```
## $Id: parcel.r,v 1.7 2012-10-24 17:35:53 jmuelmen Exp $

## Parcel model that solves the hydrostatic bouyancy and energy conservation equations,
## with the possibility of entrainment of ambient air

## constants
g <- 9.8 ## J/m
L.e <- 2.5e6 ## J/kg
c.p <- 1004 # J/kg/K

## parameters, switches etc
entrainment.rate <- 0

## Evaporation and condensation in the parcel (expressed as a mixing
## ratio)
##
## A possible project is to implement a more realistic model of
## condensation that does not condense all the available water vapor
## instantaneously
##
## T is parcel temperature, w is parcel water vapor mixing ratio
parcel.condensation <- function(T, w, wL, p, W) {
  w.s <- w.sat(T, p)
  if (is.nan(w.s) || is.infinite(w.s)) {
    print(paste("saturation vapor pressure is NaN at ", T, w, p))
    return(0)
  }
  if (W > 0.1) {
    if (w >= w.s)
      ## if water vapor mixing ratio is at or above saturation,
      ## condensation will occur
      dw.sat.dT(T, p)
    else
      ## otherwise, no evaporation or condensation occurs
      0
  } else if (W < -0.1) {
    if (wL > 0 && w <= w.s)
      ## if water vapor mixing ratio is at or below saturation and
      ## liquid water is available, evaporation will occur
      dw.sat.dT(T, p)
    else 0
  }
  else 0
}

## Evaporation and condensation in the entrained air
##
## One of the possible (advanced) projects is to implement entrainment
## of moist or dry ambient air.  The default code does not implement entrainment.
entrain.condensation <- function() {
  0
}

## Heating and cooling of the parcel during the ascent
##
##
parcel.Delta.T <- function(Delta.z, T, w, wL, p, T., W) {
  Delta.T <- -Delta.z * 1e3 * ## (to convert z in km to m)
    (g + entrainment.rate * (L.e * entrain.condensation() + c.p * (T - T.))) /
      (c.p + L.e * parcel.condensation(T, w, wL, p, W))
}

## Buoyant force on the parcel
##
## T is the temperature of the parcel, T. is the density of the
## ambient air, wL is the mixing ratio of liquid water in the parcel
buoyancy <- function(T, T., wL) {
  g * ((T - T.) / T. - wL)
}

## Saturation water vapor mixing ratio over liquid water
w.sat <- function(T, p) {
  T.C <- T - 273.15
  e.sat <- 6.107799961 + T.C*(4.436518521e-1 + T.C*(1.428945805e-2 + T.C*(2.650648471e-4 + T.C*(3
.031240396e-6 + T.C*(2.034080948e-8 + 6.136820929e-11*T.C))))) ## polynomial approximation taken
from Appendix 4, Pruppacher & Klett, 2nd ed; valid for -50 deg C < T < 50 deg C
  epsilon <- 0.622 # M_a / M_a
  return(epsilon * e.sat / (p - e.sat))
}
```

Vary this value for entrainment studies

To model slower-than-instantaneous condensation, change the return value to a fraction < 1 of dwsat/dT

Implement condensation/evaporation similar to the function above for entrainment studies

Add a drag term if you are implementing entrainment (see P&K eq. (12.25))

```
## Derivative of saturation water vapor mixing ratio with respect to temperature
dw.sat.dT <- function(T, p) {
  T.C <- T - 273.16
  de.sat.dT <- 4.436518521e-1 + T.C*(1.428945805e-2 + T.C*(2.650648471e-4 + T.C*(3.031240396e-6 +
 T.C*(2.034080948e-8 + 6.136820929e-11*T.C)))) ## polynomial approximation taken from Appendix 4,
 Pruppacher & Klett, 2nd ed; valid for -50 deg C < T < 50 deg C
  epsilon <- 0.622 # M_w / M_a
  return(epsilon * de.sat.dT / (p - de.sat.dT))
}

## Vertical profiles of pressure, temperature and humidity.  Many
## project possibilities here, such as:
##
## * ascent of a parcel in a stable/unstable/conditionally stable atmosphere
## * effect of condensation on the vertical development of convection
## * effect of a temperature/humidity inversion
##
## US standard atmospheric temperature profile with a surface
## temperature of 15 deg C and a lapse rate of 6.5 K/km (to 11 km
## height)
T.US.standard.atmosphere <- function(z) {
  pmax(288.15 - 6.5 * z, 0)
}
## Simple exponential atmospheric pressure profile
p.US.standard.atmosphere <- function(z) {
  1013 * exp(-z / 8.3)
}
## Dummy profile of humidity (80% relative humidity throughout)
w.default <- function(z) {
  0.8 * w.sat(T.US.standard.atmosphere(z), p.US.standard.atmosphere(z))
}
T.with.inversion <- function(z) {
  ## this atmosphere has an inversion at 1000 m, above which the
  ## temperature is a constant 15 deg C
  sapply(z, function(z) {
    if (z < 1)
      pmax(288.15 - 9.8 * z, 0)
    else 288.15
  })
}

## Run the parcel model
##
## T.prof is the ambient temperature profile; supply a function that
## takes height as an argument; default is the US standard atmosphere
## with 15 deg C at sea level and a lapse rate of 6.9 K/km.
##
## w.prof is the ambient humidity profile; default is 80% RH throughout.
##
## T.parcel and w.parcel are the starting temperature and humidity of the parcel
##
## t.end is the simulation end time in seconds; Delta.t is the time step
parcel.model <- function(T.prof = T.US.standard.atmosphere, w.prof = w.default,
                         p.prof = p.US.standard.atmosphere,
                         T.parcel, w.parcel,
                         t.end, Delta.t) {
  ## These are the variables we need to keep track of during the ascent
  t <- 0 ## time (s)
  z <- 0 ## height (km)
  T <- T.parcel ## parcel temp (K)
  w <- w.parcel ## parcel water vapor mixing ratio (kg/kg)
  wL <- 0 ## parcel liquid water mixing ratio (kg/kg)
  W <- 0 ## parcel vertical velocity (m/s)

  ## Store model output for later plotting and analysis
  time.series <- data.frame(t = t, z = z, T = T, w = w, wL = wL, W = W)

  ## Iterate the model
  while (t <= t.end) {
##    w <- 0
##    wL <- 0
    T. <- T.prof(z) ## ambient temp
    ## w. <- w.prof(z) ## ambient humidity -- only needed if entrainment is desired
    p <- p.prof(z) ## parcel and ambient pressure are equal
    W <- W + buoyancy(T, T., wL) * Delta.t ## buoyancy accelerates the parcel vertically
    Delta.z <- W * Delta.t * 1e-3 ## all heights in km
    z <- z + Delta.z
    Delta.T <- parcel.Delta.T(Delta.z = Delta.z, T = T, w = w, wL = wL, p = p, T. = T., W = W) ##
```

Vary the ambient lapse rate (stable/ conditionally stable/unstable)

```
 adiabatic (unless entrainment is added) cooling
                                                        ## (due to ascent) and heating (due to conden
sation)
    T <- T + Delta.T
    Delta.wL <- parcel.condensation(T, w, wL, p, W) * Delta.T
    wL <- wL - Delta.wL  ## condensation increases liquid water mixing ratio
    w <- w + Delta.wL    ## and decreases water vapor mixing ratio
    ## Store model output for later plotting and analysis
    time.series <- rbind(time.series,
                         data.frame(t = t, z = z, T = T, w = w, wL = wL, W = W))
    t <- t + Delta.t
    if (t %% 1000 == 0) {
      print(t)
    }
  }

  ## return the output for plotting
  return(time.series)
}

## Function to plot the model output and save it to a PDF file
plot.results <- function(results, file.name = "output.pdf",
                         T.prof = T.US.standard.atmosphere, w.prof = w.default,
                         p.prof = p.US.standard.atmosphere) {
  pdf(file.name)

  par(cex = 1) ## increase default font size to something legible

  ## plot vertical profile of parcel and ambient temperature
  with(results, {
    z.prof <- seq(0, 2, by = 0.1)
    plot(T.prof(z.prof), z.prof, col = "blue", type = "l",
         xlab = "T (K)", ylab = "z (km)", xlim = c(275, 290))
    lines(T, z)
    legend("topright",
       c("Ambient T", "Parcel T"),
       col = c("blue", "black"), lty = c("solid", "solid"))
  })

  ## plot ambient and parcel lapse rate
  par(cex = 1)
  with(results, {
    par(mfrow = c(1,2))
    plot(-filter(diff(T) / diff(z), rep(0.01, 100)), z[-1], type = "l",
         xlab = "-dT/dz (K/km)", ylab = "z (km)")
    plot(-filter(diff(T.US.standard.atmosphere(z)) / diff(z), rep(0.01, 100)), z[-1],
         type = "l", col = "blue", xlab = "-dT/dz (K/km)", ylab = "z (km)")
    par(mfrow = c(1,1))
  })

  ## plot vertical profile of parcel and ambient water vapor and liquid water mixing ratios
  par(cex = 1)
  with(results, {
    z.prof <- seq(0, 2, by = 0.1)
    plot(w.sat(T.prof(z.prof), p.prof(z.prof)), z.prof, col = "blue", type = "l",
         xlab = "w (kg/kg)", ylab = "z (km)", xlim = c(0, 0.012))
    lines(w.sat(T, p.prof(z)), z, col = "blue", lty = "dashed")
    lines(w, z)
    lines(wL, z, lty = "dashed")
    legend("topright",
       c(expression(paste("Ambient ", w[s])), expression(paste("Parcel ", w[s])),
         "Parcel w", expression(paste("Parcel ", w[L]))),
       col = c("blue", "blue", "black", "black"), lty = c("solid", "dashed"))
  })

  ## make an aerological diagram with p^(R/cp) on the height axis and
  ## temperature on the abscissa
  ## y.transform <- function(p) -p^(2/7)
  ## with(results, {
  ##   z.prof <- seq(0, 12, by = 0.1)
  ##   plot(T.prof(z.prof), y.transform(p.prof(z.prof)), col = "blue", type = "l",
  ##        xlab = "T (K)", ylab = "p", axes = FALSE, xlim = c(200, 290))
  ##   lines(T, y.transform(p.prof(z)))
  ##   legend("topright",
  ##      c("Ambient T", "Parcel T"),
  ##      col = c("blue", "black"), lty = c("solid", "solid"))
  ##   p.points <- c(1000, 950, 900, 850, 800, 700, 600, 500, 400, 200)
  ##   axis(1)
  ##   axis(2, labels = p.points, at = y.transform(p.points))
```

```
  ##   ## constant potential temperature lines
  ##   for (theta in c(300, 295, 290, 285, 280, 275, 250, 220)) {
  ##     ## lines(c(theta, 275), c(-(1000)^(2/7), -(1000)^(2/7) * 275 / theta), lty = "dashed")
  ##     lines(c(theta, 200), c(-(1000)^(2/7), -(1000)^(2/7) * 200 / theta), lty = "dashed")
  ##     text(theta, -(1000)^(2/7), as.expression(paste("theta = ", theta)))
  ##   }
  ##   box()
  ## })

  dev.off()
}

## Uncomment the model configuration you are interested in

## ## dry parcel:
## results <- parcel.model(T.prof, w.prof, p.prof, T.parcel = 289.15, w.parcel = 0.0, t.end = 500
, Delta.t = 0.1)
## plot.results(results)

## ## moist parcel, slightly below saturation at the surface:
results <- parcel.model(T.parcel = 289.15, w.parcel = 0.01, t.end = 600, Delta.t = 0.1)
plot.results(results)

## ## moist parcel, in an atmosphere with an inversion
## results <- parcel.model(T.prof = T.with.inversion, T.parcel = 289.15, w.parcel = 0.01, t.end =
 1000, Delta.t = 0.1)
## plot.results(results, T.prof = T.with.inversion)

## Save the model output to a text file suitable for later analysis
write.csv(results, file = "output.txt")
```

Vary w.parcel to simulate different initial parcel humidities

Uncomment this paragraph to simulate an atmosphere with an inversion (and see the T.with.inversion function above)