**Lab 8: Using Numerical Integration to Estimate Energy Usage**

**Background**

Smart meters take thousands of measurements of voltage and current every day and use these values to calculate instantaneous power and energy usage. Energy usage information is transmitted several times per day to a customer’s utility company which allows the utility company to monitor and analyze when energy is being used. This detailed data allows utility companies to run their generation and distribution operations more efficiently. Both commercial and industrial customers can take advantage of smart billing and reduce monthly utility bills.

Electric power is the rate at which energy is transferred by an electric circuit:

Power = Voltage \* Current

Power will be in units of watts (W) if voltage and current are in the standard units of volts (V) and amps (A) respectively.

The power company bills customers for energy usage. If power was a constant, then energy would just be the product of power and usage time. However, power demand is certainly not constant in a household or business. So, energy usage is:

A smart meter takes thousands of measurements of voltage and current every day. Each time a measurement is taken, power is calculated. A processor in the meter uses numerical integration to calculate the energy usage.

**Part A**

1. Fill in the column for Power in Table 1 using the equation:

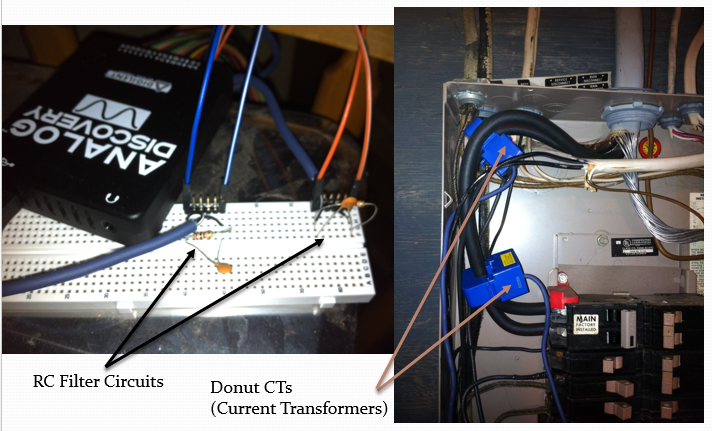
**Table 1: Calculating Energy using Numerical Integration**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Min** | **Second** | **Current (A)** | **Power (W)** | **DeltaT**  **(s)** | **Energy**  **(Ws)** | **Energy (kWh)** |
| **0** | **47** | **3** | **360** | **0** | **0** | **0** |
| **0** | **53** | **3.4** | **408** | **6** | **2340** | **0.0006400** |
| **1** | **10** | **3.6** | **432** | **17** | **7140** | **0.0019833** |
| **3** | **4** | **4** | **480** | **114** | **51984** | **0.0144400** |
| **5** | **24** | **3.6** | **432** | **140** | **63840** | **0.0177333** |

1. Fill in the column for DeltaT (time elapsed since previous measurement in seconds).
2. Estimate the Energy usage in watt-seconds (Ws) over each time period using the Trapezoidal Rule for numerical integration:
3. Convert energy values from watt-seconds (Ws) to kilowatt-hours (kWh) to complete the table. ***Note: utility companies use units of kWh for billing purposes.***
4. Check table values with T.A.s

**Part B**

Dr. Talaga, a computer science professor here at CEAS, decided to make his own smart meter using a couple of donut current transformers, a simple RC conditioning circuit, a data acquisition device (similar to the MyDAQ), and MATLAB. Figure 1 shows the basic set-up.



**Figure 1: Dr Talaga’s Smart Meter**

The donut current transformers were wrapped around the incoming power lines and produced a voltage proportional to the line currents. He wrote a MATLAB script which sent a command to the DAQ to take measurements about every seven seconds and calculated energy usage from these measurements. He even created a webpage that would allow him to monitor his energy usage throughout the day as shown in Figure 2.



**Figure 2: Webpage to Monitor Home Energy Usage**

Dr. Talaga took data from July 7th ,2013 through December 27th, 2013 and ended up with 1.87 million measurements (too big to open in excel by the way). We will be using a smaller subset of the data for this lab (September 13th through October 13th ).

1. Go to the Blackboard site under Lab 8 and download Lab8.mat. Save it in your current MATLAB directory.
2. At the MATLAB command prompt, type the command: >> load Lab8 to load a dataset array called Data.
3. Double click on Data in the Workspace Window to open it in the Variable Editor Window. Data is a dataset array with the following fields:

* Data.Month is a numerical vector with the month at which measurement was taken
* Data.Day is a numerical vector with the day at which measurement was taken
* Data.Year is a numerical vector with the year at which the measurement was taken
* Data.Hour is a numerical vector with the hour at which the measurement was taken
* Data.Min is a numerical vector with the minute at which the measurement was taken
* Data.Second is a numerical vector with the second at which the measurement was taken
* Data.Current is a numerical vector with the sum of the current measurements from the two input power lines into the house.

***Note: This is a very large dataset with over 300,000 measurements. So, you really need to remember to use semicolons to suppress outputs or you will be staring at your screen for quite some time watching data being displayed in the command window!***

1. Add a column to the dataset array called Power that calculates the power using the following command:

Data.Power = 120\*Data.Current;

Verify that a new column for Power has been added to the dataset array Data.

1. Add a column to the dataset array called DeltaT that calculates the time that has elapsed between the current measurement and the previous measurement using Data.Min and Data.Second (it is not necessary to use the Data.Hour values). Refer back to the calculations you did in Part A for DeltaT.

**Important Notes!**

It is possible to do this without a for loop. Just initialize Data.DeltaT(1) = 0 then use smart indexing on the minute and second columns to calculate the rest.

If you do decide to use a for loop, you want to avoid reading from and writing to the dataset array inside the loop. Why? Since the dataset array is so large, it will take a long time to execute (11 minutes on my computer)!

So, how do you avoid reading from and writing to the dataset array inside your loop. Simple. Prior to the for loop, create vectors for minutes and seconds: Min = Data.Min; and Sec = Data.Second;. Also create a column vector for DeltaT initially filled with zeros: DeltaT = zeros(# of rows in dataset array,1); Use the Min and Sec vectors inside your for loop to calculate correct values for the DeltaT vector. After your for loop ends, write the values to the dataset array: Data.DeltaT = DeltaT;

How much time is saved by not reading from and writing to the dataset array inside the loop? On my computer, the loop ran in 0.012 seconds (about 55,000 times faster).

MATLAB Commands to add DeltaT Column to Data:

%Kyle O'Connor

home;

clear all;

clc;

load Lab8;

Data.Power = 120\*Data.Current;

min = Data.Min;

size = 337678;

sec = Data.Second;

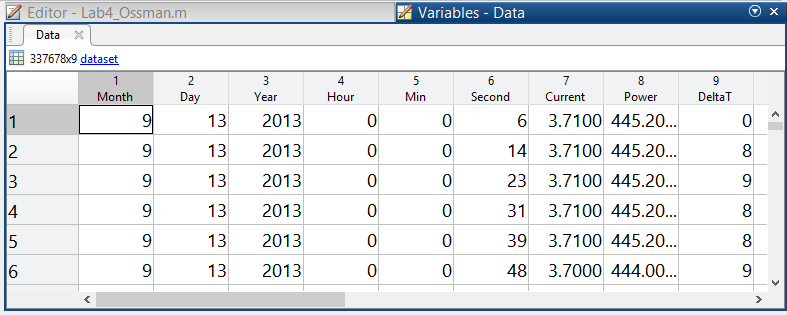
dt = zeros(size,1);

for c = 2:size

dt(c,1) = ((min(c,1)\*60) + sec(c,1)) - ((min(c-1,1)\*60) + sec(c-1,1));

end

1. Look at Data in the variable editor window. Verify that the first six rows have the same values for Power and DeltaT shown in Figure 3.



**Figure 3: Updated Dataset with Power and DeltaT**

1. Type the following in the command window (without semicolons) and paste results:

Data(10238:10240,:)

MATLAB Output:

Month Day Year Hour Min Second

9 13 2013 23 59 44

9 13 2013 23 59 52

9 14 2013 0 0 1

Current Power DeltaT

3.45 414 8

3.45 414 8

3.45 414 -3591

Data(40731:40733,:)

MATLAB Output:

Month Day Year Hour Min Second

9 16 2013 23 59 47

9 16 2013 23 59 56

9 17 2013 0 0 4

Current Power DeltaT

3.99 478.8 9

3.98 477.6 9

3.98 477.6 -3592

Clearly there is a problem with DeltaT when we roll to a new day. Determine the correct value for DeltaT in each of the cases above.

What do you need to add to the negative values to get the correct DeltaT:

for c = 1:size

if dt(c,1) < 0

dt(c,1) = (60 - Data.Second(c-1)) + Data.Second(c);

end

end

1. Use the find command to find all of the rows where Data.DeltaT is negative. Then fix these values. Re-run the commands in Step 7 to make sure the values for DeltaT are now correct.
2. Calculate the maximum DeltaT value and the average of the DeltaT values. Check results with your T.A.

Maximum DeltaT value = 415 seconds

Average of the DeltaT values = 7.9318 seconds

1. Now, add a column for Energy (in kWh) using the Trapezoidal Estimate for Numerical Integration:

**Units are in Ws**. **Convert to units of kWh.**

*Same issue here as with the DeltaT column. You can avoid a for loop using smart indexing, but if you can’t see how to do this without a loop, then make sure you avoid reading from and writing to the dataset array inside the loop – unless you have time to kill!*

Check your results by typing the following command at the MATLAB command prompt:

>> format short e; Data.Energy(1:6)

You should get the following output:

0

9.8933e-04

1.1130e-03

9.8933e-04

9.8933e-04

1.1115e-03

MATLAB Commands to add Energy Column to Data:

for c = 2:size

energy(c,1) = ((power(c,1) + power(c-1,1))/2)\*dt(c,1);

end

energy = (energy/1000/3600);

energy(size) = 0;

Data.Energy = energy;

1. As previously mentioned, this particular dataset has measurements from September 13th through October 13th in the year 2013. Calculate the total energy usage over this period of time. Check your answer with your T.A.

MATLAB Command:

totalEnergy = 0;

for c = 1:length(energy)

totalEnergy = totalEnergy + energy(c);

end

Total Energy Usage = 3.04.9957 kWh

1. Dr. Talaga’s bill from Duke Energy indicated that he used 320 kWh during this period of time. Calculate the % Error between the actual usage of 320 kWh and your estimated total usage from Step 10.

Percent Error = 4.8251%

1. Calculate the total energy usage of September 15th.

*Hint: use the find command to find the rows where Data.Month is 9 AND Data.Day is 15. Then add up the energy just in these rows.*

MATLAB Commands: Sept15 = sum(Data.Energy((find(Data.Month == 9 & Data.Day == 15))));

Total Energy Usage on September 15th = 16.3003 kWh

1. Calculate and plot the hourly usage on September 15th. Data.Hour uses a military clock so it starts with Hour 0 and ends with Hour 23. Include title and axis labels (with units).

Hint: for loop and find command.

MATLAB Commands: for i = 0:23

Sept15Hourly(i+1) = sum(Data.Energy((find(Data.Month == 9 & Data.Day == 15 & Data.Hour==i))));

end

plot(0:23,Sept15Hourly,'r\*')

count = 0;

PLOT:



What time of the day was the highest hourly usage?

18 Hours into the day, or 6pm

What time of the day was the lowest hourly usage?

5 Hours into the day, or 5am

1. Calculate and plot the daily usage from September 13th through October 13th. Include title and axis labels (with units). On the x-axis, you can just put 1 to 31 to represent the 31 days represented.

MATLAB Commands:

count = 0;

for i = 13:30

count = count +1;

Sept15DailyUse(count) = sum(Data.Energy((find(Data.Month == 9 & Data.Day == i))));

end

for i = 1:13

count = count + 1;

Sept15DailyUse(count) = sum(Data.Energy((find(Data.Month == 10 & Data.Day == i))));

end

plot(1:31,Sept15DailyUse,'b\*')

xlabel('Day (Sept 13th - Oct 13th')

ylabel('Usage (kWh)')

title('Daily Usage from Sept 13th - Oct 13th')

PLOT:

