

# *Investigating the Use of Hydroelectric Power*

New Mexico Supercomputing Challenge

Final Report

April 4, 2012

Team #77

Los Alamos Middle School

Team Member:

Claire DeCroix

Teacher:

Pauline Stephens

Project Mentor:

David DeCroix

## Executive Summary:

In this project a way to power a house, city, etc. without using fossil fuels was found. On Earth there are many rivers; for example the Rio Grande. These rivers are constantly flowing, some can create enough power to power a city and others can only power a cell phone. A way has been found to use the stream flow and turn it into electricity.

## Introduction:

Water wheels have been used for grinding grain, sawing wood, and powering textile mills since ancient Egypt. The Greeks and Romans knew about water wheels in the 3rd Century, but they were not used until the 14<sup>th</sup>. By the end of the 18<sup>th</sup> Century approximately 10,000 water wheels were used in New England. The problem was that the wheel's power output was limited to 100kW. Benoit Fourneyron in 1833 developed the first modern hydroelectric turbine, and in 1901 George Westinghouse installed the current equipment at Niagara Falls, expanding the uses of hydropower.

Lester Allan Pelton invented the Pelton wheel and patented it in October of 1880. The Pelton Wheel extracts energy from the impulse (the momentum) of moving water.

Pelton's design worked as follows: the water enters through a tube at the front of the wheel then is pushed out of two jets. One jet is shorter and higher up on the wheel and the second is lower shooting water toward one of the bottom cups. This turns the wheel, which is connected to an electrical generator. The energy is extracted from the water and turned into electrical power.

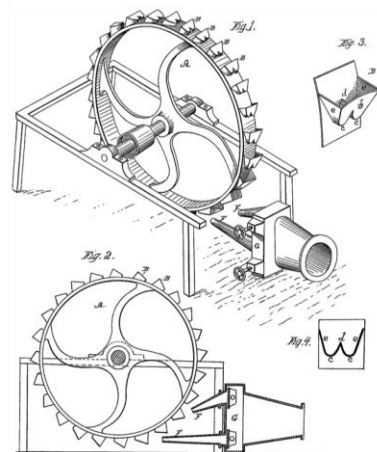


Figure 1 Lester Pelton's drawings from the original patent application.

Pelton designed the cups on the wheel in a very efficient but odd way. The cups are called double cups. They look like two glasses of water set next to each other, or an "M", shown in Figure 2. The water hits the middle ridge then splits on either side of the ridge and pushes the cup backward.

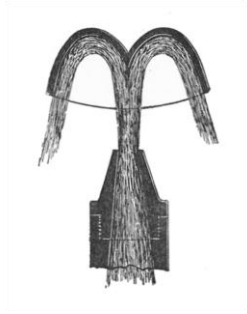


Figure 2 Pelton's cup design.

### Problem:

It was asked if there is a way to use stream water, or running water, to produce power. A simulated hydroelectric power generator was placed in a stream to extract the energy from the running water then convert it to power.

### Problem Solution:

The Mathworks' Matlab program was used to create a model of the physical system for this project. A power system to place in a stream or river to generate electricity was simulated. The power system has blades that turn a shaft and a shaft that turns an electrical generator. In Figure 3, a schematic diagram of the Pelton wheel is shown. The water flows through a nozzle and impinges on the bucket attached to the wheel. The wheel rotates a shaft, which can be connected to an electrical power generator.

The maximum performance of the system was looked at. The stream flow in a river is used to generate the power, the velocity in particular, not the traditional hydropower plant using the depth of water or "head". The change in velocity was looked at. "As you increase the velocity how much more power could you get out?" As speed increases, the propeller turns faster, and you produce more power from the water. The water temperature wasn't one of the concerns because it shouldn't have an impact on the performance of the system.

The mechanical conversion of water velocity into turning a shaft and the shaft turning/running a generator was also looked into. The generator is a black box that has directly effected by parameters, such as flow speed and blade efficiency. These relate directly to the power output of the generator. The most efficient generator will be determined by understanding these differences on power output.

The force of water impinging on a Pelton wheel blade can be expressed as:

$$F = 2\rho Q(V - u)$$

Where:

$V$ = Jet Velocity

$u$  = Bucket Velocity  
 $Q$  = Flow Rate  
 $\rho$  = Water Density

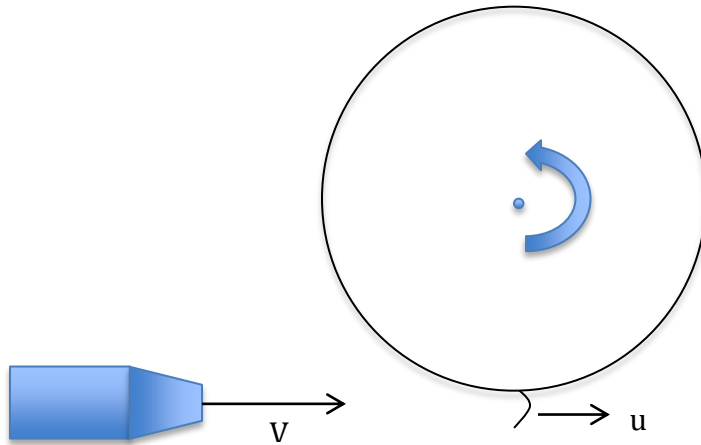


Figure 3 Pelton wheel schematic.

With that equation it is possible to calculate the Power with “Power = Force \* Velocity”, which simplifies to this equation:

$$P = (2\rho Q(V - u)) * (u)$$

For maximum power, it can be shown that  $V=2u$ , thus the power equation can be simplified to:

$$P = \frac{\rho Q V^2}{2}$$

A graphical user interface (GUI) was created using GUIDE, Matlab’s Graphical User Interface Development Environment, to calculate the maximum power in the above equation, shown in Figure 4. The GUI has editable text and stationary text. It is possible to enter the minimum flow rate, the maximum flow rate, the flow rate increment, and the jet velocity, for which it plots the calculated power as a function of flow rate. The GUI calculates the power output then plots the data into a graph.

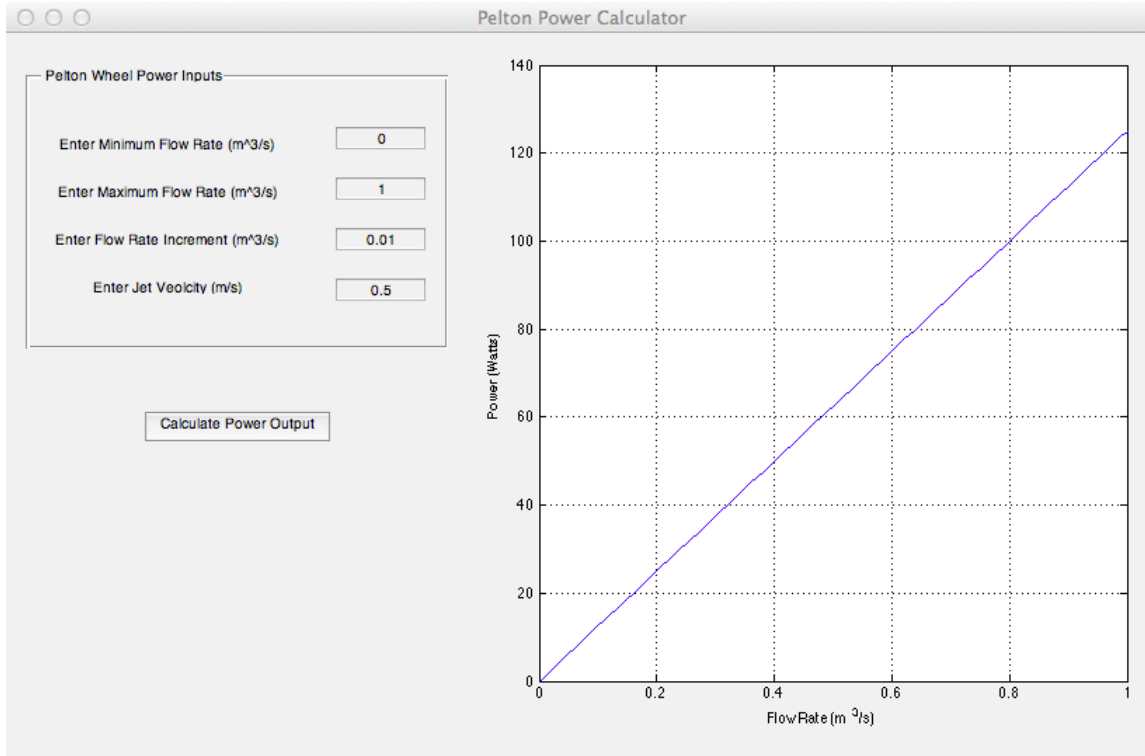


Figure 4 The GUI layout.

But in order for the GUI to work a conversion from text (string) to a number had to occur. There are four major element in the GUI, text boxes, which are descriptions of the variable, editable text boxes, which are the numbers for flow rate and jet velocity, a push button, that takes the numbers and calls the power calculation routine, and a graphical display area, which plots the power as a function of the input flow rates and jet velocity. The editable text box contains strings, or characters, which need to be converted into numbers using the “STR2double” function.

**Results:**

As shown in Figure 5 there is a different colored line for each different jet velocity; blue for a jet velocity of 1, green for a jet velocity of 1.5, and red for a jet velocity of 2. The flow rate was varied from zero to one m<sup>3</sup>/s (cubic meters per second). A cell phone charger needs two watts of power, where as a (my Grandmother’s) hair dryer needs 1800 watts of power. So, as shown in the graph, a cell phone charger can use a jet velocity of one and still get plenty of power without needing a stream like the Rio Grande, whose flow rate is about 42 m<sup>3</sup>/s. But a hairdryer needs a jet velocity of two and the water has to be flowing at least 0.9 m<sup>3</sup>/s.

Rivers and their flow rates:

River:	Rio Grande, NM	Boulder Creek, CO	Jemez River, NM
Flow Rate:	42 m <sup>3</sup> /s	0.85 m <sup>3</sup> /s	4.8 m <sup>3</sup> /s

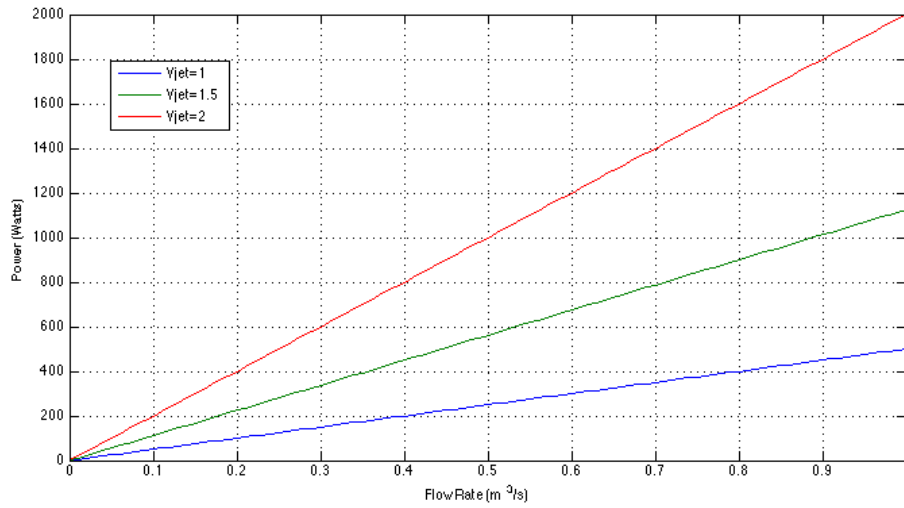


Figure 5 Pelton wheel power curves.

### Conclusions:

It is possible to power a cell phone or an appliance using a stream or a river. The next step would be to investigate the practicality of the theoretical simulation and propose designs for different power requirements. One could actually design a portable Pelton wheel for backpackers or military persons using a small stream flow, or if a larger stream or river were available, one could design a system to generate power for a household.

**References:**

1. Gulliver, John S. and Roger E. Arndt, "History and Technology of Hydropower," Encyclopedia of Energy, Vol. 3, 2004.
2. Haidar, Ahmed M.A. et.al., "Utilization of pico hydro generation in domestic and commercial loads," Renewable and Sustainable Energy Reviews, Vol 16, pp. 518-524, 2012.
3. Agar, D. and M. Rasi, "On the use of a laboratory scale Pelton wheel water-turbine in renewable energy education," Renewable Energy, Vol 33, pp. 1517-1522, 2008.
4. Pelton Wheel, Wikipedia.
5. Colorado Division of Water Resources website (<http://www.dwr.state.co.us>)

**Acknowledgements:**

Thank you to Mrs. Pauline Stephens, the sponsor, David DeCroix, my mentor. And thank you to the judges at the interim presentation and Nico Marrero for their input.

## Appendix: The Computer Program



```

function varargout = PeltonGUI(varargin)
% PELTONGUI MATLAB code for PeltonGUI.fig
% PELTONGUI, by itself, creates a new PELTONGUI or raises the
existing
% singleton*.
%
% H = PELTONGUI returns the handle to a new PELTONGUI or the
handle to
% the existing singleton*.
%
% PELTONGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
% function named CALLBACK in PELTONGUI.M with the given input
arguments.
%
% PELTONGUI('Property','Value',...) creates a new PELTONGUI or
raises the
% existing singleton*. Starting from the left, property value
pairs are
% applied to the GUI before PeltonGUI_OpeningFcn gets called.
An
% unrecognized property name or invalid value makes property
application
% stop. All inputs are passed to PeltonGUI_OpeningFcn via
varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help PeltonGUI

% Last Modified by GUIDE v2.5 18-Mar-2012 15:19:04

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @PeltonGUI_OpeningFcn, ...
                  'gui_OutputFcn', @PeltonGUI_OutputFcn, ...
                  'gui_LayoutFcn', [] , ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before PeltonGUI is made visible.
function PeltonGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)
% varargin    command line arguments to PeltonGUI (see VARARGIN)

% Enter out input data here
handles.count = 0;

%handles.peaks=peaks(35);
handles.membrane=membrane;
%[x,y]=(-8,0.5,8);
%r=sqrt(x.^2+y.^2)+eps;
%sinc=sin(r)./r;
%handles.sinc=sinc
handles.current_data=handles.membrane;
surf(handles.current_data)

% Choose default command line output for PeltonGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes PeltonGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = PeltonGUI_OutputFcn(hObject, eventdata,
handles)
% varargout    cell array for returning output args (see VARARGOUT);
% hObject     handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function minFlowRate_Callback(hObject, eventdata, handles)
% hObject     handle to minFlowRate (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of minFlowRate as
text
%           str2double(get(hObject,'String')) returns contents of
minFlowRate as a double

handles.minFlowRate=str2double(get(hObject,'String'));
% Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function minFlowRate_CreateFcn(hObject, eventdata, handles)
% hObject     handle to minFlowRate (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%           See ISPC and COMPUTER.

```

```

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function maxFlowRate_Callback(hObject, eventdata, handles)
% hObject      handle to maxFlowRate (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of maxFlowRate as
text
%           str2double(get(hObject,'String')) returns contents of
maxFlowRate as a double
handles.maxFlowRate=str2double(get(hObject,'String'));
% Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function maxFlowRate_CreateFcn(hObject, eventdata, handles)
% hObject      handle to maxFlowRate (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function incFlowRate_Callback(hObject, eventdata, handles)
% hObject      handle to incFlowRate (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of incFlowRate as
text
%           str2double(get(hObject,'String')) returns contents of
incFlowRate as a double
handles.incFlowRate=str2double(get(hObject,'String'));
% Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function incFlowRate_CreateFcn(hObject, eventdata, handles)
% hObject      handle to incFlowRate (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

end

```
function jetVelocity_Callback(hObject, eventdata, handles)
% hObject      handle to jetVelocity (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of jetVelocity as
text
%      str2double(get(hObject,'String')) returns contents of
jetVelocity as a double
handles.jetVelocity=str2double(get(hObject,'String'));
% Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function jetVelocity_CreateFcn(hObject, eventdata, handles)
% hObject      handle to jetVelocity (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbuttonCalculateAndPlot.
function pushbuttonCalculateAndPlot_Callback(hObject, eventdata,
handles)
% hObject      handle to pushbuttonCalculateAndPlot (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

%Count number of times we've entered this routine
handles.count = handles.count + 1;

% Calculate the power using this code
%g = 9.81; %Acceleration of gravity (m/s^2)
rho = 1000; %Density of water (kg/m^3)

%Dimension flow rate vector
flowRate =
(handles.minFlowRate:handles.incFlowRate:handles.maxFlowRate)';

tmp = size(flowRate);
nmax = tmp(1);
clear tmp

power=zeros(nmax,1);

for i=1:nmax
    [power(i)]=0.5*rho*flowRate(i)*handles.jetVelocity^2; % Maximum
power
end
```

```

%u_bucket = v_jet/2;
%head = 0.5*rho*g*v_jet^2;

%Plots
%subplot(2,1,1);
if (handles.count ==1)
    plot(flowRate,power, 'b');grid on;ylabel('Power
(Watts)');xlabel('Flow Rate (m^3/s)');
    hold on;
elseif (handles.count == 2)
    plot(flowRate,power, 'r');grid on;ylabel('Power
(Watts)');xlabel('Flow Rate (m^3/s)');
elseif (handles.count == 3)
    plot(flowRate,power, 'g');grid on;ylabel('Power
(Watts)');xlabel('Flow Rate (m^3/s)');
elseif (handles.count >= 4)
    plot(flowRate,power, 'c');grid on;ylabel('Power
(Watts)');xlabel('Flow Rate (m^3/s)');
end

% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes2

% --- Executes on mouse press over axes background.
function axes2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```