

Exercise 4.1: Duffing attractor

The Duffing oscillator describes the motion of a particle in a two-well potential, with damping and a periodic driving force:

$$\begin{aligned}dx/dt &= y \\ dy/dt &= x(1 - x^2) - cy + F \cos(z) \\ dz/dt &= w\end{aligned}$$

where x is the position and y the velocity of the particle; $z = wt$ is the phase and F the amplitude of the driving force, respectively; c is the damping term. For a discussion on the Duffing system see De Souza-Machado et al., *American Journal of Physics*, **58**, 321 (1990). As the damped driven pendulum (Sect 4.5) the Duffing system displays a variety of behaviours by varying the parameters. For instance, fixed c and w ($c = 0.5, w = 1$), with $F = 0.325$ the system converges to a period 1 limit cycle. If F is increased further, (e.g., $F = 0.34875$ a period doubling occurs, finally for $F = 0.42$ the system displays a chaotic behaviour.

Exercise: Study the Duffing system with the code below. The reader is also encouraged to explore what happens by varying the values of w, F, c .

Solution: Run the code `Duffing attractor: Complete`. Here the code is written for the chaotic regime ($F = 0.42$).

```
# Duffing attractor: Complete
library(tseriesChaos)
parms<- c(0.5,1,0.42) # parameters: c, w, F
# F=0.325 period 1; F=0.34875 period 2; F=0.42 chaos
tinit<- 0
tfin<- 650
step<- 0.1
times<- seq(tinit,tfin,by=step)
funct<- function(t,integ,parms) {
  x<-integ[1]
  y<-integ[2]
  z<-integ[3]
  c<- parms[1]
  w<- parms[2]
  F<- parms[3]

  dx <- y
  dy <- x*(1-x^2) - c*y + F*cos(z)
  dz <- w
  list(c(dx,dy,dz))
} # end of funct

require(deSolve)
cinit<-c(0,0,0)
xyz<-lsoda(cinit,times,funct,parms)
#xyz # comment if you do not wish the xyz values printed
par(mfrow=c(2,1))
par(mar = c(6.3, 4.8, 1., 3))
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(-1.5,1.5))
plot(xyz[,1],xyz[,3],type="l",xlab="t",ylab="y(t)", # y(t) vs t
xlim=c(tinit,tfin),ylim=c(-1,1))
#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(-10,20),
cex.lab=1.4,cex.axis=1.2)

windows()
par(mai=c(1.02,1.,0.82,0.42)+0.1)
plot(xyz[,2],xyz[,3],type="p",pch=20,cex=0.7,xlab="x(t)",ylab="y(t)",
cex.lab=1.5,cex.axis=1.2,lwd=3,lty=1,xlim=c(-1.5,1.5),ylim=c(-1,1))

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(2,1))
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(-1.5,1.5))
plot(t_time,y,type="l",xlab="t",ylab="y(t)",
xlim=c(t_start,tfin),ylim=c(-1,1))
#plot(t_time,z,type="l",xlab="t",ylab="z(t)",
#xlim=c(t_start,tfin),ylim=c(0,50))

windows()
par(mai=c(1.02,1.,0.82,0.42)+0.1)
plot(x,y,type="p",pch=20,cex=0.7,xlab="x(t)",ylab="y(t)",
cex.lab=1.5,cex.axis=1.2,lwd=2,lty=1,xlim=c(-1.5,1.5),ylim=c(-1,1))

m.max<- 6 # embedding dimensions: from 1 to m_max
d<- 40 # 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<- 24 # choose time delay
xyz <- embedd(x,m,d) # embed the 'observed' series
windows()
par(mai=c(1.02,1.,0.82,0.42)+0.1)
plot(xyz[,2],xyz[,3],type="p",pch=20,cex=0.7,xlab="x(t)",ylab="y(t)",
cex.lab=1.5,cex.axis=1.2,lwd=3,lty=1,xlim=c(-1.5,1.5),ylim=c(-2,2))

windows()
n<- 1000
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<- 1
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=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=24,t=50,k=5,ref=4200,s=400,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<- 60
lmax<- 160
# 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=30,t=50,eps.min=0.005,neps=100) # correlation integral, m = 1,...
C.m <- data.frame(unclass(C.m))                # class attribute removed
C.3 <- subset(C.m,eps > 0.08 & eps < 0.8, select=c(eps,m3)) # eps in [0.08, 0.8]
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(0.004,100),ylim=c(0.000001,1))

# 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,lty=1)

```

The analogous figures resulted from the code `Rosslor attractor`: Complete for the Rössler attractor, are yielded by the code `Duffing attractor`: Complete.

1. Solutions of the Duffing equations.
2. Phase space portrait of the chaotic attractor.
3. Projection of the trajectory in the three-dimensional phase space onto the (x, y) plane
4. As the first figure, but without the transient.
5. Projection of the trajectory in the three-dimensional phase space onto the (x, y) plane without the transient.
6. Percentage of false nearest neighbours as a function of the embedding dimension, for the “observed” series x .
7. The average mutual information as a function of the lag, for the “observed” series x .
8. Plot in the reconstructed phase space of the Duffing strange attractor.
9. Thresholded recurrence plot.
10. Unthresholded recurrence plot.
11. Evolution of the logarithm of the mean distance $S(\nu)$ as a function of the time step ν .

12. Estimated correlation integral with the embedding dimension from $m = 1$ to $m = 6$.

The estimates of the invariants result to be $\hat{\lambda} = 0.105$ and $\hat{D}_2 = 2.12$.

Notice that in the figure concerning the average mutual information as a function of the lag, the minimum is for the $m = 4$, even though the values for $m = 3$ and $m = 5$ are almost equal. So we have chosen $m = 3$, however in similar cases, it is cautious to run the code also with $m = 4$. In this case, the estimates of the invariants are again $\hat{\lambda} = 0.105$ and $\hat{D}_2 = 2.12$.

More on Duffing attractor

We have written that the Duffing oscillator describes the motion of a particle in a two-well potential.

Exercise: Plot this potential

Solution: We see that the second equation of the system eqn (1) is the Newton $F = ma$ plus the damping and the driving force. Therefore, we have:

$$F_x = x(1 - x^2) = -\frac{dU(x)}{dx}$$

so

$$U(x) = -\frac{x^2}{2} + \frac{x^4}{4} \quad (1)$$

depicted in Fig. 1 by the code:

```
# Duffing oscillator potential: #U(x) = - x^2/2 + x^4/4

duff.pot <- function(x) - x^2/2 + x^4/4
par(mar = c(6.3, 4.8, 1., 3))
curve(duff.pot, type="l", cex.lab=2, cex.axis=1.5, lwd=3, font.lab=3,
xlab = "x", ylab = "U(x)", xlim=c(-2, 2), ylim=c(-0.3, 0.5), tck=-.02)
library(Hmisc) # to add minor tick marks
minor.tick(nx=5, ny=2, tick.ratio=0.5)
```

There are two *stable* fixed points at $x = -1$ and $x = +1$ and one *unstable* fixed point at $x = 0$. When the system is in the limit cycles regime is either in the “left well” or in the “right well”, depending on the initial conditions, but cannot bounce between the wells.

Exercise: Study the Duffing system in periodic regime

Solution: The code below, Duffing attractor: Limit cycle, is the code Duffing attractor: Complete up to the line `trans<- 2000` # integration time step considered as transient excluded, but with a second initial point. Note that now the parameters are `parms<- c(0.5, 1, 0.325)`.

```
# Duffing attractor: Limit cycle
library(tseriesChaos)
parms<- c(0.5, 1, 0.325) # parameters: c, w, F
# F=0.325 period 1; F=0.35 period 2; F=0.42 chaos
tinit<- 0
tfin<- 200
```

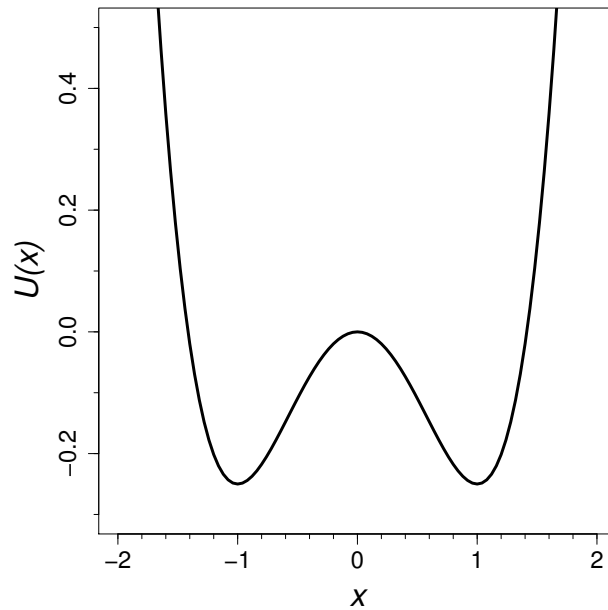


Figure 1 Duffing oscillator potential $U(x)$

```

step<- 0.2
times<- seq(tinit,tfin,by=step)
funct<- function(t,integ,parms) {
  x<-integ[1]
  y<-integ[2]
  z<-integ[3]
  c<- parms[1]
  w<- parms[2]
  F<- parms[3]
  dx <- y
  dy <- x*(1-x^2) - c*y + F*cos(z)
  dz <- w
  list(c(dx,dy,dz))
} # end of funct

require(deSolve)
cinit<-c(1,1,0)
xyz<-lsoda(cinit,times,funct,parms)
#xyz # comment if you do not wish the xyz values printed
par(mfrow=c(2,1))
par(mar = c(6.3, 4.8, 1., 3))
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(-1.5,1.5))
plot(xyz[,1],xyz[,3],type="l",xlab="t",ylab="y(t)", # y(t) vs t
      xlim=c(tinit,tfin),ylim=c(-1,1))
#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(-10,20),
              cex.lab=1.4,cex.axis=1.2)

```

```

windows()
par(mai=c(1.02,1.,0.82,0.42)+0.1)
plot(xyz[,2],xyz[,3],type="p",pch=20,cex=0.7,xlab="x(t)",ylab="y(t)",
cex.lab=1.5,cex.axis=1.2,lwd=3,lty=1,xlim=c(-1.5,1.5),ylim=c(-1,1))
## second initial point
cinit<-c(-1,1,0)
xyz<-lsoda(cinit,times,funct,parms)
points(xyz[,2],xyz[,3],col="red",pch=1,cex=1)

```

The projection of two trajectories onto the (x, y) plane is shown in Fig. 2. The plot displays clearly two limit cycles, reached after a transient time, one in the left well and one in the right well. As we did for others system, we can suppose that the the first compo-

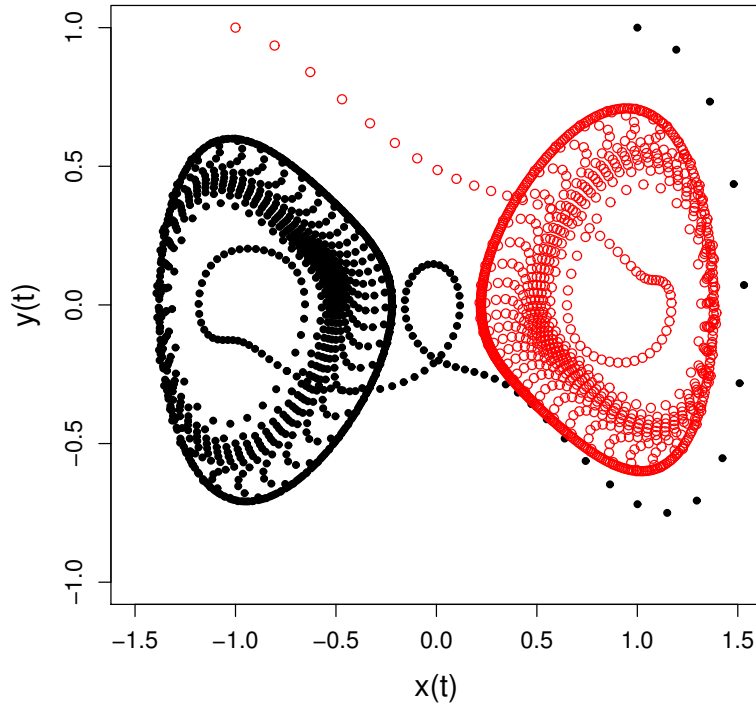


Figure 2 Trajectories projected onto the (x, y) plane of the Duffing system with parameters $c = 0.5$, $w = 1$, $F = 0.325$, initial conditions $(1, 1, 0)$ (full circles) and $(-1, 1, 0)$ (open circles).

nent of the system $x(t)$, is the “observed” series and reconstruct the phase space, after the transient time = 2000. We put `tfin<- 400` and `step<- 0.1`, with $m = 3$ and $d = 12$, the reconstructed limit cycle is obtained. You can see clearly the difference between the recurrence plots between of the chaotic attractor and the limit cycle (`n<- 300` and `eps<- 0.2`). Obviously it should be useless to try to search for a positive Lyapunov maximum exponent or a fractional correlation dimension.