第四章のおまけ集

ここに書いてあることは僕が勝手に作ったものですので、理論・コードなどが間違っている可能性があります。間違いを発見された方は、その旨、柴田泰宙 d10tf005@ynu.ac.jp(@を@に変えてください)までお知らせいただけると、ありがたいです。責任を持って修正させていただきます。ただし、使用後の結果の責任までは負いかねますので、ご容赦ください。

Appendix 1

p125 の解法(特性方程式を使う)

$$0 = D_b \frac{\partial^2 C}{\partial x^2} - \lambda C$$

ここで、解が

$$C = Ae^{ax}$$

の形で書けるとする。これを上式に代入して、

$$D_b \frac{\partial^2}{\partial x^2} Ae^{ax} - \lambda Ae^{ax} = 0$$

第一項だけが微分の形になっているので、微分して

$$D_b Aa^2 e^{ax} - \lambda Ae^{ax} = 0$$

まとめて、

$$Ae^{ax}\left(a^2 - \frac{\lambda}{D_b}\right) = 0$$

すなわち、

$$\left(a^2-\frac{\lambda}{D_b}\right)=0$$

なので、これをaについて解くと、

$$a=\pm\sqrt{\frac{\lambda}{D_b}}$$

C の式に代入すると、

$$C=Ae^{\sqrt{\frac{\lambda}{D_b}}x},\ C=Ae^{-\sqrt{\frac{\lambda}{D_b}}x}$$

ここで、$\frac{\partial C}{\partial x}\Big|_{x=\infty}=0$ という条件から、

$$C=Ae^{-\sqrt{\frac{\lambda}{D_b}}x}$$

が解となる。また$x=0$で$C(x)=C_0$より、

$$C=C_0e^{-\sqrt{\frac{\lambda}{D_b}}x}$$

となる。左辺が x の関数であることを明示的に書くと、

$$C(x)=C_0e^{-\sqrt{\frac{\lambda}{D_b}}x}$$

となり、p126 と同じ式が導けた。

$$\log(C(x))=a+bx$$

Appendix2(Fig. 4. 5 が気持ち悪いので色々変更)

```
###ワーキングディレクトリの変更。ここに pdf ファイルを生成する###
setwd("C:/")

###pdf の名前###
pdf("Fig.4.5.pdf")
x <- 0.5:9.5
y <- c(3.9, 1.7, 1.1, 0.5, 0.3, 0.2, 0.1, 0.05, 0.03, 0.02)

###教科書は x と y が逆で非常に気持ち悪い###
plot(x, y, pch=16, xlab="x as depth,cm", ylab="y as dpm/cm3",
      main="Pb210")

###ラムダは既知###
lam <- 0.031

###y だけ対数変換###
LL <- lm(log(y) ~ x)
###p126 の式になるように切片と傾きを解釈###
C0 <- exp(coef(LL)[1])
Db <- lam/((coef(LL)[2])^2)

xx <- seq(0, 10, 0.1)
lines(xx, C0*exp(-sqrt(lam/Db)*xx))
###もちろんこれでも一緒の図になる###
points(xx, exp(coef(LL)[1] + coef(LL)[2]*xx), col="red", lwd=2, type="l")

fit <- nls(y ~ C0*exp(-sqrt(lam/Db)*x),
            start = c(C0=5, Db=0.1))

C02 <- coef(fit)[1]
Db2 <- coef(fit)[2]

lines(xx, C02*exp(-sqrt(lam/Db2)*xx), lty=2, col="blue", lwd=2)
legend("topright", lty=c(1, 2), col=c("red", "blue"),
       c("linear fit", "nonlinear fit"), lwd=c(2, 2))
```

```
par(new=T, fig=c(0.5, 1, 0.2, 0.8))
plot(x, log(y), pch=16, ylab="log(y)", xlab="x", las=1)
lines(xx, log(C0*exp(-sqrt(lam/Db)*xx)), col="red", lwd=2)
lines(xx, log(C02*exp(-sqrt(lam/Db2)*xx)), col="blue", lty=2, lwd=2)
dev.off()
```

Appendix 3
y=ax+b の a,b を推定してみる

```
set.seed(1)

###x は 1 から 100 まで###
x <- seq(1, 100)

###真の a, b の値の設定###
a <- 5
b <- 3

###y の生成。誤差も加える###
y <- a + b*x + rnorm(100)

###a と b の値をたくさん生成しておく（ベイズでいうと、事前分布みたいなもの）###
###今回は、計算時間短縮のため、0-10 の間で生成###
a2 <- runif(100, 0, 10)
b2 <- runif(100, 0, 10)

###計算したコストの値の格納場所###
cost <- matrix(0, 100, 1)

###コストはまぁ単純に最小二乗法で###
cost_func <- function(a3, b3) {
sum((a3 + b3*x - y)^2)
}

###コストの計算###
for(i in 1：100) {
cost[i] <- cost_func(a3=a2[i], b3=b2[i])
}

###何回パラメータ推定に費やすか###
iter <- 2000
```

```
###各パラメータの格納場所###
a_mean <- b_mean <- a_mirror <- b_mirror <-
new_a <- new_b <- matrix(0, iter, 1)

###while 文で iter で決めた回数になるまで繰り返し、パラメータ推定###
i <- 1
while(i < iter) {

###大量に発生させた中から、3 つランダムに取ってきて、それを mean として作成###
a_mean[i] <- mean(sample(a2, 3))
b_mean[i] <- mean(sample(b2, 3))

###大量に発生させた中から、ミラー用に一つだけ取ってくる###
a_mirror[i] <- sample(a2, 1)
b_mirror[i] <- sample(b2, 1)

###例えば、a_mean が 5 だったとして、a_mirror が 3、すると、2*5-3=7 となり、3 を中心に等しい距離だけ移動したことになる###
###何故この作業が必要なのかというと、もしどこかに一度トラップされてしまうと抜け出すことがこのアルゴリズム###
###は困難。なので、これで一気に移動してコストの小さいところを探索しているのだと思う。###
###新しく作成されたパラメータを new_a,b に格納###
new_a[i] <- 2*a_mean[i] - a_mirror[i]
new_b[i] <- 2*b_mean[i] - b_mirror[i]

###もし、新しく作ったパラメータで計算したコストが、最初に計算したコストの最大値よりも小さかったならば###
if (max(cost) > cost_func(a3=new_a[i], b3=new_b[i])) {

###まず、コスト群の中で、最大値が何番目になっているのか j に格納###
j <- which(cost == max(cost))

###j 番目パラメータを新しく作ったパラメータで更新###
a2[j] <- new_a[i]
b2[j] <- new_b[i]
```

```
###コストも更新###
cost[j] <- cost_func(a3=a2[j], b3=b2[j])

###i も更新###
i <- i+1

###もしそうでないなら###
} else {

###なんもせんでよろし###
###あえて明示的に、else の場合を書き下した###
i <- i
 }
}
hist(a2)
hist(b2)
```

Appendix 4

個体群増加モデルを解いてみる

```
###ode に与える関数は、(時間、状態変数、パラメータ)の関数でないといけない###
###実際には、t のところは何も明示的に t <-　とする必要はなく、ode の times 引数で###
###勝手にやってくれる###
library(deSolve)
Growth_model <- function (t, state, pars) {

###必要なものは、外から与える仕組み###
  with (as.list(c(state, pars)), {

###微分方程式のまま書いていて OK###
  dN <- r * N *(K - N)/K

###返り値はリストの形でしかダメらしい###
 return(list(dN, r * N *(K - N)/K))
})
}
###パラメータやら初期値やら時間変化を与える###
pars <- c(r=0.1, K=100)
ini <- c(N=1)
times <- seq(0, 100, length=200)

###微分方程式が勝手に離散に直されて、解かれる。y のところは初期値###
out <- as.data.frame(ode(func=Growth_model, y=ini,
                         parms=pars, times=times))
names(out) <- c("time", "N", "dN/dt")
###作図領域の設定###
par(oma=c(0, 0, 0, 2))

###N のプロット###
plot(out[, 1], out[, 2],xlab="time", ylab="N", col="blue", type="l", lwd=3)

###上書き準備###
par(new=T)
```

```
###dN/dt のプロット###
plot(out[, 1], out[, 3], axes=F, xlab="", ylab="", col="red", type="l", lwd=3)

###dN/dt の軸###
axis(4)
mtext(side=4, "dN/dt", outer=T)

###凡例###
legend("topleft", col=c("blue", "red"), leg=c("N", "dN/dt"), lwd=c(2, 2))
```

Appendix5

```
###p137 の図を MCMC で###
library(deSolve)
###教科書と一緒。コストを計算する関数を定義###
###事前分布の平均を決めるためだけに使う###
costf <- function(params)
 {with(as.list(params),
  {
     Carbon <- meanDepo*mult/k
     outtimes <- as.vector(oxcon$time)
     outmin <- ode(Carbon, outtimes, minmod, params)
     costt <- sum((outmin[, 3] - oxcon$cons)^2)
     return(costt)
  })
 }
###教科書と一緒。微分方程式を定義###
###事前分布の平均を決めるためだけに使う###
minmod <- function(t, Carbon, parameters)
 { with(as.list(c(Carbon, parameters)),
    {
     minrate <- k*Carbon
     Depo <- approx(Flux[, 1], Flux[, 2], xout=t)$y
     dCarbon <- mult*Depo - minrate
    list(dCarbon, minrate)
  })
 }

Flux <- matrix(ncol=2, byrow=TRUE, data=c(
   1, 0.654, 11, 0.167, 21, 0.060, 41, 0.070,
  73, 0.277, 83, 0.186, 93, 0.140,103, 0.255,
113, 0.231,123, 0.309,133, 1.127,143, 1.923,
153, 1.091,163, 1.001,173, 1.691,183, 1.404,
194, 1.226,204, 0.767,214, 0.893,224, 0.737,
234, 0.772,244, 0.726,254, 0.624,264, 0.439,
274, 0.168,284, 0.280,294, 0.202,304, 0.193,
315, 0.286,325, 0.599,335, 1.889,345, 0.996,
```

```
355, 0.681,365, 1.135))

oxcon <- as.data.frame(matrix(ncol=2, byrow=T, data=c(
   68, 0.387, 69, 0.447, 71, 0.473, 72, 0.515,
  189, 1.210,190, 1.056,192, 0.953,193, 1.133,
  220, 1.259,221, 1.291,222, 1.204,230, 1.272,
  231, 1.168,232, 1.168,311, 0.963,312, 1.075,
  313, 1.023)))

names(oxcon) <- c("time", "cons")

###初期値###
meanDepo <- mean(approx(Flux[, 1], Flux[, 2], xout=seq(1, 365, by=1))$y)

###事前分布のアタリをつけるために計算###
multser <- seq(1, 1.5, by=.05)
numms <- length(multser)
kseries <- seq(0.001, 0.5, by=0.005)
numks <- length(kseries)

outcost <- matrix(nrow=numms, ncol=numks)

for(m in 1:numms)
 {
 for(i in 1:numks)
   {
   pars <- c(k=kseries[i], mult=multser[m])
   outcost[m, i] <- costf(pars)
   }
 }

minpos <- which(outcost==min(outcost), arr.ind=T)
multm <- multser[minpos[1]]
ki <- kseries[minpos[2]]

###事後分布を計算するための関数。パラメータの事前分布は大体アタリをつけて縛った
```

```
###
cost_likelihood <- function(params)
{with(as.list(params),
  {
    Carbon <- meanDepo*mult/k
    outtimes <- as.vector(oxcon$time)
    outmin <- ode(Carbon, outtimes, minmod, params)
    costt <- sum(log(dnorm(outmin[, 3] - oxcon$cons, 0, 1))) +
                    log(dnorm(k, ki, 1)) + log(dnorm(mult, multm, 1))
    return(costt)
  })
 }
###事後分布の大体のイメージ図###
###教科書のもの(p138)と大体一緒###
mat <- expand.grid(k=seq(1e-3, 0.05, by=0.002),
                        mult=seq(1, 2, by=0.05))
for(i in 1：nrow(mat)) {
mat[i, 3] <- cost_likelihood(c(k=mat[i, 1], mult=mat[i, 2]))
}
library(lattice)
contourplot(mat[, 3] ~ mat[, 2] + mat[, 1])

###事後分布計算###
set.seed(1)
###すごく時間がかかるので、試すときは iter の数字をすごく小さくしてからのほうが良い
です###
iter <- 5000

###ランダムウォークの幅や採択棄却の乱数をあらかじめ用意###
###初期値は事前分布のど真ん中から###
u <- runif(iter, 0, 1)
k2 <- mult2 <- numeric(iter+1)
k2[1] <- ki
mult2[1] <- multm
lag <- 0.01
lag2 <- 0.4
```

```
###メトロポリス法で MCMC###
for(i in 1：iter) {
par <- c(k2[i], mult2[i])

###ジオメトリックランダムウォーク(k が負だと困るので)###
k3 <- k2[i] *exp(runif(1, -lag, lag))

###こちらは普通のランダムウォーク###
mult3 <- mult2[i] + runif(1, -lag2, lag2)
par2 <- c(k3, mult3)
if (u[i] < exp(cost_likelihood(c(k=par2[1], mult=par2[2])) -
                    cost_likelihood(c(k=par[1], mult=par[2])))) {
k2[i+1] <- k3
mult2[i+1] <- mult3
 } else {
k2[i+1] <- k2[i]
mult2[i+1] <- mult2[i]
 }
}
###pricefit は p130 で定義されているので、それを使ってください###
optpar <- pricefit(par=c(k=ki, mult=multm), minpar=c(0.001, 1),
              maxpar <- c(0.5, 1.5), func <- costf, npop=50, numiter=500,
              centroid=3, varleft=1e-8)

outtimes <- seq(1, 365, by=1)
Carbon <- meanDepo*optpar$par[2]/optpar$par[1]
names(Carbon) <- "Carbon"
out <- as.data.frame(ode(Carbon, outtimes, minmod, optpar$par))
names(out) <- c("times", "Carbon", "minrate")

###事後分布###
hist(k2, xlab="k")
x11();hist(mult2, xlab="mult")

###パラメータの 95%信頼区間を使って予測値の 95%信頼区間を求めてみる###
```

```
k2_25 <- as.numeric(quantile(k2, 0.025))
k2_975 <- as.numeric(quantile(k2, 0.975))
k2_median <- as.numeric(median(k2))

mult2_25 <- as.numeric(quantile(mult2, 0.025))
mult2_975 <- as.numeric(quantile(mult2, 0.975))
mult2_median <- as.numeric(median(mult2))

Carbon95_lo <- meanDepo * mult2_25/k2_25
Carbon95_up <- meanDepo * mult2_975/k2_975
Carbon_median <- meanDepo * mult2_median/k2_median

par3 <- c(k=k2_25, mult=mult2_25)
par4 <- c(k=k2_975, mult=mult2_975)
par5 <- c(k=k2_median, mult=mult2_median)

out25 <- as.data.frame(ode(Carbon95_lo, outtimes, minmod, par3))
out975 <- as.data.frame(ode(Carbon95_up, outtimes, minmod, par4))
out_median <- as.data.frame(ode(Carbon_median, outtimes, minmod, par5))

names(out25) <- c("times", "Carbon", "minrate")
names(out975) <- c("times", "Carbon", "minrate")
names(out_median) <- c("times", "Carbon", "minrate")

par(oma=c(0, 0, 0, 2))
plot(Flux, type="l", xlab="daynr", ylab="mmol/m2/d",
        main="Sediment-detritus model", lwd=2)
lines(out$time, out$minrate, lwd=2, col="black")
points(oxcon$time, oxcon$cons, pch=25, col="black", bg="darkgray", cex=2)
lines(out25$time, out25$minrate, lwd=2, col="blue")
lines(out975$time, out975$minrate, lwd=2, col="green")
lines(out_median$time, out_median$minrate, lwd=2, col="red")

par(new=T)
plot(out$time, out$Carbon,axes=F, xlab="",ylab="", type="l" ,
        lty=2, col="black", lwd=2)
```

```
axis(4)
mtext(side=4, "mmolC/m2", outer=T)

points(out_median$time, out_median$Carbon,type="l",
       lty=2, col="red", lwd=2)
points(out25$time, out25$Carbon, type="l",
       lty=2, col="blue", lwd=2)
points(out975$time, out975$Carbon, type="l",
       lty=2, col="green", lwd=2)
legend("topleft", col=c("green", "red", "blue"),
          leg = c("97.5%", "Median", "2.5%"),
          lwd=c(2, 2, 2), lty=c(1, 1, 2))
```