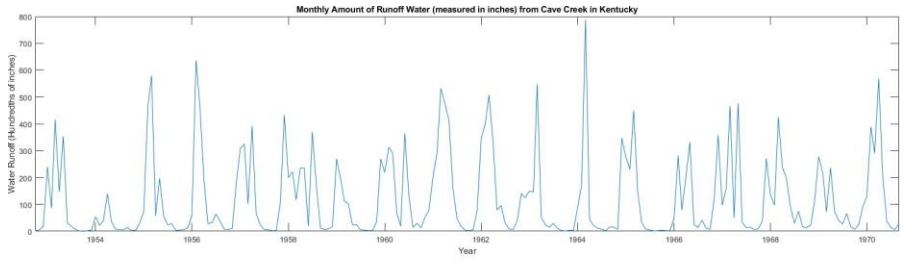
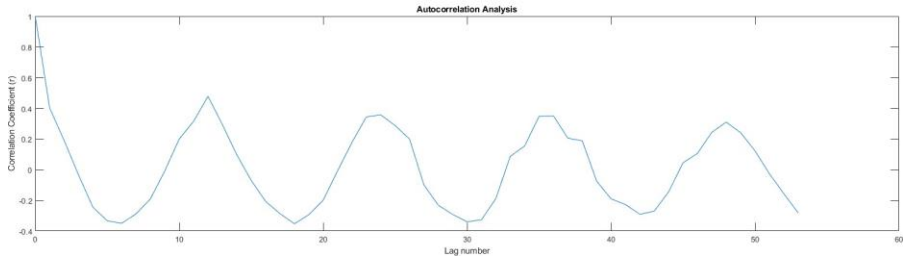
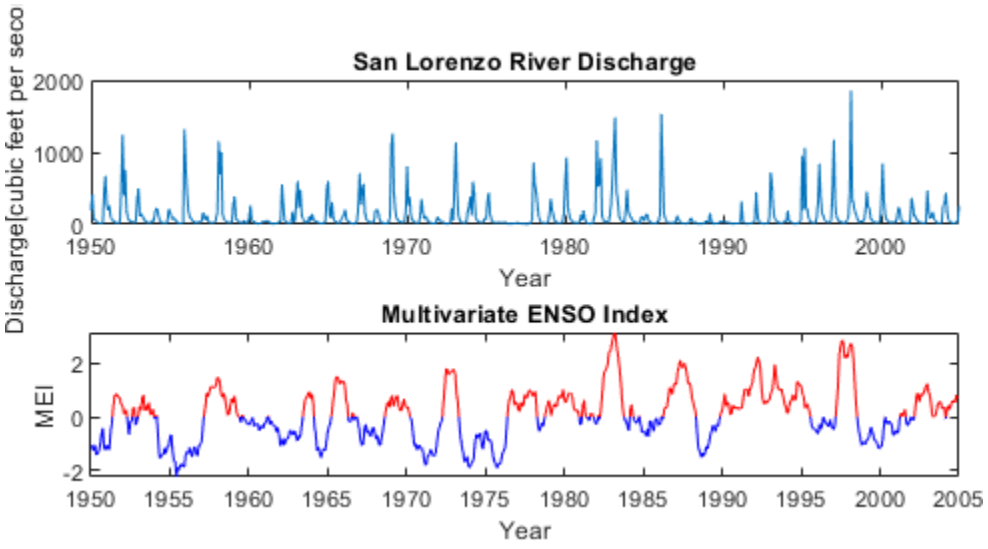
```
clear variables
load cavecreek.txt
time=cell(1,length(cavecreek));
month=10;
for i=1:length (cavecreek)
    temp=datestr(datenum(1952,month,1,0,0,0));
    time{i}=temp(4:end);
    month=month+1;
    end
time=time(1:10);
%plot(cavecreek)
figure(1)
subplot(2,1,1)
t53=datetime(1952,10:225,1);
datetick()
plot(t53',cavecreek)
%set(gca,'xticklabel','time','XTick',0:10:length(cavecreek));
%xlim([0,length(cavecreek)])

xlabel('Year')
ylabel('Water Runoff (Hundredths of inches)')
title('Monthly Amount of Runoff Water (measured in inches) from Cave
 Creek in Kentucky')


x = cavecreek;
N = length(x);
n=53;
cor = 1;
bi=0;
[lag1 r1]= ser_corr_fcn(x, n, cor, bi);
%step=10;
%time=time(1:step:end);
subplot(2,1,2)
plot(lag1, r1)
xlabel('Lag number');
ylabel('Correlation Coefficient (r)');
title('Autocorrelation Analysis')
```





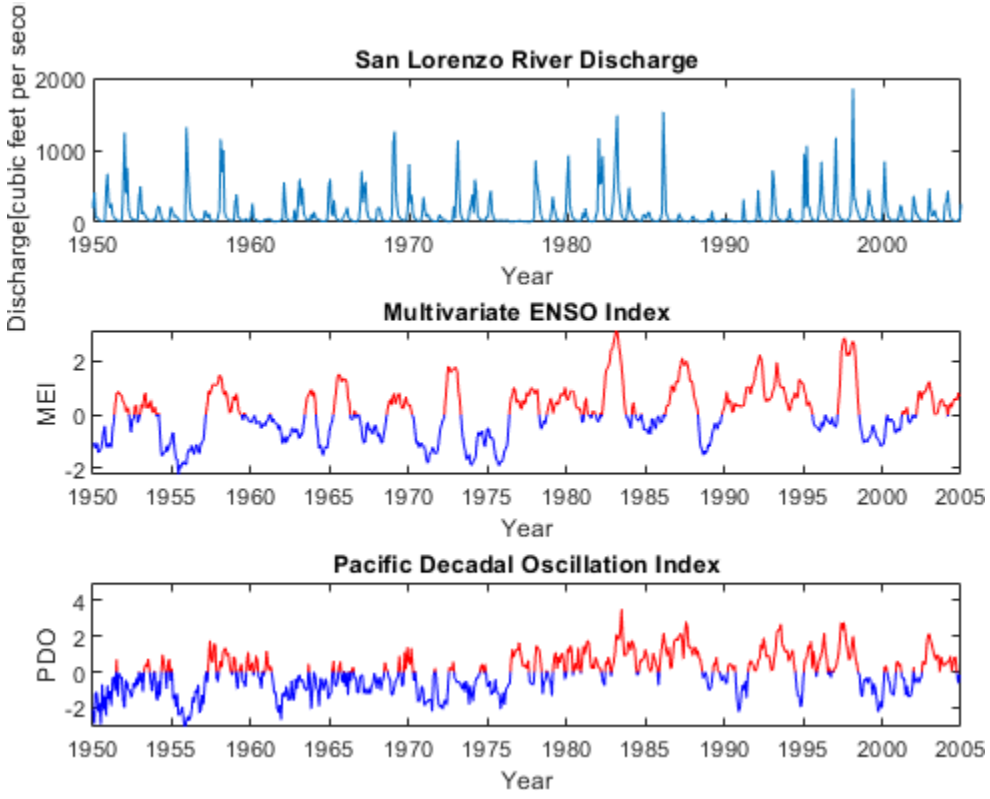

```
%Multivariate ENSO Index
%Pacific Decadal Oscillation Index
%San Lorenzo River Discharge

%usgs=importdata('usgs_monthly_slr.txt');
%mei=importdata('MEI_clipped.txt');
```

```
%pdo=importdata('pdo.txt');
%plot(pdo)
fid = fopen('usgs_monthly_slr.txt');
% read the first 37 lines and do nothing to them
for i=1:37
    tline=fgets(fid);
end
%Now read all the other ones
for i=1:840
    tline=fgets(fid);
    temp=str2num(tline(23:end));

    USGS (i,1:3)=temp;
end
%reading only the 3rd column

t1=datetime(1950,1:660,1);
USGS=USGS(160:819,:);
figure(2)
datetick()
subplot(3,1,1)
datetick()
plot(t1,USGS(:, 3))
title('San Lorenzo River Discharge')
xlabel('Year')
ylabel('Discharge[cubic feet per second]')


MEI=zeros (55,13);
fid = fopen('MEI_clipped.txt');
for i=1:10
    tline=fgetl(fid);
end

for i=1:55
    tline=fgets(fid);
    temp=str2num(tline);
    MEI (i,1:13)=temp;

end
t2=datetime(1950,1:660,1);
%A=1:55
%B=reshape(A,715,1)
MEI1=MEI(:,2:end)';
MEI2= MEI1(:)';
subplot(3,1,2)
t5=datenum(t2);

%plot(t2,MEI)
splitcolorplot(t5,MEI2, 0, 'r', 'b')
datetick()
title('Multivariate ENSO Index')
xlabel('Year')
ylabel('MEI')
```



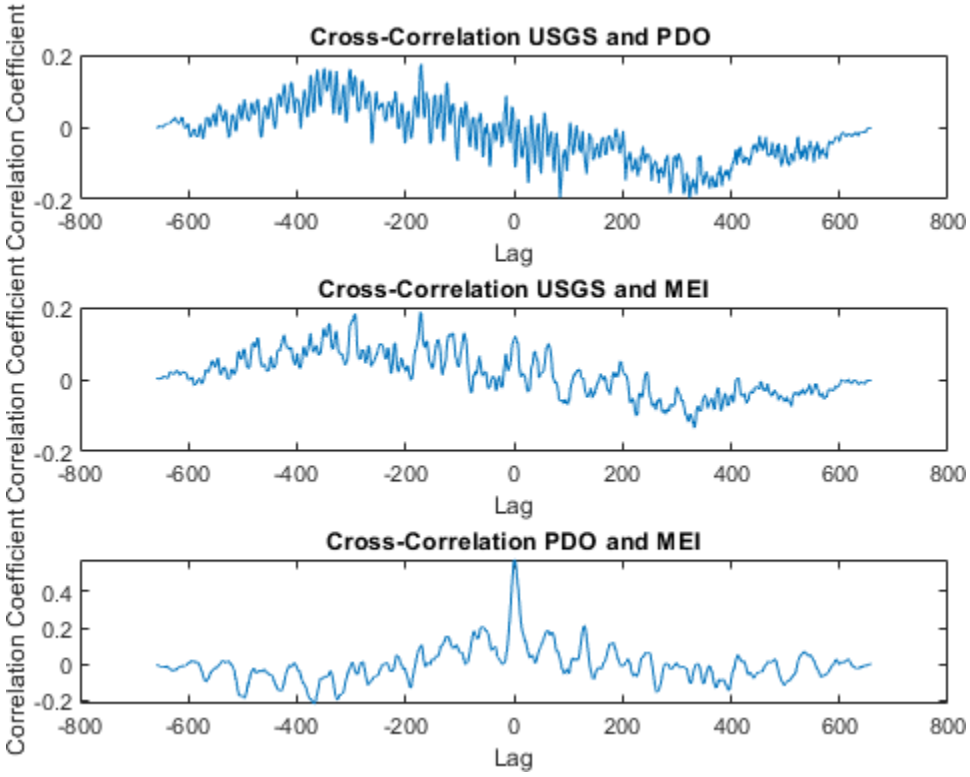

```
fid = fopen('pdo.txt');
for i=1:55
    tline=fgets(fid);
    temp=str2num(tline);
    pdo(i,1:13)=temp;
end
t3=datetime(1950,1:660,1);
pdo = pdo(:, 2:end);
pdo = reshape(pdo.',1,[]);


%datestring

%pdo=(t3,pdo);
subplot(3,1,3)
%{
aboveLine = pdo2 > 0; % or whatever.
hold on;
plot(t3(aboveLine), pdo2(aboveLine),'red')
plot(t3(~aboveLine), pdo2(~aboveLine), 'blue')
hold off
%}
%datestringdatenum,(datetime)(t3)
t4=datenum(t3);
step=36;
```

```
%time_step= time(1)(step:end);
splitcolorplot(t4,pdo, 0, 'r', 'b')
datetick()
%xlabel('months from 1950')
%set(gca,'xticklabel','XTick',time_step
%datestr(t3))
%DateString = {'01/01/1950';'01/01/2004'};
%ticlocs = datetime(DateString);
title('Pacific Decadal Oscillation Index')
xlabel('Year')
ylabel('PDO')
```




```
reUSGS=reshape(USGS(1:end,3:end),[1,660]);
[rUSGS,lag]=xcorr(reUSGS,pdo,'coeff');
figure(3)
subplot(3,1,1)
plot(lag,rUSGS)
title('Cross-Correlation USGS and PDO ')
xlabel('Lag')
ylabel('Correlation Coefficient')

hold on
subplot(3,1,2)
[USGSMEI,lag]=xcorr(reUSGS,MEI2,'coeff');
%[USGSMEI,lag]=xcorr(MEI2,reUSGS,'coeff');
plot(lag,USGSMEI)
```

```
title('Cross-Correlation USGS and MEI')
xlabel('Lag')
ylabel('Correlation Coefficient')

subplot(3,1,3)
[PDOMEI,lag]=xcorr(pdo,MEI2,'coeff');
plot(lag,PDOMEI)
title('Cross-Correlation PDO and MEI')
xlabel('Lag')
ylabel('Correlation Coefficient')


hold off
```



```
%SPLITCOLORPLOT(Y)                              [index used for X & VAL =
 0]
%     SPLITCOLORPLOT(X,Y)                       [VAL = 0]
%     SPLITCOLORPLOT(Y,VAL)                     [index used for X]
%     SPLITCOLORPLOT(X,Y,VAL)
%     SPLITCOLORPLOT(X,Y,FMT1,FMT2)             [VAL = 0]
%     SPLITCOLORPLOT(Y,VAL,FMT1,FMT2)           [index used for X]
%     SPLITCOLORPLOT(X,Y,VAL,FMT1,FMT2)
%
%   Example:
%   >> x = 0:0.1:10;
%   >> y = sin(4*x);
%   >> splitcolorplot(x,y,0.4)
%   >> splitcolorplot(y,-0.2,'rx--','k*:')
%Parse inputs (nested function used to set x, y, val, and all
 necessary
%  format options)
checkinputs(nargin)
%  Find zero crossings ("zero" = specified level)
if isrow(y)
    idx = [0,find(sign(y(2:end)-val)~=sign(y(1:end-1)-val)),lngth];
else
    idx = [0;find(sign(y(2:end)-val)~=sign(y(1:end-1)-val));lngth];
end
nidx = length(idx);
%Make an invisible plot -- this means that splitcolorplot has the
 same
%  overwriting behavior as plot
plot(x,y,'visible','off')
%  Loop over each segment
for k=2:nidx
    %  Get first and last indices
    k1 = idx(k-1)+1;
    k2 = idx(k);
    %  Choose format, depending on whether y > level or not
    fmt = (y(k1)<=val) + 1;
    %  Plot line segment

 line(x(k1:k2),y(k1:k2),'color',clr{fmt},'marker',mrkr{fmt},'linestyle',lntp{fmt})
    %  Plot linear interpolation to beginning of next line segment
    if k<nidx
        %  Start point is last point of previous line segment
        %  End point is beginning of next segment
        x1 = x(k2);
        x2 = x(k2+1);
        %  Get associated y values
        y1 = y(k2);
        y2 = y(k2+1);
        %  Slope
        m = (y2-y1)/(x2-x1);
        %  Find point on line where y = val
        x0 = x1 + (val-y1)/m;
        %  Plot from end of first segment to y = val
        line([x1,x0],[y1,val],'color',clr{fmt},'linestyle',lntp{fmt})
```

```
        %  Change formats and plot from y = val to start of 2nd
 segment
        fmt = mod(fmt,2) + 1;
        line([x0,x2],[val,y2],'color',clr{fmt},'linestyle',lntp{fmt})
    end
end
    %  Parse all the inputs
    function checkinputs(n)
        %  As long as we have some inputs, get the length of the first
        %  input (sort out other possible errors later)
        if n
            lngth = length(x);
        end
        %  Assign values and defaults as best we can (and trust the
 later
        %  error checking to sort out any problems)
        %  How many Romans?!
        switch n
            %  i
            case 1
                % Single input is vector of y values
                y = x;
                % Everything else is default
                x = 1:lngth;
                val = 0;
                fmt1 = 'o-';
                fmt2 = 'o-';
            %  ite!
            case 2
                % Two inputs could be x & y or y & level
                % Two equal-sized vectors => x & y
                if isequal(size(x),size(y))
                    val = 0;
                % Scalar y => y is level (and x is actually y)
                elseif numel(y)==1
                    val = y;
                    y = x;
                    x = 1:lngth;
                else
                    error('x & y must be same size, and level must be
 a scalar')
                end
                fmt1 = 'o-';
                fmt2 = 'o-';
            %  if you don't understand these comments, you seriously
 need
            %  to watch "Life of Brian"
            case 3
                % x, y, and level are all set, so set default formats
                fmt1 = 'o-';
                fmt2 = 'o-';
            case 4
                % Four inputs => last two are formats
                fmt2 = fmt1;
```

```
                fmt1 = val;
                % First two inputs could be x & y or y & level
                % Two equal-sized vectors => x & y
                if isequal(size(x),size(y))
                    val = 0;
                % Scalar y => y is level (and x is actually y)
                elseif numel(y)==1
                    val = y;
                    y = x;
                    x = 1:lngth;
                else
                    error('x & y must be same size, and level must be
a scalar')
                end
            case 5
                % Nothing to do here
            otherwise
                % No.
                error('Wrong number of inputs')
        end
        %  Now error-check the hell out of the x, y, val, fmt1, & fmt2
that
        %  we ended up with.
        if ~isequal(size(x),size(y))
            error('x & y must be same size')
        elseif ~isvector(x)
            error('x & y must be vectors')
        elseif ~isscalar(val)
            error('Level must be a scalar')
        elseif ~isnumeric(x) || ~isnumeric(y) || ~isnumeric(val)
            error('x, y, and level must be numeric')
        elseif ~ischar(fmt1) || ~ischar(fmt2)
            error('Formats must be strings')
        end
        %  Parse and deconstruct the format strings
        %  Look for color specifiers first
        [clr{1},fmt1] = getcolor(fmt1,'b');
        [clr{2},fmt2] = getcolor(fmt2,[0 0.5 0]);
        %  And now for markers
        [mrkr{1},fmt1] = getmarker(fmt1);
        [mrkr{2},fmt2] = getmarker(fmt2);
        %  And what's left is a linestyle
        fmt1 = getlnstyle(fmt1,mrkr{1});
        fmt2 = getlnstyle(fmt2,mrkr{2});
        lntp = {fmt1,fmt2};
    end
    function [c,fmt] = getcolor(fmt,def)
        %  Possible values
        clrs = '[bgrcymkw]';
        %  See if fmt1 includes any of the possible values
        idx = regexp(fmt,clrs);
        if isempty(idx)
            % No?  Set default
            c = def;
```

```
        else
            % Yes?  Cool, that's the color.  Remove all color
specifiers
            % from the string.
            if numel(idx)~=1
                warning('SPLITCOLORPLOT:multipleColors',...
                    'More than one color detected')
            end
            c = fmt(idx(1));
            fmt(idx) = [];
        end
    end
    function [m,fmt] = getmarker(fmt)
        marks = '[.ox+*sdv^<>ph]';
        idx = regexp(fmt,marks);
        %  Special case: if there's a dot (.), we need to make sure it
        %  isn't part of the linestyle specifier -.
        if ~isempty(regexp(fmt,'.','once'))
            %  Go through all possibilities (.s) in reverse order --
            %  reverse because we will remove any that shouldn't be
there,
            %  and that will mess up the indexing
            for j=length(idx):-1:1
                %  If there's a - in front of the ., remove it from
the
                %  list of possibilities (it's a line specifier, not a
                %  marker)
                if idx(j)>1 && strcmp(fmt(idx(j)-1:idx(j)),'-.')
                    idx(j) = [];
                end
            end
        end
        % OK, now go through and work out marker
        if isempty(idx)
            % Nothing given?  Set default
            m = 'none';
        else
            % Otherwise, that's the marker.  Remove all marker
specifiers
            % from the string.
            if numel(idx)~=1
                warning('SPLITCOLORPLOT:multipleMarkers',...
                    'More than one marker type detected')
            end
            m = fmt(idx(1));
            fmt(idx) = [];
        end
    end
    function fmt = getlnstyle(fmt,m)
        %  If there's nothing left, no linestyle was specified.
However,
        %  the depends on whether a marker was given
        if isempty(fmt)
            % No marker (and no linestyle) => default solid line
```

```
                if strcmp(m,'none')
                    fmt = '-';
                % Marker given but no linestyle => no line (just marker)
                else
                    fmt = 'none';
                end
            %  Something was specified.  Was it valid?
            elseif ~ismember(fmt,{'-','--',':','-.'})
                error('Unknown format specification')
            end
        end
    end
```