Thermodynamics Project 2 User Manual

Overview

There are several files contained in this repository, many of which are helper/utility functions. To obtain the outputs for project 2, one needs only to run the *project2_main.py* file. Each part of each question is a separate function that can be run independently — if one is interested in testing a particular question, one can simply comment out the other function calls.

Input Files:

I use YAML format for input files and are read in as a Python dictionary. PyYAML is a dependency for the code to run. This project has 3 separate input files — one for problems 2a & 2b, one for problem 2c, and one for problem 3; they can be found in the Input_Files/ directory. The first two input files are used for problem 2d. Setting Pvap = None will automatically initialize vapor pressure to the vapor pressure given by Wilson's correlation. You should be able to simply change the P, T, and z in the input file and rerun HW8_main.py. I also specify the desired critical temperature, critical pressure, and acentric factor for each fluid as a list (state variables grouped together). To change the components, simply change the Tc, Pc, and w in the appropriate fields. The order of the list matters! Each element of the list corresponds to a particular component.

The binary interaction parameters (BIP) are read in as nested lists where each list corresponds to a row of the BIP matrix (see input_file_3.yml) as an example. The input function automatically generates the Kij matrix.

Output Files:

The code automatically creates an output directory if one does not already exist. For this assignment, the parent output directory will be called Project2_Output/. There will be 2 subdirectories, Problem 2/ and Problem 3/.

Within the Project2_Output/Problem 2/ directory will be 5 files:

A .txt file with stdout outputs.

A .png file titled Problem2A.png with the P-x diagram assuming BIP = 0

A .png file titled Problem2B.png with the P-x diagram assuming BIP = 0 and experimental data from Joyce et al. (2000).

A .png file titled Problem2C.png with the P-x diagram assuming a non-zero BIP and experimental data from Joyce et al. (2000).

A .png file titled Problem2D.png with the Gibbs free energy plotted for BIP=0 and non-zero BIP.

Within the Project2_Output/Problem 3/ directory will be 4 file:

A .txt file with stdout outputs.

A .png file titled Problem3A.png with a ternary diagram indicating the two-phase region, 5 tie-lines, and the estimated critical point.

A .png file titled Problem3B_1.png with a ternary diagram with the extended pseudo-component line along which the subsequent Gibbs free energy analysis was performed.

A .png file titled Problem3B_2.png with plots of the Gibbs free energy and gradient of Gibbs free energy vs. mixing ratio. The equilibrium phase positions are also marked.

Brief Description of Module Files

eos.py

Equation of State and EoS utility functions.

io_utils.py

Functions related to input/output.

solve.py

Cardano's method function and single phase flash calculations are found in solve.py.

pr_utils.py

Departure functions and fugacity calculations for this assignment are in the pr_utils.py file.

singlecomponent_utils.py

Functions for single component flash and vapor pressure calculations

multicomponent_utils.py

Utilities for calculating bubble point, dew point, phase compositions, attraction and covolume parameters, phase fugacity, root selection (via Gibbs Free Energy), and TBD for stability analysis.

multicomponent_solve.py

General-use function for performing Newton-Raphson iteration for root finding, a lambda function for the Rachford-Rice equation, and a function for finding the root of the Rachford-Rice equation. New additions include two-phase flash, single-phase stability analysis, and a combination function that performs the two-phase flash contingent on an unstable single phase.

unit_conversions.py

A class to help with unit conversions.

ternary_plot_utils.py

Functions to initialize a ternary plot and convert compositions to the coordinate convention used in the ternary-python library. Additional function to analytically find the axis intercepts on a ternary diagram.