Table of Contents

Cross-Sectional Area Analysis

Author: Manny Rodriguez Upwork Profile: upwork.com/fl/emmanueljrodriguez

```
% Requirements: the matlab code should output:
% 1. date/time (which is the A column in the 'time-series' sheet of
 the spreadsheet)
% 2. depth or Water Level (which is the B column in the 'time-series'
 sheet of the spreadsheet)
% 3. cross-sectional area for each water-level depth.

% Known issues:
% 1. Water levels greater than river bank depth are excluded from
% cross-sectional area calc.
% 2. Some water levels are LESS than zero. These values are excluded
 from
% cross-sectional area calc.
% 3. Some water levels were really close the river bank depth, and
% therefore only one x-coordinate point is output from the conditional
% statement. At least two x-coordinates (domain of integration) are
% needed to perform the integration using the trapz function,
% so any data point water level depths that contain only one x-
coordinate
% are excluded.

% The above conditions are applied to all data sets.
```

Read and Process Data

Import data

Date and Time

```
[~, date_time16] =
 xlsread('xsectionflow.11.13_xsArea.xlsx','WY16','A2:A35213');
```

```
[~, date_time17] =
 xlsread('xsectionflow.11.13_xsArea.xlsx','WY17','A2:A43174');
[~, date_time18] =
 xlsread('xsectionflow.11.13_xsArea.xlsx','WY18','A2:A48130');
[~, date_time20] =
 xlsread('xsectionflow.11.13_xsArea.xlsx','WY20','A2:A46071');
% The first output argument is ignored by using '~' which corresponds
 to
% numeric data; the second output argument applies to text data, which
% returns as a cell array of strings.
% Convert from cell array to datetime array.
dateTimeWL16 = datetime(date_time16,'InputFormat','MM/dd/yyyy h:mm:ss
 a');
dateTimeWL17 = datetime(date_time17,'InputFormat','MM/dd/yyyy h:mm:ss
 a');
dateTimeWL18 = datetime(date_time18,'InputFormat','MM/dd/yyyy h:mm:ss
 a');
dateTimeWL20 = datetime(date_time20,'InputFormat','MM/dd/yyyy h:mm:ss
 a');
```

Water Level / Depth

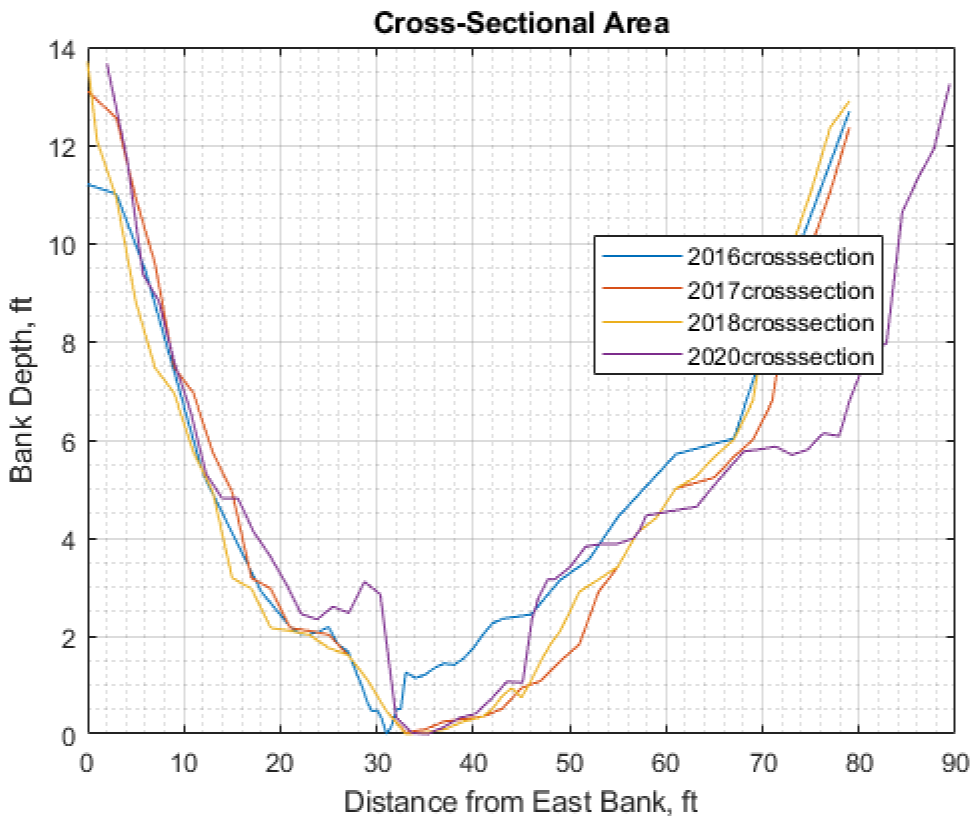
```
WL16 = xlsread('xsectionflow.11.13_xsArea.xlsx','WY16','B2:B35213');
WL17 = xlsread('xsectionflow.11.13_xsArea.xlsx','WY17','B2:B43174');
WL18 = xlsread('xsectionflow.11.13_xsArea.xlsx','WY18','B2:B48130');
WL20 = xlsread('xsectionflow.11.13_xsArea.xlsx','WY20','B2:B46071');

% **** original cross section ***************
%xdata = xlsread('cross sectional area.xlsx','cross
 section','A2:A42');
%ydata = xlsread('cross sectional area.xlsx','cross
 section','B2:B42');

% Invert ydata to produce a better representation of cross-section
%negydata = -1*ydata;

% _*x-y Coordinates*_
xdata16 =
 xlsread('xsectionflow.11.13_xsArea.xlsx','2016crosssection','A2:A42');
ydata16 =
 xlsread('xsectionflow.11.13_xsArea.xlsx','2016crosssection','B2:B42');
xdata17 =
 xlsread('xsectionflow.11.13_xsArea.xlsx','2017crosssection','A2:A40');
ydata17 =
 xlsread('xsectionflow.11.13_xsArea.xlsx','2017crosssection','B2:B40');
xdata18 =
 xlsread('xsectionflow.11.13_xsArea.xlsx','2018crosssection','A2:A47');
ydata18 =
 xlsread('xsectionflow.11.13_xsArea.xlsx','2018crosssection','B2:B47');
xdata20 =
 xlsread('xsectionflow.11.13_xsArea.xlsx','2020crosssection','A2:A56');
ydata20 =
 xlsread('xsectionflow.11.13_xsArea.xlsx','2020crosssection','B2:B56');
```

Translate data to place min point on the horizontal axis.

```
%trans_negydata = negydata + abs(min(negydata));
```

Plot Cross-Section

```
plot(xdata16,ydata16,xdata17,ydata17,xdata18,ydata18,xdata20,ydata20);
legend('2016crosssection','2017crosssection','2018crosssection','2020crosssection'

grid on;
grid minor;
title('Cross-Sectional Area');
xlabel('Distance from East Bank, ft');
ylabel('Bank Depth, ft');
```

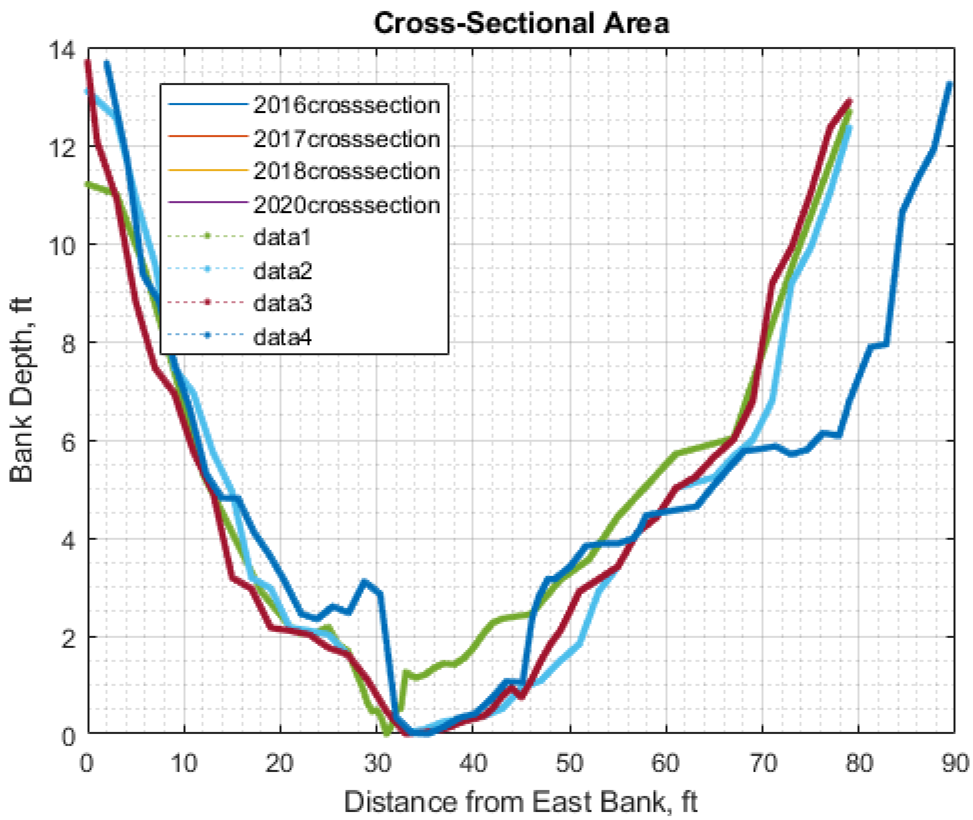


Interpolate data to increase resolution

Create "query points" Creates a column vector from 0 to max value in xdata array in increments of 0.1 - this defines the query points to be a finer sampling over the range of x.

```
res = 0.01; % Set resolution
qxdata16 = 0:res:max(xdata16);
qxdata17 = 0:res:max(xdata17);
qxdata18 = 0:res:max(xdata18);
```

```
qxdata20 = 0:res:max(xdata20);
%
% **** original cross section ***************
% Interpolate the function at the query points and plot the results.
qydata16 = interp1(xdata16,ydata16,qxdata16);
qydata17 = interp1(xdata17,ydata17,qxdata17);
qydata18 = interp1(xdata18,ydata18,qxdata18);
qydata20 = interp1(xdata20,ydata20,qxdata20);

hold on;
plot(qxdata16,qydata16,':.',qxdata17,qydata17,':.',qxdata18,qydata18,':.',qxdata20
hold off;
```



Cross-Sectional Area Calculations

2016

Calculate Cross-Section Area

Calculate the depth at water level

Program structure to find the integration limits according to the water level

```
nqydata16 = length(qydata16); % Outputs the size of the y-coordinate
 vector
```

```
nWL16 = length(WL16); % Size of Water Level vector

% Initialize variables column vectors to zeros
yWL16 = zeros(size(nWL16,1),1); % Assigns zeros to the first column of
 size nWL array of the first column (dimension)
xsAreaWL16 = zeros(size(WL16,1),1); % Assigns zeros to the first
 column of size WL array of the first column
timeIterCount = 0;
tic % start stopwatch timer to measure the time the code execution
 time
for l = 1:nWL16
    timeIterCount = timeIterCount + 1;
    % *******To be used with 'cross section' data:********
    %yWL(l) = min(negqydata) + WL(l); % Calcs the y-depth at Water
 Level for each date/time value

    yWL16(l) = min(qydata16) + WL16(l); % Calcs the y-depth at Water
 Level for each date/time value

    % The data set needs to be translated DOWN so that the water level
    % becomes the horizontal axis; this needs to be done for every
    % timeIterCount water level iteration.

    transWLqydata16 = qydata16 - yWL16(l);

    % Find the x-coordinates corresponding to the water level on
 either
    % side of the cross-section.
    % Initialize and set x and y variables to NaN
    x16 = [nan,nan];
    iterCount = 0;
    Int = 0;
    for k = 1:nqydata16
        % **** original cross section ***************
        % if isalmost(round(yWL(l),2), round(negqydata(k),2), 0.01)
 && ~isnan(WL(l)) % The isalmost function tests if water level value
 is approximately equal to the y-coordinate within the specified
 tolerance. Only perform the comparison if the cell is populated with
 numeric data.

        if isalmost(round(yWL16(l),2), round(qydata16(k),2), 0.01) &&
 ~isnan(WL16(l)) && WL16(l) < max(qydata16) && WL16(l) >= 0
            % The isalmost function tests if water level value
 is approximately equal to the y-coordinate within the specified
 tolerance. Only perform the comparison if the cell is populated with
 numeric data.
            % Some water levels are greater than the river bank depth,
 these
            % values are excluded. See known issues in the preamble.
            % Some water levels are less than zero, these values are
 excluded.

            % Note: Because rounding issues cause exact testing to
 fail to
```

```
            % find all relevant values in the water level vector a
            % tolerance is added.
            iterCount = iterCount + 1;
            x16(iterCount,:) = [k, qxdata16(k)]; % Stores the k-value
when true and the corresponding x-coordinate for the water level as a
column vector

        end
    end

    if ~isnan(WL16(l)) && WL16(l) < max(qydata16) && WL16(l) >= 0 &&
numel(x16(:,2)) > 1
        % ~isnan(WL16(l))
        % Some water level cells don't contain data, so these cells
are excluded from the xsAreaWL calculation.
        % WL16(l) < max(qydata16)
        % Some water levels are greater than the river bank depth,
these
        % values are excluded. See known issues in the preamble.
        % WL16(l) >= 0
        % Some water levels are less than zero, these values are
excluded.
        % numel(x16(:,2)) > 1
        % In some instances the water level is too close to the river
depth and therefore only
        % one x-coordinate is given; at least two x-coordinates are
needed to
        % perform the integration using trapz, so any data point water
        % level depths that contain only one x-coordinate are exluded.

        yWL_x16 = transWLqydata16(x16(1,1):x16(end,1)); % Extracts the
y-coordinates at the water level corresponding to the x-coordinates.
        % Set integration domain values
        xdom16 = x16(1,2):res:x16(end,2); % Important to use the last
value (end) of the array, since multiple points may be in close
proximity to each other - only the first and last coordinates are
needed to determine the water level boundary on the x-axis.

        % Hold cross-sectional area as a function of time, perform
trapezoidal
        % integration
        xsAreaWL16(timeIterCount) = abs(trapz(xdom16,yWL_x16));
    end
end
toc
```

Elapsed time is 526.890850 seconds.

2017

Calculate Cross-Section Area

Calculate the depth at water level

Program structure to find the integration limits according to the water level

```
nqydata17 = length(qydata17); % Outputs the size of the y-coordinate
 vector
nWL17 = length(WL17); % Size of Water Level vector

% Initialize variables column vectors to zeros
yWL17 = zeros(size(nWL17,1),1); % Assigns zeros to the first column of
 size nWL array of the first column (dimension)
xsAreaWL17 = zeros(size(WL17,1),1); % Assigns zeros to the first
 column of size WL array of the first column
timeIterCount = 0;
tic % start stopwatch timer to measure the time the code execution
 time
for l = 1:nWL17
    timeIterCount = timeIterCount + 1;
    % *******To be used with 'cross section' data:********
    %yWL(l) = min(negqydata) + WL(l); % Calcs the y-depth at Water
 Level for each date/time value

    % **** 2017crosssection ***************
    yWL17(l) = min(qydata17) + WL17(l); % Calcs the y-depth at Water
 Level for each date/time value

    % The data set needs to be translated DOWN so that the water level
    % becomes the horizontal axis; this needs to be done for every
    % timeIterCount water level iteration.

    % *******To be used with 'cross section' data:********
    % transWLnegqydata = negqydata - yWL(l);

    % *******To be used with '2017crosssection' data:******
    transWLqydata17 = qydata17 - yWL17(l);

    % Find the x-coordinates corresponding to the water level on
 either
    % side of the cross-section.
    % Initialize and set x and y variables to NaN
    x17 = [nan,nan];
    iterCount = 0;
    Int = 0;
    for k = 1:nqydata17
        % **** original cross section ***************
        % if isalmost(round(yWL(l),2), round(negqydata(k),2), 0.01)
 && ~isnan(WL(l)) % The isalmost function tests if water level value
 is approximately equal to the y-coordinate within the specified
 tolerance. Only perform the comparison if the cell is populated with
 numeric data.

        % **** 2017crosssection ******
        if isalmost(round(yWL17(l),2), round(qydata17(k),2), 0.01) &&
 ~isnan(WL17(l)) && WL17(l) < max(qydata17) && WL17(l) >= 0
            % The isalmost function tests if water level value
 is approximately equal to the y-coordinate within the specified
```

```
tolerance. Only perform the comparison if the cell is populated with
numeric data.
            % Some water levels are greater than the river bank depth,
these
            % values are excluded. See known issues in the preamble.
            % Some water levels are less than zero, these values are
excluded.

            % Note: Because rounding issues cause exact testing to
fail to
            % find all relevant values in the water level vector a
            % tolerance is added.
            iterCount = iterCount + 1;
            x17(iterCount,:) = [k, qxdata17(k)]; % Stores the k-value
when true and the corresponding x-coordinate for the water level as a
column vector

        end
   end

   if ~isnan(WL17(l)) && WL17(l) < max(qydata17) && WL17(l) >= 0 &&
numel(x17(:,2)) > 1
        % ~isnan(WL18(l))
        % Some water level cells don't contain data, so these cells
are excluded from the xsAreaWL calculation.
        % WL17(l) < max(qydata17)
        % Some water levels are greater than the river bank depth,
these
        % values are excluded. See known issues in the preamble.
        % WL17(l) >= 0
        % Some water levels are less than zero, these values are
excluded.
        % numel(x17(:,2)) > 1
        % In some instances the water level is too close to the river
depth and therefore only
        % one x-coordinate is given; at least two x-coordinates are
needed to
        % perform the integration using trapz, so any data point water
        % level depths that contain only one x-coordinate are exluded.

        % **** 2017crosssection ******
        yWL_x17 = transWLqydata17(x17(1,1):x17(end,1)); % Extracts the
y-coordinates at the water level corresponding to the x-coordinates.
        % Set integration domain values
        xdom17 = x17(1,2):res:x17(end,2); % Important to use the last
value (end) of the array, since multiple points may be in close
proximity to each other - only the first and last coordinates are
needed to determine the water level boundary on the x-axis.

        % Hold cross-sectional area as a function of time, perform
trapezoidal
        % integration
        xsAreaWL17(timeIterCount) = abs(trapz(xdom17,yWL_x17));
   end
```

```
end
toc
```

Elapsed time is 688.972339 seconds.

2018

Calculate Cross-Section Area

Calculate the depth at water level

Program structure to find the integration limits according to the water level

```
nqydata18 = length(qydata18); % Outputs the size of the y-coordinate
 vector
nWL18 = length(WL18); % Size of Water Level vector

% Initialize variables column vectors to zeros
yWL18 = zeros(size(nWL18,1),1); % Assigns zeros to the first column of
 size nWL array of the first column (dimension)
xsAreaWL18 = zeros(size(WL18,1),1); % Assigns zeros to the first
 column of size WL array of the first column
timeIterCount = 0;
tic % start stopwatch timer to measure the time the code execution
 time
WL18(14571) = nan; % This data point is excluded from cross-section
 area
% calc, the water level is too close to the river depth and therefore
 only
% one x-coordinate is given; at least two x-coordinates are needed to
% perform the integration using trapz.

for l = 1:nWL18
    timeIterCount = timeIterCount + 1;
    % *******To be used with 'cross section' data:********
    %yWL(l) = min(negqydata) + WL(l); % Calcs the y-depth at Water
 Level for each date/time value

    yWL18(l) = min(qydata18) + WL18(l); % Calcs the y-depth at Water
 Level for each date/time value

    % The data set needs to be translated DOWN so that the water level
    % becomes the horizontal axis; this needs to be done for every
    % timeIterCount water level iteration.

    transWLqydata18 = qydata18 - yWL18(l);

    % Find the x-coordinates corresponding to the water level on
 either
    % side of the cross-section.
    % Initialize and set x and y variables to NaN
    x18 = [nan,nan];
    iterCount = 0;
    Int = 0;
```

```
   for k = 1:nqydata18
       % **** original cross section ***************
       % if isalmost(round(yWL(l),2), round(negqydata(k),2), 0.01)
&& ~isnan(WL(l)) % The isalmost function tests if water level value
is approximately equal to the y-coordinate within the specified
tolerance. Only perform the comparison if the cell is populated with
numeric data.

       if isalmost(round(yWL18(l),2), round(qydata18(k),2), 0.01) &&
~isnan(WL18(l)) && WL18(l) < max(qydata18) && WL18(l) >= 0
           % The isalmost function tests if water level value
is approximately equal to the y-coordinate within the specified
tolerance. Only perform the comparison if the cell is populated with
numeric data.
           % Some water levels are greater than the river bank depth,
these
           % values are excluded. See known issues in the preamble.
           % Some water levels are less than zero, these values are
excluded.

           % Note: Because rounding issues cause exact testing to
fail to
           % find all relevant values in the water level vector a
           % tolerance is added.
           iterCount = iterCount + 1;
           x18(iterCount,:) = [k, qxdata18(k)]; % Stores the k-value
when true and the corresponding x-coordinate for the water level as a
column vector

       end
   end

   if ~isnan(WL18(l)) && WL18(l) < max(qydata18) && WL18(l) >= 0 &&
numel(x18(:,2)) > 1
       % ~isnan(WL18(l))
       % Some water level cells don't contain data, so these cells
are excluded from the xsAreaWL calculation.
       % WL18(l) < max(qydata18)
       % Some water levels are greater than the river bank depth,
these
       % values are excluded. See known issues in the preamble.
       % WL18(l) >= 0
       % Some water levels are less than zero, these values are
excluded.
       % numel(x18(:,2)) > 1
       % In some instances the water level is too close to the river
depth and therefore only
       % one x-coordinate is given; at least two x-coordinates are
needed to
       % perform the integration using trapz, so any data point water
       % level depths that contain only one x-coordinate are exluded.

       yWL_x18 = transWLqydata18(x18(1,1):x18(end,1)); % Extracts the
y-coordinates at the water level corresponding to the x-coordinates.
```

```
        % Set integration domain values
        xdom18 = x18(1,2):res:x18(end,2); % Important to use the last
 value (end) of the array, since multiple points may be in close
 proximity to each other - only the first and last coordinates are
 needed to determine the water level boundary on the x-axis.

        % Hold cross-sectional area as a function of time, perform
 trapezoidal
        % integration
        xsAreaWL18(timeIterCount) = abs(trapz(xdom18,yWL_x18));
    end
end
toc
```

Elapsed time is 726.134353 seconds.

2020

Calculate Cross-Section Area

Calculate the depth at water level

Program structure to find the integration limits according to the water level

```
nqydata20 = length(qydata20); % Outputs the size of the y-coordinate
 vector
nWL20 = length(WL20); % Size of Water Level vector

% Initialize variables column vectors to zeros
yWL20 = zeros(size(nWL20,1),1); % Assigns zeros to the first column of
 size nWL array of the first column (dimension)
xsAreaWL20 = zeros(size(WL20,1),1); % Assigns zeros to the first
 column of size WL array of the first column
timeIterCount = 0;
tic % start stopwatch timer to measure the time the code execution
 time
for l = 1:nWL20
    timeIterCount = timeIterCount + 1;
    % *******To be used with 'cross section' data:********
    %yWL(l) = min(negqydata) + WL(l); % Calcs the y-depth at Water
 Level for each date/time value

    yWL20(l) = min(qydata20) + WL20(l); % Calcs the y-depth at Water
 Level for each date/time value

    % The data set needs to be translated DOWN so that the water level
    % becomes the horizontal axis; this needs to be done for every
    % timeIterCount water level iteration.

    transWLqydata20 = qydata20 - yWL20(l);

    % Find the x-coordinates corresponding to the water level on
 either
```

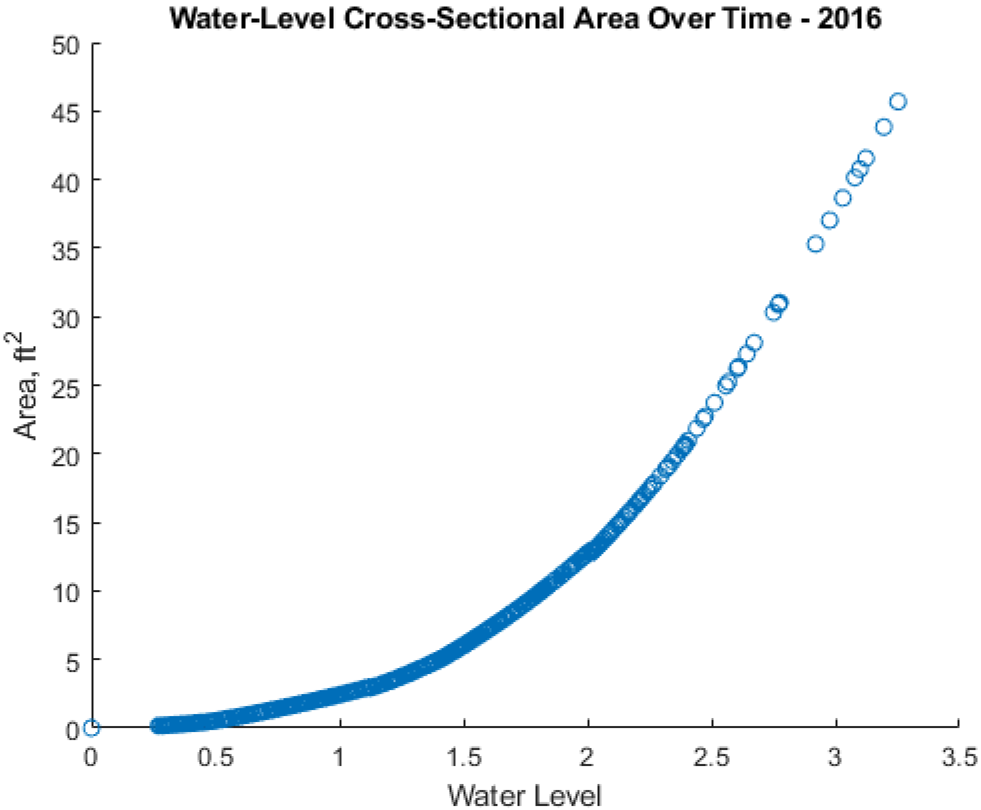
```
    % side of the cross-section.
    % Initialize and set x and y variables to NaN
    x20 = [nan,nan];
    iterCount = 0;
    Int = 0;
    for k = 1:nqydata20
        % **** original cross section ***************
        % if isalmost(round(yWL(l),2), round(negqydata(k),2), 0.01)
&& ~isnan(WL(l)) % The isalmost function tests if water level value
is approximately equal to the y-coordinate within the specified
tolerance. Only perform the comparison if the cell is populated with
numeric data.

        if isalmost(round(yWL20(l),2), round(qydata20(k),2), 0.01) &&
~isnan(WL20(l)) && WL20(l) < max(qydata20) && WL20(l) >= 0
            % The isalmost function tests if water level value
is approximately equal to the y-coordinate within the specified
tolerance. Only perform the comparison if the cell is populated with
numeric data.
            % Some water levels are greater than the river bank depth,
these
            % values are excluded. See known issues in the preamble.
            % Some water levels are less than zero, these values are
excluded.

            % Note: Because rounding issues cause exact testing to
fail to
            % find all relevant values in the water level vector a
            % tolerance is added.
            iterCount = iterCount + 1;
            x20(iterCount,:) = [k, qxdata20(k)]; % Stores the k-value
when true and the corresponding x-coordinate for the water level as a
column vector

        end
    end

    if ~isnan(WL20(l)) && WL20(l) < max(qydata20) && WL20(l) >= 0 &&
numel(x20(:,2)) > 1
        % ~isnan(WL18(l))
        % Some water level cells don't contain data, so these cells
are excluded from the xsAreaWL calculation.
        % WL20(l) < max(qydata20)
        % Some water levels are greater than the river bank depth,
these
        % values are excluded. See known issues in the preamble.
        % WL20(l) >= 0
        % Some water levels are less than zero, these values are
excluded.
        % numel(x20(:,2)) > 1
        % In some instances the water level is too close to the river
depth and therefore only
        % one x-coordinate is given; at least two x-coordinates are
needed to
```

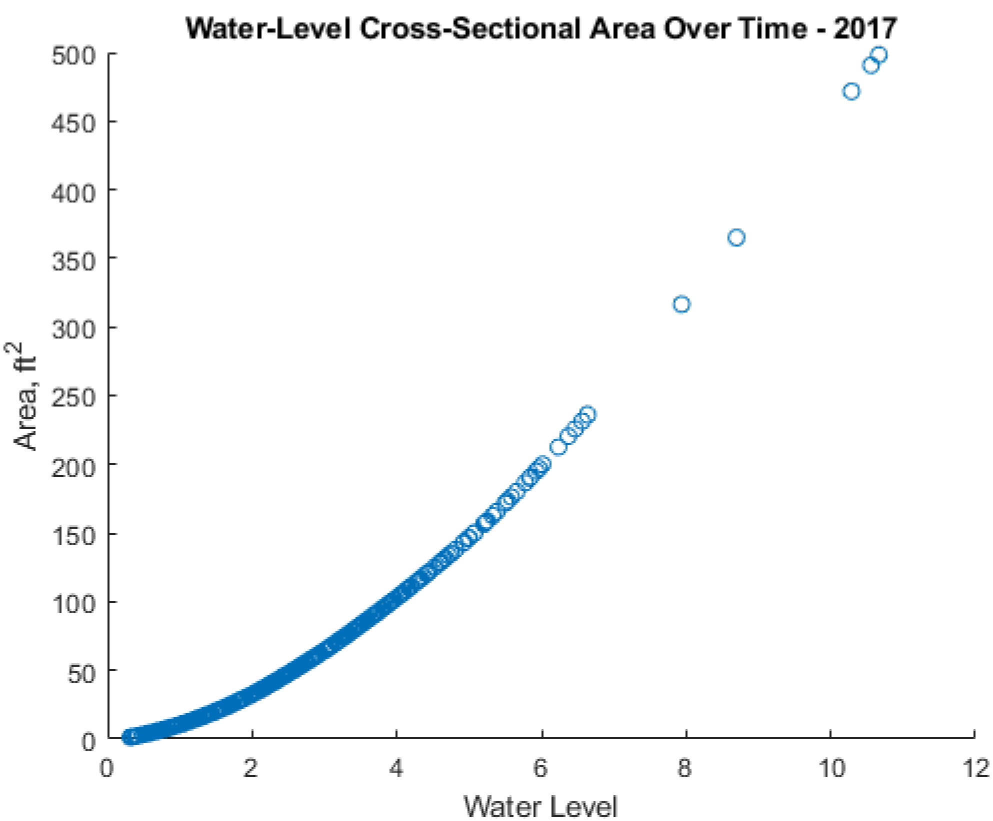
```
        % perform the integration using trapz, so any data point water
        % level depths that contain only one x-coordinate are exluded.

        yWL_x20 = transWLqydata20(x20(1,1):x20(end,1)); % Extracts the
 y-coordinates at the water level corresponding to the x-coordinates.
        % Set integration domain values
        xdom20 = x20(1,2):res:x20(end,2); % Important to use the last
 value (end) of the array, since multiple points may be in close
 proximity to each other - only the first and last coordinates are
 needed to determine the water level boundary on the x-axis.

        % Hold cross-sectional area as a function of time, perform
 trapezoidal
        % integration
        xsAreaWL20(timeIterCount) = abs(trapz(xdom20,yWL_x20));
    end
end
toc
```

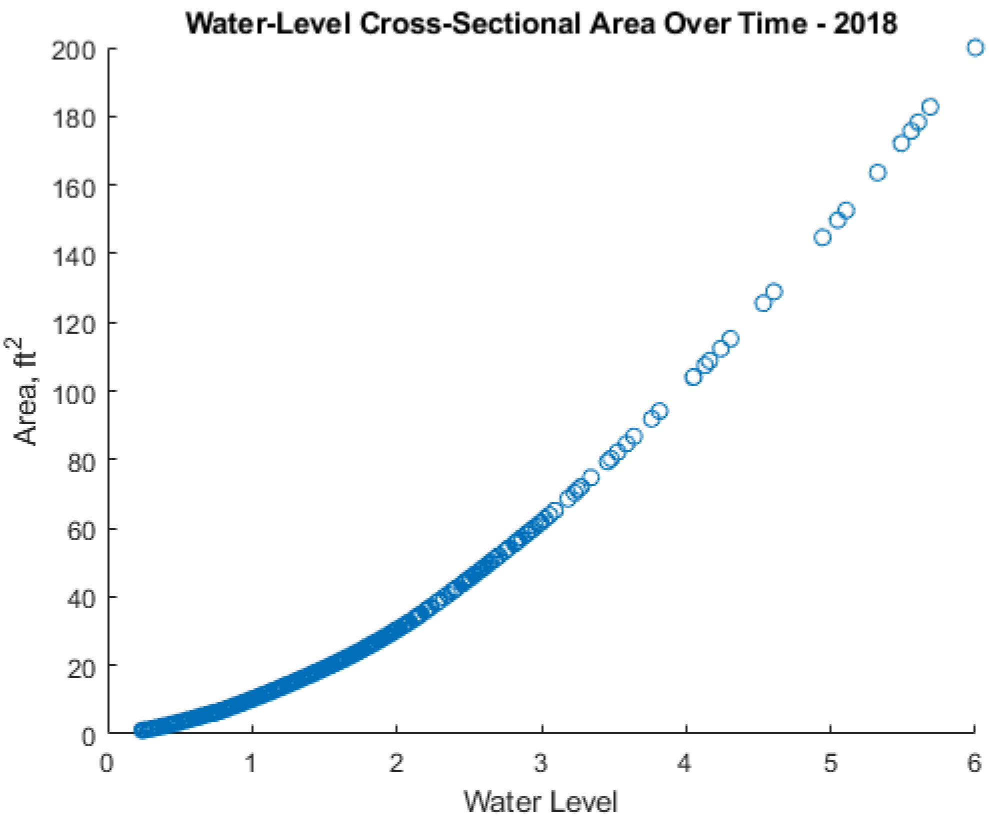
Elapsed time is 754.513148 seconds.

Other Plots

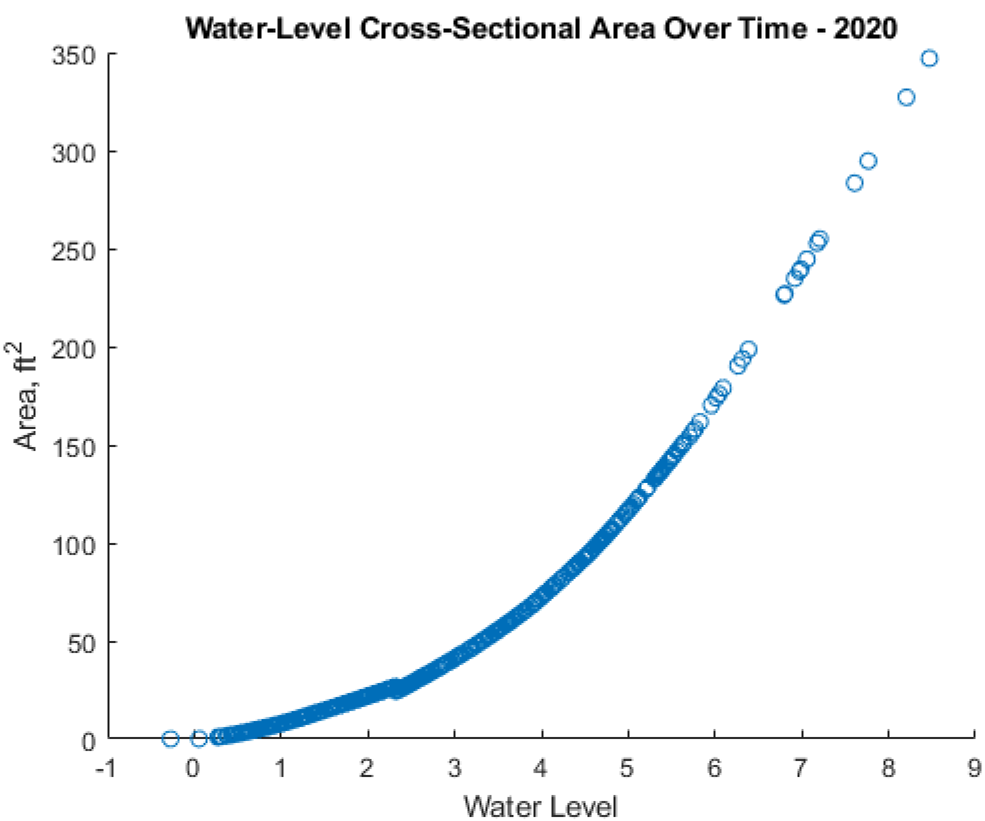
```
figure;
scatter(WL16,xsAreaWL16)
title('Water-Level Cross-Sectional Area Over Time - 2016')
xlabel('Water Level')
ylabel('Area, ft^{2}')

figure;
scatter(WL17,xsAreaWL17)
title('Water-Level Cross-Sectional Area Over Time - 2017')
xlabel('Water Level')
ylabel('Area, ft^{2}')

figure;
scatter(WL18,xsAreaWL18)
title('Water-Level Cross-Sectional Area Over Time - 2018')
xlabel('Water Level')
ylabel('Area, ft^{2}')

figure;
scatter(WL20,xsAreaWL20)
title('Water-Level Cross-Sectional Area Over Time - 2020')
xlabel('Water Level')
ylabel('Area, ft^{2}')
```


Water-Level Cross-Sectional Area Over Time - 2016
Area, ft²
Water Level
0
5
10
15
20
25
30
35
40
45
50
0
0.5
1
1.5
2
2.5
3
3.5


Water-Level Cross-Sectional Area Over Time - 2017
Area, ft²
Water Level
0
50
100
150
200
250
300
350
400
450
500
0
2
4
6
8
10
12



Water-Level Cross-Sectional Area Over Time - 2018
Area, ft^2
Water Level
0
20
40
60
80
100
120
140
160
180
200
0
1
2
3
4
5
6



Water-Level Cross-Sectional Area Over Time - 2020
Area, ft^2
Water Level
0
50
100
150
200
250
300
350
-1
0
1
2
3
4
5
6
7
8
9

Deliverables

1. date/time for the four data sets

```
dateTimeWL16;
dateTimeWL17;
dateTimeWL18;
dateTimeWL20;
```

2. depth or Water Level for the four data sets

```
WL16;
WL17;
WL18;
WL20;
```

3. cross-sectional area for each water-level depth.

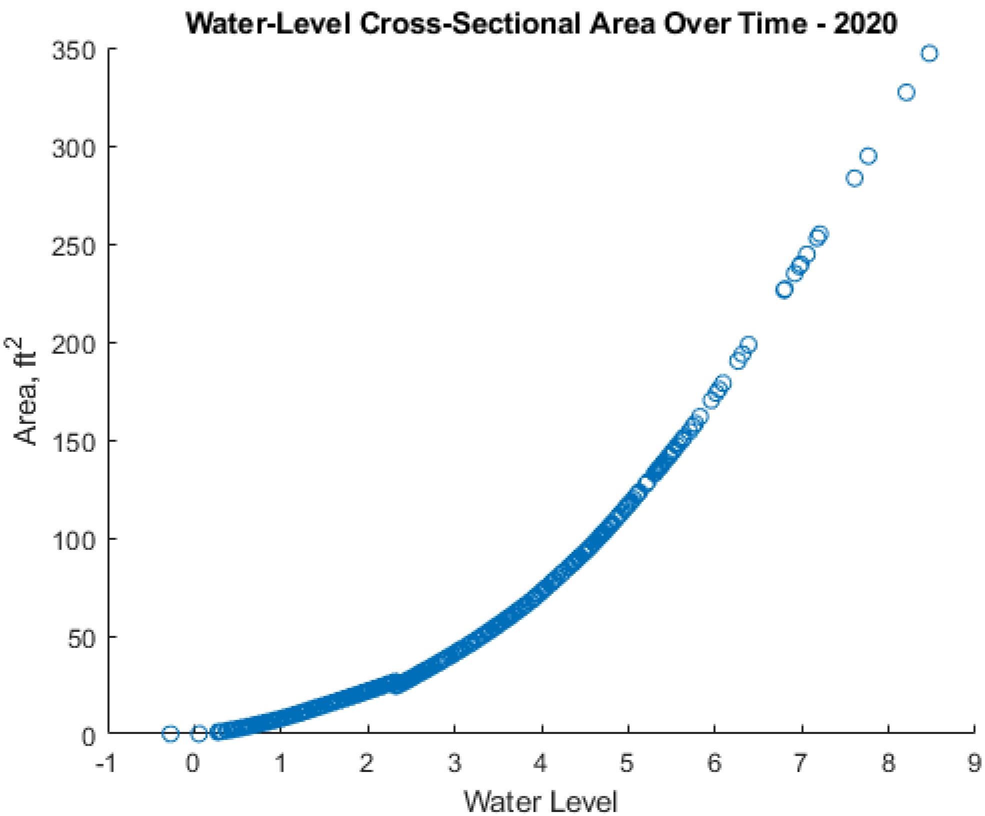
```
xsAreaWL16;
xsAreaWL17;
xsAreaWL18;
xsAreaWL20;
```

Write Data

Export data to MS Excel

```
% Write cross section data
tic
writematrix(xsAreaWL16,'xsectionflow.11.13_xsArea.xlsx','Sheet','WY16','Range','C2
writematrix(xsAreaWL17,'xsectionflow.11.13_xsArea.xlsx','Sheet','WY17','Range','C2
writematrix(xsAreaWL18,'xsectionflow.11.13_xsArea.xlsx','Sheet','WY18','Range','C2
writematrix(xsAreaWL20,'xsectionflow.11.13_xsArea.xlsx','Sheet','WY20','Range','C2
toc

% ***NOTE: Time to complete execution of this section of code is about
 55
% seconds.***

% Publish by entering ...
% 'publish('Cross_Sectional_Area_Analysis_v1_04.m','pdf')' in the
 command window
% END
```

Elapsed time is 16.928455 seconds.



Published with MATLAB® R2020a