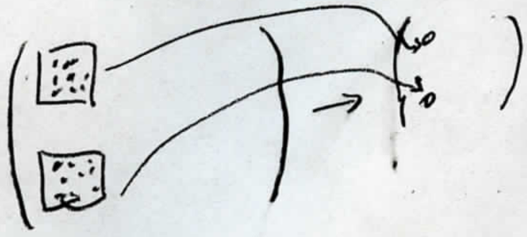
MRN 105 11-16-11

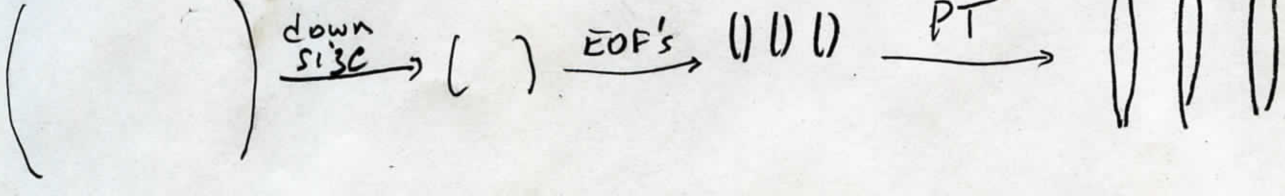
A student asked me how to compute EOF's for the earth in the case where the matrix of cross-correlations was verly large.

I suggested that one could block-average the matrix to produce a smaller matrix .



and then use first order perturbation Theory to correct the EOF's of the small matrix back to the large.

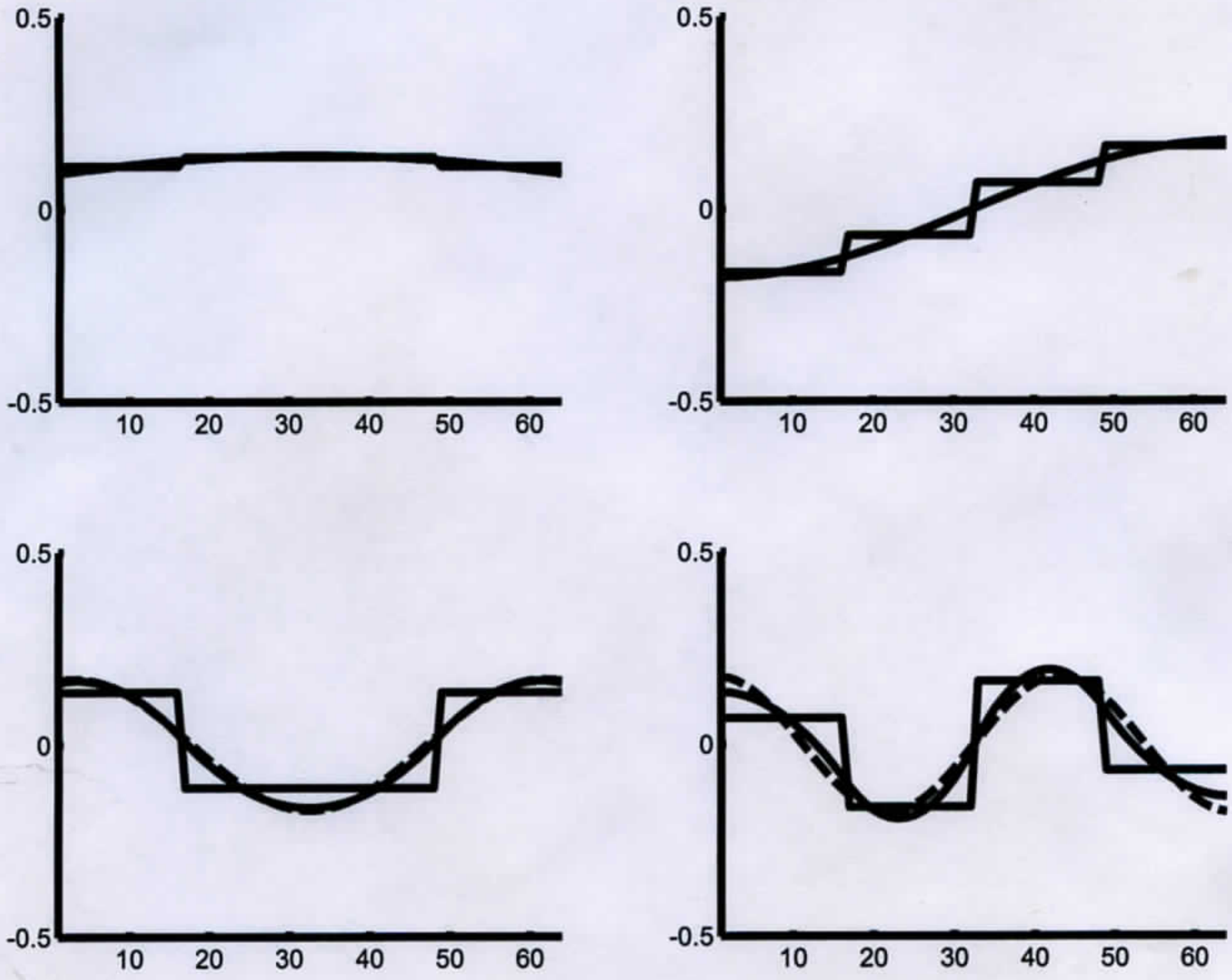
This only works for the number of EOF's needed to represent the small matrix, since the rest are degenerate, but a test seup indicated it works well for those



```
% compute the eigenvectors and eigenvalues of a big matrix of
% block-constant values given the eigenvalues and eigenvectors
% of the corresponding little matrix of values

clear all

N=64; % size of big matrix
r=16;  % size reduction factor (must divide evenly)
n=4;  % size of little matrix

% generate symmetric matrix A
sigma=0.001;
AP = toeplitz( [N:-1:1]', [N:-1:1] ) + random('Normal',0,sigma,N,N);
A = 0.5*( AP + AP' );

% little matrix is block-average of big matrix A
a = zeros(n,n);
for i=[1:n]
for j=[1:n]
    I = (i-1)*r+1;
    J = (j-1)*r+1;
    a(i,j)=mean(mean(A(I:I+r-1,J:J+r-1)));
end
end

% make a r-block version of A
% AM has r hy r constant blocks
AM = zeros(N,N);
for i=[1:n]
for j=[1:n]
    I = (i-1)*r+1;
    J = (j-1)*r+1;
    AM(I:I+r-1,J:J+r-1)=a(i,j)*ones(r,r);
end
end

% eigenvectors and values of little matrix
[v, d] = eig(a);

% eigenvectors and values of r by r block of ones
[vb, db] = eig(ones(r,r));

% two index arrays
i1 = [];
i2 = [];
for i=[1:n]
    i1 = [ i1, i*ones(1,r) ];
    i2 = [ i2, [1:r] ];
end
i1 = i1';
i2 = i2';
```

```
% reconstruct the eigenvalues of the block-averaged matrix AM
% it has n non-zero eigenvalues and N-n zero eigenvalues.
% first construct them
DUvec = zeros(N,1);
for j=[1:N]
    i = i1(j);
    k = i2(j);
    DUvec(j) = d(i,i)*db(k,k);
end
% then sort them, ones with big absolute value first,
% keeping track of original order
sDUvec = sign(DUvec);
[ DUvec, i3 ] = sort( abs(DUvec), 'descend' );
DUvec = DUvec .* sDUvec(i3);
% eigenvalue matrix
DU = diag(DUvec);

% reconstructed eigenvectors
UU = zeros(N,N);
for j=[1:N]

    % (i,k) or original ordering
    i = i1(i3(j));
    k = i2(i3(j));

    % r-basis
    vr = vb(:,k);
    c = db(k,k);

     % make a length-N eigenvector out of a length n eigenvector
     Uvec = zeros(N,1);
     for q=[1:n]
        J = (q-1)*r+1; % start of block
        Uvec(J:J+r-1)=v(q,i)*vr; % this is the key statement
                                 % each block combines its own r-basis
                                 % with the corresponding element of the
                                 % eigenvector of the little matrix
     end
     Uvec = Uvec/sqrt(Uvec'*Uvec); % normalize to unit length

     % save reconstructed eigenvectors & eigenvalues
     UU(:,j)=Uvec;

end

% test that eigenvalue reconstruction is really AM
AMR = UU*DU*UU';
E = sum(sum((AM - AMR).^2));
fprintf('AM reconstruction error: %f\n', E );
```

```
% perturbation theory for eigenvalues and eigenvectors of A
DA = A - AM;

% exact eigenvalues and eigenvecros for reference
[Vo, Do] = eig(A);
Dovec = diag(Do);
% sort eigenvalues, ones with big absolute value first,
% keeping track of original order
sDovec = sign(Dovec);
[ Dvec, ind ] = sort( abs(Dovec), 'descend' );
Dvec = Dvec .* sDovec(ind);
% eigenvalue matrix
D = diag(Dvec);
% eigenvector matrix
V=zeros(N,N);
for i=[1:N]
    V(:,i) = Vo(:,ind(i));
end

% we can only do the n non-zero eigenvalues, because
% the zero ones are degenerate and to do a degenerate
% perturbation problem we would have to a eigenvalue
% problem of size equal to the order or the degeneracy,
% which would defeat the purpose of this approximation
DVRvec = zeros(n,1);
VR = zeros(N,n);
JOFI = zeros(n,1);
signlist = ones(n,1);

for i=[1:n]

    fprintf('Unperturbed eigenvector %d\n', i );

    % the ith reconstructed eigenvector and eigenvalue
    Uvec = UU(:,i);
    Uval = DU(i,i);

    % the ith perturbed eigenvector and eigenvalue
    nuvec = Uvec;
    nuval = Uval;

    % which exact eigeinvector is Uvec closest to?
    [mymax, jofi] = max(abs(V'*Uvec));
    fprintf('    is closest to true eigenvector %d\n', jofi );

    % perturbation of eigenvalues
    % I believe that these are all zero
    % but I calculate them anyway
    DUval = Uvec'*DA*Uvec;
    nuval = nuval + DUval;
    DUvec = zeros(N,1);
    % perturbation of eigenvectors
    for j=[1:N]
        if( i==j ) % skip self
            continue;
        end
        DUvec = DUvec + UU(:,j)*(UU(:,j)'*DA*Uvec)/(Uval-DU(j,j));
    end
    nuvec = nuvec + DUvec;
    nuvec = nuvec / sqrt(nuvec'*nuvec);  % normalize to unit length

    % possibility that might be antiparallel to V(:,jofi)
    if ( (V(:,jofi)'*nuvec) < 0 )
        nuvec = -nuvec;
        Uvec = -Uvec;
        signlist(i) = -1;
    end

    % comparison;
    EU  = (V(:,jofi)-Uvec)'*(V(:,jofi)-Uvec);
    Enu = (V(:,jofi)-nuvec)'*(V(:,jofi)-nuvec);

    fprintf('    Unperturbed error %f  Perturbed error %f\n', EU, Enu);

    % save results
    DVRvec(i) = nuval;
    VR(:,i) = nuvec;
    JOFI(i) = jofi;

end

% plot first 4 eigenvectors
figure(1);
clf;
for i = [1:4]
    subplot(2,2,i);
    set(gca,'LineWidth',3);
    hold on;
    axis( [1, N, -0.5, 0.5 ] );
    plot( [1:N], signlist(i)*UU(:,i), 'r-', 'LineWidth', 3 );
    plot( [1:N], VR(:,i), 'g-', 'LineWidth', 3 );
    plot( [1:N], V(:,JOFI(i)), 'k--', 'LineWidth', 3 );
end
```


0.5
0
-0.5
10
20
30
40
50
60