```
%% Collect PE initial realizations to create Do-Pe modes

addpath(genpath('/home/deepakns/Matlab/DeepakUtils'));
addpath(genpath('/home/deepakns/Matlab/mseas/Test'));
addpath(genpath('/home/deepakns/Matlab/mseas/do_pe_plots'));
addpath('/software/Matlab/SeaWater');
addpath ('/home/deepakns/Matlab/mseas/do_init_for_pe/ThreeD_Modes/StrmFn_TS_modes');
% addpath ('/home/deepakns/Matlab/mseas/do_init_for_pe/ThreeD_Modes','-end')
% addpath ('/software/MSEASwork/PE_mask/trunk/Src');
% addpath ('/software/HOPS/Cond_Topo/Matlab');
% v2 - no dx,dy,dz
% v2p1 - read PE forecasts ensembles
%% Load params
createmodes_in;

km2cm = 1e5;
m2cm = 1e2;
km2m = 1e3;
grav = 980.6; % cm/s^2

x_cor = x_cor*km2cm; % in km
y_cor = y_cor*km2cm;

x_var = x_var*km2cm;
y_var = y_var*km2cm;


if isempty(out_file)
    out_dir = pi_file(1:find(pi_file==filesep,1,'last'));
end
out_dir1 = pwd;

% if ~exist(out_dir1,'dir')
%     mkdir(out_dir1)
% end


%% Load geometry from pi_file
ncpeoutid = netcdf(pi_file);
landt = ncpeoutid{'landt'}(:);
landv_ls = ncpeoutid{'landv'}(:); %landv is different from landt, it is 0 on land, 1 on the coa
and 2 on sea. So we want > 1.
landv = double(landv_ls>1);
%tgrid2 = ncpeoutid{'tgrid2'}(:);
%vgrid2 = ncpeoutid{'vgrid2'}(:);
%tgrid3 = ncpeoutid{'tgrid3'}(:);
%vgrid3 = ncpeoutid{'vgrid3'}(:); % vgrid2/3, tgrid2/3: not used
%vbath = abs(ncpeoutid{'vbath'}(:));
%tbath = abs(ncpeoutid{'tbath'}(:));
dxt = ncpeoutid{'gridx'}(1);
dyt = ncpeoutid{'gridy'}(1);
dzt = ncpeoutid{'dzt'}(:)*100; % in cm
dzv = ncpeoutid{'dzv'}(:)*100;
imt = ncpeoutid{'imt'}(:);
jmt = ncpeoutid{'jmt'}(:);
km = ncpeoutid{'km'}(:);
nt = ncpeoutid{'nt'}(:); %number of tracers
```

```
eta_subtide = ncpeoutid{'srfpress'}(1,:,:)./grav;
[ny nx] = size (eta_subtide);
n = [2:ny ny];
s = 1:ny;
e = [2:nx nx];
w = 1:nx;
%etav_subtide = 0.25.*(eta_subtide(n,w)+eta_subtide(n,e)+eta_subtide(s,w)+eta_subtide(s,e));
gfile = ncpeoutid.grd_file(:);
%temp_mean = ncpeoutid{'temp'}(1,:,:,:);
%salt_mean = ncpeoutid{'salt'}(1,:,:,:);
close(ncpeoutid)

%%
utotrealiz = zeros(jmt,imt,km,num_realiz);
vclinrealiz = zeros(jmt,imt,km,2,num_realiz);
vtotrealiz = zeros(jmt,imt,km,num_realiz);
vbarorealiz = zeros(jmt,imt,2,num_realiz);
temprealiz = zeros(jmt,imt,km,num_realiz);
saltrealiz = zeros(jmt,imt,km,num_realiz);
srfprealiz = zeros(jmt,imt,num_realiz);

tmask = repmat(landt,[1 1 km]);
vmask = repmat(landv,[1 1 km]);

tmask = repmat(reshape(tmask,jmt*imt*km,1),[1 num_realiz]);
vmask = repmat(reshape(vmask,jmt*imt*km,1),[1 num_realiz]);

pmask = repmat(reshape(landt,jmt*imt,1),[1 num_realiz]);

%volt = sqrt(reshape(dxt.*dyt.*dzt,jmt*imt*km,1));
%volv = sqrt(reshape(dxt.*dyt.*dzv,jmt*imt*km,1));
%areat = sqrt(dxt.*dyt);

if ~exist('ens_id','var')
    ens_id = 'Ens';
end

if ~exist('itt','var')
    itt = 1;
end
itt
if ~exist('ensncfilename','var')
    ensncfilename = 'pi_ini.nc';
end
misrun = false(num_realiz,1);

if ~isempty(skprun)
    misrun(skprun)=true;
end

for irealiz=1:num_realiz
    fprintf('irealiz=%03d\n',irealiz);
    outdir = sprintf('%s%03d',ens_id,irealiz);
    outfile = [outdir,filesep,ensncfilename];
    ncid = netcdf (outfile);
    if isempty(ncid) || length(ncid{'time'}(:))<itt || misrun(irealiz)==true
        misrun(irealiz) = true;
```

```
        continue
    end
    utotrealiz(:,:,:,irealiz) = ncid{'vclin'}(itt,:,:,:,1) + repmat(ncid{'vbaro'}(itt,:,:,1),[1
km]);
    vtotrealiz(:,:,:,irealiz) = ncid{'vclin'}(itt,:,:,:,2) + repmat(ncid{'vbaro'}(itt,:,:,2),[1
km]);
    vclinrealiz(:,:,:,:,irealiz) = ncid{'vclin'}(itt,:,:,:,:);
    vbarorealiz(:,:,:,irealiz) = ncid{'vbaro'}(itt,:,:,:);
    temprealiz(:,:,:,irealiz) = ncid{'temp'}(itt,:,:,:);
    saltrealiz(:,:,:,irealiz) = ncid{'salt'}(itt,:,:,:);
    srfprealiz(:,:,irealiz) = ncid{'srfpress'}(itt,:,:);
    close(ncid)
end

clear outfile;
clear ncid;
utotrealiz(utotrealiz>1e30)=nan;
vtotrealiz(vtotrealiz>1e30)=nan;
vclinrealiz(vclinrealiz>1e30)=nan;
vbarorealiz(vbarorealiz>1e30)=nan;
temprealiz(temprealiz>1e30)=nan;
saltrealiz(saltrealiz>1e30)=nan;
srfprealiz(srfprealiz>1e30)=nan;

vtotmean = zeros(2,jmt,imt,km);

vtotmean(1,:,:,:) = nanmean(utotrealiz,4);
vtotmean(2,:,:,:) = nanmean(vtotrealiz,4);
vclinmean=nanmean(vclinrealiz,5);
vbaromean=nanmean(vbarorealiz,4);
tempmean=nanmean(temprealiz,4);
saltmean=nanmean(saltrealiz,4);
srfpmean=nanmean(srfprealiz,3);

% size(vtotmean)
% size(vclinmean)
% size(vbaromean)
% size(tempmean)
% size(saltmean)
% size(srfpmean)
% exit
clear vclinrealiz vbarorealiz

Ru = reshape(utotrealiz,jmt*imt*km,num_realiz).*vmask;
ustd = std(utotrealiz,[],4);
clear utotrealiz;
Rv = reshape(vtotrealiz,jmt*imt*km,num_realiz).*vmask;
vstd = std(vtotrealiz,[],4);
clear vtotrealiz;
Rt = reshape(temprealiz,jmt*imt*km,num_realiz).*tmask;
tstd = std(temprealiz,[],4);
clear temprealiz;
Rs = reshape(saltrealiz,jmt*imt*km,num_realiz).*tmask;
sstd = std(saltrealiz,[],4);
clear saltrealiz;
Rp = reshape(srfprealiz,jmt*imt,num_realiz).*pmask;
clear srfprealiz;

```

```
Ru(:,misrun) = [];
Rv(:,misrun) = [];
Rt(:,misrun) = [];
Rs(:,misrun) = [];
Rp(:,misrun) = [];

vmask(:,misrun) = [];
tmask(:,misrun) = [];
pmask(:,misrun) = [];


Rumean = mean(Ru,2);
Rvmean = mean(Rv,2);
Rtmean = mean(Rt,2);
Rsmean = mean(Rs,2);
Rpmean = mean(Rp,2);

Ru = bsxfun(@minus,Ru,Rumean);
Rv = bsxfun(@minus,Rv,Rvmean);
Rt = bsxfun(@minus,Rt,Rtmean);
Rs = bsxfun(@minus,Rs,Rsmean);
Rp = bsxfun(@minus,Rp,Rpmean);
clear Rumean Rvmean Rtmean Rsmean Rpmean;

Runan = Ru;Runan(vmask==0)=nan;
Rvnan = Rv;Rvnan(vmask==0)=nan;
Rtnan = Rt;Rtnan(tmask==0)=nan;
Rsnan = Rs;Rsnan(tmask==0)=nan;
Rpnan = Rp;Rpnan(pmask==0)=nan;

Rustd = std(Runan(:),'omitnan');
Rvstd = std(Rvnan(:),'omitnan');
Rtstd = std(Rtnan(:),'omitnan');
Rsstd = std(Rsnan(:),'omitnan');
Rpstd = std(Rpnan(:),'omitnan');
clear Runan Rvnan Rtnan Rsnan Rsnan Rpnan;


% Ruvol = Ru.*repmat(volv,1,num_realiz);Ruvolnan = Ruvol;Ruvolnan(vmask==0)=nan;
% Rvvol = Rv.*repmat(volv,1,num_realiz);Rvvolnan = Rvvol;Rvvolnan(vmask==0)=nan;
% Rtvol = Rt.*repmat(volt,1,num_realiz);Rtvolnan = Rtvol;Rtvolnan(tmask==0)=nan;
% Rsvol = Rs.*repmat(volt,1,num_realiz);Rsvolnan = Rsvol;Rsvolnan(tmask==0)=nan;
% Rpvol = Rp.*areat;Rpvolnan = Rpvol;Rpvolnan(pmask==0)=nan;
%
% Ruvolstd = std(Ruvolnan(:),'omitnan');
% Rvvolstd = std(Rvvolnan(:),'omitnan');
% Rtvolstd = std(Rtvolnan(:),'omitnan');
% Rsvolstd = std(Rsvolnan(:),'omitnan');
% Rpvolstd = std(Rpvolnan(:),'omitnan');

R = [Ru./Rustd;Rv./Rvstd;Rt./Rtstd;Rs./Rsstd;Rp./Rpstd];
% Rvol = [Ruvol./Ruvolstd;Rvvol./Rvvolstd;Rtvol./Rtvolstd;Rsvol./Rsvolstd;Rpvol./Rpvolstd];
clear Ru Rv Rt Rs Rp;


R(isnan(R)) = 0;

```

```
% R(:,misrun) = [];

[U,S,V] = svds(R,num_modes_do);
% [Util,Stil,Vtil] = svds(Rvol,num_modes_do);

Z = V*S';
if ~isfile(strcat('SVD_',out_sfx,'.mat'))
        save(strcat('SVD_',out_sfx,'.mat'),'U','V','S','Z');
end

if ~isfile(strcat('SVD_full_itt_',num2str(itt),'.mat'))
        [U_full,S_full,V_full] = svds(R,num_realiz-length(skprun));
        Z_full = V_full*S_full';
        save(strcat('SVD_full_itt_',num2str(itt),'.mat'),'U_full','V_full','S_full','Z_full');
        clear Z_full U_full S_full V_full
        % return
        uncertainty_vs_modes
end
% Ztil = Vtil*Stil';
%keyboard

mmin = M_Min;                % minimum number of mixture components
mmax = M_Max;                % maximum number of mixture components
MaxIter = 2e03;            % maximum number of EM iterations
TolFun = 1e-07;            % EM termination tolerance
Options = statset('MaxIter', MaxIter, 'TolFun', TolFun);
Replicates = 5;           % number of EM random starts
fits = cell(mmax, 1);       % maximum likelihood GMM fits
BICs = zeros(mmax, 1);  % BICs of maximum likelihood GMM fits
% fit Yi GMM
warning('off','stats:gmdistribution:IllCondCov')
for m = mmin : mmax
    try
        Cur_Time = toc;
        switch GMM_Fit_Cov
            case 'Full_NotShared'
                fits{m} = fitgmdist(Z, m, 'Regularize', 0.000001,...
                    'CovType', 'full', 'SharedCov', false,...
                    'Replicates', Replicates, 'Options', Options);
            case 'Full_Shared'
                fits{m} = fitgmdist(Z, m, 'Regularize', 0.01,...
                    'CovType', 'full', 'SharedCov', true,...
                    'Replicates', Replicates, 'Options', Options);
            case 'Diag_NotShared'
                fits{m} = fitgmdist(Z, m, 'Regularize', 0.01,...
                    'CovType', 'diagonal', 'SharedCov', false,...
                    'Replicates', Replicates, 'Options', Options);
            case 'Diag_Shared'
                fits{m} = fitgmdist(Z, m, 'Regularize', 0.01,...
                    'CovType', 'diagonal', 'SharedCov', true,...
                    'Replicates', Replicates, 'Options', Options);
        end
        BICs(m) = fits{m}.BIC;
        fprintf('GMM %02d fit: t = %2.2f s, LL = %06.2e, BIC =%06.2e\n',...
            m, toc-Cur_Time, -fits{m}.NlogL, -BICs(m));
    catch err
        fprintf('GMM %02d fit: t = %2.2f s, error\n', m, toc-Cur_Time)
```

```
    end
end
dlmwrite(strcat('createmodes_in_',out_sfx,'.txt'),strcat(BICs),'-append');
[~,M] = min(BICs(mmin:mmax)) ;     % number of components with minimum BIC
M     = M + (mmin-1) ;             % (have to correct the index if mmin>1)
GMMf  = fits{M} ;           % prior phi GMM
pijf  = GMMf.PComponents ;  % component weights
mujf  = GMMf.mu' ;          % component means (note: transposed)
Sigjf = GMMf.Sigma ;        % component covariances
% switch GMM_Fit_Cov
%     case 'Full_Shared'
%         Sigjf = reshape(repmat(Sigjf, 1, M), app.S, app.S, M);
%     case 'Diag_NotShared'
%         Temp_Sig = Sigjf;
%         Sigjf = zeros(app.S, app.S, M);
%         for GMM_Comp = 1 : M
%             Sigjf(:,:,GMM_Comp) = diag(Temp_Sig(:,:,GMM_Comp));
%         end
%     case 'Diag_Shared'
%         Sigjf = diag(Sigjf);
%         Sigjf = reshape(repmat(Sigjf, 1, M), app.S, app.S, M);
% end



%ustd = std(utotrealiz,[],4);
%vstd = std(vtotrealiz,[],4);
%tstd = std(temprealiz,[],4);
%sstd = std(saltrealiz,[],4);

% SS = diag(S);
SS = var(Z);

% coeff = normrnd(0,1, num_realiz_do, num_modes_do);
% docoeffs = coeff.*repmat(sqrt(SS(1:num_modes_do)),[num_realiz_do, 1]);

docoeffs = random(GMMf,num_realiz_do);

[sval,sid] = sort(docoeffs(:,1));
docoeffs = docoeffs(sid,:);
Yi = docoeffs;

ui = U(1:jmt*imt*km,:).*Rustd;
vi = U(jmt*imt*km+(1:jmt*imt*km),:).*Rvstd;
ti = U(2*jmt*imt*km+(1:jmt*imt*km),:).*Rtstd;
si = U(3*jmt*imt*km+(1:jmt*imt*km),:).*Rsstd;
spi = U(4*jmt*imt*km+(1:jmt*imt),:).*Rpstd;

% UI = bsxfun(@times,volv,ui);
% VI = bsxfun(@times,volv,vi);
% TI = bsxfun(@times,volt,ti);
% SI = bsxfun(@times,volt,si);
% SPI = bsxfun(@times,areat,spi);

UI = ui;
VI = vi;
TI = ti;
```

```
SI = si;
SPI = spi;

if rewrite_appip == 1
    varm  = sum(repmat(var(docoeffs),size(UI,1),1).*(UI.^2),2);
    sdvar(1) = sqrt(mean(varm(:)));
    varm  = sum(repmat(var(docoeffs),size(VI,1),1).*(VI.^2),2);
    sdvar(2) = sqrt(mean(varm(:)));
    varm  = sum(repmat(var(docoeffs),size(SPI,1),1).*(SPI.^2),2);
    sdvar(3) = sqrt(mean(varm(:)));
    varm  = sum(repmat(var(docoeffs),size(TI,1),1).*(TI.^2),2);
    sdvar(4) = sqrt(mean(varm(:)));
    varm  = sum(repmat(var(docoeffs),size(SI,1),1).*(SI.^2),2);
    sdvar(5) = sqrt(mean(varm(:)));
    appip([1 2 3 4 5]) = 1./sdvar([1 2 3 4 5]).^2;
    %     appip(3) = 1./(dxt*dyt);
elseif rewrite_appip == 2
    appip = [1/Rustd.^2 1/Rvstd.^2 1/Rpstd.^2 1/Rtstd.^2 1/Rsstd.^2];
end


Gram = appip(1).*UI'*UI + appip(2).*VI'*VI + appip(3).*SPI'*SPI + appip(4).*TI'*TI +
appip(5).*SI'*SI ;

% [VC, DC] = eig(Gram); % Gram is the innerproduct with appip included
% DC = sqrt(DC);
% Yi = Yi * VC * DC * VC; % Update coefficients to ensure realizations are preserved.
% DCR = diag(1./diag(DC)); % Inverse of eigenvalue matrix needed for the next two lines.
% ui = ui * VC * DCR * VC;      % This multiplication by VC and DC ensures that the inner produ
of the new modes is I.
% vi = vi * VC * DCR * VC;
% ti = ti * VC * DCR * VC;
% si = si * VC * DCR * VC;
% spi = spi * VC * DCR * VC;
%
% CYY = cov(Yi);
% [VC2, ~] = eig(CYY) ; %Diagonal Covariance, and Diagonal Vectors
% ui = fliplr(ui * VC2); % Preserve orthonormality of the modes.
% vi = fliplr(vi * VC2); % Preserve orthonormality of the modes.
% ti = fliplr(ti * VC2); % Preserve orthonormality of the modes.
% si = fliplr(si * VC2); % Preserve orthonormality of the modes.
% spi = fliplr(spi * VC2); % Preserve orthonormality of the modes.
% Yi = fliplr(Yi * VC2);
Yi_mean = mean(Yi);
%if ~isfile(strcat('Y_mean_',out_sfx,'.mat'))
%       save(strcat('Y_mean_',out_sfx,'.mat'),'Yi_mean');
%end
Yi = bsxfun(@minus,Yi,Yi_mean);
clear Yi_mean;

utotmode_s = reshape(ui,jmt,imt,km,num_modes_do);
vtotmode_s = reshape(vi,jmt,imt,km,num_modes_do);
tempmode_s = reshape(ti,jmt,imt,km,num_modes_do);
saltmode_s = reshape(si,jmt,imt,km,num_modes_do);
srfpmode = reshape(spi,jmt,imt,num_modes_do);

docoeffs = Yi;

```

```
%% Decompose to clinic and baro

Hv = sum(dzv,3);
fracv = 1+eta_subtide./Hv;
Hv = Hv.*fracv;
ubaromode = zeros(jmt,imt,num_modes_do);
vbaromode = zeros(jmt,imt,num_modes_do);
uclmode = zeros(size(utotmode_s));
vclmode = zeros(size(vtotmode_s));
utotmode_test = uclmode;
vtotmode_test = vclmode;

DZVZ = dzv.*repmat(fracv,[1 1 km]);


for imodes=1:num_modes_do
    ftmp = squeeze(utotmode_s(:,:,:,imodes)).*DZVZ;
    ftmp = sum(ftmp,3);
    ubaromode(:,:,imodes) = squeeze(ftmp./Hv);
    uclmode(:,:,:,imodes) = squeeze(utotmode_s(:,:,:,imodes))-
repmat(squeeze(ubaromode(:,:,imodes)),[1 1 km]);
    utotmode_test(:,:,:,imodes) = uclmode(:,:,:,imodes)+repmat(squeeze(ubaromode(:,:,imodes)),[
km]);
    ftmp = squeeze(vtotmode_s(:,:,:,imodes)).*DZVZ;
    ftmp = sum(ftmp,3);
    vbaromode(:,:,imodes) = squeeze(ftmp./Hv);
    vclmode(:,:,:,imodes) = squeeze(vtotmode_s(:,:,:,imodes))-
repmat(squeeze(vbaromode(:,:,imodes)),[1 1 km]);
    vtotmode_test(:,:,:,imodes) = vclmode(:,:,:,imodes)+repmat(squeeze(vbaromode(:,:,imodes)),[
km]);
end
% ui1 = reshape(utotmode_test(2:end-2,2:end-2,:,:),(jmt-3)*(imt-3)*km,num_modes);
% vi1 = reshape(vtotmode_test(2:end-2,2:end-2,:,:),(jmt-3)*(imt-3)*km,num_modes);
% ti1 = reshape(tempmode_s(2:end-2,2:end-1,:,:),(jmt-3)*(imt-2)*km,num_modes);
% si1 = reshape(saltmode_s(2:end-2,2:end-1,:,:),(jmt-3)*(imt-2)*km,num_modes);
% spi1 = reshape(srprmode(2:end-1,2:end-1,:),(jmt-2)*(imt-2),num_modes);
%
% UI1 = bsxfun(@times,sqrt(dxt*dyt*dzv_1d),ui1);
% VI1 = bsxfun(@times,sqrt(dxt*dyt*dzv_1d),vi1);
% TI1 = bsxfun(@times,sqrt(dxt*dyt*dzt_1d),ti1);
% SI1 = bsxfun(@times,sqrt(dxt*dyt*dzt_1d),si1);
% SPI1 = bsxfun(@times,(sqrt(dxt*dyt)).*reshape(landt(2:end-1,2:end-1),(jmt-2)*(imt-2),1),spi1)
%
%
% Gram1 = appip(1).*UI1'*UI1 + appip(2).*VI1'*VI1 + appip(3).*SPI1'*SPI1 + appip(4).*TI1'*TI1 +
appip(5).*SI1'*SI1 ;

%% Prepare for call to add_modes
tracermodes = zeros([num_modes_do km imt jmt nt]);
tracermodes(:,:,:,:,1) = permute(tempmode_s,[4 3 2 1]);
tracermodes(:,:,:,:,2) = permute(saltmode_s,[4 3 2 1]);
uclinmodes = zeros([num_modes_do 2 km imt jmt]);
uclinmodes(:,1,:,:,:) = permute(uclmode,[4 3 2 1]);
uclinmodes(:,2,:,:,:) = permute(vclmode,[4 3 2 1]);
ubaromodes = zeros([num_modes_do 2 imt jmt]);
ubaromodes(:,1,:,:) = permute(ubaromode,[3 2 1]);
ubaromodes(:,2,:,:) = permute(vbaromode,[3 2 1]);
%srfpressmode = zeros([num_modes imt jmt]);
```

```
srfpressmode = permute(srfpmode,[3 2 1]);

%% Add modes to the ncoutfile

if ~isempty(out_sfx)
    out_file = [out_dir1,filesep,'pido_',out_sfx,'_GMM',num2str(M),'.nc'];
else
    out_file = [out_dir1,filesep,'pido.nc'];
end

tracernames = {'temp' 'salt'};
ncid = add_modes_v5
(pi_file,out_file,docoeffs,tracermodes,tracernames,uclinmodes,ubaromodes,srfpressmode,appip);%w
starting from nomodes

netcdf.close(ncid);

ncid = netcdf(out_file,'write');
ncid{'num_modes'}(:) = num_modes_do;
ncid{'num_realiz'}(:) = num_realiz_do;

% ncid{'vclin'}(:,:,:,:) = vclinmean;
% ncid{'vbaro'}(:,:) = vbaromean;
% ncid{'vtot'}(:,:,:,:) = vtotmean;
% ncid{'temp'}(:,:,:) = tempmean;
% ncid{'salt'}(:,:,:) = saltmean;
% ncid{'srfpress'}(:,:) = srfpmean;

close(ncid);

% fid = fopen([out_dir1,filesep,'appip.txt'],'w');
dlmwrite([out_dir1,filesep,'appip',out_sfx,'.txt'],appip,' ')
```