Real Time and Historical Flow and Height Patterns of Lansing's Grand River using MATLAB Functions

December 14, 2010

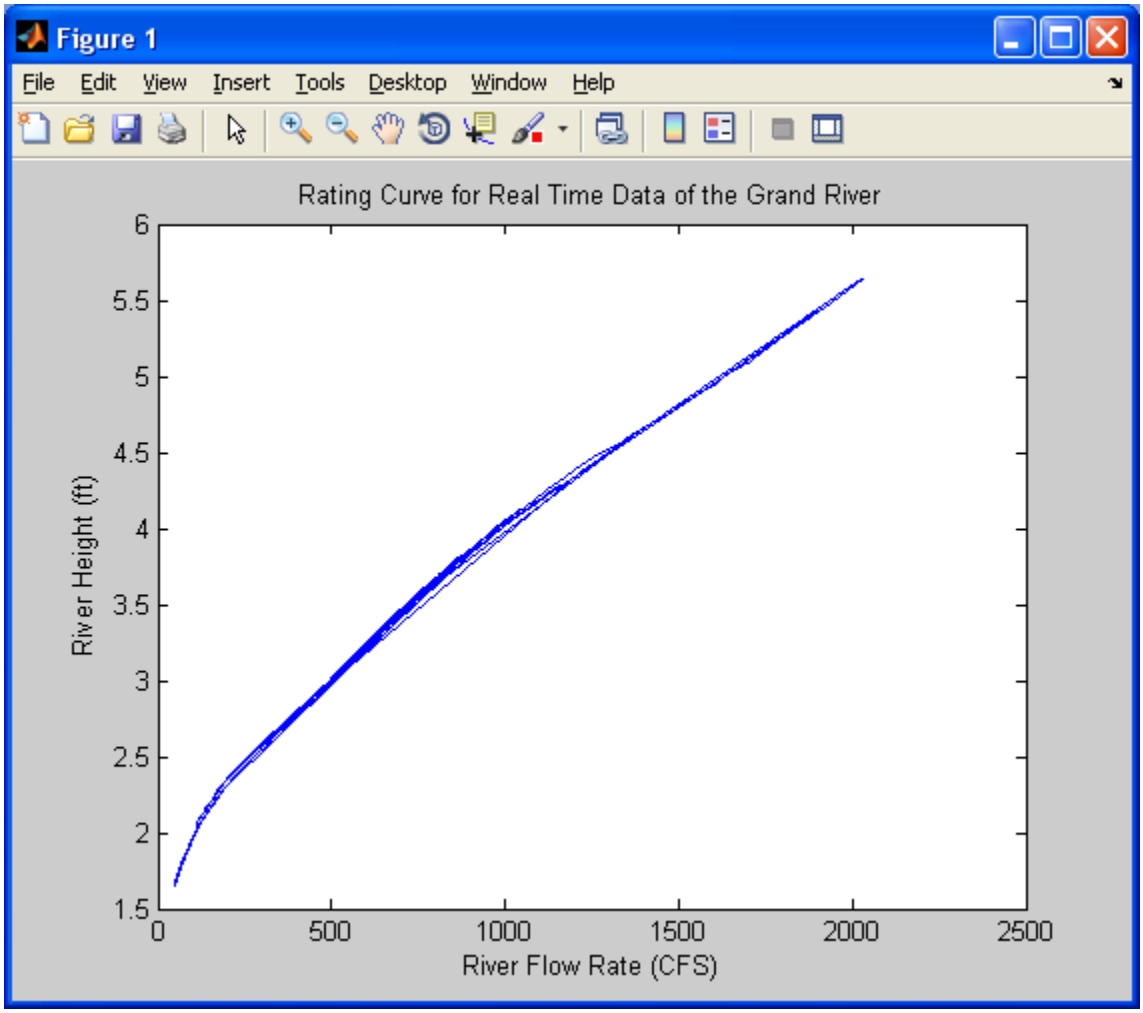
<student names removed>

INTRODUCTION

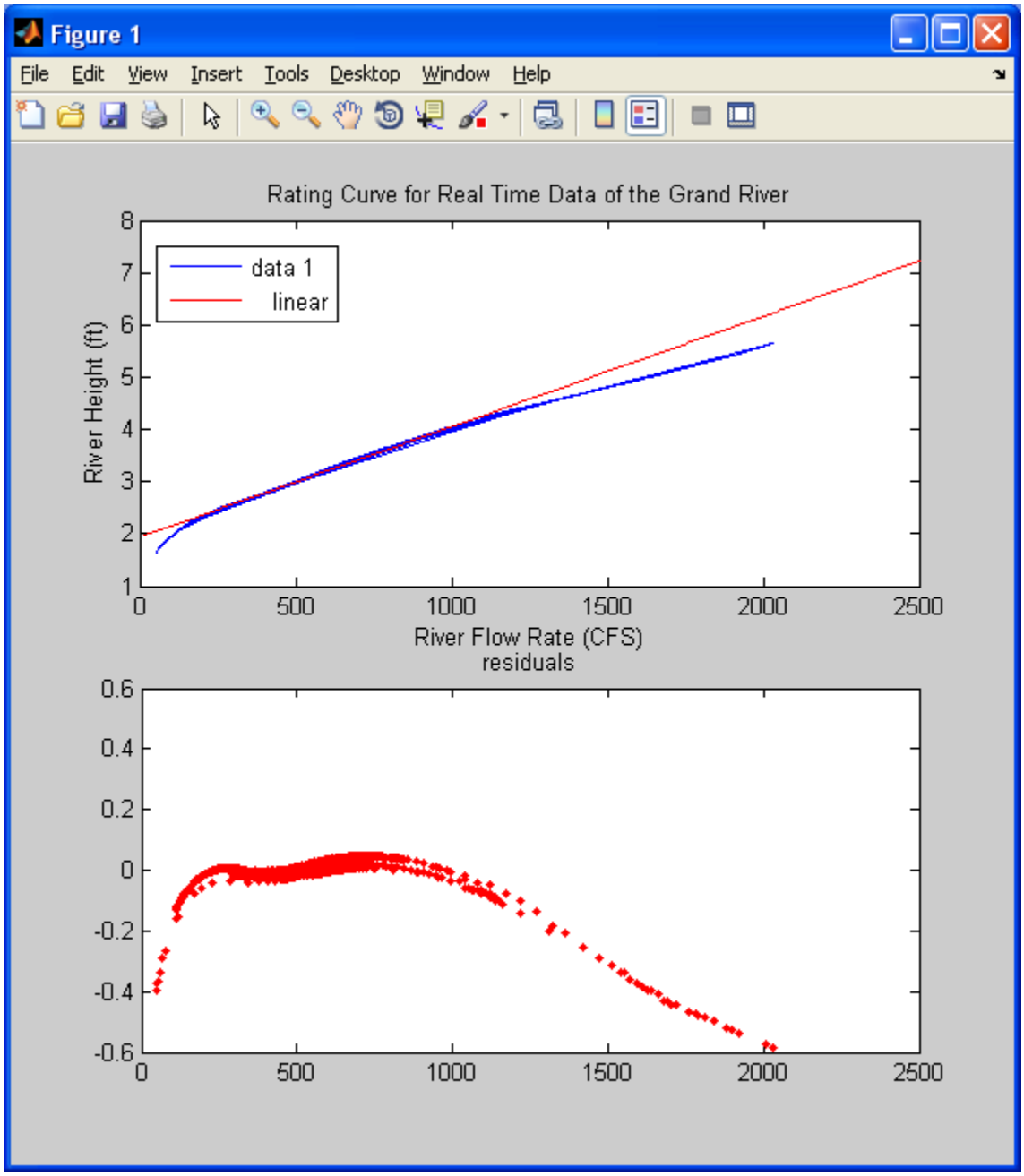
In the greater Lansing area, the Grand River is one of three rivers that could potentially cause flooding. Riverine flooding, when excess storm water overflows the river's banks, is of great concern for safety and financial reasons. For these reasons, the Grand River is continuously monitored. The United States Geological Survey maintains a website which provides historical river data. The website gives a variety of information concerning waterways in the United States. The river height (stage) and river flow (discharge) are two pieces of information that help predict the possibility of flooding. Using this information imported into MATLAB and through the use of MATLAB functions, we have provided graphs to show the flow and height patterns of the Grand River.

Part 1

Following each step to get the USGS information, for the Grand River, into a text format was not difficult as each step was clearly stated. Step 4 was helpful in showing how to get data into MATLAB. Using the *xlsread* function to import data from the Excel file we created showed just one more capability of MATLAB. From the data provided for us in step 4, we see the relationship between the first column and the second column: The number in the first column is cubed then multiplied by 2 to get the number in the second column. After downloading the information on the Grand River, from the USGS website, creating a rating curve using a MATLAB function we created was simple.



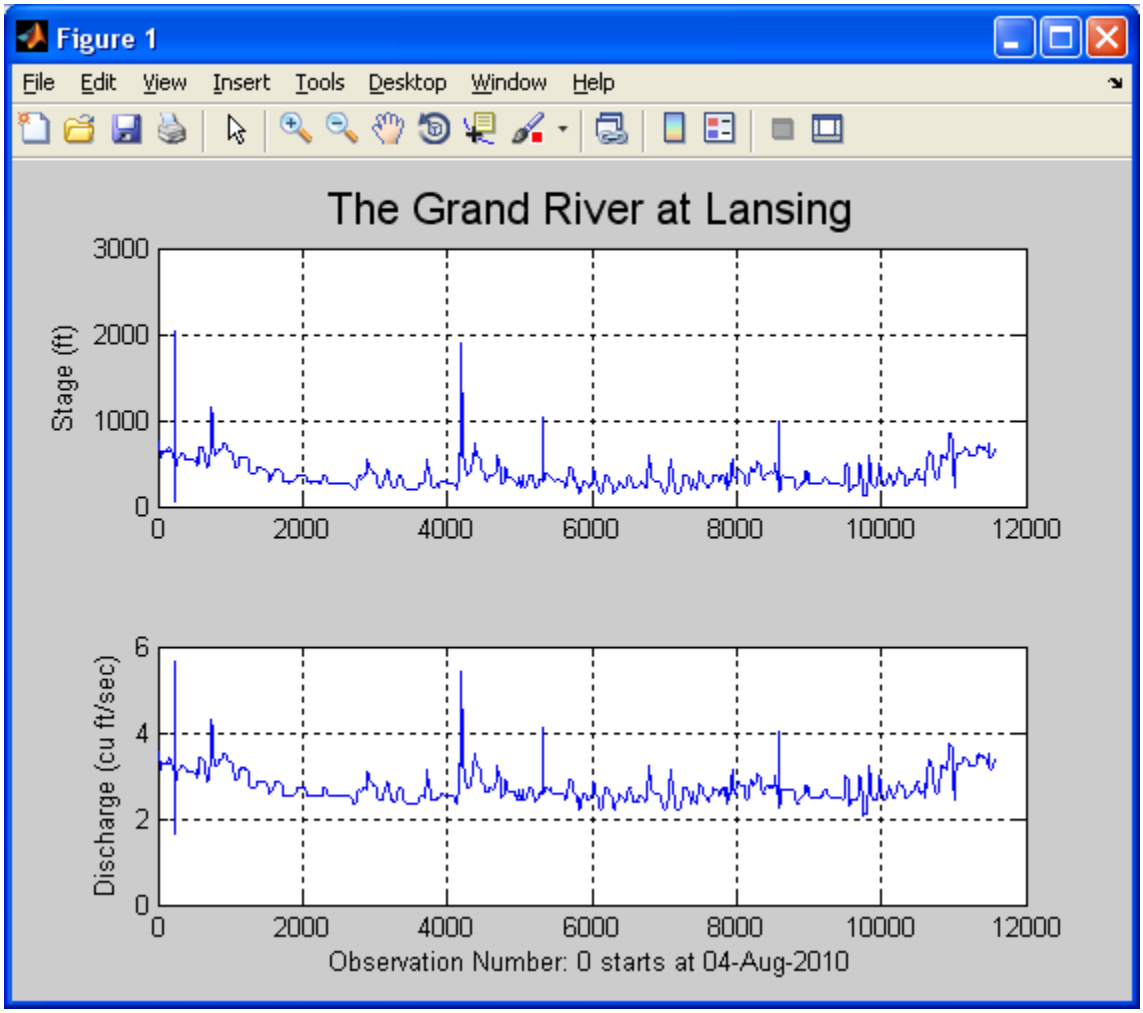
Using MATLAB graphing tools from the figure window allowed us to further explore the the data. In the basic fitting window we could display the "norm of residuals" as well as the polynomial coefficients that best fit the data. Considering all the data points we were given, the tools from the basic fitting window were very helpful.

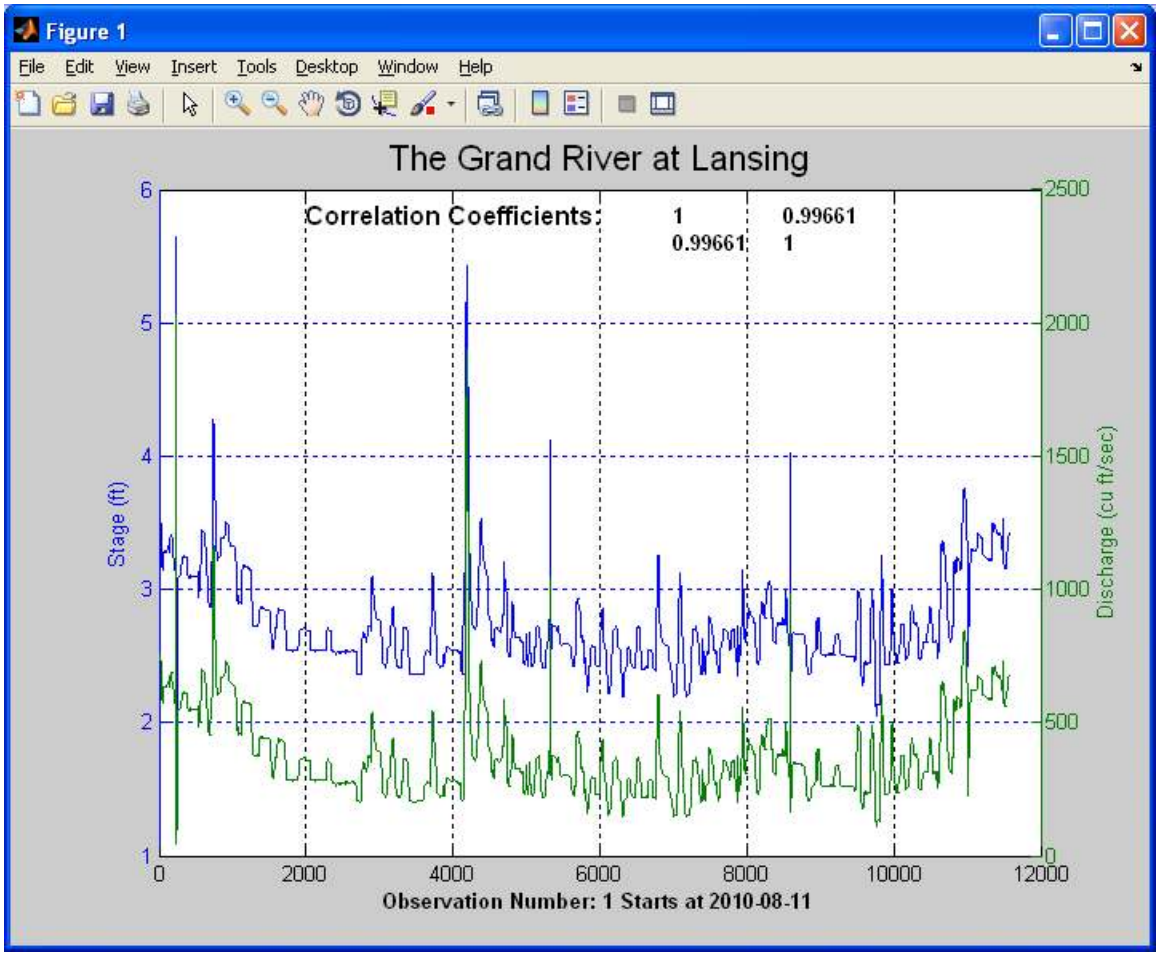


Understanding the limitations of a rating curve is important when interpreting data. Extraploation may not be the best method to use since it predicts the data values beyond the x data range. We could run into a problem if we were using extrapolation for high volume of flow because it assumes the data is continuous. It would not take into account the outliers of the data.

Part 2

To show the correlation between the river flow and river height, we used MATLAB to create subplots using the *subplot* function. This was not a too involved step since the programming was minimal and we were familiar with the functions. To further show the relationship between river flow and height, the *plotYY* function was implemented. This function allows for two independent data sets to be plotted in the same area using the left side as the Y-axis for one data set and the right side as the Y-axis for the other data set.



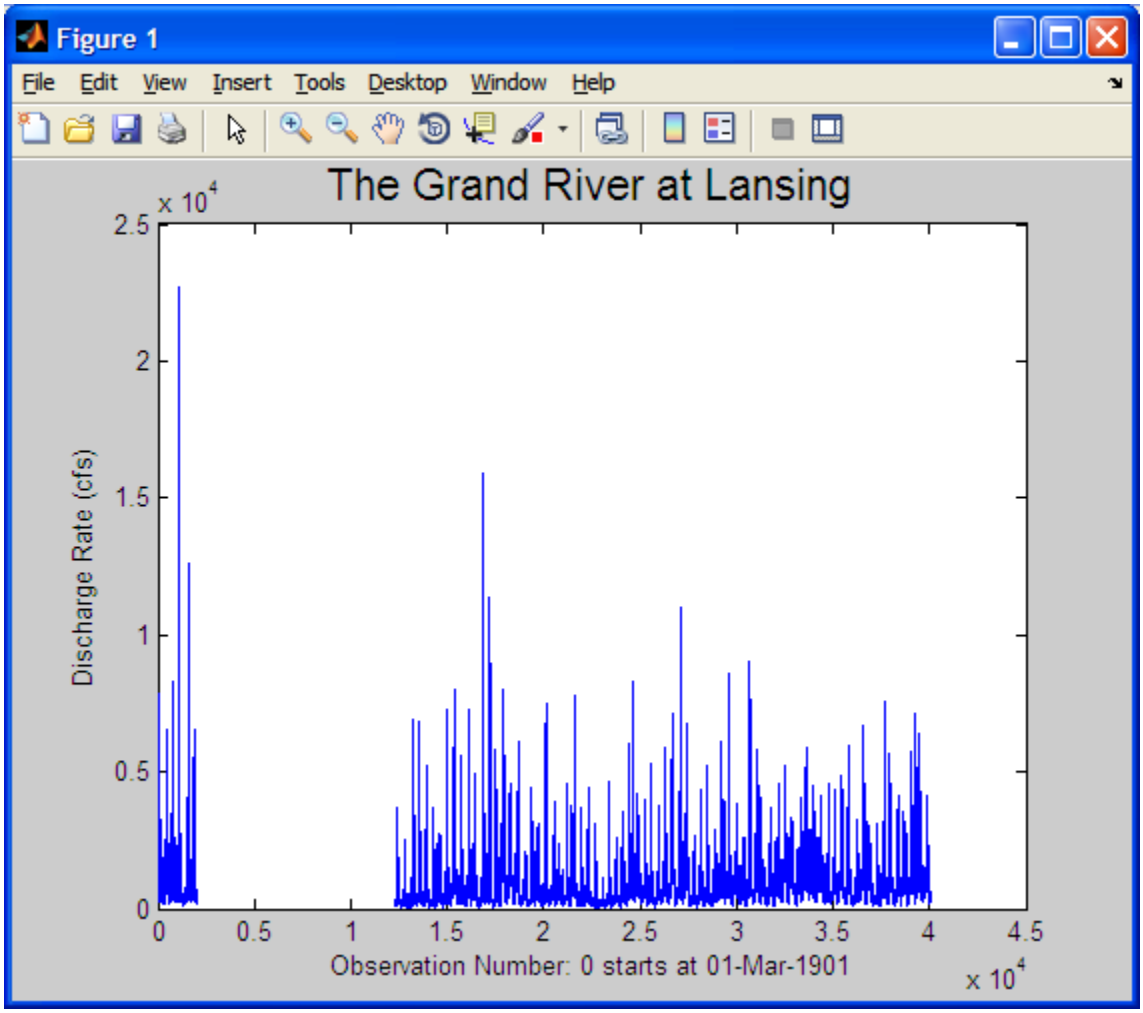


Looking at the data, we can see the higher the river level was, the faster the discharge was. This makes logical sense because the more water you have, the greater the pressure is, causing the river to flow faster. With the correlation coefficients so close to one, we see how strong the correlation between stage and discharge is. This further proves that when the stage increases so does the discharge rate. In the beginning of the plotted data we see that the level and the discharge had a low point. After utilizing outside resources we learned that on August 6, 2010 the North Lansing Dam was opened up to let debris pass. With that information, we can confidently say the water data for the Grand River, recorded by the USGS, was taken near the North Lansing Dam.

In order for us to explore the data on the plot, interactive tools in MATLAB were utilized. The *ginput* function was used to produces a pair of data points, which is translated as the date/time of selected point. This was one of the more involved steps we had to do, but given its functionality it's easy to see why. The *ginput* is one of the better interactive functions we have used. The use of this function allows for a quick check of a data point if we were trying to pull exact data from the graph.

```
M = readDataInFromTXT('MyRawProjectData.txt');
        GrandRiver2(M)
Get value for specific point? (1 for yes, 0 for no) : 1
Select a point on the plot
Output given in form (date, time, observation #, stage, discharge)
  '2010-11-01'    '10:45'    [8585]    [2.5800]    [298]
Another point? (1 for yes, 0 for no) :
```

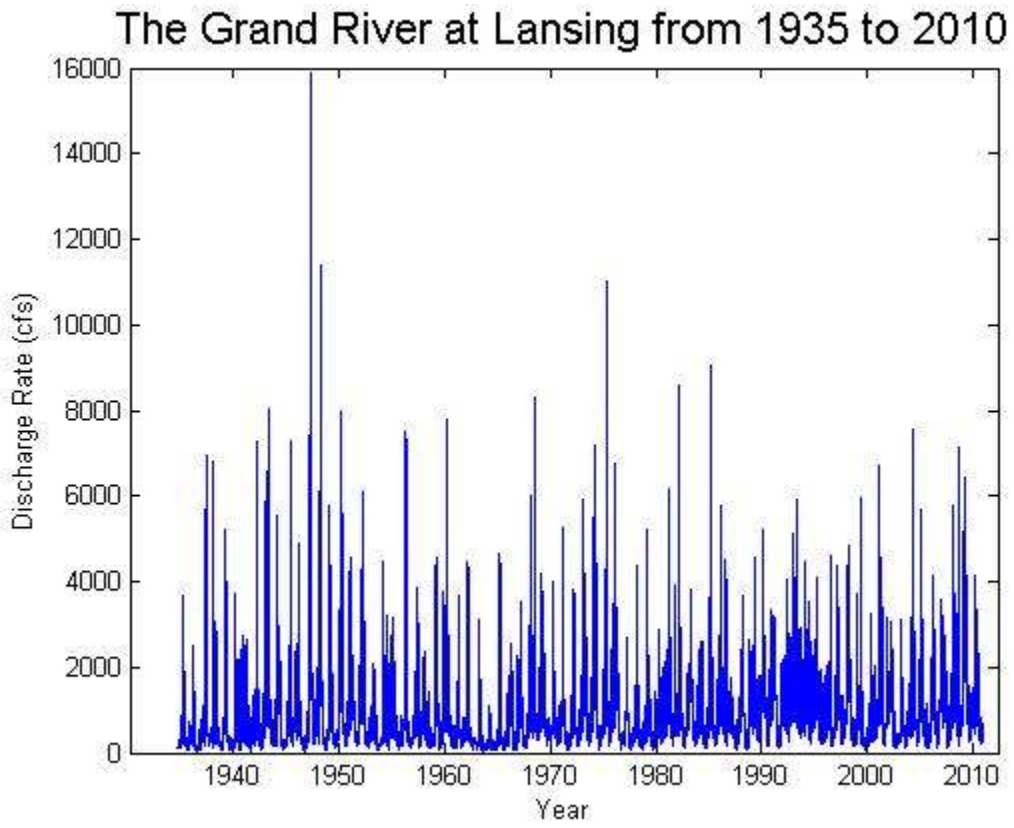
Exploring the historical data the USGS website provides, we can investigate the patterns of the river flow and height dating back to 1901. Using MATLAB to produce a plot, we can see the peak where the flood of 1904 occurred and that there is period where no data is provided. Looking at the table of data points, there is no data from 1906 to 1934. There is not an explanation of why no data was recorded, but a simple explanation could be the information was lost. Some other explanations could be there was no one to record the data or the city was in a financial downturn and money could not be allocated to pay someone to record the data.



With such a big gap in the plot because of the missing data, it makes sense to look at the data *after* the gap. Using the *isNaN*, a built in function from MATLAB, producing a plot of data points after the gap was fairly simple. We could have used just *NaN* if our information was not in the form of an array.

The initial graph of the data points after the gap provided us with a pattern of the Grand River, but did not include a date on the x-axis. To do this, the built in MATLAB function *datetick* was used. In MATLAB there are numerous ways to represent the date. The data

we were provided with gave us the date in the form of yyyy-mm-dd. MATLAB provides us with a list of date forms, for our data points we initially used #29 in the date format. The only problem with using the built in form was the graph continued on after 2010, resulting in a gap in the graph. To fix this problem the *keeplimits* function was used to eliminate the gap after 2010.



To find the wettest period over the last hundred years in Lansing we had to do figure out exactly what length a period would be. To determine the length of a period, we took into account that there were over 40,000 data points to investigate. We needed to make sure our period wasn't too short or too long. We decided that creating a function where the user could input the length of the period would be more useful since we all can argue why *our* length of period is the right length. For the lengths given it finds the highest

average and of those values returns the one closest to the max for the inputted time periods

Several MATLAB functions were used to complete the code, including: *datenum ,isnan, max, length, maxfinder, zeros, datestr, mean, avg*.

```
>> daysNum = [2 10 50 100 500 1000 2000 5000];
>> [PeriodStart,BestLengthInDays] = maxFinder(daysNum)
PeriodStart =
24-Feb-1985
BestLengthInDays =
   50
```

Conclusion

This project was by far the best example of using MATLAB to solve real world problems. Although the functions we created throughout the semester helped us see how MATLAB could help us with our homework, seeing crucial information, like the height of a river, taught us that this is not just software to help us complete homework. Steps 2 and 7 were by far the most difficult to complete probably because it involved some of the MATLAB functions we hadn't yet utilized. These steps helped us appreciate how thorough engineers/programmers must be when tackling real world problems like this. The project showed us how we could conclude what was happening with the river given either the stage or discharge. Looking at the historic data maintained by USGS helped us predict what might happen with the Grand River in the future.

APPENDIX

Part 1

```
function M = readDataInFromTXT(MyRawProjectData)
% reads data from a tab delimited TXT
% file into a MATLAB
% INPUT: fileName - the name of the TXT file to be read in
% OUTPUT: M is a cell array with seven elements, one for
% each column of data in fileName.
fid = fopen('MyRawProjectData.txt'); % open the file for reading
theFormat = '%s %s %s %s %s %f %f';
M = textscan(fid, theFormat);
fclose(fid); % close the file
x=M{:,7};
y=M{:,6};
plot(x,y)
title('Rating Curve for Real Time Data of the Grand River')
xlabel('River Flow Rate (CFS)')
ylabel('River Height (ft)')
```

```
>> M = readDataInFromTXT('MyRawProjectData.txt')

M =

 Columns 1 through 4

   {11577x1 cell}   {11577x1 cell}   {11577x1 cell}   {11577x1 cell}

 Columns 5 through 7

   {11577x1 cell}   [11577x1 double]   [11577x1 double]
```

Part 2

Step 1

```
function GrandRivers1p2(M)
%This function plots the height of the river in feet over the
obsevation time in
%subplot 1 and the discharge rate of the river in cu ft/sec over the
%observation time in subplot2

%Inputs: M, the data collected by the USGS, string
x=M{7};
y=M{6};
subplot(2,1,1)
```

```
plot(x)
grid on
ylabel('Stage (ft)')
title('The Grand River at Lansing','fontsize',16)
subplot(2,1,2)
plot(y)
grid on
ylabel('Discharge (cu ft/sec)')
xlabel('Observation Number: 0 starts at 04-Aug-2010')


function GrandRivers1p2yy(M)
%This function plots the height of the river in feet over the
obsevation time in
%subplot 1 and the discharge rate of the river in cu ft/sec over the
%observation time in subplot2

%Inputs: M, the data collected by the USGS, string
y1=M{6};
y2=M{7};
x1=[1:1:length(y1)];
YAxes = plotyy(x1,y1,x1,y2);
title('The Grand River at Lansing','fontsize',16)
xlabel('Observation Number: 1 Starts at 2010-08-
11','FontWeight','bold')
grid on
set(get(YAxes(1),'Ylabel'),'String','Stage (ft)')
set(get(YAxes(2),'Ylabel'),'String','Discharge (cu ft/sec)')
Coeffs = corrcoef(y1, y2);
A = Coeffs(1);
B = Coeffs(2);
C = Coeffs(3);
D = Coeffs(4);
text(2000,5.8,'Correlation Coefficients:', 'FontWeight', 'bold',
'FontSize', 12)
text(7000,5.8,num2str(A), 'FontWeight', 'bold')
text(8500,5.8,num2str(B), 'FontWeight', 'bold')
text(7000,5.6,num2str(C), 'FontWeight', 'bold')
text(8500,5.6,num2str(D), 'FontWeight', 'bold')
```

Step 2

```
function GrandRiver2(M)
% This function plots data set M and will give the date, time,
observation #, stage, and discharge for points
% that the user clicks on the plot.
% INPUTS: M, data set with 7 columns where column 3 is the date, column
4 is the time, column 6 is the 'stage' and
%            column 7 is the 'discharge' amount, array
% OUTPUTS: inqPt, date, time, observation #, stage, and discharge,
vector
%
% Sample Call: M = readDataInFromTXT('MyRawProjectData.txt');
%                  GrandRiver2(M)
```

```
x1 = 1:length(M{7});
y1 = M{6};
y2 = M{7};
date = M{3};
time = M{4};
YAxes = plotyy(x1,y1,x1,y2);
xlabel('Observation Number: 1 Starts at 2010-08-
11','FontWeight','bold')
set(get(YAxes(1),'Ylabel'),'String','Stage (ft)')
set(get(YAxes(2),'Ylabel'),'String','Discharge (cu ft/sec)')

k = input('Get value for specific point? (1 for yes, 0 for no) : ');
if k == 1
    keepGoing = 1;
    while keepGoing == 1
       disp('Select a point on the plot')
       [xIn yIn] = ginput(1);
       inqPt = [date(round(xIn)) time(round(xIn)) round(xIn)
y1(round(xIn)) y2(round(xIn))];
       disp('Output given in form (date, time, observation #, stage,
discharge)')
       disp(inqPt)
       keepGoing = input('Another point? (1 for yes, 0 for no) : ');
    end
end
```

Step 3

how to obtain historical data

```
function M2 = readDataInFromTXT2(fileName)
%function reads data from a tab delimited TXT
%file into a MATLAB Cell array
%INPUT: fileName- the name of the TXT file to be read in
%OUTPUT: M2 is a cell array with 5 elements, 4 strings, 1 double

fid = fopen(fileName); %open file
    readFormat = '%s %s %s %f %s';
    M2 = textscan(fid,readFormat); %read data in format given

fclose(fid); %close file

x = M2{4};

figure;
plot(x);
title('Discharge of Grand River');
xlabel('Discharge (cu ft/second)');

function GrandRiverHistoricNew(M2)
```

```
%This function plots the discharge rate vs time for the Historic USGS
data
%from the Grand River at Lansing for the data after the gap in data.

%Inputs: M2, the data collected by the USGS, string
x=M2{4};
rangenan=find(isnan(x));
xstart=max(rangenan);
NewData=x(xstart+1:end);
plot(NewData)
xlabel('Day')
ylabel('Discharge Rate (cfs)')
title('The Grand River at Lansing Historic','fontsize',16)
```

Step 4

```
function GrandRiverHistoric(M2)
%This function plots a scatter plot of the height of the water vs. the
%discharge rate for the USGS data from the Grand River at Lansing.

%Inputs: MH, the data collected by the USGS, string
x=M2{4};
plot(x)
xlabel('Observation Number: 0 starts at 01-Mar-1901')
ylabel('Discharge Rate (cfs)')
title('The Grand River at Lansing','fontsize',16)
```

Step 5

```
function GrandRiverHistoricNew(M2)
%This function plots the discharge rate vs time for the Historic USGS
data
%from the Grand River at Lansing for the data after the gap in data.

%Inputs: M, the data collected by the USGS, string
x=M2{4};
rangenan=find(isnan(x));
xstart=max(rangenan);
NewData=x(xstart+1:end);
plot(NewData)
xlabel('Day')
ylabel('Discharge Rate (cfs)')
title('The Grand River at Lansing Historic','fontsize',16)
dates=M{3};
```

Step 6

With the Years

```
function ModifiedDataPlot(M2)
```

```
%This function plots the discharge rate vs time for the Historic USGS
data
%from the Grand River at Lansing for the data after the gap in data.

%Inputs: M2, the data collected by the USGS, string
y= M2{4};
x= datenum(M2{3});
rangenan= find(isnan(y));
start= max(rangenan);
NewData= y(start+1:end);
xrange = x(start+1:end);


plot(xrange,NewData)
xlabel('Year')
ylabel('Discharge Rate (cfs)')
title('The Grand River at Lansing from 1935 to 2010','fontsize',16)
set(gca,'XTick',xrange)

datetick('x','yyyy','keeplimits')
```

Step 7

Now outputs best period length and starting point

```
function [PeriodStart,BestLength] = maxFinder(daysNum)
% This function gives starting date and number of days of wettest time
% period of the past hundred years.  The user provides a vector of
possible
% time lengths in days and the function will choose between them. The
% function will also work with a single value for daysNum.
% INPUTS: daysNum, lengths of time in days, vector
% OUTPUTS: PeriodStart, the starting date of the period, string
%
% Sample Call: daysNum = [2 10 100 500 1000];
%              [PeriodStart,BestLengthInDays] = maxFinder(daysNum)
%


%
M2 = readDataInFromTXT2('MyRawProjectData2.txt');
y= M2{4};
x= datenum(M2{3});
rangenan= find(isnan(y));
start= max(rangenan);

NewData= y(start+1:end);
NewDates = x(start+1:end);
startVal = zeros(length(daysNum),1);
StartDates = zeros(length(daysNum),1);
for i = 1:length(daysNum)

    avgW = zeros(length(NewData),1);
```

```
    for j = 1:length(NewData)-daysNum(i)+1
        Avg = NewData(j:j+daysNum(i)-1);
        avgW(j) = mean(Avg);
    end
    maxAvg = max(avgW);
    StartDates(i) = NewDates(logical(maxAvg==avgW));
    startVal(i) = maxAvg;

end
idealVal = mean(startVal);
valDiffs = abs(startVal-idealVal);
minVal = min(valDiffs);

for k = 1:length(valDiffs)
    if minVal == valDiffs(k)
        bestVal = k;
    end
end


bestStartVal = startVal(bestVal);

PeriodStart = datestr(StartDates(logical(bestStartVal==startVal)));

BestLength = daysNum(logical(bestStartVal==startVal));
```