The Thermodynamic Models in PYroMat

Christopher R. Martin

March 4, 2021

1 Introduction

As of PYroMat version 2.1.0, there are three general model formulations used to evaluate the thermodynamic properties of gases and multi-phase (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.

2 Ideal Gases

2.1 Introduction

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 PYroMat for 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.

When they are spread so sparsely that intermolecular forces between molecules are only relevant for the brief moments known as "collisions," ideal gases are well described by the relation

$$p = \rho R T. \tag{1}$$

p is the pressure in Pa, ρ is density in kg/m^3, R is the gas constant for the gas, and T is the temperature in K. When density is expressed in molar terms (usually called concentration) R may be replaced by the universal gas constant, R_u.

In our treatment of gas properties, we will require the first law of thermodynamics, which states that energy must be conserved, and

$$\delta q = \mathrm{d}e + p\mathrm{d}v. \tag{2}$$

Here, q is heat in Joules, e is the internal energy of the substance (energy stored in electrical, vibrational, chemical, nuclear, and translational energy of molecules) in Jounles per kilogram per Kelvin, p is pressure in Pascals, and v is specific volume (or $1/\rho$).

Some will be irritated that u is traditionally used for internal energy, but we have abandoned that tradition in an attempt (perhaps hopeless) to resolve some of the collisions in notation between fluid mechanics and thermodynamics. We reserve v for volume, u for velocity, and e for energy. One possible objection might be that e should be reserved for the natural number, which appears in many derivations. Instead, we will always use the expression, exp(), when we must.

The specific heat is the energy required to obtain a finite increase in temperature of finite amount of a substance. When heat is applied in such a manner that the substance's volume is constant, $\mathrm{d}v = 0$, and

$$\begin{aligned}\delta q|_v &= \mathrm{d}e \\ &= \left(\frac{\partial e}{\partial T}\right)_v \mathrm{d}T.\end{aligned} \tag{3}$$

Thus, the constant-volume specific heat is, by definition,

$$c_v \equiv \left(\frac{\partial e}{\partial T}\right)_v. \tag{4}$$

In an ideal gas, c_v is treated as *only* a function of temperature, so we may drop the specification of constant-volume. Much could be said about this assumption; it implicitly assumes that when a system of colliding molecules come to equilibrium, the fraction of internal energy expressed in translational kinetic energy does not depend on how tightly packed the molecules are. This goes hand-in-hand with the assumption that intermolecular forces need not be modeled in detail.

When one considers addition of heat under constant pressure, a trick application of the chain rule for the term, pv, lets us transition the differential on volume into a differential on pressure. The definition for enthalpy, $h = e + pv$, appears naturally.

$$\begin{aligned}\delta q &= \mathrm{d}e + p\mathrm{d}v + v\mathrm{d}p - v\mathrm{d}p \\ &= \mathrm{d}(e + pv) - v\mathrm{d}p\end{aligned} \tag{5}$$

To engineers practiced in the arts of fluid power system design will immediately recognize pv as the term quantifying the mechanical energy carried by a flowing fluid. Including it alongside e merely means that we are accounting for mechanical energy communicated by the bulk material alongside the energy stored in the individual molecules of the substance.

Of course, when pressure is constant, $\mathrm{d}p = 0$, and

$$\begin{aligned} \delta q|_p &= \mathrm{d}h \\ &= \left(\frac{\partial h}{\partial T}\right)_p \mathrm{d}T \end{aligned} \tag{6}$$

Thus, constant-pressure specific heat is, by definition,

$$c_p \equiv \left(\frac{\partial h}{\partial T}\right)_p . \tag{7}$$

$$= \left(\frac{\partial e}{\partial T}\right)_p + \left(\frac{\partial (pv)}{\partial T}\right)_p . \tag{8}$$

We have already established that, for an ideal gas, internal energy is only a function of temperature, and $pv = RT$. So,

$$c_p(T) = c_v(T) + R, \tag{9}$$

and c_p is also only a function of temperature.

Calorimetry provides means by which the specific heats of gases may be very precisely measured under different conditions. So, provided with a sufficiently accurate function for specific heat, it is possible to calculate the other properties by its integral.

Enthalpy is readily calculated from c_p,

$$h^\circ(T) = \Delta h_f^\circ(T_{ref}) + \int_{T_{ref}}^{T} c_p(\tau)\mathrm{d}\tau. \tag{10}$$

The notation h° emphasizes that this is the enthalpy at standard pressure (usually 1bar for ideal gas data), but h may be presumed to be insensitive to pressure so long as the ideal gas assumption holds.

The integration constant, Δh_f°, is called the enthalpy of formation. It describes the enthalpy consumed when forming the species from other "reference" species. Reference species (like O_2, N_2, H_2, C(s), He, Ar, Kr, and others) are arbitrarily defined to have zero enthalpy at 298.15K and 1bar, and calorimetry experiments including reactions with them allow the enthalpies of all other species to be constructed. The enthalpy of formation is expressed in terms of the reference temperature, since the choice of reference temperature is arbitrary. It is common to use $T_{ref} = 298.15\text{K}$.

There is a similar expression for standard entropy, which can be obtained from the definition of entropy and the first law. For an ideal gas,

$$\begin{aligned}\mathrm{d}s &= \frac{\delta q}{T} = \frac{\mathrm{d}h}{T} - \frac{v\mathrm{d}p}{T} \\ &= c_p\frac{\mathrm{d}T}{T} - R\frac{\mathrm{d}p}{p}. \end{aligned} \tag{11}$$

The definition for enthalpy had no absolute zero reference, so we had to construct one from a convention. However, the entropy for a substance *can* be defined as zero when absolute temperature is zero. Therefore, it is possible to define the entropy at a reference temperature and pressure.

$$s^\circ(T_{ref}, p_{ref}) = \int_0^T \frac{c_p(\tau)}{\tau}\mathrm{d}\tau \tag{12}$$

From that state,

$$\begin{aligned} s(T,p) &= s^\circ(T_{ref}, p_{ref}) + \int_{T_{ref}}^T \frac{c_p(\tau)}{\tau}\mathrm{d}T - \int_{p_{ref}}^p \frac{R}{\pi}\mathrm{d}\pi \\ &= s^\circ(T_{ref}, p_{ref}) + \int_{T_{ref}}^T \frac{c_p(\tau)}{\tau}\mathrm{d}\tau - \ln\left(\frac{p}{p_{ref}}\right) \end{aligned} \tag{13}$$

PYroMat calculates $s(T,p)$, but tables usually only list entropy as a function of temperature. Why? Since the pressure contribution to entropy can be calculated explicitly, the "standard enthalpy," s°, is normally what is tabulated,

$$s^\circ(T) = s(T, p_{ref}) = s^\circ(T_{ref}, p_{ref}) + \int_{T_{ref}}^T \frac{c_p}{T}\mathrm{d}T. \tag{14}$$

Note that we could propose conditions at pressures so large, $s(T,p)$ would evaluate to be zero or even negative. Long before that occurs, the ideal gas assumption would be violated. At such extreme densities, intermolecular forces would become relevant, and this expression would be beyond the scope of its usefulness.

Once $c_p(T)$ and $h(T)$ are well defined, it is also possible to evaluate internal energy,

$$e(T) = h(T) - RT, \tag{15}$$

constant-volume specific heat,

$$c_v(T) = c_p(T) - R, \tag{16}$$

specific heat ratio,

$$\gamma(T) = \frac{c_p(T)}{c_p(T) - R}, \tag{17}$$

speed of sound

$$a(T) = \sqrt{\gamma(T) R T}, \tag{18}$$

and others.

The central problem, then, is how to calculate $c_p(T)$.

As becomes clear in the next sections, polynomials are important to these formulations; so much so that we devote a separate section to their efficient evaluation.

2.2 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.

The Shomate equation takes the form

$$t = \frac{T}{T_s} \tag{19}$$

$$c_p(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + \frac{c_4}{t^2}, \tag{20}$$

where the scaling temperature, T_s is 1000K for all species. Reducing the argument to the polynomial, t, to by on the order of 1 is an intelligent step to help reduce numerical errors, but it should be obvious that there is no attempt to base the formulation on 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 $298\text{K} \le T < 1100\text{K}$ and $1100\text{K} \le T \le 6000\text{K}$.

The enthalpy can be explicitly calculated from (10),

$$\begin{aligned}
h(T) &= h_0 + \int c_p(T)\mathrm{d}T \\
&= h_0 + T_s \int c_p(t)\mathrm{d}t \\
&= T_s \left(c_0 t + \frac{c_1}{2}t^2 + \frac{c_2}{3}t^3 + \frac{c_3}{4}t^4 - \frac{c_4}{t} + c_5 \right). \qquad (21)
\end{aligned}$$

It is important to emphasize that h_0 is not the same as the enthalpy of formation, Δh_f°. Instead, it is merely an integration constant, which can be alternately expressed as a new coefficient, c_5.

Because of the temperature term in the denominator, no multiple of T_s appears in entropy when the integration is changed to t,

$$\begin{aligned}
s(T,p) &= s^\circ(T_{ref}) + \int_{T_{ref}}^{T} \frac{c_p(\tau)}{\tau}\mathrm{d}\tau - R\ln\left(\frac{p}{p_{ref}}\right) \\
&= s^\circ(T_{ref}) + \int_{T_{ref}}^{T} \frac{c_p(\tau)}{\tau}\mathrm{d}\tau - R\ln\left(\frac{p}{p_{ref}}\right) \\
s(T,p) &= c_0 \ln t + c_1 t + \frac{c_2}{2}t^2 + \frac{c_3}{3}t^3 - \frac{c_4}{2t^2} + c_6 - R\ln\left(\frac{p}{p_{ref}}\right). \qquad (22)
\end{aligned}$$

Just like in the enthalpy integral, a new coefficient, c_6, has been introduced to represent the integration constant.

Internal energy is readily calculated from the definition of enthalpy in (**??**),

$$e(T) = h(T) - RT \qquad (23)$$

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

$$c_v(T) = c_p(T) - R \qquad (24)$$

$$\gamma(T) = \frac{c_p(T)}{c_p(T) - R} \qquad (25)$$

2.3 IG2: The NASA polynomial

Like the Shomate equation, the NASA polynomials are a piece-wise empirical formulation to the specific heat of an ideal gas. Unlike the Shomate equation, there is no $1/t^2$ term, they make no attempt to scale temperature prior to evaluating the polynomial, and they are scaled with respect to the species' ideal gas constant.

$$\frac{c_p(T)}{R} = R\left(c_0 + c_1 T + c_2 T^2 + c_3 T^3 + c_4 T^4\right) \tag{26}$$

There are nearly identical formulations for enthalpy,

$$h(T) = R\left(c_0 T + \frac{c_1}{2}T^2 + \frac{c_2}{3}T^3 + \frac{c_3}{4}T^4 + \frac{c_4}{5}T^5 + c_5\right), \tag{27}$$

and entropy

$$s(T) = R\left(c_0 \ln(T) + c_1 T + \frac{c_2}{2}T^2 + \frac{c_3}{3}T^3 + \frac{c_4}{4}T^4 + c_6\right). \tag{28}$$

Here, just as in the Shomate equations, c_5 and c_6 are introduced as integration constants in enthalpy and entropy.

Internal energy is readily calculated from the definition of enthalpy in (**??**),

$$e(T) = h(T) - RT \tag{29}$$

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

$$c_v(T) = c_p(T) - R \tag{30}$$

$$\gamma(T) = \frac{c_p(T)}{c_p(T) - R} \tag{31}$$

3 Multi-phase properties

4 Efficiently evaluating polynomials and their derivatives

It must be clear from the previous sections that polynomials are an important tool for accurately modeling thermodynamic properties. However, how they are best evaluated in code is more nuanced than it may seem.

For example, if we were to evaluate x^4, we might use "x**4" in Python or we might use "x*x*x*x". Which is better? The answer depends on how much data we are analyzing. Recall that PYroMat supports arrays.

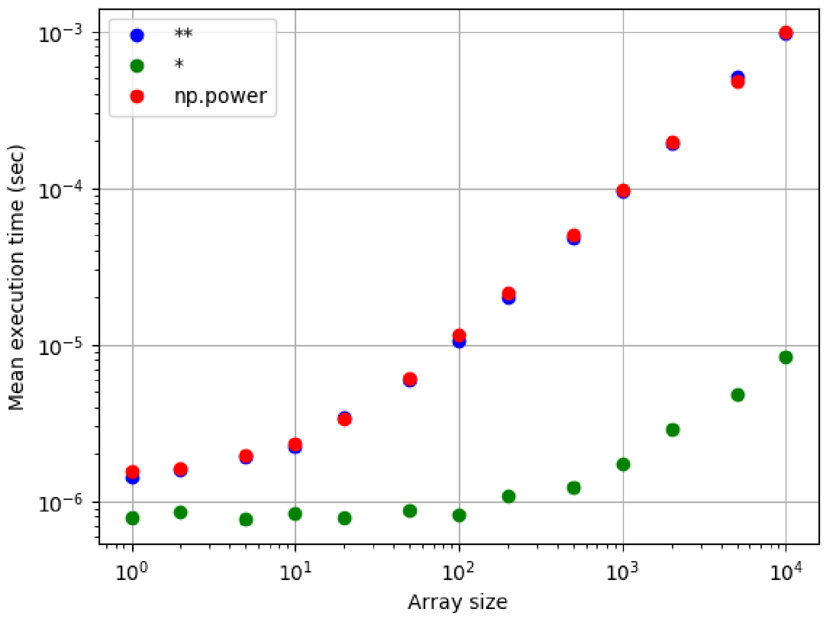

Figure 1: Computation time for array multiplication, and power operations.

Figure 1 shows the results of a computation time study on the time to multiply Numpy arrays of different sizes. The "**" and "np.power" operations were shown to be equivalent. Certainly on faster or slower processors, the time scale might be faster or slower, but the shape of the curve should be roughly the same.

For small data sets (fewer than 10 or so elements), the overhead associated with Python being an interpreted language dominates the computational time, so adding array elements has little or no impact. We could probably only perform two or three multiplications before the costs start to appear equal. On the other hand, the computational costs are so small in these cases, it probably doesn't matter whether we use multiplication or exponents.

The real cost appears when data sets are larger. By the time the arrays are over 1,000 elements, it is clear that multiplication and exponents diverge in their cost. On a 10,000 element array, 100 multiplications can be performed for the cost of a single exponent and that trend only worsens with larger arrays.

The conclusion is that exponents should be used rarely (if ever) and repeated multiplications are dramatically more efficient. This is somewhat less true on small data sets where there is significant computational overhead, but these operations are also so inexpensive to begin with, that there is little pressure on the optimization of the code.

The remaining question is how best to minimize the number of multiplications.

4.1 Polynomials of one variable

Polynomials of one variable may be expressed

$$p(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \dots \tag{32}$$

We have made the case that it is preferable to express x^3 as $x \times x \times x$, but if that is the case, $x \times x$ would already have been available when calculating x^2, so we should absolutely avoid redundant operations. An N^{th} order polynomial would imply $N(N-1)$ multiplications and N additions. On the other hand, if the polynomial were to be evaluated as

$$p(x) = c_0 + x(c_1 + x(c_2 + x(c_3 + \dots \tag{33}$$

there are only N multiplications and N additions.

While numerically expedient, it leads to a very awkward notation. Instead, consider the polynomial constructed as a series. For an N^{th} order polynomial,

$$\eta_N(x) = c_N \tag{34a}$$
$$\eta_k(x) = c_k + x\eta_{k+1} \quad \forall\ k\ :\ 0 \le k < N \tag{34b}$$
$$\eta_0(x) = p(x). \tag{34c}$$

4.2 Polynomials of two variables

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

$$p(x, y) = \sum_{i,j} c_{i,j} x^i y^j \tag{35}$$

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

4.3 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,

$$x = X^a \tag{36}$$

$$y = Y^b. \tag{37}$$

The new polynomial formed is

$$p(X, Y) = \sum_{i,j} c_{i,j} X^{ai} Y^{bj}. \tag{38}$$

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

$$P(X, Y) = X^\alpha Y^\beta p(X, Y). \tag{39}$$

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

4.4 Efficient evaluation of the polynomial

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

$$p(x, y) = \sum_{i,j} c_{i,j} x^i y^j \tag{40}$$

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

$$q(y) = c_0 + y(c_1 + y(c_2 + y(\ldots \tag{41}$$

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

$$q_n = c_n \tag{42}$$
$$q_j(y) = c_j + y\, q_{j+1}(y) \tag{43}$$
$$q_0(y) = q(y). \tag{44}$$

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.

$$p(x, y) = \sum_i q_i(y) x^i \tag{45}$$
$$q_i(y) = \sum_j c_{i,j} y^j \tag{46}$$

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,

$$q_{i,n}(y) = c_{n,j} \tag{47a}$$
$$q_{i,j}(y) = c_{i,j} + y\, q_{i+1,j}(y) \tag{47b}$$
$$q_{i,0}(y) = q_i(y). \tag{47c}$$

If there are m x terms,

$$p_m(x) = q_m(x) \tag{48a}$$
$$p_i(x, y) = q_i(x) + y\, p_{i+1}(x, y) \tag{48b}$$
$$p_0(x, y) = p(x, y). \tag{48c}$$

4.5 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.

$$q_{i,n|y} = 0 \tag{49a}$$
$$q_{i,j|y} = q_{i+1,j} + y\, q_{i+1,j|y} \tag{49b}$$
$$q_{i,0|y} = q_{i|y} \tag{49c}$$

$$q_{i,n|yy} = 0 \tag{50a}$$
$$q_{i,j|yy} = 2q_{i,j+1|y} + y\, q_{i,j+1|yy} \tag{50b}$$
$$q_{i,0|yy} = q_{i|yy} \tag{50c}$$

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

$$p_{n|y} = 0 \tag{51a}$$
$$p_{j|y} = q_{i|y} + x\, p_{j+1|y} \tag{51b}$$
$$p_{0|y} = p_y \tag{51c}$$

$$p_{m|yy} = 0 \tag{52a}$$
$$p_{i|yy} = q_{i|yy} + x\, q_{i+1|yy} \tag{52b}$$
$$p_{0|yy} = p_{yy} \tag{52c}$$

The derivatives on x appear

$$p_{m|x} = 0 \tag{53a}$$
$$p_{i|x} = p_{i+1} + x\, p_{i+1|x} \tag{53b}$$
$$p_{0|x} = p_x \tag{53c}$$

$$p_{n|xx} = 0 \tag{54a}$$
$$p_{i|xx} = 2p_{i+1|x} + x\, p_{i+1|xx} \tag{54b}$$
$$p_{0|xx} = p_{xx} \tag{54c}$$

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

$$p_{n|xy} = 0 \tag{55a}$$

$$p_{i|xy} = p_{i+1|y} + y\, p_{i+1|xy} \tag{55b}$$

$$p_{0|xy} = p_{xy} \tag{55c}$$

4.6 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

$$p(X, Y) = \sum_k = 0^{N-1} c_k X^{a_k} Y^{b_k}. \tag{56}$$

In this way, the polynomial

$$p(X, Y) = -0.1X^2 + XY + 0.5Y^2 - Y - 0.2 \tag{57}$$

may be represented by

$$a = [2,\ 1,\ 0,\ 0,\ 0] \tag{58}$$

$$b = [0,\ 1,\ 2,\ 1,\ 0] \tag{59}$$

$$c = [-0.1,\ 1,\ 0.5,\ -1,\ 0.2] \tag{60}$$

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.
  1: $p, p_x, p_y, p_{xx}, p_{xy}, p_{yy} \leftarrow 0$
  2: $i_{max} \leftarrow a[0]$
  3: $k \leftarrow 0$
  4: for $i = i_{max}$ to 0 do
  5:    if $k < N$ and $a_k$ equals $i$ then
  6:       $j_{max} \leftarrow b_k$
  7:       $q, q_y, q_{yy} \leftarrow 0$
  8:       for $j = j_{max}$ to 0 do
  9:          $q_{yy} \leftarrow 2q_y + yq_{yy}$
 10:          $q_y \leftarrow q + yq_y$
 11:          if $k < N$ and $a_k$ is $i$ and $b_k$ is $j$ then
 12:             $q \leftarrow c_k + yq$
 13:             $k \leftarrow k + 1$
 14:          else
 15:             $q \leftarrow yq$
 16:          end if
 17:       end for
 18:       $p_{yy} \leftarrow q_{yy} + xp_{yy}$
 19:       $p_{xx} \leftarrow 2p_x + xp_{xx}$
 20:       $p_{xy} \leftarrow p_y + xp_{xy}$
 21:       $p_x \leftarrow p + xp_x$
 22:       $p_y \leftarrow q_y + xp_y$
 23:       $p \leftarrow q + xp$
 24:    else
 25:       $p_{yy} \leftarrow xp_{yy}$
 26:       $p_{xx} \leftarrow 2p_x + xp_{xx}$
 27:       $p_{xy} \leftarrow p_y + xp_{xy}$
 28:       $p_x \leftarrow p + xp_x$
 29:       $p_y \leftarrow xp_y$
 30:       $p \leftarrow xp$
 31:    end if
 32: end for
```