Developing a Graphical User Interface for Thermodynamic Models *Thermopack* in Python with PyQt

Bendik Støa Sannes

January 3, 2025

Contents

1 Introduction

One tool for contributing to lower CO_2 emissions is Carbon Capture and Storage (CCS) technology. SINTEF has since 2011 developed a set of thermodynamic models, *Thermopack*, to describe physical properties for CO_2 mixtures. This is used to test, varify and compare both existing and new theoretical models. To make *Thermopack* more accessible, the aim of this project has been to develop a Graphical User Interface (GUI), where the user can interact with *Thermopack* in an easier way.

To create this GUI, a Python plugin called PyQt5 was used. This is a Python bundle for Qt, which is a well documented, open-source tool for creating cross-platform GUIs. It comes with a designer tool, Qt Designer, which makes it easy and fast to design and create code.

2 PyQt5

This project has used the Python bundle for the latest Qt version, PyQt5. The whole PyQt5 application is built of many individual widgets. These widgets can contain other widgets, and they can send signals to each other to pass data or handle user actions. The Qt Designer tool is a convenient way of creating a layout, where you can see what you make without having to create the layout code, which saves a lot of time and energy. The different widgets can be styled and customized if desired with a stylesheet.

2.1 Widgets

All parts of the GUI are called *Widgets*. These can contain information, take inputs from the user, or contain a set of other widgets. The main widget containing the others is called the *Window*. Some examples of simple widgets are the *QPushButton* which is a pushable button, or the *QLineEdit* which can take in user input from the keyboard. Other built-in widgets are the *QTabWidget* which can hold other widgets in different tabs, or the *QStackedWidget* which contains a stack of multiple widgets, and only displays one widget at the time. A complete list of available widgets is found in the Qt documentation `https://doc.qt.io/archives/qt-5.11/widget-classes.html`. This documentation is for C++, but most of the functionality can easily be used in the same fashion in PyQt.

2.2 Signals and Slots

For widgets to communicate with each other, or to handle user interaction or input, they use *signals* and *slots*. Different widgets have different built-in signals, but it is also possible to create your own. A slot is a function which is called if a signal is emitted and connected to it. As an example, a QPushButton for closing a window emits a signal *clicked*, and can be connected to a slot in the following way:

```
self.close_button.clicked.connect(self.close)
```

More on signals and slots can be found in the Qt documentation `https://doc.qt.io/qt-5/signalsandslots.html`.

2.3 Qt Designer

The Qt Designer is a handy GUI builder tool. Here, you are shown a list of different widgets which you can drag-and-drop into a preview of the application window. You can set different layouts of your widgets (horizontal layout, vertical layout, grid layout, form layout) saving a lot of time and many lines of code. You can also set basic properties of your widgets including its object name to be referenced in further code. To install the Qt Designer for PyQt5, it is necessary to install the Python plugin *pyqt5-tools*:

```
pip install pyqt5-tools
```

When this is done, the application is found in the /Lib/site-packages/pyqt5_tools/Qt/bin folder where you have installed your Python environment on your system. With the Qt Designer, you can save your layouts as .ui files. To set the layout for a widget, one can do as follows:

```
from PyQt5.QtWidgets import QWidget
from PyQt5.uic import LoadU

class MyWidget(QWidget):
    def __init__(self):
        super().__init__()
        loadUi("path/to/layout.ui", self)
```

2.4 Stylesheet

To make the widgets somewhat more intersting and less geeneric, it is possible to style them by setting their styleSheet. This can be done in several ways, either in the code by calling a widgets setStyleSheet() method, in the Qt Designer or by having a stylesheet.qss file and load this for the whole application. Different widgets have different parts that can be styled, and an overview of this is found in the documentation `https://doc.qt.io/qt-5/stylesheet-reference.html`. The styling syntax is very similar to CSS, but not all properties found in CSS is availbale for all widgets.

3 Thermopack GUI

The application is built up of three main parts. The first window that is opened is where the compositions are chosen, and the model setup is set. When this is done, you can go to Plot Mode or to Calculation Mode. In Plot Mode, it is possible to choose between a set of plots and download the plotted data.

In Calculation Mode, a table showing physical data from flash calculations is displayed, which can also be downloaded. It is possible to save your setup as a JSON-file, and later load it.

3.1 Main Window

The main window consists of a menu to the left displaying the currently set compositions and models. From here you can open the ComponentSelectWidget where you can create or edit your compositions, or the ModelSelectWidget for setting your models. Depending on your model setup, you can see and change different parameters. The currently availbale ParameterWidgets are for the Van der Waals, Huron Vidal 1 and 2 mixing rules, and for the CPA, PC-SAFT and SAFT-VR Mie models. When both a composition and a model is set, you may proceed to either Plot Mode or Calculation Mode by pressing the icons in the top toolbar.

3.2 Plot Mode

When entering Plot Mode, one composition and one model setup has to be chosen. Here, you choose a plot type, specify your parameters and plotting preferences, and set molar fractions for the composition. Plots will be shown in a matplotlib sub-window (on the MplCanvas), and the plotted data can be downloaded as a csv file.

3.3 Calculation Mode

As for the Plot Mode, one composition and one model setup must be chosen before entering Calculation Mode. Here, you choose the type of flash calculation, set initial guesses and molar fractions. Calculated values will be shown in a table, which is also available to download to a csv file.

3.4 Storing data

To keep track of data shared between the different widgets and windows, a Python dictionary is used. This is because if one part of the application makes changes to the dictionary, it is automatically changed in other parts of the application. When the user wants to save data, the dictionary is converted to JSON and stored in a json file, done with Python's json module.

4 Conclusion

PyQt has been nice to work with. First of all, Qt is well documented, its community is very helpful and solutions to many questions of all levels can be found. The Qt Designer saves a lot of time, so PyQt is a good way of creating GUI's in Python. To create such a GUI, one could as well have used C++ and gotten the same result. It would take somewhat longer time, but most of the

examples on the Internet uses C++, so this seems like a more common way to create GUI's. And there are some things you can't control in the same way in Python as in C++, such as with pointers.

As of now, there are still things left to be done with the GUI. More of the widgets can be styled to create a more complete feel. In the Plot Mode, it could be added an option for plotting experimental data and compare this to the existing models. Plot Mode can also have options for changing units. Not all calculations throw an exception if the solution does not converge, and thus sometimes the application stops. This has to be fixed before distributing the application.