

CS 109 – C/C ++ Programming for Engineers w. MatLab – Spring 2012

Homework Assignment 2 – First Draft

Sidescan Sonar Distances in Fresh and Salt Water

Due: Thursday February 2nd by 8:00 p.m., via Blackboard. Optional hard copy may be turned in during lab. (Note: no late submissions will be accepted for this assignment.)

Overall Assignment

For this assignment you are to write a program that generates tables of distances from the boat as a function of the round-trip echo times of a sidescan sonar ping, for a range of times in either fresh or salt water. This assignment will make heavy use of control structures, for generating the results and for getting and checking user input.

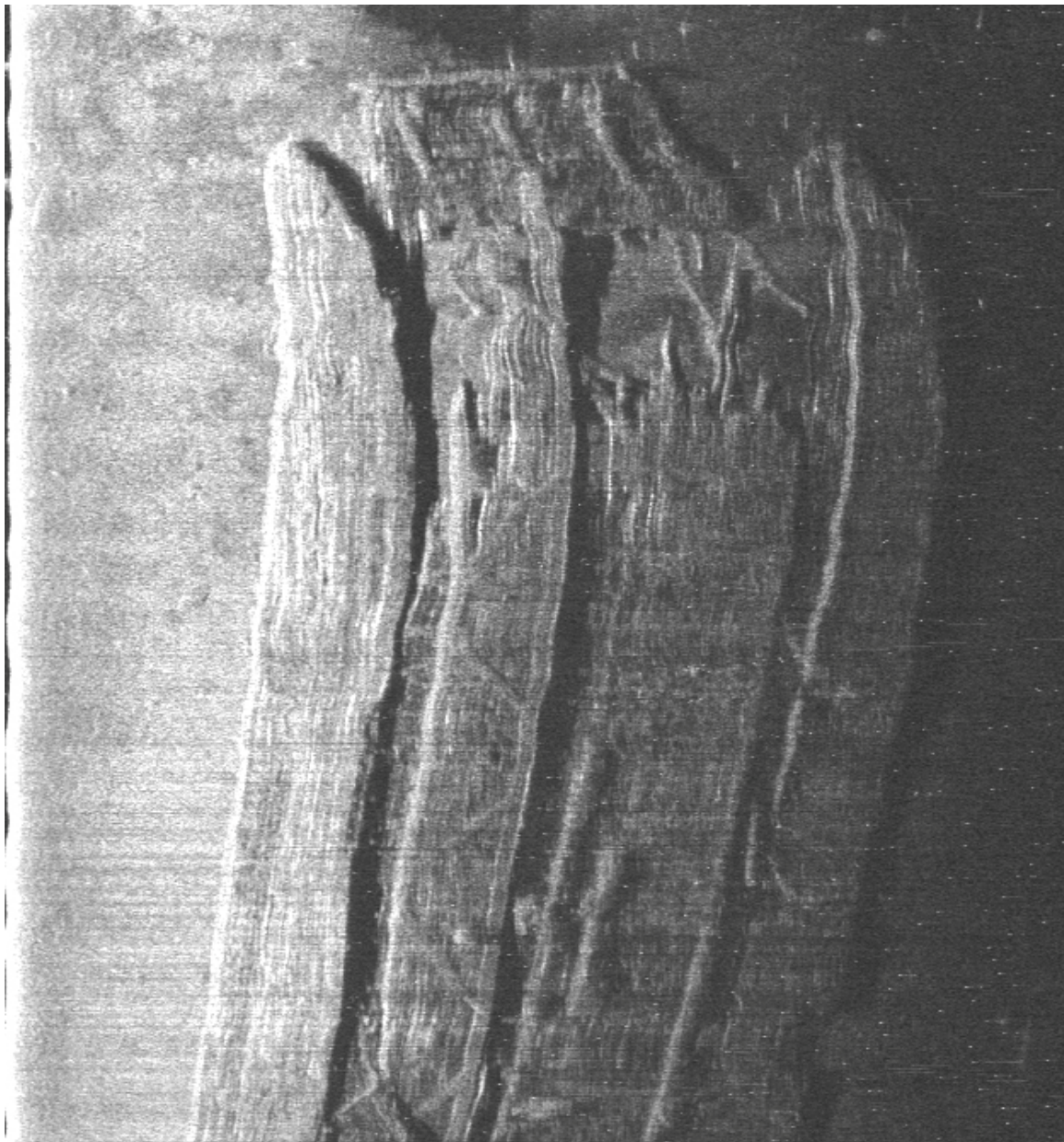


Figure 1: Sidescan sonar image of the Railroad Car Ferry Number 2 in southern Lake Michigan. Note the echo signal strengths are stronger (brighter) on the left side of the image, closer to the boat.

Background: Speed of Sound in Water Calculations

Recall the formula used to predict the speed of sound in fresh water given in HW1 as:

$$c = C_0 + C_1 T + C_2 T^2 + C_3 T^3 + C_4 T^4 + C_5 T^5 \quad (1)$$

where:

- c is the speed of sound in fresh water, in m/s,
- T is the temperature of the water, in degrees C in the range 0 to 95, and
- C_0 to C_5 are polynomial coefficients, reported as:

Coefficient	Value
C_0	1402.388
C_1	5.03711
C_2	-0.0580852
C_3	$0.3342 * 10^{-3}$
C_4	$-0.1478 * 10^{-5}$
C_5	$0.315 * 10^{-8}$

For salt water one must also take into account salinity (and normally depth, which we will ignore for this assignment.) We will use the estimation equation proposed by Leroy in 1969:

$$c = C_0 + C_1 (T - 10) + C_2 (T - 10)^2 + C_3 (T - 18)^2 + C_4 (S - 35) + C_5 (T - 18)(S - 35) \quad (2)$$

where:

- c is the speed of sound in salt water, in m/s,
- T is the temperature of the water, in degrees C in the range -2 to 23,
- S is the salinity in ppt, (parts per trillion), in the range 30 to 40,
- (the depth term, $D / 61$ is omitted for this assignment), and
- C_0 to C_5 are function coefficients, reported as:

Coefficient	Value
C_0	1492.9
C_1	3.0
C_2	-0.006
C_3	-0.04
C_4	1.2
C_5	-0.01

Background: Sidescan Sonar Distance Calculations

As shown in Figure 2 below, sidescan sonar works by emitting a sonar ping and then listening for the strength of the echo over time. Initially there won't be any echo, until the ping has a chance to travel to the bottom of the lake and back again. The time required to get the first echo back determines the depth under the boat, as you calculated in your first homework assignment. (Labeled as point $2h/c$ in the chart below. This is actually the depth under the sensor, but for this assignment we will assume the sensor is mounted directly onto the boat, so the boat-to-sensor distance is zero.) Then as time progresses, echoes come back from further and further distances from the boat, as shown below.

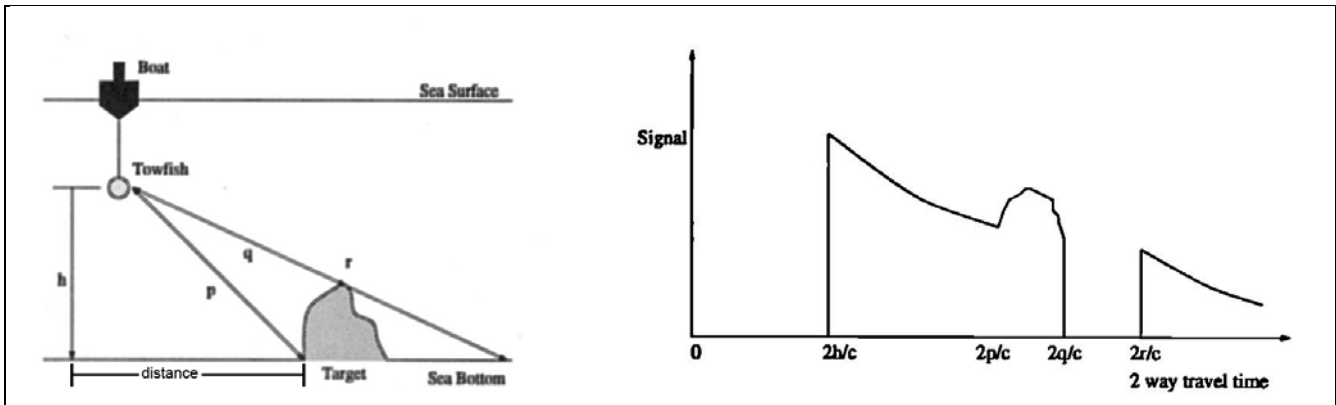


Figure 2 - Sidescan sonar schematic, corresponding to one line of a sidescan sonar image.

Given the height of the sensor above the sea floor, h , and the straight-line echo distance of a return signal, p , the distance away from the boat of the reflected surface can be calculated as:

$$distance = \sqrt{p^2 - h^2} \quad (3)$$

where distance, p , and h are defined as shown in Figure 2 above.

Program Details

For this assignment you are to write a computer program that calculates the speed of sound in either fresh or salt water, and also gives users the choice of calculating either the depth under the boat or a table of distances based on a range of echo times.

- Your program should first print out your name and ACCC account name (e.g. astudent), and explain to the user what the program does.
- Next use a **switch** block to give the user four choices: (1) A single depth in fresh water, (2) a table of distances in fresh water, (3) a single depth in salt water, and (4) a table of distances in salt water. You may also want to give the user options for (H) Help and (Q) Quit.
 - Within the switch block the only thing your program should do is to set the value of two boolean variables, **fresh** and **table**. For example, if the user wants a single value in fresh water, then **fresh** would be set to **true** and **table** to **false**.
- After the switch has completed, your program needs to ask the user for the necessary input, which will vary depending on the values of **fresh** and **table**. For tabulated results, your

program should ask for the minimum value, maximum value, and the number of values desired in the table. (See sample outputs below.)

- For this (and all future) assignment(s), you need to check the validity of user input. You can assume that the user enters a number when you ask for a number, (i.e. you don't have to check for non-numeric input), but you need to check the values to verify that they are within a valid range. If the user enters a bad number, then you should loop back and ask again, and keep asking until they enter a valid number.
- Finally your program needs to perform the necessary calculations and report the result. For the tables you will need to report each value as it is calculated, since we have not covered arrays yet. Your output should be properly formatted, with the columns of tables lined up nicely. (You should report the depth under the boat in every case. If the user requests a table of distances, that should be *in addition to* the depth under the boat.)

Evolutionary Development:

When developing a large complex program it is best **not** to try and write the whole program all at once. A better approach is to develop the program incrementally, starting with a simple program having limited functionality, and gradually adding additional features one by one. You should test the program and save a copy after each step is completed, so that if the next step of development breaks the program you can always back up to a version that was working, and to make sure you have something to hand in that at least compiles and runs.

- Since you already have a program that calculates the depth under the boat at a single echo time in fresh water, it is recommended you start with (a copy of) that.
- Without adding any control logic you can enhance your program to calculate the speed of sound in salt water after calculating the speed of sound in fresh water. You will now have a program that performs two of the four types of calculations, one after the other.
- The easiest control logic to add is the user input checking. The course notes on looping constructs have some good examples.
- You can then add the Boolean variables **fresh** and **table**. At this point you can just assign them initial values which will change what calculations are performed when you edit those initial values. Modify your program so that it performs **either** the freshwater **or** saltwater calculations, but not both, depending on the value of the Boolean variable **fresh**.
- Next add the code for a table of fresh water distances. If **table** is true, you should ask the user for the minimum echo time to calculate for, the number of time samples to calculate, and the change in echo time between samples. For example the user may ask for 20 samples starting at 10 milliseconds and increasing in steps of 1 millisecond. The results should be a table with two columns, showing the round-trip echo times in the first column and the distance away from the boat in the second column. (Use the minimum round-trip echo time to determine the depth under the boat.) See the course notes on looping.
- Next add a table of distances in salt water. This table will be a function of two variables – round-trip echo time and salinity. Just to mix things up a bit, the user should specify their salinity range as the minimum salinity to calculate for, the maximum salinity to calculate for, and the number of salinity values to calculate. Again, check the course notes on looping.

- Next add the code to present the user with a menu of choices, and use a switch block to set the values of the **fresh** and **table** based on their menu choice.
- When all this is done and working, then you can work on any final polishing, refinements, and optional enhancements, such as putting the whole thing inside an infinite while loop to repeat multiple problems, with a “quit” choice on the menu to get out of the loop.
- By saving your work (as a backup copy) whenever you accomplish a milestone, you will have something to hand in if your next set of changes breaks the program.

Special Notes:

- The special notes from HW1 continue to apply to this assignment.
- See the course notes on looping, and how to increment floating point variables in a loop.
- All user input needs to be checked for validity. Note that the valid range is different for salt water temperatures and fresh water temperatures.

Sample Output – Table for Fresh Water

- Two columns, one for round trip echo time and one for distance.
- N rows, (not counting headers), one for each time calculated.
- Properly labeled, including units
- In the following example the user has requested 20 rows of output, starting from 10 milliseconds in 1 millisecond intervals.
- The following numbers have not been checked for accuracy.

Distance from Boat in Fresh Water at a depth of xx.xxxx feet.

Echo time milliseconds	Distance feet
10.00	xx.xxxx
11.00	xx.xxxx
12.00	xx.xxxx
13.00	xx.xxxx
14.00	xx.xxxx
15.00	xx.xxxx
16.00	xx.xxxx
17.00	xx.xxxx
18.00	xx.xxxx
19.00	xx.xxxx
20.00	xx.xxxx
21.00	xx.xxxx
22.00	xx.xxxx
23.00	xx.xxxx
24.00	xx.xxxx
25.00	xx.xxxx
26.00	xx.xxxx
27.00	xx.xxxx
28.00	xx.xxxx
29.00	xx.xxxx

What to Hand In:

- Your code, **including a user documentation file**, should be handed in electronically using Blackboard.
- The intended audience for the documentation file is a general end user, who might want to use this program to perform some work. They do not get to see the inner workings of the code, and have not read the homework assignment. You can assume, however, that they are familiar with the problem domain (e.g. sonar.)
- A secondary purpose of the documentation file is to make it as easy as possible for the grader to understand your program. If there is anything special the grader should know about your program, be sure to document it in the documentation file. In particular, if you do any of the optional enhancements, then you need to document what they are and anything special the TA needs to do to run your program and understand the results.
- If there are problems that you know your program cannot handle, it is best to document them as well, rather than have the TA wonder what is wrong with your program.
- Make sure that your name appears at the beginning of each of your files. Your program should also print this information when it runs.

Optional Enhancements:

It is course policy that students may go above and beyond what is called for in the base assignment if they wish. These optional enhancements will not raise any student's score above 100 for any given assignment, but they may make up for points lost due to other reasons.

- Ask the user if they would like to solve additional problems, and if so, repeat until they indicate they are done. (Or employ an infinite loop, and break out of it when they select the "quit" menu choice.)
- Echo the input temperature in degrees F as well as degrees C, and report the speed and distance using both the metric system (meters) and the English system (feet.)
- Adjust the width of the columns (and precisions) in the salt water table dynamically, depending on how many columns the user has requested.
- Other enhancements that you think of – Check with TA for acceptability.