G-Function Library Guide (V 1.0)

Timothy West, Jack C. Cook, Jeffrey D. Spitler

Oklahoma State University

June 30, 2021

Thermal response functions, known as g-functions, are commonly used to simulate ground heat exchangers used with ground-source heat pump systems. G-functions were originally developed by Prof. Johan Claesson and his graduate students at the University of Lund in Sweden. (Claesson and Eskilson 1988, Eskilson 1987, Hellström 1991) G-functions are used by ground heat exchanger design tools (BLOCON 2017, Spitler 2000) and whole building energy simulation tools (Liu and Hellström 2006, Mitchell and Spitler 2020). G-functions are unique to a specific borehole configuration (geometry, e.g. 5 rows of 10 boreholes spaced 5m apart) and depth.

Calculation of g-functions can be quite computationally time-consuming, particularly as the number of boreholes gets large. However, once the g-function is computed, the actual simulation time can be quite short, particularly if a hybrid time-step (Cullin and Spitler 2011) approach is used. Because of this, pre-computed g-function libraries are commonly used in design tools and building simulation tools. In practice, the g-functions are pre-calculated for specific configurations; for each configuration multiple depths are pre-computed for interpolation purposes. Then, a design tool, can iteratively adjust the depth to find the correct-sized ground heat exchanger. Furthermore, the g-functions scale with several non-dimensional parameters that allow wider application than the specific horizontal spacing and depths used in the pre-calculation.

Currently available libraries, implemented in eQuest (Liu and Hellström 2006), GLHEPRO (Spitler 2000), and EED (BLOCON 2015) have less than a 1000 possible configurations and are proprietary. This document gives a brief description of a new, publicly available library containing g-functions for 34,321 configurations at 5 depths. Some of the configurations are available in existing libraries; others are new. The new configurations are C-shapes, lopsided-U-shapes and zoned rectangles (Cook and Spitler 2021). In anticipation of possible future expansions to the library, this version is referred to as V1.0; this is included in the JSON file names.

Library Overview

This library contains g-functions for standard, regularly spaced vertical borehole ground heat exchangers. In total, it contains 34, 321 configurations. To permit interpolation, each configuration has g-functions for heights of 24, 48, 96, 192, and 384 m. All the g-functions were calculated with burial depths of 2m, and borehole diameters of 15 to 17.5 cm, depending on height. Deeper boreholes commonly have larger diameters, which is why these values were chosen. The g-function can be corrected to any reasonable diameter, though. In configurations with uniform spacing, the spacing between the boreholes is set to 5m, though it can be scaled to other horizontal spacings.

This library comes in the form of seven JSON[1] files each containing a specific kind of borehole configuration. The seven configurations are rectangles, zoned rectangles, open rectangles, C-shapes, U-shapes, lopsided U-shapes, and L-shapes. Table 1 contains a summary of the number of configurations in each library as well as a brief description of each type of configuration. Appendices 1-7 contain more detailed explanations of each library.

Table 1 Library Contents Overview

Configuration Name	Number of Cases	Notes
Rectangle	1,651	Standard *N*x*M* cases (i.e. *N* rows, *M* columns) with uniform spacing. Only one key is required to access a specific configuration.
Zoned Rectangle	12,615	Similar to the Rectangle configurations, this configuration type has had rows/columns removed from the interior in order to represent configurations where the interior spacing of the bore field is greater than that of the exterior (or perimeter) spacing. This library defines a specific configuration using the *N* and *M* values for the exterior as well as *Ni* and *Mi* values for the interior section. Two keys are required to access a specific configuration
Open Rectangle	2,332	These configurations represent *N* by *M* rectangular cases where there boreholes only around the perimeter, but the perimeter can have more than one row of boreholes. The number of rows around the perimeter is specified with the key "*t*." Two keys are required to access a specific configuration.
C	4,525	This type of configuration may be thought of as an open rectangle configuration that has had some number of boreholes removed from the top side. The current C configurations in the library all have one row of boreholes around the perimeter. The number of holes removed is represented by the key "*r*." Two keys are required to access a specific configuration.
L	495	These configurations consist of a line *N* boreholes long and M boreholes wide. The L cases have a single row of boreholes. Only one key is required to access a specific configuration.
U	3, 248	This type of configuration is U-shaped, with the opening at the top. The U may have up to 3 perimeter rows of boreholes around all sides of the U. The number of perimeter rows is represented with "*t*." Two keys are required to access a specific configuration.
LopU (Lopsided U)	9,455	These configurations consist of U cases that have had some number of bore holes removed from their right side. These configurations all have a single row of perimeter boreholes. The number of bore holes removed is represented by "*r*." Two keys are required to access a specific configuration.

[1] JSON "JavaScript Object Notation" files are a standard file format commonly used with Python, and other languages, including Java, C#, C++, and PHP.

Each library requires two JSON keys to access a specific bore hole configuration. The first required key is consistent across the seven libraries. It is "M_N" where M represents the number of bore holes along the x-axis and N represents the number of boreholes along the y-axis. It should be noted that N>=M. The second key is different depending on the library.

Additional keys are required to access the data for a specific configuration. Each configuration has three keys: "bore_locations," "g," and "logtime." The "bore_locations" key accesses a 2d array containing the x and y locations for each bore hole in the configuration. The "g" key returns a dictionary containing the g-functions that were calculated for the configuration. The "logtime" key returns a 1d array containing $\ln\left(\frac{t}{t_s}\right)$ values for which the configurations g-functions were evaluated.

The "g" dictionary requires one more key to access a 1d array containing the values of the g-function for a specific configuration under a specific set of conditions. The key uses the format "B._H._r_b". "B" represents the spacing of the configuration (5m for all cases currently in the libraries). "H" represents the height of the bore holes, and r_b represents the radius of each bore hole. Currently, each height has a specific bore hole radius pairing. The possible keys are the following: "5._24._0.075", "5._48._0.075", "5._96._0.075","5._192._0.08", and "5._384._0.0875".

The python file "LibraryAccessExample.py" contains examples regarding how to use python to access g-functions from the library. The first example demonstrates how to access a configuration from the "Rectangle" library. It also demonstrates how the information from the configuration can be accessed/used (such as outputting it to a csv file). The second example also illustrates how to use the information for a configuration, but it shows how to access a configuration in the "LopU" library. The main difference between accessing a configuration in the two libraries is that the "Rectangle" library requires one key to access a configuration, but the "LopU" library require two. The remaining examples are similar to the second one, but access one of the other five libraries (the 2nd key varies from library to library). Appendix 8 also contains the code for the example.

Calculation Methodology

This section briefly describes the procedure used to calculate the g-functions. The g-functions are calculated with a tool that we call "cpgfunction" (Cook and Spitler 2021). It is based on the finite line source methodology developed by Cimmino (2018a, 2018b) for an open-source tool written in Python, called pygfunction. Cpgfunction is written in C++. Cpgfunction was developed with an eye towards reducing memory consumption, which can be quite high for large numbers of boreholes, exceeding 96 GB in many cases. For calculating large numbers of g-functions, as was done here, the memory requirements can become critical when running on a cluster. Keeping the memory requirements below 96 GB allowed us to fully use the most common compute nodes on the Oklahoma State University High Performance Computing Cluster (OSUHPCC 2020). The time requirement is also improved for most cases, but large numbers of regularly spaced boreholes the computation times are similar. For further information on cpgfunction, see Cook and Spitler (2021).

The g-functions are calculated with the "Uniform borehole wall temperature" (UBHWT) boundary condition. That is, the heat input at each segment is adjusted to give uniform (but changing with time) temperatures at the borehole walls. This is the method commonly used to develop other g-function libraries and has been used to size ground heat exchangers for commercial systems for the last 30 years.

Arguably, the "Uniform inlet fluid temperature" (UIFT) conditions are more physically realistic, since the boreholes in ground heat exchangers are generally plumbed in parallel, and all receive approximately the same inlet fluid temperature at any time. However, the g-functions calculated with UIFT conditions will be slightly different than the UBHWT, and they have a dependence on the local borehole thermal resistance and the mass flow rate of the fluid. The g-functions also depend on the number of segments used – like most numerical analyses, increasing the number of cells or volumes increases the accuracy, with diminishing returns. We did a grid-independency analysis using typical values of borehole thermal resistance and mass flow rate using the UIFT boundary conditions. We then compared results for the UBHWT boundary conditions and found that we could use a smaller number of segments with the UBHWT boundary conditions and still closely approximate the g-functions calculated with UIFT boundary conditions, and typical borehole thermal resistances and mass flow rates. This investigation yielded the number of segments summarized in Table 2. We utilized these values in calculating the g-functions and refer to this method as the "adaptive discretization scheme."

Table 2 Adaptive discretization scheme

Depth (m)	Range (NBH)	Segment Length (m)	Number of Segments / BH
24	All	8	3
48	All	12	4
96	All	12	8
192	NBH < 120 BH	16	12
192	NBH >= 120 BH	12	16
384	NBH < 220 BH	16	24
384	NBH >= 220 BH	12	32

The library presented here represents weeks of computation time on the OSU High Performance Computing Cluster, and it would not have been feasible to develop such a library without access to this or a similar resource.

Interpolation Methodology

Whether for design or energy simulation purposes, borehole configurations often will have different spacings, different depths, and different borehole radii than the values used to calculate the library g-functions. This section discusses a recommended procedure for interpolating between library g-functions. There are four parameters that can be used to non-dimensionally scale the results:

- B – the borehole spacing
- H – the "active" borehole depth – the length of the borehole over which heat transfer to the ground takes place. For a grouted borehole, this would be the length from the connection to the header to the bottom of the U-tube. In Scandinavia, groundwater-filled boreholes are

common, and, in that case the active portion of the borehole starts at the water table. Above the water table, the U-tube is suspended in air, with relatively little heat transfer.

- *D* – the depth from the ground surface to the top of the active portion of the borehole.
- r_b – the borehole radius

As discussed by Cimmino and Bernier (2014), the dimensionless g-function values depend on four dimensionless parameters:

- $\frac{t}{t_s}$, the dimensionless time
- $\frac{r_b}{H}$, the ratio of the borehole radius to the length of the borehole
- $\frac{B}{H}$, the ratio of the spacing between boreholes to the borehole length
- $\frac{D}{H}$, the ratio of the depth of the top of the borehole to the borehole length

Claesson and Eskilson (1987) – See Eskilson (1987) concluded that $\frac{D}{H}$ was relatively unimportant on the basis of varying the depth between 2 and 8 m and only finding a 0.1°C difference in extraction temperature. Presumably, for this reason, Eskilson (1987) does not specify the $\frac{D}{H}$ value for g-functions published in his thesis, but it appears that the g-functions correspond to $\frac{D}{H}$ values of about 0.06; that is for a 100m deep borehole, the presumed burial depth is 6m. This value makes sense for Scandinavian groundwater-filled boreholes, but in ground heat exchangers installed in grouted boreholes, the typical depth is considerably less; we used a value of 2m.[2] We used a value of 5m for the spacing (B). A range of *H* values were used and borehole radius was varied to match typical designs, as shown in Table 3. Table 4 summarizes the dimensional and dimensionless values.

Table 3 Borehole depths and diameters

Depth (m)	BH Diameter (mm)
24	150
48	150
96	150
192	160
384	175

[2] G-functions show greater response as the burial depth increases. We chose 2m as a conservative value.

Table 4 Summary of dimensional and non-dimensional parameters

B (m)	*D* (m)	*H* (m)	r_b (mm)	*B/H*	*D/H*	r_b/H
5	2	24	75	0.20833	0.08333	0.003125
5	2	48	75	0.10417	0.04167	0.001563
5	2	96	75	0.05208	0.02083	0.000781
5	2	192	80	0.02604	0.01042	0.000417
5	2	384	87.5	0.01302	0.00521	0.000228

Any borehole configuration has three dimensionless geometric parameters. Calculation of a library with enough variations in all three parameters to allow for 3-d interpolation is not computationally feasible. Therefore, we rely on the relative importance of each dimensionless parameter to establish procedures for calculating library g-functions and interpolating them during the design phase.

First, g-functions are much more sensitive to the value of *B/H* than to the values of *D/H* and $\frac{r_b}{h}$. Therefore, we compute the library g-functions for the parameters shown in Table 4, and use *B/H* to interpolate to get the g-function. This has the consequence that the interpolated g-function has corresponding values of *D/H* and $\frac{\mathrm{r_b}}{\mathrm{h}}$ that may be estimated with interpolation.

Regarding the value of *D/H*, as discussed above, Claesson and Eskilson (1987) found that variations in *D* in the range of 2-8m have very little impact on the g-function. As we have computed all g-functions for a burial depth of 2m, the interpolation will result in depths around 2m, but not exactly 2m. Again, this approximation has only a small effect and is deemed acceptable in order to keep the time required to compute the library feasible.

Regarding the value of $\frac{r_b}{h}$, the interpolated g-function values can be corrected using an expression given by Claesson and Eskilson (1988):

$$g\left(\frac{t}{t_s},\frac{B}{H},\frac{D}{H},\frac{r_b^*}{H}\right) = g\left(\frac{t}{t_s},\frac{B}{H},\frac{D}{H},\frac{r_b}{H}\right) - ln\left(\frac{r_b^*}{r_b}\right) \quad (1)$$

Where r_b^* is the borehole radius for the actual design;

r_b is the borehole radius that was interpolated for the specific *B/H* value.

The procedure for determining the g-function values that correspond to a specific borehole configuration, borehole depth, and borehole radius is summarized in Figure 1.

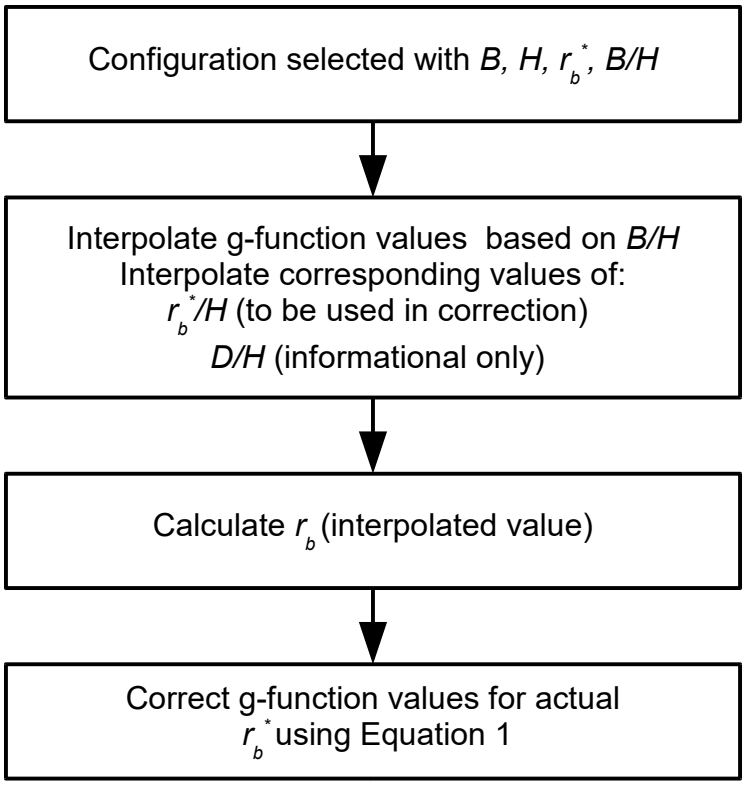

Figure 1 Summary of g-function interpolation procedures

As an example, consider a 5x8 rectangular borehole configuration with *B*=7m; *H*=150m; *D*=2m, and r_b=80mm. The library entries for this configuration are shown in Table 5, in the two columns labeled "Library" (brown font, yellow background). The interpolation inputs (are shown in the interpolation column – the *B* and *H* rows; the value of *B/H* is calculated from the inputs. The other values (green font with green background) are calculated by interpolation, based on *B/H*. The interpolations for *D/H* and r_b/H are graphically illustrated in Figure 2. The resulting *D* value (2.8m) does not match the actual D value, but the effect is small. The resulting r_b value does not match the actual value, but it can be corrected using Equation 1.

Table 5 Example interpolation

Parameter	Library	Interpolated	Library
D	2	2.8	2
B	5	7	5
H	96	150	192
r_b	0.075	0.106	0.08
B/H	0.052083	0.046667	0.026042
D/H	0.020833	0.018667	0.010417
r_b/H	0.000781	0.000705	0.000417

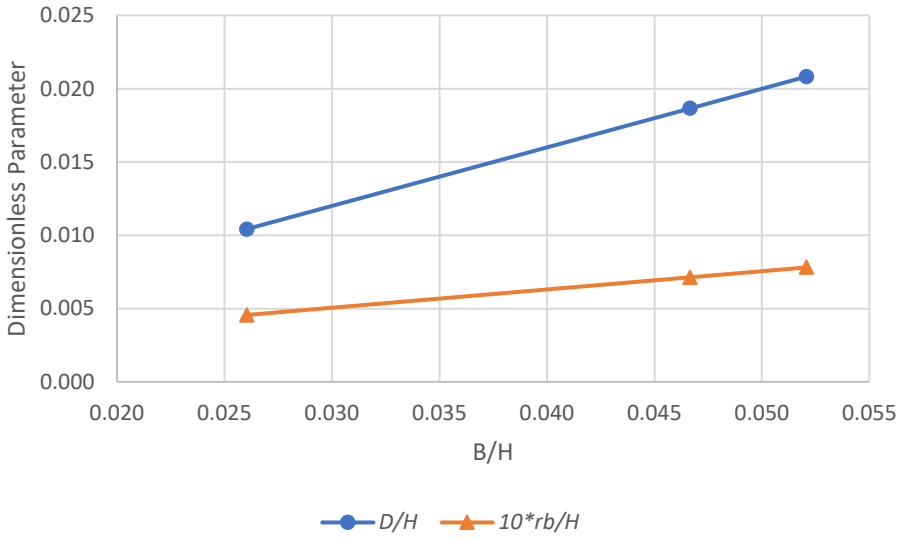

Figure 2 Interpolation illustrated

The B/H value for the field with 7m spacing is 20.8% of the way between the B/H for the 96m deep field and the B/H value for the 192m deep field. In this example, linear interpolation has been done; more sophisticated interpolations are possible. Figure 3 shows the results; the correction for the actual borehole radius is very small, so the curve for the interpolated g-function and the corrected g-function nearly lie on top of each other.

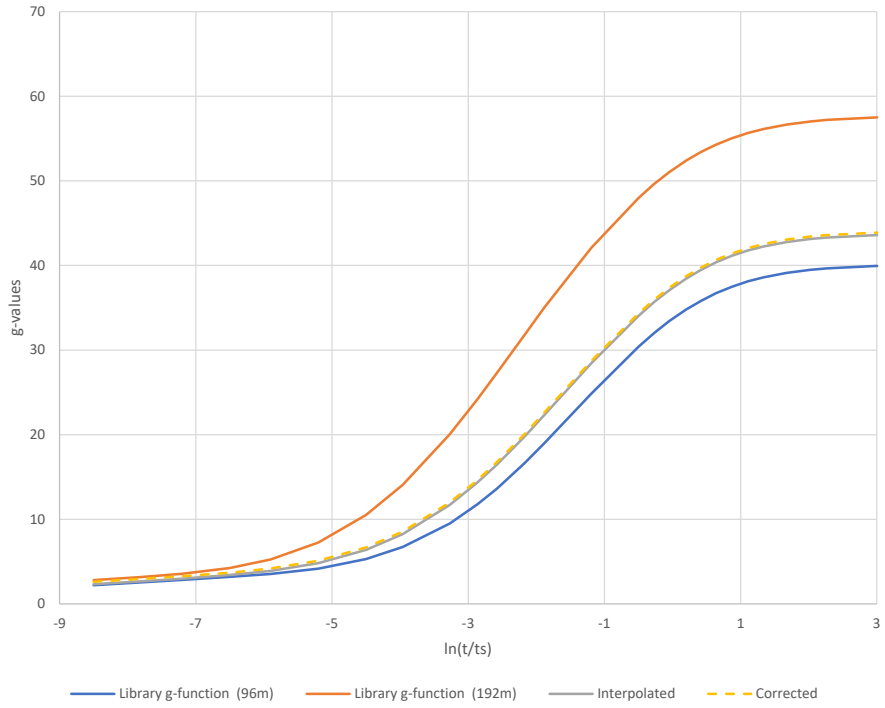

Figure 3 Interpolation to obtain g-function for a 5x8 rectangular borefield with B=7m; H=150m; D=2.8m, and r_b=80mm.

Table 6 Example interpolation g-function values

$\ln(t/t_s)$	Library g-function (96m)	Library g-function (192m)	Interpolated g-function	Corrected g-function
-8.5	2.209	2.835	2.339	2.619
-7.8	2.555	3.191	2.688	2.967
-7.2	2.852	3.560	2.999	3.279
-6.5	3.203	4.250	3.421	3.700
-5.9	3.554	5.267	3.910	4.190
-5.2	4.187	7.263	4.827	5.107
-4.5	5.320	10.513	6.400	6.680
-3.963	6.729	14.076	8.257	8.537
-3.27	9.518	20.060	11.711	11.991
-2.864	11.769	24.198	14.354	14.633
-2.577	13.644	27.314	16.488	16.767
-2.171	16.649	31.816	19.804	20.083
-1.884	18.984	34.992	22.313	22.593
-1.191	24.874	42.089	28.455	28.734
-0.497	30.407	47.997	34.066	34.345
-0.274	31.984	49.596	35.647	35.927
-0.051	33.399	51.010	37.062	37.342
0.196	34.776	52.371	38.436	38.715
0.419	35.832	53.409	39.488	39.768
0.642	36.732	54.292	40.384	40.664
0.873	37.497	55.044	41.146	41.426
1.112	38.129	55.668	41.777	42.056
1.335	38.598	56.132	42.245	42.524
1.679	39.128	56.659	42.774	43.054
2.028	39.490	57.020	43.136	43.416
2.275	39.668	57.197	43.314	43.594
3.003	39.959	57.487	43.605	43.885

Acknowledgements

Development of this library was funded through Department of Energy contract DE-AC05-00OR22725, via a subcontract from Oak Ridge National Laboratory. Computation of the library g-functions was made possible by the Oklahoma State University High Performance Computing Center. Development of the g-function calculation tool, cpgfunction, was supported by Oklahoma State University via the OG&E Energy Technology Chair.

References

BLOCON. 2015. “Earth Energy Designer (EED) Version 3.2 Manual.” https://buildingphysics.com/eed-2/.

BLOCON. 2017. “Earth Energy Designer (EED) Version 4 Update Manual.” https://buildingphysics.com/eed-2/.

Cimmino, M. and M. Bernier. 2014. A semi-analytical method to generate g-functions for geothermal bore fields. International Journal of Heat and Mass Transfer 70: 641-650.

Cimmino, M. 2018a. "Fast calculation of the g-functions of geothermal borehole fields using similarities in the evaluation of the finite line source solution." Journal of Building Performance Simulation 11(6): 655-668.

Cimmino, M. 2018b. pygfunction: an open-source toolbox for the evaluation of thermal. eSim 2018, Montreál, IBPSA Canada.

Claesson, J. and P. Eskilson 1987. Conductive Heat Extraction by a Deep Borehole. Analytical Studies. Lund, Sweden, University of Lund.Claesson, J and P. Eskilson. 1988. Conductive heat extraction to a deep borehole: Thermal analyses and dimensioning rules. Energy (Oxford), 13(6), 509–527. https://doi.org/10.1016/0360-5442(88)90005-9

Cook, J. C. and J. D. Spitler. 2021. Faster computation of g-functions used for modeling of ground heat exchangers with reduced memory consumption. Building Simulation 2021. Bruges, Belgium, IBPSA.

Cullin, J. R. and J. D. Spitler. 201). "A computationally efficient hybrid time step methodology for simulation of ground heat exchangers." Geothermics 40(2): 144-156.

Eskilson, P. 1987. Thermal Analysis of Heat Extraction Boreholes. Ph.D. thesis. University of Lund.

Hellström, G. 1991. Ground heat storage: thermal analyses of duct storage systems. Ph.D. thesis. University of Lund.

Liu, X. and G. Hellström (2006). Enhancements of an Integrated Simulation Tool for Ground-Source Heat Pump System Design and Energy Analysis. Ecostock 2006. Stockton State College, Pomona, NJ.

Mitchell, M. S. and J. D. Spitler (2020). "An Enhanced Vertical Ground Heat Exchanger Model for Whole-Building Energy Simulation." Energies 13(16): 4058.

OSUHPCC. (2020). "OSU's newest supercomputer "Pete" is available for all OSU researchers." Retrieved 29 January, 2021, from https://hpcc.okstate.edu/pete-supercomputer.html.

Spitler, J.D. 2000. GLHEPRO -- A Design Tool For Commercial Building Ground Loop Heat Exchangers. Proceedings of the Fourth International Heat Pumps in Cold Climates Conference, Aylmer, Québec. August 17-18, 2000.

Appendix 1: Rectangle Library

This library contains rectangles of uniformly distributed bore holes, and has no secondary key, so it only requires the first “M_N” key to access a configuration. Figure 4 gives an example of a rectangular configuration (with the key “5_8”).

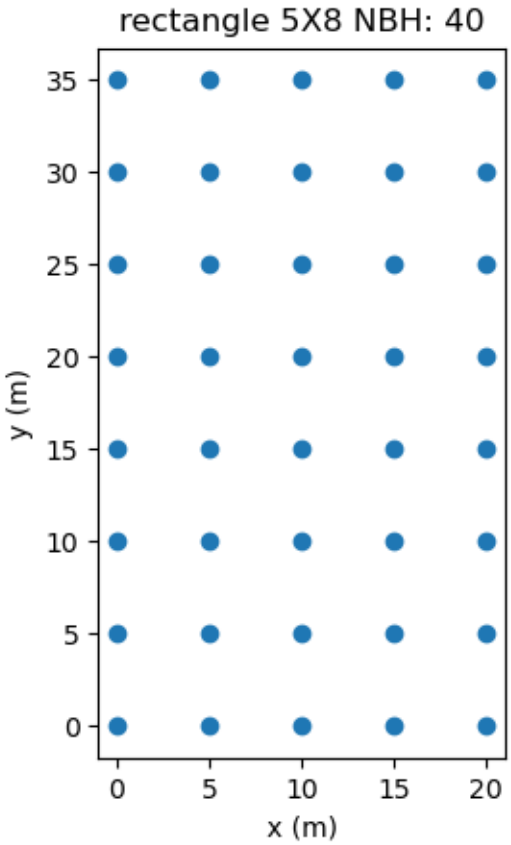

Figure 4 Rectangle Configuration Example

This library contains a variety of “M_N” values which sum up to 1,651 entries. Table 7 details the available configurations in the library.

Table 7 Configurations Available in Rectangle Library

M	N (minimum)	N (maximum)
1	1	100
2	2	100
3	3	100
4	4	100
5	5	100
6	6	100
7	7	100
8	8	100
9	9	100
10	10	100
11	11	93
12	12	85
13	13	78
14	14	73
15	15	68
16	16	64
17	17	60
18	18	56
19	19	53
20	20	51
21	21	48
22	22	46
23	23	44
24	24	42
25	25	40
26	26	39
27	27	37
28	28	36
29	29	35
30	30	34
31	31	33
32	32	32

Appendix 2: Zoned Rectangle Library

This library contains Zoned Rectangles. Zoned Rectangles are rectangularly-shaped, but the inner portions of the rectangle have rows/columns removed and increased spacing. The perimeter boreholes have uniform spacing. The interior boreholes have "bi-uniform" spacing; that is, spacing that is uniform in both directions, but which may be different. The interior spacing is determined to have uniform spacing in each direction. The spacing from the perimeter to the first borehole in a row or column is the same as the spacing between the boreholes in that row or column. Figure 5 shows one example of a zoned rectangle with "5_11" as key 1 and "2_6" as key 2.

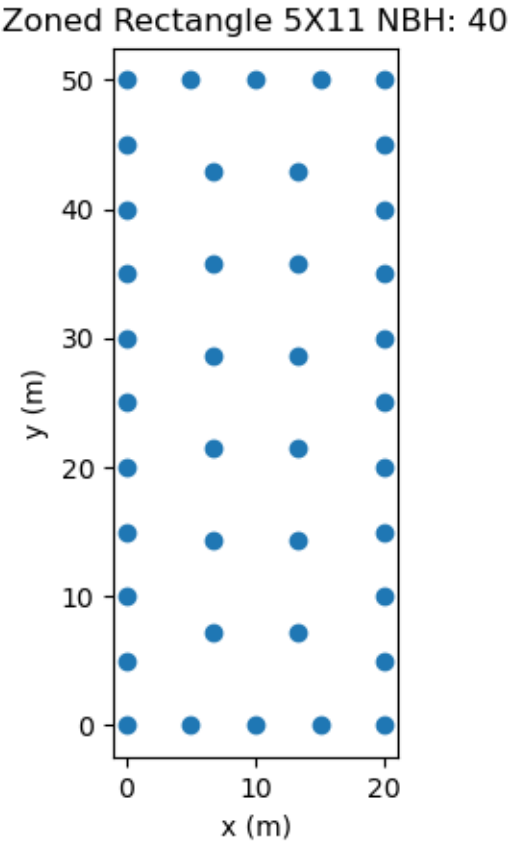

Figure 5 Zoned Rectangle Configuration Example 1 (key 2 = "2_6")

The second key for this configuration is "*Mi_Ni*". *Mi* represents the *M* value of the inner rectangle, and Ni represents the N value of the inner rectangle. Figure 6 has keys of "10_15" and "1_3".

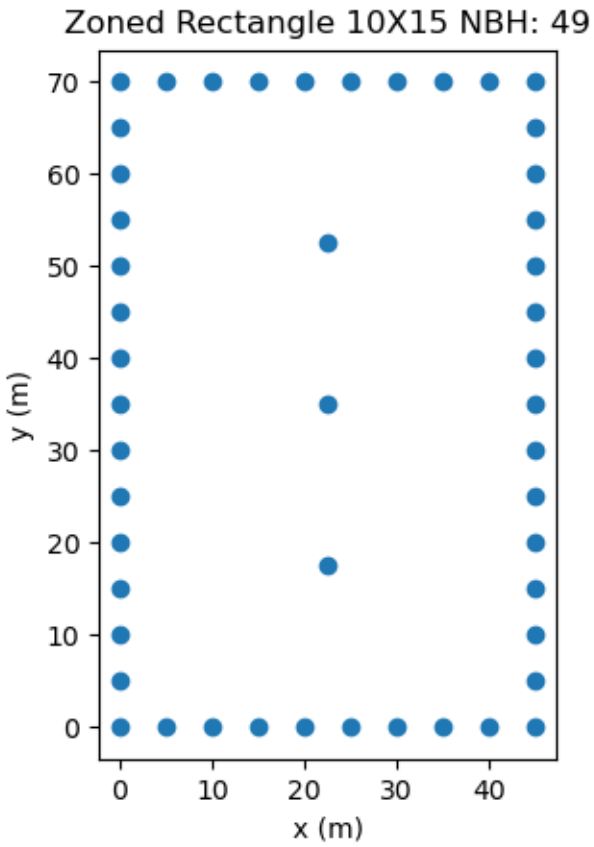

Figure 6 Zoned Rectangle Configuration Example 2 (key 2 = "1_3")

This library has 12,615 configurations. It has *M* values from 4 to 32 (inclusive). For each *M* value, *N* values from that *M* value to 32 are contained (inclusive). For instance, for *M* = 8, there are N values from 8 to 32 (inclusive). For the vast majority of "*M_N*" keys, the "*Mi_Ni*" keys essentially go from "1_1" to one iterative step below what a full interior rectangle would be. The steps add one to either *Ni* or *Mi* while maintaining *Mi* < *Ni*. Not every possible *Ni, Mi* pair is contained in the library. Rather, a stepwise process was used to eliminate rows or columns. At each step, a row or column was eliminated; the decision was based on which elimination would result in the *x* and *y* spacing being closest to uniform. The reader may wish to consult the list of available configurations contained in "ZRectsContained.xlsx".

Appendix 3: Open Rectangle Library

This library contains Open Rectangle configurations. Open Rectangle Configurations in the library have one, two, or three rows of perimeter boreholes. Figure 7 illustrates one example of an Open Rectangle Configuration with a key 1 of "5_10" and key 2 of "2".

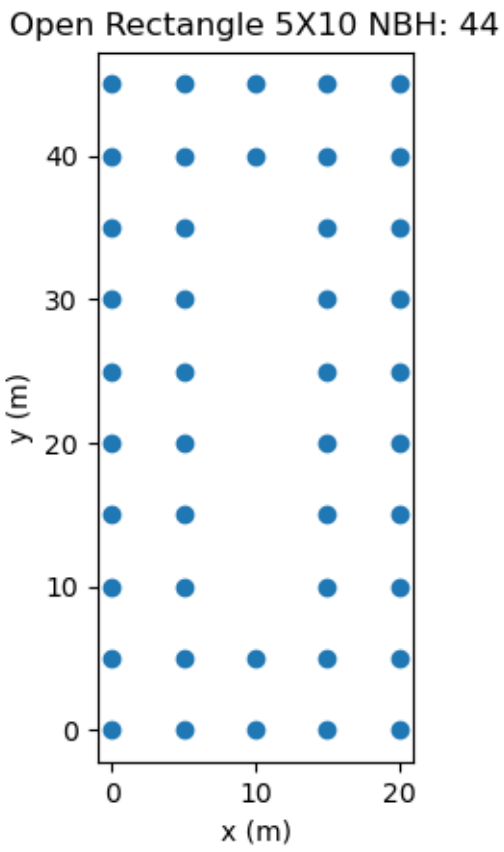

Figure 7 Open Rectangle Configuration Example 1 (key 2 = "2")

The second key for this configuration is "t" where "t" represents the number of rows of perimeter boreholes. A *t* value of 1 would represent a rectangle with only 1 row of perimeter boreholes going around the perimeter of the rectangle. Figure 8 shows a field with keys of "6_17" and "1".

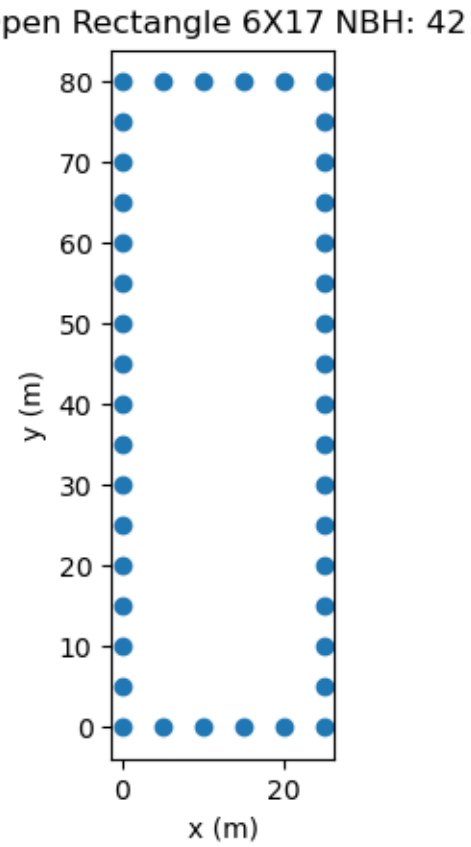

Figure 8 Open Rectangle Configuration Example 2 (key 2 = "1")

This library has 2,332 configurations. It has M values from 3 to 32 (inclusive). For each *M* value, *N* values from that *M* value to 32 are contained (inclusive). For instance, for *M* = 8, there are N values from 8 to 32 (inclusive). The range of "*t*" values change based on how large *M* is. For *M*<5, only *t*=1 is available. For 5<=*M*<7, *t*=1 and *t*=2 is available. For the other *M* values, *t*=1, *t*=2, and *t*=3 are available.

Appendix 4: C Library

This library contains C configurations. C configurations are configurations where some amount of bore holes have been removed from the topside of an Open configuration, but not enough bore holes have been removed to make the configuration a U configuration. Figure 9 and Figure 10 illustrate examples of this type of configuration. The bore holes are removed in such a way as to nearly maintain vertical symmetry. When an odd number of boreholes are removed from an edge with an odd number of boreholes, or an even number are removed from an edge with an even number of boreholes, symmetry is maintained. However, this is not possible in the other cases, as shown in Figure 9 (key 1 = "8_12" and key 2 = "3").

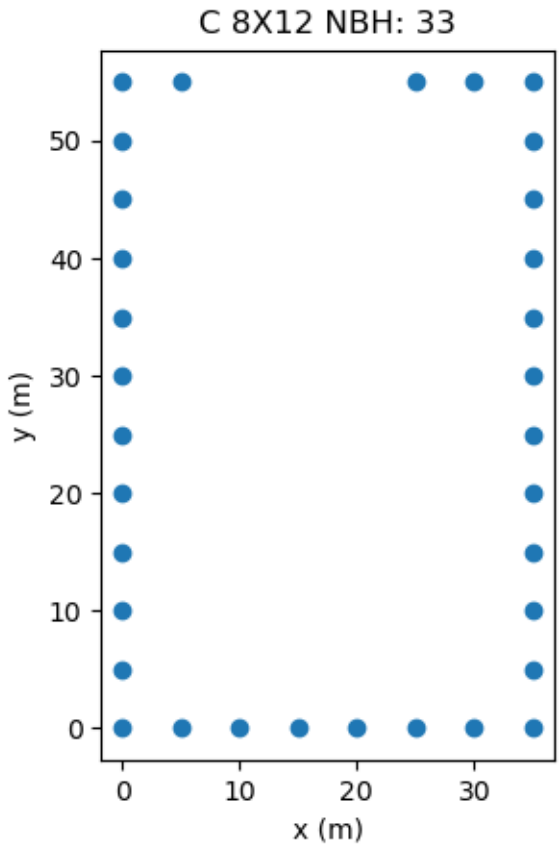

Figure 9 C Configuration Example 1 (key 2 = "3")

The second key for C configurations is "r" where "r" represents the number of boreholes that have been removed from the topside of the configuration. The keys for the configuration in Figure 10 are "10_14" and "7".

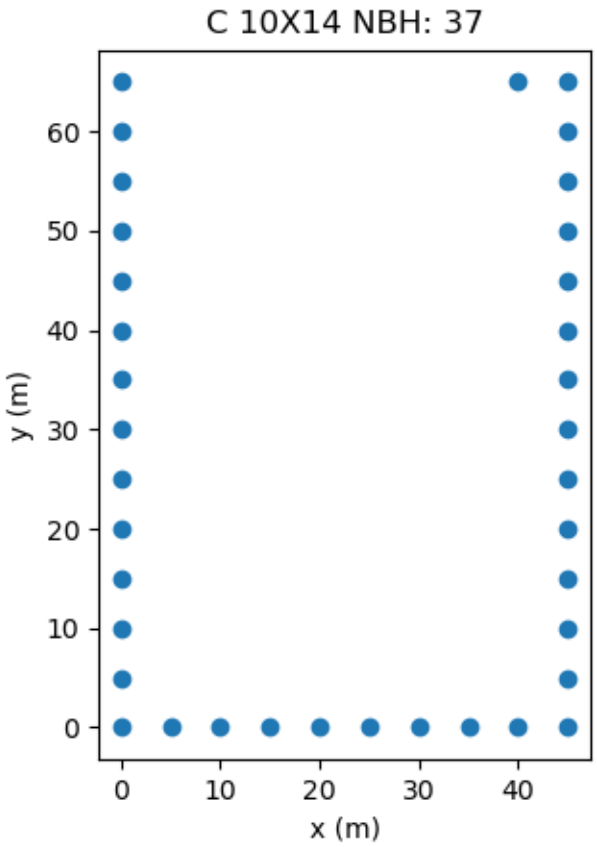

Figure 10 C Configuration Example 2 (key 2 = "7")

This library has 4,525 configurations. It contains *M* values from 3 to 32 (inclusive). For each *M* value, *N* values from that *M* value to 32 are contained (inclusive). For instance, for *M* = 8, there are *N* values from 8 to 32. For a specific "*M*_*N*", the r values range from 1 to *M*-2 or *M*-3 (inclusive). It is *M*-2 for cases where *M* is odd, and *M*-3 for cases where *M* is even.

Appendix 5: L Library

This library contains L Configurations. L configurations are cases where there are lines along the y axis, and the x-axis that meet at the origin. Figure 11 illustrates an example of an L configuration (key 1 = "7_13").

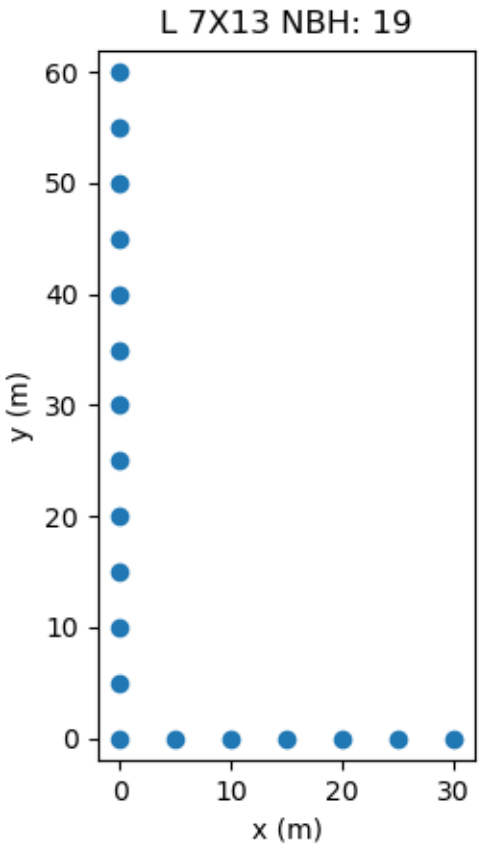

Figure 11 L Configuration Example 1

Currently, there is only one key for the L configurations. The key for the field in Figure 12 is "11_14."

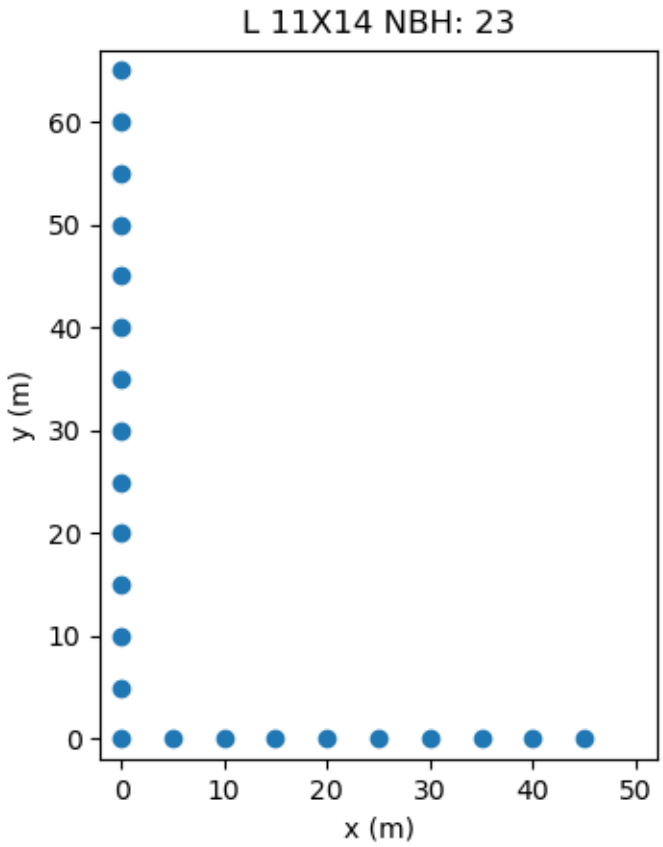

Figure 12 L Configuration Example 2

The library has 495 configurations. It has *M* values from 2 to 32 (inclusive). For each *M* value, *N* values from that *M* value to 32 are contained (inclusive); however, there is not a *N*=2, *M*=2 configuration. For instance, for *M* = 8, there are *N* values from 8 to 32 (inclusive).

Appendix 6: U Library

This library contains U configurations. Figure 13 shows an example of a U configuration (where key 1 = “5_10” and key 2 = “1”).

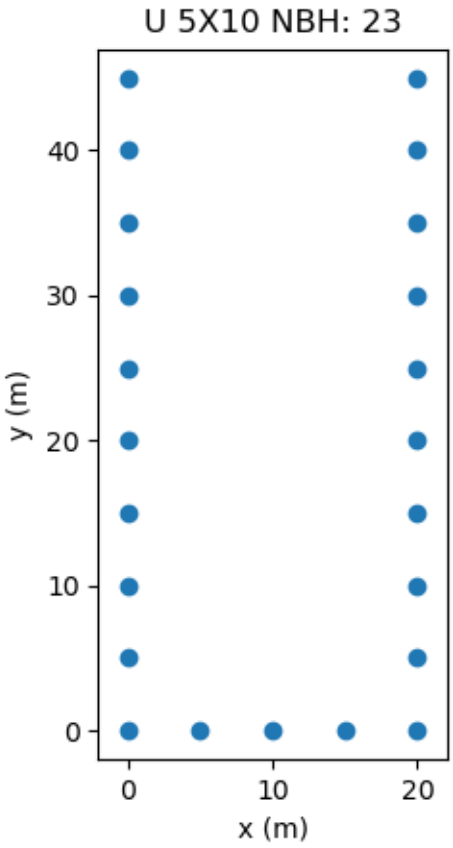

Figure 13 U Configuration Example 1 (key 2 = “1”)

The second key for this configuration is “*t*” where “*t*” represents the number of perimeter borehole rows. A t value of 1 would represent one row of perimeter boreholes, as shown in Figure 13. A *t* value of 3 would represent 3 rows of perimeter boreholes, as shown in Figure 141. The keys for the field in

Figure 11 are “8_11” and “3”.

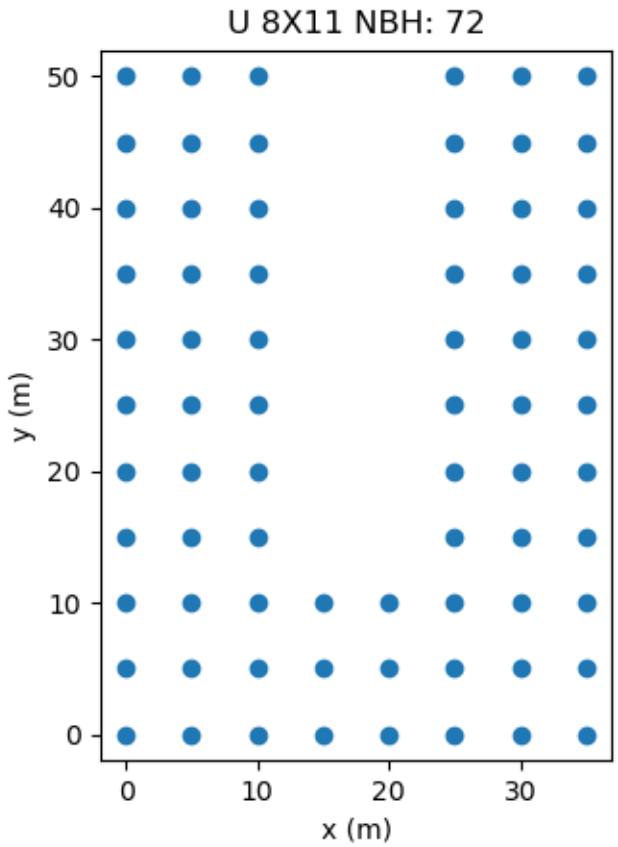

Figure 14 U Configuration Example 2 (key 2 = “3”)

This library has 3, 248 configurations. It has *M* values from 3 to 50 (inclusive). For each *M* value, *N* values from that *M* value to 50 are contained (inclusive). For instance, for *M* = 8, there are *N* values from 8 to 50 (inclusive). The “*t*” values available for the library consist of only “1” for *M* < 5, “1” and “2” for 5 <= *M* < 7, and “1,” “2,” and “3” for *M* >= 7.

Appendix 7: LopU Library

This library contains “LopU”(Lopsided U) configurations. LopU configurations consist of U configurations that have had points removed from the right side. Figure 15 shows an example of a LopU Configuration (key 1 = “5_10” and key 2 = “6”).

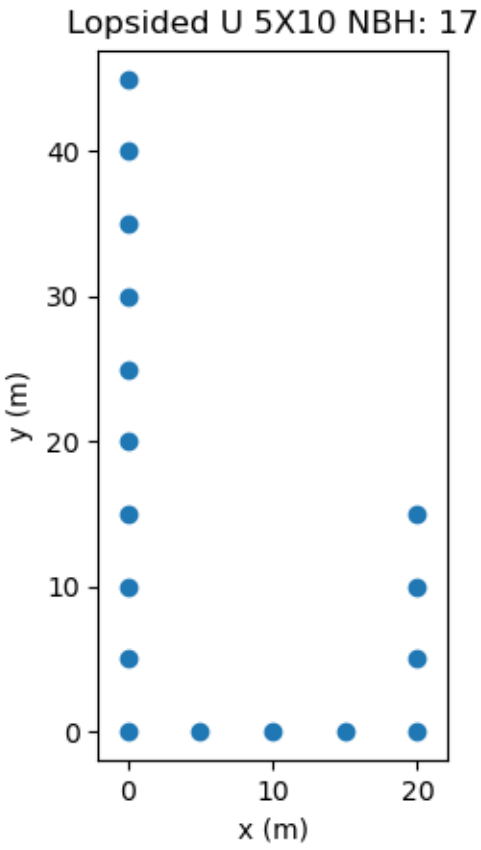

Figure 15 LopU Configuration Example 1 (key 2 = “6”)

The second key for this configuration is “*r*” where “*r*” represents the number of holes removed from the right side of a “U” configuration. The keys for the field illustrated in Figure 13 are “7_20” and “13”.

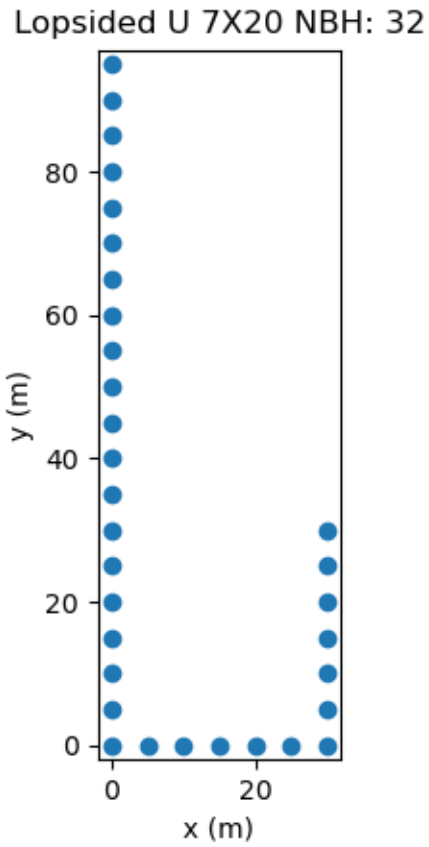

Figure 16 LopU Configuration Example 2 (key 2 = "13")

The library has 9,455 configurations. It has *M* values from 3 to 32 (inclusive). For each *M* value, N values from that *M* value to 32 are contained (inclusive). For instance, for *M* = 8, there are *N* values from 8 to 32 (inclusive). For a specific "M_*N*", the r values go between 1 and N-2 (inclusive).

Appendix 8: Library Python Example

```
import json
import csv
import matplotlib.pyplot as plt


def main ():

    #EXAMPLE FROM RECTANGLE LIBRARY

    #load library file using the standard JSON python library
    rectLibrary = None
    with open("rectangle_5m_v1.0.json",'r') as libraryFile:
        rectLibrary = json.load(libraryFile)

    #defining m and n values (remember that n<m)
    n = 15
    m = 8

    #making key
    key1 = "{m:d}_{n:d}".format(m=m,n=n)

    #getting specific configuration from the library
    configuration = rectLibrary[key1]

    #Getting values of configuration
    gVals = configuration["g"]
    lntVals = configuration["logtime"]
    boreHoleLocations = configuration["bore_locations"]

    #Graphing Bore Hole Locations

    #Separating the points into x and y coordinates
    boreHoleX = [boreHoleLocation[0] for boreHoleLocation in
boreHoleLocations]
    boreHoleY = [boreHoleLocation[1] for boreHoleLocation in
boreHoleLocations]

    #Setting title for Graphs and csv file
    title = "Rectangular {n:d}X{m:d} Field".format(n=n,m=m)

    #plot the bore hole locations
    plt.scatter(boreHoleX,boreHoleY)
    plt.title( title)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.xlabel("Distance (m)")
    plt.ylabel("Distance (m)")
    plt.show()

    #Getting a Specific G-Function for the configuration
    B = 5
    H=96
```

```
    rb = .075
    gFunc = gVals["{B:d}_{H:d}_{r:.3f}".format(B=B,H=H,r=rb)]

    #Graphing G-Function
    plt.clf()
    plt.title( title + " G-Function for H = {H:d}m".format(H=H))
    plt.xlabel("ln(t/ts)")
    plt.ylabel("g")
    plt.plot(lntVals,gFunc)
    plt.show()

    #Saving the G-Function to file
    with open( title + ".csv","w",newline="") as outputFile:
        cW = csv.writer(outputFile)
        cW.writerow(["ln(t/ts)","g"])
        for i in range(len(lntVals)):
            cW.writerow([lntVals[i],gFunc[i]])
```

```
    # Zoned Rectangle EXAMPLE

    # This is example is to help demonstrate how to use the libraries that
require tow keys to access a configuration

    # load library file using the standard JSON python library
    Library = None
    with open("zoned_rectangle_5m_v1.0.json", 'r') as libraryFile:
       Library = json.load(libraryFile)

    # defining m and n values (remember that n<m)
    n = 17
    m = 13
    mi = 6
    ni = 8

    # making keys (there are two for this library)
    key1 = "{m:d}_{n:d}".format(m=m,n=n)

    # This second key will very based on the library being accessed (check
the library overview document for more information)
    key2 = "{mi:d}_{ni:d}".format(mi=mi,ni=ni)

    # getting specific configuration from the library
    configuration = Library[key1][key2]

    # Getting values of configuration
    gVals = configuration["g"]
    lntVals = configuration["logtime"]
    boreHoleLocations = configuration["bore_locations"]
```

```
    # Graphing Bore Hole Locations

    # Separating the points into x and y coordinates
    boreHoleX = [boreHoleLocation[0] for boreHoleLocation in
boreHoleLocations]
    boreHoleY = [boreHoleLocation[1] for boreHoleLocation in
boreHoleLocations]

    # Setting title for Graphs and csv file
    title = "Zoned Rectangle {n:d}X{m:d} {ni:d}X{mi:d} Field".format(n=n,
m=m, ni=ni,mi=mi)

    # plot the bore hole locations
    plt.scatter(boreHoleX, boreHoleY)
    plt.title(title)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.xlabel("Distance (m)")
    plt.ylabel("Distance (m)")
    plt.show()

    # Getting a Specific G-Function for the configuration
    B = 5
    H = 384
    r = .0875
    gFunc = gVals["{B:d}._{H:d}._{r:.4f}".format(B=B, H=H, r=r)]

    # Graphing G-Function
    plt.clf()
    plt.title( title + " G-Function for H = {H:d}m".format(H=H))
    plt.xlabel("ln(t/ts)")
    plt.ylabel("g")
    plt.plot(lntVals, gFunc)
    plt.show()

    # Saving the G-Function to file
    with open( title + ".csv","w",newline="") as outputFile:
        cW = csv.writer(outputFile)
        cW.writerow(["ln(t/ts)", "g"])
        for i in range(len(lntVals)):
            cW.writerow([lntVals[i], gFunc[i]])




    # Open Rectangle EXAMPLE

    # This is example is to help demonstrate how to use the libraries that
require tow keys to access a configuration

    # load library file using the standard JSON python library
    Library = None
```

```
    with open("Open_configurations_5m_v1.0.json", 'r') as libraryFile:
        Library = json.load(libraryFile)

    # defining m and n values (remember that n<m)
    n = 20
    m = 8
    t = 1

    # making keys (there are two for this library)
    key1 = "{m:d}_{n:d}".format(m=m,n=n)

    # This second key will very based on the library being accessed (check
the library overview document for more information)
    key2 = "{t:d}".format(t=t)

    # getting specific configuration from the library
    configuration = Library[key1][key2]

    # Getting values of configuration
    gVals = configuration["g"]
    lntVals = configuration["logtime"]
    boreHoleLocations = configuration["bore_locations"]

    # Graphing Bore Hole Locations

    # Separating the points into x and y coordinates
    boreHoleX = [boreHoleLocation[0] for boreHoleLocation in
boreHoleLocations]
    boreHoleY = [boreHoleLocation[1] for boreHoleLocation in
boreHoleLocations]

    # Setting title for Graphs and csv file
    title =  "Open (Rectangle) {n:d}X{m:d} Field that is  {t:d}
Thick".format(n=n, m=m, t=t)

    # plot the bore hole locations
    plt.scatter(boreHoleX, boreHoleY)
    plt.title(title)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.xlabel("Distance (m)")
    plt.ylabel("Distance (m)")
    plt.show()

    # Getting a Specific G-Function for the configuration
    B = 5
    H = 192
    r = .08
    gFunc = gVals["{B:d}._{H:d}._{r:.2f}".format(B=B, H=H, r=r)]

    # Graphing G-Function
    plt.clf()
    plt.title( title + " G-Function for H = {H:d}m".format(H=H))
    plt.xlabel("ln(t/ts)")
```

```
    plt.ylabel("g")
    plt.plot(lntVals, gFunc)
    plt.show()

    # Saving the G-Function to file
    with open( title + ".csv","w",newline="") as outputFile:
        cW = csv.writer(outputFile)
        cW.writerow(["ln(t/ts)", "g"])
        for i in range(len(lntVals)):
            cW.writerow([lntVals[i], gFunc[i]])







    # C EXAMPLE

    # This is example is to help demonstrate how to use the libraries that
require tow keys to access a configuration

    # load library file using the standard JSON python library
    Library = None
    with open("C_configurations_5m_v1.0.json", 'r') as libraryFile:
        Library = json.load(libraryFile)

    # defining m and n values (remember that n<m)
    n = 25
    m = 13
    t = 3

    # making keys (there are two for this library)
    key1 = "{m:d}_{n:d}".format(m=m,n=n)

    # This second key will very based on the library being accessed (check
the library overview document for more information)
    key2 = "{t:d}".format(t=t)

    # getting specific configuration from the library
    configuration = Library[key1][key2]

    # Getting values of configuration
    gVals = configuration["g"]
    lntVals = configuration["logtime"]
    boreHoleLocations = configuration["bore_locations"]

    # Graphing Bore Hole Locations

    # Separating the points into x and y coordinates
    boreHoleX = [boreHoleLocation[0] for boreHoleLocation in
boreHoleLocations]
```

```
    boreHoleY = [boreHoleLocation[1] for boreHoleLocation in
boreHoleLocations]

    # Setting title for Graphs and csv file
    title =  "C {n:d}X{m:d} Field with {t:d} Removed".format(n=n, m=m, t=t)

    # plot the bore hole locations
    plt.scatter(boreHoleX, boreHoleY)
    plt.title(title)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.xlabel("Distance (m)")
    plt.ylabel("Distance (m)")
    plt.show()

    # Getting a Specific G-Function for the configuration
    B = 5
    H = 48
    r = .075
    gFunc = gVals["{B:d}._{H:d}._{r:.3f}".format(B=B, H=H, r=r)]

    # Graphing G-Function
    plt.clf()
    plt.title( title + " G-Function for H = {H:d}m".format(H=H))
    plt.xlabel("ln(t/ts)")
    plt.ylabel("g")
    plt.plot(lntVals, gFunc)
    plt.show()

    # Saving the G-Function to file
    with open( title + ".csv","w",newline="") as outputFile:
        cW = csv.writer(outputFile)
        cW.writerow(["ln(t/ts)", "g"])
        for i in range(len(lntVals)):
            cW.writerow([lntVals[i], gFunc[i]])




    # L EXAMPLE

    # This is example is to help demonstrate how to use the libraries that
require tow keys to access a configuration

    # load library file using the standard JSON python library
    Library = None
    with open("L_configurations_5m_v1.0.json", 'r') as libraryFile:
        Library = json.load(libraryFile)

    # defining m and n values (remember that n<m)
    n = 28
    m = 28
```

```
    # making keys (there is only one for this library)
    key1 = "{m:d}_{n:d}".format(m=m,n=n)


    # getting specific configuration from the library
    configuration = Library[key1]

    # Getting values of configuration
    gVals = configuration["g"]
    lntVals = configuration["logtime"]
    boreHoleLocations = configuration["bore_locations"]

    # Graphing Bore Hole Locations

    # Separating the points into x and y coordinates
    boreHoleX = [boreHoleLocation[0] for boreHoleLocation in
boreHoleLocations]
    boreHoleY = [boreHoleLocation[1] for boreHoleLocation in
boreHoleLocations]

    # Setting title for Graphs and csv file
    title = "L {n:d}X{m:d} Field".format(n=n, m=m, r=r)

    # plot the bore hole locations
    plt.scatter(boreHoleX, boreHoleY)
    plt.title(title)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.xlabel("Distance (m)")
    plt.ylabel("Distance (m)")
    plt.show()

    # Getting a Specific G-Function for the configuration
    B = 5
    H = 24
    r = .075
    gFunc = gVals["{B:d}._{H:d}._{r:.3f}".format(B=B, H=H, r=r)]

    # Graphing G-Function
    plt.clf()
    plt.title( title + " G-Function for H = {H:d}m".format(H=H))
    plt.xlabel("ln(t/ts)")
    plt.ylabel("g")
    plt.plot(lntVals, gFunc)
    plt.show()

    # Saving the G-Function to file
    with open( title + ".csv","w",newline="") as outputFile:
        cW = csv.writer(outputFile)
        cW.writerow(["ln(t/ts)", "g"])
        for i in range(len(lntVals)):
            cW.writerow([lntVals[i], gFunc[i]])
```

```
    #U EXAMPLE

    # This is example is to help demonstrate how to use the libraries that
require tow keys to access a configuration

    # load library file using the standard JSON python library
    Library = None
    with open("U_configurations_5m_v1.0.json", 'r') as libraryFile:
        Library = json.load(libraryFile)

    # defining m and n values (remember that n<m)
    n = 24
    m = 15
    t = 2

    # making keys (there are two for this library)
    key1 = "{m:d}_{n:d}".format(m=m,n=n)

    # This second key will very based on the library being accessed (check
the library overview document for more information)
    key2 = "{t:d}".format(t=t)

    # getting specific configuration from the library
    configuration = Library[key1][key2]

    # Getting values of configuration
    gVals = configuration["g"]
    lntVals = configuration["logtime"]
    boreHoleLocations = configuration["bore_locations"]

    # Graphing Bore Hole Locations

    # Separating the points into x and y coordinates
    boreHoleX = [boreHoleLocation[0] for boreHoleLocation in
boreHoleLocations]
    boreHoleY = [boreHoleLocation[1] for boreHoleLocation in
boreHoleLocations]

    # Setting title for Graphs and csv file
    title = "U {n:d}X{m:d} Field that is  {t:d} thick".format(n=n, m=m, t=t)

    # plot the bore hole locations
    plt.scatter(boreHoleX, boreHoleY)
    plt.title(title)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.xlabel("Distance (m)")
    plt.ylabel("Distance (m)")
    plt.show()
```

```
    # Getting a Specific G-Function for the configuration
    B = 5
    H = 192
    r = .08
    gFunc = gVals["{B:d}_{H:d}_{r:.2f}".format(B=B, H=H, r=r)]

    # Graphing G-Function
    plt.clf()
    plt.title( title + " G-Function for H = {H:d}m".format(H=H))
    plt.xlabel("ln(t/ts)")
    plt.ylabel("g")
    plt.plot(lntVals, gFunc)
    plt.show()

    # Saving the G-Function to file
    with open( title + ".csv","w",newline="") as outputFile:
        cW = csv.writer(outputFile)
        cW.writerow(["ln(t/ts)", "g"])
        for i in range(len(lntVals)):
            cW.writerow([lntVals[i], gFunc[i]])
```

```
    # LopU EXAMPLE

    # This is example is to help demonstrate how to use the libraries that
require tow keys to access a configuration

    # load library file using the standard JSON python library
    LopULibrary = None
    with open("LopU_configurations_5m_v1.0.json", 'r') as libraryFile:
        LopULibrary = json.load(libraryFile)

    # defining m and n values (remember that n<m)
    n = 24
    m = 15
    r = 5

    # making keys (there are two for this library)
    key1 = "{m:d}_{n:d}".format(m=m,n=n)

    # This second key will very based on the library being accessed (check
the library overview document for more information)
    key2 = "{r:d}".format(r=r)

    # getting specific configuration from the library
    configuration = LopULibrary[key1][key2]
```

```
    # Getting values of configuration
    gVals = configuration["g"]
    lntVals = configuration["logtime"]
    boreHoleLocations = configuration["bore_locations"]

    # Graphing Bore Hole Locations

    # Separating the points into x and y coordinates
    boreHoleX = [boreHoleLocation[0] for boreHoleLocation in
boreHoleLocations]
    boreHoleY = [boreHoleLocation[1] for boreHoleLocation in
boreHoleLocations]

    # Setting title for Graphs and csv file
    title = "LopU {n:d}X{m:d} Field with {r:d} Removed".format(n=n, m=m, r=r)

    # plot the bore hole locations
    plt.scatter(boreHoleX, boreHoleY)
    plt.title(title)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.xlabel("Distance (m)")
    plt.ylabel("Distance (m)")
    plt.show()

    # Getting a Specific G-Function for the configuration
    B = 5
    H = 192
    r = .08
    gFunc = gVals["{B:d}._{H:d}._{r:.2f}".format(B=B, H=H, r=r)]

    # Graphing G-Function
    plt.clf()
    plt.title( title + " G-Function for H = {H:d}m".format(H=H))
    plt.xlabel("ln(t/ts)")
    plt.ylabel("g")
    plt.plot(lntVals, gFunc)
    plt.show()

    # Saving the G-Function to file
    with open( title + ".csv","w",newline="") as outputFile:
        cW = csv.writer(outputFile)
        cW.writerow(["ln(t/ts)", "g"])
        for i in range(len(lntVals)):
            cW.writerow([lntVals[i], gFunc[i]])
```

```
if __name__ == "__main__":
    main()
```