6-4LORENTZsensitivity

10/01/2019

This R repository is for demonstration of algorithms involved in the book Mathematical Modeling (4th Edition) written by Prof. Mark. M. Meerschaert coded, edited and tested by Hao Li during Dec. 2018 - Jan. 2019.

# Effect of different variables on the pattern of the kinetic system

# Wrap 6-4 code into functions

#  
  
  
#Lorentz for different conditions:  
#  
  
xLorentz\_df = function(xi,param,  
 init,end,h){  
 #xi -initial x value,   
 #param -list of parameters  
 #t Domain: from ,to,N  
 t = seq(from = init, to = end,by=h)  
 #Copy an paste into this function  
 #from LORENTZ\_plain\_code.R, collapse the text using your editor to   
 #tidy up  
 x1p = function(x1,x2,x3,Sigma) -Sigma\*x1+Sigma\*x2  
 x2p = function(x1,x2,x3,r) -x2+r\*x1-x1\*x3  
 x3p = function(x1,x2,x3,b) -b\*x3+x1\*x2  
 #Using Euler s Method  
 xLorentz = function(x,dt,Sigma,r,b){  
 c(x[1]+dt\*x1p(x[1],x[2],x[3],Sigma = Sigma),  
 x[2]+dt\*x2p(x[1],x[2],x[3],r=r),  
 x[3]+dt\*x3p(x[1],x[2],x[3],b=b))  
 }   
 Sigma = param$Sigma  
 b = param$b  
 #Initial condition#xi= c(7,1,2)#r = param$r  
 dt = t[2] - t[1]  
 xdf = matrix(NA,length(t),3)#x1,x2,x3 then cbind t to the left  
 xdf[1,] = xi  
 #system.time({  
 for(i in seq\_along(t)[-1]){  
 xdf[i,] =xLorentz(xdf[i-1,],dt,Sigma,r=param$r,b)  
 }  
 #})  
 cbind(t,xdf)  
}

## Visualization function, default and plot3D(for non-interactive 3D plot)

plot.particle = function(xdf,  
 type = 'default',  
 grid =T,  
 add=F){  
 if(type == 'default'){  
 layout(matrix(1:4,2,2))  
 plot(xdf[,1],xdf[,2],  
 xlab = 't',  
 ylab = 'x1',type = 'l')  
 if(grid==T) grid()  
 plot(xdf[,1],xdf[,3],  
 xlab = 't',  
 ylab ='x2',type = 'l')  
 if(grid==T) grid()  
 plot(xdf[,1],xdf[,4],  
 xlab = 't',  
 ylab ='x3',type = 'l')  
 if(grid==T) grid()  
 plot3D::scatter3D(x = xdf[,2],  
 y = xdf[,3],  
 z = xdf[,4],  
 colvar =xdf[,1],add = F)  
 layout(matrix(1,1))  
 title("Default plots of 3D particle dynamic system")  
 }else if(type =='3d'){  
 #require package: plot3D3d  
 plot3D::scatter3D(x = xdf[,2],  
 y = xdf[,3],  
 z = xdf[,4],  
 colvar =xdf[,1],add = add)  
 #if(grid ==T) 3d::grid3d(side = c('x','y','z'))  
 }  
}

# 1 Figure 6-35 6-36

Compare different timestep setting: ##Case1: r=18, (x1,x2,x3) = (6.7,6.7,17),h = .005 ##Case2: r=18, (x1,x2,x3) = (6.7,6.7,17),h = .01

#  
  
#Case1: r=18, (x1,x2,x3) = (6.7,6.7,17),h = .005  
#Case2: r=18, (x1,x2,x3) = (6.7,6.7,17),h = .01  
  
require(doParallel)

## Loading required package: doParallel

## Loading required package: foreach

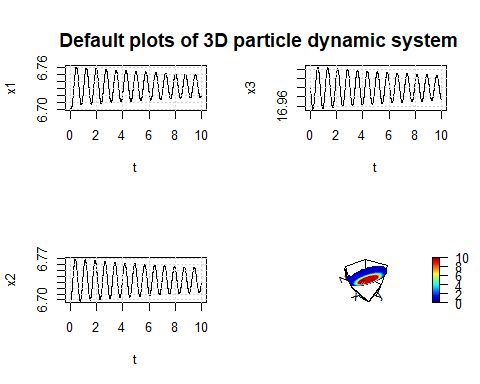
## Loading required package: iterators

## Loading required package: parallel

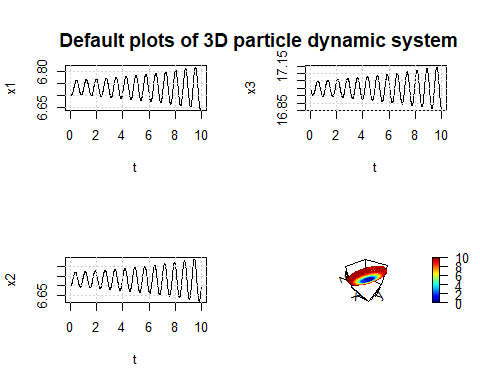
registerDoParallel(2)#Only 2 needed in this case  
  
comp1 =foreach(i = c(0.005,0.01)) %dopar% {  
 xLorentz\_df(xi = c(6.7,6.7,17),  
 param = list(Sigma = 10, b =8/3, r =18),  
 init = 0,end = 10,h=i)  
}  
  
str(comp1)

## List of 2  
## $ : num [1:2001, 1:4] 0 0.005 0.01 0.015 0.02 0.025 0.03 0.035 0.04 0.045 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : NULL  
## .. ..$ : chr [1:4] "t" "" "" ""  
## $ : num [1:1001, 1:4] 0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : NULL  
## .. ..$ : chr [1:4] "t" "" "" ""

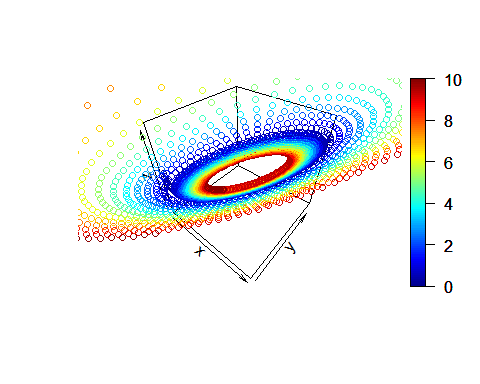
#Default Visualization plots defined in the previous code  
  
plot.particle(comp1[[1]])



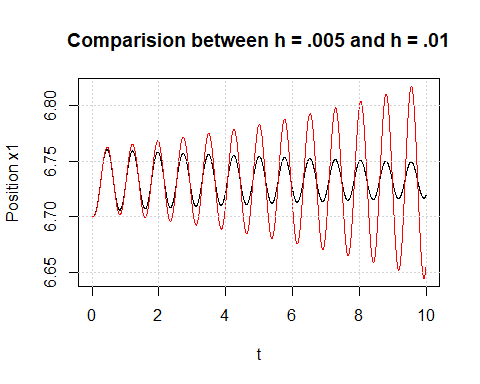
plot.particle(comp1[[2]])



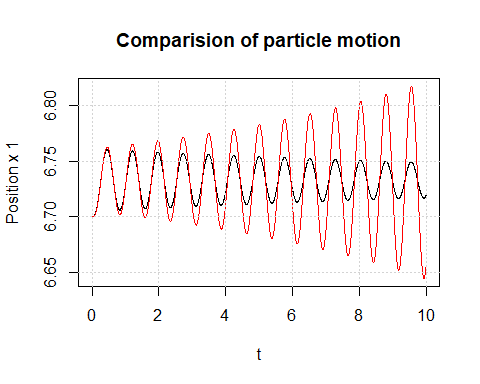
plot.particle(comp1[[1]],type = '3d')  
plot.particle(comp1[[2]],type = '3d',add = T)



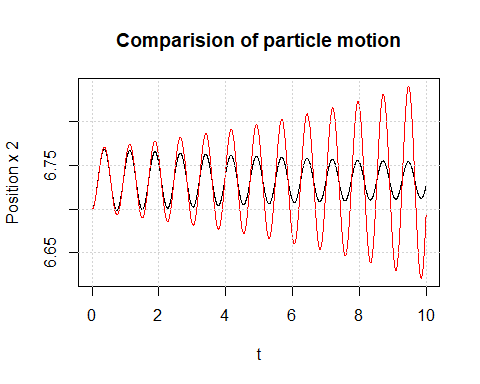
#This is not intuitive for comparision  
#tmin = 0;tmax = 2.5  
  
#Make a matplot with x axis: t  
# y axis: Position x1  
# we can later write this as a function  
maxt1 = max(comp1[[1]][,1])  
maxt2 = max(comp1[[2]][,1])  
mint1 = min(comp1[[1]][,1])  
mint2 = min(comp1[[2]][,1])  
max1 = max(comp1[[1]][,2])  
max2 = max(comp1[[2]][,2])  
min1 = min(comp1[[1]][,2])  
min2 = min(comp1[[2]][,2])  
  
plot(c(min(mint1,mint2),max(maxt1,maxt2)),c(min(min1,min2),max(max1,max2)),  
 xlab = 't',ylab = 'Position x1',type = 'n')  
lines(comp1[[1]][,1],comp1[[1]][,2],col = 1)  
lines(comp1[[2]][,1],comp1[[2]][,2],col = 2)  
grid()  
title('Comparision between h = .005 and h = .01')



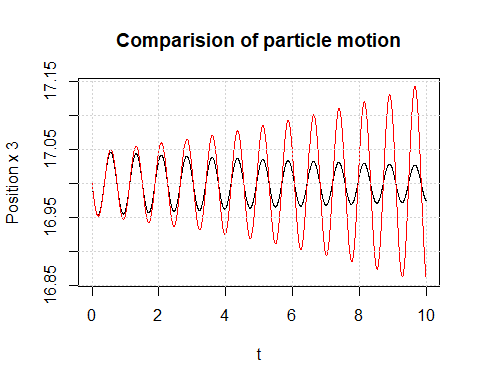
plot.particleTCompare = function(compList,  
 asp1 =1,  
 asp2= 2,  
 tIndex =1,  
 xIndex =2){  
 maxt1 = max(compList[[asp1]][,tIndex])  
 maxt2 = max(compList[[asp2]][,tIndex])  
 mint1 = min(compList[[asp1]][,tIndex])  
 mint2 = min(compList[[asp2]][,tIndex])  
 max1 = max(compList[[asp1]][,xIndex])  
 max2 = max(compList[[asp2]][,xIndex])  
 min1 = min(compList[[asp1]][,xIndex])  
 min2 = min(compList[[asp2]][,xIndex])  
   
 plot(c(min(mint1,mint2),max(maxt1,maxt2)),c(min(min1,min2),max(max1,max2)),  
 xlab = 't',ylab = paste("Position x",as.character(xIndex - 1)),type = 'n')  
 lines(compList[[asp1]][,tIndex],compList[[asp1]][,xIndex],col = asp1)  
 lines(compList[[asp2]][,tIndex],compList[[asp2]][,xIndex],col = asp2)  
 grid()  
 title('Comparision of particle motion')  
}  
  
  
plot.particleCompare = function(compList,  
 asp1 =1,  
 asp2= 2,  
 xIndex =1,  
 yIndex =2,  
 xlab = 'x',  
 ylab = 'y'){  
 maxt1 = max(compList[[asp1]][,xIndex])  
 maxt2 = max(compList[[asp2]][,xIndex])  
 mint1 = min(compList[[asp1]][,xIndex])  
 mint2 = min(compList[[asp2]][,xIndex])  
 max1 = max(compList[[asp1]][,yIndex])  
 max2 = max(compList[[asp2]][,yIndex])  
 min1 = min(compList[[asp1]][,yIndex])  
 min2 = min(compList[[asp2]][,yIndex])  
   
 plot(c(min(mint1,mint2),max(maxt1,maxt2)),c(min(min1,min2),max(max1,max2)),  
 xlab = xlab,ylab = ylab,type = 'n')  
 lines(compList[[asp1]][,xIndex],compList[[asp1]][,yIndex],col = asp1)  
 lines(compList[[asp2]][,xIndex],compList[[asp2]][,yIndex],col = asp2)  
 grid()  
 title('Comparision of particle motion')  
}  
  
#layout(matrix(1:3,3))  
plot.particleTCompare(comp1,xIndex = 2)



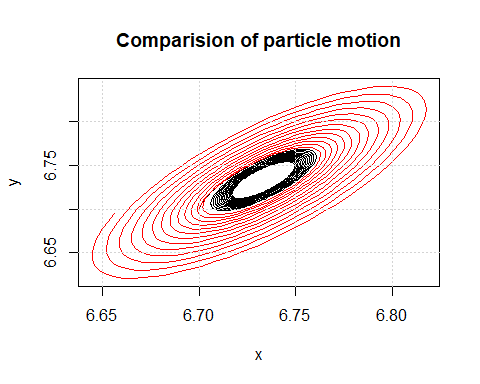
plot.particleTCompare(comp1,xIndex = 3)



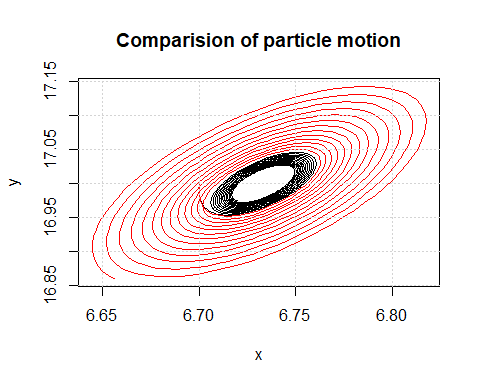
plot.particleTCompare(comp1,xIndex = 4)



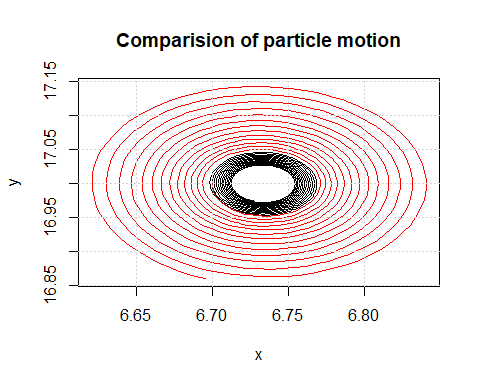
layout(1)  
plot.particleCompare(comp1,xIndex = 2,yIndex = 3)



plot.particleCompare(comp1,xIndex = 2,yIndex = 4)



plot.particleCompare(comp1,xIndex = 3,yIndex = 4)



#I am using Dell Latitute 4250 with 16GB memory in this case  
  
memory.limit()

## [1] 16245

# 2 Figure 6-37 6-38 Sensitivity to the initial condition

## Case1: x1[1]=9.00

## Case2: x1[1]=9.01

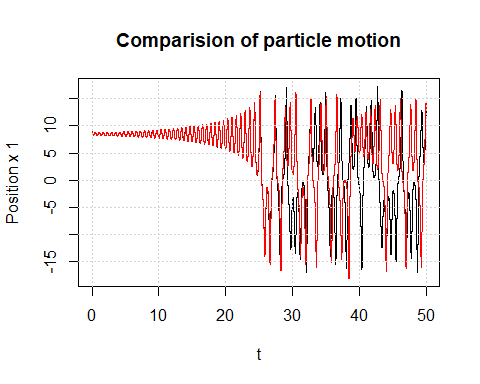
system.time({  
comp2 = foreach(i=c(9,9.01)) %dopar% ({  
 xLorentz\_df(xi =c(i,8,27),param = list(Sigma = 10,b = 8/3,r = 28),  
 init = 0, end = 50,h=.0005)  
 })  
})

## user system elapsed   
## 0.03 0.02 1.18

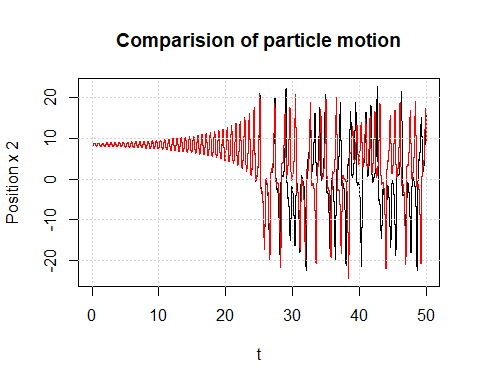
str(comp2)

## List of 2  
## $ : num [1:100001, 1:4] 0 0.0005 0.001 0.0015 0.002 0.0025 0.003 0.0035 0.004 0.0045 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : NULL  
## .. ..$ : chr [1:4] "t" "" "" ""  
## $ : num [1:100001, 1:4] 0 0.0005 0.001 0.0015 0.002 0.0025 0.003 0.0035 0.004 0.0045 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : NULL  
## .. ..$ : chr [1:4] "t" "" "" ""

plot.particleTCompare(comp2,xIndex = 2)



plot.particleTCompare(comp2,xIndex = 3)



plot.particle(comp2[[1]],type ='3d')

