概日リズム３変数

ryamada

2019年6月6日

６月５日に扱った、「概日リズム」周期的振動を作り出すモデル

$$\frac{dM}{dt} = \frac{k}{h^n + P^n} - \frac{aM}{a' + M}$$
$$\frac{dR}{dt} = \frac{sM}{s' + M} - \frac{dR}{d' + R} - \frac{uR}{u' + R} + \frac{vP}{v' + P}$$
$$\frac{dP}{dt} = \frac{uR}{u' + R} - \frac{vP}{v' + P}$$

離散的に近似する～差分方程式

dX, dtは無限小だけれど、そこそこに小さいのであれば、有限小の$\Delta X, \Delta t$にしても、ま、いいか、という式

$$\frac{\Delta M}{\Delta t} = \frac{k}{h^n + P^n} - \frac{aM}{a' + M}$$
$$\frac{\Delta R}{\Delta t} = \frac{sM}{s' + M} - \frac{dR}{d' + R} - \frac{uR}{u' + R} + \frac{vP}{v' + P}$$
$$\frac{\Delta P}{\Delta t} = \frac{uR}{u' + R} - \frac{vP}{v' + P}$$

変化量を書き直す

$$\Delta M = \Delta t \times (\frac{k}{h^n + P^n} - \frac{aM}{a' + M})$$
$$\Delta R = \Delta t \times (\frac{sM}{s' + M} - \frac{dR}{d' + R} - \frac{uR}{u' + R} + \frac{vP}{v' + P})$$
$$\Delta P = \Delta t \times (\frac{uR}{u' + R} - \frac{vP}{v' + P})$$

シミュレーションする

- 有限小時間幅Δtを定める
- ステップ数 n を定める。$\Delta t \times n$がシミュレーションする時間になる
- 出てくる変数を２種類に分ける
  - 定数(Constants)
  - 時間で変化する変数(時間の関数) (Variables)
- Constantsを与える
- Variablesにステップ数に見合う長さの記録用ベクトルを作る
- Variablesに初期値を与える
- Δtごとに、Variables の変化量 ΔXを計算し、加算する
- 算出した値は保管する

Δt

```
dt <- 10^(-2)
N <- 10000
```

定数

```
k <- 1
h <- 1
n <- 1
a <- 1
a. <- 1
s <- 2
s. <- 2
d <- 1
d. <- 1
u <- 2
u. <- 2
v <- 1
v. <- 1
```

Variables 時間の変数

```
M <- rep(0,N)
P <- rep(0,N)
R <- rep(0,N)

M[1] <- 1
P[1] <- 1
R[1] <- 1
```

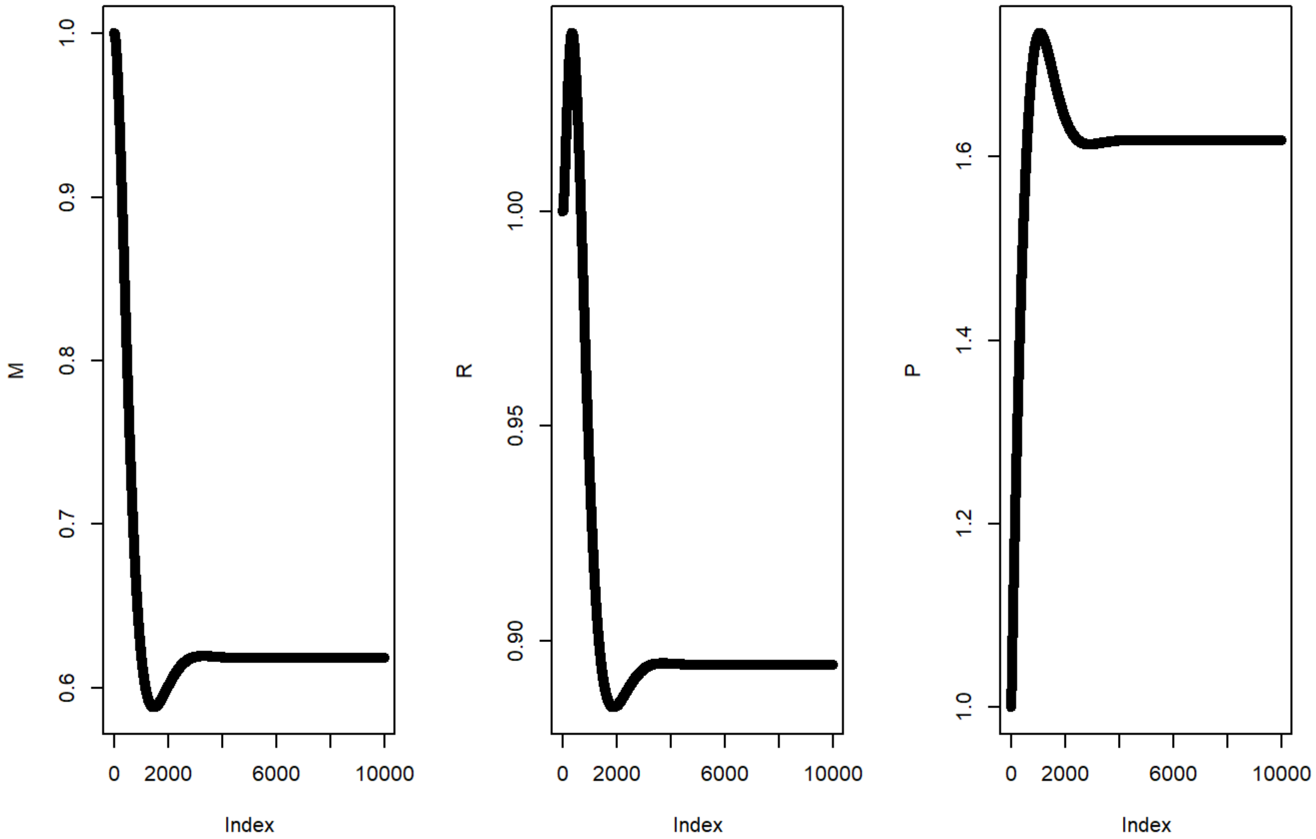
時間を進める

```
for(i in 2:N){
  # 今の時刻のVariablesの値
  M.now <- M[i-1]
  P.now <- P[i-1]
  R.now <- R[i-1]
  # 変化量を計算する
  dM <- dt * (k/(h^n + P.now^n) - (a*M.now)/(a.+M.now))
  dR <- dt * ((s*M.now)/(s. +M.now) - (d*R.now)/(d. + R.now) - (u*R.now)/(u. + R.now) + (v*P.now)/(v.
+P.now))
  dP <- dt * ((u*R.now)/(u. + R.now) - v*P.now/(v.+P.now))

  # dt後の値を格納する
  M[i] <- M.now + dM
  R[i] <- R.now + dR
  P[i] <- P.now + dP
}
```

様子を見てみる

時間軸に沿って変化具合を見る

```
par(mfcol=c(1,3)) # 画面を１行３列に分ける
plot(M)
plot(R)
plot(P)
```



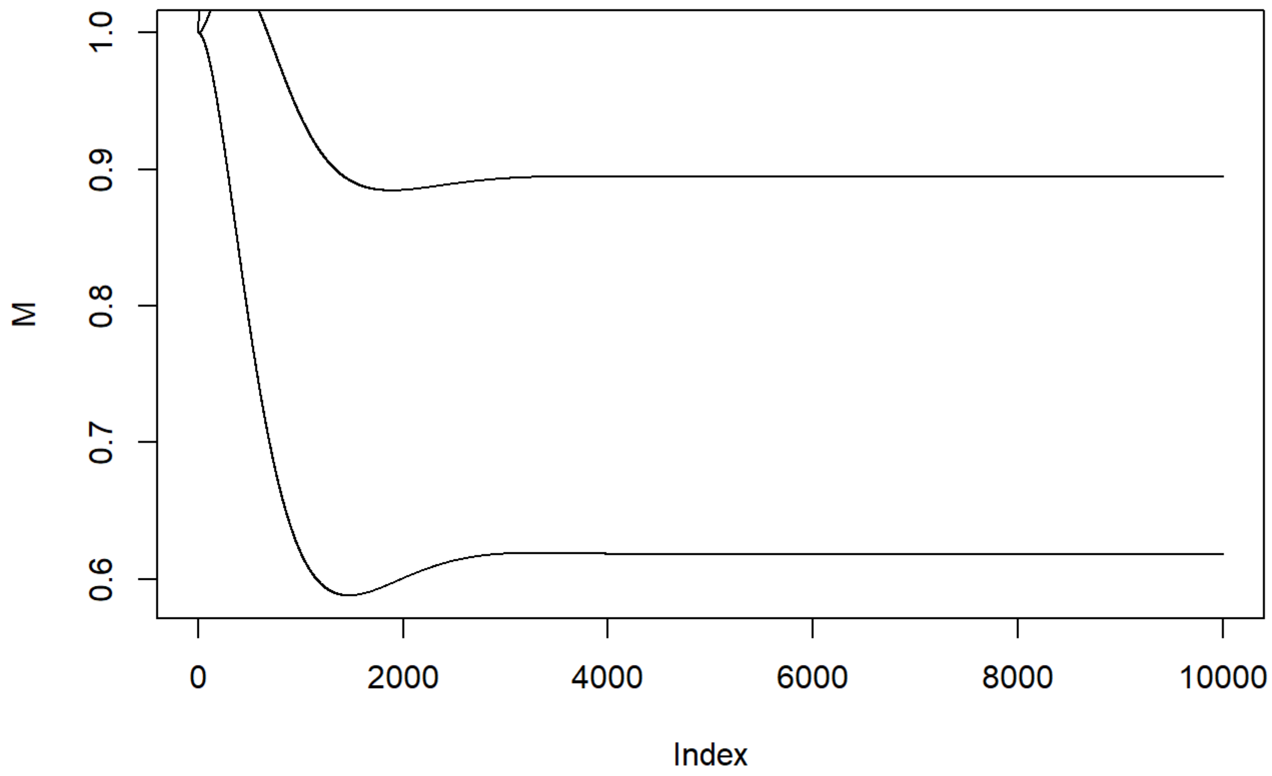
３変数併せてプロットする

```
par(mfcol = c(1,1))
plot(M,type="l")
points(R,type="l",ncol=2)
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "ncol" はグラフィッ
## クスパラメータではありません
```

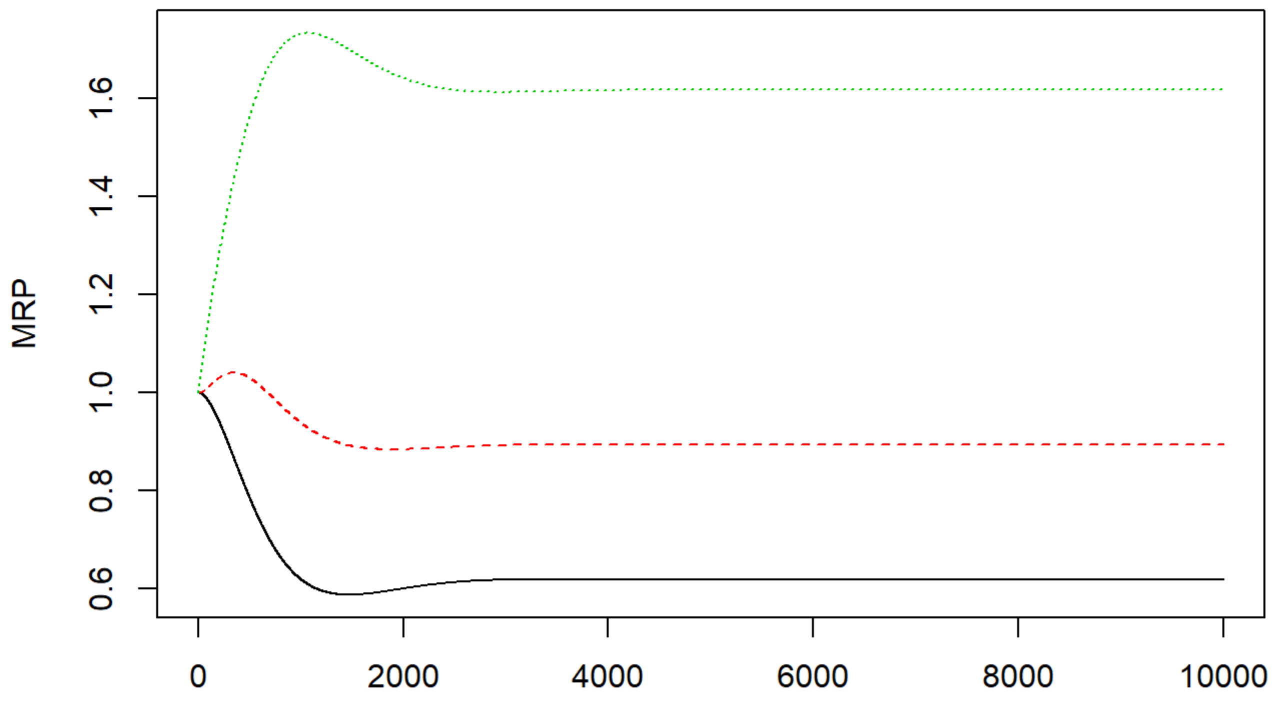
```
points(P,type="l",ncol=3)
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "ncol" はグラフィッ
## クスパラメータではありません
```



別法

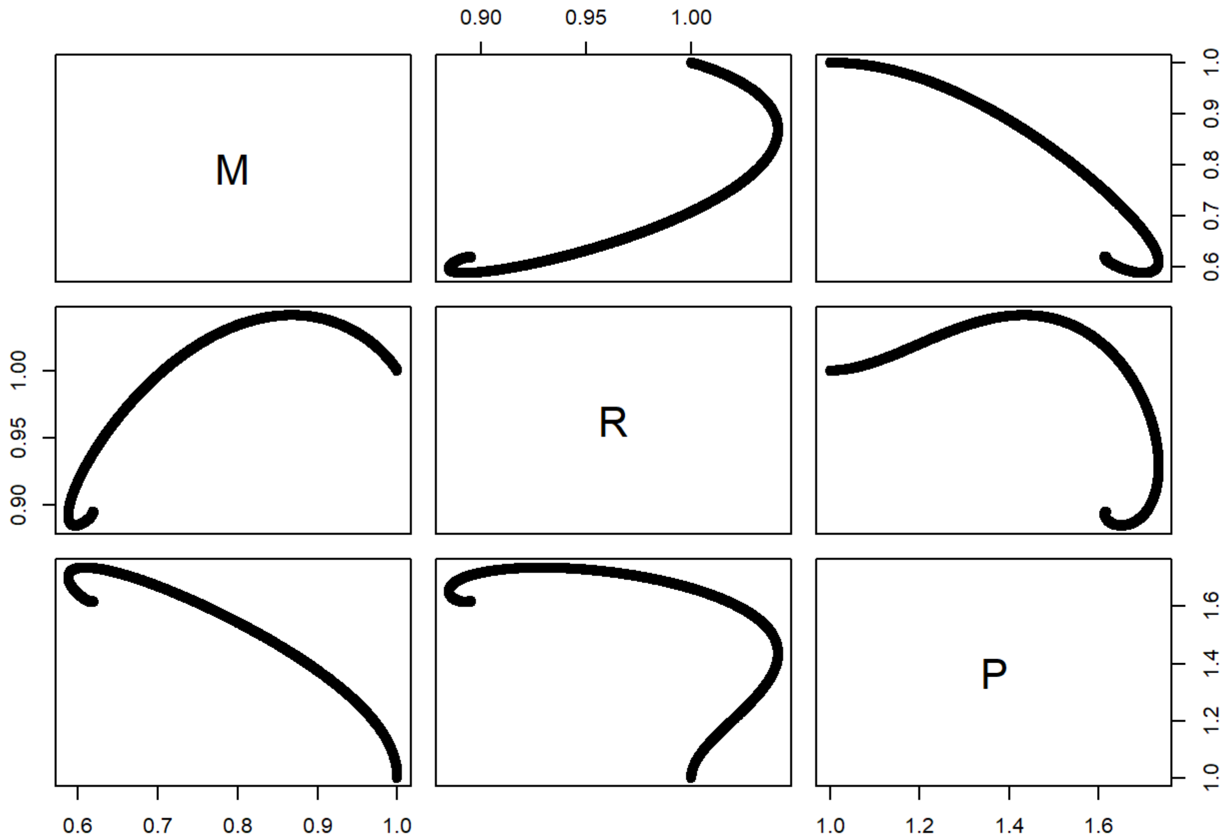
```
MRP <- cbind(M,R,P)
matplot(MRP,type="l")
```



状態空間で見てみる

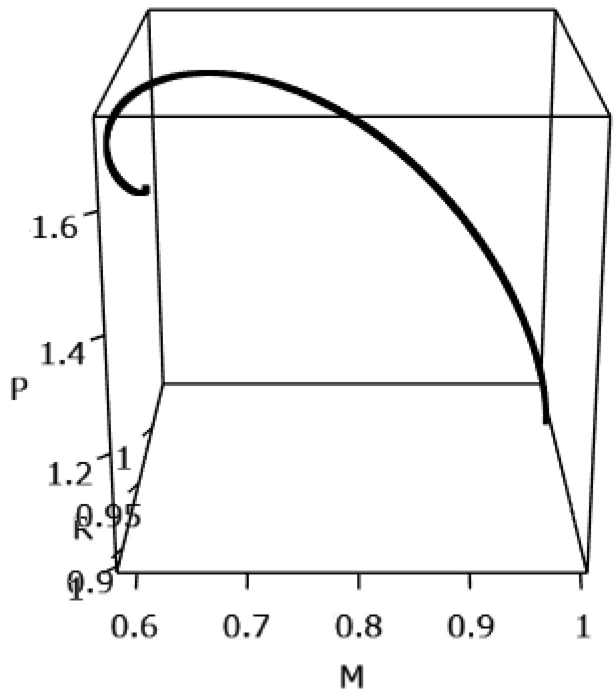
２変数ごとにみる

```
pairs(MRP)
```



３変数を３次元空間で見る

```
plot3d(MRP)
```



発展～微分方程式を指定すると、計算してくれるパッケージもある

```
# install.packages("deSolve")
```

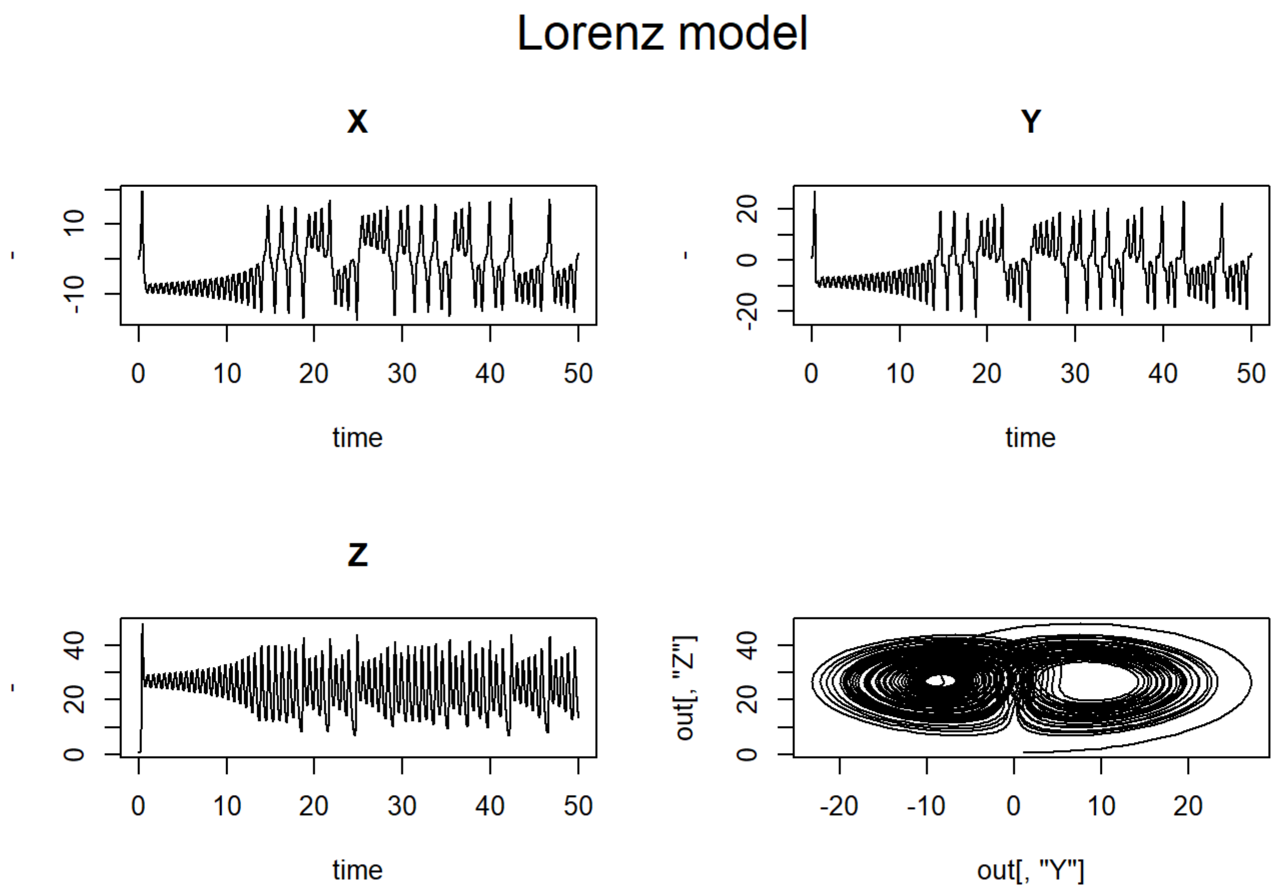
こちら (https://www.sixhat.net/lorenz-attractor-in-r.html)のページの例は「ローレンツアトラクタと呼ばれる非線形微分方程式の例

定数parametersと、初期値 stateを定め、時間刻み情報tとにより、連立微分方程式を以下のように書く

```
parameters <- c(s = 10, r = 28, b = 8/3)
state <- c(X = 0, Y = 1, Z = 1)

Lorenz <- function(t, state, parameters) {
    with(as.list(c(state, parameters)), {
        dX <- s * (Y - X)
        dY <- X * (r - Z) - Y
        dZ <- X * Y - b * Z
        list(c(dX, dY, dZ))
    })
}
```

ode()関数がシミュレーションしてくれる

```
times <- seq(0, 50, by = 0.01)
library(deSolve)
out <- ode(y = state, times = times, func = Lorenz, parms = parameters)

par(oma = c(0, 0, 3, 0))
plot(out, xlab = "time", ylab = "-")
plot(out[, "Y"], out[, "Z"], pch = ".", type = "l")
mtext(outer = TRUE, side = 3, "Lorenz model", cex = 1.5)
```

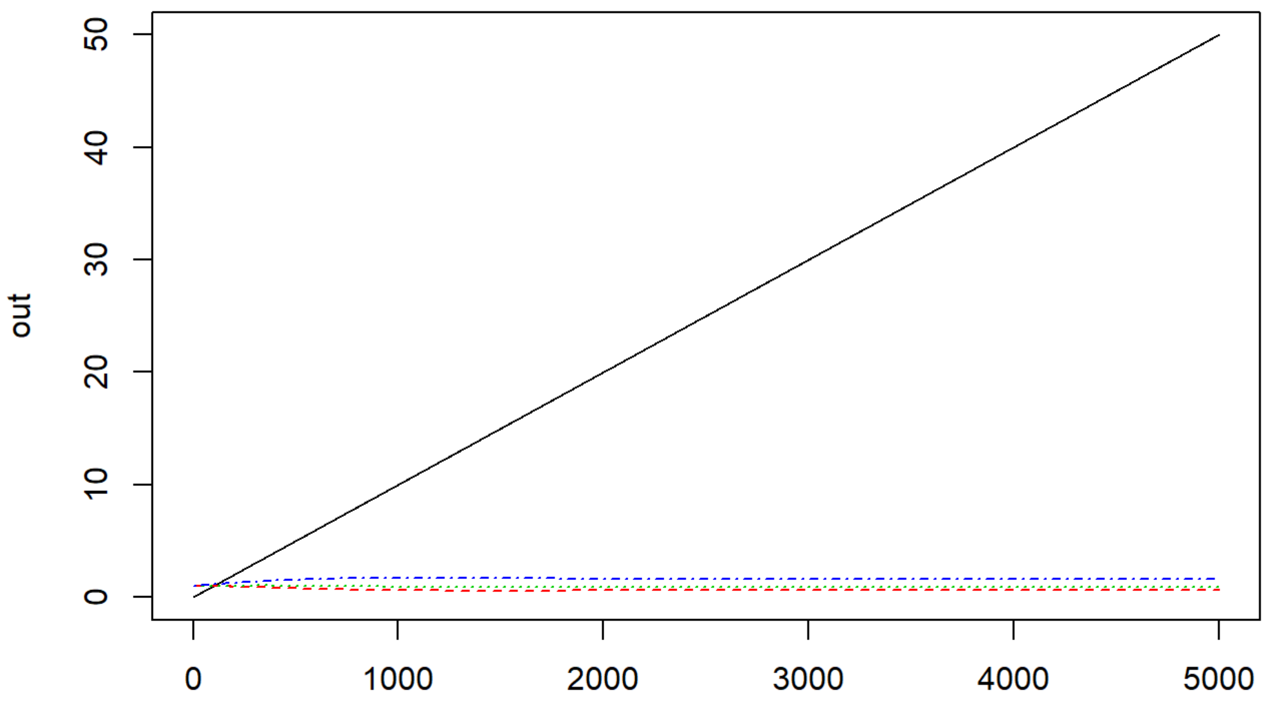


これに沿って、概日リズムモデルを書き換えてみよう

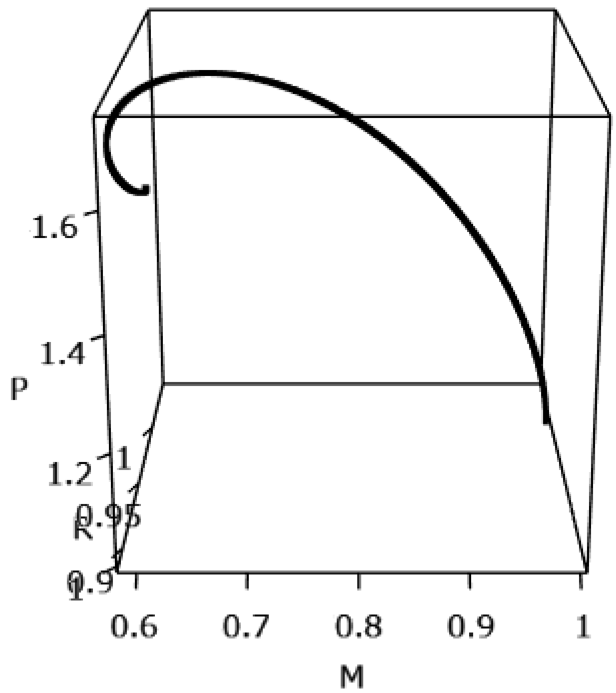
```
parameters <- c(k = 1,h = 1, n = 1,a = 1,a. = 1,s = 2,s. =2, d = 1,d. = 1,u = 2, u. = 2, v = 1, v. = 1
)
state <- c(M = 1, R = 1, P = 1)

our.model <- function(t, state, parameters) {
    with(as.list(c(state, parameters)), {
        dM <- (k/(h^n + P^n) - (a*M)/(a.+M))
        dR <- ((s*M)/(s. +M) - (d*R)/(d. + R) - (u*R)/(u. + R) + (v*P)/(v. +P))
        dP <- ((u*R)/(u. + R) - v*P/(v.+P))

        list(c(dM, dR, dP))
    })
}
```

```
times <- seq(0, 50, by = 0.01)

#out <- ode(y = state, times = times, func = Lorenz, parms = parameters)
out <- ode(y = state, times = times, func = our.model, parms = parameters)
```

```
matplot(out,type="l")
```




```
plot3d(out[,2:4]) # 第１列は時刻情報
```


1.6
1.4
P
1.2
1
0.95
R
0.9
1
0.6
0.7
0.8
0.9
1
M