The movement of compounds in the environment is driven by two processes, advection and diffusion. Of course, these processes occur in three dimensions, but for this class we'll begin with one dimensional processes before getting to more complecated examples.

1 Session Outcomes

1. Describe Advection Mathematically
2. Analyze 1-dimensional movement using advection equations
3. Describe Diffuions mathematically
4. Analyze 1-dimensional movement using Fick's law.
5. Two dimensional analysis of advention

2 Material Transport

3 Theory of Pollution Movement

3.1 Diffusion versus Advection

4 R as a Calculator

5 One Dimensional Diffusion

5.1 Steady-state solution of 2-D PDEs

within `rootSolve` Package, has a function, `steady.2D()` that can efficiently fine the steady-state of 2-dimensional problems.

(1)

a substance C is consumed at a quadratic rate (r dot C2), while dispersing in X- and Y-direction. At certain positions (x,y) the substance is produced (rate p).

The model is solved on a square (100*100) grid. There are zero-flux boundary conditions at the 4 boundaries.

The term Dx . . . ,

i.e. it is the negative of the ux gradient, where the ux is due to diffusion. In the numerical approximation fo the ux, the concentration gradient is approximated as the subtraction of two matrices, with the columns or rows shifted (e.g. Conc[2:n,]-Conc[1:(n-1),]).

The flux gradient is then also approximated by subtracting entire matrices (e.g. Flux[2:(n+1),]-Flux[1:(n),]). This is very fast. The zero-flux at the boundaries is imposed by binding a column or row with 0-s.

```
> #library(rootSolve)
> diffusion2D <- function(t,conc,par){
+ Conc <- matrix(nr=n,nc=n,data=conc) # vector to 2-D matrix
+ dConc <- -r*Conc*Conc # consumption
+ BND <- rep(1,n) # boundary concentration
+
+ # constant production in certain cells
+ dConc[ii]<- dConc[ii]+ p
+
+ #diffusion in X-direction; boundaries=imposed concentration
+
+ Flux <- -Dx * rbind(rep(0,n),(Conc[2:n,]-Conc[1:(n-1),]),rep(0,n))/dx
+ dConc <- dConc - (Flux[2:(n+1),]-Flux[1:n,])/dx
+
+ #diffusion in Y-direction
+ Flux <- -Dy * cbind(rep(0,n),(Conc[,2:n]-Conc[,1:(n-1)]),rep(0,n))/dy
+ dConc <- dConc - (Flux[,2:(n+1)]-Flux[,1:n])/dy
+
+ return(list(as.vector(dConc)))
+ }
```

After specifying the values of the parameters, 10 cells on the 2-D grid where there will be substance produced are randomly selected (ii).

Figure 5: Steady-state solution of the nonlinear 2-Dimensional model

```
> # parameters
> dy <- dx <- 1 # grid size
> Dy <- Dx <- 1.5 # diffusion coeff, X- and Y-direction
> r <- 0.01 # 2-nd-order consumption rate (/time)
> p <- 20 # 0-th order production rate (CONC/t)
> n <- 100
> # 10 random cells where substance is produced at rate p
> ii <- trunc(cbind(runif(10)*n+1,runif(10)*n+1))
>
```

The steady-state is found using function steady.2D. It takes as arguments a.o. the dimensionality of the problem (dimens) and lrw=1000000, the length of the work array needed by the solver. If this value is set too small, the solver will return with the size needed. It takes about 0.5 second to solve this 10000 state variable model.

```
> Conc0 <- matrix(nr=n,nc=n,10.)
> # print(system.time(
> # not working yet...
>
> #ST3 <- steady.2D(Conc0,func=diffusion2D,parms=NULL,pos=TRUE,dimens=c(n,n), lrw=1000000,at
>
```

The S3 image method is used to generate the steady-state plot.

```
> #image(ST3,main="2-D diffusion+production", xlab="x", ylab="y")
```