AspenTech: VAPOR PRESSURES

Technical Memo No. XX-1

Subject: Vapor pressure interpolation error analysis
To: *Ms. Sam K. Safobeen*
Submitted by: *Student 2*
Date: May 11, 2010

Summary: I have compiled vapor pressure functions for the following four species: methane (CH_4), ethane (C_2H_6), ethylene (C_2H_4), and propane (C_3H_8). The functions accept temperature inputs in Kelvin and return the vapor pressure of the species in atm. The error in each approximation was calculated and presented in tabular form.

Findings: I am writing because I have successfully fit the provided vapor pressure data to separate MATLAB functions. Each function accepts a temperature input, in Kelvin, and will output the corresponding vapor pressure in atm. I have enclosed a sample MATLAB function file (Psat1.m) for methane.

The approach used for all four functions was a variation of **Newton's forward divided-difference form of the interpolatory polynomial**. The algorithm generates a 4th-order polynomial using the data points given from the experimental data closest to the temperature input. This method is preferred over a more comprehensive polynomial because of how the polynomial generated. A large order polynomial is overly concerned with fitting the given data points exactly than observing a general trend in the data. Fitting only a handful of points to a polynomial stresses the importance of the points around the input temperature rather than placing equal weight on all of the available points.

The estimated error for this approach is computed by generating the 5th-order polynomial at the same temperature and taking the difference between the 4th-order and 5th-order approximations. Since the error varies between data points, I have listed the error bounds for each function below:

Species	MATLAB Function	Valid Temp (K)	Maximum Error (atm)
methane	**Psat1.m**	110 - 190	± 0.1
ethane	**Psat2.m**	110 - 270	± 0.01
ethylene	**Psat3.m**	110 - 270	± 0.009
propane	**Psat4.m**	110 - 270	± 0.002

The maximum error is computed by comparing the estimated error to the error inherent in the experimental data; whichever contributes the most significant source of error for each function is the maximum possible error. The error beyond the temperature ranges cannot be bound in such a manner. The data obtained from my vapor pressure functions, therefore, will be accurate up to the respective maximum error listed within the range of valid temperatures.

Attached: Psat1.m

Listing 1: **Psat1.m** - Vapor Pressure as a function of Temperature for Methane

```
function [Psat] = Psat1(T)
%PSAT1 Obtain the partial pressure of methane (CH4) as a function of
%temperature in Kelvin.
%   Psat = Psat1(T) From a given set of divided difference coefficients,
%   the appropriate Newton forward divided-difference form of interpolatory
%   polynomial is evaluated at the specified T and the appropriate partial
%   pressure is returned.
%
%   The pressure is given in atm.  (1 atm = 1.01325 bar)
%
%   Student 2
%   04/21/2010

T_given = 110:10:190; % Temperature, K
Psat_given = [0.884 ...
              1.919 ...
              3.681 ...
              6.422 ...
              10.41 ...
              15.94 ...
              23.81 ...
              32.86 ...
              45.20]; % Pressure, bar

[T_pivot,index_pivot] = min(abs(T_given - T)); % find the closest T within
    T_given
if (index_pivot == 1)
    index_pivot = 2;
elseif (index_pivot >= 8)
    index_pivot = 7;
end

T_sample = [T_given(index_pivot - 1) T_given(index_pivot) T_given(index_pivot
    +1) T_given(index_pivot+2)];
Psat_sample = [Psat_given(index_pivot - 1) Psat_given(index_pivot) Psat_given(
    index_pivot+1) Psat_given(index_pivot+2)];

dim_T = size(T_sample);
dim_P = size(Psat_sample);

N = dim_T(2); % could easily be dim_P(2), both should be the same value

P_temp = [Psat_sample' zeros(N,N-1)]; % initialize P_temp, first column values
     are from given initial values
                           % It holds all the relevant divided differences
                           % calculations.
coeff = zeros(1,N); % initialize the relevant coeffs used in the polynomial
coeff(1) = Psat_sample(1); % First coeff is not even touched by the algorithm
    and must be set manually

for i = 2:N
    for j = 2:i
        P_temp(i,j) = (P_temp(i,j-1) - P_temp(i-1,j-1))/(T_sample(i)-T_sample(
            i-j+1));
        if (i == j)
            coeff(i) = P_temp(i,j); % the needed coeffs only occur at the
                diagonals of the triangular matrix,
```

```
                                % which is when i = j
            end
        end
    end

    Psat = 0; % initialize Psat

    for i = length(coeff):-1:1
        Psat = Psat*(T - T_sample(i)) + coeff(i); % very concise'd version which
            deals with all of the polynomial terms of divided differences through
            backwards nested loop
    end

    Psat = Psat / 1.01325; % conversion from bar to atm
```

AspenTech: Flash Calculator

EPCflash.m - User Manual

EPCflash.m is a powerful flash drum calculator in MATLAB. This user manual will walk you through an example flash calculation to get you up and running as soon as possible for your flash calculation needs.

Before we begin, please make sure the following files are in your current directory:

- **Psat1.m** - This function calculates the vapor pressure for methane (CH_4)
- **Psat2.m** - Same as above, for ethane (C_2H_6)
- **Psat3.m** - Same as above, for ethylene (C_2H_4)
- **Psat4.m** - Same as above, for propane (C_3H_8)
- **flash1.m** - Performs a flash calculation given (z,F,T,P)
- **flash2.m** - Performs a flash calculation given (z,F,T,V)
- **flash3.m** - Performs a flash calculation given (z,F,T,y1)
- **EPCflash.m** - Integrates all of the above, the primary function to use. Input (z,F,T,[P V y1]).

To get started, enter the following in the command window:

```
>> z = [0.2 0.4 0.3 0.1];
```

z is a vector of mole fractions corresponding to the input feed stream. It is important that **z** is always a vector of length **four**, must always contain values between 0 and 1, and must always add up to 1. In this example, the mole fraction of our input feed is 0.2 methane, 0.4 ethane, 0.3 ethylene, and 0.1 propane.

```
>> F = 200;
>> T = 200;
>> P = 3;
```

Here we have set the input feed stream (**F**) to be 200; this stream is in units of mol per unit time. The operating temperature, **T**, is set to 200 K, and the pressure, **P**, is set to 3 atm. We are now ready to perform our first flash calculation. This is done by entering the following in the command window:

```
>> [x y L V P] = EPCflash(z,F,T,[P 0 0]);
```

The last parameter, [P V y1], indicates which method will be called to perform the flash calculation: if P is provided, then *flash1.m* will be invoked, if V is provided, then *flash2.m* will be used, and if y1 is provided, then *flash3.m* will be called. Only one of the three possible variables should be used when using this program; the other two should be marked as 0. The flash calculation should be run correctly and no error messages should be displayed. After this, you should have these following variables with the corresponding values:

```
x =
    0.0141    0.4943    0.2249    0.2666
y =
    0.2922    0.3532    0.3373    0.0173
L =
   66.3212
V =
  133.6788
P =
     3
```

The way *flash1.m* works is by substituting all of the known data into each vapor mole fraction of **y** and using a bisection method to solve for **L**, the liquid molar flow rate that satisfies the given conditions. The extensive mathematical details will not be discussed here. **x** is the liquid mole fraction of the feed, and **V** is the vapor molar flow rate of the exit stream. As we can see here, of the 200 mol per unit time initial feed, 66.3212 exits as a liquid under 200 K and 3 atm, which has a composition of 1.41% methane, 49.43% ethane, 22.49% ethylene, and 26.66% propane.

Let us perform another flash calculation using a different value for the last parameter. Enter **clear** in the command window and input the same values for **z, F,** and **T** from before. Let us find the conditions for an equimolar flow of both liquid and vapor output streams at 200 K by entering the following:

```
>> V = 100;
>> [x y L V P] = EPCflash(z,F,T,[0 V 0]);
x =
    0.0228    0.5117    0.2749    0.1903
y =
    0.3772    0.2883    0.3251    0.0097
L =
   100
V =
   100
P =
    3.8048
```

This method uses *flash2.m* to run the flash calculation, which does so by obtaining the pressure,

P, in a similar manner as *flash1.m* obtains **L**, and then uses Runge-Kutta order 4 for systems of initial value problems. This is made possible through the method of false transients, in which the vapor fraction **y** is assumed to vary with time, and the solution to this problem would be when **y** reaches steady state. From this example we can see that at 200 K and 3.8048 atm, we will obtain an equimolar liquid and vapor flow.

Suppose we want to set a fixed vapor mole fraction for methane. We will use the same **z** and **F**, but let us set $\mathbf{T = 150}$ this time just so we are not performing flash calculations at the same temperature every time. Input a value for **y1** into the last input parameter as follows:

```
>> y1 = 0.4;
>> [x y L V P] = EPCflash(z,F,T,[0 0 y1]);
x =
    0.0079    0.5406    0.2580    0.1936
y =
    0.4000    0.2536    0.3438    0.0026
L =
  102.0200
V =
   97.9800
P =
    0.2034
```

This method uses *flash3.m* to calculate the remaining values, which is done by using Newton's method for solving initial value problems. At every iteration, the pressure is recomputed until everything converges. As you can see, in order to obtain 40% methane in the vapor exit stream at 150 K, it will be necessary to lower the pressure to 0.2034 atm.

Of the possible final inputs, vapor molar flow rate is easily bounded by the input feed (since the output rate cannot exceed the input rate) and **y1** must lie between 0 and 1. Pressure is difficult to bound and arbitrary selections of temperature and pressure will easily lead to poor *flash1.m* output values. The following table presents general guidelines for valid temperature and pressure ranges:

Temperature (K)	Pressures (atm)
120	0.01 - 0.03
140	0.04 - 0.06
160	0.13 - 0.64
180	0.4 - 3
200	1.4 - 9
220	3.6 - 22
240	8.8 - 42
260	17 - 86
280	28 - 140
300	45 - 210

This data was compiled through trial and error using an input stream composition of 25% methane, 25% ethane, 25% ethylene, and 25% propane. The pressure ranges will also vary depending on the input stream composition; this table only serves as a guideline for what will actually be computable by the flash programs. Even so, at most of the extreme conditions, although values will be properly generated, most species will be below their boiling points or above their critical temperature, and thus the output will not be accurate under these circumstances. Essentially, if the operating pressure lies well within the guidelines provided, the more likely that the flash calculations will approximate realistic scenarios.

Error Messages

When an error occurs, almost all return values will be -1 to denote that no flash calculation took place.

z related errors (From *flash1.m*,*flash2.m*, and *flash3.m*):

```
Error: Incorrect number of mole fractions — please provide the mole fractions in
    the following format:
>> z = [ 0.2 0.4 0.3 0.1 ];
     where 0.2 corresponds to mole fraction of inlet methane,
     0.4 corresponds to inlet ethane,
     0.3 corresponds to inlet ethylene,
     and 0.1 corresponds to inlet propane.

Error: Mole fractions must be within the range of 0 and 1. Please check the mole
    fractions to see if they are valid and try again.

Error: Mole fractions provided do not add up to 1. Please check the mole fractions
    to see if they are valid and try again.
```

These are caused by **z** inputs either not having 4 elements, not consisting of real fractions, and not totaling up to 1.

F related errors (From *flash1.m*,*flash2.m*, and *flash3.m*):

```
Error: Feed molar flow rate must be positive. Please check to see if the molar flow
     rate (F) is greater than 0.
```

This is caused by **F** being negative or equal to 0. **P** related errors (From *flash1.m*):

```
Error: Operating pressure must be positive. Please check to see if the pressure (P)
     is greater than 0.
```

This is caused by an input pressure **P** less than or equal to 0.

V related errors (From *flash2.m*):

```
Error: Vapor flow rate cannot exceed that of the input feed molar flow rate. Please
     make sure V is less than F.

Error: Vapor flow rate cannot be negative. Please make sure V is a positive number
    less than the molar flow rate (F).
```

This is caused by an input vapor molar flow rate **V** greater than F. The vapor flow rate must also be greater than or equal to 0.

y1 related errors (From *flash3.m*):

```
Error: Desired mole fraction of methane must be between 0 and 1. Please check to
    see if this value (y1) is within this range.
```

This is caused by the desired vapor mole fraction of methane **y1** not within [0,1].

var related errors (From *EPCflash.m*):

```
Error: Not enough inputs in var. Please input a vector [P V y1] of which only one
    value is defined whereas the others are 0.

Error: No valid var parameter defined. Please input a vector [P V y1] for in which
    only one out of the three are defined.
```

Either **var** was not input as a vector of correct length or no value of **P, V,** or **y1** was detected to be usable. The format is strict in order for the program to work.

Other messages (From *EPCflash.m*):

```
Warning: x values do not add up to 1; These values are most likely unusable. Please
     revise input conditions for better results.
Warning: y values do not add up to 1; These values are most likely unusable. Please
     revise input conditions for better results.

Warning: Negative x values detected; These values are most likely unusable. Please
    revise input conditions for better results.
Warning: Negative x values detected; These values are most likely unusable. Please
    revise input conditions for better results.
```

These are warnings and not errors because the flash calculations may have taken place and values may be usable to some degree. These warnings will almost always accompany every error message to let the user know that the mole fractions most likely do not make sense and should not be used or interpreted to have any meaning. In certain cases will the **x** and **y** values be usable; although a leniency of their sums adding up to 0.995 to 1.005 is allotted, anything beyond those bounds trigger the first type of warning message.

AspenTech - EPCQcalc

Executive Summary
Heat and Flash Drum Calculator

Problem: When operating a flash drum, there are many parameters which must be taken into account. Given a handful of operating conditions, it is possible to derive and approximate the behavior of such a system. This process is tedious and time consuming, and the derivation is not easily obtained analytically. Furthermore, based on what the given conditions are, the approach to solving the rest of the flash drum conditions will vary as well. The task is not an easy one, and not all operating conditions are guaranteed to be solvable.

Solution: EPCQcalc is a numerical approach to solving flash drum simulations. The function simulates feeding a mixture into a flash drum at a known temperature and pressure and 'flashing' it to a specified temperature and either pressure, volume, or vapor mole fraction. This process requires heat to either be added or released from the system, depending on whether the feed temperature is increased or decreased respectively. A pressure drop will also require heat to be added to the system and heat to be removed for a pressure increase.

EPCQcalc calls upon a sister function, EPCflash, which tackles on the following flash tank simulations:

- **Feed rate composition, molar flow rate, temperature, and pressure given**
  With this information, either the liquid or vapor exit flow rates can be expressed as a single non-linear equation, solvable by any root solving algorithm. The flash program uses bisection method in this case. If the algorithm converges, a liquid molar flow rate is found and thus, the vapor flow rate can be found as well. From here, both vapor and liquid stream compositions can be found.

- **Feed rate composition, molar flow rate, temperature, and vapor molar flow rate given**
  The pressure of this system can be solved for using the same method as the previous case, through a root solving algorithm. Bisection method is once again used for this parameter. With this, a system of non-linear equations can be solved for by treating them as initial value problems. This contrived system of initial value problems can be solved for by treating one of the mole fractions as changing with time and finding when such a change equals 0, also known as the method of false transients. When this system converges, the vapor mole fraction composition is complete and is used to calculate the missing liquid mole fraction.

- **Feed rate composition, molar flow rate, temperature, and desired vapor mole fraction of light key given**
  Using the desired vapor mole fraction of the species with the lowest boiling point, a system of non-linear equations are established. This system is composed of the other unknown vapor mole fractions, and either vapor molar flow rate or liquid molar flow rate; one can be expressed in terms of the other. The flash program uses Newton's method for solving non-linear systems, which involves taking the Jacobian of the set of functions and solving for each unknown until convergence is reached. The liquid composition and the missing molar flow rate are then subsequently calculated.

No built-in MATLAB solver functions are required to use this program; all algorithms are implemented using MATLAB commands. The only potentially 'special' command used is the `inline` function, which creates an inline object out of a string.

Future Changes

The program is fixed to run for a set number of species, currently 4, and the species themselves are also fixed (methane, ethane, ethylene, and propane). The next step to improving this program would be to encorporate a database of vapor pressure values and their corresponding relationship with temperature. This would allow for user specification in selecting which species to be used in the flash operation, as well as allow the number of active species be specified by the user.

Temperatures below the boiling point and above the critical temperature have significant impact on individual species. These substances would be at either pure liquid or pure vapor compositions; such effects are not taken into account in this program. This is because the documented boiling point and critical temperatures only apply at 1 atm–the dew and bubble points for a species changes with pressure as well. Implementing some kind of algorithm which calculates each active species' boiling point and critical temperature at the operating pressures and factoring them into the stream composition would also be the next step towards improving the validity of the program's results.