14 November 2014

TO: Ranadu documentation
FROM: Al Cooper
SUBJECT: Skew-T diagram

1 Introduction and goal

The skew-T diagram (formally, the Skew-T log-p diagram) and related thermodynamic diagrams (tephigram, emagram, St?ve diagram) are standard tools for analysis of stability in the atmosphere. There are many versions of this diagram available. A fairly recent and very well documented source is this URL, provided by Bret Whissel. Blank diagrams and also source code are available here. The Storm Prediction Center, National Centers for Environmental Prediction, Norman, OK, uses a sounding analysis system called NSHARP, but it has been updated frequently since its introduction and I haven't yet learned the details of how it is generated. NCL (see this link) provides a skew-T diagram based on the USAF diagram (form dod-wpc 9-16-1) but this dates back to the 1970s and so probably uses the Rossby form for equivalent potential temperature.

There have been two recent developments that have potential influences on the pseudo-adiabats in the skew-T diagram. First, Davies-Jones (Davies-Jones, R., 2009: On formulas for equivalent potential temperature. Mon. Wea. Review, 137, 3137– 3148) developed an improved representation for the pseudoadiabatic equivalent potential temperature, taking into account factors like the variation of the latent heat of vaporization with temperature and making other adjustments. Second, Murphy and Koop (Q. J. R. Meteorol. Soc. (2005), 131, pp. 1539–1565) developed an improved representation of the equilibrium water vapor pressure as a function of temperature. So far as I have been able to find, these advances have not been incorporated into thermodynamic diagrams that are readily available. Furthermore, any definition of pseudo-adiabatic equivalent potential temperature will likely involve approximations because the specific heat and latent heat of vaporization of water varies with temperature, and the former has not been included even in the Davies-Jones formula, which did not use the Murphy-Koop representation of equilibrium water vapor pressure. For that reason, the goal here is to construct new diagrams based on those two advances and compare the results to standard diagrams. For the pseudo-adiabatics, the calculation will be based on direct integration of the differential equation representing constant entropy, as discussed in a 2011 memo referenced from the document ProcessingAlgorithms.pdf.

2 The basic equations and general approach

2.1 The coordinate transformation

The basic coordinates in a skew-T log-p diagram are an ordinate that is based on the base-10 logarithm of the pressure and isotherms that are geometrically at 45° slope with respect to both the

abscissa and ordinate. This leads to an abscissa coordinate that is a function of both temperature and pressure and is expressed as a value in the range 0–1 (the plot limits):

$$x = \frac{T - T_l}{T_h - T_l} - \frac{\log_{10}(p/p_l)}{\log_{10}(p_l - p_h)} \tag{1}$$

where T and p are the respective temperature [°C] and pressure [hPa] and $\{T_l, T_h\}$ and $\{p_l, p_h\}$ are the respective lower and upper limits for temperature along the bottom axis and for pressure. For generating the diagram, this is coded into a function the provide the abscissa for the plot, as follows:

```
# references tBot, tTop, pBot, pTop in global environment!! caution!!
Xplot <- function (.T, .p) {
  return ((.T-tBot) / (tTop-tBot) - log10(.p/pBot) / log10(pBot/pTop))
}
```

2.2 Basic lines: isotherms, isobars, dry adiabats

Plotting the isobars and isotherms is then straightforward. For dry adiabats, solving the equation for potential temperature gives the following equation for the temperature T corresponding to pressure p:

$$T = (T_R)\left(\frac{p}{p_l}\right)^{R_d/c_{pd}} - T_0 \tag{2}$$

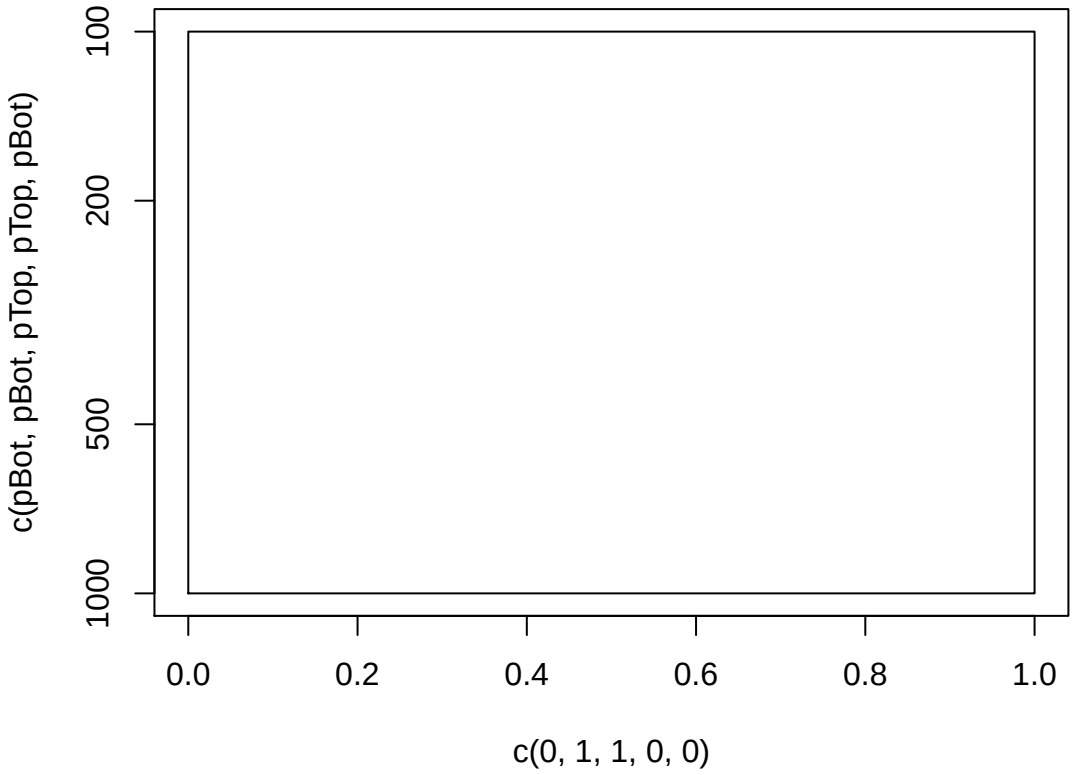
where T_R is the reference temperature for the potential-temperature line (in kelvin) at the lower limit for p (or, conventionally, 1000 hPa, where T_R is also the potential temperature) and T_0 is 273.15 K. In this equation, dry-air values are conventionally used for the gas constant (R_d) and the specific heat at constant pressure (c_{pd}), although this can introduce errors of 1 K or more for moist air. For this reason, it may be helpful to plot a range about the dry adiabats to represent this uncertainty, using as an indication of error the value of T the difference between the above value and that obtained for 100% relative humidity, using the relationship

$$\frac{R}{c_p} = \frac{R_d}{c_{pd}(1 + \frac{1}{5}\frac{e}{p})}'$$

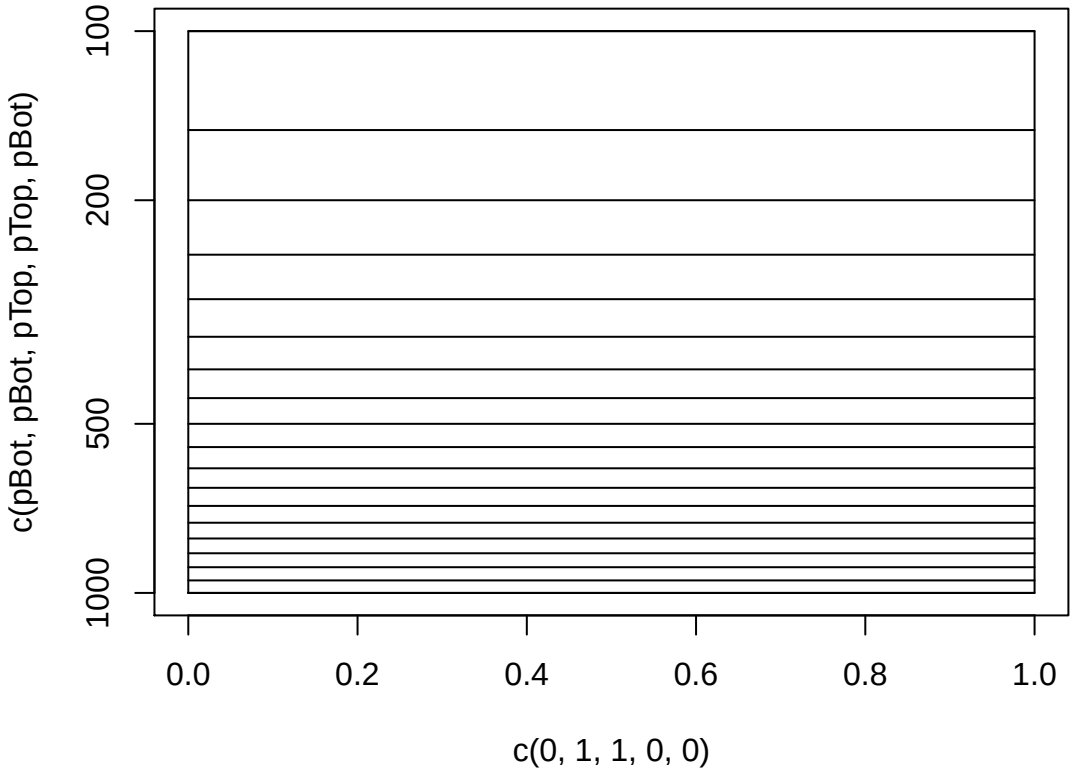
where quantities with subscript d refer to dry air and ε is the ratio of the molecular weight of water to that of dry air (cf. ProcessingAlgorithms.pdf, p. 38).

```
pBot <- 1000
pTop <- 100
tTop <- 40
tBot <- -40
```

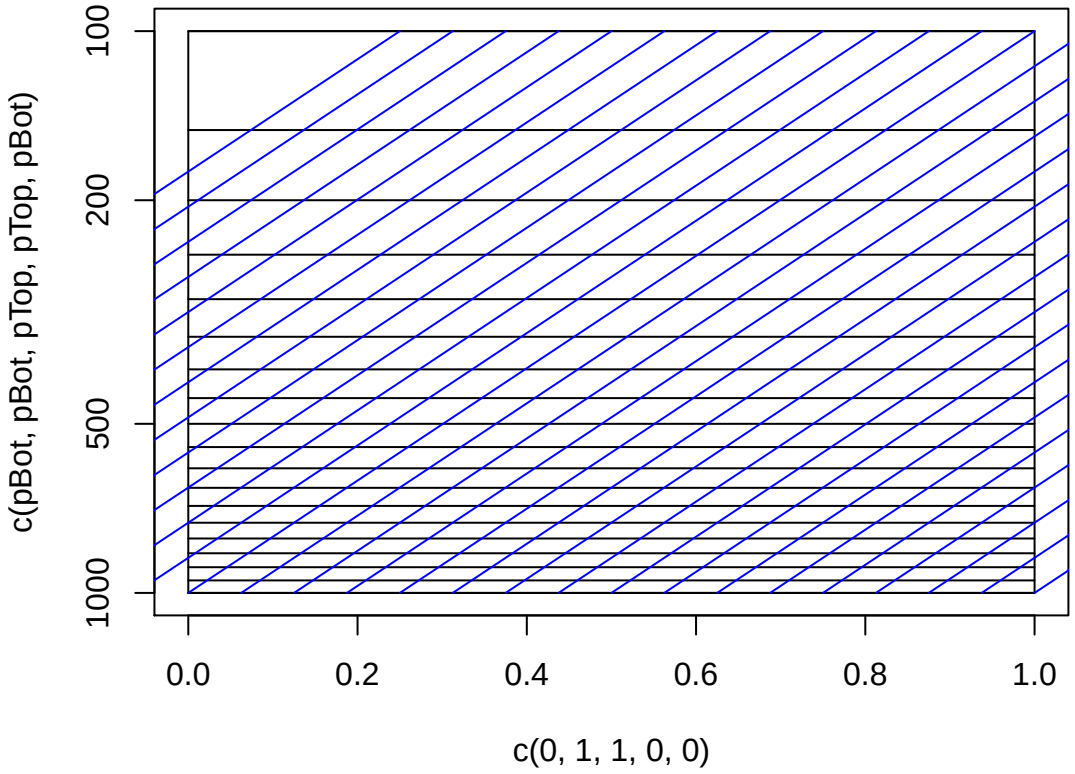
```
tMin <- -100
CP <- SpecificHeats ()
RoverCP <- CP[3] / CP[1]
pLevels <- seq (pBot, pTop, by=-50)
tLevels <- seq (tMin, tTop, by=5)

plot (c (0, 1, 1, 0, 0), c(pBot, pBot, pTop, pTop, pBot), ylim=c(1000,100), log='y', type='l')
```



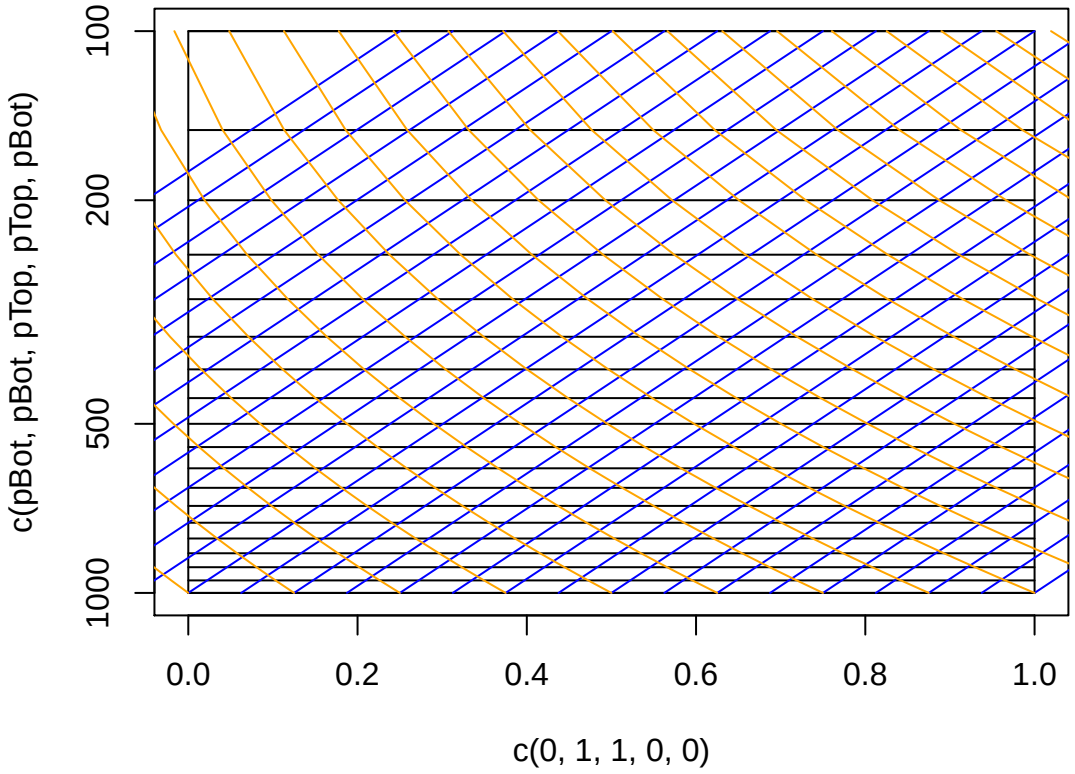

```
for (p in pLevels) {
  lines (c(0., 1.), c(p, p))
}
```




```
for (t in tLevels) {
  x1 <- Xplot (t, pBot)
  y1 <- pBot
  x2 <- x1 + 1
  y2 <- pTop
  lines (c(x1, x2), c(y1, y2), col='blue')
}
```




```
Theta <- seq (TZERO - 100, TZERO + 200, by=10)
for (theta in Theta) {
  x1 <- Xplot(theta / ((1000/pLevels)^RoverCP) - TZERO, pLevels)
  lines (x1, pLevels, col='orange')
}
```



2.3 Mixing ratio

Lines representing constant mixing ratio on a thermodynamic diagram represent the equilibrium value at the specified temperature and pressure, and so are often considered in relationship to measurements of dew point in sounding plots. Expressed in units of grams per kilogram of dry air, the mixing ratio MR for moist air in equilibrium with a plane water surface at temperature T and pressure p is

$$\mathrm{MR} = 1000\,\varepsilon\frac{\mathrm{e_s(T)}}{(\mathrm{p}-\mathrm{e_s(T)})} \tag{3}$$

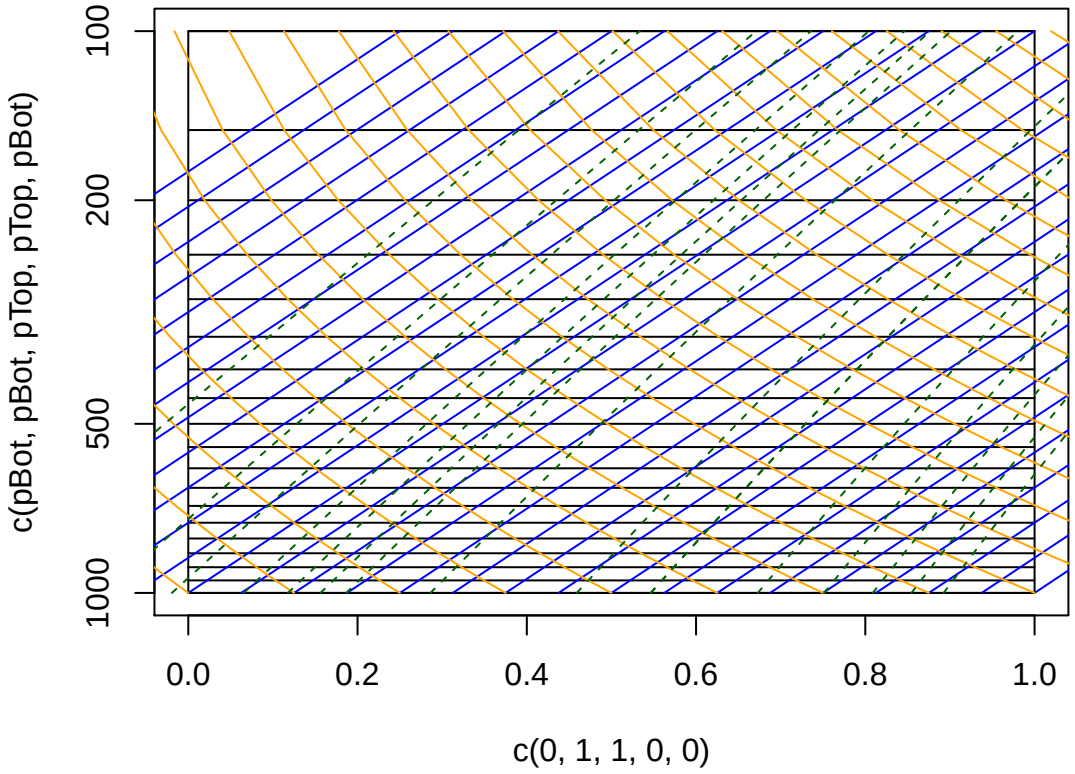
where $e_s(T)$ is the equilibrium water vapor pressure. Given a specified value of MR, the value of T giving that mixing ratio at a specified pressure can be found numerically by solving

$$\mathrm{MR} - 1000\,\varepsilon\frac{\mathrm{e_s(T)}}{(\mathrm{p}-\mathrm{e_s(T)})} = 0$$

numerically, varying T with p amd MR fixed. The R routine 'nleqslv' is used here to find the temperature, which can be used with (1) to find the plotted abscissa coordinate. This is implemented using this function in the call to 'nleqslv':

Using this function with calls to 'nleqslv' for a sequence of pressures gives a sequence of temperatures and hence a sequence of plot points for constructing the diagram.

```
pTop <- 100
tTop <- 40
tBot <- -40
tMin <- -100
CP <- SpecificHeats ()
RoverCP <- CP[3] / CP[1]
pLevels <- seq (pBot, pTop, by=-50)
tLevels <- seq (tMin, tTop, by=5)

plot (c (0, 1, 1, 0, 0), c(pBot, pBot, pTop, pTop, pBot), ylim=c(1000,100), log='y', type='l')
for (p in pLevels) {
  lines (c(0., 1.), c(p, p))
}
for (t in tLevels) {
  x1 <- Xplot (t, pBot)
  y1 <- pBot
  x2 <- x1 + 1
  y2 <- pTop
  lines (c(x1, x2), c(y1, y2), col='blue')
}
Theta <- seq (TZERO - 100, TZERO + 200, by=10)
for (theta in Theta) {
  x1 <- Xplot(theta / ((1000/pLevels)^RoverCP) - TZERO, pLevels)
  lines (x1, pLevels, col='orange')
}
rMix <- c(0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 1., 3., 5, seq (5, 30, by=5)) * 0.001
tt <- pLevels  # overwritten below, just a quick allocation
for (rmix in rMix) {
  for (i in 1:length (pLevels)) {
    tt[i] <- nleqslv::nleqslv (10., TfromRmix, jac=NULL, rmix, pLevels[i])$x
  }
  lines (Xplot (tt, pLevels), pLevels, col='darkgreen', lty=2)
}
```



2.4 Pseudo-adiabatic equivalent potential temperature[1]

The representation of pseudo-adiabats can be done in two ways, either via the formula of Davies-Jones (2009), as presented in ProcessingAlgorithms.pdf, p. 50, or by direct integration of the differential equation obtained by setting the total derivative of entropy with pressure to zero. Previous integrations, discussed in the memo referenced above, demonstrate good agreement between these two approaches, but the direct integration is more basic and is the test against which the Davies-Jones formula was developed so that will be used here.

For integrations that include representations of the temperature dependence of the specific heats and latent heat as well as the Murphy and Koop (2006) representation of equilibrium water vapor pressure, the appropriate differential form of the entropy can be used to find the pseudo-adiabatic (or adiabatic) derivative of temperature with respect to pressure. For an adiabatic process where

[1] Sometimes called, incorrectly, pseudo-equivalent potential temperature (it is pseudo-adiabatic in the sense that all water condensate is removed as it appears) or equivalent potential temperature (a term better understood to refer to the true adiabatic process).

all changes occur in equilibrium, the molar entropy s' is related to temperature T and pressure p (for a perfect gas) via

$$Tds' = c'_p dT - v' dp \tag{4}$$

where primed quantities refer to molar quantities, such that c'_p and $v'$are the molar heat capacity at constant pressure and the molar volume, respectively. There are three contributions to the entropy to consider for a moist air parcel: the entropy of the (i) dry air; (ii) water vapor; and (iii) liquid water. If the respective mole numbers of these three components are n'_d, n'_v, and n'_w, then

$$Tds' = (n'_d c'_{pd} + n'_v c'_{pv} + n'_w c'_w)dT - n'_d v'_d dp_d - n'_v v'_v de + L'_v dn'_v \tag{5}$$

where p_d is the pressure of dry air, e is the water vapor pressure, and L'_v is the molar latent heat of vaporization of liquid water. The last term in (5) arises because there is an entropy change associated with the phase change from liquid water to water vapor, and this last term is the heat released by that phase change.[2] The other terms arise from summing (4) for the three individual components.

The mixing ratios r andr_w are, respectively, the masses of water vapor and liquid water per unit mass of dry air:

$$\begin{aligned} r &= \frac{n'_v M_w}{n'_d M_d} \\ r_w &= \frac{n'_w M_w}{n'_d M_d} \end{aligned} \tag{6}$$

where M_w is the molecular weight of water (mass of water per mole) and M_d that of dry air. Dividing (5) by $n'_d M_d T$ and setting ds' to zero for an isentropic process leads to

$$(c_{pd} + r c_{pv} + r_w c_w)\frac{dT}{T} - \frac{v_d}{T} dp_d - r\frac{v_v}{T} de + \frac{L_v}{T} dr = 0 \tag{7}$$

where unprimed quantities c_{pd}, c_{pv}, c_w, v_d, v_v, and L_v are specific quantities (i.e., per unit mass of dry air for c_{pd} and v_d and per unit mass of water for c_{pv}, c_w, v_v and L_v; e.g., $c_{pd} = c'_{pd}/M_d$ and $c_{pv} = c'_{pv}/M_w$). For perfect gases, $v_d/T = R_d/p_d$ and $v_v/T = R_w/e$ where R_d and R_w are the gas constants for dry air and water vapor, respectively. Also, the ideal-gas form of the Clausius-Clapeyron equation is

$$\frac{de_s}{e_s} = \frac{L_v dT}{R_w T^2} \tag{8}$$

[2] An additional contribution arises from the entropy increase associated with mixing of the water vapor and the dry air, but this is insignificant and will be neglected.

and Kirchhoff's equation (cf., e.g., Emanuel 1994, Eq. 4,4,3) is

$$dL_V = (c_{pv} - c_w)\,dT\ . \tag{9}$$

With $e = e_s(T)$ and $r = r_s(T) = \frac{M_w}{M_d} e_s(T)/p_d$ as corresponds to a saturated parcel, and with some additional transformations as follow, all terms in (7) can be transformed into differential relationships that only involve derivatives of T and p:

$$\frac{L_v dr}{T} = \frac{d(L_v r)}{T} - r\frac{dL_v}{dT}\frac{dT}{T} = \frac{d(L_v r)}{T} - r(c_{pv} - c_w)\frac{dT}{T}$$

$$rR_w \frac{de_s}{e_s} = rR_w \frac{L_v dT}{R_w T^2} = \frac{L_v r}{T}\frac{dT}{T}$$

$$\frac{d(L_v r)}{T} = d\left(\frac{L_v r}{T}\right) + \frac{L_v r}{T}\frac{dT}{T}$$

$$d\left(\frac{L_v r}{T}\right) = \left(\frac{\partial\left(\frac{L_v r}{T}\right)}{\partial T}\right)_{p_d} dT + \left(\frac{\partial\left(\frac{L_v r}{T}\right)}{\partial p_d}\right)_{T} dp_d = \frac{\varepsilon T\, d\left(\frac{L_v e_s(T)}{T}\right)}{p_d dT}\frac{dT}{T} - \frac{L_v r}{T}\frac{dp_d}{p_d}$$

where the transformation to dependence on T is a consequence of assuming that the parcel remains saturated, and where the temperature dependence of L_v and the specific heats is implicit. Gathering terms in (7) after these transformations leads to

$$\left[(c_{pd} + r_t c_w) + \frac{T\varepsilon}{p_d}\left(\frac{\partial\left(\frac{L_v e_s(T)}{T}\right)}{\partial T}\right)_{p_d}\right]\frac{dT}{T} = \left[R_d + \frac{L_v r}{T}\right]\frac{dp_d}{p_d}$$

where $r_t = r + r_w$ is the total water mixing ratio and where $\varepsilon = M_w/M_d$. The result then gives a relationship between T and p_d:

$$\frac{dT}{dp_d} = \frac{TR_d + L_v r}{p_d}\left[(c_{pd} + r_t c_w) + \frac{T\varepsilon}{p_d}\left(\frac{\partial\left(\frac{L_v e_s(T)}{T}\right)}{\partial T}\right)_{p_d}\right]^{-1} \tag{10}$$

The result is a derivative that can be used for numerical integrations that take into account the temperature dependence of the specific heats and the latent heat of vaporization and improved

representation of the equilibrium vapor pressure $e_s(T)$, as in Murphy and Koop (2006). This is also used below to evaluate the accuracy of representations of the equivalent potential temperatures.

Equation (10) is appropriate for the adiabatic process and so can lead to the wet-equivalent potential temperature Θ_q. The similar formula for the pseudo-adiabatic equivalent potential temperature Θ_p can be obtained by neglecting the heat capacity of the liquid water, and so would be the same as (10) but with r_t replaced by r.

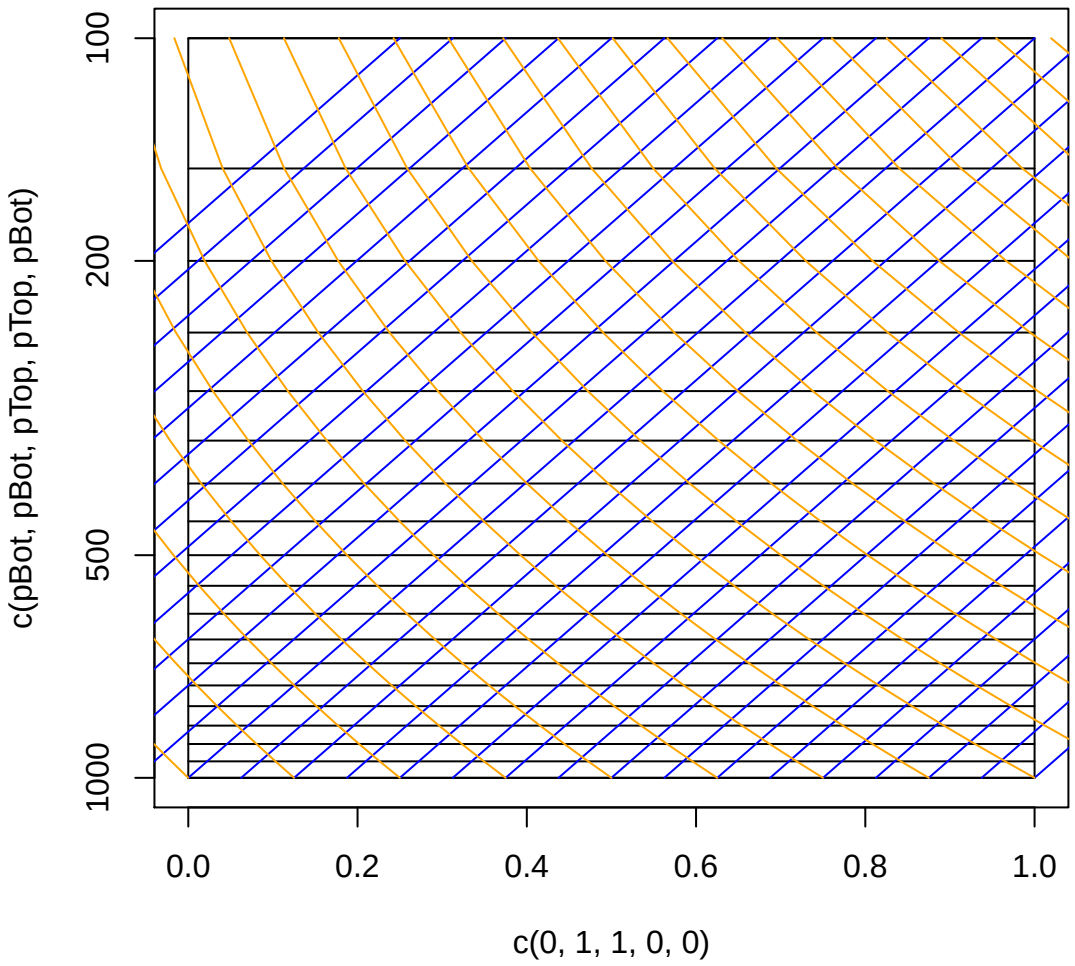
The pseudo-adiabats are then constructed by integration of (10) either upward from a fixed starting point at 1000 hPa (which would represent the wet-bulb pseudo-adiabatic potential temperature) or starting from a pressure where humidity is negligible (which would result in pseudo-adiabats labeled by their value equivalent to potential temperature after all water is condensed). Here the former is chosen. The integration is performed using the following R code:

```
# this chunk includes functions including one to integrate between two levels
# if the specific-heat-of-liquid-water data have not been read, read and store them
if (file.exists ("./SpecificHeatWater.Rdata")) {
  load ("./SpecificHeatWater.Rdata") # reloads cw.table
} else {
  cw.table <- read.table ("./MurphyKoopFig6.txt", sep=',', col.names=c("ID", "T", "cw"))
  cw.table$cw[cw.table$ID == 'n'] <- cw.table$cw[cw.table$ID == 'n'] / StandardConstant("MWW")
  save (cw.table, file="./SpecificHeatWater.Rdata")
}
CWData <- cw.table [ , 2:3]
CWData <- CWData[order(CWData), ]
CWData <- CWData[!is.na(CWData[ ,1]), ]
load ("./CPV.Rdata")

LatentHeatApprox <- function (.T) {  # input in deg.C
  return (2.501e6 - 2370 * .T)
}
CPWaterVapor <- function (.T) {
  tk <- .T + TZERO
  # note that there was an error in this formula in the earlier ThetaE memo, 3 Jan 2011
  return (1000 * LagrangeInterpolate (tk, 4, CPV))
}
SpecificHeatLiquidWater <- function (.T) {
  tk <- .T + TZERO
  return (1000. * LagrangeInterpolate (tk, 4, CWData))
}
# not used; here for reference
LCLfn <- function (.p, RbyCP, thetam, mr) {  # used by LCL function call to nleqslv
  tt <- thetam / (1000/.p)^RbyCP
  ee <- MurphyKoop (tt-TZERO)
  return (ee - mr * .p / (mr + StandardConstant("MWW")/StandardConstant("MWD")))
}

# not used; here for reference; for pseudoadiabat, always at 100% RH
LCL <- function (.RH, .T, .p) {  # .RH in a fraction (not %), .T in deg.C, .p in hPa
  et <- .RH * MurphyKoop (.T)
```

```
  mr <- MixingRatio (et/.p)
  CPM <- SpecificHeats (et/.p)
  RbyCP <- CPM[3] / CPM[1]
  thetam <- .T+TZERO * (1000 / .p)^RbyCP
  pLCL <- nleqslv::nleqslv (.p, LCLfn, jac=NULL, RbyCP, thetam, mr)
  tLCL <- thetam / (1000/pLCL)^RbyCP
  return (data.frame ("pLCL"=pLCL, "tLCL"=tLCL))
}

deriv1 <- function (.T, .LV) { # evaluate last partial deriv. in (10)
  delt <- 0.1
  tc <- .T + delt
  LVp <- .LV + (CPWaterVapor(tc)-SpecificHeatLiquidWater(tc)) * delt
  esp <- MurphyKoop (tc)
  delp <- LVp * esp / (.T + delt + TZERO)
  tc <- .T - delt
  LVm <- .LV - (CPWaterVapor(tc)-SpecificHeatLiquidWater(tc)) * delt
  esm <- MurphyKoop (tc)
  delm <- LVm * esm / (.T - delt + TZERO)
  return ((delp-delm) / (2*delt))
}

dTdpdF <- function (.T, .pd, .LV, .rtot=NA, .aflag=FALSE) {
  e <- MurphyKoop (.T)
  p <- .pd + e
  if (.aflag) {  # adiabatic
    rx <- .rtot
  } else {
    rx <- MixingRatio (e/p)
  }
  tk <- .T + TZERO
  cp_x <- CP[1] + rx * SpecificHeatLiquidWater(.T)
  A <- ((tk*SpecificHeats(0)[3]+.LV*MixingRatio(e/p))/.pd) /
       (cp_x + (EPS * tk / .pd) * deriv1 (.T, .LV))
  # print (sprintf("cp_x=%f, rs=%f. A=%f, for .aflag=%d and p=%f", cp_x, rx, A, .aflag, p))
  return (A)
}

# integration step: call repeatedly to construct plottable profile
# tk Initial temperature, kelvin
# p1 Initial pressure, hPa
# p2  Final pressure, hPa
# value returned, temperature at p2
# aflag  FALSE for pseudoadiabatic, adiabatic otherwise
IntegrationStep <- function (tc, pd1, pd2, rtot, aflag) {
  nsteps <- 2
  delpd <- (pd1 - pd2) / nsteps
  pd <- pd1
  tk <- tc + TZERO
  e <- MurphyKoop (tc)
  p <- pd + e
```

```
  r <- MixingRatio (e/p)

  # ready to calculate the derivative dT/dp_d, adiabatic and pseudo-adiabatic cases
  while (pd > pd2) {
    if ((pd-delpd) < pd2) {delpd <- pd-pd2}
    dTdpd <- dTdpdF (tc, pd, LV, rtot, aflag)
    tch <- tc - dTdpd * delpd/2
    ex <- MurphyKoop (tch)
    LVh <- LV - (CPWaterVapor (tch) - SpecificHeatLiquidWater (tch)) * dTdpd * delpd / 2
    dTdpd <- dTdpdF (tch, pd-delpd/2, LVh, rtot, aflag)
    tc <- tc - dTdpd * delpd  # take full step with derivative evaluated half-step
    pd <- pd - delpd
    LV <- LV + (CPWaterVapor (tch) - SpecificHeatLiquidWater (tch)) * dTdpd * delpd
  }
  return (tc)
}
```

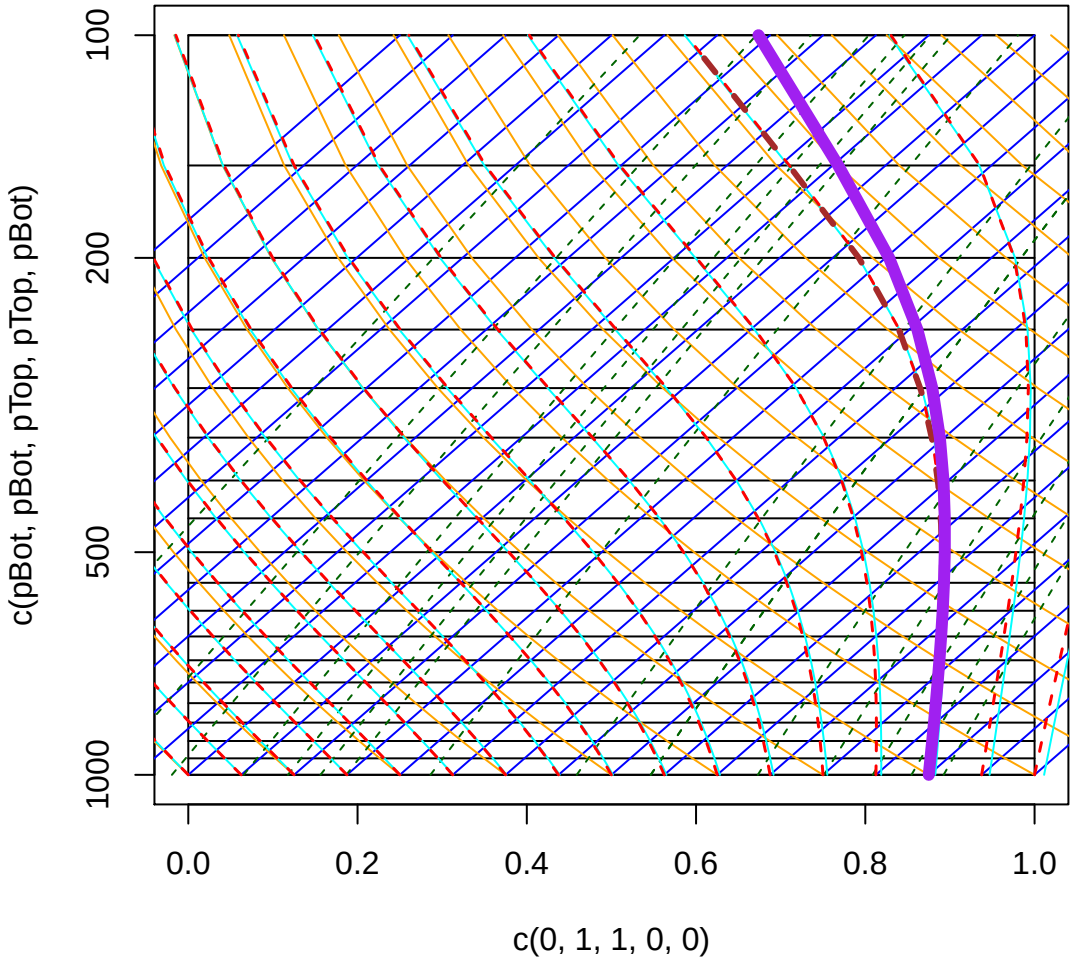
```
pBot <- 1000
tTop <- 40
tBot <- -40
tMin <- -100
CP <- SpecificHeats ()
RoverCP <- CP[3] / CP[1]
pLevels <- seq (pBot, pTop, by=-50)
tLevels <- seq (tMin, tTop, by=5)

plot (c (0, 1, 1, 0, 0), c(pBot, pBot, pTop, pTop, pBot), ylim=c(1000,100), log='y', type='l')
for (p in pLevels) {
  lines (c(0., 1.), c(p, p))
}
for (t in tLevels) {
  x1 <- Xplot (t, pBot)
  y1 <- pBot
  x2 <- x1 + 1
  y2 <- pTop
  lines (c(x1, x2), c(y1, y2), col='blue')
}
Theta <- seq (TZERO - 100, TZERO + 200, by=10)
for (theta in Theta) {
  x1 <- Xplot(theta / ((1000/pLevels)^RoverCP) - TZERO, pLevels)
  lines (x1, pLevels, col='orange')
}
```




```
pBot <- 1000
pTop <- 100
tTop <- 40
tBot <- -40
tMin <- -100
CP <- SpecificHeats ()
RoverCP <- CP[3] / CP[1]
pLevels <- seq (pBot, pTop, by=-50)
tLevels <- seq (tMin, tTop, by=5)

plot (c (0, 1, 1, 0, 0), c(pBot, pBot, pTop, pTop, pBot), ylim=c(1000,100), log='y', type='l')
for (p in pLevels) {
  lines (c(0., 1.), c(p, p))
}
```

```
for (t in tLevels) {
  x1 <- Xplot (t, pBot)
  y1 <- pBot
  x2 <- x1 + 1
  y2 <- pTop
  lines (c(x1, x2), c(y1, y2), col='blue')
}
Theta <- seq (TZERO - 100, TZERO + 200, by=10)
for (theta in Theta) {
  x1 <- Xplot(theta / ((1000/pLevels)^RoverCP) - TZERO, pLevels)
  lines (x1, pLevels, col='orange')
}
rMix <- c(0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 1., 3., 5, seq (5, 30, by=5)) * 0.001
tt <- pLevels  # overwritten below, just a quick allocation
for (rmix in rMix) {
  for (i in 1:length (pLevels)) {
    tt[i] <- nleqslv::nleqslv (10., TfromRmix, jac=NULL, rmix, pLevels[i])$x
  }
  lines (Xplot (tt, pLevels), pLevels, col='darkgreen', lty=2)
}

## first show conventional solution
TWB <- seq (tBot, tTop, by=5)
ThetaE <- EquivalentPotentialTemperature (1000., TWB, MurphyKoop(TWB))


## Warning in if (E <= 0) {:  the condition has length > 1 and only the first element will
be used


TfromEPT <- function (.T, .thetaE, .P) {
  e <- MurphyKoop (.T)
  r <- MixingRatio (e / .P)
  lhv <- 2.501e6 - 2370. * .T    # latent heat of vaporization, temp-dependent
  expn=lhv * r / (CP[1] * (TZERO + .T))
  return (.thetaE - (TZERO+.T)*(1000/(.P-e))^RoverCP * exp(expn))
}
tt <- pLevels  # shortut to define new vector
for (thetaE in ThetaE) {
  for (i in 1:length(pLevels)) {
    tt[i] <- nleqslv::nleqslv (10., TfromEPT, jac=NULL, thetaE, pLevels[i])$x
  }
  lines (Xplot (tt, pLevels), pLevels, col='cyan')
}
# this is solution of Davies-Jones formula; do integration later
DJTfromEPT <- function (.T, .thetaE, .P) {
  L0 <- 2.56313e6
  L1 <- 1754.
  K2 <- 1.137e6
  TK <- .T + TZERO
  e <- MurphyKoop (.T)
  r <- MixingRatio (e/.P)
```

```
  CP <- SpecificHeats(0.)      # need dry-air value, don't need vector
  TL = 2840./(3.5*log(TK)-log(e)-4.805)+55.
  TDL <- TK * (1000./(.P-e))**0.2854*(TK/TL)**(0.28e-3*r)
  THETAP <- TDL * exp (r*(L0-L1*(TL-TZERO)+K2*r)/(CP[1]*TL))
  return (.thetaE - THETAP)
}
for (thetaE in ThetaE) {
  for (i in 1:length(pLevels)) {
    tt[i] <- nleqslv::nleqslv (10., DJTfromEPT, jac=NULL, thetaE, pLevels[i])$x
  }
  lines (Xplot (tt, pLevels), pLevels, col='red', lty=2, lwd=1.5)
}
TQfromWBPT <- function (.T, .ThetaQ, .P, .rtot) {
  e <- MurphyKoop (.T)
  r <- MixingRatio (e / .P)
  lwc <- ifelse ((.rtot > r), 1000 * (.rtot - r) * (100 * (.P-e) / (SpecificHeats(0)[3] * (.T+TZERO))),
  #print (sprintf ("lwc at p/t=%f %f is %f TQ=%f", .P, .T, lwc, WetEquivalentPotentialTemperature (.P,
  return (.ThetaQ - WetEquivalentPotentialTemperature (.P, .T, 0, lwc))
}

# now add lines obtained by integration:
CP <- SpecificHeats(0)
for (twb in TWB[15]) {
  p1 <- pBot - MurphyKoop (twb)
  ptLevels <- pLevels
  ptLevels[1] <- pBot
  rtot <- MixingRatio (MurphyKoop (twb) / pBot)
  LV <- LatentHeatApprox (twb)
  tt[1] <- twb
  for (i in 1:(length(pLevels)-1)) {
    tt[i+1] <- IntegrationStep (tt[i], p1, pLevels[i+1], rtot, aflag=FALSE)
    p1 <- pLevels[i+1]
    tch <- (tt[i+1]+tt[i])/2
    LV <- LV + (CPWaterVapor (tch) - SpecificHeatLiquidWater (tch)) * (tt[i+1]-tt[i])
  }
  ptLevels <- pLevels + MurphyKoop(tt)
  ptLevels[1] <- pBot
  lines (Xplot (tt, ptLevels), ptLevels, col='brown', lty=2, lwd=3)
}

# add a wet-adiabat
ThetaQ <- WetEquivalentPotentialTemperature (1000, TWB[15])
print (sprintf ("ThetaQ line for %f", ThetaQ))

## [1] "ThetaQ line for 373.744935"

rTot <- MixingRatio (MurphyKoop (TWB[15]) / 1000)
for (i in 1:length(pLevels)) {
  tt[i] <- nleqslv::nleqslv (10., TQfromWBPT, jac=NULL, ThetaQ, pLevels[i], rTot)$x
}
lines (Xplot (tt, pLevels), pLevels, col='purple', lty=1, lwd=6)
```



– End of Memo –

Reproducibility:

PROJECT:	SkewT
ARCHIVE PACKAGE:	SkewT.zip
CONTAINS:	attachment list below
PROGRAM:	SkewT.Rnw
ORIGINAL DATA:	/scr/raf_data/DEEPWAVE/rf16.nc
GIT:	git@github.com:WilliamCooper/SkewT.git

Attachments: SkewT.Rnw

SkewT.pdf
SessionInfo

Attachments: ProgramFile
Document.pdf