

Exercise 4.1: Lorenz attractor

For the reader convenience, we reported below the complete “Lorenz code”, discussed in the book in various places. By running the code, we obtain in sequence the Figures 3.7, 3.9, 3.11, 3.12, 3.13(b), 4.3, 4.9, 4.13, 4.14(a).

```
#Lorenz attractor: Complete
#install.packages("tseriesChaos", dep = TRUE) # to install the package if not present
library(tseriesChaos)
parms<- c(10,8/3,28) # parameters: sigma, beta, rho
tinit<- 0
tfin<- 100
step<- 0.01
times<-seq(tinit,tfin,by=step)
funct<- function(t,integ,parms) {
  x<-integ[1]
  y<-integ[2]
  z<-integ[3]
  sigma<- parms[1]
  beta<- parms[2]
  rho<- parms[3]
  dx<- sigma*(y-x) # that is dx/dt = sigma*(y-x)
  dy<- x*(rho-z)-y # that is dy/dt = x*(rho-z)-y
  dz<- x*y-beta*z # that is dz/dt = xy -beta z
  list(c(dx,dy,dz))
} # end of funct

require(deSolve)
cinit<-c(1,1,1)
xyz<-lsoda(cinit,times,funct,parms)
#xyz # comment if you do not wish the xyz values printed
par(mfrow=c(3,1))
par(mar = c(6.3, 4.8, 1., 3)) # to control the margin size
par(cex.lab=2,cex.axis=1.6,lwd=1,lty=1)
plot(xyz[,1],xyz[,2],type="l",xlab="t",ylab="x(t)", # x(t) vs t
      xlim=c(tinit,tfin),ylim=c(-30,30))
plot(xyz[,1],xyz[,3],type="l",xlab="t",ylab="y(t)", # y(t) vs t
      xlim=c(tinit,tfin),ylim=c(-30,30))
plot(xyz[,1],xyz[,4],type="l",xlab="t",ylab="z(t)", # z(t) vs t
      xlim=c(tinit,tfin),ylim=c(0,50))
windows()
# phase space portrait
require(scatterplot3d)
scatterplot3d(xyz[,2],xyz[,3],xyz[,4],type="l",xlim=c(-30,30),
              cex.lab=1.4,cex.axis=1.2)

trans<- 2000 # integration time step considered as transient
# discard initial transient:
x <- window(xyz[,2],trans)
y <- window(xyz[,3],trans)
z <- window(xyz[,4],trans)
t_start<- trans*step - step # new initial time
t_time<-seq(t_start,tfin,by=step) # new time interval
windows()
par(mfrow=c(3,1))
par(mar = c(6.3, 4.8, 1., 3))
```

```

par(cex.lab=2,cex.axis=1.6,lwd=1,lty=1)
# x(t), y(t), z(t) after the transient:
plot(t_time,x,type="l",xlab="t",ylab="x(t)",
      xlim=c(t_start,tfin),ylim=c(-30,30))
plot(t_time,y,type="l",xlab="t",ylab="y(t)",
      xlim=c(t_start,tfin),ylim=c(-30,30))
plot(t_time,z,type="l",xlab="t",ylab="z(t)",
      xlim=c(t_start,tfin),ylim=c(0,50))

m.max<- 6 # embedding dimensions: from 1 to m_max
d<- 18 # tentative time delay (see below)
tw<- 100 # Theiler window
rt<- 10 # escape factor
eps<- sd(x)/10 # neighbourhood diameter
fn <- false.nearest(x,m.max,d,tw,rt,eps)
fn
windows()
plot(fn)

windows()
lm<- 60 # largest lag
mutual(x,lag.max = lm) # average mutual information to suggest d

# embedding Procedure
m<- 3 # choose embedding dimension
d<- 10 # choose time delay (d<- 10)
xyz <- embedd(x,m,d) # embed the 'observed' series
windows()
scatterplot3d(xyz, type="l")

windows()
n<- 800
par(mai=c(1.02,1.,0.82,0.42)+0.1)
plot(0,0,type="n",xlab="i",ylab="j",xlim=c(1,n),ylim=c(1,n),cex.lab=1.6,cex.axis=1.2)
family="Times",font.lab=3)
distx<- matrix(,n,n)
eps<- 5
xx<- xyz[,1]
yy<- xyz[,2]
zz<- xyz[,3]
for(i in 1:n){ # construction of the thresholded recurrence plot
  for(j in 1:n){
    # Euclidean distance:
    distx[i,j]<- sqrt( (xx[i]-xx[j])^2 + (yy[i]-yy[j])^2 + (zz[i]-zz[j])^2 )
    if(distx[i,j]<eps) points(i,j,pch=19,cex=0.01)
  }
}

# unthresholded recurrence plot

time<- seq(1:n)
windows()
par(mai=c(1.02,1.,0.82,0.42)+0.1,cex.lab=1.6,cex.axis=1.2)
require(gplots)
distx<- distx/max(distx)
distx<- 1-distx

```

```

#filled.contour(time,time,distx,col=colorpanel(10,"white","black"),nlevels=10,
filled.contour(time,time,distx,col=gray.colors(10,start=1,end=0),nlevels=10,
xlab = "i", ylab = "j", main = "",xlim=c(0,n),ylim=c(0,n),las=0,
key.axes = axis(4, las=1) )

# MCLE: maximum characteristic Lyapunov exponent (Kantz)

S_nu <- lyap_k(x,m=3,d=10,t=20,k=6,ref=5000,s=600,eps=5)
windows()
par(mai=c(1.02,1.,0.82,0.42)+0.1)
plot(S_nu,xlab = expression(paste(nu)),ylab=expression(paste("S", (nu))),
lwd=2,lty=1,cex=1.5)
lmin<- 80
lmax<- 280
# to add the vertical lines delimiting the linear region:
abline(v=lmin,lty=4,lwd=2,col="black")
abline(v=lmax,lty=4,lwd=2,col="black")
lr<- S_nu[lmin:lmax] # output of lyap_k(.) in [lmin, lmax]
l_lr<- length(lr)-1
lt<- seq(lmin*step,lmax*step,by=step)
lm(lr~lt) # regression analysis in R
abline(lyap(S_nu,lmin,lmax),lty=2,lwd=2,col="black")

# estimate of D2

C.m <- d2(x,m=6,d=10,t=100,eps.min=0.01,neps=100) # correlation integral, m = 1,...,6
C.m <- data.frame(unclass(C.m)) # class attribute removed
C.3 <- subset(C.m,eps > 0.1 & eps < 1.2, select=c(eps,m3)) # eps in [0.1, 1.2]
lm(log(m3) ~ log(eps), data = C.3) # D2 estimate with m=3
windows() # if other plots are made before
par(mai=c(1.02,1.,0.82,0.42)+0.2)
plot(C.m[,1],C.m[,4], type="l",log="xy",main="",
xlab=expression(paste(epsilon)),ylab=expression(paste(widehat(C),(epsilon))),
lwd=2,lty=1,cex.axis=1.3,cex.lab=2.0, xlim=c(0.01,100),ylim=c(0.00000002,1))

# stop here to plot only C(m=3)
#####

# add the lines below to plot C(m=1,...,6)
n.m<- ncol(C.m)
for(i in (n.m-0):2) lines(C.m[,c(1,i)],lwd=2)

```

Some comments are in order. We repeat that the length of the series is $t_{fin}/step$ ($t_{init}=0$). So, if $t_{fin}=100$, the series contains 10000 “observations”. Actually $10000+1$, including the starting point = 0. The Lorenz original parameters are $\sigma = 10$, $\beta = 8/3$, $\rho = 28$, as used in the book. However, we find in the literature also $\sigma = 16$, $\beta = 4$, $\rho = 45.92$. With these values, it results $\hat{\lambda} = 1.54$, linear region = $[60, 160]$ and $k(n_{fin}) = 8$; with $k = 3$, it is $\hat{\lambda} = 1.46$.