



Indian Institute of Technology Bombay

Summer Fellowship Report

ON

GUI for OpenModelica Chemical Process Simulator (Based on PyQt5)

Author:

Anosh Billimoria,
(Bennett University)
Anuj Goyal,
(Chandigarh College of Engg. & Tech.)

Supervisor:

Prof. Kannan M. Moudgalya
Chemical Engineering Dpt.
IIT Bombay

July 12, 2019

Acknowledgement

We would like to take this opportunity to express our greatest gratitude to our mentor, Mr. Pravin Kumar Dalve for guiding, supporting and helping us in every possible way. We were extremely fortunate to have him as our mentor as he provided insightful solutions to problems faced by us thus contributing immensely towards the completion of this project. We would like to thank FOSSEE Team and IIT Bombay for giving us this opportunity and providing a platform to exhibit our skills. We would also like to express our deepest gratitude to the faculty of our respective universities for giving us an opportunity to be a part of this fellowship.

Contents

1	Introduction	3
1.1	Vision	3
1.2	Approach	3
2	Technology Stack	4
2.1	Python	4
2.2	PyQt	4
2.3	OpenModelica	4
2.4	Git:	5
3	Implementation	5
4	Features	7
4.1	Compound Selector	7
4.2	Component Selector	7
4.3	Mode Selection	8
4.4	Message Browser	8
4.5	Canvas	9
4.6	Result	9
4.7	Miscellaneous	10
5	Simulation	10
5.1	Equation oriented mode	10
5.2	Sequential Oriented mode	11

1 Introduction

OPENMODELICA is an open-source Modelica-based modeling and simulation environment intended for industrial and academic usage. Its long-term development is supported by a non-profit organization – the Open Source Modelica Consortium (OSMC).

Our GUI for OM chemical simulator is responsible for implementing the usage of OMChemSim library created by FOSSEE team.

1.1 Vision

The existing process of creating chemical simulations requires the knowledge of, how to use the openmodelica and learn the syntax of correctly writing code that implements the library and creates flowsheet model.

Once the model is created openmodelica provides only solution in Equation Oriented Mode only. This creates a problem in getting the final solution to converge in most cases.

Chemical simulation of dynamic nature requires a Sequential Oriented Mode, wherein each unit operation must be simulated in isolation with respect to the other.

After successful simulation openmodelica provides a tree structure of viewing the result variables with not a lot of explanation of what they might be or belong to. So, having component wise result selection screen was our goal, with well-defined titles for each result variable

We aim to eliminate the hassle of learning a coding paradigm for chemical engineers and layman enthusiasts who may want to simulate any process.

1.2 Approach

We have used the python language to create and system involving an object-oriented approach to each of the components, their connections and defining properties of each of them.

Our primary graphical view and UI was built using PyQt5 which is a python wrapper library over the C/C++ Qt framework.

The GUI empowers user to select compounds from a wide database by searching its name and corresponding properties. Further select the specific components required for the flowsheet model and connections by a simple click and drag.

Parameters and modes of these components can be edited to users will. Adding and removing of components is fluently implemented.

We have separate triggers for Sequential oriented and Equation oriented mode of solving. The visual order appearing on screen is implemented as the standard order for Sequential oriented simulation.

We have managed to separate the result variables for each component and display them to the user.

A messages window reflects every action made by the user as a feedback for successful action.

2 Technology Stack

The technologies that we used for the development of the GUI Software. Following are the technology stack we have use:

2.1 Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

2.2 PyQt

PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. PyQt is free software developed by the British firm Riverbank Computing.

We used Latest version of PyQt i.e PyQt5. PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android.

2.3 OpenModelica

OpenModelica is a free and open source environment based on the Modelica modeling language for modeling, simulating, optimizing and analyzing complex dynamic systems. This software is actively developed by Open Source Modelica Consortium, a non-profit, non-governmental organization.

We used the om compiler for doing calculations and generating desired output. This is also a pre-requisite to our GUI application.

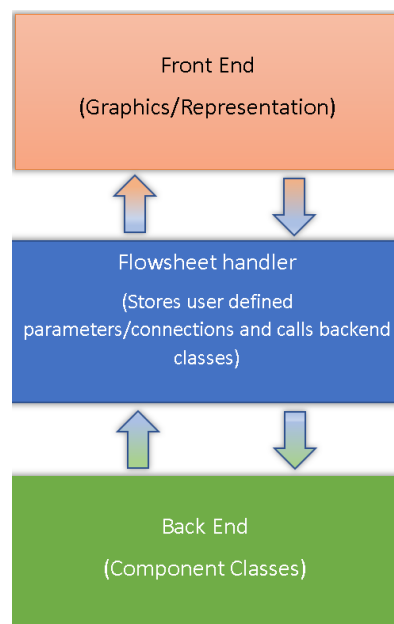
2.4 Git:

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

We Used Github for the Git Version Control System.

3 Implementation

We can model our application in the following manner.



Front End: Using the power of Graphic widgets from PyQt5 we have created separate handlers for lines, component icons, and sockets.

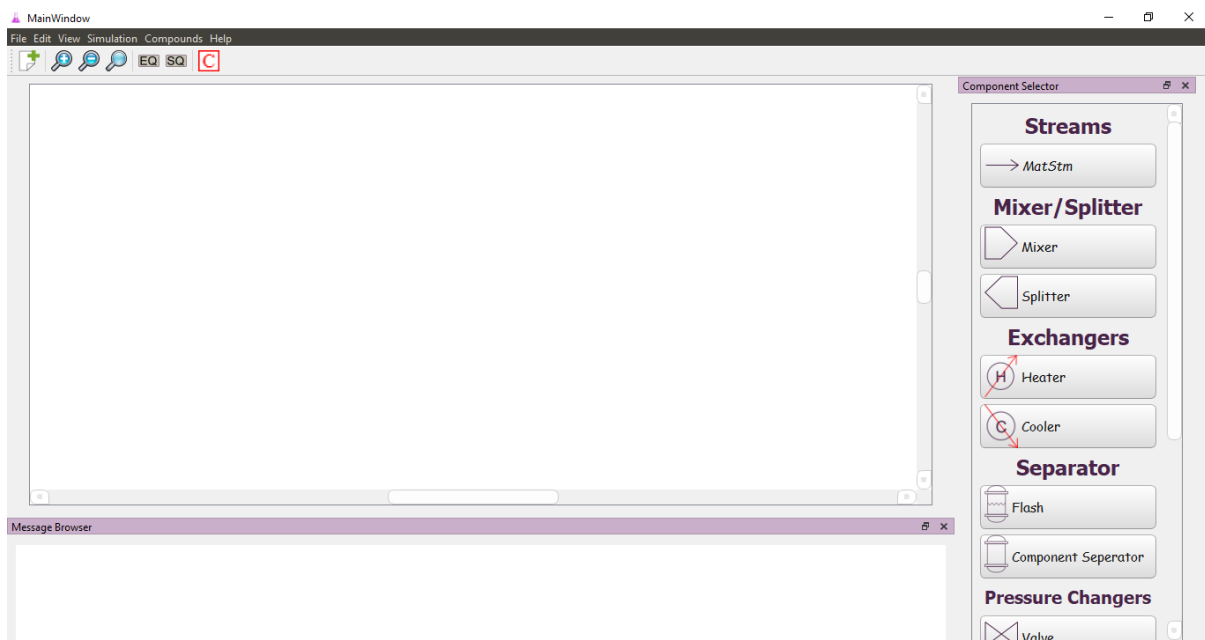
Any interaction of the icons with the user is an input taken seriously and must be collected.

Interactions may include:

- Add new component (Material Streams or Unit Operations)
- Removing a component
- Editing parameters according to mode selected
- Making the connections between each component
- Compound selection

Flowsheet Handler: A single class connecting the user inputs and necessary backend functionality. This handler brings the user input to of each component representation to its specific object instances. For sending it over to the writing of the (.mo) script file which will ultimately run on the Openmodelica compiler installed on the system.

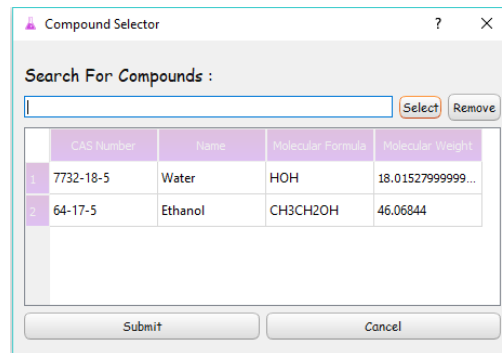
Back End: This involves all the classes whose object instances have been feed with user input and flowsheet design details. Every component is responsible for writing its definition in these files and the corresponding equations demanded by the Openmodelica syntax.



4 Features

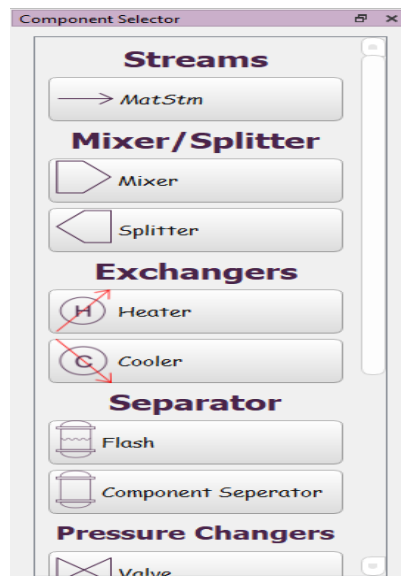
Following are the main features of The GUI application:

4.1 Compound Selector



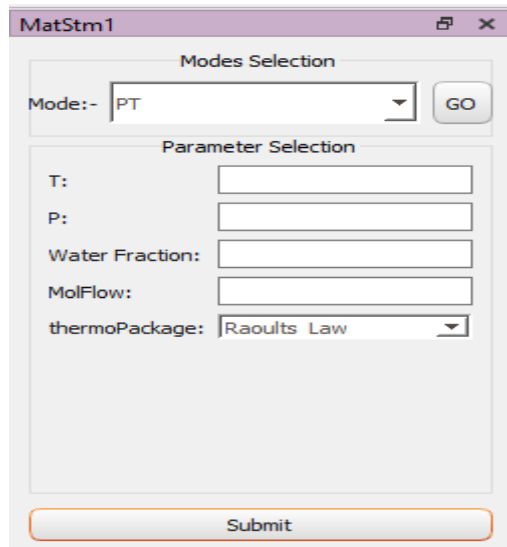
Compound Selector is a dialog Box in which user can select the required compounds. Compounds can be searched in autocomplete search box. After selecting the compounds user can view all the details of selected compounds like the molecular formulae, molecular weight etc. Also, user can remove the required compounds in the case not needed.

4.2 Component Selector



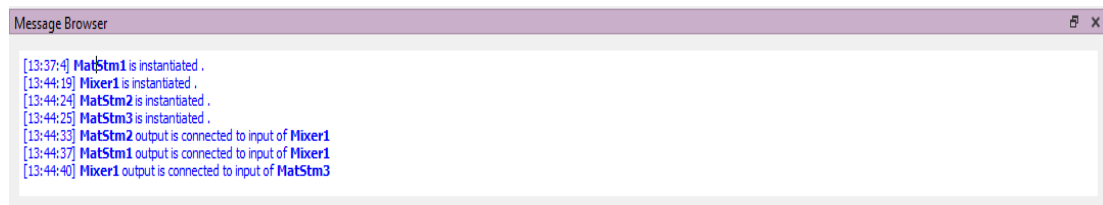
Component Selector is a dockWidget with a scrollable area. In this user can click on the button of the required component like Mixer ,Matstm ,Splitter etc. The respective component will be instantiated and shown on the canvas as soon as user selects the component. Component selector is the one which handles all the thing of respective components.

4.3 Mode Selection



The screenshot shows a window titled "MatStm1" with two main sections: "Modes Selection" and "Parameter Selection". In the "Modes Selection" section, there is a dropdown menu labeled "Mode:-" with "PT" selected, and a "GO" button to its right. The "Parameter Selection" section contains five input fields: "T:", "P:", "Water Fraction:", "MolFlow:", and "thermoPackage:". The "thermoPackage:" field has a dropdown menu with "Raoult's Law" selected. At the bottom of the window is a large "Submit" button.

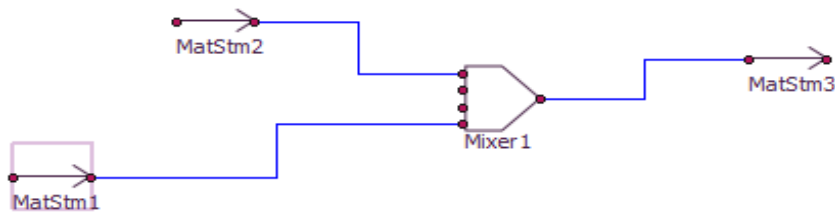
Some of the Unit Operations have different modes for the input parameters. So, we have set default mostly used modes for components but the user can change the modes according to needs. User can change the mode at the same place of where he/she has to input the parameters. So, that its convenient for user to change the modes.



4.4 Message Browser

Message Browser is another dockwidget window placed at the bottom. The main function of this message browser to give the feedback to the user what is done. It gives the feedback to the user with time of operations. It prints out the operations that the user has done like instantiation of the new components or connection of one component to another etc. It also gives the time taken in simulation and other things like that.

4.5 Canvas



Canvas is the area where user can draw the required flowsheet. User can connect multiple components with each other by connecting the nodes on the different components with a line. The canvas can be zoomed in or Out using buttons on the toolbar. This is the area where the flowsheet designer will spend most of his time.

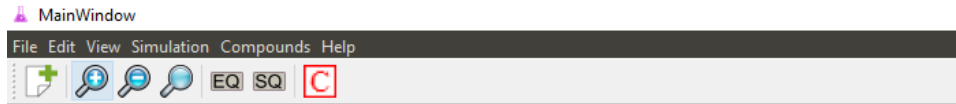
4.6 Result

The screenshot shows a window titled 'Results' with a dropdown menu set to 'MatStm1' and a 'Submit' button. Below the dropdown is a table with 18 rows of data. The table has two columns: 'Property' and 'Value'.

	Property	Value
1	Pressure	101325
2	Temperature	310
3	Liquid Phase Mol Fraction	1
4	Liquid Phase Mass Fraction	1
5	Vapour Phase Mol Fraction	0
6	Vapour Phase Mass Fraction	0
7	Molar Flow	100
8	Mass Flow	1801.5
9	Mixer Phase Molecular Weight	18.015
10	Liquid Phase Molecular Weight	18.015
11	Vapour Phase Molecular Weight	0
12	Mixer Phase molar Heat Capacity	75.5482659514592
13	Mixer Phase Molar Enthalpy	-43090.212500850
14	Liquid Phase molar Heat Capacity	75.5482659514592
15	Liquid Phase Molar Enthalpy	-43090.212500850
16	Liquid Phase Molar Entropy	-138.97545674452
17	Vapour Phase molar Heat Capacity	0
18	Vapour Phase Molar Enthalpy	0

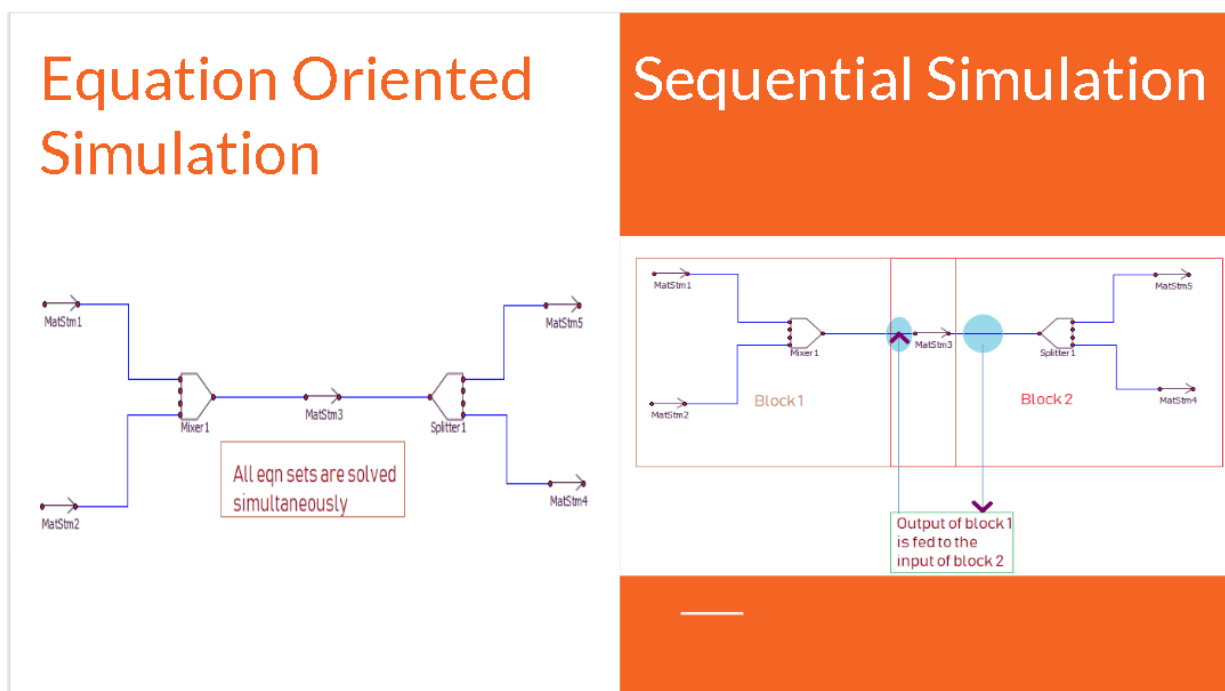
Results are displayed in the Dockwidget which is viewed on the left side. Results widget will have a dropdown where user can select from the list of components, he has used and select for which the component he needs the result. User will be able to view the result that Dockwidget. We used this approach as for reference user can easily view the design he has made on the flowsheet.

4.7 Miscellaneous



- Menu Bar
- Tool Bar

5 Simulation



5.1 Equation oriented mode

Once all user input for each component is received and every connection is duly made user will click on the simulation button for equation oriented solution. In doing so a flowsheet with the extension of (.mo) is created within the directory structure. At the same time a file with extension (.mos) is also created which holds all necessary commands for the Open modelica compiler installed on the system. We instruct the compiler to load the following dependencies in order. Modelica, package.mo and flowsheet.mo then a command for simulate.

OM provides the output in various formats such as (.mat),(.plt) and (.csv). We continue our program flow with the (.csv) file and print results onto the screen with clear formatting of each variable and it's full name.

5.2 Sequential Oriented mode

Once the simulation of Equation oriented is completed. The user can choose to re-simulate the same flowsheet design in sequential oriented mode. Although any can be simulated first.

Under this process, python selects only the first unit operation from the flowsheet and creates an openmodelica script with extension (.mo) and simulates this similar to Equation oriented approach.

The output of this flowsheet is saved in the connecting material stream of the flowsheet design, as the program progresses to the next unit operation in the pipeline it makes sure that the order is maintained and instantiates the starting values from prior simulation only. After successful simulation of all the unit operations, the results are shown on screen formatted according to the components used.