```
%% Supplement title: MATLAB function for generating predictions using the SSR
forecasting method.

function [bestMSEandPhis,bestPreds] = SSR_forecast_code(train,test,p,phis)
% This function creates forecasts using state space reconstruction (SSR).
% Here, test should be the last x points in the train set.
% The forecasts are compared to the test set using the specified params.
%
% Input variables: 'train', 'test', 'p', 'phis'
% 'train' is the training set
% 'test' is the test set
% 'p' is the number of steps ahead to predict
% 'phis' are the weight parameters (one for each species)

%2. Log input data to account for multiplicative error
train0=log(train+1); test0=log(test+1);

%3. Get info about the data.
ltr=size(train0,1); lte=size(test0,1);
num_sp=size(train0,2);

%4. Set the embedding dimension.
dim=3;

%% 5. Set up the master difference matrix. This will be a 3-d matrix, where
% each sheet is a species and each r,c is the difference between point r of
% the train set and point c of the test set.

% a. Shift train data so it's 3d.
train3d=reshape(train0,ltr,1,num_sp);
train3d_rep=train3d(:,ones(size(test0,1),1),:);

% b. Transpose and shift test data so it is 3-d.
test3d=shiftdim(test0,-1); %<-Takes the transpose then turns it into a 3-d
matrix.
test3d_rep=test3d(ones(size(train0,1),1),:,:); %repeats each col until it is
length(train) wide

% c. Calculate difference b/t each train and test point for all species.
master_diff = (train3d_rep-test3d_rep).^2; %<-Take square root later.


%% 6. Generate distance matrix for each species.
        index2=diff_index(ltr,lte,dim);  %<-Generate the index.

        for i=1:num_sp
        master_diff2=master_diff(:,:,i); %<-Extract the species diff sheet
        master_diff3(:,:,i)=master_diff2(index2);
        end
        % In 2-d, each pair of columns in the master_diff3 matrix is the
        % distance between a pair of test points and every training point.
        % So each pair should then be summed and square rooted to calculate
        % the distance matrix. This will then be weighted later to make the
        % predictions.
        % So, next, sum the columns in the matrix, grouped according to the
        % embedding dimension.
```

```
        %b.
        for k=1:size(master_diff3,2)/dim
            c=1+(k-1)*dim;%<-Counter
            master_dist(:,k,:)=...
                (sum(master_diff3(:,c:c+dim-1,:),2)).^.5;%<-Took square root
here
        end


%7. Create a phi matrix
phi_vect=linspace(0,500,15);
phi_permute=phi_vect(combinator(length(phi_vect),num_sp,'p','r'));

% Trap that allows code to run if num_sp==1
if num_sp==1
    phi_permute=phi_permute';
end

% Trap that bypasses search if we have already fit the NL model
if isempty(phis)==0 %If it is not empty
    phi_permute=phis;
end


% 8. This multiplies each combination of distances by a unique phi.  So it
goes
% through the entire phi_index for each element of the d_index and
% multiplies it by phi_index(i).
%
% So, dists_weighted(r,c) is the rth dimension combination, and the cth
% phi combination.  **So dists_weighted should be length(d_index) long and
% length(phi_index) wide.

% Set up phi matrix for 3-d multiplications.
phi2use=reshape(phi_permute,[size(phi_permute,1),1,num_sp]);
% Repeat the master dist matrix row-wise so it can be multiplied by phi2use
master_distRep=repmat({master_dist},[size(phi2use,1),1,1]);
distsWeighted=cell(size(master_distRep,1),1);
for j=1:size(phi2use,1)
    distsWeighted{j}=multiprod(master_distRep{j},-1*phi2use(j,1,:));
end


%% 9. Make predictions

        preds=cell(size(master_distRep)); %<-Holder for predictions
        mse=-9999*ones(size(master_distRep));%<-Holder for MSE calculations

for i=1:size(master_distRep,1)
    % The final weight matrix is formed by summing the weight matrices
    % for each species.
    w=exp(sum(distsWeighted{i},3)); %weight matrix

    % Chop off the end of w because we can't make predictions for
    % points > length(train)-p.
    % twpCol stands for 'total weight per column'.
    twpCol=sum(w(1:end-p,:),1);
```

```
    wShort=w(1:end-p,:)./twpCol(ones(size(w,1)-p,1),:);

    wShortT=wShort'; %Transpose the w_chopped matrix

    %Multiply the weight matrix by the train(max_d+p) to get
    %the predictions. Here, preds(r,1,s) each s is a species and
    %each r is the prediction for test point r+dim-1.

    preds0=multiprod(wShortT(:,:,ones(num_sp,1)),train3d(dim+p:end,:,:));

    %Exponentiate so that the mse can be compared to the other methods
    %subtract one for same reason.
    preds{i}=exp(preds0)-1; test1=exp(test0)-1;

    %Calculate the Mean Squared Error.
    mse(i)=sum((preds{i}(1:end-p,1,1)-
test1(p+dim:end,1)).^2)./length(preds{i}(1:end-p,1,1));

    %Plot Actual vs Pred
    %{
    figure(num_sp), plot(preds{i}(1:end-p,1,1),'o-b'),legend('predicted'),
    hold on,
    plot(test1(p+dim:end,1),'o-r'),
    pause(.01)
    hold off,
    %}

end
%% Find optimal phi and dimension
[bestMSE,mseInd]=min(mse);
bestPhis=phi_permute(mseInd,:);
bestPreds=preds{mseInd};
bestMSEandPhis=[bestMSE,bestPhis];

% Plot the best Actual vs Predicted for the best phi
%{
if isempty(bestPhis)==0
figure(num_sp), plot(preds{mseInd}(1:end-p,1,1),'o-b'), legend('predicted')
hold on
plot(test1(p+dim:end,1),'o-r'), pause
hold off
end
%}
```