The Thermodynamic Models in PYroMat

Christopher R. Martin

March 3, 2021

1 Introduction

As of PYroMatversion 2.1.0, there are three general model formulations used to evaluate the thermodynamic properties of gases and multiphase (liquid-vapor) substances. Each pure substance is referred to as a "species." For ideal gases, calculating the properties of mixtures is almost trivial, but when inter-molecular forces rise to relevance, this task becomes difficult enough to require special treatment with a unique mixture model.

1.1 Ideal Gases

In ideal gas models, inter-molecular forces are unimportant, so only the molecule's variable specific heat needs to be resolved in order to construct an entire thermodynamic model. There are two classes in PYroMatfor the two commonly used standards: the IG1 class manages the Shomate equation of state, and IG2 manages the so-called NASA polynomials equation of state. In either case, constant-pressure specific heat, c_p, is constructed purely as a function of temperature. These are addressed in the first two sections below.

2 IG1: The Shomate Equation

PYroMat's IG1 class is built on the Shomate equation for constant-pressure specific heat c_p. This is the formulation used by the NIST/JANAF thermophysical property database, and all other properties are derived from c_p and the ideal gas assumption.

The Shomate equation takes the form

$$t = \frac{T}{1000\text{K}} \tag{1}$$

$$c_p(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + \frac{c_4}{t^2}. \tag{2}$$

It may be obvious that there is little attempt to formulate the Shomate equation around fundamental physics; this is a purely empirical formula. Physics-based expressions for the specific heat of gases are available, but they become so numerically cumbersome, equations as simple as these are far preferable when available.

Because of its simplicity, the Shomate equations lacks the degrees of freedom to express specific heat over wide ranges, so data are usually given in piece-wise formulations. For example, tungsten dioxide (WO_2), has a set of coefficients for $298\text{K} \leq T < 1100\text{K}$ and $1100\text{K} \leq T \leq 6000\text{K}$.

Given an explicit analytical formulation for

$c_p(T)$, it is possible to express enthalpy

$$\begin{aligned} h(T) &= h_0 + \int c_p(T)\mathrm{d}T \\ &= h_0 + (1000K)\int c_p(t)\mathrm{d}t \\ &= h_0 + (1000K)\left(c_0 t + \frac{c_1}{2}t^2 + \dots \right. \\ &\quad \left. \frac{c_2}{3}t^3 + \frac{c_3}{4}t^4 - \frac{c_4}{t}\right) \end{aligned} \tag{3}$$

which is not a function of pressure for an ideal gas.

There are many ways one could select a value for the reference temperature, T_0, and the enthalpic integration constant, h_0, would seem to be arbitrary. That is not so, however, when one is dealing with chemical reactions. These values must account for the energy released or absorbed in chemical reactions by being defined appropriately relative to the species that can react to make the one in question. As a result, h_0, must be explicitly provided with the model, and the models must be constructed with the same conventions.

Entropy is calculated from its definition with the benefit of the ideal gas equation of state,

$$\mathrm{d}s = \frac{\mathrm{d}h}{T} - \frac{v\mathrm{d}p}{T} \tag{4}$$

$$s(T,p) = s_0 + \int \frac{c_p(T)}{T}\mathrm{d}T - R\ln\left(\frac{p}{p_0}\right) \tag{5}$$

$$\tag{6}$$

Here, s_0 appears in the same way as h_0, and must be explicitly defined to account for changes in entropy due to chemical reactions. Because of the temperature term in the denominator, no multiple of 1000 appears in entropy,

$$\int \frac{c_p(T)}{T}\mathrm{d}T = \int \frac{c_p(t)}{t}\mathrm{d}t. \tag{7}$$

Therefore,

$$\begin{aligned} s(T,p) = s_0 + c_0\ln t + c_1 t + \frac{c_2}{2}t^2 + \dots \\ \frac{c_3}{3}t^3 - R\ln\left(\frac{p}{p_0}\right) \end{aligned} \tag{8}$$

To the irritation of some, PYroMatuses e (rather than the traditional u) for internal energy. This is motivated by a (perhaps hopeless) attempt to resolve some of the many collisions between notations in thermodynamics and fluid mechanics (wherein u is usually used for velocity) and v is used in both to represent a volume.

Internal energy is readily calculated from the definition of enthalpy,

$$\begin{aligned} e(T) &= h(T) - \frac{p}{\rho} \\ &= h(T) - RT. \end{aligned} \tag{9}$$

There is a similarly simple relationship to determine constant-volume specific heat and specific heat ratio,

$$c_v(T) = c_p(T) - R \tag{10}$$

$$\gamma(T) = \frac{c_p(T)}{c_p(T) - R} \tag{11}$$

3 Polynomials of two variables

In the evaluaiton of empirical thermodynamic properties, there is often the problem of evaluating polynomial expansions on two variables. These expansions are typically of the form

$$p(x,y) = \sum_{i,j} c_{i,j} x^i y^j \tag{12}$$

where a and b are real coefficients such that i and j are integer indices.

3.1 Modifying polynomials for non-integer and negative powers

Fractional and negative exponents are also possible within this framework if we were to accept input values X, and Y, and adjust them according to pre-exponentials a and b,

$$x = X^a \tag{13}$$

$$y = Y^b. \tag{14}$$

The new polynomial formed is

$$p(X, Y) = \sum_{i,j} c_{i,j} X^{ai} Y^{bj}. \tag{15}$$

For example, were $a = 0.25$, then the expansion on X could proceed in fractions of 4 even though the evaluation algorithm we use is purely expressed in integer exponents. It is relatively computationally inexpensive to apply these exponentials prior to the code's execution. It would be far more costly to apply them to each term.

Similarly, negative exponentials can be achieved by using post-exponential terms

$$P(X, Y) = X^\alpha Y^\beta p(X, Y). \tag{16}$$

When α or β are non-zero, the effective exponents are all shifted positively or negatively by a single multiplication operation.

3.2 Efficient evaluation of the polynomial

The PYroMat polynomial evaluation algorithm is an expansion with purely integer exponents.

$$p(x, y) = \sum_{i,j} c_{i,j} x^i y^j \tag{17}$$

However, evaluating each term individually requires two expensive calls to a `pow` function and two floating point multiplications.

The widely accepted method for evaluating a polynomial of one variable is to construct a recursive expansion

$$q(y) = c_0 + y(c_1 + y(c_2 + y(\dots \tag{18}$$

If there are n coefficients, then this amounts to only m multiplications with no `pow` calls. In order to extend this algorithm to two variables, more elegant notation will be helpful. If we name the intermediate value calculated in the process of these recursions q, then a polynomial with n terms implies the series

$$q_n = c_n \tag{19}$$

$$q_j(y) = c_j + y\, q_{j+1}(y) \tag{20}$$

$$q_0(y) = q(y). \tag{21}$$

This is a series beginning with q_n, and proceeding backwards through the values of j to q_0, which is the final value for $q(y)$. In practice, there is no need to keep the old values of q, so a single register may be used to hold the latest value.

How can this be extended to a polynomial of two variables? We may consider the polynomials to be nested; the evaluation of a polynomial on Y determines the individual coefficients for a polynomial on X.

$$p(x, y) = \sum_i q_i(y) x^i \tag{22}$$

$$q_i(y) = \sum_j c_{i,j} y^j \tag{23}$$

We only need a minor modification to the intermediate values for the x polynomial since there will be a separate expansion for each value of i. If there are n j terms,

$$q_{i,n}(y) = c_{n,j} \tag{24a}$$

$$q_{i,j}(y) = c_{i,j} + y\, q_{i+1,j}(y) \tag{24b}$$

$$q_{i,0}(y) = q_i(y). \tag{24c}$$

If there are m x terms,

$$p_m(x) = q_m(x) \tag{25a}$$
$$p_i(x,y) = q_i(x) + y\, p_{i+1}(x,y) \tag{25b}$$
$$p_0(x,y) = p(x,y). \tag{25c}$$

3.3 Efficient evaluation of derivatives

The partial derivatives of the polynomial can be efficiently evaluated along with the polynomial itself. To relax the already cumbersome notation, the functional dependencies (y) and (x,y) will be dropped. For the purpose of thermodynamic property evaluation, the first two derivatives will suffice.

Let us begin with the simpler task of calculating the derivatives of q_j.

$$q_{i,n|y} = 0 \tag{26a}$$
$$q_{i,j|y} = q_{i+1,j} + y\, q_{i+1,j|y} \tag{26b}$$
$$q_{i,0|y} = q_{i|y} \tag{26c}$$

$$q_{i,n|yy} = 0 \tag{27a}$$
$$q_{i,j|yy} = 2q_{i,j+1|y} + y\, q_{i,j+1|yy} \tag{27b}$$
$$q_{i,0|yy} = q_{i|yy} \tag{27c}$$

The derivatives on p are constructed somewhat differently because they can be in both x and y. Beginning with y,

$$p_{n|y} = 0 \tag{28a}$$
$$p_{j|y} = q_{i|y} + x\, p_{j+1|y} \tag{28b}$$
$$p_{0|y} = p_y \tag{28c}$$

$$p_{m|yy} = 0 \tag{29a}$$
$$p_{i|yy} = q_{i|yy} + x\, q_{i+1|yy} \tag{29b}$$
$$p_{0|yy} = p_{yy} \tag{29c}$$

The derivatives on x appear

$$p_{m|x} = 0 \tag{30a}$$
$$p_{i|x} = p_{i+1} + x\, p_{i+1|x} \tag{30b}$$
$$p_{0|x} = p_x \tag{30c}$$

$$p_{n|xx} = 0 \tag{31a}$$
$$p_{i|xx} = 2p_{i+1|x} + x\, p_{i+1|xx} \tag{31b}$$
$$p_{0|xx} = p_{xx} \tag{31c}$$

Finally, the cross-term (both x and y) appears

$$p_{n|xy} = 0 \tag{32a}$$
$$p_{i|xy} = p_{i+1|y} + y\, p_{i+1|xy} \tag{32b}$$
$$p_{0|xy} = p_{xy} \tag{32c}$$

3.4 Implementation of the algorithm

In practice, this cumbersome notation can be drastically simplified in code because it is not necessary to distinguish between p and q in their various incarnations; provided care is taken not to overwrite a value before it is needed.

In most practical polynomials of two variables of given order, very few of the possible coefficients may be non-zero, so storing and looping over all $m \times n$ coefficients may not be sensible. Instead, it is common to take an approach closer to spare matrix storage.

If we have one-dimensional arrays of polynomial coefficients, c_k, and exponents, a_k and b_k, the polynomial will be constructed as

$$p(X,Y) = \sum_k = 0^{N-1} c_k X^{a_k} Y^{b_k}. \tag{33}$$

In this way, the polynomial

$$p(X,Y) = -0.1X^2 + XY + 0.5Y^2 - Y - 0.2 \tag{34}$$

may be represented by

$$a = [2,\ 1,\ 0,\ 0,\ 0] \tag{35}$$
$$b = [0,\ 1,\ 2,\ 1,\ 0] \tag{36}$$
$$c = [-0.1,\ 1,\ 0.5,\ -1,\ 0.2] \tag{37}$$

For the algorithm to function efficiently, it is reasonable to impose some prior sorting of the exponent values. Since the series above requires that we interact with higher-order terms first, let us assert that the polynomial should be expressed in order of descending exponents on X and then Y.

In Algorithm 1, we employ an outer loop on the values of i and an inner loop on the values of j. If coefficients are absent from the arrays (if the i, j pair is not found), then the coefficient is presumed to be zero. Starting with the maximum value for each exponent, the indices are reduced incrementally until the i, j combination corresponding to the next row (k) is found.

This is extremely practical for polynomials where most of the possible combinations are not represented, and does not cost much for cases where they are.

Algorithm 1 Efficient evaluation of a polynomial of two variables

Require: $a, b, c \in \mathbb{R}^N$
Require: The coefficient order is sorted in descending order of a and then b so that
$a_k \geq a_{k+1}\ \forall\ k \in [0, N-1)$
$b_k > b_{k+1}\ \forall\ k : a_k = a_{k+1}$
Ensure: p, p_x, p_y, p_{xx}, p_{xy}, p_{yy} are equal to the polynomial and its derivatives.

$p, p_x, p_y, p_{xx}, p_{xy}, p_{yy} \leftarrow 0$
$i_{max} \leftarrow a[0]$
$k \leftarrow 0$
for $i = i_{max}$ to 0 **do**
  if $k < N$ **and** a_k equals i **then**
    $j_{max} \leftarrow b_k$
    $q, q_y, q_{yy} \leftarrow 0$
    for $j = j_{max}$ to 0 **do**
      $q_{yy} \leftarrow 2q_y + yq_{yy}$
      $q_y \leftarrow q + yq_y$
      if $k < N$ **and** a_k is i **and** b_k is j **then**
        $q \leftarrow c_k + yq$
        $k \leftarrow k + 1$
      else
        $q \leftarrow yq$
      end if
    end for
    $p_{yy} \leftarrow q_{yy} + xp_{yy}$
    $p_{xx} \leftarrow 2p_x + xp_{xx}$
    $p_{xy} \leftarrow p_y + xp_{xy}$
    $p_x \leftarrow p + xp_x$
    $p_y \leftarrow q_y + xp_y$
    $p \leftarrow q + xp$
  else
    $p_{yy} \leftarrow xp_{yy}$
    $p_{xx} \leftarrow 2p_x + xp_{xx}$
    $p_{xy} \leftarrow p_y + xp_{xy}$
    $p_x \leftarrow p + xp_x$
    $p_y \leftarrow xp_y$
    $p \leftarrow xp$
  end if
end for