```
# install.packages("magrittr") # package installations are only needed the first time you use it
# install.packages("dplyr")    # alternative installation of the %>%
library(magrittr) # needs to be run every time you start R and want to use %>%
library(dplyr)    # alternatively, this also loads %>%
# Part 6 - PPM scheme advection

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
library(tidyverse)
```

```{r}
# Create the grid and initial conditions
imax = 1000             # number of grid points in x-direction
delx = 100.             # horizontal grid spacing (m)
delt = 10.              # time increment (s)
u = 5.                  # horizontal wind speed (m/s)
```

```{r}
# Create initial concentration anomaly distribution in the x-direction
conc <- rep(0.0, imax)   # initial concentration of background is zero
cmax = 10.0                      # max initial concentration
conc[100:150] <- seq(0., cmax, len = 51)        # insert left side of triangle
conc[150:200] <- seq(cmax, 0., len = 51)        # insert right side of triangle
conc[20:40] <- seq(0., -0.5*cmax, len = 21)    # insert left side of triangle
conc[40:60] <- seq(-0.5*cmax, 0., len = 21)    # insert right side of triangle
conc_orig <- conc
```



```{r}
# create ideal solution
cideal <- rep(0.0, imax)   # initial concentration of ideal background is zero
cideal[800:850] <- seq(0., cmax, len = 51)    # insert left side of triangle
cideal[850:900] <- seq(cmax, 0., len = 51)    # insert right side of triangle
cideal[720:740] <- seq(0., -0.5*cmax, len = 21)    # insert left side of triangle
cideal[740:760] <- seq(-0.5*cmax, 0., len = 21)    # insert right side of triangle
```



```{r}
nsteps = (imax - 300) / (u * delt / delx)
xvals = seq(1,1000)
```


Plot 1
This has only the original concentration and the ideal solution

```

```{r}
plot(xvals, cideal, col = 'red', type = "l")
lines(xvals, conc, col = 'blue')
legend(400, 10, legend=c("cideal", "conc orig"),
       col=c("red", "blue"), lty=1, cex=0.8)
# plot <- ggplot() +
#   geom_line(aes(x = xvals, y = conc), color ="blue") +
#   geom_line(aes(x = xvals, y = cideal), color ="red") +
#   theme_bw() +
#   xlab("grid index (i)") +
#   ylab("quantity")+
#   ggtitle("PPM plot")
```
PPM scheme code

Here I use the code that was provided for the homework.

```{r}
#### USING THE CODE PROVIDED
# ================================
# 6) Use the HPPM method from CMAQ
# CW refers to the paper by Colella and Woodward.
# 1-D domain covers grid points i = 1 to imax.  But 1 and imax are boundary-
# condition cells.  The main interior computation is for i = 2 to (imax-1).
# Pre-calculate some constants
sixth = 1.0/6.0
two3rds = 2.0/3.0
oneoverdelx = 1.0 / delx
# Allocate the vectors
dc = numeric(imax)       # nominal difference in concentration across a cell
clfirst = numeric(imax)  # first guess of conc at left edge of cell i
cr = numeric(imax)       # conc at right edge of cell i
cl = numeric(imax)       # conc at left edge of cell i
c6 = numeric(imax)       # this corresponds to parabola parameter a6 of CW
eq.(1.4)
FL = numeric(imax)       # pollutant flux into the left side of a grid cell
FR = numeric(imax)       # pollutant flux into the right side of a grid cell
# Iterate forward in time
for (n in 1:nsteps) {                  # for each time step n


    # To guarantee that solution is monotonic, check that the left edge of
cell i
    #    (which is between cells i and i-1) should not have a concentration
lower
    #    or higher than the concentrations in those two neighboring cells
    #    Namely, is clfirst between c[i] and c[i-1].  If not, then fix.
    for (i in 2:(imax - 1)) {          # for each interior grid point i
        del_cl = conc[i] - conc[i-1]    # concentration difference with cell
at left
        del_cr = conc[i+1] - conc[i]    # concentration difference with cell
at right
        dc[i] = 0.5*(del_cl + del_cr)   # 1st guess of avg conc difference
across cell i
        if ((del_cl*del_cr)>0.0) {      # then revise average difference
across cell i
            dc[i] = sign(dc[i]) * min( abs(dc[i]) , 2*abs(del_cl) ,
2*abs(del_cr) )
        } else {dc[i]=0.0}             # for the special case of constant
conc across cell
```

```
104     }                                   # end of grid-point (i) loop
105 
106     # First guess for concentration at left edge of each cell, using revised
    dc value
107     for (i in 2:(imax - 1)) {           # for each interior grid point i
108         clfirst[i] = 0.5*(conc[i]+conc[i-1]) - sixth*(dc[i]-dc[i-1])
109     }                                   # end of grid-point (i) loop
110 
111     # find parameters for the piecewise-continuous parabola in cell i
112     for (i in 2:(imax - 1)) {           # for each interior grid point i
113 
114         # conc at the right edge (cr) of cell i equals concen at left edge of
    cell i+1
115         cr[i] = clfirst[i+1]            # concentration at right edge of cell
    i
116         cl[i] = clfirst[i]              # concentration at left edge of cell
    i
117 
118         # Check whether cell i is an extremum (is a peak or valley in the
    conc plot)
119         if (( (cr[i]-conc[i]) * (conc[i] - cl[i]) )   > 0.0) {   # then not
    extremum
120 
121             # Find the two coefficients of the parabola: dc and c6:
122             dc[i] = cr[i] - cl[i]       # updated concen diff. between right
    and left edges
123             c6[i] = 6*( conc[i] - 0.5*(cl[i]+cr[i]) )
124 
125             if ( (dc[i]*c6[i]) > (dc[i]*dc[i]) ) {   # then adjust for
    overshoot at left edge
126                 cl[i] = 3.0*conc[i] - 2.0*cr[i]
127             } else if ((-dc[i]*dc[i]) > (dc[i]*c6[i])) {  # then adjust for
    overshoot at right
128                 cr[i] = 3.0*conc[i] - 2.0*cl[i]
129             }                           # end of block of "not extremum"
    calculations
130 
131         } else {                        # For an extremum, don't use a
    parabola.
132             cl[i] = conc[i]             # Instead, assume concen is constant
    across the cell,
133             cr[i] = cl[i]               # Thus, left and right concentrations
    equal average conc.
134         }                               # end of grid-point (i) loop
135 
136         # second guess of coefficients for the parabola, from CW eq. (1.5)
137         dc[i] = cr[i] - cl[i]
138         c6[i] = 6.0*(conc[i] - 0.5*(cl[i] + cr[i]))
139 
140     }                                   # end of grid-point (i) loop
141 
142 
143     # Initialize to 0 the fluxes into the left and right sides of cell i
144     FL <- rep(0.0, imax)
145     FR <- rep(0.0, imax)
146 
147 
148     # Next, use parabolic fits within each cell to calculate the fluxes
    betweeen cells
149 
```

```
150     # At left side of whole domain (i = 1), assume constant flux. Use FR[1] =
        FR[2]
151     if (u > 0.0) {                      # if wind enters left boundary of
        domain
152         y = u*delt                      # distance traversed by wind during
        delt
153         x = y*oneoverdelx               # Courant number is fraction of grid
        cell traversed
154         # Find the flux leaving the right side of left boundary cell
155         FR[1] = y*( cr[2] - 0.5*x*(dc[2] - c6[2]*(1.0 - two3rds*x))  )    #
        parabolic in x
156     }
157
158     # In interior of whole domain, use parabola eqs. CW (1.12) to find the
        fluxes
159     for (i in 2:(imax-1)) {             # for each interior grid point i
160
161         if (u < 0.0) {                  # for wind from right to left
162             y = -u*delt                 # distance traversed by wind during
        delt
163             x = y*oneoverdelx           # Courant number is fraction of grid
        cell traversed
164             FL[i] = y*( cl[i] + 0.5*x*(dc[i] + c6[i]*(1.0 - two3rds*x))  )
        # parabolic in x
165         }
166
167         if (u > 0.0) {                  # for wind from left to right
168             y = u*delt                  # distance traversed by wind during
        delt
169             x = y*oneoverdelx           # Courant number is fraction of grid
        cell traversed
170             FR[i] = y*( cr[i] - 0.5*x*(dc[i] - c6[i]*(1.0 - two3rds*x))  )
        # parabolic in x
171         }
172
173     }                                   # end of loop over all interior grid
        cells
174
175     # At right side of whole domain (i = imax), assume const. flux. Use
        FL[imax] = FL[imax-1]
176     if (u < 0.0) {                      # if wind enters right boundary of
        domain
177         y = -u*delt                     # distance traversed by wind during
        delt
178         x = y*oneoverdelx               # Courant number is fraction of grid
        cell traversed
179         FL[imax] = y*( cl[imax-1] + 0.5*x*(dc[imax-1] + c6[imax-1]*(1.0 -
        two3rds*x))  )
180     }
181
182
183     # For a realistic case, you would want to impose the actual fluxes at the
        boundaries.
184     # But for our simple HW, impose boundry conditions of zero pollutant flux
        entering the domain.
185     if (u > 0.0) FR[1] = 0.0
186     if (u < 0.0) FL[1] = 0.0
187
188
```

```
    # Update the concentrations in each grid cell.  *** This is the forecast equation.***

    for (i in 2:(imax-1)) {             # for each interior grid point i
        conc[i] <- conc[i] + oneoverdelx* (FR[i-1] - FR[i] + FL[i+1] - FL[i])  # CW eq. 1.13
    }                                   # end of loop over all interior grid cells i

}                                       # end of loop over all time iterations n
```

Make PPM plot

```{r}
df <- data.frame("Initial Concentration" = conc_orig,
                 "Final Ideal" = cideal,
                 "PPM" = conc,
                 "Grid Index" = xvals)
```

```{r}
df %>% ggplot(aes(x = Grid.Index)) +
  geom_line(aes(y = Initial.Concentration, color = "Initial")) +
  geom_line(aes(y = Final.Ideal, color ="Final")) +
  geom_line(aes(y = PPM, color ="PPM")) +
  theme_bw() +
  xlab("Grid Index (i)") +
  ylab("Quantity")+
  scale_color_manual(values = c("Initial" = 'blue',
                                "Final" = "red",
                                "PPM" = "green")) +
  ggtitle("Piecewise Parabolic Method  CR: 0.5")
```