Lorenz Attractors

Atul Singh Arora

March-April 2015

1 Introduction

The Lorenz attractor is the consequence of a set of 3 ordinary differential equations, which for a given range of parameters, results in chaotic solutions. This is of particular interest because the resultant solutions, when plotted in 3 dimensional space, move in a very contrained part of the space, intuitively suggesting that its dimension must be between 1 and 3. What is further fascinating is that even though the solution is chaotic, it is possible to synchronize two lorenz attractors, which is to say that while the motion of neither can be modelled (chaotic), yet both perform the same chaotic motion. In this paper, I show numerically that the dimension of the 'Lorenz attractor' is ~ 2 and demonstrate that it is indeed possible to synchronize two lorenz attractors. Further, a peculiar observation was made; the distance between the two solutions (a measure of synchrony), although appeared to drop exponentially initially, in accordance with the theory, there seemed to be a periodic increase and decrease which I haven't been able to account for theoretically.

2 Lorenz Attractor, equations

The lorenz equations are given as

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = rx - y - xz$$
$$\dot{z} = xy - bz$$

where $\sigma, r, b > 0$ are parameters. These were designed to model convection flows in the atmosphere. One can show various interesting properties of the attractor, including the fact that volumes in phase space contract under the flow, which results in elimination of existence of sources/repellers. For $r < 1$, it can be shown that the origin is globally stable. One can further show that, given

$$r_H \equiv \frac{\sigma(\sigma + b + 3)}{\sigma - b - 1}$$

for

$$1 < r < r_H$$

there exist two more stable fixed points, C^+ and C^-. For $r > r_H$ is where there's chaos, where chaos maybe defined as "aperiodic long-term behaviour in a deterministic system that exhibits sensitive dependence on initial conditions".

3 Syncing two Lorenz attractors

Upon appropriate scaling ($u = x/10, v = y/10, w = z/20$), I can define the same lorenz equations as

$$\begin{aligned}\dot{u} &= \sigma(v - u) \\ \dot{v} &= ru - v - 20uw \\ \dot{w} &= 5uv - bw\end{aligned}$$

I can couple another lorenz attractor as

$$\begin{aligned}\dot{u}_r &= \sigma(v_r - u) \\ \dot{v}_r &= ru - v - 20uw_r \\ \dot{w}_r &= 5uv_r - bw_r\end{aligned}$$

where note that the coupling is done by replacing u_r with u. The claim is that regardless of the initial conditions, both attractors synchronize nearly perfectly, exponentially fast. This is particularly interesting because even though we're feeding in the u variable, the other attractor is able to generate v and w to be in perfect sync.

Quantitatively, the claim is that if I define

$$\begin{aligned}\mathbf{x} &= (u, v, w) \\ \mathbf{y} &= (u_r, v_r, w_r) \\ \mathbf{d} &= \mathbf{x} - \mathbf{y}\end{aligned}$$

then $d \to 0$ as $t \to \infty$. The proof is as follows.

$$\begin{aligned}\dot{d}_1 &= \sigma(d_2 - d_1) \\ \dot{d}_2 &= -d_2 - 20ud_3 \\ \dot{d}_3 &= 5ud_2 - bd_3\end{aligned}$$

We now construct the following by multiplying the second equation with d_2 and the third by $4d_3$ and then adding. We have

$$\begin{aligned}d_2\dot{d}_2 + 4d_3\dot{d}_3 &= -d_2^2 - 20ud_2d_3 + 20ud_2d_3 - 4bd_3^2 \\ &= -d_2^2 - 4bd_3^2\end{aligned}$$

where we have managed to remove the u term which was known to be chaotic. Note also that the left

hand side is essentially

$$\frac{1}{2}\frac{d}{dt}\left(d_2^2 + 4d_3^2\right)$$

Next we define the function

$$E(\mathbf{d}, t) = \frac{1}{2}\frac{d}{dt}\left(\frac{1}{\sigma}d_1^2 + d_2^2 + 4d_3^2\right)$$

Note now that for $\sigma > 0$, $E > 0$ for all t. If I can show that $\dot{E} < 0$, then that must mean that eventually, $d \to 0$. I have already evaluated two terms in the sum. Lets evaluate the first term.

$$\begin{aligned}\dot{E} &= \left[\frac{1}{\sigma}d_1\dot{d}_1\right] + -d_2^2 - 4bd_3^2 \\ &= -\left[d_1^2 - d_1 d_2\right] - d_2^2 - 4bd_3^2\end{aligned}$$

Completing square (for the bracketed term)

$$\begin{aligned}\dot{E} &= -\left[d_1^2 - d_1 d_2\right] - d_2^2 - 4bd_3^2 \\ &= -\left[d_1^2 - \frac{1}{2}d_2\right]^2 + \left(\frac{1}{2}d_2\right)^2 - d_2^2 - 4bd_3^2 \\ &= -\left[d_1^2 - \frac{1}{2}d_2\right]^2 - \frac{3}{4}d_2^2 - 4bd_3^2\end{aligned}$$

So for $b > 0$, we know that $\dot{E} \leq 0$ with equality only if $\mathbf{d} = \mathbf{0}$. It thus proves that $\mathbf{d} = \mathbf{0}$ is globally stable.

4 Correlation Dimension

This is to briefly describe the concept since this will be used in finding the dimension of the Lorenz attractor. First we find a large set of points

$$X = \{\mathbf{x}_i | i = 1, \ldots, n\}$$

Then for a given point $\mathbf{x}$, we consider a ball of radius ϵ. We evaluate $N_{\mathbf{x}}(\epsilon)$ which measures the number of points in X that are inside the ball. We do this computation $\forall\ \mathbf{x}_i$ and find the average, viz.

$$< N_{\mathbf{x}_i}(\epsilon) > \equiv C(\epsilon)$$

It is known emperically that

$$C(\epsilon) \propto \epsilon^d$$

where d is defined to be the *correlation dimension*. It is also claimed that $d_{\text{correlation}} \leq d_{\text{box}}$ but is usually very close. It is almost immediate that ϵ must be in a specific range, say

(minimum seperation between points in X)$\ll \epsilon \ll$('diameter' of X)

5 Simulation

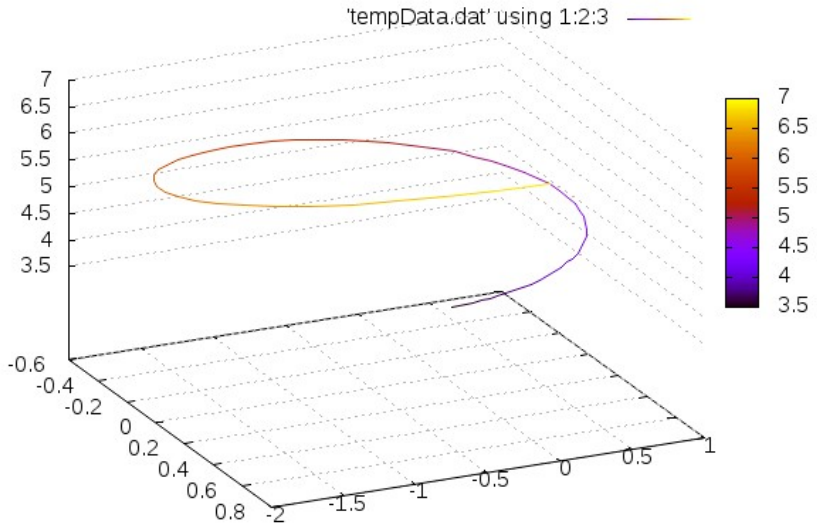
The simulation was written in fortran. The time evolution was done using RK4, for the following parameter values

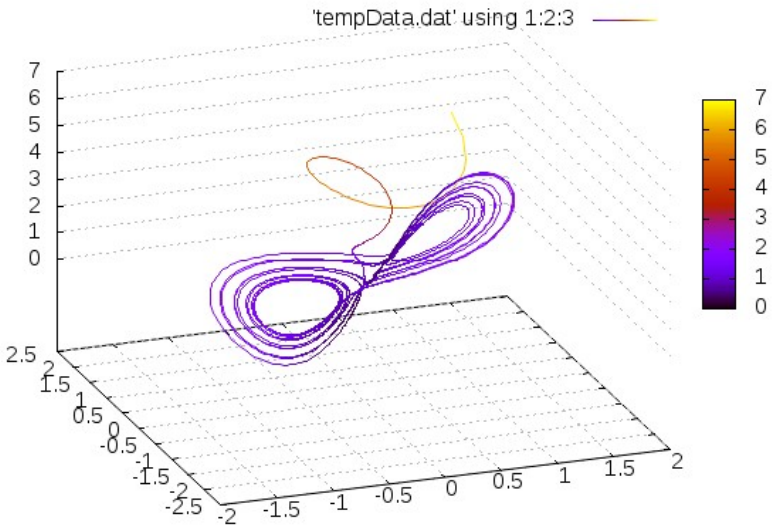
$$\sigma = 10, \rho = 29, \beta = 8.0/3, \gamma = 20.0, \delta = 5.0, \epsilon = 2.85$$

where γ and δ have been introduced because of scaling as has been discussed. The time-step was taken to be both $dt = 4.0 \times 10^{-3}$ and $dt = 4.0 \times 10^{-2}$ depending on the required resolution.

5.1 Single attractor

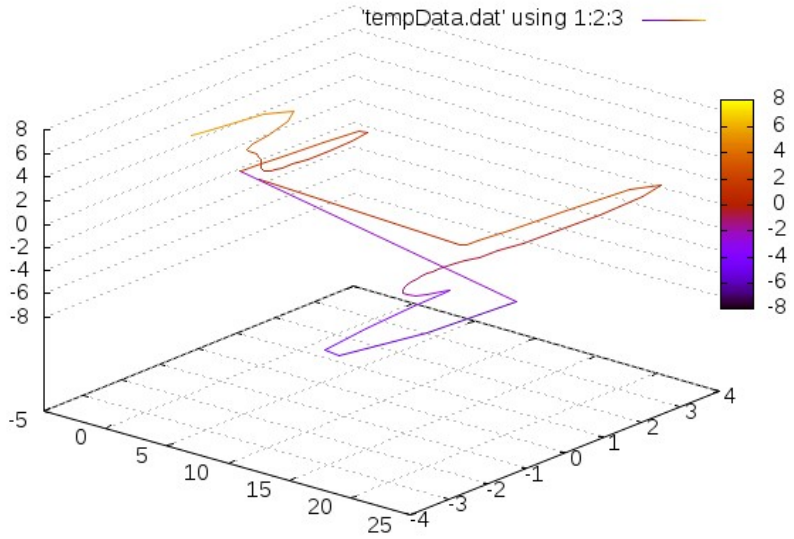
Following are the results from frame 65 and 4300 respectively.

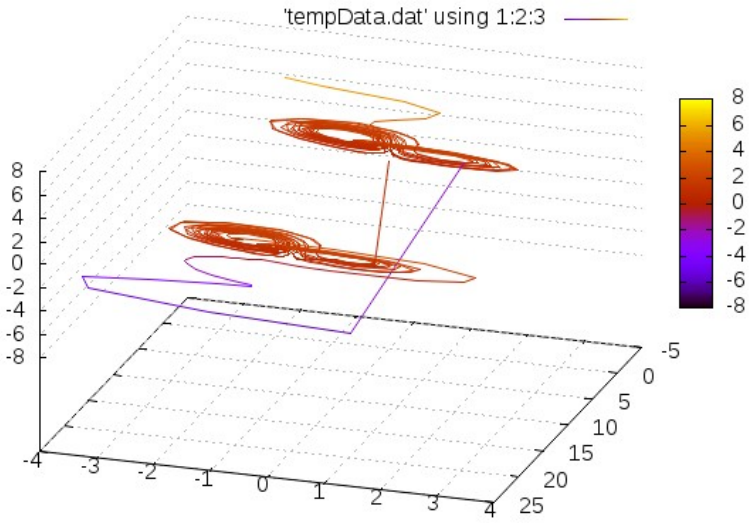


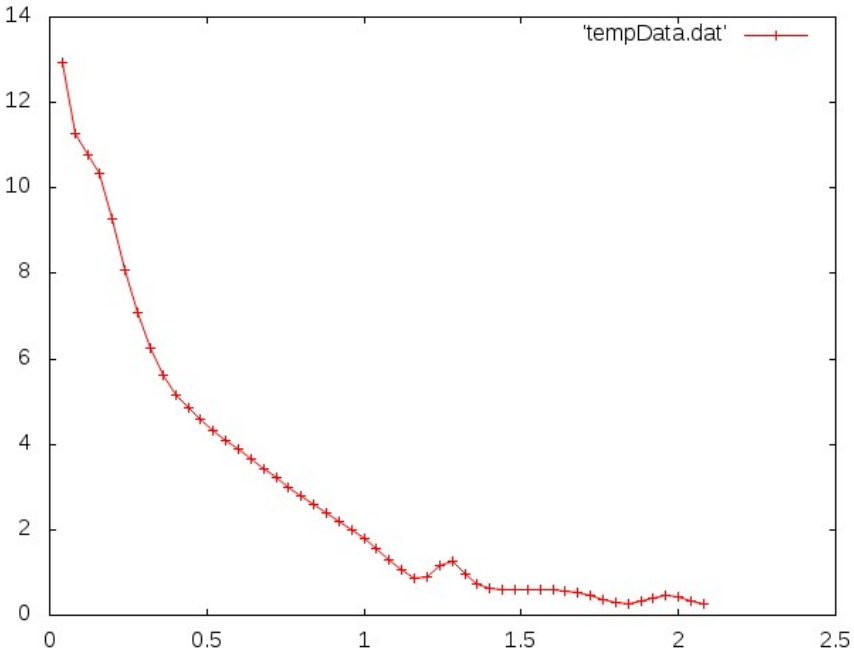


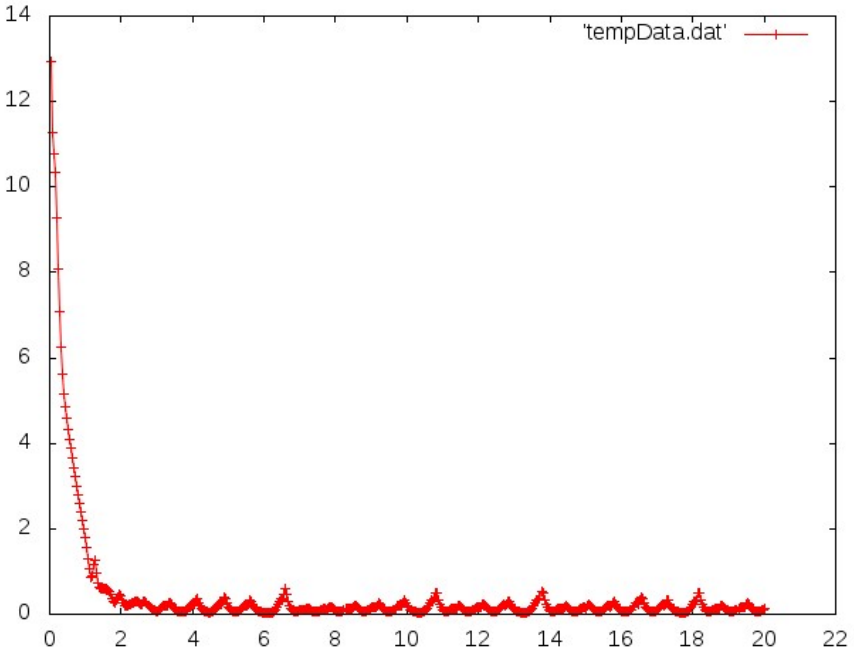
5.2 Synchronized Attractors

This has been obtained by plotting one attractor at an offset (by $\mathbf{y}_f = (20, 0, 0)$ and using the synchronization equations described earlier. The first image shows that the initial conditions were distinct, explicitly $x_0 = (0.8, 0.3068, 7.0)$, $y_0 = (-2.8, -0.3068, -7.0)$. The second image shows that the two synchronize soon after. The last two images show a plot between $\mathbf{d}^2$ and time. Note how despite being evolved for long enough, and despite showing that $\dot{E} \leq 0$, the error term $\mathbf{d}^2$ does increase slightly and decrease, in what seems to be a periodic way.


'tempData.dat' using 1:2:3
8
6
4
2
0
-2
-4
-6
-8
-5
0
5
10
15
20
25
-4
-3
-2
-1
0
1
2
3
4

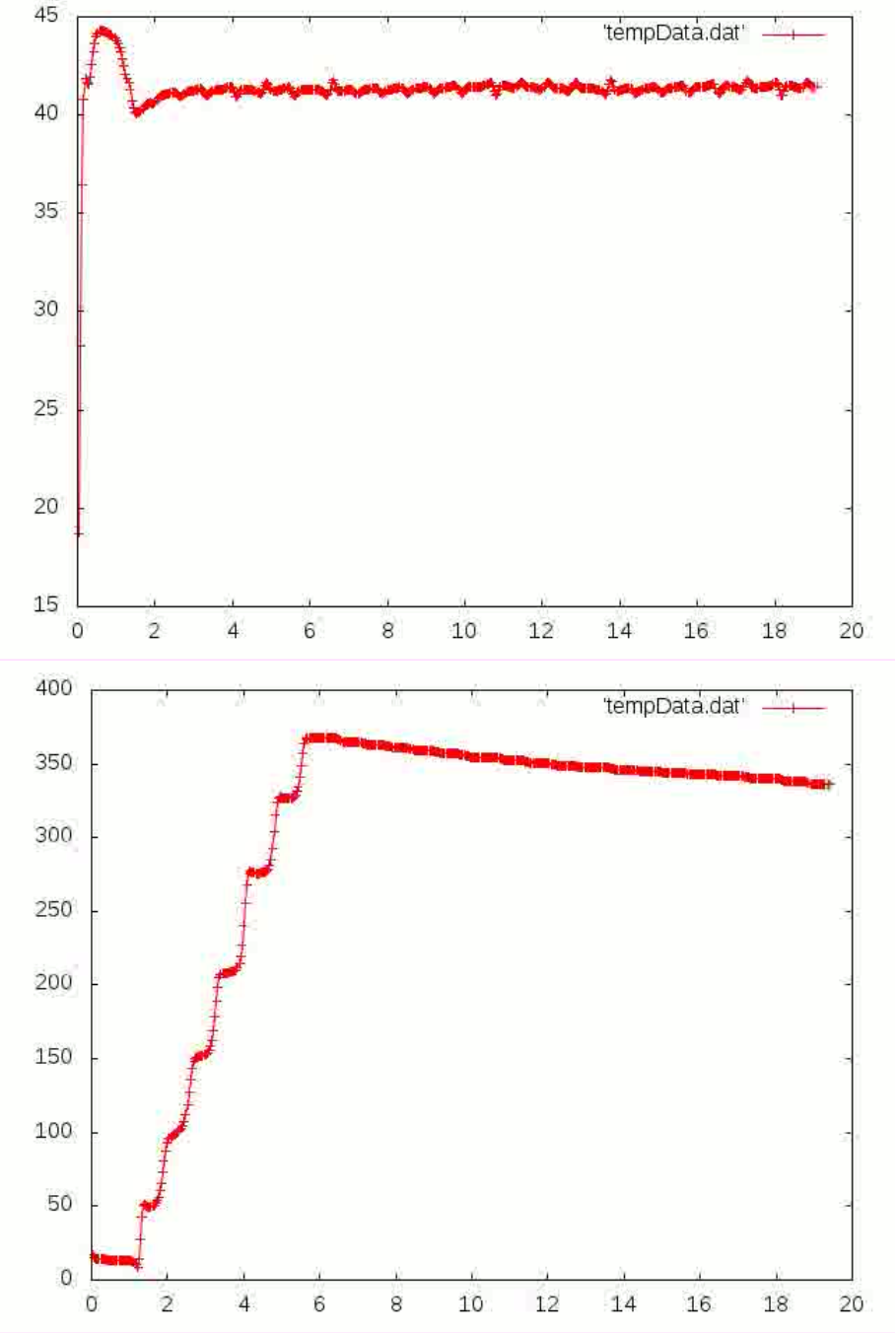

'tempData.dat' using 1:2:3
8
6
4
2
0
-2
-4
-6
-8
-4
-3
-2
-1
0
1
2
3
4
-5
0
5
10
15
20
25





5.2.1 Alternate Synchronizations

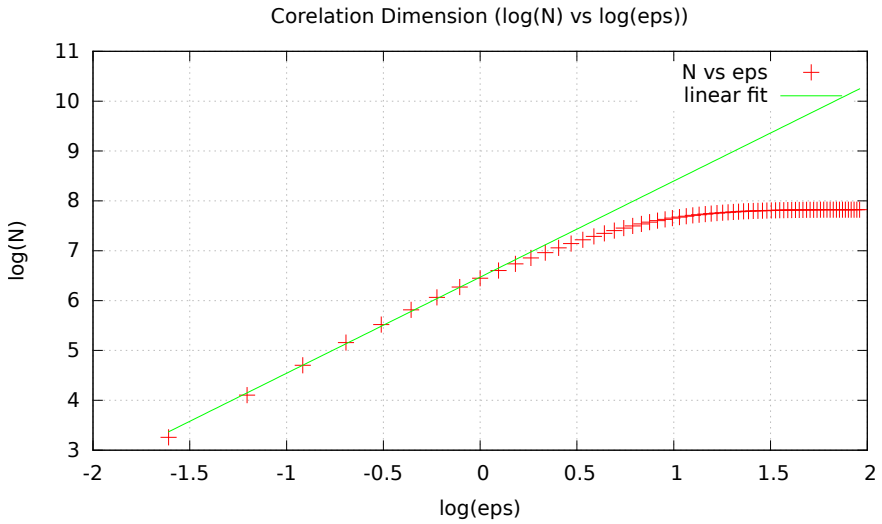
I also attempted feeding in v instead of u to the second oscillator, and then did the same with w. In neither of the cases, did $\mathbf{d}^2 \to 0$ however it did seem to stabilize. The first graph is that for v and second that for w.

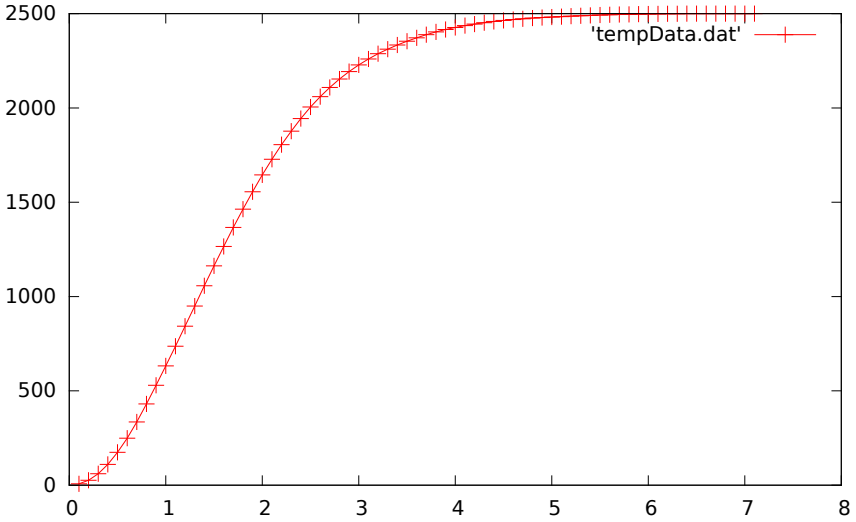


6 Dimension

The correlation dimension of the attractor was evaluated by the method described earlier. The dimension was found to be $1.92733 \pm 0.02832(1.469\%)$ which is slightly less than what is given in the text (~ 2.01) but is intuitively in the right range, close to 2. This was found by iterating through 2250

points. The graph ($\log N$ $\log E$ and N vs E) and detailed initial conditions have been given below.






```
Final set of parameters            Asymptotic Standard Error
=======================            =========================

m                = 1.92733         +/- 0.02832      (1.469%)
c                = 6.47041         +/- 0.01696      (0.2621%)


iterations/points = 2250

INITIAL CONDITIONS
real, parameter :: sigma=10,rho=29,rho2=35, beta=8.0/3, gamma=20.0 ,
    ↪ delta=5.0 ,dt=4.0E-2,epsilon=2.85,maxT=100, maxEps=7,deltaEps=0.1,
```

```
    ↪ edg=2.0
integer , parameter :: maxI = maxT/dt , snakeSize=10, maxSpheres=200,
    ↪ maxEpsI=maxEps/deltaEps
integer :: i,j,k,l
real :: relativeDistance,t,eps

logical :: plotGraphs = .false.
logical :: findDimension = .false.

real, dimension(3) :: x0 = (/ 0.8, 0.3068, 7.0 /), y0 = (/ -2.8, -0.3068,
    ↪  -7.0 /), yf= (/ 20,0,0/)
real, dimension(maxI,3) :: xp,yp !yf = reshape( (/ (40,i=1,3*maxI) /), (/
    ↪  maxI, 3 /) )
real, dimension(2*maxI,3) :: xy
real, dimension (3) :: x,xc,k1,k2,k3,k4
real, dimension (3) :: y,yc,m1,m2,m3,m4
real, dimension (maxI,2) :: dis
real, dimension (maxEpsI,2) :: epscount
```

7 Conclusion

A lorenz attractor was simulated using RK4. Two lorenz attractors were shown to get coupled, despite being in the chaotic regime. Interesting periodic oscillations of the relative distance (measure of synchrony) were found. These seem to naively said, contrad the theory. Two other obvious ways of synchronization were attempted which seem to result in a constant 'phase difference' synchronization. This is indicated by the relative distance seemingly varying slowly after the transience. And finally, the correlation dimension of the lorenz attractor was obtained to be very close to ~ 2 as was expected.

8 References and Acknowledgement

I refered to the textbook for the course, titled **Nonlinear Dynamics and Chaos**, *S. H. Strogatz*, Perseus Books Publishing, LCC

I would like to acknowledge our instructor, *Prof. Sudeshna Sinha*, the instructor for the course by the same name.

I also mention the contribution of my friends (and colleagues). Foremost, *Vivek Sagar,* discussions with him accelerated the completion of the work. In addition I thank *Yosman Bapatdhar* and *Arjit Kant Gupta*, who were there when I wrote the first instance of the code.

9 Source Code

The source code is available online at `https://github.com/toAtulArora/chaosTerm.git`. The gnuplot_fortran library has been mostly written by me (built on top of a tutorial). Here I have appended only the main code.

```
program lorenz
use gnuplot_fortran
implicit none
!dt=4.0E-2,
real, parameter :: sigma=10,rho=29,rho2=35, beta=8.0/3, gamma=20.0 ,
    ↪ delta=5.0 ,dt=4.0E-3,epsilon=2.85,maxT=20, maxEps=7,deltaEps=0.1,
    ↪ edg=2.0
integer, parameter :: maxI = maxT/dt, snakeSize=10, maxSpheres=200,
    ↪ maxEpsI=maxEps/deltaEps
integer :: i,j,k,l
real :: relativeDistance,t,eps

logical :: plotGraphs = .true., plotBoth=.false.
logical :: findDimension = .false.

real, dimension(3) :: x0 = (/ 0.8, 0.3068, 7.0 /), y0 = (/ -2.8, -0.3068,
    ↪  -7.0 /), yf= (/ 20,0,0/)
real, dimension(maxI,3) :: xp,yp !yf = reshape( (/ (40,i=1,3*maxI) /), (/
    ↪  maxI, 3 /) )
real, dimension(2*maxI,3) :: xy
real, dimension (3) :: x,xc,k1,k2,k3,k4
real, dimension (3) :: y,yc,m1,m2,m3,m4
real, dimension (maxI,2) :: dis
real, dimension (maxEpsI,2) :: epscount

type sphereLike
   integer :: count
   real, dimension(3):: origin
end type sphereLike

type(sphereLike), dimension(maxI)::sphere

call startPlot()
```

```
call setXrange(0.0,100.0)
call setYrange(0.0,100.0)
call setZrange(0.0,100.0)

call srand(100)

write (*,*) "Iterating through..(will do ", maxI, "iterations)"

i=1
t=0
do while (t<maxT) !i<3000)
   t=t+dt

   k1=xDot(x0)
   k2=xDot(x0 + 0.5*dt*k1)
   k3=xDot(x0 + 0.5*dt*k2)
   k4=xDot(x0 + dt*k3)
   xc = x0 + dt*(1.0/6)*(k1+2*k2+2*k3+k4)
   x0=xc

   k1=yDot(xc,1,y0)
   k2=yDot(xc,1,y0 + 0.5*dt*k1)
   k3=yDot(xc,1,y0 + 0.5*dt*k2)
   k4=yDot(xc,1,y0 + dt*k3)
   yc = y0 + dt*(1.0/6)*(k1+2*k2+2*k3+k4)
   y0=yc




   xp(i,:)=xc
   yp(i,:)=yc + yf
   dis(i,2)=sqrt(sum((xc-yc)*(xc-yc)))
   dis(i,1)=t

   xy(1:i,:)=xp(1:i,:)
   xy(i:2*i,:)=yp(1:i,:)
   if(plotGraphs) then
```

```
      if(plotBoth) then
         call nextPlot3d(xy(1:2*i,1),xy(1:2*i,2),xy(1:2*i,3))
      else
         call nextPlot3d(xp(1:i,1),xp(1:i,2),xp(1:i,3))
      end if
      call nextPlot2d(dis(:i,1),dis(:i,2))
   end if
   i=i+1

end do


if(findDimension) then
   write (*,*) "Finding the box dimension.."

   j=0
   eps=1.0E-10
   sphere(:)%origin(1) = xp(:,1)
   sphere(:)%origin(2) = xp(:,2)
   sphere(:)%origin(3) = xp(:,3)


   do while (eps<maxEps)
      j=j+1
      eps=eps + deltaEps
      do k=1,maxI
         sphere(k)%count=0
         do i=1,maxI
            sphere(k)%count=sphere(k)%count + isInsideSphere(sphere(k)%
               ↪ origin,eps,xp(i,:))
         end do
      end do
      epscount(j,1)=eps
      epscount(j,2)=sum(sphere(1:maxSpheres)%count)/real(maxSpheres)

   end do

   call plot2dSave(log(epscount(1:j,1)),log(epscount(1:j,2)+1.0E-13),
```

```
      ↪ filename="dimensionLogLog.pdf",picFormat=1)
   call plot2dSave(epscount(1:j,1),epscount(1:j,2),filename="dimension.
      ↪ pdf",picFormat=1)

   open(file="dimension.dat",unit=4)
   do i=2,j
      write (4,*) epscount(i,1),epscount(i,2)
   end do
end if

call endPlot()
call system("xdg-open result3d.avi")


contains
  function xDot(x)
    real, dimension(3)::xDot,x
    xDot(1)=sigma*(x(2)-x(1))
    xDot(2)=rho*x(1) - x(2) - gamma*x(1)*x(3)
    xDot(3)=(delta*x(1)*x(2)) - (beta*x(3))
  end function xDot

  function yDot(x,l,y)
    real, dimension(3)::yDot,y,x
    real :: u
    integer :: l
    if(l==1) then
       u=x(1)
       yDot(1)=sigma*(y(2)-y(1))
       yDot(2)=rho*u - y(2) - gamma*u*y(3)
       yDot(3)=(delta*u*y(2)) - (beta*y(3))
    else if(l==2) then
       yDot(1)=sigma*(x(2)-y(1))
       yDot(2)=rho*y(1) - x(2) - gamma*y(1)*y(3)
       yDot(3)=(delta*y(1)*x(2)) - (beta*y(3))
    else if(l==3) then
       yDot(1)=sigma*(y(2)-y(1))
       yDot(2)=rho*y(1) - y(2) - gamma*y(1)*x(3)
```

```
        yDot(3)=(delta*y(1)*y(2)) - (beta*x(3))
    else
        stop "Not implmented"
    end if
  end function yDot

  function isInsideSphere(r0,eps,r)
    real, dimension(3) :: r0,r
    real :: eps
    integer:: isInsideSphere
    isInsideSphere=0
    if(eps*eps > sum( (r0-r)*(r0-r)) )  then
        isInsideSphere=1
    end if
  end function isInsideSphere

  subroutine initOrigin(spheres)
    type(sphereLike), dimension(:) :: spheres
    do l=1,maxSpheres
        spheres(l)%origin(1)=(0.5-rand())*2.0*edg
        spheres(l)%origin(2)=(0.5-rand())*2.0*edg
        spheres(l)%origin(3)=(0.5-rand())*2.0*edg
    end do
  end subroutine

end program lorenz
```