New Calculation of Saturation Specific Humidity and Saturation Vapor Pressure in the COLA Atmospheric General Circulation Model

L. Marx

Center for Ocean-Land-Atmosphere Studies
Institute of Global Environment and Society, Inc.
4041 Powder Mill Road, Suite 302
Calverton, MD 20705

E-mail: marx@cola.iges.org

November 2002

Abstract

A new method of computing the saturation specific humidity, q_s, and saturation vapor pressure, e_s, is presented. This has been recently implemented in the Center for Ocean-Land-Atmosphere Studies (COLA) atmospheric general circulation model (AGCM). The new method offers high accuracy extending beyond the observed atmospheric temperature domain and yet is computationally competitive with most other methods used for this calculation. In addition a corresponding continuous function for the derivative of e_s with respect to temperature is also given. An inverse function to e_s is available as well. Included are small corrections to e_s to assure high accuracy. The design of the implementation permits the user to obtain whatever level of accuracy and detail might be desired. With this even detailed microphysical calculations with high computational efficiency can be done with small changes when the need arises.

1. Introduction

Water vapor is a key constituent of the atmosphere and plays a significant role in radiative and thermodynamic processes that affect the evolution of the atmospheric state through time. Typically water vapor is represented by the variable specific humidity, the amount of water vapor in the mixture of water vapor and dry air, in both analysis of the atmospheric state as well as various physically based models describing the evolution of the atmospheric state. For example, the Center for Ocean-Land-Atmosphere Studies (COLA) atmospheric general circulation model (AGCM) uses specific humidity to represent water vapor. Numerous physical processes require the local value of saturation specific humidity, q_s, which is the maximum amount of specific humidity that the air can contain at a given temperature and pressure. Previously, this calculation has been done in the COLA AGCM in a different way for nearly every physical process requiring q_s. This is undesirable, because a somewhat different atmospheric state is represented in each process and some moisture may become artificially lost or created. Furthermore, none of the saturation specific humidity calculations account for the reduced amount of water vapor as temperatures fall below freezing and ice becomes the predominant form of condensate. It therefore became necessary to revise the method of calculating q_s in the AGCM to one that is consistent, accurate and computationally efficient. Other less significant but important dependencies, such as the reduction in q_s due to sea surface salinity, can also be simply addressed as part of this revision.

Calculation of q_s is a fairly simple function of atmospheric pressure, p, and saturation vapor pressure, e_s. It is the calculation of e_s, which is usually viewed as a function of

atmospheric temperature only, that has caused difficulty in past treatments and has lead to numerous approximate calculation formulas. Many of these are discussed in Buck (1981) and will only be summarized here. In the past, detailed computation of e_s required accounting for many microphysical dependencies that were first given by the Goff and Gratch (1946) formulas for values of e_s over water and ice, which were also shown in List (1949). These formulas were based on older physical constant standards. Subsequently, Wexler (1976 and 1977) produced new formulas based on the newer Systemé International (SI) physical constants as well as newer microphysical theories and measurements with newer instruments. Due to the computational expense in computing the formulas based on microphysics, other approximate formulas have been used in atmospheric models. Since most of them have been based on the Goff and Gratch (1946) formulas or even predate them, they will not be detailed here. Some of the methods, however, will be briefly reviewed. These include simpler exponential and logarithm formulas, polynomial fits, and look-up tables with linear interpolation.

Formulas based on exponential functions that sometimes include an additional logarithm calculation have the advantage that they are generally not domain limited, at least over observed atmospheric temperatures (&90°C to +70°C). These formulas are usually constructed so that both the inverse of the function and the derivative of the function can be determined analytically. The exponential-based formulas are easy to implement in most AGCMs, and two formulas are used in the COLA AGCM for large scale precipitation processes and separately for land and water surfaces. This approach can be limited in computational efficiency depending on the local machine's EXP (and LOG) function speed. The actual accuracy is dependent on the specific formulas used. The higher accuracy formulas given by Buck (1981) are probably as accurate as

can be done by this approach but still can have errors as large as 0.05%.

Polynomial-based formulas are generally the most computationally efficient on most machines, but have a more limited domain (typically &50°C to +50°C) and are usually somewhat less accurate than exponential-based formulas. These are often used where iterative calculations may require frequent calculation of e_s and some loss of accuracy can be accepted. In the COLA AGCM, both the relaxed Arakawa-Schubert (RAS, Moorthi and Suarez, 1992) and Kuo (1965) convection schemes use separate sixth-order polynomial formulas. When the temperature falls outside the accepted argument domain, exponential formulas are sometimes used or other corrective steps are taken. In some cases, an error or error code may be generated. These contingencies all increase the complexity of implementation. While the domain can be increased by increasing the order of the polynomial, local accuracy may be reduced along with increasing the computation time. Local behavior may even cease to be monotonic if not carefully constructed. In any case the new function values in the original domain will no longer exactly match the original function values so that extending the domain is even more of a problem. While constructing the derivative of the function is easy, computing the inverse function can be involved. Typically the Newton-Raphson method works best, but can require several iterations for convergence.

Look-up tables using linear interpolation are used where lower accuracy is generally expected; and while appearing to be computationally efficient, can become quite slow if coded poorly, as will be shown later. The COLA AGCM uses this method when computing rain evaporation from convection. While the inverse function can be done by using a hunt/search algorithm, which will be discussed below, the derivative is discontinuous at each node and thus

is not well behaved where convergent calculations are required. Nevertheless, the domain can be easily extended as needed and some increase in accuracy can be obtained by reducing the interval.

Changes in computer hardware technology with the use of memory cache along with a high performance microprocessor suggest an alternative approach to approximating e_s: cubic spline fit using a sufficiently small interval can give both a very accurate and reasonably fast approximation for use in atmospheric modeling, even those involved in detailed microphysics. For AGCMs, an interval of 1°C provides enough accuracy to accommodate even fairly involved microphysics while gaining added speed by eliminating multiplicative factors of one. Only highly detailed microphysical calculations used in such applications as cloud resolving models might require additional accuracy. In that case, an interval of say $\frac{1}{16}$°C should be more than adequate to meet these needs while adding no more than 10% computation time on most computers. This method is generally faster than most of the above methods except for polynomial approximation where it usually takes from 1.5 to 3 times longer to compute but has none of the disadvantages and is far more accurate than any of the above methods. All these will be addressed in the next sections.

2. Implementation

Cubic spline fits are generally done such that the value of the function and its derivative are continuous at each interval node. This gives a function that is piecewise smooth over its entire domain as well as its derivative. Since there is often a need to compute both the values of

e_s and its derivative, this approach seems best even though the error over each interval is not minimized. Instead, decreasing the interval size can be used to reduce the error to whatever level is desired. While the size of the interval can vary over its domain using this method, choosing a fixed interval simplifies the computer code necessary to implement the fit and adds to the efficiency. In addition it will be seen that using a 1°C interval further simplifies the code while retaining enough accuracy to satisfy most needs. Since the formulas for e_s are different over water and ice, a separate fit is needed for each. The choice of which to use when the temperature falls below freezing can be left to the user. In the COLA AGCM, the value over ice will always be used for the present time until more sophisticated microphysics are employed. This will also serve to compensate excess moisture found in the model at lower temperatures. Attempts to fit between the two over some limited interval may be subject to local conditions that are not correctly accounted for presently. It also makes it easy to maintain a consistent heat budget by switching the latent heat calculation from the vaporization value to the sublimation value when the temperature falls below freezing. Calculation of the values used at each interval are based on standard software: *Numerical Recipes in FORTRAN* (Press, et al., 1992) with some modifications.

The separate formulas for e_s (in Pascals) over water and ice as given by Wexler (1976 and 1977) as well as Buck (1981) are:

$$e_{sw} = \exp[-2991.2729\, t^{-2} - 6017.0128\, t^{-1} + 18.87643854 - 0.028354721\, t + 0.17838301\times10^{-4}\, t^2 - 0.84150417\times10^{-9}\, t^3 + 0.44412543\times10^{-12}\, t^4 + 2.858487 \ln t] \quad (1)$$

$$e_{si} = \exp[-5865.3696\, t^{-1} + 22.241033 + 0.013749042\, t - 0.34031775\times10^{-4}\, t^2 + 0.26967687\times10^{-7}\, t^3 + 0.6918651 \ln t] \quad (2)$$

where *t* is the temperature in K.

Since the formulas are supposed to have the same value at triple point (273.16K), (2) has been slightly modified so that when using higher precision arithmetic, such as IEEE 64-bit, the constant term (22.241033) is extended until the resulting value matches as closely as possible the triple point value from (1) without exceeding it. The exact value for this term becomes machine, compiler and compiler optimization choice dependent. For example, on the Compaq ES series with maximum optimization (O5), the value 22.241033076380849 is used. The complexities of (1) and (2) require use of higher precision arithmetic to retain stable values. Similarly the resulting cubic spline fits in 64-bit arithmetic retain enough stability and accuracy even at a 1°C interval that they are usually more accurate and stable than (1) and (2) evaluated in 32-bit arithmetic.

As discussed in *Numerical Recipes*, correct implementation of a cubic spline can be done by evaluating the function at interval nodes and solving a tridiagonal system to arrive at the second derivatives at each interval. Over a given domain, the correct solution requires having the exact value of the first derivative at the domain endpoints. However, if the domain can be extended, the tridiagonal solution will provide the correct values of the first derivative in the interior of the domain even when choosing zero for the first derivative at the domain endpoints. This method relies on decreasingly inaccurate values of the first derivative as one goes further into the interior. Eventually the roundoff of the arithmetic precision used prevents further error propagation. For this implementation, 32 extra intervals at each end of the domain are sufficient to eliminate further error propagation when using 64-bit arithmetic. This can be tested by adding more intervals and finding that the interior values remain unchanged. Thus when the tables are

constructed with these extra intervals, values requiring the use of the extra intervals are considered out of domain.

Once the tridiagonal system is solved, the spline can be evaluated at each interval based on the argument value: First the interval index, *i*, is determined by:

$$i = (t \,\&\, tb) \cdot kint + 1 \qquad (3)$$

where *t* is the argument being evaluated, *tb* is the lowest domain value (base temperature) of the spline, and *kint* is the integer reciprocal of the interval size Δt.

Then the spline for the interval between *i* and *i*+1 may be evaluated

for $t_h = i \cdot \Delta t + tb$ and $t_R = t_h \,\&\, \Delta t$ (the values of *t* at the interval nodes)

and $a = (t_h \,\&\, t)/\Delta t$ and $b = (t \,\&\, t_R)/\Delta t$ (the normalized value of *t* inside the interval)

as:

$$e_{sat} = a \cdot e_{s(i)} + b \cdot e_{s(i+1)} + \frac{(\Delta t)^2}{6}\left[\left(a^3 - a\right) \cdot e''_{s(i)} + \left(b^3 - b\right) \cdot e''_{s(i+1)}\right] \qquad (4)$$

where e_{sat} = approximate value of e_s at temperature *t*.
$e_{s(i)}$ = value of e_s at interval node *i* from (1) or (2)
$e0_{s(i)}$ = 2nd derivative of e_s at interval node *i* by tridiagonal solution separately for (1) and (2)

By this method e_{sat} is computed by linear interpolation plus a cubic correction. By using a fixed interval, some of the terms can be combined before evaluation so that the new terms:

$$e''^{*}_{s(i)} = \frac{(\Delta t)^2}{6} \cdot e''_{s(i)} \quad \text{and} \quad e^{*}_{s(i)} = e_{s(i)} - e''^{*}_{s(i)}$$

give a computationally reduced spline evaluation of:

$$e_{sat} = a \cdot \left(e^{*}_{s(i)} + a^2 \cdot e''^{*}_{s(i)}\right) + b \cdot \left(e^{*}_{s(i+1)} + b^2 \cdot e''^{*}_{s(i+1)}\right) \qquad (5)$$

In addition, if the interval is 1, the evaluation is further simplified since $t_h = i + tb$ and $t_R = t_h \,\&\, 1$

as well as a = t_h & t and b = t & t_R. Since the cubic spline can be evaluated analytically to obtain the derivative, the derivative of (4) reduces to:

$$e'_{sat} = \frac{(e_s(i+1) - e_s(i))}{\Delta t} + \frac{\Delta t}{6}\left[\left(1-3a^2\right)\cdot e''_s(i) + \left(3b^2-1\right)\cdot e''_s(i+1)\right] \qquad (6)$$

where e'_{sat} = approximate value of the 1st derivative of e_s at temperature t.
Similarly, a computationally reduced version of (6) can also be obtained.

The choice of 1°C interval (or any other interval) is based on using triple point to determine the base temperature for both spline fits. Thus each node is actually 0.01°C above the true Celsius temperature value as will be the case in the discussion below. Nevertheless, the small temperature difference still gives nearly exact results at the true Celsius temperatures. When choosing an interval, only powers of two (positive or negative) have been used to help preserve accuracy even though triple point cannot be exactly represented in a binary computer. Since an interval of 1°C is a simpler formula than any other interval, little is gained by choosing larger intervals. The only advantage is increased speed in computing the inverse function (see below).

The most efficient way to use the spline method to evaluate e_{sat} is to compute as many values at one time as possible using a single subroutine call or by evaluating directly from the tables. This helps to keep the table values memory cache resident as long as possible. However on some machines the control over cache residency may be limited.

While (1) is based on measurements from 0° to 100° C and (2) is based on measurements from &80° to 0°C, their physical basis is used to extend them to &100°C to allow for unobserved but potentially possible very cold free atmosphere temperatures. Outside the domain of &100° to +100°C, the value of &1 is returned to indicate that corrective action should be taken. In the COLA AGCM, the model is stopped to begin further investigation.

It is possible to also compute the inverse function: *t* as a function of e_s. This is not as simple as an analytically based inverse function but can be done such that the exact inverse value is obtained. Combining code from *Numerical Recipes,* this is done in two stages:

1. *Hunt/Search stage*

   A. *Hunt phase*

      Starting from a first guess value, the interval index is increased or decreased by increasing powers of two until the first guess value and new interval node index value of $e_s(i)$ bracket the argument value.

   B. *Search phase*

      Using a binary search algorithm, the index difference between the two bracketing indexes is successively reduced by half and the new bracketing index pair determined. This continues until the bracketing indexes differ by one. These are the interval node indexes (i, i+1) of the interval containing the argument value of e_s. This method works because e_s is a monotonically increasing function of temperature.

2. *Newton-Raphson stage*

   Using (5) and the computationally reduced version of (6) and starting with a value of temperature, *t*, that is halfway in the interval, the value of *t* is successively adjusted by subtracting $(e_{sat}\&e_s)/e\mathbb{N}_{sat}$ until little or no change to *t* is detected.

It turns out that depending on the choice of interval, a fixed number of iterations will guarantee the exact value of *t* without having to check the closeness of the solution at each iteration. For a 1°C interval and the &100° to +100°C domain, 4 iterations are enough. For $\frac{1}{16}$°C interval and the

same domain, 3 iterations are enough. Closeness is checked at the end of all iterations to be assured of convergence. If the exact result is not desired and computational time saving is important, the iteration loop can be unrolled and the check can be done after the second-to-last iteration.

Several other assumptions are made to obtain the inverse function. First, performance depends on how close the first guess index is to the bracketing index. Typically the last value calculated serves as a good first guess. Second, during iteration intermediate values of *t* may fall outside the interval. It is assumed that the original interval table values can still be used to obtain a solution. This causes no problem with the intervals tested and the number of iterations used. Even a value of *t* only one bit away from the interval node value, the most difficult to solve, poses no problem. The need for an extra iteration for the 1°C interval is an indication that this assumption can begin to have problems. Use of a larger interval may require more iterations to obtain a solution or may even fail to converge. Third, the choice of the initial index increment during the hunt phase depends on the interval choice and the distribution of the solution temperatures. If the value of the initial index increment is too small, extra time will be spent on the hunt phase; if too large, more time will be spent in the search phase. For horizontal pressure level data and an interval of 1°C, an index increment of 1 seems to give the best performance. For the same data at $\frac{1}{16}$°C interval a better choice seems to be 2 or 4 since adjacent locations are rarely less than $\frac{1}{16}$°C apart. Similar or larger values may also be better for vertically indexed data. Finally, it has been assumed that an analytical solution to the cubic equation instead of a Newton-Raphson solution would require more time. Since a cube root is involved, the standard Fortran method of using logarithms and exponentials as well as computing the remaining parts of the

calculation would make it less efficient. On some machines, a built-in cube root is available and the four-iteration solution for a 1°C interval may prove more expensive in that case. Since the analytical method using a built-in cube root is neither portable nor gives an exact solution, this has not been tested. The computer time to perform each stage is significant so that tradeoffs to improve performance are difficult. The four-iteration 1°C interval solution takes a little less total time (~10%) on some machines due to the reduced time in the hunt/search stage than the three-iteration $\frac{1}{16}$°C interval solution.

In the COLA AGCM, the inverse function is not required presently, so less emphasis has been given to improving its performance than might be desired. The inverse function is used in the post-processing and having an exact solution here is deemed more important.

3. Error analysis

Most approximations of e_s tend to cross or meet the original function a small number of times over the domain. This makes it possible to describe the errors by indicating mean values, root mean square values, extreme values or other statistics over fixed intervals or by using a table. Alternatively, the errors can be presented graphically over the domain. In using a spline approximation which has many matches to the original function throughout the domain, these methods of indicating error behavior are less satisfactory and can be incomplete. What follows is both a description of the errors over the domain and a graphical representation of the error distribution.

When using 64-bit precision arithmetic, the errors are zero at each interval node and

usually negative within the interval: the spline approximation is less than the original function. Regardless of the size of the interval, the errors tend to have a parabolic shape with maximum absolute value near the interval midpoint. The relative error magnitudes are largest at the lowest temperature, with values over ice somewhat larger than the values over water.

Because the relative errors here are quite small (<<1), it becomes convenient to represent the errors in a compact yet precise way: quantitative errors or Q_{err}. For discussion purposes Q_{err} can be defined as:

$$Q_{err} = \frac{(a - c)}{\text{spacing}(c)} \qquad (7)$$

where a = approximate value
c = correct value
spacing is the Fortran 90 spacing function: the absolute spacing of the number nearest the argument according to machine representation.

Note: a similar definition can also be made for quantitative difference, Q_{diff}, where neither value is assumed to be more correct than the other.

Conceptually, while Q_{err} will have only integer values, there is no guarantee that the division will result in an exact integer value. Actual computation requires non-Fortran methods such as unnormalized arithmetic or stripping (zeroing) the exponents (if they are the same) and computing the integer difference. The concept of Q_{err} comes from the fact that computers are limited in their ability to represent real or floating point data and that differences between two similar values can only be represented as integral multiples of the smallest possible difference that can be represented for either value (the error quantum). When $Q_{err}=\pm1$, the error is the smallest that can

be detected for a given precision. When errors are small multiples of this, it is a much more compact way to express the error.

There is a relationship between the standard relative error, RES=(*a*&*c*)/*c* with *a* and *c* as defined in (7), and the relative error computed from Q_{err}, REQ:

> REQ = Q_{err}/Machine Mantissa Precision
> The machine mantissa precision is a fixed function of the machine arithmetic precision. For most 32-bit representations, the machine mantissa precision is 2^{24} or 16,777,216. For IEEE 64-bit arithmetic, the machine mantissa precision is 2^{53} or 9,007,199,254,740,992.

As the values of *a* or *c* approach a power of two from below, REQ=RES. When the power of two value is just exceeded, REQ=½RES. Thus, over each power of two of the function result, REQ increases from ½RES to RES. This causes REQ (and Q_{err} as well) to behave unsteadily as the domain value increases. This makes Q_{err} less useful as its value increases and the functional behavior has a steadily larger scale error fluctuation over the domain. However, when errors are small and the behavior even over small intervals is nearly random or highly varied, Q_{err} can provide a better description of error behavior over the entire domain. While Q_{err} (or REQ) behaves less steadily than RES, it more accurately describes the actual error behavior on a given computer in terms of the realizable departure from its correct value.

The results presented here will focus on two interval choices: 1°C or $\frac{1}{16}$°C. Other choices are possible, but are not detailed since they are slower (…1°C), not precise enough (>1°C), not distinct enough from the others (between 1°C and $\frac{1}{16}$°C) or seem to provide little added benefit (<$\frac{1}{16}$°C). For the 64-bit spline fit at a 1°C interval, the 32-bit Q_{err} will be shown. For the 64-bit spline fit at $\frac{1}{16}$°C interval, the 64-bit Q_{err} will be described since all errors are within 32-bit precision but vary too much to be displayed graphically.

With Q_{err} only having integer values, it becomes convenient to use each value as a bin for expressing error distribution when the errors are small enough. We start first with the errors of 32-bit vs. 64-bit representations of (1) and (2). This will serve as a guide for evaluating spline fit error distributions.

Fig. 1 shows the distribution of Q_{err} at 1°C interval for 32-bit vs. 64-bit representations of (1) over the entire domain of &100° to +100°C. Each interval is the error for each temperature in the interval at 1/32768°C increments. For temperatures above 256K, all possible values of *t* are evaluated in IEEE 32-bit arithmetic. Below 256K every other value of temperature is evaluated. While potentially missing some values, the sample should still give a good estimate of the error behavior. Retaining the same increment presents a more uniform graphical display. It can be seen that the multiple terms in the exponential function of (1) give a fairly wide distribution of the Q_{err}. Above &70°C the distribution range remains the same and only the increasing separation of the succeeding power of two result values changes the shape of each power of two error distribution pattern. This is what one would expect from a nearly random error distribution, though perhaps larger than expected. The slight negative bias of the mean value of the distribution (~&7) probably comes from the inability of the 32-bit function to come closer to the mean due to the high amplification of differences: a Q_{diff}=1 in temperature gives Q_{diff} . 30 in e_s throughout most of the domain. Below &70°C, the errors become larger, nearly doubling at around &100°C. This error increase toward colder temperatures will be seen in the other figures as well and may be due to a reduction of significance or other numerically related problem.

Fig. 2 shows the distribution of Q_{err} at 1°C interval for 32-bit vs. 64-bit representations of (2) over its domain of &100° to 0°C. The errors are smaller and fairly uniform down to about

&85°C. The bias seems to be slightly positive (~+3). The smaller errors and bias magnitude may all be due to fewer terms in the exponential for (2): 6 instead of 8.

The corresponding distributions for the 64-bit 1°C interval cubic spline fits to (1) and (2) using the same temperatures can be seem in Figs. 3 and 4. Fig. 3 shows that the errors over water are quite small over much of the domain with 32-bit Q_{err} either &1 or 0 for temperatures above &20°C and gradually reaching &37 at &100°C, and Q_{err} of +1 only seen below &50°C. In Fig. 4, larger errors are reached at warmer temperatures, but since the values over ice are smaller than over water, the tendency for errors to increase with smaller function values is also enhanced. Positive Q_{err} still never exceed +1. While both fits show a negative bias, they are usually much smaller than the errors in their 32-bit full function counterparts. It should be noted that since the splines are computed in 64-bit arithmetic, the distributions shown in Figs. 3 and 4 only represent a sample of all possible error values. Within each interval, however, the spline fits are well behaved enough that the sampling of errors here should provide a good estimate of the actual total error distribution.

For comparison the best approximating function of e_s over water reviewed by Buck (1981), e_{w4}, has much larger values of Q_{err} making it difficult to display them graphically. Comparing figure 1 from Buck (1981) and limiting the domain from &25° to +35°C, the extreme values of Q_{err} are ~+1500 at &15°C and ~&2500 at +15°C. Values outside this domain can be several times larger. An additional advantage of the 64-bit 1°C interval cubic spline fits is the diminishing error with increasing temperature, giving a better estimate of the total atmospheric moisture content. Even the best approximating functions reviewed by Buck (1981) have errors several orders of magnitude larger in this regard.

While the errors from the 64-bit 1°C interval cubic spline fits are below the observational errors of tens of parts per million (ppm) for (1) and from 16 ppm at 0°C to 3560 ppm at &80°C for (2), microphysical calculations may require even greater accuracy in the approximation functions to better represent local behavior about the original physically based function. For this case the choice of $\frac{1}{16}$°C interval may prove more satisfactory. The 64-bit Q_{err} are too large to display graphically but have a similar pattern to the 1°C interval cubic spline 32-bit Q_{err}. The largest values within each interval range from ~&341000 at &99.65°C to ~&2527 near 0°C to ~&320 near +60°C over water and from ~&396000 at &98.40°C to ~&5204 near 0°C over ice. No positive values have been found so far. With a small loss of performance (<10% on most machines), a large gain in accuracy can be obtained: about 4-6 orders of magnitude. For consistency, the most accurate enhancement factors given in Buck (1981) Eq. (6) and Table 3 should also be used. This will be discussed more below.

4. Sample Computational Times

The main purpose of this implementation is to retain high accuracy over a wide domain while sustaining competitive computational performance. A small sample of calculation times on a few platforms using some of the existing COLA AGCM algorithms as well as the new cubic spline fit is shown in Table I. While unable to outperform the polynomial fit, the cubic spline fit outperforms the exponential-based formula and linear look-up table method on the sample platforms. Note that the linear look-up method as implemented gives the poorest performance on the tested platforms suggesting a new formula would be better suited on these platforms. Each

algorithm was given identical optimization for a given platform and compiler. The shift in the relative performance between the cubic spline fit and the polynomial fit on the different computing environments suggests a dependency on optimization used. Even though many optimization options are available, it has been difficult to improve the relative performance beyond what is shown in table. The ability of the cubic spline fit to outperform the non-polynomial fit algorithms will have to prove satisfactory for now since it exceeds all the algorithms in accuracy and domain coverage.

5. Corrections to the Saturation Vapor Pressure and Calculation of Saturation Specific Humidity

As mentioned in Buck (1981) accurate measurements show that vapor pressure behaves as a non-ideal gas, often expressed as higher order virial coefficients in the equation of state. These are discussed in detail in Hyland (1975). Buck (1981) used the tabular data given in Hyland (1975) to create different correction formulas depending on the degree on accuracy desired. The formulas give an enhancement factor that modifies the saturation vapor pressure value. While these enhancement factors have both pressure and temperature dependencies, he suggested that only detailed microphysical calculations need to consider both sets of dependencies and that most applications can use the pressure dependent corrections only. In this way the pressure dependent enhancement factor can be applied after calculating the temperature-only dependent e_{sat}. This also makes it easier to compute the inverse function. The new COLA AGCM uses the most accurate pressure-only dependent formulas (f_{w3} and f_{i3}) for enhancement factor given by Buck (1981):

$f_w = 1.0007+3.47\times10^{-8}$ P (8)

$f_i = 1.0003+4.18\times10^{-8}$ P (9)

Where f_w is the enhancement factor over water
f_i is the enhancement factor over ice
P is the pressure in Pascals

The constant terms in each factor will lead to some error at lower pressures, but these should add little to total error in atmospheric moisture since values tend to be quite small at low pressure.

The formulas of Wexler (1976 and 1977) are based on pure water. While natural water contains many combinations of isotopic hydrogen and oxygen, their relative concentrations and individual evaporation/condensation rates are temperature dependent and require very detailed microphysics. While their effects are small, they can be of importance in paleoclimate studies. See Bradley (1999) for a detailed discussion. These small effects fall outside the COLA AGCM level of detail.

The reduction of vapor pressure over sea water is not quite as small and can have an important affect on climate simulations through reduced surface evaporation as shown in Sud and Walker (1997). Their formula (Witting, 1908) is based in part on an empirical fit of chlorinity (that includes bromine and iodine as well as chlorine) to salinity over typical sea surface salinity values of ~32-37‰. As water becomes fresher, the relationship breaks down giving a non-zero correction at 0 ‰. In Sverdrup, et al. (1942) a simpler relationship also based on Witting (1908) is given between vapor pressure reduction and salinity (also found in List, 1949):

$$e_{sw} = e_{dw} (1 ! 0.000537 S) \quad (10)$$

Where: e_{sw} is the vapor pressure over sea water
e_{dw} is the vapor pressure over distilled water
S is the sea surface salinity (‰)

Using the Levitus (1982) data, sea surface salinity can interpolated daily or more frequently and

the reductions in saturation vapor pressure computed. Eq. (10) will hold even when the water becomes entirely fresh.

The final saturation vapor pressure $e_{\$}$ is determined by using the appropriate value of e_{sat} (over water or ice) multiplied by the corresponding enhancement factor (f_w or f_i) and the sea surface salinity reduction factor, if applicable, over water. Specifically in the free atmosphere, over ice covered surfaces, or away from water covered surfaces:

$$e_{\$} = e_{sat}(\text{over water}) \cdot f_w \qquad (11)$$

or

$$e_{\$} = e_{sat}(\text{over ice}) \cdot f_i \qquad (12)$$

If over water covered surfaces and $f_S = (1 ! 0.000537\ S)$:

$$e_{\$} = e_{sat}(\text{over water}) \cdot f_w \cdot f_S \qquad (13)$$

With $e_{\$}$ the saturation specific humidity, q_s, can now be computed. A key factor in this calculation is $e = m_w / m_d$ with m_w = molecular weight of water vapor and m_d = molecular weight of dry air. While the molecular weight of natural water vapor does depend on its isotopic components, as mentioned above, this dependency is too complex to treat in the COLA AGCM. The National Center for Atmospheric Research Community Atmosphere Model Version 2.0 (NCAR CAM2.0) constant value of 18.016 is used. The molecular weight of dry air can be treated as a constant unless significant global change scenarios (e.g. $4\times CO_2$) are being investigated, which currently fall outside the COLA AGCM area of investigation. Here the NCAR CAM2.0 constant value of 28.966 is used. Saturation specific humidity is then computed in the usual fashion:

$$q_s = \frac{e \cdot e_{\$}}{p - (1-e) \cdot e_{\$}} \qquad (14)$$

Where $e_{\$}$ is as defined in 11, 12 or 13
p is the atmospheric pressure in Pascals

6. Conclusions

An accurate, large domain coverage, computationally efficient method of calculating saturation vapor pressure and specific humidity has been presented. Based on more recent formulas and including non-ideal gas and sea surface salinity corrections, this method should be satisfactory for most modeling and data analysis applications. With some small changes, even detailed microphysical calculations can be accommodated and reasonable computational efficiency retained. The inverse function and the temperature derivative of e_s can also be computed. The exact level of accuracy and detail can be kept entirely under control of the user.

REFERENCES

Software

Press, W. H., S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, 1992: Numerical recipes in FORTRAN: the art of scientific computing, 2nd ed. Cambridge University Press, 994 pp.

Publications

Bradley, R. S., 1999: *Paleoclimatology*, 2nd ed. Academic Press, 610 pp.

Buck, A. L., 1981: New equations for computing vapor pressure and enhancement factor. *J. Appl. Meteor.*, **20**, 1527-1532.

Goff, J. A. and S. Gratch, 1946: Low-pressure properties of water from -160° to 212°F. *Trans. Amer. Soc. Heat. Vent. Eng.*, **52**, 95-121.

Hyland, R. W., 1975: A correlation for the second interaction virial coefficients and enhancement factors for moist air. *J. Res. Natl. Bur. Stand.*, **79A**, 551-560.

Kuo, H. L., 1965: On the formulation and intensification of tropical cyclones through latent heat release by cumulus convection. *J. Atmos. Sci.*, **22**, 40-63.

Levitus, S., 1982: *Climatological Atlas of World Ocean*, National Oceanic and Atmospheric Administration, 173 pp.

List, R. J., 1949: *Smithsonian Meteorological Tables*, 6th ed. Smithsonian Institution Press, 350 pp.

Moorthi, S. and M. J. Suarez, 1992: Relaxed Arakawa-Schubert: A parameterization of moist convection for general circulation models. *Mon. Wea. Rev.*, **120**, 978-1002.

Sud, Y. C. and G. K. Walker, 1997: Simulation errors associated with the neglect of oceanic salinity in an atmospheric GCM. *Earth Interactions*, **1**, Paper No. 4, 1-19.

Sverdrup, H. U., M. W. Johnson and R. H. Fleming, 1942: *The Oceans, their Physics, Chemistry and General Biology*, Prentice-Hall, 1087 pp.

Wexler, A., 1976: Vapor pressure formulation for water in the range 0° to 100°C—A Revision. *J. Res. Natl. Bur. Stand.*, **80A**, p. 775 ff.

———, 1977: Vapor pressure formulation for ice. *J. Res. Natl. Bur. Stand.*, **81A**, 5-20.

Witting, R., 1908: Untersuchungen zur Kenntnis der Wasserbewegungen und der Wasserumsetzung in den Finnland umgebenden Meeren. *Finländische Hydrogr.-Biologische Untersuchungen*, no. 2, 246 pp. *Cf*. p. 173.

TABLE I.
Time (in Fsec) to compute the saturation vapor pressure, e_s, for all atmospheric temperatures (144×73×17) in the NCEP/NCAR Reanalysis for 00Z 1 January 1990 for 4 different approximate formulas (3 old formulas plus cubic spline replacement) in the COLA/AGCM on 4 different computing platforms in 64-bit arithmetic.

	Polynomial	Exponential	Linear look-up	Cubic spline
SGI Origin 2000	12211	26199	74657	20502
Pentium P4	13357	30830	50751	23654
Itanium	7961	24556	54605	21730
Compaq DS 20	6832	21472	42944	19520