Data visualisation at Höganäs AB

MORGAN ASK
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Data visualisation at Höganäs AB

Morgan Ask
morgan.ask.615@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Anders J Johansson, LTH
Ola Litström, Höganäs AB

Examiner: Ove Edfors

September 11, 2023

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

This thesis work was carried out for Höganäs AB, the world leading metal powder manufacturer. The goal was to create a program that automatically compiled logged data from their heat treating ovens and present it graphically, based on input in the form of time intervals. The program should also be able to compare outcomes during different time intervals and outcomes in different assignments.

Due to one of the limitations was that the program was to be written in an open source coding language, Python was chosen due to it being a very readable language, and thus easy to continue the work for another user.

For importing data from a .txt or .csv file the Python packages pandas was chosen, for presenting data graphically the package matplotlib was chosen, and finally to create the GUI the built in package tkinter was used.

This thesis work resulted in a program that fulfilled all the requirements that the client requested. A lot of emphasise was put into making the program easy to use, and the code being readable so that changes could easily be made in the future.

Table of Contents

List of Keywords

1. Python
2. PyCharm
3. Pandas
4. Matplotlib
5. tkinter
6. Data presentation
7. Writing program

Chapter 1

Introduction

1.1 Background

Höganäs AB is the world leader on the market for iron and metal powders with a yearly capacity of 500.000 tons. Höganäs AB was established in 1797 by Count Eric Ruuth as a coal mining company, and today has about 2400 employees all over the world. The applications of metal powders are many, and Höganäs AB uses them in areas such as brazing, additive manufacturing, pressing and sintering, soft magnetic composites, welding, electromagnetic applications, surface coating, water treatment, and many more.

Höganäs AB operates a customer development centre in the city of Höganäs, Customer Development Centre EMEA, where work is done with their customer's processes. One of the main processes is sintering, where pressed components of metal powder are heated to a high temperature which is around 1100-1300 degrees Celsius, to achieve the necessary strength.

This thesis work will be assisting in analysing their pressing and sintering process data. Metal powder is pressed into a shape, and with the sintering process the pressed particles forms a solid mass of material by heat, this is done without melting the material to a liquefied state. More practically this is done by letting the pressed iron powder to pass through different heat zones to acquire the best possible result.

1.2 Thesis goal and contributions

The goal of this thesis work is to create a program that automatically compiles logged data and presents it graphically, based on input in the form of time intervals and/or selected assignments. It should be possible, for example, to compare outcomes during different time intervals and outcomes in different assignments. Basic statistical data such as averages, standard deviations, and min/max values should be presented on the user's input. The program should be able to load data files in commonly used formats such as .csv and .txt, and be written in open source.

1.3 Scope

A program needed to be developed that could load .csv and .txt data files. The program should include a GUI, which makes it easy for a user to specify from which oven the data should be loaded. The program should also be able to present data in graphic form, where the user should be able to choose which data is presented. The program should preferably be created with an open source programming language. Documentation of the program how it is installed should also be included.

Examples of data are:

1. Temperatures from oven zones
2. Track speed
3. Input and output of oven
4. Gas flows
5. Gas composition based on analytical data

Documentation and presentation must be done in English

1.4 Outline

Data to load and use is already available at the start of work, through the use of a data logger that logs data from two furnaces into .csv files. Implementation and testing is done on 1 or two furnaces.

Chapter 2
Preparation and execution

2.1 Method

To start off the logged files needed to be visualized, after these were viewed a clearer picture of what needed to be done, and how it could be done could be formed. The furnace data were logged into .csv file and such were hard to visualize without using a parser, a program to analyze a string of symbols. To solve this problem fast and easy without delving too much into any certain programming language a visualization of the data structure was done by the program Jupyter notebook.

Below in figures 2.1 and 2.2 it is clearly shown the advantage to quickly use something like Jupyter notebook to visualize the content of a .csv file so that further strategy into solving the problem can be made.

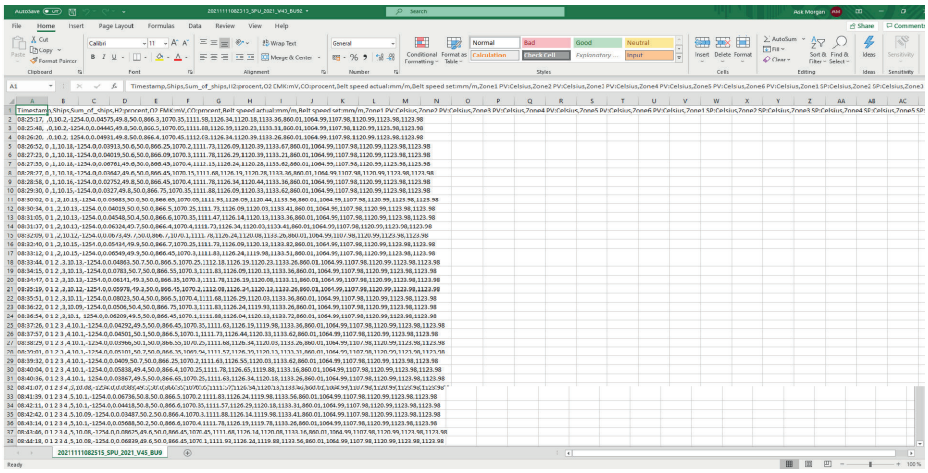

Figure 2.1: Example of .csv file

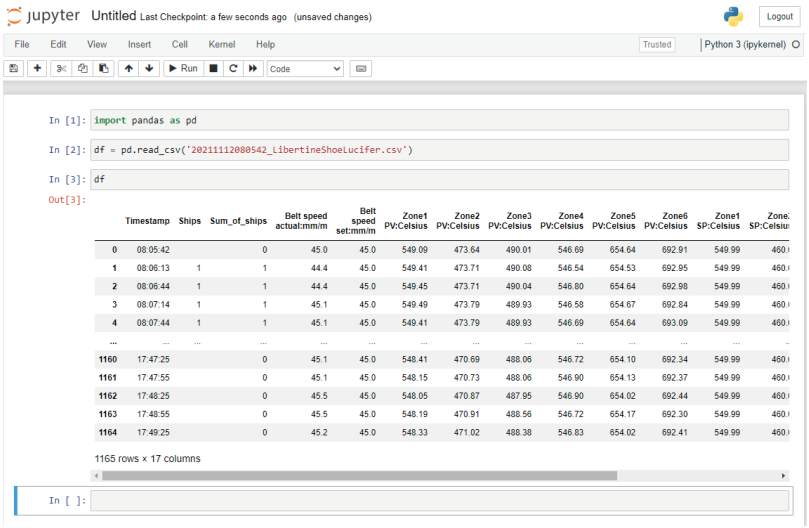

Figure 2.2: .csv file visualized through Jupyter notebook

After the visualization progress had been completed, a strategy to create the appropriate program for the client could be shaped. Many different ways could be found to do this, but Jupyter notebook was chosen due to easy access with the .pip installer that is found within Python[5], and Jupyter being a known program utilized for analyzing data. After the visualization had been made decision could be made which programming language to use.

2.2 Research

Several things needed to be looked into before dedicating oneself to a certain programming language, first off consideration had to be taken to the requirement of the client, which were:

1. Open source programming language.
2. Able to analyse data both numerically and graphically.
3. Able to load data files in commonly used formats, such as .csv and .txt.

I decided to create a program that would act as a platform for further development, deciding to write easily readable code, and for the code and the GUI to easily be adapted for further development on start. With this into consideration the coding language Python was decided upon.

Python has a very easily readable code due to its use of significant indentation, this means that in many coding languages like Java or C++ brackets are use to decide where methods and the likes start and end, but in Pythons case a space indentation is used instead, like a tab for example. Thus the code will be cleaner to look at and easier to read.

Further research for this project consisted of reading up on the Python language and its packages, and how they could be implemented to create a basic, easy to use program, that had a high probability of being supported in the future. Which was considered to be beneficial for the client if they felt the need to change the program in the future, by adding functions or adapting it for another similar type of furnace data collection within the company.

2.3 Deciding on packages and implementation

When choosing packages and implementing them, a great deal of thought was put into finding packages that in the future would be supported. One way to go about this was to chose packages that were popular and implemented into Python, thus easily installed through the use of the pip-installer. The pip(package management system) is used to install packages, pip connects to an online repository of public packages called the Python Package Index. In short pip is used with a command and will then easily install chosen package into your Python text editor. The theory here was that if it was popular and had been used a lot, it should be easier to find well explained tutorials and have a lot of support in the future.

The packages that were decided upon:

1. ***Pandas***[4], is a software library for Pythons programming language, it is used for data manipulation and analysis, and is especially useful for data structures and for manipulating numerical tables and time series.
2. ***Matplotlib***[2], is a software library written for the Python programming language that uses Pythons numerical mathematics extension NumPy[1].
3. ***Tkinter***[3], is the standard GUI when it comes to the Python programming language.

With this in mind a program could be created to analyse and visualise data from .csv files.

Chapter 3
Program

3.1 Program explanation

The program written is built up with the use of five python files:

- Run.py
- Thesis_v4.py
- Thesis_v4_support.py
- MyMethods.py
- globVar.py

Run.py is the programs main file, where the code is initiated, from there the program is started with the use of the def run().

Thesis_v4.py was planned to only consist of GUI initiation, but some things like the dropDown list were easier to build within the GUI-builder, which lead to some methods being located in this file.

Thesis_v4_support.py is a support module for Thesis_v4.py generated by the software PAGE, which is a drag and drop GUI creator for tKinter, PAGE was used to easily build up the structure of the GUI before fine tuning the appearance and functions for the widgets.

MyMethods.py is the file which was planned to have all the methods within it, unfortunately some methods ended up in Thesis_v4.py.

globVar.py is the module where control variables and dataFrames are stored, such as if the checkButton has been clicked, which column was chosen in the dropDown list, and the dataFrames themself are stored here.

To start the program define run() is used, which initiates the GUI creation with the define createGUIthesis(). Thus the main program has been created, and is awaiting instructions.

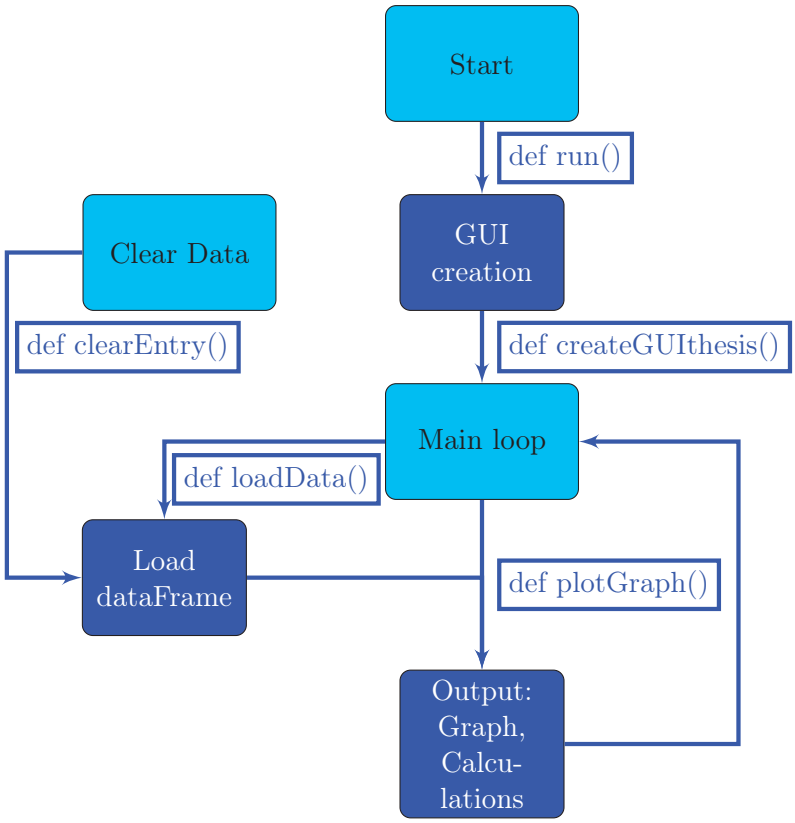

Figure 3.1: Flowchart showing the program from start to finish, and the definitions used to progress through the program.

Now the option of loading data into the program can be used with define loadData(), which opens up a window where you can search for the data in the harddrive that you want analysed, and load it into the program.

There is also an option to clear the data within the program with the define clearEntry()

Finally with define plotGraph() the data will be plotted into a graph, either together in one chart, or two separate graphs, so comparing different data with vastly different timelines can be done easily. Also the statistical data will be calculated and presented by the program.

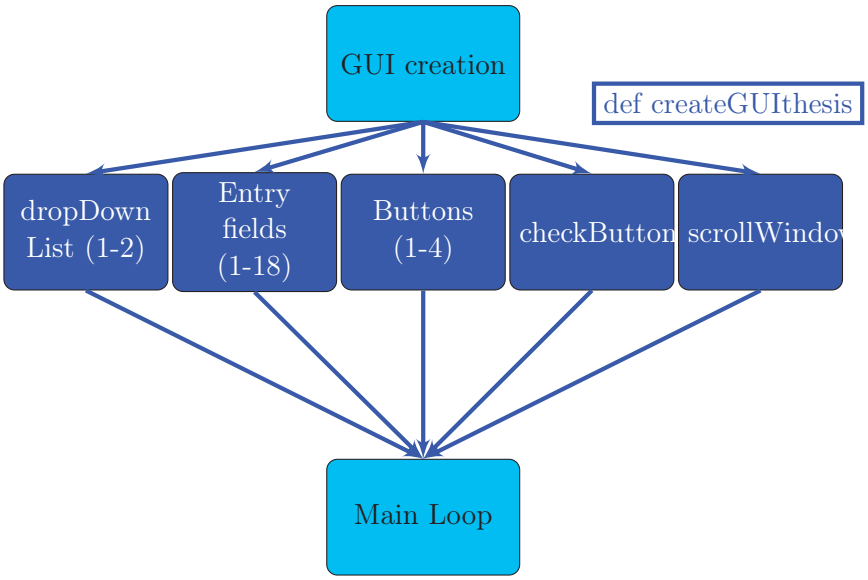

Figure 3.2: Closer look into the "GUI creation" section from Figure 3.1.

Figure 3.2 shows in more detail the happenings with define createGUIthesis(). Here a number of graphical representations are created as listed below:

1. Two dropDown lists, for selection between different data to be presented and calculated within the data file.
2. 18 different entry fields, these are used to present text and numbers, either hard coded such as Minimum/Maximum/Average/StDev, or printed by the program such as .cls file location and its statistical data.
3. Four buttons, used for communicating with the program by the user, options such as load dataframes, run, and reset.
4. checkButton, used to tell the program how the two different dataframes will be plotted in the program.
5. scrollWindow, which is the window where the graphical representation of the data will be presented.

After this the program enters the main loop awaiting further commands.

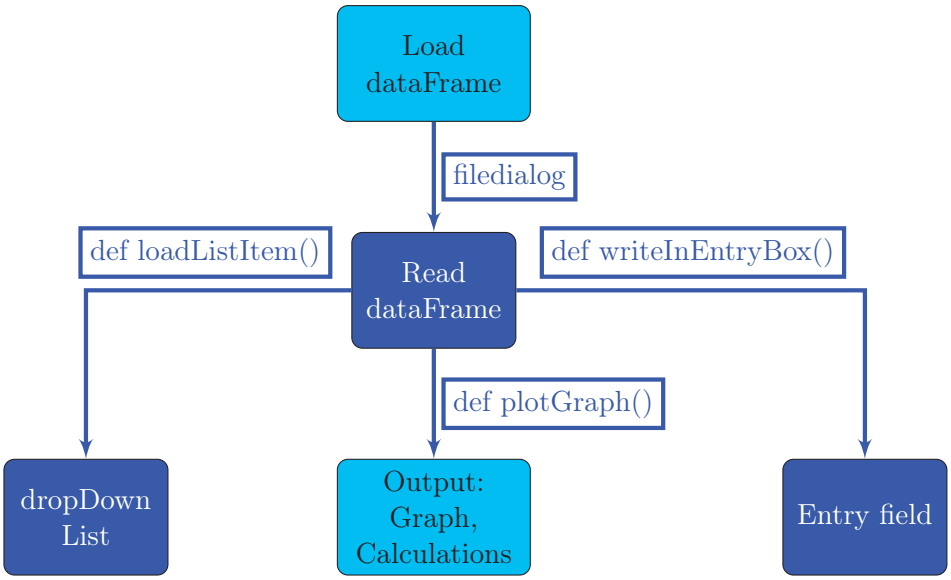

Figure 3.3: Closer look into the "Load dataFrame" section from Figure 3.1.

Figure 3.3 shows in more detail what happens when you load the dataFrame within the program. When pressing the load dataframe 1 or 2 button, the program will with the help of the pandas package use the filedialog in pandas to search your computer for the correct file. When the correct file is chosen its column names are loaded into the correct dropDown list, the name of the file is loaded into an entry field. And after this the program is ready to plot/calculate the data.

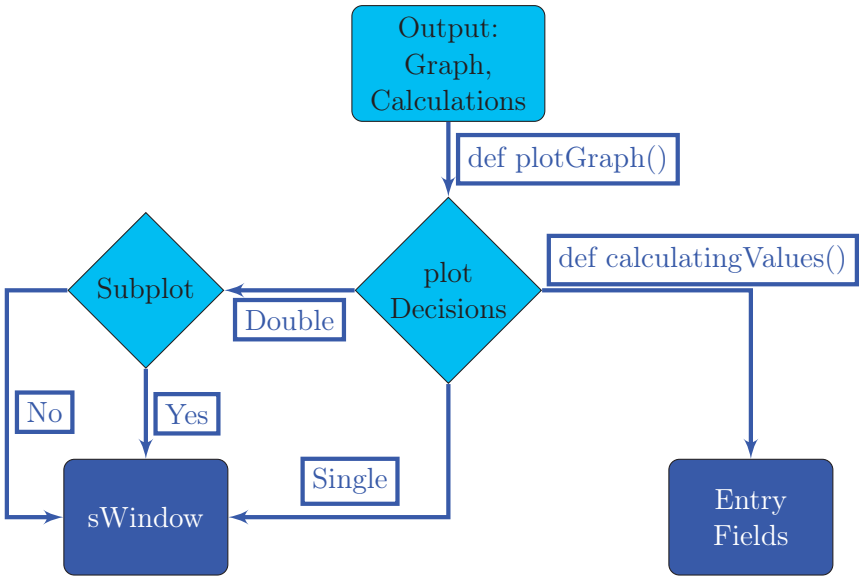

Figure 3.4: Closer look into the "Output: Graph, Calculations" section from Figure 3.1.

Figure 3.4 shows in more detail what happens when you plot the data with define plotGraph(). First the program looks how many files are loaded into the program(current version is maximum of two), then it looks if you want it subplotted or not. After this the data is plotted in the scrollWindow with the help of the package matplotlib. The data is also calculated statistically and presented in its chosen entry fields.

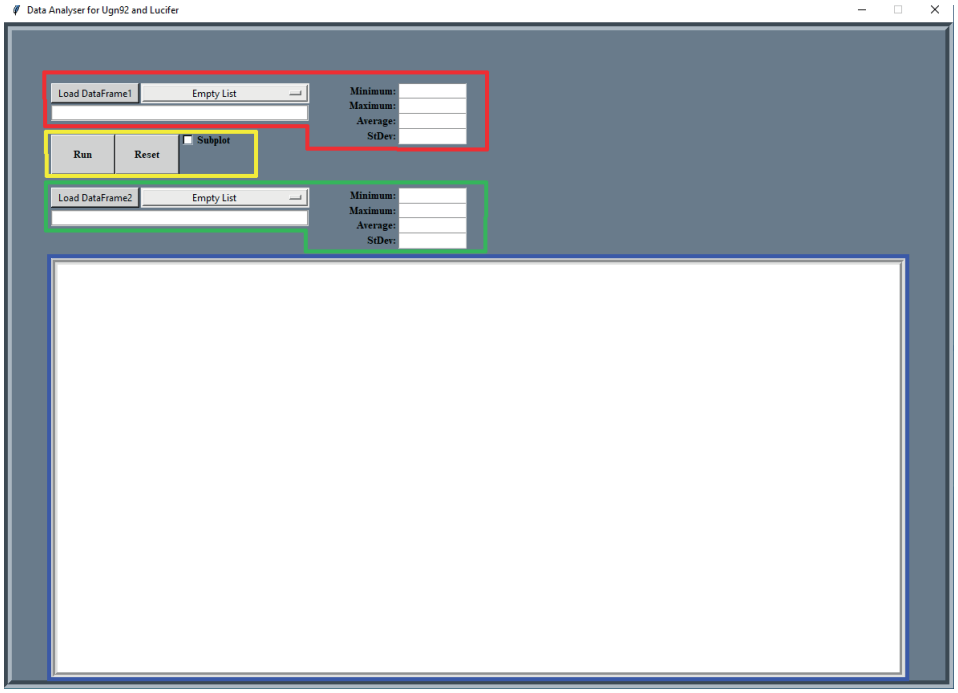

Figure 3.5: Shows the programs main window after it has been started and the GUI has been loaded.

Figure 3.5 shows the programs Main Window after the GUI has been created and no data has been entered.

- Red rectangle, In this rectangle the GUI-gadgets for dataFrame 1 is placed. On the left side button "Load DataFrame 1" is located that can activate def loadData(), and load a .csv file into the program. The name of the file will display just below in the entryField, and to the right of the button a dropDown list will display the column names that are within that file. On the right side of the red rectangle the statistical data for the chosen .csv file will be presented after it has been calculated.
- Green rectangle, is identical to the red rectangle but this is where you can load your second .csv file if needed.
- Yellow rectangle, here are two buttons and a checkBox, the button "Run" initializes the def plotGraph(), that will calculate the statistical data for the chosen dataFrames, and plot the graphs chosen in the dropDown lists. The button "Reset" will clear the dataFrames loaded into the program. The checkButton "Subplot" will determine if the program choses to play two dataFrames into the same graph or separate graphs.
- Blue rectangle, is the scrollWindow where the graphs will be presented within.

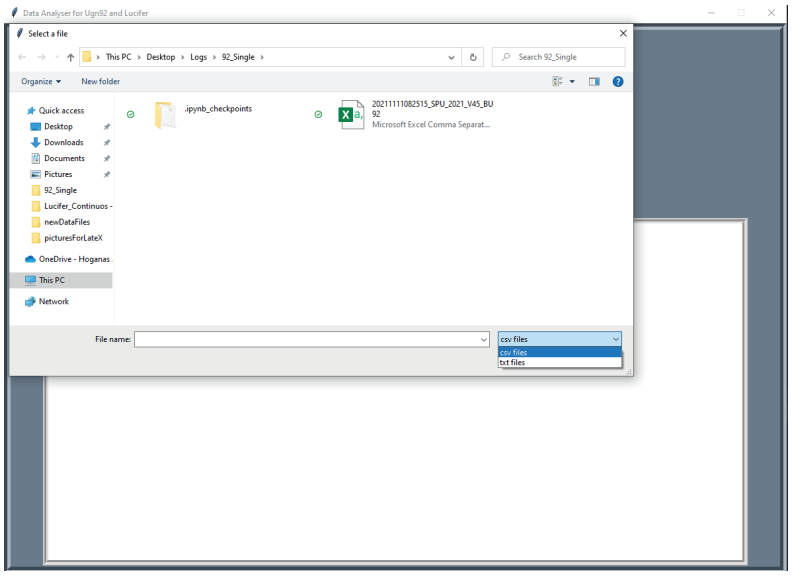

Figure 3.6: Shows the program after the button "Load dataFrame (1 or 2)" has been pressed.

Figure 3.6 shows the action taken by the program from clicking on a "Load DataFrame" button, now you can look for a .csv file that is located within the computer or on a network connected to the computer.

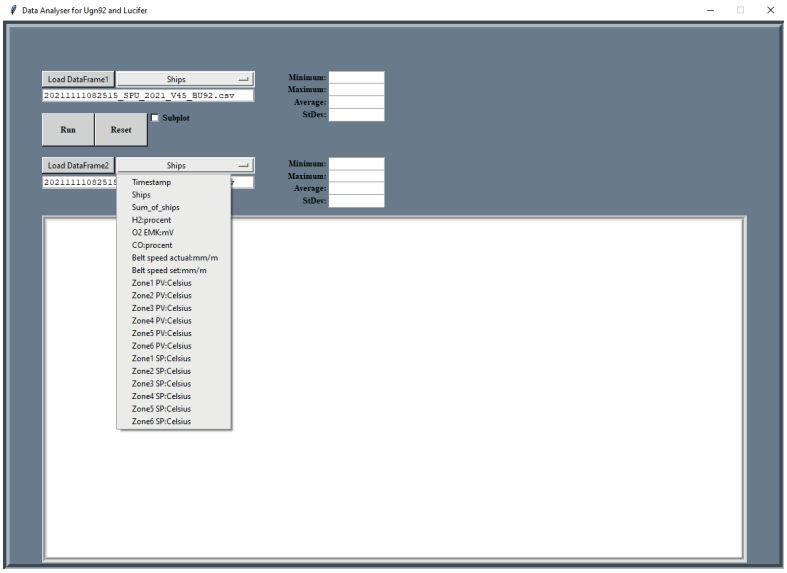

Figure 3.7: Shows the program after a dataFrame has been loaded. The different columns names can be shown in the dropDown-List, and also the name of the .csv file can be seen in the entry field.

Figure 3.7 shows the program after you have loaded data into the program, you can clearly see all the collumn names in the selected dataFrame, in this case in the second dataFrame loaded into the program. Also shown below the "Load DataFrame1" and "Load DataFrame2" respectively the entryField will display the file name of each .csv file.

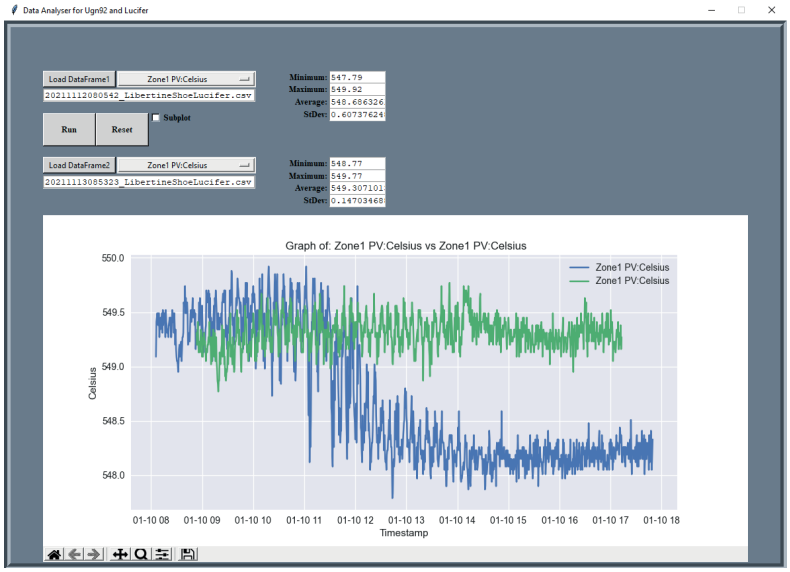

Figure 3.8: Shows the program after two dataFrames has been plotted into the same graph inside the scrollWindow. Also the results from the statistical calculations can be shown in its selected entryField.

Figure 3.8 shows the joined graph of two DataFrames in the scrollWindow, the statistical data for each dataFrame, and also the toolbar accessible when the graphs has been plotted. The toolbar is very versatile and can be used to zoom into the graph, save the graph into a .png, format the appearance of the graph. Also the timeFrame on the x-axis is very versatile thanks to the pandas package, it will adapt its data to how far into the graph you have zoomed. And if you have years of data presented, it will show only the date, and if you zoom very far into the graph, only the daily time will be presented.

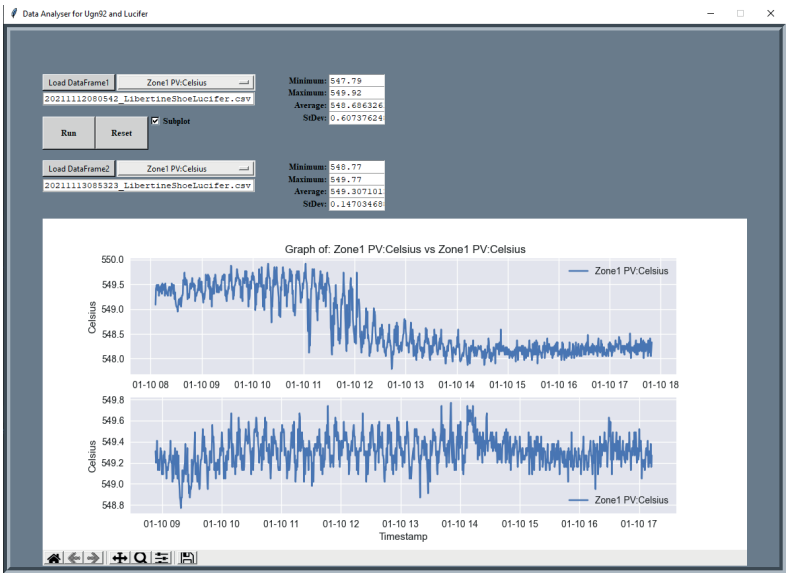

Figure 3.9: Shows the program after two dataFrames has been plotted in a subplot mode.

Figure 3.9 shows the same data as in figure 3.8, but subplotted, now you can individually zoom in on different parts of the graphs for further investigation/-comparison.

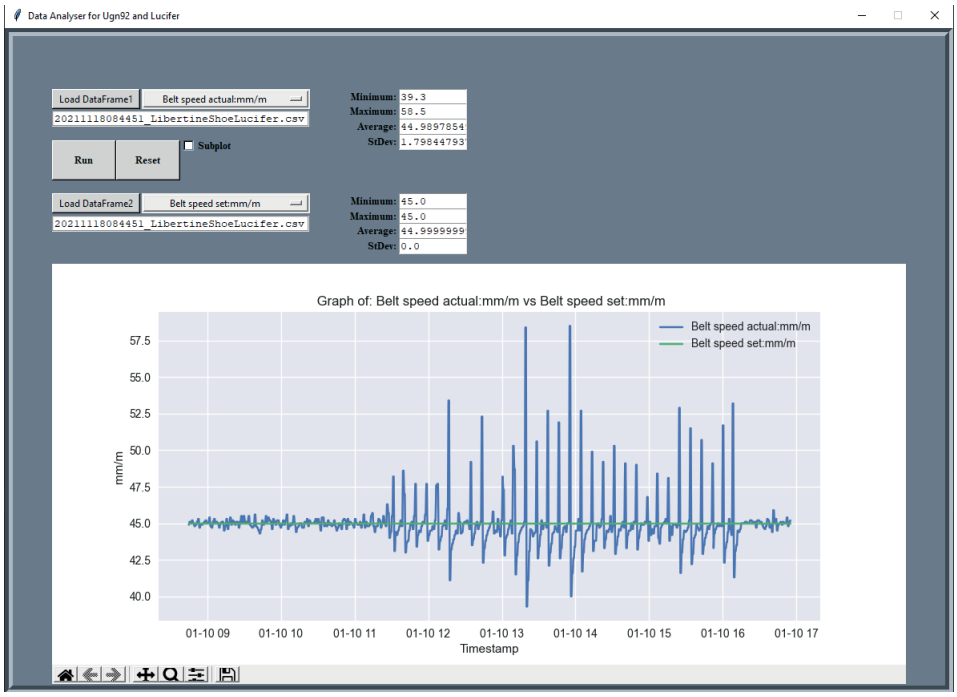

Figure 3.10: Comparing actual vs set speed

Figure 3.10 shows the set speed by the furnace user, and what the actual speed is.

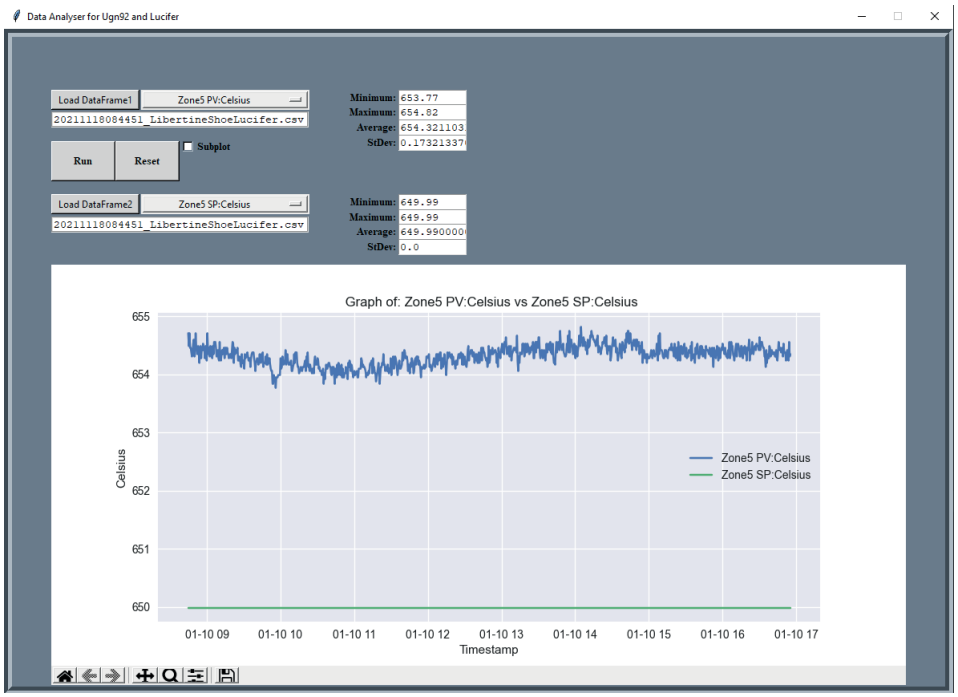

Figure 3.11: Comparing actual vs set temperature

Figure 3.11 shows the set temperature in zone 5 by the furnace user, versus the actual temperature in zone 5.

Figure 3.12-3.13 Demonstrates an example on how the program could be used. In this case the speed set vs the actual speed has a lot of hops. If comparison is taken due to the actual speed and amount of shipments on the belt is shown as in Figure 3.13, we can see that everytime a shipment goes off the belt, the speed of the belt increases shortly before normal speed resumes.

Figure 3.12: Looking into reason for speed difference

Figure 3.13: Speed change due to changing amount of ships on belt

Chapter 4
Conclusions

4.1 Discussion

The program was written to last, a lot of effort was put into making it viable for the future. Python was chosen as a programming language since it is easy to read, a versatile language with its many packages. The packages chosen were some of the most basic and popular ones in its field, to ensure that updates and support would continue and such making it possible to update and adapt the program in the coming years.

A lot of effort was put into making the program easy to use, and foreseeing errors that can occur in the future. The program is not only to be used as a specific furnace data analyzer, but can also be used for other situations, as long as the data is in the correct format. Just load the data into the program and compare curves, you will easily see what data you have loaded into the program and be able to compare it with the timeline, the timeline itself is very adaptable, from a yearly perspective down to seconds with the help of the Pandas package.

The toolbar is easy to use and can be used to save and adapt data to whatever you wish to present for others, or yourself to see.

4.1.1 Was the scope fulfilled?

The goal of the project was fulfilled all but being able to read .txt data files. Otherwise a program was created that includes a Graphical User Interface, it presents data in a graphic form, and the user can choose which data is to be presented.

The programming language used is open source and a well known language that is easy to use. And the documentation and presentation is done in English.

Time wise the project itself was well managed, the program was created on time and delivered to Höganäs AB. This report however has been delayed due to personal reasons.

4.1.2 Lessons learned

If I would do this project again, there are certain things I would do different, such as I went out of the standard Python syntax when I chose to present the program in a popupWindow, thus the code writing became a bit more complex and took unecessary time to implement new functions. The reasoning for making this change was only for aesthetic.

Over-engineering, I took a lot of time thinking about problems that could occur in the future while implementing new things, and thus a great deal of effort was put taking into account things that will never happen.

I took for granted that reading a .txt file would be easy, but there was no parsing for a simple .txt file and such in the end of the project that part was not implemented.

Due to time running up, the comments in the code was not very well done, in the future while writing programs some effort will be done to write comments simultaneously while writing the code itself, to explain methods and other functions.

4.1.3 Future works

There are many things that could be done with the program in the future, as a lot of time was taken to prepare for future upgrades, a list will follow with things that could be done.

1. There are many packages for Python that can be used to improve the program, so a good future works project would be adding more packages
2. Improve the graphic
3. Add more ways of reading data, such as json and directly from an sql database. The Pandas package currently comes with 18 readers for different sources of data
4. Add button or ways to increase the amount of comparative data that can be loaded into the program. This could be beneficial if more ovens are added to the sintering process

With the amount of people using the python languages, and the amount of packages being added on a yearly basis, there are a lot of possibilities to adapt the program for a specialized usage.

To use the program written by this bachelor thesis, just unzip the program into a folder, and start it with the run application.

Chapter 5
Appendix

5.0.1 Program Code

Run.py

```
#Run the app by python run.py
import Thesis_v4
def run():

    Thesis_v4.vp_start_gui()

if __name__ == '__main__':
    run()
```

Thesisv4.py

```
#! /usr/bin/env python
#  -*- coding: utf-8 -*-
#
# GUI module generated by PAGE version 6.2
#  in conjunction with Tcl version 8.6
#    Dec 07, 2021 01:26:04 PM CET  platform: Windows NT
import statistics
import sys

import myMethods
import globVar
from tkinter import *

try:
    import Tkinter as tk
except ImportError:
    import tkinter as tk

try:
```

```
    import ttk
    py3 = False
except ImportError:
    import tkinter.ttk as ttk
    py3 = True

import Thesis_v4_support

def vp_start_gui():
    '''Starting point when module is the main routine.'''
    global val, w, root
    root = tk.Tk()
    top = GUIthesis (root)
    Thesis_v4_support.init(root, top)
    root.mainloop()

w = None
def create_GUIthesis(rt, *args, **kwargs):
    '''Starting point when module is imported by another
                                    module.
       Correct form of call: 'create_GUIthesis(root, *args, **
                                    kwargs)' .'''
    global w, w_win, root
    root = rt
    w = tk.Toplevel (root)
    top = GUIthesis (w)
    Thesis_v4_support.init(w, top, *args, **kwargs)
    return (w, top)

def destroy_GUIthesis():
    global w
    w.destroy()
    w = None

#Manually written methods------------------

def clearSwindow(self,top):

    #Destroy old window
    self.Scrolledwindow1_f.destroy()

    #Create new window
    self.Scrolledwindow1 = ScrolledWindow(top)
    self.Scrolledwindow1.place(relx=0.042, rely=0.35,
                                    relheight=0.647
                               , relwidth=0.916)
    self.Scrolledwindow1.configure(background="white")
    self.Scrolledwindow1.configure(borderwidth="6")
    self.Scrolledwindow1.configure(highlightbackground="#
                                    d9d9d9")
    self.Scrolledwindow1.configure(highlightcolor="black")
```

```
    self.Scrolledwindow1.configure(insertbackground="black")
    self.Scrolledwindow1.configure(relief="groove")
    self.Scrolledwindow1.configure(selectbackground="blue")
    self.Scrolledwindow1.configure(selectforeground="white")
    self.color = self.Scrolledwindow1.cget("background")
    self.Scrolledwindow1_f = tk.Frame(self.Scrolledwindow1,
                                      background=self.color)
    self.Scrolledwindow1.create_window(0, 0, anchor='nw',
                            window=self.Scrolledwindow1_f)

def writeInEntryBox(self,id,fileLoc):
    if id == 'Button1':
        self.Entry1.delete(0,'end')
        newFileLoc1 = fileLoc.split("/")
        self.Entry1.insert(0,newFileLoc1[-1])
    if id == 'Button2':
        self.Entry2.delete(0, 'end')
        newFileLoc2 = fileLoc.split("/")
        self.Entry2.insert(0, newFileLoc2[-1])
    if id == 'Clear':
        self.Entry1.delete(0,'end')
        self.Entry2.delete(0,'end')
        self.Entry3.delete(0, 'end')
        self.Entry4.delete(0, 'end')
        self.Entry5.delete(0, 'end')
        self.Entry6.delete(0, 'end')
        self.Entry7.delete(0, 'end')
        self.Entry8.delete(0, 'end')
        self.Entry9.delete(0, 'end')
        self.Entry10.delete(0, 'end')

def calculatingValues(self,id,values1,values2):
    if id == 'both':
        self.Entry3.delete(0, 'end')
        self.Entry3.insert(0, min(values1))
        self.Entry4.delete(0, 'end')
        self.Entry4.insert(0, max(values1))
        self.Entry5.delete(0, 'end')
        self.Entry5.insert(0, sum(values1 / len(values1)))
        self.Entry6.delete(0, 'end')
        self.Entry6.insert(0, statistics.stdev(values1))
        self.Entry7.delete(0, 'end')
        self.Entry7.insert(0, min(values2))
        self.Entry8.delete(0, 'end')
        self.Entry8.insert(0, max(values2))
        self.Entry9.delete(0, 'end')
        self.Entry9.insert(0, sum(values2 / len(values2)))
        self.Entry10.delete(0, 'end')
        self.Entry10.insert(0, statistics.stdev(values2))

    if id == 'first':
```

```
        self.Entry3.delete(0, 'end')
        self.Entry3.insert(0,min(values1))
        self.Entry4.delete(0, 'end')
        self.Entry4.insert(0,max(values1))
        self.Entry5.delete(0, 'end')
        self.Entry5.insert(0, sum(values1/len(values1)))
        self.Entry6.delete(0, 'end')
        self.Entry6.insert(0, statistics.stdev(values1))
    if id == 'second':
        self.Entry7.delete(0, 'end')
        self.Entry7.insert(0, min(values2))
        self.Entry8.delete(0, 'end')
        self.Entry8.insert(0, max(values2))
        self.Entry9.delete(0, 'end')
        self.Entry9.insert(0, sum(values2 / len(values2)))
        self.Entry10.delete(0, 'end')
        self.Entry10.insert(0, statistics.stdev(values2))

def loadListItems(self,id,top):

    if id == 'drop1':
        self.df1List = globVar.df1.columns
        self.clicked1 = StringVar()
        self.clicked1.set(self.df1List[1])
        self.drop1 = OptionMenu(top, self.clicked1, *self.
                                    df1List)
        self.drop1.place(relx=0.138, rely=0.081, height=25,
                                    width=210)

    if id == 'drop2':
        self.df2List = globVar.df2.columns
        self.clicked2 = StringVar()
        self.clicked2.set(self.df2List[1])
        self.drop2 = OptionMenu(top, self.clicked2, *self.
                                    df2List)
        self.drop2.place(relx=0.138, rely=0.242, height=25,
                                    width=210)

def getNewWindow(self):
    return self.Scrolledwindow1

#---------------------------------------------

class GUIthesis:
    def __init__(self, top=None):
        '''This class configures and populates the toplevel
                                    window.
           top is the toplevel containing window.'''
        _bgcolor = '#d9d9d9'  # X11 color: 'gray85'
        _fgcolor = '#000000'  # X11 color: 'black'
        _compcolor = '#d9d9d9' # X11 color: 'gray85'
```

```
        _ana1color = '#d9d9d9' # X11 color: 'gray85'
        _ana2color = '#ececec' # Closest X11 color: 'gray92'
        self.style = ttk.Style()
        if sys.platform == "win32":
            self.style.theme_use('winnative')
        self.style.configure('.',background=_bgcolor)
        self.style.configure('.',foreground=_fgcolor)
        self.style.map('.',background=
            [('selected', _compcolor), ('active',_ana2color)])

        top.geometry("1184x830+310+124")
        top.minsize(120, 1)
        top.maxsize(5764, 1073)
        top.resizable(False,  False)
        top.title("Data Analyser for Ugn92 and Lucifer")
        top.configure(borderwidth="6")
        top.configure(background="slategray")
        top.configure(highlightbackground="#d9d9d9")
        top.configure(highlightcolor="black")
        top.configure(border=10)
        top.configure(relief="groove")


        #DropDownList with available columnNames

        globVar.df1Ready = False
        globVar.df2Ready = False

        self.df1List = []
        self.clicked1=StringVar()
        self.clicked1.set('Empty List')
        self.drop1 = OptionMenu(top, self.clicked1,'Empty')
        self.drop1.place(relx=0.138,rely=0.081, height=25,
                                        width=210)

        #self.Entry15.configure(font=('Times New Roman', 10, '
                                        bold'))


        self.df2List = []
        self.clicked2 = StringVar()
        self.clicked2.set('Empty List')
        self.drop2 = OptionMenu(top, self.clicked1, 'Empty')
        self.drop2.place(relx=0.138, rely=0.242, height=25,
                                        width=210)


        #Creating the GUI
        self.Scrolledwindow1 = ScrolledWindow(top)
        self.Scrolledwindow1.place(relx=0.042, rely=0.35,
                                        relheight=0.647
                , relwidth=0.916)
```

```
self.Scrolledwindow1.configure(background="white")
self.Scrolledwindow1.configure(borderwidth="6")
self.Scrolledwindow1.configure(highlightbackground="#
                               d9d9d9")
self.Scrolledwindow1.configure(highlightcolor="black")
self.Scrolledwindow1.configure(insertbackground="black
                               ")
self.Scrolledwindow1.configure(relief="groove")
self.Scrolledwindow1.configure(selectbackground="blue"
                               )
self.Scrolledwindow1.configure(selectforeground="white
                               ")
self.color = self.Scrolledwindow1.cget("background")
self.Scrolledwindow1_f = tk.Frame(self.Scrolledwindow1
                               ,
                     background=self.color)
self.Scrolledwindow1.create_window(0, 0, anchor='nw',
                   window=self.Scrolledwindow1_f)

globVar.checkButton = BooleanVar()
self.Checkbutton1 = tk.Checkbutton(top)
self.Checkbutton1.place(relx=0.18, rely=0.154,
                               relheight=0.03
, relwidth = 0.055)
self.Checkbutton1.configure(variable=globVar.
                               checkButton)
self.Checkbutton1.configure(activebackground="#ececec"
                               )
self.Checkbutton1.configure(activeforeground="#000000"
                               )
self.Checkbutton1.configure(background="slategray")
self.Checkbutton1.configure(disabledforeground="#
                               a3a3a3")
self.Checkbutton1.configure(foreground="#000000")
self.Checkbutton1.configure(highlightbackground="#
                               d9d9d9")
self.Checkbutton1.configure(highlightcolor="black")
self.Checkbutton1.configure(justify='left')
self.Checkbutton1.configure(relief="flat")
self.Checkbutton1.configure(text='Subplot')
self.Checkbutton1.configure(font=('Times New Roman',
                               10, 'bold'))

self.Button1 = tk.Button(top)
self.Button1.place(relx=0.042, rely=0.081, height=25,
                               width=110)
self.Button1.configure(activebackground="#ececec")
self.Button1.configure(activeforeground="#000000")
self.Button1.configure(background="#d9d9d9")
self.Button1.configure(command=lambda:myMethods.
                               loadFile(self,'Button1'
```

```
                                    ,top))
        self.Button1.configure(disabledforeground="#a3a3a3")
        self.Button1.configure(foreground="#000000")
        self.Button1.configure(highlightbackground="#d9d9d9")
        self.Button1.configure(highlightcolor="black")
        self.Button1.configure(pady="0")
        self.Button1.configure(text='''Load DataFrame1''')

        self.Button2 = tk.Button(top)
        self.Button2.place(relx=0.042, rely=0.242, height=25,
                                    width=110)
        self.Button2.configure(activebackground="#ececec")
        self.Button2.configure(activeforeground="#000000")
        self.Button2.configure(background="#d9d9d9")
        self.Button2.configure(command=lambda:myMethods.
                                    loadFile(self,'Button2'
                                    ,top))
        self.Button2.configure(disabledforeground="#a3a3a3")
        self.Button2.configure(foreground="#000000")
        self.Button2.configure(highlightbackground="#d9d9d9")
        self.Button2.configure(highlightcolor="black")
        self.Button2.configure(pady="0")
        self.Button2.configure(text='''Load DataFrame2''')

        self.Button3 = tk.Button(top)
        self.Button3.place(relx=0.042, rely=0.161, height=50,
                                    width=80)
        self.Button3.configure(activebackground="#ececec")
        self.Button3.configure(activeforeground="#000000")
        self.Button3.configure(background="#d9d9d9")
        self.Button3.configure(command=lambda: myMethods.
                                    plotGraph(self,top))
        self.Button3.configure(disabledforeground="#a3a3a3")
        self.Button3.configure(foreground="#000000")
        self.Button3.configure(highlightbackground="#d9d9d9")
        self.Button3.configure(highlightcolor="black")
        self.Button3.configure(pady="0")
        self.Button3.configure(text='''Run''')
        self.Button3.configure(font=('Times New Roman', 10, '
                                    bold'))

        self.Button5 = tk.Button(top)
        self.Button5.place(relx=0.111, rely=0.161, height=50,
                                    width=80)
        self.Button5.configure(activebackground="#ececec")
        self.Button5.configure(activeforeground="#000000")
        self.Button5.configure(background="#d9d9d9")
        self.Button5.configure(command=lambda: myMethods.
                                    clearEntry(self, 'Clear
                                    ', top))
        self.Button5.configure(disabledforeground="#a3a3a3")
```

```
        self.Button5.configure(foreground="#000000")
        self.Button5.configure(highlightbackground="#d9d9d9")
        self.Button5.configure(highlightcolor="black")
        self.Button5.configure(pady="0")
        self.Button5.configure(text='''Reset''')
        self.Button5.configure(font=('Times New Roman', 10, '
                                     bold'))

        self.Entry1 = tk.Entry(top)
        self.Entry1.place(relx=0.042, rely=0.115, height=20,
                                     relwidth=0.276)
        self.Entry1.configure(background="white")
        self.Entry1.configure(relief='groove')
        self.Entry1.configure(borderwidth=2)
        self.Entry1.configure(disabledforeground="#a3a3a3")
        self.Entry1.configure(font="TkFixedFont")
        self.Entry1.configure(foreground="#000000")
        self.Entry1.configure(highlightbackground="#d9d9d9")
        self.Entry1.configure(highlightcolor="black")
        self.Entry1.configure(insertbackground="black")
        self.Entry1.configure(selectbackground="blue")
        self.Entry1.configure(selectforeground="white")

        self.Entry2 = tk.Entry(top)
        self.Entry2.place(relx=0.042, rely=0.276, height=20,
                                     relwidth=0.276)
        self.Entry2.configure(background="white")
        self.Entry2.configure(relief='groove')
        self.Entry2.configure(borderwidth=2)
        self.Entry2.configure(disabledforeground="#a3a3a3")
        self.Entry2.configure(font="TkFixedFont")
        self.Entry2.configure(foreground="#000000")
        self.Entry2.configure(highlightbackground="#d9d9d9")
        self.Entry2.configure(highlightcolor="black")
        self.Entry2.configure(insertbackground="black")
        self.Entry2.configure(selectbackground="blue")
        self.Entry2.configure(selectforeground="white")

        self.Entry3 = tk.Entry(top)
        self.Entry3.place(relx=0.414, rely=0.081, height=20,
                                     relwidth=0.073)
        self.Entry3.configure(background="white")
        self.Entry3.configure(disabledforeground="#a3a3a3")
        self.Entry3.configure(font="TkFixedFont")
        self.Entry3.configure(foreground="#000000")
        self.Entry3.configure(highlightbackground="#d9d9d9")
        self.Entry3.configure(highlightcolor="black")
        self.Entry3.configure(insertbackground="black")
        self.Entry3.configure(selectbackground="blue")
        self.Entry3.configure(selectforeground="white")
```

```
        self.Entry4 = tk.Entry(top)
        self.Entry4.place(relx=0.414, rely=0.104, height=20,
                                        relwidth=0.073)
        self.Entry4.configure(background="white")
        self.Entry4.configure(disabledforeground="#a3a3a3")
        self.Entry4.configure(font="TkFixedFont")
        self.Entry4.configure(foreground="#000000")
        self.Entry4.configure(highlightbackground="#d9d9d9")
        self.Entry4.configure(highlightcolor="black")
        self.Entry4.configure(insertbackground="black")
        self.Entry4.configure(selectbackground="blue")
        self.Entry4.configure(selectforeground="white")

        self.Entry5 = tk.Entry(top)
        self.Entry5.place(relx=0.414, rely=0.127, height=20,
                                        relwidth=0.073)
        self.Entry5.configure(background="white")
        self.Entry5.configure(disabledforeground="#a3a3a3")
        self.Entry5.configure(font="TkFixedFont")
        self.Entry5.configure(foreground="#000000")
        self.Entry5.configure(highlightbackground="#d9d9d9")
        self.Entry5.configure(highlightcolor="black")
        self.Entry5.configure(insertbackground="black")
        self.Entry5.configure(selectbackground="blue")
        self.Entry5.configure(selectforeground="white")

        self.Entry6 = tk.Entry(top)
        self.Entry6.place(relx=0.414, rely=0.15, height=20,
                                        relwidth=0.073)
        self.Entry6.configure(background="white")
        self.Entry6.configure(disabledforeground="#a3a3a3")
        self.Entry6.configure(font="TkFixedFont")
        self.Entry6.configure(foreground="#000000")
        self.Entry6.configure(highlightbackground="#d9d9d9")
        self.Entry6.configure(highlightcolor="black")
        self.Entry6.configure(insertbackground="black")
        self.Entry6.configure(selectbackground="blue")
        self.Entry6.configure(selectforeground="white")

        self.Entry7 = tk.Entry(top)
        self.Entry7.place(relx=0.414, rely=0.242, height=20,
                                        relwidth=0.073)
        self.Entry7.configure(background="white")
        self.Entry7.configure(disabledforeground="#a3a3a3")
        self.Entry7.configure(font="TkFixedFont")
        self.Entry7.configure(foreground="#000000")
        self.Entry7.configure(highlightbackground="#d9d9d9")
        self.Entry7.configure(highlightcolor="black")
        self.Entry7.configure(insertbackground="black")
        self.Entry7.configure(selectbackground="blue")
        self.Entry7.configure(selectforeground="white")
```

```
self.Entry8 = tk.Entry(top)
self.Entry8.place(relx=0.414, rely=0.265, height=20,
                                relwidth=0.073)
self.Entry8.configure(background="white")
self.Entry8.configure(disabledforeground="#a3a3a3")
self.Entry8.configure(font="TkFixedFont")
self.Entry8.configure(foreground="#000000")
self.Entry8.configure(highlightbackground="#d9d9d9")
self.Entry8.configure(highlightcolor="black")
self.Entry8.configure(insertbackground="black")
self.Entry8.configure(selectbackground="blue")
self.Entry8.configure(selectforeground="white")

self.Entry9 = tk.Entry(top)
self.Entry9.place(relx=0.414, rely=0.288, height=20,
                                relwidth=0.073)
self.Entry9.configure(background="white")
self.Entry9.configure(disabledforeground="#a3a3a3")
self.Entry9.configure(font="TkFixedFont")
self.Entry9.configure(foreground="#000000")
self.Entry9.configure(highlightbackground="#d9d9d9")
self.Entry9.configure(highlightcolor="black")
self.Entry9.configure(insertbackground="black")
self.Entry9.configure(selectbackground="blue")
self.Entry9.configure(selectforeground="white")

self.Entry10 = tk.Entry(top)
self.Entry10.place(relx=0.414, rely=0.311, height=20,
                                relwidth=0.073)
self.Entry10.configure(background="white")
self.Entry10.configure(disabledforeground="#a3a3a3")
self.Entry10.configure(font="TkFixedFont")
self.Entry10.configure(foreground="#000000")
self.Entry10.configure(highlightbackground="#d9d9d9")
self.Entry10.configure(highlightcolor="black")
self.Entry10.configure(insertbackground="black")
self.Entry10.configure(selectbackground="blue")
self.Entry10.configure(selectforeground="white")

self.Entry11 = tk.Entry(top)
self.Entry11.place(relx=0.34, rely=0.081, height=20,
                                relwidth=0.073)
self.Entry11.configure(background="slategray")
self.Entry11.configure(relief='flat')
self.Entry11.configure(disabledforeground="#a3a3a3")
self.Entry11.configure(font="TkFixedFont")
self.Entry11.configure(foreground="#000000")
self.Entry11.configure(highlightbackground="#d9d9d9")
self.Entry11.configure(highlightcolor="black")
self.Entry11.configure(insertbackground="black")
```

```
        self.Entry11.configure(selectbackground="blue")
        self.Entry11.configure(selectforeground="white")
        self.Entry11.insert(0, 'Minimum:')
        self.Entry11.configure(justify=RIGHT)
        self.Entry11.configure(font=('Times New Roman', 10, '
                                    bold'))

        self.Entry12 = tk.Entry(top)
        self.Entry12.place(relx=0.34, rely=0.104, height=20,
                                    relwidth=0.073)
        self.Entry12.configure(background="slategray")
        self.Entry12.configure(relief='flat')
        self.Entry12.configure(disabledforeground="#a3a3a3")
        self.Entry12.configure(font="TkFixedFont")
        self.Entry12.configure(foreground="#000000")
        self.Entry12.configure(highlightbackground="#d9d9d9")
        self.Entry12.configure(highlightcolor="black")
        self.Entry12.configure(insertbackground="black")
        self.Entry12.configure(selectbackground="blue")
        self.Entry12.configure(selectforeground="white")
        self.Entry12.insert(0, 'Maximum:')
        self.Entry12.configure(justify=RIGHT)
        self.Entry12.configure(font=('Times New Roman', 10, '
                                    bold'))

        self.Entry13 = tk.Entry(top)
        self.Entry13.place(relx=0.34, rely=0.127, height=20,
                                    relwidth=0.073)
        self.Entry13.configure(background="slategray")
        self.Entry13.configure(relief='flat')
        self.Entry13.configure(disabledforeground="#a3a3a3")
        self.Entry13.configure(font="TkFixedFont")
        self.Entry13.configure(foreground="#000000")
        self.Entry13.configure(highlightbackground="#d9d9d9")
        self.Entry13.configure(highlightcolor="black")
        self.Entry13.configure(insertbackground="black")
        self.Entry13.configure(selectbackground="blue")
        self.Entry13.configure(selectforeground="white")
        self.Entry13.insert(0,'Average:')
        self.Entry13.configure(justify=RIGHT)
        self.Entry13.configure(font=('Times New Roman', 10, '
                                    bold'))

        self.Entry14 = tk.Entry(top)
        self.Entry14.place(relx=0.34, rely=0.150, height=20,
                                    relwidth=0.073)
        self.Entry14.configure(background="slategray")
        self.Entry14.configure(relief='flat')
        self.Entry14.configure(disabledforeground="#a3a3a3")
        self.Entry14.configure(font="TkFixedFont")
        self.Entry14.configure(foreground="#000000")
```

```
        self.Entry14.configure(highlightbackground="#d9d9d9")
        self.Entry14.configure(highlightcolor="black")
        self.Entry14.configure(insertbackground="black")
        self.Entry14.configure(selectbackground="blue")
        self.Entry14.configure(selectforeground="white")
        self.Entry14.insert(0,'StDev:')
        self.Entry14.configure(justify=RIGHT)
        self.Entry14.configure(font=('Times New Roman', 10, '
                                     bold'))

        self.Entry15 = tk.Entry(top)
        self.Entry15.place(relx=0.34, rely=0.242, height=20,
                                     relwidth=0.073)
        self.Entry15.configure(background="slategray")
        self.Entry15.configure(relief='flat')
        self.Entry15.configure(disabledforeground="#a3a3a3")
        self.Entry15.configure(font="TkFixedFont")
        self.Entry15.configure(foreground="#000000")
        self.Entry15.configure(highlightbackground="#d9d9d9")
        self.Entry15.configure(highlightcolor="black")
        self.Entry15.configure(insertbackground="black")
        self.Entry15.configure(selectbackground="blue")
        self.Entry15.configure(selectforeground="white")
        self.Entry15.insert(0,'Minimum:')
        self.Entry15.configure(justify=RIGHT)
        self.Entry15.configure(font=('Times New Roman', 10, '
                                     bold'))

        self.Entry16 = tk.Entry(top)
        self.Entry16.place(relx=0.34, rely=0.265, height=20,
                                     relwidth=0.073)
        self.Entry16.configure(background="slategray")
        self.Entry16.configure(relief='flat')
        self.Entry16.configure(disabledforeground="#a3a3a3")
        self.Entry16.configure(font="TkFixedFont")
        self.Entry16.configure(foreground="#000000")
        self.Entry16.configure(highlightbackground="#d9d9d9")
        self.Entry16.configure(highlightcolor="black")
        self.Entry16.configure(insertbackground="black")
        self.Entry16.configure(selectbackground="blue")
        self.Entry16.configure(selectforeground="white")
        self.Entry16.insert(0,'Maximum:')
        self.Entry16.configure(justify=RIGHT)
        self.Entry16.configure(font=('Times New Roman', 10, '
                                     bold'))

        self.Entry17 = tk.Entry(top)
        self.Entry17.place(relx=0.34, rely=0.288, height=20,
                                     relwidth=0.073)
        self.Entry17.configure(background="slategray")
        self.Entry17.configure(relief='flat')
```

```
        self.Entry17.configure(disabledforeground="#a3a3a3")
        self.Entry17.configure(font="TkFixedFont")
        self.Entry17.configure(foreground="#000000")
        self.Entry17.configure(highlightbackground="#d9d9d9")
        self.Entry17.configure(highlightcolor="black")
        self.Entry17.configure(insertbackground="black")
        self.Entry17.configure(selectbackground="blue")
        self.Entry17.configure(selectforeground="white")
        self.Entry17.insert(0,'Average:')
        self.Entry17.configure(justify=RIGHT)
        self.Entry17.configure(font=('Times New Roman', 10, '
                                    bold'))

        self.Entry18 = tk.Entry(top)
        self.Entry18.place(relx=0.34, rely=0.311, height=20,
                                    relwidth=0.073)
        self.Entry18.configure(background="slategray")
        self.Entry18.configure(relief='flat')
        self.Entry18.configure(disabledforeground="#a3a3a3")
        self.Entry18.configure(font="TkFixedFont")
        self.Entry18.configure(foreground="#000000")
        self.Entry18.configure(highlightbackground="#d9d9d9")
        self.Entry18.configure(highlightcolor="black")
        self.Entry18.configure(insertbackground="black")
        self.Entry18.configure(selectbackground="blue")
        self.Entry18.configure(selectforeground="white")
        self.Entry18.insert(0,'StDev:')
        self.Entry18.configure(justify=RIGHT)
        self.Entry18.configure(font = ('Times New Roman',10, '
                                    bold'))

# The following code is added to facilitate the Scrolled
                                widgets you specified.
class AutoScroll(object):
    '''Configure the scrollbars for a widget.'''
    def __init__(self, master):
        #  Rozen. Added the try-except clauses so that this
                                    class
        #  could be used for scrolled entry widget for which
                                    vertical
        #  scrolling is not supported. 5/7/14.
        try:
            vsb = ttk.Scrollbar(master, orient='vertical',
                                        command=self.yview)
        except:
            pass
        hsb = ttk.Scrollbar(master, orient='horizontal',
                                    command=self.xview)
        try:
            self.configure(yscrollcommand=self._autoscroll(vsb
                                        ))
```

```
        except:
            pass
        self.configure(xscrollcommand=self._autoscroll(hsb))
        self.grid(column=0, row=0, sticky='nsew')
        try:
            vsb.grid(column=1, row=0, sticky='ns')
        except:
            pass
        hsb.grid(column=0, row=1, sticky='ew')
        master.grid_columnconfigure(0, weight=1)
        master.grid_rowconfigure(0, weight=1)
        # Copy geometry methods of master  (taken from
                                        ScrolledText.py)
        if py3:
            methods = tk.Pack.__dict__.keys() | tk.Grid.
                                                __dict__.keys() \
                  | tk.Place.__dict__.keys()
        else:
            methods = tk.Pack.__dict__.keys() + tk.Grid.
                                                __dict__.keys() \
                  + tk.Place.__dict__.keys()
        for meth in methods:
            if meth[0] != '_' and meth not in ('config', '
                                                configure'):
                setattr(self, meth, getattr(master, meth))

    @staticmethod
    def _autoscroll(sbar):
        '''Hide and show scrollbar as needed.'''
        def wrapped(first, last):
            first, last = float(first), float(last)
            if first <= 0 and last >= 1:
                sbar.grid_remove()
            else:
                sbar.grid()
            sbar.set(first, last)
        return wrapped

    def __str__(self):
        return str(self.master)

def _create_container(func):
    '''Creates a ttk Frame with a given master, and use this
                                    new frame to
    place the scrollbars and the widget.'''
    def wrapped(cls, master, **kw):
        container = ttk.Frame(master)
        container.bind('<Enter>', lambda e:
                                        _bound_to_mousewheel(e,
                                         container))
```

```
        container.bind('<Leave>', lambda e:
                                        _unbound_to_mousewheel(
                                        e, container))
        return func(cls, container, **kw)
    return wrapped

class ScrolledWindow(AutoScroll, tk.Canvas):
    '''A standard Tkinter Canvas widget with scrollbars that
                                    will
    automatically show/hide as needed.'''
    @_create_container
    def __init__(self, master, **kw):
        tk.Canvas.__init__(self, master, **kw)
        AutoScroll.__init__(self, master)

import platform
def _bound_to_mousewheel(event, widget):
    child = widget.winfo_children()[0]
    if platform.system() == 'Windows' or platform.system() ==
                                    'Darwin':
        child.bind_all('<MouseWheel>', lambda e:
                                        _on_mousewheel(e, child
                                        ))
        child.bind_all('<Shift-MouseWheel>', lambda e:
                                        _on_shiftmouse(e, child
                                        ))
    else:
        child.bind_all('<Button-4>', lambda e: _on_mousewheel(
                                        e, child))
        child.bind_all('<Button-5>', lambda e: _on_mousewheel(
                                        e, child))
        child.bind_all('<Shift-Button-4>', lambda e:
                                        _on_shiftmouse(e, child
                                        ))
        child.bind_all('<Shift-Button-5>', lambda e:
                                        _on_shiftmouse(e, child
                                        ))

def _unbound_to_mousewheel(event, widget):
    if platform.system() == 'Windows' or platform.system() ==
                                    'Darwin':
        widget.unbind_all('<MouseWheel>')
        widget.unbind_all('<Shift-MouseWheel>')
    else:
        widget.unbind_all('<Button-4>')
        widget.unbind_all('<Button-5>')
        widget.unbind_all('<Shift-Button-4>')
        widget.unbind_all('<Shift-Button-5>')

def _on_mousewheel(event, widget):
    if platform.system() == 'Windows':
```

```
        widget.yview_scroll(-1*int(event.delta/120),'units')
    elif platform.system() == 'Darwin':
        widget.yview_scroll(-1*int(event.delta),'units')
    else:
        if event.num == 4:
            widget.yview_scroll(-1, 'units')
        elif event.num == 5:
            widget.yview_scroll(1, 'units')

def _on_shiftmouse(event, widget):
    if platform.system() == 'Windows':
        widget.xview_scroll(-1*int(event.delta/120), 'units')
    elif platform.system() == 'Darwin':
        widget.xview_scroll(-1*int(event.delta), 'units')
    else:
        if event.num == 4:
            widget.xview_scroll(-1, 'units')
        elif event.num == 5:
            widget.xview_scroll(1, 'units')

if __name__ == '__main__':
    vp_start_gui()
```

Thesisv4support.py

```
#! /usr/bin/env python
#  -*- coding: utf-8 -*-
#
# Support module generated by PAGE version 6.2
#  in conjunction with Tcl version 8.6
#    Dec 07, 2021 01:26:08 PM CET  platform: Windows NT

import sys

try:
    import Tkinter as tk
except ImportError:
    import tkinter as tk

try:
    import ttk
    py3 = False
except ImportError:
    import tkinter.ttk as ttk
    py3 = True

def init(top, gui, *args, **kwargs):
    global w, top_level, root
    w = gui
```

```
    top_level = top
    root = top

def destroy_window():
    # Function which closes the window.
    global top_level
    top_level.destroy()
    top_level = None

if __name__ == '__main__':
    import Thesis_v4
    Thesis_v4.vp_start_gui()
```

myMethods.py

```
import tkinter
from tkinter import filedialog
import pandas as pd
import Thesis_v4
import globVar
from matplotlib.backends.backend_tkagg import
                                FigureCanvasTkAgg,
                                NavigationToolbar2Tk
from matplotlib.figure import Figure
from matplotlib import style

def loadFile(self,id,top):
    fileLoc = filedialog.askopenfilename(initialdir="", title=
                                'Select a file',
                filetypes=(('csv files', '*.csv'), ('txt files
                                        ', '*.txt')))

    if id == 'Button1':
        globVar.df1 = pd.read_csv(fileLoc)
        globVar.df1Ready = True
        Thesis_v4.writeInEntryBox(self,id,fileLoc)
        Thesis_v4.loadListItems(self, 'drop1',top)

    if id == 'Button2':
        globVar.df2 = pd.read_csv(fileLoc)
        globVar.df2Ready = True
        Thesis_v4.writeInEntryBox(self,id,fileLoc)
        Thesis_v4.loadListItems(self, 'drop2', top)

def clearEntry(self, id,top):

    if id == 'Clear':

        Thesis_v4.clearSwindow(self,top)
```

```
        Thesis_v4.writeInEntryBox(self,'Clear','not required
                                        for clearing')

def plotGraph(self, top):

    Thesis_v4.clearSwindow(self,top)
    sWindow = Thesis_v4.getNewWindow(self)

    f = Figure(figsize=(5,5), dpi=100)
    style.use('seaborn')


    if globVar.df1Ready and globVar.df2Ready:
        x0 = globVar.df1[r'Timestamp']
        y0 = globVar.df1[self.clicked1.get()]
        x1 = globVar.df2[r'Timestamp']
        y1 = globVar.df2[self.clicked2.get()]

        x0 = pd.to_datetime(x0,yearfirst=True)
        x1 = pd.to_datetime(x1,yearfirst=True)

        if globVar.checkButton.get():

            a = f.add_subplot(211)
            a.plot(x0, y0, label=y0.name)
            a.legend()
            temp = y0.name.split(":")
            a.set_ylabel(temp[-1])

            title = " Graph of: " + str(y0.name) + " vs " +
                                            str(y1.name)
            a.set_title(title)

            a = f.add_subplot(212)
            a.plot(x1, y1, label=y1.name)
            a.legend()
            temp = y1.name.split(":")
            a.set_ylabel(temp[-1])
            a.set_xlabel("Timestamp")

        else:

            a = f.add_subplot(111)

            temp = y0.name.split(":")
            temp1 = y1.name.split(":")

            a.plot(x0, y0, label=y0.name)
            a.set_ylabel(temp[-1],color='blue')
            a2 = a.twinx()
            a.plot(x1, y1, label=y1.name)
```

```
            a2.set_ylabel(temp1[-1],color='green')

            title = " Graph of: " + str(y0.name) + " vs " +
                                            str(y1.name)
            a.set_title(title)
            a.legend()
            a.set_xlabel("Timestamp")

        Thesis_v4.calculatingValues(self, 'both', y0, y1)

    if globVar.df1Ready and not globVar.df2Ready:
        x0 = globVar.df1[r'Timestamp']
        y0 = globVar.df1[self.clicked1.get()]
        x0 = pd.to_datetime(x0, yearfirst=True)

        a = f.add_subplot(111)
        a.plot(x0, y0, label=y0.name)

        title = "Graph of: \n"+str(y0.name)
        a.set_title(title)
        temp = y0.name.split(":")
        a.set_ylabel(temp[-1])
        a.legend()
        a.set_xlabel("Timestamp")

        Thesis_v4.calculatingValues(self,'first',y0,0)

    if not globVar.df1Ready and globVar.df2Ready:
        x1 = globVar.df2[r'Timestamp']
        y1 = globVar.df2[self.clicked2.get()]
        x1 = pd.to_datetime(x1, yearfirst=True)

        a = f.add_subplot(111)
        a.plot(x1, y1, label=y1.name)

        title = "Graph of: \n" + str(y1.name)
        a.set_title(title)
        temp = y1.name.split(":")
        a.set_ylabel(temp[-1])
        a.legend()
        a.set_xlabel("Timestamp")

        Thesis_v4.calculatingValues(self, 'second', 0, y1)

    canvas = FigureCanvasTkAgg(f, sWindow)
    canvas.draw()
    canvas.get_tk_widget().pack(side=tkinter.TOP, fill=tkinter
                                    .BOTH, expand=True)
    toolbar = NavigationToolbar2Tk(canvas, sWindow)
    toolbar.update()
```

```
    canvas._tkcanvas.pack(side=tkinter.TOP, fill=tkinter.BOTH,
                                        expand=True)
```

globVar.py

```
    #Theoretical constants that will be used
#and of course important variables, nameChange inbound in 3...
                                2.........


global df1
global df1Ready
global df1Clicked

global df2
global df2Ready
global df2Clicked

global checkButton
```

References

[1] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[2] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[3] Fredrik Lundh. An introduction to tkinter. *URL: www. pythonware. com/library/tkinter/introduction/index. htm*, 1999.

[4] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[5] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Printed by Tryckeriet i E-huset, Lund 2024

LUND
UNIVERSITY

Series of Master´s theses
Department of Electrical and Information Technology
LU/LTH-EIT 2024-1029
http://www.eit.lth.se