Loam: Soil Analysis

By

Christopher Diaz

Approved by:

Dr. Tanmay Bhowmik (Major Professor)
Dr. Stephen Torri
Dr. Shuvashis Dey
Dr. TJ Jankun-Kelly (Graduate Coordinator)
Dr. Jason M. Keith (Dean, Bagley College of Engineering)

A Final Project Report
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science
in the Department of Computer Science

Mississippi State, Mississippi

May 2023

Name: Christopher Diaz

Date of Degree: May 12, 2023

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: Dr. Tanmay Bhowmik

Title of Study: Loam: Soil Analysis

Pages in Study: 20

Candidate for Degree of Master of Science

Loam: Soil Analysis, also referred to as Loam, is a software application developed to assist in analyzing and visualizing data obtained from a sensor. This sensor is placed into soil and then gives an amplitude value which can be input into the software along with the depth of the sensor. Loam then gives a two-dimensional and three-dimensional graphical visualization of this data in two different plots while calculating the volumetric moisture content (VMC). The VMC is then used to report the moisture level quality in the tested soil area.

ACKNOWLEDGEMENTS

Thank you to Dr. Dey for working with me and supplying various datasets for this project. Thank you also to Dr. Bhowmik for supporting this program, and originally organizing the initial project meeting. I appreciate all project committee members for their time and any support they have provided.

TABLE OF CONTENTS

B. CODE 19

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRO TO LOAM: SOIL ANALYSIS

1.1 About

As mentioned previously, Loam: Soil Analysis is a software application created to visualize and analyze data reported from a soil sensor. The data reported from the sensor consists of an amplitude value given in decibels and a sensor depth value given in centimeters. These values are input into Loam using the textboxes, labeled Depth(cm) and Amplitude(dB), on the left side of the window. These values are then used to create a data point on the Amplitude vs. Depth plot and the Amplitude vs. Depth vs. Dielectric Constant plot. To create the point on the three-dimensional plot, the dielectric constant is calculated using bi-linear interpolation. This will be discussed more in-depth in a later chapter. The dielectric constant is then used to calculate the volumetric moisture content.

1.2 Features

This section goes more in-depth on the features provided by Loam. These include the two plots previously discussed, and the reporting of the volumetric moisture content.

1.2.1 Amplitude vs. Depth Plot

The Amplitude vs. Depth plot is generated at the start of the application. An example reference graph, seen in Appendix A, and the numerical data used to generate the plot were provided by the client [1]. The plot produced by Loam can be viewed in Figure 1.1. The data in

this plot represents amplitude levels at various depths. Using the text boxes, labeled “Depth (cm)” and “Amplitude (dB),” values from the sensor can be input into Loam. Depth values must be from “0 – 8.15” centimeters while amplitude values must between “-6.0 – 2.0” decibels. Inputs are sanitized to ensure they are within these ranges. These values are then used to plot a new point on the Amplitude vs. Depth plot.

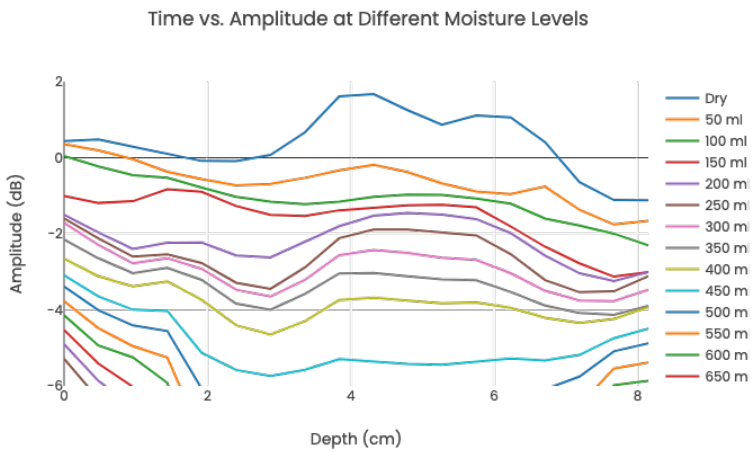

Figure 1.1 Extrapolated “Time vs. Amplitude at different moisture levels” [1]

This data was recorded in uniform sandy soil [1] and contains extrapolated data.

1.2.2 Amplitude vs. Depth vs. Dielectric Constant Plot

Like the previous plot, the Amplitude vs. Depth vs. Dielectric Constant is generated at the start of the application; however, this plot is originally hidden. To view this plot, the user must click one of the arrows on either side of the Amplitude vs. Depth plot or the “unselected” gray dot underneath the same plot. These buttons will remain when the new plot is shown and can be used to switch between the two plots.

A reference graph and numerical data were provided by the client for this plot. These can be found in Appendix A [1]. The graph produced in Loam can be seen in Figure 1.2.

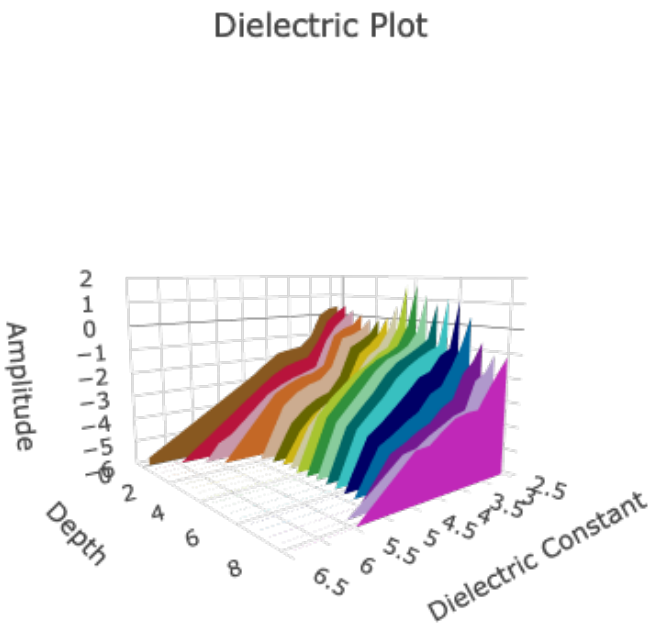

Figure 1.2 Extrapolated “Calibration Curve” [1]

This data is also based on the data from the previous plot [1] and contains extrapolated data as well

When amplitude and depth data are entered, a new point is also generated on this plot. The dielectric constant is calculated based on the given amplitude and depth values to generate this point. The details of this calculation will be discussed in a later section.

1.2.3 Volumetric Moisture Content and Recommendations

Once the dielectric constant for the given input is determined, the volumetric moisture content is calculated and displayed in the “water droplet” on the top left side of the window. The circle around the droplet will also change depending on the volumetric constant. The circle contains an indicator that will move proportionally to the volumetric constant. For example, if the volumetric moisture constant is 0.25, the indicator will move to cover a quarter of the height of the containing circle. The indicator will be blue if the volumetric moisture content is at an

acceptable level and red if the soil contains too much or too little moisture. Below, is a figure that shows the output of the moisture indicator when the acceptable volumetric moisture content is calculated.

Figure 1.3 Example of Acceptable Moisture Indicator Output

CHAPTER II

CALCULATIONS AND ASSUMPTIONS

2.1 Extrapolation of Data

As requested by the client, the data was extrapolated to include data up to a volumetric moisture content value of 0.4 [1]. The following equation for linear extrapolation was used to extrapolate each data set [2]:

$$y = y_1 + \frac{x - x_1}{x_2 - x_1}(y_2 - y_1) \quad \text{(Eq. 2.1)}$$

The first dataset to be extrapolated was the volumetric moisture content and dielectric constants. This resulted in an extrapolated dataset to include dielectric constants corresponding to the moisture content of 0.4. Next, amplitude values were extrapolated from the extrapolated dielectric constant values.

2.2 Calculating the Dielectric Constant

Once the user has entered an amplitude value and depth value, the corresponding dielectric constant is calculated using bi-linear interpolation and data from the "Calibration Curve" table in Appendix A [1]. Bi-linear interpolation consists of using a series of linear interpolations to find a point between two lines [2]. The equation used for linear interpolation is given in Equation 2.1 below.

$$y = \frac{y_0\,(x_1 - x) + y_1\,(x - x_0)}{(x_1 - x_0)} \quad \text{(Eq. 2.2)}$$

2.3 Calculating the Volumetric Moisture Content

After the dielectric constant is calculated, the volumetric moisture content can be calculated using Equation 1 along with the data from Table 2.1 [1], which includes extrapolated data. The two dielectric constants, and corresponding volumetric constants, that surround the calculated dielectric constant are selected to be used in the equation. For example, if the calculated dielectric constant, x, is 3.5, the dielectric constants used in the equation for x_0 and x_1 will be 3.42 and 3.67 respectively. Because these values correspond with VMC values of 0.1 and 0.125, these will be y_0 and y_1. The resulting y value will then give the estimated VMC value at a dielectric constant of 3.5.

Table 2.1 VMC vs. Dielectric Constant

Volumetric Moisture Content	Dielectric Constant
0	2.53
0.025	2.74
0.05	2.96
0.075	3.18
0.1	3.42
0.125	3.67
0.15	3.92
0.175	4.18
0.2	4.45
0.225	4.73
0.25	4.92
0.275	5.17
0.3	5.41
0.325	5.66
0.35	5.90
0.375	6.15
0.4	6.39

2.4 Determining the Upper and Lower Bounds of Acceptable VMC Values

To give feedback on the volumetric moisture content, an upper bound and lower bound for acceptable VMC values must be determined. These represent too much or too little moisture in the tested area, respectively.

The upper bound is given by the field capacity (FC) which is the threshold at which the force of gravity drains water between larger openings in the grains of soil [3]. This means that the FC is directly related to the type of soil being used. Moisture above this level provides no benefit since the additional water will drain deep enough to pass the depth of the roots meaning it is lost to drainage [3]. An example of these concepts can be seen in Figure 2.1 below.

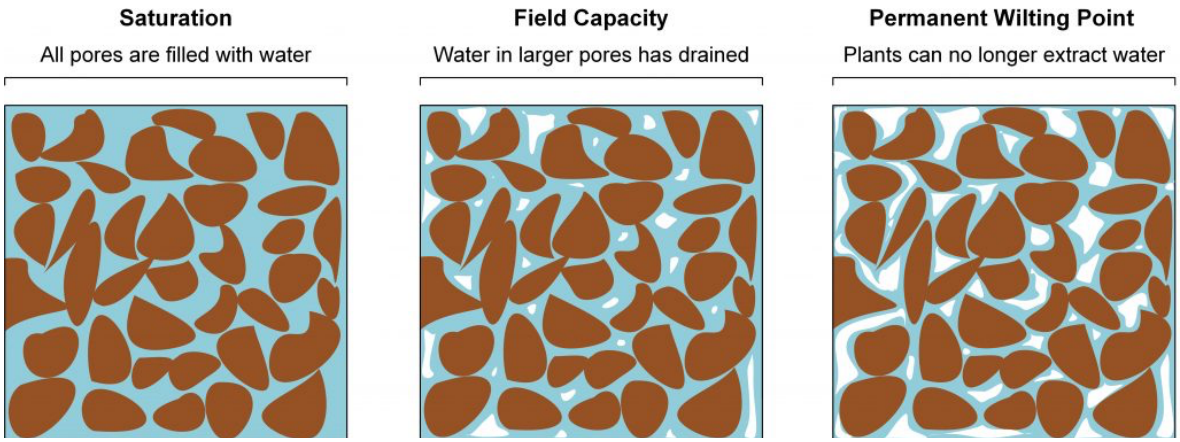

Figure 2.1 "Soil water content at saturation, field capacity, and permanent wilting point thresholds." [3]

While the permanent wilting point (PWP), seen in the figure above, could technically be considered a lower bound for acceptable VMC values, it would be dangerous to do so since at this point, the grains of soil are holding moisture with force greater than a plant can overcome to extract it [3]. This means that a plant will die if it remains at this amount of moisture for too long. Therefore, it could be viewed as a point at which planting should be redone, but this was not

applied to the scope of this project. The values used for FC, PWP, and TAW are found in Table 2.2 below. Because the data provided by the client was completed in uniform sandy soil, the values for sand were used for all calculations.

Table 2.2 "Typical soil water thresholds for different soil textures sampled across the U.S." [3

Soil Texture	FC (%)	PWP (%)	TAW (%)
Sand	10	4	6
Loamy Sand	16	7	9
Sandy Loam	21	9	12
Loam	27	12	15
Silt Loam	30	15	15
Sandy Clay Loam	36	16	20
Sandy Clay	32	18	14
Clay Loam	29	18	11
Silty Clay Loam	28	15	13
Silty Clay	40	20	20
Clay	40	22	18

Because the lower boundary should be an acceptable value to promote plant growth, total available water (TAW) and management allowable depletion (MAD) values are used. TAW is the theoretical amount of water available to a plant which is "estimated as the difference between soil water content at FC and PWP" [3]. This means that TAW is also directly correlated to the type of soil being used in the tested area which is why this value can also be found in the previous table. MAD, however, is dependent on the crop being grown, because it is the amount of TAW that can be depleted before any plant stress or growth reduction occurs [3]. For this project, MAD values corresponding to soybean, rice, and potatoes were used. These values were obtained from the following table.

Table 2.3 "Management allowable depletion (MAD) and maximum root zone depths for selected crops." [3]

Type of crop	MAD*	Maximum root depth (ft.)
Cotton	**0.65**	**3.3-5.6**
Barley and Oats	**0.55**	**3.3-4.5**
Maize	**0.50-0.55**	**2.6-6.0**
Sorghum	**0.50 – 0.55**	**3.3 – 6.6**
Rice	**0.2**	**1.6 – 3.3**
Beans	**0.45**	**1.6 – 4.3**
Soybeans	**0.5**	**2.0 – 4.1**
Alfalfa	**0.50 – 0.60**	**3.3 – 9.9**
Cool season – Turf grass	**0.4**	**1.6 – 2.2**
Warm season – Turf grass	**0.5**	**1.6 – 2.2**
Citrus	**0.5**	**2.6 – 5.0**
Walnut orchard	**0.5**	**5.6 – 8.0**
Carrots	**0.35**	**1.5 – 3.3**
Cantaloupes/watermelons	**0.40 – 0.45**	**2.6 – 5.0**
Lettuce	**0.3**	**1.0 – 1.6**
Onions	**0.3**	**2.0 – 3.0**
Potatoes	**0.65**	**1.0 – 2.0**
Sweet peppers	**0.3**	**1.0 – 2.0**
Cucumbers	**0.5**	**2.0-4.0**

After the MAD value is determined based on the plant being used and the TAW, FC, and PWP values are found based on the type of soil, an expression for the proper lower bound of VMC can be determined. This expression is given in Equation 3.1 below.

$$\text{lower\_bound} = \text{FC} - (\text{MAD} * \text{TAW}) \qquad \text{(Eq. 3.1) [3]}$$

Finally, once the upper bound and lower bound are found, any value within this range is defined as an acceptable VMC value. However, values above the upper bound are over-watered while values under the lower bound are under-watered. This value is displayed on the moisture indicator with the corresponding recommendation for irrigation.

CHAPTER III

FRAMEWORKS AND LIBRARIES USED

3.1 ElectronJS

To complete this project, ElectronJS was used to develop the application in JavaScript, HTML, and CSS. Electron JS is a runtime framework that allows for the creation of a desktop application in these languages [4]. There are a few main reasons for selecting this framework that are discussed in the following subsections.

3.1.1 Cross-platform Applications

One of the contributing factors in selecting ElectronJS is its ability to develop cross-platform applications [4]. Because the user's and future user's operating system of choice was not known, the capability of creating a cross-platform application became desirable. However, there are many more options for developing cross-platform applications which means this was not the sole reason for selecting ElectronJS.

Packages are developed using Electron Forge, which has the capability of producing various forms of packages. This includes dmg files that are compatible with MacOS, squirrel.windows which are compatible with Windows operating systems and deb files which are compatible with Linux.

3.1.2 Compatibility with Plotly

Because two of the three features of this project are developing plots, it was important to find a quality graphing library. The JavaScript language was ultimately chosen to take advantage of the Plotly graphing library, which will be discussed more in-depth later. While Plotly is available in a few different languages, the only other available language that is also well-known is

Python. Considering the future development of this application as another person's project, it was important to select a language that is well-known which limited the options to Python and JavaScript. The selection between these two was decided with the same logic as my final point for choosing ElectronJS.

3.1.3 Personal Growth

The final reason for selecting ElectronJS was for my personal growth as a developer. Choosing this framework provided a challenge since I was not previously familiar with the technology, and it seemed extremely interesting. This was also a framework that was mentioned multiple times in my job search which led to a desire to learn it.

3.2 Plotly

As mentioned previously, graphing is a major aspect of this project. While it was technically possible to create these graphs from scratch, the time constraints were a concern, and the amount of work to develop what a library can already accomplish appeared unnecessary in this context. This led to the search for a graphing library which resulted in the selection of the Plotly graphing library.

Another main reason for selecting Plotly was its capability of creating three-dimensional graphs [5]. While there are many options for plotting libraries, many are only capable of displaying two-dimensional plots. Plotly appeared to be the best option given the requirement of this project to display both two-dimensional and three-dimensional plots.

REFERENCES

[1] S. Dey, private communication, August 2022.

[2] X-engineer, “Linear Interpolation Extrapolation Calculator,” https://x-engineer.org/linear-interpolation-extrapolation-calculator/

[3] H. R. Kang, “Three-Dimensional Lookup Table with Interpolation,” in Computation Color Technology. SPIE Press, 2006, ch. 9, pp. 151 – 159.

[4] S. Datta, S. Taghvaeian, J. Stivers, “Understanding Soil Water Content and Thresholds for Irrigation Management,” Oklahoma State University. https://extension.okstate.edu/fact-sheets/understanding-soil-water-content-and-thresholds-for-irrigation-management.html (accessed Feb. 2, 2023).

[5] ElectronJS, “ElectronJS Docs,” https://www.electronjs.org/docs/latest/ (accessed Aug 2022).

[6] Plotly, “Plotly JavaScript Open Source Graphing Library,” https://plotly.com/javascript/ (accessed Aug 2022).

APPENDIX A

DATA TABLES

A.1 Amplitude vs. Depth Data

The following table contains the numerical data on amplitude levels at various sensor depth values. The client provided this data, and it is used to generate the amplitude vs depth plot [1].

Table A.1 "Time vs Amplitude at different moisture levels" [1]

Depth	Time	2.53	Amplitude	Amplitude	Amplitude	Amplitude	Amplitude	Amplitude	Amplitude	Amplitude	Amplitude
cm	ns	dB	dB	dB	dB	dB	dB	dB	dB	dB	dB
		Dry (er=2.53)	50 ml (er=2.74)	100 ml (er=2.96)	150 ml (er=3.18)	200 ml (er=3.42)	250 ml (er=3.67)	300 ml (er=3.92)	350 ml (er=4.18)	400 ml (er=4.45)	450ml (er=4.73)
0	16.15	0.43653	0.353297	0.0435421	-1.005547	-1.500341	-1.594382	-1.710513	-2.149558	-2.655376	-3.089209
0.48	16.2	0.48389624152076	0.190211186093047	-0.229938272336168	-1.18868904002001	-1.96511644332166	-2.11995022411206	-2.29615294317159	-2.645414016008	-3.11855213306653	-3.64977925312656
0.96	16.25	0.285882980590295	-0.039246761380690 3	-0.459375346723362	-1.14356549194597	-2.39855063101551	-2.60034892946473	-2.77802094687344	-3.03527442421211	-3.38481151175588	-3.99542033866934
1.44	16.3	0.100441990195097	-0.371107429214607	-0.525096044972486	-0.830189112356178	-2.235384783 89195	-2.539371400 70035	-2.642068817 2086	-2.897276759 37969	-3.250193817 90895	-4.028500404 2021
1.92	16.35	-0.084999000 2001002	-0.560742096548274	-0.78363	-0.898005435217609	-2.231341496 54827	-2.767763173 58679	-2.926125930 66533	-3.214559478 73937	-3.737626464 23212	-5.129620544 27214
2.4	16.4	-0.088284188 1940968	-0.723827910 455228	-1.031707226 51326	-1.270394249 92496	-2.567562030 01501	-3.287524014007	-3.469934449 32466	-3.836718442 22111	-4.403869965 98299	-5.582581329 16458
2.88	16.45	0.072012261 1305655	-0.686832835 917959	-1.155744772 43622	-1.504409078 83942	-2.619968765 98299	-3.447930022 01101	-3.653939676 33817	-3.998762306 15308	-4.650288795 3977	-5.737739167 08354
3.36	16.5	0.666052043 921961	-0.529436062 031016	-1.221499616 05803	-1.534135724 26213	-2.204637518 75938	-2.887517495 74787	-3.214709718 95948	-3.582430368 18409	-4.299185992 9965	-5.579412991 49575
3.84	16.55	1.612115401 70085	-0.332216008 004002	-1.15798	-1.381517357 57879	-1.801667320 56028	-2.11514	-2.561127	-3.042649	-3.739044	-5.299392680 84042
4.32	16.6	1.672512661 03051	-0.188093660 830415	-1.030938140 02001	-1.310295453 12656	-1.524779822 41121	-1.888647500 75038	-2.433069490 64532	-3.030207651 82591	-3.680736885 44272	-5.354448687 84392
4.79	16.65	1.251341598 09905	-0.376313699 849925	-0.970976703 601801	-1.251283018 009	-1.449073910 95548	-1.887042801 4007	-2.506274483 54177	-3.121426447 22361	-3.758313183 5918	-5.419514877 93897
5.27	16.7	0.867887346 773387	-0.674040127 563782	-0.975459988 394197	-1.237533185 993	-1.496045897 24862	-1.962538789 3947	-2.631768757 07854	-3.203289468 73437	-3.831326170 08504	-5.443185118 55928

Table A.1 (Continued)

5.75	16.75	1.1130989991996	-0.890223648324162	-1.071103397298 65	-1.304685267333 67	-1.612239758079 04	-2.043842161080 54	-2.689286965782 89	-3.222541322661 33	-3.806664090045 02	-5.360601108054 03
6.23	16.8	1.0607234 4032016	-0.954555364682341	-1.207096369334 67	-1.807308421610 81	-1.985543438619 31	-2.534566083041 52	-3.037010682041 02	-3.529513102551 28	-3.950504720360 18	-5.282654367183 59
6.71	16.85	0.4101084 4012006	-0.759104689344672	-1.600131002 8014	-2.340455249224 61	-2.573929372186 09	-3.216933666833 42	-3.497156351675 84	-3.887371453726 86	-4.208331828914 46	-5.329921981490 75
7.19	16.9	-0.639676149574788	-1.369728318159 08	-1.782451251025 51	-2.777961233716 86	-3.038704815507 75	-3.536339769884 94	-3.755988290845 42	-4.083842705352 68	-4.349794490245 12	-5.187276872436 22
7.67	16.95	-1.114279362281 14	-1.758479386193 1	-1.997648921060 53	-3.127966021310 66	-3.241426019509 75	-3.513238091045 52	-3.771195899649 82	-4.132678347173 59	-4.238604225112 56	-4.746828816408 2
8.15	17	-1.122131	-1.66196	-2.30849	-3.006337	-3.007367	-3.118336	-3.473147	-3.898784	-3.935144	-4.498764

A.2 Amplitude vs. Depth vs. Dielectric Constant

The following table contains the numerical data on dielectric constants at different amplitude levels and sensor depth values. The client provided this data, and it is used to generate the Amplitude vs Depth vs Dielectric Constant plot [1].

Table A.2 "Calibration Curve" data [1]

Dielectric Constant	**0**	**0.48**	**0.96**	**1.44**	**1.92**	**2.88**	**2.88**	**3.36**	**3.84**	**4.32**	**4.79**	**5.27**	**5.75**	**6.23**	**6.71**	**7.19**	**7.67**	**8.15**
\g(e\-(r))	dB	dB	dB	dB	dB	dB	dB	dB	dB	dB	dB	dB	dB	dB	dB	dB	dB	dB
	Depth =0	Depth =0.48	Depth =0.96	Depth =1.44	Depth =1.92	Depth =2.4	Depth =2.88	Depth =3.36	Depth =3.84	Depth =4.32	Depth =4.79	Depth =5.27	Depth =5.75	Depth =6.23	Depth =6.71	Depth =7.19	Depth =7.67	Depth =8.15
2.53	0.43653	0.483896241520 76	0.285882 9805902 95	0.10044 1990195 097	-0.084999 0002001 002	0.18828 4188194 097	0.072012 2611305 655	0.66605 2043921 961	1.61211 5401700 85	1.67251 2661030 51	1.25134 1598099 05	0.86788 7346773 387	1.11309 8999199 6	1.06072 3440320 16	0.41010 8440120 06	-0.63967 6149574 788	-1.11427 936228 114	-1.1221 31
2.74	0.35329 7	0.19021 1186093 047	-0.039246 7613806 903	-0.37110 7429214 607	-0.560742 0965482 74	-0.72382 7910455 228	-0.686832 8359179 59	-0.52943 6062031 016	-0.33221 6008004 002	-0.18809 3660830 415	-0.37631 3699849 925	-0.67404 0127563 782	-0.89022 3648324 162	-0.95455 5364682 341	-0.75910 4689344 672	-1.36972 8318159 08	-1.75847 938619 31	-1.6619 6
2.96	0.04354 21	-0.22993 8272336 168	-0.459375 3467233 62	-0.52509 6044972 486	-0.78363	-1.03170 7226513 26	-1.155744 7724362 2	-1.22149 9616058 03	-1.15798	-1.03093 8140020 01	-0.97097 6703601 801	-0.97545 9988394 197	-1.07110 3397298 65	-1.20709 6369334 67	-1.60013 1002801 4	-1.78245 1251025 51	-1.99764 892106 053	-2.3084 9
3.18	-1.00554 7	-1.18868 9040020 01	-1.143565 4919459 7	-0.83018 9112356 178	-0.898005 4352176 09	-1.27039 4249924 96	-1.504409 0788394 2	-1.53413 5724262 13	-1.38151 7357578 79	-1.31029 5453126 56	-1.25128 3018009	-1.23753 3185993	-1.30468 5267333 67	-1.80730 8421610 81	-2.34045 5249224 61	-2.77796 1233716 86	-3.12796 602131 066	-3.0063 37

Table A.2 (Continued)

3.42	-1.500341	-1.965116443321 66	-2.398550631015 51	-2.235384783891 95	-2.231341496548 27	-2.367562030015 01	-2.619968765982 99	-2.204637518759 38	-1.801667320560 28	-1.524779822411 21	-1.449073910955 48	-1.496045897248 62	-1.612239758079 04	-1.985543438619 31	-2.573929372186 09	-3.038704815507 75	-3.241426019509 75	-3.007367
3.67	-1.594382	-2.119950224112 06	-2.600348929464 73	-2.539371400700 35	-2.767763173586 79	-2.937524014007	-3.447930022011 01	-2.887517495747 87	-2.11514	-1.888647500750 38	-1.887042801400 7	-1.962538789394 7	-2.043842161080 54	-2.534566083041 52	-3.216933666833 42	-3.536339769884 94	-3.513238091045 52	-3.118336
3.92	-1.710513	-2.296152943171 59	-2.778020946873 44	-2.642068817208 6	-2.926125930665 33	-3.219934449324 66	-3.653939676338 17	-3.214709718959 48	-2.56112 7	-2.433069490645 32	-2.506274483541 77	-2.631768757078 54	-2.689286965782 89	-3.037010682041 02	-3.497156351675 84	-3.755988290845 42	-3.771195899649 82	-3.473147
4.18	-2.149558	-2.645414016008	-3.035274424212 11	-2.897276759379 69	-3.214559478739 37	-3.736718442221 11	-3.998762306153 08	-3.582430368184 09	-3.042649	-3.030207651825 91	-3.121426447223 61	-3.203289468734 37	-3.222541322661 33	-3.529513102551 28	-3.887371453726 86	-4.083842705352 68	-4.132678347173 59	-3.898784
4.45	-2.655376	-3.118552133066 53	-3.384811511755 88	-3.250193817908 95	-3.737626464232 12	-4.403869965982 99	-4.650288795397 7	-4.299185992996 5	-3.739044	-3.680736885442 72	-3.758313183591 8	-3.831326170085 04	-3.806664090045 02	-3.950504720360 18	-4.208331828914 46	-4.349794490245 12	-4.238604225112 56	-3.935144
4.73	-3.089209	-3.649779253126 56	-3.995420338669 34	-4.028500404202 1	-5.129620544272 14	-5.582581329164 58	-5.737739167083 54	-5.579412991495 75	-5.299392680840 42	-5.354448687843 92	-5.419514877938 97	-5.443185118559 28	-5.360601108054 03	-5.282654367183 59	-5.329921981490 75	-5.187276872436 22	-4.746828816408 2	-4.498764
4.92	-3.383595678571 43	-4.010254798881 58	-4.409762042646 33	-4.556637016329 59	-6.074187955727 87	-6.382421182752 09	-6.475651919298 93	6.448138454763 1	-6.358200714267 84	-6.490181696616 16	-6.546758884817 4	-6.536946547881 08	-6.415058370274 43	-6.186613056099 47	-6.091001013596 09	-5.755568488923 04	-5.091695503358 81	-4.881220428571 43
5.17	-3.770946571428 57	-4.484564727506 6	-4.954948495247 63	-5.251553611234 19	-7.317039812906 45	-7.434842042735 65	-7.446589751161 29	-7.591198274851 71	-7.751369179303 93	-7.984567234474 37	-8.029974683341 66	-7.976106323304 51	-7.802502136353 9	-7.376032383620 38	-7.092420792682 06	-6.503320615879 38	-5.545467459872 78	-5.384452571428 57
5.41	-4.142803428571 43	-4.939902258986 63	-5.251553611234 19	-5.918673542342 61	-8.510177595797 9	-8.445166068319 87	-8.378690069749 15	-8.688535702136 78	-9.088810905738 58	-9.419177350818 26	-9.453861849924 95	-9.357699707711	-9.134448151790 19	-8.517874938040 44	-8.053783780604 59	-7.221162657757 46	-5.981088538126 18	-5.867555428571 43

Table A.2 (Continued)

5.66	-4.530154321428571	-5.414212187611661	-6.023513942346181	-6.613590137247211	-9.753029452976491	-9.497586928303431	-9.349627901611511	-9.831595522225541	-10.481979370774701	-10.913562888676501	-10.937077648449201	-10.796859483134401	-10.521891917869701	-9.707294265561351	-9.055203559690561	-7.968914784713801	-6.434860494640151	-6.370787571428571
5.9	-4.902011178571431	-5.869549719091681	-6.546892936843441	-7.280710068355621	-10.946167235867901	-10.507910953887601	-10.281728220199401	-10.928932949510501	-11.819421097209301	-12.348173005020401	-12.360964815032501	-12.178452867540901	-11.853837933303301	-10.849136819981401	-10.016566547613101	-8.686756826591891	-6.870481572893551	-6.853890428571431
6.15	-5.289362071428571	-6.343859647716701	-7.092079389444741	-7.975626663260221	-12.189019093046501	-11.560331813871201	-11.252666052061701	-12.071992769599101	-13.212589562245401	-13.842558542878601	-13.844180613556701	-12.178452867540901	-13.241281699385401	-12.038556147502301	-11.017986326699101	-9.434508953548221	-7.324253529407521	-7.357122571428571
6.39	-5.661218928571431	-6.799197179196731	-7.615458383941991	-8.642746594368641	-13.382156875938	-12.570655839455401	-12.184766370640961	-13.169330196884101	-14.550031288680	-15.277168659222501	-15.26806778014	-14.999206027370801	-14.573227714821701	-13.180398701922401	-11.979349314621601	-10.152350995426301	-7.759874607660921	-7.840225428571431

APPENDIX B

CODE

main.js

```
const { app, BrowserWindow } = require('electron');

const createWindow = () => {
   const win = new BrowserWindow({
     width: 1000,
     height: 600
   })

   win.loadFile('index.html')
 }
 app.whenReady().then(() => {
   createWindow();

   app.on('activate', () => {
     if (BrowserWindow.getAllWindows().length === 0) {
       createWindow();
     }
   });
 });

 app.on('window-all-closed', () => {
   if (process.platform !== 'darwin') {
     app.quit();
   }
 });Hidden text to allow template to find last page in document
```

input_validation.js

```
function setInputFilter(textbox, inputFilter, errMsg) {
   ["input", "keydown", "keyup", "mousedown", "mouseup", "select", "contextmenu", "drop",
"focusout", "keypress"].forEach(function(event) {
     textbox.addEventListener(event, function(e) {
       if (event === "keypress" && e.key === "Enter") {
          if (document.activeElement === document.getElementById("amplitudeInput")) {
             // console.log(document.activeElement);
             UpdatePlot();
             Update3dPlot();
          } else if (document.activeElement === document.getElementById("depthInput")){
             document.getElementById("amplitudeInput").focus();
             // console.log(document.activeElement);
          }
       }
       if (inputFilter(this.value)) {
         // Accepted value
         if (["keydown", "mousedown", "focusout"].indexOf(e.type) >= 0) {
           this.classList.remove("input-error");
           this.setCustomValidity("");
         }
         this.oldValue = this.value;
         this.oldSelectionStart = this.selectionStart;
         this.oldSelectionEnd = this.selectionEnd;
       } else if (this.hasOwnProperty("oldValue")) {
         // Rejected value - restore the previous one
         this.classList.add("input-error");
         this.setCustomValidity(errMsg);
         this.reportValidity();
         this.value = this.oldValue;
         this.setSelectionRange(this.oldSelectionStart, this.oldSelectionEnd);
       } else {
         // Rejected value - nothing to restore
         this.value = "";
       }
     });
   });
}
```

```
setInputFilter(document.getElementById("depthInput"), function(value) {
   return /^-?\d*[.,]?\d*$/.test(value) && (value === "" || parseFloat(value) <= 8.15);
}, "Must be a value between 0 and 8.15");
setInputFilter(document.getElementById("amplitudeInput"), function(value) {
   return /^-?\d*[.,]?\d*$/.test(value) && ((value === "" || value === "-") || (parseFloat(value) >=
-6.0 && parseFloat(value) <= 2.0));
}, "Must be a value between -6.0 and 2.0");
```

moisture_indicator.js

```
const indicatorElement = document.querySelector(".moisture-indicator");
const MAX_MOISTURE = 0.4;
var veg = "soybean";

function setIndicator(indicator, moisture) {

   moisture = +moisture;

   if (moisture < 0 || moisture > MAX_MOISTURE) {
      return;
   }

   let perc_moisture = 100 - ((moisture / MAX_MOISTURE) * 100);
   let TAW = 0.06;
   let FC = 0.1;
   let PWP = 0.04;
   let MAD = 0.5;

   if (veg == "soybean") {
      MAD = 0.5;
      // console.log(veg);
   } else if (veg == "rice") {
      MAD = 0.2;
      // console.log(veg);
   } else if (veg == "potato") {
      MAD = 0.65;
      // console.log(veg);
   }

   let highMoisture = FC;
   let lowMoisture = FC - (MAD * TAW);
   let replantMoisture = PWP;

   console.log(lowMoisture, moisture, highMoisture);

   if (moisture > highMoisture) {
      indicator.querySelector(".indicator-fill").style.background = '#fc4349';
```

```
            document.getElementById("report").innerHTML = "Land too hydrated. Refrain
from irrigation";
            document.getElementById("report").style.color = '#fc4349';
        } else if (lowMoisture < moisture && moisture <= highMoisture) {
            indicator.querySelector(".indicator-fill").style.background = '#3498db';
            document.getElementById("report").innerHTML = "Land properly irrigated";
            document.getElementById("report").style.color = '#3498db';
        } else if (moisture <= lowMoisture) {
            indicator.querySelector(".indicator-fill").style.background = '#fc4349';
            document.getElementById("report").innerHTML = "Land needs moisture. Irrigate
land as soon as possible";
            document.getElementById("report").style.color = '#fc4349';
        } else {
            indicator.querySelector(".indicator-fill").style.background = '#fc4349';
        }

        // console.log(perc_moisture);
        // document.getElementById("report").innerHTML = moisture;

        indicator.querySelector(".indicator-fill").style.transform =
        `translateY(${
            perc_moisture
        }%)`;
        indicator.querySelector(".indicator-cover p").textContent =
        `${
            moisture
        }`;
    }

    function selectVeg(selected) {
        veg = selected;

        var depth = +document.getElementById('depthInput').value;
        var amplitude = +document.getElementById('amplitudeInput').value;

        if (depth != 0 && amplitude != 0) {
            UpdatePlot();
            Update3dPlot();
```

```
        }

        return;
}
```

navigation.js

```
let plotIndex = 1;
showPlot(plotIndex);

// Next/previous controls
function changePlot(n) {
  showPlot(plotIndex += n);
}

// Thumbnail image controls
function currentPlot(n) {
  showPlot(plotIndex = n);
}

function showPlot(n) {
    let i;
    let plot = document.getElementsByClassName("plots");
    let dots = document.getElementsByClassName("dot");
    if (n > plot.length) {plotIndex = 1}
    if (n < 1) {plotIndex = plot.length}
    for (i = 0; i < plot.length; i++) {
      plot[i].style.display = "none";
    }
    for (i = 0; i < dots.length; i++) {
      dots[i].className = dots[i].className.replace(" active", "");
    }
    // console.log(plot[plotIndex-1].style)
    plot[plotIndex-1].style.display = "block";
    dots[plotIndex-1].className += " active";
}
```

plot.js

```
TESTER = document.getElementById('ampVsTimePlot');

const DEPTHS =
[0,0.48,0.96,1.44,1.92,2.4,2.88,3.36,3.84,4.32,4.79,5.27,5.75,6.23,6.71,7.19,7.67,8.15];

const AMPLITUDES = [[0.43653,0.48389624152076, 0.285882980590295,
0.100441990195097, -0.0849990002001002, -0.0882841881940968,
   0.0720122611305655, 0.666052043921961, 1.61211540170085, 1.67251266103051,
1.25134159809905, 0.867887346773387,
   1.1130989991996, 1.06072344032016, 0.41010844012006, -0.639676149574788, -
1.11427936228114, -1.122131],
   [0.353297, 0.190211186093047, -0.0392467613806903, -0.371107429214607, -
0.560742096548274, -0.723827910455228, -0.686832835917959,
   -0.529436062031016, -0.332216008004002, -0.188093660830415, -0.376313699849925, -
0.674040127563782, -0.890223648324162,
   -0.954555364682341, -0.759104689344672, -1.36972831815908, -1.7584793861931, -
1.66196],
   [0.0435421, -0.229938272336168, -0.459375346723362, -0.525096044972486, -0.78363, -
1.03170722651326, -1.15574477243622,
   -1.22149961605803, -1.15798, -1.03093814002001, -0.970976703601801, -
0.975459988394197, -1.07110339729865, -1.20709636933467,
   -1.6001310028014, -1.78245125102551, -1.99764892106053, -2.30849],
   [-1.005547, -1.18868904002001, -1.14356549194597, -0.830189112356178, -
0.898005435217609, -1.27039424992496, -1.50440907883942,
   -1.53413572426213, -1.38151735757879, -1.31029545312656, -1.251283018009, -
1.237533185993, -1.30468526733367, -1.80730842161081,
   -2.34045524922461, -2.77796123371686, -3.12796602131066, -3.006337],
   [-1.500341, -1.96511644332166, -2.39855063101551, -2.23538478389195, -
2.23134149654827, -2.56756203001501, -2.61996876598299,
   -2.20463751875938, -1.80166732056028, -1.52477982241121, -1.44907391095548, -
1.49604589724862, -1.61223975807904,
   -1.98554343861931, -2.57392937218609, -3.03870481550775, -3.24142601950975, -
3.007367],
   [-1.594382, -2.11995022411206, -2.60034892946473, -2.53937140070035, -
2.76776317358679, -3.287524014007, -3.44793002201101,
```

-2.88751749574787, -2.11514, -1.88864750075038, -1.8870428014007, -1.9625387893947, -2.04384216108054, -2.53456608304152,
-3.21693366683342, -3.53633976988494, -3.51323809104552, -3.118336],
[-1.710513, -2.29615294317159, -2.77802094687344, -2.6420688172086, -2.92612593066533, -3.46993444932466, -3.65393967633817,
-3.21470971895948, -2.561127, -2.43306949064532, -2.50627448354177, -2.63176875707854, -2.68928696578289, -3.03701068204102,
-3.49715635167584, -3.75598829084542, -3.77119589964982, -3.473147],
[-2.149558, -2.645414016008, -3.03527442421211, -2.89727675937969, -3.21455947873937, -3.83671844222111, -3.99876230615308,
-3.58243036818409, -3.042649, -3.03020765182591, -3.12142644722361, -3.20328946873437, -3.22254132266133, -3.52951310255128,
-3.88737145372686, -4.08384270535268, -4.13267834717359, -3.898784],
[-2.655376, -3.11855213306653, -3.38481151175588, -3.25019381790895, -3.73762646423212, -4.40386996598299, -4.6502887953977,
-4.2991859929965, -3.739044, -3.68073688544272, -3.7583131835918, -3.83132617008504, -3.80666409004502, -3.95050472036018,
-4.20833182891446, -4.34979449024512, -4.23860422511256, -3.935144],
[-3.089209, -3.64977925312656, -3.99542033866934, -4.0285004042021, -5.12962054427214, -5.58258132916458, -5.73773916708354,
-5.57941299149575, -5.29939268084042, -5.35444868784392, -5.41951487793897, -5.44318511855928, -5.36060110805403,
-5.28265436718359, -5.32992198149075, -5.18727687243622, -4.7468288164082, -4.498764],
[-3.38359567857143, -4.01025479888158, -4.40976204264633, -4.55663701632959, -6.07418795572787, -6.38242118275209,
-6.47565191929893, -6.4481384547631, -6.35820071426784, -6.49018169661616, -6.5467588848174, -6.53694654788108, -6.41505837027443,
-6.18661305609947, -6.09100101359609, -5.75556848892304, -5.09169550335881, -4.88122042857143],
[-3.77094657142857, -4.4845647275066, -4.95494849524763, -5.25155361123419, -7.31703981290645, -7.43484204273565, -7.44658975116129,
-7.59119827485171, -7.75136917930393, -7.98456723447437, -8.02997468334166, -7.97610632330451, -7.8025021363539, -7.37603238362038,
-7.09242079268206, -6.50332061587938, -5.54546745987278, -5.38445257142857],
[-4.14280342857143, -4.93990225898663, -5.2515536112341925, -5.91867354234261, -8.5101775957979, -8.44516606831987, -8.37869006974915,

```
    -8.68853570213678, -9.08881090573858, -9.41917735081826, -9.45386184992495, -
9.357699707711, -9.13444815179019, -8.51787493804044,
    -8.05378378060459, -7.22116265775746, -5.98108853812618, -5.86755542857143],
    [-4.53015432142857, -5.41421218761166, -6.02351394234618, -6.61359013724721, -
9.75302945297649, -9.49758692830343, -9.34962790161151,
    -9.8315955222254, -10.4819793707747, -10.9135628886765, -10.9370776484492, -
10.7968594831344, -10.5218919178697, -9.70729426556135,
    -9.05520355969056, -7.9689147847138, -6.43486049464015, -6.37078757142857],
    [-4.90201117857143, -5.86954971909168, -6.54689293684344, -7.28071006835562, -
10.9461672358679, -10.5079109538876, -10.2817282201994,
    -10.9289329495105, -11.8194210972093, -12.3481730050204, -12.3609648150325, -
12.1784528675409, -11.853837933306, -10.8491368199814,
    -10.0165665476131, -8.68675682659189, -6.87048157289355, -6.85389042857143],
    [-5.28936207142857, -6.3438596477167, -7.09207938944474, -7.97562666326022, -
12.1890190930465, -11.5603318138712, -11.2526660520617,
    -12.0719927695991, -13.2125895622454, -13.8425585428786, -13.8441806135567, -
13.617612642964353, -13.2412816993854, -12.0385561475023,
    -11.0179863266991, -9.43450895354822, -7.32425352940752, -7.35712257142857],
    [-5.66121892857143, -6.79919717919673, -7.61545838394199, -8.64274659436864, -
13.382156875938, -12.5706558394554, -12.1847663706496,
    -13.1693301968841, -14.55003128868, -15.2771686592225, -15.26806778014, -
14.9992060273708, -14.5732277148217, -13.1803987019224,
    -11.9793493146216, -10.1523509954263, -7.75987460766092, -7.84022542857143]
];
class Line {
    x;
    y;
    mode;
    name;

    constructor(x, y, mode, name) {
        this.x = x;
        this.y = y;
        this.mode = mode;
        this.name = name;
    }
}
```

```
let dataLine1 = new Line(DEPTHS, AMPLITUDES[0], 'lines', 'Dry');
let dataLine2 = new Line(DEPTHS, AMPLITUDES[1], 'lines', '50 ml');
let dataLine3 = new Line(DEPTHS, AMPLITUDES[2], 'lines', '100 ml');
let dataLine4 = new Line(DEPTHS, AMPLITUDES[3], 'lines', '150 ml');
let dataLine5 = new Line(DEPTHS, AMPLITUDES[4], 'lines', '200 ml');
let dataLine6 = new Line(DEPTHS, AMPLITUDES[5], 'lines', '250 ml');
let dataLine7 = new Line(DEPTHS, AMPLITUDES[6], 'lines', '300 ml');
let dataLine8 = new Line(DEPTHS, AMPLITUDES[7], 'lines', '350 ml');
let dataLine9 = new Line(DEPTHS, AMPLITUDES[8], 'lines', '400 ml');
let dataLine10 = new Line(DEPTHS, AMPLITUDES[9], 'lines', '450 ml');
let dataLine11 = new Line(DEPTHS, AMPLITUDES[10], 'lines', '500 ml');
let dataLine12 = new Line(DEPTHS, AMPLITUDES[11], 'lines', '550 ml');
let dataLine13 = new Line(DEPTHS, AMPLITUDES[12], 'lines', '600 ml');
let dataLine14 = new Line(DEPTHS, AMPLITUDES[13], 'lines', '650 ml');
let dataLine15 = new Line(DEPTHS, AMPLITUDES[14], 'lines', '700 ml');
let dataLine16 = new Line(DEPTHS, AMPLITUDES[15], 'lines', '750 ml');
let dataLine17 = new Line(DEPTHS, AMPLITUDES[16], 'lines', '800 ml');

let inputValue = {
   mode: 'markers',
   type: 'scatter',
   name: 'Input'
};

let trace2d = [dataLine1, dataLine2, dataLine3, dataLine4, dataLine5, dataLine6, dataLine7,
dataLine8, dataLine9, dataLine10,
   dataLine11, dataLine12, dataLine13, dataLine14, dataLine15, dataLine16, inputValue]

let layout = {
   title: "Time vs. Amplitude at Different Moisture Levels",
   xaxis: {
      title: 'Depth (cm)',
   },
   yaxis: {
      title: 'Amplitude (dB)',
      range: [-6.0, 2.0]
   },
   legend: {
```

```
        y: 0.5,
    },
    font: {
        family: 'Poppins'
    },
    paper_bgcolor: 'rgba(0,0,0,0)',
    plot_bgcolor: 'rgba(0,0,0,0)',
}


function UpdatePlot() {
    var graphDiv = document.getElementById('ampVsTimePlot')
    var depth = +document.getElementById('depthInput').value;
    var amplitude = +document.getElementById('amplitudeInput').value;

    let newInput = {
        x: [depth],
        y: [amplitude],
        mode: 'markers',
        type: 'scatter',
        name: 'Input'
    };

    newData = trace2d[trace2d.length - 1] = newInput;
    // console.log(graphDiv);
    // console.log(newData);
    // console.log(layout);
    console.log(newData.x);

    Plotly.animate(graphDiv, {
        data: [{x: [depth], y: [amplitude]}],
        traces: [trace2d.length - 1],
        layout: {}
      }, {
        transition: {
          duration: 500,
          easing: 'cubic-in-out'
        },
```

```
        frame: {
          duration: 500
        }
      })
    //Plotly.react(graphDiv, newData, layout);
}

Plotly.newPlot( TESTER, trace2d, layout);
console.log("Creating 2d plot")
```

plot3d.js

```
const DIELECTRIC_CONSTANTS = [2.53, 2.74, 2.96, 3.18, 3.42, 3.67, 3.92, 4.18, 4.45, 4.73,
4.92, 5.17, 5.41, 5.66, 5.90, 6.15, 6.39];
const DC_EXTRAPOLATED = [4.45, 4.73, 4.92, 5.17, 5.41, 5.66, 5.90, 6.15, 6.39];
const VMI_VALUES = [0, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275,
0.3, 0.325, 0.35, 0.375, 0.4];
const VMI_EXTRAPOLATED = [0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4];
let amp_extrapolated = [[-2.655376, -3.11855213306653, -3.38481151175588, -
3.25019381790895, -3.73762646423212, -4.40386996598299, -4.6502887953977, -
4.2991859929965, -3.739044, -3.68073688544272, -3.7583131835918, -3.83132617008504, -
3.80666409004502, -3.95050472036018, -4.20833182891446, -4.34979449024512, -
4.23860422511256, -3.935144],
  [-3.089209, -3.64977925312656, -3.99542033866934, -4.0285004042021, -5.12962054427214,
-5.58258132916458, -5.73773916708354, -5.57941299149575, -5.29939268084042, -
5.35444868784392, -5.41951487793897, -5.44318511855928, -5.36060110805403, -
5.28265436718359, -5.32992198149075, -5.18727687243622, -4.7468288164082, -4.498764]];

// let extrapolatedAmp = extrapolateAmp();
let dielectricConstant = createDielectricConstantArray(DIELECTRIC_CONSTANTS);
let amp3d = createAmplitudeArray();
let depth3d = createDepthArray();


class Trace {
  x;
  y;
  z;
  name;
  type;
  showscale;
  colorscale;

  constructor(x, y, z, name, colorscale) {
     this.x = x;
     this.y = y;
     this.z = z;
     this.name = name;
```

```
    this.type = 'surface';
    this.showscale = false;
    this.colorscale = colorscale;
  }
}

let trace1 = new Trace(
  dielectricConstant,
  depth3d[0],
  amp3d[0],
  DEPTHS[0].toString(),
  [['0', 'rgb(170,110,40)'], ['1.0', 'rgb(170,110,40)']],
)

let trace2 = new Trace(
  dielectricConstant,
  depth3d[1],
  amp3d[1],
  DEPTHS[1].toString(),
  [['0', 'rgb(230,25,75)'], ['1.0', 'rgb(230,25,75)']],
)

let trace3 = new Trace(
  dielectricConstant,
  depth3d[2],
  amp3d[2],
  DEPTHS[2].toString(),
  [['0', 'rgb(250,190,212)'], ['1.0', 'rgb(250,190,212)']],
)

let trace4 = new Trace(
  dielectricConstant,
  depth3d[3],
  amp3d[3],
  DEPTHS[3].toString(),
  [['0', 'rgb(245,130,48)'], ['1.0', 'rgb(245,130,48)']],
)
```

```
let trace5 = new Trace(
  dielectricConstant,
  depth3d[4],
  amp3d[4],
  DEPTHS[4].toString(),
  [['0', 'rgb(255,215,180)'], ['1.0', 'rgb(255,215,180)']],
)

let trace6 = new Trace(
  dielectricConstant,
  depth3d[5],
  amp3d[5],
  DEPTHS[5].toString(),
  [['0', 'rgb(128,128,0)'], ['1.0', 'rgb(128,128,0)']],
)

let trace7 = new Trace(
  dielectricConstant,
  depth3d[6],
  amp3d[6],
  DEPTHS[6].toString(),
  [['0', 'rgb(255,225,25)'], ['1.0', 'rgb(255,225,25)']],
)

let trace8 = new Trace(
  dielectricConstant,
  depth3d[7],
  amp3d[7],
  DEPTHS[7].toString(),
  [['0', 'rgb(255,250,200)'], ['1.0', 'rgb(255,250,200)']],
)

let trace9 = new Trace(
  dielectricConstant,
  depth3d[8],
  amp3d[8],
  DEPTHS[8].toString(),
  [['0', 'rgb(210,245,60)'], ['1.0', 'rgb(210,245,60)']],
```

```
)

let trace10 = new Trace(
  dielectricConstant,
  depth3d[9],
  amp3d[9],
  DEPTHS[9].toString(),
  [['0', 'rgb(60,180,75)'], ['1.0', 'rgb(60,180,75)']],
)

let trace11 = new Trace(
  dielectricConstant,
  depth3d[10],
  amp3d[10],
  DEPTHS[10].toString(),
  [['0', 'rgb(170, 255, 195)'], ['1.0', 'rgb(170, 255, 195)']],
)

let trace12 = new Trace(
  dielectricConstant,
  depth3d[11],
  amp3d[11],
  DEPTHS[11].toString(),
  [['0', 'rgb(0,128,128)'], ['1.0', 'rgb(0,128,128)']],
)

let trace13 = new Trace(
  dielectricConstant,
  depth3d[12],
  amp3d[12],
  DEPTHS[12].toString(),
  [['0', 'rgb(70,240,240)'], ['1.0', 'rgb(70,240,240)']],
)

let trace14 = new Trace(
  dielectricConstant,
  depth3d[13],
  amp3d[13],
```

```
  DEPTHS[13].toString(),
  [['0', 'rgb(0,0,128)'], ['1.0', 'rgb(0,0,128)']],
)

let trace15 = new Trace(
  dielectricConstant,
  depth3d[14],
  amp3d[14],
  DEPTHS[14].toString(),
  [['0', 'rgb(0, 130, 200)'], ['1.0', 'rgb(0, 130, 200)']],
)

let trace16 = new Trace(
  dielectricConstant,
  depth3d[15],
  amp3d[15],
  DEPTHS[15].toString(),
  [['0', 'rgb(145,30,180)'], ['1.0', 'rgb(145,30,180)']],
)

let trace17 = new Trace(
  dielectricConstant,
  depth3d[16],
  amp3d[16],
  DEPTHS[16].toString(),
  [['0', 'rgb(220,190,255)'], ['1.0', 'rgb(220,190,255)']],
)

let trace18 = new Trace(
  dielectricConstant,
  depth3d[17],
  amp3d[17],
  DEPTHS[17].toString(),
  [['0', 'rgb(240,50,230)'], ['1.0', 'rgb(240,50,230)']],
)

var data2 = [trace1, trace2, trace3, trace4, trace5, trace6, trace7, trace8, trace9, trace10,
  trace11, trace12, trace13, trace14, trace15, trace16, trace17, trace18];
```

```
var layout2 = {
  title: 'Dielectric Plot',
  showscale: false,
  autosize: true,
  // width: 600,
  // height: 600,
  scene: {
    aspectmode: 'manual',
    aspectratio: {
      x: 1.0, y: 1.0, z: 0.7
    },
    xaxis: {
      title: 'Dielectric Constant',
      range: [2.4, 6.5],
    },
    yaxis: {
      title: 'Depth',
      range: [-0.5, 8.5]
    },
    zaxis: {
      title: 'Amplitude',
      range: [-6.0, 2.0],
      font: {
        family: 'Poppins',
      }
    },
  },
  paper_bgcolor: 'rgba(0,0,0,0)',
  plot_bgcolor: 'rgba(0,0,0,0)',
};


function createDepthArray() {
  let newDepth = [];
  for (let j = 0; j < DEPTHS.length; j++) {
    let trace = [];
    for (let i = 0; i < DIELECTRIC_CONSTANTS.length; i++) {
```

```
      trace.push([DEPTHS[j], DEPTHS[j]]);
    }
    newDepth.push(trace);
  }
  // console.log(newDepth);
  return newDepth;
}


function createAmplitudeArray() {
  let newAmp = [];
  let MIN_AMP = -6.0;

  for (let j = 0; j < DEPTHS.length; j++) {
    let trace = [];
    for (let i = 0; i < DIELECTRIC_CONSTANTS.length; i++) {
      trace.push([AMPLITUDES[i][j], MIN_AMP]);
      //console.log(AMPLITUDES[i][index]);
    }
    newAmp.push(trace);
  }
  // console.log(newAmp);
  return newAmp;
}


function createDielectricConstantArray(arr) {
  let newArr = [];
  // let minDielectricConstant = Math.min(...arr);
  for (let i = 0; i < arr.length; i++) {
    newArr.push([arr[i], arr[i]/*minDielectricConstant*/]);
  }
  // console.log(newArr)
  return newArr;
}


function calcVolMoisture(dielectricConstant) {
```

```
  let index = 0;
  for (let i = 0; i < DIELECTRIC_CONSTANTS.length; i++) {
    if (dielectricConstant < DIELECTRIC_CONSTANTS[i]) {
      index = i;
      break;
    }
  }

  // console.log(DIELECTRIC_CONSTANTS[index-1], VMI_VALUES[index-1],
DIELECTRIC_CONSTANTS[index], VMI_VALUES[index]);
  let vmi = interpolate(dielectricConstant, DIELECTRIC_CONSTANTS[index-1],
VMI_VALUES[index-1], DIELECTRIC_CONSTANTS[index], VMI_VALUES[index]);
  return (vmi.toFixed(3));
}


function calcDielectric(depth, amplitude) {

  let traceFloor, traceCeiling = [];
  let x00, x01, y00, y01, z00, z01 = 0;
  let x10, x11, y10, y11, z10, z11 = 0;

  //find which two traces to use (using depth value)
  for (let k = 0; k < data2.length; k++) {
    // console.log(data2[k].y[0][0]);
    if (data2[k].y[0][0] > depth) {
      traceFloor = data2[k-1];
      traceCeiling = data2[k];

      // console.log(traceFloor);
      // console.log(traceCeiling);
      break;
    }
  }

  //find which AMPLITUDES to use
  for (let m = 0; m < traceFloor.z.length; m++) {
    // console.log(traceFloor.z[m], amplitude);
```

```
  if (traceFloor.z[m][0] < amplitude) {
    console.log(traceFloor.x[m][0], traceFloor.y[m][0], traceFloor.z[m][0]);
    if (traceFloor.x[m][0] == 2.53) {
      console.log("error");
      setIndicator(indicatorElement, "NaN");
      return "error";
    }
    x00 = traceFloor.x[m][0];
    y00 = traceFloor.y[m][0];
    z00 = traceFloor.z[m][0];
    x01 = traceFloor.x[m-1][0];
    y01 = traceFloor.y[m-1][0];
    z01 = traceFloor.z[m-1][0];
    console.log(x00, x01);
    break;
  }
}
for (let n = 0; n < traceCeiling.z.length; n++) {
  if (traceCeiling.z[n][0] < amplitude) {
    x10 = traceCeiling.x[n][0];
    y10 = traceCeiling.y[n][0];
    z10 = traceCeiling.z[n][0];
    x11 = traceCeiling.x[n-1][0];
    y11 = traceCeiling.y[n-1][0];
    z11 = traceCeiling.z[n-1][0];
    // console.log(x10, x11);
    break;
  }
}

// console.log(depth, y00, x00, y10, x10, z00, z01, z10, z11);
let dielectricConstant_floor = interpolate(depth, y00, x00, y10, x10);
let dielectricConstant_ceil = interpolate(depth, y01, x01, y11, x11);

let ampFloor = interpolate(depth, y00, z00, y10, z10);
let ampCeil = interpolate(depth, y01, z01, y11, z11);

// console.log(dielectricConstant_floor, dielectricConstant_ceil, ampFloor, ampCeil);
```

```
  let dielectricConstant = interpolate(amplitude, ampFloor, dielectricConstant_floor, ampCeil,
dielectricConstant_ceil);
  // console.log(dielectricConstant);

  //interpolate - fix from using average values
  return dielectricConstant.toFixed(2);
}

//returns y-value
function interpolate(x, x0, y0, x1, y1) {
  return (((y0 * (x1 - x)) + (y1 * (x - x0))) / (x1 - x0))
}


// function extrapolate(x, y) {
//   let xSum = 0, ySum = 0, xxSum = 0, xySum = 0;
//   let count = x.length;
//   for (var i = 0; i < count; i++) {
//     xSum += x[i];
//     ySum += y[i];
//     xxSum += x[i] * x[i];
//     xySum += x[i] * y[i];
//   }
//   var slope = (count * xySum - xSum * ySum) / (count * xxSum - xSum * xSum);
//   var intercept = (ySum / count) - (slope * xSum) / count;

//   var xValues = [];
//   var yValues = [];
//   for (var j = 0.25; j <= 0.425; j+= 0.025) {
//     xValues.push(j.toFixed(3));
//     yValues.push((j*slope + intercept).toFixed(3));
//   }
//   console.log(slope, xValues, yValues);
//   return [xValues, yValues];
// }

// function extrapolateAmp() {
```

```
//   // console.log(extrapolate(DC_EXTRAPOLATED[0],
DIELECTRIC_CONSTANTS[DIELECTRIC_CONSTANTS.length-2], amp3d[i][j-2][0],
DIELECTRIC_CONSTANTS[DIELECTRIC_CONSTANTS.length-1], amp3d[i][j-2][0]));
//   // console.log(extrapolate(DC_EXTRAPOLATED[1],
DIELECTRIC_CONSTANTS[DIELECTRIC_CONSTANTS.length-1], amp3d[i][j-2][0],
DC_EXTRAPOLATED[0], amp3d[i][j-2][0]));
//   for (let i = 0; i < DC_EXTRAPOLATED.length-2; i++) {
//     let tmp = []
//     for (let j = 0; j < DEPTHS.length; j++) {
//       let newAmp = extrapolate(DC_EXTRAPOLATED[i+2], DC_EXTRAPOLATED[i],
amp_extrapolated[i][j], DC_EXTRAPOLATED[i+1], amp_extrapolated[i+1][j]);
//       tmp.push(newAmp);
//       // console.log(newAmp, DC_EXTRAPOLATED[i+2], DC_EXTRAPOLATED[i],
amp_extrapolated[i+1][j]);
//     }
//     amp_extrapolated.push(tmp);
//   }
// }

function extrapolate(x, x1, y1, x2, y2) {
  return (y1 + (((x - x1) / (x2 - x1)) * (y2 - y1)))
}


function Update3dPlot() {
  var graph3dDiv = document.getElementById('dielectricPlot');
  var depth = +document.getElementById('depthInput').value;
  var amplitude = +document.getElementById('amplitudeInput').value;
  let dielectricConstant = calcDielectric(depth, amplitude);
  // console.log(dielectricConstant);

  if (dielectricConstant != "error") {
    let new3dInput = {
      x: [dielectricConstant],
      y: [depth],
      z: [amplitude],
      mode: 'markers',
      type: 'scatter3d',
```

```
    name: 'Input',
    marker: {
      size: 6,
      color: 'rgb(180, 180, 180)',
      line: {
        color: 'rgb(0,0,0)',
        width: 2,
      }
    },
  };

  let vmi = calcVolMoisture(dielectricConstant);

  setIndicator(indicatorElement, vmi);

  new3dData = data2.concat(new3dInput);
//  console.log(new3dInput);
  // console.log(new3dData);
  // console.log(layout2);
  Plotly.react(graph3dDiv, new3dData, layout2);
  } else {
    setIndicator(indicatorElement, "NaN");
  }

}

console.log("Creating 3d Plot");
console.log(data2[0]);

Plotly.newPlot('dielectricPlot', data2, layout2);
```

preload.js

```
window.addEventListener('DOMContentLoaded', () => {
    const replaceText = (selector, text) => {
      const element = document.getElementById(selector)
      if (element) element.innerText = text
    }

    for (const type of ['chrome', 'node', 'electron']) {
      replaceText(`${type}-version`, process.versions[type])
    }
  })
```

forge.config.js

```
module.exports = {
  packagerConfig: {},
  rebuildConfig: {},
  makers: [
    {
      name: '@electron-forge/maker-squirrel',
      config: {},
    },
    {
      name: '@electron-forge/maker-zip',
      platforms: ['darwin'],
    },
    {
      name: '@electron-forge/maker-deb',
      config: {},
    },
    {
      name: '@electron-forge/maker-rpm',
      config: {},
    },
  ],
};
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Loam: Soil Analysis</title>
   <link rel="stylesheet" href="style.css">
   <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Poppins">
   <script src="plotly-2.14.0.min.js"></script>
   <script src='https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.17/d3.min.js'></script>
</head>

<body>
   <div class="sidebar">

     <div class="sidebar-element">
       <div class="moisture-indicator">
         <div class="indicator-body">
           <div class="indicator-fill"></div>
           <div class="indicator-cover">
             <p>--</p>
           </div>
         </div>
       </div>
     </div>

     <div class="sidebar-element">
       <label for="options">Select a type:</label>
       <select id="options" name="options" onchange="selectVeg(this.value)">
         <option value="soybean" selected>Soybean</option>
         <option value="rice">Rice</option>
         <option value="potato">Potato</option>
       </select>
     </div>

     <div class="sidebar-element">
       <label for="depthInput">Depth (cm):</label>
       <input id="depthInput" placeholder="0 - 8.15">
```

```
    </div>
    <div class="sidebar-element">
      <label for="amplitudeInput">Amplitude (dB):</label>
      <input id="amplitudeInput" placeholder="-6.0 - 2.0">
    </div>

    <div class="sidebar-element">
      <input type="button" value="Enter" id="plotBtn">
    </div>

    <div class="sidebar-element">
      <p id="report"></p>
    </div>
  </div>

  <div class="plot-card">

    <div class="plot-container">
      <div class="plots">
        <div id="ampVsTimePlot"></div>
      </div>
      <div class="plots">
        <div id="dielectricPlot"></div>
      </div>
    </div>
    <br>

    <div style="text-align:center">
      <a class="prev" onclick="changePlot(-1)">&#10094;</a>
      <div class="dot" onclick="currentPlot(1)"></div>
      <div class="dot" onclick="currentPlot(2)"></div>
      <a class="next" onclick="changePlot(1)">&#10095;</a>
    </div>

  </div>

<script src="navigation.js"></script>
<script src="moisture-indicator.js"></script>
```

```
  <script src="plot.js"></script>
  <script src="plot3d.js"></script>
  <script src="input_validation.js"></script>
</body>

</html>
```

style.css

```
* {box-sizing:border-box; font-family: 'Poppins';}

body {
  background-color: #ffffff;
}

:focus {
    outline-color: #3498db;
}

.input-error{
    outline-color: #fc4349;
}

.sidebar {
  margin: 0;
  padding: 0;
  top: 0;
  left: 0;
  width: 200px;
  background-color: #2c3e50;
  color: #ffffff;
  position: absolute;
  height: 100%;
  overflow: auto;
  float: left
}

.sidebar .sidebar-element {
  display: block;
  padding: 8px;
}

.plot-card {
  /* display: inline-block;
  vertical-align: middle; */
  padding: 0;
```

```
  margin-left: 245px;
  /* box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2); */
  width: 700px;
  max-width: 1000px;
}

/* Slideshow container */
.plot-container {
  /* max-width: 1000px; */
  position: relative;
  margin: auto;

}

/* Hide the images by default */
.plots {
  /* border: 2px solid black; */
  border-radius: 5px;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
  display: none;
  animation: fadeIn 0.6s;
}

/* Next & previous buttons */
.prev, .next {
  cursor: pointer;
  /* position: absolute */
  padding: 10px;
  margin: 0px 50px;
  color: #2c3e50;
  font-weight: bold;
  font-size: 24px;
  transition: 0.6s ease;
  border-radius: 3px 3px 3px 3px;
  user-select: none;
}

/* .prev {
```

```
  right: -10%;
} */
/* Position the "next button" to the right */
/* .next {
  left: -10%;
} */

/* On hover, add a black background color with a little bit see-through */
.prev:hover, .next:hover {
  background-color: rgba(44,62,80,0.5);
  color: white;
}


/* The dots/bullets/indicators */
.dot {
  cursor: pointer;
  height: 15px;
  width: 15px;
  margin: 0 2px;
  background-color: rgba(44,62,80,0.5);
  border-radius: 50%;
  display: inline-block;
  transition: background-color 0.6s ease;
}

.active, .dot:hover {
  background-color: rgba(44,62,80,1.0);
}


.moisture-indicator {
  height: 10em;
  width: 10em;
  color: #004033;
  /* display: inline-block; */
  /* vertical-align: middle; */
}
```

```
.indicator-body {
    width: 100%;
    height: 100%;
    padding-bottom: 50%;
    background: #ffffff;
    position: relative;
    border-radius: 50%;
    overflow: hidden;
}

.indicator-fill {
    position: absolute;
    left: 0;
    width: inherit;
    height: 100%;
    background: #3498db;
    transform-origin: center bottom;
    transform: translateY(50%);
    transition: transform 0.2s ease-out;
}

.indicator-cover {
    width: 75%;
    height: 75%;
    background-color: #d7dadb;
    position: absolute;
    top: 20%;
    left: 12.5%;
    transform: rotate(45deg);
    border-bottom-right-radius: 60%;
    border-bottom-left-radius: 60% 80%;
    border-top-right-radius: 80% 60%;
    font-size: 32px;
    display: flex;
    align-items: center;
    justify-content: center;
    box-sizing: border-box;
```

```
}

.indicator-cover p {
   transform: rotate(-45deg);
}

#report {
  background-color: #ffffff;
  color: #004033;
  text-align: center;
  border-radius: 5px;
  padding: auto;
}

@keyframes fadeIn {
  0% { opacity: 0; transform: translateX(100%);}
  100% { opacity: 1; transform: translateX(0%);}
}
```