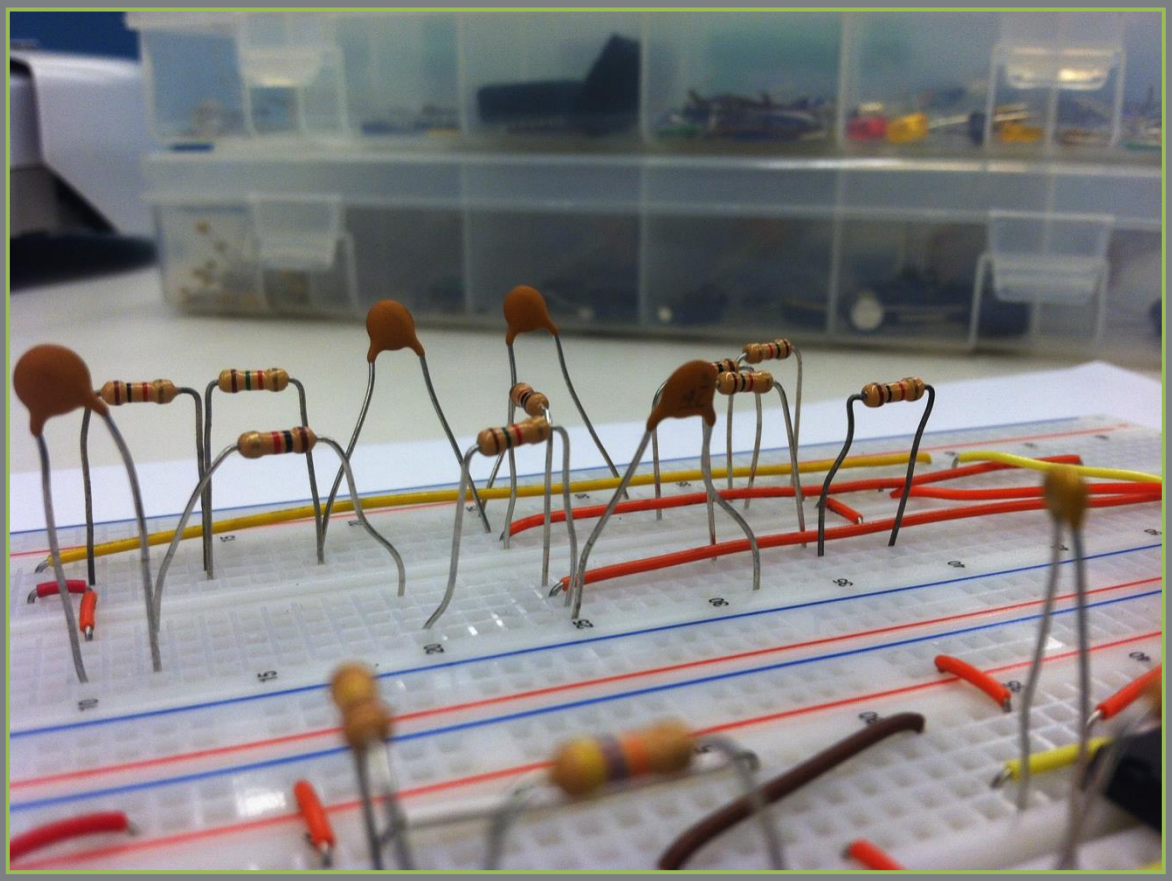


DESIGN DOCUMENT

MAY14-11, IMPEDANCE MEASUREMENT DEVICE FOR DETECTION OF CYANO-BACTERIA

GROUP MEMBERS
Tyler Bohlke
David Callen
Danielle Kimler
Watson Mulder

CLIENTS/ADVISORS
Dr. Degang Chen
Dr. Nathan Neihart

System Design Overview

Requirements

The device has several requirements to allow for ease of use for the intended customer.

Speed
The current testing speed for the average user is on the order of days. The goal for this device is to not only provide the user with same-day results, but to have results within 10 minutes.

Size
With this device, the lab comes to the user—without the size of laboratory-grade equipment. The intention is for the size of the product to be small enough to be held in the customer's hand.

Accuracy
The device is being designed with a goal of 1% accuracy in microcystin concentration measurement. Precision in the measurement of this toxin is vital to the application, as the lives of animals and people may depend on it.

Usability
The final user of this product will not need to have a technical background to utilize the device. All measurement can be done via a simple user interface.

Microcystin-LR concentration	**~0.5ug/L->20ug/L**
Capacitance	160-175 nF (3200-3500 nF/cm^2)
MCLR Capactiance change	1.5-3.0 nF (30-60 nF/cm^2)
Area of probe	0.049087 cm^2
Size of PCB	3"x4"
Supply Voltage	Run off 9V battery or perhaps 8 AAs (12V)
Output resolution	10 bits - between 160 and 175 nF: Range of 15nF, 14.65 pF step
Measurement Time	Below 10 minutes
Measurement Size	500mL sample size
Impedance measurement technique	Bridge method
Probe model	Series RC circuit
LCD Display	6 cm x 8cm

Definitions

- ***Zu* -** The internal impedance of the input to the microcontroller. This input reads the voltage difference across Zu to determine if the bridge is balanced.
- ***Ct* -** The tunable capacitance of the tuning branch in the bridge, adjusted by the microcontroller.
- ***Rt* -** The tunable resistance of of the tuning branch in the bridge, adjusted by the microcontroller.
- ***Bridge* -** The part of the circuit that is for impedance measurement, consisting of four impedance branches. When the ratio of the branches is satisfied, the circuit is balanced, and a value for capacitance can be extrapolated. Insert pic of bridge here
- ***MATLAB* -** Simulation software used for determining the characteristics of our bridge.

Functional Decomposition

Before the device is used, it undergoes an initial calibration step to make sure that the bridge is balanced, so that it outputs an accurate reading of the capacitance. The microcontroller accomplishes this by adjusting the tunable components of the circuit until the bridge is as balanced as possible.

After this, the device is immersed in the suspected water. If there is microcystin present in the water, then the bridge will become unbalanced, due to the capacitance change. The microcontroller again goes to work, adjusting the tunable capacitance until the bridge is balanced again. The microcontroller keeps track of the capacitance it adjusts, and uses this to calculate the capacitance change from the microcystin, and the corresponding microcystin concentration.

Finally, the microcontroller sends the information to an LCD, which will give a reading of the capacitance and the microcystin concentration, along with a diagnosis of the safety level of the water.

Analysis

The Circuit

The main emphasis of our analysis of the bridge circuit was finding a relationship between the voltage across Zu and the known impedances, along with the input voltage from the oscillator. The branches of the bridge can be expressed by the matrix:

$$\begin{bmatrix} \boldsymbol{Z_P + Z_C} & \boldsymbol{-Z_P} & \boldsymbol{-Z_C} \\ \boldsymbol{-Z_P} & \boldsymbol{Z_P + Z_\mu + Z_{P+M}} & \boldsymbol{-Z_\mu} \\ \boldsymbol{-Z_C} & \boldsymbol{-Z_\mu} & \boldsymbol{Z_C + Z_\mu + Z_t} \end{bmatrix} * \begin{bmatrix} \boldsymbol{I_1} \\ \boldsymbol{I_2} \\ \boldsymbol{I_3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V_S} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}$$

When solving this matrix (in MATLAB), the intent was to find the current through Zu and then multiplying by said impedance.

$$\boldsymbol{V_{Z_\mu}(V_S, Z_P, Z_t, Z_C, Z_{P+M}, Z_\mu) = (I_2 - I_3) * Z_\mu}$$

The final expression for Zu can be found in the APPENDIX. Once we obtained this function of known variables, we were able to use it to simulate several parameters of the circuit; including how the output across Zu is affected by different oscillator wave types, and what the voltage change across Zu is for different values of Ct.

Design Specifications

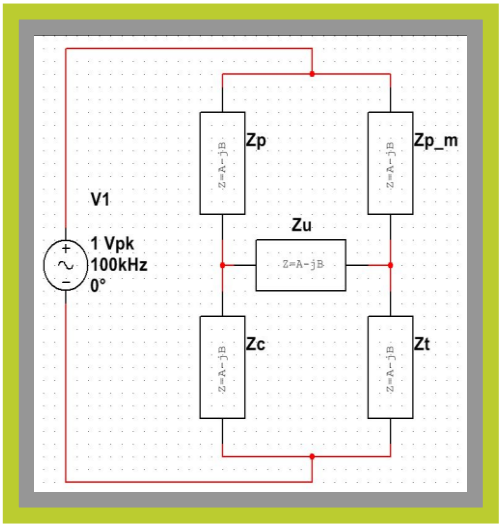
Input/Output

Input

The input to our circuit is composed of two components: an oscillator, and an electrode painted with antibodies that changes in capacitance under specific ranges of microcystin concentration.

- The electrode is the most crucial input component. This piece changes in capacitance based on microcystin concentration. When the capacitance changes, the bridge becomes unbalanced. The bridge is then balanced with the microcontroller, which yields values for concentration and capacitance.
- The oscillator provides a signal for the input of the bridge and serves as a reference for the voltage across Zu.

Output
The output of our circuit is sent from the microcontroller to the LCD. It provides the user with data regarding the capacitance measured and the concentration of microcystin.



User Interface
The user interface is simple enough that a non-technical individual will be able to successfully use the device. One will simply turn the circuit on, place the designated electrode into suspect water, and wait. A few moments after the electrode has been submerged, the LCD will present the capacitance and the concentration of the microcystin, and whether or not the water is safe.

Hardware
The final circuit will have six main hardware components: an oscillator, a balancing bridge, a feedback network from the microcontroller to the bridge, an amplifier for the voltage across Zu, a microcontroller to control the bridge and to serve as a user interface, and an LCD.

1. **The Oscillator -** The oscillator serves as the primary input to the bridge. Its purpose is to facilitate a usable frequency to be measured across the bridge. It will be composed of a sinusoidal waveform of an amplitude yet to be determined. The operating frequency of the oscillator is 100 kHz. This frequency is best suited for accurate measurements of voltage across our bridge.
2. **The Balancing Bridge -** The balancing bridge is the core of our circuit. It is composed of four separate impedances, one of which is tunable by the microcontroller. Once the bridge satisfies the impedance ratio, the voltage across Zu, will be essentially zero, and we will be able to extrapolate a value for Cm.
3. **The Feedback Network -** The feedback network is, in itself, part of the bridge, but it is its own separate piece of hardware. It is composed of an electronically tunable resistor and capacitor. These two elements are controlled by the microcontroller in order to balance the bridge by satisfying the impedance ratio.
4. **The Amplifier -** The amplifier is needed to deliver a measurable voltage from Zu to the microcontroller. When the voltage across Zu gets too small (in the process of balancing the bridge), it will not produce a signal large enough for the microcontroller to recognize. When the small signal is amplified, the microcontroller can tune it to a point where it is small enough for a precise measurement.
5. **The Microcontroller -** The microcontroller is the interface between the bridge, user input, and LCD output. It will have a least a 10-bit ADC for taking voltage inputs. It is embedded with the tuning algorithm which adjusts the capacitor and resistor for tuning the bridge, and for tuning the feedback resistors for the amplifier to continuously increase the amplification as the voltage across Zu drops. Because it tunes the capacitor values, it will know the final value of capacitance (and thus the concentration of microcystin) and will output those values to the LCD.

6. The LCD - The LCD serves as the display for the user. The values obtained from the microcontroller (capacitance and concentration of microcystin) will be displayed for the user to determine if the levels of microcystin are safe for any purpose.

Software

The software used for the algorithm is simply block modules of MATLAB code (will be implemented in C for the microcontroller in the spring of 2014). Each block tunes a specific element and works in conjunction with the microcontroller. One block of code is used for tuning the resistor Rt, and another block of code tunes the capacitor Ct. In order to balance the bridge, the microcontroller uses a binary search algorithm.

For example, in order to balance the bridge in response to a capacitive change, the microcontroller uses the following procedure:

1. Start by setting the upper limit of Ct at the maximum, and the lower limit at the minimum.
2. Tune Ct to the center of the range between the upper and lower limits (this will be Ct1), and read the voltage Vu. This will be Vu1.
3. Tune Ct one step higher (this will be Ct2), and read Vu again. This will be Vu2.
4. If Vu2 is less than Vu1, we need to adjust the capacitance higher. Set the lower limit of the Ct range to be at Ct1, and repeat from step 2.
5. If Vu2 is greater than Vu1, we need to adjust the capacitance lower. Set the upper limit of the Ct range to be at Ct1, and repeat from step 2.

The program breaks out of the loop when we reach the limit of our precision, i.e., when Ct2 exceeds the upper limit. The microcontroller then uses the current value of Ct1 to calculate the capacitance change from the microcystin.

Simulation

We simulated all of our design parameters in MATLAB. The simulations completed were:

- A simulation of the change in voltage across Zu as a function of a variable capacitance Ct
- A simulation to find the change in tunable values of Ct (range and step size)as the concentration of microcystin increased
- A simulation for the amplitude of the voltage across Zu as the waveform of the oscillator changed (adding harmonics to a sine wave)
- A simulation of the tuning algorithm that includes
    - Tuning Ct using a given step size
    - Tuning Rt using a given step size
    - Tuning the amplifier with a given step size

Testing

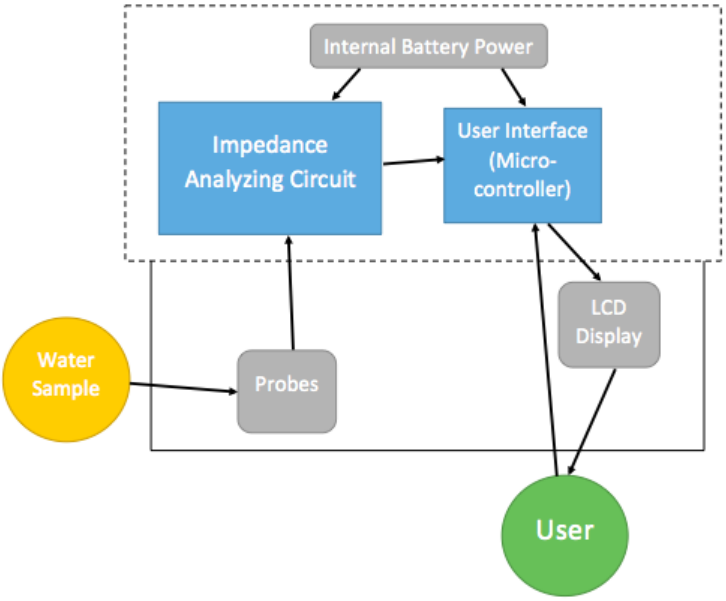
Testing will occur once the prototype is complete. This process will occur in the spring of 2014 and will be comprised of debugging any hardware or software issues. The hardware will be installed on the PCB in the spring, and the tuning algorithm will be tested and implemented in C on the microcontroller by then as well.

Prototype

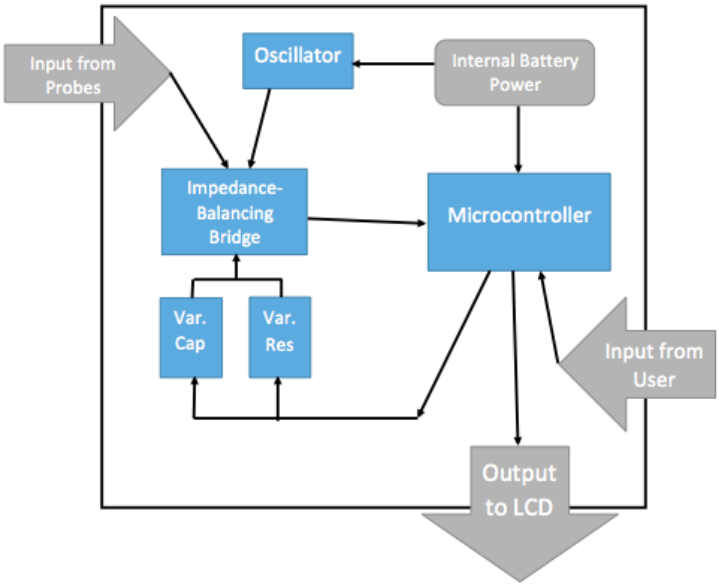
Our team is working towards a prototype by the end of the semester. The prototype will include a fully laid-out printed circuit board, and all components will be selected. We are at the point where we know what components we need, but we don't know the specific part numbers yet. After Thanksgiving break we will have most of the simulations, along with the algorithm complete, and the necessary components selected. This will lead us into board layout, and then testing by the spring of 2014.

Figures

General System Block Diagram

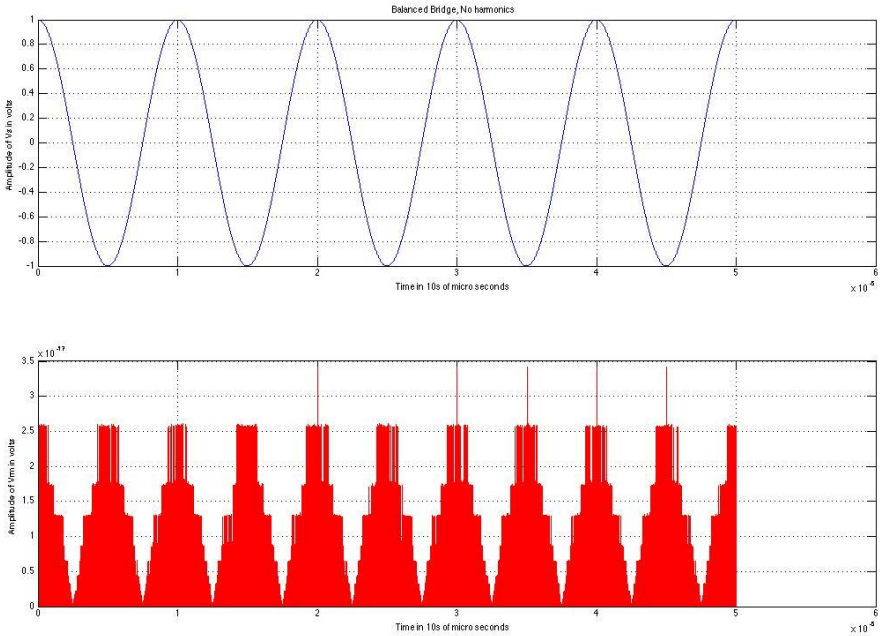


Impedance Analyzing Circuit and Microcontroller

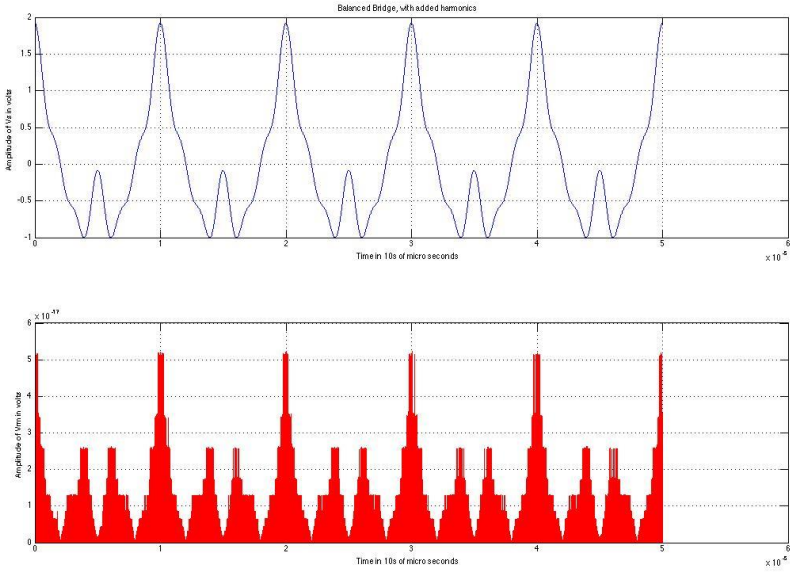


The Necessity of Sine Wave Purity at the Bridge Input

A balanced bridge with no harmonic distortion in Vs

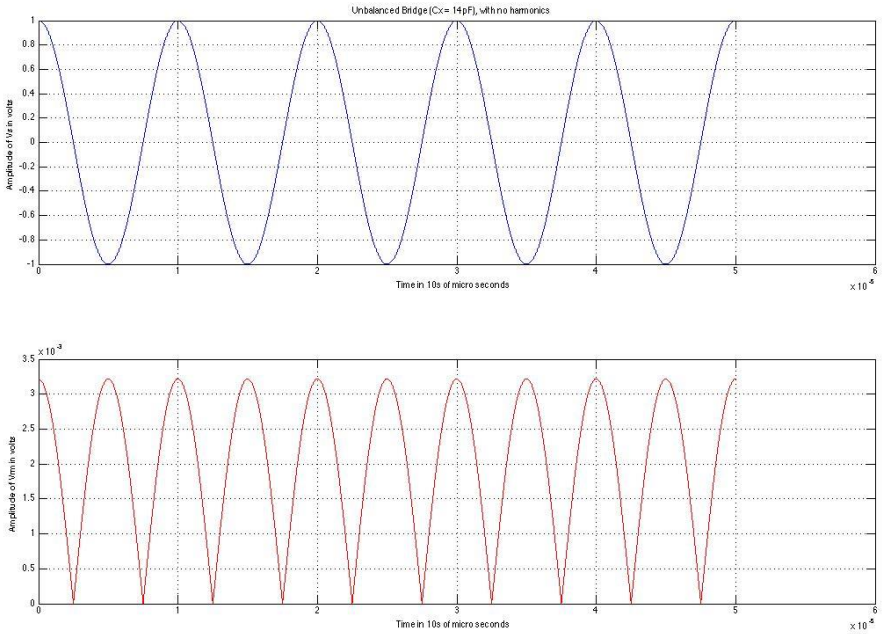


A balanced bridge with three even harmonics at the input (Vs)

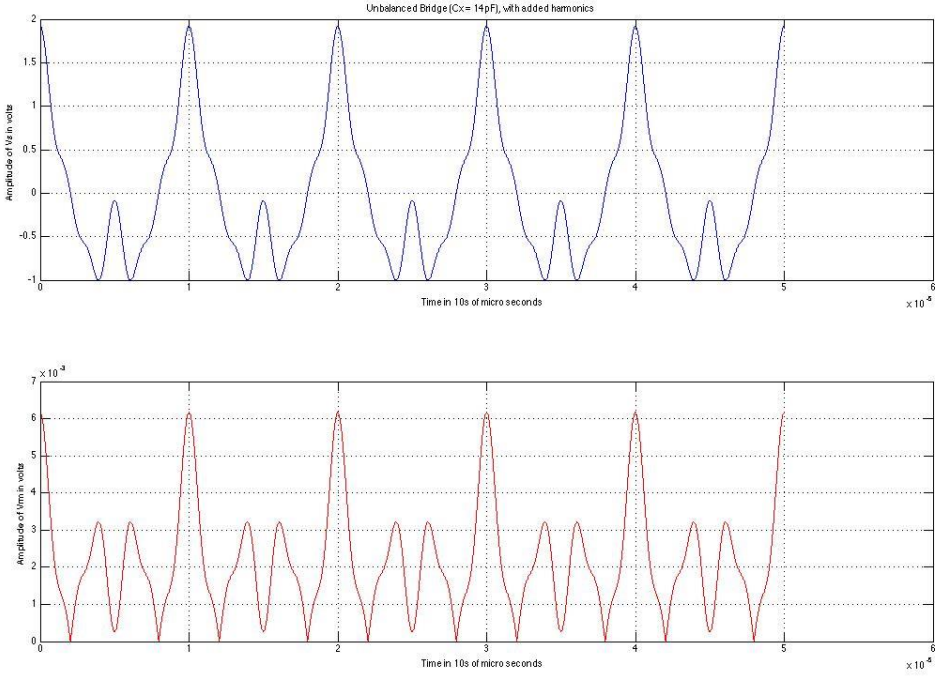


An unbalanced bridge with no harmonic distortion in Vs

Cx = 14pF rather than 16pF



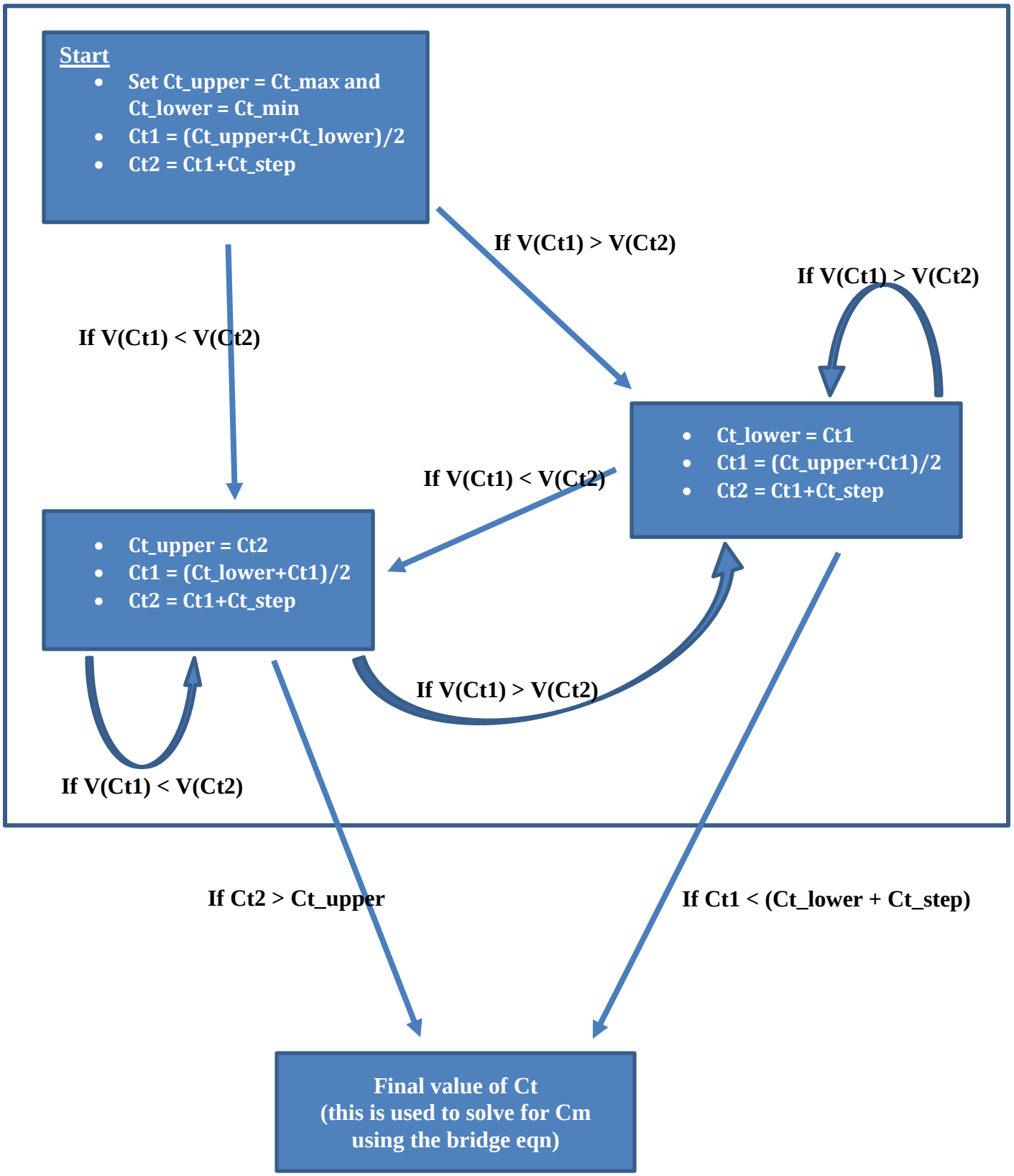
An unbalanced bridge with three even harmonics at the input (Vs)
Cx = 14pF rather than 16pF




```
f = 100000;
w = 2*pi*f;
```

```
Fs = 1/10000000000; %sampling rate
t = [0:Fs:0.00005];

Rm = 10000; % estimated Rin of microcontroller

Ra = 2500; % the value guiven to us by the grad students
Rp = 2500;
Rt = 2500;
Rx = 2500;

Ca = 16e-12; % in Farads, the e-12 means pico
Cp = 16e-12;
Ct = 16e-12;
Cx = 14e-12;

Xa = 1/(w*Ca);
Xp = 1/(w*Cp);
Xt = 1/(w*Ct);
Xx = 1/(w*Cx);

Za = Ra-1i*Xa; % we can define our imedances however we want right now,
Zp = Rp-1i*Xp; % so i just used a series res and cap
Zt = Rt-1i*Xt;
Zx = Rx-1i*Xx;

Vs = cos(t*w)+cos(2*t*w)/2+cos(4*t*w)/4+cos(6*t*w)/6;

%The expression for Vrm in terms of Vs and known impedances given by
%the Vrm solution code
Vrm = -Rm*((Vs*Zp + (Rm*Za*(Rm*Vs*Za + Rm*Vs*Zp + Vs*Za*Zp +
Vs*Za*Zx))...
/(Rm*Za*Zt + Rm*Za*Zx + Rm*Zp*Zt + Rm*Zp*Zx + Za*Zp*Zt + Za*Zp*Zx + ...
Za*Zt*Zx + Zp*Zt*Zx) + (Rm*Zp*(Rm*Vs*Za + Rm*Vs*Zp + Vs*Za*Zp + ...
Vs*Za*Zx))/(Rm*Za*Zt + Rm*Za*Zx + Rm*Zp*Zt + Rm*Zp*Zx + Za*Zp*Zt + ...
Za*Zp*Zx + Za*Zt*Zx + Zp*Zt*Zx) + (Za*Zp*(Rm*Vs*Za + Rm*Vs*Zp + ...
Vs*Za*Zp + Vs*Za*Zx))/(Rm*Za*Zt + Rm*Za*Zx + Rm*Zp*Zt + Rm*Zp*Zx...
+ Za*Zp*Zt + Za*Zp*Zx + Za*Zt*Zx + Zp*Zt*Zx))/(Rm*Za + Rm*Zp + Za*Zp...
+ Za*Zx + Zp*Zx) - (Rm*Vs*Za + Rm*Vs*Zp + Vs*Za*Zp + Vs*Za*Zx)...
/(Rm*Za*Zt + Rm*Za*Zx + Rm*Zp*Zt + Rm*Zp*Zx + Za*Zp*Zt + Za*Zp*Zx...
+ Za*Zt*Zx + Zp*Zt*Zx));

figure
subplot(2,1,1);
plot(t,Vs);
grid on
title('Unbalanced Bridge (Cx = 14pF), with added harmonics');
xlabel('Time in 10s of micro seconds');
ylabel('Amplitude of Vs in volts');
subplot(2,1,2);
plot(t,abs(Vrm),'red');
grid on
xlabel('Time in 10s of micro seconds');
ylabel('Amplitude of Vrm in volts');
```

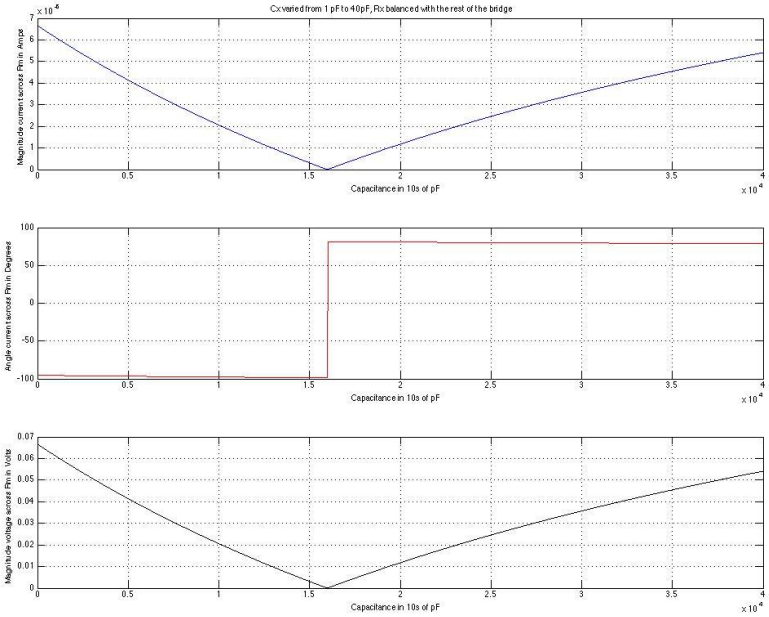
Tuning Algorithm



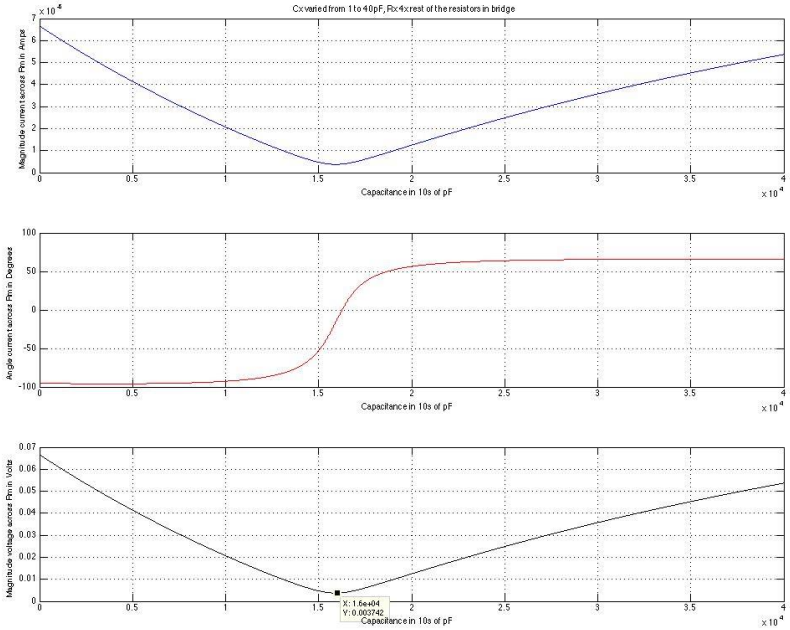
The Effects of Unbalanced Bridge Conditions

The current (angle and magnitude) and voltage (magnitude) across Rm as Cx varies from 1pF to 40pF while Rx remains balanced

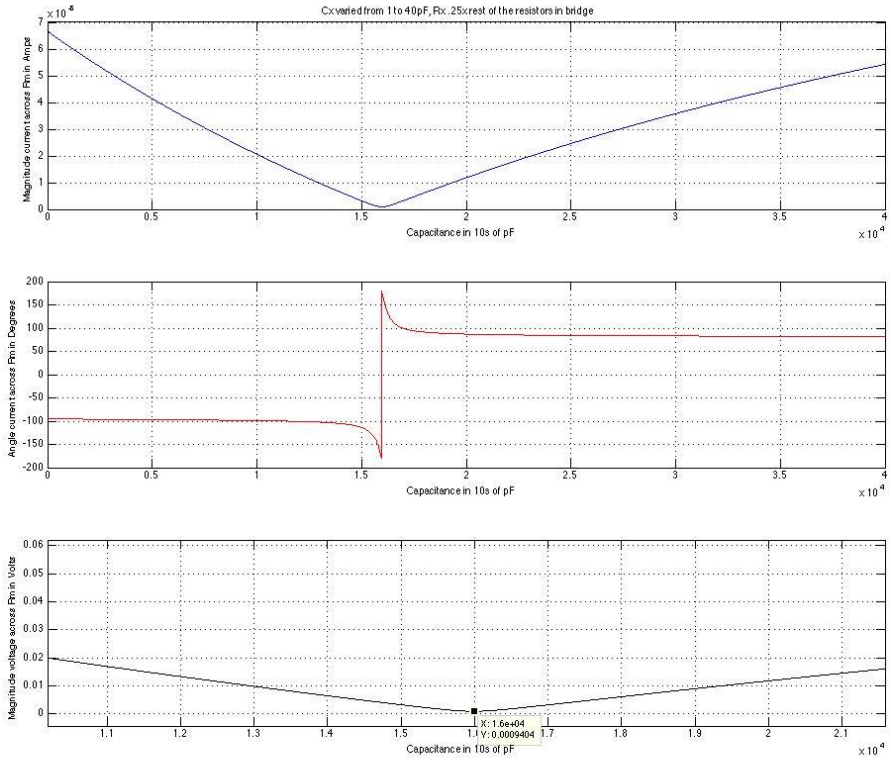
7e-5



The current (angle and magnitude) and voltage (magnitude) across Rm as Cx varies from 1pF to 40pF while Rx is 4x the size of the rest of the bridge resistors



The current (angle and magnitude) and voltage (magnitude) across Rm as Cx varies from 1pF to 40pF while Rx is 0.25x the size of the rest of the bridge resistors




```
w = 2*pi*100000;
Vs = 2*cos(w);  %random voltage value for now

Rm = 10000; % estimated Rin of microcontroller, will probably be high
            % but the smaller the better
Ra = 2500; % the value given to us by the grad students
Rp = 2500;
Rt = 2500;
Rx = 625;

Ca = 16e-12; % in Farads, the e-12 means pico
Cp = 16e-12;
Ct = 16e-12;
Cx = 16e-12;

Xa = 1/(w*Ca);
Xp = 1/(w*Cp);
Xt = 1/(w*Ct);
Xx = 1/(w*Cx);

Za = Ra-1i*Xa; % we can define our imedances however we want right now,
Zp = Rp-1i*Xp; % so i just used a series res and cap
```

```
Zt = Rt-1i*Xt;
Zx = Rx-1i*Xx;

i2=[];
i3=[];
C = 1:1:40000;  %define an array for our variable cap

for n = 1:1:40000

Xx1 = 1/(w*n*1e-15); %sweeps the capacitance from 1fF to 40pF
Zx1 = Rx-j*Xx1;

%Matrix solved by watson and i
impedance_matrix = [Zp+Za -Zp -Za; -Zp Zp+Rm+Zx1 -Rm; -Za -Rm
Za+Rm+Zt];
voltage_matrix = [Vs;0;0];
current_matrix = linsolve(impedance_matrix, voltage_matrix);

i2=[i2 current_matrix(2,1)];%stores the range of i2 values
i3=[i3 current_matrix(3,1)];%stores the range of i3 values

end

total_current = i2-i3;
voltage_Rm = total_current.*Rm;

figure
subplot(3,1,1);
plot(C,abs(total_current))%looking at magnitude of difference in
current
title('Cx varied from 1 to 40pF, Rx .25x rest of the resistors in
bridge');
xlabel('Capacitance in 10s of pF');
ylabel('Magnitude current across Rm in Amps');
grid on
subplot(3,1,2);
plot(C,angle(total_current)*180/pi,'red')%angle of difference in
current
xlabel('Capacitance in 10s of pF');
ylabel('Angle current across Rm in Degrees');
grid on
subplot(3,1,3);
plot(C,abs(total_current).*Rm,'k')%resistance across Rm
xlabel('Capacitance in 10s of pF');
ylabel('Magnitude voltage across Rm in Volts');
grid on
```