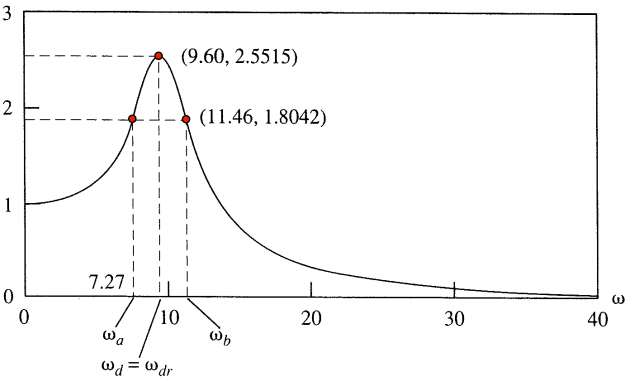
The matrix G will be the **inertance** FRF for our system. By dividing each column of G by the corresponding frequency , we can attain the **compliance** FRF. The columns of G represent complex mode shapes at that particular frequency.

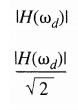
(7)

These mode shapes can be realized as shown below:[[1]](#footnote-1)

(8)

Frequency and damping can be taken from the half power points from the compliance FRF.



****

Finally, a test-analysis correlation for the TAM will be performed. Cross-Orthogonality, MAC **(APPENDIX A)** and frequency errors will be used to compare the modes. A test-FEM comparison will also be done.

**RESULTS**

 ****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mode # |  |  | % Error |  | % Error |
| 1 | 0.00 | - | - | 0.00 | - |
| 2 | 0.00 | - | - | 0.00 | - |
| 3 | 10.98 | - | - | 11.05 | -0.63 |
| 4 | 30.22 | 30.22 | 0.00 | 30.36 | -0.48 |
| 5 | 59.16 | 59.16 | 0.00 | 66.67 | -12.69 |
| 6 | 97.67 | 97.67 | 0.00 | - | - |
| 7 | 145.71 | 145.71 | 0.00 | - | - |
| 8 | 203.24 | 203.24 | 0.00 | - | - |

****

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Mode # |  |  |  |  |  |  |  |
| 4 | 38 | 0.235 | 0.166 | 36 | 40 | 0.053 | 38.106 |
| 5 | 64 | 0.113 | 0.080 | 62 | 66 | 0.031 | 64.063 |
| 6 | 96 | 0.063 | 0.044 | 92 | 98 | 0.031 | 96.094 |
| 7 | 136 | 0.032 | 0.023 | 132 | 138 | 0.022 | 136.066 |
| 8 | 184 | 0.011 | 0.008 | 180 | 188 | 0.022 | 184.087 |

**APPENDIX A**

**Static Reduction**

To statically reduce a system, begin by partitioning your DOF vector into the DOF you wish to keep, and those you wish to reduce out of your system: (

We assume the DOF being reduced possess little mass and that no load is applied to them. The EOM for an undamped system with this new partitioned DOF vector, it will look as follows:

Working with the second equation, we can derive the following Ritz transformation matrix:

Which, when applied to our sorted Mass and Stiffness matrices, will yield the matrices for our reduced system:

**MaC, self and cross-orthogonality**

Modal comparison between FEM and test modes can be made by computing the Modal Assurance Criterion (MAC). MAC stems from the inequality of vector products:

Which, in our case, we are working with the squares of the modes, so our expression becomes:

Where a value of 0 indicates that modes are orthogonal and a value of 1 indicates modes are parallel.

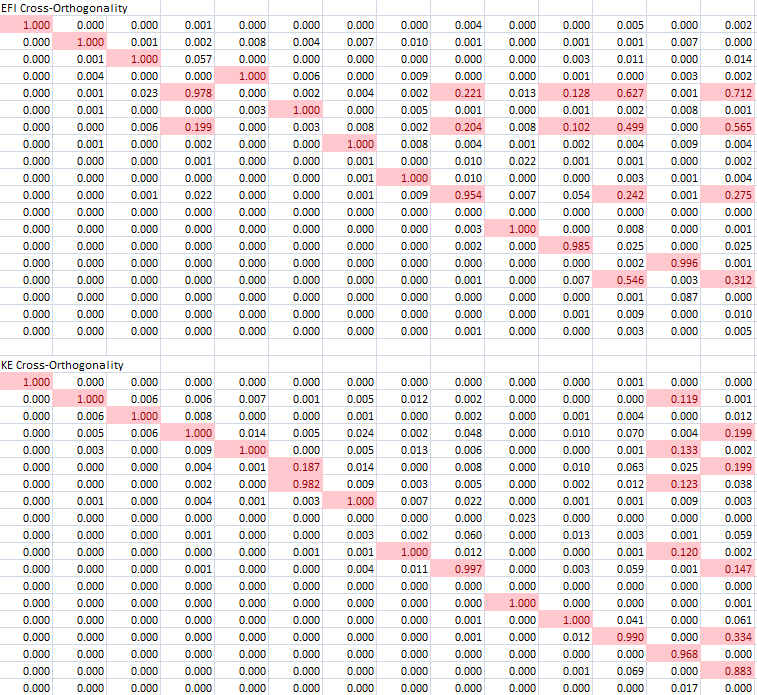
After modes have been matched by using MAC, a mass-weighted cross and self-orthogonality test will can done. The expressions used will be:

In an ideal world, the self-orthogonality check for test modes will yield an identity; however, this will rarely be the case. In the cross-orthogonality computation, the criteria for ideal mode matching is to have diagonal values of and off-diagonal values of .

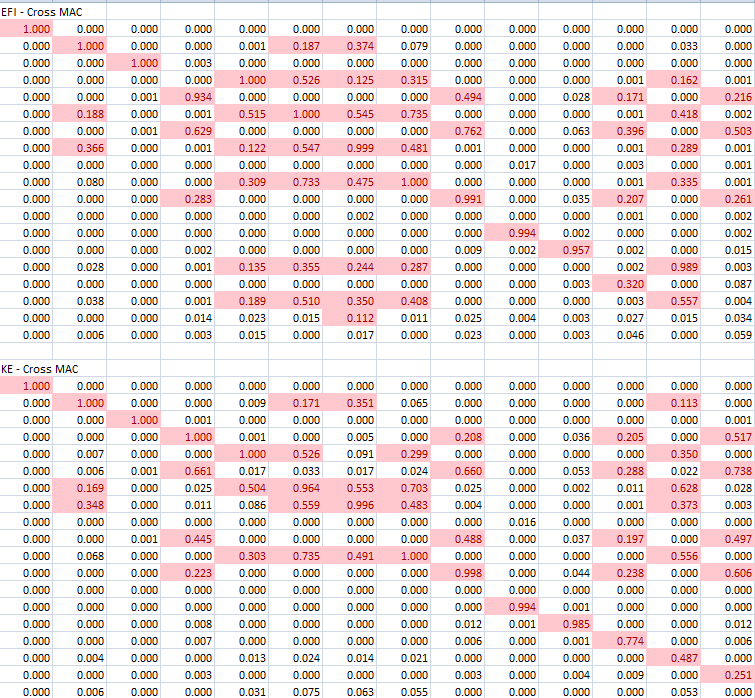
Frequency errors will be computed between the FEM and TEST frequencies (in Hertz), by using the formula:

**APPENDIX B – Numerical Values for MAC, SO, CO**

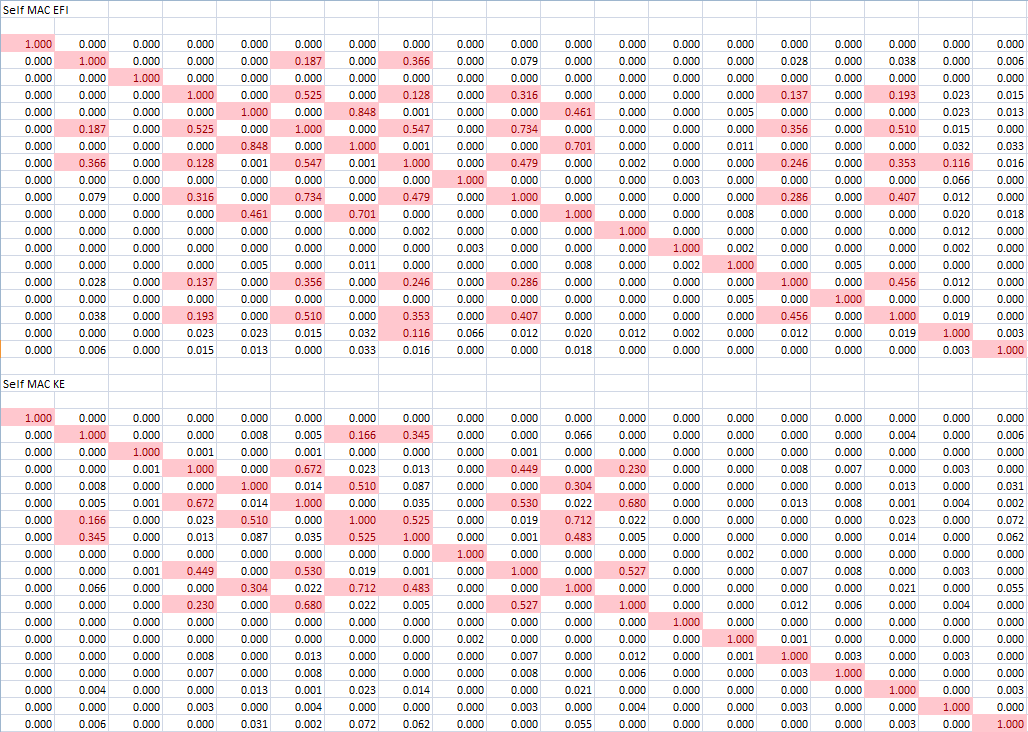
Cross-Orthogonality for Effective Independence. The Rows correspond to the EFI and KE set modes, while the Columns correspond to the target modes.

****

Cross-Modal Assurance Criterion, The rows represent the EFI and KE modes while the columns represent the target modes.



Self-Modal Assurance Criterion. The columns and rows represent the EFI and KE modes.

****

**APPENDIX C – MATLAB CODE**

**getEffectiveIndependence**

function [Efi,DOF,xtraDOF,detFish,kedetFish] = getEffectiveIndependence(PHI,M,K, ntargetdof,DOF)

%Effective Independce can be explained here:

% https://ay16-17.moodle.wisc.edu/prod/pluginfile.php/171634/

% mod\_resource/content/3/eCOWI\_Resources/lecturepre/

% Topic%2010.2%20-%20Sensor%20Placement%20using%20Effective

% %20Independence%20Pres.pdf

%And here:

% https://ay16-17.moodle.wisc.edu/prod/pluginfile.php/171635/

% mod\_resource/content/1/AIAA-JGCD-91.pdf

%It uses Modal Kinetic Energy to pick an initial candidate set of DOF which

%is 2x ntargetdof. It then iteratively removes dof by ranking via effective

%independence until a set of target dof is chosen (with ntargetdof dof).

format long;

%xtraDOF - DOF FOR HW3. SET OF nM+5 dof based solely on MKE.

[n,nM] = size(PHI);

iniset = 3\*ntargetdof;

% iniset = n;

for i = 1:n

for j = 1:nM

KE(i,j) = PHI(i,j)\*M(i,:)\*PHI(:,j);

end

end

KEnorm = (1/nM)\*sum(KE,2);

%idx is the index of the positions in the PHI vector of the sorted kinetic

%energy by modes.

[KEsort, idx] = sort(KEnorm,'descend');

xtraDOF = DOF(idx(1:ntargetdof));

%DOF for only the initial candidate set of modes

DOF = DOF(idx(1:iniset));

PHIb = PHI;

%Reduce our target modes to the initial set of DOF selected by Modal

%Kinetic Energy

PHI = PHI(idx(1:iniset),:);

k = iniset;

detFish = zeros(iniset-ntargetdof,1);

kedetFish = zeros(iniset-ntargetdof,1); %HW 3 only

j = 0;

figure; hold on;

while k > ntargetdof

PHIKE = PHIb(idx(1:iniset-j),:);

B = PHIKE'\*PHIKE;

kedetFish(iniset-k+1) = det(B); %HW 3 only

j = j+1;

% A = zeros(nM,nM);

% for i = 1:size(PHI,1)

% A = A+PHI(i,:)'\*PHI(i,:);

% end

A = PHI'\*PHI;

[V,D] = eig(A);

[D, ascindx] = sort(diag(D),'ascend'); %Sort eigenvalues.

V = V(:,ascindx); %sort eigenvectors accordingly.

G = (PHI\*V).^2;

% F = G.\*repmat((1./D'),38,1);

F = G\*inv(diag(D));

Efi = F\*ones(nM,1);

[Efi,index] = sort(Efi,'descend');

%Remove the lowest contributing dof from original DOF list and from PHI.

DOF = DOF(index(1:end-1));

PHI = PHI(index(1:end-1),:);

detFish(iniset-k+1) = det(A);

k = k-1;

end

plot(detFish,'bx-');

plot(kedetFish,'ro-');

xlabel('Iteration #');

ylabel('det(Q)');

legend('Q - efi' ,'Q- ke');

end

**getEffectiveMass**

function [E,MR, MRo,M,K,dofA,dofO] = getEffectiveMass(M,K,PHI, DOFint, DOF, PHIR )

% Effective Mass can be described here:

% https://ay16-17.moodle.wisc.edu/prod/pluginfile.php/171644/...

% mod\_resource/content/1/Topic%2012.1%20-%20Selection%20of%20Target...

% %20Modes%20-%20Effective%20Mass%20Pres.pdf

% Returns E, the effective mass matrix. It's columns sum to the diagonal

% terms of the rigid body mass matrix Mro (PHI'\*Moo\*PHI)

% Returns MR, the rigid body mass matrix, MRo, the o-set partition of MR.

% M and K, sorted mass and stiffness.

% Takens in the mass matrix M, FIXED-INTERFACE MODES PHIro,

% modes PHI, the list of DOF at the interface DOFint and

% the total list of DOF, DOF.

% Mass matrix and rows of PHI should be sorted according to DOF.

nDOFint = length(DOFint); % # kept dof.

nDOF = length(DOF); % # dof total.

dofA = zeros(nDOFint,1); % INDEX OF ASET wrt M and K

for i = 1:nDOFint

[tmpind,~] = find(DOF==DOFint(i));

dofA(i) = tmpind;

end

dofO = setxor(1:nDOF,dofA); % INDEX OSET DOF wrt M and K

DOF = DOF([dofA;dofO],:); % ordered DOF list, [ASET;OSET]

Moo = M(dofO,dofO);

for i = 1:numel(PHI(1,:))

mnorm(i) = 1/(sqrt(PHI(:,i)'\*Moo\*PHI(:,i)));

PHI(:,i) = mnorm(i)\*PHI(:,i);

end

% Moa = M(DOFind,DOFCOMP);

% Mao = M(DOFCOMP,DOFind);

% Maa = M(DOFCOMP,DOFCOMP); `

%M AND K PARTITIONED ACCORDING TO DOF

M = [ M(dofA,dofA), M(dofA,dofO); ...

M(dofO,dofA),M(dofO,dofO)];

K = [ K(dofA,dofA), K(dofA,dofO); ...

K(dofO,dofA),K(dofO,dofO)];

%Orthogonality Check- turn on to see if PHI is orthogonal in a general

%sense.

% surf(PHI'\*Moo\*PHI);

% OSET MASS WEIGHTED FIXED INTERFACE MODES ORTHOGONALITY CHECK.

%Compute PHIro \*\*\* NOTE, this restricts rigid body modes from the first 6

%DOF of the interface.

% PHIr = [ eye(6); -inv(K(7:end,7:end))\*K(7:end,1:6)];

% PHIRo = PHIR(nDOFint+1:end,:);

PHIRo = PHIR(dofO,:);

PHIR = PHIR([dofA; dofO],:);

MR = PHIR'\*M\*PHIR;

MRo = PHIRo'\*Moo\*PHIRo; %Modal mass matrix contributing to o-set partition.

%%

A = repmat((1./diag(MRo))',30,1);

E = ([PHI'\*Moo\*PHIRo].^2).\*A;

% E = ([PHI'\*Moo\*PHIRo].^2)\*(1./diag(MRo));

end

**Wrapper Function**

close all; clear all; clc;

load('gpsc.mat');

wtmass = 386.4; %parameter to convert weight to mass for M.

% M = wtmass.\*M; %Convert to propper units.

%Sort DOF1 and DOF2 in ascending order. Reorder M and K according to DOF1

%sort, sort PHI according to DOF2 sort.

[DOF1, DOF1ind] = sort(DOF1,'ascend');

[DOF2, DOF2ind] = sort(DOF2,'ascend');

M = M(DOF1ind,DOF1ind);

K = K(DOF1ind,DOF1ind);

PHI = PHI(DOF2ind,:);

PHIFI = PHI; %fixed interface modes;

%interface dof set

aSET = [44.1; 44.2; 44.3; 45.1; 45.2; 45.3; 48.1; 48.2; 48.3; 49.1; ...

49.2; 49.3];

%Compute RBM about node 50

PHIR = [ -inv(K(1:end-6,1:end-6))\*K(1:end-6,end-5:end); eye(6)];

PHIR = PHIR(1:end-3,:); %Remove the rotational DOF from PHIR

DOF1 = DOF1(1:end-3); %Remove the rotation DOF from list.

%Static reduction of the 3 rotations at DOF 50 (50.4,50.5,50.6);

T = [eye(150); -inv(K(151:end,151:end))\*K(151:end,1:150)];

M = T'\*M\*T;

K = T'\*K\*T;

%% Generate a function that computes effective mass and use it to rank the

%fixed interface modes of the craft (PHIsort). Select a set of target modes

%based on the fact that each of them needs at least 5% of the effective

%mass in each of the six rigid body directions. Determine the total

%effective mass for the target mode set in each rigid body direction and

%comment on the sets dynamic completeness.

% M = wtmass.\*M; %Convert to propper units.

[E,MR,MRo,Msort,Ksort,dofA,dofO] = getEffectiveMass(M,K,PHIFI,aSET,DOF1,PHIR);

Esum = sum(E);

%%

% DOF1 = dofO;

% MR = wtmass.\*MR;

% MRo = wtmass.\*MRo;

%%

disp('Effective Mass (30 modes):');

disp(Esum);

%Selects all modes that contribute to more than 4.65% of mass in any rigid

%body direction. The assignment specifies 5%, however Dr. Kammer suggested

%"Things close to 5% RBM".

I = find(E>=0.048);

modindx = sort(mod(I,numel(E(:,1))),'ascend');

modindx = unique(modindx);

masscmplt = sum(E(modindx,:));

disp('Selected Modes Total Effective Mass:');

disp(masscmplt);

disp('The Modes that should be kept are:');

disp(modindx');

%Fixed interface M and K. These matrices were sorted in the

%getEffectiveMass call. (O-set partition)

PHIK = PHIFI(:,modindx); %Kept fixed-interface modes.

MF = Msort(13:end,13:end);

KF = Ksort(13:end,13:end);

%Generate functions to rank candidate sensor locations based on Modal

%Kinetic Energy and Effective Independence. Pick a single initial candidate

%set of sensor locations and use these functions to select a final set that

%will identify your target modes. The final sensor location should have

%n-target modes + 5 sensors.

[Efi,EFIdof,KEdof,efiDetQ,keDetQ] = getEffectiveIndependence(PHIK,MF,KF,numel(modindx)+5,dofO);

%Use MAC and any other measure to determine which of the sensor sets

%(initial and final) produces the most independent target mode partitions

%and the greatest target mode response signal strength.

EFIdof = sort(EFIdof,'ascend'); %DOF in original 150x150 mass matrix.

KEdof = sort(KEdof,'ascend');

for i = 1:numel(EFIdof)

[efiDOFind(i),~] = find(EFIdof(i)==dofO);

end

for i = 1:numel(KEdof)

[keDOFind(i),~] = find(KEdof(i)==dofO);

end

efiDOFind = efiDOFind';

keDOFind = keDOFind';

disp('The DOF set selected by EFI are:');

disp(DOF1(EFIdof)');

disp('The DOF set selected by KE are:');

disp(DOF1(KEdof)');

grid on;

%% DEBUG ONLY

% i = [ 130:135, 142:147]';

% j = setxor(1:150,i);

% [PHI,D,wn,wnhz,sortindx] = getEigSort(K(j,j),M(j,j));

%%

%Static reductions to EFIdof and KEdof.

[Kke,Mke,keREDind,keREDcomp,Kkesort,Mkesort] = ...

getStaticTAM(KF,MF,[1:138]',keDOFind);

[Kefi,Mefi,efiREDind,efiREDcomp,Kefisort,Mefisort] = ...

getStaticTAM(KF,MF,[1:138]',efiDOFind);

%Eigenproblem for EFIdof and KEdof.

[efiPHI,efiD,efiwn,efiwnhz,efisortindx] = getEigSort(Kefi,Mefi);

[kePHI,keD,kewn,kewnhz,kesortindx] = getEigSort(Kke,Mke);

PHImacefi = PHI(efiREDind,modindx);

PHImacke = PHI(keREDind,modindx);

%MAC

efimac = mac(efiPHI,PHImacefi);

set(gca, 'XTickLabel', modindx);

title('Modal Assurance Criterion (MAC) between PHI\_e\_f\_i and PHI');

xlabel('Target Fixed-Interface Mode #');

efiSmac = mac(efiPHI,efiPHI);

title('Self - Modal Assurance Criterion (MAC) for PHI\_e\_f\_i');

ylabel('PHI\_e\_f\_i Mode #');

xlabel('PHI\_e\_f\_i Mode #');

kemac = mac(kePHI,PHImacke);

set(gca, 'XTickLabel', modindx);

title('Modal Assurance Criterion (MAC) between PHI\_k\_e and PHI');

xlabel('Target Fixed-Interface Mode #');

keSmac = mac(kePHI,kePHI);

title('Self - Modal Assurance Criterion (MAC) for PHI\_k\_e');

ylabel('PHI\_k\_e Mode #');

xlabel('PHI\_k\_e Mode #');

%Cross and Self Orthogonality

% [keCO] = corl8(kePHI,PHImacke,Mke);

[Cke,~,~] = Kammercorl8(kePHI,Mke,PHImacke);

set(gca, 'XTickLabel', modindx);

title('Cross Orthogonality (CO) between PHI\_k\_e and PHI');

ylabel('PHI\_k\_e Mode #');

xlabel('Target Fixed-Interface Mode #');

% [keSO] = corl8(kePHI,kePHI,Mke);

[Sefi,~,~] = Kammercorl8(kePHI,Mke,kePHI);

% set(gca, 'XTickLabel', modindx);

title('Self Orthogonality (SO) for PHI\_k\_e');

ylabel('PHI\_k\_e Mode #');

xlabel('Target Fixed-Interface Mode #');

% [efiCO] = corl8(efiPHI,PHImacefi,Mefi);

[Cefi,p1,p2] = Kammercorl8(efiPHI,Mefi, PHImacefi);

set(gca, 'XTickLabel', modindx);

title('Cross Orthogonality (CO) between PHI\_e\_f\_i and PHI');

ylabel('PHI\_e\_f\_i Mode #');

xlabel('Target Fixed-Interface Mode #');

% [efiSO] = corl8(efiPHI,efiPHI,Mefi);

[Sefi,~,~] = Kammercorl8(efiPHI,Mefi, efiPHI);

% set(gca, 'XTickLabel', modindx);

title('Self Orthogonality (SO) for PHI\_e\_f\_i');

ylabel('PHI\_e\_f\_i Mode #');

xlabel('Target Fixed-Interface Mode #');

%Frequency error

% efimodematching = [1,1; 2,4; 3,3; 4,5; 5,6; 7,8; 10,9; 11,12; 12,13; 15,18; 16,19]; %FULL SET

kemodematching = [1,1; 2,3; 3,4; 4,5; 5,6; 7,8; 8,9; 11,12; 12,13; 14,18; 15,19; 16,20; 17,21; 18,22]; %FULL SET (REDUCED IS THE SAME)

efimodematching = [1,1; 2,3; 3,4; 4,6; 5,5; 6,8; 8,9; 10,12; 11,13; 13,18; 14,19; 15,21]; %REDUCED BEGINING SET

% kemodematching = [1,1; 2,3; 3,4; 4,5; 5,6; 7,9; 11,12; 12,13; 14,18; 15,19;16,20;17,21; 18,22];

eficompare = [w(efimodematching(:,2)) efiwnhz(efimodematching(:,1))];

kecompare = [w(kemodematching(:,2)) kewnhz(kemodematching(:,1))];

1. https://sem.org/wp-content/uploads/2016/07/sem.org-IMAC-XI-11th-Int-11-39-6-Realization-Complex-Mode-Shapes.pdf [↑](#footnote-ref-1)