ENGR 19500H Spring 2013

Homework 4 – C, Python, and MATLAB Programming

You are a chemical engineering co-op student working at the NASA Goddard Space Flight Center in the Earth Observing System (EOS) Program Office. You have been assigned to work with the SOlar Radiation and Climate Experiment (SORCE) mission team. It's your first day on the job and you go to you new boss's office to introduce yourself. After she makes a brief "welcome aboard" statement, she invites you to a meeting with the planning team of engineers looking into the next generation of instrumentation they will be putting on a future satellite.

Upon your arrival, you are surprised to see a whole team of engineers seated at the conference table in your boss's office. She introduces you and a discussion ensues. After lengthy discourses by several engineers, dealing with things you have never heard about, you boss turns to you and says "Let's have our new intern work on designing two of the gas pressure vessels that will be used aboard the satellite. I will assist you to get things started."

After the meeting is over and everyone has left the office, she begins explaining that oxygen and carbon dioxide are needed to operate various instruments aboard the satellite. She tells you that in order to design the pressure vessels, you will need to estimate the molal volume (*v*) of carbon dioxide and oxygen for a number of different temperature and pressure combinations. You think for a moment and then blurt out that this is simply an application of the *ideal gas law*. On a piece of paper you write

$$pV = nRT \tag{1}$$

where *p* is the absolute pressure, *V* is the volume, *n* is the number of moles, *R* is the universal gas constant, and *T* is the absolute temperature. She says, "While the *ideal gas law* is widely used, it is only accurate over a limited range of pressures and temperatures. In addition, it is more appropriate for some gases than for others." She goes on to explain that an alternative equation of state for gases is given by the *van der Walls equation* and writes

$$\left(p + \frac{a}{v^2}\right)(v - b) = RT \tag{2}$$

where $v = V/n$ is the molal volume and *a* and *b* are empirical constants that depend on the particular gas. She explains that, for all of the various gasses to be stored on the satellite, the significant changes in temperature and pressure that are expected during operation of the satellite could produce wide fluctuations in the molal volume of the gas, which in turn could impact the performance of the instruments that depend on that gas for proper operation. **She explains one of the first things the team will need to know is how *v* will change, for a given gas, over a specified pressure range at a constant temperature. Therefore, she recommends you write a program to compute *v* for a given gas and illustrate how it changes with pressure at a constant temperature. Some of the current operating specifications for the vessel include:**

Temperature: -40-60 C
Pressure: 6 psi to 1500 psi

She then proceeds to rewrite eq. (2) such that

$$f(v) = \left(p + \frac{a}{v^2}\right)(v - b) - RT = 0 \qquad (3)$$

She says if you find a value of *v* that satisfies eq. (3), the problem is solved. You also reason that it has to be a positive, real number or else the answer does not make much sense. Since this is nothing more than finding the root of an equation, you look up all the different methods that are available to do this and decide to use a method called the *Newton-Raphson method*. An excerpt from a book you found in the library describes the *Newton-Raphson method* as follows:

> Perhaps the most widely used of all root-locating formulas is the Newton-Raphson equation. If the initial guess of the root is x_i, a tangent can be extended from the point [x_i, $f(x_i)$]. The point where this tangent crosses the *x*-axis usually represents an improved estimate of the root.
>
> The Newton-Raphson method can be derived on the basis of this geometrical interpretation. As shown in the fig. 1, the first derivative at x_i is equivalent to the slope:
>
> $$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$
>
> which can be rearranged to yield
>
> $$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$
>
> and is commonly called the *Newton-Raphson formula*. This equation can be used in an iterative procedure until a converged solution (a root) is obtained. Convergences is determined by comparing a previous estimate to the root to the current estimate of the root, such that
>
> $$|\varepsilon| = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \leq tolerance$$
>
> where the tolerance is some small number.

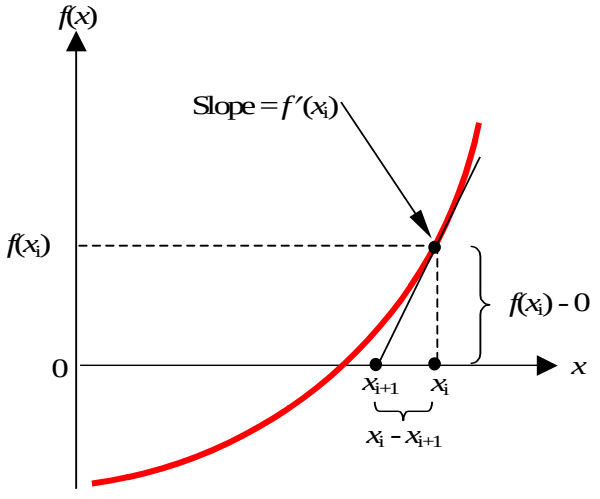

Fig. 1 Graphical depiction of the Newton-Raphson method.

For the particular problem at hand, you reason that application of the Newton-Raphson formula will yield something like the following

$$for\ i = 1: \quad v_2 = v_1 - \frac{f(v_1)}{f'(v_1)} \quad \rightarrow \quad |\varepsilon| = \left|\frac{v_2 - v_1}{v_2}\right| > tolerance$$

$$for\ i = 2: \quad v_3 = v_2 - \frac{f(v_2)}{f'(v_2)} \quad \rightarrow \quad |\varepsilon| = \left|\frac{v_3 - v_2}{v_3}\right| > tolerance$$

$$for\ i = 3: \quad v_4 = v_3 - \frac{f(v_3)}{f'(v_3)} \quad \rightarrow \quad |\varepsilon| = \left|\frac{v_4 - v_3}{v_4}\right| > tolerance$$

$$\vdots$$

$$for\ i = n-1: \quad v_n = v_{n-1} - \frac{f(v_{n-1})}{f'(v_{n-1})} \quad \rightarrow \quad |\varepsilon| = \left|\frac{v_n - v_{n-1}}{v_n}\right| \leq tolerance$$

Now that you have the *Newton-Raphson method* figured out, you go to another reference book and find the empirical constants *a* and *b* (for both carbon dioxide and oxygen), as well as the universal gas constant. The values you found are:

$$R = 0.082054 \frac{L \cdot atm}{mol \cdot K}$$

Carbon Dioxide
a = 3.59200
b = 0.04267

Oxygen
a = 1.36000
b = 0.03183

You are now ready to design your program. You first decide the input for your program should be:

Table 1. Input Data
Temperature (K)
initial pressure (atm)
final pressure (atm)
number of pressure increments
gas type
a (for a gas, if **not** carbon dioxide or oxygen)
b (for a gas, if **not** carbon dioxide or oxygen)
maximum number of iterations
convergence criteria
Output file name

In addition, on a piece of engineering paper you sketch out what the output will look like. For example:

Table 2. Sample Output Data

initial pressure (atm) = xxxxx
final pressure (atm) = xxxxx
number of pressure increments = xxxxx

a = xxxxx
b = xxxxx

maximum number of iterations = xxxxx
convergence criteria = xxxxx

Temperature (K)	Pressure (atm)	gas type Molal Volume, L/mol (vdW)	Number Iterations	gas type Molal Volume, L/mol (IGL)
xxxxx	xxxx	xxxxx	xxxxx	xxxx
xxxxx	xxxx	xxxxx	xxxxx	xxxx
xxxxx	xxxx	xxxxx	xxxxx	xxxx

***Note: the xxxxx's are not intended to denote the number of digits to be displayed; they are simply there to indicate that a number should be printed*.**

Finally, you decide to use as a test case, a temperature of 300.0 K and a pressure range of 1.0 to 100.0 atm.

SPECIFIC REQUIREMENTS:

Part A

1. **INDIVIDUALLY**, develop an appropriate algorithm and sub-algorithms (one to input the data, another to compute the function $f(v_i)$, another to compute the derivative $f'(v_i)$, and another to describe the Newton-Raphson method) in the form of a problem statement and associated flow diagram (**please remember to use the posted Programming Problem Solving Presentation Method, see posted example, also make sure you flow diagram is programming language independent**), that will estimate the molal volume (v) of carbon dioxide and oxygen for a number of different temperature and pressure combinations based on the input data described in Table 1 using the *van der Walls equation* (eq. 2). The algorithm should denote that each input data be output to ensure it is correct and upon entry of a single character (Y, y, N, or n), the flow of the algorithm be appropriately altered. In addition, if the gas type is not carbon dioxide or oxygen, the algorithm should indicate the constants a and b are also to be input. Lastly, from an input perspective, the algorithm should allow for input of the maximum number of allowable iterations and specification of a convergence criteria (see above), but **there should not be** any check of these data. All other sub-algorithms that will allow for the calculation of the molal volume should be included.
2. **Part A of this homework assignment is due on Monday, April 15, 2013 at the start of class.**

Part B

1. **AS A TEAM** decide who will convert his/her algorithmic solution into a C program, MATLAB script, or Python script. For teams of four, two individuals must select to do this assignment in C, one in Python and the other in MATLAB. For teams of three, each one of the programming languages must be chosen by one team member. **Note that there are no restrictions on the choice of programming language used based on prior homework assignments – each team member may choose any of**

the languages indicated so long as the distribution of languages meets the previously stated requirements.

2. **INDIVIDUALLY** create a program/script that represents the algorithmic solution to problem described above.
3. The program should input all data from a keyboard prompt as described below:
    a. The order of the data input should be exactly as described above in Table 1.
    b. For each input request the user should be prompted with a specific entry request (e.g. *Enter the temperature*). After each data entry is entered, it should be displayed on the screen and the user asked to verify that the entry is correct before being prompted to enter the next the value. Should the entry be incorrect, the user should be re-directed to enter the value again. The acceptable responses to the verification question will be Y, y, N, or n. Any other input should cause the program to tell the user what the appropriate answer format is and then re-ask the verification question.
        i. For the C and Python programs, input will be done via UNIX redirection. (This will be explained in the Homework 2 Supplement Document, but it is basically re-directing information that is typically typed in by the user at the keyboard to instead come from a file). For the MATLAB program, assume that all inputs will be read from a file and use appropriate file I/O commands to read in the data. All input/output will be of the list-directed type.
    c. All input, except for the gas type, should be declared to be of type ***double***. The gas type should be input as a single character (i.e. "c" for carbon dioxide, "o" for oxygen)
    d. If a gas type other than carbon or oxygen is requested, then the program should prompt for values for the constants *a* and *b*.
    e. When entering the output data file name, the user should be able to enter both a file path as well as a file name (i.e., you should reserve space for a file name containing up to and including 100 characters). As described in 2b), the file name should be displayed to the screen and the user given an opportunity to verify his/her entry. If the file already exists in the specified directory, the program should re-prompt the user for a new file name and start the verification process over. If after five (5) attempts to create an output file, the program is unable to create a "new" file your program should notify the user and exit gracefully.
4. You can assume that the maximum number of pressure increments will not exceed 200.
5. ALL input must be done in a function that is designed specifically to input all data for this program. Program output (i.e., results) should be done in the main program. The program should create two forms of output, one to the output file described about and the other to the printer. **The output table of results should not be created until all the values have been computed**. Once all the values have been computed, the table of results should appear as shown above (both in the output file as well as for the printer). The table should be able to be read/interpreted by anyone looking at the output, as in Table 2.
6. Separate functions (of type double) should be used to:
    1) input the data
    2) compute the function $f(v_i)$
    2) compute the derivative $f'(v_i)$
    3) perform the Newton-Raphson method (including the convergence check).

    No variables should be hard coded in the functions (any required data should be passed through a parameter list).
7. If the Newton-Raphson method fails to converge after the maximum number of iterations, a value of 0.0 should be returned as the root along with the maximum number of iterations performed.

8. Use BlackBoard to submit your script using the name `HW04_login.c`, `HW04_login.m`, or `HW04_login.py` as appropriate.

Part B to be submitted electronically via Blackboard by Friday April 19, 2013 at 11:59 pm.