Groups: You are allowed to work with two other persons to complete this assignment. You should submit your work by way of a github repository url. Make sure to invite the instructor and TA to be collaborators on your repository.

**P1:** Write a program using the Model-View-Controller (MVC) pattern to read and display information from a pump file as shown below.

*Your program will have the following features*:

1. It will have a graphical user-interface (GUI) for selecting the file and presenting the output.
2. The clicked signal of the *Read File and Calculate* button will open a dialog box for searching the directories of your hard drive to navigate to location of the data file.
3. Upon selecting the data file, the path to the data file is stored, such that a second click of the button brings you back to the same location. Then, it will ***read and parse*** the selected file and display the required output (as demonstrated in the screen capture below).
4. The plot should have labels on both y-axes, the x axis, legends for each curve and a title. You should use the minimized sum of squared errors for a quadric and cubic fit to the Head and Efficiency data, respectively. (see <https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html> )

Diagram

Description automatically generated with low confidence

**P2**: I have modified the Rankine cycle program to utilize the Model-View-Controller design pattern. Note that in Rankine\_Classes.py, you now have the classes: rankineModel, rankineController, and rankineView. In Rankine\_app\_MVC.py, we instantiate a single controller object (self.RC) and then submit a tuple of widgets to self.RC.updateModel in the calculate function. The updating of the model data AND the view are handled internally by the controller.

Modifications you need to make:

1. The current program reads from saturated properties and superheated property tables to calculate the thermodynamic states. You should update the program to use pyXsteam (see work from HW7) instead.
2. When the user selects the radio button T High, change the text in the Turbine Inlet line edit equal to the saturation temperature at P High.
3. When the user changes the P High or P Low values, update the saturated properties displayed in the labels to reflect the new isobar properties. Also, update T High if appropriate. (see <https://doc.qt.io/qt-5/qlineedit.html#editingFinished> )
4. Add a pair of radio buttons that allows the user to select to work in SI or English units. Update all labels and values (including the plot) to reflect the selected units whenever the user clicks one of the radio buttons. (Hint: only one radio button in a container can be checked at a time, so you will likely need to place your radio buttons in a group box to isolate them from the existing Quality and T High radio buttons.) The only function in the slot connected to the clicked signal of these radio buttons should be:

self.RC.updateUnits(self.widgets, self.otherwidgets, SI=??)

That is, all your modifications to the view (GUI) should be done by the controller.