Ascend Geo: Levitation Test

Client: Michael Ewing, Ascend Geo
Team: Ryan Lee, Tyler Tift

Ascend Geo is a company that designs hardware and software for geophysical applications. One of the most commonly used tools in this field is a geophone, a device used to measure ground velocity in conjunction with many others. Often, however, a geophone is planted improperly or becomes flooded, causing it to behave differently. The purpose of this project was to create a simulation modeling a geophone's reaction to input currents and design an algorithm to determine a geophone array's functionality. The simulation's purpose was to aid in determining what bounds should be set on the data so that the algorithm can decide to reject further input from an improperly functioning geophone array.

Ascend Geo Levitation Test

Client: Michael Ewing, Ascend Geo

Team: Tyler Tift, Ryan Lee

Table of Contents

Introduction

Ascend Geo is a subsidiary of Aspect Energy, a company dedicated to finding new natural gas and oil reserves. Ascend Geo develops both hardware and software, used in geophysical applications, to aid the discovery of new resources. One of the most commonly used tools to search for reservoirs is a geophone. A geophone is a device that measures ground velocity through a mass-spring system. A mass with a copper wire wrapped around it sits inside a magnet. The magnet, in turn, is attached to a solid plastic casing. This casing is mounted on a spike, which is then planted in the ground. After setting up all of the geophones to be tested, they are excited by a ground movement, which is usually caused by an explosion or some other large shock. As a geophone moves, the casing, attached to the ground through the spike, moves around the mass. As the mass then proceeds to oscillate through the magnetic field, a current is induced in the geophone's internal circuit. This produces an output voltage from the geophone relative to the ground velocity. In order to obtain meaningful data, many of these geophones are connected in series, making what is known as a "line." These lines are then attached in parallel with each other in what is called a geophone array. All the information is collected by a recording device and analyzed later at a host machine. Using this data, some fairly accurate estimates as to the composition of the ground beneath the surface can be made.

The problems often experienced in the field are that geophones are poorly planted in the ground or the information they give becomes otherwise unusable due to ambient conditions, including floods, landslides, or various animals attempting to eat the geophones, which cause spurious noise. Unfortunately, the recording device that a geophone array is attached to indicates a functioning status as long as anything that has some internal resistance is attached to it. In this case, all the geophones attached to a device may not be returning any data at all, rendering the entire endeavor useless.

There are two goals for this project: create an accurate geophone simulation, and develop an algorithm to help determine whether geophones are functioning. The simulation will be modeled after what is referred to as a "levitation test." In this situation, instead of the geophones being excited by a shockwave in the ground, a constant current is introduced to the geophone array. This current is abruptly cut off and the resulting voltage produced by the geophones is recorded. This test is typically used to determine whether a geophone is still functioning properly after particularly rough handling or to help calibrate various elements of the geophone. The simulation allows the alteration of various parameters, including internal resistance and natural frequency of the geophone. Using this simulation, practically any condition in the field can be emulated. At the same time, ideal functionality of a geophone array can also be simulated. The algorithm for making a decision about array functionality will work by rejecting any data that falls too far outside of the ideal values that should be obtained from the geophone array. This algorithm will be implemented on the recording devices, and with the aid of the host machine to calculate the values to be compared, can end the recording process entirely so that analysts don't have to slog through useless data.

Requirements

I. Functional requirements

1. Characterization of input parameters and system response for a geophone.

2. Creation of a simulator to capture input parameters and generate data response.

3. The simulator will have inputs of tilt of the geophone, physical characteristics of geophones, and number of geophones.

4. Graphical representation of the geophone or array of geophones.

5. Graph the results of the simulations.

6. Analysis of data response and comparison to known responses for the creation of a device state algorithm and approximation of device parameters.

Of these requirements, the first is the most important for the simulation. The client absolutely requires that requirement 6 be met, but in order to do so, the first two requirements must first be achieved. Therefore, creation of the simulation with an accurate geophone response is paramount to creating an algorithm to determine geophone array functionality.

II. Non-Functional requirements

1. Students will need to be able to meet with employees on-site (Golden area) at various points in the project

2. Must create interactive GUI for simulation

3. Algorithm must be .NET 2.0 compatible

4. Client would prefer use of Visual C++ 2005 as IDE

5. The simulation should be stand alone.

6. Output should be viewable in Excel

7. The algorithm needs to be under a few KB to fit on the recording device

8. Accessible user interface with intuitive design

The client specifically stated that the algorithm's final implementation must be able to run in .NET 2.0. In addition, the client stated that he would greatly prefer the IDE used for this project to be Visual C++ 2005 Express Edition. These requirements determined the developing environment for the project.

As a final note, the simulation should be able to run regardless of what computer it's on as long as the proper resources are included in its folder. The simulation should be able to run in any Windows 95 or greater environment. Also, the data generated by the simulation should be able to be exported to an Excel file so that it can be viewed outside of the GUI should the user choose to do so.

III. Scope

This project encompasses the creation of a simulation of geophones and an algorithm analyzing functional status. The simulation must be accurate enough to allow for the modification of geophone parameters to emulate most conditions that can be experienced in the field. The algorithm must be small enough to fit into a very limited space so that it can be implemented on the recording devices. By the end of the 6th week, the algorithm must be provided to the client and the simulation must provide a reasonable response.

System Design

I. Design Goal and Risks

The goal of the project is to create a simulation that accurately represents a geophone's response in a levitation test and use these results to determine an algorithm and tolerance values for geophone functionality. In addition, an individual geophone's values in the simulation should be alterable in order to simulate field conditions.

Required Tasks

1. Set up developing environment: Windows Forms Application in Visual C++ 2005

2. Create GUI skeleton for later editing

3. Derive equation to approximate geophone response

4. Write functional classes

5. Implement GUI inputs

6. Implement graphing area

7. Implement visual representation of geophone array

8. Add additional elements requested by client

Risks

1. Technology Risks

The primary risk for this project was the ability to accurately determine the response of a geophone in given situations.

Limiting the algorithm to the codespace available on the recording devices was a bit daunting until it was actually implemented. As it worked out, the algorithm simply requires some values to be calculated on a host and then compared on the geophone recording device.

2. Miscellaneous Risks

A lot of the simulation's effectiveness depended on being able to correct it to better resemble actual data. As we never actually received this data, there was no way to actually verify the model.

II. High-level Design

1. User Interaction

The system will accept input from the user (individual geophone eccentricities, environment, etc.) and output a resulting graph and optionally a text file for later analysis, as shown below in Figure 1. The program will also output a functioning status based on a tolerance input by the user.

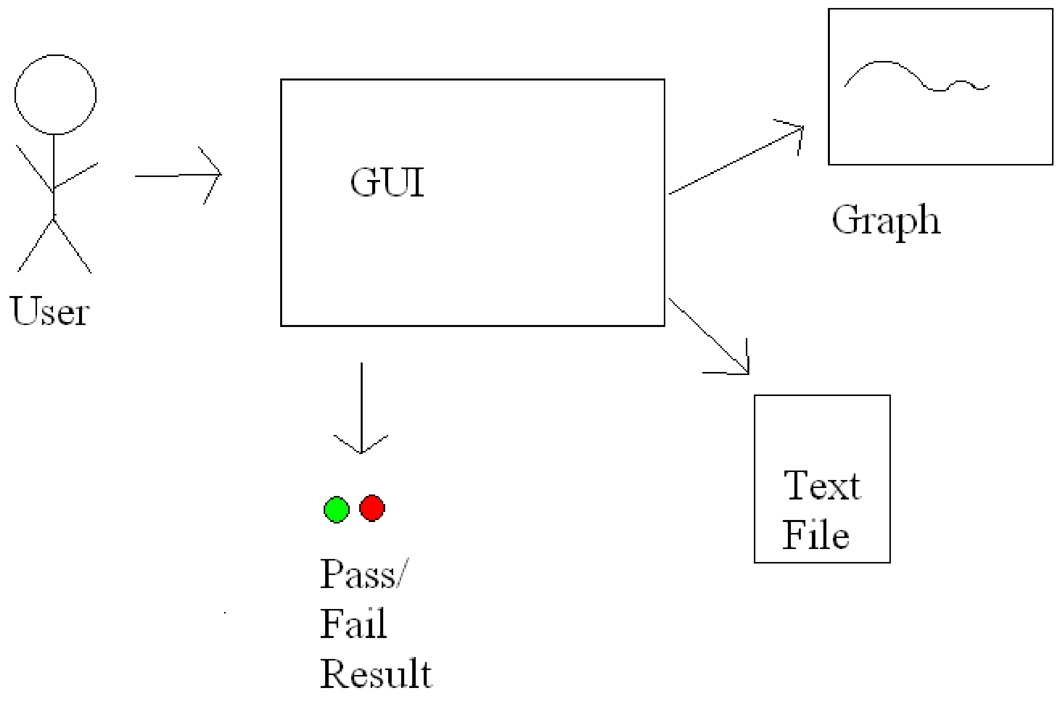


Figure 1: User interaction

2. Design Process

Several steps were taken, in order to accurately simulate a geophone's response to electrical input. First, a simple harmonic oscillator without damping was modeled. The system was then modified to under-damped harmonic motion using a rough approximation of a geophone's response. Then, based on physical data about a geophone the under-damped model was rectified to better represent a geophone's output. The next phase of enhancing the simulation was to include analyzing actual data to determine whether the simulation model is comparable to realistic response. Unfortunately, due to uncontrolled circumstances, no actual data was ever received.

The algorithm decides whether a line is functioning within given tolerances of precalculated theoretical values.

The GUI consisted of designing and modifying a Visual Studio form application. The form uses an external graphing utility and takes several key inputs, e.g. current, shunt resistance, etc. The GUI also contains an interactive display of the chain of geophones that allows for changes in input data to be made to individual units.

III. Detail Design

1. UML Diagram

A UML diagram detailing the interaction between proposed classes is shown in figure 2 below.

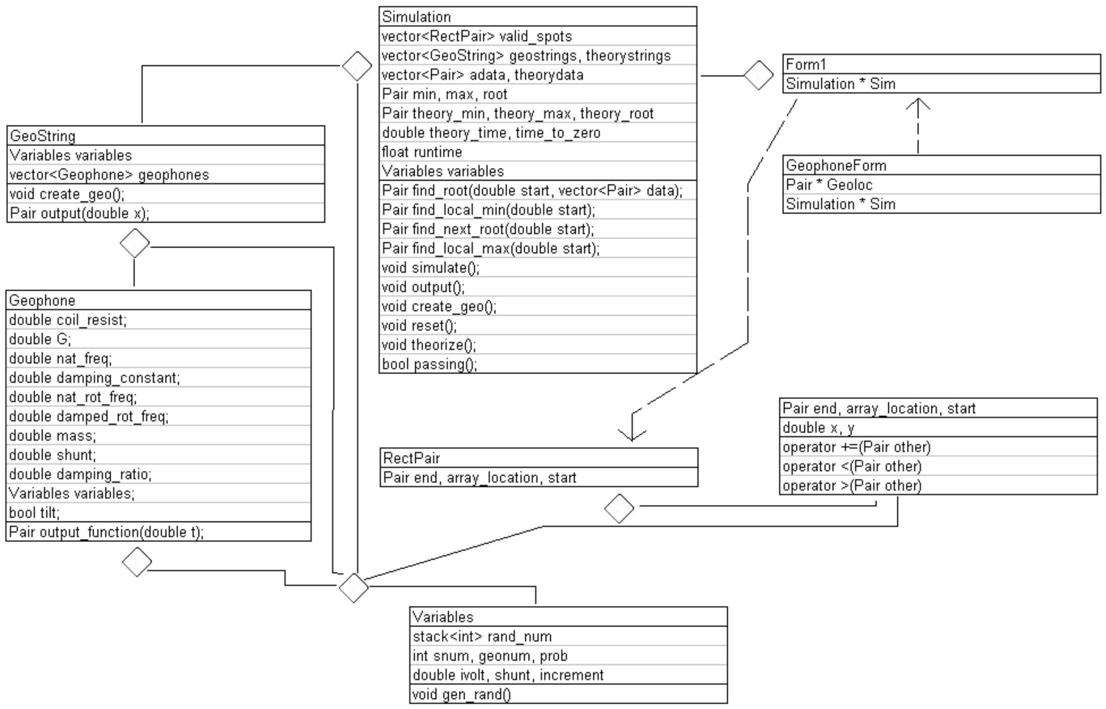


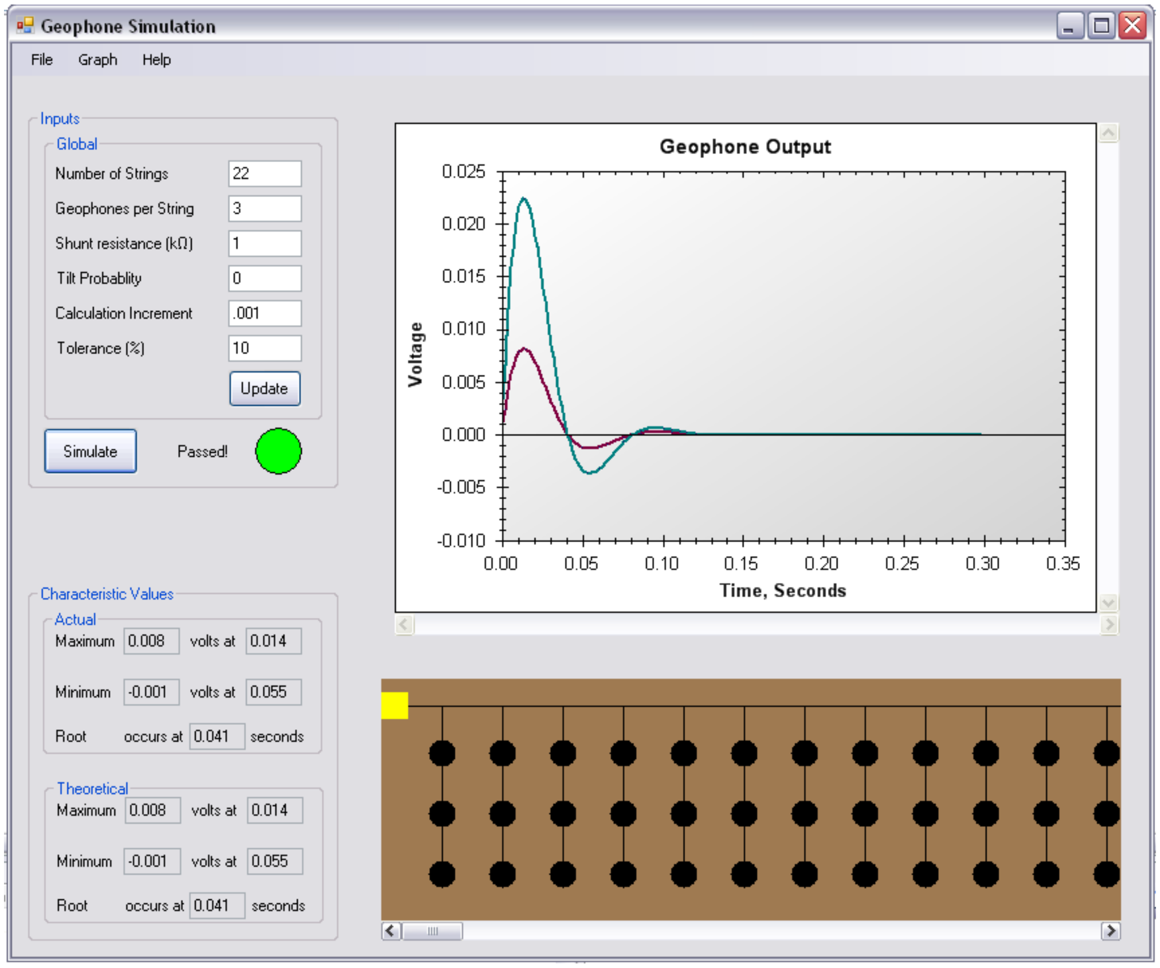
Figure 2: UML diagram

2. Description

Form1 contains an instance of Simulation and dynamically creates a GeophoneForm when needed. Simulation contains a vector of GeoStrings, each of which contains a vector of Geophones. Simulation, GeoString, and Geophone all contain an instance of the Variables class and several Pairs. Form1 also uses RectPair for determining whether a click in the interactive area is valid.

C. GUI

Below is a screenshot of the GUI



Inputs

Global

Number of Strings 22

Geophones per String 3

Shunt resistance (kΩ) 1

Tilt Probablity 0

Calculation Increment .001

Tolerance (%) 10

Passed!

Characteristic Values

Actual

Maximum 0.008 volts at 0.014

Minimum -0.001 volts at 0.055

Root occurs at 0.041 seconds

Theoretical

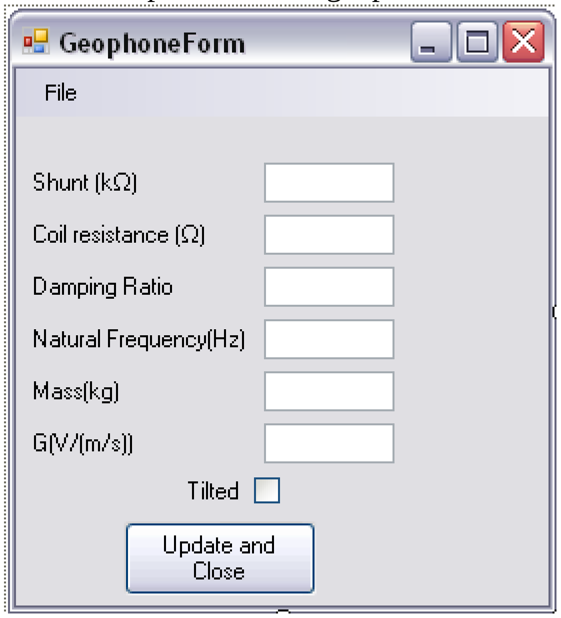
Maximum 0.008 volts at 0.014

Minimum -0.001 volts at 0.055

Root occurs at 0.041 seconds

Figure 3: GUI ScreenShot

Below is a picture of the geophone alteration window.



Shunt (kΩ)	
Coil resistance (Ω)	
Damping Ratio	
Natural Frequency(Hz)	
Mass(kg)	
G(V/(m/s))	

Figure 4: GeophoneForm

IV. Formula Design

In order to accurately simulate a geophone's response to a step current, quite a bit of research was required; a significant portion of the work included determining exactly how a geophone works and how certain physical characteristics interact with each other. After determining these relationships and acquiring the necessary data from our client, we were able to produce a second order linear differential equation modeling the motion of the geophone given an input current. After going through the process of integrating this equation and taking into account voltage sensitivity, we finally arrived at an equation for output voltage in terms of an input current. This input current is determined by the number of geophones in the array and their individual impedances.

The equation for output voltage in terms of input current is defined as

$$V(t) = -G e^{-\zeta \omega_0 t} ((\zeta \omega_0 \sin(\omega_d t) - \omega_d \cos(\omega_d t)) B + (\omega_d \sin(\omega_0 t) - \zeta \omega_0 \cos(\omega_d t)) A) \qquad \text{Eq. 1}$$

where

$$\omega_d = \omega_0 \sqrt{1 - \zeta^2} \qquad \text{Eq. 2}$$

$$B = \frac{1}{\omega_d} \zeta \omega_0 \frac{-G i_0}{k} \qquad \text{Eq. 3}$$

$$A = \frac{-G i_0}{k} \qquad \text{Eq. 4}$$

G is the voltage sensitivity, ω_0 is the natural frequency of the system, i_0 is the input current, ζ is the damping ratio, k is the spring constant, m is the mass, and t is the time.

Design Decision

We did not have to choose the language as the client asked us to use Visual Studio C++ 2005 and .NET 2.0. Our client also asked us to have a graphical display of the simulation. He suggested we use .NET to control Excel in order to facilitate data manipulation. However, we had trouble getting .NET and Excel to talk to each other so we decided to make a menu item that will output the data to a comma separated value (CSV) file. We then had to find a way to graph the data since we were no longer using Excel. We thought about writing a graphing function ourselves, but that might have been limited in functionality due to the time constraint. Another option was to find some previously written code that we would be able to use. We ended up going with the latter option and we were able to find an exceptional graphing utility: ZedGraph.

ZedGraph has much more functionality then we would have been able to come up with in the short time we had to work on it. Some of the functionality of ZedGraph that helped us decide to use it was that it has zooming and panning functions. These functions are useful, but would have been beyond the scope of this project for us to implement ourselves. After Finding Zedgraph the next problem was its licensing. ZedGraph is licensed under Library General Public License (LGPL) which states that if any modifications to the library are made, then the source code must be made public. However, if no changes are made to the library the program, it is just a work that uses the Library and is outside of the scope of the license. Since no changes were made to the library in our development, there is no fear of violating the license agreement.

We wanted an interactive interface to show the setup of the geophone array. The first step to this was to dynamically draw a picture representing the orientation of the geophone array. This part was fairly straight forward. We are using the System::Drawing library to make the graphics in a PictureBox. After we implemented the graphics, we ran into a problem. In some cases the Graphics object we were trying to draw was bigger then the box we drawing it into. Since the PictureBox class does not have an auto scroll feature we had to put scrollbars in manually. When we did this the scrollbars would not grab and move the graphics that were drawn into it. Our first approach was to use a child class of the Image class, Drawing::Imaging::Metafile. We drew the graphics to a metafile image then saved a copy of the image to the computer and reloaded it. However, when we reloaded the image to the PictureBox it was still drawing it as a Graphics object instead of as an Image. This is because metafiles are vector graphics; it saves how the picture is drawn and not the final product. To get around this, we used another child class of Image, Imaging::Bitmap, to load the picture and get rid of the vector graphics. Since we were doing this in such a convoluted way, when we displayed the Image the transparent color of the Bitmap was also displayed. We then realized we could just draw the graphics onto a bitmap. This made it so we had scrolling functionality, but didn’t have problems with the transparent color and weren’t creating superfluous files.

The other major problem we had was with the environment. We were using the Windows::Forms designer to facilitate the creation of our GUI. After our project started to get fairly sizable, the designer would not load, giving as a reason that it had a parser error; we had declared and implemented some variables in the designer created code that had apparently caused this problem. We fixed this by going through the code line by line and commenting certain parts out to see where the problems were occurring.

Conclusion

The environment we used assured the algorithm was .NET 2.0 compatible. The algorithm was only a few lines of code so it was under the few KB limit. The simulation was within acceptable standards. The client was impressed with the program’s functionality and its presentation. We were able to design and implement a simulation for geophone levitation. With this simulation we were able to develop an algorithm determining whether a levitation test passed or failed. Future work includes implementing hydrophones. Another area of future study is including more responses to different types of geophone failure.