Standardized Schoeller Diagrams with Matlab

A User Manual

Alexander Dietz and Rafael Schäffer

December 14, 2023

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Mathematics
Institute of Applied Geosciences

Contents

Abstract

Water analyses typically result in numerous characteristic values. The bunch of parameters hamper temporal or spatial comparisons of different samples. In order to facilitate the evaluation and interpretation of hydrochemical data, hydrogeologists use various special diagrams. One of them is the Schoeller diagram, in which the concentrations of different chemical species are plotted on logarithmic scales. Entries of an analysis are connected and form a characteristic signature. In the Schoeller diagram, parallel or subparallel signatures indicate relatedness of waters. Here we present the idea of standardized Schoeller diagrams: different logarithmic axes are shifted with respect to each other so that the signatures of a selectable sample forms a straight line.

In this manual, we describe in detail how to use the matlab file to create (standardized) Schoeller diagrams. We also explain how data can be imported into matlab, how that plot can be customized, and how they can be exported to images or vector graphics, useful for presentations or publications.

Contact & Licence

For any suggestions concerning the code or the manual you can write to

dietz@mathematik.tu-darmstadt.de

and for any suggestions concerning (standardized) Schoeller diagrams to

rafael.schaeffer@tu-darmstadt.de

Schäffer, R., Dietz, A. (2023): Standardized Schoeller diagrams — A Matlab plotting tool. Grundwasser 28, 345–355. https://doi.org/10.1007/s00767-023-00556-3

If it is necessary to cite the matlab code, or the manual itself, you can cite the data repository [1].

Acknowledgement

We would like to thank Cäcilia Boller and Professor Dr. Ulrich Reif for fruitful discussions and careful proofreading.

Example Graphics

All graphics in this manual are created with the plotting tool out of the file ExampleDataSet.xlsx. At any point, the code lines of this manual are used with the addition

```
plotSchoeller([...],'LineWidth',2,...
'CreateFigure','Print','./exampleX','-depsc','FigureSize',[640,480])
```

to highlight the lines, export it to a .eps file, and put it into an adjusted shape for this document. You can find more information about the line options in section 4.12, about exporting in section 4.11, and about the scaling in section 4.10. Especially the left example on each figure is created with the command

```
plotSchoeller('ExampleDataSet.xlsx',1,'LineWidth',2,...
'CreateFigure','Print','./example1','-depsc','FigureSize',[640,480])
```

The full command and the additional plot options are not added in the example code lines to get a clearer notation and to focus on the part command itself. You can find all commands in the

```
createAllFigures.m
```

file and reproduce it as you like.

1 Overlay

The plot tool is an easy way to plot Schoeller diagrams. In this manual, we explain the tool for matlab beginners and give additional hints for advantaged use. Nevertheless, we assume some basic skills like navigating into another folder and the use of the command window to be known. For a fundamental explanation into matlab we recommend [2]. To use the plot tool, you have to do these previous steps:

1. Download the plotSchoeller.m file [1].
2. Copy it into a folder of your choice (for beginners we recommend the folder of your data sheet).
3. Start matlab and navigate to your folder.

The plot command consists of three main components: the data, the pivot line, and optional several plotting commands. So the stencil for this command looks like the following:

```
plotSchoeller(data,pivot,plotoption)
```

There are two ways to use this plot command. The first one is to write the command directly into the command window. This way can be used to generate a graphic in a quick way. The second is to generate a script with the button 'New Script' and write the plot command into the script. This method has some advantages. One of them is, that at first the plot command can get quite long and therefore incomprehensible. As a solution within the script, the command can be split into several lines like

```
plotSchoeller(...
    data,...
   pivot,...
 plotoption...
      )
```

Also, a script can be saved and edited, wherefore you can save your plot command for later uses. You can also add more commands to the script to edit your plot afterwards. To run such a script, you have to save it and then write the saved name into the command window. If your saved file is called 'myPlot1.m', you have to write

```
myPlot1
```

into the command line. In the next sections we explain the plot components and start with the data table. To get a quick guide of the possibilities in matlab, you can use the command

```
help plotSchoeller
```

2 Parse Your Data Sheet

First of all, we describe how to parse your data sheet in this section. For the plot tool, there are two different ways to read your data sheet.
The first one (which we recommend for beginners) is, to read the data table from an external file like .txt, .csv, .ods or .xlsx. For this option you have to put the data path into the first command option. For example

```
plotSchoeller('ExampleDataSet.xlsx',pivot,plotoption)
```

if the file ExampleDataSet.xlsx is in the same directory as the plotSchoeller.m file, or

```
plotSchoeller('c:/myFolder/ExampleDataSet.xlsx',pivot,plotoption)
```

for a file in any directory. Notice that it is mandatory to put the name or data path of the file into single quotes ' '. As the import of your data sheet into matlab might be the hardest part of using this tool, it is important to parse your sheet in a proper way. Your file has to fulfill the following requirements:

1. The entries of the first line correspond to the name of the columns. This name is a technical name to address the columns. They could be set arbitrarily and will not appear in the plot.
2. The entries in the second line correspond to the captions of the columns printed in the plot. You can use TEX code here.
3. The entries in the first column correspond to the name (and if no second column for description is used, also for the description) of one sample.
4. The entries in the second column can have two abilities. The first option is to use it as the first column of the data entries. The second option is to use it for a description of one sample. One advantage of the description line is that you can use shorter names to address one sample, and the second is, that you can use special fonts for the legend. **If you want to use the second column as a description line, it is mandatory to keep the cell B2 (the cell in the second line and the second column) empty.**
5. Any other column corresponds to one element, and any other line to one sample. You can add unbounded many columns for elements and lines for samples.
6. Any line of the table has to be a line in the file.
7. Not every cell needs a value. If there is no value, or the value is equal to zero, the value will be set by default as < 0.01 in the plot. You can adjust the value of 0.01 within a plot option. More to this topic is available in section 4.5.

If you are using the .csv file format, you have to mention:

1. Every entry has to be separated with a tab.
2. Any floating point number has to be written with a dot. So if you use another language (e.g. German), be sure that your numbers are written with a dot, not with a comma.

And if you are using the .xlsx format, you have to mention:

1. Any number cell has to be formatted as a number cell (and not as an text or string cell). Here it is not necessary to write it with a dot. The German language style is therefore possible here. You can check whether your cell is formatted as a number or a text by the text alignment. Typically, a right text alignment means a number and a left alignment means a text or string.

As a simple example for this manual, we use the following tables

Element Name	NH4+	K+	Mg2+	TEC	TDS
Element Caption	\bf NH_{4}^{+} \rm \newline (mg/L)	\bf K^{+} \rm \newline (mg/L)	\bf Mg^{2+} \rm \newline (mg/L)	\bf TEC \rm \newline (mmol(eq)/L)	\bf TDS \rm \newline (mg/L)
\bf Sprudel-1 1898		862.5	423	1048.5	34980
\bf Sprudel-2 1901	3.51	548.7	377.8	938.2	30940.2
\bf Huttenquelle 1886		743.2	255.8	609.1	19959.8
\bf Ottoquelle 1886	6.18	160	161.5	643.51	21236.1
\bf Ottoquelle 1957		57.3	22.1		3406.9

Table 1: Example data without a sample caption column.

Element Name	Sample Caption	NH4+	K+	Mg2+	TEC	TDS
Element Caption		\bf NH_{4}^{+} \rm \newline (mg/L)	\bf K^{+} \rm \newline (mg/L)	\bf Mg^{2+} \rm \newline (mg/L)	\bf TEC \rm \newline (mmol(eq)/L)	\bf TDS \rm \newline (mg/L)
S1	\bf Sprudel-1 1898		862.5	423	1048.5	34980
S2	\bf Sprudel-2 1901	3.51	548.7	377.8	938.2	30940.2
H1	\bf Huttenquelle 1886		743.2	255.8	609.1	19959.8
O1	\bf Ottoquelle 1886	6.18	160	161.5	643.51	21236.1
O2	\bf Ottoquelle 1957		57.3	22.1		3406.9

Table 2: Example data with a sample caption column.

Concerning the caption, we recommend the following TEX stencil

```
\bf  NH_{4}^{+}\rm\newline (mg/L)
```

In detail, the commands means: [3],[4]

- `\bf` writes the following in bold letters
- `_{}` creates an lower index
- `^{}` creates an upper index
- `\rm` writes the following in normal letters
- `\newline` creates a new line

Additional useful commands are:

- `\fontname{}` changes the font of the text
- `\fontsize{}` changes the font size of the text
- `\color{}` changes the color of the text

For more additional commands, we recommend [3]. To put the first line in the middle of the column caption, it could help to add additional spaces in the first line. You can use the upper commands either to adjust the second line (to change the caption of each column) or to adjust the second column (to change the style of the legend entries).

Alternatively to this method, you can generate a DataTable directly in Matlab and put it into the first argument. The command looks like the following:

```
plotSchoeller(myDataTable,pivot,plotoption)
```

3 The Pivot Element

The second entry of the plot command is the pivot element. If this entry is set to zero, as in the following example, with command

```
plotSchoeller(data,0,plotoption)
```

the program plots a usual Schoeller diagram. If the number n is greater than zero, the program plots a standardized Schoeller diagram with the pivot line n. This means, that the data set of line n is plotted as a horizontal line and any other line is plotted relative to this. If n is greater than the amount of lines in the data table, the program throws an error. An example for a standardized Schoeller diagram command is

```
plotSchoeller(data,1,plotoption)
```

You can see the difference in figure 1. If any cell has no entry, the line is set to the floor of the plot (e.g. in figure 1 (left) for the NH_4^+ column). If the pivot line has no entry in a cell, the lower bound of < 0.01 is set up to the constant line (e.g. in figure 1 (right) for the NH_4^+ column). The value, of these data point is

$$\text{lowerDisplayFactor} \cdot \text{lowerBound} = 0.5 \cdot 0.01 = 0.005.$$

You can find more information about these two values in section 4.5 and 4.8.

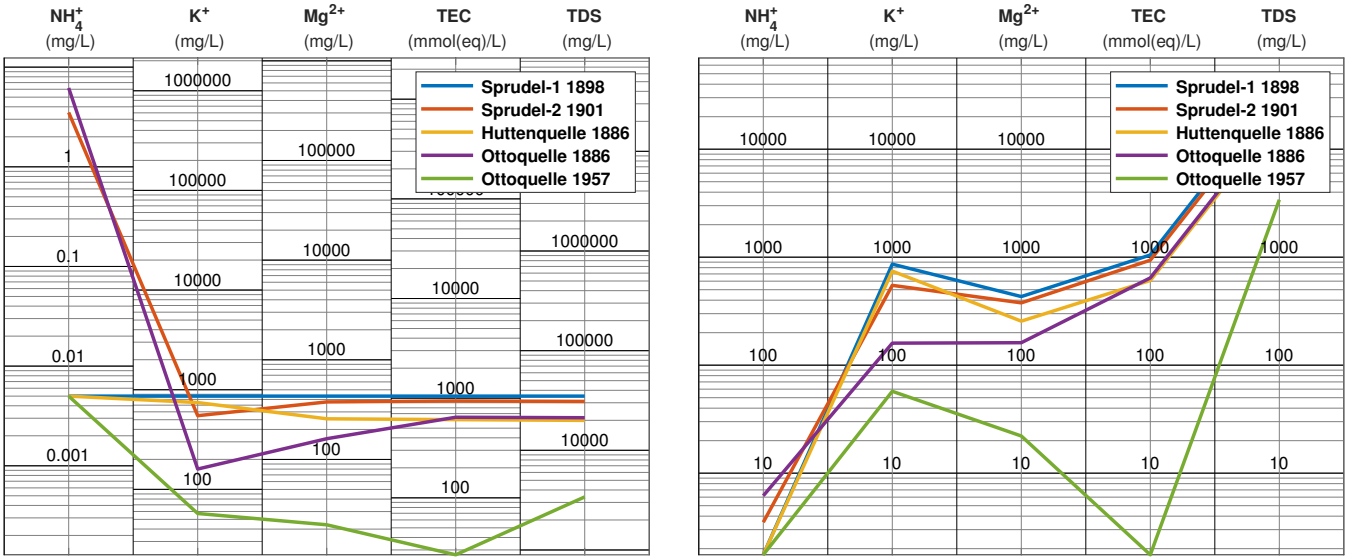

Figure 1: Two Schoeller diagrams with pivot element 1 (left) and pivot element 0 (right).

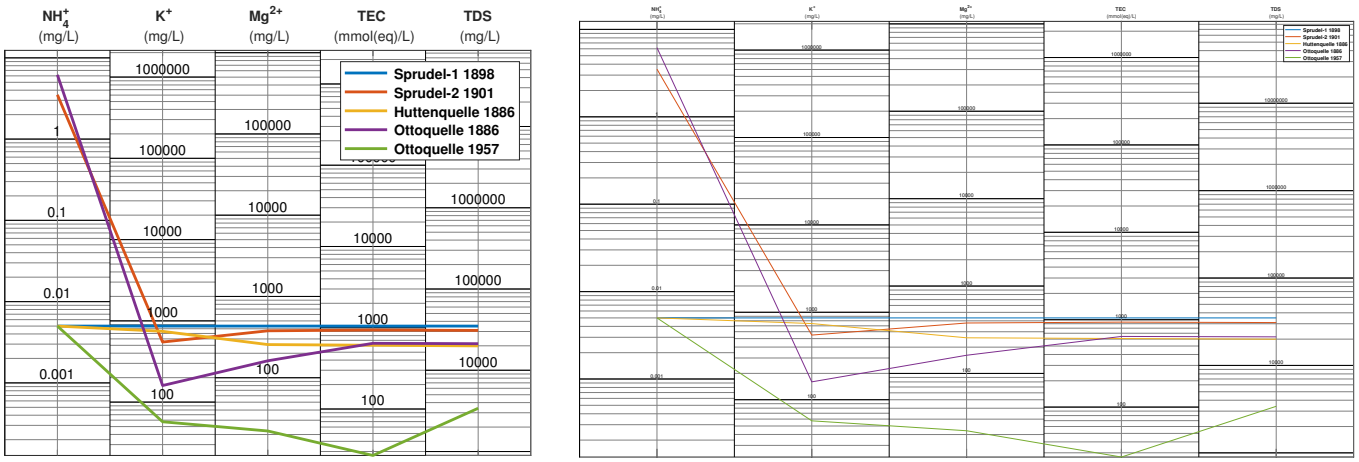

Figure 2: Two Schoeller diagrams with optimized plot options for this manual (left) and with no plot options (right).

4 The Plot Commands

To customize the plots, we provide a bunch of plot options. You can combine any of this commands as you like. The structure behind the commands is the following: Most plot option command uses a key word and then one or two values. An example code for this is

```
plotSchoeller(data,pivot,'plotoption1',1,'plotoption2',3,'plotoption3',7)
```

As you can see, you can string together as much plot option commands, as you like in arbitrary order. Of course you can also use no plot command. The example Matlab code for this is just

```
plotSchoeller(data,pivot)
```

For any of these plot options, there is a standard value. Therefore, if you wish not to customize the plot option, the program will always use the standard option. You can see the difference in figure 2. In the following subsections we explain the different plot options and give examples on how to use them.

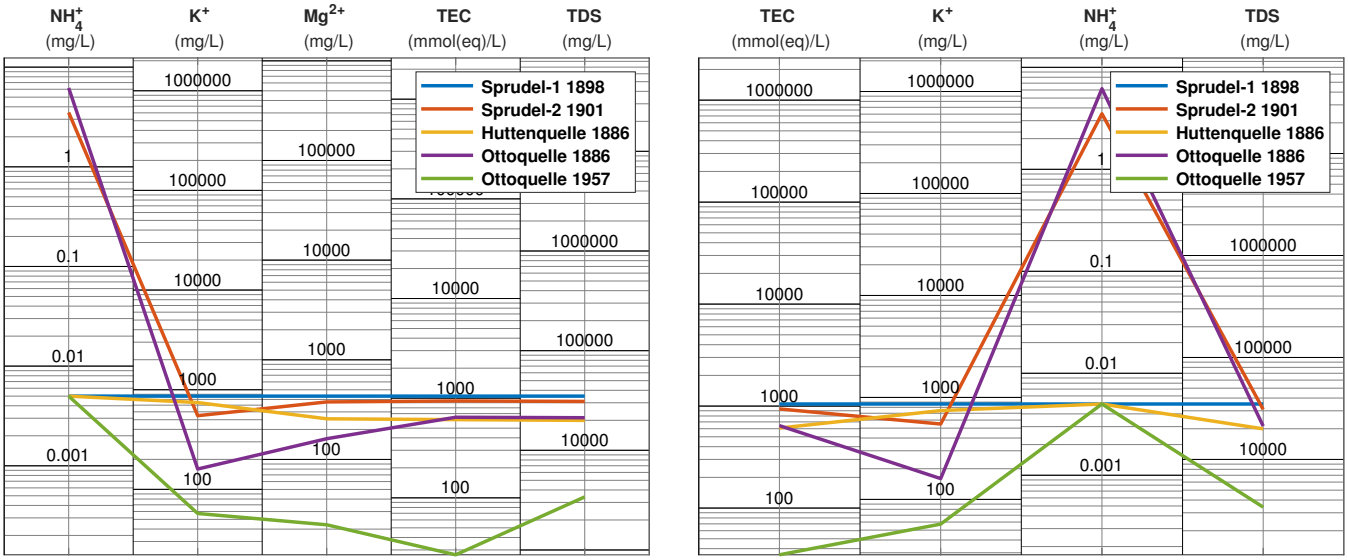

Figure 3: Two Schoeller diagrams with columns as in the data table (left) and adjusted columns (right).

4.1 Columns

The first option, that you can manipulate, is the columns displayed and the order of them. If you do not manipulate this value, the program will plot any column in the order of the given data table or data sheet. If you want to change the sequence or if you want to display just a few of the columns, you can use this command

```
plotSchoeller(data,pivot,'Columns',{'Column1','Column2',...,'ColumnLast'})
```

or as an concrete example,

```
plotSchoeller(data,pivot,'Columns',{'TEC','K+','NH4+','TDS'})
```

As you can see, the key word is `'Columns'` and the value is

```
{'TEC','K+','NH4+','TDS'}.
```

In the new plot, only the columns TEC, K+, NH4+, and TDS appear, and in contrast to the first plot, the order of K+ and NH4+ is swapped. You can see the difference in figure 3. Here it is important to know that the names in the curly braces have to correspond with the entries in the element line (which is the first line) of the data table.

4.2 Rows

In the same way, you can manipulate the rows displayed and the order of it by using the command

```
plotSchoeller(data,pivot,'Rows',{'Row1';'Row2';...;RowLast'})
```

or as an example

```
plotSchoeller(data,pivot,'Rows',{'O2';'S1'})
```

Here, just the lines Ottoquelle 1957 and Sprudel-1 1898 are displayed. The legend is also sorted in this order. If you do not use this command, all lines will be displayed in the order of the data sheet. You can see the difference in figure 4 and see that the pivot element is the line Ottoquelle 1957, and not Sprudel-1 1898. The reason for that is, that the pivot element also refers to the order of the lines. If you do not use the description column, as seen in table 2, the command will be much longer:

```
plotSchoeller(data,pivot,'Rows',{'\bf Ottoquelle 1957';'\bf Sprudel-1 1898'})
```

and is also vulnerable for spelling mistakes. Thus we highly recommend the use of the description column.

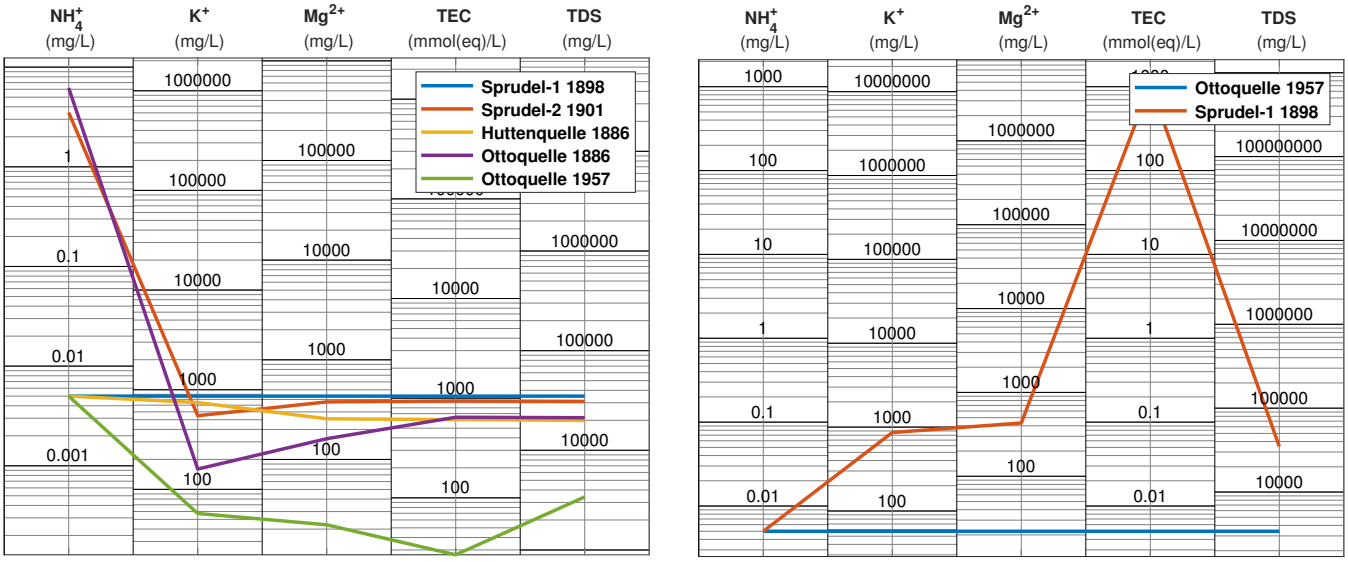

Figure 4: Two Schoeller diagrams with lines as in the data table (left) and adjusted lines (right).

4.3 Title

To add a title to your plot, you can use the command

```
plotSchoeller(data,pivot,'Title','MyTitle')
```

or as an example

```
plotSchoeller(data,pivot,'Title','\bf \fontsize{16} Notice Element NH_{4}^{+}')
```

The title will appear above the plot. You can also use the TEX commands of section 2 here. In the example seen in figure 5, the text is bold, has a bigger font size, and used the style commands for NH_4^+.

4.4 Legend

The legend of plot can be manipulated in two different ways. One way is to use the command 'legend' after the plot command. Herefore you have to save the plot as a variable and then adjust the legend options by using the legend command:

```
fig1=plotSchoeller(data,pivot);
legend(fig1,'Location','MyLocation','NumColumns',Mynumber);
```

For more information and command options see [5].
Because the location, the amount of columns, and the style of the legend are the most used commands, we integrated them into the plot command to provide an easier use of them. The following subsections explain the usage.

4.4.1 Legend Column

This plot option can manipulate the amount of columns in the legend. With the command

```
plotSchoeller(data,pivot,'LegendColumns',colums)
```

or as an example

```
plotSchoeller(data,pivot,'LegendColumns',2)
```

you can change the amount of columns for the legend. You can see an example in figure 6.

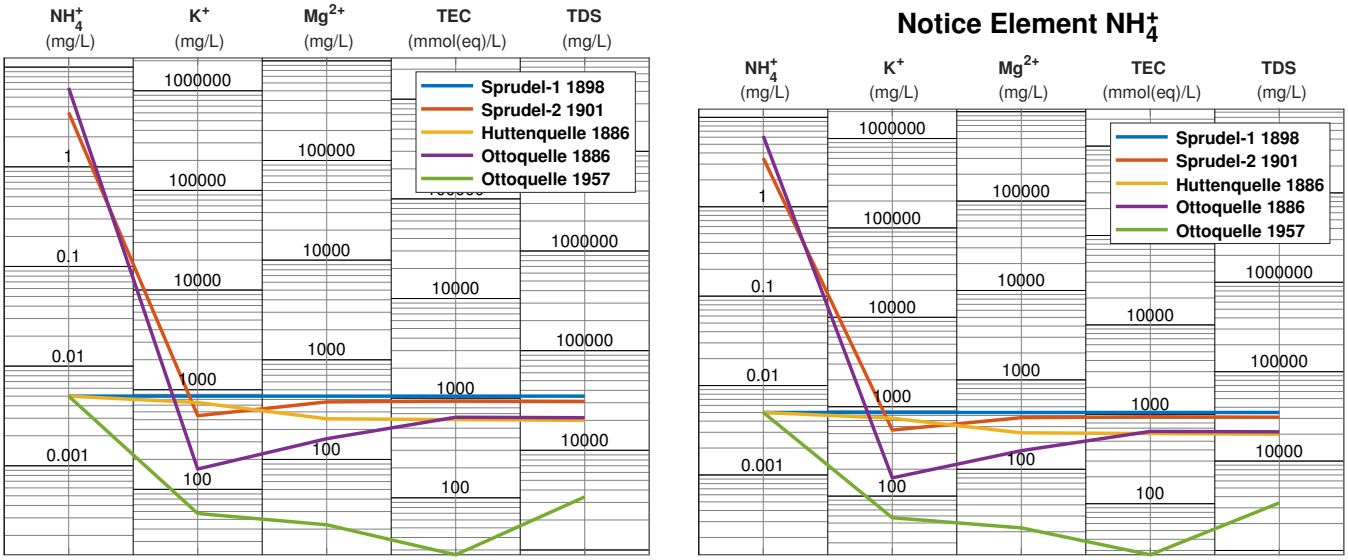

Figure 5: Two Schoeller diagrams with no title (left) and a customed title (right).

4.4.2 Legend Position

This plot option can manipulate the position of the legend. With the command

```
plotSchoeller(data,pivot,'LegendLocation','MyLocation')
```

or as an example

```
plotSchoeller(data,pivot,'LegendLocation','southwest')
```

you can change the position of the legend. The standard value here is northeast. Possible values in the value part of the command are the cardinal points. You can also determine whether the legend should be outside or not by adding the suffix 'outside' e.g.

```
plotSchoeller(data,pivot,'LegendLocation','eastoutside')
```

Notice, that for an outside legend the size of the whole figure will not change. The consequence is, that the plot itself looks clinched. You can fix this issue by adjusting the figure size

```
plotSchoeller(data,pivot,'LegendLocation','eastoutside','FigureSize',[800,480])
```

You can find more information about the figure size in section 4.10. The difference of the commands is shown in figure 6. For more information about the legend and a list of all possible legend locations, you can look into the matlab documentary [5]. If you use the value 'none', no legend will be displayed.

4.4.3 Legend Text Style

You can manipulate the style of the legend text by editing your data sheet as described in section 2. You can either change the column Element Name or use a Sample Caption column as we recommended. You can use all the TEX commands we described in section 2. In the example graphics, we printed the texts of the legend as bold.

4.5 Lower Bound

With the lower bound of the plot, we mean the value below which no measurement is possible. The standard value of it is < 0.01. Any value in the data table, which has no entry or is set to zero is interpreted as < 0.01. To adjust this value you can use the command

```
plotSchoeller(data,pivot,'LowerBound',myValue)
```

or for example

```
plotSchoeller(data,pivot,'LowerBound',0.001)
```

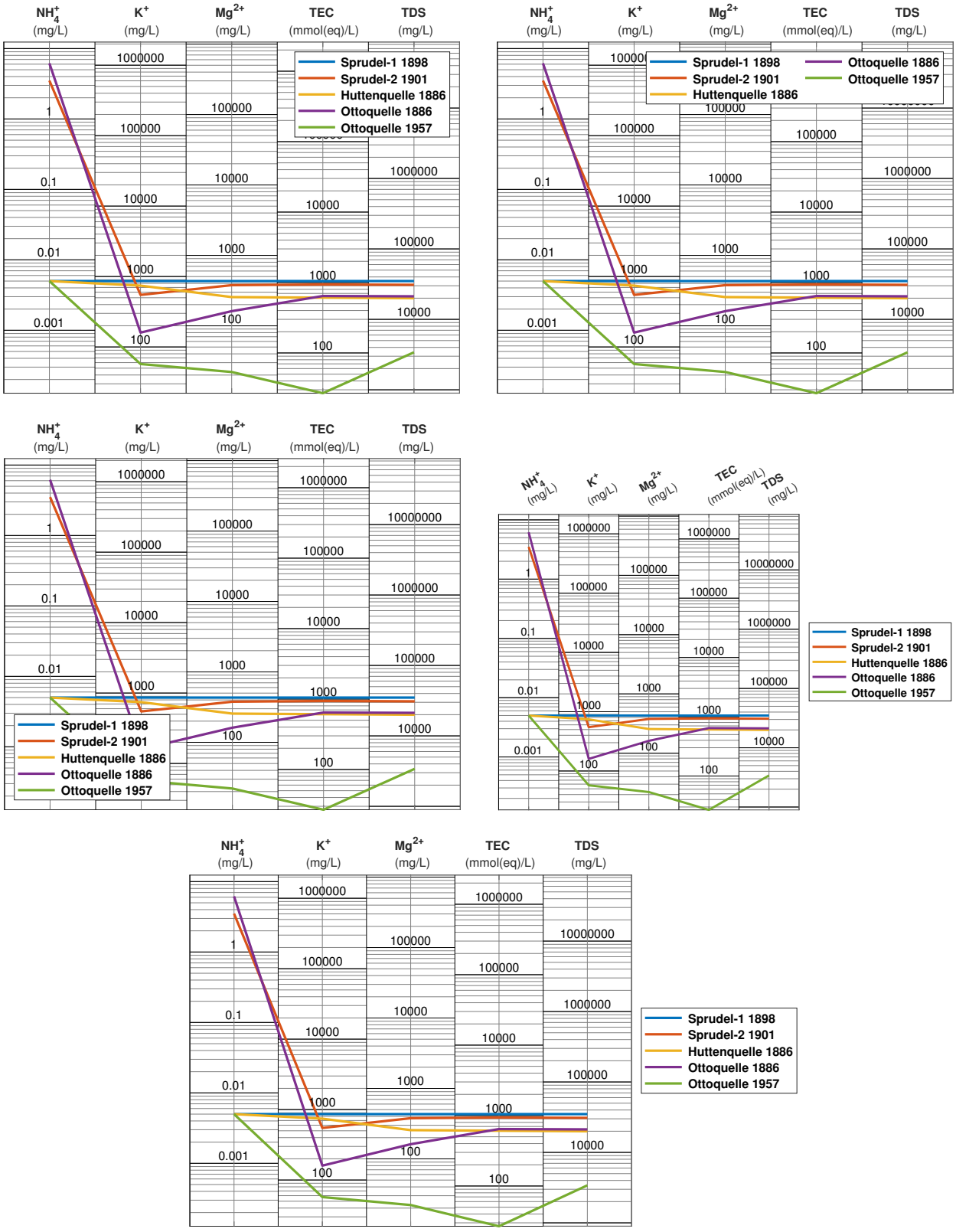

Figure 6: Five Schoeller diagrams with standard legend attributes (first), two legend columns (second), different legend location (third), outside legend (fourth) and outside legend with rescaled plot (fivth).

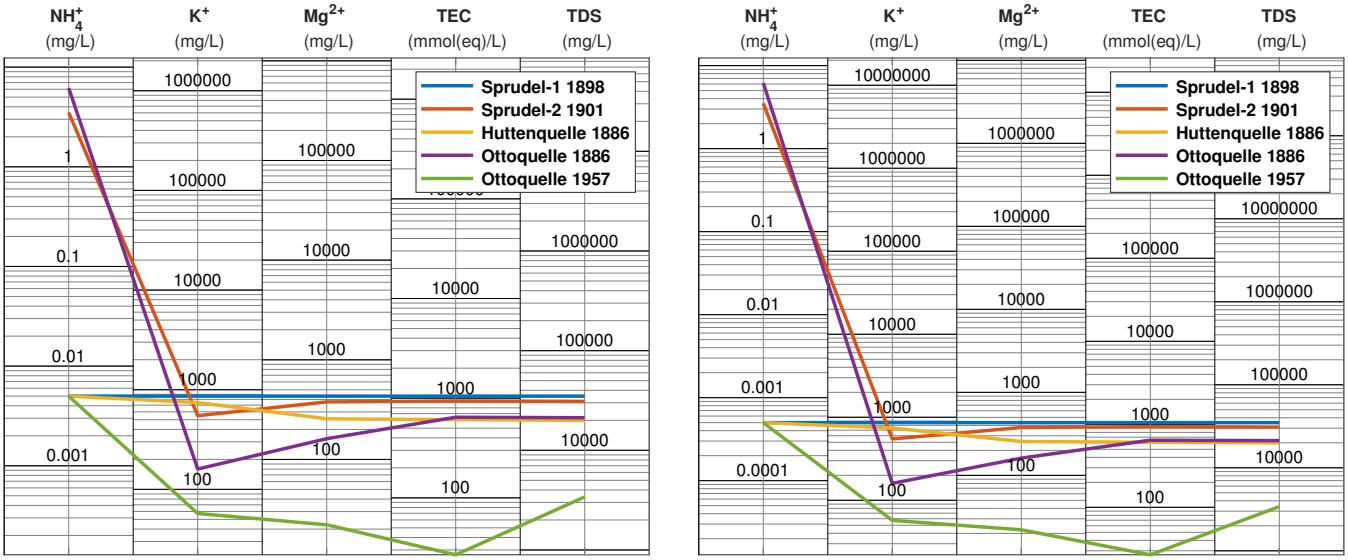

Figure 7: Two Schoeller diagrams with a lower bound of ≤ 0.01 (left) and a lower bound of ≤ 0.001 (right).

This will set the lower bound to the value 0.001. You can see the difference in figure 7. The lower bound can also be defined for every column, separated by a comma

```
plotSchoeller(data,pivot,'LowerBound',[value1,value2,...,valueLast])
```

or for example

```
plotSchoeller(data,pivot,'LowerBound',[0.001,1,100,10,1000])
```

Unfortunately, for this data set we will not see a difference at this step. This command also adjusts the caption of the measurement boundary described in section 4.6, where we will see a difference. Due to the nature of a logarithmic plot, the value zero is not valid.

4.6 Lower Bound Text

In addition to section 4.5, you can decide whether the lower bound captions are displayed or not. The command is quite similar to the upper command

```
plotSchoeller(data,pivot,'LowerBoundText',1)
```

The value for the lower bound text must be 0, 1 or a vector out of 0 and 1 like

```
plotSchoeller(data,pivot,'LowerBoundText',[0,0,1,0,1])
```

If you use 1 as an argument, the corresponding text is displayed, if you use 0, none of the text is displayed. If you use a vector, you can decide for every column, if the text is displayed or not. The standard value is 0, so all texts are not displayed. As we can see in figure 8, usually the lower bound text is displayed at the bottom of each column. If the pivot element has no entry, the lower bound text is displayed at the constant line of the pivot element. To highlight, that the displayed value is not the actual value, the lower bound text is red. In our example it might be useful just to display the lower bound text of column 1 and 4. It might also be useful to set the lower bound of column 1 to 0.001 and the lower bound of column 4 to 10. We can combine both commands with the code line

```
plotSchoeller(data,pivot,'LowerBoundText',[1,0,0,1,0],'LowerBound',[0.001,0.01,0.01,10,0.01])
```

to create the last plot of figure 8.

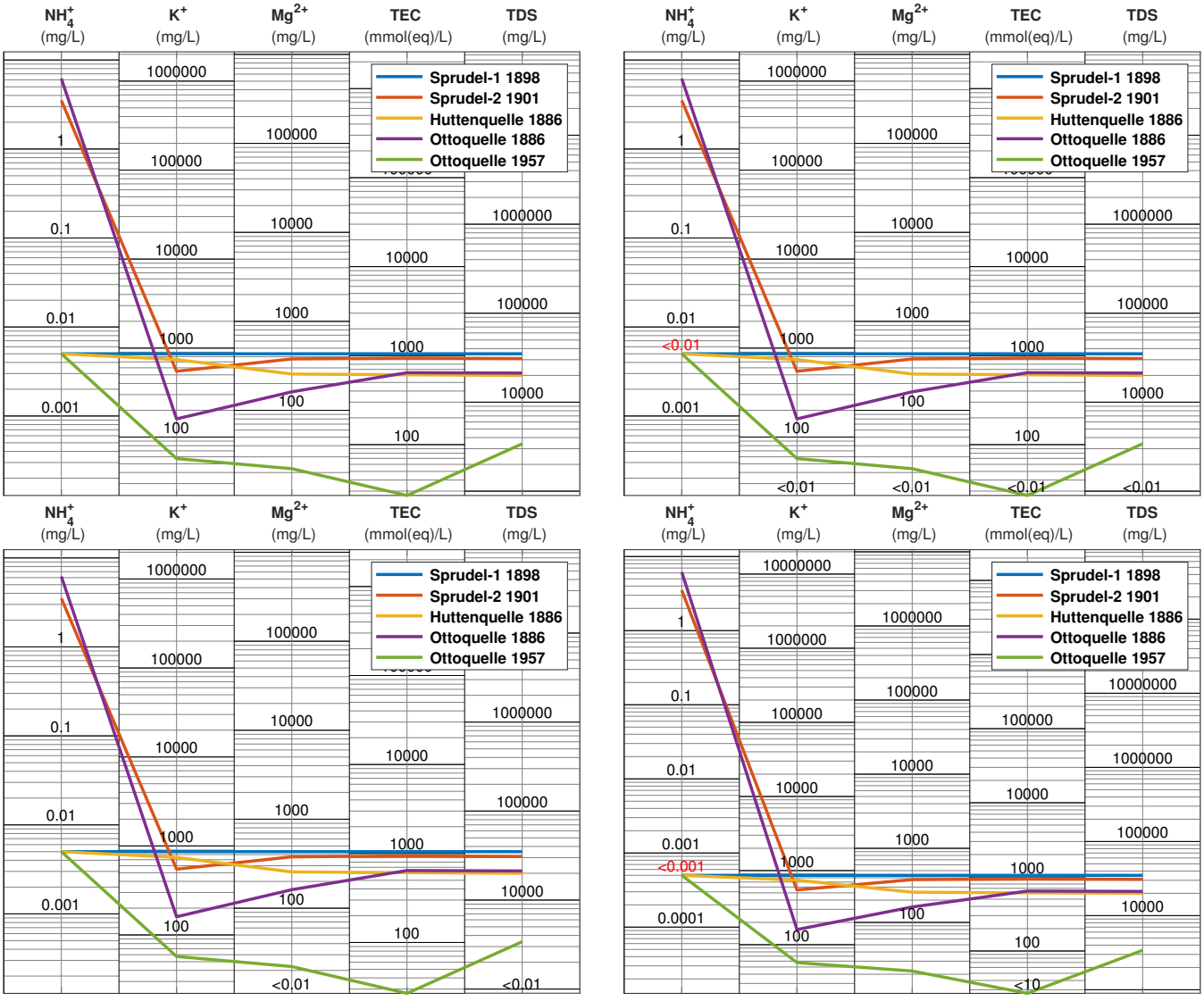

Figure 8: Four Schoeller diagrams with no lower bound text (first), all lower bound texts (second), just lower bound texts at column 3 and 5 (third), and lower bound texts on column 1 and four with adjusted lower bound values (fourth).

4.7 Floor Lower Values

This option sets all values which are lower than the lower bound value (of each column separately) as an empty cell (cf. 4.6). The idea behind this is, that we declare a limit in our plots for empty fields, under which no measurement is possible. If there are entries below this limit, the plot might not be consistent anymore (e.g., if the plot displays, that one line has values below 0.01 and plots another line with a value under this point). This command allows you to cut all these lower entries and setting them to the bottom of the plot. The code example is

```
plotSchoeller(data,pivot,'FloorLowerValues',1)
```

The limited value here is the lower bound value of section 4.5. If no lower bound value is defined, it is 0.01. If the lower bound value is different for each column, the command will set the empty cells automatically for each column. Unfortunately, we have no entries in our data sheet that is below 0.01. To see a difference, we extend the example command to

```
plotSchoeller(data,pivot,'FloorLowerValues',1,'LowerBound',[0.01,100,400,0.01,0.01])
```

As you can see in figure 9 for the second column, the green line is set to the bottom of the column (because the green data sample is the only one, which is smaller than 100 for K^{+}), and for the third column any line, except the blue one, is set to the bottom (because all other samples are smaller than 400 for Mg^{2+}). You can also use the FloorLowerValue command for each column separately, analogue to the LowerBoundText and the LowerBound command. The command for that is

```
plotSchoeller(data,pivot,'FloorLowerValues',[column1,column2,...,columnLast])
```

or as an example

```
plotSchoeller(data,pivot,'FloorLowerValues',[0,1,0,0,0],'LowerBound',[0.01,100,400,0.01,0.01])
```

As you can see in figure 9, just the green line in column 2 is set to the bottom, while the lines in column 3 are as usual.

4.8 Upper and Lower Display Factor

The upper and lower display factor adjust the height between the highest data point and the upper boundary of the plot, respectively, as well as the lowest data point and the lower boundary of the plot. Are the values equal to 1, the lowest element is on the lower boundary, respectively, and the highest element is on the upper boundary of the plot. The standard values are 2 for the upper display factor and 0.5 for the lower display factor. The value for the upper display factor needs to be greater or equal to 1, the factor for the lower display factor needs to be smaller or equal to 1. To adjust them, you can use the commands

```
plotSchoeller(data,pivot,'UpperDisplayFactor',upperDisplayFactor)
plotSchoeller(data,pivot,'LowerDisplayFactor',lowerDisplayFactor)
```

or as an example

```
plotSchoeller(data,pivot,'UpperDisplayFactor',5)
plotSchoeller(data,pivot,'LowerDisplayFactor',1/5)
```

and combined for an example image

```
plotSchoeller(data,pivot,'UpperDisplayFactor',5,'LowerDisplayFactor',1/5)
```

As you can see in figure 10, the space between plot points and boundary is higher.

4.9 Font Size

This command specifies the size of all fonts to not be defined specifically. The command is

```
plotSchoeller(data,pivot,'FontSize',value)
```

or as an example

```
plotSchoeller(data,pivot,'FontSize',15)
```

The standard value here is 10. Notice that this command just manipulate all fonts that are not extra defined. If you set the font of a line in the legend, a column description or the title with the TEX commands of section 2, this font will appear as you define it in the data sheet (or in the title command). You can see the difference in figure 11.

It is important to know, that the font size command is very useful in combination of the scaling command of section 4.10 and the export command in section 4.11. Due to the resolution of your plot, the font size might look too big or too small. With this command you can adjust this to get the plot in a good shape.

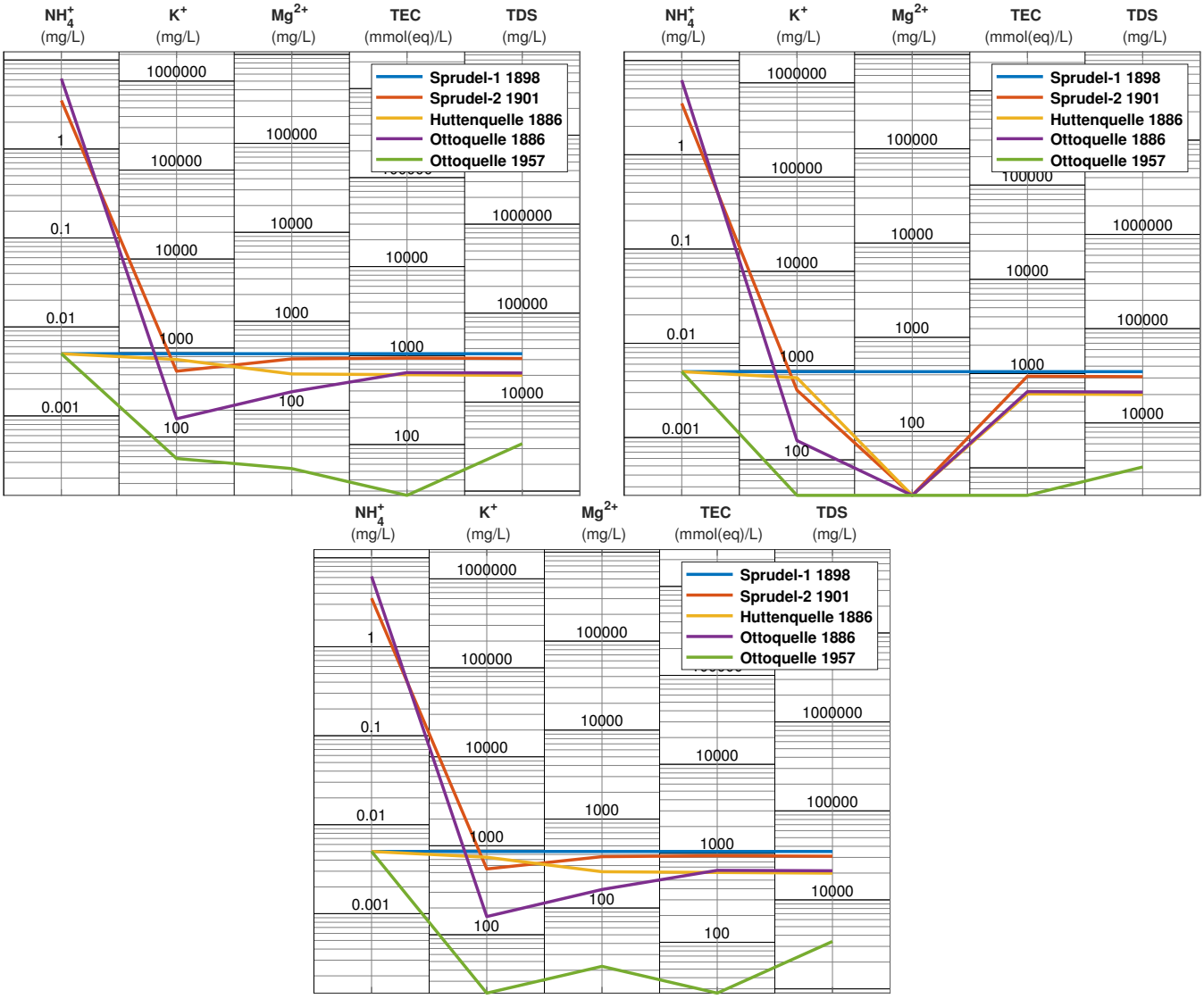

Figure 9: Three Schoeller diagrams with no floorLowerValues (first), flowLowerValue for each column (second) and just floorLowerValue for the second column (third).

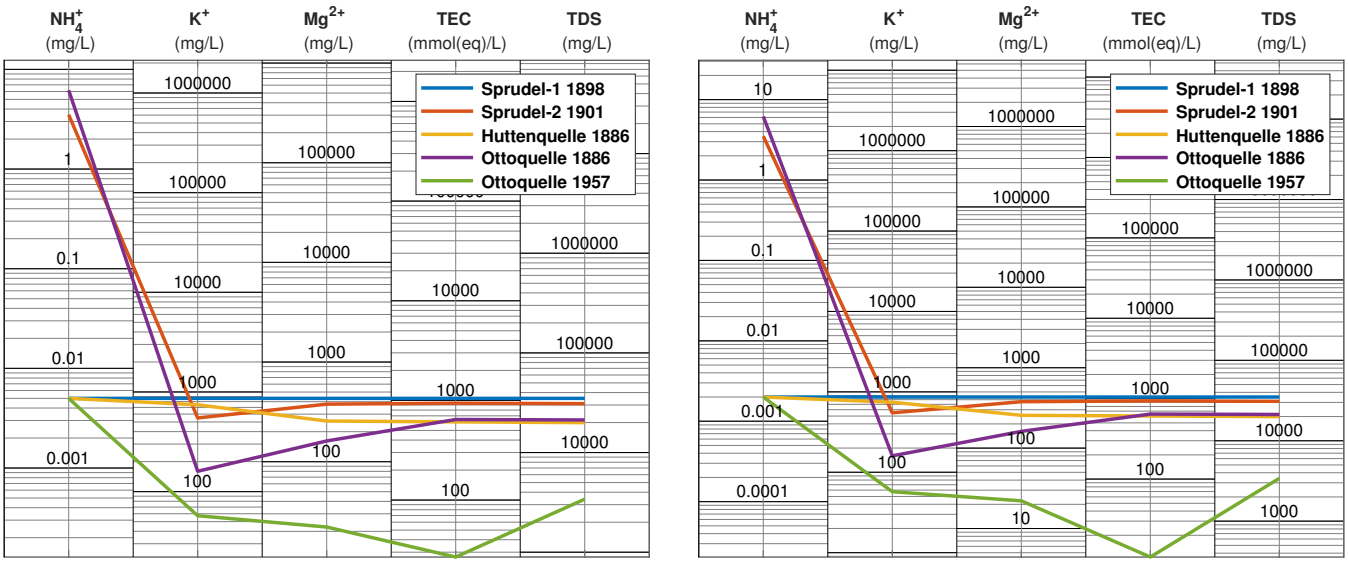

Figure 10: Three Schoeller diagrams with standard display factors (left) and adjusted display factors (right).

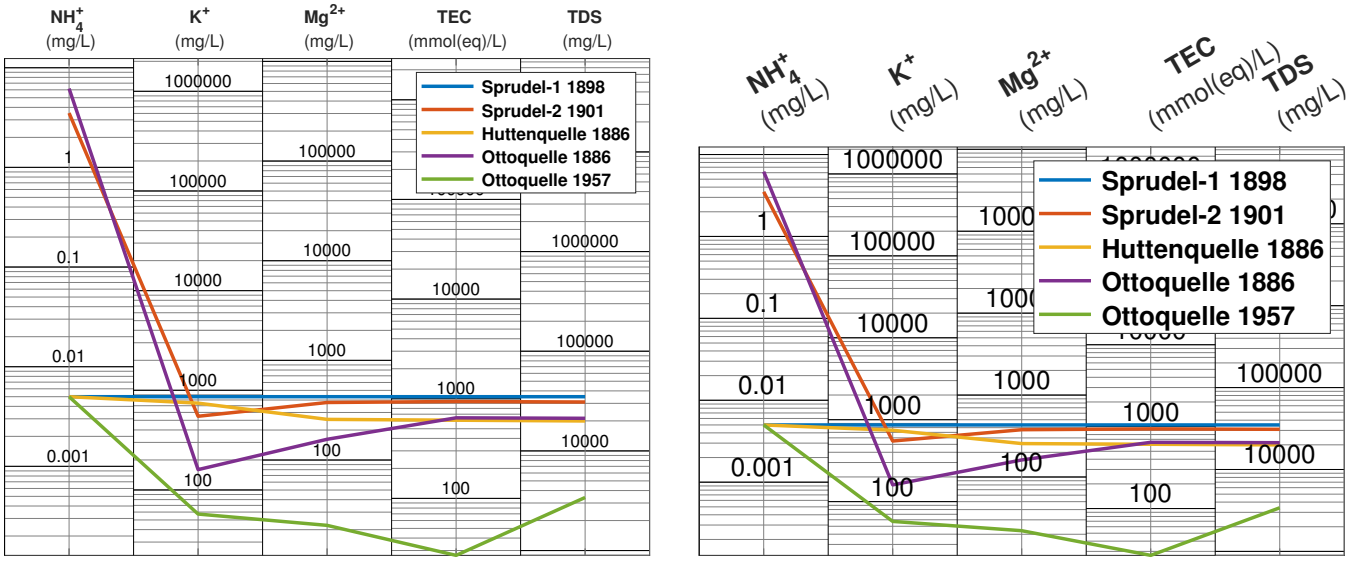

Figure 11: Two Schoeller diagrams with a font size of 10 (left) and a font size of 15 (right).

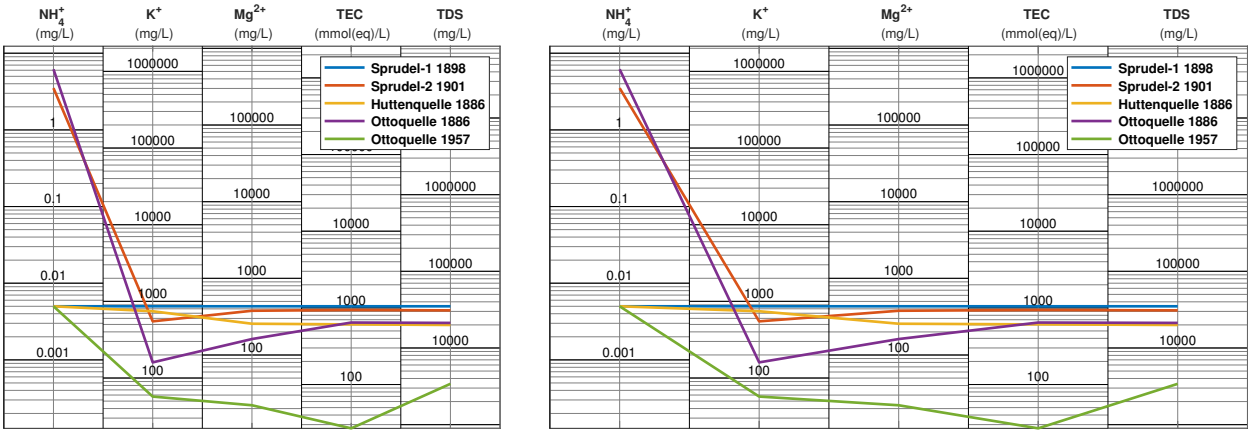

Figure 12: Two Schoeller diagrams with a resolution of 640×480 (left) and a resolution of 900×480 (right).

4.10 Scaling and Figure Size

This command is especially useful to export the plot to a graphic. With the scaling command

```
plotSchoeller(data,pivot,'FigureSize',[width,height])
```

or as an example

```
plotSchoeller(data,pivot,'FigureSize',[900,480])
```

the size of the plot will be width $\times$ height pixel big. The values reference to the whole graphic (including element names and p.r.n. the title). Especially for exporting to a .png graphic, you might get a high resolution to get a graphic of high quality. The standard value here is the screen size of your monitor. You can see the difference in figure 12. Please notice, for a high resolution the standard font size of 10 might be too small to get a good readability. You can adjust them with the font size command of section 4.9. It is also important to know, that the absolute figure size is just important for image graphics like .png or .jpg. For vector graphics (as used in this manual) just the relative difference between height, width and font size is relevant (as vector graphics are scaleable).

4.11 Export

One of the most useful commands is to export the plot to several files. First we recommend the command[1]

```
plotSchoeller(data,pivot,'CreateFigure')
```

This yields matlab to produce a plot, based on a vector graphic format, which produces better pictures of the plot. To export, you can use the following

```
plotSchoeller(data,pivot,'CreateFigure','Print',printpath, printproperties)
```

This saves the plot to the file printpath with print properties printproperties. As an example, we can use

```
plotSchoeller(data,pivot,'CreateFigure','Print','./mySubFolder/myName','-dpng')
```

This saves the plot in the directory you are in, to the subfolder mySubFolder with the file name myName. The file format is set with the plot property -dpng, which produces a .png file. Another example is

```
plotSchoeller(data,pivot,'CreateFigure','Print','c:/myFolder/myPlot','-dpdf')
```

which saves the plot in the directory c:/myFolder/myPlot as an .pdf file. Note: If the file format supports vector graphics, matlab will produce a vector graphic. Some useful commands for different output formats are [6]

[1] Matlab announced that this option might not be available in the future. We still keep it in the code but in a further version it might be erased. If so just erase the part `'CreateFigure'` in the commands.

command	description	vector graphic
-dpng	PNG 24-bit	no
-djpeg	JPEG 24-bit	no
-dbmp	BMP 24-bit	no
-dpdf	Full page Portable Document Format (PDF) color	yes
-depsc	Encapsulated PostScript (EPS) Level 3 color	yes

You can find a full list on [6]. We would recommend the -depsc command. It produces full scaleable pictures and can easily be used in LaTeX. You can see the benefit by zooming into any graphic in this manual. None of them will appear pixelated. All of the images in this manual are .eps files. Unfortunately, Microsoft Word or Power Point will not support any vector graphics. Therefore, if you are forced to use this programs, we recommend to produce .png files with a preferably high resolution as explained in section 4.10. Note that it might be necessary to adjust the font size as explained in section 4.9.

4.12 Line Styles

In addition to these commands, you can use the full spectrum of matlab line properties for plots. Any of them can be found in [7] and [8]. Just enter the plot properties to the command as in the normal plot command, e.g.:

```
plotSchoeller(data,pivot,'.--','Markersize',15,'Color','r','LineWidth',4)
```

In this example, the first command `'.--'` specifies the appearance of the line. Here, matlab plots a line with markers on the values and dashes the line. The second command `'Markersize',15` sets the size of these markers to 15 pixel. Command `'Color','r'` plots all lines in red and command `'LineWidth',4` sets the thickness of all lines to 4 pixel. You can see the result in figure 13.

You can also set the line styles separately for any line. Therefore, you have to set the properties as a vector of values. If you want to specify e.g., the marker size for any line separately, you can use the vector

```
[value1;value2;...;valueLast]
```

or as an example

```
[5;10;25;5;15]
```

Be aware that the delimiter between the entries are `;` and not `,`. The idea behind is:

- column/element delimiter `,`
- row/line/sample delimiter `;`

It also might be necessary to use a combination of both delimiter, e.g., if you want to specify the line colors with RGB values. An example for that might be

```
[1,0,0;0,1,0;0,0,1;1,1,0;0,1,1]
```

As you can see, the color for the first line is `1,0,0` (red), for the second line `0,1,0` (green), and so on. In an analogue way, you can write a vector of text elements, e.g., to set the color of any line separately. Unfortunately though, the notation might get a little confusing. For a vector of text, you have to use `"` and not `'` as text separator elements. The vector therefore looks like

```
["text1";"text2";...;"textLast"]
```

or as an example

```
["-";".-";"--";".--";"d-"]
```

If we want to adapt the upper command to different lines, it could be

```
plotSchoeller(data,pivot,["-";".-";"--";".--";"d-"],'Markersize',[5;10;25;5;15],...
        'Color',[1,0,0;0,1,0;0,0,1;1,1,0;0,1,1],'LineWidth',[3;4;2;5;1])
```

Note that the upper commands are just examples. Matlab provides much more commands to customize the line styles and you can find them in [7] and [8].

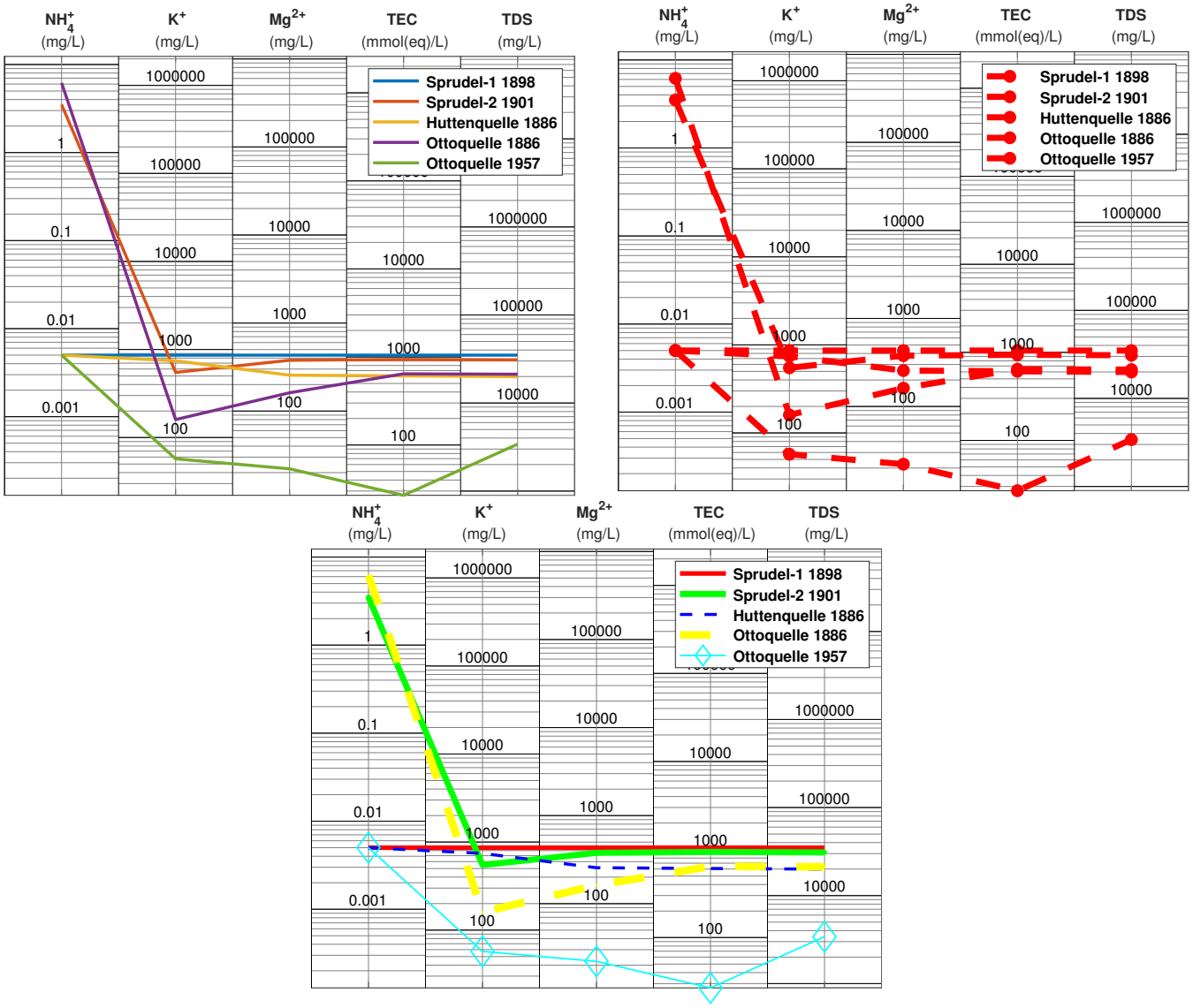

Figure 13: Three Schoeller diagrams with standard line styles for this manual (first), the same customized line styles for each line (second) and customized line styles for each line (third).

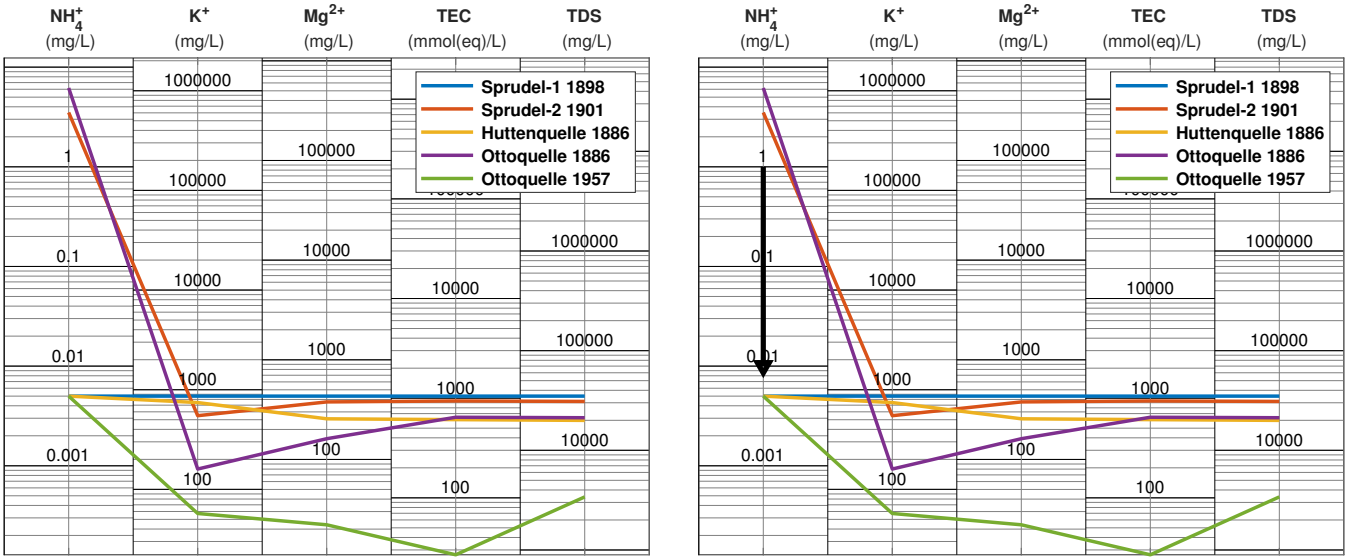

Figure 14: Two Schoeller diagrams with no annotation (left), and a custom annotation (right).

5 Annotations and Relative Data Points

After you created the plot, you can use the full bandswitch of annotations matlab provides for plots. You can find all the commands in [9] or you can add annotations over the matlab plot, overlying in the register insert. The disadvantages of such matlab annotations are, that they work with the relative position of the plot window, not with the data values. This means, e.g., if you place an arrow as an annotation on the plot, the values for starting and endpoints of it are between 0 and 1. If you want to zoom in or shift the graphic to any direction afterwards, the arrow stays at its position and will not change its place.

It might be necessary for you to place an relative annotation to your figure. However, there is one problem here. Due to the different scales of each column in a standardized Schoeller diagram, it is difficult to get the values you need for your additional annotation, or possibly for other things for which you need the absolute data points of the plot. For this case, we provide the command

```
[PointMatrix]=plotSchoeller(data,pivot,'RelativeDataPoints',...
  [xPoint1,yPoint1;xPoint2,yPoint2;...;xPointLast,yPointLast])
```

As you can see, the x- and y- value of a point are separated with a `,` and two points are separated with a `;`. With this command you can put the visible points of the plot into the command and return the points which you need.
The visible x-values of any point can be determined by the following: The left side of the first column starts at 1, has the vertical middle bar at 1.5 and ends with 1.9999. The second column starts with 2, and so on. The visible values of the y-axis can simply be obtained by looking at the tick lines of each column.

Now we construct an example to use the command RelativeDataPoint. This example might look a little difficult and might require additional matlab commands that you might not yet know. At this stage we just want to show one example, how the command can be used. For more information of other matlab commands, we recommend [2].
Assuming that you want to draw a vertical arrow in the first column from value 1 to value 0.01, then you can use the command

```
[PointMatrix]=plotSchoeller(data,pivot,'RelativeDataPoints',[1.5,1;1.5,0.01])
```

In this point matrix, there are 2 lines. The first one is the relative point for point (1.5,1), and the second point is the relative point to value (1.5,0.01). Now you can draw an arrow in the following way

```
plot(PointMatrix([1,2],1),PointMatrix([1,2],2),'-',...
  'HandleVisibility','off','LineWidth',3,'Color','k')
plot(PointMatrix(2,1),PointMatrix(2,2),'v','HandleVisibility','off','LineWidth',3,'Color','k')
```

The parameter `'HandleVisibility','off'` are just to suppress an entry in the legend list. Of course, you can also create lines between different columns like

```
[PointMatrix]=plotSchoeller(data,pivot,'RelativeDataPoints',[1.5,1;3.5,1000])
```

To export the plot with the customized annotation, we use the command

```
print('./exampleX','-depsc')
```

and we can see the result in figure 14. This print command can also be used after doing manual annotation on the figure, e.g., with the register insert. The annotation will be exported with the plot together.

References

[1] Alexander Dietz and Rafael Schäffer. "Standardized Schoeller Diagrams with Matlab - The Plotting Tool". In: TUDataLib (2022). DOI: https://doi.org/10.48328/tudatalib-877.

[2] Daniel T. Valentine and Brian Hahn. Essential MATLAB for Engineers and Scientists. 7th ed. Elsevier Science, 2019. ISBN: 9780081029978.

[3] Greek Letters and Special Characters in Chart Text. URL: https://de.mathworks.com/help/matlab/creating_plots/greek-letters-and-special-characters-in-graph-text.html (visited on 12/15/2021).

[4] URL: https://de.mathworks.com/matlabcentral/answers/102143-how-do-i-get-a-new-line-with-the-text-function-using-tex-formatting-in-matlab-7-2-r2006a (visited on 12/01/2021).

[5] Matlab Command Legend. URL: https://de.mathworks.com/help/matlab/ref/legend.html.

[6] Matlab Command Print. URL: https://de.mathworks.com/help/matlab/ref/print.html (visited on 12/19/2021).

[7] Matlab Command Plot. URL: https://de.mathworks.com/help/matlab/ref/plot.html#btzitot_sep_mw_3a76f056-2882-44d7-8e73-c695c0c54ca8 (visited on 12/19/2021).

[8] Matlab Line Properties. URL: https://de.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.line-properties.html (visited on 12/19/2021).

[9] Matlab Command Annotation. URL: https://de.mathworks.com/help/matlab/ref/annotation.html (visited on 12/19/2021).